SAND2022-14375C

Supplementary Material for: The Evaluation and Calibration of Epistemic Uncertainty Estimates

A. ADDITIONAL EXPERIMENTS

In Section 5.3, we created calibration datasets that resembled the evaluation datasets to evaluate our EU estimators. Since we cannot guarantee that this will hold in practice, in this section, we present two modifications to the toy data experiments from Section 5.3 in order to explore the effectiveness of calibration when the calibration datasets differ from the evaluation datasets.

A.1. Modification 1: Different Calibration Circles

For this experiment, we carve three large circles instead of twelve small circles in the calibration dataset, as shown in Fig. 1. Note that this also results in the calibration dataset having fewer points removed than the evaluation dataset. Also, note that due to there being only three circles, we can only choose three of the ratios used to carve the circles from the evaluation datasets.

In the top half of Table 1, we display the evaluation metrics, averaged over the six seeds. The results are very similar to the results from Section 5.3; one difference is that the standard deviation of ρ_{epi} for $\hat{\mathcal{E}}_{\text{M}}$ is lower. From Fig. 4, we see that the calibration curve for $\hat{\mathcal{E}}_{\text{E}}$ looks almost identical to the corresponding curve from Section 5.3. On the other hand, the curves for $\hat{\mathcal{E}}_{\text{M}}$ look slightly different in that the yellow calibrated $\hat{\mathcal{E}}_{\text{M}}$ curve slightly overestimates the accuracy gain in the middle intervals. These results suggest when the shapes carved out in the evaluation and calibration datasets differ, the performance of $\hat{\mathcal{E}}_{\text{M}}$ may decrease; however, in both cases, creating the calibration datasets helps improve the EECE of $\hat{\mathcal{E}}_{\text{M}}$. The performance of $\hat{\mathcal{E}}_{\text{E}}$ appears to be less sensitive to a mismatch in the shape of the region carved out.

A.2. Modification 2: High Aleatoric Uncertainty

In our experiments on toy and real data, we found that in general, $\hat{\mathcal{E}}_{\text{E}}$ was the superior EU estimator: it has a higher epistemic correlation, and results in lower EECE after calibration. The only advantage that $\hat{\mathcal{E}}_{\text{M}}$ has is that its uncalibrated estimates are useable as direct estimators of EU. In this section, we show that in the presence of aleatoric uncertainty, the performance of $\hat{\mathcal{E}}_{\text{E}}$ can decrease.

For this experiment, of the twelve circles we carve out, half of them do not have many points removed. For the circles where we did not remove many points, we only include test points on the boundary of the two classes, which are the points with high aleatoric uncertainty. However, the calibration datasets are created as normal: $\mathcal{D}_{\text{test},2}$ are uniformly sampled from each ball. This setup is visualized in Fig. 3.

From the bottom half of Table 1, we observe that the epistemic correlation of $\hat{\mathcal{E}}_{\text{E}}$ is decreases on $\mathcal{D}_{\text{test}}$, and $\hat{\mathcal{E}}_{\text{M}}$ is comparatively better. From Fig. 4, we see that the calibrated EU for $\hat{\mathcal{E}}_{\text{E}}$ overestimates the true accuracy gain for the higher intervals, so the calibrated EECE is higher than the EECE of $\hat{\mathcal{E}}_{\text{M}}$. This is because in the regions of high aleatoric uncertainty without many points removed, $\hat{\mathcal{E}}_{\text{E}}$ still assigns high EU to the test points because they lie on the boundary between the two classes; $\hat{\mathcal{E}}_{\text{M}}$ correctly estimates the EU of these points to be small, so its calibrated EECE does not deteriorate. Although the performance of $\hat{\mathcal{E}}_{\text{E}}$ decreases overall, it still is able to estimate the EU for the lower intervals properly after calibration.

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

B. TRAINING THE LARGE SCALE GAUSSIAN PROCESS CLASSIFIER

We use the GPyTorch implementation[1] of Wenzel et al., 2019. We use the squared exponential kernel:

$$k(\boldsymbol{x}_1, \boldsymbol{x}_2) = \sigma^2 \exp\left(\frac{1}{2l}||\boldsymbol{x}_1 - \boldsymbol{x}_2||^2\right).$$

The large scale Gaussian process classifier has several hyperparameters:

- The number of inducing points
- The initial guess for the inducing point locations
- The initial guess of the kernel parameters

We initialize the kernel parameters with $\sigma^2 = 0.4805, l = 0.2$, which are the default values from the GPyTorch tutorial. The number of inducing points should be large enough to produce a good approximation, but small enough to make computation times reasonable. We use 2500 inducing points for the toy data experiments, and 2000 inducing points for the real data experiments. We initialize the inducing point locations by randomly sampling points from the training data. In addition, there are also several optimization hyperparameters:

- The number of epochs
- The batch size
- The initial step sizes

The large scale GP classifier implementation uses two optimizers: natural gradient descent for learning the inducing point locations and variational parameters, and Adam for learning the kernel parameters. We set the initial step sizes to be 0.025 and 0.01, respectively. For the toy data, we train for 150 epochs; for the real data, we train for 250 epochs. These numbers were picked to ensure that the loss curve indicates a local minimum is reached. We found that the batch size should be set to as large as possible in order to improve optimization. We set the batch size to 17,264, which is the maximum we were able to set without running into out-of-memory errors. Training was done on one compute node with a CUDA 11.2.2 GPU.

C. DETAILS ON CREATING THE DATASETS

C.1. Pseudocode for the Data Splitting Procedure

Algorithm 1: `train_test_aug_split` Procedure

```
1  given dataset $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$, number of balls to carve out $M$, radius size $R$, ratios $p_j, q_j$ for
     $j \in \{1, \ldots, M\}$;
2  initialize $\mathcal{I}_{\text{carve}} = [1, 2, \ldots, N], \mathcal{I}_{\text{train}} = [\,], \mathcal{I}_{\text{add}} = [\,], \mathcal{I}_{\text{test}} = [\,]$;
3  for $j = 1$ to M do
4      $\boldsymbol{c} \leftarrow$ get_ball_center($j, \ldots$);
5      $\mathcal{I}_{\text{ball}} \leftarrow \{i \in \mathcal{I}_{\text{carve}} : \|\boldsymbol{x}_i - \boldsymbol{c}\| < R\}$;
6      $\mathcal{I}_{\text{carve}} \leftarrow \mathcal{I}_{\text{carve}} \setminus \mathcal{I}_{\text{ball}}$;
7      $\mathcal{I}_{\text{ball\_train}}, \mathcal{I}_{\text{ball\_test}} \leftarrow$ train_test_split($\mathcal{I}_{\text{ball}}, p_j$);
8      $\mathcal{I}_{\text{ball\_train}}, \mathcal{I}_{\text{ball\_add}} \leftarrow$ train_test_split($\mathcal{I}_{\text{ball\_train}}, q_j$);
9      $\mathcal{I}_{\text{train}} \leftarrow \mathcal{I}_{\text{train}} \cup \mathcal{I}_{\text{ball\_train}}$;
10     $\mathcal{I}_{\text{add}} \leftarrow \mathcal{I}_{\text{add}} \cup \mathcal{I}_{\text{ball\_add}}$;
11     $\mathcal{I}_{\text{test}} \leftarrow \mathcal{I}_{\text{test}} \cup \mathcal{I}_{\text{ball\_test}}$;
12 end
13 $\mathcal{D}_{\text{train}} \leftarrow \{(\boldsymbol{x}_i, y_i) \in \mathcal{D} : i \in \mathcal{I}_{\text{carve}} \cup \mathcal{I}_{\text{train}}\}$;
14 $\mathcal{D}_{\text{aug}} \leftarrow \mathcal{D}_{\text{train}} \cup \{(\boldsymbol{x}_i, y_i) : i \in \mathcal{I}_{\text{add}}\}$;
15 $\mathcal{D}_{\text{test}} \leftarrow \{(\boldsymbol{x}_i, y_i) \in \mathcal{D} : i \in \mathcal{I}_{\text{test}}\}$;
16 return $\mathcal{D}_{\text{train}}, \mathcal{D}_{\text{aug}}, \mathcal{D}_{\text{test}}$
```

[1] https://docs.gpytorch.ai/en/latest/examples/04_Variational_and_Approximate_GPs /PolyaGamma_Binary_Classification.html

Algorithm 1 is the procedure we use to create $\mathcal{D}_{\text{train}}$, $\mathcal{D}_{\text{test}}$, and $\mathcal{D}_{\text{aug}}$ from a given dataset $\mathcal{D}$ for each of our experiments. It on the following sub-procedures:

- `train_test_split` is a function that takes in a set and a ratio p and returns two sets with the first set having $p\%$ of the elements and the second set having the remaining elements.
- `get_ball_center` is a function that returns the center of the region to carve out in each iteration. We implement it differently for the toy data and the real data.

C.1.1 `get_ball_center` Pseudocode

On the toy data, `get_ball_center` simply returns a ball center from a prespecified list of locations:

Algorithm 2: `get_ball_center` Procedure for Toy Data

```
1 given dataset $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{N}$, ball centers $\boldsymbol{c}_1, \ldots, \boldsymbol{c}_M$, iteration number $j$;
2 return $\boldsymbol{c}_j$
```

On the real data, `get_ball_center` randomly samples K available points and returns a point that contains roughly L neighbors within radius R:

Algorithm 3: `get_ball_center` Procedure for Real Data

```
1 given dataset $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{N}$, available indices $\mathcal{I}_{\text{carve}}$, desired number of neighbors $L$, number of
    samples $K$, radius size $R$;
2 sample $i_1, \ldots, i_K$ from $\mathcal{I}_{\text{carve}}$ without replacement;
3 $n_i \leftarrow |\{m \in \mathcal{I} : \|\boldsymbol{x}_i - \boldsymbol{x}_m\| < R\}|$ for $i \in \{i_1, \ldots, i_K\}$;
4 $\mathcal{J} \leftarrow \{i \in \{i_1, \ldots, i_K\} : n_i \leqslant L\}$;
5 $i^* \leftarrow \arg\max_{i \in \mathcal{J}} n_i$;
6 return $\boldsymbol{x}_{i^*}$
```

C.2. Pseudocode for Section A.2 Experiment

We use a slightly different `train_test_aug_split` procedure for the experiment from Section A.2. We need modify the step where we add the test data, since we want some of the points to have high aleatoric uncertainty. We add a new input to the algorithm: a Boolean value for each ball to carve that determines whether or not the test data for that ball will have high aleatoric uncertainty. We present the Pseudocode in Algorithm 4.

In Algorithm 4, $\texttt{knn}(\mathcal{D}, \boldsymbol{x})$ is a function that returns the labels of the six nearest neighbors to $\boldsymbol{x}$ in $\mathcal{D}$. For a given ball to carve out, to determine which test points to add, if A_j is set to True, we remove all points from $\mathcal{I}_{\text{ball},2}$ whose neighbors do not have sufficient disagreement; that is, there are 5 or more members of the same class. Otherwise, we uniformly subset from $\mathcal{I}_{\text{ball},2}$. Also, note that we split the data in the first step, reserving 7/8 of the data for to add to the test data. Because of this, when we generate the data, we sample eight times more data than in the regular toy data experiments.

C.3. Algorithm Parameters for Each Experiment

We list the arguments used in Algorithms 1, 2, 3 and 4 to create the evaluation and calibration datasets for each experiment in Tables 2 and 3. The parameters M, R, $\boldsymbol{p}$ and $\boldsymbol{q}$ are selected so that:

- $\mathcal{D}_{\text{train}}$ is large enough that the estimate of the kernel parameters in the large scale GP does not change significantly from $\mathcal{D}_{\text{aug}}$.
- There is variety in how much data is subsetted, so that in some regions, $\mathcal{D}_{\text{aug}}$ improves the prediction significantly over $\mathcal{D}_{\text{train}}$, while in other regions, there is not much improvement.
- $\mathcal{D}_{\text{test}}$ is sufficiently large so that we can see whether there is a trend in how the accuracy gain changes versus estimated EU value.

For the real datasets, L is smaller for EMNIST because it has fewer observations than K-49, so we cannot remove as many points. We set K to be larger when creating the calibration datasets because after subsetting the data, we need to search longer to find data-rich regions. We display the resulting dataset sizes in Table 4, where we give the average and standard deviation of the sizes over the six seeds.

Algorithm 4: `train_test_aug_split` Procedure with High Aleatoric Uncertainty Test Data

```
1  given dataset $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{N}$, number of balls to carve out $M$, radius size $R$, ratios $p_j, q_j$, high
     aleatoric uncertainty determiners $A_j$ for $j \in \{1, \ldots, M\}$;
2  initialize $\mathcal{I}_{\text{carve}} = [1, 2, \ldots, N], \mathcal{I}_{\text{train}} = [\,], \mathcal{I}_{\text{add}} = [\,], \mathcal{I}_{\text{test}} = [\,]$;
3  $\mathcal{I}_{\text{reserve}}, \mathcal{I}_{\text{carve}} \leftarrow$ train_test_split$(\mathcal{I}_{\text{carve}}, 0.875)$;
4  for $j = 1$ to M do
5      $\boldsymbol{c} \leftarrow$ get_ball_center$(j, \ldots)$;
6      $\mathcal{I}_{\text{ball}} \leftarrow \{i \in \mathcal{I}_{\text{carve}} : \|\boldsymbol{x}_i - \boldsymbol{c}\| < R\}$;
7      $\mathcal{I}_{\text{carve}} \leftarrow \mathcal{I}_{\text{carve}} \backslash \mathcal{I}_{\text{ball}}$;
8      $\mathcal{I}_{\text{ball\_train}}, \_ \leftarrow$ train_test_split$(\mathcal{I}_{\text{ball}}, p_j)$;
9      $\mathcal{I}_{\text{ball\_train}}, \mathcal{I}_{\text{ball\_add}} \leftarrow$ train_test_split$(\mathcal{I}_{\text{ball\_train}}, q_j)$;
10     $\mathcal{I}_{\text{train}} \leftarrow \mathcal{I}_{\text{train}} \cup \mathcal{I}_{\text{ball\_train}}$;
11     $\mathcal{I}_{\text{add}} \leftarrow \mathcal{I}_{\text{add}} \cup \mathcal{I}_{\text{ball\_add}}$;
12     $\mathcal{I}_{\text{ball,2}} \leftarrow \{i \in \mathcal{I}_{\text{reserve}} : \|\boldsymbol{x}_i - \boldsymbol{c}\| < R\}$;
13     if $A_j$ then
14         $y_{i1}, \ldots, y_{i6} \leftarrow$ knn$(\mathcal{D}, \boldsymbol{x}_i)$  for  $i \in \mathcal{I}_{\text{ball,2}}$;
15         $\mathcal{I}_{\text{ball\_test}} \leftarrow \{i \in \mathcal{I}_{\text{ball,2}} : 2 \leqslant y_{i1} + \cdots + y_{i6} \leqslant 4\}$;
16     else
17         $\mathcal{I}_{\text{ball\_test}}, \_ \leftarrow$ train_test_split$(\mathcal{I}_{\text{ball,2}}, 0.035)$
18     end
19     $\mathcal{I}_{\text{test}} \leftarrow \mathcal{I}_{\text{test}} \cup \mathcal{I}_{\text{ball\_test}}$;
20 end
21 $\mathcal{D}_{\text{train}} \leftarrow \{(\boldsymbol{x}_i, y_i) \in \mathcal{D} : i \in \mathcal{I}_{\text{carve}} \cup \mathcal{I}_{\text{train}}\}$;
22 $\mathcal{D}_{\text{aug}} \leftarrow \mathcal{D}_{\text{train}} \cup \{(\boldsymbol{x}_i, y_i) : i \in \mathcal{I}_{\text{add}}\}$;
23 $\mathcal{D}_{\text{test}} \leftarrow \{(\boldsymbol{x}_i, y_i) \in \mathcal{D} : i \in \mathcal{I}_{\text{test}}\}$;
24 return $\mathcal{D}_{\text{train}}, \mathcal{D}_{\text{aug}}, \mathcal{D}_{\text{test}}$
```

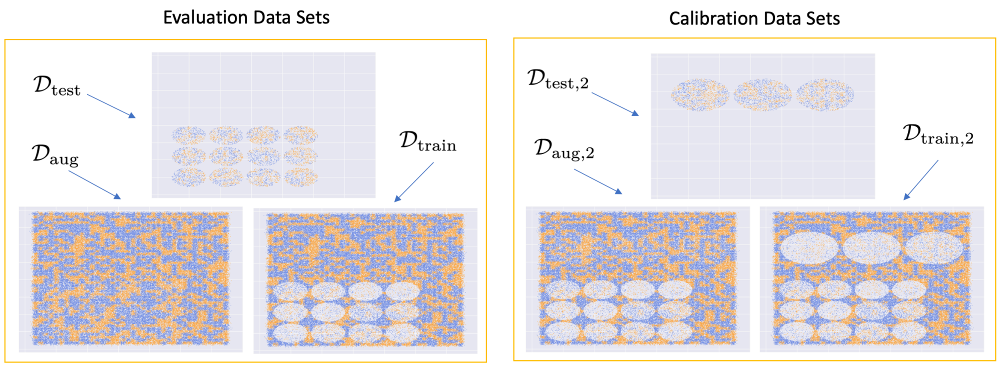

Figure 1: Toy Data, Modification 1

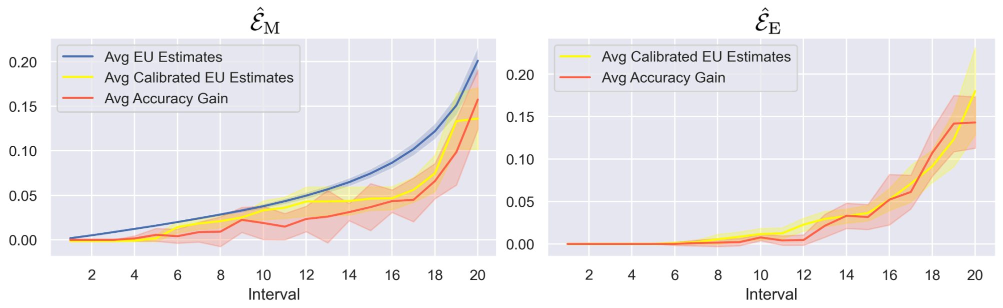

Figure 2: Average Epistemic Uncertainty for each Interval on Toy Data, Modification 1

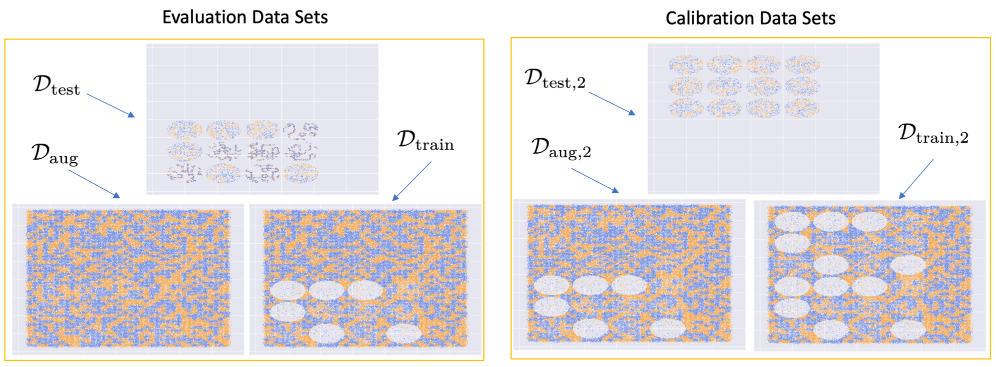

Figure 3: Toy Data, Modification 2

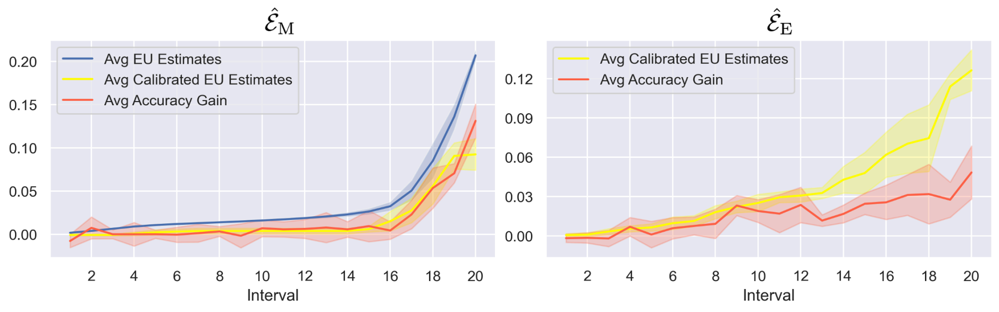

Figure 4: Average Epistemic Uncertainty for each Interval on Toy Data, Modification 2

Table 1: Evaluation Metrics on Modified Toy Data

	Method	ρ_{epi}	EECE	Calib. EECE
Mod. 1	$\hat{\mathcal{E}}_{\mathrm{M}}$	0.098 $\pm$ 0.013	$\mathbf{0.0271 \pm 0.005}$	0.0180 $\pm$ 0.004
	$\hat{\mathcal{E}}_{\mathrm{E}}$	$\mathbf{0.122 \pm 0.024}$	0.391 $\pm$0.006	$\mathbf{0.0154 \pm 0.0027}$
Mod. 2	$\hat{\mathcal{E}}_{\mathrm{M}}$	$\mathbf{0.064 \pm 0.01}$	$\mathbf{0.0214 \pm 0.003}$	$\mathbf{0.0114 \pm 0.0017}$
	$\hat{\mathcal{E}}_{\mathrm{E}}$	0.034 $\pm$ 0.011	0.56 $\pm$ 0.011	0.0234 $\pm$ 0.003

Table 2: Algorithm Inputs for Toy Data

Parameter	Reg.	Reg. (Calib.)	Mod. 1	Mod. 1 (Calib.)	Mod. 2	Mod. 2 (Calib.)
M	12	12	12	3	12	12
R	0.55	0.55	0.55	1.7	0.55	0.55
$\boldsymbol{p}$	[0.08, 0.08, 0.08, 0.08, 0.12, 0.12, 0.12, 0.12, 0.2, 0.2, 0.24, 0.24]	[0.08, 0.08, 0.08, 0.08, 0.12, 0.12, 0.12, 0.12, 0.2, 0.2, 0.24, 0.24]	[0.08, 0.08, 0.08, 0.08, 0.12, 0.12, 0.12, 0.12, 0.2, 0.2, 0.24, 0.24]	[0.08,0.12,0.16]	[0.06, 0.06, 0.06, 0.06, 0.06, 0.06, 0.7, 0.7, 0.7, 0.7, 0.7, 0.7]	[0.06, 0.06, 0.06, 0.06, 0.06, 0.06, 0.7, 0.7, 0.7, 0.7, 0.7, 0.7]
$\boldsymbol{q}$	[0.225, 0.225, 0.225, 0.225, 0.225, 0.225, 0.225, 0.225, 0.225, 0.225, 0.225, 0.225]	[0.225, 0.225, 0.225, 0.225, 0.225, 0.225, 0.225, 0.225, 0.225, 0.225, 0.225, 0.225]	[0.225, 0.225, 0.225, 0.225, 0.225, 0.225, 0.225, 0.225, 0.225, 0.225, 0.225, 0.225]	[0.225,0.225,0.225	[0.225, 0.225, 0.225, 0.225, 0.225, 0.225, 0.225, 0.225, 0.225, 0.225, 0.225, 0.225]	[0.225, 0.225, 0.225, 0.225, 0.225, 0.225, 0.225, 0.225, 0.225, 0.225, 0.225, 0.225]
$\boldsymbol{c}$	[(-1, -1), (0.2, -1), (1.4, -1), (2.6, -1), (-1, 0.2), (0.2, 0.2) (1.4, 0.2), (2.6, 0.2), (-1, 1.4), (0.2, 1.4), (1.4, 1.4), (2.6,1.4)]	[(-1, 2.6), (0.2, 2.6), (1.4, 2.6), (2.6, 2.6), (-1, 3.8), (0.2, 3.8) (1.4, 3.8), (2.6, 3.8), (-1, 5), (0.2, 5), (1.4, 5), (2.6,5)]	[(-1, -1), (0.2, -1), (1.4, -1), (2.6, -1), (-1, 0.2), (0.2, 0.2) (1.4, 0.2), (2.6, 0.2), (-1, 1.4), (0.2, 1.4), (1.4, 1.4), (2.6,1.4)]	[(-0.6, 3.8), (1.4, 3.8), (3.4, 3.8)]	[(-1, -1), (0.2, -1), (1.4, -1), (2.6, -1), (-1, 0.2), (0.2, 0.2) (1.4, 0.2), (2.6, 0.2), (-1, 1.4), (0.2, 1.4), (1.4, 1.4), (2.6,1.4)]	[(-1, 2.6), (0.2, 2.6), (1.4, 2.6), (2.6, 2.6), (-1, 3.8), (0.2, 3.8), (1.4, 3.8), (2.6, 3.8), (-1, 5), (0.2, 5), (1.4, 5), (2.6,5)]
$\boldsymbol{A}$	–	–	–	–	[False, False, False, False, False, False, True, True, True, True, True, True]	–

Table 3: Algorithm Inputs for Real Data

Parameter	EMNIST	EMNIST (Calib.)	K-49	K-49 (Calib.)
M	16	16	16	16
R	6.5	6.5	6	6
$\boldsymbol{p}$	[0.04, 0.04, 0.04, 0.0,4 0.08, 0.08, 0.08, 0.08, 0.12, 0.12, 0.12, 0.12, 0.16, 0.16, 0.16, 0.16]	[0.04, 0.04, 0.04, 0.04, 0.08, 0.08, 0.08, 0.08, 0.12, 0.12, 0.12, 0.12, 0.16, 0.16, 0.16, 0.16]	[0.1, 0.1, 0.1, 0.1, 0.2, 0.2, 0.2, 0.2, 0.3, 0.3, 0.3, 0.3, 0.4, 0.4, 0.4, 0.14]	[0.1, 0.1, 0.1, 0.1, 0.2, 0.2, 0.2, 0.2, 0.3, 0.3, 0.3, 0.3, 0.4, 0.4, 0.4, 0.4]
$\boldsymbol{q}$	[0.225, 0.225, 0.225,0.225, 0.225, 0.225, 0.225, 0.225, 0.225, 0.225, 0.225, 0.225]	[0.225, 0.225, 0.225, 0.225, 0.225, 0.225, 0.225, 0.225, 0.225, 0.225, 0.225, 0.225]	[0.225, 0.225, 0.225, 0.225, 0.225, 0.225, 0.225, 0.225, 0.225, 0.225, 0.225, 0.225]	[0.225, 0.225, 0.225, 0.225, 0.225, 0.225, 0.225, 0.225, 0.225, 0.225, 0.225, 0.225]
L	1100	1100	2400	2400
K	500	1000	500	1000

Table 4: Dataset Sizes

Dataset	$\mathcal{D}_{\text{train}}$	$\mathcal{D}_{\text{train},2}$	$\mathcal{D}_{\text{test}}$	$\mathcal{D}_{\text{test},2}$	$\mathcal{D}_{\text{aug}}$	$\mathcal{D}_{\text{aug},2}$
Toy (Reg)	112557 ± 38	83565 ± 42	6538 ± 9	6528 ± 4	135050 ± 9	106028 ± 39
Toy (Mod. 1)	112578 ± 47	90984 ± 65	6533 ± 10	4860 ± 6	135055 ± 10	107717 ± 50
Toy (Mod. 2)	121441 ± 117	100134 ± 97	8952 ± 233	6320 ± 4	138423 ± 23	117152 ± 112
EMNIST	116834 ± 39	100584 ± 107	3329 ± 9	3312 ± 15	128269 ± 9	111964 ± 57
K-49	244175 ± 91	217309 ± 207	6023 ± 21	6052 ± 29	264890 ± 21	238123 ± 114