Machine Learning (2025) 114:193
https://doi.org/10.1007/s10994-025-06796-z

Robust partial-label learning by leveraging class activation values

Tobias Fuchs[1] · **Florian Kalinke**[1]

Received: 4 April 2025 / Revised: 4 April 2025 / Accepted: 5 May 2025 /
Published online: 15 July 2025

Abstract
Real-world training data is often noisy; for example, human annotators assign conflicting class labels to the same instances. Partial-label learning (PLL) is a weakly supervised learning paradigm that allows training classifiers in this context without manual data cleaning. While state-of-the-art methods have good predictive performance, their predictions are sensitive to high noise levels, out-of-distribution data, and adversarial perturbations. We propose a novel PLL method based on subjective logic, which explicitly represents uncertainty by leveraging the magnitudes of the underlying neural network's class activation values. Thereby, we effectively incorporate prior knowledge about the class labels by using a novel label weight re-distribution strategy that we prove to be optimal. We empirically show that our method yields more robust predictions in terms of predictive performance under high PLL noise levels, handling out-of-distribution examples, and handling adversarial perturbations on the test instances.

Keywords Weakly-supervised learning · Partial-label learning · Learning under noise · Robust classification · Deep learning

1 Introduction

In real-world applications, one often encounters ambiguously labeled data. In crowd labeling, for example, human annotation produces instances with multiple conflicting class labels (Huiskes & Lew, 2008). Other examples with ambiguous data include web mining (Guillaumin et al., 2010; Zeng et al., 2013) and audio classification (Briggs et al., 2012). Partial-label learning (PLL; Grandvalet, 2002; Nguyen & Caruana, 2008; Xu et al., 2023; Zhang et al., 2017) is a weakly-supervised learning paradigm that

Editor: Maria Concepcion Bielza Lozoya.

✉ Tobias Fuchs
tobias.fuchs@kit.edu

Florian Kalinke
florian.kalinke@kit.edu

[1] Karlsruhe Institute of Technology, Karlsruhe, Germany

targets classification with such inexact supervision, where training data can have several candidate labels of which only one is correct. While cleaning is costly, PLL algorithms allow to handle such ambiguously labeled data directly.

As predictions by machine-learning systems often impact actions or decisions by humans, they should be *robust* regarding several criteria to limit mispredictions 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 (Lambrou et al., 2011; Reamaroon et al., 2019; Yang et al., 2009) or financial fraud detection (Berkmans & Karthick, 2023; Cheng et al., 2020; Xiang et al., 2023), where all three criteria 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.

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.[1]
- 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 2 discusses related work. Section 3 defines the problem and our notations. We propose our PLL method in Sect. 4 and show our experiments in Sect. 5. Section 6 concludes. We defer all proofs and hyperparameter choices to the appendices.

2 Related work

This section separately details related work on PLL and on making predictions more robust regarding aspects (a)–(c).

2.1 Partial-label learning (PLL)

PLL is a typical weakly-supervised learning problem. Early approaches apply common supervised learning frameworks to the PLL context: Grandvalet (2002) propose a logistic regression formulation, Jin and Ghahramani (2002) propose an expectation-maximization strategy, Hüllermeier and Beringer (2005) propose a nearest-neighbors method, Nguyen and Caruana (2008) propose an extension of support-vector machines, and Cour et al. (2011) introduce an average loss formulation allowing the use of any supervised method.

As most of the aforementioned algorithms struggle with non-uniform noise, several extensions and novel methods have been proposed: Zhang and Yu (2015), Xu et al. (2019), Feng and An (2019), Ni et al. (2021) leverage ideas from representation learning, Yu and Zhang (2017), Feng and An (2019), Ni et al. (2021) extend the maximum-margin idea, Liu and Dietterich (2012), Lv et al. (2020) propose extensions of the expectation-maximization strategy, Zhang et al. (2017), Tang and Zhang (2017), Wu and Zhang (2018) propose stacking and boosting ensembles, and Lv et al. (2020), Cabannes et al. (2020) introduce a minimum loss formulation, which iteratively disambiguates the partial labels.

The progress of deep-learning techniques also yields advances in PLL. Feng et al. (2020), Lv et al. (2020) provide risk-consistent loss formulations for PLL, Xu et al. (2021), Wang et al. (2022), Zhang et al. (2022), Fan et al. (2024), Tian et al. (2024) use advances in deep representation learning and data augmentation to strengthen inference from PLL data, and Xu et al. (2023), Fuchs and Kalinke (2025) gradually refine the partial-label learning candidate sets by removing unlikely labels. Similar to our approach, Cavl (Zhang et al., 2022) makes use of the class activation values to identify the correct labels in the candidate sets. While they use the activation values as a feature map, we use the activation values to build multinomial opinions in subjective logic, which reflect the involved uncertainty in prediction-making.

Similar to Proden (Lv et al., 2020), Pop (Xu et al., 2023), and CroSel (Tian et al., 2024), among many others, we iteratively update a label weight vector keeping track of

[1] https://github.com/mathefuchs/robust-pll

the model's knowledge about the labeling of all instances. However, those three methods do not provide any formal reasoning for their respective update rules. In contrast, we prove our update rule's optimality in Propositions 4.3 and 4.6 for the mean-squared error and cross-entropy error, respectively.

We note that, at a first glance, our update rule also appears to be similar to the label smoothing proposed by Gong et al. (2024). Based on a smoothing hyperparameter $r \in [0, 1]$, they iteratively update the label weights: $r = 1$ uniformly allocates weight among all class labels, while $r = 0$ only allocates label weight on the most likely label. In contrast, our update strategy does not involve any hyperparameter and allocates probability mass based on the uncertainty involved in prediction-making.

2.2 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 & 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 & Welling, 2014; Kendall & Gal, 2017) or ensembles[2] (Lakshminarayanan et al., 2017; Wimmer et al., 2023). To recognize OOD samples (b), one commonly employs techniques from representation learning (Xu et al., 2021; Zhang & Yu, 2015) 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. 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 (Sect. 4.3) and using an optimal label-weight update strategy (Sect. 4.4). 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.

[2] 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 Sect. 5.

3 Problem statement and notations

This section defines the partial-label learning problem, establishes the notations used throughout this work, and briefly summarizes subjective logic.

3.1 Partial-label learning (PLL)

Given a real-valued feature space $\mathcal{X} = \mathbb{R}^d$ and a label space $\mathcal{Y} = [k] := \{1, \ldots, k\}$ with $3 \leq k \in \mathbb{N}$ class labels, we denote a partially-labeled training dataset with $\mathcal{D} = \{(\boldsymbol{x}_i, S_i) \mid i \in [n]\}$ consisting of n instances with features $\boldsymbol{x}_i \in \mathcal{X}$ and a non-empty set of candidate labels $S_i \subseteq \mathcal{Y}$ for all $i \in [n]$. The ground-truth label $y_i \in \mathcal{Y}$ of instance i is unknown. However, one assumes $y_i \in S_i$ (Jin & Ghahramani, 2002; Liu & Dietterich, 2012; Lv et al., 2020). The goal is to train a classifier $g : \mathcal{X} \to \mathcal{Y}$ that minimizes the empirical loss with weak supervision only, that is, the exact ground truth labels y_i are unavailable during training. We use label weight vectors $\boldsymbol{\ell}_i \in [0,1]^k$ with $\|\boldsymbol{\ell}_i\|_1 = 1$ to represent the model's knowledge about the labeling of instance $i \in [n]$. Thereby, $\|\boldsymbol{\ell}_i\|_p = (\sum_{j=1}^k |\ell_{ij}|^p)^{1/p}$ denotes the p-norm. $\ell_{ij} \in [0,1]$ denotes class j's weight regarding instance i. We typeset vectors in bold font.

3.2 Subjective logic (SL)

Inspired by Dempster–Shafer theory (Dempster, 1967; Shafer, 1986), Jøsang (2016) proposes a theory of evidence, called subjective logic, 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 $\|\mathfrak{a}_i\|_1 = 1$ and requires additivity, that is, $\mathfrak{u}_i + \|\mathfrak{b}_i\|_1 = 1$ for all $i \in [n]$. The projected probability is defined by $\bar{\boldsymbol{p}}_i = \mathfrak{b}_i + \mathfrak{a}_i \mathfrak{u}_i \in [0,1]^k$ for $i \in [n]$. $\bar{\boldsymbol{p}}_i$ 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 $\boldsymbol{x}_i \in \mathcal{X}$, and a prediction model $f : \mathbb{R}^d \to \mathbb{R}^k_{\geq 0}$, we set $\mathfrak{b}_i = f(\boldsymbol{x}_i)/(k + \|f(\boldsymbol{x}_i)\|_1)$, $\mathfrak{a}_i$ using prior knowledge, and $\mathfrak{u}_i = k/(k + \|f(\boldsymbol{x}_i)\|_1)$. The multinomial opinion ω_i can be expressed in terms of a Dirichlet-distributed random variable with parameters $\boldsymbol{\alpha}_i = f(\boldsymbol{x}_i) + 1$, that is, $\mathbb{E}_{\boldsymbol{p}_i \sim \mathrm{Dir}(\boldsymbol{\alpha}_i)}[\boldsymbol{p}_i] := \boldsymbol{\alpha}_i / \|\boldsymbol{\alpha}_i\|_1 = \mathfrak{b}_i + \mathfrak{a}_i \mathfrak{u}_i = \bar{\boldsymbol{p}}_i$, 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, $\bar{\boldsymbol{p}}_i = \mathfrak{a}_i$ for $i \in [n]$. In the following, we provide an example of multinomial opinions in subjective logic.

Example 3.1 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 $\bar{\boldsymbol{p}}_1 = \bar{\boldsymbol{p}}_2 = (2/3, 1/6, 1/6)$. 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.

4 Our 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, $\text{softmax} : \mathbb{R}^k \to [0,1]^k$, 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 & Waegeman, 2021; Sale et al., 2023), which is also evident from Example 3.1, 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 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 1 outlines our method: RobustPll. First, we initialize the label weights ℓ_{ij} in Line 1. These weights ℓ_{ij} represent the degree to which instance i has the correct label j. Section 4.1 discusses their initialization and interpretation. In Line 2, we set up our model $f : \mathbb{R}^d \to \mathbb{R}^k_{\geq 0}$ and its parameters $\boldsymbol{\theta}$. 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 $\hat{\mathcal{R}}(f;\lambda_t)$, which is discussed in Sect. 4.3. In Line 5, we then compute the empirical risk $\hat{\mathcal{R}}(f;\lambda_t)$ and update the model parameters $\boldsymbol{\theta}$ in Line 6. Those steps are discussed in Sect. 4.2. Thereafter, we update the label weights ℓ_{ij} in Line 7 as shown in Sect. 4.4.

The remainder of Sect. 4 presents further analyses. In Sect. 4.5, we discuss our reinterpretation of the label weight update within SL. Section 4.6 bounds the rate of change of $\boldsymbol{\ell}_i$ and Sect. 4.7 demonstrates why the squared error loss is superior to the cross-entropy loss in our setting. Section 4.8 discusses our approach's runtime.

Algorithm 1 RobustPll (Our proposed method)

Input: Partially-labeled dataset $\mathcal{D} = \{(\boldsymbol{x}_i, S_i) \in \mathcal{X} \times 2^{\mathcal{Y}} \mid i \in [n]\}$ with n instances, and $k = |\mathcal{Y}|$ classes;
Output: Model parameters $\boldsymbol{\theta}$;
1: Init weights $\ell_{ij} \leftarrow \mathbb{1}_{\{j \in S_i\}}/|S_i|$ for $i \in [n]$, $j \in [k]$;
2: Init model f and its parameters $\boldsymbol{\theta}$;
3: **for** $t = 1, \ldots, T$ **do**
4: $\quad \lambda_t \leftarrow \min(2t/T, 1)$
5: $\quad$ Compute the empirical risk $\hat{\mathcal{R}}(f; \lambda_t)$ by (4);
6: $\quad$ Update $\boldsymbol{\theta}$ by backpropagation;
7: $\quad$ Set label weights $\boldsymbol{\ell}_i$ $(i \in [n])$ to the solution $\boldsymbol{\ell}_i^*$ of (5) using the closed-form solution in Proposition 4.3;

4.1 Initializing the label weights

The label weights ℓ_{ij} represent the current knowledge of our method about instance i having the correct label j. They must sum to one, that is, $\|\boldsymbol{\ell}_i\|_1 = 1$, and must be zero if j is not a candidate label of instance i, that is, $\ell_{ij} = 0$ if $j \notin S_i$ $(i \in [n])$. We initialize the label weights with $\ell_{ij} = \mathbb{1}_{\{j \in S_i\}}/|S_i|$, where $\mathbb{1}_{\{\cdot\}}$ denotes the indicator function. ℓ_{ij} satisfies both requirements. Also, ℓ_{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, $\boldsymbol{\ell}_i = \mathfrak{b}_i + \mathfrak{a}_i \mathfrak{u}_i$ 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 $\boldsymbol{\ell}_i = \mathfrak{a}_i$ at initialization: The label weights $\boldsymbol{\ell}_i$ are solely determined by prior knowledge about the candidate sets encoded in $\mathfrak{a}_i$.

4.2 Training a model

We interleave learning the parameters $\boldsymbol{\theta}$ of a model $f : \mathbb{R}^d \to \mathbb{R}^k_{\geq 0}$ (Lines 4–6) with updating the label weights $\boldsymbol{\ell}_i$ 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 4.1. Instead, the model f outputs evidence supporting a particular class label, which parameterizes a Dirichlet distribution $\mathrm{Dir}(\boldsymbol{\alpha}_i)$ with

$$\boldsymbol{\alpha}_i = f(\boldsymbol{x}_i; \boldsymbol{\theta}) + 1 \in \mathbb{R}^k_{\geq 1} \quad \text{for} \quad i \in [n]. \tag{1}$$

To fit f to the label weights $\boldsymbol{\ell}_i$, 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 $\boldsymbol{\ell}_i$ and $\boldsymbol{p}_i \sim \mathrm{Dir}(\boldsymbol{\alpha}_i)$ with $\boldsymbol{\alpha}_i$ as in (1). For an instance $i \in [n]$ with features $\boldsymbol{x}_i \in \mathcal{X}$ and label weights $\boldsymbol{\ell}_i \in [0, 1]^k$, the squared error using the bias-variance decomposition is

$$\mathcal{L}(f(\boldsymbol{x}_i; \boldsymbol{\theta}), \boldsymbol{\ell}_i) = \mathbb{E}_{\boldsymbol{p}_i \sim \mathrm{Dir}(\boldsymbol{\alpha}_i)} \|\boldsymbol{\ell}_i - \boldsymbol{p}_i\|_2^2 \tag{2}$$

$$= \sum_{j=1}^{k} \mathbb{E}\left[(\ell_{ij} - p_{ij})^2\right] \overset{(i)}{=} \sum_{j=1}^{k} \left[\left(\ell_{ij} - \mathbb{E}\left[p_{ij}\right]\right)^2 + \mathrm{Var}\left[p_{ij}\right]\right]$$

$$\overset{(ii)}{=} \sum_{j=1}^{k} \left[\underbrace{\left(\ell_{ij} - \bar{p}_{ij}\right)^2}_{=:\mathcal{L}_{ij}^{\mathrm{err}}} + \underbrace{\frac{\bar{p}_{ij}(1-\bar{p}_{ij})}{1+\|\boldsymbol{\alpha}_i\|_1}}_{=:\mathcal{L}_{ij}^{\mathrm{var}}}\right], \tag{3}$$

with $\bar{\boldsymbol{p}}_i = \mathbb{E}_{\boldsymbol{p}_i \sim \mathrm{Dir}(\boldsymbol{\alpha}_i)}[\boldsymbol{p}_i] = \boldsymbol{\alpha}_i / \|\boldsymbol{\alpha}_i\|_1$. (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 (2). One can directly compute $\mathcal{L}$ using the outputs of f only. In the following, we give an example highlighting the differences to softmax normalization.

Example 4.1 Let $n = 2$, $k = 3$, $\mathcal{Y} = \{1, 2, 3\}$, $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathcal{X}$, $f(\boldsymbol{x}_1) = (4, 1, 1)$, and $f(\boldsymbol{x}_2) = (7, 4, 4)$. Using softmax normalization, both predictions yield the same discrete probabilities, that is, $\mathrm{softmax}(f(\boldsymbol{x}_1)) = \mathrm{softmax}(f(\boldsymbol{x}_2)) \approx (0.910, 0.045, 0.045)$, although $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ have different activation values. In our setting, $\boldsymbol{\alpha}_1 = f(\boldsymbol{x}_1) + 1 = (5, 2, 2)$ and $\boldsymbol{\alpha}_2 = f(\boldsymbol{x}_2) + 1 = (8, 5, 5)$ by (1). The predicted probabilities are $\bar{\boldsymbol{p}}_1 = \mathbb{E}_{\boldsymbol{p}_1 \sim \mathrm{Dir}(\boldsymbol{\alpha}_1)}[\boldsymbol{p}_1] = (5/9, 2/9, 2/9)$ and $\bar{\boldsymbol{p}}_2 = \mathbb{E}_{\boldsymbol{p}_2 \sim \mathrm{Dir}(\boldsymbol{\alpha}_2)}[\boldsymbol{p}_2] = (4/9, 5/18, 5/18)$. 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, $\mathrm{Var}_{\boldsymbol{p}_1 \sim \mathrm{Dir}(\boldsymbol{\alpha}_1)}[\boldsymbol{p}_1] \approx (0.025, 0.017, 0.017)$ and $\mathrm{Var}_{\boldsymbol{p}_2 \sim \mathrm{Dir}(\boldsymbol{\alpha}_2)}[\boldsymbol{p}_2] \approx (0.013, 0.011, 0.011)$. Since $f(\boldsymbol{x}_2)$ has higher activation values, there is less associated uncertainty across all classes. Hence, $\mathrm{Dir}(\boldsymbol{\alpha}_2)$ has less variance than $\mathrm{Dir}(\boldsymbol{\alpha}_1)$.

The loss term $\mathcal{L}$ in (3) can be separated into an error and variance component, $\mathcal{L}_{ij}^{\mathrm{err}}$ and $\mathcal{L}_{ij}^{\mathrm{var}}$, respectively. $\mathcal{L}_{ij}^{\mathrm{err}}$ enforces model fit and $\mathcal{L}_{ij}^{\mathrm{var}}$ 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 $\mathcal{L}_{ij}^{\mathrm{err}} > \mathcal{L}_{ij}^{\mathrm{var}}$ if ℓ_{ij} and $\bar{p}_{ij}$ deviate too much, which we discuss in the following.

Proposition 4.2 *Given instance* $(\boldsymbol{x}_i, S_i) \in \mathcal{D}$, *parameters* $\boldsymbol{\theta}$, *label weights* $\boldsymbol{\ell}_i$, *and* $\bar{\boldsymbol{p}}_i = \boldsymbol{\alpha}_i / \|\boldsymbol{\alpha}_i\|_1$, *it holds that* $\mathcal{L}_{ij}^{\mathrm{err}} < \mathcal{L}_{ij}^{\mathrm{var}}$ *if and only if* $\bar{p}_{ij} - \sqrt{\mathcal{L}_{ij}^{\mathrm{var}}} < \ell_{ij} < \bar{p}_{ij} + \sqrt{\mathcal{L}_{ij}^{\mathrm{var}}}$, *for all* $i \in [n]$ *and* $j \in [k]$.

Proposition 4.2 sheds light on the magnitudes of $\mathcal{L}_{ij}^{\mathrm{err}}$ and $\mathcal{L}_{ij}^{\mathrm{var}}$. When ℓ_{ij} is within one standard deviation from $\bar{p}_{ij}$, ℓ_{ij} is close to $\bar{p}_{ij}$. In this regime, reducing variance is more important than improving model fit, that is, $\mathcal{L}_{ij}^{\mathrm{err}} < \mathcal{L}_{ij}^{\mathrm{var}}$. Reducing the variance of the Dirichlet distribution is equivalent to a reduction of the uncertainty about the prediction. When ℓ_{ij} is outside one standard deviation from $\bar{p}_{ij}$, model fit is more important, that is, $\mathcal{L}_{ij}^{\mathrm{err}} > \mathcal{L}_{ij}^{\mathrm{var}}$. 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 features.

4.3 Regularization

Given an instance $\boldsymbol{x}_i \in \mathcal{X}$, the correct label $y_i \in \mathcal{Y}$ is hidden within $S_i \subseteq \mathcal{Y}$ (Sect. 3). Therefore, our model f should not allocate any evidence to incorrect labels, that is, $f_j(\boldsymbol{x}_i;\boldsymbol{\theta})$ 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 $\mathrm{Dir}(\tilde{\boldsymbol{\alpha}}_i)$ and $\mathrm{Dir}(1)$. We compute the empirical risk as

$$\hat{\mathcal{R}}(f;\lambda_t) = \frac{1}{n}\sum_{i=1}^{n}\Big[\mathcal{L}\left(f(\boldsymbol{x}_i;\boldsymbol{\theta}),\boldsymbol{\ell}_i\right) + \lambda_t \,\mathrm{KL}\left(\mathrm{Dir}(\tilde{\boldsymbol{\alpha}}_i)\|\,\mathrm{Dir}(1)\right)\Big]. \tag{4}$$

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 $\mathrm{Dir}(\tilde{\boldsymbol{\alpha}}_i)$ and $\mathrm{Dir}(1)$ respectively, their KL divergence permits a closed-form expression (Penny, 2001). One updates the parameters $\boldsymbol{\theta}$ by backpropagation of (4). Note that the KL term also depends on the parameters $\boldsymbol{\theta}$.

4.4 Updating the label weights

After updating the parameters $\boldsymbol{\theta}$, we extract the learned knowledge about the class labels to iteratively disambiguate the candidate sets S_i. For a fixed instance $(\boldsymbol{x}_i, S_i) \in \mathcal{D}$ and model parameters $\boldsymbol{\theta}$, we want to find the optimal label weights $\boldsymbol{\ell}_i \in [0,1]^k$ that minimize (3), while maintaining all prior information about the candidate set membership, that is, $\ell_{ij} = 0$ if $j \notin S_i$. We cannot directly assign $\bar{\boldsymbol{p}}_i$ to the label weights $\boldsymbol{\ell}_i$ since f can allocate evidence to incorrect labels, that is, $f_j(\boldsymbol{x}_i;\boldsymbol{\theta}) \geq 0$ for $j \notin S_i$. In Line 7 of Algorithm 1, we assign $\boldsymbol{\ell}_i \in [0,1]^k$ to the solution of

$$\min_{\boldsymbol{\ell}'_i \in [0,1]^k} \mathcal{L}\left(f(\boldsymbol{x}_i;\boldsymbol{\theta}),\boldsymbol{\ell}'_i\right) \quad \text{subject to} \quad \|\boldsymbol{\ell}'_i\|_1 = 1 \text{ and } \ell'_{ij} = 0 \text{ if } j \notin S_i. \tag{5}$$

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

Proposition 4.3 *Given a fixed instance* $(\boldsymbol{x}_i, S_i) \in \mathcal{D}$, *model parameters* $\boldsymbol{\theta}$, *and* $\bar{\boldsymbol{p}}_i = \boldsymbol{\alpha}_i / \|\boldsymbol{\alpha}_i\|_1$, *the optimization problem* (5), *with* $\mathcal{L}$ *as in* (3), *has the solution*

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

The proof of Proposition 4.3 (Appendix A.2) first shows that ℓ^*_{ij} is a feasible solution for the constraints in (5) and then establishes optimality using the Lagrangian multiplier method since $\mathcal{L}$ is continuous and differentiable. The solution ℓ^*_{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 4.3 differs from the heuristic ones proposed in related work (Lv et al., 2020; Tian et al., 2024; Xu et al., 2023).

4.5 Reinterpreting the label weights

Recall from Sect. 3 that subjective logic allows decomposing the projected probabilities $\bar{\boldsymbol{p}}_i$ into a multinomial opinion ω_i with a belief and uncertainty term, that is, $\bar{\boldsymbol{p}}_i = \mathfrak{b}_i + \mathfrak{a}_i \mathfrak{u}_i$. This representation directly allows quantifying the uncertainty involved in prediction-making. It is desirable that the label weight update $\boldsymbol{\ell}_i^*$ in Sect. 4.4 can also be written as such a multinomial opinion to allow for the direct quantification of the involved uncertainty.

Proposition 4.4 *Given a fixed instance* $(\boldsymbol{x}_i, S_i) \in \mathcal{D}$ *and parameters* $\boldsymbol{\theta}$, *the solution* ℓ_{ij}^* *of* (5), *which is given by Proposition* 4.3, *is equivalent to* $\boldsymbol{\ell}_i^* = \mathfrak{b}_i + \mathfrak{a}_i \mathfrak{u}_i$ *with*

$$\mathfrak{b}_{ij} = \mathbb{1}_{\{j \in S_i\}} \frac{f_j(\boldsymbol{x}_i;\boldsymbol{\theta})}{\|\boldsymbol{\alpha}_i\|_1} \text{ with } \boldsymbol{\alpha}_i = f(\boldsymbol{x}_i;\boldsymbol{\theta}) + 1,\ \mathfrak{u}_i = 1 - \sum_{j \in S_i} \mathfrak{b}_{ij}, \text{ and } \mathfrak{a}_{ij} = \frac{\mathbb{1}_{\{j \in S_i\}}}{|S_i|}.$$

Proposition 4.4 allows reinterpreting $\boldsymbol{\ell}_i^*$ (Proposition 4.3) as such a multinomial opinion ω_i about instance i. Proposition 4.4 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 ℓ_{ij} (Sect. 4.1). The prior weights are uniformly distributed among all candidate labels. The result in Proposition 4.4 establishes that our proposed update strategy (Proposition 4.3) is valid within subjective logic.

4.6 Bounding the label weights

This section examines how the model's probability outputs $\bar{\boldsymbol{p}}_i$ and the label weights $\boldsymbol{\ell}_i$ interact with each other. In the following, we provide an upper bound of the change of $\boldsymbol{\ell}_i$'s values over time. As the label weights $\boldsymbol{\ell}_i$ are the prediction targets in (3), it is desirable that the $\boldsymbol{\ell}_i$ do not oscillate, which we detail in the following.

Proposition 4.5 *Let* $(\boldsymbol{x}_i, S_i) \in \mathcal{D}$, *its label weights* $\boldsymbol{\ell}_i^{(t)} \in [0,1]^k$, *and the model's probability outputs* $\bar{\boldsymbol{p}}_i^{(t)}$ *at epoch* $t \in \mathbb{N}$. *Then*, $0 \leq \|\boldsymbol{\ell}_i^{(t+1)} - \boldsymbol{\ell}_i^{(t)}\|_2^2 \leq \|\bar{\boldsymbol{p}}_i^{(t+1)} - \bar{\boldsymbol{p}}_i^{(t)}\|_2^2$ for $i \in [n]$.

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

4.7 Cross-entropy loss

Although we use the squared error loss (2), it is worth considering the commonly used cross-entropy loss. Given an instance $\boldsymbol{x}_i \in \mathcal{X}$, a model f with parameters $\boldsymbol{\theta}$, and label weights $\boldsymbol{\ell}_i$, a cross-entropy formalization similar to Sensoy et al. (2018) is given by

$$\mathcal{L}_{\mathrm{CE}}(f(\boldsymbol{x}_i;\boldsymbol{\theta}), \boldsymbol{\ell}_i) = \mathbb{E}\left[-\sum_{j=1}^{k} \ell_{ij} \log p_{ij}\right] \overset{(i)}{=} \boldsymbol{\ell}_i \cdot \boldsymbol{\Psi}_i, \tag{6}$$

with $\Psi_{ij} = \psi(\|\boldsymbol{\alpha}_i\|_1) - \psi(\alpha_{ij})$ and ψ denoting the digamma function. (i) holds because of the linearity of the expected value and $\mathbb{E}_{p_{ij} \sim \mathrm{Dir}_j(\boldsymbol{\alpha}_i)} \log p_{ij} = \psi(\alpha_{ij}) - \psi(\|\boldsymbol{\alpha}_i\|_1)$. In the following, we establish the optimal choice of $\boldsymbol{\ell}_i$ in our optimization problem (5) using the cross-entropy loss (6).

Proposition 4.6 *Given a fixed instance* $(\boldsymbol{x}_i, S_i) \in \mathcal{D}$ *and parameters* $\boldsymbol{\theta}$, *optimization problem* (5), *using the cross-entropy loss* (6), *has the closed-form solution*

$$\ell_{ij}^{*} = \begin{cases} 1 & \text{if } j = \arg\min_{j' \in S_i} \Psi_{ij'}, \\ 0 & \text{else.} \end{cases}$$

This suggests that the cross-entropy loss (6) enforces an aggressive label-weight update strategy setting all mass on one class label. Also, the label weights given by Proposition 4.6 cannot be reinterpreted in SL as discussed in Sect. 4.5. 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 4.3.

4.8 Runtime analysis

Recall from Sect. 4.2 that one does not need to numerically approximate the integral within the expectation value in (2). Given label weights $\boldsymbol{\ell}_i$, one can directly compute $\mathcal{L}$ using the outputs of f only. The computation of the KL divergence between two Dirichlet-distributed random variables with parameters $\mathrm{Dir}(\tilde{\boldsymbol{\alpha}}_i)$ and $\mathrm{Dir}(1)$, respectively, admits a closed-form expression (Penny, 2001), leading to an overall linear runtime in n to compute $\hat{\mathcal{R}}(f;\lambda_t)$ (Algorithm 1, Line 5). In Line 7 of Algorithm 1, Proposition 4.3 also permits updating ℓ_{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 Experiments

Section 5.1 summarizes all methods that we compare against and Sect. 5.2 outlines the experimental setup. Thereafter, Sect. 5.3 analyzes the methods' robustness against PLL noise, Sect. 5.4 against out-of-distribution samples, and Sect. 5.5 against adversarial perturbations.

5.1 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 & Beringer, 2005), PLSVM (Nguyen & Caruana, 2008), IPAL (Zhang & 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., 2022), 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[3] (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 (Feng et al., 2020; Lv 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.[1]

5.2 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 MNIST-like datasets with added noise and six real-world PLL datasets and refer to Appendix B.2 for an overview of 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 NotMNIST (Bulatov, 2011). For the real-world datasets, we use *bird-song* (Briggs et al., 2012), *lost* (Cour et al., 2011), *mir-flickr* (Huiskes & Lew, 2008), *msrc-v2* (Liu & 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 $g : \mathbb{R}^d \to [0,1]^k$, which outputs probabilities $g_j(\boldsymbol{x}_i)$ for instance $i \in [n]$ and class labels $j \in [k]$. Given an instance's features $\boldsymbol{x}_i \in \mathcal{X}$ with correct label $y_i \in \mathcal{Y}$, a flipping probability of $\xi_j(\boldsymbol{x}_i) = g_j(\boldsymbol{x}_i) / \max_{j' \in \mathcal{Y} \setminus \{y_i\}} g_{j'}(\boldsymbol{x}_i)$ determines whether to add the incorrect label $j \neq y_i$ to the candidate set S_i. Additionally, one divides $\xi_j(\boldsymbol{x}_i)$ by the mean probability $\frac{1}{k-1} \sum_{j' \neq y_i} \xi_{j'}(\boldsymbol{x}_i)$ 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 NotMNIST. We remark that five out of the ten datasets

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

do not contain a single instance with a candidate set that only consists of the ground truth label. This prohibits the application of algorithms from related fields, for example, semi-supervised learning. We publicly provide all code and data for reproducibility.[1]

5.3 Robustness under PLL noise

Robust PLL algorithms should exhibit good predictive performance when confronted with PLL noise from ambiguous candidate sets. Table 1 shows the accuracies of all methods on the MNIST-like and real-world datasets. All supervised datasets have added noise (Sect. 5.2). 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 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

Table 1 Average test-set accuracies (± std.) on the MNIST-like and real-world datasets

All methods	MNIST-like datasets with inst.-dep. noise				Real-world datasets
	MNIST	FMNIST	KMNIST	NotMNIST	
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)

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 Sect. 5.2. The column for the real-world datasets contains averages across all six real-world datasets; the corresponding non-aggregated results are collected in Table 5. 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

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 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 2 The difference in the predictive entropies on the test and OOD sets

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**

The models have been trained on the MNIST train dataset with added noise as discussed in Sect. 5.2. 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 Fig. 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

Table 3 Average test-set accuracies (± std.) on the real-world datasets

Deep-learning methods	Corrupted real-world datasets with adversarial parameter ϵ				
	$\epsilon = 0.0$	$\epsilon = 0.1$	$\epsilon = 0.2$	$\epsilon = 0.3$	$\epsilon = 0.4$
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)

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 $\epsilon = 0.0$ is identical to the last column of Table 1. The triangles indicate our proposed methods

Table 2 shows the differences in the normalized entropies (range 0–1) 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 NotMNIST test set. Samples from the NotMNIST 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 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).

5.5 Performance on adversarial examples

In recent years, many attacks on neural networks have been discussed in the literature (Goodfellow et al., 2015; Moosavi-Dezfooli et al., 2016; Madry et al., 2018; Szegedy et al., 2014). 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 sign $\nabla_x f(x;\boldsymbol{\theta})$ 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 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 3 therefore matches the last column of Table 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 3 as it is trained on adversarial examples, that 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.

6 Conclusions

In this work, we presented a novel PLL method that leverages class activation values within the subjective logic framework. We formally analyzed our method showing our update rule's optimality with respect to the mean-squared error and its reinterpretation in subjective logic. We empirically showed that our approach yields more robust predictions than other state-of-the-art approaches in terms of prediction quality under high PLL noise, dealing with out-of-distribution examples, as well as handling instance features corrupted by adversarial noise. To the best of our knowledge, we are the first to address these aspects in the PLL setting.

Appendix A. Proofs

This section collects all proofs of the propositions in the main text. The proof of Proposition 4.2 is in Appendix A.1, that of Proposition 4.3 is in Appendix A.2, that of Proposition 4.4 is in Appendix A.3, and that of Proposition 4.5 is in Appendix A.4.

A.1 Proof of Proposition 4.2

Solving $\mathcal{L}_{ij}^{\text{err}} = \mathcal{L}_{ij}^{\text{var}}$ for the label weights ℓ_{ij} yields

$$\mathcal{L}_{ij}^{\text{err}} = \mathcal{L}_{ij}^{\text{var}} \Leftrightarrow \left(\ell_{ij} - \bar{p}_{ij}\right)^2 = \frac{\bar{p}_{ij}\left(1-\bar{p}_{ij}\right)}{1+\|\boldsymbol{\alpha}_i\|_1}$$
$$\Leftrightarrow \ell_{ij} = \bar{p}_{ij} \pm \sqrt{\frac{\bar{p}_{ij}\left(1-\bar{p}_{ij}\right)}{1+\|\boldsymbol{\alpha}_i\|_1}}$$
$$\Leftrightarrow \ell_{ij} = \bar{p}_{ij} \pm \sqrt{\mathcal{L}_{ij}^{\text{var}}}.$$

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

A.2 Proof of Proposition 4.3

The proof first shows that $\boldsymbol{\ell}_i^*$ is a feasible solution for the constraints in (5) and then establishes that $\boldsymbol{\ell}_i^*$ is indeed optimal using the Lagrangian multiplier method.

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

$$\begin{aligned}\sum_{j=1}^{k} \ell_{ij}^* &= \sum_{j\in S_i}\left(\bar{p}_{ij} + \frac{1}{|S_i|}\left(1 - \sum_{j'\in S_i}\bar{p}_{ij'}\right)\right)\\ &= \sum_{j\in S_i}\bar{p}_{ij} + \frac{1}{|S_i|}\sum_{j\in S_i}\left(1 - \sum_{j'\in S_i}\bar{p}_{ij'}\right)\\ &= \sum_{j\in S_i}\bar{p}_{ij} + 1 - \sum_{j'\in S_i}\bar{p}_{ij'} = 1.\end{aligned}$$

Constraint (ii) follows directly from the definition of ℓ_{ij}^* in Proposition 4.3.

Optimality. Since the loss $\mathcal{L}$ is differentiable, continuous, and convex in $\boldsymbol{\ell}_i$, we incorporate the constraints (i) and (ii) using the Lagrangian multiplier method as follows:

$$\mathrm{L}(\boldsymbol{\ell}_i;\lambda_i) = \sum_{j=1}^{k}(\ell_{ij} - \bar{p}_{ij})^2 + \lambda_i\left(\sum_{j=1}^{k}\ell_{ij} - 1\right),$$

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

$$\begin{aligned}\frac{\partial}{\partial \ell_{ij}}\mathrm{L}(\boldsymbol{\ell}_i;\lambda_i) = 0 &\Leftrightarrow 2\left(\ell_{ij} - \bar{p}_{ij}\right) + \lambda_i = 0\\ &\Leftrightarrow \ell_{ij} = \bar{p}_{ij} - \frac{\lambda_i}{2},\end{aligned} \tag{7}$$

for $i \in [n]$ and $j \in S_i$. For $j \notin S_i$, $\ell_{ij} = 0$. Inserting (7) into constraint (i) yields

$$\begin{aligned} \|\boldsymbol{\ell}_i\|_1 = 1 \Leftrightarrow \sum_{j=1}^{k} \ell_{ij} = \sum_{j \in S_i} \left(\bar{p}_{ij} - \frac{\lambda_i}{2} \right) = \sum_{j \in S_i} \bar{p}_{ij} - \frac{|S_i| \lambda_i}{2} = 1 \\ \Leftrightarrow \lambda_i = \frac{2}{|S_i|} \left(\sum_{j \in S_i} \bar{p}_{ij} - 1 \right). \end{aligned} \tag{8}$$

Putting (8) back into (7) gives us

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

which is the optimal solution to (5). Note that we do not need to show dual feasibility and complementary slackness as there are no inequality constraints.

A.3 Proof of Proposition 4.4

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

$$\begin{aligned} \ell_{ij}^* &\overset{(i)}{=} \bar{p}_{ij} + \frac{1}{|S_i|} \left(1 - \sum_{j' \in S_i} \bar{p}_{ij'} \right) \\ &\overset{(ii)}{=} \frac{\alpha_{ij}}{\|\boldsymbol{\alpha}_i\|_1} + \frac{1}{|S_i|} \left(1 - \sum_{j' \in S_i} \frac{\alpha_{ij'}}{\|\boldsymbol{\alpha}_i\|_1} \right) \\ &\overset{(iii)}{=} \frac{f_j(\boldsymbol{x}_i;\boldsymbol{\theta}) + 1}{\|\boldsymbol{\alpha}_i\|_1} + \frac{1}{|S_i|} \left(1 - \sum_{j' \in S_i} \frac{f_{j'}(\boldsymbol{x}_i;\boldsymbol{\theta}) + 1}{\|\boldsymbol{\alpha}_i\|_1} \right) \\ &\overset{(iv)}{=} \frac{f_j(\boldsymbol{x}_i;\boldsymbol{\theta})}{\|\boldsymbol{\alpha}_i\|_1} + \frac{1}{\|\boldsymbol{\alpha}_i\|_1} + \frac{1}{|S_i|} \left(1 - \frac{|S_i|}{\|\boldsymbol{\alpha}_i\|_1} - \sum_{j' \in S_i} \frac{f_{j'}(\boldsymbol{x}_i;\boldsymbol{\theta})}{\|\boldsymbol{\alpha}_i\|_1} \right) \\ &\overset{(v)}{=} \underbrace{\frac{f_j(\boldsymbol{x}_i;\boldsymbol{\theta})}{\|\boldsymbol{\alpha}_i\|_1}}_{=\mathfrak{b}_{ij}} + \underbrace{\frac{1}{|S_i|}}_{=\mathfrak{a}_{ij}} \underbrace{\left(1 - \sum_{j' \in S_i} \frac{f_{j'}(\boldsymbol{x}_i;\boldsymbol{\theta})}{\|\boldsymbol{\alpha}_i\|_1} \right)}_{=\mathfrak{u}_i} \\ &\overset{(vi)}{=} \mathfrak{b}_{ij} + \mathfrak{a}_{ij} \mathfrak{u}_i, \end{aligned}$$

where (i) holds by Proposition 4.3, (ii) by $\bar{\boldsymbol{p}}_i = \boldsymbol{\alpha}_i / \|\boldsymbol{\alpha}_i\|_1$, (iii) by $\boldsymbol{\alpha}_i = f(\boldsymbol{x}_i;\boldsymbol{\theta}) + 1$, (iv) by separating summands, (v) by simplifying, and (vi) by the definitions in Proposition 4.4. 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.

A.4 Proof of Proposition 4.5

The proof of Proposition 4.5 proceeds as follows:

$$
\begin{aligned}
0 &\leq \|\boldsymbol{\ell}_i^{(t+1)} - \boldsymbol{\ell}_i^{(t)}\|_2^2 \\
&\overset{(i)}{=} \sum_{j \in S_i} \left(\bar{p}_{ij}^{(t+1)} + \frac{1}{|S_i|} \left(1 - \sum_{j' \in S_i} \bar{p}_{ij'}^{(t+1)} \right) - \bar{p}_{ij}^{(t)} - \frac{1}{|S_i|} \left(1 - \sum_{j' \in S_i} \bar{p}_{ij'}^{(t)} \right) \right)^2 \\
&\overset{(ii)}{=} \sum_{j \in S_i} \left(\left(\bar{p}_{ij}^{(t+1)} - \bar{p}_{ij}^{(t)} \right) + \frac{1}{|S_i|} \sum_{j' \in S_i} \left(\bar{p}_{ij'}^{(t)} - \bar{p}_{ij'}^{(t+1)} \right) \right)^2 \\
&\overset{(iii)}{=} \sum_{j \in S_i} \left(\bar{p}_{ij}^{(t+1)} - \bar{p}_{ij}^{(t)} \right)^2 + \frac{2}{|S_i|} \sum_{j \in S_i} \left(\bar{p}_{ij}^{(t+1)} - \bar{p}_{ij}^{(t)} \right) \sum_{j' \in S_i} \left(\bar{p}_{ij'}^{(t)} - \bar{p}_{ij'}^{(t+1)} \right) \\
&\quad + \frac{1}{|S_i|^2} \sum_{j \in S_i} \left(\sum_{j' \in S_i} \left(\bar{p}_{ij'}^{(t)} - \bar{p}_{ij'}^{(t+1)} \right) \right)^2 \\
&\overset{(iv)}{=} \sum_{j \in S_i} \left(\bar{p}_{ij}^{(t+1)} - \bar{p}_{ij}^{(t)} \right)^2 - \frac{2}{|S_i|} \left(\sum_{j \in S_i} \left(\bar{p}_{ij}^{(t+1)} - \bar{p}_{ij}^{(t)} \right) \right)^2 + \frac{1}{|S_i|} \left(\sum_{j' \in S_i} \left(\bar{p}_{ij'}^{(t)} - \bar{p}_{ij'}^{(t+1)} \right) \right)^2 \\
&\overset{(v)}{=} \sum_{j \in S_i} \left(\bar{p}_{ij}^{(t+1)} - \bar{p}_{ij}^{(t)} \right)^2 - \frac{1}{|S_i|} \left(\sum_{j \in S_i} \left(\bar{p}_{ij}^{(t+1)} - \bar{p}_{ij}^{(t)} \right) \right)^2 \\
&\overset{(vi)}{\leq} \sum_{j \in S_i} \left(\bar{p}_{ij}^{(t+1)} - \bar{p}_{ij}^{(t)} \right)^2 \\
&\overset{(vii)}{\leq} \sum_{j=1}^{k} \left(\bar{p}_{ij}^{(t+1)} - \bar{p}_{ij}^{(t)} \right)^2 \\
&= \|\bar{\boldsymbol{p}}_i^{(t+1)} - \bar{\boldsymbol{p}}_i^{(t)}\|_2^2,
\end{aligned}
$$

where (i) holds by Proposition 4.3, (ii) by reordering, (iii) by the binomial theorem, (iv) by

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

and (v) by

$$
\left(\sum_{j \in S_i} \left(\bar{p}_{ij}^{(t)} - \bar{p}_{ij}^{(t+1)} \right) \right)^2 = \left(- \sum_{j \in S_i} \left(\bar{p}_{ij}^{(t+1)} - \bar{p}_{ij}^{(t)} \right) \right)^2 = \left(\sum_{j \in S_i} \left(\bar{p}_{ij}^{(t+1)} - \bar{p}_{ij}^{(t)} \right) \right)^2.
$$

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

A.5 Proof of Proposition 4.6

The proof first shows that ℓ^*_{ij} is a feasible solution for the constraints in (5) and then establishes that ℓ^*_{ij} is optimal.

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

Optimality. As the cross-entropy loss $\mathcal{L}_{\text{CE}}$ in Sect. 4.7 is a linear combination of ℓ_{ij} with coefficients $\Psi_{ij} = \psi(\|\boldsymbol{\alpha}_i\|_1) - \psi(\alpha_{ij})$ for fixed $i \in [n]$ and $\ell_{ij} \in [0, 1]$, we minimize $\mathcal{L}_{\text{CE}}$ by assigning all label weight to the minimal coefficient Ψ_{ij}, that is, $\arg\min_{j \in S_i} \Psi_{ij}$.

Appendix B. Experiments

This section augments Sect. 5 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 & Beringer, 2005): We use $k = 10$ nearest neighbors.
- PLSVM (Nguyen & Caruana, 2008): We use the PEGASOS optimizer (Shalev-Shwartz et al., 2007) and $\lambda = 1$.
- IPAL (Zhang & 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 (LV et al., 2020; Hoerl & Kennard, 1970): 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.
- CAVL (Zhang et al., 2022): 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 (LV et al., 2020; Srivastava et al., 2014): 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.

B.2. Datasets

Table 4 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 5 augments Table 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

Table 4 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
NotMNIST	70 000	784	10	6.342	0.00

Table 5 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)

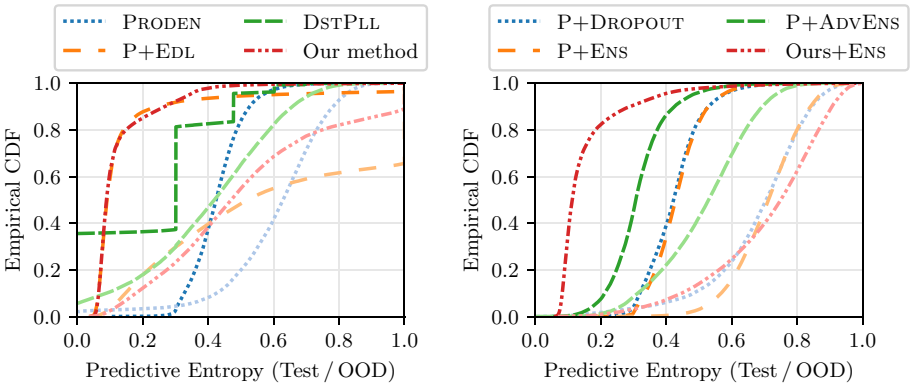

Fig. 1 Empirical CDF of the normalized entropy (range 0–1) of predictions on MNIST (darker color) and NotMNIST (lighter color) for models trained on MNIST. The left plot shows the four best non-ensemble approaches according to Table 2 (highest metrics). We exclude methods that are too similar, for example, PRODEN-L2 and RC behave similarly to PRODEN, which is shown. All methods' performances can be observed in Table 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)

ensemble methods separately. Our proposed algorithms, which are indicated by the triangles, perform comparably on all considered datasets.

B.4. Adversarial perturbations

To complement Table 2 in the main text, Fig. 1 provides the empirical cumulative distribution functions on the test and OOD set of the four best non-ensemble methods (regarding Table 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 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 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).

Acknowledgements We thank the anonymous reviewers for their constructive comments. This work was supported by the German Research Foundation (DFG) Research Training Group GRK 2153: *Energy Status Data --- Informatics Methods for its Collection, Analysis and Exploitation* and by the KiKIT (The Pilot Program for Core-Informatics at the KIT) of the Helmholtz Association.

Author contributions T.F. and F.K. defined the scope of the work. T.F. conducted all experiments and drafted the main manuscript. T.F. and F.K. revised and reviewed the manuscript.

Funding Open Access funding enabled and organized by Projekt DEAL. This work was supported by the German Research Foundation (DFG) Research Training Group GRK 2153: *Energy Status Data — Informatics Methods for its Collection, Analysis and Exploitation* and by the KiKIT (The Pilot Program for Core-Informatics at the KIT) of the Helmholtz Association.

Data availability All code and data is available at https://github.com/mathefuchs/robust-pll.

Declarations

Conflict of interest The authors declare no competing interests.

References

Ao, S., Rueger, S., & Siddharthan, A. (2023). Two sides of miscalibration: Identifying over and under-confidence prediction for network calibration. In: *Uncertainty in Artificial Intelligence*, vol. 216, pp. 77–87.

Berkmans, T. J., & Karthick, S. (2023). 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.

Bishop, C. M. (2007). *Pattern recognition and machine learning*. Springer.

Briggs, F., Fern, X Z., & Raich, R. (2012). Rank-loss support instance machines for MIML instance annotation. In: Conference on Knowledge Discovery and Data Mining, pp. 534–542.

Bulatov, Y. (2011). NotMNIST dataset. http://yaroslavvb.blogspot.com/2011/09/notmnist-dataset.html. Accessed 13 Jun 2024

Cabannes, V., Rudi, A., & Bach, F. R. (2020). Structured prediction with partial labelling through the infimum loss. In: *International Conference on machine learning*, vol. 119, pp. 1230–1239

Cheng, D., Xiang, S., Shang, C., Zhang, Y., Yang, F., & Zhang, L. (2020). Spatio-temporal attention-based neural network for credit card fraud detection. In: *AAAI Conference on Artificial Intelligence*, pp. 362–369.

Clanuwat, T., Bober-Irizar, M., Kitamoto, A., Lamb, A., Yamamoto, K., & Ha, D. (2018). Deep learning for classical Japanese literature. Technical report. arXiv:1812.01718.

Cour, T., Sapp, B., & Taskar, B. (2011). Learning from partial labels. *Journal of Machine Learning Research, 12*, 1501–1536.

Dempster, A. P. (1967). Upper and lower probabilities induced by a multivalued mapping. *Annals of Statistic, 28*, 325–339.

Fan, J., Huang, L., Gong, C., You, Y., Gan, M., & Wang, Z. (2024). KMT-PLL: K-means cross-attention transformer for partial label learning. *Transactions on Neural Networks and Learning Systems*, *2*, 2789–2800.

Feng, L., & An, B.(2019). Partial label learning with self-guided retraining. In: *AAAI Conference on Artificial Intelligence*, pp. 3542–3549.

Feng, L., Lv, J., Han, B., Xu, M., Niu, G., Geng, X., An, B., & Sugiyama, M. (2020). Provably consistent partial-label learning. In: *Advances in neural information processing systems*, pp. 10948–10960.

Fuchs, T., & Kalinke, F. (2025). Partial-label learning with conformal candidate cleaning. Technical report. arXiv:2502.07661

Fuchs, T., Kalinke, F., & Bohm, K. (2025). Partial-label learning with a reject option. *Transactions on Machine Learning Research*.

Gong, X., Bisht, N., & Xu, G. (2024). Does label smoothing help deep partial label learning? In: *International Conference on machine learning*.

Goodfellow, I. J., Shlens, J., & Szegedy, C. (2015). Explaining and harnessing adversarial examples. In: *International Conference on learning representations*.

Grandvalet, Y. (2002). Logistic regression for partial labels. In: *Information processing and management of uncertainty in knowledge-based systems*. https://storm.cis.fordham.edu/~gweiss/selected-papers/logistic-regression-partial-labels.pdf

Guillaumin, M., Verbeek, J., spsampsps Schmid, C. (2010). Multiple instance metric learning from automatically labeled bags of faces. In: *European Conference on Computer Vision*, vol. 6311, pp. 634–647.

He, K., Zhang, X., Ren, S., & Sun, J.(2016). Deep residual learning for image recognition. In: *Conference on computer vision and pattern recognition*, pp. 770–778.

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

Huiskes, M. J., & Lew, M. S. (2008). The MIR flickr retrieval evaluation. In: *International conference on multimedia information retrieval*, pp. 39–43

Hüllermeier, E., & Beringer, J. (2005). Learning from ambiguously labeled examples. *Intelligent Data Analysis, 10*(5), 168–179.

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

Jin, R., & Ghahramani, Z. (2002). Learning with multiple labels. In: *Advances in neural information processing systems*, pp. 897–904.

Jøsang, A. (2016). *Subjective logic—A formalism for reasoning under uncertainty*. Springer.

Jürgens, M., Meinert, N., Bengs, V., Hüllermeier, E., & Waegeman, W. (2024). Is epistemic uncertainty faithfully represented by evidential deep learning methods? In: *International Conference on Machine Learning*.

Kendall, A., & Gal, Y.(2017). What uncertainties do we need in Bayesian deep learning for computer vision? In: *Advances in Neural Information Processing Systems*, pp. 5574–5584.

Kingma, D. P., & Welling, M. (2014). Auto-encoding variational bayes. In: *International Conference on learning representations*.

Lakshminarayanan, B., Pritzel, A., & Blundell, C. (2017). Simple and scalable predictive uncertainty estimation using deep ensembles. In: *Advances in neural information processing systems*, pp. 6402–6413.

Lambrou, A., Papadopoulos, H., & Gammerman, A. (2011). Reliable confidence measures for medical diagnosis with evolutionary algorithms. *Transactions on Information Technology in Biomedicine, 15*(1), 93–99.
LeCun, Y., Cortes, C., & Burges, C. J. C. (1999). The MNIST database of handwritten digits. http://yann.lecun.com/exdb/mnist. Accessed 13 Jun 2024
LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE, 86*(11), 2278–2324.
Liu, L., & Dietterich, T. G. (2012). A conditional multinomial mixture model for superset label learning. In: *Advances in neural information processing systems*, pp. 557–565.
Lv, J., Xu, M., Feng, L., Niu, G., Geng, X., & Sugiyama, M. (2020). Progressive identification of true labels for partial-label learning. In: *International conference on machine learning*, vol. 119, pp. 6500–6510.
Madry, A., Makelov, A., Schmidt, L., Tsipras, D., & Vladu, A. (2018). Towards deep learning models resistant to adversarial attacks. In: *International conference on learning representations*.
Moosavi-Dezfooli, S., Fawzi, A., & Frossard, P. (2016). DeepFool: A simple and accurate method to fool deep neural networks. In: *Conference on computer vision and pattern recognition*, pp. 2574–2582.
Mortier, T., Bengs, V., Hüllermeier, E., Luca, S., & Waegeman, W. (2023). On the calibration of probabilistic classifier sets. In: *International Conference on artificial intelligence and Statistics*, vol. 206, pp. 8857–8870.
Nguyen, N., & Caruana, R. (2008). Classification with partial labels. In: *Conference on knowledge discovery and data mining*, pp. 551–559.
Ni, P., Zhao, S., Dai, Z., Chen, H., & Li, C. (2021). Partial label learning via conditional-label-aware disambiguation. *Journal of Computer Science and Technology, 36*(3), 590–605.
Penny, W. D. (2001). *Kullback-Leibler divergences of Normal, Gamma*. Wellcome Department of Cognitive Neurology: Dirichlet and Wishart densities. Technical report.
Reamaroon, N., Sjoding, M. W., Lin, K., Iwashyna, T. J., & Najarian, K. (2019). Accounting for label uncertainty in machine learning for detection of acute respiratory distress syndrome. *Journal of Biomedical and Health Informatics, 23*(1), 407–415.
Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature, 323*(6088), 533–536.
Sale, Y., Caprio, M., & Hüllermeier, E. (2023). Is the volume of a credal set a good measure for epistemic uncertainty? In: *Uncertainty in Artificial Intelligence*, vol. 216, pp. 1795–1804.
Sensoy, M., Kaplan, L. M., & Kandemir, M. (2018). Evidential deep learning to quantify classification uncertainty. In: *Advances in neural information processing systems*, pp. 3183–3193.
Shafer, G. (1986). The combination of evidence. *Intelligent Systems, 1*(3), 155–179.
Shalev-Shwartz, S., Singer, Y., & Srebro, N. (2007). Pegasos: Primal Estimated sub-GrAdient SOlver for SVM. In: *International Conference on machine learning*, vol. 227, pp. 807–814.
Srivastava, N., Hinton, G. E., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research, 15*(1), 1929–1958.
Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I.J., & Fergus, R. (2014). Intriguing properties of neural networks. In: *International conference on learning representations*.
Tang, C., & Zhang, M. (2017). Confidence-rated discriminative partial label learning. In: *AAAI conference on artificial intelligence*, pp. 2611–2617.
Tian, S., Wei, H., Wang, Y., & Feng, L. (2024). CroSel: Cross selection of confident pseudo labels for partial-label learning. In: *Conference on computer vision and pattern recognition*.
Wang, H., Xiao, R., Li, Y., Feng, L., Niu, G., Chen, G., & Zhao, J. (2022). PiCO: Contrastive label disambiguation for partial label learning. In: *International conference on learning representations*.
Werbos, P. (1974). Beyond regression: New tools for prediction and analysis in the behavioral sciences. PhD thesis, Harvard University.
Wimmer, L., Sale, Y., Hofman, P., Bischl, B., & Hüllermeier, E. (2023). Quantifying aleatoric and epistemic uncertainty in machine learning: Are conditional entropy and mutual information appropriate measures? In: *Uncertainty in Artificial Intelligence*, vol. 216, pp. 2282–2292.
Wu, X., & Zhang, M. (2018). Towards enabling binary decomposition for partial label learning. In: *International Joint Conference on Artificial Intelligence*, pp. 2868–2874.
Xiang, S., Zhu, M., Cheng, D., Li, E., Zhao, R., Ouyang, Y., Chen, L., & Zheng, Y. (2023). Semi-supervised credit card fraud detection via attribute-driven graph representation. In: *AAAI Conference on artificial intelligence*, pp. 14557–14565.
Xiao, H., Rasul, K., & Vollgraf, R. (2018). Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms. Technical report . arXiv:1708.07747
Xu, N., Liu, B., Lv, J., Qiao, C., & Geng, X. (2023). Progressive purification for instance-dependent partial label learning. In: *International Conference on machine learning*, vol. 202, pp. 38551–38565.

Xu, N., Lv, J., & Geng, X. (2019). Partial label learning via label enhancement. In: *AAAI Conference on artificial intelligence*, pp. 5557–5564.
Xu, N., Qiao, C., Geng, X., & Zhang, M. (2021). Instance-dependent partial label learning. In: *Advances in neural information processing systems*, pp. 27119–27130.
Yang, F., Wang, H., Mi, H., Lin, C., & Cai, W. (2009). Using random forest for reliable classification and cost-sensitive learning for medical diagnosis. *BMC Bioinformatics,10*(1), 1–14.
Yu, F., & Zhang, M. (2017). Maximum margin partial label learning. In: *Asian Conference on machine learning*, vol. 106, pp. 573–593.
Zeng, Z., Xiao, S., Jia, K., Chan, T., Gao, S., Xu, D., & Ma, Y. (2013). Learning by associating ambiguously labeled images. In: *Conference on computer vision and pattern recognition*, pp. 708–715.
Zhang, F., Feng, L., Han, B., Liu, T., Niu, G., Qin, T., & Sugiyama, M. (2022). Exploiting class activation value for partial-label learning. In: *International Conference on learning representations*
Zhang, M., & Yu, F. (2015). Solving the partial label learning problem: An instance-based approach. In: *International Joint Conference on artificial intelligence*, pp. 4048–4054.
Zhang, Y., Zheng, S., Wu, P., Goswami, M., & Chen, C. (2021). Learning with feature-dependent label noise: A progressive approach. In: *International Conference on learning representations*.
Zhang, M., Yu, F., & Tang, C. (2017). Disambiguation-free partial label learning. *Transactions on Knowledge and Data Engineering, 29*(10), 2155–2167.