Accurate and Robust Weakly Supervised Learning with Candidate Labels

Zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften

von der KIT-Fakultät für Informatik des
Karlsruher Instituts für Technologie (KIT)

genehmigte
Dissertation

von

Tobias Fuchs

Tag der mündlichen Prüfung: 19. Januar 2026
1. Referentin: Prof. Dr. Nadja Klein
2. Referent: Prof. Dr. Eyke Hüllermeier
Betreuer: Prof. Dr. Nadja Klein, Prof. Dr.-Ing. Klemens Böhm

Acknowledgements

First and foremost, I would like to express my gratitude to everyone who supported me throughout my doctoral journey. I am grateful to my supervisors Prof. Klemens Böhm and Prof. Nadja Klein for their invaluable guidance. They consistently pointed me in the right direction while giving me the freedom to pursue my own research interests. I also would like to thank Prof. Eyke Hüllermeier for reviewing this manuscript.

I am thankful to all my colleagues for making this such a great environment to work in. In particular, I want to thank Florian Kalinke for our collaborative work, Jakob Bach and Markus Iser for introducing me into research, and Jose Cribeiro for sharing an office with me and your many funny stories.

Finally, I want to thank my family for their unwavering support. To my partner, Jana: thank you for sharing your life with me and making every day better.

Abstract

Machine learning algorithms underpin many modern technologies but typically rely on supervised learning, which assumes access to fully annotated, noise-free training data. In practice, this assumption rarely holds: real-world datasets are often noisy, incomplete, or ambiguous. In crowdsourcing, for example, experts can assign conflicting labels to the same instances. While such datasets can be manually cleaned, sanitizing data is costly.

Partial-label learning (PLL) offers a principled framework to address such ambiguous data. Thereby, each training instance is associated with a set of candidate labels, of which only one is correct but unknown. The PLL framework aims at training classifiers in this setting that perform well on unseen samples. However, the absence of exact ground truth makes it particularly difficult to construct reliable and accurate classifiers. Ensuring that such models make reliable predictions thus requires robust algorithms that can deal with uncertainty, abstain from unreliable decisions, or adapt to imperfect data.

This thesis addresses these challenges by developing accurate and robust partial-label learning methods. First, we introduce a method that allows models to abstain from uncertain predictions through a reject option, enabling safer deployment in human-in-the-loop scenarios such as medical diagnosis or crowdsourcing. Second, we present a probabilistic approach that models predictive uncertainty via a Dirichlet distribution over class probabilities, enhancing robustness against label noise, out-of-distribution samples, and adversarial perturbations. Third, we propose a refinement technique based on conformal prediction that iteratively cleans candidate label sets, providing confidence guarantees while improving overall accuracy. Finally, we formulate label disambiguation as a latent-variable inference problem and employ amortized variational inference to approximate the posterior distribution over true labels, achieving state-of-the-art accuracy in both synthetic and real-world experiments. Together, these contributions establish a comprehensive framework for accurate and robust weakly supervised learning with PLL candidate labels.

Contents

1. Introduction

Machine learning algorithms lie at the core of countless applications powering today's technology. In many cases, these algorithms rely on supervised learning, which assumes fully annotated, noise-free pairs of features and corresponding ground-truth labels to predict the class labels of unseen samples. However, real-world data rarely meets these ideal conditions—it is often noisy, incomplete, or ambiguous—violating the assumptions underlying conventional supervised classification and thereby limiting its applicability and performance.

Such noisy and conflicting data arises across a wide range of domains, including crowdsourcing (Lin et al., 2022), web mining (Guillaumin et al., 2010; Zeng et al., 2013), audio classification (Briggs et al., 2012), and automated theorem proving (Zombori and Indruck, 2025), among many others. In crowdsourcing, for instance, different human annotators may assign conflicting class labels to the same training instance, reflecting differing levels of expertise, subjective judgment, or simple human error. Although it is often possible to clean such datasets manually or to aggregate multiple annotations through majority voting, this process is expensive, time-consuming, and not always reliable. The challenge is particularly pronounced in medical domains, where labels must be provided by trained specialists, such as radiologists or pathologists, making large-scale annotation prohibitively costly (Wang et al., 2020).

In web mining, the problem manifests differently: data collected from the internet is inherently noisy due to the uncurated and heterogeneous nature of online content (Gheisari et al., 2023). For instance, Guillaumin et al. (2010) train image classifiers using labels derived from the alternative text of online images. These textual descriptions often vary widely in quality and relevance—some accurately reflect the image content, while others are misleading or unrelated—yet they enable the creation of large-scale datasets without manual labeling. Similar challenges appear in audio classification, where sound recordings may contain overlapping acoustic events, such as multiple speakers talking simultaneously or environmental noise obscuring the target signal (Briggs et al., 2012). In these cases, it is not obvious how to derive a single definitive label for each training instance.

A more recent and conceptually distinct example arises in automated theorem proving (Zombori and Indruck, 2025), where machine learning models guide the search for proofs by predicting which logical atom to branch on next. Here, the class labels correspond to proof decisions that may not be uniquely correct, since multiple proof paths can lead to valid results. This structure creates an intrinsic form of label ambiguity that mirrors noise in other domains.

Across all these scenarios, the data available for training is noisy, ambiguous, or conflicting. Such imperfections fundamentally violate the assumptions underlying standard supervised classification, which assumes unambiguous, fully labeled data. To deal with these real-world challenges, weakly supervised learning has emerged as a broad paradigm that relaxes the strict requirements of full supervision and enables model training under weaker forms of label information.

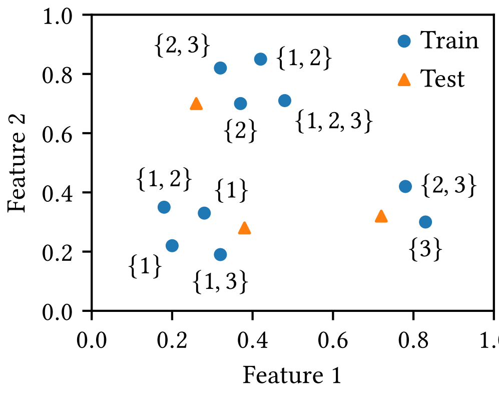
Weakly supervised learning encompasses a wide range of problem settings, each reflecting a different way in which supervision can be incomplete or imprecise. Examples include semi-supervised learning (Hady and Schwenker, 2013), where only a subset of the data is labeled; noisy label learning (Bylander, 1994), where annotations may be incorrect; complementary label learning (Ishida et al., 2019), where the learner is told only what class a training instance does not belong to; and partial-label learning (Feng et al., 2020)—the main focus of this thesis—where each training instance is associated with a set of possible candidate labels.

Among these settings, partial-label learning (PLL; Jin and Ghahramani 2002; Lv et al. 2020; Xu et al. 2021; Tian et al. 2024) offers a principled approach to handling ambiguous or conflicting annotations without the need for costly data cleaning. In PLL, each training instance is associated with a set of possible labels, one of which is correct but unknown to the learning algorithm. This framework captures the previously discussed scenarios—disagreement among annotators in crowdsourcing, inconsistent or noisy information in web mining, overlapping sources in audio classification, and multiple valid proof paths in automated theorem proving—where uncertainty about the correct class label is intrinsic to the data itself. PLL methods aim to leverage such weak supervision to train multi-class classifiers capable of disambiguating candidate sets and predicting the correct class of unseen samples.

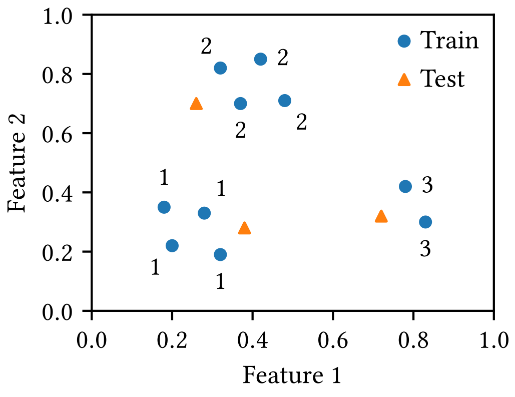
Figure 1.1 illustrates the distinction between standard supervised classification and partial-label learning: while the former assumes access to precise one-to-one mappings between training instances and labels, the latter operates under one-to-many supervision, where the correct label must be inferred from a set of possibilities.

The absence of exact class labels makes it particularly challenging to construct reliable classifiers, as uncertainty arises at multiple levels of the learning process. First, there is label ambiguity within each candidate set: the true label is hidden among several plausible alternatives, and the likelihood of candidate labels can depend on both the instance's features and the relationships between classes. In some regions of the feature space, annotators, or the processes generating the labels, are more prone to mislabeling, leading to systematically biased or overlapping candidate sets. Moreover, certain classes may be inherently more similar or harder to distinguish, further increasing the ambiguity.

Second, there is uncertainty concerning how the candidate labels of one example relate to those of others. Since the reliability of a candidate label may vary across the dataset, it becomes difficult to determine which training instances provide useful information for guiding the learning process. Both types of uncertainty—within individual examples and



(a) An example of a partial-label learning dataset.



(b) An example of a supervised classification dataset.

Figure 1.1.: Comparison of partial-label learning (left) and supervised multi-class classification (right), using the possible class labels $\{1, 2, 3\}$. Figure 1.1a shows an example of a partial-label learning dataset, where each training data point (●) is associated with a set of candidate labels, of which only one unknown label is correct. PLL admits training classifiers in this context without the need for manual data cleaning. In contrast, Figure 1.1b shows an example of a standard multi-class classification dataset, where each training data point (●) is associated with a known ground-truth class label. Note that the dataset in Figure 1.1b is a cleaned version of that in Figure 1.1a. In both cases, the goal is to predict the class labels of unseen test instances (▲).

across the dataset—contribute to the difficulty of constructing reliable classifiers under the PLL setting.

Given this inherent uncertainty, robustness naturally becomes a central concern. Machine-learning models must not only resolve label ambiguity but also remain reliable when faced with noisy, conflicting, or otherwise imperfect supervision. This is especially important when model predictions influence human decisions or safety-critical systems. In high-stakes domains such as medical diagnosis (Yang et al., 2009; Lambrou et al., 2011; Reamaroon et al., 2019), financial fraud detection (Cheng et al., 2020; Berkmans and Karthick, 2023; Xiang et al., 2023), or autonomous driving (Xu et al., 2014; Varshney and Alemzadeh, 2017; Hubmann et al., 2017; Shafaei et al., 2018; Michelmore et al., 2020), misclassifications can have serious consequences, ranging from incorrect treatments to financial losses or physical harm.

Although robustness has been extensively studied in conventional supervised learning, it remains relatively underexplored in partial-label learning. Most existing PLL methods focus on disambiguating candidate labels to improve predictive accuracy, yet few explicitly consider how these models behave when faced with corrupted, adversarial, or out-of-distribution data. However, because PLL inherently operates on uncertain and imperfect supervision, as already elaborated above, robustness considerations are crucial. Developing algorithms that are both accurate and robust under weak supervision is therefore essential for ensuring trustworthy and dependable learning systems.

In the following, we detail our contributions toward building such accurate and robust algorithms within the PLL framework.

1.1. Contributions

In this work, we propose four partial-label learning methods that address the aforementioned challenges of accurate and robust prediction-making in the PLL setting. We summarize our contributions in the following.

PLL with a Reject Option (Chapter 4). Even the best algorithms make incorrect predictions, which can have severe consequences in safety-critical domains. To mitigate such risks, we employ what are known as reject options. Reject options allow an algorithm to abstain from certain predictions if unsure and, instead, let humans decide on the label of an instance or the actions to take. In the weakly supervised PLL setting, obtaining sensible reject options is more challenging than in the supervised case as ground truth is not available. Nevertheless, a reject option allows for mitigating incorrect predictions and improving prediction quality. To the best of our knowledge, we are the first to address reject options in the PLL context. Such a mechanism is especially valuable in crowdsourcing scenarios, where rejected instances can be prioritized for manual re-annotation by human workers, thereby improving data quality without excessive labeling effort. Similarly, in medical image classification, rejected cases can be escalated to domain experts for verification, reducing the risk of harmful misclassifications.

To introduce reject options in the PLL setting, we propose a nearest-neighbor-based PLL method grounded in Dempster-Shafer theory. Unlike existing PLL approaches that rely on point estimates of candidate-label weights, our method maintains a feasible region that captures the range of plausible label assignments. This representation provides a principled way to assess predictive uncertainty and decide when to reject a sample.

Robust PLL by Leveraging Class Activation Values (Chapter 5). Predictions of machine learning algorithms should be robust regarding several criteria to limit incorrect predictions and their effects on downstream applications. Three central robustness desiderata are: (a) resilience to high noise levels, (b) stability under out-of-distribution inputs, and (c) resistance to adversarial attacks. These aspects are critical in the PLL setting, where supervision is inherently noisy and ambiguous. While prior work has examined robustness to label noise (a), the impact of distributional shifts (b) and adversarial perturbations (c) on PLL algorithms has remained largely unexplored.

To address this gap, we propose RobustPll, a novel PLL method that leverages class activation values to construct reliable internal representations of the observed data. Specifically, RobustPll learns a model that parameterizes a Dirichlet distribution over class probabilities, providing a principled quantification of predictive uncertainty. This distributional modeling allows the algorithm to distinguish confident from ambiguous predictions, improving robustness against label noise, out-of-distribution samples, and adversarial perturbations.

This capability is especially beneficial in real-world scenarios such as web mining, where the closed-world assumption does rarely hold, or autonomous systems, where adversarial or out-of-context inputs may arise unexpectedly. RobustPll maintains strong predictive performance across all three robustness dimensions, establishing it as one of the first PLL approaches to explicitly address this broad spectrum of robustness challenges.

PLL with Conformal Candidate Cleaning (Chapter 6). A wide range of algorithms have been proposed to address the PLL problem. Recent work has introduced several extension mechanisms designed to enhance their predictive performance. Such extensions are particularly valuable because different PLL classifiers tend to excel on different datasets, making broadly compatible improvements highly desirable. However, most existing extensions rely on heuristic rules to refine candidate label sets, lacking theoretical guarantees about their reliability.

To overcome this limitation, we propose a principled extension that alternates between training a PLL classifier and pruning candidate sets using conformal prediction. The conformal prediction step outputs refined candidate sets that contain the true label with a user-specified confidence level, thereby providing a statistically grounded mechanism for reducing label ambiguity. This iterative conformal candidate cleaning process leads to cleaner supervision signals and, consequently, more effective classifier training.

The approach is broadly applicable across existing PLL methods and demonstrates substantial improvements in predictive accuracy across diverse datasets and experimental settings. In practical terms, this kind of principled candidate refinement is especially useful in crowdsourcing and web-mining scenarios, where label noise is pervasive and heuristic filtering often fails to provide reliable confidence estimates.

Amortized Variational Inference for PLL (Chapter 7). Early PLL methods approximate the posterior distribution over true labels but are often computationally expensive and limited in scalability. More recent deep learning approaches address scalability but typically rely on surrogate loss functions or heuristic label refinement, which can compromise probabilistic soundness and interpretability.

To bridge this gap, we propose a probabilistic framework that directly models the posterior distribution over true labels using amortized variational inference. In our approach, neural networks predict the variational parameters from input data, allowing for efficient inference. This formulation combines the expressiveness of deep architectures with the theoretical rigor of probabilistic modeling, while remaining architecture-agnostic.

Our method demonstrates strong empirical performance, consistently achieving state-of-the-art accuracy compared to existing PLL methods across a diverse set of benchmark experiments. These results highlight the effectiveness of modeling label uncertainty through amortized variational inference. Such gains in predictive performance are particularly valuable in domains like crowdsourcing or medical diagnosis, where precise label inference is essential for guiding reliable downstream reasoning and decision-making.

1.2. Publications

This work is based on the following publications.

- Tobias Fuchs, Florian Kalinke, and Klemens Böhm. Partial-label learning with a reject option. *Transactions on Machine Learning Research*, January 2025. `https://openreview.net/pdf?id=wS1fD0ofay`.
- Tobias Fuchs and Florian Kalinke. Robust partial-label learning by leveraging class activation values. *Machine Learning*, 114(193), 2025. `https://doi.org/10.1007/s10994-025-06796-z`.
- Tobias Fuchs and Florian Kalinke. Partial-label learning with conformal candidate cleaning. In *Uncertainty in Artificial Intelligence*, pages 1337–1357, 2025. `https://proceedings.mlr.press/v286/fuchs25a.html`.
- Tobias Fuchs and Nadja Klein. Amortized variational inference for partial-label learning: A probabilistic approach to label disambiguation. *CoRR*, abs/2510.21300, 2025. `https://doi.org/10.48550/arXiv.2510.21300`.

This dissertation reuses their content, with adaptations made to ensure a coherent and consistent monograph. These prior publications contain the core contributions of this work—yet, their presentation has been restructured where appropriate. Algorithms, definitions, equations, figures, etc. may not be identical to the originals, as adjustments were made for consistency. Chapters 4–7 explicitly indicate which prior works they draw upon.

All code and data associated with this dissertation has been released under permissive licenses. Chapters 4–7 specify where the resources for reproducing the corresponding experiments can be found. Each repository includes detailed instructions for reproducing our results.

1.3. Outline

The remainder of this dissertation is organized as follows. Chapter 2 introduces notation and reviews the partial-label learning problem, Chapter 3 discusses related work, and Chapters 4–7 contain our main contributions, summarized in Section 1.1. Chapter 8 concludes and discusses future work.

Appendix A, B, and C complement Chapter 4, 5, and 6, respectively, and contain additional proofs, results, and detail the experimental setups.

2. Notation and Problem Statement

This section establishes the notations used throughout our work and states the partial-label learning (PLL) problem. Table 2.1 gives an overview of all notations.

Partially-labeled dataset. Given a d-dimensional real-valued feature space $\mathcal{X} = \mathbb{R}^d$ and a set $\mathcal{Y} = [k] := \{1, \dots, k\}$ of $k \in \mathbb{N}$ ($k \geq 3$) classes, a partially-labeled training dataset $\mathcal{D} = \{(x_i, s_i) \in \mathcal{X} \times 2^{\mathcal{Y}} : i \in [n], s_i \neq \emptyset\}$ contains n training instances with associated feature vectors $x_i \in \mathcal{X}$ and non-empty candidate labels $s_i \in 2^{\mathcal{Y}}$, $s_i \neq \emptyset$, for each $i \in [n]$. Their respective ground-truth labels $y_i \in \mathcal{Y}$ are unknown during training, but $y_i \in s_i$.

PLL model. In PLL, the observations (x_i, s_i) and their associated ground-truth labels y_i are random ($i \in [n]$), which we model as follows. Let $\Omega = \mathcal{X} \times \mathcal{Y} \times 2^{\mathcal{Y}}$. Underlying PLL is the measurable space $(\Omega, \mathcal{B}(\Omega))$ with $\mathcal{B}$ denoting the Borel σ-algebra. Further, let $\mathbb{P}, \mathbb{Q} \in \mathcal{M}_1^+(\Omega, \mathcal{B}(\Omega))$ be probability measures with $\mathcal{M}_1^+(\Omega, \mathcal{B}(\Omega))$ denoting the set of all probability measures on $(\Omega, \mathcal{B}(\Omega))$. We denote by $X : \Omega \longrightarrow \mathcal{X}$, $Y : \Omega \longrightarrow \mathcal{Y}$, and $S : \Omega \longrightarrow 2^{\mathcal{Y}}$ the random variables governing the occurrence of an instance's features, ground-truth label, and its candidate labels, respectively. Their realizations are denoted by x_i, y_i, and s_i ($i \in [n]$). We denote by $\mathbb{P}_X$ the marginal distribution of X and by $\mathbb{P}_{XY}$ and $\mathbb{P}_{XS}$ the joint distribution of (X, Y) and (X, S), respectively. $\mathbb{P}_{XY}$ coincides with the probability measure usually underlying the supervised setting. We denote by $\mathbb{P}_n := \mathbb{P}_{XS}^n$ the n-fold product of $\mathbb{P}_{XS}$. Assume $\mathbb{P}, \mathbb{Q} \in \mathcal{M}_1^+(\Omega, \mathcal{B}(\Omega))$ are absolutely continuous with respect to a suitable product measure composed of Lebesgue and counting measures, so that Radon-Nikodym densities p and q exist. The cumulative distribution function of a real-valued random variable $Z : \Omega \longrightarrow \mathbb{R}$ is $F_Z(t) = \mathbb{P}_Z(Z \leq t)$ and its empirical counterpart is $\hat{F}_Z(t) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}_{\{Z_i \leq t\}}$, where $Z_1, \dots, Z_n \overset{i.i.d.}{\sim} \mathbb{P}_Z$.

PLL objective. Having established our notations, we define the PLL problem as follows. Let $\Delta^{\text{k-1}} = \{y \in [0,1]^k \mid \sum_{j=1}^{k} y_j = 1\}$, where y_j denotes the j-th component of y, and $\ell : \Delta^{\text{k-1}} \times \mathcal{Y} \longrightarrow \mathbb{R}_{\geq 0}$ denote a measurable loss function, e.g., the log-loss $(\hat{p}, y) \mapsto -\log \hat{p}_y$ ($\hat{p}_y \neq 0$). Similar to the supervised classification setting, PLL aims to train a probabilistic classifier $f : \mathcal{X} \longrightarrow \Delta^{\text{k-1}}$ that minimizes the risk

$$R^*(f) = \mathbb{E}_{XY}[\ell(f(X), Y)] . \tag{2.1}$$

Table 2.1.: Overview of the notations used throughout this thesis.

Notation	Description	Chapter
d, k, n	Number of feature dimensions, classes, and instances	4, 5, 6, 7
$\mathcal{X}$, $\mathcal{Y}$	Instance and label space	4, 5, 6, 7
$2^{\mathcal{Y}}$	Power set of the label space	4, 5, 6, 7
Ω, $\mathcal{B}(\Omega)$	Sample space and Borel σ-algebra	4, 5, 6, 7
X, Y, S	Random variables in the PLL setting	4, 5, 6, 7
x_i, y_i, s_i	Realizations of X, Y, and S	4, 5, 6, 7
$\mathcal{M}_1^+(\Omega, \mathcal{B}(\Omega))$	Set of probability measures	4
$\mathbb{P}$, $\mathbb{Q}$	Probability measures	4, 5, 6, 7
$\mathbb{P}_{XS}$, $\mathbb{P}_{XY}$, etc.	Joint measures	4, 5, 6, 7
$\mathbb{P}_X$, $\mathbb{P}_Y$, $\mathbb{P}_S$	Marginal measures	4, 5, 6, 7
$\mathbb{P}_n$	n-fold product of $\mathbb{P}_{XS}$	6
$\mathbb{E}$, $\mathbb{E}_{XS}$	Expectation w.r.t. $\mathbb{P}$ and $\mathbb{P}_{XS}$	5, 6, 7
p, q	Densities	6, 7
$F_X(t)$, $\hat{F}_X(t)$	(Empirical) cumulative distribution function of X	6
$\Delta^{\text{k-1}}$	Probability simplex	4, 5, 6, 7
$\mathcal{H}$	Hypothesis space	6
f, g	Classifiers	4, 5, 6, 7
ℓ	Loss function, e.g., log loss	4, 5, 6, 7
$R(f)$, $\hat{R}(f)$	True, empirical risk of f	4, 5, 6, 7
f^*, $\hat{f}$	True, empirical risk minimizer	4, 5, 6, 7
$W_{X,S,y}$, w_{iy}	Label weights	4, 5, 6, 7

However, since Y is unobserved in PLL, one commonly considers

$$R(f) = \mathbb{E}_{XS}\left[\sum_{y=1}^{k} W_{X,S,y}\, \ell(f(X), y)\right], \tag{2.2}$$

where $(W_{X,S,y})_{y=1}^{k} \in \Delta^{\text{k-1}}$ are label weights to control the influence of different loss terms. Common instantiations for $W_{X,S,y}$ include the average strategy $W_{X,S,y}^{(\text{avg})} = \mathbb{1}_{\{y \in S\}}/|S|$ (Cour et al., 2011) and the minimum strategy

$$W_{X,S,y}^{(\min)} = \frac{\mathbb{P}_{Y|X}(Y = y)}{\sum_{y' \in S} \mathbb{P}_{Y|X}(Y = y')} \tag{2.3}$$

(Lv et al., 2020), which weights the loss based on the relevancy of each label. Thereby, one assumes that $\sum_{y' \in S} \mathbb{P}_{Y|X}(Y = y') > 0$ almost surely, so the above fractions are well-defined. For the minimum strategy in (2.3), the true risk takes the form

$$R(f) = \mathbb{E}_{XS}\left[\sum_{y=1}^{k} \frac{\mathbb{P}_{Y|X}(Y = y)}{\sum_{y' \in S} \mathbb{P}_{Y|X}(Y = y')} \ell(f(X), y)\right]. \tag{2.4}$$

Under the uniform candidate-set generation model (Feng et al., 2020), where each nonempty subset $s \subseteq \mathcal{Y}$ containing the true label y is drawn uniformly, the expected PLL risk in (2.4) is an unbiased estimator of the supervised classification risk (2.1).

The empirical version of (2.4) is obtained by using a sample mean:

$$\hat{R}(f) = \frac{1}{n} \sum_{i=1}^{n} \sum_{y=1}^{k} w_{iy} \ell(f(x_i), y), \tag{2.5}$$

where $(x_i, s_i) \in \mathcal{D}$ and $(w_{iy})_{y=1}^{k} \in \Delta^{\text{k-1}}$ approximates the label relevancy $W_{X,S,y}^{(\min)}$ in (2.3) using

$$w_{iy} = \begin{cases} f_y(x_i) / \sum_{y' \in s_i} f_{y'}(x_i) & \text{if } y \in s_i, \\ 0 & \text{else,} \end{cases} \tag{2.6}$$

where $f : \mathcal{X} \to \Delta^{\text{k-1}}$ is a classifier and $f_y(x)$ denotes the y-th entry of $f(x) \in \Delta^{\text{k-1}}$.

$\mathcal{H} = \{f : \mathcal{X} \to \Delta^{\text{k-1}} \mid f \text{ measurable } \}$ denotes the hypothesis space, $f^* = \arg\min_{f \in \mathcal{H}} R(f)$ the true risk minimizer, and $\hat{f} = \arg\min_{f \in \mathcal{H}} \hat{R}(f)$ the empirical risk minimizer. We make the common assumption that the hypothesis space $\mathcal{H}$ is well-specified in the sense that the Bayes posterior $x \mapsto \mathbb{P}_{Y|X=x}$ is contained in $\mathcal{H}$, implying an exact minimizer of (2.1) exists in $\mathcal{H}$ (Tsybakov, 2004; van Erven et al., 2015). The class label of each instance $x \in \mathcal{X}$ with the highest probabilistic prediction, that is, $\hat{y}_x = \arg\max_{y \in \mathcal{Y}} \hat{f}_y(x)$, is called *pseudo-label* (with ties broken at random).

3. Related Work

Partial-label learning (PLL) is one out of many weakly supervised learning frameworks. In PLL, each instance is associated with a set of candidate labels, among which only one is correct. Section 3.1 reviews recent developments in the PLL literature.

Given that even the best machine-learning algorithms can give incorrect predictions—potentially causing serious downstream consequences—Chapter 4 studies the integration of reject options in the PLL setting. Section 3.2 discusses related work on reject options in the supervised case. Also, machine-learning algorithms should be robust to reduce the impact of misclassifications. Section 3.3 presents related work on robust decision-making. Later, in Chapter 5, we propose a novel PLL algorithm with these robustness considerations in mind. Finally, Section 3.4 discusses related work on variational inference: while many recent PLL methods employ deep learning to optimize surrogate loss functions, variational inference offers a principled approach to approximating the true label posterior (Chapter 7).

3.1. Partial-Label Learning

Partial-label learning is a weakly supervised learning problem, where training instances are annotated with multiple candidate labels. The PLL problem has gained significant attention over the last decades. Most approaches adapt common supervised classification algorithms to the PLL setting. Examples include a logistic regression formulation (Grandvalet, 2002), expectation-maximization strategies (Jin and Ghahramani, 2002; Liu and Dietterich, 2012), nearest-neighbor methods (Hüllermeier and Beringer, 2005; Zhang and Yu, 2015), support-vector classifiers (Nguyen and Caruana, 2008; Cour et al., 2011; Yu and Zhang, 2017), custom stacking and boosting ensembles (Zhang et al., 2017; Tang and Zhang, 2017; Wu and Zhang, 2018), and label propagation strategies (Zhang and Yu, 2015; Zhang et al., 2016; Xu et al., 2019; Wang et al., 2019; Feng and An, 2019).

Recent state-of-the-art methods (Lv et al., 2020; Feng et al., 2020; Xu et al., 2021; Zhang et al., 2022a; Wang et al., 2022; Xu et al., 2023; Tian et al., 2024) leverage deep learning methods to tackle the PLL problem. Since ground-truth labels are unavailable, however, risk minimization is performed using surrogate loss functions. For example, Lv et al. (2020); Feng et al. (2020) propose the minimum loss formulation, that is, they minimize (2.5) with the label weights updated as in (2.6), Xu et al. (2021) introduce a self-training strategy based on pseudo-labels, Zhang et al. (2022a) leverage the magnitudes of class activation values, Wang et al. (2022) use ideas from contrastive learning, Xu et al. (2023) utilize level sets to iteratively remove incorrect labels from the candidate sets, Tian et al. (2024) propose a cross-model selection strategy, where multiple models are trained simultaneously, Gong et al. (2024) introduce a smoothing component to the loss term, and Yang et al. (2025) use a pseudo-labeling framework based on the feature representations.

These methods iteratively refine the PLL candidate sets by alternating between training a model using empirical risk minimization and updating the label weights w_{ij} in (2.6) using the trained classifier. Feng et al. (2020) further show that their classifier is risk consistent with the Bayes classifier f^*, if the small-ambiguity-degree condition holds (Cour et al., 2011; Liu and Dietterich, 2012). The condition requires that there is no incorrect label $\bar{y} \neq y$, which co-occurs with the correct label y in a candidate set with a probability of one. Formally, one imposes that $\sup_{x \in \mathcal{X}, y \in \mathcal{Y}, \bar{y} \in \mathcal{Y}, \bar{y} \neq y} \mathbb{P}_{S \mid X=x, Y=y}(\bar{y} \in S) < 1$.

Because of the huge variety of PLL methods, there are recent extensions that can be combined with any of the above to improve prediction performance further. Wang and Zhang (2022) propose a feature augmentation technique based on class prototypes and Bao et al. (2021, 2022); Zhang et al. (2022b) propose feature selection strategies for PLL data. Existing state-of-the-art methods achieve significantly better accuracies when trained on these modified feature sets. Xu et al. (2023) propose the method Pop, which gradually removes unlikely class labels from the candidate sets if the margin between the most likely and the second-most likely class label exceeds some heuristic threshold.

3.2. Reject Options

Recently, much attention has been given to the study of reject options (Mozannar et al., 2023; Mao et al., 2024; Narasimhan et al., 2024). A reject option allows one to abstain from predictions and defer them to humans rather than making possibly harmful decisions.

There are two common strategies in rejecting predictions in the supervised setting: The confidence-based and the classifier-rejector approach (Ni et al., 2019; Cao et al., 2022). The confidence-based strategy uses a threshold on the models' confidence in order to accept or reject predictions. Common model choices for quantifying the confidences are Bayesian methods (Kingma and Welling, 2014; Kendall and Gal, 2017) and ensembles (Lakshminarayanan et al., 2017; Wimmer et al., 2023). In contrast, the classifier-rejector approach jointly learns the classifier and rejector (Ni et al., 2019; Mao et al., 2024), which can have beneficial theoretical properties. However, the classifier-rejector approach is less flexible than the confidence-based strategy as it is coupled with the concrete loss formulation of the classifier. We propose an extension of the confidence-based strategy for partial-label learning in Chapter 4.

Calibration methods (Naeini et al., 2015; Guo et al., 2017; Ao et al., 2023) are also related to the confidence-based rejection strategy as both are used to make statements about the certainty of predictions. While reject options provide a binary decision, calibration methods modify the predicted confidences such that they align with the observed accuracies. In this sense, both approaches are orthogonal and cannot directly be compared. In Chapter 4, we focus on reject options.

3.3. Robust Prediction-Making

Robust prediction-making encompasses a variety of aspects out of which we consider (a) good predictive performance under high PLL noise (Zhang et al., 2021), (b) robustness against out-of-distribution examples (OOD; Sensoy et al. 2018), and (c) robustness against adversarial examples (Madry et al., 2018) to be the most important in PLL. Real-world applications of PLL often entail web mining use cases, where the closed-world assumption usually does not hold (requiring (b)). Also, PLL training data is commonly human-based and therefore a possible surface for adversarial attacks (requiring (c)). Other robustness objectives that we do *not* consider are, for example, the decomposition of the involved uncertainties (Kendall and Gal, 2017; Wimmer et al., 2023) or the calibration of the confidences (Ao et al., 2023; Mortier et al., 2023).

To address (a) in the supervised setting, one commonly employs Bayesian methods (Kingma and Welling, 2014; Kendall and Gal, 2017) or ensembles[1] (Lakshminarayanan et al., 2017; Wimmer et al., 2023). To recognize OOD samples (b), one commonly employs techniques from representation learning (Zhang and Yu, 2015; Xu et al., 2021) or leverages negative examples using regularization or contrastive learning (Sensoy et al., 2018; Wang et al., 2022). To address (c), methods incorporate adversarially corrupted features already in the training process to strengthen predictions (Lakshminarayanan et al., 2017). To the best of our knowledge, we are the first to propose a method that addresses (a), (b), and (c) in PLL (see Chapter 5). Tackling all three aspects is particularly challenging in the PLL domain as there is no exact ground truth on which an algorithm can rely to build robust representations.

While evidential deep-learning (Sensoy et al., 2018) fails to learn a well-calibrated epistemic uncertainty measure with respect to a reference distribution, it excels at forming a relative notion of uncertainty, which is sufficient for most downstream tasks. We further improve its performance in this respect by adding a regularization term (Section 5.3.3) and using an optimal label-weight update strategy (Section 5.3.4). With these additions, we empirically observe strong performances for the downstream tasks (a), (b), and (c) in the partial-label learning setting. We refer to Jürgens et al. (2024) for an extended discussion regarding the limitations of evidential deep-learning in the supervised setting.

[1] Ensemble techniques also benefit (b) and (c) and are easy to implement. Therefore, we also consider an ensemble approach of one of our competitors in our experiments as a strong baseline in Chapter 5.

3.4. Variational Inference

Variational methods offer a principled framework for approximate Bayesian inference by formulating posterior estimation as an optimization problem. Early methods (Jordan et al., 1999; Attias, 1999; Beal, 2003) introduce mean-field approximations and EM algorithms for latent variable models. These approaches typically rely on model-specific derivations and coordinate ascent updates. Kingma and Welling (2014) introduce variational auto-encoders (VAE) and amortized VI, which employ a neural network (the encoder) to predict the variational parameters directly from input data. They also make use of the reparameterization trick to enable backpropagation through stochastic variables, facilitating scalable and efficient inference. Rezende et al. (2014) propose a similar approach in the context of deep latent Gaussian models. Subsequent extensions include VI with normalizing flows (Rezende and Mohamed, 2015) and a model-agnostic optimization formulation (Ranganath et al., 2014). Sohn et al. (2015) introduce conditional variational auto-encoders (CVAE), which extend VAEs by conditioning the variational parameters on additional input data. Our method builds on this formulation and is detailed in Chapter 7. An overview of recent developments in amortized VI in general is that of Margossian and Blei (2024).

4. Partial-Label Learning with a Reject Option

This chapter's contents are based on the following publication.

- Tobias Fuchs, Florian Kalinke, and Klemens Böhm. Partial-label learning with a reject option. *Transactions on Machine Learning Research*, January 2025. `https://openreview.net/pdf?id=wS1fD0ofay`.

All code and data for reproducing this chapter's experiments are available at `https://github.com/mathefuchs/pll-with-a-reject-option`.

4.1. Overview

Even the best machine-learning algorithms can give incorrect predictions. These errors can have severe consequences when they impact actions or decisions. Consider, for example, safety-critical domains such as the classification of medical images (Yang et al., 2009; Lambrou et al., 2011; Kendall and Gal, 2017; Reamaroon et al., 2019) or the control of self-driving cars (Xu et al., 2014; Varshney and Alemzadeh, 2017; Hubmann et al., 2017; Shafaei et al., 2018; Michelmore et al., 2020). One option to limit fallacies is to employ so-called *reject options*, which allow one to abstain from certain predictions if unsure and, instead, let humans decide on the label of an instance or the actions to take (Mozannar et al., 2023). Naturally, there is a trade-off arising between the number and accuracy of non-rejected predictions. In the supervised setting, reject options have already been studied, both, for multi-class classification (Charoenphakdee et al., 2021; Cao et al., 2022; Mao et al., 2024; Narasimhan et al., 2024) and regression tasks (Zaoui et al., 2020; Cheng et al., 2023).

In the weakly supervised PLL setting, obtaining sensible reject options is more challenging than in the supervised case as ground truth is not available. Still, a reject option allows for mitigating misclassifications and improving prediction quality. Determining whether to reject a prediction of a PLL algorithm is difficult as the uncertainty involved in prediction-making originates from multiple sources. On the one hand, there is inherent uncertainty due to ambiguously labeled data. This ambiguity is influenced by both instances and classes: In some regions of the instance space, annotators are more likely to mislabel instances than in others. Additionally, some class labels are inherently more similar than others. On the other hand, there is uncertainty due to the lack of knowledge regarding the relevance of each candidate label. To the best of our knowledge, we are the first to study reject options in the context of PLL.

We propose a novel partial-label learning approach based on Dempster-Shafer theory (DST; Dempster 1967; Shafer 1986). In contrast to existing PLL approaches, which use point estimates of the candidate label weights (Liu and Dietterich, 2012; Zhang and Yu, 2015; Ni et al., 2021; Xu et al., 2023), our method maintains a feasible region, also known as *credal set*. Maintaining such a credal set is beneficial in assessing whether to reject predictions as our experiments show.

In the PLL context, we consider (Hüllermeier and Beringer, 2005; Zhang and Yu, 2015; Zhang et al., 2016; Xu et al., 2019; Wang et al., 2019) as most closely related to our proposed method as these approaches consider an instance's neighborhood to infer its class label. To the best of our knowledge, we are the first to propose an extension of the l-NN classifier leveraging Dempster-Shafer theory to tackle reject options in the PLL context.

Contributions. We summarize our contributions as follows.

- *Algorithm with reject option.* We introduce DstPll, a novel nearest-neighbor-based partial-label learning algorithm with a reject option that learns from ambiguously labeled data. The algorithm effectively determines whether to reject a prediction and provides the best trade-off between the number and accuracy of non-rejected predictions when compared to other state-of-the-art PLL methods.
- *Experiments.* Extensive experiments on artificial and real-world data support our claims. Our code and data are openly available.
- *Theoretical analysis.* We analyze DstPll and show several desirable properties of our reject option. The runtime analysis shows that the proposed method's runtime is dominated by l-nearest-neighbor search, which has an average time complexity of $\mathcal{O}(dl \log n)$, with d features and n training instances.

Outline. We discuss relevant background information in Section 4.2 and propose our method in Section 4.3. Section 4.4 features experiments and Section 4.5 contains all proofs. Auxiliary results and additional experiments are in Appendix A.

4.2. Background

In order to make well-informed decisions about when to reject predictions, we use Dempster-Shafer theory, which we elaborate on in the following. Dempster-Shafer theory (DST; Dempster 1967; Shafer 1986) allows for dealing with uncertainty by assigning probability mass to sets of events without specifying the probabilities of individual labels; incorrect labels do not obtain any probability mass. DST builds upon two core quantities, so-called *belief* and *plausibility*. Informally, belief collects all evidence that supports a hypothesis and plausibility collects all evidence that does not contradict a hypothesis. We argue that DST is a perfect fit for partial-label learning as one may interpret the ambiguous candidate sets as evidence regarding a label hypothesis. We further exploit belief and plausibility to inform our reject option, taking into account the difference between the supporting and non-conflicting evidence. In contrast, existing PLL approaches (Hüllermeier and Beringer, 2005; Cour et al., 2011; Liu and Dietterich, 2012; Zhang and Yu, 2015; Ni et al., 2021; Xu et al., 2023) initially assign some probability mass to each label candidate and subsequently refine them. By doing so, most probability mass is first allocated to labels that are certainly incorrect, as only one candidate is the true label. This known method renders handling the noise coming from incorrect candidate labels challenging.

With these intuitions, we now recall DST formally. In DST, a basic probability assignment (bpa) $\mathrm{m} : 2^{\mathcal{Y}} \rightarrow [0, 1]$ assigns probability mass to subsets of $\mathcal{Y}$. m satisfies $\mathrm{m}(\emptyset) = 0$ and $\sum_{A \subseteq \mathcal{Y}} \mathrm{m}(A) = 1$. This differs from standard probability as $\mathbb{P}(\mathcal{Y}) = 1$ for any $\mathbb{P} \in \mathcal{M}_1^+(\mathcal{Y}, 2^{\mathcal{Y}})$ but $\mathrm{m}(\mathcal{Y}) \leq 1$. Also, the mass allocated to non-intersecting sets does not necessarily add up to the mass allocated to the union, that is, one can have $\mathrm{m}(\{1, 2\}) \neq \mathrm{m}(\{1\}) + \mathrm{m}(\{2\})$. In this sense, DST allows for more flexibility as one can allocate mass on the set $\{1, 2\}$ without needing to specify any mass for $\{1\}$ and $\{2\}$ if uncertain. The sets $A \subseteq \mathcal{Y}$ with $\mathrm{m}(A) > 0$ are called *focal sets* of m. The mass allocated to the set of all possible alternatives $\mathrm{m}(\mathcal{Y})$ can be interpreted as the degree of ignorance; it is the mass not supporting a specific alternative within $\mathcal{Y}$.

The basic probability assignment m does not induce a single probability measure $\mathbb{P}$ on $(\mathcal{Y}, 2^{\mathcal{Y}})$ but rather a set of probability measures $\mathcal{C}_{\mathrm{m}}(\mathcal{Y}, 2^{\mathcal{Y}})$, which is called *credal set* (Abellán et al., 2006; Cuzzolin, 2021). The probability measures $\mathbb{P} \in \mathcal{C}_{\mathrm{m}}(\mathcal{Y}, 2^{\mathcal{Y}})$ are restricted by imposing lower and upper bounds, which are called *belief* and *plausibility*, respectively. They are defined as

$$\mathrm{bel}_{\mathrm{m}}(A) := \sum_{B \subseteq A} \mathrm{m}(B), \qquad \mathrm{pl}_{\mathrm{m}}(A) := 1 - \mathrm{bel}_{\mathrm{m}}(\mathcal{Y} \setminus A) = \sum_{B \subseteq \mathcal{Y}, A \cap B \neq \emptyset} \mathrm{m}(B), \tag{4.1}$$

for $A \in 2^{\mathcal{Y}}$. Recall that *belief* collects all evidence that supports a hypothesis $A \in 2^{\mathcal{Y}}$ (or a more specific one $B \subseteq A$), and *plausibility* collects all evidence that does not contradict a hypothesis $A \in 2^{\mathcal{Y}}$ (or an overlapping one $A \cap B \neq \emptyset$). With belief and plausibility as above, the set of all probability measures supporting m is

$$\mathcal{C}_{\mathrm{m}}(\mathcal{Y}, 2^{\mathcal{Y}}) := \left\{ \mathbb{P} \in \mathcal{M}_1^+(\mathcal{Y}, 2^{\mathcal{Y}}) \mid \mathrm{bel}_{\mathrm{m}}(A) \leq \mathbb{P}(A) \leq \mathrm{pl}_{\mathrm{m}}(A) \text{ for all } A \subseteq \mathcal{Y} \right\}. \tag{4.2}$$

Further, DST provides rules to combine m-s from multiple sources (Dempster, 1967; Yager, 1987a,b). This is beneficial in the PLL setting as there is several conflicting evidence about the class labels within a neighborhood of instances. Estimating a credal set $\mathcal{C}_{\mathrm{m}}(\mathcal{Y}, 2^{\mathcal{Y}})$ from such a neighborhood allows us to construct an effective reject option, which we detail in Section 4.3.2.

Several methods already leverage DST in supervised learning (Mandler and Schümann, 1988; Denoeux, 1995; Tabassian et al., 2012; Sensoy et al., 2018; Denoeux, 2019; Tong et al., 2021). We consider the nearest neighbor approach by Denoeux (1995) to be most closely related to our approach. Here, basic probability assignments are constructed from the nearest neighbors of an instance. Then, Dempster's rule is used to combine them into a single bpa. Their analysis is, however, not transferable to our case because they only have singletons or the full label space as focal sets, making set intersections in the combination rule easy to handle. In this sense, we examine a more general setting since we allocate probability mass to arbitrary subsets.

4.3. Proposed Method: DstPll

This section introduces our novel partial-label learning method DstPll. Based on the labeling information of an instance's nearest neighbors, we construct basic probability assignments within Dempster-Shafer theory. These bpas inform the prediction and rejection decisions as discussed in Section 4.3.1 and Section 4.3.2, respectively. Regarding the reject option, we propose a novel variation of the confidence-based rejection strategy: The confidence threshold is adaptively selected on a per-instance basis dependent on the amount of incorrect label noise. The more noise from incorrect labels there is, the more confident the model needs to be to accept a prediction.

Algorithm 1 outlines DstPll, which we summarize in the following. We denote by $\mathrm{NN}_l(\tilde{x}) \subseteq \mathcal{X} \times 2^{\mathcal{Y}}$ the set of the l-nearest neighbors of instance $\tilde{x}$ with their associated candidate labels. To predict the class label of an instance $\tilde{x}$ (Line 9), the algorithm first transforms information from $\tilde{x}$'s neighbors $\mathrm{NN}_l(\tilde{x})$ into bpas m_i (Lines 3–6), collects these into evidence set $\mathcal{E}$ (Line 7), and combines the bpas into $\tilde{\mathrm{m}}$ using Yager's rule (Line 8; Yager 1987a,b). Section 4.3.1 elaborates on these steps. Section 4.3.2 elaborates on how we extract our reject option from $\tilde{\mathrm{m}}$ (Line 10). We analyze our algorithm's runtime in Section 4.3.3.

4.3.1. Making Predictions

Basic probability assignments. Following the standard assumption that neighboring instances in feature space are also close in label space, we combine the evidence from the l-nearest neighbors $(x_i, s_i) \in \mathrm{NN}_l(\tilde{x})$ of a given instance $\tilde{x} \in \mathcal{X}$ with its candidate labels $\tilde{s} \subseteq \mathcal{Y}$ ($\tilde{s} = \mathcal{Y}$ if $\tilde{x}$ is a test instance).

When looking at a neighboring instance $(x_i, s_i) \in \mathrm{NN}_l(\tilde{x})$, there are generally two cases: either (i) (x_i, s_i) provides information about the correct label of $\tilde{x}$ or (ii) (x_i, s_i) is irrelevant for finding the correct label of $\tilde{x}$. To address (i), we allocate probability mass on the candidates s_i of neighbor x_i. To address (ii), we allocate probability mass on the full label space $\mathcal{Y}$ indicating uncertainty about the correct label of $\tilde{x}$.

More formally, for fixed $i \in [l]$, the candidate labels s_i do not provide any valuable information if they support all ($\tilde{s} \subseteq s_i$) or none ($\tilde{s} \cap s_i = \emptyset$) of the labels in $\tilde{s}$; we use a bpa of $\mathrm{m}_i(\tilde{s}) = 1$ (Line 4). We set $\mathrm{m}_i(A) = 1/2$ if $A = \tilde{s}$ or $A = \tilde{s} \cap s_i$, else $\mathrm{m}_i(A) = 0$ (Line 6), where $1/2$ equally weights evidence. We later elaborate further on this choice and demonstrate the application of the proposed classification rule in Example 4.3.1. Note that we make

Algorithm 1 DstPll (Our proposed method)

```
Input: PLL dataset $\mathcal{D} = \{(x_i, s_i) \in \mathcal{X} \times 2^{\mathcal{Y}} \mid i \in [n]\}$, number of nearest neighbors $l$,
    instance $\tilde{x} \in \mathcal{X}$ for inference with candidate labels $\tilde{s} \subseteq \mathcal{Y}$ ($\tilde{s} = \mathcal{Y}$ if $\tilde{x}$ is an unseen test
    instance);
Output: Prediction $g(\tilde{x})$ and reject option $\Gamma_g(\tilde{x})$ for instance $\tilde{x}$;
 1: $\mathcal{E} \leftarrow \emptyset$
 2: for $(x_i, s_i) \in \mathrm{NN}_l(\tilde{x})$ do
 3:     if $\tilde{s} \subseteq s_i$ or $\tilde{s} \cap s_i = \emptyset$ then
 4:         $\mathrm{m}_i : 2^{\mathcal{Y}} \longrightarrow [0,1],\ A \mapsto \begin{cases} 1 & \text{if } A = \tilde{s}, \\ 0 & \text{else} \end{cases}$
 5:     else
 6:         $\mathrm{m}_i : 2^{\mathcal{Y}} \longrightarrow [0,1],\ A \mapsto \begin{cases} 1/2 & \text{if } A = \tilde{s} \text{ or } A = \tilde{s} \cap s_i, \\ 0 & \text{else} \end{cases}$
 7:     $\mathcal{E} \leftarrow \mathcal{E} \cup \{\mathrm{m}_i\}$
 8: $\tilde{\mathrm{m}} \leftarrow$ yager_combination($\tilde{s}$, $\mathcal{E}$)
 9: $g(\tilde{x}) \leftarrow \begin{cases} \arg\max_{y \in \tilde{s}} \tilde{\mathrm{m}}(\{y\}) & \text{if } \max_{y \in \tilde{s}} \tilde{\mathrm{m}}(\{y\}) > 0, \\ \text{Randomly pick from } \arg\max_{A \subseteq \tilde{s}} \tilde{\mathrm{m}}(A) & \text{else;} \end{cases}$
10: $\Gamma_g(\tilde{x}) \leftarrow \begin{cases} \{g(\tilde{x})\} & \text{if } \Delta_{\tilde{\mathrm{m}}} > 0 \text{ as defined in (4.5),} \\ \emptyset & \text{else;} \end{cases}$
11: return $(g(\tilde{x}), \Gamma_g(\tilde{x}))$
```

the common assumption that the true label of instance $\tilde{x}$ is always in $\tilde{s}$ (Cour et al., 2011; Liu and Dietterich, 2012; Lv et al., 2020; Ni et al., 2021). While our definition of the m_i-s is similar to (Denoeux, 1995), we target a more general setting as our focal sets can be arbitrary subsets instead of only singletons or the full label set.

The bpa m_i has the following four effects on belief and plausibility as defined in (4.1): (i) A set of candidates A has maximal belief, that is, $\mathrm{bel}_{\mathrm{m}_i}(A) = 1$, if it covers $\tilde{s}$, that is, $\tilde{s} \subseteq A$. (ii) A set of candidates A is plausible, that is, $\mathrm{pl}_{\mathrm{m}_i}(A) > 0$, if it supports at least one of the candidate labels in $\tilde{s}$, that is, $A \cap \tilde{s} \neq \emptyset$. (iii) There is a gap, that is, $\mathrm{bel}_{\mathrm{m}_i}(A) < \mathrm{pl}_{\mathrm{m}_i}(A)$, if A supports some candidate in $\tilde{s} \cap s_i$ but does not cover all candidates in $\tilde{s} \cap s_i$ or supports some candidate in $\tilde{s}$ but does not cover all candidates of $\tilde{s}$. (iv) Class labels $y \in \tilde{s} \cap s_i$ are maximally plausible, that is, $\mathrm{pl}_{\mathrm{m}_i}(\{y\}) = 1$.

Evidence weighting. Our definition of the m_i-s (Algorithm 1, Line 6) also permits a more general view, that is, $\mathrm{m}_i(A) = \alpha$ if $A = \tilde{s}$, $\mathrm{m}_i(A) = 1 - \alpha$ if $A = \tilde{s} \cap s_i$, and $\mathrm{m}_i(A) = 0$ otherwise, for some $\alpha \in (0, 1)$. However, without further assumptions, one cannot know how relevant the information from a particular neighbor is. The setting of $\alpha = 1/2$, which we use, weights supporting and conflicting evidence of all neighbors equally. In other words, if a neighbor's evidence excludes some candidate labels from consideration, it is of equal importance compared to supporting some candidate labels. We set $\alpha = 1/2$.

Evidence combination. Given the set $\mathcal{E} = \{\mathrm{m}_i \mid i \in [l]\}$, we combine all m_i-s using Yager's rule (Yager, 1987a,b). Dempster's original rule (Dempster, 1967) enforces $\tilde{\mathrm{m}}(\emptyset) = 0$ by normalization, which is criticized for its unintuitive results when facing high conflict (Zadeh, 1984). Instead, Yager's rule first collects overlapping evidence in $\mathrm{q} : 2^{\mathcal{Y}} \to [0,1]$ and creates a valid bpa $\tilde{\mathrm{m}} : 2^{\mathcal{Y}} \to [0,1]$ by

$$\mathrm{q}(A) := \sum_{\substack{A_1,\dots,A_k \subseteq \mathcal{Y} \\ \bigcap_{i=1}^{l} A_i = A}} \prod_{j=1}^{l} \mathrm{m}_j(A_j) \qquad \text{and} \qquad \tilde{\mathrm{m}}(A) := \begin{cases} 0 & \text{if } A = \emptyset, \\ \mathrm{q}(\mathcal{Y}) + \mathrm{q}(\emptyset) & \text{if } A = \mathcal{Y}, \\ \mathrm{q}(A) & \text{else.} \end{cases} \tag{4.3}$$

We implement this efficiently with hash maps storing only the focal sets.

Classification rule. After the combination into $\tilde{\mathrm{m}}$ by (4.3), we extract a prediction $g(\tilde{x})$ for instance $\tilde{x}$ (Line 9). We predict the class label with the highest probability mass $\arg\max_{y \in \tilde{s}} \tilde{\mathrm{m}}(\{y\})$ if any has non-zero mass, or else randomly pick from the subset with the most mass $\arg\max_{A \subseteq \tilde{s}} \tilde{\mathrm{m}}(A)$. When tied, we use the subset with the smallest cardinality. In the following, we present an example of our classification rule.

Example 4.3.1 (Classification rule). Let $l = 3$, $\tilde{x}$ an unseen test instance, $(x_i, s_i) \in \mathrm{NN}_l(\tilde{x})$, $\mathcal{Y} = \{1,2,3\}$, $s_1 = \{1\}$, $s_2 = \{1,2\}$, and $s_3 = \{1,3\}$. Then, $\mathrm{m}_1(\{1\}) = \mathrm{m}_1(\mathcal{Y}) = 1/2$, $\mathrm{m}_2(\{1,2\}) = \mathrm{m}_2(\mathcal{Y}) = 1/2$, and $\mathrm{m}_3(\{1,3\}) = \mathrm{m}_3(\mathcal{Y}) = 1/2$. All other subsets receive a mass of zero. Using Yager's combination rule, we obtain $\tilde{\mathrm{m}}(\{1\}) = 5/8$, $\tilde{\mathrm{m}}(\{1,2\}) = \tilde{\mathrm{m}}(\{1,3\}) = \tilde{\mathrm{m}}(\mathcal{Y}) = 1/8$, and $\tilde{\mathrm{m}}(A) = 0$ for the remaining $A \subseteq \mathcal{Y}$. Therefore, we predict label 1 to be the class label of instance $\tilde{x}$.

4.3.2. Reject Option

As misclassifications can be quite harmful, we look at the possibility of rejecting predictions, that is, abstaining from making these predictions and, instead, deferring the decisions to humans. In the PLL setting, a reject option $\Gamma_g : \mathcal{X} \to 2^{\mathcal{Y}}$ associated with a trained classifier $g : \mathcal{X} \to \mathcal{Y}$ either returns g's prediction (*accept*) or abstains from making any prediction at all (*reject*), that is, $\Gamma_g(x) \in \{\emptyset, \{g(x)\}\}$ for $x \in \mathcal{X}$, with $\Gamma_g(x) = \emptyset$ denoting a reject. We then define the rejection probability $\mathrm{r}(\Gamma_g) = \mathbb{P}_X(\Gamma_g(X) = \emptyset)$ and the expected error

$$\mathrm{err}(\Gamma_g) = \mathbb{E}_{XS}\left[\mathbb{1}_{\{\Gamma_g(X) \neq \emptyset\}} \sum_{y=1}^{k} W_{X,S,y} \; \ell_{01}(g(X), y) \right]$$

of accepted predictions, where $\ell_{01} : \mathcal{Y}^2 \to [0,1]$, $(y, y') \mapsto \mathbb{1}_{\{y \neq y'\}}$ is the 0-1-loss and $W_{X,S,y}$ weights the loss terms similar to (2.2). Naturally, a trade-off arises between the number and accuracy of accepted predictions, leading us to the risk

$$\mathrm{R}_\lambda(\Gamma_g) = \mathbb{E}_{XS}\left[\mathbb{1}_{\{\Gamma_g(X) \neq \emptyset\}} \sum_{y=1}^{k} W_{X,S,y}\ \ell_{01}(g(X), y) + \lambda \mathbb{1}_{\{\Gamma_g(X) = \emptyset\}}\right] = \mathrm{err}(\Gamma_g) + \lambda\, \mathrm{r}(\Gamma_g), \quad (4.4)$$

for $\lambda \geq 0$; $\mathrm{err}(\Gamma_g)$ characterizes the cost of misclassification and $\lambda\, \mathrm{r}(\Gamma_g)$ the cost of rejecting a prediction.

Our method provides such a reject option Γ_g, that is, the algorithm can abstain from individual predictions if unsure (Algorithm 1, Line 10). Our formulation builds on the confidence-based rejection strategy (compare Section 3.2), that is, $\Delta := \mathrm{conf}(g) - \theta$ with $\mathrm{conf}(g) \in [0,1]$ being the model's confidence and $\theta \in [0,1]$ the confidence threshold. We adapt this setting to the PLL context by changing the confidence threshold $\theta_{\tilde{\mathrm{m}}}$ based on the amount of noise present.

Recall from (4.2) that the belief and plausibility regarding $\tilde{\mathrm{m}}$ act as a lower and upper bound of the probability mass, respectively. The intuition of our reject option is as follows. If the lower bound (belief) on the probability mass of our predicted label exceeds the maximal upper bound (plausibility) on the probability mass regarding any other label, we can safely make the prediction.

In other words, if there is a class label different from the predicted one that is quite plausible (high $\max_{y \in \tilde{s} \setminus \{\hat{y}\}} \mathrm{pl}_{\tilde{\mathrm{m}}}(\{y\})$), we require a high belief mass to be certain about the prediction, that is, the belief must satisfy $\mathrm{bel}_{\tilde{\mathrm{m}}}(\{\hat{y}\}) > \max_{y \in \tilde{s} \setminus \{\hat{y}\}} \mathrm{pl}_{\tilde{\mathrm{m}}}(\{y\})$. If, instead, there is no other plausible candidate label, we can be sure of our prediction with less belief mass.

We formalize this intuition in the following. Let $\tilde{\mathrm{m}}$ be the resulting bpa as determined by Algorithm 1 and

$$\Delta_{\tilde{\mathrm{m}}} := \underbrace{\mathrm{bel}_{\tilde{\mathrm{m}}}(\{\hat{y}\})}_{(=\mathrm{conf}(g))} - \underbrace{\max_{y \in \tilde{s} \setminus \{\hat{y}\}} \mathrm{pl}_{\tilde{\mathrm{m}}}(\{y\})}_{(=\theta_{\tilde{\mathrm{m}}})} \quad \text{with} \quad \hat{y} := \arg\max_{y \in \tilde{s}} \tilde{\mathrm{m}}(\{y\}). \quad (4.5)$$

In other words, we instantiate the model's confidence with the model's belief mass $\mathrm{conf}(g) = \mathrm{bel}_{\tilde{\mathrm{m}}}(\{\hat{y}\})$ of the predicted instance $\hat{y}$ and the confidence threshold $\theta_{\tilde{\mathrm{m}}}$ based on the amount of noise regarding other labels $y \neq \hat{y}$, that is, $\theta_{\tilde{\mathrm{m}}} = \max_{y \in \tilde{s} \setminus \{\hat{y}\}} \mathrm{pl}_{\tilde{\mathrm{m}}}(\{y\})$. Note that the dependence on $\tilde{\mathrm{m}}$ allows for setting the threshold adaptively.

(4.5) satisfies several desirable properties, which we collect in Theorem 4.3.2 and elaborate on in the following.

Theorem 4.3.2. *Let $\mathcal{Y}$ be the label space, $\tilde{x} \in \mathcal{X}$ the instance of interest, $\tilde{s} \subseteq \mathcal{Y}$ its candidate labels ($\tilde{s} = \mathcal{Y}$ if $\tilde{x}$ is a test instance), $g(\tilde{x})$ our algorithm's prediction, and $\tilde{\mathrm{m}}$ the resulting probability mass as determined by Algorithm 1. Then, the following hold:*

(i) If $g(\tilde{x})$ has been picked randomly (Algorithm 1, Line 9, second case), then $\Delta_{\tilde{\mathrm{m}}} \leq 0$,

(ii) if $\Delta_{\tilde{\mathrm{m}}} > 0$*, then* $\mathbb{P}(\{g(\tilde{x})\}) > \mathbb{P}(\{y\})$ *for all* $\mathbb{P} \in C_{\tilde{\mathrm{m}}}(\mathcal{Y}, 2^{\mathcal{Y}})$ *and* $y \in \tilde{s} \setminus \{g(\tilde{x})\}$*, and*

(iii) if $\tilde{\mathrm{m}}(\{g(\tilde{x})\}) > 1/2$*, then* $\Delta_{\tilde{\mathrm{m}}} > 0$*. The converse of (iii) does not hold.*

Based on these considerations, we define the accept and reject regions of our method as

1. **Accept.** When $\Delta_{\tilde{\mathrm{m}}} > 0$, we are in the *accept* region and $\Gamma_g(\tilde{x}) = \{\hat{y}\}$: The lower-bound probability of the class label $\hat{y}$ is greater than the upper-bound probability of any other label $y \neq \hat{y}$, that is, $\mathbb{P}(\{\hat{y}\}) > \mathbb{P}(\{y\})$ for all $\mathbb{P} \in C_{\tilde{\mathrm{m}}}(\mathcal{Y}, 2^{\mathcal{Y}})$ and $y \neq \hat{y}$.
2. **Reject.** When $\Delta_{\tilde{\mathrm{m}}} \leq 0$, we are in the *reject* region and $\Gamma_g(\tilde{x}) = \emptyset$: The lower-bound probability of the class label $\hat{y}$ is less than or equal to the upper-bound probability of another class label $y \neq \hat{y}$. There exists $\mathbb{P} \in C_{\tilde{\mathrm{m}}}(\mathcal{Y}, 2^{\mathcal{Y}})$ and $y \neq \hat{y}$ with $\mathbb{P}(\{\hat{y}\}) \leq \mathbb{P}(\{y\})$.

We remark that Theorem 4.3.2 (iii) implies that DstPll rejects fewer predictions than the l-nearest neighbor reject option by Hellman (1970), which requires more than 1/2 of all votes to be certain. Our method can accept predictions with less than 1/2 of all probability mass on a single class label, while still satisfying Theorem 4.3.2 (ii), that is, if a prediction is accepted, the decision remains unchanged independent of $\mathbb{P} \in C_{\tilde{\mathrm{m}}}(\mathcal{Y}, 2^{\mathcal{Y}})$. It follows that having fewer rejections does not come at the expense of an increase in the expected error. In the following, we provide an example of the proposed reject option.

Example 4.3.3 (Reject option). Assume the setting and result of Example 4.3.1: $\tilde{\mathrm{m}}(\{1\}) = 5/8$, $\tilde{\mathrm{m}}(\{1,2\}) = \tilde{\mathrm{m}}(\{1,3\}) = \tilde{\mathrm{m}}(\mathcal{Y}) = 1/8$, and $\tilde{\mathrm{m}}(A) = 0$ for the remaining $A \subseteq \mathcal{Y}$. Our prediction is $\hat{y} = \arg\max_{y \in \tilde{s}} \tilde{\mathrm{m}}(\{y\}) = 1$ with $\tilde{s} = \mathcal{Y}$. Hence, $\Delta_{\tilde{\mathrm{m}}} = \mathrm{bel}_{\tilde{\mathrm{m}}}(\{\hat{y}\}) - \max_{y \in \tilde{s} \setminus \{\hat{y}\}} \mathrm{pl}_{\tilde{\mathrm{m}}}(\{y\}) = \mathrm{bel}_{\tilde{\mathrm{m}}}(\{1\}) - \mathrm{pl}_{\tilde{\mathrm{m}}}(\{2\}) = 5/8 - 2/8 = 3/8$. Since $\Delta_{\tilde{\mathrm{m}}} = 3/8 > 0$, we accept the prediction $\hat{y} = 1$.

4.3.3. Runtime Analysis

We decompose the overall runtime of our approach in Algorithm 1 into l-times querying one nearest neighbor and creating its bpa (Lines 3–7), the cost of Yager's rule (Line 8), as well as extracting predictions in Lines 9-10. Using the ball-tree data structure (Omohundro, 1989), querying one neighbor takes $O(d \log n)$ time on average. In the worst case, query time is $O(dn)$. One builds a ball-tree in $O(dn \log n)$ time. We construct a bpa m_i by storing its focal sets within a hash map and combine all m_i-s as defined in (4.3). There are at most $\min(2^k, 2^l)$ focal sets of $\tilde{\mathrm{m}}$: Each m_i has at most two focal sets producing 2^l combinations and there are at most 2^k possible subsets of $\mathcal{Y}$. We take the minimum as both are upper bounds. Looking up a focal set in the hash-map requires $O(k)$ time as the key length is variable. Extracting a prediction and the reject option then requires $O(k^2)$ time. Combining the above yields a worst-case complexity of $O(dln + k\left(\min(2^k, 2^l) + \max(k, l)\right))$. Since l and k are constant, the nearest-neighbor search dominates. The average total runtime of the search is $O(dl \log n)$.

4.4. Experiments

Section 4.4.1 lists the tested methods, Section 4.4.2 shows our experimental setup, and Section 4.4.3 collects our main findings. Additional results and a description of all hyperparameters can be found in Appendix A.

4.4.1. Algorithms for Comparison

While many PLL algorithms exist (see Section 3.1), we focus on state-of-the-art methods commonly used in the literature. We consider ten methods: PlKnn (Hüllermeier and Beringer, 2005), PlSvm (Nguyen and Caruana, 2008), Ipal (Zhang and Yu, 2015), PlEcoc (Zhang et al., 2017), Proden (Lv et al., 2020), Cc (Feng et al., 2020), Valen (Xu et al., 2021), Pop (Xu et al., 2023), CroSel (Tian et al., 2024), and DstPll (our proposed method).

We choose the parameters of all methods as recommended by their respective authors and use the same base models for all of the neural network approaches. Appendix A.1 discusses the choices of all hyperparameters in more detail. As base models, we pick the LeNet architecture (LeCun et al., 1998) for the *mnist*-like datasets and a d-300-300-300-k MLP (Werbos, 1974) for all other datasets. For PlKnn and our method, we use variational auto-encoders (Kingma and Welling, 2014) to reduce the feature space dimensionality of the *mnist*-like datasets and compute the nearest neighbors on the hidden representations. As our competitors do not provide a reject option, we use a threshold on their model's confidences, that is, the maximum probability outputs, to evaluate the trade-off between the fraction and accuracy of non-rejected predictions.

4.4.2. Experimental Setup

Data. Following the default protocol (Cour et al., 2011; Zhang and Yu, 2015; Lv et al., 2020; Xu et al., 2023), we conduct several experiments using datasets for supervised learning with added artificial noise as well as experiments on real-world partially-labeled data. We repeat all experiments five times to report averages and standard deviations. For the supervised datasets, we use the *ecoli* (Horton and Nakai, 1996), *multiple-features* (Duin, 2002), *pen-digits* (Alpaydin and Alimoglu, 1998), *semeion* (Buscema and Terzi, 2008), *solar-flare* (Dodson and Hedeman, 1989), *statlog-landsat* (Srinivasan, 1993), and *theorem* datasets (Bridge et al., 2013) from the UCI repository (Bache and Lichman, 2013). These datasets contain between 336 and 10 992 instances each. Also, we use the popular *mnist* (LeCun et al., 1999), *kmnist* (Clanuwat et al., 2018), and *fmnist* datasets (Xiao et al., 2018), which contain 60 000 images each similar to other datasets like *cifar10* and *cifar100* (Krizhevsky, 2009). For the partially labeled data, we use the *bird-song* (Briggs et al., 2012), *mir-flickr* (Huiskes and Lew, 2008), *yahoo-news* (Guillaumin et al., 2010), and *msrc-v2* datasets (Liu and Dietterich, 2012). They contain between 1755 and 22 762 instances.

Table 4.1.: Average test-set accuracies (± std.) on the UCI, *mnist*-like, and real-world datasets. The UCI and *mnist*-like datasets are artificially augmented with class-dependent and instance-dependent noise. The best algorithms (highest accuracy) as well as algorithms with non-significant differences are emphasized. The significance analysis uses a paired student t-test with level $\alpha = 0.05$.

Algorithms	UCI datasets		*mnist*-like datasets		Real-world
	Class-dep.	Inst.-dep.	Class-dep.	Inst.-dep.	
PlKnn (2005)	**81.2** (± 13.9)	75.8 (± 11.1)	92.2 (± 4.1)	84.8 (± 7.2)	53.4 (± 10.9)
PlSvm (2008)	62.3 (± 16.3)	43.8 (± 16.4)	67.5 (± 9.8)	47.8 (± 8.2)	39.1 (± 9.6)
Ipal (2015)	79.3 (± 17.1)	75.3 (± 18.3)	92.9 (± 4.3)	88.5 (± 6.6)	58.7 (± 9.8)
PlEcoc (2017)	63.7 (± 13.1)	66.5 (± 12.5)	64.3 (± 14.3)	51.7 (± 10.7)	46.2 (± 10.2)
Proden (2020)	**81.6** (± 14.0)	**78.1** (± 13.2)	**93.9** (± 4.4)	88.2 (± 6.0)	**64.2** (± 8.2)
Cc (2020)	**81.3** (± 14.2)	**78.8** (± 13.6)	**93.9** (± 4.5)	**89.6** (± 5.7)	49.2 (± 29.7)
Valen (2021)	79.7 (± 15.3)	75.6 (± 12.7)	91.8 (± 4.2)	83.4 (± 8.4)	**63.6** (± 9.7)
Pop (2023)	**81.5** (± 14.0)	**78.1** (± 13.1)	**93.9** (± 4.5)	88.1 (± 6.0)	**63.6** (± 8.4)
CroSel (2024)	79.9 (± 17.2)	**78.4** (± 14.5)	**94.2** (± 4.5)	**88.9** (± 6.7)	46.3 (± 28.9)
DstPll (ours)	**80.8** (± 14.1)	75.4 (± 10.7)	92.0 (± 4.2)	84.5 (± 7.2)	52.0 (± 11.6)

Noise Generation. We use three noise generation strategies to introduce candidate labels into the supervised datasets: uniform, class-dependent, and instance-dependent noise. Uniform noise (Liu and Dietterich, 2012) adds three uniform random noise labels to a fraction of 70 % of all instances. Class-dependent noise (Cour et al., 2011) randomly partitions all class labels into pairs and adds the partner label as noise to 70 % of all instances having the other label. Instance-dependent noise (Zhang et al., 2021) first trains a supervised probabilistic classifier $g : \mathcal{X} \rightarrow \Delta^{\mathrm{k-1}}$. Given an instance x with true label y, a flipping probability of $\xi_{\bar{y}}(x) := g_{\bar{y}}(x) / \max_{y' \in \mathcal{Y} \setminus \{y\}} g_{y'}(x)$ for $\bar{y} \neq y$ determines which noise labels to randomly pick.

4.4.3. Results

Prediction Performance. Table 4.1 shows the average test-set accuracies and standard deviations over all UCI datasets with class- and instance-dependent noise, *mnist*-like datasets with class- and instance-dependent noise, and the real-world datasets. The algorithms with the highest accuracies as well as the algorithms with non-significant differences using a paired t-test with level $\alpha = 0.05$ are emphasized. Our approach (DstPll) performs comparably to the other methods. We note that none of the methods is best across all settings. For example, Cc performs best in four out of five settings but is significantly outperformed by our approach on the real-world experiments.

Reject Option. To compare our reject option with the other methods, we use a threshold of $\Delta_{\tilde{\mathrm{m}}} > 0$ for our proposed approach (Algorithm 1, Line 10), a confidence threshold of 90 %

Table 4.2.: Average empirical reject option risk $\hat{R}_\lambda(\Gamma_g)$ as defined in (4.4) for various values of λ. The best algorithms (smallest risk) as well as algorithms with non-significant differences are emphasized. The significance analysis uses a paired student t-test with level $\alpha = 0.05$.

Algorithms	Average $\hat{R}_\lambda(\Gamma_g)$ (± std.) across all experimental settings				
	$\lambda = 0.00$	$\lambda = 0.05$	$\lambda = 0.10$	$\lambda = 0.15$	$\lambda = 0.20$
PlKnn (2005)	0.11 (± 0.17)	0.15 (± 0.18)	0.18 (± 0.19)	0.21 (± 0.20)	0.25 (± 0.21)
PlSvm (2008)	0.19 (± 0.27)	0.23 (± 0.28)	0.27 (± 0.29)	0.31 (± 0.30)	0.35 (± 0.31)
Ipal (2015)	0.19 (± 0.17)	0.20 (± 0.18)	0.21 (± 0.19)	0.22 (± 0.20)	0.23 (± 0.21)
PlEcoc (2017)	0.25 (± 0.27)	0.29 (± 0.27)	0.34 (± 0.28)	0.38 (± 0.28)	0.43 (± 0.28)
Proden (2020)	0.12 (± 0.11)	0.13 (± 0.11)	0.14 (± 0.12)	0.15 (± 0.12)	**0.16** (± 0.13)
Cc (2020)	0.16 (± 0.18)	0.16 (± 0.19)	0.17 (± 0.20)	0.18 (± 0.20)	0.19 (± 0.21)
Valen (2021)	0.20 (± 0.14)	0.20 (± 0.14)	0.20 (± 0.14)	0.20 (± 0.14)	0.21 (± 0.14)
Pop (2023)	0.12 (± 0.11)	0.13 (± 0.11)	0.14 (± 0.12)	0.15 (± 0.12)	**0.16** (± 0.13)
CroSel (2024)	0.15 (± 0.20)	0.16 (± 0.21)	0.17 (± 0.21)	0.18 (± 0.21)	0.19 (± 0.22)
DstPll (ours)	**0.05** (± 0.07)	**0.07** (± 0.08)	**0.10** (± 0.10)	**0.12** (± 0.11)	**0.15** (± 0.12)

Table 4.3.: Fraction of rejects and non-rejected test-set accuracy of all methods with std. across all experimental settings and five repetitions with different random seeds. To obtain the results, we tune the thresholds θ such that each competitor has a number of rejects that is comparable to that of our method.

Algorithms	Fraction of rejects (± std.)	Non-rejected test accuracy (± std.)
PlKnn (2005)	50.19 % (± 20.98 %)	91.23 % (± 10.12 %)
PlSvm (2008)	50.19 % (± 20.98 %)	74.40 % (± 19.77 %)
Ipal (2015)	50.19 % (± 20.98 %)	83.52 % (± 16.08 %)
PlEcoc (2017)	50.19 % (± 20.98 %)	73.92 % (± 17.11 %)
Proden (2020)	50.19 % (± 20.98 %)	94.03 % (± 8.23 %)
Cc (2020)	50.19 % (± 20.98 %)	90.13 % (± 17.89 %)
Valen (2021)	50.19 % (± 20.98 %)	86.81 % (± 13.04 %)
Pop (2023)	50.19 % (± 20.98 %)	94.02 % (± 8.20 %)
CroSel (2024)	50.19 % (± 20.98 %)	89.71 % (± 18.67 %)
DstPll (ours)	50.15 % (± 21.16 %)	**95.49 %** (± 7.01 %)

for classifiers outputting a probability distribution over the class labels, and a threshold of 50 % of all votes for PlKnn to not reject a prediction, which is in line with the reject option by Hellman (1970).

Table 4.2 shows the average empirical reject-option risk and standard deviation across all experiments for varying λ. We compute $\hat{R}_\lambda(\Gamma_g) = \hat{\mathrm{err}}(\Gamma_g) + \lambda \hat{r}(\Gamma_g)$ by using ground-truth information to calculate the non-reject error $\hat{\mathrm{err}}(\Gamma_g)$ and counting the number of rejects to calculate the reject rate $\hat{r}(\Gamma_g)$. The algorithms with the lowest risks as well as the algorithms

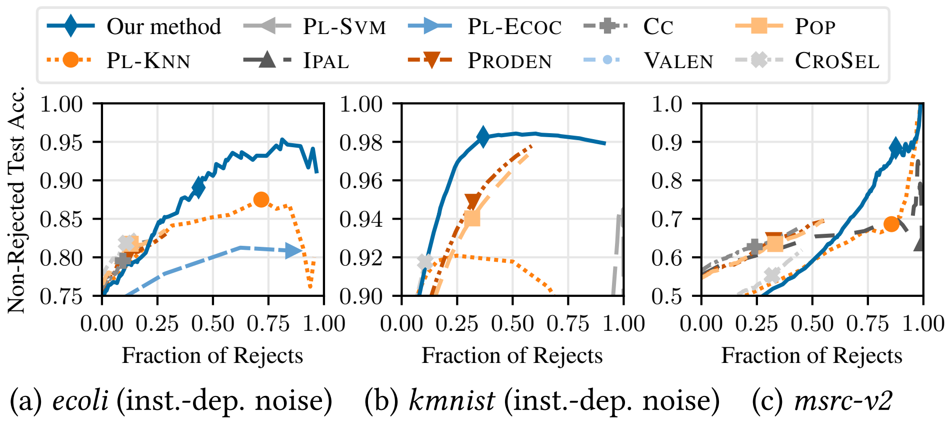


Figure 4.1.: Trade-off between the fraction of rejected predictions and the accuracy of non-rejected predictions for three experiments: *ecoli* with instance-dependent noise, *kmnist* with instance-dependent noise, and the real-world dataset *msrc-v2*. We show the trade-off curves for varying confidence (0 to 1) and $\Delta_{\tilde{m}}$ (-1 to 1) thresholds. We highlight the points corresponding to a threshold of $\Delta_{\tilde{m}} = 0$ for our method, a confidence threshold of 90 % for methods with a probability output, and a threshold of 50 % of all votes for PlKnn. We refer to Appendix A.3 for all reject trade-off curves across all experimental settings.

with non-significant differences using a paired t-test with level $\alpha = 0.05$ are emphasized. When misclassification is costly ($\lambda \leq 0.2$), our method provides the significantly best trade-off compared to our competitors. In contrast, when rejecting predictions is costly ($\lambda > 0.2$), the methods in Table 4.1 are to be preferred.

Table 4.3 shows the non-rejected test accuracy of all methods across all experimental settings for a fixed fraction of rejects. To obtain the results, we tune the confidence thresholds $\theta \in [0, 1]$ such that each competitor rejects a similar number of instances as our proposed approach. Our approach uses the threshold $\Delta > 0$, for which we have proved several desirable guarantees in Theorem 4.3.2. Our method achieves superior test-set accuracy on the non-rejected predictions.

Figure 4.1 shows the reject trade-off for varying confidence (0 to 1) and $\Delta_{\tilde{m}}$ (-1 to 1) thresholds on the *ecoli* and *knnist* datasets with instance-dependent noise as well as on the *msrc-v2* real-world dataset. The x-axes show the fractions of predictions that are rejected. The y-axes show the accuracies of predictions that are not rejected. The plots show (fraction of rejects, non-rejected test-set accuracy)-pairs corresponding to different settings of the thresholds. Most methods have monotonic growth: The more predictions are rejected, the more accurate are non-rejected predictions. Also, it is desirable to be close to the top-left corner of the plots as one wants to achieve high accuracy while rejecting as few predictions as possible. Note that the point (1, 1) cannot be observed as the test-set accuracy is undefined if all predictions are rejected. Given a desired rejection rate $\hat{r}(\Gamma_g)$, Figure 4.1 also allows for numerically finding the appropriate value of λ that minimizes $\hat{R}_\lambda(\Gamma_g)$. Varying λ in (4.4), while having $\hat{err}(\Gamma_g)$ and $\hat{r}(\Gamma_g)$ fixed, yields a straight line in Figure 4.1 showing all possible trade-offs.

Our method provides the significantly best trade-offs compared to competitors (Table 4.2). These results empirically demonstrate our method's superiority in reliably rejecting unsure predictions.

4.5. Proof of Theorem 4.3.2

Part (i)**.** Given an instance $\tilde{x} \in \mathcal{X}$ with its candidate set $\tilde{s} \subseteq \mathcal{Y}$ and its associated prediction $g(\tilde{x})$ as described in Algorithm 1, we assume that $g(\tilde{x})$ has been picked randomly from $\arg\max_{A \subseteq \tilde{s}} \tilde{\mathrm{m}}(A)$ (Algorithm 1, Line 9, second case), so $\max_{y \in \tilde{s}} \tilde{\mathrm{m}}(\{y\})$ must be zero because we are not in the first case in Line 9. Therefore, $\mathrm{bel}_{\tilde{\mathrm{m}}}(\{y\}) \overset{(4.1)}{=} \tilde{\mathrm{m}}(\{y\}) = 0$ for all $y \in \mathcal{Y}$. Substitution in $\Delta_{\tilde{\mathrm{m}}}$ yields

$$\Delta_{\tilde{\mathrm{m}}} = \mathrm{bel}_{\tilde{\mathrm{m}}}(\{y\}) - \max_{y' \in \tilde{s} \setminus \{y\}} \mathrm{pl}_{\tilde{\mathrm{m}}}(\{y'\}) = 0 - \max_{y' \in \tilde{s} \setminus \{y\}} \mathrm{pl}_{\tilde{\mathrm{m}}}(\{y'\}) \leq 0,$$

independent of the choice of $y \in \mathcal{Y}$. Therefore, $\Delta_{\tilde{\mathrm{m}}} \leq 0$.

Part (ii)**.** Given an instance $\tilde{x} \in \mathcal{X}$ with its candidate set $\tilde{s} \subseteq \mathcal{Y}$ and its associated prediction $g(\tilde{x})$ as described in Algorithm 1, we assume that $\Delta_{\tilde{\mathrm{m}}} > 0$. By the contraposition of part (i), $g(\tilde{x}) = \hat{y}$ with $\hat{y} = \arg\max_{y \in \tilde{s}} \tilde{\mathrm{m}}(\{y\})$. From (4.5) and $\Delta_{\tilde{\mathrm{m}}} > 0$, it follows that $\mathrm{bel}_{\tilde{\mathrm{m}}}(\{\hat{y}\}) > \max_{y \in \tilde{s} \setminus \{\hat{y}\}} \mathrm{pl}_{\tilde{\mathrm{m}}}(\{y\})$, which is equivalent to $\mathrm{bel}_{\tilde{\mathrm{m}}}(\{\hat{y}\}) > \mathrm{pl}_{\tilde{\mathrm{m}}}(\{y\})$ for all $y \in \tilde{s} \setminus \{\hat{y}\}$. For all $\mathbb{P} \in \mathcal{C}_{\tilde{\mathrm{m}}}(\mathcal{Y}, 2^{\mathcal{Y}})$, it holds that $\mathrm{bel}_{\tilde{\mathrm{m}}}(A) \leq \mathbb{P}(A) \leq \mathrm{pl}_{\tilde{\mathrm{m}}}(A)$ for all $A \subseteq \mathcal{Y}$ by (4.2). Therefore, $\mathbb{P}(\{y\}) \leq \mathrm{pl}_{\tilde{\mathrm{m}}}(\{y\}) < \mathrm{bel}_{\tilde{\mathrm{m}}}(\{\hat{y}\}) \leq \mathbb{P}(\{\hat{y}\})$ for all $y \in \tilde{s} \setminus \{\hat{y}\}$.

Part (iii)**.** $(\Rightarrow)$: Given an instance $\tilde{x} \in \mathcal{X}$ with its candidate set $\tilde{s} \subseteq \mathcal{Y}$ and its associated prediction $g(\tilde{x})$ as described in Algorithm 1, we assume that $\tilde{\mathrm{m}}(\{g(\tilde{x})\}) > 1/2$. Because $\max_{y \in \tilde{s}} \tilde{\mathrm{m}}(\{y\}) > 0$, we are in the first case in Line 9 of Algorithm 1. Therefore, $g(\tilde{x}) = \hat{y}$ with $\hat{y} = \arg\max_{y \in \tilde{s}} \tilde{\mathrm{m}}(\{y\})$. As $\sum_{A \subseteq \mathcal{Y}} \tilde{\mathrm{m}}(A) = 1$ and $\tilde{\mathrm{m}}(\{\hat{y}\}) > 1/2$, it holds that $\sum_{A \subseteq \mathcal{Y}, A \neq \{\hat{y}\}} \tilde{\mathrm{m}}(A) < 1/2$. Then, for all $y \in \tilde{s}$ with $y \neq \hat{y}$, $\mathrm{pl}_{\tilde{\mathrm{m}}}(\{y\}) = \sum_{A \subseteq \mathcal{Y}, A \cap \{y\} \neq \emptyset} \tilde{\mathrm{m}}(A) < 1/2$. Hence, $\mathrm{pl}_{\tilde{\mathrm{m}}}(\{y\}) < \mathrm{bel}_{\tilde{\mathrm{m}}}(\{\hat{y}\})$ for all $y \in \tilde{s}$ with $y \neq \hat{y}$. Therefore, $\Delta_{\tilde{\mathrm{m}}} = \mathrm{bel}_{\tilde{\mathrm{m}}}(\{\hat{y}\}) - \max_{y \in \tilde{s} \setminus \{\hat{y}\}} \mathrm{pl}_{\tilde{\mathrm{m}}}(\{y\}) > 0$.

$(\not\Leftarrow)$: In the following, we provide a counter-example. Let $\tilde{s} = \mathcal{Y} = \{1, 2, 3\}$ and $\tilde{\mathrm{m}}$ be defined by $\tilde{\mathrm{m}}(A) = 0.4$ if $A = \{1\}$, $\tilde{\mathrm{m}}(A) = 0.3$ if $A = \{1, 2\}$ or $A = \{1, 3\}$, else $\tilde{\mathrm{m}}(A) = 0$. Then, $y = 1$ is our prediction since it has the highest probability mass. The prediction is not rejected because $\Delta_{\tilde{\mathrm{m}}} = 0.4 - 0.3 = 0.1 > 0$. However, $\tilde{\mathrm{m}}(\{y\}) < 1/2$.

5. Robust Partial-Label Learning by Leveraging Class Activation Values

This chapter's contents are based on the following publication.

- Tobias Fuchs and Florian Kalinke. Robust partial-label learning by leveraging class activation values. *Machine Learning*, 114(193), 2025. `https://doi.org/10.1007/s10994-025-06796-z`.

All code and data for reproducing this chapter's experiments are available at `https://github.com/mathefuchs/robust-pll`.

5.1. Overview

As predictions by machine-learning systems often impact actions or decisions by humans, they should be *robust* regarding several criteria to limit misclassifications and their effects. Three common criteria are robustness against (a) high noise levels (Zhang et al., 2021), (b) out-of-distribution data (Sensoy et al., 2018), and (c) adversarial attacks (Madry et al., 2018). Consider, for example, safety-critical domains such as medical image classification (Yang et al., 2009; Lambrou et al., 2011; Reamaroon et al., 2019) or financial fraud detection (Cheng et al., 2020; Berkmans and Karthick, 2023; Xiang et al., 2023), where all three criteria (a) – (c) are of interest.

Robustness in terms of (a), (b), and (c) is especially important in PLL because of its noisy and inexact supervision. While different noise generation processes (a) are well-examined in PLL, it is still open to investigate the impact of (b) and (c) on PLL algorithms. Dealing with out-of-distribution data (b) is essential in the web mining use case of PLL as the closed-world assumption usually does not hold, that is, an algorithm should recognize instances that do not belong to any known class. Addressing adversarial modifications of input features (c) is also critical, given that much of the training data is human-based, presenting a potential vulnerability. Tackling (a) – (c) is particularly challenging in the PLL domain as there is no exact ground truth on which an algorithm can rely to build robust representations. Our proposed PLL method is unique in its ability to perform well across all three aspects.

In this work, we propose a novel PLL deep-learning algorithm that leverages the magnitudes of class activation values within the subjective logic framework (Jøsang, 2016). Subjective logic allows for explicitly representing uncertainty in predictions, which is highly beneficial in dealing with the challenges (a) – (c). The details are as follows. When dealing with noise from the PLL candidate sets (a), having good uncertainty estimates supports the propagation of meaningful labeling information as it allows us to put more weight on class labels with a low uncertainty and to restrict the influence of noisy class labels, which have high uncertainty. We tackle out-of-distribution data (b) by optimizing for high uncertainty when the correct class label is excluded from the set of all possible class labels. Adversarial modifications of the input features (c) are addressed similarly to (a), as our approach provides reliable uncertainty estimates near the decision boundaries of the class labels.

The supervised classification approach by Sensoy et al. (2018) is the most similar to the proposed approach as both employ the subjective logic framework. However, it is highly non-trivial to extend the methods from the supervised to the PLL setting as the existing work relies on exact ground truth, which is generally unavailable in PLL. We attack this problem by proposing a novel representation of partially-labeled data within the subjective logic framework and give an optimal update strategy for the candidate label weights with respect to the model's loss term. Subjective logic allows us to deal with the partially labeled data in a principled fashion by jointly learning the candidate labels' weights and their associated uncertainties.

Contributions. Our contributions are as follows.

- We introduce RobustPll, a novel partial-label learning algorithm, which leverages the model's class activation values within the subjective logic framework.
- We empirically demonstrate that RobustPll yields more robust predictions than our competitors. The proposed method achieves state-of-the-art prediction performance under high PLL noise and can deal with out-of-distribution examples and examples corrupted by adversarial noise more reliably. Our code and data are publicly available.
- Our analysis of RobustPll shows that the proposed label weight update strategy is optimal in terms of the mean-squared error and allows for reinterpretation within the subjective logic framework. Further, we discuss our method's runtime and show that it yields the same runtime complexity as other state-of-the-art PLL algorithms.

Outline. Section 5.2 gives relevant background information on the subjective logic framework. We propose our PLL method in Section 5.3 and show our experiments in Section 5.4. Section 5.5 contains all proofs. Appendix B lists all hyperparameter choices and shows additional results.

5.2. Background

Inspired by Dempster-Shafer theory (Dempster, 1967; Shafer, 1986), Jøsang (2016) proposes a theory of evidence, called subjective logic (SL), that explicitly represents (epistemic) uncertainty in prediction-making. In subjective logic, the tuple $\omega_i = (\mathfrak{b}_i, \mathfrak{a}_i, \mathfrak{u}_i) \in [0,1]^k \times [0,1]^k \times [0,1]$ denotes a multinomial opinion about instance $i \in [n]$ with $\mathfrak{b}_i$ representing the belief mass of the class labels, $\mathfrak{a}_i$ representing the prior knowledge about the class labels, and $\mathfrak{u}_i$ explicitly represents the uncertainty involved in predicting i. ω_i satisfies $\|\mathfrak{a}_i\|_1 = 1$, where $\|\mathfrak{a}_i\|_p = (\sum_{j=1}^{k} |\mathfrak{a}_{ij}|^p)^{1/p}$ denotes the p-norm, and requires additivity, that is, $\mathfrak{u}_i + \|\mathfrak{b}_i\|_1 = 1$ for all $i \in [n]$. The projected probability is defined by $\bar{p}_i = \mathfrak{b}_i + \mathfrak{a}_i \mathfrak{u}_i \in \Delta^{\text{k-1}}$ for $i \in [n]$. $\bar{p}_i$ induces a probability measure $\mathbb{P}_i$ on the measurable space $(\mathcal{Y}, 2^{\mathcal{Y}})$ with $\mathbb{P}_i(A) = \sum_{j \in A} \bar{p}_{ij}$ for all $A \in 2^{\mathcal{Y}}$. Given instance $i \in [n]$, features $x_i \in \mathcal{X}$, and a prediction model $f : \mathbb{R}^d \to \mathbb{R}^k_{\geq 0}$, we set $\mathfrak{b}_i = f(x_i)/(k + \|f(x_i)\|_1)$, $\mathfrak{a}_i$ using prior knowledge, and $\mathfrak{u}_i = k/(k + \|f(x_i)\|_1)$. The multinomial opinion ω_i can be expressed in terms of a Dirichlet-distributed random variable with parameters $\alpha_i = f(x_i) + 1$, that is, $\mathbb{E}_{p_i \sim \text{Dir}(\alpha_i)}[p_i] := \alpha_i / \|\alpha_i\|_1 = \mathfrak{b}_i + \mathfrak{a}_i \mathfrak{u}_i = \bar{p}_i$, with $\mathfrak{b}_i$ and $\mathfrak{u}_i$ as defined above, and uniform prior $\mathfrak{a}_{ij} = 1/k$ ($i \in [n]$, $j \in [k]$). A multinomial opinion ω_i that is maximally uncertain, that is, $\mathfrak{u}_i = 1$, defaults to prior knowledge $\mathfrak{a}_i$: In this case, $\bar{p}_i = \mathfrak{a}_i$ for $i \in [n]$. In the following, we provide an example of multinomial opinions in subjective logic.

Example 5.2.1. Let $n = 2$, $k = 3$, $\mathcal{Y} = \{1,2,3\}$, $\mathfrak{b}_1 = (2/3, 1/6, 1/6)$, $\mathfrak{b}_2 = (1/2, 0, 0)$, $\mathfrak{a}_1 = \mathfrak{a}_2 = (1/3, 1/3, 1/3)$, $\mathfrak{u}_1 = 0$, and $\mathfrak{u}_2 = 1/2$. Then, both multinomial opinions, $\omega_1 = (\mathfrak{b}_1, \mathfrak{a}_1, \mathfrak{u}_1)$ and $\omega_2 = (\mathfrak{b}_2, \mathfrak{a}_2, \mathfrak{u}_2)$, yield the same projected probabilities $\bar{p}_1 = \bar{p}_2 = (2/3, 1/6, 1/6)$. While ω_1 and ω_2 both induce the same probability measure on $(\mathcal{Y}, 2^{\mathcal{Y}})$, ω_2 contains more uncertainty than ω_1, that is, there is more evidence that supports ω_1 than ω_2. Also, both multinomial opinions induce a Dirichlet-distributed random variable with equal mean but different variances indicating different degrees of certainty about the labeling, which will be helpful in disambiguating the PLL data.

5.3. Proposed Method: RobustPll

We present a novel PLL method that yields robust predictions in terms of good predictive performance, robustness against out-of-distribution examples, and robustness against adversarial examples. Tackling all is especially challenging in PLL as full ground truth is not available.

Given a prediction, one commonly uses softmax normalization, softmax : $\mathbb{R}^k \to [0,1]^k$, to output a discrete probability distribution over possible targets (Bishop, 2007). It has been noted, however, that softmax normalization cannot represent the uncertainty involved in prediction-making (Hüllermeier and Waegeman, 2021; Sale et al., 2023), which is also evident from Example 5.2.1, where different amounts of uncertainty can be associated with the same probability measure. In partial-label learning, where candidate labels are iteratively refined, it is crucial to accurately reflect the uncertainty involved in the candidate labels to effectively propagate labeling information. Our method explicitly represents uncertainty through the SL framework by using a neural-network that parameterizes a Dirichlet distribution rather than using softmax normalization. By jointly learning the candidate label weights as well as their associated uncertainty, our approach builds robust

Algorithm 2 RobustPll (Our proposed method)

```
Input: PLL dataset $\mathcal{D} = \{(x_i, s_i) \in \mathcal{X} \times 2^{\mathcal{Y}} \mid i \in [n]\}$ with $n$ instances, $k$ classes;
Output: Model parameters $\theta$;
1: Init weights $\mathrm{w}_{ij} \longleftarrow \mathbb{1}_{\{j \in s_i\}}/|s_i|$ for $i \in [n]$, $j \in [k]$;
2: Init model $f$ and its parameters $\theta$;
3: for $t = 1, \ldots, T$ do
4:     $\lambda_t \longleftarrow \min(2t/T, 1)$
5:     Compute the empirical risk $\hat{\mathrm{R}}(f; \lambda_t)$ by (5.4);
6:     Update $\theta$ by backpropagation;
7:     Set label weights $\mathrm{w}_i$ ($i \in [n]$) to the solution $\mathrm{w}_i^*$ of (5.5) using the closed-form
       solution in Proposition 5.3.3;
```

representations, which help in dealing with out-of-distribution data and adversarially corrupted features.

Similar to an expectation-maximization procedure, we interleave learning the parameters of our prediction model with updating the current labeling information of all instances based on the discovered knowledge and prior information.

Algorithm 2 outlines our method: RobustPll. First, we initialize the label weights w_{ij} in Line 1. These weights $\mathrm{w}_i \in \Delta^{\mathrm{k}-1}$ represent the model's knowledge about the labeling of instance $i \in [n]$. $\mathrm{w}_{ij} \in [0, 1]$ denotes class j's weight regarding instance i. Section 5.3.1 discusses their initialization and interpretation. In Line 2, we set up our model $f : \mathbb{R}^d \rightarrow \mathbb{R}^k_{\geq 0}$ and its parameters θ. Our framework is independent of the concrete model choice. For example, one may use MLPs (Rumelhart et al., 1986), LeNet (LeCun et al., 1998), or ResNet (He et al., 2016). One only needs to modify the last layer, which is required to be a ReLU layer to enforce non-negative outputs for the SL framework. Lines 3–7 contain the main training loop of our approach. We train for a total of T epochs. Note that, in practice, we make use of mini-batches. We set the annealing coefficient λ_t in Line 4. The coefficient controls the influence of the regularization term in $\hat{\mathrm{R}}(f; \lambda_t)$, which is discussed in Section 5.3.3. In Line 5, we then compute the empirical risk $\hat{\mathrm{R}}(f; \lambda_t)$ and update the model parameters θ in Line 6. Those steps are discussed in Section 5.3.2. Thereafter, we update the label weights w_{ij} in Line 7 as shown in Section 5.3.4.

The remainder of Section 5.3 presents further analyses. In Section 5.3.5, we discuss our reinterpretation of the label weight update within SL. Section 5.3.6 bounds the rate of change of w_i and Section 5.3.7 demonstrates why the squared error loss is superior to the cross-entropy loss in our setting. Section 5.3.8 discusses our approach's runtime.

5.3.1. Initializing the Label Weights

The label weights w_{ij} represent the current knowledge of our method about instance i having the correct label j. They must sum to one, that is, $\| \mathrm{w}_i \|_1 = 1$, and must be zero if j

is not a candidate label of instance i, that is, $\mathrm{w}_{ij} = 0$ if $j \notin s_i$ ($i \in [n]$). We initialize the label weights with $\mathrm{w}_{ij} = \mathbb{1}_{\{j \in s_i\}} / |s_i|$, where $\mathbb{1}_{\{\cdot\}}$ denotes the indicator function. w_{ij} satisfies both requirements. Also, w_{ij} can be written as a multinomial opinion $\omega_i = (\mathfrak{b}_i, \mathfrak{a}_i, \mathfrak{u}_i)$ in SL with maximal uncertainty, that is, $\mathrm{w}_i = \mathfrak{b}_i + \mathfrak{a}_i \mathfrak{u}_i$ with zero belief $\mathfrak{b}_{ij} = 0$, uniform prior weights $\mathfrak{a}_{ij} = \mathbb{1}_{\{j \in s_i\}} / |s_i|$, and maximal uncertainty $\mathfrak{u}_i = 1$ ($i \in [n]$, $j \in [k]$). Note that $\mathrm{w}_i = \mathfrak{a}_i$ at initialization: The label weights w_i are solely determined by prior knowledge about the candidate sets encoded in $\mathfrak{a}_i$.

5.3.2. Training a Model

We interleave learning the parameters θ of a model $f : \mathbb{R}^d \to \mathbb{R}^k_{\geq 0}$ (Lines 4–6) with updating the label weights w_i based on the discovered knowledge (Line 7). Our model f does not directly output discrete probabilities (e.g., via softmax) as a single probability mass function cannot reflect the degree of uncertainty involved in prediction-making, which we illustrate in Example 5.3.1. Instead, the model f outputs evidence supporting a particular class label, which parameterizes a Dirichlet distribution $\mathrm{Dir}(\alpha_i)$ with

$$\alpha_i = f(x_i; \theta) + 1 \in \mathbb{R}^k_{\geq 1}, \tag{5.1}$$

for $i \in [n]$. To fit f to the label weights w_i, we use a loss formulation similar to Sensoy et al. (2018). The loss regarding a fixed instance i is characterized by the expected value of the squared distance of w_i and $p_i \sim \mathrm{Dir}(\alpha_i)$ with α_i as in (5.1). For an instance $i \in [n]$ with features $x_i \in \mathcal{X}$ and label weights $\mathrm{w}_i \in \Delta^{\text{k-1}}$, the squared error using the bias-variance decomposition is

$$\begin{aligned}
\ell(f(x_i; \theta), \mathrm{w}_i) &= \mathbb{E}_{p_i \sim \mathrm{Dir}(\alpha_i)} \left\| \mathrm{w}_i - p_i \right\|_2^2 && (5.2)\\
&= \sum_{j=1}^{k} \mathbb{E}[(\mathrm{w}_{ij} - p_{ij})^2] \overset{(i)}{=} \sum_{j=1}^{k} \left[(\mathrm{w}_{ij} - \mathbb{E}[p_{ij}])^2 + \mathrm{Var}[p_{ij}] \right] \\
&\overset{(ii)}{=} \sum_{j=1}^{k} \Big[\underbrace{(\mathrm{w}_{ij} - \bar{p}_{ij})^2}_{=:\ell^{\text{err}}_{ij}} + \underbrace{\frac{\bar{p}_{ij}(1 - \bar{p}_{ij})}{1 + \|\alpha_i\|_1}}_{=:\ell^{\text{var}}_{ij}} \Big], && (5.3)
\end{aligned}$$

with $\bar{p}_i = \mathbb{E}_{p_i \sim \mathrm{Dir}(\alpha_i)}[p_i] = \alpha_i / \|\alpha_i\|_1$. (i) holds by expansion of the squared term and rearrangement and (ii) by the known variance of Dirichlet random variables. Fortunately, one does not need to numerically approximate the integral of the expected value in (5.2). One can directly compute ℓ using the outputs of f only. In the following, we give an example highlighting the differences to softmax normalization.

Example 5.3.1. Let $n = 2$, $k = 3$, $\mathcal{Y} = \{1, 2, 3\}$, $x_1, x_2 \in \mathcal{X}$, $f(x_1) = (4, 1, 1)$, and $f(x_2) = (7, 4, 4)$. Using softmax normalization, both predictions yield the same discrete probabilities, that is, $\mathrm{softmax}(f(x_1)) = \mathrm{softmax}(f(x_2)) \approx (0.910, 0.045, 0.045)$, although x_1 and x_2 have different activation values. In our setting, $\alpha_1 = f(x_1) + 1 = (5, 2, 2)$ and $\alpha_2 = f(x_2) + 1 = (8, 5, 5)$ by (5.1). The predicted probabilities are $\bar{p}_1 = \mathbb{E}_{p_1 \sim \mathrm{Dir}(\alpha_1)}[p_1] = (5/9, 2/9, 2/9)$

and $\bar{p}_2 = \mathbb{E}_{p_2 \sim \mathrm{Dir}(\alpha_2)}[p_2] = (4/9, 5/18, 5/18)$. While both probabilities are still close, similar to the softmax normalization, the different variances of the Dirichlet distributions represent different degrees of uncertainty, that is, $\mathrm{Var}_{p_1 \sim \mathrm{Dir}(\alpha_1)}[p_1] \approx (0.025, 0.017, 0.017)$ and $\mathrm{Var}_{p_2 \sim \mathrm{Dir}(\alpha_2)}[p_2] \approx (0.013, 0.011, 0.011)$. Since $f(x_2)$ has higher activation values, there is less associated uncertainty across all classes. Hence, $\mathrm{Dir}(\alpha_2)$ has less variance.

The loss term ℓ in (5.3) can be separated into an error and variance component, ℓ_{ij}^{err} and ℓ_{ij}^{var}, respectively. ℓ_{ij}^{err} enforces model fit and ℓ_{ij}^{var} acts as regularization term and incentivizes the decrease of the variance of the Dirichlet distribution parameterized by f. To prioritize model fit, it is desirable that $\ell_{ij}^{\mathrm{err}} > \ell_{ij}^{\mathrm{var}}$ if w_{ij} and $\bar{p}_{ij}$ deviate too much, which we discuss in the following.

Proposition 5.3.2. *Given instance $(x_i, s_i) \in \mathcal{D}$, parameters θ, label weights w_i, and $\bar{p}_i = \alpha_i / \|\alpha_i\|_1$, it holds that $\ell_{ij}^{\mathrm{err}} < \ell_{ij}^{\mathrm{var}}$ if and only if*

$$\bar{p}_{ij} - \sqrt{\ell_{ij}^{\mathrm{var}}} < \mathrm{w}_{ij} < \bar{p}_{ij} + \sqrt{\ell_{ij}^{\mathrm{var}}},$$

for all $i \in [n]$ and $j \in [k]$.

Proposition 5.3.2 sheds light on the magnitudes of ℓ_{ij}^{err} and ℓ_{ij}^{var}. When w_{ij} is within one standard deviation from $\bar{p}_{ij}$, w_{ij} is close to $\bar{p}_{ij}$. In this regime, reducing variance is more important than improving model fit, that is, $\ell_{ij}^{\mathrm{err}} < \ell_{ij}^{\mathrm{var}}$. Reducing the variance of the Dirichlet distribution is equivalent to a reduction of the uncertainty about the prediction. When w_{ij} is outside one standard deviation from $\bar{p}_{ij}$, model fit is more important, that is, $\ell_{ij}^{\mathrm{err}} > \ell_{ij}^{\mathrm{var}}$. This property guarantees that one can jointly learn the candidate label weights as well as their associated uncertainty seamlessly, which helps in creating robust representations to deal with out-of-distribution data and adversarial modifications of the instance features.

5.3.3. Regularization

Given an instance $x_i \in \mathcal{X}$, the correct label $y_i \in \mathcal{Y}$ is hidden within $s_i \subseteq \mathcal{Y}$ (Section 2). Therefore, our model f should not allocate any evidence to incorrect labels, that is, $f_j(x_i; \theta)$ should be zero for $i \in [n]$ and $j \notin s_i$. Similar to Sensoy et al. (2018), we add a regularization term to the risk computation to avoid evidence supporting incorrect labels $j \notin s_i$. Let $\tilde{\alpha}_{ij} = \alpha_{ij}$ if $j \notin s_i$, else $\tilde{\alpha}_{ij} = 1$, for $i \in [n]$, $j \in s_i$. We then achieve maximal uncertainty about predicting $j \notin s_i$ by considering the KL-divergence between $\mathrm{Dir}(\tilde{\alpha}_i)$ and $\mathrm{Dir}(1)$. We compute the empirical risk as

$$\hat{\mathrm{R}}(f; \lambda_t) = \frac{1}{n} \sum_{i=1}^{n} \left[\ell(f(x_i; \theta), \mathrm{w}_i) + \lambda_t D_{\mathrm{KL}}(\mathrm{Dir}(\tilde{\alpha}_i) \| \mathrm{Dir}(1)) \right]. \tag{5.4}$$

This has a positive effect on classification as f also learns from *negative* examples, that is, f should be maximally uncertain about predicting $j \notin s_i$. However, to avoid our model f

from being uncertain about all labels, we gradually increase the regularization coefficient λ_t. This regularization directly benefits the robustness of our method when dealing with out-of-distribution and adversarial data. Given two Dirichlet-distributed random variables with parameters $\mathrm{Dir}(\tilde{\alpha}_i)$ and $\mathrm{Dir}(1)$ respectively, their KL divergence permits a closed-form expression (Penny, 2001). One updates the parameters θ by backpropagation of (5.4). Note that the KL term also depends on the parameters θ via $\tilde{\alpha}_i$.

5.3.4. Updating the Label Weights

After updating the parameters θ, we extract the learned knowledge about the class labels to iteratively disambiguate the candidate sets s_i. For a fixed instance $(x_i, s_i) \in \mathcal{D}$ and model parameters θ, we want to find the optimal label weights $\mathrm{w}_i \in [0, 1]^k$ that minimize (5.3), while maintaining all prior information about the candidate set membership, that is, $\mathrm{w}_{ij} = 0$ if $j \notin s_i$. We cannot directly assign $\bar{p}_i$ to the label weights w_i since f can allocate evidence to incorrect labels, that is, $f_j(x_i; \theta) \geq 0$ for $j \notin s_i$. In Line 7 of Algorithm 2, we assign $\mathrm{w}_i \in [0, 1]^k$ to the solution of

$$\min_{\mathrm{w}'_i \in [0,1]^k} \ell(f(x_i; \theta), \mathrm{w}'_i) \quad \text{subject to} \quad \| \mathrm{w}'_i \|_1 = 1 \text{ and } \mathrm{w}'_{ij} = 0 \text{ if } j \notin s_i. \tag{5.5}$$

(5.5) permits a closed-form solution, which is as follows.

Proposition 5.3.3. *Given a fixed instance $(x_i, s_i) \in \mathcal{D}$, model parameters θ, and $\bar{p}_i = \alpha_i / \|\alpha_i\|_1$, the optimization problem* (5.5)*, with ℓ as in* (5.3)*, has the solution*

$$\mathrm{w}^*_{ij} = \begin{cases} \bar{p}_{ij} + \frac{1}{|s_i|}(1 - \sum_{j' \in s_i} \bar{p}_{ij'}) & \text{if } j \in s_i, \\ 0 & \text{else.} \end{cases}$$

The proof of Proposition 5.3.3 in Section 5.5.2 first shows that w^*_{ij} is a feasible solution for the constraints in (5.5) and then establishes optimality using the Lagrangian multiplier method since ℓ is continuous and differentiable. The solution w^*_{ij} uniformly re-distributes all weight of labels not in s_i to labels, which are in s_i. This guarantees a minimal loss. Notably, the update strategy in Proposition 5.3.3 differs from the heuristic ones proposed in related work (Lv et al., 2020; Xu et al., 2023; Tian et al., 2024).

5.3.5. Reinterpreting the Label Weights

Recall from Section 5.2 that subjective logic allows decomposing the projected probabilities $\bar{p}_i$ into a multinomial opinion ω_i with a belief and uncertainty term, that is, $\bar{p}_i = \mathfrak{b}_i + \mathfrak{a}_i \mathfrak{u}_i$. This representation directly allows quantifying the uncertainty involved in prediction-making. It is desirable that the label weight update w^*_i in Section 5.3.4 can also be written as such a multinomial opinion to allow for the direct quantification of belief and uncertainty.

Proposition 5.3.4. *Given a fixed instance $(x_i, s_i) \in \mathcal{D}$ and parameters θ, the solution w_{ij}^* of* (5.5)*, which is given by Proposition 5.3.3, is equivalent to $\mathrm{w}_i^* = \mathfrak{b}_i + \mathfrak{a}_i \mathfrak{u}_i$ with*

$$\mathfrak{b}_{ij} = \mathbb{1}_{\{j \in s_i\}} \frac{f_j(x_i;\theta)}{\|\alpha_i\|_1}, \text{ where } \alpha_i = f(x_i;\theta) + 1,\ \mathfrak{u}_i = 1 - \sum_{j \in s_i} \mathfrak{b}_{ij}, \text{ and } \mathfrak{a}_{ij} = \frac{\mathbb{1}_{\{j \in s_i\}}}{|s_i|}.$$

Proposition 5.3.4 allows reinterpreting w_i^* (Proposition 5.3.3) as such a multinomial opinion ω_i about instance i. Proposition 5.3.4 establishes that the belief of our prediction model in non-candidate labels directly contributes to the uncertainty $\mathfrak{u}_i$ in predicting instance $i \in [n]$. The uncertainty term $\mathfrak{u}_i$ arises from unallocated belief mass $\mathfrak{b}_{ij}$, that is, $1 - \sum_{j \in s_i} \mathfrak{b}_{ij}$ for $i \in [n]$. Also, the prior weights $\mathfrak{a}_{ij}$ are defined similarly to the initial label weights w_{ij} (Section 5.3.1). The prior weights are uniformly distributed among all candidate labels. The result in Proposition 5.3.4 establishes that our proposed update strategy (Proposition 5.3.3) is valid within subjective logic.

5.3.6. Bounding the Label Weights

This section examines how the model's probability outputs $\bar{p}_i$ and the label weights w_i interact with each other. In the following, we provide an upper bound of the change of w_i's values over time. As the label weights w_i are the prediction targets in (5.3), it is desirable that the w_i do not oscillate, which we detail in the following.

Proposition 5.3.5. *Let $(x_i, s_i) \in \mathcal{D}$, its label weights $\mathrm{w}_i^{(t)} \in [0,1]^k$, and the model's probability outputs $\bar{p}_i^{(t)}$ at epoch $t \in \mathbb{N}$. Then,*

$$0 \leq \| \mathrm{w}_i^{(t+1)} - \mathrm{w}_i^{(t)} \|_2^2 \leq \|\bar{p}_i^{(t+1)} - \bar{p}_i^{(t)}\|_2^2,$$

for $i \in [n]$.

This indicates that the label weights w_i change at most as fast as the model's probability outputs $\bar{p}_i$ between consecutive epochs. An immediate consequence is that the convergence of the model training and its probability outputs $\bar{p}_i$, that is, $\|\bar{p}_i^{(t+1)} - \bar{p}_i^{(t)}\|_2^2 \longrightarrow 0$ for $t \longrightarrow \infty$, implies the convergence of the label weight vectors w_i, that is, $\| \mathrm{w}_i^{(t+1)} - \mathrm{w}_i^{(t)} \|_2^2 \longrightarrow 0$, which are extracted from the model. This property is desirable as it shows that the label weights w_i do not oscillate if the model's probability outputs $\bar{p}_i$, which depend on the model parameters θ, converge.

5.3.7. Cross-Entropy Loss

Although we use the squared error loss (5.2), it is worth considering the commonly used cross-entropy loss. Given an instance $x_i \in \mathcal{X}$, a model f with parameters θ, and label weights w_i, a cross-entropy formalization similar to Sensoy et al. (2018) is given by

$$\ell_{\mathrm{CE}}(f(x_i;\theta), \mathrm{w}_i) = \mathbb{E}[-\sum_{j=1}^{k} \mathrm{w}_{ij} \log p_{ij}] \overset{(i)}{=} \mathrm{w}_i \cdot \Psi_i, \tag{5.6}$$

with $\Psi_{ij} = \psi(\|\alpha_i\|_1) - \psi(\alpha_{ij})$ and ψ denoting the digamma function. (i) holds because of the linearity of the expected value and $\mathbb{E}_{p_{ij} \sim \mathrm{Dir}_j(\alpha_i)} \log p_{ij} = \psi(\alpha_{ij}) - \psi(\|\alpha_i\|_1)$. In the following, we establish the optimal choice of w_i in our optimization problem (5.5) using the cross-entropy loss (5.6).

Proposition 5.3.6. *Given a fixed instance $(x_i, s_i) \in \mathcal{D}$ and parameters θ, optimization problem* (5.5)*, using the cross-entropy loss* (5.6)*, has the closed-form solution*

$$\mathrm{w}^*_{ij} = \begin{cases} 1 & \textit{if } j = \arg\min_{j' \in s_i} \Psi_{ij'}, \\ 0 & \textit{else.} \end{cases}$$

This suggests that the cross-entropy loss (5.6) enforces an aggressive label-weight update strategy setting all mass on one class label. Also, the label weights given by Proposition 5.3.6 cannot be reinterpreted in SL as discussed in Section 5.3.5. The squared error loss also performs better than the cross-entropy loss empirically. For these reasons, all experiments are conducted using the update strategy in Proposition 5.3.3.

5.3.8. Runtime Analysis

Recall from Section 5.3.2 that one does not need to numerically approximate the integral within the expectation value in (5.2). Given label weights w_i, one can directly compute ℓ using the outputs of f only. The computation of the KL divergence between two Dirichlet-distributed random variables with parameters $\mathrm{Dir}(\tilde{\alpha}_i)$ and $\mathrm{Dir}(1)$, respectively, admits a closed-form expression (Penny, 2001), leading to an overall linear runtime in n to compute $\hat{\mathrm{R}}(f; \lambda_t)$ (Algorithm 2, Line 5). In Line 7 of Algorithm 2, Proposition 5.3.3 also permits updating w_{ij} in linear time regarding n. Therefore, our method's runtime is dominated solely by the forward and backward pass of the employed model f.

5.4. Experiments

Section 5.4.1 summarizes all methods that we compare against and Section 5.4.2 outlines the experimental setup. Thereafter, Section 5.4.3 analyzes the methods' robustness against PLL noise, Section 5.4.4 against out-of-distribution samples, and Section 5.4.5 against adversarial perturbations.

5.4.1. Algorithms for Comparison

There are many PLL algorithms from which we pick the best-performing and commonly used ones for comparison. We cover classic algorithms and deep-learning techniques and complement these methods with strong baselines.

We consider 13 methods: PlKnn (Hüllermeier and Beringer, 2005), PlSvm (Nguyen and Caruana, 2008), Ipal (Zhang and Yu, 2015), PlEcoc (Zhang et al., 2017), Proden (Lv et al., 2020), Rc (Feng et al., 2020), Cc (Feng et al., 2020), Valen (Xu et al., 2021), Cavl (Zhang et al., 2022a), Pop (Xu et al., 2023), CroSel (Tian et al., 2024), DstPll (Fuchs et al., 2025), and RobustPll (our method).

Additionally, we benchmark various extensions known to obtain robust results in the supervised domain: Proden with L2-regularization (Proden+L2), Proden with dropout[1] (Proden+Dropout; Srivastava et al. 2014), Proden for disambiguating the partial labels and then training an evidential-deep-learning classifier (Sensoy et al., 2018) in a supervised manner (Proden+Edl), an ensemble of 5 Proden classifiers (Proden+Ens; Lakshminarayanan et al. 2017), an ensemble of 5 Proden classifiers trained on adversarial examples (Proden+AdvEns; Lakshminarayanan et al. 2017), and an ensemble of our method (RobustPll+Ens).

For a fair comparison, we use the same base model, that is, a d-300-300-300-k MLP (Werbos, 1974), which is a common choice in the literature (Lv et al., 2020; Feng et al., 2020; Xu et al., 2023), for all neural-network-based approaches. Appendix B.1 discusses the specific hyperparameter choices, including the neural network architectures, of all approaches in more detail. We publicly provide all code and data for reproducibility.

5.4.2. Experimental Setup

As is common in the literature (Zhang et al., 2017; Xu et al., 2023), we conduct experiments on supervised datasets with added noise as well as on real-world partially-labeled datasets. We use four supervised datasets with added noise and six real-world PLL datasets and refer to Appendix B.2 for the dataset characteristics. For the supervised datasets, we use *mnist* (LeCun et al., 1999), *kmnist* (Clanuwat et al., 2018), *fmnist* (Xiao et al., 2018), and *not-mnist* (Bulatov, 2011). For the real-world datasets, we use *bird-song* (Briggs et al., 2012), *lost* (Cour et al., 2011), *mir-flickr* (Huiskes and Lew, 2008), *msrc-v2* (Liu and Dietterich, 2012), *soccer* (Zeng et al., 2013), and *yahoo-news* (Guillaumin et al., 2010).

We use instance-dependent noise to introduce partial labels into the supervised datasets (Zhang et al., 2021). This strategy first trains a supervised classifier $g : \mathbb{R}^d \to \Delta^{\text{k-1}}$, which outputs probabilities $g_j(x_i)$ for instance $i \in [n]$ and class labels $j \in [k]$. Given an instance's features $x_i \in \mathcal{X}$ with correct label $y_i \in \mathcal{Y}$, a flipping probability of $\xi_j(x_i) = g_j(x_i)/\max_{j' \in \mathcal{Y} \setminus \{y_i\}} g_{j'}(x_i)$ determines whether to add the incorrect label $j \neq y_i$ to the candidate set s_i. Additionally, one divides $\xi_j(x_i)$ by the mean probability $\frac{1}{k-1} \sum_{j' \neq y_i} \xi_{j'}(x_i)$ of incorrect labels (Xu et al., 2021, 2023), which makes all labels more likely to appear. While all *mnist*-like datasets have ten class labels, the averages (± std.) of the candidate set cardinalities are 6.30 (± 0.06) for *mnist*, 5.95 (± 0.05) for *fmnist*, 6.34 (± 0.04) for *kmnist*, and 6.34 (± 0.09) for *not-mnist*. We remark that five out of the ten datasets do not contain a single instance with a candidate set that only consists of the ground truth label. This

[1] Dropout is also applied in testing to form an explicit ensemble.

Table 5.1.: Average test-set accuracies (± std.) on the *mnist*-like and real-world datasets. All experiments are repeated five times with different seeds to report mean and standard deviations. The *mnist*-like datasets have added instance-dependent noise as discussed in Section 5.4.2. The column for the real-world datasets contains averages across all six real-world datasets; the corresponding non-aggregated results are collected in Table B.2. We emphasize the best algorithm per dataset, as well as non-significant differences, using a student t-test with level $\alpha = 0.05$. We consider non-ensemble and ensemble methods separately. The triangles indicate our proposed methods.

All methods	*mnist*-like datasets with inst.-dep. noise				Real-world datasets
	mnist	*fmnist*	*kmnist*	*not-mnist*	
PlKnn (2005)	46.6 (± 0.5)	41.9 (± 0.4)	52.2 (± 0.4)	31.3 (± 0.9)	50.3 (± 8.8)
PlSvm (2008)	32.4 (± 5.0)	37.3 (± 1.9)	31.6 (± 4.2)	39.2 (± 3.9)	40.7 (± 10.1)
Ipal (2015)	**96.0** (± 0.4)	75.1 (± 0.7)	**80.8** (± 0.9)	61.5 (± 1.6)	57.8 (± 7.1)
PlEcoc (2017)	61.6 (± 2.9)	49.6 (± 4.5)	40.6 (± 2.7)	39.8 (± 6.1)	42.7 (± 19.8)
Proden (2020)	93.2 (± 0.5)	**77.8** (± 2.5)	76.6 (± 0.5)	84.6 (± 1.3)	**64.1** (± 7.4)
Proden+L2	93.3 (± 0.4)	**78.1** (± 1.7)	76.4 (± 0.6)	84.6 (± 1.2)	**64.1** (± 7.5)
Proden+Edl	92.0 (± 0.5)	74.9 (± 2.4)	74.5 (± 0.7)	80.8 (± 0.5)	49.6 (± 21.0)
Rc (2020)	93.0 (± 0.4)	**78.0** (± 2.3)	76.5 (± 0.7)	84.1 (± 1.6)	62.1 (± 8.9)
Cc (2020)	93.1 (± 0.2)	78.9 (± 0.9)	77.5 (± 0.8)	83.5 (± 0.9)	43.8 (± 31.2)
Valen (2021)	50.3 (± 5.3)	59.6 (± 1.9)	37.3 (± 1.3)	50.3 (± 2.4)	53.8 (± 9.0)
Cavl (2022)	79.5 (± 6.4)	72.9 (± 2.4)	64.6 (± 6.5)	61.5 (± 6.8)	61.4 (± 6.7)
Pop (2023)	92.5 (± 0.6)	**79.0** (± 1.6)	77.6 (± 0.2)	84.5 (± 1.8)	**64.1** (± 7.5)
CroSel (2024)	95.3 (± 0.1)	**79.6** (± 0.9)	79.6 (± 0.6)	**86.6** (± 0.7)	41.9 (± 30.1)
DstPll (2024)	62.2 (± 0.9)	50.3 (± 1.0)	68.4 (± 1.0)	38.2 (± 0.7)	48.5 (± 9.6)
► RobustPll	**96.0** (± 0.1)	**79.6** (± 3.0)	**81.7** (± 0.3)	**83.7** (± 1.9)	59.5 (± 6.8)
Proden+Dropout	92.5 (± 0.6)	72.7 (± 2.8)	72.1 (± 1.1)	78.0 (± 2.5)	65.0 (± 8.1)
Proden+Ens	93.7 (± 0.2)	78.0 (± 2.3)	77.3 (± 0.5)	**85.6** (± 0.7)	65.8 (± 8.3)
Proden+AdvEns	95.3 (± 0.6)	77.9 (± 2.3)	77.7 (± 0.9)	**84.3** (± 1.5)	**66.7** (± 9.0)
► RPll+Ens	**96.3** (± 0.1)	**80.4** (± 2.3)	**82.9** (± 0.5)	**85.9** (± 1.6)	63.6 (± 7.8)

prohibits the application of algorithms from related fields, for example, semi-supervised learning. We publicly provide all code and data for reproducibility.

5.4.3. Robustness under PLL Noise

Robust PLL algorithms should exhibit good predictive performance when confronted with PLL noise from ambiguous candidate sets. Table 5.1 shows the accuracies of all methods on the *mnist*-like and real-world datasets. All supervised datasets have added noise (Section 5.4.2). We repeat all experiments five times to report averages and standard deviations. The best algorithm per dataset, as well as algorithms with non-significant differences, are emphasized. Thereby, we consider non-ensemble methods (top) and

Table 5.2.: The difference in the predictive entropies on the test and OOD sets. The models have been trained on the *mnist* train dataset with added noise as discussed in Section 5.4.2. We report the area between the empirical CDFs, the KS statistic, and the maximum-mean discrepancy using the RBF kernel. A value of one is optimal. Also compare Figure B.1 for a graphical representation. Negative values indicate that the predictions on the out-of-distribution set are taken more confidently than the predictions on the test set.

All methods	Difference in entropy on MNIST and NotMNIST		
	CDF Area	KS stat.	MMD
PlKnn (2005)	0.0172	0.1587	0.0429
PlSvm (2008)	0.0114	0.2944	0.0296
Ipal (2015)	0.0896	0.3665	0.1285
PlEcoc (2017)	-0.0216	-0.3674	-0.0544
Proden (2020)	0.1769	0.6550	0.4240
Proden+L2	0.1853	0.6844	0.4410
Proden+Edl	**0.4379**	0.7171	**0.6714**
Rc (2020)	0.1402	0.5560	0.3495
Cc (2020)	0.0607	0.5587	0.1378
Valen (2021)	-0.7668	-0.9434	-1.2137
Cavl (2022)	0.0087	0.1555	0.0205
Pop (2023)	0.1345	0.5570	0.3361
CroSel (2024)	0.2278	**0.8360**	0.5202
DstPll (2024)	0.1723	0.5097	0.3243
► RobustPll	**0.3855**	0.7345	**0.6707**
Proden+Dropout	0.2541	0.7662	0.5700
Proden+Ens	0.2741	0.8559	0.6144
Proden+AdvEns	0.2017	0.6435	0.4506
► RobustPll+Ens	**0.5560**	**0.8866**	**0.9996**

ensemble methods (bottom) separately for fairness. We use a paired student t-test with level $\alpha = 0.05$ to test for significance.

Our method (RobustPll) performs best on the four *mnist*-like datasets and comparably on the real-world datasets. We observe a similar behavior regarding our ensemble method (RobustPll+Ens). Our non-ensemble method even achieves comparable performance to the ensemble methods on the *mnist*-like datasets. Summing up, we perform most consistently well under high PLL noise levels.

5.4.4. Out-of-distribution Robustness

Out-of-distribution examples (OOD) are instances that are not represented within the dataset. Since all methods output a discrete probability distribution over known class labels, we evaluate the entropy of the predicted probability outputs. Test-set instances should receive minimal predictive entropy, that is, the model is confident about one label, while the OOD examples should receive maximal predictive entropy, that is, no known class label matches the features. Robust algorithms should maximize the distance between the predictive entropies on the test and OOD sets. This is especially challenging in PLL as no exact ground truth is available.

Table 5.2 shows the differences in the normalized entropies (range 0 to 1) on the test and OOD samples for all methods. All methods are trained on the *mnist* train set, evaluated on the *mnist* test set, and evaluated on the *not-mnist* test set. Samples from the *not-mnist* test set contain letters instead of digits and are hence OOD examples. We measure the differences between the entropies on the test and OOD set using the area between the empirical CDFs, the value of the Kolmogorov-Smirnov statistic, and the maximum-mean discrepancy with RBF kernel using the median distance heuristic to set the kernel's parameter. We highlight the best (and close-to-best) values and consider non-ensemble and ensemble methods separately for fairness. Positive values indicate that test predictions are taken more confidently and negative values indicate that OOD predictions are taken more confidently.

Our methods (RobustPll and RobustPll+Ens) are among the best in almost all the three settings in Table 5.2. Some other methods even give negative values, which means that they are more sure about predicting the OOD than the test samples. Appendix B also contains further results. OOD examples mislead most of the state-of-the-art PLL methods into confidently predicting an incorrect label. In contrast, RobustPll+Ens achieves almost perfect differences indicating small predictive entropies on the test set (one class label receives most of the probability mass) and high predictive entropies on the OOD set (class probabilities are almost uniformly distributed).

5.4.5. Performance on Adversarial Examples

In recent years, many attacks on neural networks have been discussed in the literature (Szegedy et al., 2014; Goodfellow et al., 2015; Moosavi-Dezfooli et al., 2016; Madry et al., 2018). Using the *projected gradient descent* (PGD; Madry et al. 2018), we modify all test set examples, which are min-max-normalized to the range $[0, 1]$, by iteratively adding $\alpha := \varepsilon/10$ times $\operatorname{sign} \nabla_x f(x; \theta)$ to an instance's features $x \in \mathcal{X}$ and then projecting the newly obtained features back to an ε-ball around the original feature values x. We repeat those steps $T = 10$ times. The perturbed instances remain similar but moving against the gradient with respect to an instance's features decreases prediction performance rapidly.

Table 5.3 shows how all neural-network-based methods perform for varying values of the adversarial parameter $\varepsilon \in \{0.0, 0.1, 0.2, 0.3, 0.4\}$ on the real-world datasets. A value of $\varepsilon = 0.0$ indicates no added adversarial noise. The first column in Table 5.3 therefore matches the last column of Table 5.1. For values of $\varepsilon \geq 0.1$, RobustPll and Proden+Edl perform best among all non-ensemble techniques. Among the ensemble techniques, our method (RobustPll+Ens) performs best. In general, all ensembling techniques make Proden more robust against the corrupted features. Note that Proden+AdvEns has an unfair advantage in the analysis in Table 5.3 as it is trained on adversarial examples, that

Table 5.3.: Average test-set accuracies (± std.) on the real-world datasets. The instance features, which are min-max-normalized to the range $[0, 1]$, are corrupted using the projected gradient descent method (Madry et al., 2018). As this attack applies to neural networks, we report only the performances of the deep learning PLL methods. Note that the column with $\varepsilon = 0.0$ is identical to the last column of Table 5.1. The triangles indicate our proposed methods.

Deep-learning methods	Corrupted real-world datasets with adversarial parameter ε				
	$\varepsilon = 0.0$	$\varepsilon = 0.1$	$\varepsilon = 0.2$	$\varepsilon = 0.3$	$\varepsilon = 0.4$
Proden (2020)	**64.1** (± 7.4)	28.9 (± 7.1)	22.6 (± 5.7)	18.8 (± 5.2)	17.2 (± 5.9)
Proden+L2	**64.1** (± 7.5)	28.4 (± 7.6)	22.5 (± 6.1)	19.2 (± 5.3)	17.2 (± 5.3)
Proden+Edl	49.6 (± 21.0)	**36.2** (± 14.8)	**32.0** (± 14.0)	**29.0** (± 13.8)	**27.1** (± 13.0)
Rc (2020)	62.1 (± 8.9)	29.0 (± 6.5)	21.3 (± 6.4)	17.9 (± 6.0)	14.7 (± 5.3)
Cc (2020)	43.8 (± 31.2)	20.2 (± 14.9)	14.6 (± 11.2)	11.8 (± 9.1)	9.8 (± 7.7)
Valen (2021)	53.8 (± 9.0)	25.4 (± 7.8)	19.6 (± 6.9)	17.2 (± 6.5)	15.4 (± 6.5)
Cavl (2022)	61.4 (± 6.7)	25.8 (± 7.2)	19.3 (± 5.7)	16.4 (± 5.2)	13.9 (± 4.8)
Pop (2023)	**64.1** (± 7.5)	28.6 (± 7.1)	22.3 (± 6.3)	18.8 (± 5.0)	16.7 (± 4.9)
CroSel (2024)	41.9 (± 30.1)	22.6 (± 16.8)	16.3 (± 12.6)	13.4 (± 10.3)	11.5 (± 8.6)
► RobustPll	59.5 (± 6.8)	**40.3** (± 12.0)	**31.8** (± 10.3)	**27.4** (± 9.3)	23.8 (± 8.4)
Prod.+Dropout	65.0 (± 8.1)	30.7 (± 6.1)	23.4 (± 4.8)	19.8 (± 5.0)	17.9 (± 5.4)
Prod.+Ens	65.8 (± 8.3)	42.8 (± 6.9)	33.1 (± 8.4)	27.4 (± 8.7)	24.7 (± 10.3)
Prod.+AdvEns	**66.7** (± 9.0)	**48.7** (± 7.0)	37.1 (± 7.6)	30.5 (± 8.7)	26.6 (± 9.9)
► RPll+Ens	63.6 (± 7.8)	**51.0** (± 10.3)	**42.0** (± 10.2)	**37.0** (± 9.4)	**33.3** (± 9.1)

is, it has access to the corrupted features during training. Nevertheless, our ensemble method RobustPll+Ens is significantly better for $\varepsilon \geq 0.1$. RobustPll and RobustPll+Ens consistently perform among the best for $\varepsilon \geq 0.1$.

In summary, our non-ensemble and ensemble methods consistently perform the best across almost all settings considered. Our methods are robust against high PLL noise, out-of-distribution examples, and adversarial perturbations.

5.5. Proofs

This section collects all proofs of the propositions in the main text. The proof of Proposition 5.3.2 is in Section 5.5.1, that of Proposition 5.3.3 is in Section 5.5.2, that of Proposition 5.3.4 is in Section 5.5.3, and that of Proposition 5.3.5 is in Section 5.5.4.

5.5.1. Proof of Proposition 5.3.2

Solving $\ell_{ij}^{\text{err}} = \ell_{ij}^{\text{var}}$ for the label weights w_{ij} yields

$$\begin{aligned}\ell_{ij}^{\text{err}} = \ell_{ij}^{\text{var}} &\Leftrightarrow (\mathrm{w}_{ij} - \bar{p}_{ij})^2 = \frac{\bar{p}_{ij}(1-\bar{p}_{ij})}{1+\|\alpha_i\|_1} \\ &\Leftrightarrow \mathrm{w}_{ij} = \bar{p}_{ij} \pm \sqrt{\frac{\bar{p}_{ij}(1-\bar{p}_{ij})}{1+\|\alpha_i\|_1}} \\ &\Leftrightarrow \mathrm{w}_{ij} = \bar{p}_{ij} \pm \sqrt{\ell_{ij}^{\text{var}}}.\end{aligned}$$

Since $\ell_{ij}^{\text{err}} = (\mathrm{w}_{ij} - \bar{p}_{ij})^2$ reaches its minimum when $\mathrm{w}_{ij} = \bar{p}_{ij}$, we have shown the statement to be demonstrated.

5.5.2. Proof of Proposition 5.3.3

The proof first shows that w_i^* is a feasible solution for the constraints in (5.5) and then establishes that w_i^* is indeed optimal using the Lagrangian multiplier method.

(Primal) Feasibility. To prove our solution's feasibility, we need to show that (i) $\| \mathrm{w}_i^* \|_1 = 1$ and (ii) $\mathrm{w}_{ij}^* = 0$ for all $j \notin s_i$. Constraint (i) holds as

$$\begin{aligned}\sum_{j=1}^{k} \mathrm{w}_{ij}^* &= \sum_{j\in s_i} \Big(\bar{p}_{ij} + \frac{1}{|s_i|}(1 - \sum_{j'\in s_i} \bar{p}_{ij'})\Big) \\ &= \sum_{j\in s_i} \bar{p}_{ij} + \frac{1}{|s_i|} \sum_{j\in s_i}(1 - \sum_{j'\in s_i} \bar{p}_{ij'}) \\ &= \sum_{j\in s_i} \bar{p}_{ij} + 1 - \sum_{j'\in s_i} \bar{p}_{ij'} = 1.\end{aligned}$$

Constraint (ii) follows directly from the definition of w_{ij}^* in Proposition 5.3.3.

Optimality. Since the loss ℓ is differentiable, continuous, and convex in w_i, we incorporate the constraints (i) and (ii) using the Lagrangian multiplier method as follows:

$$\mathrm{L}(\mathrm{w}_i; \lambda_i) = \sum_{j=1}^{k} (\mathrm{w}_{ij} - \bar{p}_{ij})^2 + \lambda_i \Big(\sum_{j=1}^{k} \mathrm{w}_{ij} - 1 \Big),$$

for instance $i \in [n]$. Constraint (ii) directly determines the value of w_{ij} for all $j \notin s_i$. We then need to check the following Lagrange conditions:

$$\begin{aligned}\frac{\partial}{\partial \mathrm{w}_{ij}} \mathrm{L}(\mathrm{w}_i; \lambda_i) = 0 &\Leftrightarrow 2(\mathrm{w}_{ij} - \bar{p}_{ij}) + \lambda_i = 0 \\ &\Leftrightarrow \mathrm{w}_{ij} = \bar{p}_{ij} - \frac{\lambda_i}{2},\end{aligned} \tag{5.7}$$

for $i \in [n]$ and $j \in s_i$. For $j \notin s_i$, $\mathrm{w}_{ij} = 0$. Inserting (5.7) into constraint (i) yields

$$\begin{aligned} \| \mathrm{w}_i \|_1 = 1 &\Leftrightarrow \sum_{j=1}^{k} \mathrm{w}_{ij} = \sum_{j \in s_i} (\bar{p}_{ij} - \frac{\lambda_i}{2}) = \sum_{j \in s_i} \bar{p}_{ij} - \frac{|s_i| \lambda_i}{2} = 1 \\ &\Leftrightarrow \lambda_i = \frac{2}{|s_i|} \Big(\sum_{j \in s_i} \bar{p}_{ij} - 1 \Big). \end{aligned} \tag{5.8}$$

Putting (5.8) back into (5.7) gives us

$$\mathrm{w}^*_{ij} = \begin{cases} \bar{p}_{ij} + \frac{1}{|s_i|}(1 - \sum_{j' \in s_i} \bar{p}_{ij'}) & \text{if } j \in s_i, \\ 0 & \text{else,} \end{cases}$$

which is the optimal solution to (5.5). Note that we do not need to show dual feasibility and complementary slackness as there are no inequality constraints.

5.5.3. Proof of Proposition 5.3.4

We prove the statement by distinguishing two cases. (a) If $i \in [n]$ and $j \notin s_i$, $\mathrm{w}^*_i = \mathfrak{b}_i + \mathfrak{a}_i \mathfrak{u}_i$ is true as both sides are zero. (b) If $i \in [n]$ and $j \in s_i$, it follows

$$\begin{aligned} \mathrm{w}^*_{ij} &\overset{(i)}{=} \bar{p}_{ij} + \frac{1}{|s_i|}(1 - \sum_{j' \in s_i} \bar{p}_{ij'}) \\ &\overset{(ii)}{=} \frac{\alpha_{ij}}{\|\alpha_i\|_1} + \frac{1}{|s_i|}\Big(1 - \sum_{j' \in s_i} \frac{\alpha_{ij'}}{\|\alpha_i\|_1}\Big) \\ &\overset{(iii)}{=} \frac{f_j(x_i;\theta) + 1}{\|\alpha_i\|_1} + \frac{1}{|s_i|}\Big(1 - \sum_{j' \in s_i} \frac{f_{j'}(x_i;\theta) + 1}{\|\alpha_i\|_1}\Big) \\ &\overset{(iv)}{=} \frac{f_j(x_i;\theta)}{\|\alpha_i\|_1} + \frac{1}{\|\alpha_i\|_1} + \frac{1}{|s_i|}\Big(1 - \frac{|s_i|}{\|\alpha_i\|_1} - \sum_{j' \in s_i} \frac{f_{j'}(x_i;\theta)}{\|\alpha_i\|_1}\Big) \\ &\overset{(v)}{=} \underbrace{\frac{f_j(x_i;\theta)}{\|\alpha_i\|_1}}_{=\mathfrak{b}_{ij}} + \underbrace{\frac{1}{|s_i|}}_{=\mathfrak{a}_{ij}} \underbrace{\Big(1 - \sum_{j' \in s_i} \frac{f_{j'}(x_i;\theta)}{\|\alpha_i\|_1}\Big)}_{=\mathfrak{u}_i} \\ &\overset{(vi)}{=} \mathfrak{b}_{ij} + \mathfrak{a}_{ij} \mathfrak{u}_i, \end{aligned}$$

where (i) holds by Proposition 5.3.3, (ii) by $\bar{p}_i = \alpha_i / \|\alpha_i\|_1$, (iii) by $\alpha_i = f(x_i;\theta) + 1$, (iv) by separating summands, (v) by simplifying, and (vi) by the definitions in Proposition 5.3.4. Note that we add the factor $\mathbb{1}_{\{j \in s_i\}}$ to $\mathfrak{b}_{ij}$ and $\mathfrak{a}_{ij}$ to combine both cases, that is, (a) $j \notin s_i$ and (b) $j \in s_i$, into a single formula.

5.5.4. Proof of Proposition 5.3.5

The proof of Proposition 5.3.5 proceeds as follows:

$$\begin{aligned}
0 &\leq \| \mathrm{w}_i^{(t+1)} - \mathrm{w}_i^{(t)} \|_2^2 \\
&\overset{(i)}{=} \sum_{j \in s_i} \Big(\bar{p}_{ij}^{(t+1)} + \frac{1}{|s_i|}\big(1 - \sum_{j' \in s_i} \bar{p}_{ij'}^{(t+1)}\big) - \bar{p}_{ij}^{(t)} - \frac{1}{|s_i|}\big(1 - \sum_{j' \in s_i} \bar{p}_{ij'}^{(t)}\big)\Big)^2 \\
&\overset{(ii)}{=} \sum_{j \in s_i} \Big(\big(\bar{p}_{ij}^{(t+1)} - \bar{p}_{ij}^{(t)}\big) + \frac{1}{|s_i|} \sum_{j' \in s_i} \big(\bar{p}_{ij'}^{(t)} - \bar{p}_{ij'}^{(t+1)}\big)\Big)^2 \\
&\overset{(iii)}{=} \sum_{j \in s_i} \big(\bar{p}_{ij}^{(t+1)} - \bar{p}_{ij}^{(t)}\big)^2 + \frac{2}{|s_i|} \sum_{j \in s_i} \big(\bar{p}_{ij}^{(t+1)} - \bar{p}_{ij}^{(t)}\big) \sum_{j' \in s_i} \big(\bar{p}_{ij'}^{(t)} - \bar{p}_{ij'}^{(t+1)}\big) \\
&\qquad + \frac{1}{|s_i|^2} \sum_{j \in s_i} \Big(\sum_{j' \in s_i} \big(\bar{p}_{ij'}^{(t)} - \bar{p}_{ij'}^{(t+1)}\big)\Big)^2 \\
&\overset{(iv)}{=} \sum_{j \in s_i} \big(\bar{p}_{ij}^{(t+1)} - \bar{p}_{ij}^{(t)}\big)^2 - \frac{2}{|s_i|} \Big(\sum_{j \in s_i} \big(\bar{p}_{ij}^{(t+1)} - \bar{p}_{ij}^{(t)}\big)\Big)^2 + \frac{1}{|s_i|} \Big(\sum_{j' \in s_i} \big(\bar{p}_{ij'}^{(t)} - \bar{p}_{ij'}^{(t+1)}\big)\Big)^2 \\
&\overset{(v)}{=} \sum_{j \in s_i} \big(\bar{p}_{ij}^{(t+1)} - \bar{p}_{ij}^{(t)}\big)^2 - \frac{1}{|s_i|} \Big(\sum_{j \in s_i} \big(\bar{p}_{ij}^{(t+1)} - \bar{p}_{ij}^{(t)}\big)\Big)^2 \\
&\overset{(vi)}{\leq} \sum_{j \in s_i} \big(\bar{p}_{ij}^{(t+1)} - \bar{p}_{ij}^{(t)}\big)^2 \\
&\overset{(vii)}{\leq} \sum_{j=1}^{k} \big(\bar{p}_{ij}^{(t+1)} - \bar{p}_{ij}^{(t)}\big)^2 \\
&= \|\bar{p}_i^{(t+1)} - \bar{p}_i^{(t)}\|_2^2,
\end{aligned}$$

where (i) holds by Prop. 5.3.3, (ii) by reordering, (iii) by the binomial theorem, (iv) by

$$\sum_{j' \in s_i} \big(\bar{p}_{ij'}^{(t)} - \bar{p}_{ij'}^{(t+1)}\big) = - \sum_{j' \in s_i} \big(\bar{p}_{ij'}^{(t+1)} - \bar{p}_{ij'}^{(t)}\big),$$

and (v) by

$$\Big(\sum_{j \in s_i} \big(\bar{p}_{ij}^{(t)} - \bar{p}_{ij}^{(t+1)}\big)\Big)^2 = \Big(- \sum_{j \in s_i} \big(\bar{p}_{ij}^{(t+1)} - \bar{p}_{ij}^{(t)}\big)\Big)^2 = \Big(\sum_{j \in s_i} \big(\bar{p}_{ij}^{(t+1)} - \bar{p}_{ij}^{(t)}\big)\Big)^2.$$

(vi) holds as $|s_i| \geq 1$ and $\sum_{j=1}^{k} \big(\bar{p}_{ij}^{(t+1)} - \bar{p}_{ij}^{(t)}\big)^2 \geq 0$ and (vii) holds by including further non-negative summands within the summation.

5.5.5. Proof of Proposition 5.3.6

The proof first shows that w_{ij}^* is a feasible solution for the constraints in (5.5) and then establishes that w_{ij}^* is optimal.

Feasibility. As w_{ij}^{*} is one for exactly one class label $j \in s_i$, it holds that $\| \mathrm{w}_i \|_1 = 1$ for $i \in [n]$. Also, $\mathrm{w}_{ij}^{*} = 0$ for $j \notin s_i$ as only class labels $j \in s_i$ can be different from zero.

Optimality. As the cross-entropy loss ℓ_{CE} in Section 5.3.7 is a linear combination of w_{ij} with coefficients $\Psi_{ij} = \psi(\|\alpha_i\|_1) - \psi(\alpha_{ij})$ for fixed $i \in [n]$ and $\mathrm{w}_{ij} \in [0, 1]$, we minimize ℓ_{CE} by assigning all label weight to the minimal coefficient Ψ_{ij}, that is, $\arg\min_{j \in s_i} \Psi_{ij}$.

6. Partial-Label Learning with Conformal Candidate Cleaning

This chapter's contents are based on the following publication.

- Tobias Fuchs and Florian Kalinke. Partial-label learning with conformal candidate cleaning. In *Uncertainty in Artificial Intelligence*, pages 1337–1357, 2025. `https://proceedings.mlr.press/v286/fuchs25a.html`.

All code and data for reproducing this chapter's experiments are available at `https://github.com/mathefuchs/pll-with-conformal-candidate-cleaning`.

6.1. Overview

Many algorithms targeting the PLL problem exist. Recently, several extensions (Bao et al., 2021, 2022; Wang and Zhang, 2022; Zhang et al., 2022b; Xu et al., 2023) that can be combined with a wide range of PLL methods have been proposed, which aim at further improving their predictive performance. Typically, different PLL classifiers perform best on different datasets. In this sense, having extensions that are applicable to a multitude of different PLL algorithms is extremely beneficial. These extensions include feature selection and candidate cleaning techniques, which clean the instance space and candidate label space, respectively. However, many of these extensions depend on heuristics.

In contrast, this article proposes a novel method that alternates between training a PLL classifier through empirical risk minimization and pruning the candidate sets using conformal prediction, which output sets of possible labels that contain the correct label with a specified confidence level (Lei, 2014; Sadinle et al., 2019). In our pruning step, we remove candidate labels if they are not part of these predicted conformal sets. This principled way of reducing the candidate set ambiguity benefits the training of the PLL classifier when compared to the existing heuristic thresholds. Our extension significantly improves the prediction quality of several state-of-the-art PLL methods across a variety of datasets and experimental settings. To guarantee the validity of the conformal classifier used in the pruning step, one usually requires a labeled validation set for the calibration of the coverage guarantee. In the PLL setting, however, ground truth is unavailable. To resolve this serious issue, we propose a strategy that trains a PLL classifier, uses its predictions to label the validation set, calibrates the conformal sets with the validation set, and prunes candidate labels that are not part of these conformal sets. We show that our method preserves the conformal validity with respect to the unknown ground truth.

Contributions. Our contributions can be summarized as follows.

- *Algorithm.* We propose a novel candidate cleaning method that alternates between training a PLL classifier and pruning the PLL candidate sets. Our algorithm significantly improves the predictive performance of several state-of-the-art PLL approaches.
- *Experiments.* Extensive experiments on artificial and real-world partially labeled data support our claims. An ablation study further demonstrates the usefulness of the proposed strategy. Our code and data are openly available.
- *Theoretical analysis.* We analyze our method and show that the pruning step yields valid conformal sets.

Outline. Section 6.2 contains relevant background information on conformal prediction, Section 6.3 details our contributions, and Section 6.4 shows our experimental setup and results. All proofs are in Section 6.5. Appendix C.1 and C.2 contain supplementary material for our proofs, Appendix C.3 lists all hyperparameters used within our experiments in detail, and Appendix C.4 contains additional experiments.

6.2. Background

Recent methods in supervised multi-class classification (Lei, 2014; Barber et al., 2023; Mozannar et al., 2023; Mao et al., 2024; Narasimhan et al., 2024) explore training set-valued predictors $C : \mathcal{X} \to 2^{\mathcal{Y}}$ rather than single-label classifiers as they offer more flexibility in representing the uncertainty involved in prediction-making.

In conformal prediction, classifiers output sets of class labels $C(x) \subseteq \mathcal{Y}$. *Valid* conformal predictors guarantee that

$$\mathbb{P}_{XY}(Y \in C(X)) \geq 1 - \alpha, \tag{6.1}$$

which means that the correct label is part of a conformal set with a given error level of at most $\alpha \in (0, 1)$. The conformal predictor C that outputs $C(x) = \mathcal{Y}$, for $x \in \mathcal{X}$, is trivially valid as it covers the correct label with a probability of one. To avoid this case, one searches for conformal predictors C with minimal expected cardinality $\mathbb{E}_X \, |C(X)|$, while still being valid. In the supervised setting, this is captured by the following optimization problem (Sadinle et al., 2019):

$$\min_{C:\mathcal{X}\to 2^{\mathcal{Y}}} \mathbb{E}_X \, |C(X)| \qquad \text{subject to} \qquad \mathbb{P}_{XY}(Y \in C(X)) \geq 1 - \alpha. \tag{6.2}$$

Optimal solutions to (6.2) are of the form $C(x) = \{y \in \mathcal{Y} : \mathbb{P}_{Y|X=x}(Y = y) \geq t_\alpha\}$, for $x \in \mathcal{X}$, where t_α is set to

$$t_\alpha = \sup \left\{ t \in [0, 1] : \mathbb{P}_{XY}\left[(x, y) : \mathbb{P}_{Y|X=x}(Y = y) \geq t \right] \geq 1 - \alpha \right\}, \tag{6.3}$$

where we assume that the quantile function of $\mathbb{P}_{Y|X=x}(Y = y)$ is continuous at t_α.[1] In practice, one approximates t_α by computing the empirical distribution function on a hold-out validation set. One splits the dataset $\mathcal{D}$ into a dataset $\mathcal{D}_t$ for model training and $\mathcal{D}_v$ for calibrating the conformal predictor C with respect to the confidence level α. The validation set $\mathcal{D}_v$ is assumed to be exchangeable with respect to the joint distribution $\mathbb{P}_{XY}$.

Conformal prediction is also a natural fit to PLL as both deal with sets of class labels. Javanmardi et al. (2023) examine different ways of achieving valid conformal sets in the PLL context. However, they do not propose any new PLL method against which we can compare. Rather, they analyze the properties of different non-conformity measures in this context. In contrast, our focus is on constructing and evaluating new PLL methods.

In the subsequent section, we propose a novel candidate cleaning method that is based on conformal prediction and adapts (6.2) to the PLL setting to yield valid conformal sets. The optimization problem (6.2) cannot directly be transferred to the PLL context as ground truth for the calibration of the validity property is unavailable. We propose a strategy that uses the PLL classifier to label the validation set and then leverages these pseudo-labels for calibration. We show that this preserves the validity with respect to the ground truth.

6.3. PLL with Conformal Candidate Cleaning

We propose a novel candidate cleaning strategy that iteratively cleans the candidate sets of the PLL dataset $\mathcal{D}$ by reducing the candidate set cardinalities. Our method alternates between training a PLL classifier through empirical risk minimization and pruning the candidate sets based on conformal prediction. Conformal predictors $C : \mathcal{X} \to 2^{\mathcal{Y}}$ cover the correct label y_i of instance x_i with a specified probability; see (6.1). This coverage property is calibrated using a separate validation set of exchangeable PLL data points that are labeled using the trained PLL algorithm. As the classifier can give wrong predictions, however, we propose a novel correction strategy that accounts for possible misclassifications when calibrating the coverage of the correct labels against the validation set, which maintains the validity guarantee. We remove class labels from the candidate sets s_i if they are not part of the predicted conformal set $C(x_i)$ since the correct label y_i is in $C(x_i)$ with a specified confidence level.

This procedure iteratively removes noise from incorrect candidate labels, which benefits the training of the PLL classifier by having to account for less and less noise in each training step. Many PLL algorithms (Lv et al., 2020; Xu et al., 2023; Tian et al., 2024) proceed in a similar manner. They have in common that they alternate between training a PLL classifier and using its predictions to refine the candidate label weights. This can equivalently be expressed from an expectation-maximization perspective (Wang et al., 2022, Section 5). These label propagation strategies are state-of-the-art in many weakly supervised learning domains. In contrast to the existing heuristic update rules, however,

[1] See Sadinle et al. (2019, Theorem 1) for the general case.

our proposed method provides a principled way of iteratively cleaning the candidate sets using conformal predictors C.

In the following sections, we discuss our method in detail. Section 6.3.1 elaborates on the notion of conformal validity in the PLL context, Section 6.3.2 details how to correct for the ambiguity in PLL compared to the supervised setting, Section 6.3.3 outlines the proposed algorithm, Section 6.3.4 discusses the method's runtime complexity, and Section 6.3.5 discusses the placement of our method with respect to related work.

6.3.1. PLL Validity

Since we use the conformal predictions $C(x_i)$ to clean the associated candidate sets s_i, for $(x_i, s_i) \in \mathcal{D}$, we require that $s_i \cap C(x_i)$ is nonempty with a specified confidence level as otherwise $C(x_i)$ does not contain the unknown correct label y_i. Hence, we adapt (6.1) to our setting and consider a conformal classifier C valid with respect to the PLL candidate sets if it holds that

$$\mathbb{P}_{XS}(S \cap C(X) \neq \emptyset) \geq 1 - \alpha, \tag{6.4}$$

for a given error level $\alpha \in (0, 1)$. In other words, conformal predictions $C(x_i)$ need to cover the observed ambiguously labeled candidate sets s_i with a specified probability. Recall that $C(x) = \mathcal{Y}$, for $x \in \mathcal{X}$, trivially satisfies (6.4). One therefore also wants to minimize the cardinalities $|C(x)|$. Given the standard PLL assumption that the correct label y_i is within the respective candidate set s_i, which implies that $\mathbb{P}_{S|X=x,Y=y}(y \in S) = 1$ for any $(x, y) \in \mathcal{X} \times \mathcal{Y}$, an optimal solution to (6.2) is also valid in the sense of (6.4). Theorem 6.3.1 captures this relationship and underpins our proposed cleaning method, which we detail in Section 6.3.3.

Theorem 6.3.1. *Assume that $\mathbb{P}_{S|X=x,Y=y}(y \in S) = 1$, for any $(x, y) \in \mathcal{X} \times \mathcal{Y}$, and $\alpha \in (0, 1)$. Then, an optimal solution C of* (6.2) *satisfies* (6.4)*: $\mathbb{P}_{XS}(S \cap C(X) \neq \emptyset) \geq 1 - \alpha$.*

6.3.2. Correcting for Misclassification

Recall that, in the PLL setting, the ground-truth labels y are unavailable during training, which hinders the approximation of (6.3) needed for the solution of (6.2). Because a solution to (6.2) is, however, also desirable in the PLL setting (Theorem 6.3.1), we make use of existing PLL algorithms to generate pseudo-labels. This strategy iteratively learns a prediction model $f : \mathcal{X} \to [0, 1]^k$ that minimizes the empirical risk in (2.5). We use the trained model f to predict the labels on the validation set $\mathcal{D}_v$, which in turn is used for the calibration of the validity guarantee. Notably, this strategy results in a valid conformal predictor (Theorem 6.3.4). We note that it remains open to establish the minimality of the resulting conformal sets (analogous to solutions of (6.2)).

At first glance, it might be counter-intuitive to use the trained model f to label the validation set and build conformal sets based on it. However, we want to recall that the used base PLL classifier is risk consistent (Feng et al., 2020, Theorem 4). With this result and additional mild assumptions, we can prove that the PLL classifier's predictions cannot be arbitrarily bad (Lemma 6.3.3) and, leveraging this, that our conformal predictor is valid for some adapted threshold and error level (Theorem 6.3.4).

One of our central assumptions is a Bernstein condition (Audibert, 2004; Bartlett and Mendelson, 2006; Grünwald and Mehta, 2020) on the loss difference (Assumption 6.3.2). The Bernstein condition is defined as follows.

Assumption 6.3.2 (Bernstein Condition)**.** Let $B > 0$, $\beta \in (0, 1]$, $f^* = \arg\min_{f \in \mathcal{H}} R(f)$ the true risk minimizer, and $\ell : [0, 1]^k \times \mathcal{Y} \to \mathbb{R}_{\geq 0}$ a loss function. We assume that the excess loss $L_f(x, y) := \ell(f(x), y) - \ell(f^*(x), y)$ satisfies the (β, B)-Bernstein condition, that is, for $f \in \mathcal{H}$,

$$\mathbb{E}_{XY}\left[L_f(X,Y)^2\right] \leq B\left(\mathbb{E}_{XY}\left[L_f(X,Y)\right]\right)^{\beta}.$$

Assumption 6.3.2 is frequently made in ERM as it allows controlling the variance of the resulting losses, since $\operatorname{Var}_{XY}[\ell(f(X), Y) - \ell(f^*(X), Y)] \leq \mathbb{E}_{XY}[(\ell(f(X), Y) - \ell(f^*(X), Y))^2]$. In other words, the tail of the distribution of the excess loss must be well-behaved.

Building upon Assumption 6.3.2, we prove the results in the following Lemma 6.3.3, which are the main building blocks underlying the proof of our main result.

Lemma 6.3.3. *Let $\hat{f} = \arg\min_{f \in \mathcal{H}} \hat{R}(f)$ the empirical risk minimizer, $f^* = \arg\min_{f \in \mathcal{H}} R(f)$ the true risk minimizer, $\hat{y}_x = \arg\max_y \hat{f}_y(x)$, $y^*_x = \arg\max_y f^*_y(x)$, and Assumption 6.3.2 hold for the excess loss $L_{\hat{f}}$.*

(i) Then, for any $\delta_1 \in (0, 1)$ and some constant $M_1 > 0$, it holds, with $\mathbb{P}_n$-probability at least $1 - \delta_1$, that

$$\mathbb{E}_{XY}\left[|\hat{f}_Y(X) - f^*_Y(X)|\right] \leq M_1 \left(\frac{\log(1/\delta_1)}{n}\right)^{\frac{1}{4}\beta},$$

assuming that $\ell : (0, 1]^k \times \mathcal{Y} \to \mathbb{R}_{\geq 0}$, $(p, y) \mapsto -\log p_y$ is the log-loss.

(ii) Also, for any $\delta_2 \in (0, 1)$ and some constant $M_2 > 0$, it holds, with $\mathbb{P}_n$-probability at least $1 - \delta_2$, that

$$\mathbb{P}_X\left[\hat{y}_X \neq y^*_X\right] \leq M_2 \left(\frac{\log(1/\delta_2)}{n}\right)^{\frac{1}{2}\beta},$$

given that, for any $x \in \mathcal{X}$ and some constant $\delta_5 \in [0, 1)$, $\mathbb{P}_{Y|X=x}\left(Y \in \{\hat{y}_x, y^*_x\}\right) \geq 1 - \delta_5$.

Intuitively, Lemma 6.3.3 (i) and (ii) state that, under mild assumptions, a consistent PLL classifier cannot, in expectation, provide arbitrarily bad predictions. More precisely, Lemma 6.3.3 (i) states that the expected absolute difference in the probabilistic predictions of the empirical and true risk minimizer are upper-bounded. Lemma 6.3.3 (ii) states that the probability of class label predictions of the empirical and true risk minimizer not matching is upper-bounded. Note that, for $n \to \infty$, both upper-bounds tend to zero.

In the following, we comment on the assumptions made. Lemma 6.3.3 (i) requires the loss function to be the log-loss as it is a local proper loss function (Gneiting and Raftery, 2007), that is, ℓ is a proper loss function that only uses the y-th entry of the vector p in the computation of $\ell(p, y)$, which we use in our proof. Lemma 6.3.3 (ii) requires that the correct label y_x^* and pseudo-label $\hat{y}_x$ have some lower-bound for their conditional probability mass. Intuitively, the assumption captures that the true class posterior of the correct label y_x^* must have a probability mass that is not arbitrarily close to zero.

Based on the upper bounds in Lemma 6.3.3, one can adapt the threshold and confidence levels in (6.2) and (6.3) such that the conformal guarantee is still valid when using the pseudo-labels on the validation set. Theorem 6.3.4 states this result.

Theorem 6.3.4. *Assume the setting of Lemma 6.3.3* (i) *and* (ii) *and, for any* $\delta_6 \in (0,1)$, $\mathbb{P}_{Y|X=x}(Y = y_X^*) \geq 1 - \delta_6$ *with* $y_X^* = \arg\max_{y' \in \mathcal{Y}} f_y^*(X)$. *For any* $\alpha \in (0,1)$, *let*

$$t_\alpha = \sup\{t \in [0,1] \mid \hat{F}_{\hat{f}_{\hat{y}_X}(X)}(t) \leq \alpha\}, \tag{6.5}$$

with $\hat{y}_x = \arg\max_{y \in \mathcal{Y}} \hat{f}_y(x)$. *Then, the conformal set*

$$C(x) = \{y \in \mathcal{Y} \mid \hat{f}_y(x) \geq t_\alpha - \delta_3\} \tag{6.6}$$

is valid, that is, $\mathbb{P}_X(y_X^* \in C(X)) \geq 1 - \alpha_n'$ *holds with a* $\mathbb{P}_n$*-probability of at least* $1 - (\delta_1 + \delta_2 + \delta_4)$, *where, for any* $\delta_1, \delta_2, \delta_4 \in (0,1)$ *and some constants* $\beta \in (0,1]$, $B, \delta_3, M_1, M_2 > 0$,

$$\alpha_n' := \frac{1}{\delta_3(1-\delta_6)} M_1 \left(\frac{\log(1/\delta_1)}{n}\right)^{\frac{1}{4}\beta} + M_2 \left(\frac{\log(1/\delta_2)}{n}\right)^{\frac{1}{2}\beta} + \alpha + \left(\frac{\log(2/\delta_4)}{2n}\right)^{\frac{1}{2}}.$$

Intuitively, the tighter the upper bounds in Lemma 6.3.3 are, the smaller the necessary correction of the threshold and confidence level in Theorem 6.3.4. In other words, loose upper bounds in Lemma 6.3.3 lead to high cardinalities of $C(x)$ in (6.6). In contrast, tight upper bounds in Lemma 6.3.3 lead to small cardinalities of $C(x)$ in (6.6). The following Remark 6.3.5 details how to obtain conformal validity for a fixed error level.

Remark 6.3.5. Alternatively, one obtains a fixed error level $\alpha_2 \in (0,1)$ in Theorem 6.3.4, that is, $\mathbb{P}_X(y_X^* \in C(X)) \geq 1 - \alpha_2$, by using

$$\alpha'' = \alpha_2 - \frac{1}{\delta_3(1-\delta_6)} M_1 \left(\frac{\log(1/\delta_1)}{n}\right)^{\frac{1}{4}\beta} - M_2 \left(\frac{\log(1/\delta_2)}{n}\right)^{\frac{1}{2}\beta} - \left(\frac{\log(2/\delta_4)}{2n}\right)^{\frac{1}{2}}, \tag{6.7}$$

in the computation of the threshold $t_{\alpha''}$ in (6.5).

While Remark 6.3.5 follows from a simple substitution, it explicitly links Theorem 6.3.4 to the setting usually considered in conformal prediction: One wants to have a conformal predictor that is valid regarding some specified confidence level α_2, which Remark 6.3.5 achieves by using an altered α'' in the computation of $t_{\alpha''}$. If $\alpha'' \leq 0$, the resulting conformal predictor defaults to $C(x) = \mathcal{Y}$, for $x \in \mathcal{X}$, which is valid. In contrast, given

Algorithm 3 Conformal Candidate Cleaning

```
Input: PLL dataset $\mathcal{D} = \{(x_i, s_i) \in \mathcal{X} \times 2^{\mathcal{Y}} : i \in [n]\}$; conformal error level $\alpha \in (0, 1)$;
    number of epochs $R$; number of warm-up rounds $R_{\text{warmup}}$;
Output: Predictor $f : \mathcal{X} \to [0, 1]^k$, $\sum_{y \in \mathcal{Y}} f_y(x) = 1$;
 1: $(\mathcal{D}_t, \mathcal{D}_v) \leftarrow$ Partition $\mathcal{D}$ into $\mathcal{D}_t$ for model training and $\mathcal{D}_v$ for calibrating the con-
    formal sets;
 2: $n' \leftarrow |\,\mathcal{D}_t\,|$;
 3: $(f, \theta) \leftarrow$ Initialize model $f$ and its weights $\theta$;
 4: $(w_{ij})_{i \in [n'], j \in [k]} \leftarrow 1/|s_i|$ if $j \in s_i$, else 0;
 5: for $r = 1, \ldots, R$ do
 6:     ▷ Update predictions on the validation set
 7:     $\mathcal{S} \leftarrow \{\max_{y \in s_i} f_y(x_i; \theta) : (x_i, s_i) \in \mathcal{D}_v\}$;
 8:     ▷ Update $f$'s weights $\theta$
 9:     $\hat{R}(f; w, \theta) \leftarrow -\frac{1}{n'} \sum_{i=1}^{n'} \sum_{j=1}^{k} w_{ij} \log f_j(x_i; \theta)$;
10:     Update $\theta$ by backpropagation on $-\nabla \hat{R}(f; w, \theta)$;
11:     ▷ Clean candidate sets $s_i$
12:     if $r \geq R_{\text{warmup}}$ then
13:         $\alpha_r \leftarrow$ Estimate the adapted error level in (6.7);
14:         for $(x_i, s_i) \in \mathcal{D}_t$ do
15:             $C(x_i) \leftarrow$ Construct the conformal predictor as defined in (6.6) using $\mathcal{S}$ and $\alpha_r$;
16:             if $s_i \cap C(x_i) \neq \emptyset$ then
17:                 $s_i \leftarrow s_i \cap C(x_i)$;
18:     ▷ Update label weights $w_{ij}$
19:     $(w_{ij})_{i \in [n'], j \in [k]} \leftarrow \frac{f_j(x_i)}{\sum_{j' \in s_i} f_{j'}(x_i)}$ if $j \in s_i$, else 0;
20: return predictor $f(\,\cdot\,; \theta)$;
```

some confidence level α, Theorem 6.3.4 gives a conformal predictor that is valid with the confidence level $\alpha'_n \neq \alpha$.

Theorem 6.3.4 enables our proposed algorithm. When using a consistent PLL classifier to label the validation set, a conformal predictor with a threshold set based on these pseudo-labels still satisfies a conformal validity guarantee for an adapted threshold and error level. The subsequent section discusses our approach.

6.3.3. Proposed Algorithm

Based on the conformal predictor in Theorem 6.3.4, we propose a novel candidate cleaning strategy that alternates between training a neural-network-based PLL classifier and pruning the candidate labels by conformal prediction. We outline our method in Algorithm 3. In the following, we provide an overview. Thereafter, we discuss all parts in detail.

First, we randomly partition the dataset $\mathcal{D}$ into $\mathcal{D}_t$ for training the model and $\mathcal{D}_v$ for calibrating the conformal predictor C based on the current state of the prediction model f (Line 1). The training set consists of 80 % and the validation set of 20 % of all instances. We initialize the model f and the label weights w_{ij} in Lines 3–4. Lines 5–23 contain the main training loop, which can be divided into four phases: (1) Updating the predictions on the validation set $\mathcal{D}_v$ for calibration (Lines 6–7), (2) updating the model's weights θ through backpropagation (Lines 8–10), (3) cleaning the candidate sets s_i based on the predicted conformal sets $C(x_i)$ (Lines 11–20), and (4) updating the label weights w_{ij} (Lines 21–22). We detail these phases in the following.

In **phase 1** (Lines 6–7), we use the current model f to predict the labels on the hold-out validation set $\mathcal{D}_v$, which are required for the computations in phase 3.

In **phase 2** (Lines 8–10), we update the model weights θ of the neural network f by performing backpropagation on the risk term (2.5). As our candidate cleaning method is agnostic to the concrete PLL classifier used, one can also use other commonly-used PLL strategies instead.

In **phase 3** (Lines 11–18), we compute the conformal predictor C, which is used to clean the candidate sets. After completing R_{warmup} warm-up epochs, we start with our pruning procedure. In Line 13, we compute α_r for the current epoch r. While it is desirable to use the exact value of α'' in (6.7) in Line 13 of Algorithm 3, its computation is unfortunately infeasible as the constants B and β, for which the Bernstein condition (Assumption 6.3.2) holds, cannot be known unless the true distribution $\mathbb{P}_{XY}$ is known. As the employed PLL classifiers are consistent, that is, they converge to the Bayes classifier with enough samples, we approximate the estimation error terms in (6.7) by the probability mass that the PLL classifier allocates on false class labels, that is, class labels that are not part of the candidate sets and hence cannot be the correct label. Given $(x_i, s_i) \in \mathcal{D}_t$, we set $\alpha_r = \frac{1}{n'} \sum_{i=1}^{n'} \sum_{j \notin s_i} f_j(x_i)$ with $n' = |\, \mathcal{D}_t \,|$. Then, we compute the conformal prediction sets $C(x_i)$ in Line 15 for all training instances $(x_i, s_i) \in \mathcal{D}_t$ using the empirical distribution function of the adapted scores on the validation set; this conformal predictor C is valid by Theorem 6.3.4. We use the conformal sets $C(x_i)$ to prune the candidate sets s_i. If $C(x_i)$ and s_i have a nonempty intersection, which is implied with high probability by the conformal validity (Theorem 6.3.1), we assign $s_i \cap C(x_i)$ to s_i in Line 17.

Finally, in **phase 4** (Lines 21–22), we update the label weights w_{ij} based on the cleaned candidate sets s_i with (2.6).

6.3.4. Runtime Analysis

The main runtime cost of our cleaning method arises from the computation of the conformal sets $C(x_i)$ in Line 15 of Algorithm 3. Finding the rank of $f_y(x_i)$ within $\mathcal{S}$ can be done by first sorting $\mathcal{S}$ and then using a binary search. This requires a total runtime of $O(Rn \log n)$, as we prune candidate labels in each epoch and, both, the training set $\mathcal{D}_t$ and validation set $\mathcal{D}_v$ have a size of $\mathcal{O}(n)$. Note that the runtime of our method is not dependent on the number of feature dimensions d as the considered scores $\mathcal{S}$ are scalars.

6.3.5. Placement regarding Related Work

In this section, we provide a brief comparison of our cleaning strategy with the one employed by Pop (Xu et al., 2023). Pop uses level sets, which we sketch in the following. Let $e > 0$, $(x_i, s_i) \in \mathcal{D}$, the predicted label $\hat{y}_{x_i} = \arg\max_{j \in s_i} f_j(x_i)$, and the second-most likely label $\hat{o}_{x_i} = \arg\max_{j \in s_i, j \neq \hat{y}} f_j(x_i)$. The level sets are of the form $L(e) = \{x \in \mathcal{X} : f_{\hat{y}_x}(x) - f_{\hat{o}_x}(x) \geq e\}$ to gradually clean the candidate labels for instances in $L(e)$. In other words, one is confident in the predicted labels if the distance between the most likely and second-most likely label exceeds some margin. Given $x \in L(e)$, this implies

$$f_{\hat{y}_x}(x) - f_{\hat{o}_x}(x) \geq e \overset{(\dagger)}{\Leftrightarrow} 2 f_{\hat{y}_x}(x) - 1 + \underbrace{\sum_{j' \in \mathcal{Y} \setminus \{\hat{y}_x, \hat{o}_x\}} f_{j'}(x)}_{\leq 1} \geq e \implies f_{\hat{y}_x}(x) \geq \frac{1}{2} e, \tag{6.8}$$

with ($\dagger$) holding as $f_{\hat{o}_x}(x) = 1 - \sum_{j' \in \mathcal{Y} \setminus \{\hat{y}_x, \hat{o}_x\}} f_{j'}(x) - f_{\hat{y}_x}(x)$. Pop gradually decreases the value of e to enlarge the reliable region $L(e)$, which in turn requires $f_{\hat{y}_x}(x) \geq \frac{1}{2} e$ by (6.8). In contrast, in Theorem 6.3.4, we find an appropriate value t such that $f_{\hat{y}_x}(x) \geq t$ holds with a specified probability. The conformal predictor C can therefore also be interpreted as a level set. However, our approach satisfies the conformal validity guarantee.

6.4. Experiments

Section 6.4.1 lists all PLL methods that we compare against, Section 6.4.2 summarizes the experimental setup, and Section 6.4.3 shows our results.

6.4.1. Algorithms for Comparison

In our experiments, we benchmark six state-of-the-art PLL methods. These are Proden (Lv et al., 2020), Cc (Feng et al., 2020), Valen (Xu et al., 2021), Cavl (Zhang et al., 2022a), Pop (Xu et al., 2023), and CroSel (Tian et al., 2024). For each dataset, we use the same base models across all approaches. For the colored-image datasets, we use a ResNet-9 architecture (He et al., 2016). Else, we use a standard d-300-300-300-k MLP (Werbos, 1974). We train all models from scratch. An in-depth overview of all hyperparameters is in Appendix C.3. Appendix C.4 contains additional experiments, including the use of the pre-trained Blip-2 model (Li et al., 2023) on the vision datasets.

6.4.2. Experimental Setup

Data. Using the standard PLL experimentation protocol (Lv et al., 2020; Zhang et al., 2022a; Tian et al., 2024), we perform experiments on real-world PLL datasets and on supervised datasets with artificially added incorrect candidate labels. To report averages and standard deviations, we repeat all experiments five times with different seeds. For the supervised multi-class datasets, we use *mnist* (LeCun et al., 1999), *fmnist* (Xiao et al., 2018), *kmnist* (Clanuwat et al., 2018), *cifar10* (Krizhevsky, 2009), *cifar100* (Krizhevsky, 2009), and *svhn* (Netzer et al., 2011). Regarding the real-world PLL datasets, we use *bird-song* (Briggs et al., 2012), *lost* (Cour et al., 2011), *mir-flickr* (Huiskes and Lew, 2008), *msrc-v2* (Liu and Dietterich, 2012), *soccer* (Zeng et al., 2013), and *yahoo-news* (Guillaumin et al., 2010). An overview of the dataset characteristics is in Appendix C.3.

Candidate generation. As is common in related work, we use two kinds of candidate label generation methods to augment labeled multi-class data with partial labels: Uniform (Hüllermeier and Beringer, 2005; Liu and Dietterich, 2012) and instance-dependent (Xu et al., 2021). For *cifar10* and *cifar100*, we use the uniform generation strategy as in Wang et al. (2022) and the instance-dependent strategy for all other datasets. For adding instance-dependent candidate labels, we first train a supervised MLP classifier $g : \mathcal{X} \to [0, 1]^k$. Then, given an instance $x \in \mathcal{X}$ with correct label $y \in \mathcal{Y}$, we add the incorrect label $\bar{y} \in \mathcal{Y} \setminus \{y\}$ to the candidate set s with a binomial flipping probability of $\xi_{\bar{y}}(x) = g_{\bar{y}}(x) / \max_{y' \in \mathcal{Y} \setminus \{y\}} g_{y'}(x)$. For *cifar10*, we use a constant flipping probability of $\xi_{\bar{y}}(x) = 0.1$. In the *cifar100* dataset, all class labels $\mathcal{Y}$ are partitioned into 20 categories (for example, aquatic mammals consisting of the labels beaver, dolphin, otter, seal, and whale) and we use a constant flipping probability of $\xi_{\bar{y}}(x) = 0.1$ if $\bar{y}$ and y belong to the same category, else we set $\xi_{\bar{y}}(x) = 0$.

6.4.3. Results

Predictive performance. Table 6.1 presents the average test-set accuracies for all competitors on all datasets. We benchmark our conformal candidate cleaning technique combined with all approaches in Section 6.4.1, which is marked by Conf+Method. An overview of significant differences is in Table 6.2. There, we compare the respective method to all the other approaches. All significance tests use a paired t-test with a confidence level of 5 %.

The approaches Conf+Proden, Conf+Pop, and Conf+CroSel that combine the respective approaches with our candidate cleaning strategy win most often (Table 6.2). Conformal candidate cleaning makes Proden win 18 more direct comparisons, Pop win 17 more direct comparisons, and CroSel win 13 more direct comparisons advancing the state-of-the-art prediction performance. These methods significantly benefit from our pruning.

The approaches Cc, Valen, and Cavl yield similar performances when combined with conformal candidate cleaning. For Valen and Cavl, we attribute this to the fact that their

Table 6.1.: Average test-set accuracies (std.) on the real-world datasets (top) and the supervised datasets with added candidate labels (bottom). We benchmark our strategy (Conf+) combined with all existing methods.

Method	*bird-song*	*lost*	*mir-flickr*	*msrc-v2*	*soccer*	*yahoo-news*
Proden (2020)	75.55 (1.08)	78.94 (3.01)	**67.05** (1.18)	54.33 (1.76)	54.18 (0.55)	65.25 (1.00)
Conf+P. (no)	76.27 (0.94)	79.56 (1.96)	66.07 (1.63)	53.00 (2.24)	54.63 (0.81)	65.42 (0.36)
Conf+Proden	**76.99** (0.90)	**80.09** (4.40)	66.91 (1.57)	**54.60** (3.42)	**54.77** (0.84)	**65.93** (0.42)
Cc (2020)	74.49 (1.57)	78.23 (2.11)	62.39 (1.87)	50.96 (2.03)	55.28 (0.96)	**65.03** (0.51)
Conf+Cc	**75.01** (1.84)	**79.38** (1.79)	**63.37** (0.45)	**52.45** (3.64)	**55.52** (0.74)	64.35 (0.64)
Valen (2021)	**72.30** (1.83)	**70.18** (3.44)	**67.05** (1.48)	**49.20** (1.37)	**53.20** (0.88)	**62.25** (0.45)
Conf+Valen	71.22 (1.03)	68.41 (2.95)	61.61 (2.79)	48.37 (2.24)	52.49 (1.00)	62.16 (0.74)
Cavl (2022)	69.78 (3.00)	**72.12** (1.08)	**65.02** (1.34)	**52.67** (2.32)	**55.06** (0.48)	61.91 (0.46)
Conf+Cavl	**72.00** (1.22)	71.24 (3.81)	64.42 (0.89)	51.63 (5.03)	54.85 (0.92)	**62.43** (0.43)
Pop (2023)	75.17 (1.04)	77.79 (2.11)	**67.93** (1.44)	53.83 (0.69)	55.31 (0.71)	65.09 (0.64)
Conf+Pop	**77.58** (1.01)	**78.41** (2.13)	66.21 (2.19)	**54.82** (3.60)	**56.49** (1.10)	**65.25** (0.23)
CroSel (2024)	75.11 (1.79)	**81.24** (3.68)	**67.58** (1.16)	52.23 (2.83)	52.64 (1.21)	**67.72** (0.32)
Conf+CroSel	**77.76** (0.50)	81.15 (2.57)	65.93 (1.94)	**54.10** (2.75)	**54.97** (0.65)	67.55 (0.22)

Method	*mnist*	*fmnist*	*kmnist*	*svhn*	*cifar10*	*cifar100*
Proden (2020)	87.21 (0.83)	71.18 (2.95)	59.31 (1.22)	83.71 (0.37)	**86.42** (0.39)	**61.58** (0.20)
Conf+P. (no)	**91.74** (0.34)	**78.38** (0.50)	**66.88** (0.76)	**87.31** (0.30)	85.39 (0.49)	61.50 (0.20)
Conf+Proden	91.55 (0.23)	78.09 (0.33)	66.43 (0.38)	86.99 (0.41)	85.29 (0.44)	61.45 (0.49)
Cc (2020)	**86.29** (2.18)	**66.19** (2.77)	**58.29** (0.32)	83.40 (0.42)	**85.61** (0.27)	60.43 (0.53)
Conf+Cc	85.20 (4.16)	59.75 (2.68)	57.07 (0.66)	**84.32** (0.31)	84.10 (0.38)	**60.49** (0.37)
Valen (2021)	**78.91** (0.80)	66.53 (2.65)	58.48 (0.45)	54.87 (15.83)	**84.83** (0.23)	58.67 (0.17)
Conf+Valen	74.20 (21.99)	**69.09** (2.71)	**60.95** (2.59)	**78.31** (3.15)	84.35 (0.22)	**59.57** (0.71)
Cavl (2022)	71.11 (3.92)	**59.85** (6.49)	48.15 (5.07)	**72.57** (3.14)	**84.00** (0.94)	**61.97** (0.25)
Conf+Cavl	**71.86** (4.57)	59.54 (6.62)	**52.14** (3.89)	70.53 (2.94)	82.82 (1.58)	61.79 (0.36)
Pop (2023)	87.08 (0.58)	72.30 (2.63)	60.63 (1.15)	83.69 (0.28)	**86.76** (0.29)	61.27 (0.60)
Conf+Pop	**91.19** (0.29)	**79.15** (1.23)	**67.37** (0.28)	**85.89** (0.48)	85.32 (0.38)	**61.38** (0.30)
CroSel (2024)	91.84 (0.44)	76.34 (1.21)	**65.55** (0.81)	75.95 (3.91)	**87.32** (0.22)	63.69 (0.29)
Conf+CroSel	**91.85** (0.61)	**77.31** (0.46)	64.73 (1.52)	**77.70** (3.84)	87.05 (0.09)	**64.55** (0.31)

methods already use pseudo-labeling internally, that is, they treat the most likely label as the possible correct label, which diminishes the positive effect of pruning candidates.

Ablation study. Additionally, we perform an ablation experiment regarding our correction method proposed in Theorem 6.3.4. The approach Conf+Proden (no correction) uses conformal predictions based on the labels provided by the PLL classifier without our

Table 6.2.: Number of significant differences compared to all six methods on all twelve datasets using a paired student t-test (level 5 %).

Comparison vs. all others	Wins	Ties	Losses
Proden (2020)	26	**36**	10
Conf+Proden (no correction)	**37**	24	11
Conf+Proden	**44**	21	7
Cc (2020)	17	**36**	19
Conf+Cc	19	**28**	25
Valen (2021)	3	31	**38**
Conf+Valen	4	26	**42**
Cavl (2022)	8	28	**36**
Conf+Cavl	5	29	**38**
Pop (2023)	27	**38**	7
Conf+Pop	**44**	22	6
CroSel (2024)	**36**	29	7
Conf+CroSel	**49**	19	4

proposed correction method, which is equivalent to a fixed α_r. Table 6.2 shows that, while Conf+Proden (no correction) is already a significant improvement over Proden, our PLL correction strategy improves performance even further by incorporating the possible approximation error of the trained classifier. We limit our ablation study to Proden due to runtime constraints.

Our experiments show that the proposed method yields significant improvements over a wide range of existing PLL models (Proden, Pop, and CroSel) and advances the state-of-the-art prediction performance with the method Conf+CroSel.

6.5. Proofs

This section collects our proofs. Section 6.5.1 contains the proof of Theorem 6.3.1, Section 6.5.2 that of Lemma 6.3.3, and Section 6.5.3 that of Theorem 6.3.4.

6.5.1. Proof of Theorem 6.3.1

Let C be an optimal solution of (6.2). Then, we have

$$\begin{aligned}
\mathbb{P}_{XS}(S \cap C(X) \neq \emptyset) &= 1 - \mathbb{P}_{XS}(S \cap C(X) = \emptyset) = 1 - \mathbb{P}_{XS}(\forall y \in S, y \notin C(X)) \\
&\geq 1 - \mathbb{P}_{XS}(\exists y \in S, y \notin C(X)) \\
&\overset{(a)}{=} 1 - \sum_{y \in \mathcal{Y}} \mathbb{P}(Y = y, y \in S, y \notin C(X)) = 1 - \mathbb{P}(Y \in S, Y \notin C(X)) \\
&\overset{(b)}{=} 1 - \int_{\mathcal{X} \times \mathcal{Y}} \mathbb{P}_{S|X=x,Y=y}(y \in S, y \notin C(x)) \, \mathrm{d}\,\mathbb{P}_{XY}(x, y) \\
&\overset{(c)}{=} 1 - \int_{\mathcal{X} \times \mathcal{Y}} \underbrace{\mathbb{P}_{S|X=x,Y=y}(y \in S)}_{\overset{(d)}{=} 1} \mathbb{P}_{S|X=x,Y=y,y\in S}(y \notin C(x)) \, \mathrm{d}\,\mathbb{P}_{XY}(x, y) \\
&= 1 - \int_{\mathcal{X} \times \mathcal{Y}} \mathbb{P}_{S|X=x,Y=y,y\in S}(y \notin C(x)) \, \mathrm{d}\,\mathbb{P}_{XY}(x, y) \\
&\overset{(e)}{=} 1 - \int_{\mathcal{X} \times \mathcal{Y}} \mathbb{P}_{S|X=x,Y=y}(y \notin C(x)) \, \mathrm{d}\,\mathbb{P}_{XY}(x, y) \\
&\overset{(f)}{=} 1 - \int_{\mathcal{X} \times \mathcal{Y}} \mathbb{1}_{\{y \notin C(x)\}} \, \mathrm{d}\,\mathbb{P}_{XY}(x, y) \\
&= 1 - \mathbb{P}_{XY}(Y \notin C(X)) = \mathbb{P}_{XY}(Y \in C(X)) \overset{(g)}{\geq} 1 - \alpha,
\end{aligned}$$

where (a) is implied by the law of total probability holding for the discrete Y taking mutually exclusive values in $y \in \mathcal{Y}$, (b) holds by the tower rule, (c) holds by the chain rule of conditional probability, (d) holds as $\mathbb{P}_{S|X=x,Y=y}(y \in S) = 1$ for any $(x, y) \in \mathcal{X} \times \mathcal{Y}$, (e) holds by independence, (f) holds as $\mathbb{P}_{S|X=x,Y=y}(y \notin C(x))$ is either one if $y \notin C(x)$ or zero if $y \in C(x)$, and (g) holds by our imposed assumption.

6.5.2. Proof of Lemma 6.3.3

We prove parts (i) and (ii) separately in the following.

Proof of (i)**.** To proof the result, we first show that for any $\hat{f}$, one has the expectation bound

$$\mathbb{E}_{XY}\left[\left\|\hat{f}_Y(X) - f_Y^*(X)\right\|\right] \leq \frac{\lambda\sqrt{B}}{2^{\frac{1}{2}\beta}} \left(R(\hat{f}) - R(f^*)\right)^{\frac{1}{2}\beta}, \tag{6.9}$$

for some constants $\beta \in (0, 1]$ and $B, \lambda > 0$. We then apply a known result (recalled in Theorem C.2.4) to obtain the stated concentration inequality. The details are as follows.

To prove (6.9), notice that

$$\begin{aligned}
\mathbb{E}_{XY}\left[\left|\hat{f}_Y(X) - f_Y^*(X)\right|\right] &= \left(\left(\mathbb{E}_{XY}\left[\left|\hat{f}_Y(X) - f_Y^*(X)\right|\right]\right)^2\right)^{\frac{1}{2}} \overset{(a)}{\leq} \left(\mathbb{E}_{XY}\left[\left|\hat{f}_Y(X) - f_Y^*(X)\right|^2\right]\right)^{\frac{1}{2}} \\
&\overset{(b)}{\leq} \lambda\left(\mathbb{E}_{XY}\left[\left|\ell(\hat{f}(X),Y) - \ell(f^*(X),Y)\right|^2\right]\right)^{\frac{1}{2}} = \lambda\left(\mathbb{E}_{XY}\left[\left(\ell(\hat{f}(X),Y) - \ell(f^*(X),Y)\right)^2\right]\right)^{\frac{1}{2}} \\
&\overset{(c)}{\leq} \lambda\sqrt{B}\left(\mathbb{E}_{XY}\left[\ell(\hat{f}(X),Y) - \ell(f^*(X),Y)\right]\right)^{\frac{1}{2}\beta} \\
&\overset{(d)}{=} \lambda\sqrt{B}\left(\mathbb{E}_{XY}\left[\ell(\hat{f}(X),Y)\right] - \mathbb{E}_{XY}[\ell(f^*(X),Y)]\right)^{\frac{1}{2}\beta} \\
&\overset{(e)}{=} \lambda\sqrt{B}\frac{1}{2^{\frac{1}{2}\beta}}\left(R(\hat{f}) - R(f^*)\right)^{\frac{1}{2}\beta},
\end{aligned}$$

using the following observations. (a) is implied by Jensen's inequality. Next, we note that $z \mapsto -\log(z)$ satisfies the λ-bi-Lipschitz condition on $[\epsilon, 1]$ (Lemma C.1.1), implying

$$\begin{aligned}
\left|\hat{f}_Y(X) - f_Y^*(X)\right| &\leq \lambda\left|-\log\left(\hat{f}_Y(X)\right) - \left(-\log\left(f_Y^*(X)\right)\right)\right| \\
&= \left|\ell\left(\hat{f}(X),Y\right) - \ell\left(f^*(X),Y\right)\right|,
\end{aligned} \tag{6.10}$$

where we used the definition of ℓ for the equality. Using (6.10) together with the monotonicity of the L_2-norm yields (b). (c) holds by the assumed (β, B)-Bernstein condition. The linearity of expectations gives (d) and Theorem C.2.3 yields (e).

Now, to obtain the probabilistic bound, we observe that

$$\begin{aligned}
&\mathbb{P}_n\left(\mathbb{E}_{XY}\left[|\hat{f}_Y(X) - f_Y^*(X)|\right] \leq M_1\left(\frac{\log(1/\delta_1)}{n}\right)^{\frac{1}{4}\beta}\right) \\
&\overset{(6.9)}{\geq} \mathbb{P}_n\left(\lambda\sqrt{B}\frac{1}{2^{\frac{1}{2}\beta}}\left(R(\hat{f}) - R(f^*)\right)^{\frac{1}{2}\beta} \leq M_1\left(\frac{\log(1/\delta_1)}{n}\right)^{\frac{1}{4}\beta}\right) \\
&\overset{(a)}{=} \mathbb{P}_n\left(\lambda^{\frac{2}{\beta}}B^{\frac{1}{\beta}}\frac{1}{2}\left(R(\hat{f}) - R(f^*)\right) \leq M_1\left(\frac{\log(1/\delta_1)}{n}\right)^{\frac{1}{2}}\right) \overset{(b)}{\geq} 1 - \delta_1.
\end{aligned}$$

In (a), we notice that both sides of the inequality are nonnegative and apply the function $z \mapsto z^{2/\beta}$, which is monotonically increasing on $\mathbb{R}^+$. We conclude the proof of part (i) with an application of Theorem C.2.4 in (b), where we let $M_1 = M\lambda^{\frac{2}{\beta}}B^{\frac{1}{\beta}}\frac{1}{2}$ (with M defined in the external result).

Proof of (ii)**.** The proof proceeds in three steps. In step 1, we will show that

$$\begin{aligned}
&\mathbb{E}_{XY}\left[\mathbb{1}_{\{\arg\max_{j\in\mathcal{Y}}\hat{f}_j(X)\neq\arg\max_{j\in\mathcal{Y}}f_j^*(X)\}}\right] \\
&\quad\leq \frac{1}{1-\delta_5}\mathbb{E}_{XY}\left[\left(\mathbb{1}_{\{\arg\max_{j\in\mathcal{Y}}\hat{f}_j(x)\neq y\}} - \mathbb{1}_{\{\arg\max_{j\in\mathcal{Y}}f_j^*(x)\neq y\}}\right)^2\right],
\end{aligned}$$

which will allow us to obtain, in step 2, that for some constants $\beta \in (0,1]$, $B > 0$ and $\delta_5 \in [0,1)$, one has

$$\mathbb{P}_X\left[\arg\max_{j\in\mathcal{Y}} \hat{f}_j(X) \neq \arg\max_{j\in\mathcal{Y}} f_j^*(X)\right] \leq \frac{B}{1-\delta_5}\left(R(\hat{f}) - R(f^*)\right)^{\beta}. \tag{6.11}$$

The result will then follow by an application of Theorem C.2.4, which we elaborate in step 3. The details are as follows.

Step 1. Note that we have

$$1 \overset{(a)}{\leq} \frac{1}{1-\delta_5}\mathbb{P}_{Y|X=x}\left(Y \in \{\hat{y}_x, y_x^*\}\right) \overset{(b)}{\leq} \frac{1}{1-\delta_5}\sum_{y\in\{\hat{y}_x, y_x^*\}} \mathbb{P}_{Y|X=x}(Y = y), \tag{6.12}$$

with the assumption used in (a) and a union bound implying (b).

To conclude the first step, we obtain

$$\begin{aligned}
&\mathbb{E}_{XY}\left[\mathbb{1}_{\{\arg\max_{j\in\mathcal{Y}}\hat{f}_j(X)\neq\arg\max_{j\in\mathcal{Y}} f_j^*(X)\}}\right]\\
&\overset{(a)}{\leq} \frac{1}{1-\delta_5}\mathbb{E}_X\left[\sum_{y\in\{y_X^*,\hat{y}_X\}} \mathbb{P}_{Y|X}(Y=y)\mathbb{1}_{\{\hat{y}_X\neq y_X^*\}}\right]\\
&\overset{(b)}{=} \frac{1}{1-\delta_5}\mathbb{E}_X\left[\sum_{y\in\{y_X^*,\hat{y}_X\}} \mathbb{P}_{Y|X}(Y=y)(\mathbb{1}_{\{\hat{y}_X\neq y\}} - \mathbb{1}_{\{y_X^*\neq y\}})^2\right]\\
&\overset{(c)}{\leq} \frac{1}{1-\delta_5}\mathbb{E}_X\left[\sum_{y\in\mathcal{Y}} \mathbb{P}_{Y|X}(Y=y)(\mathbb{1}_{\{\hat{y}_X\neq y\}} - \mathbb{1}_{\{y_X^*\neq y\}})^2\right]\\
&\overset{(d)}{=} \frac{1}{1-\delta_5}\mathbb{E}_{XY}\left[\left(\mathbb{1}_{\{\arg\max_{j\in\mathcal{Y}}\hat{f}_j(x)\neq y\}} - \mathbb{1}_{\{\arg\max_{j\in\mathcal{Y}} f_j^*(x)\neq y\}}\right)^2\right],
\end{aligned} \tag{6.13}$$

where (a) is implied by (6.12) and the indicator function being nonnegative. For (b), we must show that $\mathbb{1}_{\{\hat{y}_x\neq y_x^*\}} = (\mathbb{1}_{\{\hat{y}_x\neq y\}} - \mathbb{1}_{\{y_x^*\neq y\}})^2$ for any (fixed) $x \in \mathcal{X}$ and $y \in \{\hat{y}_x, y_x^*\}$; it suffices to check the three cases.

- If $y = \hat{y}_x = y_x^*$, then $0 = 0$,
- if $\hat{y}_x \neq y_x^*$ and $y = \hat{y}_x$, then $1 = 1$, and
- if $\hat{y}_x = y_x^*$ and $y \neq \hat{y}_x$, then $1 = 1$.

In (c), we add nonnegative terms and (d) holds by the definition of the expectation.

Step 2. We relax the l.h.s. in (6.11) to

$$\begin{aligned}
&\mathbb{P}_X\left[\arg\max_{j\in\mathcal{Y}} \hat{f}_j(X) \neq \arg\max_{j\in\mathcal{Y}} f_j^*(X)\right] \overset{(a)}{=} \mathbb{E}_X\left[\mathbb{1}_{\{\arg\max_{j\in\mathcal{Y}} \hat{f}_j(X)\neq\arg\max_{j\in\mathcal{Y}} f_j^*(X)\}}\right]\\
&\overset{(b)}{=} \mathbb{E}_{XY}\left[\mathbb{1}_{\{\arg\max_{j\in\mathcal{Y}} \hat{f}_j(X)\neq\arg\max_{j\in\mathcal{Y}} f_j^*(X)\}}\right]\\
&\overset{(c)}{\leq} \frac{1}{1-\delta_5}\,\mathbb{E}_{XY}\left[\left(\mathbb{1}_{\{\arg\max_{j\in\mathcal{Y}} \hat{f}_j(X)\neq Y\}} - \mathbb{1}_{\{\arg\max_{j\in\mathcal{Y}} f_j^*(X)\neq Y\}}\right)^2\right]\\
&\overset{(d)}{=} \frac{1}{1-\delta_5}\,\mathbb{E}_{XY}\left[\left(\ell(\hat{f}(X),Y) - \ell(f^*(X),Y)\right)^2\right] \overset{(e)}{\leq} \frac{B}{1-\delta_5}\left(R(\hat{f}) - R(f^*)\right)^{\beta},
\end{aligned}$$

obtaining the r.h.s. and establishing (6.11). The details are as follows. In (a), we use that a probability can be written as the expectation of an indicator function. We notice in (b) that the integrand does not depend on Y. Regarding (c), with $\hat{y}_x, y_x^*$ defined as in the statement, we use (6.13) obtained in step 1. Defining $\ell : [0,1]^k \longrightarrow \mathbb{R}_{\geq 0}$, $(p,y) \mapsto \mathbb{1}_{\{\arg\max_{y'\in\mathcal{Y}} p_{y'}\neq y\}}$ as the usual 0-1-loss yields (d) and the (β, B)-Bernstein condition gives (e).

Step 3. It remains to obtain the probabilistic bound. We have that

$$\begin{aligned}
&\mathbb{P}_n\left(\mathbb{P}_X\left[\arg\max_{j\in\mathcal{Y}} \hat{f}_j(X) \neq \arg\max_{j\in\mathcal{Y}} f_j^*(X)\right] \leq M_2\left(\frac{\log(1/\delta_2)}{n}\right)^{\frac{1}{2}\beta}\right)\\
&\overset{(6.11)}{\geq} \mathbb{P}_n\left(\frac{B}{1-\delta_5}\left(R(\hat{f}) - R(f^*)\right)^{\beta} \leq M_2\left(\frac{\log(1/\delta_2)}{n}\right)^{\frac{1}{2}\beta}\right)\\
&\overset{(a)}{=} \mathbb{P}_n\left(\left(\frac{B}{1-\delta_5}\right)^{\frac{1}{\beta}}\left(R(\hat{f}) - R(f^*)\right) \leq M_2^{\frac{1}{\beta}}\left(\frac{\log(1/\delta_2)}{n}\right)^{\frac{1}{2}}\right) \overset{(b)}{\geq} 1-\delta_2,
\end{aligned}$$

where we apply the monotonically increasing $z \mapsto z^{1/\beta}$ in (a). In (b), we set $M_2^{1/\beta} = M\left(\frac{B}{1-\delta_5}\right)^{1/\beta}$ and apply Theorem C.2.4 (with M given there). This concludes the proof of part (ii).

6.5.3. Proof of Theorem 6.3.4

To obtain the statement, we first show that one has the following decomposition. For any $\alpha \in (0,1)$ and some $\delta_3 \in (0,1)$,

$$\begin{aligned}
&\mathbb{P}_X\left[f^*_{y^*_X}(X) \geq t_\alpha - \delta_3\right] \qquad (6.14)\\
&\geq \underbrace{\mathbb{P}_X\left[f^*_{y^*_X}(X) \geq \hat{f}_{y^*_X}(X) - \delta_3\right]}_{=:t_1} + \underbrace{\mathbb{P}_X\left[\hat{f}_{y^*_X}(X) = \hat{f}_{\hat{y}_X}(X)\right]}_{=:t_2} + \underbrace{\mathbb{P}_X\left[\hat{f}_{\hat{y}_X}(X) \geq t_\alpha\right]}_{=:t_3} - 2.
\end{aligned}$$

We will then obtain lower bounds on the individual terms t_1, t_2, and t_3, and show that their combination implies the stated result.

Decomposition. Let $A_1 = \{X : f^*_{y^*_X}(X) \geq \hat{f}_{y^*_X}(X) - \delta_3\}$, $A_2 = \{X : \hat{f}_{y^*_X}(X) = \hat{f}_{\hat{y}_X}(X)\}$, $A_3 = \{X : \hat{f}_{\hat{y}_X}(X) \geq t_\alpha\}$, and $B = \{X : f^*_{y^*_X}(X) \geq t_\alpha - \delta_3\}$. Using these definitions, we first obtain that

$$\begin{aligned}
\mathbb{P}_X\left[f^*_{y^*_X}(X) \geq t_\alpha - \delta_3\right] &\overset{(a)}{=} \mathbb{P}_X\left[B\right] \overset{(b)}{\geq} \mathbb{P}_X[A_1 \cap A_2 \cap A_3] \overset{(c)}{=} 1 - \mathbb{P}_X[(A_1 \cap A_2 \cap A_3)^c] \\
&\overset{(d)}{=} 1 - \mathbb{P}_X[A_1^c \cup A_2^c \cup A_3^c] \overset{(e)}{\geq} 1 - \mathbb{P}_X[A_1^c] - \mathbb{P}_X[A_2^c] - \mathbb{P}_X[A_2^c] \\
&\overset{(f)}{=} 1 - (1 - \mathbb{P}_X[A_1]) - (1 - \mathbb{P}_X[A_2]) - (1 - \mathbb{P}_X[A_3]) \\
&\overset{(g)}{=} \underbrace{\mathbb{P}_X[A_1]}_{=t_1} + \underbrace{\mathbb{P}_X[A_2]}_{=t_2} + \underbrace{\mathbb{P}_X[A_3]}_{=t_3} - 2,
\end{aligned} \tag{6.15}$$

with the following details. (a) is by the preceeding definition of the B set. For (b), we have to show that $B \supseteq A_1 \cap A_2 \cap A_3$, which implies that $\mathbb{P}_X\,[B] \geq \mathbb{P}_X\,[A_1 \cap A_2 \cap A_3]$. Let $x \in A_1 \cap A_2 \cap A_3$, then

$$f^*_{y^*_X}(x) \geq \hat{f}_{y^*_X}(x) - \delta_3 = \hat{f}_{\hat{y}_x}(x) - \delta_3 \geq t_\alpha - \delta_3 \quad \Longrightarrow \quad f^*_{y^*_X}(x) \geq t_\alpha - \delta_3.$$

Therefore, $x \in B$, proving (b). (c) considers complementary events and De Morgan's laws yield (d). In (e), we use the inclusion-exclusion principle, where we ignore a few positive terms to obtain the inequality. Considering complementary events gives (f) and cancellations yield (g). This proves (6.14).

Term t_1. To obtain a bound on t_1, we use an expectation bound, which together with Markov's inequality and Lemma 6.3.3 (i) will give the result. The expectation bound is

$$\begin{aligned}
\mathbb{E}_X\left[|\hat{f}_{y^*_X}(X) - f^*_{y^*_X}(X)|\right] &\overset{(a)}{\leq} \mathbb{E}_X\left[\frac{\mathbb{P}_{Y|X}(Y = y^*_X)}{1 - \delta_6}\,|\hat{f}_{y^*_X}(X) - f^*_{y^*_X}(X)|\right] \qquad (6.16) \\
&\overset{(b)}{\leq} \frac{1}{1 - \delta_6}\,\mathbb{E}_X\Big[\sum_{y \in \mathcal{Y}} \mathbb{P}_{Y|X}(Y = y)\,|\hat{f}_y(X) - f^*_y(X)|\Big] \overset{(c)}{=} \frac{1}{1 - \delta_6}\,\mathbb{E}_{XY}\Big[|\hat{f}_Y(X) - f^*_Y(X)|\Big],
\end{aligned}$$

with (a) implied by the assumption $\mathbb{P}_{Y|X}(Y = y^*_X) \geq 1 - \delta_6$ guaranteeing that $1 \leq \frac{\mathbb{P}_{Y|X}(Y = y^*_X)}{1-\delta_6}$. In (b), we use that $y^*_X \in \mathcal{Y}$ and that all terms in the sum are nonnegative. Using a property of the expectation of a joint distribution yields (c).

Next, Markov's inequality (recalled in Lemma C.2.5) implies that

$$\begin{aligned}
\mathbb{P}_X\left[|\hat{f}_{y^*_X}(X) - f^*_{y^*_X}(X)| \geq \delta_3\right] &\overset{\text{C.2.5}}{\leq} \frac{1}{\delta_3}\,\mathbb{E}_X\left[|\hat{f}_{y^*_X}(X) - f^*_{y^*_X}(X)|\right] \\
&\overset{(6.16)}{\leq} \frac{1}{\delta_3(1 - \delta_6)}\,\mathbb{E}_{XY}\left[|\hat{f}_Y(X) - f^*_Y(X)|\right].
\end{aligned} \tag{6.17}$$

Finally, we have that

$$
\begin{aligned}
&\mathbb{P}_n\left[\mathbb{P}_X\left[f^*_{y^*_X}(X) \geq \hat{f}_{y^*_X}(X) - \delta_3\right] \geq 1 - \frac{1}{\delta_3(1-\delta_6)} M_1 \left(\frac{\log(1/\delta_1)}{n}\right)^{\frac{1}{4}\beta}\right] \\
&\quad \overset{(a)}{=} \mathbb{P}_n\left[\mathbb{P}_X\left[\hat{f}_{y^*_X}(X) - f^*_{y^*_X}(X) \leq \delta_3\right] \geq 1 - \frac{1}{\delta_3(1-\delta_6)} M_1 \left(\frac{\log(1/\delta_1)}{n}\right)^{\frac{1}{4}\beta}\right] \\
&\quad \overset{(b)}{\geq} \mathbb{P}_n\left[\mathbb{P}_X\left[\left|\hat{f}_{y^*_X}(X) - f^*_{y^*_X}(X)\right| \leq \delta_3\right] \geq 1 - \frac{1}{\delta_3(1-\delta_6)} M_1 \left(\frac{\log(1/\delta_1)}{n}\right)^{\frac{1}{4}\beta}\right] \\
&\quad \overset{(c)}{\geq} \mathbb{P}_n\left[1 - \mathbb{P}_X\left[\left|\hat{f}_{y^*_X}(X) - f^*_{y^*_X}(X)\right| \leq \delta_3\right] \leq \frac{1}{\delta_3(1-\delta_6)} M_1 \left(\frac{\log(1/\delta_1)}{n}\right)^{\frac{1}{4}\beta}\right] \\
&\quad \overset{(d)}{\geq} \mathbb{P}_n\left[\mathbb{P}_X\left[\left|\hat{f}_{y^*_X}(X) - f^*_{y^*_X}(X)\right| > \delta_3\right] \leq \frac{1}{\delta_3(1-\delta_6)} M_1 \left(\frac{\log(1/\delta_1)}{n}\right)^{\frac{1}{4}\beta}\right] \\
&\quad \overset{(6.17)}{\geq} \mathbb{P}_n\left[\frac{1}{\delta_3(1-\delta_6)} \mathbb{E}_{XY}\left[|\hat{f}_Y(X) - f^*_Y(X)|\right] \leq \frac{1}{\delta_3(1-\delta_6)} M_1 \left(\frac{\log(1/\delta_1)}{n}\right)^{\frac{1}{4}\beta}\right] \\
&\quad \overset{(e)}{\geq} \mathbb{P}_n\left[\mathbb{E}_{XY}\left[|\hat{f}_Y(X) - f^*_Y(X)|\right] \leq M_1 \left(\frac{\log(1/\delta_1)}{n}\right)^{\frac{1}{4}\beta}\right] \overset{(f)}{\geq} 1 - \delta_1,
\end{aligned} \tag{6.18}
$$

where we rearrange the l.h.s. of the inequality in (a). In (b), we consider the absolute value, decreasing the overall probability. In (c), we subtract 1 on both sides and multiply by −1. In (d), we consider the complement of the event. In (e), we simplify and Lemma 6.3.3(i) yields (f).

Term t_2. The observation $\mathbb{P}_X\left[\hat{f}_{y^*_X}(X) = \hat{f}_{\hat{y}_X}(X)\right] \geq \mathbb{P}_X\left[y^*_X = \hat{y}_X\right]$ implies that

$$
\begin{aligned}
&\mathbb{P}_n\left[\mathbb{P}_X\left[\hat{f}_{y^*_X}(X) = \hat{f}_{\hat{y}_X}(X)\right] \geq 1 - M_2 \left(\frac{\log(1/\delta_2)}{n}\right)^{\frac{1}{2}\beta}\right] \\
&\quad \overset{(a)}{\geq} \mathbb{P}_n\left[\mathbb{P}_X\left[y^*_X = \hat{y}_X\right] \geq 1 - M_2 \left(\frac{\log(1/\delta_2)}{n}\right)^{\frac{1}{2}\beta}\right] \\
&\quad \overset{(b)}{=} \mathbb{P}_n\left[\mathbb{P}_X\left[y^*_X \neq \hat{y}_X\right] \leq M_2 \left(\frac{\log(1/\delta_2)}{n}\right)^{\frac{1}{2}\beta}\right] \overset{(c)}{\geq} 1 - \delta_2,
\end{aligned} \tag{6.19}
$$

where (a) holds by the preceding observation. In (b), we subtract 1 on both sides, multiply by −1, and consider the complement of the l.h.s. Inequality (c) was shown in Lemma 6.3.3(ii).

Term t_3. For bounding the third term, we use the Dvoretzky-Kiefer-Wolfowitz inequality (recalled in Theorem C.2.1) In particular, we have

$$\begin{aligned}
&\mathbb{P}_n\left[\mathbb{P}_X[\hat{f}_{\hat{y}_X}(X) \geq t_\alpha] \geq 1 - \left(\alpha + \sqrt{\frac{\log(2/\delta_4)}{2n}}\right)\right] \\
&\overset{(a)}{=} \mathbb{P}_n\left[1 - F_{\hat{f}_{\hat{y}_X}(X)}(t_\alpha) \geq 1 - \left(\alpha + \sqrt{\frac{\log(2/\delta_4)}{2n}}\right)\right] \\
&\overset{(b)}{=} \mathbb{P}_n\left[F_{\hat{f}_{\hat{y}_X}(X)}(t_\alpha) - \alpha \leq \sqrt{\frac{\log(2/\delta_4)}{2n}}\right] \\
&\overset{(c)}{\geq} \mathbb{P}_n\left[F_{\hat{f}_{\hat{y}_X}(X)}(t_\alpha) - \hat{F}_{\hat{f}_{\hat{y}_X}(X)}(t_\alpha) \leq \sqrt{\frac{\log(2/\delta_4)}{2n}}\right] \\
&\overset{(d)}{\geq} \mathbb{P}_n\left[\sup_{t_\alpha}\left|F_{\hat{f}_{\hat{y}_X}(X)}(t_\alpha) - \hat{F}_{\hat{f}_{\hat{y}_X}(X)}(t_\alpha)\right| \overset{(e)}{\leq} \sqrt{\frac{\log(2/\delta_4)}{2n}}\right] \overset{(e)}{\geq} 1 - \delta_4,
\end{aligned} \tag{6.20}$$

where (a) holds as

$$\mathbb{P}_X[\hat{f}_{\hat{y}_X}(X) \geq t_\alpha] = 1 - \mathbb{P}_X[\hat{f}_{\hat{y}_X}(X) \leq t_\alpha] = 1 - F_{\hat{f}_{\hat{y}_X}(X)}(t_\alpha).$$

We rearrange in (b). For obtaining (c), we observe that $\hat{F}_{\hat{f}_{\hat{y}_X}(X)}(t_\alpha) \leq \alpha$. In (d), we consider the supremum, reducing the probability as the inequality becomes more strict. Theorem C.2.1 gives (e).

Combination of t_1, t_2, and t_3. The desired result is obtained by combining the intermediate results using that

$$\begin{aligned}
\mathbb{P}_n\left[\mathbb{P}_X\left[y^*_X \in C(X)\right] \geq 1 - \alpha'_n\right] &\overset{(6.15a)}{=} \mathbb{P}_n\left[\mathbb{P}_X[B] \geq 1 - \alpha'_n\right] \\
&\overset{(6.15)}{\geq} \mathbb{P}_n\left[\mathbb{P}_X[A_1] + \mathbb{P}_X[A_2] + \mathbb{P}_X[A_3] - 2 \geq 1 - \alpha'_n\right] \\
&\overset{(a)}{\geq} 1 - (\delta_1 + \delta_2 + \delta_4),
\end{aligned}$$

where we use a union bound in (a) and the results obtained in (6.18), (6.19), and (6.20); further, we observe that

$$\begin{aligned}
&1 - \frac{1}{\delta_3(1-\delta_6)} M_1 \left(\frac{\log(1/\delta_1)}{n}\right)^{\frac{1}{4}\beta} + 1 - M_2\left(\frac{\log(1/\delta_2)}{n}\right)^{\frac{1}{2}\beta} + 1 - \alpha - \sqrt{\frac{\log(2/\delta_4)}{2n}} - 2 \\
&= 1 - \left(\frac{1}{\delta_3(1-\delta_6)} M_1 \left(\frac{\log(1/\delta_1)}{n}\right)^{\frac{1}{4}\beta} + M_2\left(\frac{\log(1/\delta_2)}{n}\right)^{\frac{1}{2}\beta} + \alpha + \sqrt{\frac{\log(2/\delta_4)}{2n}}\right) = 1 - \alpha'_n.
\end{aligned}$$

7. Amortized Variational Inference for Partial-Label Learning

This chapter's contents are based on the following publication.

- Tobias Fuchs and Nadja Klein. Amortized variational inference for partial-label learning: A probabilistic approach to label disambiguation. *CoRR*, abs/2510.21300, 2025. `https://doi.org/10.48550/arXiv.2510.21300`.

All code and data for reproducing this chapter's experiments are available at `https://github.com/mathefuchs/vi-pll`.

7.1. Overview

Recent state-of-the-art PLL methods (Xu et al., 2021; Tian et al., 2024; Yang et al., 2025) leverage deep learning to optimize surrogate loss functions, such as the minimum loss (Lv et al., 2020), contrastive loss (Wang et al., 2022), or discriminative loss (Yang et al., 2025); and to refine candidate label sets through heuristic strategies, including importance reweighting (Feng et al., 2020), confidence thresholds (Xu et al., 2023), or label smoothing (Gong et al., 2024). In contrast, earlier PLL methods (Jin and Ghahramani, 2002; Liu and Dietterich, 2012) approximate the posterior distribution over true labels directly—typically through expectation-maximization—rather than relying on heuristic refinements. However, these methods are computationally intensive and hardly scale. Moreover, they do not fully exploit recent advances in hardware acceleration, such as those enabling efficient training of neural networks.

Our method addresses this gap by integrating two paradigms: direct approximation of the true label posterior and the use of recent advances in neural network (NN) training techniques. Specifically, we employ amortized variational inference (VI) to approximate the posterior distribution over labels, while using NNs to predict the variational parameters directly from the input data. This formulation enables efficient and scalable inference, thereby overcoming the computational limitations of earlier expectation-maximization-based methods.

Contributions. Our contributions are as follows.

- *A principled PLL framework.* We introduce ViPll, a novel PLL method that formulates label disambiguation as amortized VI. Unlike prior approaches, ViPll directly approximates the posterior over true labels without relying on surrogate losses or heuristic refinements. Variational parameters are predicted via NNs, and the model is trained end-to-end by minimizing the evidence lower bound using stochastic gradient descent. Moreover, our algorithm is computationally efficient.
- *Strong empirical evidence.* We conduct extensive experiments on both synthetic and real-world datasets, benchmarking ViPll against nine state-of-the-art PLL methods. Our approach consistently outperforms existing baselines in predictive accuracy and achieves top performance in the majority of settings. Code and datasets are publicly available.
- *Theoretical justification.* We theoretically justify our optimization objective by deriving it from Bayes' rule and leveraging the equivalence of different causal models underlying PLL, which informs the design of our method.

Notation. Recall the PLL problem and our notations from Chapter 2. Different from these definitions, we consider the measurable space $(\Omega, \mathcal{B}(\Omega))$, where $\Omega = \mathcal{X} \times \Delta^{k-1} \times 2^{\mathcal{Y}}$ (instead of $\Omega = \mathcal{X} \times \mathcal{Y} \times 2^{\mathcal{Y}}$), which allows for more flexibility in modeling an instance's labeling. We define the random variables $X : \Omega \longrightarrow \mathcal{X}$, $Y : \Omega \longrightarrow \Delta^{k-1}$, and $S : \Omega \longrightarrow 2^{\mathcal{Y}}$ to model the generation of instances, their latent label distributions, and the observed candidate labels, respectively. We consider probability measures $\mathbb{P}$ and $\mathbb{Q}$ over $(\Omega, \mathcal{B}(\Omega))$, with corresponding densities p and q defined as the Radon-Nikodym derivatives with respect to a suitable product measure composed of Lebesgue and counting measures.

Outline. Section 7.2 contains our contributions and Section 7.3 our experiments.

7.2. Variational Inference for PLL

We propose ViPll, a novel PLL approach that employs amortized VI as a principled framework for disambiguating the candidate label sets. Specifically, we use fixed-form distributions—Dirichlet and Gaussian in our case—whose parameters are learned by NNs. This combines the flexibility of NNs with the probabilistic rigor of VI. Unlike standard VI, which optimizes variational parameters independently for each data point, amortized VI learns a shared inference model that maps input features to variational parameters via a NN. Importantly, our method is architecture-agnostic, allowing seamless integration with diverse neural architectures and facilitating adaptation to various data modalities.

Similar to an EM procedure, our method alternates between estimating all latent variables via Monte Carlo sampling and optimizing NN parameters through backpropagation. In practice, good predictive performance can be achieved with a relatively small number of

Monte Carlo samples. This allows our method to scale efficiently, in contrast to standard VI methods that are often computationally prohibitive.

Modeling the PLL problem within the VI framework offers a principled way for propagating labeling information and resolving candidate label ambiguity. The following sections provide a detailed exposition of our method. Section 7.2.1 introduces the optimization objective, Sections 7.2.2 – 7.2.4 define the individual components of the objective function, and Section 7.2.5 presents the resulting algorithm.

We note that several existing methods also incorporate probabilistic components, such as modeling labels with a Dirichlet distribution (Xu et al., 2021; Fuchs and Kalinke, 2025). However, unlike our approach, these methods employ the Dirichlet model as an auxiliary mechanism, either as a regularization term or as an enhancement to classifier training, rather than as the core of their method.

Our approach is closest to the expectation-maximization strategies by Jin and Ghahramani (2002) and Liu and Dietterich (2012). However, these methods are computationally expensive, which limits their application on real-world datasets. In contrast, our method leverages NNs to amortize inference by directly predicting variational parameters from input data, enabling efficient and scalable learning.

7.2.1. Optimization Objective

ViPll models the posterior distribution $\mathbb{P}_{\theta,\gamma}(Y \mid X, S)$, which represents the distribution over an instance's class labels given its features and candidate labels. We denote by $p_{\theta,\gamma}(Y \mid X, S)$ the respective posterior density with parameters θ and γ (compare Section 7.2.2 for details). Since the true posterior $\mathbb{P}_{\theta,\gamma}(Y \mid X, S)$ is intractable in practice, we approximate it with a fixed-form variational distribution $\mathbb{Q}_{\phi}(Y \mid X, S)$ with density $q_{\phi}(Y \mid X, S)$. We adopt an amortized VI approach, where $\mathbb{Q}_{\phi}(Y \mid X, S)$ is modeled as a k-dimensional Dirichlet distribution, with parameters $\alpha_{\phi} \in \mathbb{R}^{k}_{\geq 1}$ learned by a NN f_{ϕ}, that is,

$$q_{\phi}(y \mid x, s) = \mathrm{Dir}(y; \alpha_{\phi}) \quad \text{with} \quad \alpha_{\phi} = f_{\phi}(x, s) + 1, \tag{7.1}$$

where $f_{\phi} : \mathcal{X} \times 2^{\mathcal{Y}} \rightarrow \mathbb{R}^{k}_{\geq 0}$ denotes a NN that maps input features and candidate label sets to non-negative parameter vectors, and k is the number of classes. Non-negativity is enforced by applying a *softplus* activation function (Glorot et al., 2011) in the output layer of the network, while the Dirichlet distribution enables modeling uncertainty over class label assignments (Jøsang, 2016; Sensoy et al., 2018). Since $\alpha_{\phi} \geq 1$, $\mathrm{Dir}(y; \alpha_{\phi})$ only has a single mode, which eases its optimization.

Following the standard VI setting, we minimize the expected value of the reverse Kullback-Leibler (KL)-divergence between the variational and true posterior distributions, where the expectation is taken with respect to $\mathbb{P}_{XS}$, the joint distribution of X and S:

$$\begin{aligned}\mathcal{L}(\phi, \theta, \gamma) &= \mathbb{E}_{XS}\Big[D_{\mathrm{KL}}\Big(\mathbb{Q}_{\phi}(Y \mid X, S) \parallel \mathbb{P}_{\theta,\gamma}(Y \mid X, S)\Big)\Big] \\ &\overset{(i)}{=} \mathbb{E}_{(x,s)\sim\mathbb{P}_{XS}} \int_{\Delta^{\mathrm{k}-1}} q_{\phi}(y \mid x, s) \log \frac{q_{\phi}(y \mid x, s)}{p_{\theta,\gamma}(y \mid x, s)} \mathrm{d}y \\ &\overset{(ii)}{=} \mathbb{E}_{(x,s)\sim\mathbb{P}_{XS}}\left[\mathbb{E}_{y\sim q_{\phi}(y \mid x,s)}\left[\log q_{\phi}(y \mid x, s) - \log p_{\theta,\gamma}(y \mid x, s)\right]\right], \end{aligned} \tag{7.2}$$

where (i) applies the definition of the KL divergence, and (ii) that of the expectation.

To model $p_{\theta,\gamma}(y \mid x, s)$ in (7.2), we use Bayes' rule:

$$\mathbb{P}(X, Y, S) = \mathbb{P}(X)\,\mathbb{P}(Y \mid X)\,\mathbb{P}(S \mid X, Y) \tag{7.3}$$

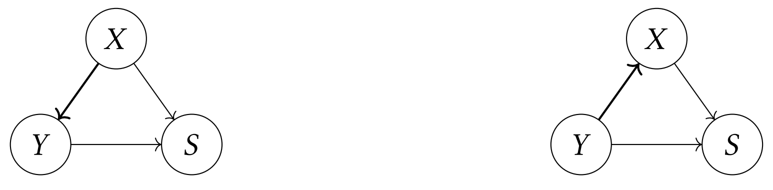
$$= \mathbb{P}(Y)\,\mathbb{P}(X \mid Y)\,\mathbb{P}(S \mid X, Y). \tag{7.4}$$

We argue that (7.4) is beneficial in our setting as it explicitly allows modeling prior information $\mathbb{P}(Y)$ on the class labels as well as a generative model of the observed features $\mathbb{P}(X \mid Y)$ given label information. Additionally, in (7.4), the unobserved variable Y is not dependent on any other variables, which eases its modeling. In contrast, existing work (Liu and Dietterich, 2012; Feng et al., 2020) relies on the factorization in (7.3). Our experiments in Section 7.3.3 confirm our argument in favor of (7.4).

In other words, (7.3) corresponds to a discriminative perspective on the PLL problem, modeling $P(Y \mid X)$, whereas (7.4) adopts a generative perspective, modeling $P(X \mid Y)$. The discriminative formulation focuses on identifying which labels are most likely given an instance's features, while the generative formulation evaluates how well instance features can be reconstructed given labeling information. In the generative setting, the underlying auto-encoder can be pre-trained by learning to reconstruct instance features, providing a useful initialization. Such pre-training is infeasible in the discriminative case, since the true labels are not available, making the generative perspective particularly advantageous.

Example 7.2.1. Consider the *cifar* dataset, which contains images of various object classes such as birds, cars, and airplanes. In the generative case, pre-training the underlying auto-encoder enables the model to uncover latent representations corresponding to visual components like bird wings, car tires, or airplane turbines. This, in turn, helps the model disambiguate labels more effectively. For example, given the label *bird*, the instance's features are expected to include representations of wings.

Using (7.4), we have $\mathbb{P}_{\theta,\gamma}(Y \mid X, S) = \mathbb{P}(Y)\,\mathbb{P}_{\theta,\gamma}(X \mid Y)\,\mathbb{P}(S \mid X, Y)/\mathbb{P}(X, S)$, where in Section 7.2.2, we model $\mathbb{P}_{\theta,\gamma}(X \mid Y)$ with a conditional variational auto-encoder (CVAE; Kingma and Welling 2014; Sohn et al. 2015), Section 7.2.3 elaborates on the prior term $p(y)$, and Section 7.2.4 details the candidate set distribution $p(s \mid x, y)$.



(a) Standard graphical model. **(b)** Markov-equivalent model.

Figure 7.1.: Different DAGs used in PLL. Figure 7.1a shows the standard model from the literature (Cour et al., 2011). Figure 7.1b shows the Markov-equivalent model (compare Proposition 7.2.3) that we use in our work.

Combined with (7.2), we obtain

$$\begin{aligned}\mathcal{L}(\phi,\theta,\gamma) &= \mathbb{E}_{(x,s)\sim\mathbb{P}_{XS}}\big[\,\mathbb{E}_{y\sim q_\phi(y|x,s)}[\log q_\phi(y\mid x,s) - \log p_{\theta,\gamma}(y\mid x,s)]\big]\\ &= \mathbb{E}_{(x,s)\sim\mathbb{P}_{XS}}\big[\,\mathbb{E}_{y\sim q_\phi(y|x,s)}\big[\underbrace{\log q_\phi(y\mid x,s) - \log p(y)}_{\overset{(i)}{=}D_{\mathrm{KL}}(q_\phi(y|x,s)\,\|\,p(y))} - \log p_{\theta,\gamma}(x\mid y)\\ &\qquad - \log p(s\mid x,y) + \underbrace{\log p(x,s)}_{(ii)\text{ constant w.r.t. }\phi,\theta,\gamma}\big]\big],\end{aligned}$$

where (i) acts as a regularization term, and (ii) can be omitted as it is constant in the parameters ϕ, θ, and γ. This induces the β-ELBO, which is defined as

$$\begin{aligned}\beta\text{-ELBO} = \mathbb{E}_{(x,s)\sim\mathbb{P}_{XS}}\big[\,&\mathbb{E}_{y\sim q_\phi(y|x,s)}[\log p_{\theta,\gamma}(x\mid y) + \log p(s\mid x,y)]\\ &- \beta D_{\mathrm{KL}}(q_\phi(y\mid x,s)\,\|\,p(y))\big],\end{aligned} \tag{7.5}$$

where $\arg\min_{\phi,\theta,\gamma}\mathcal{L}(\phi,\theta,\gamma) = \arg\max_{\phi,\theta,\gamma}\beta$-ELBO, for $\beta = 1$. The scaling parameter $\beta\in(0,1]$ allows weighting (7.5) for more flexibility (Higgins et al., 2017).

In practice, we replace $\mathbb{E}_{(x,s)\sim\mathbb{P}_{XS}}$ by a sample average over mini-batches $(x_i,s_i)\in\mathcal{D}_{\mathrm{mb}}\subseteq\mathcal{D}$. Similarly, we replace $\mathbb{E}_{y\sim q_\phi(y|x,s)}$ by taking a sample average of b Monte Carlo samples $y_i\sim q_\phi(y\mid x,s)$ for $i\in[b]$. Sampling from the variational distribution is straightforward, since $q_\phi(y\mid x,s) = \mathrm{Dir}(y;\alpha_\phi)$ and $p(y) = \mathrm{Dir}(y;\alpha^\pi)$ (compare Section 7.2.3) admit a closed-form expression for their KL divergence in (7.5) (Penny, 2001).

Remark 7.2.2. Factorizing the joint distribution $\mathbb{P}(X,Y,S)$ via either (7.3) or (7.4) has the interpretation of a directed acyclic graph (DAG). Specifically, the factorization in (7.3) corresponds to the DAG in Figure 7.1a (which is the standard in the literature; see Cour et al., 2011) and the factorization in (7.4) to that in Figure 7.1b, which we use in our work. Proposition 7.2.3 elaborates on their equivalence, that is, having offline data only, one cannot distinguish the two. Both models explain the observed data equally well.

Proposition 7.2.3 (Markov equivalence)**.** *The causal models represented by the DAGs in Figure 7.1a and 7.1b are Markov equivalent.*

Proof. Recall from (Verma and Pearl, 1990, Theorem 1) that two DAGs are Markov equivalent if and only if they have (1) the same skeleton, that is, the same edges ignoring direction, and (2) the same v-structures, that is, the same set of nodes $A\longrightarrow C\longleftarrow B$, where A and B are not connected. In our setting, (1) and (2) are satisfied. □

7.2.2. Generative Model $p_{\theta,\gamma}(x \mid y)$

To learn how an instance's class label influences its feature distribution, we propose a generative model $p_{\theta,\gamma}(x \mid y)$ based on a CVAE. In a CVAE, one encodes the labeling information using m-dimensional latent variables $Z : \Omega \longrightarrow \mathbb{R}^m$. As the true posterior of the latent variables is intractable in most cases, one approximates it using the variational distribution $r_\gamma(z \mid x, y)$, which is commonly referred to as the conditional encoder.

Using the importance sampling trick, this yields

$$\begin{aligned}
\log p_{\theta,\gamma}(x \mid y) &\overset{(i)}{=} \log \int_{\mathbb{R}} p_\theta(x \mid y, z) p(z \mid y) \mathrm{d}z \\
&\overset{(ii)}{=} \log \int_{\mathbb{R}} r_\gamma(z \mid x, y) \frac{p_\theta(x \mid y, z) p(z \mid y)}{r_\gamma(z \mid x, y)} \mathrm{d}z \\
&\overset{(iii)}{=} \log \mathbb{E}_{z \sim r_\gamma(z \mid x, y)} \left[\frac{p_\theta(x \mid y, z) p(z \mid y)}{r_\gamma(z \mid x, y)} \right] \\
&\overset{(iv)}{\geq} \mathbb{E}_{z \sim r_\gamma(z \mid x, y)} \left[\log \frac{p_\theta(x \mid y, z) p(z \mid y)}{r_\gamma(z \mid x, y)} \right] \\
&\overset{(v)}{=} \mathbb{E}_{z \sim r_\gamma(z \mid x, y)} \left[\log p_\theta(x \mid y, z) - D_{\mathrm{KL}}(r_\gamma(z \mid x, y) \,\|\, p(z \mid y)) \right], \qquad (7.6)
\end{aligned}$$

where (i) marginalizes over the latent variable Z, (ii) introduces the approximate posterior, (iii) uses the definition of the expectation, (iv) applies Jensen's inequality, and (v) recognizes that the second term is the reverse KL-divergence between r_γ and $p(z \mid y)$. We learn $p_{\theta,\gamma}(x \mid y)$ by jointly maximizing (7.6). For this, we make the following assumptions on the decoder $p_\theta(x \mid y, z)$ and the encoder $r_\gamma(z \mid x, y)$.

We assume that the encoder $r_\gamma(z \mid x, y)$ uses NNs to parameterize a Gaussian distribution:

$$r_\gamma(z \mid x, y) = \mathcal{N}(z; \mu_\gamma(x, y), \Sigma_\gamma(x, y)). \qquad (7.7)$$

To compute the KL-term in (7.6) in closed-form, we further assume $p(z \mid y) = \mathcal{N}(z; 0, I_m)$, such that

$$D_{\mathrm{KL}}(r_\gamma(z \mid x, y) \,\|\, p(z \mid y)) = \frac{1}{2} \sum_{i=1}^{m} \left[\sigma_{\gamma,i}(x, y)^2 + \mu_{\gamma,i}(x, y)^2 - 1 - 2 \log \sigma_{\gamma,i}(x, y)^2 \right]. \qquad (7.8)$$

Further, $p_\theta(x \mid y, z)$ is referred to as the conditional decoder for which we also assume a Gaussian distribution, that is,

$$p_\theta(x \mid y, z) = \mathcal{N}(x; \mu_\theta(y, z), \sigma^2 I_m), \qquad (7.9)$$

where $\mu_\theta(y, z)$ uses a NN to parameterize the decoder and σ is assumed to be fixed. Its expectation admits the following closed-form:

$$\mathbb{E}_{z \sim r_\gamma(z \mid x, y)} \log p_\theta(x \mid y, z) = -\frac{1}{2\sigma^2} \underbrace{\|x - \mu_\theta(y, z)\|^2}_{(i)} - \frac{m}{2} \log(2\pi\sigma^2), \qquad (7.10)$$

where $\| \cdot \|$ is the standard Euclidean norm. Note that (i) coincides with the standard mean-squared error and acts as the reconstruction loss of the auto-encoder. In contrast, the KL-term in (7.8) acts as regularization for the encoding step.

In practice, we approximate $p_{\theta,\gamma}(x \mid y)$ by drawing b' samples $z_i \sim r_\gamma(z \mid x, y)$ and compute the importance-weighted average

$$p_{\theta,\gamma}(x \mid y) \approx \frac{1}{b'} \sum_{i=1}^{b'} \frac{p_\theta(x \mid y, z_i) p(z_i \mid y)}{r_\gamma(z_i \mid x, y)}, \tag{7.11}$$

where we use the common *log-sum-exp* trick for numerical stability (Blanchard et al., 2021). After warming-up the CVAE, we set σ in (7.9) to an exponential moving average of the observed RMSE reconstruction loss.

7.2.3. Prior $p(y)$

Most existing work uses a non-informative prior to initialize labeling information. This prior might not satisfy the constraints provided by the candidate sets, however. This is because, given $(x_i, s_i) \in \mathcal{D}$, the class label $y \in \mathcal{Y}$ needs to occur at least $\sum_i \mathbb{1}_{\{s_i = \{y\}\}}$ and at most $\sum_i \mathbb{1}_{\{y \in s_i\}}$ times within the dataset $\mathcal{D}$. Hence, we consider the optimization problem (7.12) and find the maximum entropy prior that satisfies these constraints:

$$\begin{aligned} \max_{\pi \in \Delta^{\text{k-1}}} \ \mathrm{H}(\pi), \ \text{s.t.} \ \pi_y &\geq \frac{1}{|\mathcal{D}|} \sum_{(x_i, s_i) \in \mathcal{D}} \mathbb{1}_{\{s_i = \{y\}\}} \text{ for all } y \in \mathcal{Y}, \\ \pi_y &\leq \frac{1}{|\mathcal{D}|} \sum_{(x_i, s_i) \in \mathcal{D}} \mathbb{1}_{\{y \in s_i\}} \text{ for all } y \in \mathcal{Y}, \end{aligned} \tag{7.12}$$

where $H : \Delta^{\text{k-1}} \to \mathbb{R}, \pi \mapsto -\sum_{y=1}^{k} \pi_y \log \pi_y$ is the entropy.[1] We optimize for π_y using a numerical solver as the entropy objective is non-convex. We set $p(y) = \mathrm{Dir}(y; \alpha^\pi)$ with $\alpha_j^\pi = (\frac{\pi_j}{\min_{j' \in \mathcal{Y}} \pi_{j'}})^\delta \geq 1$ for $j \in \mathcal{Y}$, where $\delta \in [0, 1]$ allows weighting the prior information and $\delta = 0$ implies the uniform prior $p(y) = \mathrm{Dir}(y; 1_k)$.

7.2.4. Candidate Set Distribution $p(s \mid x, y)$

The candidate set distribution $p(s \mid x, y)$ governs how likely candidate sets $s \in 2^{\mathcal{Y}}$ are observed given an instance $x \in \mathcal{X}$ with associated labeling vector $y \in \Delta^{\text{k-1}}$. We use

$$p(s \mid x, y) \overset{(i)}{=} p(s \mid y) \overset{(ii)}{=} \frac{1}{2^{k-1}} \sum_{j \in s} y_j, \tag{7.13}$$

[1] Note that $\pi_y \log \pi_y$ is defined to be zero if $\pi_y = 0$.

where, in (i), we assume that x and s are conditionally independent given y, which is a common assumption in the literature (Liu and Dietterich, 2012; Feng et al., 2020). In (ii), we express $p(s \mid y)$ as the amount of labeling information that agrees with the candidate set s. Therefore, (7.13) acts as a regularization term enforcing that the constraints from the candidate sets are satisfied. Prop. 7.2.4 demonstrates that this is a valid mass function.

Proposition 7.2.4. *$p(s \mid x, y)$ in (7.13) is a valid mass function.*

Proof. Given $y \in \Delta^{\text{k-1}}$,

$$\sum_{s \in 2^{\mathcal{Y}}} p(s \mid y) \overset{(i)}{=} \sum_{s \in 2^{\mathcal{Y}}} \frac{1}{2^{k-1}} \sum_{j \in s} y_j \overset{(ii)}{=} \frac{1}{2^{k-1}} \sum_{s \in 2^{\mathcal{Y}}} \sum_{j \in s} y_j \overset{(iii)}{=} \frac{1}{2^{k-1}} \sum_{j \in \mathcal{Y}} 2^{k-1} y_j \overset{(iv)}{=} \sum_{j \in \mathcal{Y}} y_j \overset{(v)}{=} 1,$$

where (i) inserts (7.13), (ii) moves the factor $1/2^{k-1}$ to the front, (iii) holds as there are 2^{k-1} subsets $s \in 2^{\mathcal{Y}}$ that contain the label $j \in \mathcal{Y}$, (iv) moves the factor 2^{k-1} to the front, and (v) holds as $y \in \Delta^{\text{k-1}}$. ◻

7.2.5. Proposed Algorithm

Algorithm 4 summarizes ViPll, which is grouped into three phases.

Phase 1 (Lines 1–3) sets up the classifier f_ϕ with weights ϕ and the CVAE with weights θ and γ (Line 1), computes the prior according to (7.12) in Line 2, and initializes the labeling vectors $\tilde{y}_i \in \Delta^{\text{k-1}}$, for $i \in [n]$, by uniformly allocating mass on the class labels contained in the candidate sets s_i (Line 3). Later, $\tilde{y}_i$ is updated in the main training loop in Line 22 and informs the training of the CVAE, whose latent representation is conditioned on $\tilde{y}_i$.

Phase 2 (Lines 4–10) is the warm-up phase for the CVAE in which we minimize the reconstruction and regularization losses in (7.10) and (7.8), respectively, using the labeling vectors $\tilde{y}_i$. We use mini-batches and train for $T_{\text{w}} = 500$ epochs using the *Adam* optimizer (Kingma and Ba, 2015).

Phase 3 (Lines 11–22) contains our main training loop which consists of (a) computing the necessary quantities in (7.5) in Lines 14–18, (b) updating our models by backpropagation (Lines 19–20), and (c) updating the labeling vectors $\tilde{y}_i$ in Lines 21–22. Recall from Section 7.2.1 that, in step (a), we make use of Monte Carlo sampling ($b = b' = 10$) and sample averages to approximate the involved expectation terms. In step (b), we optimize the NNs' parameters using backpropagation and the *Adam* optimizer. Finally, in step (c), we update the labeling vectors $\tilde{y}_i$ using the current predictions $\alpha_i = f_\phi(x_i, s_i) + 1$ of our classification model f_ϕ. We train for $T = 1000$ epochs using mini-batches.

Remark 7.2.5. Our method's runtime scales linearly with the number of epochs T and the number of samples b, b', and their product bb'. The main runtime cost arises from the computation of the gradients as b and b' are small constants.

Algorithm 4 ViPll.

```
Input: PLL dataset $\mathcal{D} = \{(x_i, s_i) \in \mathcal{X} \times 2^{\mathcal{Y}} : i \in [n]\}$; number of epochs $T$, $T_\mathrm{w}$; mini-batch
    size $n_\mathrm{m}$; number of MC samples $b$, $b'$; parameters $\beta, \delta \in [0, 1]$;
Output: Predictor $g : \mathcal{X} \to \Delta^{\text{k-1}}$;
 1: Init classifier $f_\phi$ and the CVAE parameterized by $\gamma$, $\theta$;
 2: $\pi \leftarrow$ Compute prior by solving (7.12) numerically;
 3: $\tilde{y}_{ij} \leftarrow \frac{1}{|s_i|}\mathbb{1}_{\{j \in s_i\}}$ for $i \in [n]$, $j \in \mathcal{Y}$;
 4: ▷ Warm-up CVAE
 5: for each epoch $t = 1, \ldots, T_\mathrm{w}$ do
 6:     for each mini-batch $\mathcal{D}_\mathrm{mb} = \{(x_i, \tilde{y}_i, s_i)\}_{i \in [n_\mathrm{m}]}$ do
 7:         Draw one sample $z_i$ from encoder (7.7);
 8:         Compute reconstruction loss of $x_i$ as in (7.10);
 9:         Compute regularization term (7.8);
10:         Update the encoder $\gamma$ and decoder parameters $\theta$ using the Adam optimizer;
11: ▷ Main training loop
12: for each epoch $t = 1, \ldots, T$ do
13:     for each mini-batch $\mathcal{D}_\mathrm{mb} = \{(x_i, \tilde{y}_i, s_i)\}_{i \in [n_\mathrm{m}]}$ do
14:         Draw $b$ samples $(y_{i,o})_{o \in [b]}$ from (7.1);
15:         Compute (7.11) using $b'$ samples and the CVAE model, for each of the $b$ samples;
16:         Compute candidate regularizer (7.13), $\forall b$ samples;
17:         Compute KL term (7.8) in closed-form using $\pi$;
18:         Aggregate all quantities using a sample average;
19:         Update $f$'s params. $\phi$ using the Adam optimizer;
20:         Update CVAE params. $\theta$, $\gamma$ similar to lines 4–10;
21:         ▷ Update current labeling information $\tilde{y}_{ij}$
22:         $\tilde{y}_{ij} \leftarrow \frac{\mathbb{1}_{\{j \in s_i\}} \alpha_{ij}}{\sum_{j' \in s_i} \alpha_{ij'}}$ for $i \in [n_\mathrm{m}]$, $j \in \mathcal{Y}$;
23: return predictor $g_j(x) := \frac{f_{j,\phi}(x, \mathcal{Y}) + 1}{\sum_{j' \in \mathcal{Y}} f_{j',\phi}(x, \mathcal{Y}) + 1}$;
```

7.3. Experiments

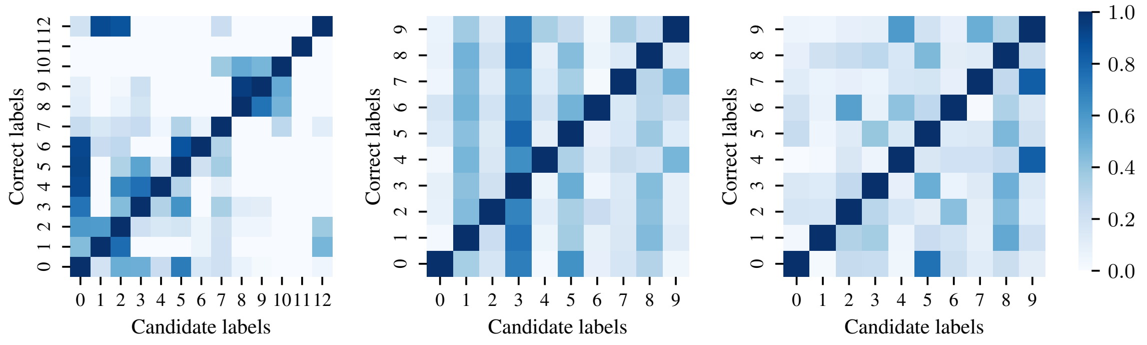
Section 7.3.1 summarizes all PLL methods that we compare against, Section 7.3.2 describes our experimental setup including the datasets and candidate generation strategies used, and Section 7.3.3 shows our main findings.

7.3.1. Competitors

Besides our method ViPll and a variant of it containing ablations, we consider nine established benchmarks from the literature. These are PlKnn (Hüllermeier and Beringer, 2005) and PlEcoc (Zhang et al., 2017), as well as the state-of-the-art deep learning methods Proden (Lv et al., 2020), Valen (Xu et al., 2021), Cavl (Zhang et al., 2022a), PiCO (Wang

Table 7.1.: Dataset characteristics of the five real-world PLL datasets (top) and the five supervised multi-class classification datasets with added candidate labels (bottom). We show the number of instances n, features d, and classes k, as well as the average candidate set sizes.

Dataset	Instances n	Features d	Classes k	Average candidates
bird-song	4 966	38	12	2.175
lost	1 122	108	14	2.217
mir-flickr	2 778	1 536	12	2.758
msrc-v2	1 755	48	22	3.156
yahoo-news	22 762	163	203	1.908
mnist	70 000	784	10	3.958
fmnist	70 000	784	10	3.242
kmnist	70 000	784	10	3.221
cifar10	60 000	3 072	10	4.593
cifar100	60 000	3 072	100	5.540



(a) The real-world partially-labeled dataset *bird-song*.
(b) The *mnist* dataset with our candidate generation strategy.
(c) The *mnist* dataset with the standard generation strategy.

Figure 7.2.: The co-occurrences of candidate labels for different partially-labelled datasets and candidate generation strategies. In Figure 7.2a, for example, the correct label 0 co-occurs most often in candidate sets with the incorrect label 5. Our candidate generation strategy in Figure 7.2b combines the instance-dependent noise in Figure 7.2c with the class imbalances that occur in real-world data (compare Figure 7.2a).

et al., 2022), Pop (Xu et al., 2023), CroSel (Tian et al., 2024), and Cel (Yang et al., 2025). For a fair comparison, we use the same base models for all approaches, that is, an MLP with ReLU activation and batch normalization. For the colored image datasets, we use the pre-trained Blip2 model (Li et al., 2023) to extract 768-dimensional feature vectors.

7.3.2. Experimental Setup

Data. As is common in the PLL literature (Lv et al., 2020; Xu et al., 2023), we use real-world PLL datasets as well as supervised multi-class classification datasets with added

Table 7.2.: Average test-set accuracies (standard deviation) on the five real-world PLL datasets. The best result per dataset is highlighted in **bold**. All methods that are not significantly worse than the best method, using a t-test with level 0.05, are indicated by $*$. ViPll gives the best results in the majority of experiments.

Methods	*bird-song*	*lost*	*mir-flickr*	*msrc-v2*	*yahoo-news*
ViPll (ours)	**76.15 (1.56)**	**78.52 (2.27)**	68.49 (2.13) $*$	**60.24 (2.45)**	63.99 (0.58)
ViPll (w/ abl.)	72.25 (1.55)	77.00 (2.20) $*$	65.25 (2.84) $*$	53.41 (2.82)	49.41 (0.52)
PlKnn (2005)	68.43 (1.38)	44.11 (2.62)	51.98 (2.44)	43.12 (1.96)	45.82 (0.16)
PlEcoc (2017)	61.04 (2.17)	64.17 (3.81)	50.68 (0.78)	28.78 (1.61)	47.64 (0.36)
Proden (2020)	71.17 (1.66)	73.44 (2.01)	**68.67 (1.74)**	55.23 (3.44)	62.65 (1.21)
Valen (2021)	71.99 (1.72)	67.02 (3.63)	64.96 (2.16)	50.11 (2.44)	59.91 (1.43)
Cavl (2022)	68.11 (1.21)	66.58 (3.33)	63.13 (4.63) $*$	53.64 (3.16)	62.60 (1.24)
PiCO (2022)	72.81 (1.10)	66.93 (3.03)	49.32 (2.38)	56.82 (5.17) $*$	61.09 (0.66)
Pop (2023)	71.71 (1.00)	72.73 (1.93)	67.27 (2.03) $*$	55.23 (2.85)	63.09 (0.68)
CroSel (2024)	75.49 (1.25) $*$	72.91 (3.01)	65.43 (1.86)	52.33 (4.11)	**67.10 (0.77)**
Cel (2025)	71.75 (1.76)	74.15 (2.11)	68.56 (2.87) $*$	53.13 (2.89)	63.77 (1.52)

Table 7.3.: Average test-set accuracies (standard deviation) on the five supervised multi-class classification datasets with added candidate labels. The best result per dataset is highlighted in **bold**. All methods that are not significantly worse than the best method, using a t-test with level 0.05, are indicated by $*$. ViPll gives the best results in the majority of experiments.

Methods	*mnist*	*kmnist*	*fmnist*	*cifar10*	*cifar100*
ViPll (ours)	**80.81 (0.60)**	58.78 (1.34) $*$	**74.87 (0.93)**	**96.64 (3.92)**	**77.76 (0.51)**
ViPll (w/ abl.)	52.93 (1.76)	40.41 (0.70)	67.36 (0.66)	88.43 (0.14)	45.74 (2.49)
PlKnn (2005)	63.73 (0.17)	46.62 (0.04)	61.70 (0.18)	76.17 (0.23)	68.05 (0.01)
PlEcoc (2017)	55.59 (1.81)	37.94 (0.87)	66.15 (1.90)	77.13 (4.88)	52.61 (1.01)
Proden (2020)	71.10 (1.41)	58.86 (0.55) $*$	69.04 (1.00)	87.17 (0.26)	77.16 (1.00) $*$
Valen (2021)	59.31 (2.30)	43.97 (0.80)	65.52 (1.74)	82.63 (0.55)	71.85 (0.26)
Cavl (2022)	72.12 (2.41)	57.88 (1.80) $*$	71.54 (1.08)	89.73 (3.79)	72.73 (1.52)
PiCO (2022)	78.45 (0.58)	56.10 (1.52)	73.89 (0.49) $*$	93.08 (4.85) $*$	70.30 (0.83)
Pop (2023)	71.88 (0.87)	58.25 (0.58)	69.57 (0.63)	87.18 (0.24)	77.42 (0.76) $*$
CroSel (2024)	73.26 (0.83)	**59.05 (0.25)**	69.55 (0.69)	88.24 (0.09)	77.68 (0.99) $*$
Cel (2025)	68.57 (1.90)	56.26 (1.38)	68.26 (1.06)	85.91 (0.14)	73.50 (0.74)

candidates. Table 7.1 summarizes the dataset characteristics. The real-world PLL datasets include the *bird-song* (Briggs et al., 2012), *lost* (Cour et al., 2011), *mir-flickr* (Huiskes and Lew, 2008), *msrc-v2* (Liu and Dietterich, 2012), and *yahoo-news* dataset (Guillaumin et al., 2010). The supervised multi-class classification datasets include the *mnist* (LeCun et al., 1999), *fmnist* (Xiao et al., 2018), *kmnist* (Clanuwat et al., 2018), *cifar10* (Krizhevsky, 2009), and *cifar100* dataset (Krizhevsky, 2009).

Candidate Generation. To obtain partial labels for the five supervised datasets, we add candidate labels using the common instance-dependent generation strategy (Xu et al., 2021; Yang et al., 2025) as follows. One first trains a supervised MLP classifier $g : \mathcal{X} \longrightarrow \Delta^{\text{k-1}}$. Then, given an instance $x \in \mathcal{X}$ with correct label $y \in \mathcal{Y}$, a binomial flipping probability of $\xi_1(x, \bar{y}) = g_{\bar{y}}(x) / \max_{y' \in \mathcal{Y} \setminus \{\bar{y}\}} g_{y'}(x)$ decides whether to add the incorrect label $\bar{y} \neq y$ to the candidate set s, that is, one samples $w_{\bar{y}} \sim \mathcal{U}(0, 1)$ for each $\bar{y} \neq y$ and adds it to x's candidate set s if $w_{\bar{y}} \leq \xi_1(x, \bar{y})$.

This generation strategy, however, leads to a rather balanced distribution of incorrect candidate labels (compare Figure 7.2c). Therefore, we combine the instance-dependent flipping probability ξ_1 with a random long-tail class imbalance $\xi_2(x, \bar{y}) = 0.025^{\frac{\pi(\bar{y})+1}{k}}$, where $\pi : \mathcal{Y} \longrightarrow \mathcal{Y}$ is a random permutation of the class labels. The resulting probability is $\xi(x, \bar{y}) = 0.3\xi_1(x, \bar{y}) + 0.7\xi_2(x, \bar{y})$ and we add $\bar{y} \neq y$ to x's candidate set s if $w_{\bar{y}} \leq \xi(x, \bar{y})$, with $w_{\bar{y}} \sim \mathcal{U}(0, 1)$. Figure 7.2 compares the co-occurrences of the candidate labels on the real-world PLL dataset *bird-song* in Figure 7.2a, on the *mnist* dataset with the instance-dependent strategy in Figure 7.2c, and on the *mnist* dataset with our mixed generation strategy in Figure 7.2b. The *mnist* dataset with the instance-dependent strategy entails a rather balanced distribution of incorrect candidates (Figure 7.2c). Real-world PLL data, however, often has a highly imbalanced distribution of incorrect candidate labels (Figure 7.2a). Class 0, for example, appears more often as an incorrect candidate label compared to class 1. We mimic this with our generation strategy in Figure 7.2b.

7.3.3. Results

Predictive Performance. We repeat all experiments five times and show means and standard deviations of the results. Table 7.2 shows all results on the real-world PLL datasets and Table 7.3 on the supervised classification datasets with added candidate labels. On both, the real-world PLL and the supervised datasets, our method ViPll has the best results in the majority of experiments. We mark methods that are not significantly worse with $*$ using a t-test with level 0.05. On the *lost* and *mnist* datasets, ViPll significantly outperforms the competitors; on the *bird-song*, *msrc-v2*, *fmnist*, *cifar10*, and *cifar100* datasets, ViPll performs best with only few competitors that are not significantly worse; and on the *mir-flickr*, *yahoo-news*, and *kmnist* datasets, ViPll performs comparable compared to the best performing method. Overall, ViPll wins most direct comparisons with its competitors.

Ablation Study. ViPll relies on the causal factorization of the PLL problem in Figure 7.1b (as discussed in Section 7.2 and 7.2.1). ViPll (with ablations), which omits the use of the

Markov equivalence shown in Proposition 7.2.3 and uses the causal model in Figure 7.1a, minimizes

$$\mathcal{L}^{\text{abl}}(\phi) = \mathbb{E}_{(x,s)\sim\mathbb{P}_{XS}}\Big[\,\mathbb{E}_{y\sim q_\phi(y|x,s)}\big[\underbrace{\log q_\phi(y \mid x, s) - \log p(y \mid x)}_{=D_{\mathrm{KL}}(q_\phi(y|x,s)\,\|\,p(y|x))} - \log p(s \mid x, y)\big] \\ - \underbrace{\log p(x) + \log p(x, s)}_{\text{constant w.r.t. }\phi}\Big], \tag{7.14}$$

where $q_\phi(y \mid x, s)$ is modeled as discussed in Section 7.2.1 and $p(s \mid x, y)$ as discussed in Section 7.2.4. The term $p(y \mid x)$ is modeled by maintaining a labeling vector y_i for each $(x_i, s_i) \in \mathcal{D}$ and updating it similarly to Line 22 in Algorithm 4. The remaining terms are constant w.r.t. ϕ.

Recall from Section 7.2.1 that the causal model in (7.4) is beneficial in our setting as it explicitly allows modeling a generative model of the observed features $\mathbb{P}(X \mid Y)$ as well as prior information $\mathbb{P}(Y)$. The results in Table 7.2 and 7.3 support this hypothesis: The approach in (7.14) is inferior to our method in most cases. ViPll (with ablations) performs comparable to ViPll only on the *lost* and *mir-flickr* datasets. On the remaining datasets, it performs worse.

To summarize our findings, ViPll performs the best in almost all cases and across a wide-range of artificial and real-world datasets. Additionally, our ablation experiments demonstrate the benefit of leveraging the Markov equivalence shown Proposition 7.2.3.

8. Conclusions and Future Work

Partial-label learning (PLL) is a form of weakly supervised learning in which each instance is annotated with a set of candidate labels, only one of which corresponds to the true class. The lack of exact class labels poses substantial challenges for constructing reliable classifiers, as ambiguity and noise are inherent in the data. In this work, we proposed four novel methods for partial-label learning that address different aspects of this problem, ranging from uncertainty handling and robustness to principled candidate refinement and probabilistic modeling. In the following, we summarize these contributions, discuss their implications, outline open research questions for future work, and elaborate on the broader setting of weakly supervised learning.

PLL with a Reject Option. When misclassifications are costly or potentially harmful, reject options provide a principled mechanism to mitigate the impact of incorrect predictions by allowing the model to abstain from making predictions when uncertain. In Chapter 4, we introduced a novel nearest-neighbor-based PLL algorithm that incorporates a reject option grounded in Dempster-Shafer theory. This approach quantifies uncertainty through a feasible region representing all plausible label distributions and decides to reject predictions that fall below a certain confidence threshold. Our analysis demonstrates that the proposed reject mechanism satisfies desirable theoretical properties, such as rejecting random guesses and ensuring that the predicted label accumulates the highest belief mass across all feasible probability measures. Experiments confirm that this approach effectively reduces the number of harmful misclassifications, which is particularly valuable in domains where human oversight remains crucial, such as crowdsourced labeling or medical image analysis.

However, some research questions remain open. A key direction is to establish theoretical principles for setting the reject threshold adaptively, based on a desired rejection rate or application-specific cost structure. Furthermore, decoupling the reject option from the nearest-neighbor framework could enhance flexibility, allowing its integration into deep or probabilistic models. Extending the theoretical analysis to characterize the trade-off between rejection rate and predictive accuracy would also provide a more comprehensive understanding of rejection-based learning under weak supervision.

Robust PLL by Leveraging Class Activation Values. Chapter 5 focused on robustness, a property essential for ensuring reliable predictions under uncertain and imperfect inputs. We proposed a novel PLL method that leverages class activation values to assess prediction confidence and explicitly model uncertainty. By parameterizing a Dirichlet distribution over class probabilities, our approach provides a principled way to quantify predictive uncertainty and separate confident from ambiguous predictions. Empirical results demonstrate that this model exhibits increased robustness to three major challenges in weak supervision: high levels of candidate-label noise, out-of-distribution inputs, and adversarial perturbations of instance features.

An open question, however, is to provide further theoretical guarantees for the robustness of the proposed classifier, which is non-trivial given that partial-label noise and adversarial noise are intertwined in the PLL setting. Also, it has been noted in the literature that evidential deep learning, underlying our approach, fails to learn a well-calibrated epistemic uncertainty measure with respect to a reference distribution. It excels at forming a relative notion of uncertainty though, which is sufficient for most downstream tasks. Further examining the implications of this finding would be beneficial. Another promising direction involves extending this framework to incorporate distributional robustness, for instance, by optimizing for worst-case risk across label uncertainties.

PLL with Conformal Candidate Cleaning. In Chapter 6, we addressed the challenge of reducing candidate-label ambiguity in a principled rather than heuristic manner. We introduced a novel conformal candidate cleaning framework that alternates between training a PLL classifier and pruning unlikely candidate labels using conformal prediction. Conformal prediction offers statistically grounded guarantees on confidence levels, ensuring that the true label remains in the refined candidate set with a user-specified probability. This iterative process substantially improves classifier training by reducing label noise and providing a clearer supervision signal. Experimental results show that conformal candidate cleaning enhances accuracy across multiple PLL methods and datasets, demonstrating its general applicability as a plug-in extension.

Future work can further deepen the theoretical understanding of this approach. In particular, while our theoretical results establish conformal validity of the predictor used to prune the candidate sets, it is yet to be shown that the predicted conformal sets achieve minimal or near-minimal cardinality. This property is desirable, since predicting the entire label space yields a trivially valid—but uninformative—conformal predictor. Although our experiments show that the conformal sets are reasonably small in practice, theoretical bounds would give further evidence. Establishing such bounds in the PLL setting is non-trivial, however, as one must account for errors in the pseudo-labels of the validation set, which is used for calibration. Moreover, exploring adaptive calibration strategies or online conformal updates could make this framework more practical for dynamic data scenarios.

Amortized Variational Inference for PLL. Chapter 7 introduced a probabilistic approach that formulates label disambiguation as a latent-variable inference problem. Using amortized variational inference, we directly approximate the posterior distribution over true labels through neural networks that predict variational parameters from input data. This formulation combines the expressive power of deep learning with the interpretability and rigor of probabilistic modeling. Our experiments demonstrated that this approach achieves state-of-the-art predictive accuracy across synthetic and real-world datasets, including crowdsourcing and web mining. By viewing the problem from a generative rather than discriminative perspective, the method provides a unified probabilistic view of PLL and offers a foundation for uncertainty-aware learning in broader weakly supervised contexts.

However, some challenges remain. For instance, there is an amortization gap as the neural network may not perfectly predict the variational parameters for each data point, which can affect both inference accuracy and generalization. Investigating how architectural choices and optimization strategies influence this bias could lead to improved model stability. Additionally, integrating more expressive posterior families, such as normalizing flows or hierarchical priors, could capture richer uncertainty structures. Finally, future work may explore combining variational inference with active learning or reinforcement learning frameworks to guide label acquisition dynamically for more stable training.

Beyond these specific contributions, this thesis contributes to the broader understanding of weakly supervised learning, a family of paradigms that includes semi-supervised learning, noisy-label learning, complementary-label learning, and partial learning, among many others. While each of these subfields has evolved independently, their underlying challenges—limited supervision, label noise, and uncertainty quantification—are deeply interconnected. Developing unified frameworks that bridge these problem formulations would yield both theoretical coherence and practical benefits, allowing algorithms to generalize across different supervision regimes. Another promising avenue is the integration of weak supervision with foundation models and large-scale pretraining. As modern machine learning increasingly relies on massive, imperfectly labeled data, the principles of PLL and related weakly supervised learning techniques can help manage uncertainty and enhance trustworthiness. Furthermore, expanding the theoretical study of robustness under weak supervision, including guarantees for confidence calibration, rejection behavior, and adversarial resilience, remains an open direction.

Overall, this thesis advances the field of weakly supervised learning by proposing four complementary approaches that together enhance the accuracy and robustness of partial-label learning methods. Through both theoretical analysis and empirical evaluation, we provide new insights into how machine learning systems can operate reliably when supervision is ambiguous or incomplete. The methods developed establish a foundation for more trustworthy weakly supervised algorithms and open several avenues for future exploration.

Bibliography

Joaquín Abellán, George J. Klir, and Serafín Moral. Disaggregated total uncertainty measure for credal sets. *International Journal of General Systems*, 35(1):29–44, 2006.

Ethem Alpaydin and Fevzi Alimoglu. Pen-based recognition of handwritten digits. UCI Machine-Learning Repository, 1998. `https://doi.org/10.24432/C5MG6K`.

Shuang Ao, Stefan Rueger, and Advaith Siddharthan. Two sides of miscalibration: Identifying over and under-confidence prediction for network calibration. In *Uncertainty in Artificial Intelligence*, pages 77–87, 2023.

Hagai Attias. Inferring parameters and structure of latent variable models by variational bayes. In *Uncertainty in Artificial Intelligence*, pages 21–30, 1999.

Jean-Yves Audibert. *PAC-Bayesian statistical learning theory.* PhD thesis, Université Paris, 2004.

Kevin Bache and Moshe Lichman. UCI machine-learning repository, 2013. `https://archive.ics.uci.edu/`.

Wei-Xuan Bao, Jun-Yi Hang, and Min-Ling Zhang. Partial label dimensionality reduction via confidence-based dependence maximization. In *Conference on Knowledge Discovery and Data Mining*, pages 46–54, 2021.

Wei-Xuan Bao, Jun-Yi Hang, and Min-Ling Zhang. Submodular feature selection for partial label learning. In *Conference on Knowledge Discovery and Data Mining*, pages 26–34, 2022.

Rina Foygel Barber, Emmanuel J. Candes, Aaditya Ramdas, and Ryan J. Tibshirani. Conformal prediction beyond exchangeability. *The Annals of Statistics*, 51(2):816–845, 2023.

Peter L. Bartlett and Shahar Mendelson. Empirical minimization. *Probability Theory and Related Fields*, 135(3):311–334, 2006.

Matthew J. Beal. *Variational algorithms for approximate Bayesian inference.* PhD thesis, University College London, 2003.

T. John Berkmans and S. Karthick. A widespread survey on machine learning techniques and user substantiation methods for credit card fraud detection. *International Journal of Business Intelligence and Data Mining*, 22(1):223–247, 2023.

Christopher M. Bishop. *Pattern recognition and machine learning*. Springer, 2007.

Pierre Blanchard, Desmond J. Higham, and Nicholas J. Higham. Accurately computing the log-sum-exp and softmax functions. *IMA Journal of Numerical Analysis*, 41(4):2311–2330, 2021.

James Bridge, Sean Holden, and Lawrence Paulson. First-order theorem proving. UCI Machine-Learning Repository, 2013. `https://doi.org/10.24432/C5RC9X`.

Forrest Briggs, Xiaoli Z. Fern, and Raviv Raich. Rank-loss support instance machines for MIML instance annotation. In *Conference on Knowledge Discovery and Data Mining*, pages 534–542, 2012.

Yaroslav Bulatov. NotMNIST dataset, 2011. `http://yaroslavvb.blogspot.com/2011/09/notmnist-dataset.html`.

Massimo Buscema and Stefano Terzi. Semeion handwritten digit. UCI Machine-Learning Repository, 2008. `https://doi.org/10.24432/C5SC8V`.

Tom Bylander. Learning linear threshold functions in the presence of classification noise. In *Conference on Computational Learning Theory*, pages 340–347, 1994.

Yuzhou Cao, Tianchi Cai, Lei Feng, Lihong Gu, Jinjie Gu, Bo An, Gang Niu, and Masashi Sugiyama. Generalizing consistent multi-class classification with rejection to be compatible with arbitrary losses. In *Advances in Neural Information Processing Systems*, 2022.

Nontawat Charoenphakdee, Zhenghang Cui, Yivan Zhang, and Masashi Sugiyama. Classification with rejection based on cost-sensitive classification. In *International Conference on Machine Learning*, pages 1507–1517, 2021.

Dawei Cheng, Sheng Xiang, Chencheng Shang, Yiyi Zhang, Fangzhou Yang, and Liqing Zhang. Spatio-temporal attention-based neural network for credit card fraud detection. In *AAAI Conference on Artificial Intelligence*, pages 362–369, 2020.

Xin Cheng, Yuzhou Cao, Haobo Wang, Hongxin Wei, Bo An, and Lei Feng. Regression with cost-based rejection. In *Advances in Neural Information Processing Systems*, 2023.

Tarin Clanuwat, Mikel Bober-Irizar, Asanobu Kitamoto, Alex Lamb, Kazuaki Yamamoto, and David Ha. Deep learning for classical japanese literature. *CoRR*, abs/1812.01718, 2018.

Timothée Cour, Benjamin Sapp, and Ben Taskar. Learning from partial labels. *Journal of Machine Learning Research*, 12:1501–1536, 2011.

Fabio Cuzzolin. *The Geometry of Uncertainty – The Geometry of Imprecise Probabilities*. Springer, 2021.

Arthur P. Dempster. Upper and lower probabilities induced by a multivalued mapping. *The Annals of Statistic*, 28:325–339, 1967.

Thierry Denoeux. A k-nearest neighbor classification rule based on dempster-shafer theory. *Transactions on Systems, Man, and Cybernetics*, 25(5):804–813, 1995.

Thierry Denoeux. Decision-making with belief functions: A review. *International Journal for Approximate Reasoning*, 109:87–110, 2019.

Helen W. Dodson and Emma R. Hedeman. Solar flare. UCI Machine-Learning Repository, 1989. `https://doi.org/10.24432/C5530G`.

Robert Duin. Multiple features. UCI Machine-Learning Repository, 2002. `https://doi.org/10.24432/C5HC70`.

Aryeh Dvoretzky, Jack Kiefer, and Jacob Wolfowitz. Asymptotic minimax character of the sample distribution function and of the classical multinomial estimator. *Annals of Mathematical Statistics*, 27(3):642–669, 1956.

Lei Feng and Bo An. Partial label learning with self-guided retraining. In *AAAI Conference on Artificial Intelligence*, pages 3542–3549, 2019.

Lei Feng, Jiaqi Lv, Bo Han, Miao Xu, Gang Niu, Xin Geng, Bo An, and Masashi Sugiyama. Provably consistent partial-label learning. In *Advances in Neural Information Processing Systems*, 2020.

Tobias Fuchs and Florian Kalinke. Robust partial-label learning by leveraging class activation values. *Machine Learning*, 114(193), 2025.

Tobias Fuchs, Florian Kalinke, and Klemens Böhm. Partial-label learning with a reject option. *Transactions on Machine Learning Research*, January 2025.

Mehdi Gheisari, Hooman Hamidpour, Yang Liu, Peyman Saedi, Arif Raza, Ahmad Jalili, Hamidreza Rokhsati, and Rashid Amin. Data mining techniques for web mining: a survey. *Artificial Intelligence and Applications*, 1(1):3–10, 2023.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *International Conference on Artificial Intelligence and Statistics*, pages 315–323, 2011.

Tilmann Gneiting and Adrian E. Raftery. Strictly proper scoring rules, prediction, and estimation. *Journal of the American Statistical Association*, 102(477):359–378, 2007.

Xiuwen Gong, Nitin Bisht, and Guandong Xu. Does label smoothing help deep partial label learning? In *International Conference on Machine Learning*, 2024.

Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*, 2015.

Yves Grandvalet. Logistic regression for partial labels. In *Information Processing and Management of Uncertainty in Knowledge-based Systems*, 2002.

Peter D. Grünwald and Nishant A. Mehta. Fast rates for general unbounded loss functions: From ERM to generalized Bayes. *Journal of Machine Learning Research*, 21:56:1–56:80, 2020.

Matthieu Guillaumin, Jakob Verbeek, and Cordelia Schmid. Multiple instance metric learning from automatically labeled bags of faces. In *European Conference on Computer Vision*, pages 634–647, 2010.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks. In *International Conference on Machine Learning*, pages 1321–1330, 2017.

Mohamed Farouk Abdel Hady and Friedhelm Schwenker. Semi-supervised learning. In *Handbook on Neural Information Processing*, volume 49, pages 215–239. Springer, 2013.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

Shuo He, Chaojie Wang, Guowu Yang, and Lei Feng. Candidate label set pruning: A data-centric perspective for deep partial-label learning. In *International Conference on Learning Representations*, 2024.

Martin E. Hellman. The nearest neighbor classification rule with a reject option. *Transactions on Systems Science and Cybernetics*, 6(3):179–185, 1970.

Irina Higgins, Loïc Matthey, Arka Pal, Christopher P. Burgess, Xavier Glorot, Matthew M. Botvinick, Shakir Mohamed, and Alexander Lerchner. β-vae: Learning basic visual concepts with a constrained variational framework. In *International Conference on Learning Representations*, 2017.

Arthur E. Hoerl and Robert W. Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 42(1):80–86, 1970.

Paul Horton and Kenta Nakai. A probabilistic classification system for predicting the cellular localization sites of proteins. In *International Conference on Intelligent Systems for Molecular Biology*, pages 109–115, 1996.

Constantin Hubmann, Marvin Becker, Daniel Althoff, David Lenz, and Christoph Stiller. Decision making for autonomous driving considering interaction and uncertain prediction of surrounding vehicles. In *Intelligent Vehicles Symposium*, pages 1671–1678, 2017.

Mark J. Huiskes and Michael S. Lew. The MIR flickr retrieval evaluation. In *International Conference on Multimedia Information Retrieval*, pages 39–43, 2008.

Eyke Hüllermeier and Jürgen Beringer. Learning from ambiguously labeled examples. *Intelligent Data Analysis*, 10(5):168–179, 2005.

Eyke Hüllermeier and Willem Waegeman. Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods. *Springer Machine Learning*, 110(3): 457–506, 2021.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456, 2015.

Takashi Ishida, Gang Niu, Aditya Krishna Menon, and Masashi Sugiyama. Complementary-label learning for arbitrary losses and models. In *International Conference on Machine Learning*, pages 2971–2980, 2019.

Alireza Javanmardi, Yusuf Sale, Paul Hofman, and Eyke Hüllermeier. Conformal prediction with partially labeled data. In *Conformal and Probabilistic Prediction with Applications*, pages 251–266, 2023.

Rong Jin and Zoubin Ghahramani. Learning with multiple labels. In *Advances in Neural Information Processing Systems*, pages 897–904, 2002.

Michael I. Jordan, Zoubin Ghahramani, Tommi S. Jaakkola, and Lawrence K. Saul. An introduction to variational methods for graphical models. *Machine Learning*, 37(2): 183–233, 1999.

Audun Jøsang. *Subjective Logic - A Formalism for Reasoning Under Uncertainty.* Springer, Cham, 2016.

Mira Jürgens, Nis Meinert, Viktor Bengs, Eyke Hüllermeier, and Willem Waegeman. Is epistemic uncertainty faithfully represented by evidential deep learning methods? In *International Conference on Machine Learning*, 2024.

Alex Kendall and Yarin Gal. What uncertainties do we need in bayesian deep learning for computer vision? In *Advances in Neural Information Processing Systems*, pages 5574–5584, 2017.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *International Conference on Learning Representations*, 2014.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009. `http://www.cs.utoronto.ca/~kriz/learning-features-2009-TR.pdf`.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems*, pages 6402–6413, 2017.

Antonis Lambrou, Harris Papadopoulos, and Alex Gammerman. Reliable confidence measures for medical diagnosis with evolutionary algorithms. *Transactions on Information Technology in Biomedicine*, 15(1):93–99, 2011.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Yann LeCun, Corinna Cortes, and Christopher J. C. Burges. The MNIST database of handwritten digits, 1999. `http://yann.lecun.com/exdb/mnist`.

Jing Lei. Classification with confidence. *Biometrika*, 101(4):755–769, 2014.

Junnan Li, Dongxu Li, Silvio Savarese, and Steven C. H. Hoi. BLIP-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *International Conference on Machine Learning*, pages 19730–19742, 2023.

Jianzhe Lin, Tianze Yu, and Z. Jane Wang. Rethinking crowdsourcing annotation: Partial annotation with salient labels for multilabel aerial image classification. *IEEE Transactions on Geoscience and Remote Sensing*, 60:1–12, 2022.

Li-Ping Liu and Thomas G. Dietterich. A conditional multinomial mixture model for superset label learning. In *Advances in Neural Information Processing Systems*, pages 557–565, 2012.

Jiaqi Lv, Miao Xu, Lei Feng, Gang Niu, Xin Geng, and Masashi Sugiyama. Progressive identification of true labels for partial-label learning. In *International Conference on Machine Learning*, pages 6500–6510, 2020.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*, 2018.

Eberhard Mandler and Jürgen Schümann. Combining the classification results of independent classifiers based on the Dempster-Shafer theory of evidence. *Machine Intelligence and Pattern Recognition*, 7:381–393, 1988.

Anqi Mao, Mehryar Mohri, and Yutao Zhong. Predictor-rejector multi-class abstention: Theoretical analysis and algorithms. In *International Conference on Algorithmic Learning Theory*, pages 822–867, 2024.

Charles C. Margossian and David M. Blei. Amortized variational inference: When and why? In *Uncertainty in Artificial Intelligence*, pages 2434–2449, 2024.

Rhiannon Michelmore, Matthew Wicker, Luca Laurenti, Luca Cardelli, Yarin Gal, and Marta Kwiatkowska. Uncertainty quantification with statistical guarantees in end-to-end autonomous driving control. In *International Conference on Robotics and Automation*, pages 7344–7350, 2020.

Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. DeepFool: A simple and accurate method to fool deep neural networks. In *Conference on Computer Vision and Pattern Recognition*, pages 2574–2582, 2016.

Thomas Mortier, Viktor Bengs, Eyke Hüllermeier, Stijn Luca, and Willem Waegeman. On the calibration of probabilistic classifier sets. In *International Conference on Artificial Intelligence and Statistics*, volume 206, pages 8857–8870, 2023.

Hussein Mozannar, Hunter Lang, Dennis Wei, Prasanna Sattigeri, Subhro Das, and David A. Sontag. Who should predict? Exact algorithms for learning to defer to humans. In *International Conference on Artificial Intelligence and Statistics*, pages 10520–10545, 2023.

Michael Naaman. On the tight constant in the multivariate Dvoretzky-Kiefer-Wolfowitz inequality. *Statistics and Probability Letters*, 173:1–8, 2021.

Mahdi P. Naeini, Gregory F. Cooper, and Milos Hauskrecht. Obtaining well calibrated probabilities using Bayesian binning. In *AAAI Conference on Artificial Intelligence*, pages 2901–2907, 2015.

Harikrishna Narasimhan, Aditya Krishna Menon, Wittawat Jitkrittum, Neha Gupta, and Sanjiv Kumar. Learning to reject meets long-tail learning. In *International Conference on Learning Representations*, 2024.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011.

Nam Nguyen and Rich Caruana. Classification with partial labels. In *Conference on Knowledge Discovery and Data Mining*, pages 551–559, 2008.

Chenri Ni, Nontawat Charoenphakdee, Junya Honda, and Masashi Sugiyama. On the calibration of multiclass classification with rejection. In *Advances in Neural Information Processing Systems*, pages 2582–2592, 2019.

Peng Ni, Suyun Zhao, Zhi-Gang Dai, Hong Chen, and Cui-Ping Li. Partial label learning via conditional-label-aware disambiguation. *Journal of Computer Science and Technology*, 36(3):590–605, 2021.

Stephen M. Omohundro. *Five balltree construction algorithms*. Berkeley, 1989.

William D. Penny. Kullback-Leibler divergences of Normal, Gamma, Dirichlet and Wishart densities. Technical report, Wellcome Department of Cognitive Neurology, 2001.

Rajesh Ranganath, Sean Gerrish, and David M. Blei. Black box variational inference. In *International Conference on Artificial Intelligence and Statistics*, volume 33, pages 814–822, 2014.

Narathip Reamaroon, Michael W. Sjoding, Kaiwen Lin, Theodore J. Iwashyna, and Kayvan Najarian. Accounting for label uncertainty in machine learning for detection of acute respiratory distress syndrome. *Journal of Biomedical and Health Informatics*, 23(1): 407–415, 2019.

Danilo Jimenez Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *International Conference on Machine Learning*, volume 37, pages 1530–1538, 2015.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *International Conference on Machine Learning*, volume 32, pages 1278–1286, 2014.

David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.

Mauricio Sadinle, Jing Lei, and Larry A. Wasserman. Least ambiguous set-valued classifiers with bounded error levels. *Journal of the American Statistical Association*, 114(525):223–234, 2019.

Yusuf Sale, Michele Caprio, and Eyke Hüllermeier. Is the volume of a credal set a good measure for epistemic uncertainty? In *Uncertainty in Artificial Intelligence*, pages 1795–1804, 2023.

Murat Sensoy, Lance M. Kaplan, and Melih Kandemir. Evidential deep learning to quantify classification uncertainty. In *Advances in Neural Information Processing Systems*, pages 3183–3193, 2018.

Sina Shafaei, Stefan Kugele, Mohd Hafeez Osman, and Alois C. Knoll. Uncertainty in machine learning: A safety perspective on autonomous driving. In *Computer Safety, Reliability, and Security*, pages 458–464, 2018.

Glenn Shafer. The combination of evidence. *Intelligent Systems*, 1(3):155–179, 1986.

Shai Shalev-Shwartz, Yoram Singer, and Nathan Srebro. Pegasos: Primal Estimated sub-GrAdient SOlver for SVM. In *International Conference on Machine Learning*, pages 807–814, 2007.

Leslie N. Smith and Nicholay Topin. Super-convergence: Very fast training of neural networks using large learning rates. In *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications*, pages 369–386, 2019.

Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. In *Advances in Neural Information Processing Systems*, pages 3483–3491, 2015.

Ashwin Srinivasan. Statlog (landsat satellite). UCI Machine-Learning Repository, 1993. `https://doi.org/10.24432/C55887`.

Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *International Conference on Learning Representations*, 2014.

Mahdi Tabassian, Reza Ghaderi, and Reza Ebrahimpour. Combining complementary information sources in the Dempster-Shafer framework for solving classification problems with imperfect labels. *Knowledge Based Systems*, 27:92–102, 2012.

Cai-Zhi Tang and Min-Ling Zhang. Confidence-rated discriminative partial label learning. In *AAAI Conference on Artificial Intelligence*, pages 2611–2617, 2017.

Shiyu Tian, Hongxin Wei, Yiqun Wang, and Lei Feng. CroSel: Cross selection of confident pseudo labels for partial-label learning. In *Conference on Computer Vision and Pattern Recognition*, 2024.

Zheng Tong, Philippe Xu, and Thierry Denoeux. An evidential classifier based on Dempster-Shafer theory and deep learning. *Neurocomputing*, 450:275–293, 2021.

Alexander B. Tsybakov. Optimal aggregation of classifiers in statistical learning. *The Annals of Statistics*, 32(1):135–166, 2004.

Tim van Erven, Peter D. Grünwald, Nishant A. Mehta, Mark D. Reid, and Robert C. Williamson. Fast rates in statistical and online learning. *Journal of Machine Learning Research*, 16:1793–1861, 2015.

Kush R. Varshney and Homa Alemzadeh. On the safety of machine learning: Cyber-physical systems, decision sciences, and data products. *Big Data*, 5(3):246–255, 2017.

Thomas Verma and Judea Pearl. Equivalence and synthesis of causal models. In *Uncertainty in Artificial Intelligence*, pages 255–270, 1990.

Cheng Wang, Larry Han, Gabriella Stein, Suzanne Day, Cedric Bien-Gund, Allison Mathews, Jason J Ong, Pei-Zhen Zhao, Shu-Fang Wei, Jennifer Walker, et al. Crowdsourcing in health and medical research: a systematic review. *Infectious diseases of poverty*, 9(8), 2020.

Deng-Bao Wang, Li Li, and Min-Ling Zhang. Adaptive graph guided disambiguation for partial label learning. In *International Conference on Knowledge Discovery and Data Mining*, pages 83–91, 2019.

Haobo Wang, Ruixuan Xiao, Yixuan Li, Lei Feng, Gang Niu, Gang Chen, and Junbo Zhao. PiCO: Contrastive label disambiguation for partial label learning. In *International Conference on Learning Representations*, 2022.

Wei Wang and Min-Ling Zhang. Partial label learning with discrimination augmentation. In *Conference on Knowledge Discovery and Data Mining*, pages 1920–1928, 2022.

Paul Werbos. *Beyond regression: New tools for prediction and analysis in the behavioral sciences.* PhD thesis, Harvard University, 1974.

Lisa Wimmer, Yusuf Sale, Paul Hofman, Bernd Bischl, and Eyke Hüllermeier. Quantifying aleatoric and epistemic uncertainty in machine learning: Are conditional entropy and mutual information appropriate measures? In *Uncertainty in Artificial Intelligence*, pages 2282–2292, 2023.

Xuan Wu and Min-Ling Zhang. Towards enabling binary decomposition for partial label learning. In *International Joint Conference on Artificial Intelligence*, pages 2868–2874, 2018.

Sheng Xiang, Mingzhi Zhu, Dawei Cheng, Enxia Li, Ruihui Zhao, Yi Ouyang, Ling Chen, and Yefeng Zheng. Semi-supervised credit card fraud detection via attribute-driven graph representation. In *AAAI Conference on Artificial Intelligence*, pages 14557–14565, 2023.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms. *CoRR*, abs/1708.07747, 2018.

Ning Xu, Jiaqi Lv, and Xin Geng. Partial label learning via label enhancement. In *AAAI Conference on Artificial Intelligence*, pages 5557–5564, 2019.

Ning Xu, Congyu Qiao, Xin Geng, and Min-Ling Zhang. Instance-dependent partial label learning. In *Advances in Neural Information Processing Systems*, pages 27119–27130, 2021.

Ning Xu, Biao Liu, Jiaqi Lv, Congyu Qiao, and Xin Geng. Progressive purification for instance-dependent partial label learning. In *International Conference on Machine Learning*, pages 38551–38565, 2023.

Wenda Xu, Jia Pan, Junqing Wei, and John M. Dolan. Motion planning under uncertainty for on-road autonomous driving. In *International Conference on Robotics and Automation*, pages 2507–2512, 2014.

Ronald R. Yager. On the Dempster-Shafer framework and new combination rules. *Information Sciences*, 41(2):93–137, 1987a.

Ronald R. Yager. Quasi-associative operations in the combination of evidence. *Kybernetes*, 16(1):37–41, 1987b.

Fan Yang, Hua-zhen Wang, Hong Mi, Cheng-de Lin, and Wei-wen Cai. Using random forest for reliable classification and cost-sensitive learning for medical diagnosis. *BMC Bioinformatics*, 10(1), 2009.

Fuchao Yang, Jianhong Cheng, Hui Liu, Yongqiang Dong, Yuheng Jia, and Junhui Hou. Mixed blessing: Class-wise embedding guided instance-dependent partial label learning. In *Conference on Knowledge Discovery and Data Mining*, pages 1763–1772, 2025.

Fei Yu and Min-Ling Zhang. Maximum margin partial label learning. In *Asian Conference on Machine Learning*, pages 573–593, 2017.

Lotfi A. Zadeh. Book review: A mathematical theory of evidence. *AI Magazine*, 5(3):81–83, 1984.

Ahmed Zaoui, Christophe Denis, and Mohamed Hebiri. Regression with reject option and application to kNN. In *Advances in Neural Information Processing Systems*, 2020.

Zinan Zeng, Shijie Xiao, Kui Jia, Tsung-Han Chan, Shenghua Gao, Dong Xu, and Yi Ma. Learning by associating ambiguously labeled images. In *Conference on Computer Vision and Pattern Recognition*, pages 708–715, 2013.

Fei Zhang, Lei Feng, Bo Han, Tongliang Liu, Gang Niu, Tao Qin, and Masashi Sugiyama. Exploiting class activation value for partial-label learning. In *International Conference on Learning Representations*, 2022a.

Min-Ling Zhang and Fei Yu. Solving the partial label learning problem: An instance-based approach. In *International Joint Conference on Artificial Intelligence*, pages 4048–4054, 2015.

Min-Ling Zhang, Bin-Bin Zhou, and Xu-Ying Liu. Partial label learning via feature-aware disambiguation. In *International Conference on Knowledge Discovery and Data Mining*, pages 1335–1344, 2016.

Min-Ling Zhang, Fei Yu, and Cai-Zhi Tang. Disambiguation-free partial label learning. *Transactions on Knowledge and Data Engineering*, 29(10):2155–2167, 2017.

Min-Ling Zhang, Jing-Han Wu, and Wei-Xuan Bao. Disambiguation enabled linear discriminant analysis for partial label dimensionality reduction. *Transactions on Knowledge Discovery from Data*, 16(4):72:1–72:18, 2022b.

Yikai Zhang, Songzhu Zheng, Pengxiang Wu, Mayank Goswami, and Chao Chen. Learning with feature-dependent label noise: A progressive approach. In *International Conference on Learning Representations*, 2021.

Zsolt Zombori and Balázs Indruck. Partial label learning for automated theorem proving. *CoRR*, abs/2507.03314, 2025.

A. Appendix to PLL with a Reject Option

This section complements Chapter 4 and contains additional experiments and details their hyperparameters. Section A.1 lists the values of all relevant hyperparameters, the parameter sensitivity of our approach is discussed in Section A.2, and Section A.3 contains reject trade-off plots for all experimental settings considered.

A.1. Hyperparameters

As mentioned in Section 4.4.1, we consider ten commonly used PLL approaches. We choose their parameters as recommended by the respective authors.

- PlKnn (Hüllermeier and Beringer, 2005): For all non-*mnist* datasets, we use $l = 10$ neighbors as recommended by the authors. For the *mnist*-like datasets, we use the hidden representation of a variational auto-encoder as instance features and use $l = 20$. The variational auto-encoder has a 768-dimensional input layer (flat *mnist* input), a 512-dimensional second layer, and 48-dimensional bottleneck layers for the mean and variance representations. The decoder uses a 48-dimensional first layer, a 512-dimensional second layer, and a 768-dimensional output layer with sigmoid activation. Otherwise, we use ReLU activations between all layers. Binary cross-entropy is used as a reconstruction loss. We choose the *AdamW* optimizer for training.
- PlSvm (Nguyen and Caruana, 2008): We use the Pegasos optimizer (Shalev-Shwartz et al., 2007) and $\lambda = 1$.
- Ipal (Zhang and Yu, 2015): We use $l = 10$ neighbors, $\alpha = 0.95$, and 100 iterations.
- PlEcoc (Zhang et al., 2017): We use $L = \lceil 10 \log_2(k) \rceil$ and $\tau = 0.1$ as recommended.
- Proden (Lv et al., 2020): For a fair comparison, we use the same base models for all neural-network-based approaches. We use a standard d-300-300-300-k MLP (Werbos, 1974) for the non-*mnist* datasets with ReLU activations, batch normalizations, and softmax output. For the *mnist*-like datasets, we use the LeNet-5 architecture (LeCun et al., 1998). We choose the *AdamW* optimizer for training.
- Cc (Feng et al., 2020): We use the same base models as mentioned above for Proden.
- Valen (Xu et al., 2021): We use the same base models as mentioned above for Proden.

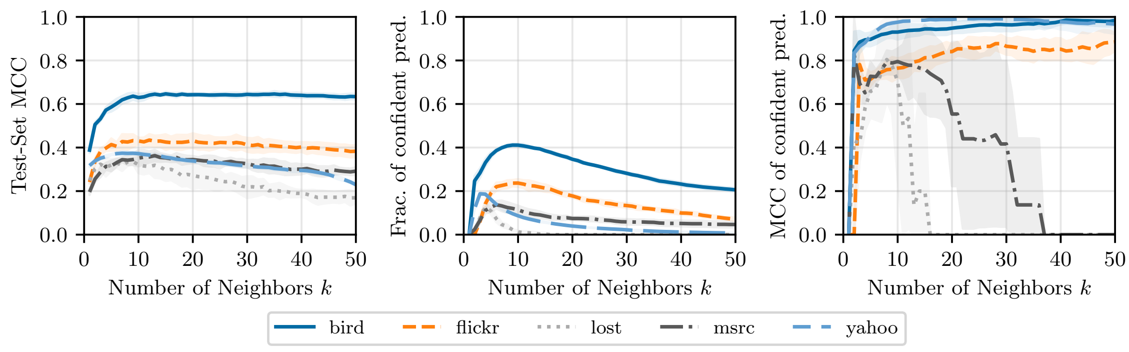


Figure A.1.: Sensitivity of the number of neighbors regarding the test-set MCC score, the fraction, and the MCC score of confident / non-rejected predictions.

- Pop (Xu et al., 2023): We use the same base models as mentioned above for Proden. Also, we set $e_0 = 0.001$, $e_{end} = 0.04$, and $e_s = 0.001$. We abstain from using the data augmentations discussed in the paper for a fair comparison.
- CroSel (Tian et al., 2024): We use the same base models as mentioned above for Proden. We use 10 warm-up epochs using Cc and $\lambda_{cr} = 2$. We abstain from using the data augmentations discussed in the paper for a fair comparison.
- DstPll (our proposed approach): Similar to PlKnn and Ipal, we use $l = 10$ neighbors for the non-*mnist* datasets. For the *mnist*-like datasets, we use the hidden representation of a variational auto-encoder as instance features and use $l = 20$. The architecture of the variational auto-encoder is the same as described above for PlKnn.

We have implemented all approaches in Python using Pytorch. All experiments need two to three days on a machine with 48 cores and one NVIDIA GeForce RTX 3090.

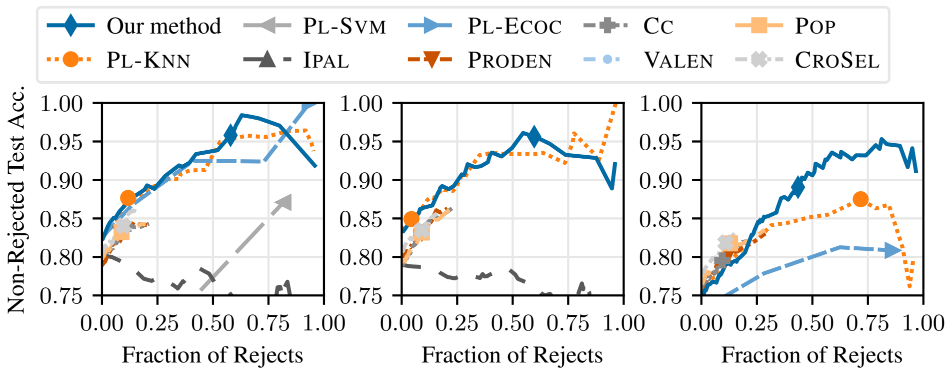
A.2. Parameter Sensitivity

Figure A.1 shows the sensitivity of the number of neighbors l regarding the test-set performance, the fraction of confident / non-rejected predictions, and the non-rejected prediction performance. The shaded areas indicate the standard deviation regarding the 5-fold cross-validation. As for default l-nearest neighbor classification, changes of l have a relatively large impact. We show parameter sensitivity for each of the real-world datasets separately. Naturally, different datasets have different optimal parameter settings. The configuration $l = 10$, which is also recommended within PlKnn (Hüllermeier and Beringer, 2005) and Ipal (Zhang and Yu, 2015), provides a good trade-off between the number of confident predictions and how accurate confident predictions are. Indeed, this setting produces a good number of confident predictions on most datasets (Figure A.1; center plot). At the same time, it produces a good MCC performance of confident predictions on most datasets (Figure A.1; right plot).

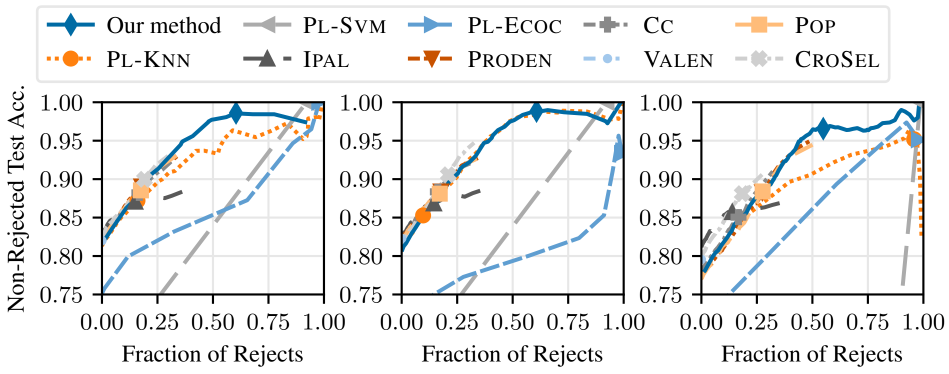
When increasing l, our method's behavior on different datasets can be assembled into two groups. On the *bird-song*, *mir-flickr*, and *yahoo-news* datasets, increasing l past ten neighbors also increases the amount of irrelevant labeling information from those neighbors. Therefore, our approach produces less confident predictions. At the same time, the MCC score of confident predictions remains at roughly the same level. This is because irrelevant labeling information from neighbors increases at most at the same rate as l. In contrast, on the *lost* and *msrc-v2* datasets, the MCC score of confident predictions drops sharply at a certain point while the number of confident predictions decreases similarly. This is because irrelevant labeling information increases more rapidly than l: The decrease of confidence in predictions is slower than the increase of irrelevant candidate labels.

A.3. Reject Trade-off Curves

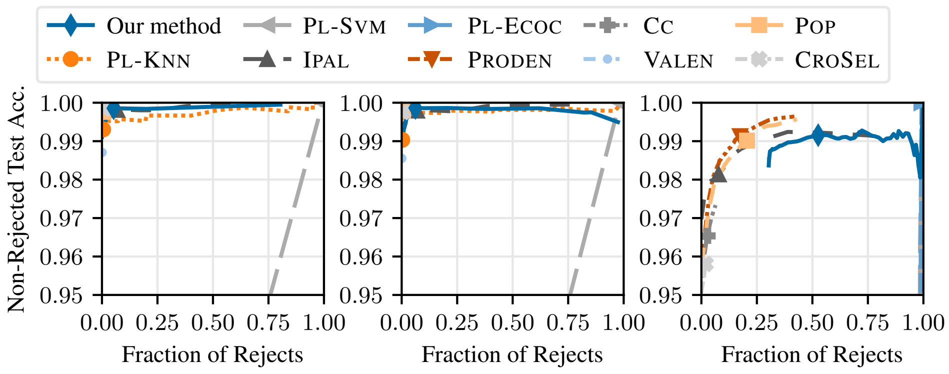
Figure A.2 (i) – (xxxiv) shows the reject trade-off for varying confidence (0 to 1) and $\Delta_{\tilde{m}}$ (-1 to 1) thresholds and augments Figure 4.1 by considering all datasets and noise generation strategies. The x-axes show the fractions of predictions that are rejected. The y-axes show the accuracies of predictions that are not rejected. The plots show (fraction of rejects, non-rejected test-set accuracy)-pairs corresponding to different settings of the thresholds. In most cases, our method provides a better trade-off between the number of rejected predictions and the accuracy of the non-rejected predictions. Table 4.2 summarizes all plots by showing the average empirical risks across all experimental settings and for different trade-off parameters λ. We recall that our method provides the significantly best trade-offs for $\lambda \in [0, 0.2]$.



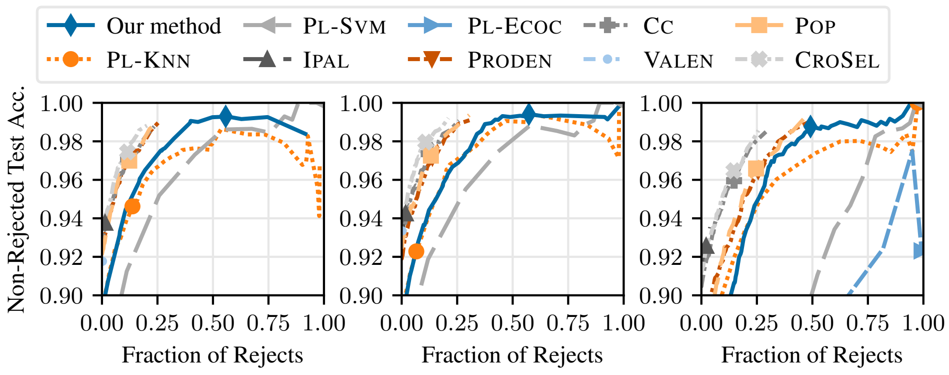
(i) *ecoli* (uniform noise) (ii) *ecoli* (class-dep. noise) (iii) *ecoli* (inst.-dep. noise)



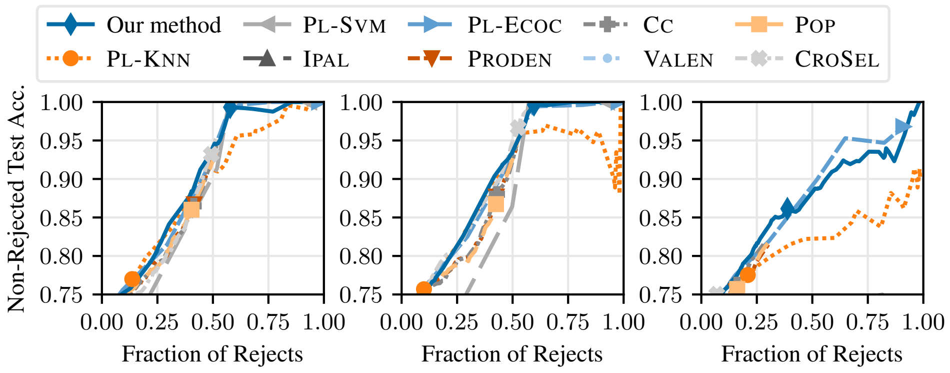
(iv) *m.-feats* (uniform noise) (v) *m.-feats* (class-dep. noise) (vi) *m.-feats* (inst.-dep. noise)



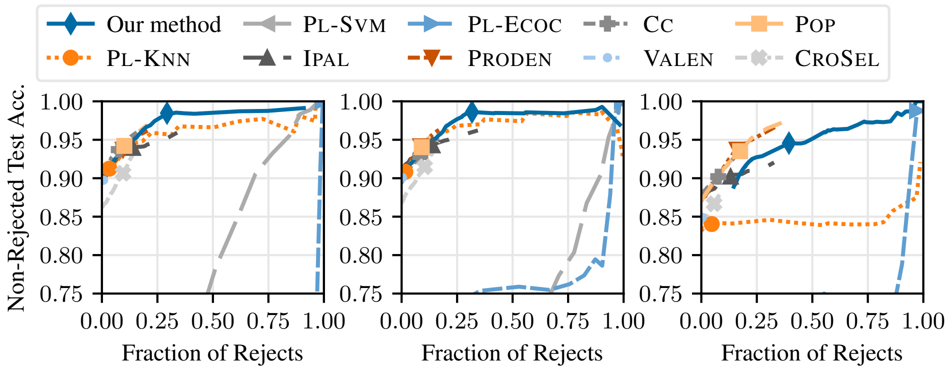
(vii) *p.-digits* (uniform noise) (viii) *p.-digits* (class-dep. noise) (ix) *p.-digits* (inst.-dep. noise)



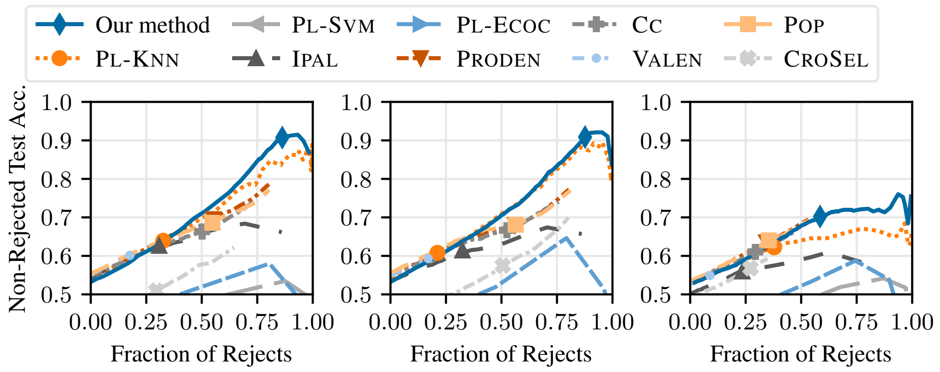
(x) *semeion* (uniform noise) (xi) *semeion* (class-dep. noise) (xii) *semeion* (inst.-dep. noise)



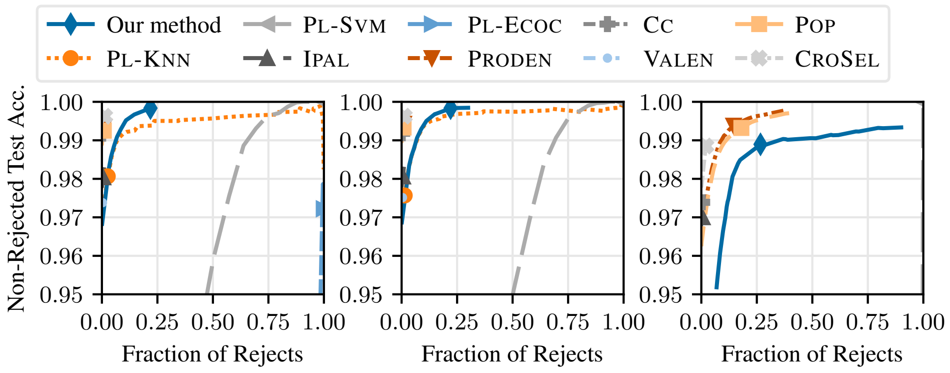
(xiii) *s.-flare* (uniform noise) (xiv) *s.-flare* (class-dep. noise) (xv) *s.-flare* (inst.-dep. noise)



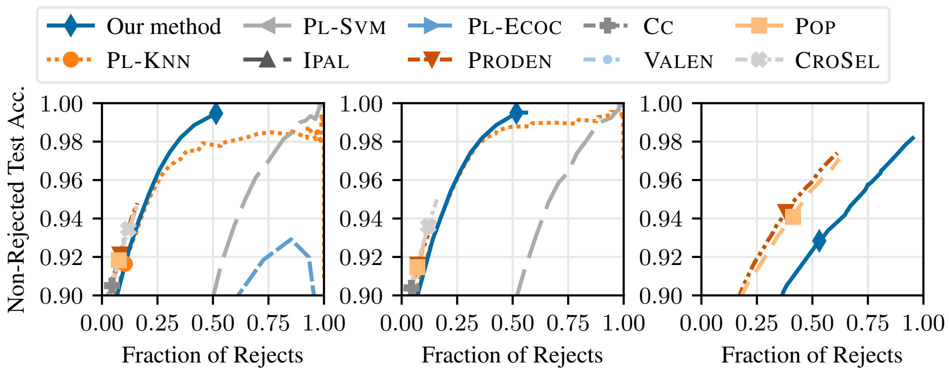
(xvi) *landsat* (uniform noise) (xvii) *landsat* (class-dep. noise) (xviii) *landsat* (inst.-dep. noise)



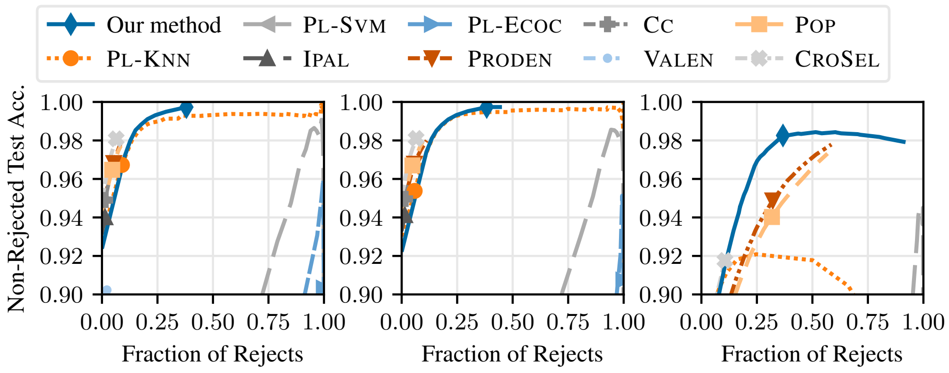
(xix) *theorem* (uniform noise) (xx) *theorem* (class-dep. noise) (xxi) *theorem* (inst.-dep. noise)



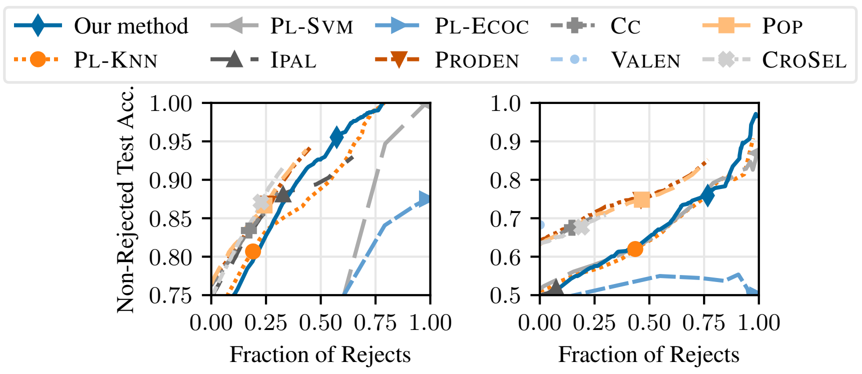
(xxii) *mnist* (uniform noise) (xxiii) *mnist* (class-dep. noise) (xxiv) *mnist* (inst.-dep. noise)



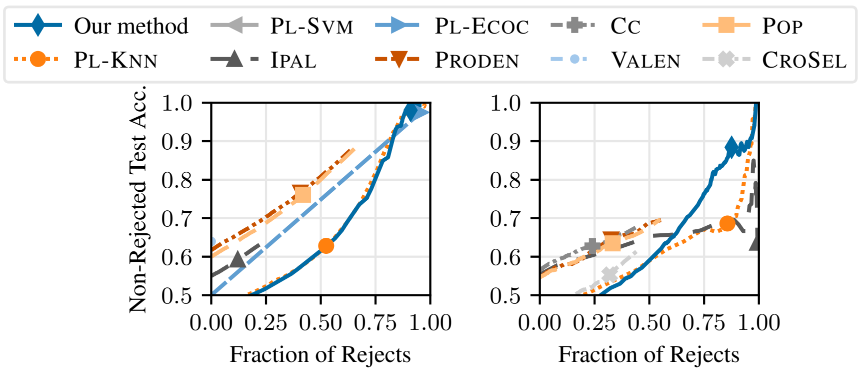
(xxv) *fmnist* (uniform noise) (xxvi) *fmnist* (class-dep. noise) (xxvii) *fmnist* (inst.-dep. noise)



(xxviii) *kmnist* (uniform noise) (xxix) *kmnist* (class-dep. noise) (xxx) *kmnist* (inst.-dep. noise)



(xxxi) *bird-song* (xxxii) *mir-flickr*



(xxxiii) *yahoo-news* (xxxiv) *msrc-v2*

Figure A.2.: Trade-off between the fraction of rejected and the accuracy of non-rejected predictions.

B. Appendix to Robust PLL by Leveraging Class Activation Values

This section augments Section 5.4 by presenting more details of our experimental setup and results. This includes the hyperparameter values of all methods (Section B.1), an overview of the datasets used (Section B.2), as well as further results (Section B.3 and B.4).

B.1. Hyperparameters

This section lists all methods, which are benchmarked in the main text, together with their respective hyperparameter choices. We set all hyperparameters as recommended by the respective authors. There are 14 non-ensemble and four ensemble methods. The non-ensemble methods and their hyperparameters are:

- PlKnn (Hüllermeier and Beringer, 2005): We use $k = 10$ nearest neighbors.
- PlSvm (Nguyen and Caruana, 2008): We use the Pegasos optimizer (Shalev-Shwartz et al., 2007) and $\lambda = 1$.
- Ipal (Zhang and Yu, 2015): We use $k = 10$ neighbors, $\alpha = 0.95$, and 100 iterations.
- PlEcoc (Zhang et al., 2017): We use $L = \lceil 10 \log_2(l) \rceil$ and $\tau = 0.1$.
- Proden (Lv et al., 2020): For a fair comparison, we use the same base model for all neural-network-based approaches. We use a standard d-300-300-300-l MLP (Werbos, 1974) with ReLU activations, batch normalizations, and softmax output. We choose the *Adam* optimizer for training and train for a total of 200 epochs.
- Proden+L2 (Hoerl and Kennard, 1970; Lv et al., 2020): We use the same base model and settings as Proden with additional L2 weight regularization.
- Proden+Edl (Sensoy et al., 2018; Lv et al., 2020): We use the Proden model to disambiguate the candidate labels with the same settings as above. Then, we use the evidential-learning algorithm by Sensoy et al. (2018) in a supervised manner.
- Rc (Feng et al., 2020): We use the same base model and settings as Proden.
- Cc (Feng et al., 2020): We use the same base model and settings as Proden.
- Valen (Xu et al., 2021): We use the same base model and settings as Proden.

Table B.1.: Overview of dataset characteristics grouped into real-world partially labeled datasets (top) and supervised datasets with added candidate labels (bottom).

Dataset	#Inst. n	#Features d	#Classes k	Avg. candidate set sizes	Fraction with ground truth
bird-song	4 998	38	13	2.167	0.33
lost	1 122	108	16	2.228	0.06
mir-flickr	2 780	1 536	14	2.764	0.00
msrc-v2	1 758	48	23	3.154	0.08
soccer	17 472	279	171	2.095	0.30
yahoo-news	22 991	163	219	1.907	0.29
mnist	70 000	784	10	6.304	0.00
fmnist	70 000	784	10	5.953	0.00
kmnist	70 000	784	10	6.342	0.00
not-mnist	70 000	784	10	6.342	0.00

- Cavl (Zhang et al., 2022a): We use the same base model and settings as Proden.
- Pop (Xu et al., 2023): We use the same base model and settings as Proden. Also, we set $e_0 = 0.001$, $e_{end} = 0.04$, and $e_s = 0.001$.
- CroSel (Tian et al., 2024): We use the same base model and settings as Proden. We use 10 warm-up epochs using Cc and $\lambda_{cr} = 2$. We abstain from using the data augmentations discussed in the paper for a fair comparison of the base approach.
- DstPll (Fuchs et al., 2025): We use $k = 20$ neighbors and a variational auto-encoder to reduce the feature dimensionality as recommended by the authors.
- RobustPll (our method): We use the same base model and settings as Proden. The parameter λ_t is set to $\min(2t/T, 1)$ with $T = 200$ epochs.

The four ensemble methods and their hyperparameters are:

- Proden+Dropout (Srivastava et al., 2014; Lv et al., 2020): We use the Proden model with additional Monte-Carlo dropout. The dropout layer is also active during inference. We repeat the predictions 1000 times to estimate the uncertainty involved across all dropout networks.
- Proden+Ens (Lakshminarayanan et al., 2017; Lv et al., 2020): We use an ensemble of 5 Proden models.
- Proden+AdvEns (Lv et al., 2020; Lakshminarayanan et al., 2017): We use an ensemble of 5 Proden models that are trained on adversarially corrupted instance features.
- RobustPll+Ens (our method): We use an ensemble of 5 RobustPll models.

Table B.2.: Average test-set accuracies (± std.) on the real-world datasets.

All methods	*bird-song*	*lost*	*mir-flickr*	*msrc-v2*	*soccer*	*yahoo-news*
PlKnn (2005)	67.8 (± 1.2)	41.6 (± 2.2)	52.6 (± 2.0)	43.6 (± 1.9)	50.0 (± 0.4)	46.1 (± 0.5)
PlSvm (2008)	32.3 (± 1.2)	53.5 (± 1.7)	45.4 (± 7.2)	26.5 (± 0.9)	49.3 (± 0.5)	37.2 (± 2.7)
Ipal (2015)	72.1 (± 0.9)	59.4 (± 4.1)	54.3 (± 1.0)	51.8 (± 2.1)	54.1 (± 0.5)	55.3 (± 0.7)
PlEcoc (2017)	58.3 (± 2.5)	64.1 (± 2.3)	49.2 (± 2.7)	30.6 (± 6.0)	5.6 (± 0.4)	48.1 (± 0.7)
Proden (2020)	72.0 (± 0.8)	**71.5** (± 2.9)	**68.4** (± 1.8)	**54.7** (± 1.2)	54.4 (± 0.5)	**63.7** (± 1.0)
Proden+L2	72.2 (± 0.9)	**71.5** (± 2.0)	**67.8** (± 2.0)	**54.4** (± 1.2)	54.4 (± 0.2)	**64.1** (± 0.8)
Proden+Edl	69.1 (± 0.4)	66.6 (± 2.8)	**66.4** (± 2.6)	**53.4** (± 1.7)	25.7 (± 1.2)	16.6 (± 0.9)
Rc (2020)	**73.7** (± 1.1)	**69.8** (± 0.7)	67.3 (± 2.0)	52.9 (± 1.5)	49.9 (± 0.5)	59.1 (± 0.5)
Cc (2020)	71.8 (± 0.7)	**71.1** (± 3.1)	65.1 (± 2.0)	**53.8** (± 1.3)	0.6 (± 0.2)	0.5 (± 0.2)
Valen (2021)	66.5 (± 2.2)	45.9 (± 5.3)	60.0 (± 2.6)	41.9 (± 1.1)	49.6 (± 0.4)	59.0 (± 1.4)
Cavl (2022)	69.4 (± 1.5)	64.7 (± 2.4)	**63.8** (± 3.6)	50.9 (± 0.7)	**54.7** (± 0.7)	**65.1** (± 0.6)
Pop (2023)	72.5 (± 0.8)	**71.6** (± 2.0)	**67.7** (± 1.6)	**53.9** (± 1.8)	**55.3** (± 0.7)	63.5 (± 0.7)
CroSel (2024)	69.5 (± 0.9)	**67.8** (± 4.5)	64.0 (± 2.0)	49.4 (± 1.5)	0.6 (± 0.2)	0.3 (± 0.2)
DstPll (2024)	67.2 (± 0.9)	37.9 (± 3.2)	50.6 (± 1.4)	41.0 (± 1.8)	49.9 (± 0.5)	44.5 (± 0.4)
► RobustPll	67.9 (± 2.5)	65.4 (± 1.5)	63.9 (± 2.0)	52.0 (± 1.0)	50.6 (± 0.6)	57.2 (± 1.0)
Proden+Dropout	72.5 (± 0.8)	73.2 (± 2.1)	**69.2** (± 2.0)	**54.2** (± 1.4)	54.0 (± 0.5)	66.7 (± 1.0)
Proden+Ens	73.5 (± 0.8)	**75.5** (± 1.9)	**68.2** (± 2.0)	**54.9** (± 2.2)	54.9 (± 0.7)	68.2 (± 0.5)
Proden+AdvEns	**74.7** (± 0.6)	**77.4** (± 3.4)	**67.0** (± 1.4)	53.9 (± 2.1)	**56.2** (± 0.5)	**71.0** (± 0.5)
► RobustPll+Ens	71.8 (± 1.0)	71.9 (± 3.2)	**66.8** (± 2.2)	**53.6** (± 0.8)	53.7 (± 0.5)	63.7 (± 0.3)

B.2. Datasets

Table B.1 shows an overview of all used datasets, including the number of instances, features, and classes. Also, we report the average candidate set sizes as well as the fraction of candidate sets with only one candidate label, which is the ground truth label. We note that five out of ten datasets do not contain a single instance with available ground truth. We recall that this prohibits the application of algorithms from related fields, for example, semi-supervised learning.

B.3. Predictive Performance

Table B.2 augments Table 5.1 and shows the detailed test-set accuracies across all real-world datasets. All experiments are repeated five times with different seeds to report mean and standard deviations. We emphasize the best algorithm per dataset, as well as non-significant differences, using a student t-test with level $\alpha = 0.05$. We consider non-ensemble and ensemble methods separately. Our proposed algorithms, which are indicated by the triangles, perform comparably on all considered datasets.

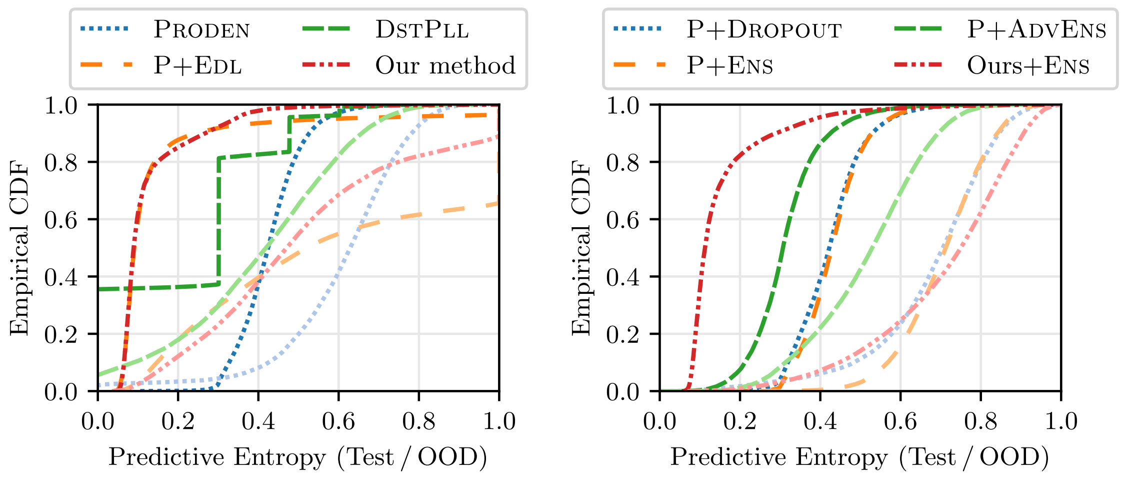


Figure B.1.: Empirical CDF of the normalized entropy (range 0 to 1) of predictions on MNIST (darker color) and NotMNIST (lighter color) for models trained on MNIST. The left plot shows the four best non-ensemble approaches according to Table 5.2 (highest metrics). We exclude methods that are too similar, for example, PRODEN-L2 and RC behave similarly to PRODEN, which is shown. All methods' performances can be observed in Table 5.2. The right plot shows the predictive entropy of all four ensemble approaches. Our ensemble approach is most certain about predictions on the test set (top-left corner) while being one of the approaches that is the most uncertain about out-of-distribution examples (bottom-right corner).

B.4. Adversarial Perturbations

To complement Table 5.2 in the main text, Figure B.1 provides the empirical cumulative distribution functions on the test and OOD set of the four best non-ensemble methods (regarding Table 5.2) on the left and of the four ensemble approaches on the right. The empirical CDFs of the entropies are normalized to a range between zero and one. The dark-colored lines represent the entropy CDFs of the predictions on the MNIST test set. The light-colored lines represent the entropy CDFs of the predictions on the NotMNIST test set (OOD). The left plot shows the four best non-ensemble approaches according to Table 5.2 (highest metrics). We exclude methods that are too similar, for example, PRODEN-L2 and RC behave similarly to PRODEN, which is shown. All methods' performances can be observed in Table 5.2. The right plot shows the predictive entropy of all four ensemble approaches. Our ensemble approach is most certain about predictions on the test set (top-left corner) while being one of the approaches that is the most uncertain about out-of-distribution examples (bottom-right corner).

C. Appendix to PLL with Conformal Candidate Cleaning

Appendix C.1 and C.2 contain supplementary material for our proofs, Appendix C.3 lists all hyperparameters used within our experiments in detail, and Appendix C.4 contains additional experiments.

C.1. Auxiliary Results

This section collects our auxiliary results.

Lemma C.1.1. *Let $\varepsilon \in (0, 1)$ and $f : [\varepsilon, 1] \rightarrow [0, -\log \varepsilon]$, $z \mapsto -\log z$. Then, f is $\frac{1}{\varepsilon}$-bi-Lipschitz, that is, for any $x_1, x_2 \in [\varepsilon, 1]$, it holds that $\varepsilon|x_1 - x_2| \leq |f(x_1) - f(x_2)| \leq \frac{1}{\varepsilon}|x_1 - x_2|$.*

Proof. f is continuous on $[\varepsilon, 1]$ and differentiable on $(\varepsilon, 1)$. Hence, by the mean value theorem, for any $x_1, x_2 \in [\varepsilon, 1]$, there exists $\xi \in (x_1, x_2)$ such that

$$|f(x_1) - f(x_2)| = |x_1 - x_2| \, |f'(\xi)| \, .$$

Using that $|f'(\xi)| = \frac{1}{\xi}$ satisfies $\varepsilon \leq f'(\xi) \leq \frac{1}{\varepsilon}$ as $\varepsilon \leq \xi \leq 1$ yields the stated stated claim. □

C.2. External Results

This section briefly summarizes external results that are necessary to prove our theorems. Theorem C.2.1 states the Dvoretzky-Kiefer-Wolfowitz inequality, Assumption C.2.2 describes the candidate generation model used in Theorem C.2.3, which relates the PLL risk (2.4) to the risk in the supervised setting. Theorem C.2.4 provides the estimation-error bound on which we build in our Lemma 6.3.3. We recall Markov's inequality in Lemma C.2.5.

Theorem C.2.1 (Dvoretzky et al. 1956; Naaman 2021, Dvoretzky-Kiefer-Wolfowitz Inequality). *Let $(\Omega, \mathcal{F}, \mathbb{P})$ be a probability space and $X, X_1, \dots, X_n \overset{i.i.d.}{\sim} \mathbb{P}$ real-valued random variables on Ω. Then, for any $\delta \in (0, 1)$,*

$$\mathbb{P}_X\left(\sup_{x \in \mathbb{R}} \left|\hat{F}_X(x) - F_X(x)\right| \leq \sqrt{\frac{\log(2/\delta)}{2n}}\right) \geq 1 - \delta,$$

with $\hat{F}_X(x) = \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}_{\{X_i \leq x\}}$ *and* $F_X(x) = \mathbb{P}(X \leq x)$.

Assumption C.2.2 (Feng et al. 2020, Eq. (5))**.** In the PLL setting (Chapter 2), assume that $\mathbb{P}_{XS}$ and $\mathbb{P}_{XY}$ have Lebesgue densities p_{XS} and p_{XY}, respectively, $p_{S|X,Y}(S) = p_{S|Y}(S)$, and the candidate generation model is of the form

$$p_{XS}(X, S) = \sum_{y=1}^{k} p_{S|Y=y}(S) p_{XY}(X, Y = y), \quad \text{with} \quad p_{S|Y=y}(S) = \begin{cases} \frac{1}{2^{k-1}-1} & \text{if } y \in S, \\ 0 & \text{else.} \end{cases}$$

The following theorem collects an identity by Feng et al. (2020).

Theorem C.2.3 (Feng et al. 2020, Eq. (6), (7), and (8))**.** *Let Assumption C.2.2 hold,* $R(f)$ *as in* (2.4)*, and the true risk of the supervised classification setting* $R_{\text{sup}}(f) := \mathbb{E}_{XY}[\ell(f(X), Y)]$. *Then,* $R_{\text{sup}}(f) = \frac{1}{2}R(f)$.

Theorem C.2.4 (Feng et al. 2020, Theorem 4)**.** *Let* $\ell : [0,1]^k \times \mathcal{Y} \to [0, M]$ *be a bounded and* λ*-Lipschitz loss function in the first argument (*$\lambda > 0$*), that is,* $\sup_{y \in \mathcal{Y}} |\ell(\mathbf{p}, y) - \ell(\mathbf{q}, y)| \leq \lambda \|\mathbf{p} - \mathbf{q}\|_2$ *for* $\mathbf{p}, \mathbf{q} \in [0,1]^k$. *Further, let* $\mathcal{H} = \{f : \mathcal{X} \to [0,1]^k \mid f$ *measurable,* $\forall x \in \mathcal{X} : \sum_{j=1}^{k} f_j(x) = 1\}$, $f^* = \arg\min_{f \in \mathcal{H}} R(f)$ *be the true risk minimizer and* $\hat{f} = \arg\min_{f \in \mathcal{H}} \hat{R}(f)$ *be the empirical risk minimizer of the risks in* (2.4) *and* (2.5)*, respectively. Then, for any* $\delta \in (0,1)$*, with* $\mathbb{P}_n$*-probability of at least* $1 - \delta$,

$$R(\hat{f}) - R(f^*) \leq 4\sqrt{2}\lambda \sum_{y=1}^{k} \mathfrak{R}_n(\mathcal{H}_y) + C\sqrt{\frac{\log(2/\delta)}{2n}},$$

where $\mathfrak{R}_n(\mathcal{H}_y)$ *is the empirical Rademacher complexity of* $\mathcal{H}_y := \{f_y \mid f \in \mathcal{H}\}$ *and some constant* $C > 0$. *Further, using that* $\mathfrak{R}_n(\mathcal{H}_y) \leq C_{\mathcal{H}}/\sqrt{n}$ *for some constants* $C_{\mathcal{H}}, M > 0$*, it holds with the same probability that*

$$R(\hat{f}) - R(f^*) \leq M\sqrt{\frac{\log(1/\delta)}{n}}.$$

Lemma C.2.5 (Markov inequality)**.** *For a real-valued random variable* X *with probability distribution* $\mathbb{P}$ *and* $a > 0$*, it holds that*

$$\mathbb{P}(|X| \geq a) \leq \frac{\mathbb{E}(|X|)}{a}.$$

C.3. Additional Setup

In our experiments, we consider twelve datasets of which Table C.1 summarizes the characteristics. As mentioned in Section 6.4.1, we consider six state-of-the-art PLL approaches and our novel candidate cleaning technique. We choose their parameters as recommended by the respective authors.

Table C.1.: Overview of dataset characteristics grouped into real-world partially labeled datasets (top) and supervised multi-class classification datasets with added candidate labels (bottom).

Dataset	#Instances n	#Features d	#Classes k	Avg. candidates
bird-song	4 966	38	12	2.146
lost	1 122	108	14	2.216
mir-flickr	2 778	1 536	12	2.756
msrc-v2	1 755	48	22	3.149
soccer	17 271	279	158	2.095
yahoo-news	22 762	163	203	1.915
mnist	70 000	784	10	6.304
fmnist	70 000	784	10	5.953
kmnist	70 000	784	10	6.342
svhn	99 289	3 072	10	4.878
cifar10	60 000	3 072	10	1.900
cifar100	60 000	3 072	100	1.399

- Proden (Lv et al., 2020): For a fair comparison, we use the same base models for each particular dataset. For the colored-image datasets, we use a ResNet-9 architecture (He et al., 2016). For all other image and non-image datasets, we use a standard d-300-300-300-k MLP (Werbos, 1974) with batch normalization (Ioffe and Szegedy, 2015) and ReLU activations (Glorot et al., 2011). We choose the *Adam* optimizer for training over a total of 200 epochs and use the one-cycle learning rate scheduler (Smith and Topin, 2019). Also, we use mini-batched training with a batch size of 16 for the small-scale datasets (less than 5000 samples) and of 256 for the large-scale datasets (more than 5000 samples). This balances training duration and predictive quality.

- Cc (Feng et al., 2020): We use the same base models and training procedures as mentioned above for Proden. Otherwise, there are no additional hyperparameters for Cc.

- Valen (Xu et al., 2021): We use the same base models and training procedures as mentioned above for Proden. Additionally, we use ten warm-up epochs and the three nearest neighbors to calculate the adjacency matrix.

- Cavl (Zhang et al., 2022a): We use the same base models and training procedures as mentioned above for Proden. Otherwise, there are no additional hyperparameters for Cavl.

- Pop (Xu et al., 2023): We use the same base models and training procedures as mentioned above for Proden. Also, we set $e_0 = 0.001$, $e_{end} = 0.04$, and $e_s = 0.001$. We abstain from using the data augmentations discussed in the paper for a fair comparison.

- CroSel (Tian et al., 2024): We use the same base models and training procedures as mentioned above for Proden. We use 10 warm-up epochs using Cc and $\lambda_{cr} = 2$. We abstain from using the data augmentations discussed in the paper for a fair comparison.
- Conf+Other method (our proposed approach): Our conformal candidate cleaning technique uses the same base models and training procedures as mentioned above for Proden. We use $R_{\text{warmup}} = 10$ warm-up epochs, a validation set size of 20 %, and $\alpha_r = \frac{1}{n'} \sum_{i=1}^{n'} \sum_{j \notin s_i} f_j(x_i)$. Otherwise, we use one of the given PLL classifiers for prediction-making.

We have implemented all approaches in Python using the Pytorch library. Running all experiments requires approximately three days on a machine with 48 cores and one NVIDIA GeForce RTX 4090. All our source code and data are available at `https://github.com/mathefuchs/pll-with-conformal-candidate-cleaning`.

C.4. Additional Experiments

In addition to our cleaning method (Conf), we also benchmark the existing cleaning method Clsp (He et al., 2024) on all datasets in Table C.2, C.3, and C.4, similar to the experiments in Section 6.4. Instead of training a ResNet-9 base model from scratch as done in Section 6.4.1, we use the pre-trained Blip-2 model (Li et al., 2023) for the experiments in Table C.3 below. We repeat all experiments five times and report means and standard deviations.

We observe that the Clsp models perform well on the image datasets (e.g., *cifar100*) but poorly on the real-world tabular PLL datasets shown in Table C.2. We attribute this to the fact that Clsp relies on the latent representation of large-scale vision models. In contrast, our method Conf gives strong results on, both, real-world and image data. This hypothesis is supported by Table C.4: The approaches Conf+Proden, Conf+Pop, and Conf+CroSel that combine the respective approaches with our candidate cleaning strategy win most frequently.

Table C.2.: Average test-set accuracies (std.) on the real-world datasets. We benchmark our strategy (Conf+) as well as the cleaning method Clsp combined with all existing methods.

Method	*bird-song*	*lost*	*mir-flickr*	*msrc-v2*	*soccer*	*yahoo-news*
Proden (2020)	75.55 (1.08)	78.94 (3.01)	67.05 (1.18)	54.33 (1.76)	54.18 (0.55)	65.25 (1.00)
Clsp+Proden	74.61 (0.84)	61.95 (2.80)	60.53 (2.95)	51.74 (1.77)	31.93 (29.15)	50.92 (0.66)
Conf+P. (no)	76.27 (0.94)	79.56 (1.96)	66.07 (1.63)	53.00 (2.24)	54.63 (0.81)	65.42 (0.36)
Conf+Proden	76.99 (0.90)	80.09 (4.40)	66.91 (1.57)	54.60 (3.42)	54.77 (0.84)	65.93 (0.42)
Cc (2020)	74.49 (1.57)	78.23 (2.11)	62.39 (1.87)	50.96 (2.03)	55.28 (0.96)	65.03 (0.51)
Clsp+Cc	74.37 (0.91)	60.88 (3.71)	59.79 (2.29)	49.64 (2.06)	53.71 (0.99)	49.89 (0.30)
Conf+Cc	75.01 (1.84)	79.38 (1.79)	63.37 (0.45)	52.45 (3.64)	55.52 (0.74)	64.35 (0.64)
Valen (2021)	72.30 (1.83)	70.18 (3.44)	67.05 (1.48)	49.20 (1.37)	53.20 (0.88)	62.25 (0.45)
Clsp+Valen	74.95 (0.27)	59.03 (2.67)	60.11 (1.95)	49.92 (1.80)	53.31 (0.84)	49.50 (0.76)
Conf+Valen	71.22 (1.03)	68.41 (2.95)	61.61 (2.79)	48.37 (2.24)	52.49 (1.00)	62.16 (0.74)
Cavl (2022)	69.78 (3.00)	72.12 (1.08)	65.02 (1.34)	52.67 (2.32)	55.06 (0.48)	61.91 (0.46)
Clsp+Cavl	73.13 (1.23)	58.76 (1.75)	59.86 (2.92)	48.65 (2.31)	53.48 (0.76)	49.48 (0.37)
Conf+Cavl	72.00 (1.22)	71.24 (3.81)	64.42 (0.89)	51.63 (5.03)	54.85 (0.92)	62.43 (0.43)
Pop (2023)	75.17 (1.04)	77.79 (2.11)	67.93 (1.44)	53.83 (0.69)	55.31 (0.71)	65.09 (0.64)
Clsp+Pop	74.25 (0.89)	60.18 (2.48)	59.61 (1.84)	50.58 (1.47)	32.08 (29.29)	50.77 (0.42)
Conf+Pop	77.58 (1.01)	78.41 (2.13)	66.21 (2.19)	54.82 (3.60)	56.49 (1.10)	65.25 (0.23)
CroSel (2024)	75.11 (1.79)	81.24 (3.68)	67.58 (1.16)	52.23 (2.83)	52.64 (1.21)	67.72 (0.32)
Clsp+CroSel	76.53 (1.34)	63.72 (2.23)	59.75 (2.79)	51.29 (1.69)	52.24 (0.84)	53.53 (0.93)
Conf+CroSel	77.76 (0.50)	81.15 (2.57)	65.93 (1.94)	54.10 (2.75)	54.97 (0.65)	67.55 (0.22)

Table C.3.: Average test-set accuracies (std.) on the supervised datasets with added incorrect candidate labels. We benchmark our strategy (Conf) as well as the cleaning method Clsp combined with all existing methods. We use the pre-trained Blip-2 model for all results in this table.

Method	*mnist*	*fmnist*	*kmnist*	*cifar10*	*cifar100*
Proden	87.21 (0.83)	71.18 (2.95)	59.31 (1.22)	99.07 (0.05)	90.51 (0.18)
Clsp+Proden	85.91 (2.42)	72.11 (2.81)	62.61 (1.00)	99.03 (0.07)	90.31 (0.13)
Conf+P. (no corr.)	91.74 (0.34)	78.38 (0.50)	66.88 (0.76)	99.03 (0.04)	91.16 (0.10)
Conf+Proden	91.55 (0.23)	78.09 (0.33)	66.43 (0.38)	99.03 (0.04)	91.16 (0.10)
Cc	86.29 (2.18)	66.19 (2.77)	58.29 (0.32)	99.07 (0.05)	73.43 (1.40)
Clsp+Cc	85.46 (1.93)	71.37 (2.34)	61.37 (1.09)	99.03 (0.06)	89.00 (1.56)
Conf+Cc	85.20 (4.16)	59.75 (2.68)	57.07 (0.66)	99.04 (0.03)	71.45 (0.94)
Valen	78.91 (0.80)	66.53 (2.65)	58.48 (0.45)	92.17 (0.54)	67.24 (2.49)
Clsp+Valen	84.72 (3.10)	68.84 (1.49)	60.76 (0.76)	98.17 (0.17)	84.53 (1.23)
Conf+Valen	74.20 (21.99)	69.09 (2.71)	60.95 (2.59)	42.63 (19.92)	60.44 (1.91)
Cavl	71.11 (3.92)	59.85 (6.49)	48.15 (5.07)	41.78 (21.40)	31.95 (1.80)
Clsp+Cavl	83.72 (3.57)	67.38 (2.59)	62.06 (2.12)	87.34 (12.71)	68.02 (1.96)
Conf+Cavl	71.86 (4.57)	59.54 (6.62)	52.14 (3.89)	29.97 (15.73)	37.34 (2.59)
Pop	87.08 (0.58)	72.30 (2.63)	60.63 (1.15)	99.06 (0.04)	90.50 (0.21)
Clsp+Pop	85.43 (2.60)	72.05 (2.41)	62.49 (0.90)	99.04 (0.07)	90.37 (0.06)
Conf+Pop	91.19 (0.29)	79.15 (1.23)	67.37 (0.28)	99.05 (0.04)	91.12 (0.09)
CroSel	91.84 (0.44)	76.34 (1.21)	65.55 (0.81)	99.07 (0.02)	75.86 (2.26)
Clsp+CroSel	91.70 (0.62)	74.42 (1.02)	67.93 (1.07)	99.08 (0.02)	88.80 (0.85)
Conf+CroSel	91.85 (0.61)	77.31 (0.46)	64.73 (1.52)	99.07 (0.03)	77.26 (0.98)

Table C.4.: Number of significant differences aggregated from Table C.2 and C.4 using a t-test (level 5 %).

Comparison vs. all others	Wins	Ties	Losses
PRODEN	27	37	8
CLSP+PRODEN	18	27	27
CONF+PRODEN (no correction)	41	25	6
CONF+PRODEN	50	20	2
CC	18	39	15
CLSP+CC	18	22	32
CONF+CC	22	30	20
VALEN	5	30	37
CLSP+VALEN	17	15	40
CONF+VALEN	5	24	43
CAVL	5	26	41
CLSP+CAVL	9	19	44
CONF+CAVL	4	26	42
POP	29	39	4
CLSP+POP	17	23	32
CONF+POP	49	22	1
CROSEL	30	33	9
CLSP+CROSEL	25	15	32
CONF+CROSEL	44	22	6