Supplemental Materials

Simulation-based Exploration for Aggregation Algorithms in Human+AI Crowd: What factors should we consider for better results?

Takumi Tamura Hiroyoshi Ito Satoshi Oyama
Atsuyuki Morishima

Contents

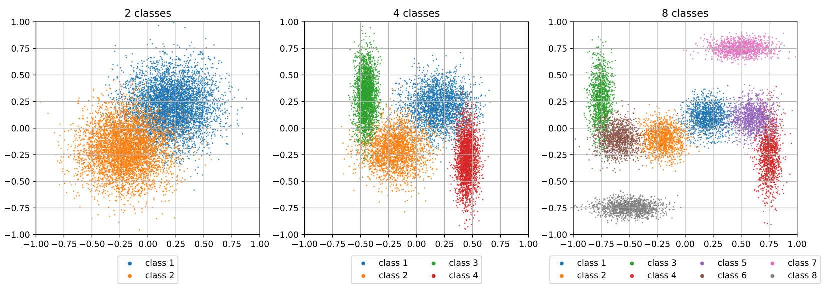


Figure 1: Datasets of our experiments

1 Human+AI Crowd Simulation Model

1.1 Model Definition

This section explains the crowd simulation model we used for the experiments with various configurations encountered in the human+AI crowd. The idea behind this model is that it has a task-feature-space centered design so that (1) workers have realistic sets of confusion matrices and (2) we can implement complex behaviors of AI workers, such as imbalanced ability, in a transparent way. These aspects are essential for simulating aggregation algorithms in the human+AI crowd.

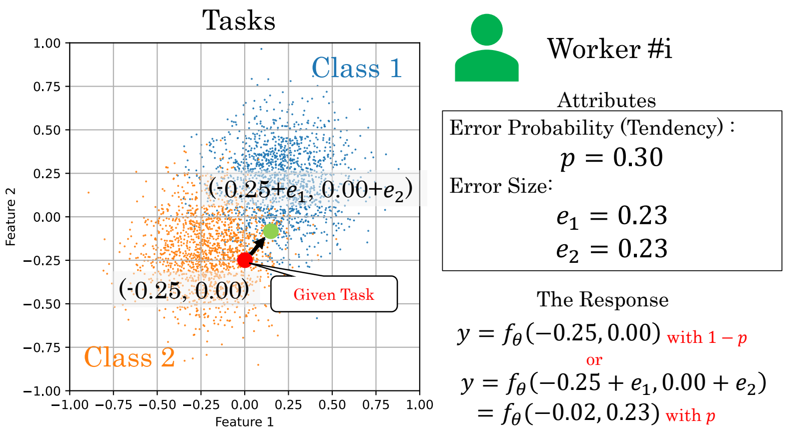
1.1.1 Task Representation and Datasets

The simulation model generates a set of multiple classification tasks based on a synthesized dataset in the n-dimensional feature space and yields the results of simulated workers' completing the tasks. Here, each data point belongs to one class, and each task asks workers to provide a classification label for the data point. Figure 1 illustrates examples of tasks represented by three two-dimensional datasets with 2, 4, and 8 classes, respectively.

The data points in a dataset are randomly generated from a Gaussian distribution and the number of data points belonging to each class is equal because it simplifies the analysis of how the aggregation results are affected from the imbalanced ability of AI workers.

The distributions of the classes in the dataset exhibit various overlap patterns in the feature space. This allows us to generate a natural variety of confusion matrices by introducing fluctuations and biases in worker responses within the feature space. For example, classes that are adjacent to each other more often cause confusion for workers than classes that are isolated in the space.

Each dataset is divided into two parts. The first serves as the training



Tasks
Class 1
Class 2
(-0.25+e_1, 0.00+e_2)
(-0.25, 0.00)
Given Task
Feature 1
Feature 2
Worker #i
Attributes
Error Probability (Tendency) :
$p = 0.30$
Error Size:
$e_1 = 0.23$
$e_2 = 0.23$
The Response
$y = f_\theta(-0.25, 0.00)$ with $1-p$
or
$y = f_\theta(-0.25 + e_1, 0.00 + e_2)$
$= f_\theta(-0.02, 0.23)$ with p

Figure 2: Overview of our human worker simulation. Note that f_θ is the pre-trained classifier.

data for machine learning models that simulate the basic behavior models (f_θ) of human workers and the behavior of AI workers. The second serves as the test data, generating tasks to be assigned to the workers.

1.1.2 Human Worker Simulation

To simulate human workers, we need to ensure that high-quality workers result in good confusion matrices. Gaussian Naive Bayes classifiers meet this requirement when data points follow the Gaussian distribution. Therefore, this model generates human workers' task results using the classifier f_θ, which is a Gaussian Naive Bayes model trained using the training data from the dataset. The answer to the task with the data point (F_1, F_2) is represented as $f_\theta(F_1, F_2)$. To simulate lower-quality workers with realistic confusion matrices, we introduce two parameters for each person: error probability p and error vector $\vec{e}$ representing the direction and size of errors in the task space.

Figure 2 illustrates a case with a two-dimension space. In the simulator, p for each worker is randomly generated from uniform distributions, and the error vector $\vec{e} = (e_1, e_2)$ in the figure is also randomly set in a specific range. With these attributes, the responses of each human worker are generated as follows. Given the data point (F_1, F_2) is provided to the worker, the response is $f_\theta(F_1, F_2)$ when the worker does not make an error with the probability $1 - p$; otherwise, the response is $f_\theta(F_1 + e_1, F_2 + e_2)$ when the worker makes an error with the probability p.

Table 1: Experimental settings

Configurations		Options (One is chosen)
#Feature Dimensions	n	2
#Data points		10,000
#Train Data		7,000
#Test Data (#Tasks)		3,000
Range of Error Vectors	$\vec{e}$	$\lvert e_1\rvert < 0.25 \land \lvert e_2\rvert < 0.25$
#Class	c	2,4,8
#Tasks per Human Workers	t	5,10,20,30,50,100
#Human Task Duplicates	r	3,5,10
#Tasks per AI workers	g	5,10,20,30,50,100, 200, 300,500,1000,1500,3000 (Where $g \geq t$)
Type of AI Workers' Ability		balanced or imbalanced
#Total Cases		**1,026 cases**

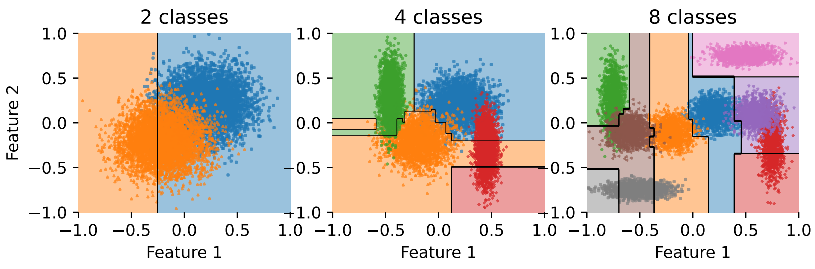


Figure 3: Decision boundaries of AI workers with imbalanced ability

1.1.3 AI Worker Simulation

When simulating black-box AI workers, their internal mechanisms do not matter. What is important is transparency so that we can analyze the behavior of AI workers to explain the simulation result. For that purpose, the model adopts decision trees for simulated AI workers. AI workers with balanced ability can be implemented by training them with the whole training data for each dataset, while a variety of AI workers with imbalanced ability can be implemented by training decision trees with subsets of the training data. Figure 3 shows examples of decision boundaries of AI workers with imbalanced ability constructed this way.

Table 2: Ability and imbalance of AI workers

Datasets	2 classes	4 classes	8 classes
AI Worker (with balanced ability)			
Accuracy	0.849	0.882	0.934
AI Worker (with imbalanced ability)			
Accuracy	0.691	0.752	0.766
The weak class	Class 2 (orange)	Class 4 (red)	Class 4 (red), Class 8 (gray)
Accuracy of the weak class	0.397	0.163	0.322 (Class 4), 0.120 (Class 8)
The drop condition	$F_1 < -0.25 \land (F_1, F_2) \in \text{Class2}$	$F_2 > -0.5 \land (F_1, F_2) \in \text{Class4}$	$(F_2 > -0.5 \land (F_1, F_2) \in \text{Class4})$, $(F_1 > -0.75 \land (F_1, F_2) \in \text{Class8})$

1.2 Datasets, Parameters, and Settings in Experiments

Table 1 summarizes the settings of the simulation model in the experiment. We added one type of AI worker (with balanced or imbalanced ability) to the worker set and aggregated their responses. For each type of AI worker, we conducted simulations with combinations of the parameters c, t, r, g. In total, we examined 1,026 cases.

1.2.1 Datasets for Tasks

They were represented by the two-dimensional datasets shown in Figure 1. Each dataset has 10,000 data points. Thirty percent (3,000) of the data points were used as test data and assigned as tasks to workers. The remaining seventy percent were used as training data. As the number of classes increases, the relationship among classes varies more.

The three datasets in the paper were generated randomly generated from Gaussian distributions with the specific parameters of each class. The specific parameters (mean and variance) are shown in Table 3. In addition, the CSV data of the datasets are available in `datasets/`.

1.2.2 AI Workers

They were trained with the training data for each dataset. AI workers with balanced ability were trained with all of the training data, while AI workers with imbalanced ability were trained with the imbalanced training data, which dropped the data points satisfying particular conditions (shown in Table 2) so that the confusion matrix shows low performance for particular classes. Figure 3 shows the decision boundaries of each AI worker with imbalanced ability. Class 2 (orange) with 2 classes, Class 4 (red) with 4 classes, and Class 4 (red) and Class 8 (gray) with 8 classes were the classes that each AI worker with imbalanced ability has problems classifying, and we call them the *weak class*. Table 2 shows the performance of AI workers in each dataset. The performance is the same for every combination of parameters in Table 1.

1.2.3 Human Workers

Given the datasets, the error vector $\vec{e} = (e_1, e_2)$ of each worker was randomly set under the condition $|e_1| \leq 0.25 \land |e_2| \leq 0.25$. This reflects the fact that people often answer an incorrect class that is close to the correct one. For the datasets, an error vector larger than this one will cause many mistakes between far classes. The performance of human workers is shown in later

Table 3: The specific parameters when generating the datasets

(a) 2 Classes

Class	Feature 1	Feature 2
0	$\mathcal{N}(0.2, 0.2^2)$	$\mathcal{N}(0.2, 0.2^2)$
1	$\mathcal{N}(-0.2, 0.2^2)$	$\mathcal{N}(-0.2, 0.2^2)$

(b) 4 Classes

Class	Feature 1	Feature 2
0	$\mathcal{N}(0.2, 0.15^2)$	$\mathcal{N}(0.2, 0.15^2)$
1	$\mathcal{N}(-0.2, 0.15^2)$	$\mathcal{N}(-0.2, 0.15^2)$
2	$\mathcal{N}(-0.45, 0.05^2)$	$\mathcal{N}(0.3, 0.2^2)$
3	$\mathcal{N}(0.45, 0.05^2)$	$\mathcal{N}(-0.3, 0.2^2)$

(c) 8 Classes

Class	Feature 1	Feature 2
0	$\mathcal{N}(0.2, 0.1^2)$	$\mathcal{N}(0.1, 0.1^2)$
1	$\mathcal{N}(-0.2, 0.1^2)$	$\mathcal{N}(-0.1, 0.1^2)$
2	$\mathcal{N}(-0.75, 0.05^2)$	$\mathcal{N}(0.25, 0.2^2)$
3	$\mathcal{N}(0.75, 0.05^2)$	$\mathcal{N}(-0.25, 0.2^2)$
4	$\mathcal{N}(0.6, 0.1^2)$	$\mathcal{N}(0.1, 0.1^2)$
5	$\mathcal{N}(-0.6, 0.1^2)$	$\mathcal{N}(-0.1, 0.1^2)$
6	$\mathcal{N}(0.5, 0.15^2)$	$\mathcal{N}(0.75, 0.05^2)$
7	$\mathcal{N}(-0.5, 0.15^2)$	$\mathcal{N}(-0.75, 0.05^2)$

figures (Figure 4 or Figure 5) as the accuracy of aggregation results of human workers.

1.2.4 Task Assignment to Human Workers

We changed the number of task assignments to human workers and the number of duplicate tasks for aggregations with the parameters t and r, respectively. The combinations of the parameters cover many cases seen in real-world crowdsourcing scenarios. Given the number of the tasks $(3,000)$, t, and r, the total number of human workers is $\frac{t}{3000} \times r$.

1.2.5 Task Assignment to AI Workers

To investigate the effects of potential significant gap in the number of tasks completed by AI and human workers, we introduced the parameter g (shown in Table 2) to represent the number of tasks assigned to AI Workers. For example, $g = 3,000$ means an AI worker completes all of the 3,000 tasks in the dataset. When $g = 1,500$ $(1,000)$, each of two (three) AI workers completes $1,500$ $(1,000)$ tasks. This way, although the total number of task assignments to AI workers remains the same, the gap becomes lower when g becomes lower. We use the same model for all of the AI workers.

1.3 Python Implementation

1.3.1 The responses of human workers

The python implementation of the human response generation are available at `human_worker_simulation_model.py` for reproducibility.

First, you have to set up the virtual environment. We used Poetry[1] for library management.

```
$ poetry install     # Set up the virtual environment and install libraries
$ poetry shell       # Activate the virtual environment
(.venv) $
```

Next, you can generate human worker responses the following commands.

```
(.venv) $ python human_worker_simulation_model.py -h
usage: human_worker_simulation_model.py [-h] seed c t r output_file

The generator of human worker responses.

positional arguments:
  seed         The random seed
  c            The number of classes of the classification tasks, 2, 4, or 8.
  t            The number of tasks per human worker
  r            The number of human duplicate task assignments
  output_file  The path of the output file

options:
  -h, --help   show this help message and exit
```

For example,

```
(.venv) $ python human_worker_simulation_model.py 123 2 5 3 sample.csv
```

Where $(c, t, r) = (2, 5, 3)$ with `seed=123`. The generated `sample.csv` have the following columns: `worker,task,label`.

1.3.2 The responses of AI Workers

The responses and `scikit-learn` models in the paper are available.

[1]https://python-poetry.org/

AI_workers/responses the responses have the following columns: `worker,task,label`.

AI_workers/models the `sklearn.tree.DecisionTreeClassifier`[2] models dumped by `pickle`[3].

2 Results in Experiment 1

Accuracy. Figure 4 shows the results with AI workers with balanced ability. Each plot shows the accuracy (Y-axis) of aggregation results in each combination of (c, t, r) with different gs (as X-axis). Similarly, Figure 5 shows the results with AI workers with imbalanced ability. Each plot shows the accuracy (Y-axis) of aggregation results in each combination of (c, t, r) with different gs (as X-axis).

Similarity to AI worker's results. This is an important criterion for evaluating aggregation algorithms, especially when the AI worker has imbalanced ability because higher similarity to AI's results means that the aggregated results inherit their imbalanced predictions. Each plot in Figure 6 shows the similarity (Cohen's κ in the Y-axis) of the aggregation results to the results with AI workers when they have imbalanced ability in the tasks belonging to the weak classes (shown in Table 2).

3 The DS+OneCoin model

3.1 Overview

The DS+OneCoin model is a variation of the basic DS model, which estimates the confusion matrix of workers using the maximum likelihood estimation (MLE) in the M-step based on the EM algorithm. The difference is that the DS+OneCoin model introduces a latent variable a to represent the accuracy of each worker (probability that the worker answers correctly) and estimates it using MLE with the estimated true labels from the previous iteration at the beginning of the M-step. The estimated a is used to regularize MLE of confusion matrices, so the DS+OneCoin model is the hybrid aggregation algorithm of the DS and OneCoin models.

More precisely, the DS+OneCoin model replaces the MLE of the confusion matrices with the MAP (maximum a posteriori) estimation using the prior distribution assumed by the accuracy variables a. Let π_j be each row of

[2]`https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html`

[3]`https://docs.python.org/3/library/pickle.html`

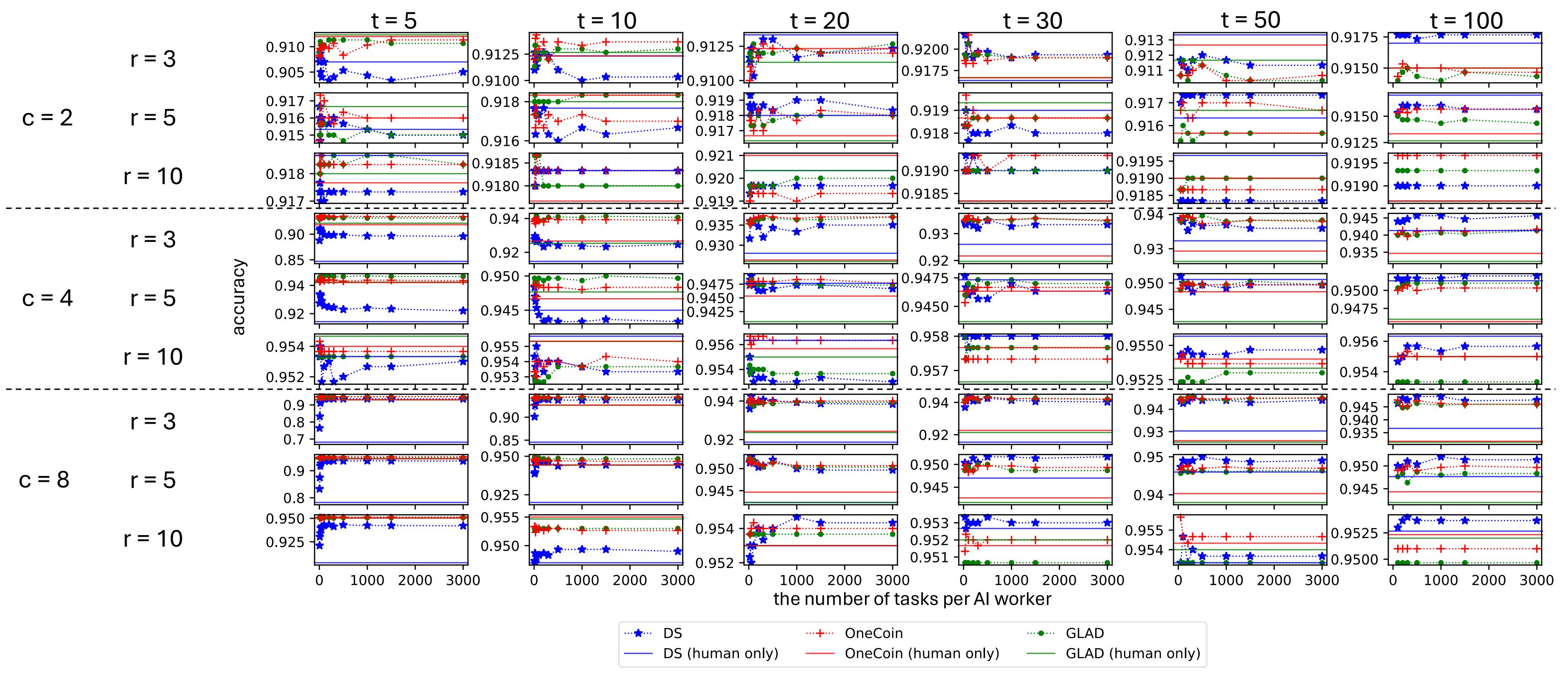


Figure 4: Accuracy of aggregation results involving AI workers with balanced ability in Experiment 1. Each Y-axis represents the accuracy of the aggregated results, and each X-axis represents the number of tasks per AI worker g. The larger g indicates that the difference in the number of tasks completed between human and AI workers is large. Note that the solid lines represent the aggregation accuracy of human workers only.

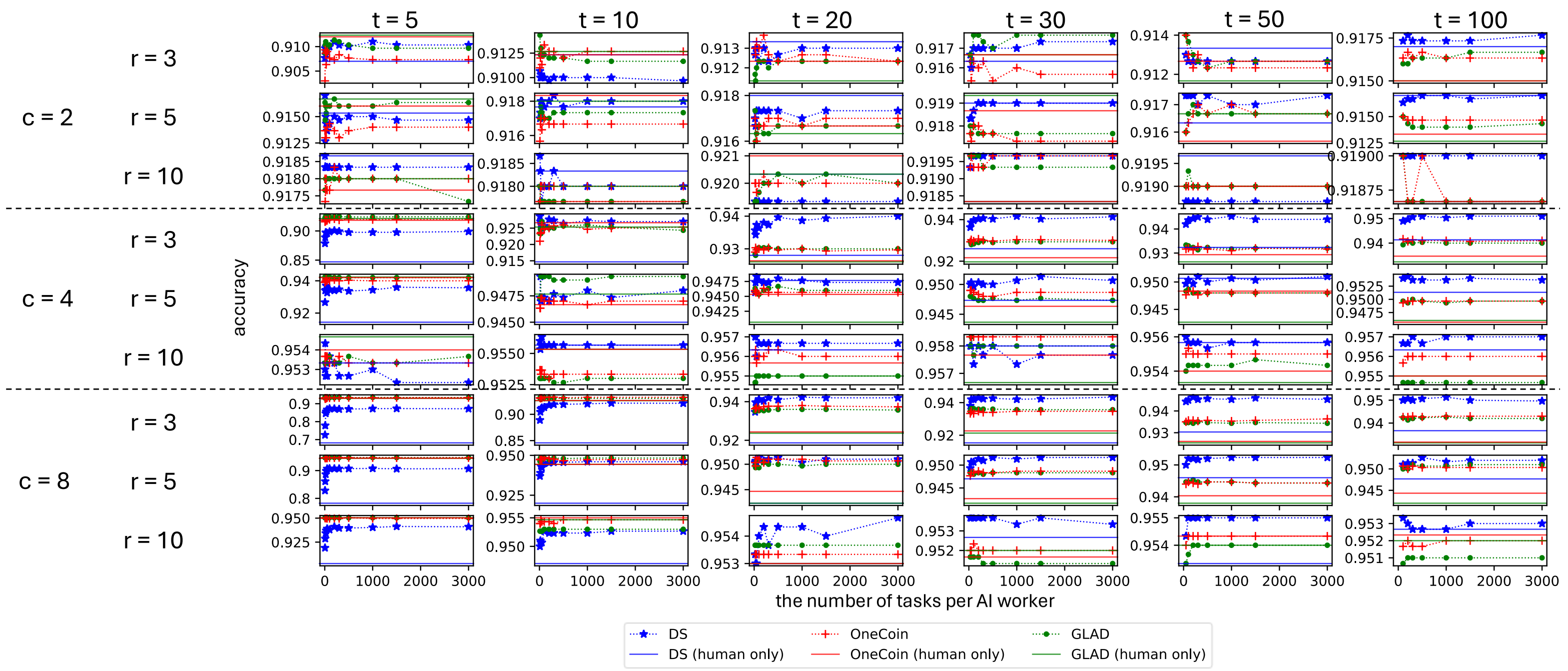


Figure 5: Accuracy of aggregation results involving AI workers with imbalanced ability in Experiment 1

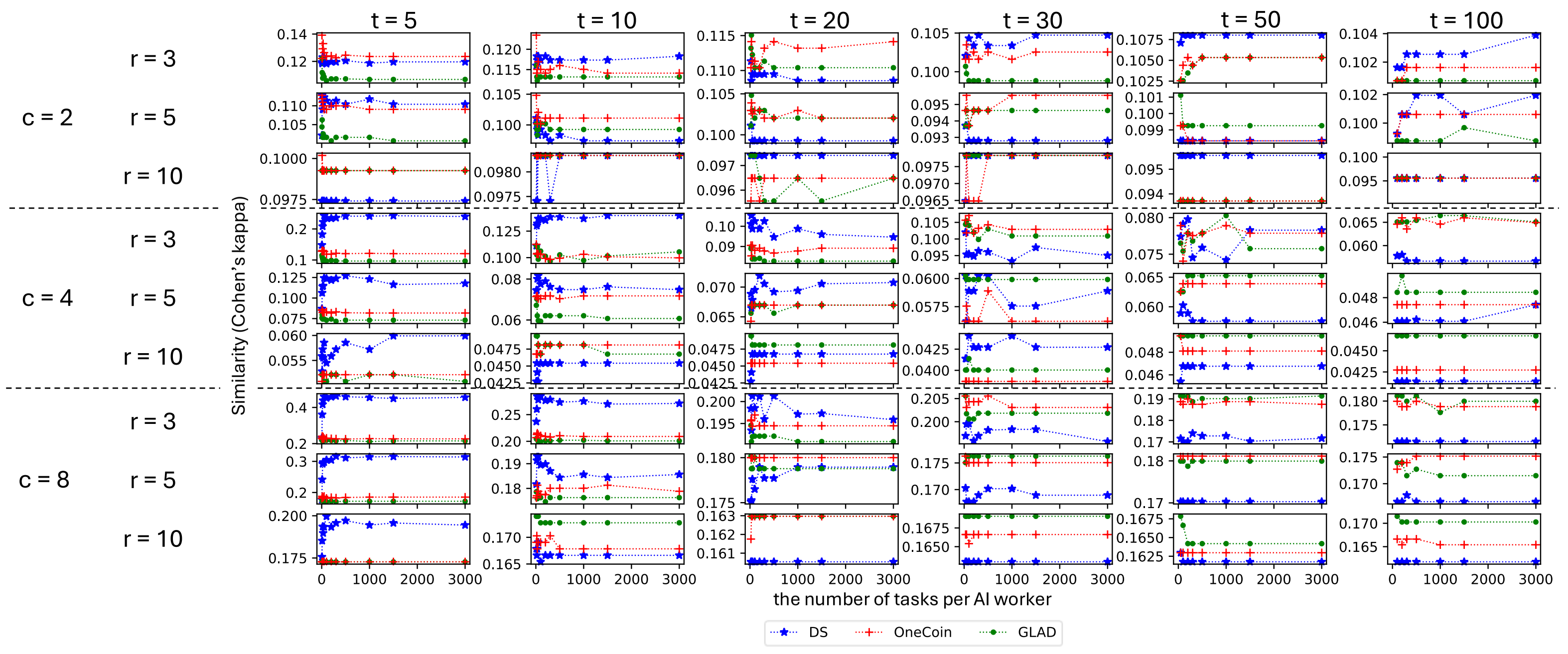


Figure 6: Similarity of aggregation results involving AI workers with imbalanced ability in Experiment 1. Each Y-axis represents Cohen's κ between the AI worker's responses and the aggregated results in the tasks belonging to the weak classes.

Table 4: The notation table of Section 3.2

Notation	Description
$\mathbb{J}$	Set of classes in the classification tasks
$\mathbb{I}$	Set of tasks
$\mathbb{K}$	Set of workers
T_{ij}	The binary value whether the true class of task i is the class j ($T_{ij} = 1$ if that is true, $T_{ij} = 0$ otherwise)
E_{ij}	Expected value of T_{ij}
$n_{ij}^{(k)}$	The number of times that the worker k answers the class j on the task i
p_j	The marginal probability of the class j
$\pi^{(k)}$	The confusion matrix of the worker k ($\lvert\mathbb{J}\rvert \times \lvert\mathbb{J}\rvert$ matrix) so $\pi_{jl}^{(k)}$ means the probability that the worker k answers the class l when the true class is j
$a^{(k)}$	The probability that the worker k answers true classes
$\xi_j^{(k)}$	The parameter of the Dirichlet distribution for $\pi_j^{(k)}$ (vector of length $\lvert\mathbb{J}\rvert$)
S	The hyperparameter of the DS+OneCoin model, an arbitrary positive integer

each confusion matrix, which is a $c \times c$ matrix. In other words, π_j means the probability distribution about a class choice when the true class of the tasks is j. π_j is the parameter of the categorical distribution, and so the DS+OneCoin model assumes the prior distribution about π_j using the Dirichlet distribution as the following equation.

$$\pi_j \sim \mathrm{Dir}(\xi_j) \tag{1}$$

where ξ_j is the parameter of the Dirichlet distribution. ξ_j is a vector of length c. Each element of ξ_j is defined as follows. Let S be an arbitrary positive integer.

$$\xi_{jl} = \begin{cases} aS & \text{if } j = l, \\ \frac{1-a}{c-1}S & \text{otherwise}, \end{cases} \tag{2}$$

where S is the hyperparameter that controls how much information from the prior distribution is used. This value is the same for all workers.

3.2 Detailed Formalization

Table 4 shows the notation table in this section.

3.2.1 The Dawid–Skene model

We introduce the Dawid-Skene (DS) model before explaining the DS+OneCoin model, because the DS+OneCoin model is the extension of the DS model.

Problem Definition: The goal of the DS model is to predict the true class of the tasks from the noisy responses of the workers. Let $\mathbb{J}$ be a set of classes in the classification tasks, and the set of workers is represented as $\mathbb{K}$. $\mathbb{I}$ is a set of tasks and T_{ij} is the ground truth, which is the binary value of whether the true class of task $i \in \mathbb{I}$ is the class $j \in \mathbb{J}$. ($T_{ij} = 1$ if this is true, $T_{ij} = 0$ otherwise). Obtaining E_{ij}, the expected value of T_{ij}, is the objection of the DS model.

Observable Variables: $n_{ij}^{(k)}$ is only the observable variables in the DS and DS+OneCoin models and it represents the number of times that the worker k answers the class $j \in \mathbb{J}$ on the task $i \in \mathbb{I}$.

Latent Variables: The DS model has two types of latent variables.

(1) Confusion matrix. Let $\pi^{(k)}$ be the confusion matrix of the worker $k \in \mathbb{K}$. $\pi^{(k)}$ is a $|\mathbb{J}| \times |\mathbb{J}|$ matrix. Each element is represented as $\pi_{jl}^{(k)}$, and $\pi_{jl}^{(k)}$ means the probability that the worker $k \in \mathbb{K}$ answers the class $l \in \mathbb{J}$ when the true class is $j \in \mathbb{J}$, so there is the following condition, $\sum_{l \in \mathbb{J}} \pi_{jl}^{(k)} = 1$. The DS model estimates workers' ability as $\pi^{(k)}$.

(2) Class Probability. Let p_j be the marginal probability of the class $j \in \mathbb{J}$. This means that the DS model estimates the task ratio of each class.

Algorithm: The DS model is based on the EM algorithm, which iterates the E-step and the M-step. In the E-step, it predicts the true class of the tasks.

$$E_{ij} = \frac{p_j \prod_{l \in \mathbb{J}} \prod_{k \in \mathbb{K}} (\pi_{jl}^{(k)})^{n_{il}^{(k)}}}{\prod_{q \in \mathbb{J}} p_q \prod_{l \in \mathbb{J}} \prod_{k \in \mathbb{K}} (\pi_{ql}^{(k)})^{n_{il}^{(k)}}}. \tag{3}$$

In the M-step, it estimates the latent variables by maximum likelihood estimation (MLE).

$$p_j = \frac{\sum_{i \in \mathbb{I}} E_{ij}}{|\mathbb{I}|}, \tag{4}$$

$$\pi_{jl}^{(k)} = \frac{\sum_{i \in \mathbb{I}} E_{ij} n_{il}^{(k)}}{\sum_{m \in \mathbb{J}} \sum_{i \in \mathbb{I}} E_{ij} n_{im}^{(k)}}. \tag{5}$$

The initial value of E_{ij} is given as follows.

$$E_{ij} = \frac{\sum_{k \in \mathbb{K}} n_{ij}^{(k)}}{\sum_{l \in \mathbb{J}} \sum_{k \in \mathbb{K}} n_{il}^{(k)}}. \tag{6}$$

3.2.2 The DS+OneCoin model

The DS+OneCoin model modifies the M-step of the DS model. It regularizes the MLE of confusion matrices to improve the estimation of workers' ability.

At the beginning of the M-step, the DS+OneCoin model estimates workers' accuracy, which is an additional latent variable. Let $a^{(k)}$ be the probability that the worker k answers true classes.

$$a^{(k)} = \frac{1}{\sum_{i\in\mathbb{I}}\sum_{j\in\mathbb{J}} n_{ij}^{(k)}} \sum_{i\in\mathbb{I}}\sum_{j\in\mathbb{J}} E_{ij} n_{ij}^{(k)}. \tag{7}$$

The DS+OneCoin model assumes the prior distribution for $\pi_j^{(k)}$ (the j-th row of $\pi^{(k)}$), which is the parameter of the Categorical distribution. Let $\xi_j^{(k)}$ be a parameter of the prior distribution.

$$\pi_j^{(k)} \sim \mathrm{Dir}(\xi_j^{(k)}). \tag{8}$$

$\xi_j^{(k)}$ is a vector of length $|\mathbb{J}|$, and each elements $\xi_{jl}^{(k)}$ is defined as follows. Let S be an arbitrary positive integer.

$$\xi_{jl}^{(k)} = \begin{cases} a^{(k)} S & \text{if } j = l, \\ \frac{1-a^{(k)}}{|\mathbb{J}|-1} S & \text{otherwise} \end{cases}. \tag{9}$$

Where S is the hyperparameter that controls how much information from the prior distribution is used ($S = 0$ is same as the basic DS model). This value is the same for all workers.

The DS+OneCoin model estimates workers' confusion matrix by MAP (maximum a posteriori) estimation in the M-step. The MLE of $\pi_{jl}^{(k)}$ is replaced as follows.

$$\pi_{jl}^{(k)} = \frac{\sum_{i\in\mathbb{I}} E_{ij} n_{il}^{(k)} + \xi_{jl}^{(k)}}{\sum_{m\in\mathbb{J}}\sum_{i\in\mathbb{I}} E_{ij} n_{im}^{(k)} + \sum_{m\in\mathbb{J}} \xi_{jm}^{(k)}}. \tag{10}$$

From the formula $\sum_{m\in\mathbb{J}} \xi_{jm}^{(k)} = S$, the above update formula is also represented as follows.

$$\pi_{jl}^{(k)} = \begin{cases} \frac{\sum_{i\in\mathbb{I}} E_{ij} n_{il}^{(k)} + a^{(k)} S}{\sum_{m\in\mathbb{J}}\sum_{i\in\mathbb{I}} E_{ij} n_{im}^{(k)} + S} & \text{if } j = l, \\ \frac{\sum_{i\in\mathbb{I}} E_{ij} n_{il}^{(k)} + \frac{1-a^{(k)}}{|\mathbb{J}|-1} S}{\sum_{m\in\mathbb{J}}\sum_{i\in\mathbb{I}} E_{ij} n_{im}^{(k)} + S} & \text{otherwise} \end{cases}. \tag{11}$$

In the DS+OneCoin model, it is the same as in the DS model that the E step, the estimation of p_j in the M step, and the initialization.

3.3 Python Implementation

`ds_plus_onecoin_model.py` is the python implementation of the DS+OneCoin model in the paper.

First, you have to set up the virtual environment. We used Poetry[4] for library management.

```
$ poetry install    # Set up the virtual environment and install libraries
```

This implementation is the extension of the Dawid-Skene model in crowd-kit[5], so you can use the same interfaces of crowd-kit. You can set the value of $S$ by the `smooth` arguments.

```
from crowdkit.datasets import load_dataset

from ds_plus_onecoin_model import DSPlusOneCoin

df, gt = load_dataset('relevance-2')
model = DSPlusOneCoin(smooth=1000) # The smooth parameter is $S$ in the paper.
result = model.fit_predict(df)
```

However, there is a important limitation that the `label` columns must be `int` and be consecutive numbers started with 0, for example, 0 or 1 in 2-class tasks, 0, 1, 2, or 3 in 4-class tasks (`relevance-2` meets this condition).

4 Results in Experiment 2

We conducted Experiment 2 to evaluate the performance of the DS+OneCoin model with the extensive set of configurations same as Experiment 1 (total 1,026 cases).

Figure 7 illustrates the results of Experiment 2 about AI workers with balanced ability. This figure corresponds Figure 4 in Experiment 1. It shows that the DS+OneCoin model performed better than the basic DS model when t or g was small.

Figure 8 shows the accuracy results about AI workers with imbalanced ability. This figure corresponds Figure 5 in Experiment 1. It clearly demonstrates the DS+OneCoin model outperformed other algorithms with smaller ts or gs.

In addition to the accuracy, Figure 9 shows the similarity of aggregation results involving AI workers with imbalanced ability in Experiment 2. This figure corresponds Figure 6 in Experiment 1. It also illustrates the DS+OneCoin model avoided to generate lower-quality results inherited from imbalanced AI workers when t was small.

[4] `https://python-poetry.org/`

[5] `https://github.com/Toloka/crowd-kit`

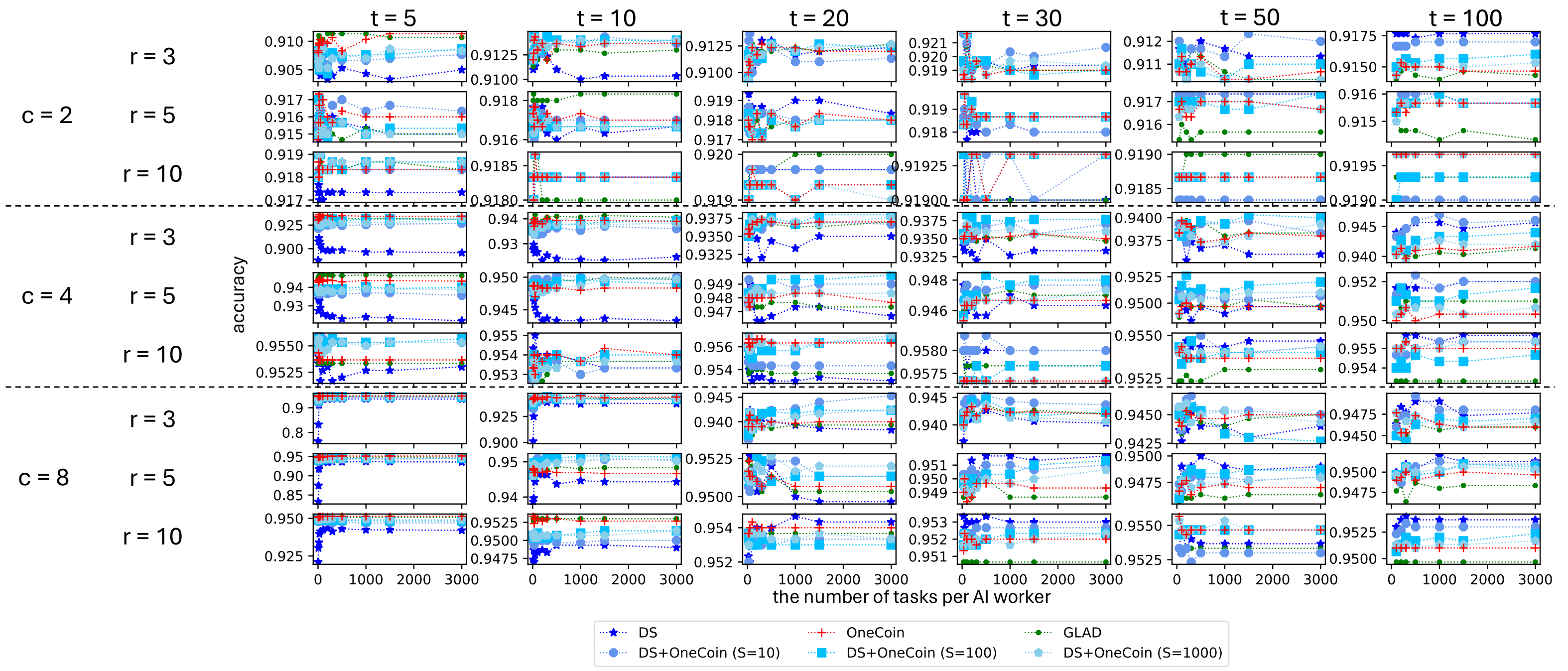


Figure 7: Accuracy of aggregation results involving AI workers with balanced ability in Experiment 2. Each Y-axis represents the accuracy of the aggregated results, and each X-axis represents the number of tasks per AI worker g. The larger g indicates that the difference in the number of tasks completed between human and AI workers is large. Note that this corresponds Figure 4

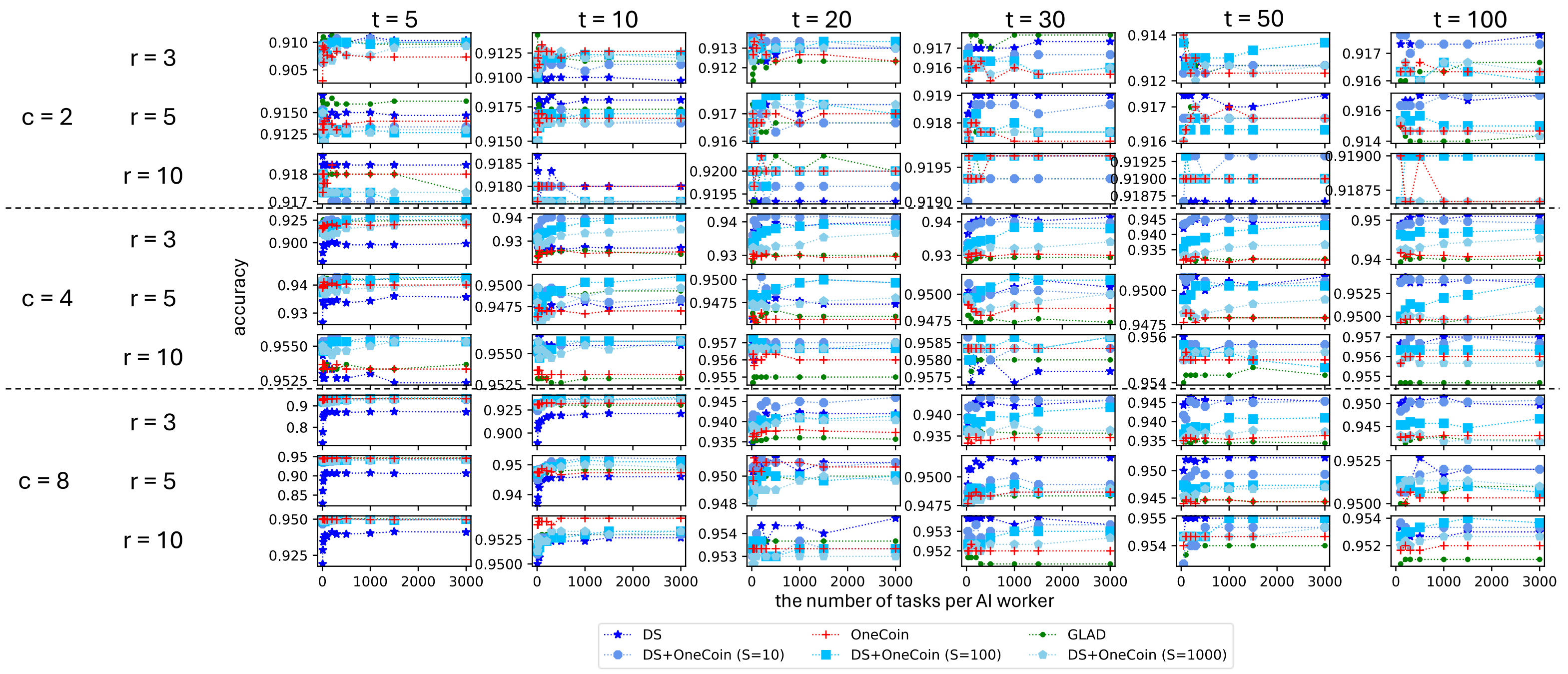


Figure 8: Accuracy of aggregation results involving AI workers with imbalanced ability in Experiment 2. Each Y-axis represents the accuracy of the aggregated results, and each X-axis represents the number of tasks per AI worker g. The larger g indicates that the difference in the number of tasks completed between human and AI workers is large. Note that this corresponds Figure 5.

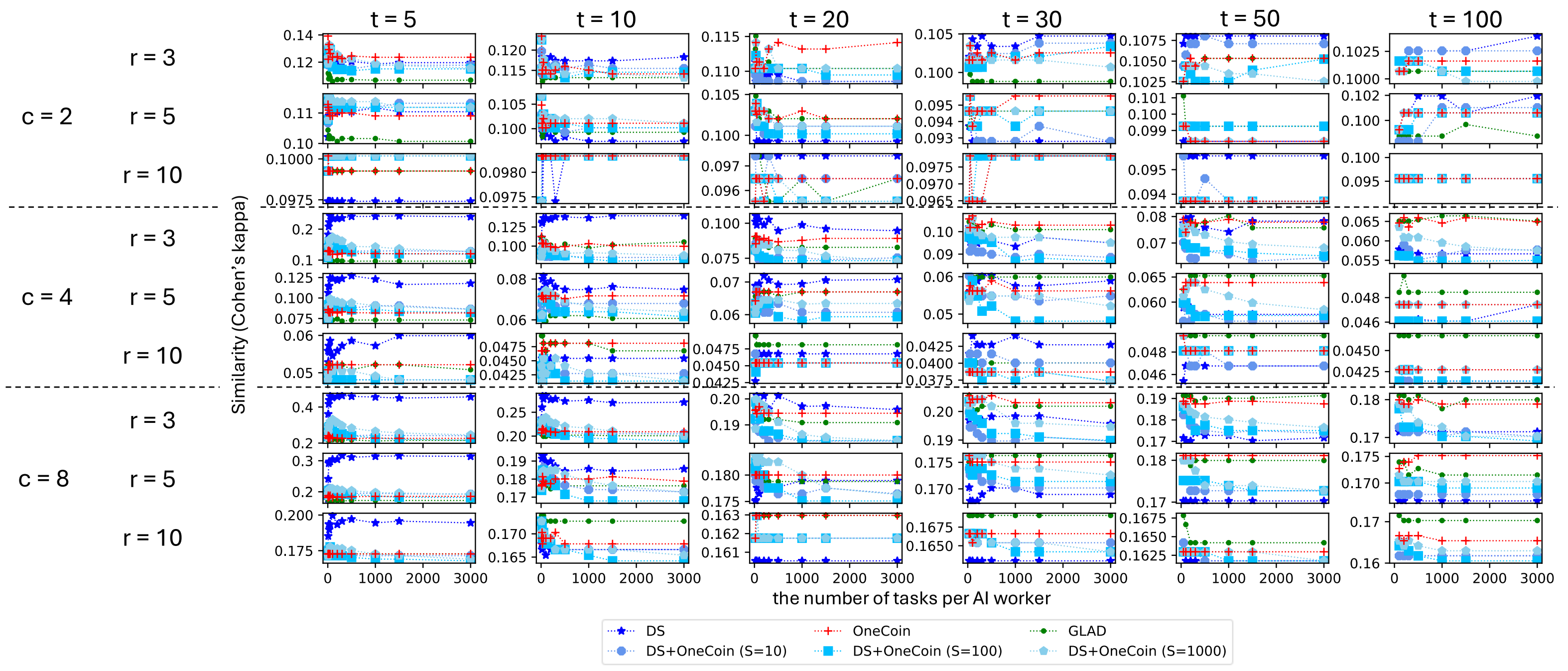


Figure 9: Similarity of aggregation results involving AI workers with imbalanced ability in Experiment 2. Each Y-axis represents the Cohen's κ between the AI worker's responses and the aggregated results in the tasks belonging to the weak classes. Note that this corresponds Figure 9.