

Peerannot: classification for crowd-sourced image datasets with Python

Tanguy Lefort  IMAG, Univ Montpellier, CNRS, Inria, LIRMM
Benjamin Charlier IMAG, Univ Montpellier, CNRS
Alexis Joly  Inria, LIRMM, Univ Montpellier, CNRS
Joseph Salmon  IMAG, Univ Montpellier, CNRS, IUF

Date published: 2024-01-18 Last modified: 2024-01-18

Abstract

Crowdsourcing is a quick and easy way to collect labels for large datasets, involving many workers. However, workers often disagree with each other. Sources of error can arise from the workers' skills, but also from the intrinsic difficulty of the task. We present `peerannot`: a Python library for managing and learning from crowdsourced labels for classification. Our library allows users to aggregate labels from common noise models or train a deep learning-based classifier directly from crowdsourced labels. In addition, we provide an identification module to easily explore the task difficulty of datasets and worker capabilities.

Keywords: crowdsourcing, label noise, task difficulty, worker ability, classification

1 Contents

2	1 Introduction: crowdsourcing in image classification	2
3	2 Notation and package structure	3
4	2.1 Crowdsourcing notation	3
5	2.2 Storing crowdsourced datasets in <code>peerannot</code>	4
6	3 Aggregation strategies in crowdsourcing	6
7	3.1 Classical models	6
8	3.1.1 Majority vote (MV)	6
9	3.1.2 Naive soft (NS)	7
10	3.1.3 Dawid and Skene (DS)	7
11	3.1.4 Variations around the DS model	8
12	3.1.5 Generative model of Labels, Abilities, and Difficulties (GLAD)	8
13	3.1.6 Aggregation strategies in <code>peerannot</code>	9
14	3.2 Experiments and evaluation of label aggregation strategies	9
15	3.2.1 Simulated independent mistakes	10
16	3.2.2 Simulated correlated mistakes	11
17	3.3 More on confusion matrices in simulation settings	12

¹Corresponding author: tanguy.lefort@umontpellier.fr

18	4 Learning from crowdsourced tasks	13
19	4.1 Popular models	13
20	4.1.1 CrowdLayer	13
21	4.1.2 CoNAL	13
22	4.2 Prediction error when learning from crowdsourced tasks	14
23	4.3 Use case with peerannot on real datasets	14
24	5 Identifying tasks difficulty and worker abilities	15
25	5.1 Exploring tasks' difficulty	16
26	5.1.1 CIFAR-1OH dataset	16
27	5.1.2 LabelMe dataset	17
28	5.2 Identification of worker reliability and task difficulty	17
29	5.2.1 CIFAR-10H	18
30	5.2.2 LabelMe	19
31	6 Conclusion	20
32	7 Appendix	20
33	7.1 Supplementary simulation: Simulated mistakes with discrete difficulty levels on tasks	20
34	7.2 Comparison with other libraries	21
35	7.3 Examples of images in CIFAR-10H and Labelme	23
36	7.4 Case study with bird sound classification	23

37 **1 Introduction: crowdsourcing in image classification**

38 Image datasets widely use crowdsourcing to collect labels, involving many workers who can annotate
 39 images for a small cost (or even free for instance in citizen science) and faster than using expert
 40 labeling. Many classical datasets considered in machine learning have been created with human
 41 intervention to create labels, such as CIFAR-10, (Krizhevsky and Hinton 2009), ImageNet (Deng et
 42 al. 2009) or Pl@ntnet (Garcin et al. 2021) in image classification, but also COCO (Lin et al. 2014),
 43 solar photovoltaic arrays (Kasmi et al. 2023) or even macro litter (Chagneux et al. 2023) in image
 44 segmentation and object counting.

45 Crowdsourced datasets induce at least three major challenges to which we contribute with peerannot:

- 46 1) **How to aggregate multiple labels into a single label from crowdsourced tasks?** This
 47 occurs for example when dealing with a single dataset that has been labeled by multiple
 48 workers with disagreements. This is also encountered with other scoring issues such as polls,
 49 reviews, peer-grading, etc. In our framework this is treated with the aggregate command,
 50 which given multiple labels, infers a label. From aggregated labels, a classifier can then be
 51 trained using the train command.
- 52 2) **How to learn a classifier from crowdsourced datasets?** Where the first question is bound
 53 by aggregating multiple labels into a single one, this considers the case where we do not need
 54 a single label to train on, but instead train a classifier on the crowdsourced data, with the
 55 motivation to perform well on a testing set. This end-to-end vision is common in machine
 56 learning, however, it requires the actual tasks (the images, texts, videos, etc.) to train on – and in
 57 crowdsourced datasets publicly available, they are not always available. This is treated with the
 58 aggregate-deep command that runs strategies where the aggregation has been transformed
 59 into a deep learning optimization problem.
- 60 3) **How to identify good workers in the crowd and difficult tasks?** When multiple answers
 61 are given to a single task, looking for who to trust for which type of task becomes necessary

62 to estimate the labels or later train a model with as few noise sources as possible. The module
 63 `identify` uses different scoring metrics to create a worker and/or task evaluation. This is
 64 particularly relevant considering the gamification of crowdsourcing experiments (Servajean et
 65 al. 2016)

66 The library `peerannot` addresses these practical questions within a reproducible setting. Indeed,
 67 the complexity of experiments often leads to a lack of transparency and reproducible results for
 68 simulations and real datasets. We propose standard simulation settings with explicit implementation
 69 parameters that can be shared. For real datasets, `peerannot` is compatible with standard neural net-
 70 work architectures from the `Torchvision` (Marcel and Rodriguez 2010) library and `Pytorch` (Paszke
 71 et al. 2019), allowing a flexible framework with easy-to-share scripts to reproduce experiments.

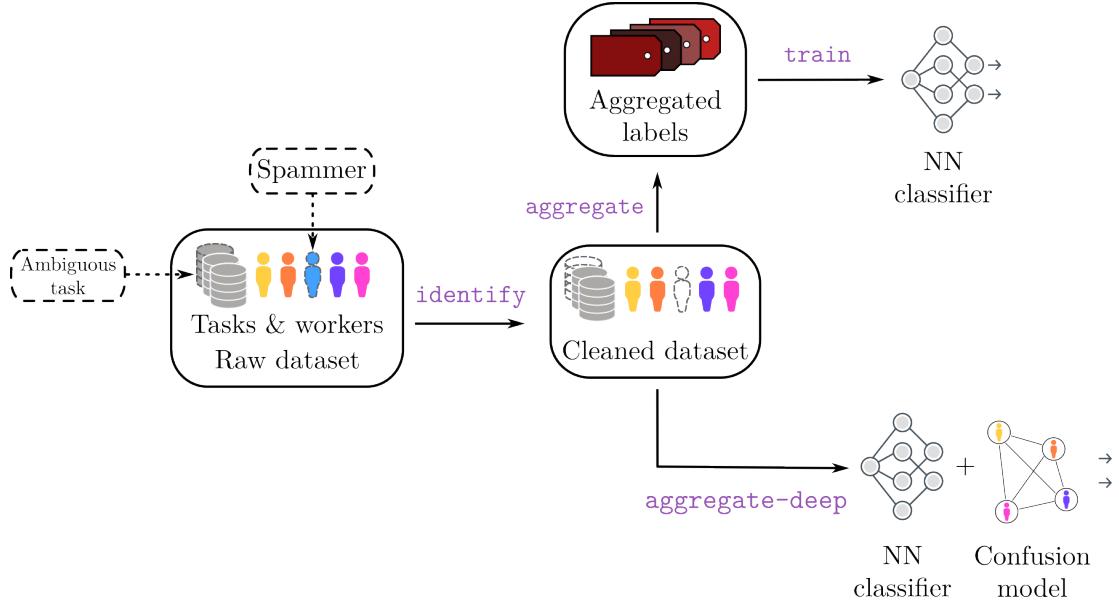


Figure 1: From crowdsourced labels to training a classifier neural network, the learning pipeline using the `peerannot` library. An optional preprocessing step using the `identify` command allows us to remove the worst-performing workers or images that can not be classified correctly (very bad quality for example). Then, from the cleaned dataset, the `aggregate` command may generate a single label per task from a prescribed strategy. From the aggregated labels we can train a neural network classifier with the `train` command. Otherwise, we can directly train a neural network classifier that takes into account the crowdsourcing setting in its architecture using `aggregate-deep`.

72 2 Notation and package structure

73 2.1 Crowdsourcing notation

74 Let us consider the classical supervised learning classification framework. A training set $\mathcal{D} =$
 75 $\{(x_i, y_i^*)\}_{i=1}^{n_{\text{task}}}$ is composed of n_{task} tasks $x_i \in \mathcal{X}$ (the feature space) with (unknown) true label $y_i^* \in$
 76 $[K] = \{1, \dots, K\}$ one of the K possible classes. In the following, the tasks considered are generally
 77 RGB images. We use the notation $\sigma(\cdot)$ for the softmax function. In particular, given a classifier \mathcal{C}
 78 with logits outputs, $\sigma(\mathcal{C}(x_i))_{[1]}$ represents the largest probability and we can sort the probabilities
 79 as $\sigma(\mathcal{C}(x_i))_{[1]} \geq \sigma(\mathcal{C}(x_i))_{[2]} \geq \dots \geq \sigma(\mathcal{C}(x_i))_{[K]}$. The indicator function is denoted $\mathbf{1}(\cdot)$. We use
 80 the i index notation to range over the different tasks and the j index notation for the workers
 81 in the crowdsourcing experiment. Note that indices start at position 1 in the equation to follow
 82 mathematical standard notation but it should be noted that, as this is a Python library, in the code

83 indices start at the 0 position.

84 With crowdsourced data the true label of a task x_i , denoted y_i^* is unknown, and there is no single
85 label that can be trusted as in standard supervised learning (even on the train set!). Instead, there is a
86 crowd of n_{worker} workers from which multiple workers $(w_j)_j$ propose a label $(y_i^{(j)})_j$. These proposed
87 labels are used to estimate the true label. The set of workers answering the task x_i is denoted by

$$\mathcal{A}(x_i) = \{j \in [n_{\text{worker}}] : w_j \text{ answered } x_i\}. \quad (1)$$

88 The cardinal $|\mathcal{A}(x_i)|$ is called the feedback effort on the task x_i . Note that the feedback effort can not
89 exceed the total number of workers n_{worker} . Similarly, one can adopt a worker point of view: the set
90 of tasks answered by a worker w_j is denoted

$$\mathcal{T}(w_j) = \{i \in [n_{\text{task}}] : w_j \text{ answered } x_i\}. \quad (2)$$

91 The cardinal $|\mathcal{T}(w_j)|$ is called the workload of w_j . The final dataset can then be decomposed as:

$$\mathcal{D}_{\text{train}} := \bigcup_{i \in [n_{\text{task}}]} \{(x_i, (y_i^{(j)})) \text{ for } j \in \mathcal{A}(x_i)\} = \bigcup_{j \in [n_{\text{worker}}]} \{(x_i, (y_i^{(j)})) \text{ for } i \in \mathcal{T}(w_j)\} .$$

92 In this article, we do not address the setting where workers report their self-confidence (Yasmin et al.
93 2022), nor settings where workers are presented a trapping set – *i.e.*, a subset of tasks where the true
94 label is known to evaluate them with known labels (Khattak 2017).

95 2.2 Storing crowdsourced datasets in peerannot

96 Crowdsourced datasets come in various forms. To store [crowdsourcing datasets](#) efficiently and in a
97 standardized way, peerannot proposes the following structure, where each dataset corresponds to a
98 folder. Let us set up a toy dataset example to understand the data structure and how to store it.

Listing 1 Dataset storage tree structure.

```
datasetname
    train
        ...
        images
        ...
    val
    test
    metadata.json
    answers.json
```

99 The `answers.json` file stores the different votes for each task as described in Figure 2. This `.json`
100 is the rosetta stone between the task ids and the images. It contains the tasks' id, the workers's
101 id and the proposed label for each given vote. Furthermore, storing labels in a dictionary is more
102 memory-friendly than having an array of size $(n_{\text{task}}, n_{\text{worker}})$ and writing $y_i^{(j)} = -1$ when the
103 worker w_j did not see the task x_i and $y_i^{(j)} \in [K]$ otherwise.

104 In Figure 2, there are three tasks, $n_{\text{worker}} = 4$ workers and $K = 2$ classes. Any available task should
105 be stored in a single file whose name follows the convention described in Listing 1. These files are
106 spread into a `train`, `val` and `test` subdirectories as in [ImageFolder](#) datasets from [torchvision](#)

107 Finally, a `metadata.json` file includes relevant information related to the crowdsourcing experiment
108 such as the number of workers, the number of tasks, *etc*. For example, a minimal `metadata.json` file
109 for the toy dataset presented in Figure 2 is:

The figure shows two representations of the same dataset. On the left, a JSON object with three entries, each containing a smiley face icon and a map of worker annotations (0 or 1) for four categories: not smiling (orange), smiling (green), neutral (purple), and happy (yellow). On the right, a 3x5 grid of icons representing the data collected by four workers. The columns represent the four categories from the JSON. The rows represent three different images. Each cell contains either a smiley face icon (if the worker annotated it as smiling) or a red X (if the worker did not). A legend at the top indicates that K=2, 0 means not smiling and 1 means smiling.

Figure 2: Data storage for the toy-data crowdsourced dataset, a binary classification problem ($K = 2$, smiling/not smiling) on recognizing smiling faces. (left: how data is stored in peerannot in a file `answers.json`, right: data collected)

```
{
  "name": "toy-data",
  "n_classes": 2,
  "n_workers": 4,
  "n_tasks": 3
}
```

110 The toy-data example dataset is available as an example [in the peerannot repository](#). Classical
 111 datasets in crowdsourcing such as CIFAR-10H (Peterson et al. 2019) and LabelMe (Rodrigues, Pereira,
 112 and Ribeiro 2014) can be installed directly using peerannot. To install them, run the `install`
 113 command from peerannot:

114 For both CIFAR-10H and LabelMe, the dataset was originally released for standard supervised
 115 learning (classification). Both datasets have been reannotated by a crowd of workers. These labels are
 116 used as true labels in evaluations and visualizations. Examples of CIFAR-10H images are available in
 117 Figure 16, and LabelMe examples in Figure 17 in Appendix. Crowdsourcing votes, however, bring
 118 information about possible confusions (see Figure 3 for an example with CIFAR-10H and Figure 4
 119 with LabelMe).

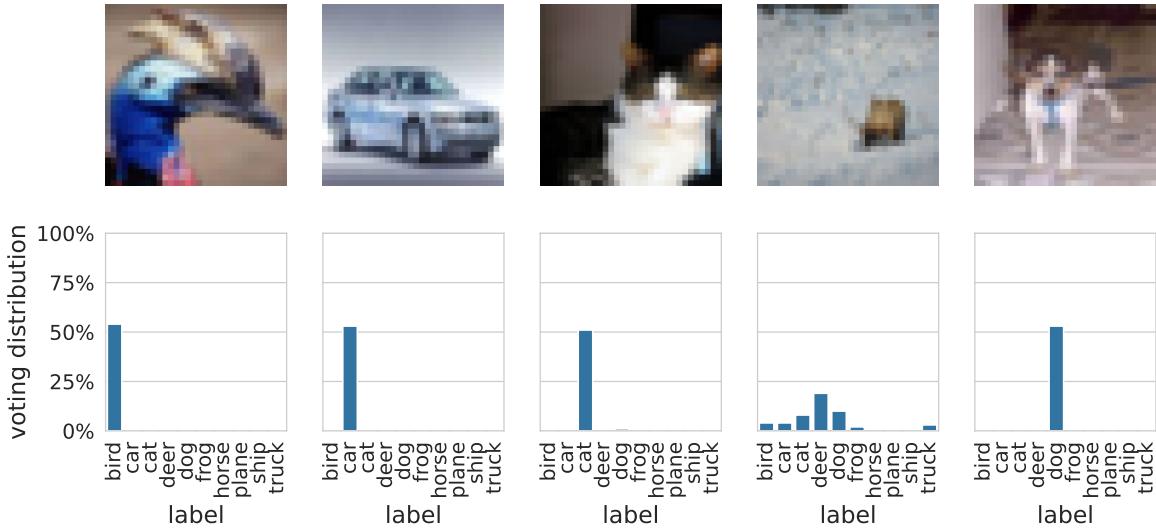


Figure 3: Example of crowdsourced images from CIFAR-10H. Each task has been labeled by multiple workers. We display the associated voting distribution over the possible classes.

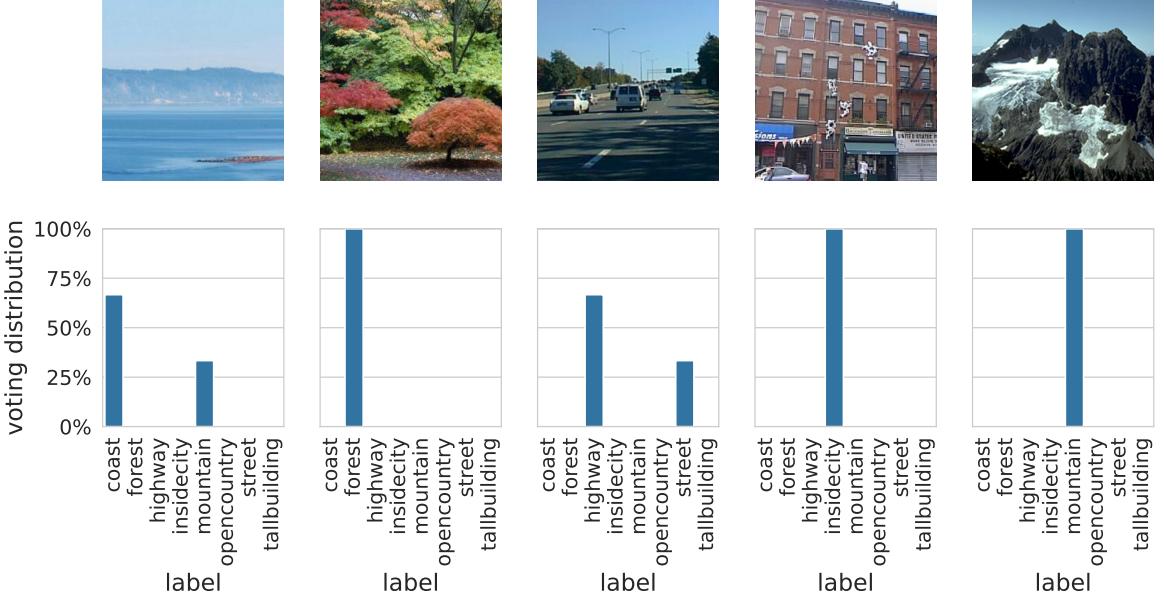


Figure 4: Example of crowdsourced images from LabelMe. Each task has been labeled by multiple workers. We display the associated voting distribution over the possible classes.

120 3 Aggregation strategies in crowdsourcing

121 The first question we address with `peerannot` is: *How to aggregate multiple labels into a single label*
 122 *from crowdsourced tasks?* The aggregation step can lead to two types of learnable labels $\hat{y}_i \in \Delta_K$
 123 (where Δ_K is the simplex of dimension $K - 1$: $\Delta_K = \{p \in \mathbb{R}^K : \sum_{k=1}^K p_k = 1, p_k \geq 0\}$) depending on
 124 the use case for each task $x_i, i = 1, \dots, n_{\text{task}}$:

- 125 • a **hard** label: \hat{y}_i is a Dirac distribution, this can be encoded as a classical label in $[K]$,
- 126 • a **soft** label: $\hat{y}_i \in \Delta_K$ can represent any probability distribution on $[K]$. In that case, each
 127 coordinate of the K -dimensional vector \hat{y}_i represents the probability of belonging to the given
 128 class.

129 Learning from soft labels has been shown to improve learning performance and make the classifier
 130 learn the task ambiguity (Zhang et al. 2018; Peterson et al. 2019; Park and Caragea 2022). However,
 131 crowdsourcing is often used as a stepping stone to create a new dataset. We usually expect a
 132 classification dataset to associate a task x_i to a single label and not a full probability distribution. In
 133 this case, we recommend releasing the anonymous answered labels and the aggregation strategy used
 134 to reach a consensus on a single label. With `peerannot`, both soft and hard labels can be produced.

135 Note that when a strategy produces a soft label, a hard label can be easily induced by taking the
 136 mode, *i.e.*, the class achieving the maximum probability.

137 3.1 Classical models

138 We list below the most classical aggregation strategies used in crowdsourcing.

139 3.1.1 Majority vote (MV)

140 The most intuitive way to create a label from multiple answers for any type of crowdsourced task is
 141 to take the **majority vote** (MV). Yet, this strategy has many shortcomings (James 1998) – there is no

¹⁴² noise model, no worker reliability estimated, no task difficulty involved and especially no way to
¹⁴³ remove poorly performing workers. This standard choice can be expressed as:

$$\hat{y}_i^{\text{MV}} = \operatorname{argmax}_{k \in [K]} \sum_{j \in \mathcal{A}(x_i)} \mathbf{1}_{\{y_i^{(j)} = k\}}.$$

¹⁴⁴ 3.1.2 Naive soft (NS)

¹⁴⁵ One pitfall with MV is that the label produced is hard, hence the ambiguity is discarded by construction.
¹⁴⁶ A simple remedy consists in using the [Naive Soft \(NS\)](#) labeling, *i.e.*, output the empirical distribution
¹⁴⁷ as the task label:

$$\hat{y}_i^{\text{NS}} = \left(\frac{1}{|\mathcal{A}(x_i)|} \sum_{j \in \mathcal{A}(x_i)} \mathbf{1}_{\{y_i^{(j)} = k\}} \right)_{j \in [K]}.$$

¹⁴⁸ With the NS label, we keep the ambiguity, but all workers and all tasks are put on the same level. In
¹⁴⁹ practice, it is known that each worker comes with their abilities, thus modeling this knowledge can
¹⁵⁰ produce better results.

¹⁵¹ 3.1.3 Dawid and Skene (DS)

¹⁵² Refining the aggregation, researchers have proposed a noise model to take into account the workers'
¹⁵³ abilities. The [Dawid and Skene](#)'s (DS) model (Dawid and Skene 1979) is one of the most studied
¹⁵⁴ (Gao and Zhou 2013) and applied (Servajean et al. 2017; Rodrigues and Pereira 2018). These types of
¹⁵⁵ models are most often optimized using EM-based procedures. Assuming the workers are answering
¹⁵⁶ tasks independently, this model boils down to model pairwise confusions between each possible
¹⁵⁷ class. Each worker w_j is assigned a confusion matrix $\pi^{(j)} \in \mathbb{R}^{K \times K}$ as described in Section 3. The
¹⁵⁸ model assumes that for a task x_i , conditionally on the true label $y_i^* = k$ the label distribution of the
¹⁵⁹ worker's answer follows a multinomial distribution with probabilities $\pi_{k,\cdot}^{(j)}$ for each worker. Each
¹⁶⁰ class has a prevalence $\rho_k = \mathbb{P}(y_i^* = k)$ to appear in the dataset. Using the independence between
¹⁶¹ workers, we obtain the following likelihood to maximize, with latent variables $\rho, \pi = \{\pi^{(j)}\}_j$ and
¹⁶² unobserved variables $(y_i^{(j)})_{i,j}$:

$$\arg \max_{\rho, \pi} \prod_{i \in [n_{\text{task}}]} \prod_{k \in [K]} \left[\rho_k \prod_{j \in [n_{\text{worker}}]} \prod_{\ell \in [K]} (\pi_{k,\ell}^{(j)})^{\mathbf{1}_{\{y_i^{(j)} = \ell\}}} \right].$$

¹⁶³ When the true labels are not available, the data comes from a mixture of categorical distributions.
¹⁶⁴ To retrieve ground truth labels and be able to estimate these parameters, Dawid and Skene (1979)
¹⁶⁵ have proposed to consider the true labels as additional unknown parameters. In this case, denoting
¹⁶⁶ $T_{i,k} = \mathbf{1}_{\{y_i^* = k\}}$ the vectors of label class indicators for each task, the likelihood with known true labels
¹⁶⁷ is:

$$\arg \max_{\rho, \pi, T} \prod_{i \in [n_{\text{task}}]} \prod_{k \in [K]} \left[\rho_k \prod_{j \in [n_{\text{worker}}]} \prod_{\ell \in [K]} (\pi_{k,\ell}^{(j)})^{\mathbf{1}_{\{y_i^{(j)} = \ell\}}} \right]^{T_{i,k}}.$$

¹⁶⁸ This framework allows to estimate ρ, π, T with an EM algorithm as follows:

- ¹⁶⁹ • With the MV strategy, get an initial estimate of the true labels T .
- ¹⁷⁰ • Estimate ρ and π knowing T using maximum likelihood estimators.
- ¹⁷¹ • Update T knowing ρ and π using Bayes formula.
- ¹⁷² • Repeat until convergence of the likelihood.

173 The final aggregated soft labels are $\hat{y}_i^{\text{DS}} = T_{i..}$. Note that DS also provides the estimated confusion
 174 matrices $\hat{\pi}^{(j)}$ for each worker w_j .

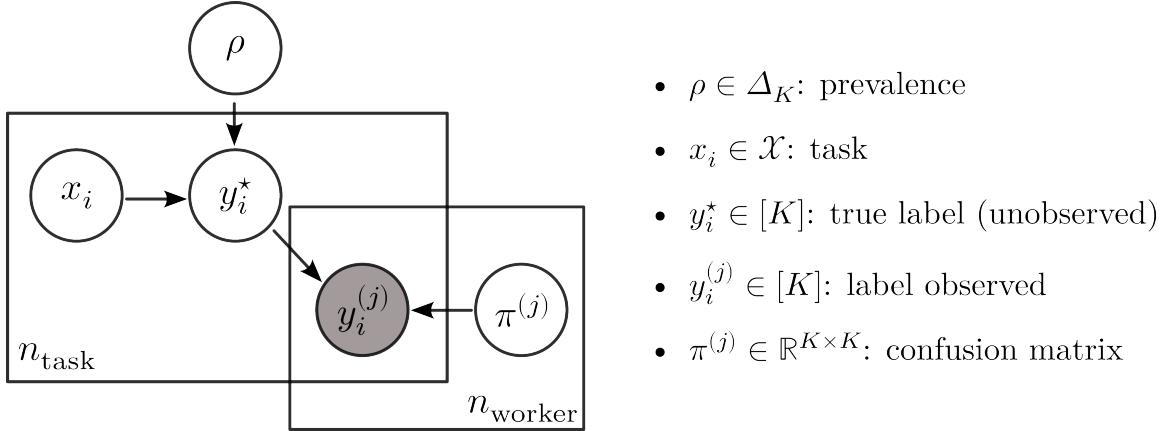


Figure 5: Bayesian [plate notation](#) for the DS model

175 3.1.4 Variations around the DS model

176 Many variants of the DS model have been proposed in the literature, using Dirichlet priors on the
 177 confusion matrices (Passonneau and Carpenter 2014), using $1 \leq L \leq n_{\text{worker}}$ clusters of workers
 178 (Imamura, Sato, and Sugiyama 2018) (DSWC) or even faster implementation that produces only hard
 179 labels (Sinha, Rao, and Balasubramanian 2018).

180 In particular, the DSWC strategy (Dawid and Skene with Worker Clustering) highly reduces the
 181 dimension of the parameters in the DS model. In the original model, there are $K^2 \times n_{\text{worker}}$ parameters
 182 to be estimated for the confusion matrices only. The DSWC model reduces them to $K^2 \times L + L$
 183 parameters. Indeed, there are L confusion matrices $\Lambda = \{\Lambda_1, \dots, \Lambda_L\}$ and the confusion matrix of a
 184 cluster is assumed drawn from a multinomial distribution with weights $(\tau_1, \dots, \tau_L) \in \Delta_L$ over Λ , such
 185 that $\mathbb{P}(\pi^{(j)} = \Lambda_\ell) = \tau_\ell$.

186 3.1.5 Generative model of Labels, Abilities, and Difficulties (GLAD)

187 Finally, we present the [GLAD](#) model (Whitehill et al. 2009) that not only takes into account the
 188 worker's ability, but also the task difficulty in the noise model. The likelihood is optimized using an
 189 EM algorithm to recover the soft label \hat{y}_i^{GLAD} .

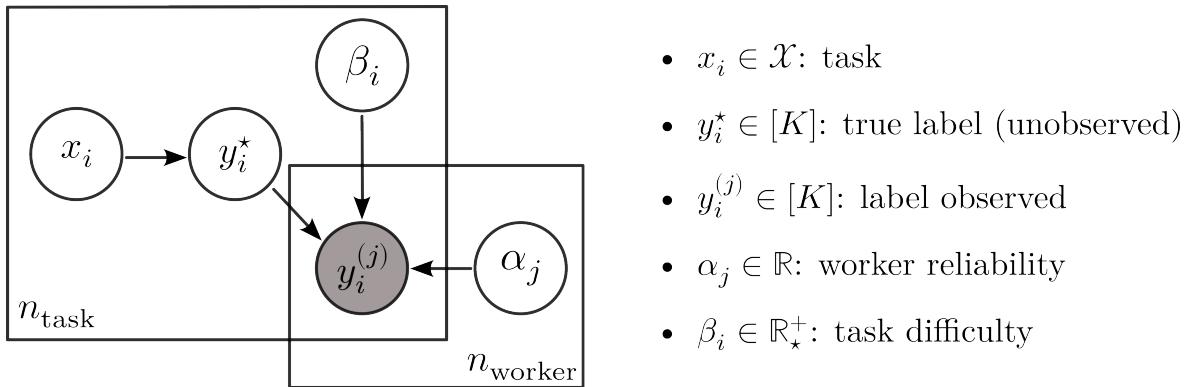


Figure 6: Bayesian [plate notation](#) for the GLAD model

190 Denoting $\alpha_j \in \mathbb{R}$ the worker ability (the higher the better) and $\beta_i \in \mathbb{R}_+^+$ the task's difficulty (the higher
 191 the easier), the model noise is:

$$\mathbb{P}(y_i^{(j)} = y_i^* | \alpha_j, \beta_i) = \frac{1}{1 + \exp(-\alpha_j \beta_i)} .$$

192 GLAD's model also assumes that the errors are uniform across wrong labels, thus:

$$\forall k \in [K], \mathbb{P}(y_i^{(j)} = k | y_i^* \neq k, \alpha_j, \beta_i) = \frac{1}{K-1} \left(1 - \frac{1}{1 + \exp(-\alpha_j \beta_i)} \right) .$$

193 This results in estimating $n_{\text{worker}} + n_{\text{task}}$ parameters.

194 3.1.6 Aggregation strategies in peerannot

195 All of these aggregation strategies – and more – are available in the peerannot library from [the](#)
 196 [peerannot.models module](#). Each model is a class object in its own Python file. It inherits from the
 197 CrowdModel template class and is defined with at least two methods:

- 198 • `run`: includes the optimization procedure to obtain needed weights (e.g., the EM algorithm for
 199 the DS model),
- 200 • `get_probas`: returns the soft labels output for each task.

201 3.2 Experiments and evaluation of label aggregation strategies

202 One way to evaluate the label aggregation strategies is to measure their accuracy. This means that
 203 the underlying ground truth must be known – at least for a representative subset. This is the case in
 204 simulation settings where the ground truth is available. As the set of n_{task} can be seen as a training
 205 set for a future classifier, we denote this metric AccTrain on a dataset \mathcal{D} for some given aggregated
 206 label $(\hat{y}_i)_i$ as:

$$\text{AccTrain}(\mathcal{D}) = \frac{1}{|\mathcal{D}|} \sum_{i=1}^{|\mathcal{D}|} \mathbf{1}_{\{y_i^* = \text{argmax}_{k \in [K]} (\hat{y}_i)_k\}} .$$

207 In the following, we write AccTrain for $\text{AccTrain}(\mathcal{D}_{\text{train}})$ as we only consider the full training set
 208 so there is no ambiguity. The AccTrain computes the number of correctly predicted labels by the
 209 aggregation strategy knowing a ground truth. While this metric is useful, in practice there are a few
 210 arguable issues:

- 211 • the AccTrain metric does not consider the ambiguity of the soft label, only the most probable
 212 class, whereas in some contexts ambiguity can be informative,
- 213 • in supervised learning one objective is to identify difficult or mislabeled tasks (Pleiss et al.
 214 2020; Lefort et al. 2022), pruning those tasks can easily artificially improve the AccTrain, but
 215 there is no guarantee over the predictive performance of a model based on the newly pruned
 216 dataset,
- 217 • in practice, true labels are unknown, thus this metric would not be computable.

218 We first consider classical simulation settings in the literature that can easily be created and repro-
 219 duced using peerannot. For each dataset, we present the distribution of the number of workers per
 220 task $(|\mathcal{A}(x_i)|)_{i=1,\dots,n_{\text{task}}}$ Equation 1 on the right and the distribution of the number of tasks per worker
 221 $(|\mathcal{T}(w_j)|)_{j=1,\dots,n_{\text{worker}}}$ Equation 2 on the left.

222 **3.2.1 Simulated independent mistakes**

223 The independent mistakes setting considers that each worker w_j answers follows a multinomial
 224 distribution with weights given at the row y_i^* of their confusion matrix $\pi^{(j)} \in \mathbb{R}^{K \times K}$. Each confusion
 225 row in the confusion matrix is generated uniformly in the simplex. Then, we make the matrix
 226 diagonally dominant (to represent non-adversarial workers) by switching the diagonal term with
 227 the maximum value by row. Answers are independent of one another as each matrix is generated
 228 independently and each worker answers independently of other workers. In this setting, the DS
 229 model is expected to perform better with enough data as we are simulating data from its assumed
 230 noise model.

231 We simulate $n_{\text{task}} = 200$ tasks and $n_{\text{worker}} = 30$ workers with $K = 5$ possible classes. Each task x_i
 232 receives $|\mathcal{A}(x_i)| = 10$ labels. With 200 tasks and 30 workers, asking for 10 leads to around $\frac{200 \times 10}{30} \approx 67$
 233 tasks per worker (with variations due to randomness in the affectations).

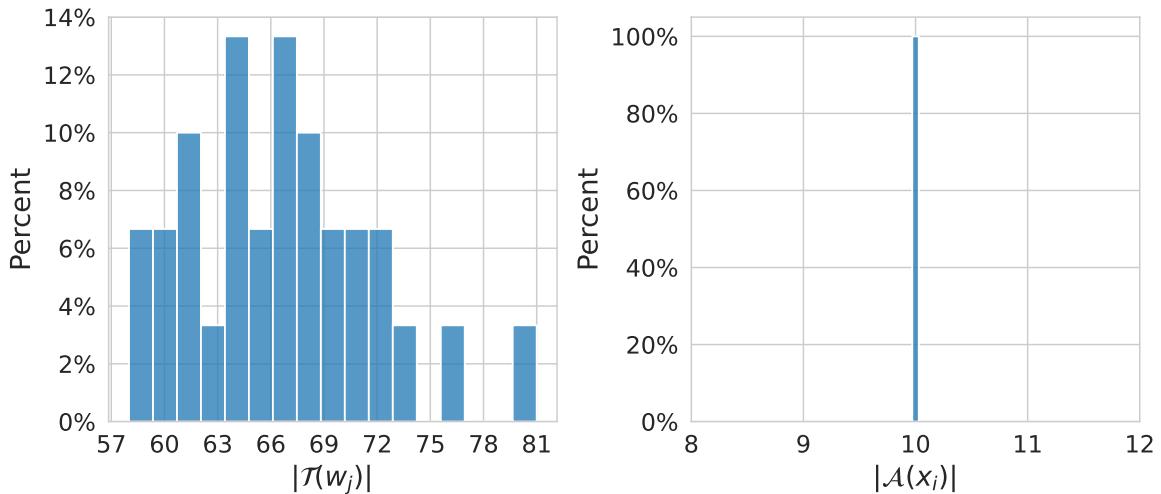


Figure 7: Distribution of number of tasks given per worker (left) and number of labels per task (right) in the independent mistakes setting.

234 With the obtained answers, we can look at the aforementioned aggregation strategies performance.
 235 The `peerannot aggregate` command takes as input the path to the data folder and the aggregation
 236 `--strategy/-s` to perform. Other arguments are available and described in the `--help` description.

Table 1: AccTrain metric on simulated independent mistakes considering classical feature-blind label aggregation strategies

Table 1

	MV	GLAD	DS	DSWC[L=5]	DSWC[L=10]	NS
AccTrain	0.770	0.775	0.890	0.775	0.770	0.760

237 As expected by the simulation framework, Table 1 fits the DS model, thus leading to better accuracy
 238 in retrieving the simulated labels for the DS strategy. The MV and NS aggregations do not consider
 239 any worker-ability scoring or the task's difficulty and perform the worst.

240 **Remark:** `peerannot` can also simulate datasets with an imbalanced number of votes chosen uniformly
 241 at random between 1 and the number of workers available. For example:

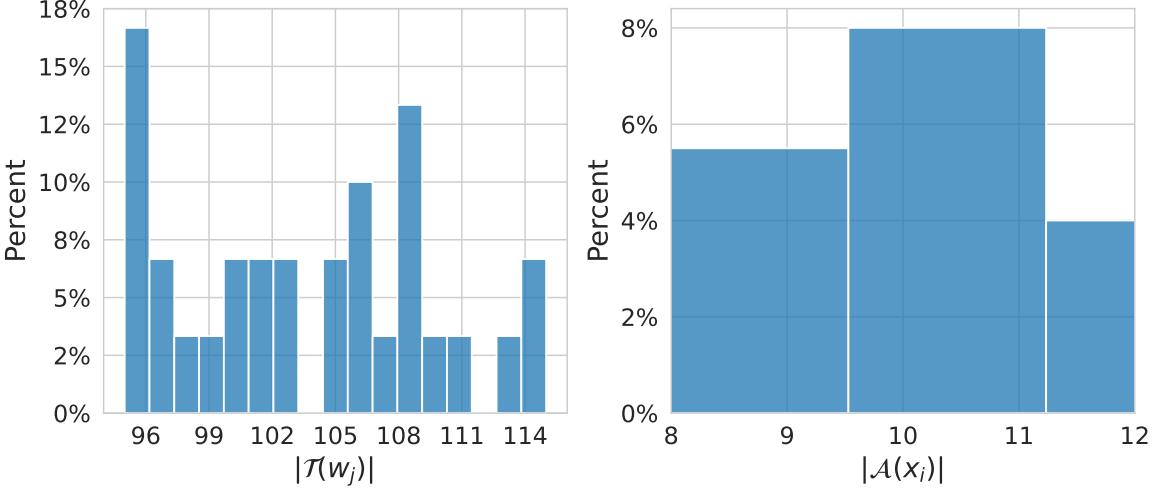


Figure 8: Distribution of the number of tasks given per worker (left) and of the number of labels per task (right) in the independent mistakes setting with voting imbalance enabled.

242 With the obtained answers, we can look at the aforementioned aggregation strategies performance:

Table 2: AccTrain metric on simulated independent mistakes with an imbalanced number of votes per task considering classical feature-blind label aggregation strategies

Table 2

	MV	GLAD	DS	DSWC[L=5]	DSWC[L=10]	NS
AccTrain	0.825	0.810	0.895	0.845	0.840	0.830

243 While more realistic, working with an imbalanced number of votes per task can lead to disrupting
 244 orders of performance for some strategies (here GLAD is outperformed by other strategies).

245 3.2.2 Simulated correlated mistakes

246 The correlated mistakes are also known as the student-teacher or junior-expert setting (Cao et al.
 247 (2019)). Consider that the crowd of workers is divided into two categories: teachers and students
 248 (with $n_{\text{teacher}} + n_{\text{student}} = n_{\text{worker}}$). Each student is randomly assigned to one teacher at the beginning
 249 of the experiment. We generate the (diagonally dominant as in Section 3.2.1) confusion matrices of
 250 each teacher and the students share the same confusion matrix as their associated teacher. Hence,
 251 clustering strategies are expected to perform best in this context. Then, they all answer independently,
 252 following a multinomial distribution with weights given at the row y_i^* of their confusion matrix
 253 $\pi^{(j)} \in \mathbb{R}^{K \times K}$.

254 We simulate $n_{\text{task}} = 200$ tasks and $n_{\text{worker}} = 30$ with 80% of students in the crowd. There are $K = 5$
 255 possible classes. Each task receives $|A(x_i)| = 10$ labels.

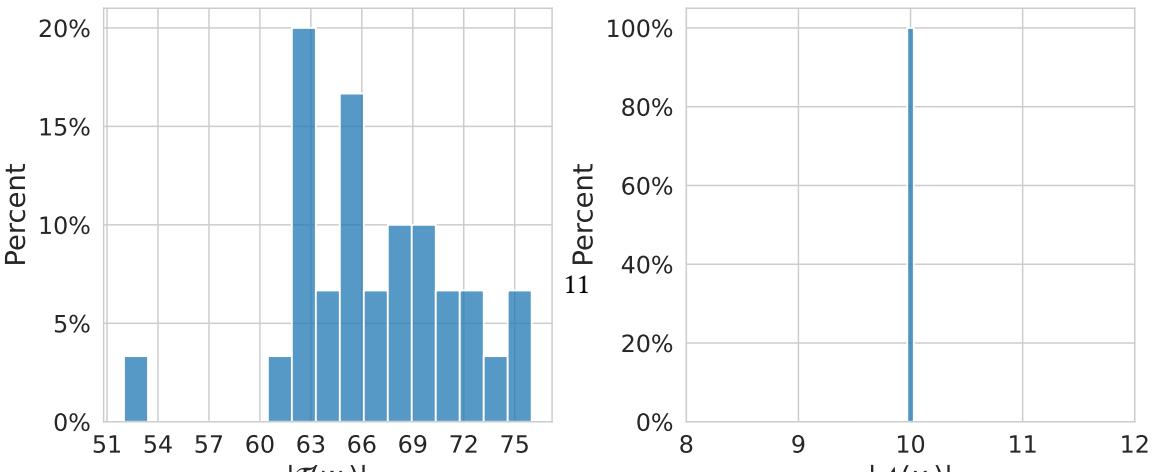


Table 3: AccTrain metric on simulated correlated mistakes considering classical feature-blind label aggregation strategies

Table 3

	MV	GLAD	DS	DSWC[L=5]	DSWC[L=6]	DSWC[L=10]	NS
AccTrain	0.735	0.645	0.755	0.795	0.780	0.815	0.690

With Table 3, we see that with correlated data (24 students and 6 teachers), using 5 confusion matrices with DSWC[L=5] outperforms the vanilla DS strategy that does not consider the correlations. The best-performing method here estimates only 10 confusion matrices (instead of 30 for the vanilla DS model).

To summarize our simulations, we see that depending on workers answering strategies, different latent variable models perform best. However, these are unknown outside of a simulation framework, thus if we want to obtain labels from multiple responses, we need to investigate multiple models. This can be done easily with `peerannot` as we demonstrated using the `aggregate` module. However, one might not want to generate a label, simply learn a classifier to predict labels on unseen data. This leads us to another module part of `peerannot`.

3.3 More on confusion matrices in simulation settings

Moreover, the concept of confusion matrices has been commonly used to represent worker abilities. Let us remind that a confusion matrix $\pi^{(j)} \in \mathbb{R}^{K \times K}$ of a worker w_j is defined such that $\pi_{k,\ell}^{(j)} = \mathbb{P}(y_i^{(j)} = \ell | y_i^* = k)$. These quantities need to be estimated since no true label is available in a crowd-sourced scenario. In practice, the confusion matrices of each worker is estimated via an aggregation strategy like Dawid and Skene's (Dawid and Skene 1979) presented in Section 3.1.

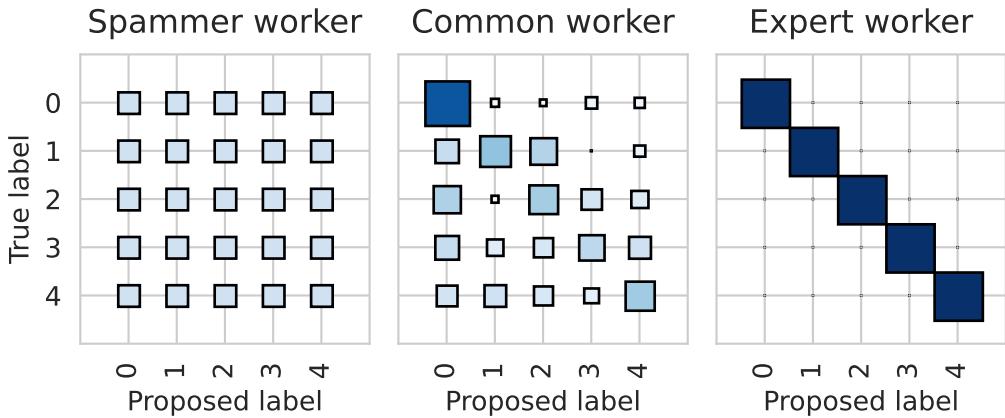


Figure 10: Three types of profiles of worker confusion matrices simulated with `peerannot`. The spammer answers independently of the true label. Expert workers identify classes without mistakes. In practice common workers are good for some classes but might confuse two (or more) labels. All workers are simulated using the `peerannot simulate` command presented in Section 3.2.

In Figure 10, we illustrate multiple workers' profile (as reflected by their confusion matrix) on a simulate scenario where the ground truth is available. For that we generate toy datasets with the `simulate` command from `peerannot`. In particular, we display a type of worker that can hurt

276 data quality: the spammer. Raykar and Yu (2011) defined a spammer as a worker that answers
 277 independently of the true label:

$$\forall k \in [K], \mathbb{P}(y_i^{(j)} = k | y_i^*) = \mathbb{P}(y_i^{(j)} = k) . \quad (3)$$

278 Each row of the confusion matrix represents the label's probability distribution given a true label.
 279 Hence, the spammer has a confusion matrix with near-identical rows. Apart from the spammer,
 280 common mistakes often involve workers mixing up one or several classes. Expert workers have a
 281 confusion matrix close to the identity matrix.

282 4 Learning from crowdsourced tasks

283 Commonly, tasks are crowdsourced to create a large annotated training set as modern machine
 284 learning models require more and more data. The aggregation step then simply becomes the first step
 285 in the complete learning pipeline. However, instead of aggregating labels, modern neural networks
 286 are directly trained end-to-end from multiple noisy labels.

287 4.1 Popular models

288 In recent years, directly learning a classifier from noisy labels was introduced. Two of the most
 289 used models: CrowdLayer (Rodrigues and Pereira 2018) and CoNAL (Chu, Ma, and Wang 2021), are
 290 directly available in peerannot. These two learning strategies directly incorporate a DS-inspired
 291 noise model in the neural network's architecture.

292 4.1.1 CrowdLayer

293 CrowdLayer trains a classifier with noisy labels as follows. Let the scores (logits) output by a given
 294 classifier neural network \mathcal{C} be $z_i = \mathcal{C}(x_i)$. Then CrowdLayer adds as a last layer $\pi \in \mathbb{R}^{n_{\text{worker}} \times K \times K}$, the
 295 tensor of all $\pi^{(j)}$'s such that the crossentropy loss (CE) is adapted to the crowdsourcing setting into
 296 $\mathcal{L}_{CE}^{\text{CrowdLayer}}$ and computed as:

$$\mathcal{L}_{CE}^{\text{CrowdLayer}}(x_i) = \sum_{j \in \mathcal{A}(x_i)} \text{CE}\left(\sigma\left(\pi^{(j)}\sigma(z_i)\right), y_i^{(j)}\right) ,$$

297 where the crossentropy loss for two distribution $u, v \in \Delta_K$ is defined as $\text{CE}(u, v) = \sum_{k \in [K]} v_k \log(u_k)$.

298 Where DS modeled workers as confusion matrices, CrowdLayer adds a layer of $\pi^{(j)}$'s into the backbone
 299 architecture as a new tensor layer to transform the output probabilities. The backbone classifier
 300 predicts a distribution that is then corrupted through the added layer to learn the worker-specific
 301 confusion. The weights in the tensor layer of $\pi^{(j)}$'s are learned during the optimization procedure.

302 4.1.2 CoNAL

303 For some datasets, it was noticed that global confusion occurs between the proposed classes. It is the
 304 case for example in the LabelMe dataset (Rodrigues et al. 2017) where classes overlap. In this case,
 305 Chu, Ma, and Wang (2021) proposed to extend the CrowdLayer model by adding global confusion
 306 matrix $\pi^g \in \mathbb{R}^{K \times K}$ to the model on top of each worker's confusion.

307 Given the output $z_i = \mathcal{C}(x_i) \in \mathbb{R}^K$ of a given classifier and task, CoNAL interpolates between the
 308 prediction corrected by local confusions $\pi^{(j)} z_i$ and the prediction corrected by a global confusion

309 $\pi^g z_i$. The loss function is computed as follows:

$$\mathcal{L}_{CE}^{\text{CoNAL}}(x_i) = \sum_{j \in \mathcal{A}(x_i)} \text{CE}(h_i^{(j)}, y_i^{(j)}) ,$$

with $h_i^{(j)} = \sigma((\omega_i^{(j)} \pi^g + (1 - \omega_i^{(j)}) \pi^{(j)}) z_i)$.

310 The interpolation weight $\omega_i^{(j)}$ is unobservable in practice. So, to compute $h_i^{(j)}$, the weight is obtained
 311 through an auxiliary network. This network takes as input the image and worker information
 312 and outputs a task-related vector v_i and a worker-related vector u_j of the same dimension. Finally,
 313 $w_i^{(j)} = (1 + \exp(-u_j^\top v_i))^{-1}$.

314 Both CrowdLayer and CoNAL model worker confusions directly in the classifier's weights to learn
 315 from the noisy collected labels and are available in peerannot as we will see in the following.

316 4.2 Prediction error when learning from crowdsourced tasks

317 The AccTrain metric presented in Section 3.2 might no longer be of interest when training a classifier.
 318 Classical error measurements involve a test dataset to estimate the generalization error. To do so, we
 319 present hereafter two error metrics. Assuming we trained our classifier \mathcal{C} on a training set and that
 320 there is a test set available with known true labels:

- 321 • the test accuracy is computed as $\frac{1}{n_{\text{test}}} \sum_{i=1}^{n_{\text{test}}} \mathbf{1}_{\{y_i^* = \hat{y}_i\}}$.
- 322 • the expected calibration error (Guo et al. 2017) over M equally spaced bins I_1, \dots, I_M partitioning
 323 the interval $[0, 1]$, is computed as:

$$\text{ECE} = \sum_{m=1}^M \frac{|B_m|}{n_{\text{task}}} |\text{acc}(B_m) - \text{conf}(B_m)| ,$$

324 with $B_m = \{x_i | \mathcal{C}(x_i)[1] \in I_m\}$ the tasks with predicted probability in the m -th bin, $\text{acc}(B_m)$
 325 the accuracy of the network for the samples in B_m and $\text{conf}(B_m)$ the associated empirical
 326 confidence. More precisely:

$$\text{acc}(B_m) = \frac{1}{|B_m|} \sum_{i \in B_m} \mathbf{1}(\hat{y}_i = y_i^*) \quad \text{and} \quad \text{conf}(B_m) = \frac{1}{|B_m|} \sum_{i \in B_m} \sigma(\mathcal{C}(x_i))[1] .$$

327 The accuracy represents how well the classifier generalizes, and the expected calibration error (ECE)
 328 quantifies the deviation between the accuracy and the confidence of the classifier. Modern neural
 329 networks are known to often be overconfident in their predictions (Guo et al. 2017). However, it has
 330 also been remarked that training on crowdsourced data, depending on the strategy, mitigates this
 331 confidence issue. That is why we propose to compare them both in our coming experiments. Note
 332 that the ECE error estimator is known to be biased (Gruber and Buettner 2022). Smaller training
 333 sets are known to have a higher ECE estimation error. And in the crowdsourcing setting, openly
 334 available datasets are often quite small.

335 4.3 Use case with peerannot on real datasets

336 Few real crowdsourcing experiments have been released publicly. Among the available ones,
 337 CIFAR-10H (Peterson et al. 2019) is one of the largest with 10000 tasks labeled by workers (the
 338 testing set of CIFAR-10). The main limitation of CIFAR-10H is that there are few disagreements
 339 between workers and a simple majority voting already leads to a near-perfect AccTrain error. Hence,
 340 comparing the impact of aggregation and end-to-end strategies might not be relevant (Peterson et al.

341 2019; Aitchison 2021), it is however a good benchmark for task difficulty identification and worker
342 evaluation scoring. Each of these dataset contains a test set, with known ground truth. Thus, we can
343 train a classifier from the crowdsourced data, and compare predictive performance on the test set.

344 The LabelMe dataset was extracted from crowdsourcing segmentation experiments and a subset of
345 $K = 8$ classes was released in Rodrigues et al. (2017).

346 Let us use `peerannot` to train a VGG-16 with two dense layers on the LabelMe dataset. Note that
347 this modification was introduced to reach state-of-the-art performance in (Chu, Ma, and Wang
348 2021). Other models from the `torchvision` library can be used, such as Resnets, Alexnet etc. The
349 `aggregate-deep` command takes as input the path to the data folder, `--output-name/-o` is the name
350 for the output file, `--n-classes/-K` the number of classes, `--strategy/-s` the learning strategy to
351 perform (e.g., CrowdLayer or CoNAL), the backbone classifier in `--model` and then optimization
352 hyperparameters for pytorch described with more details using the `peerannot aggregate-deep`
353 `--help` command.

Table 4: Generalization performance on LabelMe dataset depending on the learning strategy from
the crowdsourced labels. The network used is a VGG-16 with two dense layers for all methods.

Table 4

	method	AccTest	ECE
0	GLAD	81.061	0.189
1	CrowdLayer	85.606	0.143
2	CoNAL[scale=1e-4]	86.448	0.136
3	MV	87.205	0.117
4	NS	87.542	0.124
5	DS	88.468	0.115
6	CoNAL[scale=0]	88.889	0.112

354 As we can see, CoNAL strategy performs best. In this case, it is expected behavior as CoNAL
355 was created for the LabelMe dataset. However, using `peerannot` we can look into **why modeling**
356 **common confusion returns better results with this dataset**. To do so, we can explore the
357 datasets from two points of view: worker-wise or task-wise in Section 5.

358 5 Identifying tasks difficulty and worker abilities

359 If a dataset requires crowdsourcing to be labeled, it is because expert knowledge is long and costly to
360 obtain. In the era of big data, where datasets are built using web scraping (or using a platform like
361 [Amazon Mechanical Turk](#)), citizen science is popular as it is an easy way to produce many labels.

362 However, mistakes and confusions happen during these experiments. Sometimes involuntarily
363 (e.g., because the task is too hard or the worker is unable to differentiate between two classes) and
364 sometimes voluntarily (e.g., the worker is a spammer).

365 Underlying all the learning models and aggregation strategies, the cornerstone of crowdsourcing
366 is evaluating the trust we put in each worker depending on the presented task. And with the
367 gamification of crowdsourcing (Servajean et al. 2016; Tinati et al. 2017), it has become essential to
368 find scoring metrics both for workers and tasks to keep citizens in the loop so to speak. This is the
369 purpose of the identification module in `peerannot`.

370 Our test cases are both the CIFAR-10H dataset and the LabelMe dataset to compare the worker and
 371 task evaluation depending on the number of votes collected. Indeed, the LabelMe dataset has only
 372 up to three votes per task whereas CIFAR-10H accounts for nearly fifty votes per task.

373 5.1 Exploring tasks' difficulty

374 To explore the tasks' intrinsic difficulty, we propose to compare three scoring metrics:

- 375 • the entropy of the NS distribution: the entropy measures the inherent uncertainty of the
 376 distribution to the possible outcomes. It is reliable with a big enough and not adversarial crowd.
 377 More formally:

$$\forall i \in [n_{\text{task}}], \text{Entropy}(\hat{y}_i^{NS}) = - \sum_{k \in [K]} (\hat{y}_i^{NS})_k \log((\hat{y}_i^{NS})_k) .$$

- 378 • GLAD's scoring: by construction, Whitehill et al. (2009) introduced a scalar coefficient to score
 379 the difficulty of a task.
 380 • the Weighted Area Under the Margins (WAUM): introduced by Lefort et al. (2022), this weighted
 381 area under the margins indicates how difficult it is for a classifier \mathcal{C} to learn a task's label. This
 382 procedure is done with a budget of $T > 0$ epochs. Given the crowdsourced labels and the trust
 383 we have in each worker denoted $s^{(j)}(x_i) > 0$, the WAUM of a given task $x_i \in \mathcal{X}$ and a set of
 384 crowdsourced labels $\{y_i^{(j)}\}_j \in [K]^{|\mathcal{A}(x_i)|}$ is defined as:

$$\text{WAUM}(x_i) := \frac{1}{|\mathcal{A}(x_i)|} \sum_{j \in \mathcal{A}(x_i)} s^{(j)}(x_i) \left\{ \frac{1}{T} \sum_{t=1}^T \sigma(\mathcal{C}(x_i))_{y_i^{(j)}} - \sigma(\mathcal{C}(x_i))_{[2]} \right\} ,$$

385 where we remind that $\mathcal{C}(x_i)_{[2]}$ is the second largest probability output by the classifier \mathcal{C} for
 386 the task x_i .

387 The weights $s^{(j)}(x_i)$ are computed à la Servajean et al. (2017):

$$\forall j \in [n_{\text{worker}}], \forall i \in [n_{\text{task}}], s^{(j)}(x_i) = \langle \sigma(\mathcal{C}(x_i)), \text{diag}(\hat{\pi}^{(j)}) \rangle ,$$

388 where $\hat{\pi}^{(j)}$ is the estimated confusion matrix of worker w_j (by default, the estimation provided by
 389 DS).

390 The WAUM is a generalization of the AUM by Pleiss et al. (2020) to the crowdsourcing setting. A
 391 high WAUM indicates a high trust in the task classification by the network given the crowd labels. A
 392 low WAUM indicates difficulty for the network to classify the task into the given classes (taking into
 393 consideration the trust we have in each worker for the task considered). Where other methods only
 394 consider the labels and not directly the tasks, the WAUM directly considers the learning trajectories
 395 to identify ambiguous tasks. One pitfall of the WAUM is that it is dependent on the architecture used.

396 Note that each of these statistics could prove useful in different contexts. The entropy is irrelevant in
 397 settings with few labels per task (small $|\mathcal{A}(x_i)|$). For instance, it is uninformative for LabelMe dataset.
 398 The WAUM can handle any number of labels, but the larger the better. However, as it uses a deep
 399 learning classifier, the WAUM needs the tasks $(x_i)_i$ in addition to the proposed labels while the other
 400 strategies are feature-blind.

401 5.1.1 CIFAR-10H dataset

402 First, let us consider a dataset with a large number of tasks, annotations and workers: the CIFAR-10H
 403 dataset by Peterson et al. (2019).

404 Unable to display output for mime type(s): text/html

405 Most difficult tasks sorted by class from MV aggregation identified depending on the strategy used
406 (entropy, GLAD or WAUM) using a Resnet34.

407 Unable to display output for mime type(s): text/html

408

409 The entropy, GLAD's difficulty, and WAUM's difficulty each show different images as exhibited in
410 the interactive Figure. While the entropy and GLAD output similar tasks, in this case, the WAUM
411 often differs. We can also observe an ambiguity induced by the labels in the truck category, with the
412 presence of a trailer that is technically a mixup between a car and a truck.

413 5.1.2 LabelMe dataset

414 As for the LabelMe dataset, one difficulty in evaluating tasks' intrinsic difficulty is that there is a
415 limited amount of votes available per task. Hence, the entropy in the distribution of the votes is no
416 longer a reliable metric, and we need to rely on other models.

417 Now, let us compare the tasks' difficulty distribution depending on the strategy considered using
418 peerannot.

419 Unable to display output for mime type(s): text/html

420 Most difficult tasks sorted by class from MV aggregation identified depending on the strategy used
421 (entropy, GLAD or WAUM) using a VGG-16 with two dense layers.

422

423 Note that in this experiment, because the number of labels given per task is in $\{1, 2, 3\}$, the entropy
424 only takes four values. In particular, tasks with only one label all have a null entropy, so not just
425 consensual tasks. The MV is also not suited in this case because of the low number of votes per task.

426 The underlying difficulty of these tasks mainly comes from the overlap in possible labels. For example,
427 tallbuildings are most often found insidecities, and so are streets. In the opencountry we
428 find forests, river-coasts and mountains.

429 5.2 Identification of worker reliability and task difficulty

430 From the labels, we can explore different worker evaluation scores. GLAD's strategy estimates a
431 reliability scalar coefficient α_j per worker. With strategies looking to estimate confusion matrices,
432 we investigate two scoring rules for workers:

- 433 • The trace of the confusion matrix: the closer to K the better the worker.
- 434 • The closeness to spammer metric (Raykar and Yu 2011) (also called spammer score) that is the
435 Frobenius norm between the estimated confusion matrix $\hat{\pi}^{(j)}$ and the closest rank-1 matrix.
436 The further to zero the better the worker. On the contrary, the closer to zero, the more likely it
437 is for the worker to be a spammer. This score separates spammers from common workers and
438 experts (with profiles as in Figure 10).

439 When the tasks are available, confusion-matrix-based deep learning models can also be used. We
440 thus add to the comparison the trace of the confusion matrices with CrowdLayer and CoNAL on
441 the LabelMe datasets. For CoNAL, we only consider the trace of the confusion matrix $\pi^{(j)}$ in the
442 pairwise comparison. Moreover, for CrowdLayer and CoNAL we show in Figure 12 the weights
443 learned without the softmax operation by row to keep the comparison as simple as possible with the
444 actual outputs of the model.

445 Comparisons in Figure 11 and Figure 12 are plotted pairwise between the evaluated metrics. Each
 446 point represents a worker. Each off-diagonal plot shows the joint distribution between the scores of
 447 the y-axis row and the x-axis column. They allow us to visualize the relationship between these two
 448 variables. The main diagonal represents the (smoothed) marginal distribution of the score of the
 449 considered column.

450 **5.2.1 CIFAR-10H**

451 The CIFAR-10H dataset has few disagreements among workers. However, these strategies disagree
 452 on the ranking of good against best workers as they do not measure the same properties.

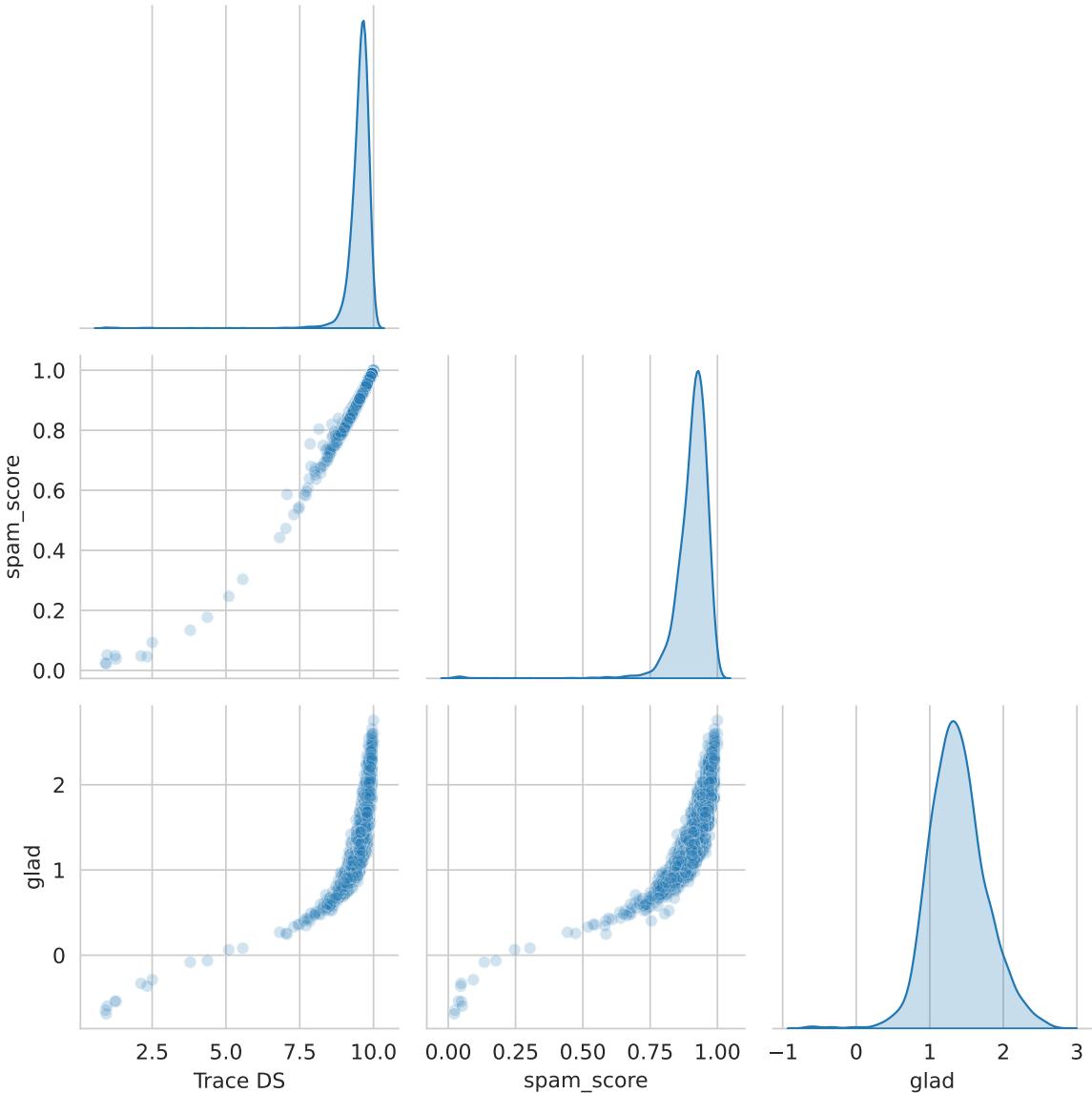


Figure 11: Comparison of ability scores by workers for the CIFAR-10H dataset. All metrics computed identify the same poorly performing workers. A mass of good and expert workers can be seen as the dataset presents few disagreements, thus few data to discriminate expert workers from the otherss.

453 From Figure 11, we can see that in this dataset, different methods easily separate the worst workers
 454 from the rest of the crowd (workers in the left tail of the distribution).

455 **5.2.2 LabelMe**

456 Finally, let us evaluate workers for the LabelMe dataset. Because of the lack of data (up to 3 labels
 457 per task), ranking workers is more difficult than in the CIFAR-10H dataset.

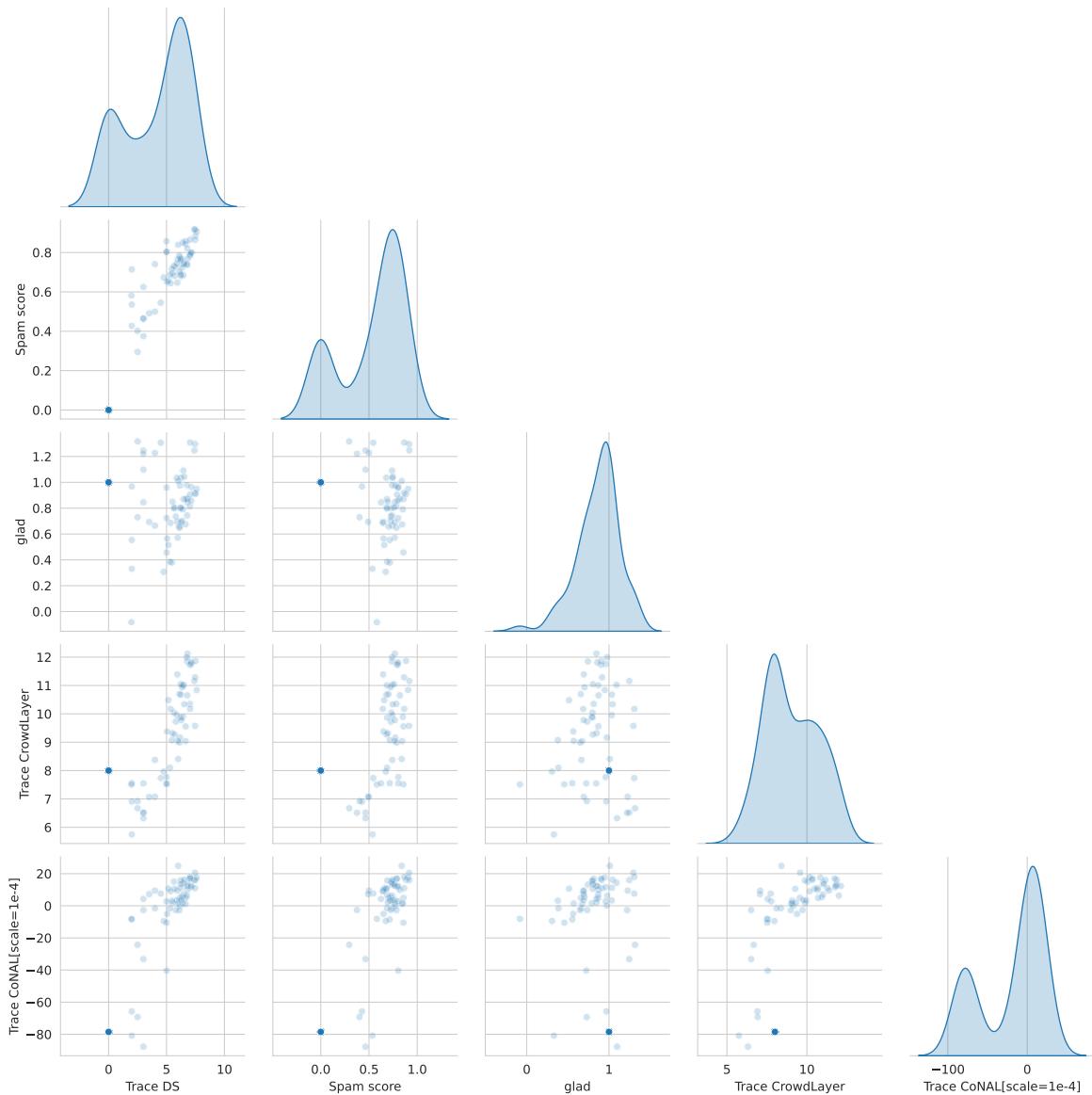


Figure 12: Comparison of ability scores by workers for the LabelMe dataset. With few labels per task, workers are more difficult to rank. It is more difficult to separate workers with their abilities in this crowd. Hence the importance of investigating the generalization performance of the methods presented in the previous section.

458 We can see in Figure 12 that the number of labels available by task highly impacts the worker
 459 evaluation scores. The spam score, DS model and CoNAL all show similar results in the distribution
 460 shape (bimodal distribution) whereas GLAD and CrowdLayer are more concentrated. However, this
 461 does not account for the ranking of a given worker by the methods considered. The exploration of
 462 the dataset lets us look at different scores, but generalization performance presented in Section 4.3
 463 should also be considered in crowdsourcing. This difference in worker evaluation scores indeed
 464 further highlights the importance of using multiple test metrics to compare the model’s prediction
 465 performance in crowdsourcing. Poorly performing workers could be removed from the dataset with

naive strategies like MV or NS. However, some label aggregation strategies like DS or GLAD can sometimes use adversarial votes as information – for example in binary classification, with a worker answering always the opposite label the confusion matrix retrieves the true label. We have seen that the library `peerannot` allows users to explore the datasets, both in terms of tasks and workers, and easily compare predictive performance in this setting.

In practice, the data exploration step can be used to detect possible ambiguities in the dataset’s tasks, but also remove answers from spammers to improve the data quality as shown in Figure 1. The easy access to the different strategies allows the user to decide if, for their collected dataset, there is a need for more recent deep-learning-based strategies to improve the results. This is the case for the LabelMe dataset. Otherwise, the user can decide that standard aggregation-based crowdsourcing strategies are sufficient and for example, if there are plenty of votes per task like in CIFAR-10H, that the entropy of the vote distribution is a criterion that identified enough ambiguous tasks for their case. As often, not a single strategy works best for all datasets, hence the need to perform easy comparisons with `peerannot`.

6 Conclusion

We introduced `peerannot`, a library to handle crowdsourced datasets. This library enables both easy label aggregation and direct training strategies with classical state-of-the-art classifiers. The identification module of the library allows exploring the collected data from both the tasks and the workers’ point of view for better scorings and data cleaning procedures. Our library also comes with templated datasets to better share crowdsourced datasets. Going beyond templating, it helps the crowdsourcing community to have openly accessible strategies to test, compare and improve to develop common strategies to analyze more and more common crowdsourced datasets.

We hope that this library helps reproducibility in the crowdsourcing community and also standardizes training from crowdsourced datasets. New strategies can easily be incorporated into the open-source code [available on GitHub](#). Finally, as `peerannot` is mostly directed to handle classification datasets, one of our future works would be to consider other `peerannot` modules to handle crowdsourcing for object detection, segmentation and even worker evaluation in other contexts like peer-grading.

7 Appendix

7.1 Supplementary simulation: Simulated mistakes with discrete difficulty levels on tasks

For an additional simulation setting, we consider the so-called discrete difficulty presented in Whitehill et al. (2009). Contrary to other simulations, we here consider that workers belong to two levels of abilities: good or bad, and tasks have two levels of difficulty: easy or hard. The keyword `ratio-diff` indicates the prevalence of each level of difficulty, it is defined as the ratio of easy tasks over hard tasks:

$$\text{ratio-diff} = \frac{P(\text{easy})}{P(\text{hard})} \text{ with } P(\text{easy}) + P(\text{hard}) = 1 .$$

Difficulties are then drawn [at random](#). Tasks that are easy are answered correctly by every worker. Tasks that are hard are answered following the confusion matrix assigned to each worker (as in Section 3.2.1). Each worker then answers independently to the presented tasks.

504 We simulate $n_{\text{task}} = 500$ tasks and $n_{\text{worker}} = 100$ with 35% of good workers in the crowd and 50% of
 505 easy tasks. There are $K = 5$ possible classes. Each task receives $|\mathcal{A}(x_i)| = 10$ labels.

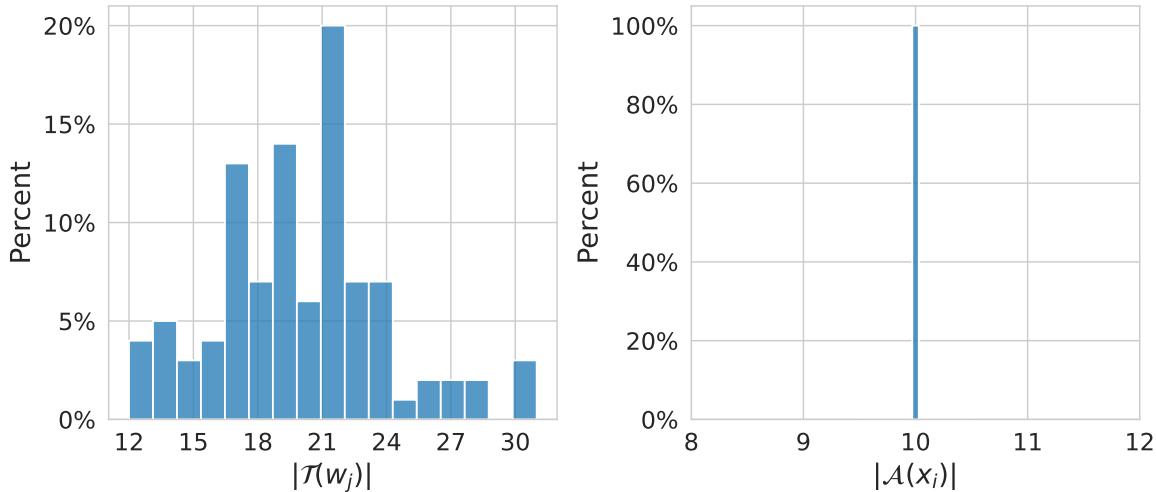


Figure 13: Distribution of the number of tasks given per worker (left) and of the number of labels per task (right) in the setting with simulated discrete difficulty levels.

506 With the obtained answers, we can look at the aforementioned aggregation strategies performance:

Table 5: AccTrain metric on simulated mistakes made when tasks are associated with a difficulty level considering classical feature-blind label aggregation strategies.

Table 5

	MV	GLAD	DS	DSWC[L=2]	DSWC[L=5]	NS
AccTrain	0.800	0.845	0.810	0.600	0.660	0.790

507 Finally, in this setting involving task difficulty coefficients, the only strategy that involves a latent
 508 variable for the task difficulty, knowing GLAD, outperforms the other strategies (see Table 5). Note
 509 that in this case, creating clusters of answers leads to worse decisions than an MV aggregation.

510 7.2 Comparison with other libraries

511 In this section, we provide several comparisons with the Ustalov, Pavlichenko, and Tseitlin (2023)
 512 library.

- 513 • Framework: `peerannot` focuses on image classification problems with categorical answers.
 514 `crowd-kit` also considers textual responses and image segmentation with three aggregation
 515 strategies for each field.
- 516 • Data storage: `peerannot` introduces this `.json` storage that can handle large datasets. `crowd-`
 517 `kit` stores the collected data in a `.csv` file with columns `task`, `worker`, `label`.
- 518 • Identification module: one of the major differences between the two libraries resides in
 519 the `identification` module of `peerannot`. This module allows us to explore the dataset
 520 and detect poorly performing workers / difficult tasks easily. `crowd-kit` only allows us to
 521 explore workers with the `accuracy_on_aggregation` metric that computes the accuracy of
 522 a worker given aggregated hard labels. `peerannot`, as demonstrated in Section 5, proposes
 523 several metrics such as the spam score, GLAD's worker ability coefficient and the trace of the

524 confusion matrices. As for the task side, peerannot proposes the different popular metrics
 525 in crowd-kit accompanied with the WAUM (and also the AUMC) metrics from Lefort et al.
 526 (2022) and GLAD's difficulty coefficients.

- 527 • Training: peerannot lets users directly train a neural network architecture from the aggregated
 528 labels. This feature is not proposed by crowd-kit.
- 529 • Simulation: peerannot created a simulate module to check strategies on. This feature is also
 530 not in the crowd-kit library.

531 Finally, to compare different strategies across libraries, we implemented a [crowdsourcing benchmark](#)
 532 in the Benchopt (Moreau et al. (2022)) library. The Benchopt library allows users to easily compare
 533 and reproduce optimization problem benchmarks between multiple frameworks. After running each
 534 strategy, we measure the cumulated time taken to reach the optimum during the optimization steps.
 535 The metric measured on the y-axis is the AccTrain. Each strategy is run 5 times until convergence.
 536 The differences in results across iterations for the MV strategy come from the randomness in the
 537 choice in case of equalities. We provide a clone of the crowdsourcing benchmark and the results are
 538 obtained by running the following command:

```
#| code-fold: false
benchopt run ./benchmark_crowdsourcing
```

539 First, let us see the performances on the [Bluebirds](#) dataset, a small dataset with 39 workers, 108 tasks
 540 and $K = 2$ classes.

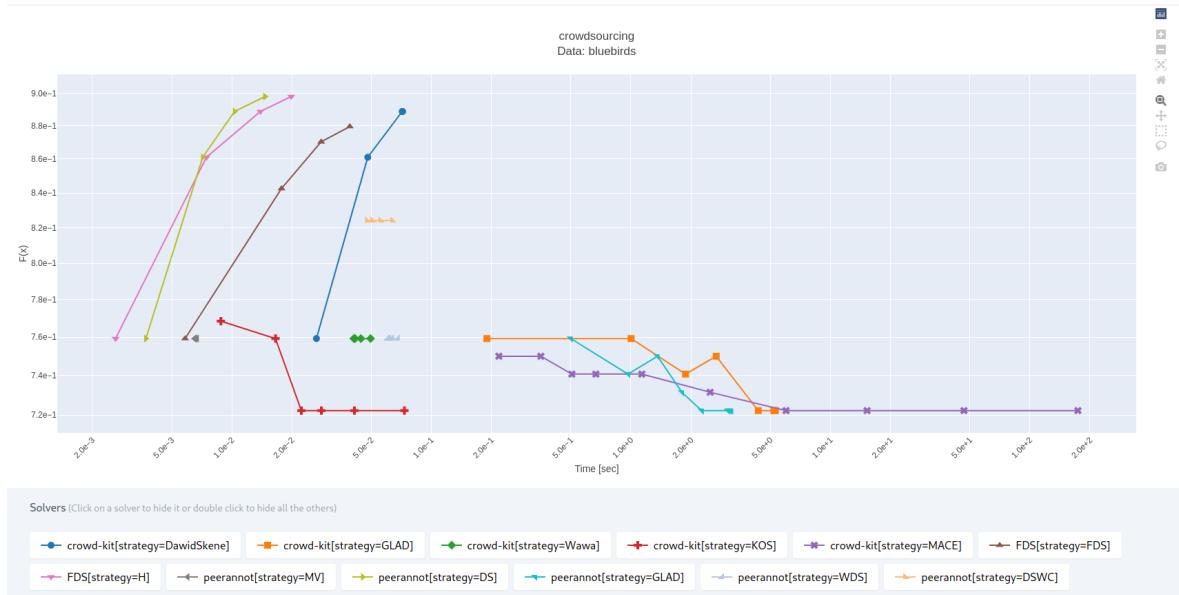


Figure 14: Aggregation strategies computational time during optimization procedure for the BlueBirds dataset with $K=2$.

541 We see in Figure 14 that the DS strategy from peerannot is the first to reach the optimum, followed
 542 by the [Fast-DS strategy](#) and then crowd-kit DS. Other strategies do not lead to better accuracy on
 543 this dataset and DS seems to be the best fitting strategy.

544 For the LabelMe dataset, DS strategy is also the best aggregation strategy, faster for crowd-kit. The
 545 sensitivity of GLAD's method to the priors on α and β parameters can lead to large performance
 546 differences for real datasets as we see in Figure 15. Note that crowd-kit's KOS strategy is not
 547 available for this dataset as it is only made for binary classification datasets.

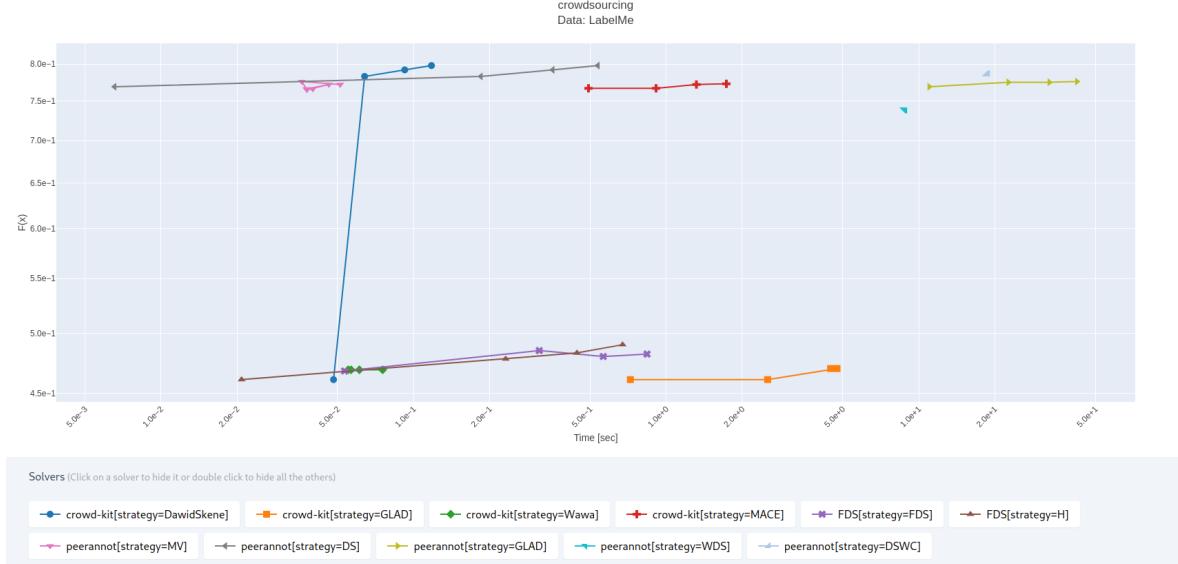


Figure 15: Aggregation strategies computational time during optimization procedure for the LabelMe dataset with K=8

548 7.3 Examples of images in CIFAR-10H and Labelme

549 In this section, we provide examples of images from the CIFAR-10H and LabelMe datasets. Both of
 550 these datasets came with known true labels. For CIFAR-10H, the true labels were from the original
 551 CIFAR-10 dataset. For LabelMe, the true labels were determined by the authors at release.

552 7.4 Case study with bird sound classification

553 We shared our results on the classical CIFAR-10H and LabelMe datasets. More recently, Lehikoinen
 554 et al. (2023) developed a platform for bird sound classification. They released the data for the
 555 following crowdsourcing experiment. Given the sample of the audio of a species (the letter), users
 556 were presented with a new audio sample (the candidate). The question is as follows: *Is the species*
 557 *vocalizing in the candidate the same as the species in the letter?* ‘The answer is a binary yes or no. In
 558 total, $n_{\text{worker}} = 205$ workers labeled $n_{\text{task}} = 79,592$ candidates. Each task received between 1 and 77
 559 annotations. Workers answered between 1 and 30,759 tasks (only one worker achieved that record,
 560 and 23% of the workers answered 100 tasks). There is no test set available as is in the original dataset.
 561 However, to have an idea of the level of performance of the label aggregation strategies, we use the
 562 fact that workers reported their level of expertise between 1 and 4. The latter corresponds to “I am
 563 bird researcher or professional birdwatcher”. This generates a test set of 13,041 tasks where the
 564 expert label is used as the current truth. This test set is only used to compute the AccTrain metric.
 565 Note that we do not perform deep-learning methods as the tasks of comparing the birds from two
 566 audio files and designing specific architectures to match this framework is out of the scope of this
 567 paper.

568 We then can run our aggregation strategies, and from Table 6 we see that strategies reach the same
 569 levels of label recovery, however naive they are. Indeed, most tasks have very few disagreements.

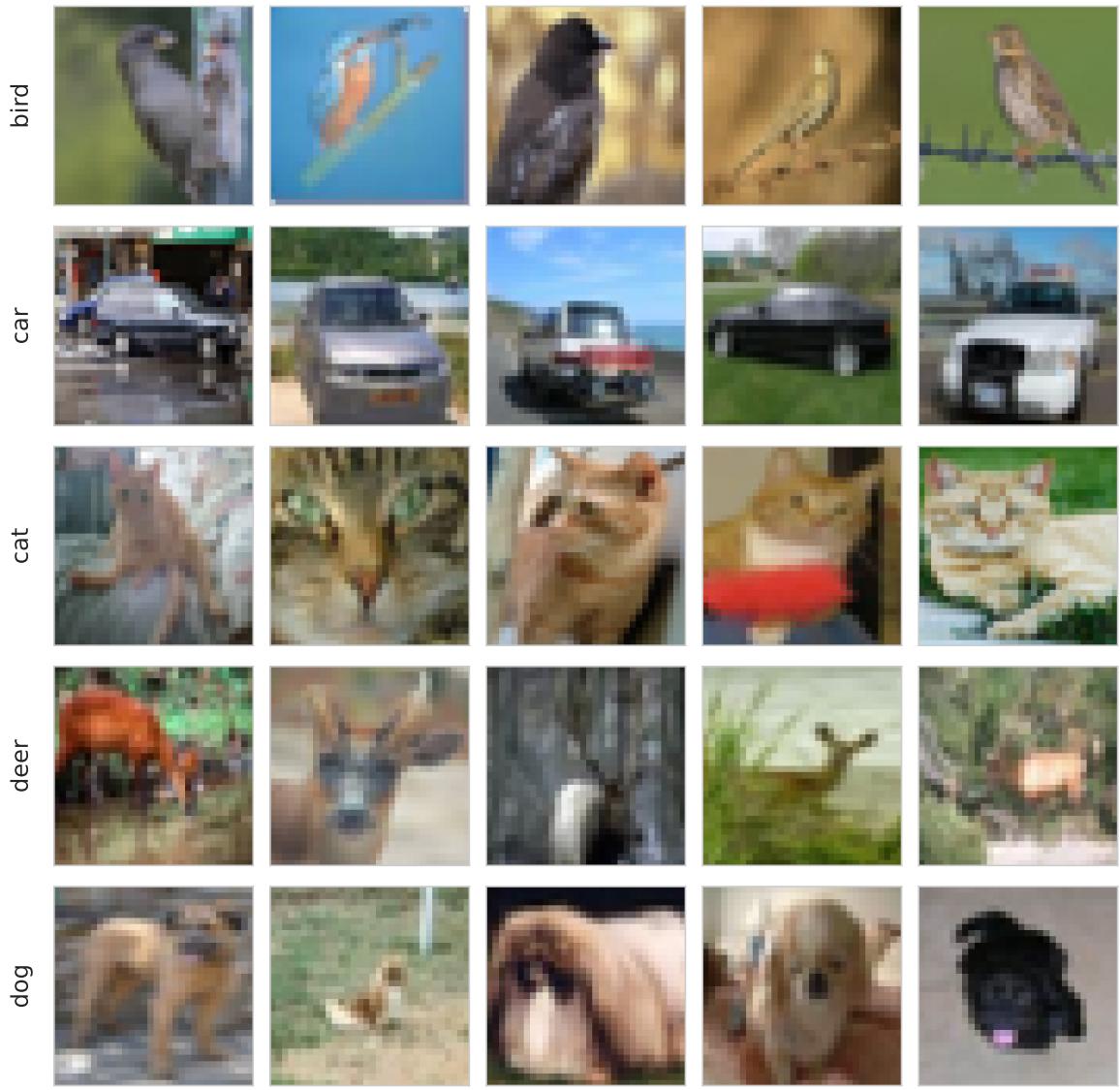


Figure 16: Example of images from CIFAR-10H. We display images row-wise according to the true label given initially in CIFAR-10.

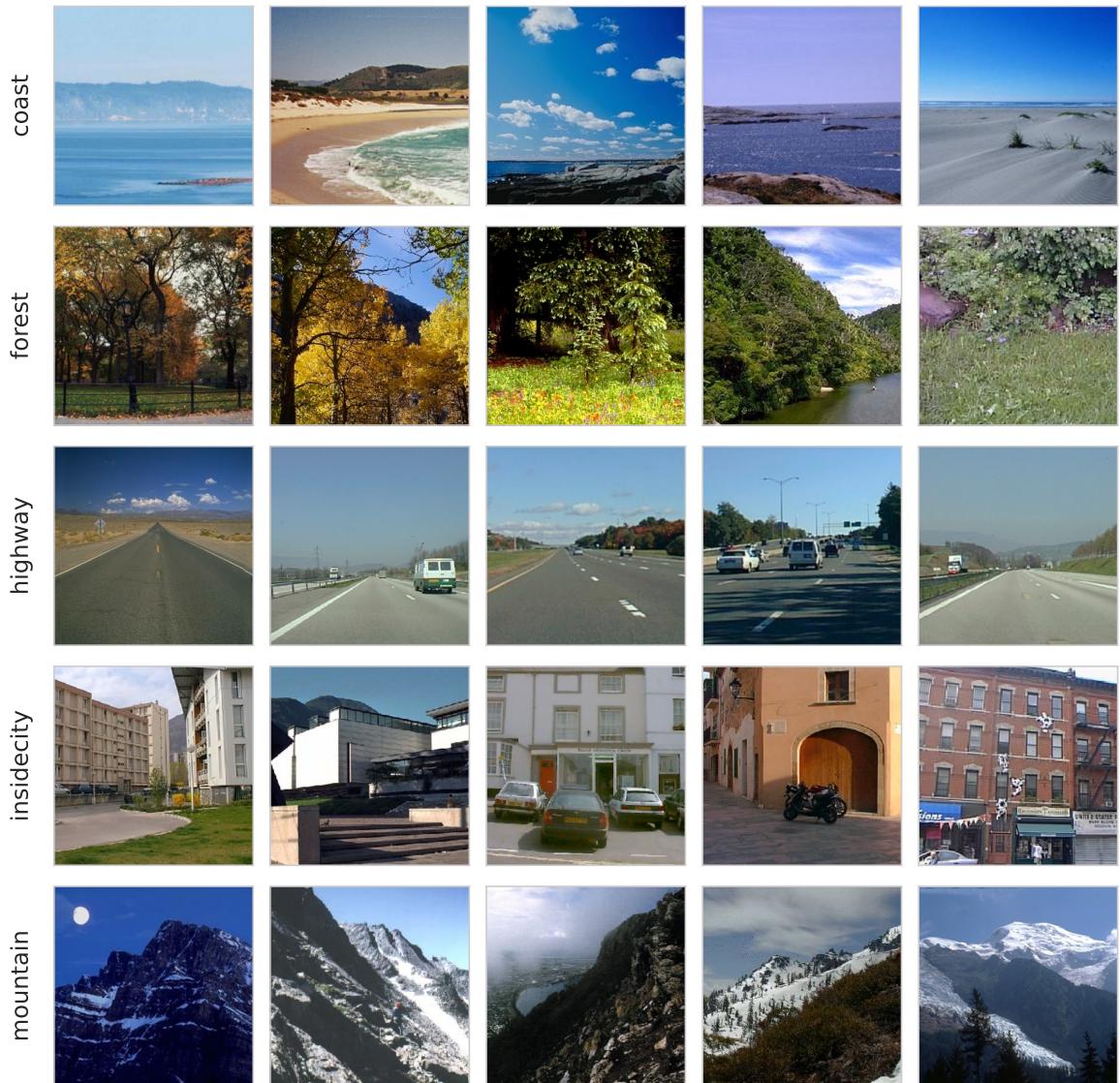


Figure 17: Example of images from LabelMe. We display images row-wise according to the true label given with the crowdsourced data.

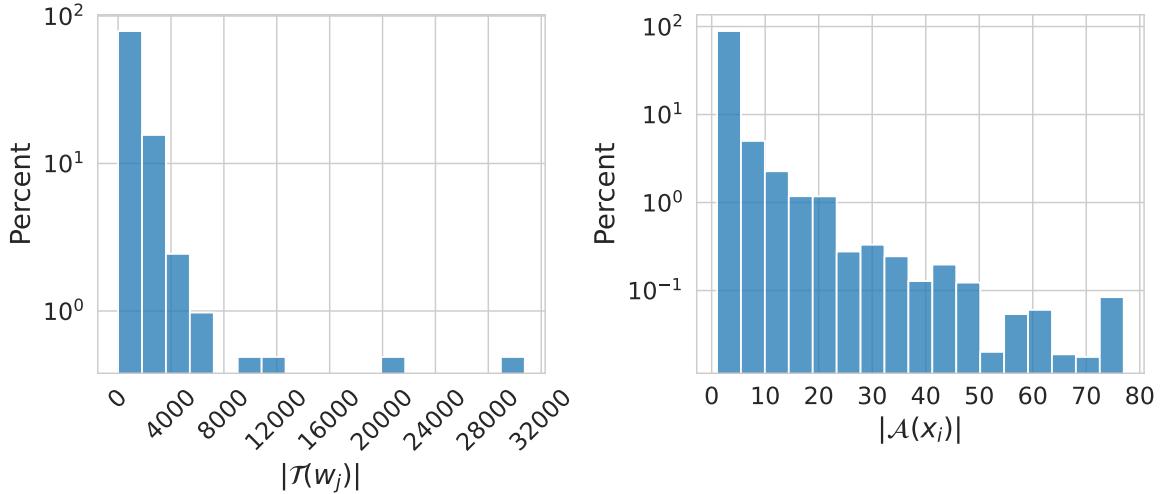


Figure 18: Distribution of the number of tasks given per worker (left) and of the number of labels per task (right) in the Audio Birds letters dataset.

Table 6: AccTrain metric on birds audio dataset considering classical feature-blind label aggregation strategies.

Table 6

	MV	DS	GLAD	NS
AccTrain	0.954	0.946	0.950	0.960

570 We can explore what tasks lead to the most disagreements depending on the entropy criterion or
 571 GLAD's difficulty-estimated latent variable.

572 Using the entropy criterion, the most difficult tasks (highest entropy) and GLAD's difficulty, we
 573 recover the index of the most ambiguous tasks.

574 Highest entropy tasks index: [28272, 25827, 40989, 2771, 55559]

575 Highest GLAD difficulty index: [2347, 435, 8710, 8992, 51700]

576 • Entropy: we obtain the candidate MRG18_20180514_000000_203.mp3 that was to be compared
 577 with the letter HL015_20180515_021439_31.mp3 (one worker agrees and another disagrees):

578 [./datasets/birds_audio/bird_sound_training_data/audio_files/MRG18_20180514_000000_203.mp3](#)

579 [./datasets/birds_audio/bird_sound_training_data/audio_files/HLO15_20180515_021439_31.mp3](#)

580 And the candidate MRG24_20180512_000000_437.mp3 that was to be compared with the letter
 581 HL012_20180511_150153_42.mp3 (one worker agrees and another disagrees):

582 [./datasets/birds_audio/bird_sound_training_data/audio_files/MRG24_20180512_000000_437.mp3](#)

583 [./datasets/birds_audio/bird_sound_training_data/audio_files/HLO12_20180511_150153_42.mp3](#)

584 • GLAD: we obtain the candidate HL004_20180511_034424_15.mp3 that was to be compared
 585 with the letter MRG11_20180519_000000_506.mp3 (53 votes, 29 agreeing and 24 disagreeing):

586 [./datasets/birds_audio/bird_sound_training_data/audio_files/HLO04_20180511_034424_15.mp3](#)

587 [./datasets/birds_audio/bird_sound_training_data/audio_files/MRG11_20180519_000000_506.mp3](#)
 588 And the candidate [MRG27_20180512_000000_597.mp3](#) that was to be compared with the letter
 589 [HL001_20180601_080126_30.mp3](#) (43 votes, 23 agreeing and 20 disagreeing):
 590 [./datasets/birds_audio/bird_sound_training_data/audio_files/MRG27_20180512_000000_597.mp3](#)
 591 [./datasets/birds_audio/bird_sound_training_data/audio_files/HL001_20180601_080126_30.mp3](#)
 592 In this dataset, a single task with two different votes has the highest entropy. GLAD's coefficient lets
 593 us explore tasks with multiple votes where workers were split.
 594 We can also explore the dataset from a worker's point of view and visualize workers' performance
 595 and how many are identified as poorly performing. This gives us an idea of the level of noise in the
 596 answers.

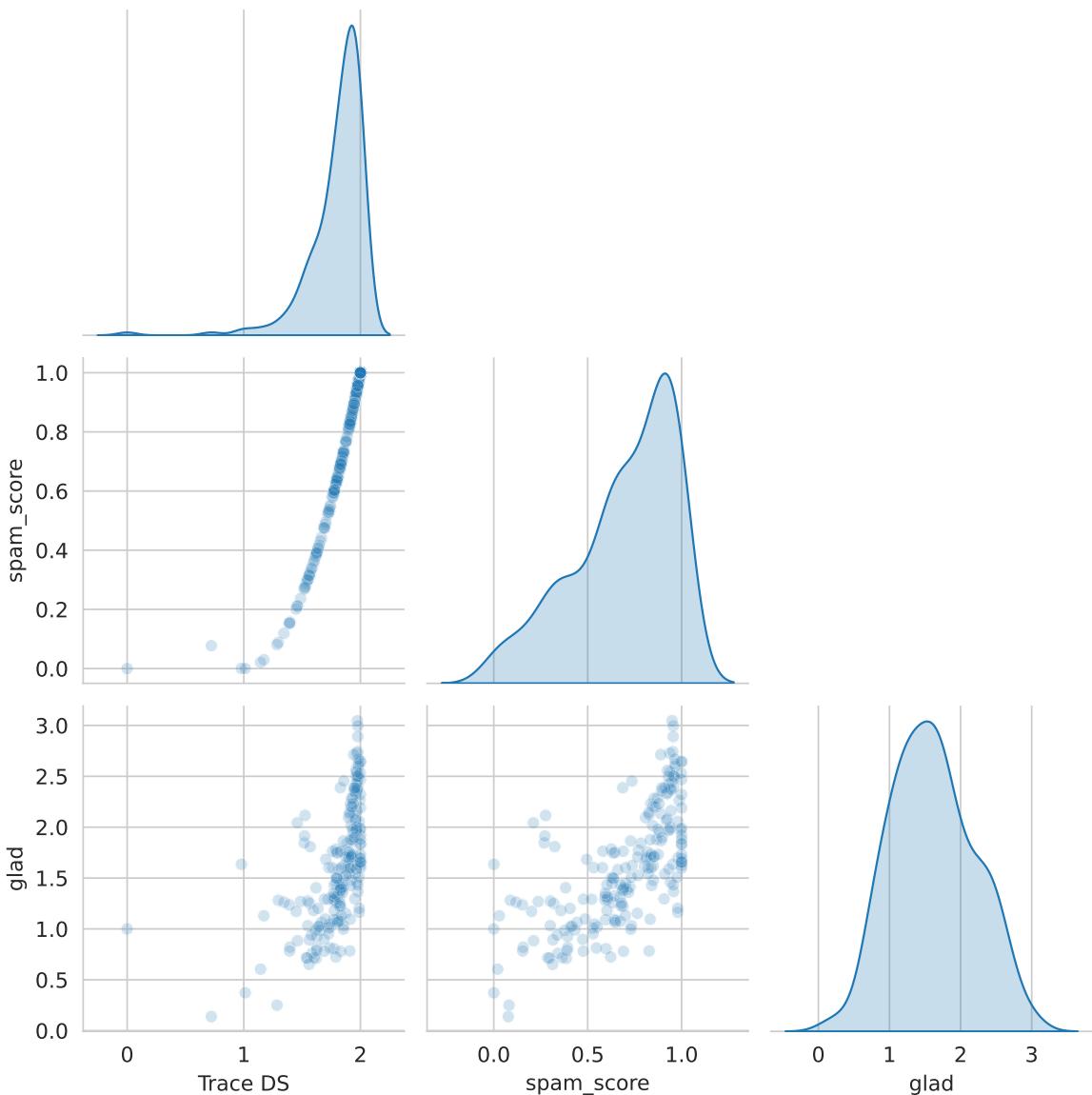


Figure 19: Comparison of ability scores by workers for the birds audio dataset. Most workers do seem to perform similarly, with very little noise voluntarily induced.

597 From Figure 19, we notice that very few workers are identified as spammers and that different

598 worker identification strategies seem to perform similarly. Here we show the worse workers' indices
599 depending on each strategy.

600 Worse workers using GLAD [94 80 109 35 45]
601 Worse workers using DS trace [69 94 172 109 35]
602 Worse workers using Spam Score [69 109 172 35 130]

603 One of the closing statements of Lehikoinen et al. (2023) is “we learned lessons for how to better
604 implement similar citizen science projects in the future”. On one hand, identifying the most ambiguous
605 tasks can help by saving only these tasks to the most expert workers and acquiring better data. On
606 the other hand, combining the task difficulty with the worker ability performance metrics could help
607 to create personal feeds of tasks to label and generate more worker participation. Finally, the label
608 aggregation step can lead to training classifiers with better labels. We hope that allowing easy access
609 thanks to the peerannot library to each of those steps can indeed help to better implement citizen
610 science projects and use the collected data.

- 611 Aitchison, L. 2021. “A Statistical Theory of Cold Posteriors in Deep Neural Networks.” In *ICLR*.
612 Cao, P, Y Xu, Y Kong, and Y Wang. 2019. “Max-MIG: An Information Theoretic Approach for Joint
613 Learning from Crowds.” In *ICLR*.
614 Chagneux, M, S LeCorff, P Gloaguen, C Ollion, O Lepâtre, and A Brûge. 2023. “Macrolitter Video
615 Counting on Riverbanks Using State Space Models and Moving Cameras.” *Computo*, February.
616 <https://computo.sfds.asso.fr/published-202301-chagneux-macrolitter>.
617 Chu, Z, J Ma, and H Wang. 2021. “Learning from Crowds by Modeling Common Confusions.” In
618 *AAAI*, 5832–40.
619 Dawid, AP, and AM Skene. 1979. “Maximum Likelihood Estimation of Observer Error-Rates Using
620 the EM Algorithm.” *J. R. Stat. Soc. Ser. C. Appl. Stat.* 28 (1): 20–28.
621 Deng, J., W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. 2009. “ImageNet: A Large-Scale Hierarchical
622 Image Database.” In *CVPR*.
623 Gao, G, and D Zhou. 2013. “Minimax Optimal Convergence Rates for Estimating Ground Truth from
624 Crowdsourced Labels.” *arXiv Preprint arXiv:1310.5764*.
625 Garcin, C., A. Joly, P. Bonnet, A. Affouard, J.-C. Lombardo, M. Chouet, M. Servajean, T. Lorieul, and
626 J. Salmon. 2021. “Pl@ntNet-300K: A Plant Image Dataset with High Label Ambiguity and a
627 Long-Tailed Distribution.” In *Proceedings of the Neural Information Processing Systems Track on
628 Datasets and Benchmarks*.
629 Gruber, S G, and F Buettner. 2022. “Better Uncertainty Calibration via Proper Scores for Classification
630 and Beyond.” In *Advances in Neural Information Processing Systems*.
631 Guo, C, G Pleiss, Y Sun, and KQ Weinberger. 2017. “On Calibration of Modern Neural Networks.” In
632 *ICML*, 1321.
633 Imamura, H, I Sato, and M Sugiyama. 2018. “Analysis of Minimax Error Rate for Crowdsourcing and
634 Its Application to Worker Clustering Model.” In *ICML*, 2147–56.
635 James, GM. 1998. “Majority Vote Classifiers: Theory and Applications.” PhD thesis, Stanford
636 University.
637 Kasmi, G, Y-M Saint-Drenan, D Trebosc, R Jolivet, J Leloux, B Sarr, and L Dubus. 2023. “A Crowd-
638 sourced Dataset of Aerial Images with Annotated Solar Photovoltaic Arrays and Installation
639 Metadata.” *Scientific Data* 10 (1): 59.
640 Khattak, FK. 2017. “Toward a Robust and Universal Crowd Labeling Framework.” PhD thesis,
641 Columbia University.
642 Krizhevsky, A, and G Hinton. 2009. “Learning Multiple Layers of Features from Tiny Images.”
643 University of Toronto.
644 Lefort, T, B Charlier, A Joly, and J Salmon. 2022. “Identify Ambiguous Tasks Combining Crowdsourced
645 Labels by Weighting Areas Under the Margin.” *arXiv Preprint arXiv:2209.15380*.

- Lehikoinen, P., M. Rannisto, U. Camargo, A. Aintila, P. Lauha, E. Piirainen, P. Somervuo, and O. Ovaskainen. 2023. “A Successful Crowdsourcing Approach for Bird Sound Classification.” *Citizen Science: Theory and Practice* 8 (1): 16. <https://doi.org/10.5334/cstp.556>.
- Lin, Tsung-Yi, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. 2014. “Microsoft COCO: Common Objects in Context.” *CoRR* abs/1405.0312. <http://arxiv.org/abs/1405.0312>.
- Marcel, S, and Y Rodriguez. 2010. “Torchvision the Machine-Vision Package of Torch.” In *Proceedings of the 18th ACM International Conference on Multimedia*, 1485–88. MM ’10. New York, NY, USA: Association for Computing Machinery.
- Moreau, Thomas, Mathurin Massias, Alexandre Gramfort, Pierre Ablin, Pierre-Antoine Bannier, Benjamin Charlier, Mathieu Dagréou, et al. 2022. “BenchOpt: Reproducible, Efficient and Collaborative Optimization Benchmarks.” In *NeurIPS*. <https://arxiv.org/abs/2206.13424>.
- Park, Seo Yeon, and Cornelia Caragea. 2022. “On the Calibration of Pre-Trained Language Models Using Mixup Guided by Area Under the Margin and Saliency.” In *ACML*, 5364–74.
- Passonneau, R J., and B Carpenter. 2014. “The Benefits of a Model of Annotation.” *Transactions of the Association for Computational Linguistics* 2: 311–26.
- Paszke, A, S Gross, F Massa, A Lerer, J Bradbury, G Chanan, T Killeen, et al. 2019. “PyTorch: An Imperative Style, High-Performance Deep Learning Library.” In *NeurIPS*, 8024–35.
- Peterson, J C., R M. Battleday, T L. Griffiths, and O Russakovsky. 2019. “Human Uncertainty Makes Classification More Robust.” In *ICCV*, 9617–26.
- Pleiss, G, T Zhang, E R Elenberg, and K Q Weinberger. 2020. “Identifying Mislabeled Data Using the Area Under the Margin Ranking.” In *NeurIPS*.
- Raykar, V C, and S Yu. 2011. “Ranking Annotators for Crowdsourced Labeling Tasks.” In *NeurIPS*, 1809–17.
- Rodrigues, F, M Lourenco, B Ribeiro, and F C Pereira. 2017. “Learning Supervised Topic Models for Classification and Regression from Crowds.” *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39 (12): 2409–22.
- Rodrigues, F, and F Pereira. 2018. “Deep Learning from Crowds.” In *AAAI*. Vol. 32.
- Rodrigues, F, F Pereira, and B Ribeiro. 2014. “Gaussian Process Classification and Active Learning with Multiple Annotators.” In *ICML*, 433–41. PMLR.
- Servajean, M, A Joly, D Shasha, J Champ, and E Pacitti. 2016. “ThePlantGame: Actively Training Human Annotators for Domain-Specific Crowdsourcing.” In *Proceedings of the 24th ACM International Conference on Multimedia*, 720–21. MM ’16. New York, NY, USA: Association for Computing Machinery.
- . 2017. “Crowdsourcing Thousands of Specialized Labels: A Bayesian Active Training Approach.” *IEEE Transactions on Multimedia* 19 (6): 1376–91.
- Sinha, V B, S Rao, and V N Balasubramanian. 2018. “Fast Dawid-Skene: A Fast Vote Aggregation Scheme for Sentiment Classification.” *arXiv Preprint arXiv:1803.02781*.
- Tinati, R, M Luczak-Roesch, E Simperl, and W Hall. 2017. “An Investigation of Player Motivations in Eyewire, a Gamified Citizen Science Project.” *Computers in Human Behavior* 73: 527–40.
- Ustalov, Dmitry, Nikita Pavlichenko, and Boris Tseitlin. 2023. “Learning from Crowds with CrowdKit.” *arXiv*. <https://arxiv.org/abs/2109.08584>.
- Whitehill, J, T Wu, J Bergsma, J Movellan, and P Ruvolo. 2009. “Whose Vote Should Count More: Optimal Integration of Labels from Labelers of Unknown Expertise.” In *NeurIPS*. Vol. 22.
- Yasmin, R, M Hassan, J T Grassel, H Bhogaraju, A R Escobedo, and O Fuentes. 2022. “Improving Crowdsourcing-Based Image Classification Through Expanded Input Elicitation and Machine Learning.” *Frontiers in Artificial Intelligence* 5: 848056.
- Zhang, H, M Cissé, Y N. Dauphin, and D Lopez-Paz. 2018. “Mixup: Beyond Empirical Risk Minimization.” In *ICLR*.