Informatics

Klassenzentrierte Visuell-Interaktive Bildbeschriftung mittels Eigenschaftsmaßen

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Data Science

eingereicht von

Matthias Matt, BSc
Matrikelnummer 01529399

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Assistant Prof. Dr.in techn. Manuela Waldner, MSc
Mitwirkung: FH-Prof. Priv.-Doz. Dipl.-Ing. Mag. Dr. Matthias Zeppelzauer

Wien, 8. Mai 2024

Matthias Matt

Manuela Waldner

Informatics

Class-Centric Visual Interactive Labeling using Property Measures

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Data Science

by

Matthias Matt, BSc
Registration Number 01529399

to the Faculty of Informatics

at the TU Wien

Advisor: Assistant Prof. Dr.in techn. Manuela Waldner, MSc
Assistance: FH-Prof. Priv.-Doz. Dipl.-Ing. Mag. Dr. Matthias Zeppelzauer

Vienna, 8th May, 2024

Matthias Matt	Manuela Waldner

Erklärung zur Verfassung der Arbeit

Matthias Matt, 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.

Wien, 8. Mai 2024

Matthias Matt

Danksagung

Zunächst möchte ich meiner Betreuerin Manuela Waldner für ihre unermüdliche Unterstützung während dieser Arbeit von ganzem Herzen danken. Ihr Fachwissen, ihre Sorgfalt und ihre kontinuierliche Begleitung waren von unschätzbarem Wert beim Schreiben dieser Arbeit und um nicht von der Bahn abzukommen. Ich möchte auch Matthias Zeppelzauer meinen aufrichtigen Dank für sein wichtiges Feedback und seine Anregungen aussprechen, die wesentlich dazu beigetragen haben, diese Arbeit in ihre jetzige Form zu bringen.

Darüber hinaus möchte ich Professor Takeo Igarashi von der Universität Tokio dafür danken, dass er mich während meines gesamten Auslandsaufenthaltes unterstützt und mir diese Möglichkeit gegeben hat.

Ich möchte mich auch bei meinen Freundinnen und Freunden bedanken, die mich in meinem Studium und auch darüber hinaus unterstützt haben. Außerdem möchte ich mich bei allen Teilnehmerinnen und Teilnehmern der Studie bedanken.

Nicht zuletzt möchte ich meinen Eltern danken, die der Grund dafür sind, dass ich heute da bin, wo ich bin. Ihre ständige Unterstützung und Ermutigung hat mich zu diesem Punkt geführt.

Acknowledgements

First, I want to express my heartfelt gratitude to my supervisor, Manuela Waldner, for her unwavering support throughout this journey. Her expertise, diligence, and continued guidance were invaluable in helping me write this thesis and keeping me on track. I would also like to sincerely thank Matthias Zeppelzauer for providing crucial feedback and input, which helped shape the thesis into its current form.

Moreover, I would like to thank Professor Takeo Igarashi from the University of Tokyo for supporting me throughout my studies abroad and providing me with this wonderful opportunity.

I would also like to thank my friends who supported me in my studies and otherwise. Furthermore, I would like to express my gratitude to all user study participants.

Last but not least, I want to thank my parents, who are the reason I am where I am today. Their constant support and encouragement have led me to this point.

Kurzfassung

Manuelles Annotieren von Bilddaten ist von fundamentaler Bedeutung für überwachtes maschinelles Lernen, bei dem beschriftete Datensätze für das Trainieren von Modellen unerlässlich sind. Traditionell wurde die Reduktion des Annotationsaufwands durch sogenanntes *active learning* erreicht, bei dem die optimale nächste Instanz für die manuelle Beschriftung auf der Grundlage von Heuristiken zur Maximierung des Nutzens ausgewählt wird. Neuere Arbeiten konzentrieren sich auf die Integration von Benutzer:innen in den Beschriftungsprozess durch *visual interactive labeling*, um den Prozess direkt und aktiv zu steuern.

In dieser Arbeit wird cVIL präsentiert, ein klassenzentrierter Ansatz für *visual interactive labeling*, der den manuellen Annotationsprozess für große und komplexe Bilddatensätze vereinfacht. Bisher waren visuelle Ansätze typischerweise instanzbasiert, wobei das System einzelne Instanzen visualisiert, die dann annotiert werden können. cVIL nutzt hingegen sogenannte Eigenschaftsmaße, um schwierige Instanzen einzeln zu annotieren und außerdem eine große Anzahl von einfacheren Fällen schnell zu annotieren. Da die Eigenschaftsmaße die Eigenschaften einer Instanz mit einem einzigen skalaren Wert ausdrücken, sind die Visualisierungen einfach und skalierbar. cVIL kombiniert den heuristischen Führungsansatz von *active learning* mit dem nutzerzentrierten Ansatz von *visual interactive labeling*.

In Simulationen konnten wir zeigen, dass Eigenschaftsmaße effektives Annotieren von einzelnen Instanzen und größeren Selektionen ermöglicht. In einer Studie zeigte cVIL eine höhere Genauigkeit und Zufriedenheit der Nutzer:innen im Vergleich zu einer instanzbasierten *visual interactive labeling* Baseline, wo Scatterplots zur Visualisierung der Daten verwendet wurden.

Abstract

Human annotation of image data is relevant for supervised machine learning, where labeled datasets are essential for training models. Traditionally, reducing the labeling effort was achieved through active learning, where the optimal next instance for labeling is selected by some heuristic to maximize utility. More recent work has focused on integrating user initiative in the labeling process through visual interactive labeling to steer the labeling process.

This thesis proposes cVIL, a class-centric approach for visual interactive labeling that simplifies the human annotation process for large and complex image datasets. Previously, visual labeling approaches were typically instance-based, where the system visualizes individual instances for the user to label. cVIL utilizes diverse property measures to enable the labeling of difficult instances individually and in batches to label simpler cases rapidly. Since the property measures express the properties of an instance using a single scalar value, the visualizations are simple and scalable. cVIL combines the heuristic guidance approach of active learning with the user-centered approach of visual interactive labeling.

In simulations, we could show that property measures can facilitate effective instance and batch labeling. In a user study, cVIL demonstrated superior accuracy and user satisfaction compared to the conventional instance-based visual interactive labeling approach that employs scatterplots. Participants also needed less time to complete the assigned tasks in cVIL compared to the baseline.

Contents

CHAPTER 1

Introduction

A good human plus a machine is the best combination.

Garry Kasparov

1.1 Motivation and Problem Statement

This work presents a class-centric approach for labeling unstructured data, an essential task for supervised machine learning or subsequent tasks. We specifically focus on images as subjects for labeling. Although repetitive and menial, it depends on human skill and experience and can hardly be accomplished without human guidance for fine-grained classification tasks. However, recent advances in representation learning have made extracting useful semantic information from complex unstructured data possible, which can assist humans in the labeling process. These lower dimensional representations provide new possibilities that are not feasible with the raw data alone, such as fast model training or efficient outlier detection.

Reducing human annotation cost has been wildly studied in the field of *active learning (AL)*, where the training process of a classification model is optimized to require as few labels as possible. Instead of labeling data samples without any particular order, in AL, the annotators are presented with the most valuable samples for the classification model at a specific stage of the training process. However, with this approach, humans are relatively passive figures in the labeling process and have to depend on the model to present beneficial samples. Consequently, several works have been published to give users more agency in the labeling process, which can be described as *visual interactive labeling* (VIL). These approaches use visualization techniques to let the user actively participate in the instance selection process. Typically, these systems show individual data instances in some spatial representation, like a scatterplot projection. We refer to such systems as instance-centric VIL systems or iVIL. While these systems proved useful, they suffer from scalability, interactivity, or data complexity issues.

We aimed to mitigate these issues and ensure that our approach effectively handles the following data characteristics:

1. Data Scale: The labeling process must remain feasible even as the number of samples in the dataset increases. However, scalability in the number of classes is not considered, and we assume that no more than ten classes are present in the data.
2. Data Complexity: The system should be able to handle complex data without affecting the core interaction techniques, and the system should mitigate the existing complexity.

To meet these requirements, our proposed approach adopts a class-centric perspective, partitioning instances into classes based on model predictions without visualizing data points as individual entities.

1.2 Contributions

We present a class-centric iterative labeling approach, which we call cVIL, that visualizes distributions of property measures [BHS$^+$21] for each class independently, where instances are divided based on the predictions of the underlying classification model. A data instance is only represented by a single scalar value based on the currently visualized property measure. These property measure values in a class partition are then shown as a distribution. The main contributions of the thesis can be summarized as follows:

- Conceptually, cVIL combines the advantages of AL and VIL into a unified approach by utilizing well-understood properties that guide the labeling process (like AL) while leaving the user entirely in control of the instance selection (like VIL). In a class-based approach, the user only has to identify false positives when deciding if a class label is correct instead of the additional possibility for true/false negative samples in instance-based systems. The use of property measures enables the labeling of large selections with a single interaction, which we call *batch labeling*, and guides the labeling of individual instances, which we call *instance labeling*.
- We implemented a labeling interface that utilizes property measures to allow users to label large datasets efficiently. To further alleviate the scalability issues of some previous systems, we incorporate representation learning as a preprocessing step and use visualizations that are not impacted by the number of samples that need to be visualized. Representation learning allows for a simple classification model, which results in fast training times and is less affected by overfitting.

- We formalized a simulation framework for evaluating different property measures within the cVIL system. We presented the results by comparing the property measures with each other and compared them to active learning. We further analyzed the utility of cVIL for actual labeling tasks in a user study.

1.3 Research Questions

The success of any labeling system primarily depends on how accurately (effectively) users can label data and how fast (efficiently) they can achieve accurate results [BHZ+18]. Another vital aspect is the scalability of the number of samples that can be labeled through the system, as characterized by the first requirement of the system. Furthermore, the system should lead to a better user experience and increase the accuracy of the labeled output compared to instance-based systems in complex real-world scenarios.

Thus, the research questions can be formulated as follows:

- **RQ1**: Is instance labeling with cVIL more effective than active learning?
- **RQ2**: Can batch labeling improve the effectiveness of cVIL?
- **RQ3**: Is the use of property measures scalable in terms of the number of instances and dimensions to process?
- **RQ4**: Does our class-centric labeling approach (**a**) reduce the labeling effort and (**b**) increase user satisfaction compared to a traditional instance-based approach?

1.4 Methodology

The system is evaluated by simulating user interactions to get robust baselines of the approach and conducting a user study where subjects have to solve label tasks manually.

To answer **RQ1** and **RQ2**, we ran simulations to compare the effectiveness of different property measures for instance labeling and batch labeling. Establishing a set of fundamental interactions for cVIL enabled us to compare the performance of different property measures across various datasets and settings by measuring the accuracy of the labeled output. This ensured that our results were both reproducible and representative.

We analyze **RQ3**, which concerns the system's scalability, by measuring the runtime of various property measures on synthetic data. By changing the number of samples and dimensionality of the data and measuring the runtime for different inputs, we can gain insights into how the properties behave in various settings.

We conducted a user study to answer **RQ4** using a prototypical implementation of cVIL. In the study, participants were required to complete a labeling task using two interfaces: cVIL and an iVIL baseline. The baseline is identical to the cVIL implementation except that it replaces the class-based visualizations with a scatterplot of t-SNE [vdMH08] projections, which was selected based on the findings of Bernard et al. [BHZ+18]. We observe the accuracy of the generated label output, labeling time, task load, and preference to compare cVIL to the iVIL baseline.

1.5 Outline

In the following chapter, we present the related work. This comprises active learning and visual interactive labeling approaches that aim to give users more agency in the labeling process. Here, we also present the property measures that were implemented for evaluation. The representation learning approach used in cVIL and labeling systems that use some form of representation learning are also introduced there.

Chapter 3 introduces cVIL with its components and interaction design, which are explained in detail. Here, we also present the prototype used for the user study and how it implements the different components of the system.

Chapter 4 details the evaluation of the system. We describe the simulation procedures on which the quantitative evaluation is based and the user study design and summarize the procedure for the participants.

Finally, in Chapter 5, the evaluation results from the simulations and user study are presented, and a conclusion with an outline of possible future work is given in Chapter 6.

A paper based on the work of this thesis has been published at the EuroVA 2024 conference.

CHAPTER 2

Background

In this chapter, we provide an overview of the related work for this thesis. We first introduce the concept of active learning, which is an algorithmic approach to determine which instances an oracle (the user) should label based on the system's current state. Then, we discuss visual interactive labeling approaches that give the user more agency during the labeling process. We also introduce property measures, which cVIL uses to combine aspects of both active learning and visual interactive labeling. As representation learning is also a crucial aspect of cVIL, we explain the specific method used to find image embeddings, which serve as input for the property measures. Finally, we present other works incorporating representation learning into the labeling workflow.

2.1 Active Learning

In active learning, a machine learning model is trained by iteratively selecting the most informative samples to maximize performance with the least amount of training data. It has been shown that active learning outperforms passive learning (random sampling) in most settings. Since providing large amounts of training data is costly, reducing the amount of labeled data to train models is advantageous to machine learning practitioners. This principle is also at the core of cVIL.

Fu et al. [FZL12] provide an overview of the most common active learning instance selection strategies, all of which could, in principle, be included in cVIL. They categorize the strategies into two major groups: IID-based and feature-correlation-based approaches. IID-based measures are based on properties determined for each sample independently, while feature-correlation-based measures try to select instances in relation to one another. Due to the computational complexity of feature-correlation-based strategies, only independent measures are directly implemented in the prototype. However, cVIL also incorporates scalable data-based selection strategies that are conceptually similar or are heuristics of these approaches.

The three main IID-based measures are uncertainty-based approaches, Expected Gradient Length, and Variance Reduction. The latter two are again excluded for computational complexity considerations. Furthermore, these are tightly coupled to a specific family of models, which would require restricting the model choice for the framework. That leaves uncertainty-based measures, which are suited for labeling.

Generally, each measure assigns a utility value to each observation $x \in X$ in the data. For uncertainty-based methods, the selection strategy always incorporates the probabilistic classification model f_θ with parameters θ, which assigns conditional probabilities $P(y|x)$ to a data sample:

$$f_\theta : X \longrightarrow [0,1]^K \tag{2.1}$$

with K classes, where the components sum to 1. Therefore, the utility of a sample is given by the function $u : \theta \times X \longrightarrow \mathbb{R}$.

In active learning, the selected sample x^* at each iteration is the sample with the highest utility, which is given by:

$$x^* = \arg\max_{x \in X} u(\theta, x) \tag{2.2}$$

Fu et al. [FZL12] present the most prominent uncertainty-based selection strategies. To simplify notation, we define $P_\theta(y_{(0)}|x), P_\theta(y_{(1)}|x), ..., P_\theta(y_{(K-1)}|x)$ to be the sorted probabilities where $P_\theta(y_{(0)}|x)$ is the largest probability value and $P_\theta(y_{(K-1)}|x)$ the smallest probability value.

Uncertainty sampling

In uncertainty sampling, the sample with the highest uncertainty is selected. That is:

$$u_{UNC}(\theta, x) = 1 - max f_\theta(x) = 1 - P_\theta(y_{(0)}|x) \tag{2.3}$$

Min-Margin sampling

In margin sampling, the utility is defined as the difference between the largest and the second largest probability:

$$u(\theta, x) = P_\theta(y_{(0)}|x) - P_\theta(y_{(1)}|x) \tag{2.4}$$

However, this leads to the smallest value being the most uncertain instance, which would contradict the definition of x^* from before, where we select the instance with maximum utility. Therefore, we have to multiply by -1:

$$u_{MA}(\theta, x) = P_\theta(y_{(1)}|x) - P_\theta(y_{(0)}|x) \tag{2.5}$$

The domain of the utility is now [-1, 0]. We can add one, if that is required, to get positive utility values.

Entropy sampling

Here, the entropy of the model prediction is used as the utility of a sample:

$$u_{ENT}(\theta, x) = -\sum_{i=0}^{K-1} P_\theta(y_i|x) \log P_\theta(y_i|x) \tag{2.6}$$

For binary classification problems, these three approaches are equivalent. A proof can be found in the appendix A.1.

Active learning, therefore, tries to select the sample with the largest utility at a specific iteration. This sample is then used for training, which results in updated model parameters θ and updated utility measures for the remaining samples. Active learning can be described using a simple algorithm [FZL12]:

Algorithm 2.1: Active Learning

```
   Input: unlabeled data $D$, initial labeled data $L$, training size $m \leq |D| + |L|$
   Output: model $f_\theta$, labeled data $L$
1  $f_\theta \leftarrow initModel(L)$
2  while $|L| \leq m$ do
3      $x^* \leftarrow \arg\max_x u(f_\theta, x)$
4      $y^* \leftarrow oracle(x^*)$
5      $L \leftarrow L \cup (x^*, y^*)$
6      $f_\theta \leftarrow trainModel(f_\theta, L)$
7      $D \leftarrow D \setminus x^*$
8  end
```

At each iteration of the labeling loop, the optimal sample x^* is selected based on the chosen utility function, and an oracle (the user) is queried for the label y^*. The chosen instance is added to the labeled instances L along with the label from the oracle. Then, the model is updated with the new information in L, and x^* is removed from the pool of unlabeled instances. The process stops once the number of labels in L exceeds the training size m. However, any other stopping criterion can be applied as well.

2.1.1 Evaluation of Active Learning Strategies

The classification accuracy measure is the most prominent metric for comparing models. Since active learning is an iterative process, we are interested in how the model improves over time as more labels are added. Reyes et al. [PAV18] argue that the area under the learning curve (ALC) is a standard metric to quantify the model performance over the whole training process.

The learning curve is determined by the model accuracy at a specific number of queried instances, and as more samples are available for training, the accuracy of the model changes, resulting in a curve that illustrates this change. As the name suggests, the area under this curve is the ALC metric, which can be calculated with the trapezoid rule. Another common way of comparing selection strategies is just a visual comparison of the learning curves. We chose visual comparison since this also allows us to compare the results at different stages of the labeling process.

Yang and Loog [YL16] thoroughly compare different active learning approaches with logistic regression as the classification model. They tested ten different instance selection strategies and found that entropy was the best strategy for the 44 real-world datasets they used in their evaluation. It had the highest win count, the highest average ranking, and the highest mean ACL score of all evaluated strategies. It also performed best across all these metrics on 80 subsets of the ImageNet dataset. Expected error reduction (EER) performed almost equally well in these tests. However, in terms of computational cost, entropy is consistently orders of magnitude faster than most other methods, especially EER, which is approximately 1000 times slower. Only the maximum model change (expected gradient length or similar approaches) is comparable regarding runtime. However, they use a particular method to estimate the model change, which only applies to logistic regression and favors uncertain instances, making it similar to entropy in that regard.

Reyes et al. [PAV18] used Support vector machines (SVM) as the classification model to analyze the performance of the uncertainty-based strategies on 26 datasets. Their evaluation found little difference between uncertainty, margin, and entropy sampling. However, all procedures performed better than random sampling with statistically significant results.

2.1.2 Batch Active Learning

We will give a brief overview of batch active learning to avoid confusion about this term and to put the cVIL system into perspective. In batch active learning or batch mode active learning, multiple instances are selected to be labeled in one iteration. This intends to mitigate potentially long training times of models and utilize a parallel labeling framework. By returning more instances, multiple labelers can work simultaneously and generate labels more efficiently.

However, batch-mode active learning requires more consideration of the instance selection strategies. Selecting the instances with the highest utility is not the optimal strategy since this does not consider information overlap among these instances. We would, therefore, query many redundant instances. Instead, there should be a focus on diversifying the instances in a batch to mitigate these problems. Several approaches have been studied to diversify batch selections [Set09].

In cVIL, there are two forms of batching in label selection. Firstly, of course, there is what we term “batch labeling,” which is the selection of instances with low uncertainty or, more generally, with low property measure values. This might seem counterintuitive since active learning selects the highest utility samples. However, these instances serve as a baseline for the model to learn from prototypical instances. These selections have a lower sample weight and, therefore, have less influence on the model than instance labels. The interaction techniques used make selecting many instances at once much more efficient. Furthermore, these instances are diversified by the class assignments. A fixed amount of instances from each class is sampled from the batch label pool at training time to prevent an unbalanced class distribution.

Secondly, since the user is not forced to train the model after a fixed amount of labels, we have batch labeling in that respect from instance labeling. Realistically, the least amount of labels per iteration is to label one sample from each class. However, theoretically, the user could look at the specific maximum value of each class and only label a single instance with the highest score, but this is not intended. So, we take the smallest batch size as the number of classes. We hypothesize that this has no negative influence on the performance since the class predictions again diversify the labels. We will examine the training behavior of different batch sizes in Chapter 5.

2.2 Visual Interactive Labeling

With active learning, the user acts as an oracle that provides labels to the model. The following approaches give users more possibilities to interact with the labeling process. They shift the focus toward user interaction, and instance selection is done actively by the user. The model mainly provides users with information, such as confidence scores. The information about the data or the model is primarily presented using visualizations, and the labeling is done by interacting with the visualizations.

The goal is to utilize human expertise to improve the label quality and give users more agency to make the labeling process less repetitive. Users can also gain knowledge about the data or the model while labeling. For example, they might gain insights about the distribution, clusters, etc., in the data or have more information about how the model performs, such as accuracy, bias, or other aspects.

Seifert et al. [SG10] published an important early work that can be attributed to visual interactive labeling. Their system visualizes model confidence of all instances in a single visualization instead of only presenting the most uncertain instances to the user, like in active learning. The visualization is a star coordinate plot based on a classifier's posterior probability, as can be seen in Figure 2.1. They place the classes on the outer rim of a circle and place samples inside it depending on how confident the model predictions are. The final position is a linear combination of the "class anchors" on the outer rim based on the confidence value for each class. The authors give the example of springs that attract a sample to a class, i.e., the higher the confidence, the stronger the attraction is to that particular class. It is important to note that the position of a sample is not unique to a specific posterior probability, and several samples can fall into one place. Therefore, the mapping function is surjective, and the output is a 2-dimensional point, which results in visualizations similar to a scatterplot for multiple instances. Users can

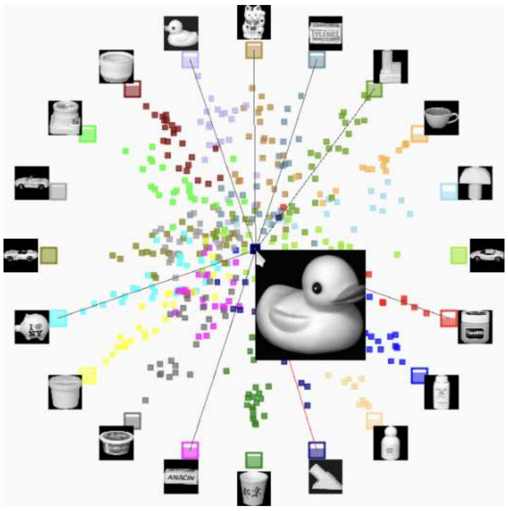


Figure 2.1: Visualization of posterior output probabilities for user-based active learning [SG10, p. 2].

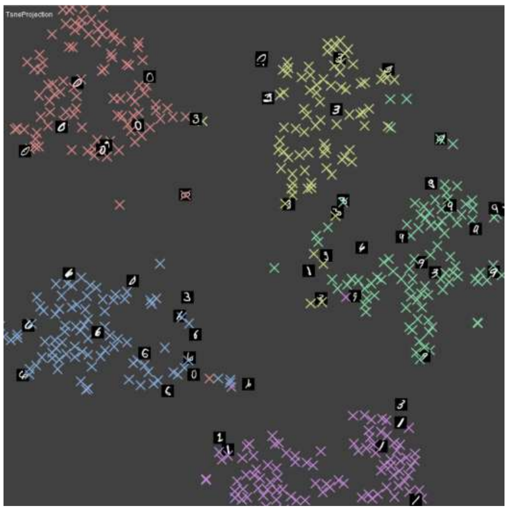


Figure 2.2: t-SNE projection used in VIAL interface [BHZ+18, p. 1].

then manually select points in the plot, where points closer to the center indicate that these instances are less certain. Instead of comparing user performance with a user study, they simulate two possible selection strategies and two label strategies, allowing them to evaluate more settings. The two selection strategies are Gaussian and Convex-hull. The Gaussian strategy selects points closest to the center. In contrast, the Convex-hull strategy constructs a convex hull around a class anchor and then chooses the point closest to the intersection point of the convex hull with the center. The labeling strategies are called “all” and “misses,” so all points chosen by the selection strategy are labeled, or only the incorrectly classified instances are labeled. They showed that with these strategies, users could outperform standard active learning methods in many cases. They also note that users could switch strategies while labeling using the visualization, which is impossible with a fixed active learning algorithm.

Another highly influential work is the paper that first introduces the visual interactive labeling framework (VIAL) by Bernard et al. [BZSA18]. The authors aim to combine model-centered and user-centered approaches into one unified and abstract process. The framework differentiates three output types: labels, the trained model, and domain knowledge. These outputs are generated with an iterative process that combines active learning with visual interfaces, either by visually guiding the AL process or by steering and verifying AL through visual means. More specifically, the framework is divided into six distinct steps:

- Step 1 - Preprocessing: This can consist of data cleaning or projecting data into more suitable representations.

- Step 2 - Learning Model: A model is trained on the user-provided labels and is also output at the end of the process.
- Step 3 - Result Visualization: Presents the current state of the labeling process to the user.
- Step 4 - Candidate Suggestion: This can either be guidance based on active learning or to help steer and verify active learning.
- Step 5 - Labeling Interface: This step is the basis of the interaction loop, where data is mapped to labels by the user.
- Step 6 - Feedback interpretation: The learning model could utilize more advanced feedback from the user, for example, relations between multiple instances. However, most commonly, the user only provides feedback through labeling.

They also analyze "pioneer" implementations of the VIAL framework and show that it can be successfully applied to labeling systems.

Bernard et al. [BHZ$^+$18] also evaluated a representative VIAL system directly against active learning. Their system is centered around projection scatter plots showing class predictions, class uncertainty, superimposed convex hulls, or superimposed butterfly plots. PCA [Jol03], non-metric MDS [Kru64], Sammons Mapping [Sam69], and t-SNE [vdMH08] were the projection methods available to users. An example of a t-SNE projection used in their system can be seen in Figure 2.2. Candidate suggestions are based on these plots, where users can decide which instances to label based on the spatial information of projection and additional information like class coloring. For the baseline, they simulated uncertainty-based active learning strategies, including query-by-committee, where several classifiers vote for the selected instance. They conducted a user study to see if VIL can outperform active learning, even in complex label settings, and when multiple instances can be selected. Furthermore, they evaluated whether users develop specific VIL strategies and how they relate to active learning. In the easy complexity setting, users had to label two classes. Medium and difficult tasks had 5 and 10 classes, respectively. The results indicated that for easy tasks, VIL outperforms active learning considerably; however, with medium and difficult tasks, the results are less clear, with VIL only being slightly better. All tested VIL-support techniques performed very similarly regardless of the setting. Class coloring based on model predictions performed marginally better than the other techniques. Most participants relied on the t-SNE projection for labeling – the default – and tended to use the others only for validation. Furthermore, participants were much more efficient when they could label multiple instances at once. Observing participants during the labeling process also revealed that participants used specific strategies while labeling. These included data-centered strategies like highest density, centroids first or equal spread, and model-based strategies like labeling class borders, class intersection, and class outliers.

Bernard et al. [BZL$^+$18] later formalized these strategies, which are central to the concept of cVIL and will be discussed in great detail in the following section. Before, we want to present additional relevant visual labeling systems.

Grimmeisen et al.[GCT22] extended VIAL with guidance based on formalized user strategies. In their work, guidance is facilitated through visual cues in the scatter plot, which assists the user in selecting specific instances to label instead of relying solely on projection or color mapping. They also indicate class prediction by color, similar to the class coloring support technique of Bernard et al. [BHZ$^+$18]. Purely user-based interactions can lead to bias, and some form of guidance can overcome this issue. Guidance in VisGIL is based on the information gain of an instance, which is calculated based on the model's uncertainty and the representativeness of a sample. Representativeness, in turn, is calculated based on the similarity of the instance to all other instances. This is realized as a density measure over the data. The authors consider two forms of guidance: orienting and directing. Orienting extends the VIAL plot by showing the utility through the glyph's size, directing further shows recommended instances for labeling using a special glyph, particularly a red star. The number of recommended instances in the directing setting is set to the number of classes k. The specific instances shown to the user are determined by finding k clusters on a subset of the data with high utility and selecting one point for each cluster closest to its center. Therefore, the user gets recommended k diverse instances to label at each labeling iteration. The authors performed a user study with orientation and guidance settings on a small subset of the MNIST dataset, with each class limited to 100 instances. The results indicate that guidance with direction outperforms guidance with orientation both in effectiveness and efficiency. The model accuracy improved with stronger guidance. They also found evidence that the accuracy per interaction increased and the percentage of correctly labeled instances increased. However, these results were not statistically significant. The results further showed that users thought the recommended instances had to be labeled and investigated less on their own. This might be the case because single instances were explicitly highlighted.

Desmond et al. [DAB$^+$21] studied instance-based label assistance by ordering the labels according to their predicted likelihood and showing the exact percentage values under the label, similar to a bar chart. The likelihood of each label of an instance is determined by label spreading, a semi-supervised labeling algorithm. In the user study, the participant's performance was evaluated with two different interfaces: a baseline without any ordering or likelihood visualization and one with both ordering and visualization of the likelihood based on classifier predictions. Furthermore, the user also tested two conditions of assistance. One where the model was trained on 21 examples, and the other was where active learning was used to train 200 samples. The latter will give better predictions. The task participants were given was in the domain of text classification with 21 labels and 420 instances in the dataset. Initially, 21 instances were selected as an example of each class, and users had to classify these training examples. Wrong answers were corrected with an explanation of why the chosen label was incorrect. Then, users labeled 79 more instances without any correction for incorrect labels. The study indicated that assistance significantly improves the results of labels, although there was little difference between weak and strong assistance.

Desmond et al. [DDB+20] also present a system that utilizes a semi-automated labeling approach that aligns much more closely with the paradigm of cVIL. Their system now uses active learning and the same label-spreading approach to implement an interactive label loop, similar to the other system discussed previously. The active learning selector chooses batches that the user labels using the assisted labeling interface. After the batch is labeled, the user reaches a checkpoint at which they can either improve the model further by labeling or apply automatic labeling to the remaining data by the model. The authors only performed an informal user study that did not allow them to draw firm conclusions.

Benato et al. [BGTF20] also try to combine automatic and user-based visual approaches for data labeling. They integrate label propagation with active learning to assist the user in labeling data. In particular, their system utilizes an autoencoder neural network to extract features, which are then again visualized as a t-SNE projection in a scatter plot. Moreover, the system allows the user to set a threshold, which divides the data into uncertain and certain instances. Label propagation is then applied to instances the user designated as certain, and the remaining instances the user determined to be uncertain must be further labeled manually. The higher the confidence threshold set by the user, the fewer instances must be labeled manually. High confidence points are displayed in the predicted class color, while low confidence points are colored in black. The user can freely choose which low-confidence samples to label next, similar to the systems before. These manual labels are then propagated to the remaining instances. The authors evaluate the system on MNIST and different subsets from microscope images of parasites, which provide a more complex label setting. They found a confidence threshold between 0.5 and 0.6 works well for all settings. With more complex datasets, the threshold is higher than for the MNIST dataset, which indicates that users do not like to delegate complex tasks as much. Their method reduces the labeling effort compared to fully manual settings and performs better for complex label settings than fully automatic approaches.

Yu et al. [YZS+15] propose another semi-automatic approach for distributed data labeling on the crowdsourcing platform Amazon Mechanical Turk. Their main contribution is the creation of a large image dataset. They achieve this using a semi-automatic approach by automatically labeling high-confidence positive samples and only using the low-confidence samples in the next label iteration. They treat each category as a binary classification problem. Therefore, it is acceptable to have an overfit classifier as long as the high-confidence positive predictions are correct since these are labeled automatically. Their system uses deep learning classifiers as predictors. The resulting dataset achieves an average precision of around 90% but has over 95% precision for specific categories. Training models using their new dataset improved the performance by over 20% compared to the baseline dataset. Using just the baseline, their model achieves an error rate of 28.6%. When both the baseline and their dataset are used, the resulting model has an error rate of only 22.2%.

Sun et al. [SLT17] explore visualizing the training progress of a classifier with several visualizations, including some specific to the domain of name entity recognition. Their Label-and-Learn system also utilizes active learning, and one visualization shows the information gain of an instance in a 2D scatterplot where one axis is the classifier uncertainty and the other is similarity. The similarity between any two items is the overlap in the words they share. Label-and-Learn is evaluated against a version using no visual information and only textual information. The results showed that the participants performed better in every aspect with visualizations available while labeling. Measures included the success of the classifier, understanding of the classifier's decisions, and self-assessed performance. Participants also rated the utility of each visualization for labeling. Surprisingly, the information gain visualization scored very low on average. However, some users reported that it was advantageous. The authors suspect that the visualization was less clear for some participants.

Chegini et al. [CBB$^+$19] designed a labeling system for multivariate datasets based on the VIAL process called mVis. A dataset can be explored using linked visualizations, and instances can be selected and labeled in any of the views available to the user. These views include parallel coordinate plots, scatter plot matrices, and projections like t-SNE. The user is supported by active learning and clustering algorithms that help to select suitable candidates for labeling. The authors analyze and discuss how linked visualizations can be integrated with machine learning models to guide the user.

Beil and Theissler [BT20] present an approach called Cluster-Clean-Label that also incorporates clustering into the VIAL process. As the name suggests, the data is first clustered and then cleaned using outlier detection so that the remaining instances in the cluster have a high accuracy. Finally, the user can add a label to each cluster. Before clustering, the data is projected to a lower dimensional space using PCA, where 90% of the variance is retained. The data is then visualized in a t-SNE projection scatterplot. This 2D projection is then used for clustering using the HDBSCAN [CMS13] algorithm. Cluster cleaning is done using an autoencoder. For each cluster, an autoencoder model is trained on the data in that specific cluster. The mean squared error (MSE) of the reconstruction of each instance in the cluster is sorted and displayed to the user for cleaning by removing instances with high MSE from the cluster. Therefore, cleaning is done with an outlier-based approach, after which the cluster is labeled. With Cluster-Clean-Label, the user can label large amounts of data with one action. However, this depends on good clustering and projection of the data. In the user study, participants were able to label around 91.5% of the MNIST dataset with an accuracy of around 99%. For clusters of the digit “9”, there was suboptimal label quality where a disproportionate number of samples were mislabeled.

Finally, Song [Son20] presents an approach that divides samples into annotation and verification sets. While the approach cannot be directly classified as interactive labeling because the label selection is determined by an adapted active learning selection criterion without any human intervention, the distinction between informative samples and samples for verification makes it nonetheless highly relevant in the context of cVIL. The annotation and verification sets are found using an optimization approach that incorporates the labeling efficiency (time cost) in the selection process. During the annotation phase, the user labels the suggested samples from the annotation set individually, similar to standard active learning. In the verification phase, the samples are partitioned into their predicted class labels, and the user can then verify the samples of each verification set and point out incorrect samples (outliers), which are then labeled in the next annotation phase. The verification phase stops once the user identifies three consecutive outliers in each verification set. This process is very similar to batch labeling in cVIL, with the distinction that the user cannot make arbitrary selections and cannot freely switch between instance labeling and batch labeling. The evaluation results show that the proposed selection process outperforms all baselines and achieves better accuracy at the beginning of the labeling process than random sampling. The system also improves labeling times compared to their previous system and baselines from the literature, as measured in the user study. On average, users only needed 328 seconds to label 400 images. Removing the verification phase drastically reduces the effectiveness of the system. The effect of the verification set only becomes negligible when the model predictions are poor. This confirms that having a component for efficient and fast labeling can be advantageous. Moreover, the system uses a learned semantic representation of the input images to train the downstream classifier. The embeddings of the images are determined beforehand, and the model is frozen during the labeling process. We discuss representation learning approaches for labeling in Section 2.4 in more detail.

2.3 Property Measures

After discovering specific strategies users adopted in visual interactive labeling, Bernard et al. [BZL+18] formalized these user strategies and analyzed their potential for labeling tasks. They found that data-based strategies can outperform active learning strategies initially, while active learning-based strategies are more robust and perform better in later stages of the training process.

Then Bernard et al. [BHR+19] generalized the concept of formalized user strategies into the notion of degree-of-interest functions in their later short paper. This paper establishes a taxonomy of these functions and presents building blocks to construct them systematically.

Bernard et al. [BHS+21] later published a more comprehensive analysis of degree-of-interest functions, which they now termed property measures (PMs), and provided a complete taxonomy fundamental for understanding these strategies. They identified 15 properties of the data or machine learning models applied to the data that can be used for instance selection. These properties enable the unification of instance selection strategies from the machine learning perspective, such as uncertainty-based active learning and user-centered strategies applied in a visually interactive labeling context. Property measures quantify given properties of the underlying system into a single numerical value. Property measures can be used to explain certain characteristics of the data as well as model decisions and development during the labeling process. The taxonomy is built along four orthogonal dimensions describing property measures.

The first dimension is the type of model output used. A property measure can use no model output at all, which means the property measure is only calculated using the data itself. If model output is used, it can either be group assignments (classification), posterior probability distributions, or scores (regression).

In the second dimension, property measures can differ in how instances are related to each other when calculating the measure. This can be done with only one instance, meaning only the model output is used to calculate the property measure. An instance can be related to its neighbors, instances within its group, or instances from other groups.

The third dimension, measure functionality, describes how model output and instance relations are combined to form the final property measure. As described before, either model output or instance relations can be ignored by model- or instance-only PMs. The third option is to use both model and instance properties to define the property measure. One example could be to take the distance between instances as weighting or use the model predictions of neighboring instances as an indication of diversity.

The fourth and last dimension is measure valance, which determines if the property measures are interpreted in ascending or descending order. So, for example, a distance measure is ascending, and model certainty is descending when estimating sample utility.

2.3.1 Explainability

Bernard et al. [BHS+21] also describe using property measures to facilitate explainability. Property measures can be used for explanations since they directly process information about the model and data, which can be visualized and provide insights during labeling. Any property of the data or model could be visualized and used to help users understand specific aspects of the underlying data or model.

This also applies to cVIL while the system retains the advantages like improved scalability. A straightforward example would be to utilize the model uncertainty to analyze for which instances the model works well and which attributes the model might misclassify by looking at samples with low or high values, respectively. Therefore, the interface can be used to analyze and correct model behavior. When using model uncertainty, cVIL is a simplistic interface for analyzing model performance through the posterior probabilities of the classifier like Alsallakh et al. [AHH+14] or Ren et al. [RAL+17]. This is also similar to Quality Assurance for Machine Learning approaches [ZTdJ+21], where the interface is used to validate an already trained model.

Property measures could be specifically designed to achieve any form of explainability within the system as long as the corresponding property measure can assign a score to each data point. However, due to the simplicity of the visualizations, the cVIL interface is less suited for developing and understanding the behavior of property measures for explainability compared to the explainer prototype by Bernard et al. [BHS+21].

Nevertheless, once the property measures are used in the labeling process, it can provide the user with the same information.

2.3.2 Property Measure Implementations

Here, we describe the prototypical implementation of each property in more detail. All property measures can be implemented with eight atomic functions, which can be composed to implement any of the 15 properties presented by Bernard et al [BHS+21]. In this section, we will express the property measures used in cVIL in terms of these atomic functions using the alphabetic designation given by Bernard et al. in Table 2 [BHS+21, p.19].

In cVIL, all data points are separated by their class assignment (Partition P), so all property measures have an implicit partitioning function applied. In the following, this initial partitioning is omitted from the description of the property measures. Staring with uncertainty-based active learning measures, seven out of 15 properties described by Bernard et al. [BHS+21] were implemented in cVIL.

The *Compactness*, *Variation*, *Collision*, and *Size* are not implemented because they operate on a partitioning of the data. Firstly, the data is already visually partitioned by class assignment, and there is no straightforward way to partition the data further within class assignments. Secondly, these property measures assign the same value for all data points in a partition, which is unsuitable for cVIL. The *Border* property cannot be easily implemented since Convex hulls, the most straightforward way to find class borders, cannot be applied to the data since the number of dimensions will often be larger than the number of instances. Finally, *Relevance* and *Agreement* properties are incompatible with cVIL due to their measure valance. These measures are inverses of *Uncertainty* and *Disagreement*, respectively.

Uncertainty

From the model posterior probabilities (Soft assignment: L) the minimum is used (Aggregation: A) to get the uncertainty of an instance. The same decomposition is used to represent **Min-Margin** and **Entropy**. All implement the *Uncertainty* property and are a direct implementation of the corresponding active learning selection functions described in Section 2.1.

$$PM(x) : L \to A$$

Eccentricity

From the data, the median and dimension-wise variance (Aggregation: A) is used to calculate the distance of each point to the median (Pairwise Measure: D), where each dimension is first scaled by the variance internally. The distance metric is implemented in SciPy [VGO+20].

$$PM(x) : A \rightarrow D$$

Separation

The data is clustered into a fixed number of clusters k (Partition: P), and the distances of each point to all cluster centers are calculated (Pairwise Measure: D). The median distance to the cluster centers (Aggregation: A) defines the property measure. The $*$ indicates that a function is applied multiple times in calculating the property measure, in this case, the distance to each of the k cluster centers. To speed up clustering, the input embeddings are projected to a lower dimensional space using PCA that preserves 75%

$$PM(x) : P \rightarrow D^* \rightarrow A$$

Density

For all k neighbors of a data point (Selection: R), the distance of the point to its neighbors (Pairwise Measure: D) is calculated. The average distance (Aggregation: A) is the density of a data point. This uses the average k-NN distance implementation of the pyOD [ZNL19] package. For all evaluations, a value of $k = 20$ was selected.

$$PM(x) : R \rightarrow D^* \rightarrow A$$

Outlierness

For each data point, an outlierness score (Scoring: S) is calculated. The specific outlier detection algorithm is ECOD [LZH+22], also implemented in the pyOD [ZNL19] package.

$$PM(x) : S$$

Disagreement

For all k neighbors of a data point (Selection: R), the classifier probabilities (Soft assignment: L) are compared to the output probability of the selected point using the Jensen–Shannon distances (Pairwise Measure: D) and the mean (Aggregation: A) of these distances defines the disagreement of a point in relation to its neighborhood. The nearest neighbor implementation of scikit-learn [PVG+11] is used and the distance metrics are implemented in SciPy [VGO+20].

$$PM(x) : R \rightarrow L^* \rightarrow D^* \rightarrow A$$

Coverage

For this measure, some training data (Selection: R) is used to calculate the distance of all data points to the training data (Pairwise Measure: D). The minimum distance (Aggregation: A) of a data point to the training data represents coverage. In cVIL, the batch labels are used as training data. When no training data is available, the mean of the whole data is used for calculating distances, which makes it identical to *Eccentricity* in that case. Again, the scikit-learn [PVG+11] implementation is used to find nearest neighbors, and distances are calculated using SciPy [VGO+20].

$$PM(x) : R \rightarrow D^* \rightarrow A$$

These property measures give users guidance on what instances to label. Instances with large property measure values should provide informative samples for instance-labeling. At the same time, data points with small values should already represent the class they were assigned to and, therefore, be suitable for batch labeling.

2.4 Representation Learning

cVIL uses DINO [CTM+21a], a representation-learning model, to extract features from images. This enables fast iterations of the labeling loop since the embedding vectors returned by the model are much smaller than the original images. Since the parameters of the backbone model are frozen, all embeddings are fixed and computed beforehand, increasing the system's throughput dramatically.

DINO uses a self-supervised approach to find vector representations of images without the need for any labels. The process is similar to knowledge distillation, where a so-called student model is trained using a fixed teacher model. Assuming the teacher model outputs a probability distribution, the student model is aligned with the teacher by minimizing the cross-entropy loss $H(a, b) = -a \log b$ between the output of the teacher model and the student model for the same input.

To adapt this approach to a self-supervised framework, the cross-entropy loss function is evaluated on several views (crops) V of the input image. The teacher model is evaluated on two large crops of the data (global views x_1^g and x_2^g) while the student model is additionally evaluated on several smaller crops (local views). The new loss function becomes the summed cross-entropy loss between the two global views from the teacher model and all the views from the student model. The loss function is given by:

$$\min_{\theta_s} \sum_{x \in \{x_1^g, x_2^g\}} \sum_{\substack{x' \in V \\ x' \neq x}} H(P_t(x), P_s(x')), \tag{2.7}$$

where P_t and P_s are the teacher and student models, respectively.

Therefore, the objective is for the output of a global view from the teacher and the local views from the student to converge. As a result, the student model learns to extract the same features from the local views to match the global views from the teacher model and thus learns robust representations of an image.

Moreover, the teacher model is not fixed but is trained along with the student model. This is done through a momentum encoder, i.e., the parameters of the teacher θ_t get updated with the parameters of the student model θ_s after each batch using the following update rule:

$$\theta_t \leftarrow \lambda\theta_t + (1-\lambda)\theta_s \tag{2.8}$$

The parameter λ follows a cosine schedule between 0.996 and 1.0 during training. So, for each batch, the student model only has a small influence on the teacher's parameters, but over time, the influence is strong enough that the model adapts to the student model. The architecture of the models is either based on a ResNet [HZRS15] or a vision transformer [DBK+20], with the exact same model architecture shared by the student and teacher model.

DINO outperforms other self-supervised methods in both linear and k-NN top-1 accuracy of classifiers trained on the features. Interestingly, the k-NN performance almost matches the linear performance. This phenomenon can only be observed when using vision transformers as the backbone model and with no other self-supervised method.

DINO also performs very well on image retrieval tasks, where a similar image should be retrieved from a database based on the vector representation of the input image [PCI+08]. It beats supervised methods on this task. Furthermore, DINO is also better in transfer learning tasks than models trained with supervision. For instance, pretraining with DINO improves the results on ImageNet [DDS+09] by around 1% compared to supervised methods.

Due to its properties, DINO is particularly well suited for the cVIL framework. Also because of the excellent kNN performance, which should allow distance-based PMs to work well. Coupled with the linear performance, the resulting features should allow the classification model and property measures to perform well and provide rich feedback to the user.

In the remainder of this section, we will present other applications of representation learning in labeling systems.

2.4.1 Representation Learning for Labeling

Bengar et al. [BvdWTR21] analyze whether self-supervised feature learning in the context of active learning can be beneficial. They use self-supervised learning to train a backbone model, freeze the weights, and use the features to train a linear model using active learning. Several informative and representative active learning sampling strategies are implemented as baselines, and results are compared with and without self-supervised pre-training. For all evaluated datasets, pre-training drastically reduces the human labeling effort. For a 1% labeling budget, pre-training achieves an accuracy of around 45% on the CIFAR100 dataset [Kri09], while using active learning only achieves an accuracy of just 10%. Only at 50% labeling budget, the active learning settings without pre-training reach the same accuracy levels as those using pre-trained features. Below a labeling budget of 1%, random sampling outperforms all active learning sampling techniques. Also, the more classes a dataset has, the better random sampling performs in the early stages of labeling.

Yi et al. [YsSPgC22] introduce a novel sampling technique for active learning with self-supervised pre-training. The loss from the pre-training task is used to divide samples into batches, which are then used to sample instances for active learning. Starting with the batch with the highest loss, the top-k instances of each batch are selected according to standard uncertainty-based sampling. This makes it a batch-mode active learning scheme since the user labels k samples before the classification model is retrained. The idea behind this approach is that the batches are supposed to split the data into subsets that are different from each other. In combination with uncertainty-based sampling within each batch, this should allow for both distributional sampling variety over all the data as well as optimal sampling at the decision boundaries in each batch. The proposed method outperforms all baselines on all evaluated datasets. Their method also works well on imbalanced data. It also lessens the effects of the cold-start problem by uniformly sampling from the batch with the highest loss at the first iteration instead of the whole data, which likely samples diverse or difficult instances. They also showed that their method indeed either samples from the class boundary (due to uncertainty sampling) or uniformly within the class distribution (due to batching).

Bai et al. [BCHS21] use self-supervised representation learning to construct a nearest neighbors graph for label propagation to reduce human labeling effort. Labels are queried by an active learning scheme and are used to get pseudo-labels through label propagation. With these pseudo-labels, the top-1 accuracy of a simple weighted k-NN classifier reaches 98.45% on CIFAR10 [Kri09] and 89.58% on CIFAR100, surpassing the results of Bengar et al. [BvdWTR21]. On CIFAR10, only 0.1% of the data needs to be labeled to reach the final target accuracy. Active learning performs much worse without pseudo-labels and only achieves the target accuracy after around 50% of the data has been labeled. Label propagation also performs well with noisy labels, where noise levels below 0.8 barely affect the final accuracy.

Lee et al. [LSB22] utilize representation learning in combination with active learning to improve the labeling efficiency of non-sematic speech tasks. Non-semantic speech tasks include speaker recognition, language detection, and speech-emotion recognition. Pre-trained models provide generic speech representations, which are used to achieve roughly the same target accuracy with only 600 labels compared to using the entire dataset for training, which contains 176.436 labels [Vox]. Their system uses a linear model trained on the representations to achieve this level of performance. Similar to previous approaches, the encoder model is not fine-tuned for any specific task, and all datasets were labeled using the same generic encoder model.

Schröder and Kiko [SK22] investigate the effectiveness of using representation learning on MorphoCluster [SKK20], a novel labeling system designed to maximize label efficiency. The goal is to label clusters of similar instances and then "grow" the cluster by selecting the first object in a list of recommended samples that are not similar to the seed images. Finally, the objects earlier in the list (i.e., those being similar) are added to the cluster. With this approach, the system enables the user to label millions of images. The original MorphoCluster approach used a supervised representation learning method with few labels of the original dataset. Now, the authors additionally evaluated the use of transfer learning on a labeled dataset if the target dataset is similar. If no similar dataset is available, unsupervised representation learning methods are the only possibility to get robust embeddings, which are also evaluated. In their experiments, unsupervised methods generally performed slightly better than transfer learning methods. With transfer learning, there was no single best source dataset. However, when the target domain labels are available (full supervision), the quality of the learned representations drastically improves. For clustering, unsupervised learning has better completeness scores than transfer learning. However, the number of clusters is much larger. This could indicate that the features are better aligned with the target domain when using transfer learning with the appropriate dataset. Supervised methods still outperform the other methods. This means that supervised representation learning methods are always preferable if labels are available for the target dataset. Unsupervised methods should be chosen over transfer learning approaches if this is not the case.

Perez et al. [PLM+22] utilize representation learning to efficiently label high-resolution astronomy data in combination with active learning. By comparing different representation learning approaches, they found that self-supervised learning performs much better than transfer learning or training a network from scratch when very few labels are available. With only 20 labels, the unsupervised method already shows a 75% agreement with humans, whereas the pre-trained or randomly initialized models perform not better than chance. After 500 labels, all methods perform very similarly and reach a performance that matches human agreement (85%) after 5000 labels. Furthermore, they show that freezing the network and using a linear classifier on the embedding matches the performance of fine-tuning the whole network. This is the case for any number of labels, but it drastically reduces the computational effort.

CHAPTER 3

cVIL: Class-centric Visual Interactive Labeling

In this chapter, we will discuss the concepts of cVIL in more detail. As mentioned before, cVIL is based on incremental labeling steps with the option to label large amounts of instances after verifying their correctness. The user labels individual data instances to get to the point where the model predictions are good enough for batch labeling to become possible. These labeling steps are repeated until the user has labeled everything or verified that the model predictions are correct for the remaining samples. The interactions are enabled by the underlying property measures presented in Section 2.3. The system should facilitate the requirements of scalability, interactivity, and data complexity, as discussed in the introductory chapter.

Batch labeling provides the model with many prototypical samples, similar to the approaches discussed in Section 2.4.1. The user can label informative samples using instance labeling based on the property measures described in Section 2.3. When using uncertainty-based property measures or other active learning selection strategies, the user can emulate active learning as described in Section 2.1. Above all, the user remains in complete control over how the labeling process evolves and can choose or switch strategies based on the information provided to the user, making it an interactive labeling as described in Section 2.2. Throughout the labeling process, the amount of unlabeled data is reduced, and the user, as well as the model, have to consider fewer remaining instances.

3.1 Components

This section provides an overview of the various components of cVIL. The interactions between the components are visualized in Figure 3.1.

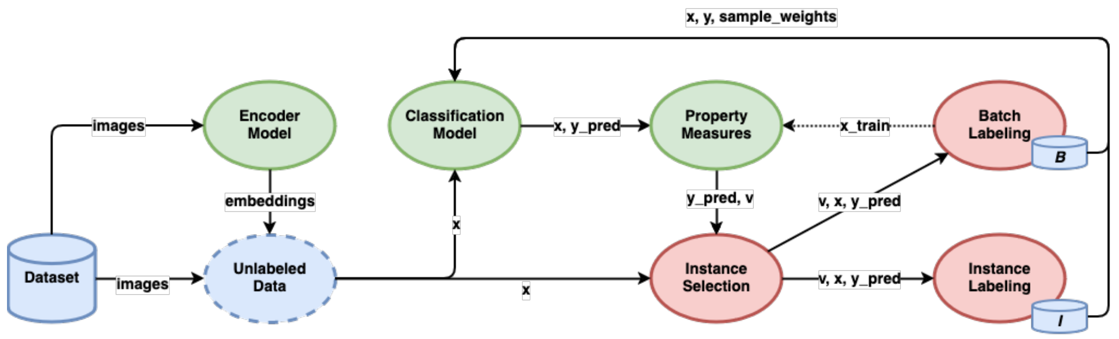


Figure 3.1: An overview of the different components of cVIL. Purely algorithmic components are highlighted in green, interface components are colored red, and data is shown in blue. Arrows visualize the flow of data. A sample x from the pool of unlabeled data contains the embeddings and the original image data. After labeling data, the labeled samples are stored in the corresponding instance label pool I or batch label pool B. These labeled samples are then used to train the classification model with the sample weights described in Section 3.1.2. The model's predictions are then used to update the property measures. Property measures, including the implementation details, were already introduced in Section 2.3. The property measure values v are visualized for instance selection in the corresponding class based on the model prediction.

3.1.1 Unlabeled Data

The unlabeled data contains raw image data as well as the embeddings of the representation learning model. All the algorithmic components (green) use embeddings, and the interface components (red) also display the original images to the user.

3.1.2 Classification Model

The classification model is trained on the embeddings along with the labels provided by the user. The relative simplicity and linearity of the embeddings [CTM$^+$21b] allows the model to be comparatively simple. The samples from instance labeling are weighted by how recently the user labeled them to give more importance to new labels. The reason for giving the most recent labels a larger weight is that they help correct specific model behavior. The samples chosen for instance labeling are usually those with large property measure values, which highlight attributes that contribute to outlierness within the class partition or are incorrectly classified. By giving these samples slightly more weight than earlier labels, we hope to reduce the number of labels needed to correct the model predictions for samples with these attributes. Additionally, we want to assign the lowest weight to the initial samples, which were selected randomly to bootstrap the system.

The model trains on both instance and batch labels. The batch labels are randomly sampled from the batch label pool. For each class in the batch label pool, $f_{batch} \cdot c_{min}$

samples are randomly selected at each training step. The value of c_{min} is defined as the number of samples of the class with the least amount of labels in the instance label pool. The sampling factor f_{batch} is set to 10. This means the number of batch labels used for training is approximately ten times the number of instance labels, given the user has generated enough batch labels. Otherwise, all available batch labels are used. To counteract the larger number of batch labels, they receive a constant weight of $\frac{1}{f_{batch}} = 0.1$, so as not to over-emphasize their influence.

The weighting function of instance labels is based on a scaled cosine function and is defined as

$$w(t) = \frac{1 - \cos(\pi t)}{2}, \tag{3.1}$$

where t is in the range $[0, 1]$ and is determined by the order in which the instances were labeled from 0.0 (earliest) to 1.0 (latest). For example, given four labeled samples, the value of t of these samples would be given by $[0, \frac{1}{3}, \frac{2}{3}, 1]$, with the corresponding samples weights of $[0, \frac{1}{4}, \frac{3}{4}, 1]$. If another label is added, the values t are recalculated accordingly to be equally spaced again. The weights are then used to scale the loss function for the corresponding sample. The choice of a cosine-based weighting function was motivated by assigning recent labels weights close to 1 before falling off. Any other function like *sigmoid* with similar properties would have been an equally valid choice. Finally, the weights are rescaled to the range of $[0.75, 1.0]$. The model then gives predictions of the remaining unlabeled data for the rest of the components.

3.1.3 Instance Selection

The instance selection is based on the property measures described in Section 2.3. The samples are partitioned into classes according to the predictions of the classification model.

In cVIL, instance selection is facilitated through a kernel density estimation (KDE) plot. The KDE plot shows the distribution of the selected property measure over all items that were predicted for the corresponding class. Therefore, it shows the property measure values on the x-axis and the estimated density of values on the y-axis. Property measures of samples that have already been labeled are excluded from the visualization.

The user can inspect images for labeling in a separate labeling view that displays images of a selection. The user can permanently select samples with a range selection, which selects all samples with property measure values in the x-range of the selection. The samples are displayed in descending order based on their property measure values, starting from the highest value in the selection. Once the user has made a selection, the user can label individual instances from the selection (instance labeling) or decide to label the whole selection (batch labeling).

The user can also hover over a point in the KDE plot to inspect samples with a similar property measure value. This hover preview is also shown in the labeling view, like the fixed range selection. Again, samples with an equal or smaller property measure value are displayed, and the hover preview can be used to validate samples quickly.

The KDE plot visualizes the output of one particular property measure. The selected property measure can be changed anytime, e.g., through a dropdown menu. The visualization can be updated instantly because values are pre-computed. The cache only needs to be updated after batch labeling or retraining the classification model since these are the only components that influence the calculation of the property measures.

3.1.4 Instance Labeling

The user can label individual samples in a range selection by assigning a class to an instance. Instance labels are stored in a separate instance label pool to calculate the specific sample weights for the classification model.

Users should consider the order of the samples determined by the selected property measure since this order is optimal according to some heuristics when choosing the most valuable item to label next. For all the implemented property measures, this is the instance with the largest property measure value.

Any labeling action, either instance or batch labeling, can be reverted with an undo action. Assigning incorrect labels to samples, especially when batch labeling, can drastically change model behavior. By undoing a labeling action, the added labels are removed from the respective labeling pool, and the model has to be retrained.

3.1.5 Batch Labeling

Batch labeling relies on the opposite idea: labeling the samples with the lowest property measure values within each class. These will most likely be prototypical samples of the class, either because of their distributional properties, e.g., being close to their respective class center, or their informativeness attributed to them by the classification model, e.g., their posterior probabilities. Users should select instances from the sample with the lowest property measure value up to a threshold, and the selection should ideally include only correct samples. Therefore, the user has to be able to inspect or validate the instances first. This can be achieved through the hover preview by inspecting the samples starting from the lowest property measure values.

3.2 Implementation

The prototype used for evaluation implements all components of the cVIL concept. It was implemented in Dash [Das], and the plots were created with Plotly [Inc15]. The user interface needs to facilitate instance labeling and batch labeling, as these depend on user input as well as instance selection through a visualization.

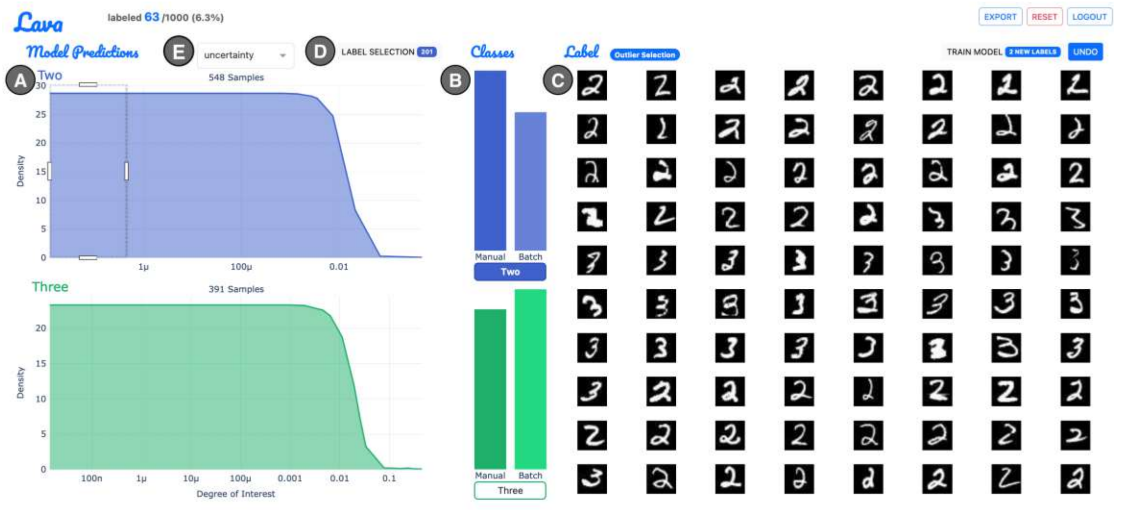


Figure 3.2: The prototype implementation of cVIL. The selected property measure is visualized using a KDE plot (A). The class distribution view (B) shows the number of labels for each class in a bar chart and allows the user to select a class for labeling. The class with the colored background has been selected, which is "Two" in this example. The labeling view (C) is used to label individual instances in a selection and shows the hover preview.

3.2.1 Unlabeled Data

The pool of unlabeled data is implemented as a PyTorch [PGM+19] dataset. The embeddings are found using the DINO ViT-B/8 model, which is based on a vision transformer with a patch size of 8x8. The resulting representations have a dimensionality of 768. All images are cropped to have a square aspect ratio and are up or downscaled to 224 by 224 pixels for inference with the DINO model. Furthermore, the values are normalized according to the mean and standard deviation of ImageNet to retrieve the embeddings. For MNIST, the original features are used directly, which have a dimensionality of 784.

3.2.2 Classification Model

The classification model is also implemented in PyTorch as a neural network with a configurable size of hidden layers. For the prototype, two layers with 50 and 20 neurons were used.

The labels should have as little class imbalance as possible to guarantee optimal performance. To that end, the class distribution view (B) helps the user to label an equal number of samples of each class. The label distribution view shows the instance label counts (left bars) and batch label counts (right bars) for each class. The instance label and batch label counts are independent, so instance labeling does not influence the visualization of the batch label counts and vice versa.

During training, a validation set is used to monitor the loss. Each time the model is retrained, 25% of the available labels generated by the user are used as a validation set for the current training step. The validation set is stratified and contains 25% of the samples in each class. The system uses early stopping with a patience of 20, so the model trains for 20 epochs after the validation loss stops decreasing. If a class has less than four labels, no validation set is used and early stopping is based on the training loss. The batch size for training is set to 64.

3.2.3 Instance Selection

The KDE plots (A) are stacked in the prototype and align with the corresponding class in the class distribution view (B). The displayed property measure can be changed with the dropdown menu (E) above the plots. The labeling view (C) is implemented as a simple grid of images.

A property measure can produce the same output for different inputs, for example, for instances with a very low uncertainty close to 0. This can lead to problems because each section in the KDE plot has to represent fewer samples than what can be shown in the labeling view for validation. When more samples have the same property measure value, this becomes impossible and would result in some samples never being shown to the user. To avoid this problem, the collections of samples with equal values have to be deduplicated. In the prototype, samples with more than one equal property measure value are interpolated using the next-largest value of the output to achieve this deduplication. This deduplication process breaks ties and provides samples with equal property measure value with a fixed and consistent ordering. An example can be seen in Figure 3.3.

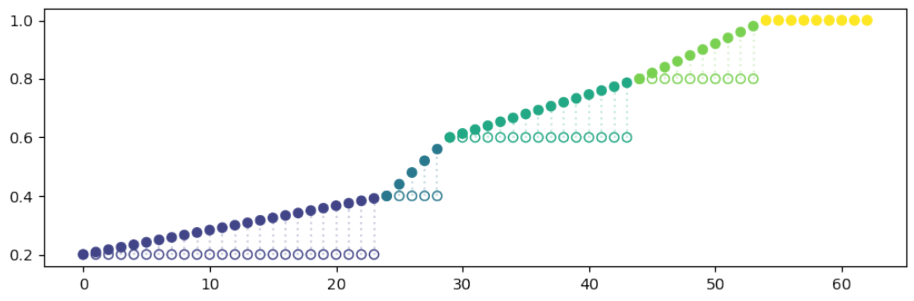


Figure 3.3: Illustration of the deduplication method used in the prototype. The points are interpolated to the next point with a different value. Before deduplication, values get rounded to eight decimal places to guarantee the required precision for interpolation. Note that the function does not map the last series of numbers to new values. However, in practice, the largest property measure will not require deduplication.

3.2.4 Instance Labeling

Once an instance selection is made, images can be labeled in the labeling view (C) by selecting a class in the class distribution view (B) and clicking on an image. A solid colored background indicates the selected class in the class distribution view (B). After labeling an image, it is added to the instance label pool and is removed from the labeling view (C), which gets updated with a new image from the current selection if one is available.

When one or more instances are labeled, the model can be retrained, after which the property measures get updated using the new posterior probabilities returned by the retrained model. In the beginning, when no samples have been labeled, each class needs at least one sample to train the model. Before, no visualization was displayed, and the labeling view (C) showed a random selection of images for the user to label to bootstrap the system.

3.2.5 Batch Labeling

Batch labeling assigns the label of the selected class from the class distribution view (B) to all instances in a selection. This is done through the batch label button (D) above the visualization. The samples are removed from the visualization, and the values of property measures in the corresponding class partition are recalculated immediately. The other class partitions not affected by the batch labeling action do not have to be updated.

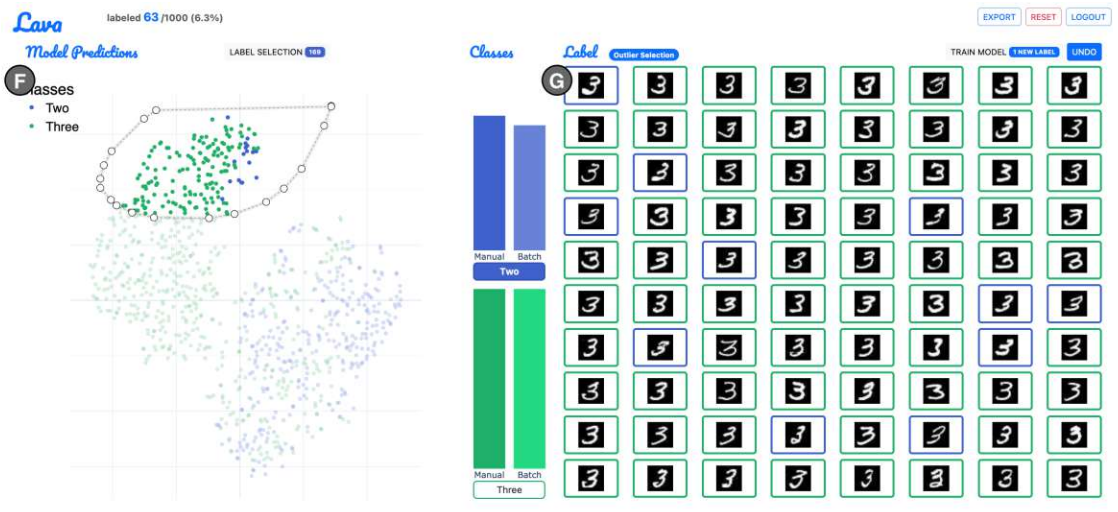


Figure 3.4: The prototype implementation of the iVIL system. The KDE plot from the cVIL prototype is replaced by a scatterplot of the t-SNE projected embeddings (F). Because the samples are not partitioned into classes, the predictions of the classification model are indicated by the border color of each image (G) in the labeling view.

3.2.6 iVIL Baseline Implementation

For the baseline iVIL interface, the KDE plot is replaced by a scatter plot showing a t-SNE [vdMH08] projection of the embeddings (F). The output of t-SNE is a 2-dimensional projection based on the 768-dimensional input embeddings from the DINO model for visualization in the scatterplot. This projection preserves the neighborhood structure from the original embeddings as best as possible. This is done by first converting Euclidean distances into conditional probabilities expressing similarity between two instances. This is also done for the output embedding, resulting in two probability distributions. The objective function then minimizes the Kullback-Leibler divergence between the distributions of the original and projected space.

The t-SNE projection is parameterized by a perplexity value, which controls the number of neighbors considered in the calculation. For the iVIL implementation, a perplexity of 30 was chosen. The scikit-learn [PVG+11] implementation, which was used for the prototype, allows the embedding to be initialized with a PCA projection. This option was chosen since it provides more stability.

The color of a data point encodes the class membership of the sample in the scatter plot. This also highlights an important distinction between the cVIL interface: cVIL is a class-based approach, while iVIL is an instance-based approach. The KDE plot from cVIL abstracts individual instances into distributions of the property measure values.

CHAPTER 4

Evaluation

To evaluate the performance of cVIL, we employ a twofold evaluation approach involving evaluation using simulated user interactions and a user study to assess actual user behavior and performance. By leveraging simulated user interactions, we aim to determine the theoretical performance of the system in various settings, compare the results to active learning, and evaluate batch labeling performance (**RQ1** and **RQ2**). The system's scalability (**RQ3**) is evaluated in isolation by running and measuring the execution time of the relevant calculations. The user study allows us to assess the real-world efficacy and usability (**RQ4**) of cVIL by testing it on actual users. Through this combined evaluation methodology, we aim to obtain a comprehensive understanding of the system's capabilities and its utility in practical data labeling tasks.

4.1 Simulation

To evaluate the system's effectiveness for a wide range of different settings in comparison to active learning (**RQ1**), we simulate user interactions in the labeling process (see Algorithm 4.1). Active learning can be easily simulated, which is regularly done in the literature [BHZ+18, SG10], by assigning the ground truth labels to the suggested instances (see Algorithm 2.1). This principle can be directly applied to any property measure as well (see *instanceSelection* Algorithm A.1), with the only difference being that the instances are selected from each class partition independently. In short, the $\lfloor \frac{k}{C} \rfloor$ instances of each class are selected, where k is the batch size and C denotes the number of classes. If C is a divisor of k, the final batch size will be k, as is the case for the following results. For example, for a batch size of 10 with ten classes like MNIST and STL10, a single instance of each class is selected. Analogously, a batch size of 10 for the binary datasets means that five samples of each class are selected in each iteration.

The only remaining user interaction is batch labeling, which is slightly more complex since it involves finding a suitable cutoff point in the KDE plot. This is implemented as

Algorithm 4.1: Simulation of user interactions

```
Input: unlabeled data $D$, initial labeled data $L$, number of batches
       $nBatches > 0$, boolean flag $enableBatchLabeling$, list of classes $\mathcal{C}$,
       instance selection size $k$
Output: accuracy at each iteration
1  $B \leftarrow \emptyset$
2  $acc \leftarrow \emptyset$
3  $f_\theta \leftarrow initModel(L)$
4  $b \leftarrow 0$
5  while $b < nBatches$ and $D \neq \emptyset$ do
6      $\{x_1^*, x_2^*, ..., x_k^*\} \leftarrow instanceSelection(D, f_\theta, L, B, k, \mathcal{C})$
7      $\{y_1^*, y_2^*, ..., y_k^*\} \leftarrow oracle(\{x_1^*, x_2^*, ..., x_k^*\})$
8      $D \leftarrow D \setminus \{x_1^*, x_2^*, ..., x_k^*\}$
9      $L \leftarrow L \cup \{(x_1^*, y_1^*), (x_2^*, y_2^*), ..., (x_k^*, y_k^*)\}$
10     $f_\theta \leftarrow trainModel(f_\theta, L, B)$
11     if $enableBatchLabeling$ then
12         for $c$ in $\mathcal{C}$ do
13             $\{x_1', x_2', ..., x_b'\} \leftarrow batchSelection(D, f_\theta, L, B, c)$
14             $D \leftarrow D \setminus \{x_1', x_2', ..., x_b'\}$
15             $B \leftarrow B \cup \{(x_1', c), (x_2', c), ..., (x_b', c)\}$
16         end
17     end
18     $acc \leftarrow acc \cup accuracy(f_\theta, L, B)$
19     $b \leftarrow b + 1$
20 end
```

the *batchSelection* Algorithm 4.1. By evaluating cVIL with and without batch labeling, we can determine the effectiveness of batch labeling (**RQ2**). All simulation runs were initialized with a model trained on a single instance per class. To find the cutoff point for the *batchSelection* procedure, we can use several methods:

1. **optimal**: This represents the best-case scenario where the best possible cutoff point is chosen using the ground truth labels. The cutoff is selected at the first sample with an incorrect label starting from the smallest property measure value.

2. **greedy**: This method starts from the largest property measure value and looks at a small window of a specific size that slides over the samples. The cutoff is selected where this window first reaches 100% accuracy.

3. **balanced**: Take the average of the optimal and greedy approaches for the cutoff point.

The selection is then created from the smallest property measure value (index 0) to the index found by the batch selection method. The procedure is described in Algorithm A.2.

4.1.1 Datasets

The evaluations are based on three datasets:

1. **MNIST**: The MNIST dataset contains 60000 images of handwritten digits from 0 to 9. It is commonly used as a benchmarking dataset [BT20, BHZ$^+$18, BGTF20] for labeling tasks. In addition to the complete dataset, a subset containing only 1000 samples of the digits 2 and 3 is used for the introductory task in the user study (MNIST23). Since the resolution of the images is only 28 by 28, the raw features of the images are small enough to use directly without any prior embedding.

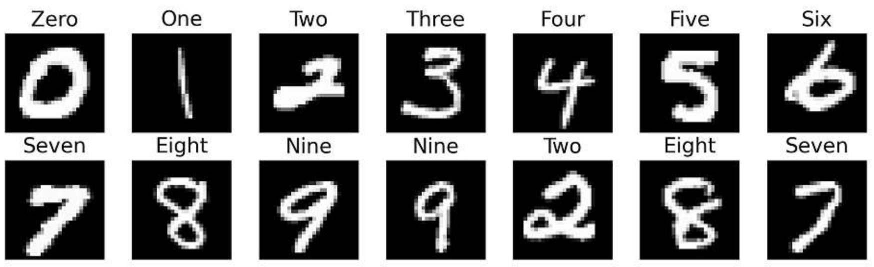


Figure 4.1: Sample images from the MNIST dataset.

2. **STL10**: The STL10 dataset includes only 5000 images in total but with a resolution of 96 by 96, and the images are in color. The dataset also has ten classes and was chosen because of its relative complexity and because the representation learning model works well for natural images out of the box, so no further pre-training is necessary.

3. **CelebHair** and **CelebGlasses**: These datasets are based on subsets of the popular CelebA dataset [LLWT15]. The subsets are either based on the person's hair color (CelebrityHair) or whether or not the depicted person wears eyeglasses (CelebGlasses). For the CelebHair dataset, only gray and black hair are used, which means both datasets are binary. The number of samples is fixed to 2000, with precisely 1000 samples in each class. The image size is 218 by 178 pixels, which are then center-cropped to 178 by 178 pixels. These datasets are also used for the user study.

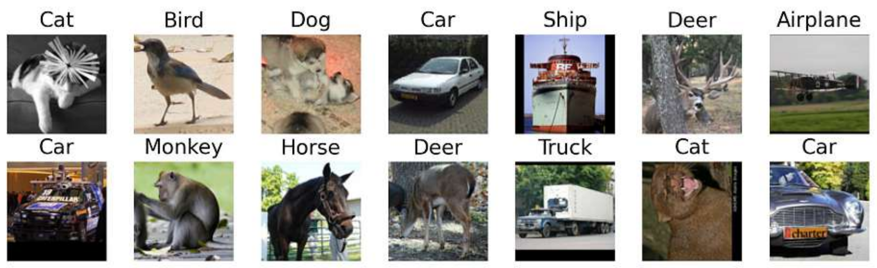


Figure 4.2: Sample images from the STL10 dataset.

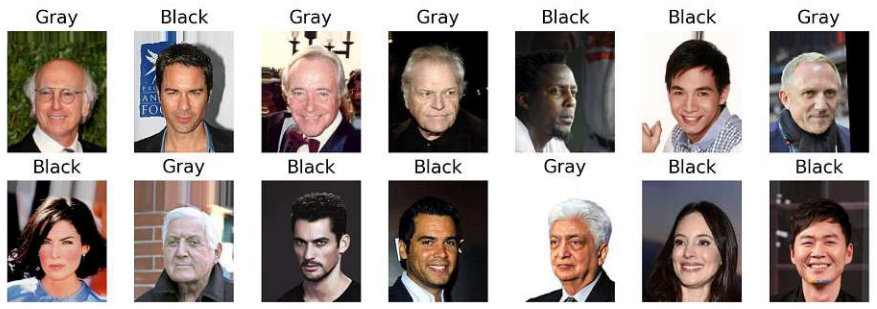


Figure 4.3: Sample images from the CelebHair dataset.

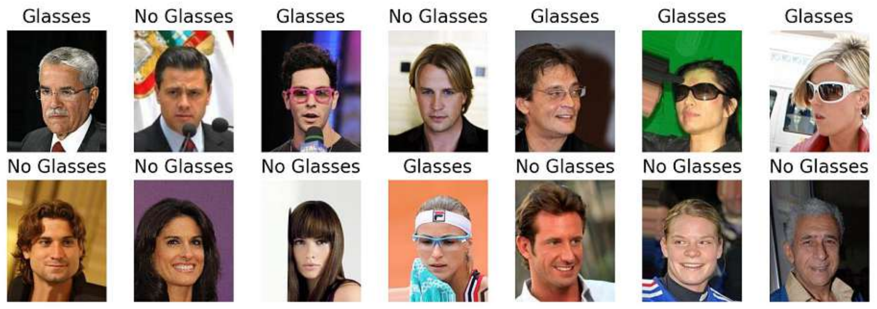


Figure 4.4: Sample images from the CelebGlasses dataset.

4.1.2 Independent Variables

The independent variables of the experiment are the batch or instance selection methods, the model complexity, the batch size, as well as the different property measures we wish to evaluate.

Batch Selection Cutoff Algorithm

The different choices of the batch selection algorithms will be influential to the quality of the selections. This will, of course, not only affect the accuracy of the labels but will, in turn, also influence the model and the quality of the training data of the property measures. The size of the sliding window can further determine the quality of the greedy and balanced approaches. Since the sliding window emulates the user behavior of looking through the samples, the window sizes were 30 and 60.

5 levels:

- optimal
- greedy (window size 30)
- greedy (window size 60)
- balanced (window size 30)
- balanced (window size 60)

Instance Selection Algorithm

The method used for the instance selection simulation also directly influences the model performance, which is crucial for all other components of cVIL. We can evaluate all selection algorithms from Section 2.1 and any property measure from Section 2.3. To simulate instance selection, a certain number of instances with the highest property measure value are labeled automatically for each class or, in the case of active learning simulations, across all classes. When evaluating property measures for batch selection, we set the instance selection method to Uncertainty for all batch labeling simulations.

Model Complexity

The model is implemented as a neural network consisting of an input layer with the same dimensionality as the embedding space and an output layer predicting class membership. The network contains either no hidden layers, one hidden layer with 50 neurons, or two hidden layers with 50 and 20 neurons. The learning rate of the model is fixed to 10^{-4} and has the weight decay set to 10^{-7}.

3 levels:

- ()
- (50)
- (50, 20)

Batch Size

We also vary the batch size for active learning before the model is retrained. Typically, the model is retrained after each labeling step. To assess the performance of larger batch size, we also performed retraining only after ten labeled samples were collected.

2 levels:

- 1
- 10

Property Measures

The goal of the simulations is to evaluate the performance of the different property measures for instance labeling and batch labeling. Therefore, the choice of the property measures is also a factor in the evaluation. We evaluated all property measures described in Section 2.3.2 and random sampling and uncertainty sampling for active learning.

4.1.3 Dependent Variables

The dependent variables tested were the combined label accuracy throughout the simulation and the size of the batch labeling pool. The accuracy is calculated after each labeling iteration to analyze how the accuracy evolves as more instances are labeled and the model is retrained after each step.

4.2 Runtime

For runtime analysis (**RQ3**), the property measures are applied to synthetic data where the number of samples and dimensions can be varied. The artificial data consists of two clusters with a Gaussian distribution, one where the center is at the origin and one where the center is at position 2.5 in all dimensions. The standard deviation of the first cluster is set to 3, while the second cluster has a standard deviation of 2. We generate the first cluster with a fixed size of k samples and then use the same amount for the second cluster, resulting in a total of $n = 2k$ samples. If training data is used, the first $\lfloor \frac{k}{2} \rfloor$ samples from the first cluster are selected as training data. For example, to create a test dataset with 64 samples, each cluster has 32 instances, and the training data consists of 16 samples from the first cluster. In the following figures, we use the size of the total training data n, so 64 in this example.

The runtime is evaluated at increasing powers of two, starting from $n = 64$ samples to $n = 2^{14}$ samples. In addition to testing the runtime on different dataset sizes, the dimensionality varies from $p = 64$ dimensions up to $p = 2^{14}$. This results in 81 combinations of n and p, where each combination is evaluated five times to get a robust average.

4.3 User Study

The user study aims to compare the results from actual users using the prototype described in Section 3.2 to the baseline iVIL implementation. The user study, therefore, serves to test the system's effectiveness on actual users in a real-world scenario as well as its usability (**RQ4**) compared to the iVIL implementation.

4.3.1 Hypothesis

To ensure the system's functionality, the user interface has to enable two crucial interaction techniques: instance labeling and batch labeling. Users have to be able to apply the theoretical concepts to an actual labeling task. We hypothesize that the cVIL prototype facilitates these interactions and outperforms iVIL for interactive labeling on complex tasks.

Since cVIL is a synthesis between active learning and visual interactive labeling, the user is neither completely reliant on the model to solve the task nor on their label selection performance. Thus, we further believe that users feel less likely to be overwhelmed when solving a task compared to the baseline, and participants prefer the cVIL prototype over the baseline.

In short, the hypothesis is that the proposed system is more effective at labeling than the baseline while being less demanding for the user.

4.3.2 Study Design

Tasks

To test the hypothesis, test subjects have to perform the given labeling task from start to finish. Participants have to label images, train the model, and decide if, when, and what to batch label. The users are expected to label the data as accurately as possible, with no strict time constraints. Users were tasked to maximize the accuracy of the final labels, whether assigned by the model or by instance or batch labeling manually. Only the *Uncertainty* property measure was available to the participants to reduce the complexity of the tasks. Due to its simplicity, it is easy to understand even for inexperienced users. While changing property measures during the labeling process would be preferable, we decided against it for simplicity.

		Dataset	
		CelebHair	CelebGlasses
Interface	iVIL	C1	C2
	cVIL	C3	C4

Table 4.1: The four task conditions by dataset and labeling interface.

Independent and dependent variables

The study used a within-subjects design, where the interface is the independent variable, either cVIL or iVIL. Users solve a single labeling task in each implementation where the dataset is different for each task to limit learning effects. The datasets used in the study are the **CelebHair** and **CelebGlasses** datasets described in Section 4.1.1. This gives conditions C1 to C4, as seen in Table 4.1. Users always label both datasets with different interfaces. Furthermore, the order of the systems is alternated between participants. This results in 4 label settings to which a participant is randomly assigned.

- **S1**: C1 → C4
- **S2**: C2 → C3
- **S3**: C3 → C2
- **S4**: C4 → C1

The dependent variables are the accuracy of the generated label output, labeling time, task load, and preference to compare cVIL to the iVIL baseline.

Data

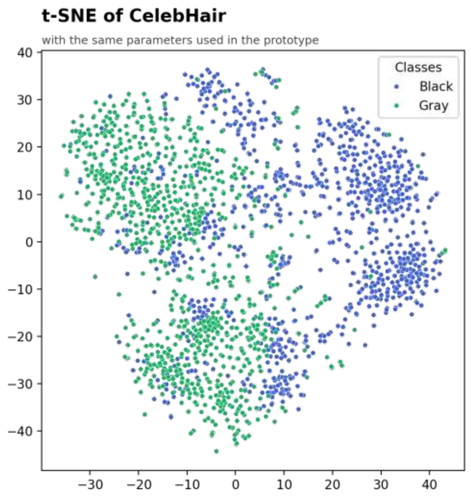
Figure 4.5 shows the t-SNE projections of the CelebHair and CelebGlasses datasets. The parameters for the t-SNE algorithm were identical to the ones used in the user study, so these visualizations mirror the plots a participant sees while solving the labeling task in iVIL. However, here, we show the ground truth labels instead of the model predictions. As we can see, the projection does not produce clean clusters. Instead, the data is quite noisy, which means the labeling task is sufficiently complex and goes beyond a simplistic task where the spatial layout of the projections already provides most of the information needed for labeling, as would be the case for MNIST or STL10.

Procedure

The study is conducted in person to reduce the latency of the system. This is particularly important when updating images based on user selections or hovering because it is essential for efficiently finding the cutoff for batch labeling.

The procedure was as follows:

1. Participants were asked to sign a GDPR form giving their consent for anonymous data collection during the study. Then, participants were given a short questionnaire, which collected information regarding demographics, experience, and other relevant factors.



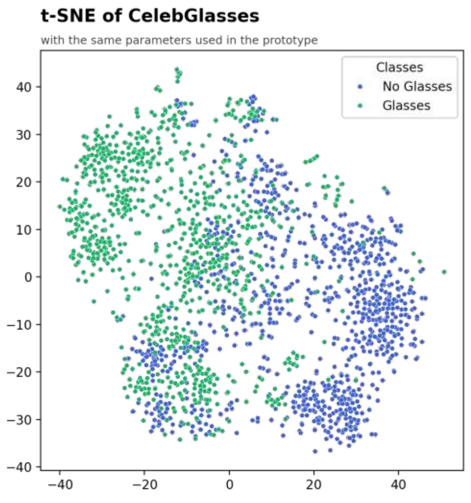


Figure 4.5: t-SNE projection of the CelebHair and CelebGlasses datasets. The parameters provided to t-SNE were identical to the prototype.

2. Before starting the label tasks, participants were introduced to the different components of the system and how to interact with them through a tutorial sheet.
3. Participants then solved a simple introductory task (labeling MNIST23) to further familiarize themselves with the system. The task did not have to be finished, and the participants could decide when to move on to the actual task.
4. Participants then solved the labeling tasks according to their assigned setting. They were given no time limit and had to decide when each task was solved.
5. After each task, they were asked to fill out a task load questionnaire (NASA TLX [HS88]) to asses how demanding the particular task was perceived. In particular, we used the aggregated TLX score described by Rubio et al. [RDMP04]. A detailed description can be found in the appendix A.2.
6. After solving both tasks, the participants are asked which system they prefer and what they liked and disliked about each interface.

To summarize, we described a simulation framework that can easily evaluate all property measures on various datasets with different conditions. The user study is conducted to compare cVIL to the baseline iVIL interface. Furthermore, the user study results can give insights into how real-world performance differs from the simulated results.

CHAPTER 5

Results

In this chapter, we will present the results from the evaluations described previously. In Section 5.1, the simulation results for active learning, cVIL instance labeling, and cVIL batch labeling are analyzed. The runtime of different property measures will be evaluated in detail in Section 5.2. Section 5.3 covers the results from the user study. We first compare the accuracy scores to the simulations before comparing the class-centric and instance-centric interfaces with each other. The task load and qualitative results from the final questionnaire are also discussed.

5.1 Simulation

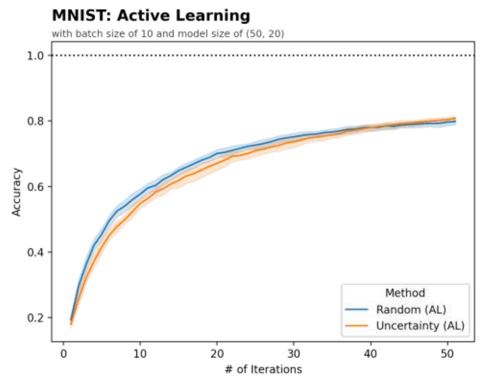
The results are primarily presented as accuracy curves as described in Section 2.1.1. Multiple runs were aggregated to get more robust average results and to show variability.

5.1.1 Active Learning

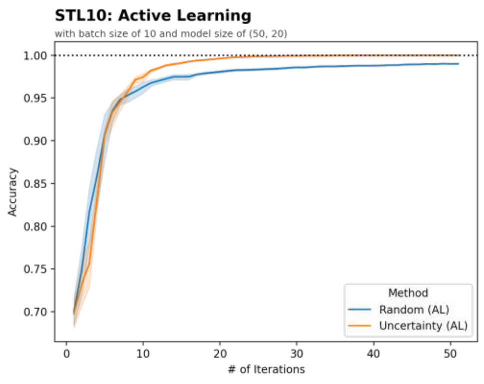
Active learning serves as the baseline for the simulation results, and two selection strategies were evaluated. The first was random selection, where a sample is chosen randomly from the pool of unlabeled data. On average, any selection strategy should outperform random sampling. The second strategy is uncertainty sampling u_{UNC} as described in Section 2.1.

Active learning is also used to compare different model sizes as described in Section 4.1.2. The model's overall performance will not fundamentally change, regardless of whether active learning or cVIL selection strategies are used, which makes active learning better suited for this evaluation. Moreover, active learning is also helpful for analyzing the effect of the batch size as it is not constrained by class partitions, where the number of classes directly gives the smallest possible batch size.

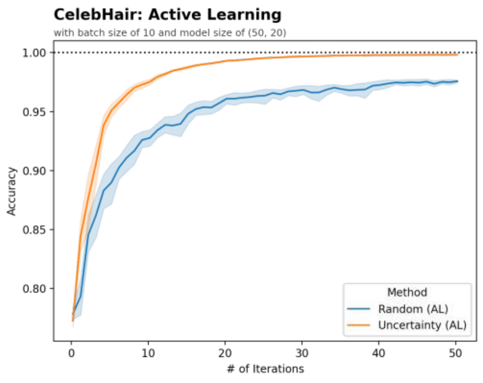
Uncertainty sampling generally performs better than random sampling, as can be seen in Figure 5.1. For MNIST and STL10, random sampling performed better than Uncertainty



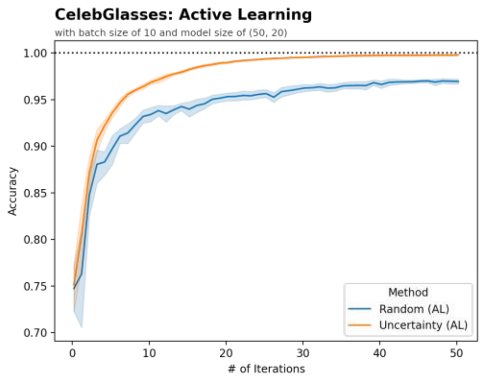
(a) MNIST



(b) STL10



(c) CelebHair

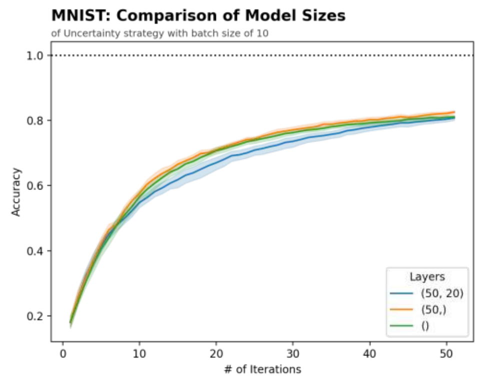


(d) CelebGlasses

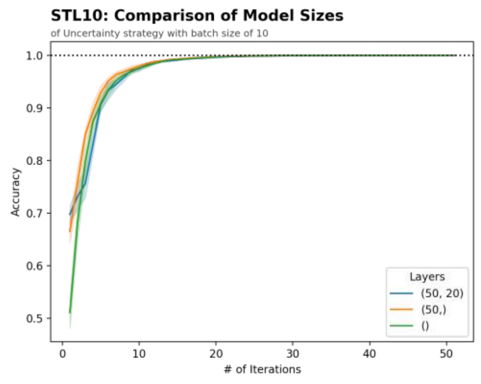
Figure 5.1: Comparison of Random and Uncertainty Sampling.

sampling initially. This is consistent with the results from Bengar et al. [BvdWTR21], which observed the same behavior in their evaluation. For STL10, the advantage is relatively small, and after around 10 iterations, Uncertainty sampling starts to outperform random sampling, whereas, for MNIST, this is only the case after around 45 iterations. This means random sampling performs better for most of the labeling process. For CelebHair and CelebGlasses, the uncertainty-based sampling already performed better initially and outperformed random sampling by a large margin throughout the labeling process. The uncertainty strategies quickly reached 100% accuracy for STL10, CelebHair, and CelebGlasses. However, both selection strategies only achieve around 80% accuracy after 500 labels for MNIST. This demonstrates the effectiveness of the DINO embeddings for classification tasks.

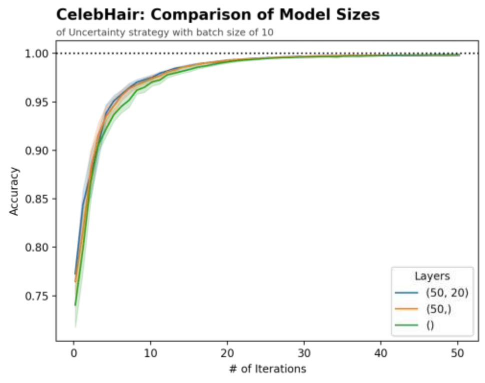
Figure 5.2 shows the comparison between different model sizes, and it is apparent that no model size has a clear advantage on all datasets. We denote the different sizes of the classification model as small, medium, and large in this context, which refers to the models (), (50,), and (50, 20), respectively, as described in Section 4.1.2. For MNIST, the large model performed worse for most of the training process and only caught up at the



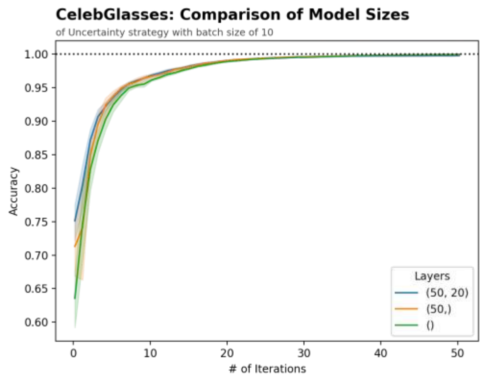
(a) MNIST



(b) STL10



(c) CelebHair

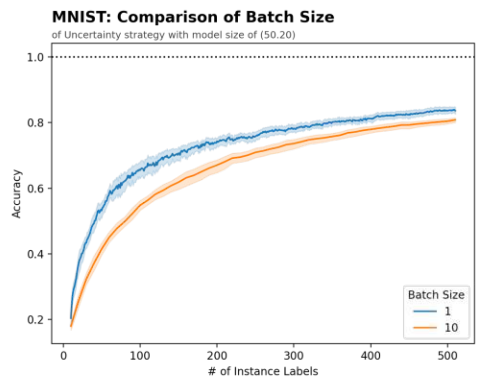


(d) CelebGlasses

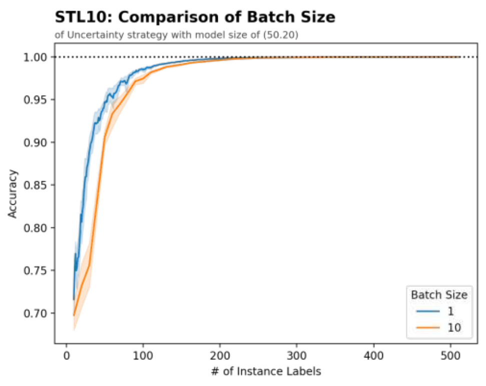
Figure 5.2: Comparison of different model sizes.

very end. The medium model performed marginally better than the small model. For STL10, the model size only affected the accuracy at the beginning of the labeling process, where the medium model performed better. In contrast, the large and small models had practically equal accuracy curves. For CelebHair and CelebGlasses, the large and small models are also very similar. However, they now outperform the medium model.

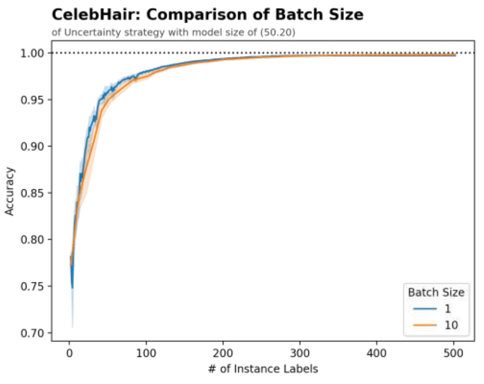
When comparing batch sizes, we find that a batch size of 1 performs better than a batch size of 10. For MNIST, this difference is considerable and remains so even after 500 labels, as shown in Figure 5.3. The difference quickly becomes negligible for all other datasets after around 250 labels. For CelebHair and CelebGlasses, the difference is relatively small, even in the beginning. This is probably due to the variability of the data. Even with a batch size of 10, the samples within a batch remain relatively diverse because of various other attributes present in the data, like age, gender, or accessories like hats, etc.



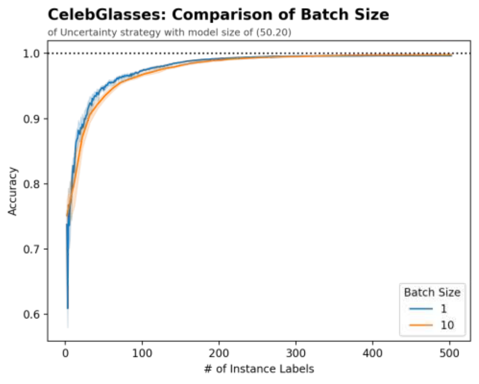
(a) MNIST



(b) STL10



(c) CelebHair



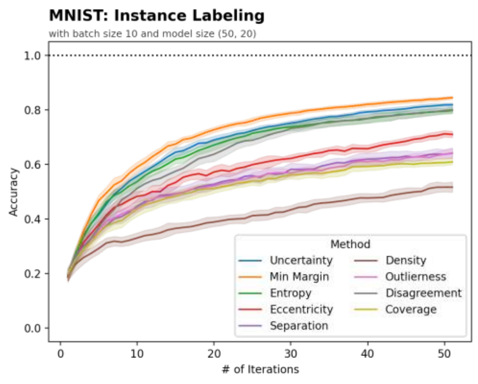
(d) CelebGlasses

Figure 5.3: Comparison of different batch sizes.

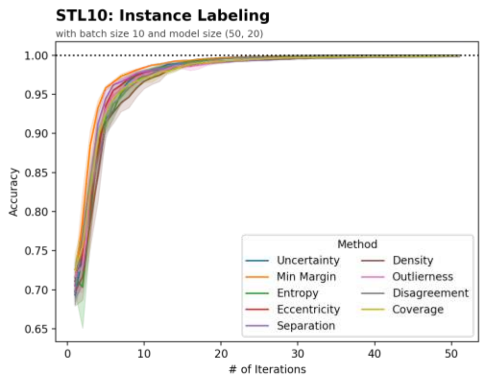
5.1.2 Instance Labeling

Figure 5.4 shows the accuracy curves of the instance labeling simulations. For MNIST, the difference between methods was quite notable. Density performs worst, followed by the other data-based strategies. Uncertainty-based strategies like Min Margin and Disagreement performed much better. When evaluated on the STL10 dataset, all methods performed similarly, with no method showing clear advantages. For the CelebHair and CelebGlasses datasets, uncertainty-based methods, including Disagreement, again outperformed all other methods by a large margin.

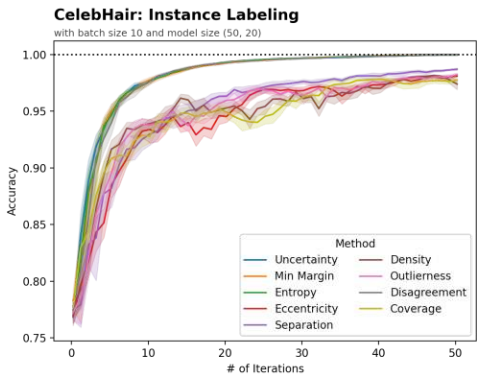
To better visualize the difference between property measures, we also plot the difference of each method to the overall mean instead of the absolute accuracy, as shown in Figure 5.5. The dotted line represents the mean accuracy of all methods at a particular point in the labeling process. For MNIST, all data-based methods perform below average, except Eccentricity, which is very close to the mean. For STL10, the Min Margin property measure performs significantly better than other methods initially, which was not as clear before. However, all methods converge after 25 iterations, and their difference becomes



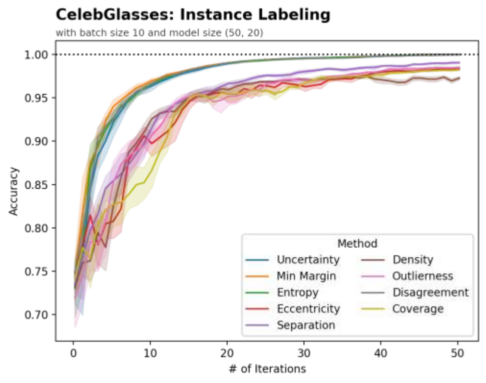
(a) MNIST



(b) STL10



(c) CelebHair



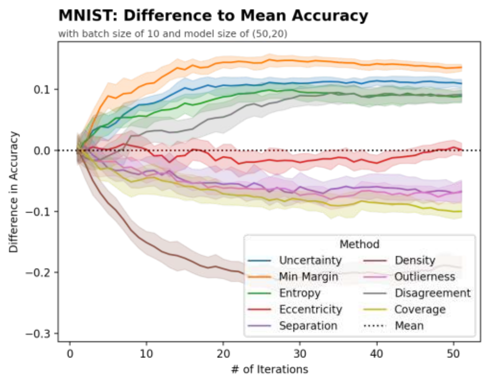
(d) CelebGlasses

Figure 5.4: Accuracy curves of property measures.

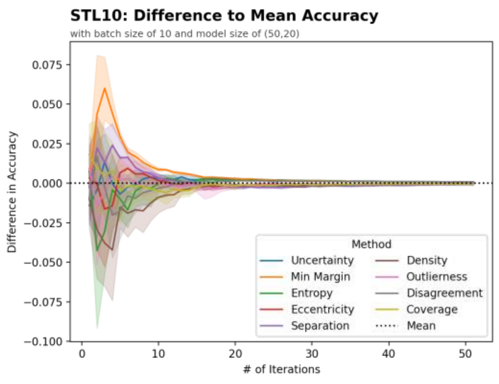
negligible. For CelebHair and CelebGlasses, the uncertainty-based methods perform better than the data-based methods. The difference between the uncertainty-based methods for both datasets becomes practically zero after ten iterations. This is to be expected since they are equivalent for binary classification tasks. They are also the only methods that reach 100% accuracy as shown in Figure 5.4c and 5.4d. For STL10, all methods converge at around 20 iterations, which also reach 100% accuracy at that point.

The cVIL instance labeling results can also be compared to the previous active learning simulations. In Figure 5.6, the difference in accuracy to the active learning Uncertainty simulation is visualized. This is especially interesting for the cVIL Uncertainty method, which allows us to directly compare cVIL instance labeling to active learning to see how class partitioning affects accuracy.

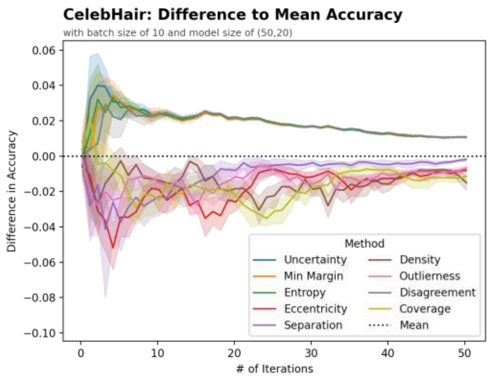
For MNIST, cVIL instance labeling with the Uncertainty property measure consistently beat the active learning counterpart. However, as we have seen before, Min Margin performed better than Uncertainty and outperformed active learning by a large margin. For STL10, all methods initially outperformed active learning but quickly converged as



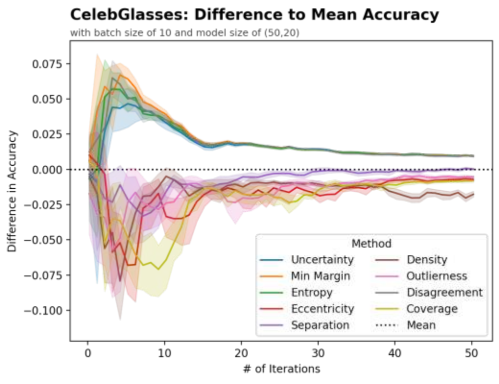
(a) MNIST



(b) STL10



(c) CelebHair



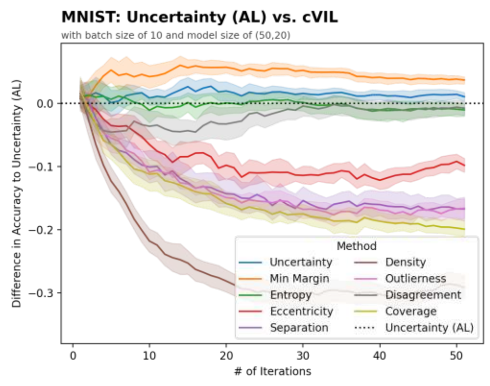
(d) CelebGlasses

Figure 5.5: Accuracy difference of each property measure to the overall mean when evaluated on instance labeling.

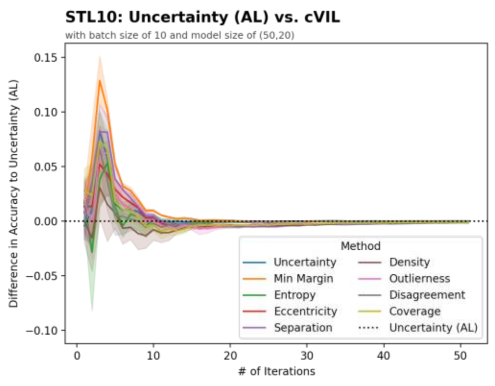
the accuracy approached 100%. For CelebHair and CelebGlasses, all uncertainty-based methods performed equally well (again due to binary classification), and no significant difference to the active learning results can be seen. With these binary classification problems, there should be samples from both classes in the active learning batch selection, so the class partitioning had a more minor impact compared to MNIST and STL10. A possible conclusion is that class partitioning helps for tasks with more classes where partitioning balances the labels across classes. The advantage of Uncertainty and Min Margin is kept throughout the labeling process in the MNIST simulation as the accuracy never reaches 100%. As for STL10, the advantage is considerable in the beginning across all methods, which indicates that partitioning works well.

5.1.3 Batch Labeling

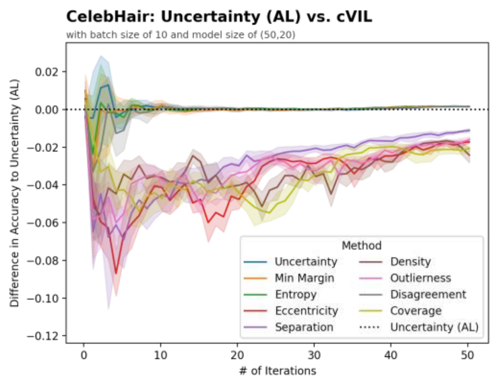
In the simulations for batch labeling, the property measures showed more similarity to each other. MNIST exhibits the most variability between methods, while the results are



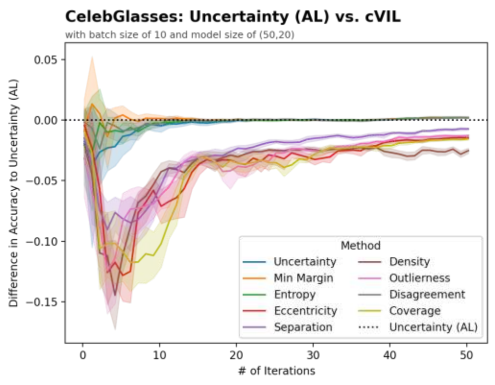
(a) MNIST



(b) STL10



(c) CelebHair



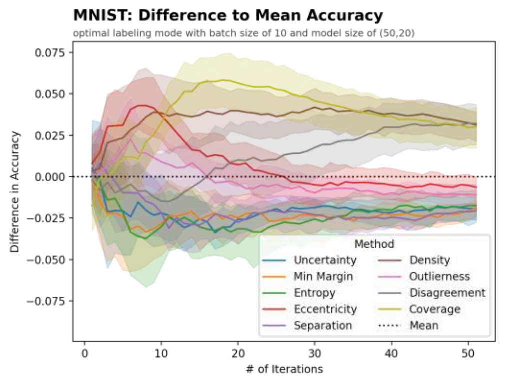
(d) CelebGlasses

Figure 5.6: Accuracy difference to uncertainty selection strategy u_{UNC}.

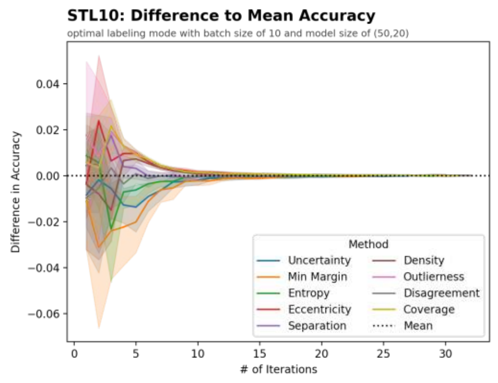
relatively similar for STL10, CelebHair, and CelebGlasses datasets. Because of batch labeling, the model has access to many prototypical samples during training, and which method is used for the selection appears less relevant. Note that the Uncertainty property measure was chosen as the designated instance labeling method for the following batch labeling simulations.

Figure 5.7 shows the difference of all property measures to the mean. It can be observed that data-based methods are more effective than uncertainty-based methods for the MNIST dataset. However, Disagreement catches up in later stages while Eccentricity and Outlierness begin to decline as training progresses. Separation is the only data-based method that underperforms from the beginning and matches the performance of the uncertainty-based methods. For STL10, there is only a small difference at the beginning, where the data-based methods again outperform model-based strategies. The results for CelebHair and CelebGlasses only show a small difference initially, and the variance makes it impossible to draw definitive conclusions about the performance of any single method.

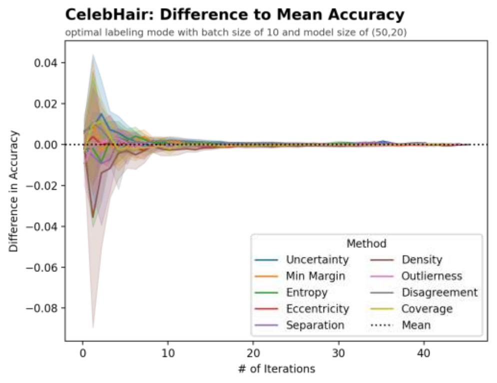
However, there are significant differences between the property measures when looking



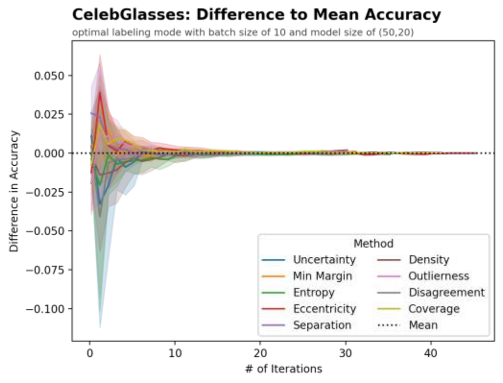
(a) MNIST



(b) STL10



(c) CelebHair

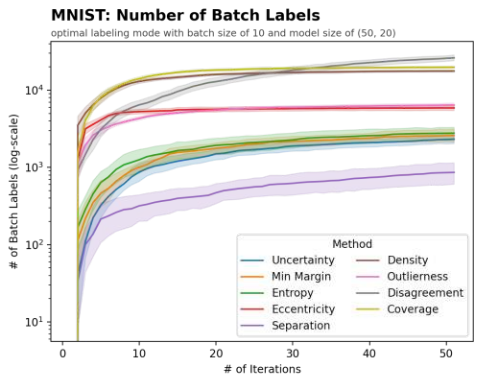


(d) CelebGlasses

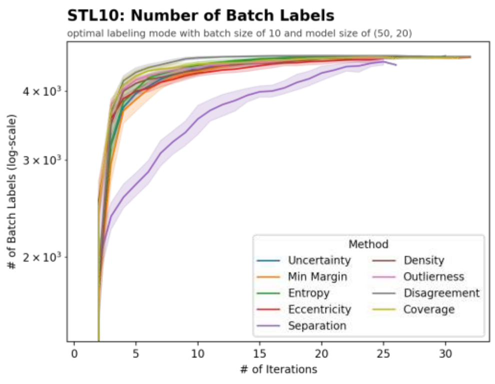
Figure 5.7: Accuracy difference of each property measure to the overall mean when evaluated on batch labeling.

at the number of batch labels in Figure 5.8. The labeling simulation occasionally stops before 50 iterations are reached because batch labeling exhausts the labeling pool when the property measures produce sufficiently accurate results. This is a bit harder to see in the accuracy curves because the accuracy converges at the end.

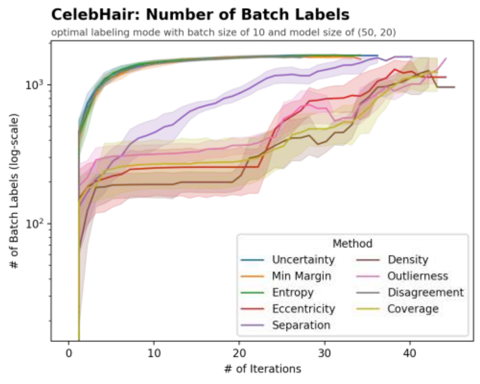
The data-based strategies that performed well for MNIST in terms of accuracy also produced many batch labels. While Disagreement did not generate many batch labels initially, it consistently increased the number of batch labels, which might have led to the delayed accuracy gain. Uncertainty-based methods and Separation have a very low number of batch labels and also perform poorly in terms of accuracy. When evaluated on STL10, all methods produced a similar amount of batch labels, except for Separation, which only reaches similar levels at the end. For MNIST, the number of batch labels of the Separation property measure is consistently below the super-sampling factor of 10, which means the batch pool is much smaller, resulting in less training data and worse model performance. To some extent, this is also true for the uncertainty-based methods, which barely reach the super-sampling limit in the beginning and then fall below the



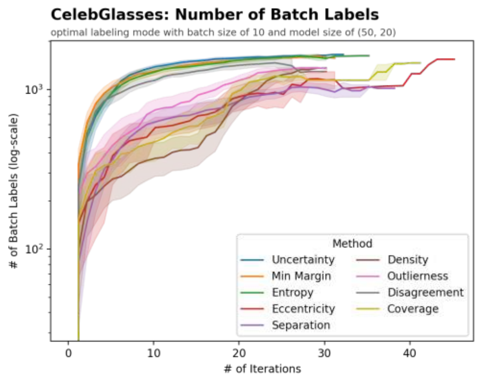
(a) MNIST



(b) STL10



(c) CelebHair



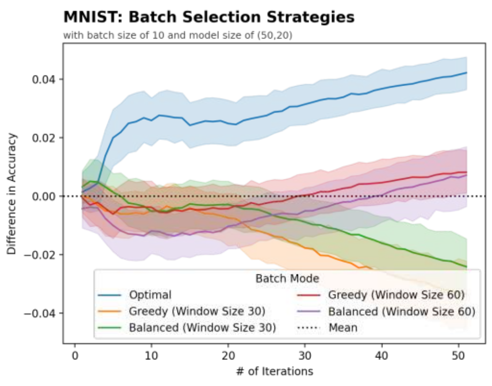
(d) CelebGlasses

Figure 5.8: Comparison of the number of batch labels produced by different property measures during batch labeling.

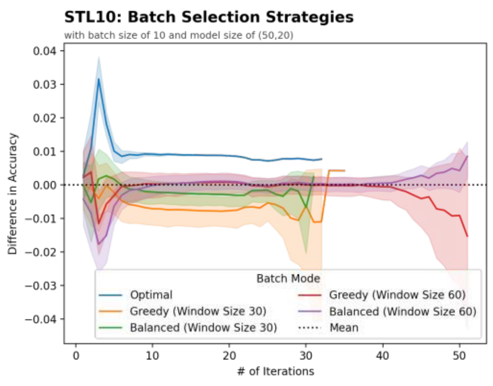
threshold faster because they never reach the maximum amount of batch labels. The other data-based methods reach more than 5000 samples very quickly, which means they can support the super-sampling until the end of the training process.

For CelebHair and CelebGlasses, the uncertainty-based methods, including Disagreement, produced significantly more batch labels than all data-based methods. However, despite an almost tenfold increase in batch labels, the accuracy of the uncertainty-based methods remained essentially unchanged. The slight difference in the overall accuracy between property measures with high and low batch label counts indicates that training on trivial instances does not fundamentally help in this case. A possible reason for the smaller number of batch labels found by data-based methods for CelebHair and CelebGlasses could be the presence of multiple attributes in the data. So the trade-off might be fewer samples to batch label but potentially labeling samples that the model was not already confident about.

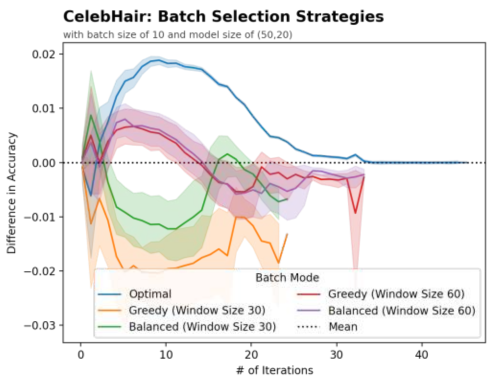
In conclusion, we can assert that data- and model-based methods perform better in different scenarios. When the data is very homogenous with no additional attributes like MNIST, the data distribution provides a lot of information, which the data-based methods can utilize to provide the model with large amounts of novel data in batch labeling. Data-based methods fail to provide large amounts of data for very noisy data with diverse attributes like CelebHair and CelebGlasses. Here, batch labeling fails to improve accuracy, but the additional batch labels also do not negatively impact accuracy.



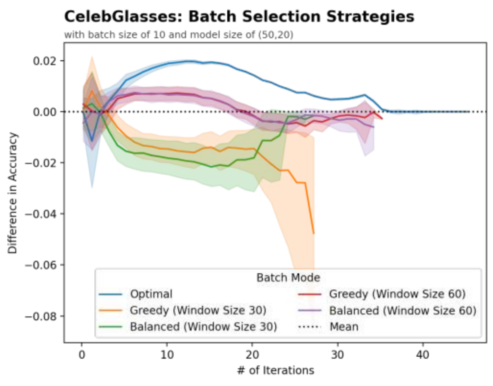
(a) MNIST



(b) STL10



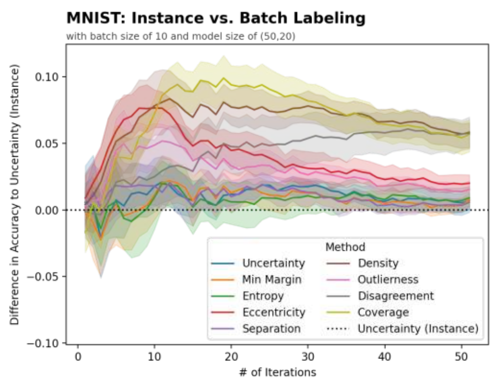
(c) CelebHair



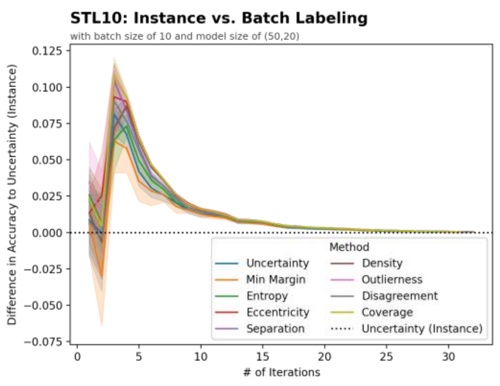
(d) CelebGlasses

Figure 5.9: Comparison of batch modes.

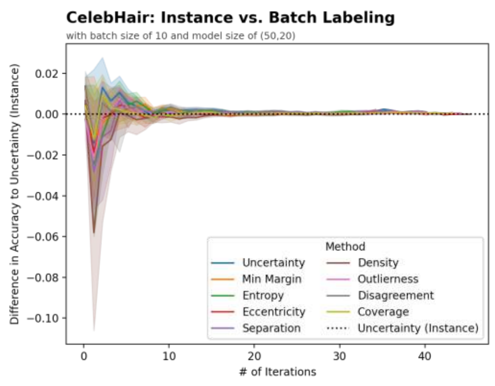
Until now, only the optimal batch labeling mode has been examined, which simulates the best possible performance of the labeler. In addition to the optimal strategy, the remaining labeling modes described in Section 4.1.2 were also evaluated. In Figure 5.9, the overall performance of the different labeling modes is compared. Unsurprisingly, the optimal labeling mode outperforms the rest substantially on all datasets. A more surprising result is that the greedy and balanced labeling modes with a window size of 60 are very similar, and the greedy mode even outperforms the balanced labeling mode for MNIST and STL10. Therefore, a window size of 60 seems to work well enough so that the values between the balanced and greedy cut-off points provide enough beneficial samples for the model to increase accuracy. For a window size of 30, the balanced selection consistently outperformed the greedy mode, which indicates that a window size of 30 is too small to find a suitable cutoff point. We can also see that the aggressive batch labeling stops after less than 30 iterations.



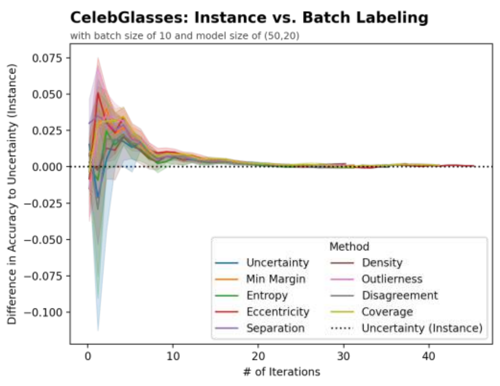
(a) MNIST



(b) STL10



(c) CelebHair



(d) CelebGlasses

Figure 5.10: Comparison of batch labeling to the instance labeling results of the Uncertainty property measure.

Figure 5.10 compares the batch labeling results to the instance labeling simulation using the Uncertainty property measure. The Uncertainty measure was chosen as a baseline because this was the designated instance selection method for batch labeling simulations. The difference is relatively small for CelebHair and CelebGlasses. CelebGlasses has some improvement initially, while batch labeling appears to perform worse for CelebHair. However, this difference quickly diminishes in both datasets.

For MNIST and STL10, all methods benefitted from batch labeling, likely because batch labeling provided more samples for the model to train on. The data-based methods increased the performance significantly, especially for MNIST. This contrasts with the instance labeling results, where uncertainty-based methods outperformed data-based methods (Figure 5.5d). In some cases, labeling instances where the model is already confident will not improve the model since the results for uncertainty-based methods barely outperformed the instance labeling results. However, for STL10, batch labeling significantly improved the results for all methods. Since STL10 closely resembles the training data of the embedding model, we expect the embeddings to be more semantically aligned. This is encouraging since pre-training on any dataset and improving the embeddings seems to translate to better batch labeling performance.

5.2 Runtime

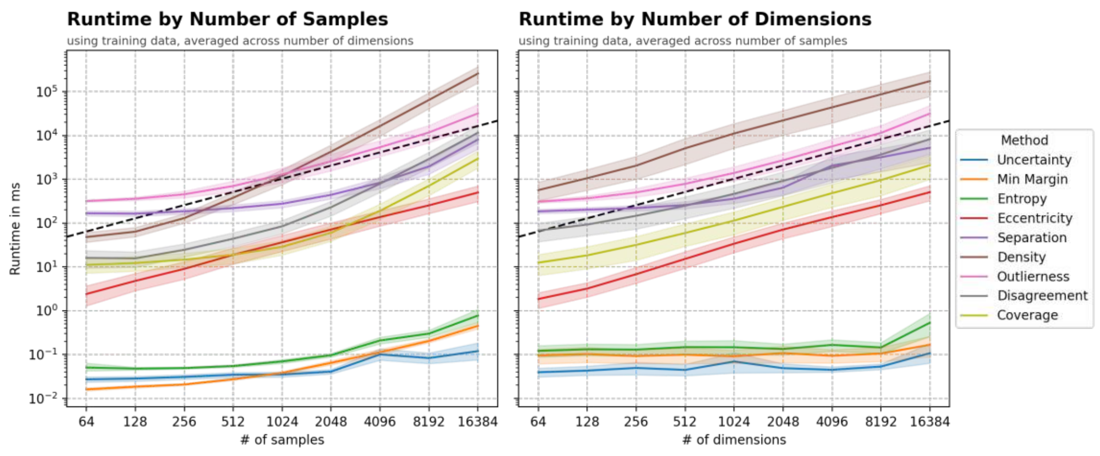


Figure 5.11: Log-log plot of runtime by number of samples and dimensions.

We evaluated the runtime for all property measures as described in Section 4.2. The results are visualized in log-log plots in Figure 5.11. In log-log plots, straight lines indicate polynomial growth, and steeper lines indicate a higher degree of polynomial growth. Curved lines on a graph indicate functions that increase faster than polynomials. The dashed black line in the plot shows linear growth where the exact function is given by $f(x) = x$. In other words, it represents the case where each additional instance adds 1ms to the overall runtime.

In addition to the line plot, Figure 5.12 shows the same results in heatmaps. While the line plot is better suited for comparing methods and their rate of change, the heatmaps provide information about how skewed the runtime is towards an influence of the number of samples or dimensions.

Eccentricity has linear growth in both samples and dimensions because the slope is perfectly parallel to the dashed reference line, which represents a linear function. The corresponding heatmap clearly shows that the runtime is not skewed. The diagonals where the size of the data matrix $n \times p$ is equal also have an equal runtime, regardless of whether the data has more samples or more dimensions.

For Outlierness, we see the same behavior for the data size where the diagonals in the heatmap have the same color and, therefore, runtime. Only here does the gradient

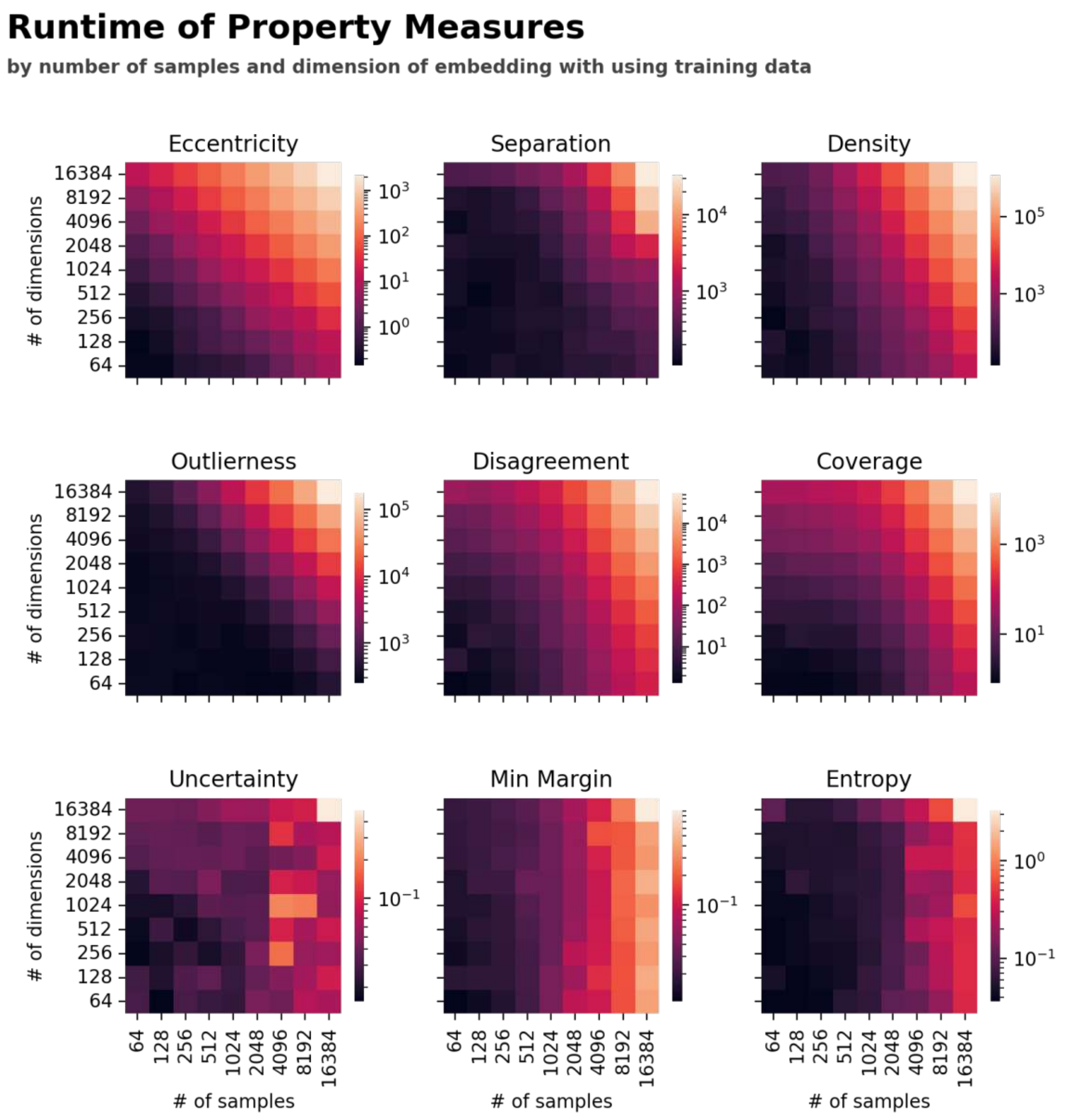


Figure 5.12: Heatmaps of the runtime of each property measure.

increase faster than for Eccentricity, meaning the number of samples or dimensions grows faster. Based on the line plot, it is evident that the Outlier property measure shows faster-than-polynomial growth as the number of samples and dimensions increases.

We can further see that Separation, Density, Coverage, and Disagreement are skewed towards an influence in the number of samples. For Separation, the heatmap indicates that for a large number of samples (the last column), the runtime is much larger than the runtime for the highest number of dimensions (top row). Separation also has exponential growth for both samples and dimensions, like Outlierness, but is more skewed towards an influence on the number of samples.

All uncertainty-based methods share a different property. Their runtime is practically unaffected by the number of dimensions whose curves are flat in the corresponding log-log plot, showing the influence as more dimensions are added. For Separation and Coverage, this is true for the opposite case, where for small sample sizes, the runtime is practically constant before growing rapidly as the sample size increases.

In terms of absolute runtime, uncertainty-based methods drastically outperform all data-based property measures. Suppose we accept a runtime of 1s (or 10^3ms). In that case, only the uncertainty-based methods and Eccentricity fall below the threshold when the number of samples and dimensions can be arbitrarily high (up to the evaluation limit of 16.384). In reality, the number of dimensions will be given by a specific representation learning model, and we want to know which methods fall below the threshold regardless of the number of samples used. In this case, we can look at the runtime for the corresponding number of dimensions. For example, if a 1024-dimensional embedding is used, we can see that Disagreement, Separation, Coverage, Eccentricity, and all uncertainty-based methods fall below the threshold of 1 second and would be a suitable choice. When the acceptable threshold is 100ms, only Eccentricity and uncertainty-based methods fall below this threshold.

5.3 User Study

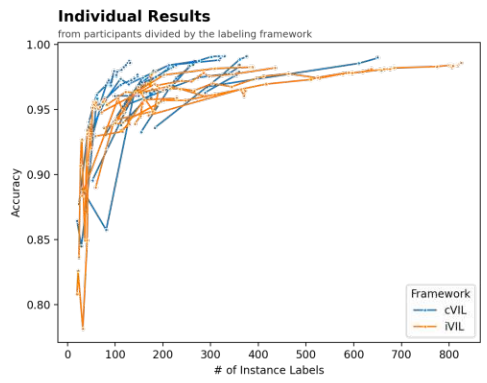
In this section, we examine the results of the user study. We first look at the quantitative results before analyzing the task load and qualitative results of the user responses in the final questionnaire. The datasets used in the user study are CelebHair and CelebGlasses, which performed very similarly in the simulations and indicate that they likely have a similar difficulty level, as was intended. The prototype used the settings described in Section 3.2.2 for the model, and the large classification model with two hidden layers was used.

5.3.1 Quantitative Results

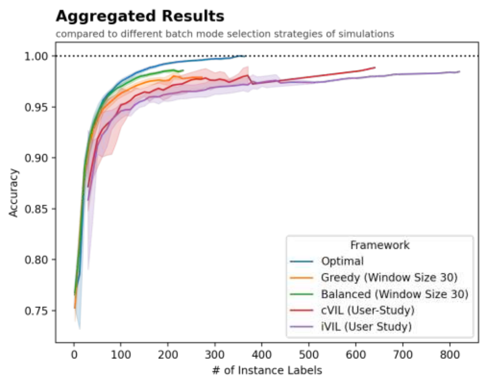
Due to an erroneous batch labeling action, Participant 2 was excluded from the analysis of the accuracy results. They mislabeled 123 instances, which caused a sudden drop in accuracy that went unnoticed during the study. The accuracy drop of Participant 2 can be seen in Figure 5.17.

Figure 5.13a shows the accuracy curves of individual participants. A point marker indicates retraining of the model where the accuracy changes are logged. When the data is aggregated, the results from the user study can be compared to the simulations, as seen in Figure 5.13b.

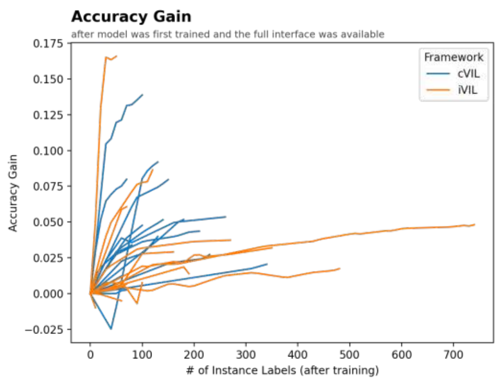
cVIL outperforms iVIL on average, but none can match the simulated results in the early stages, primarily because users decided to label more samples from the random selection



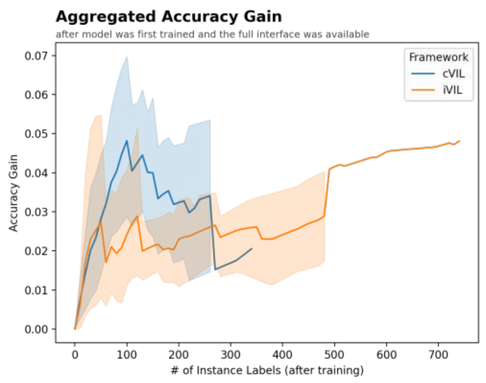
(a) Individual results.



(b) Aggregated results with results from simulations.



(c) Accuracy Gain since first labeling action.

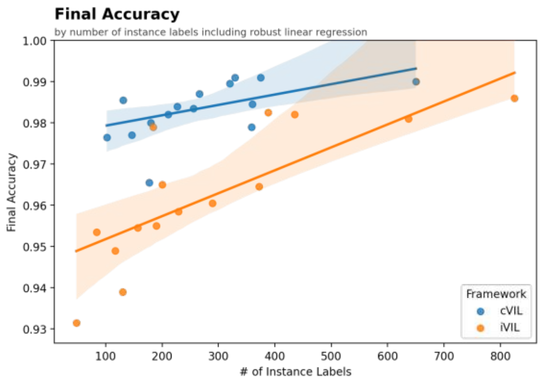


(d) Aggregated accuracy gain since first labeling action.

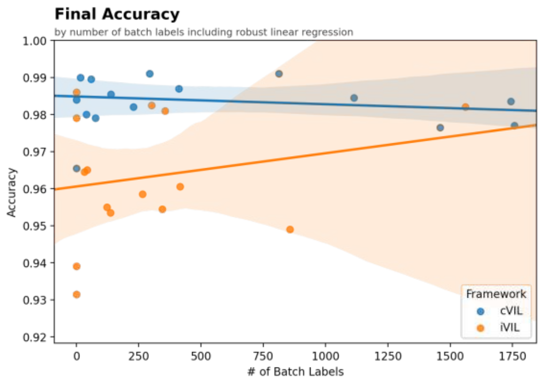
Figure 5.13: Analysis of the accuracy curves of participants during the labeling process.

initially, resulting in a delayed response. Over time, cVIL and later iVIL reached the accuracy of the greedy method from the simulations, and some users managed to achieve the accuracy of the balanced approach. The worse results are likely also due to the larger batch size when users manually label data.

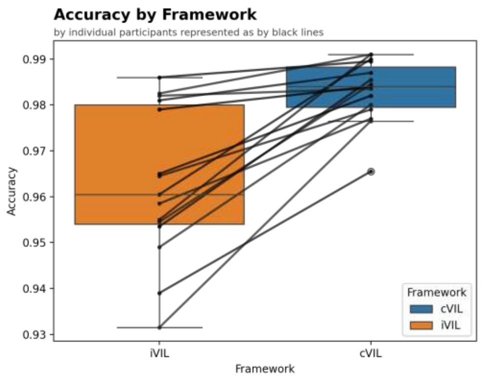
Since different participants decided to label a different number of samples before training the model, each user had a different amount of manual labels when they first got access to the full labeling interface with the visualizations. Therefore, each participant began with a different initial accuracy before using the full interface. To better observe the impact of the visualizations, we can examine the results after the model has been trained and the visualizations became available to the participants. This is shown in Figure 5.13c. One participant performed exceptionally well in iVIL; however, some participants could get close when using the cVIL interface. The aggregate results are visualized in Figure 5.13d, showing that cVIL outperforms iVIL initially. After 250 labels, only a single participant had more labels in cVIL. This participant had a lower accuracy than



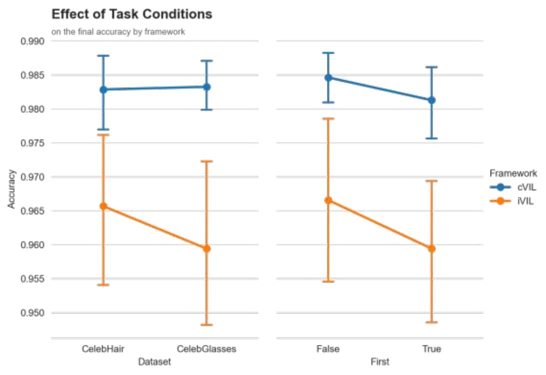
(a) Comparing frameworks based on their final label accuracy compared to the number of instance labels.



(b) Comparing frameworks based on their final label accuracy compared to the number of batch labels.



(c) Results of individual participants by framework.



(d) Analysis of task conditions on the final accuracy.

Figure 5.14: Final accuracy results of participants and analysis of confounding factors.

the iVIL results with similar amounts of labels. However, since most participants labeled less than 150 samples and the best results were achieved with around 100 labels, we can conclude that cVIL outperformed iVIL on average.

The advantage of cVIL becomes more apparent in Figure 5.14a, which shows the relationship between the final number of manual labels and the final label accuracy. cVIL outperforms iVIL considerably with fewer manual labels. The solid lines represent a robust regression estimation. cVIL achieves around the same performance with 100 labels as iVIL with around 600. iVIL would catch up to cVIL at around 900 labels, at which point both frameworks would have a projected accuracy close to 100%.

Figure 5.14b shows participants achieved these results solely through instance labeling. Interestingly, accuracy is not substantially affected by the number of generated batch labels in both conditions (except for one outlier in iVIL). This matches the result of the simulations, which showed that batch labeling with the Uncertainty property measure

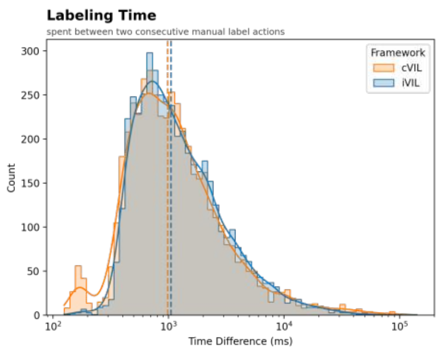


Figure 5.15: The distribution of the time difference between consecutive manual labeling actions.

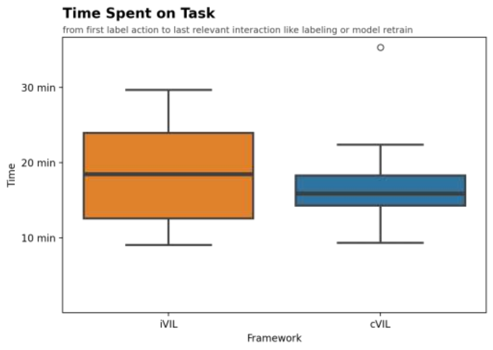


Figure 5.16: Total task completion time by framework.

does not substantially improve the model's accuracy. Due to the unavailability of other property measures for the participants, it is reasonable to conclude that this is the reason for these results.

In Figure 5.14c, it can be observed that every participant's performance improved when using cVIL. This demonstrates the effectiveness of cVIL in enhancing the performance of individual participants. When using cVIL, the accuracy gain compared to iVIL was much more substantial for results with a lower accuracy. Excellent results in iVIL with around 99% accuracy only increased slightly when the participants used cVIL. These results were obtained even though the mean accuracy of the manual labels in cVIL was 0.8 percentage points lower compared to iVIL.

Figure 5.14d shows the effects of the randomization of the task conditions on the final accuracy. For cVIL, the dataset did not affect the results at all. In contrast, iVIL had worse results for CelebGlasses. The order in which the frameworks were presented to the participants also affected the accuracy. As expected, both cVIL and iVIL performed worse when used in the first task, where the user had less prior experience with the system. The introductory task seemed insufficient to fully familiarize participants with the system. In general, the effects are relatively minor; specifically, the datasets led to a very similar task complexity.

Figure 5.15 shows the time difference between consecutive manual labeling actions. The median time indicated by the dashed lines is practically equal for both frameworks and is around 1 second. The distribution of both frameworks is very similar; only iVIL has a small cluster with a very short labeling time.

Figure 5.16 shows the total time spent on labeling tasks for the cVIL and iVIL system. Participants had a median labeling time of around 18,5 minutes for iVIL and 16 minutes for cVIL. While both systems had a minimum labeling time of around 9 minutes, cVIL generally had a lower completion time, except for one outlier. However, the difference is not statistically significant (t(14) = -1.947, p = 0.07).

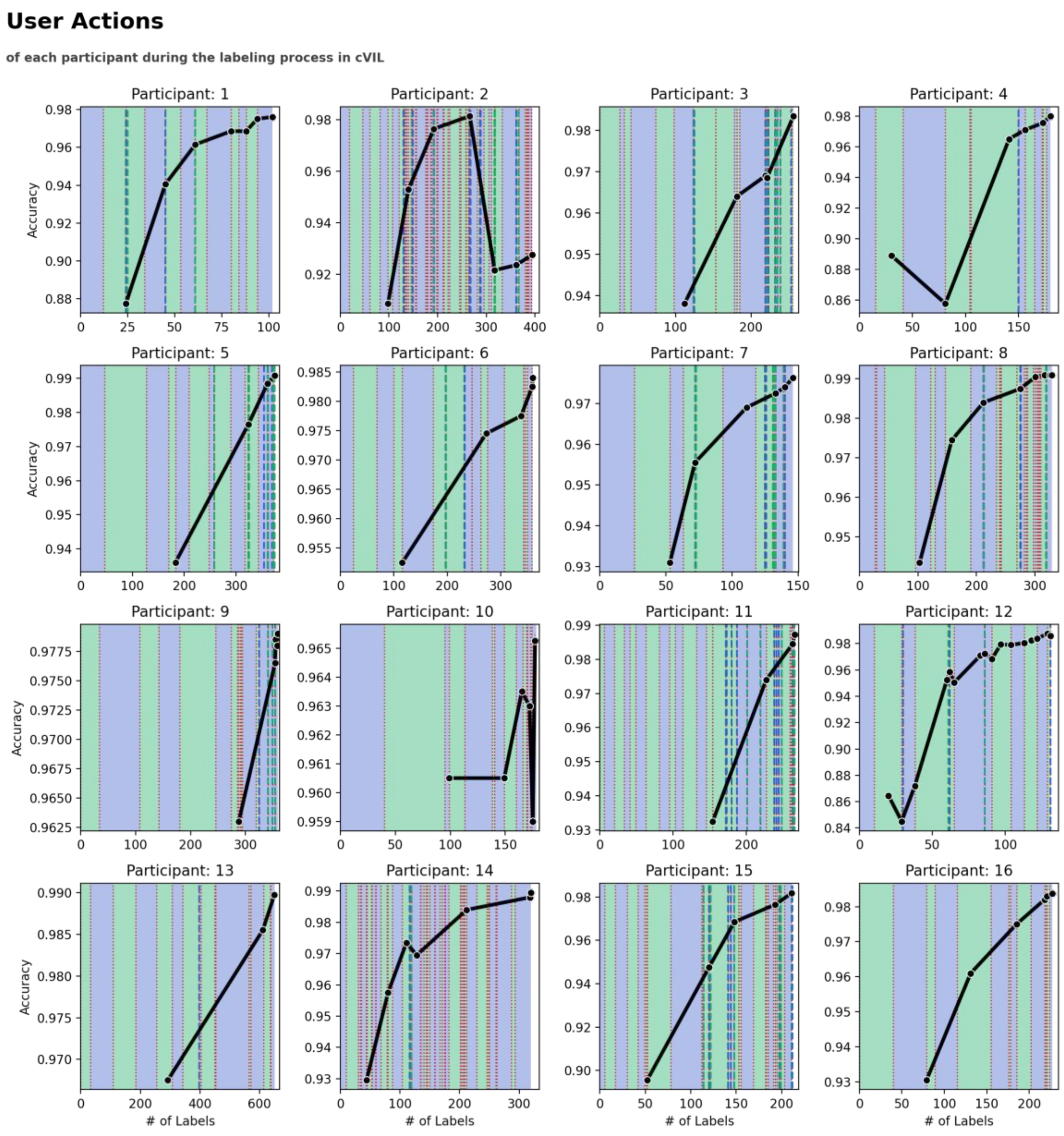


Figure 5.17: Labeling actions of each participant while working on the labeling task. The background color indicates the class of the manual labels (blue for the first and green for the second class, analogous to the prototype interface). A dotted red line marks a class switch to label the other class from the selected one. The thick dashed lines signify batch labeling by the participant. Again, the color indicates which class the samples in the batch selection were assigned to. The solid black line shows the accuracy, where markers on the line indicate model training.

Figure 5.17 shows the labeling actions of users while performing their assigned labeling task. Some participants, like Participant 1 and Participant 12, had very regular manual labeling patterns where a class switch was followed by many manual labeling actions with that class. Other participants, especially Participants 2 and 14, had very irregular labeling patterns with many class switches. Here, we can also see the accuracy drop of Participant 2 after incorrectly labeling the batch selection with class 0 instead of 1. This was a sizable selection with 123 samples where all instances were mislabeled. If, instead, the label had been of the opposite class, all samples would be correct, which leads to the conclusion that this action was performed by mistake. The results were, therefore, excluded from the previous analysis. Note that visually, Participant 10 had a similar accuracy drop. However, the size of the drop was only 0.3 percentage points, compared to more than 6 percentage points for Participant 2 and the drop was not caused by an incorrect labeling action.

Participants 10 and 16 never batch-labeled anything during the labeling process. Participants generally batch-labeled more toward the end of the labeling process when they presumably felt the model was accurate enough. This is especially prevalent in Participants 3, 5, and 11. Some participants, like Participants 5 and 13, only trained the model 3 or 4 times, while Participant 12 trained the model 16 times. Most participants first trained the model in the first half of the labeling process. However, some relied heavily on the random selection. Participant 9 stands out as they made almost 300 manual labels before training the model and labeled less than 400 labels in total.

In addition to the analysis of cVIL, the labeling actions of participants in iVIL are visualized in Figure 5.18. Of particular note are the sudden accuracy drops for Participants 1, 11, 12, 13, and 15. Participant 1 did not batch label at all, and participants 11 and 12 also exhibited accuracy drops before any batch labeling was carried out. Although participants 13 and 15 had accuracy drops after batch labeling like Participant 2 for cVIL, they did not have an incorrect batch labeling action. Furthermore, all users had a very high manual label accuracy, and participants 1 and 12 even had perfect accuracy when the drops appeared. So, these accuracy drops have to be attributed to the user's ineffective instance labeling, which seems more common for iVIL than cVIL. In iVIL, the samples were not specifically selected to benefit the model, which could explain why the accuracy could drop with correct labels. Therefore, users could not match the performance of simple uncertainty-based sampling. Furthermore, batch selections of clusters in the scatterplot projection likely contain only samples with one specific attribute. This could explain why batch labeling a single cluster sometimes leads to accuracy drops in iVIL, as the model then overfits that particular attribute. In cVIL, samples with high property measure values are diverse by design, so the user is less likely to label samples of a single cluster in a single training iteration. Similarly, samples with low property measure values can still be diverse. For example, when using the uncertainty property measure, the model can perform well on samples with different attributes, and the user can batch-label these instances all at once.

Upon examining the labeling actions of users in iVIL further, we observe that Participant

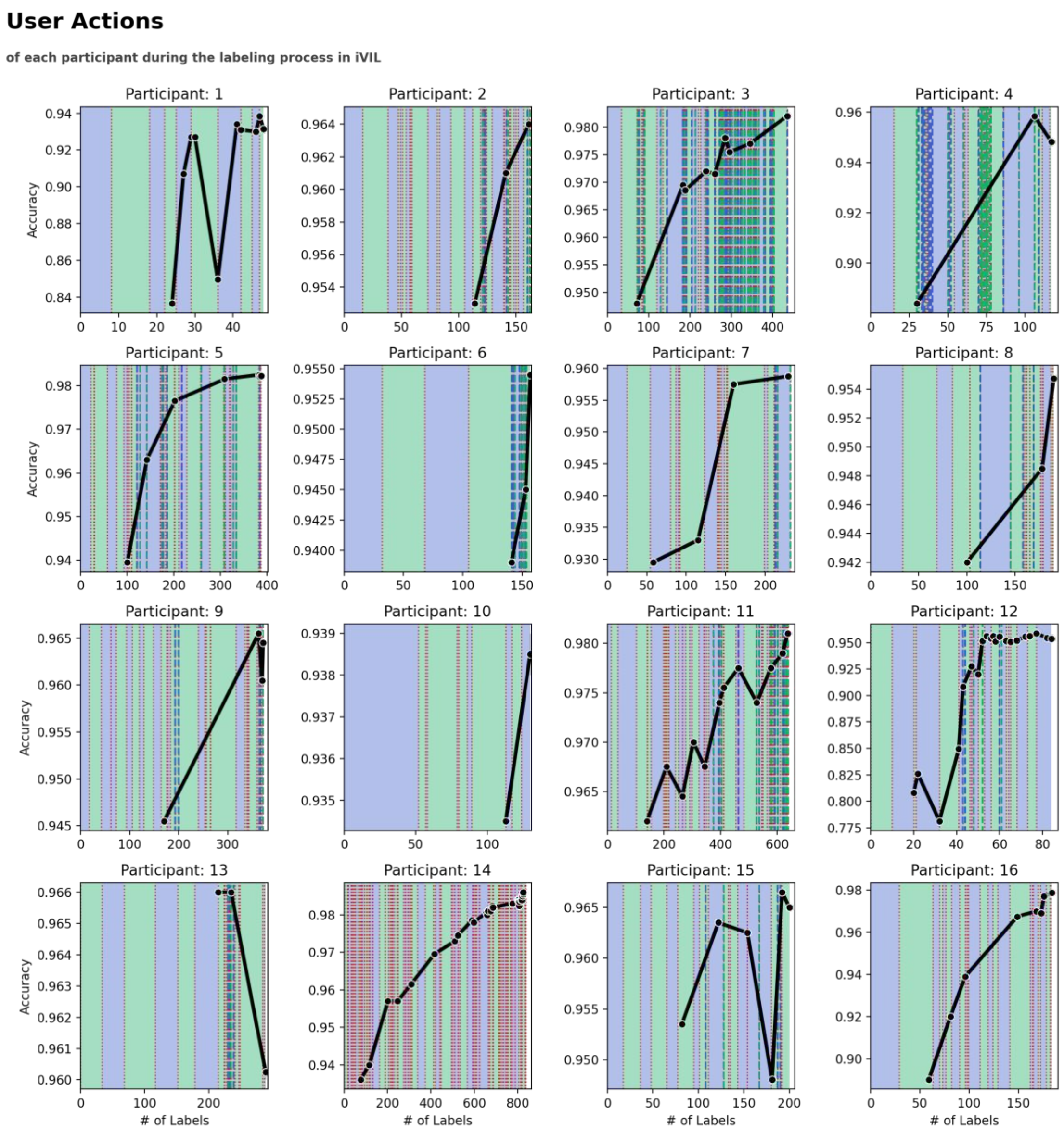


Figure 5.18: Labeling actions of each participant performed while working on the labeling task in the iVIL interface. The background color indicates the class of the manual labels (blue for the first and green for the second class, analogous to the prototype interface). A dotted red line marks a class switch to label the other class from the selected one. The thick dashed lines signify batch labeling by the participant. Again, the color indicates which class the samples in the batch selection were assigned to. The solid black line shows the accuracy, where markers on the line indicate model training.

	Sum of Sq.	df	Mean Sq.	F value	p-value
Interface	**0.00309068**	**1**	**0.00309068**	**36.4728**	**3.7e-06**
Dataset	0.00006423	1	0.00006423	0.7580	0.3929
Order	0.00018966	1	0.00018966	2.2382	0.1482
Train Length	**0.00226040**	**1**	**0.00226040**	**26.6748**	**3.1e-05**
Manual Acc.	0.00000433	1	0.00000433	0.0511	0.8232
Batch Acc.	0.00006074	1	0.00006074	0.7168	0.4059

Table 5.1: ANOVA results: the interface has the largest influence on the accuracy. It is even more influential than the number of training samples. Dataset, order, and the accuracy of manual and batch labels are not statistically significant, and the difference from these factors might have been by chance. Rows with significant p-values are bold.

14 still exhibits highly irregular labeling patterns, even more so than with cVIL. Participants 3 and 4 had significantly more batch labeling actions in iVIL. While Participant 3 labeled roughly the same amount in cVIL (1744 labels) and iVIL (1561 labels), participant 4 only had 40 batch labels in cVIL but batch-labeled 857 instances in iVIL. Again, participants 10 and 16 decided not to use batch labeling, but now, also Participant 1 did not use batch labeling once. Participants 12 and 14 frequently updated the model, labeling a small region before moving on to another cluster and retraining the model, especially towards the end. This might again have led the model to overfit on the most recent cluster.

5.3.2 Statistical Analysis

Finally, we use statistical tests to analyze the final accuracy results. Again, Participant 2 has been excluded for the reasons stated above. A paired t-test is applied to determine whether users had higher accuracy in the cVIL setting. The results are statistically significant ($t(14) = 5.784$; $p < .001$), and we can reject the null hypothesis that the mean difference is 0 and that users performed equally well in both labeling interfaces.

To remove the influence of confounding factors on the final accuracy, we report ANOVA results of a linear regression model, which received the framework, dataset, order of the interfaces, train length, and the accuracy of manual and batch labels of each participant as input variables. The train length is represented as the combined weights used in the analysis, so batch labels are weighted with 0.1 as described in Section 3.2.2.

Table 5.1 shows the ANOVA results, which are equally clear. The choice of the framework is very significant for the final accuracy. As expected, the amount of training data also directly influences the results.

5.3.3 Task Load

No difference in the aggregate task load score between the frameworks could be found in statistical analysis (Wilcoxon Signed-Rank test: $z = -0.879$; $p = 0.39$). This becomes clear in the last boxplot of Figure 5.19, which shows the aggregated task load score. The results for cVIL and iVIL are practically identical. Each score individually is quite similar, particularly the frustration score, which had the most weight in the overall score. It could be argued that cVIL has an advantage over iVIL based solely on its performance score. This could make sense as many participants stated they could better judge when to stop labeling, as discussed in the following section, which presents the results of the user responses to the final questionnaire.

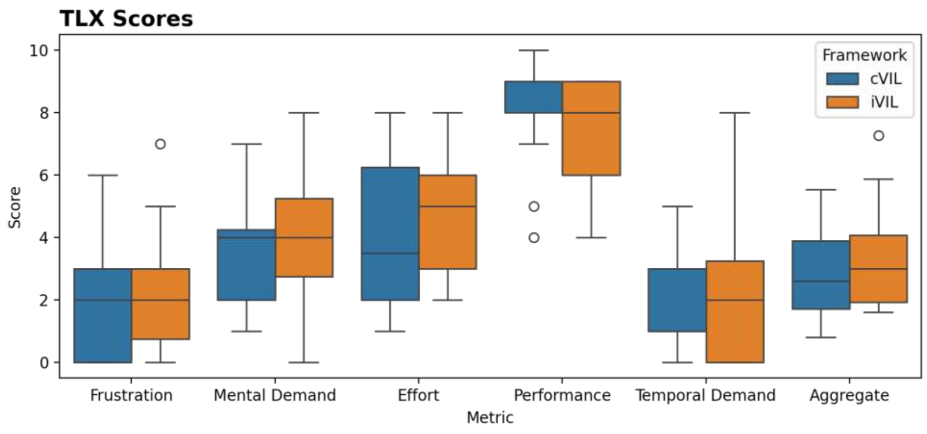


Figure 5.19: Responses to the NASA TLX evaluation by framework.

5.3.4 Qualitative Results

Participants favored cVIL, with 13 out of 16 participants choosing it over iVIL when asked for their preference after finishing both tasks. The main reason was that most participants found it easier to use. Ease of use had many different meanings to different users. Still, it was mainly interpreted as the ability to spot false positives or outliers easily and the fact that participants knew these samples would likely have large property measure values, allowing them to focus on this region of the visualization. This also made it easier to evaluate the system's current performance because the samples with the largest uncertainty values directly indicate the worst-case performance of the model. Participants were given an indication of when to stop labeling based on the model's training state and accuracy. Also, by having information about uncertainty, participants felt it was easier to correct the model when it made incorrect predictions.

The visualization made it easier to select images, and the class partitioning allowed participants to focus on a specific class and get a better overview during the labeling process. As a result, participants also described the interface as more intuitive because it was easier to think in false positives than individual clusters in iVIL.

Participants felt more efficient as they did not have to look through "obvious" images, which also tied in with the ease-of-use aspects of the interface. One participant said, "the system told [them] about all the instances that it thought might be useful in one place." Additionally, they remarked that batch labeling was more straightforward, and ambiguous instances were easier to identify.

When asked what participants disliked, they said there was too little change in the visual representation after the model updates. However, one participant said they liked the consistency of the system. Some also found the visual representation less intuitive overall. It was less clear for some participants what image features caused misclassifications, likely because outliers cannot be spatially differentiated. Their proximity in the visualization is only determined by the property measure value, and the underlying reason for this can be different and has to be found by looking at the images. One participant remarked that the presence of more classes could quickly become tedious. Also, participants thought it was difficult to gain an understanding of the data being labeled.

This is where the data projection of iVIL was well perceived. iVIL was preferred for the clear spatial layout of individual instances, which helped them to get an overview of the whole dataset. Participants who preferred iVIL said the interface was easier to understand and navigate. Participants felt more in control of the labeling process.

Due to the spatial layout, users can also see how images relate to each other and better understand class separations. Another major aspect was that the projection and the resulting clusters were more interesting and engaging to explore. Users also reported that selections were more intuitive. A possible reason is that individual instances are shown in the visualization. This also made it easier to see the number of images remaining, which helped participants to perceive the progress of the labeling task better.

Participants disliked iVIL because selections were more tedious, and ambiguous images were more challenging to find in the visualization. One participant remarked that they had to be more deliberate while labeling, as selections can also include false negatives and true negatives. This requires more focus on each image. Finally, while the clusters did help to build an understanding of the data, they did ultimately not help solve the task because clusters were sometimes noisy and did not represent the target attributes.

CHAPTER 6

Conclusion

In this thesis, we presented a novel labeling approach based on property measures and class partitioning. We justified the underlying concepts and provided detailed reasoning and explanations for each of the components of the system.

The proposed system has been thoroughly evaluated with simulations in various settings to determine the behavior of different property measures. This also included detailed evaluations of the runtime where the effects of sample size and dimensionality of the input embeddings were analyzed. In the user study, we evaluated the system with actual labeling tasks solved by real users and gathered qualitative information about the labeling process. Implementing a baseline instance-based VIL system also allowed for comparing the quantitative results of the cVIL system to a prototypical iVIL implementation.

The simulated evaluation results indicate that different property measures perform better in different scenarios. Uncertainty-based property measures are better suited to improve the model when instance labeling, while batch labeling works better with data-based methods, which can further provide useful information to the model. We demonstrated that instance labeling in cVIL improves accuracy compared to active learning for the MNIST and STL10 datasets. We also showed that batch labeling can further improve the results, specifically when using data-based property measures, answering both research questions 1 and 2.

By measuring the runtime in different settings, we answered research question 3 and showed that not all property measures are suited for real-time interaction with the system. With a threshold of 1 second, only Disagreement, Separation, Coverage, Eccentricity, and all uncertainty-based property measures are usable with the size of our embeddings. Our implementation of the Density property measure took more than 10 seconds to compute, making it hardly suitable for interactive use when property measures are updated frequently.

We observe that scatterplots can be problematic for interactive labeling because they do not scale well. This was also the case for the relatively small dataset we used in our user study, most likely due to the relative data complexity. We further conclude that scatterplots provide a less reliable way to judge model performance when there is a lot of label noise in the data.

We found that cVIL can address these problems when classes strongly overlap in the scatterplot projections and provide effective feedback with large and complex datasets while remaining fully interactive. In theory, the KDE plots used in the prototype to visualize property measure values could show an unbounded number of instances. However, the scalability in terms of the number of classes is still limited, similar to the instance-based implementation with scatterplot projections. In the user study, all participants could improve the final accuracy of the label output with cVIL compared to iVIL and took less time to complete the assigned tasks. This answers research question 4(a), and we conclude that cVIL can be more effective than instance-based approaches.

While no difference in task load could be determined based on the responses from the user study, our interface was preferred by a large majority of participants. The user study still had a relatively small and easy task, and we hypothesize that cVIL may perform even better with more complex data compared to iVIL methods. Furthermore, on average, participants achieved improved results in a shorter time than iVIL. Answering research question 4(b), we can confirm that cVIL increases user satisfaction, even though participants rated their task load equally between the two implementations.

The limitation of our evaluation was that we only used datasets with binary classes in the user study. However, we expect it will scale reasonably well to more classes or at least as well as instance-based approaches that show all classes in a single plot. Furthermore, results from one participant had to be excluded due to an erroneous batch labeling action in the cVIL system. This could have been prevented with a simple warning if the user tries to label a selection with a different class than what the KDE plot shows or if the majority of samples in a selection have the incorrect prediction in the case of iVIL.

Future work should focus on enabling the modification of the class structure during the labeling process to achieve genuinely interactive machine learning. We have also not evaluated the effectiveness of switching between several property measures in the user study. However, based on the simulation results, we can expect that specific property measures are more effective than others. What remains to be evaluated is whether users can identify the most effective property measures in an interactive label setting. We also did not systematically analyze the effect of different data set sizes, features, and class properties on the performance of cVIL and iVIL systems in the user study.

Nevertheless, we showed that cVIL can outperform active learning in our simulations. In real-world settings, we could show that our class-based approach is viable for larger amounts of complex data. For these settings, we could improve the results compared to an instance-based implementation.

APPENDIX A

Appendix

A.1 Proof of Equality of Active Learning utility functions for the binary case

Uncertainty-based active learning selection strategies always try to select instances of which the classification model is the least certain. Certainty is always defined by the posterior class probability the model assigns to an instance. As we show in Theorem 1, in a binary classification problem, when the selection algorithm can only consider probabilities of either one or the other class, the uncertainty-based active learning utility functions presented in 2.1, are equivalent.

Theorem 1. *The active learning utility functions u_{UNC}, u_{MA} and u_{ENT}, defined in Section 2.1, are equivalent for binary classification problems.*

Proof. To prove that the approaches are equivalent, we must show that they produce the same ordering of samples given the same input, i.e., the posterior probabilities. Without loss of generality assume that $u_{UNC}(\theta, x) > u_{UNC}(\theta, x')$. This implies

$$1 - P_\theta(y_0|x) > 1 - P_\theta(y'_0|x') \tag{A.1}$$

$$P_\theta(y_1|x) > P_\theta(y'_1|x') \tag{A.2}$$

Since the parameters are fixed, we will simplify the notation so that y_1 is equivalent to $P_\theta(y_1|x)$ and so forth. Therefore we have

$$y_1 > y'_1 \tag{A.3}$$

as our simplified assumption. Furthermore, since $y_1 = 1 - y_0$ holds, it is possible to rewrite the strategies exclusively in terms of y_1. We get

$$u_{MA}(y_1) = y_1 - (1 - y_1) = 2y_1 - 1 \tag{A.4}$$

and

$$u_{ENT}(y_1) = -(1 - y_1)\log(1 - y_1) - y_1 \log(y_1). \tag{A.5}$$

We further know from the definition that $0 \leq y_1' \leq y_1 \leq 0.5$ to satisfy $y_0 \geq y_1$ and $y_0 + y_1 = 1$ and the analogous constraints for y_1'. To summarize, we assume that $y_1 > y_1'$ and aim to show that $u_{MA}(y_1) > u_{MA}(y_1')$ and $u_{ENT}(y_1) > u_{ENT}(y_1')$, which means we have to show that these functions are monotonously increasing in the range $[0, 0.5]$. The standard way to show that a function is monotonously increasing is to show that its first derivative is positive in that range.

For u_{MA} we have

$$u_{MA}' = \frac{d}{dy_1} 2y_1 - 1 = 2 \tag{A.6}$$

and since $0 > 2$ we can conclude that u_{UNC} is equivalent to u_{MA} for sample selection in the binary case.

For u_{ENT} the derivative is given by:

$$\begin{aligned}
u_{ENT}' &= \frac{d}{dy_1}(-(1 - y_1)\log(1 - y_1) - y_1 \log(y_1)) && \text{(A.7)}\\
&= -\frac{d}{dy_1}((1 - y_1)\log(1 - y_1)) - \frac{d}{dy_1}(y_1 \log(y_1)) && \text{(A.8)}\\
&= -(\log(1 - y_1)\frac{d}{dy_1}(1 - y_1) + (1 - y_1)\frac{d}{dy_1}\log(1 - y_1))\\
&\quad - (log(y_1)\frac{d}{dy_1}(y_1) + y_1 \frac{d}{dy_1}(log(y_1)) && \text{(A.9)}\\
&= -(\log(1 - y_1)(-1) + (1 - y_1)(-\frac{1}{1 - y_1})) - (\log(y_1) + y_1 \frac{1}{y_1}) && \text{(A.10)}\\
&= -(-\log(1 - y_1) - 1) - (\log(y_1) + 1) && \text{(A.11)}\\
&= \log(1 - y_1) + 1 - \log(y_1) - 1 && \text{(A.12)}\\
&= \log(1 - y_1) - \log(y_1) && \text{(A.13)}
\end{aligned}$$

Given that $0 < y_1 \leq 0.5$ we have $0 < y_1 \leq 1 - y_1 \leq 1$ which implies $\log(y_1) \leq \log(1 - y_1) \leq 0$. This directly shows that $\log(1 - y_1) - \log(y_1) \geq 0$.

For the case of $y_1 = 0$ we apply the convention that $0 log(0) = 0$ so that $u_{ENT}(0) = 0$ which means that any other value $0 < y_1 \leq 0.5$ results in a larger utility.

$\square$

A.2 Aggregate NASA TLX Score

The aggregated NASA TLX Score presented by Rubio et al. [RDMP04] is calculated by weighting each individual score according to their relevance for a specific evaluation. This is done through a pairwise comparison between all scores of the NASA TLX questionnaire. The number of times a score was chosen over another defines the importance of the particular score. For the cVIL labeling task, we chose the weights as follows:

- Frustration (F): 5
- Mental Demand (M): 4
- Effort (E): 3
- Performance (P): 2
- Temporal Demand (T): 1
- Physical Demand: 0

The combined score S_{TLX} is then calculated by the weighted average of the individual scores:

$$S_{\text{TLX}} = \frac{5 \times F + 4 \times M + 3 \times E + 2 \times (10 - P) + T}{15} \tag{A.14}$$

Note that the performance score is inverted as a high self-perceived performance score is a positive result, while large values for all other scores are seen as negative.

A.3 Instance and Batch Selection Algorithms

Algorithm A.1 describes the instance selection procedure. For each class, the k_c instances with the largest property measure value get added to the selection pool. That means for each class c, k_c instances are labeled. In all our experiments, the batch size k was a multiple of the number of classes.

The batch selection algorithm A.2 selects the batch labels for a single class c. In the label simulation algorithm (Algorithm 4.1), all samples returned by this batch selection algorithm get the label c. It operates on a subset of the unlabeled data D_c, i.e., the samples the model assigns to class c. Then, the property measure values u are calculated

Algorithm A.1: Instance selection algorithm

```
Input: unlabeled data $D$, trained model $f_\theta$, labeled data $L$, batch labels $B$,
       instance selection size $k$, list of classes $\mathcal{C}$
Output: selected instances $S$
1  $S \leftarrow \emptyset$
2  $k_c \leftarrow \lfloor \frac{k}{|\mathcal{C}|} \rfloor$
3  for $c$ in $\mathcal{C}$ do
4      $D_c \leftarrow \{x \in D \setminus (S \cup L \cup B) | f_\theta(x) = c\}$
5      $x_s \leftarrow \arg\max_{S' \subset D_c : |S'| = k_c} \sum_{s \in S'} PM(s, f_\theta, B_c)$
6      $S \leftarrow S \cup x_s$
7  end
```

Algorithm A.2: Batch selection algorithm

```
Input: unlabeled data $D$, trained model $f_\theta$, labeled data $L$, batch labels $B$,
       selected class $c$
Output: selected instances $S$
1  $D_c \leftarrow \{x \in D \setminus (L \cup B) | f_\theta(x) = c\}$
2  $u \leftarrow PM(D_c, f_\theta, B_c)$
3  $i_{optimal} \leftarrow \arg\min_i \{u_{(i)} | oracle(x_{(i)}) \neq c\}$
4  $i_{greedy} \leftarrow \arg\max_i \{u_{(i)} | \forall x \in \mathcal{N}_x(u_{(i)}) : oracle(x) = c\}$
5  switch $m$ do
6      case $m$ = 'optimal' do
7          $S \leftarrow \{x_{(i)} \in D_c | i \leq i_{optimal}\}$
8      end
9      case $m$ = 'greedy' do
10         $S \leftarrow \{x_{(i)} \in D_c | i \leq i_{greedy}\}$
11     end
12     case $m$ = 'balanced' do
13         $S \leftarrow \{x_{(i)} \in D_c | i \leq \lfloor \frac{i_{optimal} + i_{greedy}}{2} \rfloor\}$
14     end
15 end
```

based on the data D_c, the model f_θ, and batch training data B_c for class c. The values $u_{(i)}$ are the ordered values of the property measures i.e. $u_{(0)} \leq \cdots \leq u_{(n)}$. Therefore, $u_{(i)}$ is the i-th largest value. Similarly, $x_{(i)}$ is the sample with the i-th largest property measure i.e. $PM(x_{(0)}, f_\theta, B_c) \leq \cdots \leq PM(x_{(n)}, f_\theta, B_c)$.

The optimal index is the smallest index of the sorted property measures where the model made an incorrect prediction. This is checked by the oracle, which has access to the ground truth labels. In the actual prototype implementation, this is done by validating the hover previews. The greedy index is found by returning the largest index where the samples x in the neighborhood $\mathcal{N}_x$ of the property measure u_i are correctly classified. The samples with the smallest absolute difference between the property measure values of a sample define its neighborhood. Similarly to the instance selection in Algorithm A.1, the neighborhood function $\mathcal{N}_x$ can be defined as $\mathcal{N}_x(u) = \arg\min_{S' \subset D_c : |S'| = n} \sum_{s \in S'} |PM(s, f_\theta, B_c) - u|$ where the size of the neighborhood is given by n. In practice, this is implemented as a simple convolution.

List of Figures

List of Tables

List of Algorithms

Bibliography

[AHH$^+$14] Bilal Alsallakh, Allan Hanbury, Helwig Hauser, Silvia Miksch, and Andreas Rauber. Visual methods for analyzing probabilistic classification data. *IEEE Transactions on Visualization and Computer Graphics*, 20:1703–1712, 2014.

[BCHS21] Haoping Bai, Mengyao Cao, Ping-Chia Huang, and Jiulong Shan. Self-supervised semi-supervised learning for data labeling and quality evaluation. *ArXiv*, abs/2111.10932, 2021.

[BGTF20] Barbara Caroline Benato, Jancarlo Ferreira Gomes, Alexandru Cristian Telea, and Alexandre Xavier Falcão. Semi-automatic data annotation guided by feature space projection. *Pattern Recognit.*, 109:107612, 2020.

[BHR$^+$19] J. Bernard, Marco Hutter, Christian Ritter, Markus Lehmann, Michael Sedlmair, and Matthias Zeppelzauer. Visual analysis of degree-of-interest functions to support selection strategies for instance labeling. In *EuroVA@EuroVis*, 2019.

[BHS$^+$21] J. Bernard, Marco Hutter, Michael Sedlmair, Matthias Zeppelzauer, and Tamara Munzner. A taxonomy of property measures to unify active learning and human-centered approaches to data labeling. *ACM Trans. Interact. Intell. Syst.*, 11:20:1–20:42, 2021.

[BHZ$^+$18] J. Bernard, Marco Hutter, Matthias Zeppelzauer, Dieter W. Fellner, and Michael Sedlmair. Comparing visual-interactive labeling with active learning: An experimental study. *IEEE Transactions on Visualization and Computer Graphics*, 24:298–308, 2018.

[BT20] David Beil and Andreas Theissler. Cluster-clean-label: an interactive machine learning approach for labeling high-dimensional data. *Proceedings of the 13th International Symposium on Visual Information Communication and Interaction*, 2020.

[BvdWTR21] Javad Zolfaghari Bengar, Joost van de Weijer, Bartlomiej Twardowski, and Bogdan Raducanu. Reducing label effort: Self-supervised meets active learning. *2021 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*, pages 1631–1639, 2021.

[BZL$^+$18] J. Bernard, Matthias Zeppelzauer, Markus Lehmann, Martin Müller, and Michael Sedlmair. Towards user-centered active learning algorithms. *Computer Graphics Forum*, 37, 2018.

[BZSA18] J. Bernard, Matthias Zeppelzauer, Michael Sedlmair, and Wolfgang Aigner. Vial: a unified process for visual interactive labeling. *The Visual Computer*, 34:1189–1207, 2018.

[CBB$^+$19] Mohammad Chegini, J. 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. *Vis. Informatics*, 3:9–17, 2019.

[CMS13] Ricardo J. G. B. Campello, Davoud Moulavi, and Jörg Sander. Density-based clustering based on hierarchical density estimates. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, 2013.

[CTM$^+$21a] Mathilde Caron, Hugo Touvron, Ishan Misra, Herv'e J'egou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9630–9640, 2021.

[CTM$^+$21b] Mathilde Caron, Hugo Touvron, Ishan Misra, Herv'e J'egou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9630–9640, 2021.

[DAB$^+$21] Michael Desmond, Zahra Ashktorab, Michelle Brachman, Kristina Brimijoin, Evelyn Duesterwald, Casey Dugan, Catherine Finegan-Dollak, Michael J. Muller, Narendra Nath Joshi, Qian Pan, and Aabhas Sharma. Increasing the speed and accuracy of data labeling through an ai assisted interface. *26th International Conference on Intelligent User Interfaces*, 2021.

[Das] Dash. `https://dash.plotly.com/`. accessed 04/30/2024.

[DBK$^+$20] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *ArXiv*, abs/2010.11929, 2020.

[DDB$^+$20] Michael Desmond, Evelyn Duesterwald, Kristina Brimijoin, Michelle Brachman, and Qian Pan. Semi-automated data labeling. In *Neural Information Processing Systems*, 2020.

[DDS+09] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, K. Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

[FZL12] Yifan Fu, Xingquan Zhu, and Bin Li. A survey on instance selection for active learning. *Knowledge and Information Systems*, 35:249–283, 2012.

[GCT22] Benedikt Grimmeisen, Mohammad Chegini, and Andreas Theissler. Visgil: machine learning-based visual guidance for interactive labeling. *The Visual Computer*, 39:5097 – 5119, 2022.

[HS88] S. G. Hart and Lowell E. Staveland. Development of nasa-tlx (task load index): Results of empirical and theoretical research. *Advances in psychology*, 52:139–183, 1988.

[HZRS15] Kaiming He, X. Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2015.

[Inc15] Plotly Technologies Inc. Collaborative data science. `https://plot.ly`, 2015. accessed 04/30/2024.

[Jol03] Ian T. Jolliffe. Principal component analysis. *Technometrics*, 45:276 – 276, 2003.

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

[Kru64] Joseph B. Kruskal. Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika*, 29:1–27, 1964.

[LLWT15] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

[LSB22] Harlin Lee, Aaqib Saeed, and A. Bertozzi. Active learning of non-semantic speech tasks with pretrained models. *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5, 2022.

[LZH+22] Zheng Li, Yue Zhao, Xiyang Hu, Nicola Botta, Cezar Ionescu, and George H. Chen. Ecod: Unsupervised outlier detection using empirical cumulative distribution functions. *IEEE Transactions on Knowledge and Data Engineering*, 35:12181–12193, 2022.

[PAV18] Oscar Gabriel Reyes Pupo, Abdulrahman H. Altalhi, and Sebastián Ventura. Statistical comparisons of active learning strategies over multiple datasets. *Knowl. Based Syst.*, 145:274–288, 2018.

[PCI$^+$08] James Philbin, Ondřej Chum, Michael Isard, Josef Sivic, and Andrew Zisserman. Lost in quantization: Improving particular object retrieval in large scale image databases. *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2008.

[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 Kopf, Edward Yang, Zachary 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. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[PLM$^+$22] Gustavo Perez, Sean T. Linden, Timothy McQuaid, Matteo Messa, Daniela Calzetti, and Subhransu Maji. An ai-assisted labeling tool for cataloging high-resolution images of galaxies. 2022.

[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.

[RAL$^+$17] Donghao Ren, Saleema Amershi, Bongshin Lee, Jina Suh, and J. Williams. Squares: Supporting interactive performance analysis for multiclass classifiers. *IEEE Transactions on Visualization and Computer Graphics*, 23:61–70, 2017.

[RDMP04] Susana Rubio, Eva Maria Carro Diaz, Jesus Martin, and José M. Puente. Evaluation of subjective mental workload: A comparison of swat, nasa-tlx, and workload profile methods. *Applied Psychology*, 53:61–86, 2004.

[Sam69] John W. Sammon. A nonlinear mapping for data structure analysis. *IEEE Transactions on Computers*, C-18:401–409, 1969.

[Set09] Burr Settles. Active learning literature survey. 2009.

[SG10] Christin Seifert and Michael Granitzer. User-based active learning. *2010 IEEE International Conference on Data Mining Workshops*, pages 418–425, 2010.

[SK22] Simon-Martin Schröder and Rainer Kiko. Assessing representation learning and clustering algorithms for computer-assisted image annotation—simulating and benchmarking morphocluster. *Sensors (Basel, Switzerland)*, 22, 2022.

[SKK20] Simon-Martin Schröder, Rainer Kiko, and Reinhard Koch. Morphocluster: Efficient annotation of plankton images by clustering. *Sensors (Basel, Switzerland)*, 20, 2020.

[SLT17] Yunjia Sun, Edward Lank, and Michael A. Terry. Label-and-learn: Visualizing the likelihood of machine learning classifier's success during data labeling. *Proceedings of the 22nd International Conference on Intelligent User Interfaces*, 2017.

[Son20] Mofei Song. Personalized image classification by semantic embedding and active learning †. *Entropy*, 22, 2020.

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

[VGO$^+$20] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.

[Vox] Voxforge.org. Free speech... recognition (linux, windows and mac) - voxforge.org. `http://www.voxforge.org/`. accessed 04/30/2024.

[YL16] Yazhou Yang and M. Loog. A benchmark and comparison of active learning for logistic regression. *Pattern Recognit.*, 83:401–415, 2016.

[YsSPgC22] John Seon Keun Yi, Min seok Seo, Jongchan Park, and Dong geol Choi. Using self-supervised pretext tasks for active learning. In *European Conference on Computer Vision*, 2022.

[YZS$^+$15] Fisher Yu, Yinda Zhang, Shuran Song, Ari Seff, and Jianxiong Xiao. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *ArXiv*, abs/1506.03365, 2015.

[ZNL19] Yue Zhao, Zain Nasrullah, and Zheng Li. Pyod: A python toolbox for scalable outlier detection. *Journal of Machine Learning Research*, 20(96):1–7, 2019.

[ZTdJ$^+$21] Yu Zhang, Martijn Tennekes, Tim de Jong, Lyana Curier, Bob Coecke, and Min Chen. Simulation-based optimization of user interfaces for quality-assuring machine learning model predictions. *ACM Transactions on Interactive Intelligent Systems*, 2021.