Appendix for 'Active Testing: Sample–Efficient Model Evaluation'

Jannik Kossen * **Sebastian Farquhar** * **Yarin Gal** **Tom Rainforth**

A. Additional Experiments

A.1. Synthetic Data

Figure A.1 shows experiments on additional synthetic data together with the experiments familiar from Fig. 3 of the main paper. We show two additional regression settings: a GP on sinusoidal data with non-uniform densities in x and a GP on the quadratic data already familiar from Fig. 3 (b). Active testing yields gains in sample-efficiency for both of these additional settings. We give details for all synthetic experiments in Appendix C.2.

A.2. Radial BNN on MNIST

We also investigate active testing of a Radial BNN on MNIST in a large data regime with $50\,000$ training points. As acquisition strategy, we here simply use the predictive entropy of the Radial BNN, which we observe to be well-calibrated in this simple setting. Consequently, we see large gains in sample-efficiency of active testing over i.i.d. acquisition for this setting as shown in Fig. A.2. We discuss experimental details for this plot in Appendix C.8

A.3. Uncertainties and Statistical Significance

In Figs. A.3 to A.5, we show a variation of Figs. 4, 6 (a), and 8 (b) wherein we plot *mean* log squared differences instead of *median* squared differences.[2] All conclusions from the main body of the paper continue to hold for the mean-based visualization. Additionally, we quantify uncertainties on the mean values via the standard errors of the log squared differences.

We also investigate the statistical significance of the results reported in the paper, computing Wilcoxon signed-rank test statistics to examine if the best active testing strategy has lower population mean rank than all other shown methods. More precisely, when comparing two methods at a fixed acquisition step, we obtain a pair of samples of squared differences for each run, and we test against the alternative hypothesis that the best performing method has *lower* squared error. We compare methods at the last displayed acquisition step. The results of the test show that we can always reject the null hypothesis at 5×10^{-3} confidence level. We give the full results of the Wilcoxon signed-rank tests in Table 1.

B. Details on Active Testing

B.1. Further Details on Clipping

As mentioned briefly in the main paper, we are clipping the predicted $q(i_m)$ to avoid overly small acquisition values. We do this because $\hat{R}_{\text{LURE}}$ is only unbiased if we have non-zero acquisition probability on *all* points remaining (Farquhar et al., 2021). In practice, we bound the value of the acquisition function from below at α times the acquisition probability assigned by a uniform acquisition strategy, where $\alpha = 0.2$.

B.2. Computational Complexity

Two components of active testing contribute significantly to computational complexity: Evaluating the acquisition function on the remaining samples of the test pool and retraining the surrogate on all observed samples.

Acquisition Function Evaluation. At each active testing iteration, we require the evaluation of the acquisition function q

[2] Reminder: We calculate the squared difference between the true test loss on the full test set (known only to an oracle) and the estimator at each acquisition step.

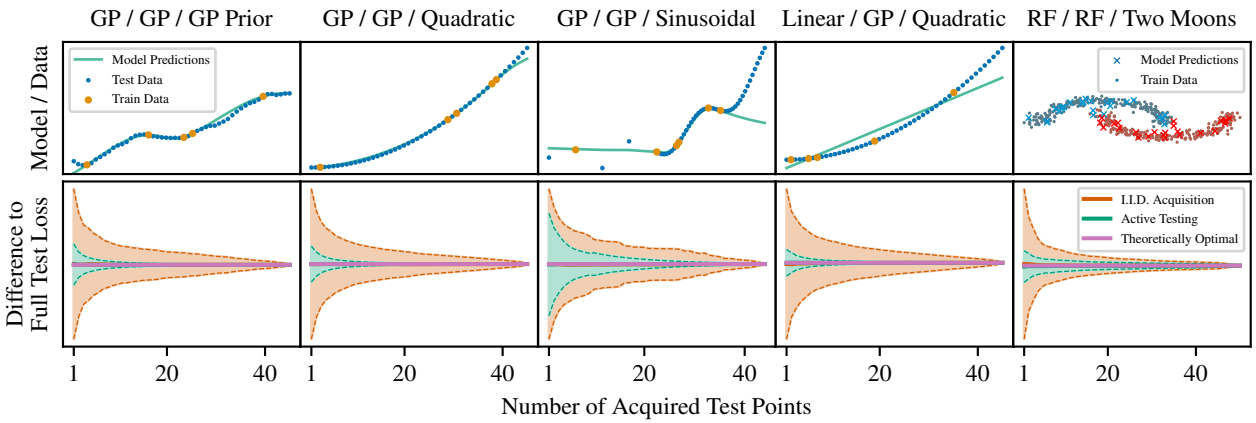

Figure A.1. Active testing on synthetic data. Columns (1, 4, 5) are repeated from Fig. 3 of the main paper, while (2–3) are exclusive to the appendix. Each column shows a different combination of *model/surrogate/data*. The first row shows model predictions, as well as the points used for training and testing for a single draw from the random data generation. The second row displays the mean difference of the estimators to the true loss on the full test set (known only to an oracle).

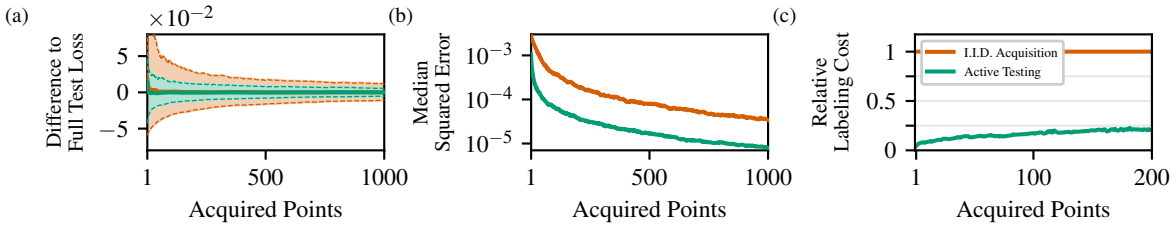

Figure A.2. Active testing for a Radial BNN on **MNIST** using model predictive entropy to estimate test cross-entropy loss. (a) Active testing gives unbiased loss estimators with (b) much lower median squared error. (c) We calculate the *relative labelling cost*: the sample-efficiency factor of i.i.d. acquisition to active testing at any number of acquired points. Lower is better and values less than 1 are gains in sample-efficiency. We plot medians and quantiles $(0.1, 0.9)$ and average over 992 runs.

on all remaining samples in the test pool $\mathcal{D}_{\text{test}}$. At first iteration, N samples remain in the test pool, contributing $\mathcal{O}(N)$ to the computational complexity. At each subsequent iteration m, $\mathcal{O}(N-m)$ additional evaluations are required. For a total of M acquisitions, we can bound the cost of evaluating the acquisition function with $\mathcal{O}(MN)$.

Surrogate Retraining. In our experiments in §5.2, we demonstrate successful active testing when retraining the surrogate every K steps or training the surrogate only once on $\mathcal{D}_{\text{train}}$. However, in active testing practice, label cost may be significantly larger than the cost of retraining. Therefore, we here assume the worst case $K = M$, i.e. retraining the surrogate after each acquisition. This gives maximum sample-efficiency because additional information from newly acquired test labels is directly used to improve surrogate predictions. At first iteration, only $\mathcal{D}_{\text{train}}$ is observed and hence retraining costs will be $\mathcal{O}(|\mathcal{D}_{\text{train}}|)$. As testing progresses, m samples of the test set become available such that retraining costs grow to $\mathcal{O}(|\mathcal{D}_{\text{train}}| + m)$ per iteration. For a total of M acquisitions, retraining costs therefore scale as $\mathcal{O}(M|\mathcal{D}_{\text{train}}| + M^2)$.

Therefore, we can bound the total computational complexity of active testing as $\mathcal{O}(M|\mathcal{D}_{\text{train}}| + M^2 + MN)$, which can be simplified to $\mathcal{O}(M|\mathcal{D}_{\text{train}}| + MN)$ as $M \leq N$ always.

C. Details on Experiments

C.1. Hardware

For the neural network architectures, we use PyTorch (Paszke et al., 2019). We use a mix of NVIDIA RTX 2080, Titan RTX, and K80 GPUs to run the experiments. No experiment requires more than one GPU in parallel. Experiments that do not require GPUs, such as the synthetic data experiments, can be run on CPUs only on conventional hardware.

Table 1. Wilcoxon signed-rank tests for experiments from the main paper, comparing the squared differences of the best method against all other shown approaches. The alternative hypothesis is that the 'best method' has lower squared error than the 'other method'.

Figure	Dataset	Best Method	Other Method	Acq. Step	p-Value
4 a	MNIST	BNN Surrogate	I.I.D. Acquisition	1000	5.438×10^{-61}
	MNIST	BNN Surrogate	Random Forrest Surrogate	1000	7.069×10^{-19}
	MNIST	BNN Surrogate	Original Model	1000	9.524×10^{-20}
4 b	Fashion-MNIST	Random Forrest Surrogate	Original Model	1000	6.586×10^{-125}
	Fashion-MNIST	Random Forrest Surrogate	I.I.D. Acquisition	1000	4.176×10^{-31}
	Fashion-MNIST	Random Forrest Surrogate	ResNet Surrogate	1000	4.249×10^{-17}
	Fashion-MNIST	Random Forrest Surrogate	ResNet Train Ensemble	1000	4.254×10^{-10}
6 a	CIFAR-100	ResNet Train Ensemble	I.I.D. Acquisition	500	3.010×10^{-11}
	CIFAR-10	ResNet Train Ensemble	I.I.D. Acquisition	500	3.988×10^{-76}
	Fashion-MNIST	ResNet Train Ensemble	I.I.D. Acquisition	500	2.457×10^{-105}
7	Fashion-MNIST	Predictive Entropy	I.I.D. Acquisition	400	2.801×10^{-6}
	Fashion-MNIST	Predictive Entropy	Mutual Information	400	8.779×10^{-4}

C.2. Figures 1 to 3: Synthetic Data

We now give details for Fig. 3 of the main paper, shown again in Fig. A.1 (columns 1, 4, 5), as well as the additional experiments (columns 3 and 4). Figures 1 and 2 use the setup of Fig. A.1 (column 1).

Regression. For the Gaussian process data (column 1), we sample $N = 50$ points from a Gaussian process with Matern-Kernel $\nu = 3/2$ and $l = 1$, using the implementation of Pedregosa et al. (2011). For each of the experiments, we sample a different set of points from the Gaussian process prior. The quadratic data (columns 2, 4) are drawn from $f(x) = y^2$. The sinusoidal data (column 3) are drawn from $f(x) = \sin(10x) + x^3$, where the density of the samples in x is non-uniform.

5 points are randomly assigned to the training set, 45 are used for testing. The test-train assignments are randomly redrawn between experiments. For each experiment, we actively evaluate the data with all acquisition functions. We perform a series of 5000 independent experiments. All regression experiments use a Gaussian process surrogate that is retrained after each acquisition on all observed data and has Matern-Kernel $\nu = 3/2$ and $l = 1$.

Two Moons Classification. For the two moons dataset (column 5), we randomly sample $N = 500$ points with noise level of 0.1 for each experiment, using the implementation of Pedregosa et al. (2011). 50 points are randomly assigned to the training set, 450 are used for testing. We perform a series of 2500 independent experiments. We use default parameters and implementation of Pedregosa et al. (2011) for the random forest. The surrogate model uses an identical random forest that is retrained after each acquisition on all observed data.

C.3. Figure 4: Radial BNN/ResNet-18 on MNIST/Fashion-MNIST

Data. For each experiment, we sample 250 training and 5000 test points randomly from the combined training and test set of the original datasets. We standardize the dataset using per-channel mean and standard deviation values over all pixels of the training set. We actively acquire 1000 samples from the test set and compute the 'full test loss' on all 5000 test points. We perform stratified sampling for both the train/test as well as the train/val splits. We retrain the surrogates after acquiring $(0, 5, 10, 20, 30, 40, 100, 250, 400, 550, 700, 850, 1000)$ test points.

Radial BNN on MNIST. We use the code obtained from Farquhar et al. (2021) to implement the Radial BNNs. We use the following hyperparameters, which are default values taken from Farquhar et al. (2021): we use a learning rate of 1×10^{-4} and weight decay of 1×10^{-4} with the ADAM optimizer (Kingma & Ba, 2014), batch size of 64, 8 variational samples during training for the BNN, 100 variational samples during testing, and convolutions with 16 channels. We train for a maximum of 500 epochs with early stopping patience of 5 and use validation sets of size 50. We have not tuned these hyperparameters.

ResNet-18 on Fashion-MNIST. For Resnet-18, we use the default hyperparameter values introduced by DeVries & Taylor (2017): we use a learning rate of 0.1, weight decay of 5×10^{-4}, and momentum of 0.9 with an SGD optimizer, and batch

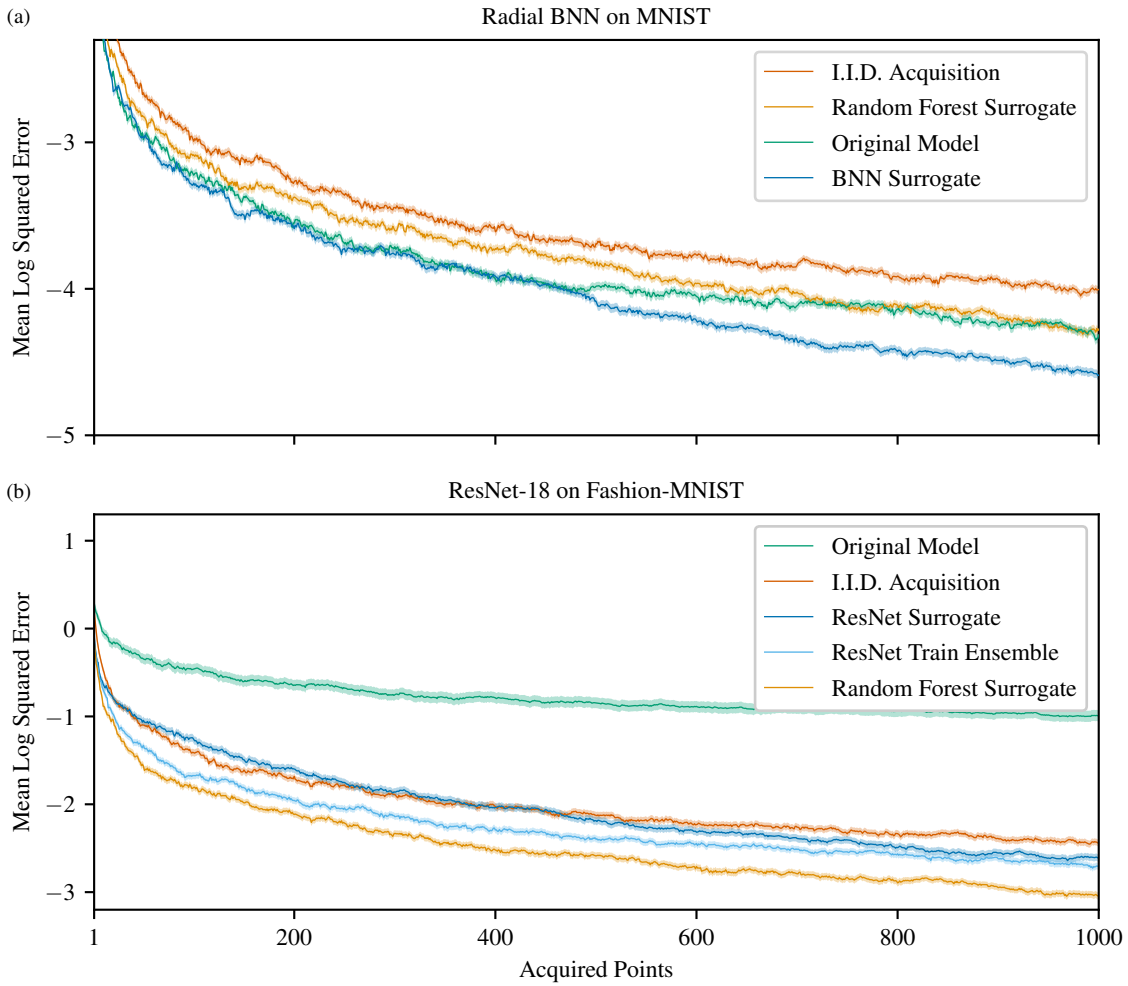

Figure A.3. Figure 4 from the main paper but now showing the mean of the log squared difference instead of the median. Additionally, shading indicates the standard error of the log squared difference. Averages over 1085 runs for (a), 872 for (b). See text and main paper for details.

size of 128. We use a cosine annealing schedule for the learning rate as provided by PyTorch. We train for a maximum of 160 epochs with early stopping patience of 20 and use validation sets of size 50. We have not tuned these hyperparameters.

Random Forest Surrogate. We use random forests with 100 estimators, split criterion 'entropy', and maximum number of features considered at each split proportional to the square root, otherwise using the default parameters and implementation of Pedregosa et al. (2011). Hyperparameters were set with a single grid-search cross-validation on one particular training set.

Training Convergence: Radial BNN on MNIST. Training of the BNN takes about 1 to 2 minutes and the early stop patience usually terminates training after around 50 to 100 epochs. A single experiment run for the Radial BNN, requiring the training of the original model and retraining of all shown surrogates, takes about 30 minutes. The validation accuracy on the initial 250 points is at about 80 to 90 percent and grows to about 90 to 98 percent after observing a total of 1250 points. We perform a total of 1085 runs.

Training Convergence: ResNet-18 on Fashion-MNIST. Training of the ResNet takes about 10 to 40 seconds and the early stop patience sets in around 30 to 80 epochs. A single run for the ResNet, requiring the training of the original model and retraining of all shown surrogates, takes about 10 minutes. The training validation accuracy on the initial 250 points is at about 68 to 78 percent and grows to about 80 to 86 percent after observing a total of 1250 points. We perform a total of 872 runs.

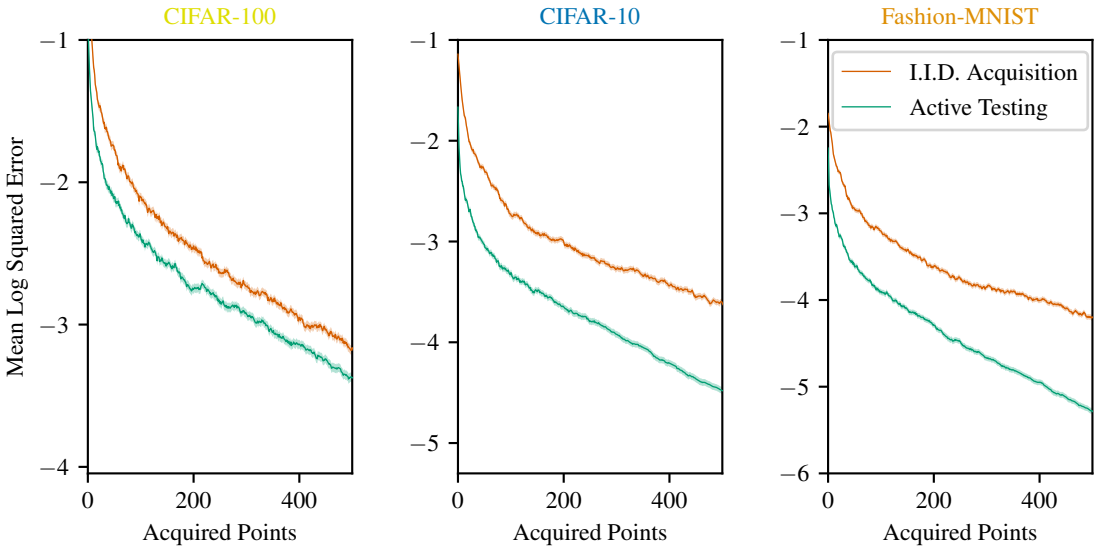

Figure A.4. Figure 6 (a) from the main paper but now showing the mean of the log squared difference instead of the median. Additionally, shading indicates the standard error of the log squared difference. Averages over 1000 random test set draws. See text and main paper for details.

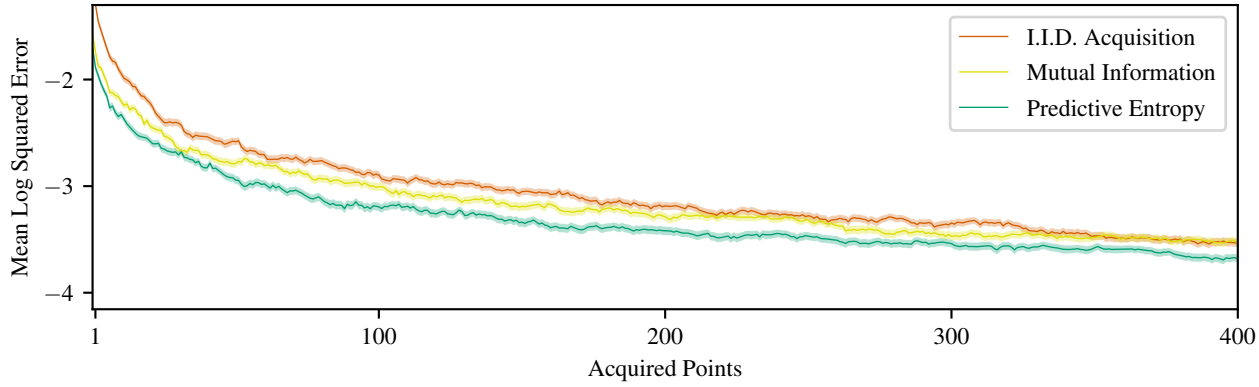

Figure A.5. Figure 8 (b) from the main paper but now showing the mean of the log squared difference instead of the median. Additionally, shading indicates the standard error of the log squared difference. Averages over 692 runs. See text and main paper for details.

C.4. Figure 5: ResNet-18 on CIFAR-100

The setup for Fig. 5 is a ResNet-18 on CIFAR-100 identical to the experiments from Fig. 6 described in the following. The model converges to 72 to 76 percent accuracy in about 12 minutes and we perform 692 runs.

C.5. Figure 6: ResNet-18/WideResNet on Fashion-MNIST, CIFAR-10/Cifar-100

Data. We now respect the original train and test indices of the dataset. We train a model on the training set and then, for each experiment, perform active testing on a subset of 1000 randomly sampled points from the original test set of size 10 000. We standardize the dataset using per-channel mean and standard deviation values over all pixels of the training set. For CIFAR-10 and CIFAR-100, we apply random crops and horizontal flips to the training data in each epoch. Given the larger training data, we now use validation sets of size 2560. We re-sample test sets for the next experiment, but keep training sets and models constant. We perform stratified sampling for both the train/test as well as the train/val splits. This, we repeat a total of 1000 times.

ResNet-18. Unless otherwise mentioned we use identical hyperparameters as for Fig. 4. On all datasets, we train for a maximum of 30 epochs with patience of 5. Again, we do not systematically tune hyperparameters and only make

adjustments to accommodate the increased data size compared to Fig. 4. The model converges to 93 to 94 percent accuracy on Fashion-MNIST in about 7 minutes and 92 to 93 percent accuracy on CIFAR-10 in about 12 minutes.

WideResNet. We use a WideResNet of depth 40 and the setup introduced by DeVries & Taylor (2017): we train for 200 epochs, use a learning rate of 0.1, weight decay of 5×10^{-4}, and momentum of 0.9 with an SGD optimizer, and batch size of 128. We also use the scheduler of DeVries & Taylor (2017) and decrease the learning rate by a factor of $\gamma = 0.2$ at epochs 60, 120, and 160. The model converges to 78 to 80 percent accuracy on CIFAR-100 in about 240 minutes.

Ensembles. For the deep ensemble of Resnet-18s, we use 10 models for Fashion-MNIST, 5 for CIFAR-10, and 15 for CIFAR-100. For the deep ensemble of WideResnets on CIFAR-100, we use a set of 10 models. The models for the ensembles are identical to the main models, i.e. use the same hyperparameters, optimizers, and training setup, and only differ in terms of the optima reached from stochastic model initialization and optimization.

Increasing the size of the ensemble did sometimes yield improved results. Following initial experiments, we increased the size of the ensemble for Fashion-MNIST from 5 to 10. However, using ensemble size of 15 on CIFAR-10 did not improve the results noticeably (nor worsen them), so we display the smaller, and computationally cheaper ensemble in the main body.

C.6. Figure 7

Figure 7 uses the same setup as Fig. 6 for CIFAR-10 with main ResNet-18 model and 'train ensemble'-style surrogates.

C.7. Figure 8

For the Radial BNN, Fig. 8 uses identical experimental setup as Fig. 4, except that the model is trained for a maximum of 30 epochs (patience 5) and we use validation sets of size 1280, to account for the increase in data. For the Fashion-MNIST data, we concatenate the original train and test datasets, from which we then sample 50 000 points without replacement for the training dataset and 10 000 for the test dataset. We acquire a total of 1000 points from the test dataset but compute the 'full test set loss' on all 10 000 points. Splits are stratified by class labels. We redraw train and test splits, and retrain models between runs. For each experiment, the model reaches about 88 to 92 percent validation accuracy in about 8 minutes. We perform a total of 962 runs.

C.8. Figure A2: Radial BNN on MNIST.

For the Radial BNN, Fig. A.2 uses identical experimental setup as Fig. 4 (a), except that the model is trained for a maximum of 50 epochs (patience 5) and we use validation sets of size 1280 to account for the increase in data. For the MNIST data, the setup is identical to Fig. 8. The model reaches about 98 percent validation accuracy in about 7 to 10 minutes. We perform a total of 992 runs.

D. Comparison to Active Risk Estimation by Sawade et al.

We here provide a brief introduction to 'Active Risk Estimation' (ARE) by Sawade et al. (2010) as well as an empirical comparison of their approach to ours. Similar to us, Sawade et al. (2010) actively acquire labels from a test pool of unlabeled samples, aiming to obtain a sample-efficient estimate of the empirical test risk. However, unlike us, they do not fully accommodate the pool-based setting. Additionally, Sawade et al. (2010) rely only on the original model to approximate outcomes $y \mid \mathbf{x}$, while we have shown that appropriate surrogate models can be crucial for sample-efficient active testing. Moreover, as we show below, active testing outperforms ARE in practice.

D.1. Background

Sawade et al. (2010) derive an 'optimal acquisition function'

$$q^*(\mathbf{x}) \propto p(\mathbf{x}) \sqrt{\int \left[\mathcal{L}(f_\theta(\mathbf{x}), y) - R\right]^2 p(y \mid \mathbf{x}) \, \mathrm{d}y}, \tag{15}$$

where $R = \mathbb{E}\left[\mathcal{L}(f(\mathbf{x}), y)\right]$ is the true model risk. For the pool-based setting, they obtain an equivalent 'optimal acquisition function' by setting $p(\mathbf{x}_{i_m}) = \frac{1}{M}$ and approximating R with the empirical test risk $\hat{R}_{\text{iid}}$ over the entire test pool. As $\hat{R}_{\text{iid}}$ is still unknown, Sawade et al. (2010) approximate $\hat{R}_{\text{iid}}$ using the original model's mean predictions $f_\theta(\mathbf{x})$ to approximate the

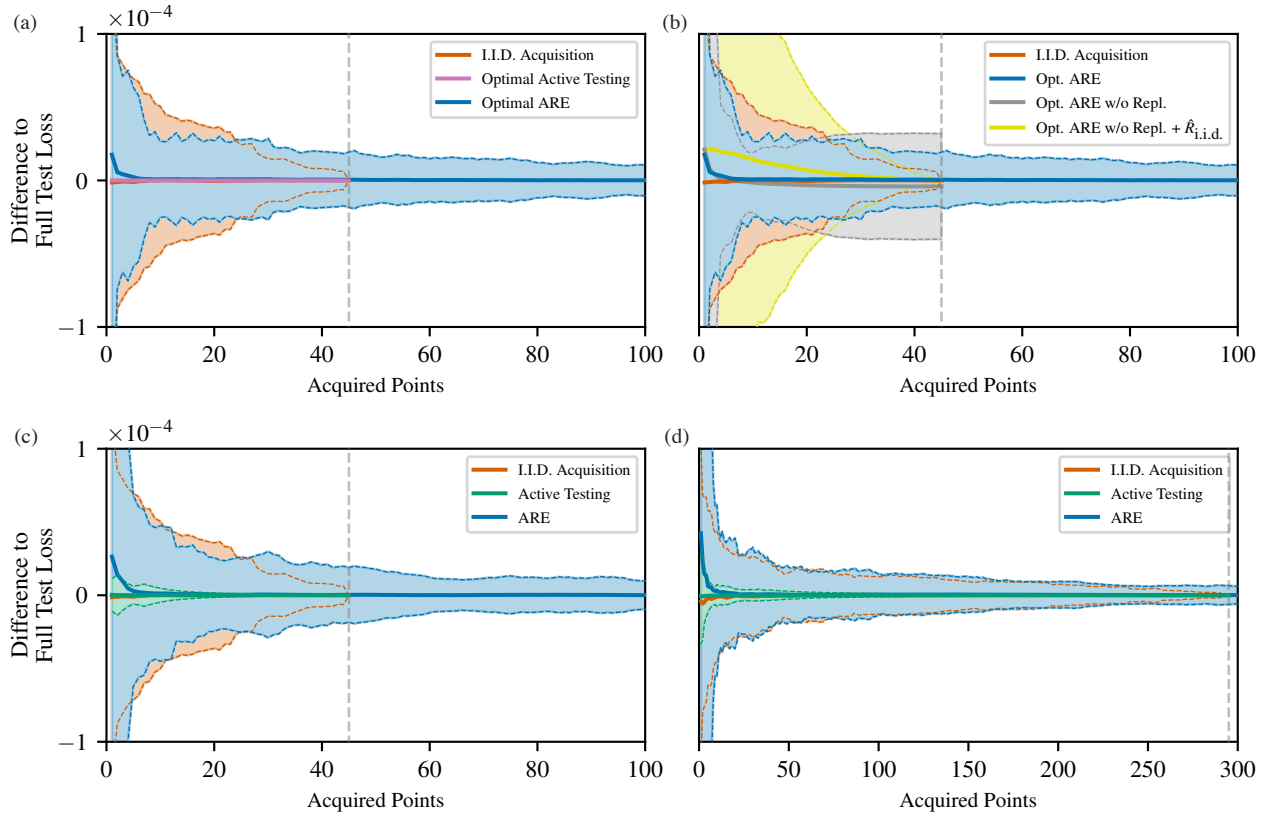

Figure D.1. Comparison of Active Testing to Active Risk Estimation (ARE) by Sawade et al. (2010). (a) While active testing can yield single-sample zero-variance optimal estimates, ARE cannot because it does not fully accommodate the pool-based setting. (b) Naive extensions of ARE to sampling without replacement are unsuccessful. (c) In practice, ARE yields lower performance than active testing. (d) As test set grows, the gap between ARE and active testing will get smaller as sampling with replacement matters less. Dashed grey line demarcates number of samples in the test pool. Solid lines show means, dashed lines standard deviations over 7000 (a–c) / 2000 (d) runs.

true outcomes, giving

$$\hat{R}_{\text{i.i.d.},\theta} = \frac{1}{M} \sum_{\mathbf{x} \in \mathcal{D}_{\text{test}}} \int \mathcal{L}(f_\theta(\mathbf{x}), \mathbf{y}) p(\mathbf{y}|\mathbf{x}; \theta) \, \mathrm{d}\mathbf{y} \tag{16}$$

$$q^*(\mathbf{x}) \propto \sqrt{\int \left[\mathcal{L}(f_\theta(\mathbf{x}), y) - \hat{R}_{\text{i.i.d.},\theta} \right]^2 p(y \mid \mathbf{x}; \theta) \, \mathrm{d}y}. \tag{17}$$

They do not consider the concept of a surrogate model.

Plugging mean squared error loss and Gaussian likelihoods into (17), they derive the acquisition function

$$q(\mathbf{x}) \propto \sqrt{3\sigma_{\mathbf{x}}^4 - 2\hat{R}_{\text{i.i.d.},\theta}\sigma_{\mathbf{x}}^2 + \hat{R}_{\text{i.i.d.},\theta}^2}, \tag{18}$$

where $\sigma_{\mathbf{x}}^2$ is the total predictive variance (aleatoric and epistemic uncertainty) of the original model's prediction at $\mathbf{x}$.

For risk estimation, ARE relies on a standard importance sampling estimator

$$\hat{R}_{\text{IS}} = \frac{1}{\sum_{m=1}^{M} q(\mathbf{x}_{i_m})} \sum_{m=1}^{M} \frac{1}{q(\mathbf{x}_{i_m})} \mathcal{L}(f_\theta(\mathbf{x}_{i_m}, y_{i_m})), \tag{19}$$

where the uniform $p(\mathbf{x}_{i_m})$ cancels from the weights. Unlike $\hat{R}_{\text{LURE}}$, $\hat{R}_{\text{IS}}$ requires sampling *with replacement* to obtain unbiased estimates. In other words, ARE does not remove samples from the test pool after querying them, and often, test points will get sampled repeatedly.

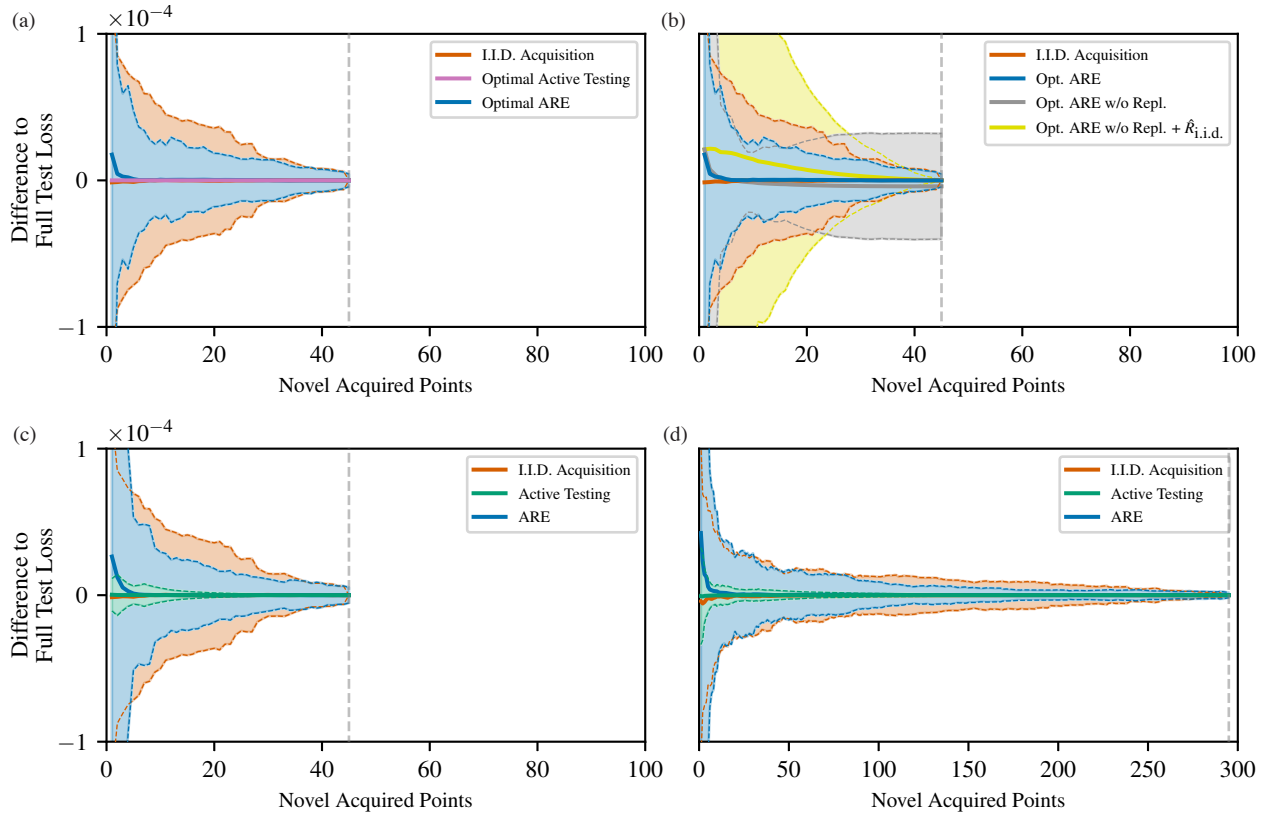

Figure D.2. Comparison of Active Testing and Active Risk Estimation. Identical to Fig. D.1, except that now, acquisition steps for ARE are only counted if ARE queries lead to novel acquisitions. Displaying results this way slightly improves ARE's performance but does not change our conclusions.

D.2. Empirical Comparison.

We compare both theoretically optimal behavior as well as observed practical performance of our active testing and ARE. For this, we take the familiar setup from Fig. A.1 (column 1).

Optimal Behavior. We have seen that active testing can ideally lead to single-sample, zero variance estimates of the test loss (cf. Fig. 8 (a)). What is the optimal behavior ARE can obtain? To test this, we apply (15) with the oracle empirical test risk $\hat{R}_{\text{iid}}$ and the true $y \mid \mathbf{x}$ (which is a δ-distribution since we generate data without noise). This constitutes the best case scenario for ARE performance.

As Fig. D.1 (a) shows, 'optimal ARE' can improve over i.i.d. acquisition but does not show the same desirable single-sample optimal behavior as active testing. Note that ARE extends beyond the test set size of 45 because it does not naturally converge due to sampling with replacement. In Fig. D.1 (b) we confirm that naive extensions of ARE to sampling *without replacement* are not successful: (i) Simply using ARE but sampling without replacement does not work; (ii) When additionally using $\hat{R}_{\text{iid}}$ instead of $\hat{R}_{\text{IS}}$, we converge to the empirical test risk but observe a bias and increased variance over i.i.d. acquisition at earlier acquisition steps.

Performance in Practice. We now investigate behavior of ARE in practice—where the oracle risk is not available and we rely on (17) instead. Figure D.1 (c) shows that while ARE can increase sample-efficiency over i.i.d. acquisition in practice, it is clearly outperformed by active testing. For Fig. D.1 (d), we increase the size of the test set from 45 to 295. We suspect that further increases in test set size will shrink the gap between ARE and active testing without surrogates, because sampling with replacement should become less important. However, still, active testing clearly outperforms ARE: active testing also varies substantially in the acquisition strategy it uses (in particular allowing the use of surrogates), with these result suggesting its approach is clearly preferable.

Redefining Acquisitions Steps. Sampling with replacement is a sub-optimal strategy for methods applied in pool-based

settings. As mentioned above, sampling with replacement in ARE leads to the same samples being acquired multiple times. For maximum fairness, we now only increase the 'acquisition step' counter by one if the sampling with replacement process leads to an acquisition that has not been previously made, i.e. if the acquisition is *novel*. This is somewhat justified in practice because we assume that acquiring novel labels is much more expensive than acquisition function evaluations.

However, find that ARE leads to a significant increase in queries compared to active testing[3] and still performs far worse even by this beneficial metric. Namely, we display the results with 'rescaled x-axis' in Fig. D.2. While the performance of ARE appears slightly improved, our overall conclusions are unaffected.

[3]For Fig. D.1 (c), ARE takes (9.8 ± 3.1) as many queries as there are samples in the test pool. This number will increase for larger test pools and shows that sampling with replacement is not a desirable strategy for pool-based active model evaulation.