On the Role of Model Uncertainties in Bayesian Optimization (Supplementary Material)

Jonathan Foldager[1,*] **Mikkel Jordahn**[1,*] **Lars Kai Hansen**[1] **Michael Riis Andersen**[1]

[1]Department of Applied Mathematics and Computer Science, Technical University of Denmark
[*]Shared first authorship.

1 HYPERPARAMETER TUNING DATASETS

When collecting our hyperparameter tuning datasets, the combinations of models and datasets are as follows:

Table 1: Model and Data Combinations for Hyperparameter Tuning

	MNIST	FashionMNIST	AG News Classification	Wine Classification
FFNN	✓	✓	✓	
CNN	✓	✓		
SVM				✓

For each of the models we then select a number of hyperparameters which we want to tune, create a grid for these hyperparameters and train a model for each of these hyperparameter sets (the BO input is thus hyperparameters and the output is validation performance). The FFNN simply has a single hidden layer with a ReLU activation function and a single dropout layer, except in the case of the AG News Classification where the "hidden layer" is an embedding layer using the `nn.EmbeddingBag` from `torch` [Paszke et al., 2019]. The CNN is a network with two convolution layers with kernel size $(5, 5)$ of output channels 16 and 32 respectively, and a single hidden and dropout layer. Max pooling is also used with a kernel size of $(2, 2)$ at every convolution layer. The SVM used is the SVC from `sklearn` [Pedregosa et al., 2011]. The hyperparameters and their grid specification can be seen here:

Table 2: Grid Specifications for Hyperparameter Tuning

	Training Epochs	**Dropout Rate**	**Learning Rate (log space)**	**Batch Size Train**	**Hidden Size**	**C (log space)**	**γ (log space)**
FFNN	np.linspace(1, 10, 10)	np.linspace(0, 0.8, 10)	np.linspace(-11.51, -2.23, 10)	np.arange(8, 256, 32)	np.linspace(1, 271, 10)		
CNN	np.linspace(1, 10, 10)	np.linspace(0, 0.8, 10)	np.linspace(-11.51, -2.23, 10)	np.arange(8, 256, 32)	np.linspace(1, 271, 10)		
SVM						np.linspace(-6.9, 4.6, 100)	np.linspace(-11.51, -2.23, 100)

2 EXPERIMENTAL DETAILS

Model details are as following. The GPs are built using GPytorch [Gardner et al., 2018] and BoTorch [Balandat et al., 2020] (BoTorch's SingleTaskGP class) and use a scale kernel and RBF kernel. The priors can be seen in Table 3. The hyperparameters of the kernel are tuned at every BO iteration using the marginal likelihood using the scipy L-BFGS-B optimizer (default settings in BoTorch). The Deep Ensembles consist of 10 neural networks with two hidden layers of size 30 and 10 respectively and use the ReLU activation function. We use an Adam optimiser with learning rate $4e^{-3}$. We train them for 200 epochs. They are implemented using torch. The BNN Small has a single hidden layer of size 10 whilst the larger BNN model has two hidden layers of size 30 and 10 respectively and are implemented using the BayesLinear layers from torchbnn [Lee et al., 2022]. We use a KL weight of 1, and use an Adam optimiser with a learning rate of 0.1. They are

Accepted for the 39th Conference on Uncertainty in Artificial Intelligence (UAI 2023).

trained for 500 epochs. The priors of these layers can be seen in Table 4. The RFs are implemented using sklearn and the hyperparameter grids we tune over are: n_estimators= $[4, 10, 20]$, max_depth=$[5, 10, 20]$ and max_features=$[1.0$, "sqrt"$]$.

Table 3: GP Priors

	Lengthscale Prior	**Outputscale Prior**
Synthetic Problems	LogNormalPrior(0.1, 1.0)	NormalPrior(1.0, 2.0)
Real Data Problems	LogNormalPrior(0.1, 5.0)	NormalPrior(1.0, 5.0)

Table 4: BNN Priors

	Prior Mean	**Prior Sigma**
Input Layer	0.0	1.0
Hidden Layer	0.0	1/900
Output Layer	0.0	1/100

We use BoTorch to perform our Bayesian Optimisation. We use their UpperConfidenceBound and ExpectedImprovement classes for UCB (with beta=1. We experimented briefly with other beta values. These results can be seen in table 5) and EI AFs respectively, and have adapted their MaxPosteriorSampling class for use as a TS AF. Due to computational reasons, we perform BO by sampling a candidate pool set of size ($N_{pool} = 5000$) at the beginning of each BO iteration that the surrogate and acquisition function can choose to sample from, rather than allowing the surrogates to sample from anywhere in the input space. Please note that $D_{test} \cap D_{pool} = \emptyset$ in the real data experiment setting where the input space is not continous, whilst this is not necesarily the case in the synthetic data experiments as the input space is continous here, and thus we allow random sampling for both D_{test} and D_{pool}. The full experimental procedure is written in pseudocode in Algorithm 1. F_S here denotes acquisition function evaluated based on surrogate model S. Please note that we invert the ys for the real data problems to make it a minimization problem (we want to optimize model accuracy).

Table 5: Experimental results when tuning beta of UCB

Surrogate	**Beta**	**Inst. Regret**	**Total Regret**
BNN Small	0.2	0.006	1.95
BNN Small	0.5	0.017	3.59
BNN Small	1	0.018	4.03
BNN Small	2	0.040	4.76
DE	0.2	0.003	1.05
DE	0.5	0.001	0.91
DE	1	0.000	0.84
DE	2	0.001	1.01
GP	0.2	0.003	1.75
GP	0.5	0.002	1.47
GP	1	0.000	1.33
GP	2	0.001	1.38
RF	0.2	0.003	0.97
RF	0.5	0.003	0.94
RF	1	0.002	0.78
RF	2	0.004	0.97

Algorithm 1 Bayesian Optimisation Experiments

Require: Surrogate Model: S, Acquisition Function: F, BO Problem: P, $N_{test} = 5000$, $N_{pool} = 5000$, $N_{init} = 10$, $i = 90$

if P is synthetic **then**
  $D_{test} \leftarrow N_{test}$random points from$P$
  $D_{pool} \leftarrow N_{pool}$ random points from P
  Standardize data s.t. mean=0, var=1 using D_{pool} metrics.
  $x_{opt}, y_{opt} \leftarrow \min D_{pool}$
  $D_{train} \leftarrow N_{init}$ random points from D_{pool}
  $D_{pool} \leftarrow D_{pool}$ - D_{train}
else if P is real data **then**
  $D_{problem} \leftarrow$ all points from P
  $D_{test} \leftarrow N_{test}$ random points from $D_{problem}$
  $D_{problem} \leftarrow D_{problem}$ - D_{test}
  $D_{pool} \leftarrow N_{pool}$ random points from $D_{problem}$
  Standardize data s.t. mean=0, var=1 using D_{pool} metrics.
  $x_{opt}, y_{opt} \leftarrow \min D_{pool}$
  $D_{train} \leftarrow N_{init}$ random points from D_{pool}
  $D_{pool} \leftarrow D_{pool}$ - D_{train}
end if
while $i > 0$ **do**
  Fit S to D_{train}
  $D_{next} \leftarrow \max F_S(D_{pool})$
  $D_{train} \leftarrow D_{train} + D_{next}$
  $D_{pool} \leftarrow D_{pool} - D_{next}$
  Fit S to D_{train}
  $y_{best} = \min D_{train}$
  Calculate regret: $y_{opt} - y_{best}$
  Calculate ECE of model based on D_{test}
  $i \leftarrow i - 1$
end while

3 MULTIPLE REGRESSION ANALYSIS

In Table 6 and Table 7 the multiple regression analysis can be seen for the real and synthetic data respectively. The regression was done using Statsmodels for Python [Seabold and Perktold, 2010].

Table 6: Multiple regression analysis for hyperparameter tuning experiments. GP is the baseline model and MNIST is the baseline dataset. The other slopes and intercepts are contrasts to these two baselines.

	coef	**std err**	**t**	**P$>$ |t|**	**[0.025**	**0.975]**
calibration_mse	8.9899	88.549	0.102	0.919	-167.409	185.389
BNN	-56.6270	89.992	-0.629	0.531	-235.899	122.646
DE	-48.5141	92.738	-0.523	0.602	-233.257	136.229
RF	-95.7785	113.183	-0.846	0.400	-321.251	129.694
BNN Small	-59.3964	89.109	-0.667	0.507	-236.911	118.118
intercept	1.7675	1.282	1.379	0.172	-0.787	4.322
DE_intercept	0.9198	1.676	0.549	0.585	-2.419	4.259
BNN_intercept	6.6811	2.856	2.339	0.022	0.992	12.370
RF_intercept	0.8637	1.615	0.535	0.594	-2.353	4.081
BNN_Small_Intercept	7.2043	2.353	3.062	0.003	2.518	11.891
fashionmnist	1.4489	0.364	3.981	0.000	0.724	2.174
mnist_cnn	-1.2256	0.460	-2.665	0.009	-2.142	-0.310
fashionmnist_cnn	0.9344	0.413	2.261	0.027	0.111	1.758
news	-0.2718	0.406	-0.669	0.505	-1.081	0.537
svm_wine	-2.6434	0.390	-6.775	0.000	-3.421	-1.866

Table 7: Multiple regression analysis for synthetic optimisation experiments. GP is the baseline model and Problem18 is the baseline dataset. The other slopes and intercepts are contrasts to these two baselines.

	coef	std err	t	P> |t|	[0.025	0.975]
calibration_mse	-2477.1182	596.466	-4.153	0.000	-3649.745	-1304.492
BNN	2049.7326	576.865	3.553	0.000	915.640	3183.825
DE	2343.7352	582.393	4.024	0.000	1198.776	3488.695
RF	1124.1523	726.204	1.548	0.122	-303.534	2551.839
BNN Small	2155.4162	578.996	3.723	0.000	1017.134	3293.698
intercept	24.4450	11.213	2.180	0.030	2.400	46.490
DE_intercept	-22.5397	8.992	-2.507	0.013	-40.217	-4.862
BNN_intercept	59.5739	17.698	3.366	0.001	24.781	94.367
RF_intercept	11.6801	11.736	0.995	0.320	-11.393	34.754
BNN Small_intercept	50.0913	13.164	3.805	0.000	24.211	75.972
MegaDomain02	-8.3801	11.598	-0.723	0.470	-31.181	14.420
Ackley	238.4477	11.751	20.292	0.000	215.346	261.549
Schwefel22	15.1786	12.368	1.227	0.220	-9.137	39.494
Problem15	-9.0384	11.668	-0.775	0.439	-31.978	13.901
Sargan	5.3031	11.621	0.456	0.648	-17.544	28.150
Quadratic	0.2745	11.537	0.024	0.981	-22.407	22.956
BartelsConn	-3.1293	11.573	-0.270	0.787	-25.881	19.622
McCourt27	68.7446	11.672	5.890	0.000	45.799	91.691
Sphere	11.3725	11.622	0.978	0.328	-11.477	34.222
Ursem04	51.9448	11.613	4.473	0.000	29.115	74.775
Plateau	36.7332	11.642	3.155	0.002	13.846	59.620
MegaDomain04	3.3478	11.668	0.287	0.774	-19.591	26.286
Problem13	-3.7408	11.551	-0.324	0.746	-26.449	18.967
SumPowers	4.3195	11.583	0.373	0.709	-18.452	27.091
MegaDomain03	-6.3915	11.623	-0.550	0.583	-29.242	16.459
Brown	24.5035	13.673	1.792	0.074	-2.378	51.385
Cigar	26.1229	11.576	2.257	0.025	3.366	48.880
Schwefel06	4.3806	11.667	0.375	0.708	-18.556	27.318
McCourt28	-0.2202	11.688	-0.019	0.985	-23.197	22.757
Step	38.1989	11.571	3.301	0.001	15.450	60.948
HimmelBlau	2.2361	11.524	0.194	0.846	-20.419	24.892
Problem18	-3.1796	11.658	-0.273	0.785	-26.098	19.739
Giunta	13.5342	11.684	1.158	0.247	-9.436	36.504
Csendes	1.7708	11.689	0.151	0.880	-21.209	24.750
Exponential	14.8954	11.656	1.278	0.202	-8.020	37.811
Problem04	-5.2547	11.595	-0.453	0.651	-28.050	17.541
Schwefel20	52.3473	11.663	4.488	0.000	29.419	75.276
Schwefel01	-1.1381	11.568	-0.098	0.922	-23.880	21.604

4 MATHEMATICAL PROOFS

Proposition 1: Let F_i be the CDF of the predictive distribution for the i'th observation and let $\{y_i\}_{i=1}^n$ be i.i.d. samples $y_i \sim p_y$. For $\mathcal{C}_y(p) = \frac{1}{n}\sum_{i=1}^n \mathbb{I}\left[y_i \leq F_i^{-1}(p)\right]$, then the variance of $C_y(p)$ is bounded by $1/n$, i.e. $\mathbb{V}[C] = \mathcal{O}(n^{-1})$.

Proof: First, we show that the variance is bounded by $\mathcal{O}(n^{-1})$. We have

$$\mathcal{C}_y(p) = \frac{1}{n}\sum_{i=1}^n \mathbb{I}\left[y_i \leq F_i^{-1}(p)\right] = \frac{1}{n}\sum_{i=1}^n z_i, \tag{1}$$

where $z_i \equiv \mathbb{I}\left[y_i \leq F_i^{-1}(p)\right]$. The variance of $\mathcal{C}_y(p)$ is then by give

$$\begin{aligned}
\mathbb{V}\left[\mathcal{C}_y(p)\right] &= \mathbb{V}\left[\frac{1}{n}\sum_{i=1}^n z_i\right] \\
&= \frac{1}{n^2}\mathbb{V}\left[\sum_{i=1}^n z_i\right] \\
&= \frac{1}{n^2}\sum_{i=1}^n \mathbb{V}\left[z_i\right] \\
&\leq \frac{1}{n^2}\sum_{i=1}^n \sup_i \mathbb{V}\left[z_i\right] \\
&\leq \frac{1}{n^2}\sum_{i=1}^n \frac{1}{2^2} \\
&= \frac{1}{n}\frac{1}{2^2}
\end{aligned} \tag{2}$$

Hence, it also follows the standard deviation of $\mathcal{C}_y(p)$ is bounded by

$$\sqrt{\mathcal{C}_y(p)} \leq \sqrt{\frac{1}{n}\frac{1}{2^2}} = \frac{1}{2\sqrt{n}} = \mathcal{O}\left(\frac{1}{\sqrt{n}}\right). \tag{3}$$

This completes the proof of the first statement.

Lemma 1: Given a perfectly calibrated model, it holds that $\mathbb{V}\left[\mathcal{C}_y(p)\right] = \frac{p(1-p)}{n}$ for all p.

Proof: In this setting, we have

$$z_i = \mathbb{I}\left[y_i \leq F_i^{-1}(p)\right] = \mathbb{I}\left[F_i(y_i) \leq p\right] = \mathbb{I}\left[u_i\right] \leq p\right], \tag{4}$$

where $u_i \sim \mathcal{U}\left[0, 1\right]$ are uniformly distributed on the unit interval due to the probability integral transform. Since $\{u_i\}_{i=1}^n$ are also independent, it follows that

$$S_n = \sum_{i=1}^n z_i \sim \text{Binomial}(n, p). \tag{5}$$

Therefore, it follows that

$$\begin{aligned}
\mathbb{V}\left[\mathcal{C}_y(p)\right] &= \mathbb{V}\left[\frac{1}{n}S\right] = \frac{1}{n^2}\mathbb{V}\left[S\right] \\
&= \frac{1}{n^2}np(1-p) = \frac{p(1-p)}{n}.
\end{aligned} \tag{6}$$

This completes the proof.

Proposition 2: Let $E_c = \sum_{j=1}^{m} w_j (p_j - \mathcal{C}_y(p_j))^2$ be the weighted mean square calibration error. Assume $w_i \in [0, 1]$ and $0 < p_1 < p_2 < ... < p_m < 1$ are fixed, and assume the CDF of the predictive distribution is equal to the true data distribution (almost everywhere), then it holds that $\mathbb{E}[E_c] = \frac{1}{n} \sum_{j=1}^{m} w_j p_j (1 - p_j) = \mathcal{O}(n^{-1})$ if $y_i \sim p_y$ are i.i.d. samples.

The calibration error E_C is defined as follows

$$E_c = \sum_{j=1}^{m} w_j (p_j - \mathcal{C}_y(p_j))^2, \tag{7}$$

where each $w_i \in [0, 1]$ is a weight and $0 \leq p_1 < p_2 < ... < p_m < 1$ is predefined set of points.

In order to compute the expectation of E_C, we first expand:

$$E = \sum_{j=1}^{m} w_j (p_j^2 + \mathcal{C}_y(p_j)^2 - 2 p_j \mathcal{C}_y(p_j)) \tag{8}$$

$$= \sum_{j=1}^{m} w_j \mathcal{C}_y(p_j)^2 - 2 \sum_{j=1}^{m} w_j p_j \mathcal{C}_y(p_j)) \tag{9}$$

Then it follows that

$$\mathbb{E}_{\mathbb{C}}[E] = \mathbb{E}\left[\sum_{j=1}^{m} w_j p_j^2 + \sum_{j=1}^{m} w_j \mathcal{C}_y(p_j)^2 - 2 \sum_{j=1}^{m} w_j p_j \mathcal{C}_y(p_j)) \right] \tag{10}$$

$$= \sum_{j=1}^{m} w_j p_j^2 + \sum_{j=1}^{m} w_j \mathbb{E}\left[\mathcal{C}_y(p_j)^2 \right] - 2 \sum_{j=1}^{m} w_j p_j \mathbb{E}\left[\mathcal{C}_y(p_j) \right]. \tag{11}$$

The first moment evaluates to

$$\mathbb{E}[C_y(p)] = \int_{-\infty}^{\infty} \mathbb{I}[y_t \leq F_t^{-1}(p)] p_y \mathrm{d}y \tag{12}$$

$$= \int_{-\infty}^{F_t^{-1}(p)} p_y \mathrm{d}y \tag{13}$$

$$= F_y(F_t^{-1}(p)) \tag{14}$$

$$= p. \tag{15}$$

Similarly, the second moment evaluates to

$$\mathbb{E}\left[\mathcal{C}_y(p)^2 \right] = \mathbb{E}\left[\left(\frac{1}{n} \sum_{i=1}^{n} z_i \right)^2 \right] \tag{16}$$

$$= \frac{1}{n^2} \mathbb{E}\left[\sum_{i=1}^{n} \sum_{j=1}^{n} z_i z_j \right] \tag{17}$$

$$= \frac{1}{n^2} \sum_{i=1}^{n} \mathbb{E}\left[z_i^2 \right] + \frac{1}{n^2} \sum_{j \neq i} \mathbb{E}\left[z_i z_j \right] \tag{18}$$

$$= \frac{1}{n^2} \sum_{i=1}^{n} p + \frac{1}{n^2} \sum_{j \neq i} \mathbb{E}\left[z_i \right] \mathbb{E}\left[z_j \right] \tag{19}$$

$$= \frac{n}{n^2} p + \frac{1}{n^2} \sum_{j \neq i} p^2 \tag{20}$$

$$= \frac{1}{n} p + \frac{1}{n^2} \left(n^2 - n \right) p^2 \tag{21}$$

Rearranging the terms yields

$$\begin{aligned}\mathbb{E}\left[\mathcal{C}_y(p)^2\right] &= \frac{1}{n}p + \frac{n^2-n}{n^2}p^2 \\ &= \frac{1}{n}p - \frac{1}{n}p^2 + p^2 \\ &= \frac{p(1-p)}{n} + p^2\end{aligned} \tag{22}$$

Substituting the moments into eq. (10) yields

$$\begin{aligned}\mathbb{E}\left[E_C\right] &= \sum_{j=1}^{m} w_j p_j^2 + \sum_{j=1}^{m} w_j \left[\frac{p_j(1-p_j)}{n} + p_j^2\right] \\ &\quad - 2\sum_{j=1}^{m} w_j p_j^2 \\ &= \sum_{j=1}^{m} w_j p_j^2 + \sum_{j=1}^{m} w_j \frac{p_j(1-p_j)}{n} \\ &\quad + \sum_{j=1}^{m} w_j p_j^2 - 2\sum_{j=1}^{m} w_j p_j^2 \\ &= \frac{1}{n}\sum_{j=1}^{m} w_j p_j(1-p_j) \\ &= \mathcal{O}(n^{-1}).\end{aligned} \tag{23}$$

This completes the proof.

IF p_y AND p_t ARE NORMAL DISTRIBUTIONS

For non-perfect models we have that $F_y(F_t^{-1}(p)) = g(p)$ where in general $g(p) \neq p$. If both p_y and p_t are normal distributions, the CDF and inverse CDF of a normal are, respectively, given by

$$\begin{aligned}F(x) &= \frac{1}{2}\left[1 + \text{erf}\left(\frac{x-\mu}{\sigma\sqrt{2}}\right)\right] \\ F^{-1}(p) &= \mu + \sigma\sqrt{2}\text{erf}^{-1}\left(2p-1\right)\end{aligned}$$

When data comes from $y_t \sim \mathcal{N}(\mu_y, \sigma_y^2)$ and the model is $\mathcal{N}(\mu_t, \sigma_t^2)$, we can write the expectation of the calibration curve as follows

$$\begin{aligned}g(p) &= F_y(F_t^{-1}(p)) \\ &= \frac{1}{2}\left[1 + \text{erf}\left(\frac{F_t^{-1}(p) - \mu_y}{\sigma_y\sqrt{2}}\right)\right] \\ &= \frac{1}{2}\left[1 + \text{erf}\left(\frac{\mu_t + \sigma_t\sqrt{2}\text{erf}^{-1}\left(2p-1\right) - \mu_y}{\sigma_y\sqrt{2}}\right)\right] \\ &= \frac{1}{2}\left[1 + \text{erf}\left(\frac{\mu_t - \mu_y}{\sigma_y\sqrt{2}} + \frac{\sigma_t}{\sigma_y}\text{erf}^{-1}\left(2p-1\right)\right)\right] \\ &= \frac{1}{2}\left[1 + \text{erf}\left(\frac{\mu_t - \mu_y}{\sigma_y\sqrt{2}} + \frac{\sigma_t}{\sigma_y}\text{erf}^{-1}\left(2p-1\right)\right)\right]\end{aligned}$$

which also evaluates to p for a perfect model:

$$\begin{aligned} g(p) &= \frac{1}{2}\left[1 + \text{erf}\left(\frac{0}{\sigma_y\sqrt{2}} + 1 \cdot \text{erf}^{-1}\left(2p-1\right)\right)\right] \\ &= \frac{1}{2}\left[1 + 2p - 1\right] \\ &= p \end{aligned}$$

5 VARIANCE OF CALIBRATION CURVES AS FUNCTION OF VALIDATION SET SIZE

We first illustrate empirically how accurately we can asses the calibration curve as a function of the size of the validation set N. This can be seen in Figure 1, where the true data generating distribution $p_y(y|x) = \mathcal{N}(y|0, 1)$ is approximated by six different model distributions $p_t(y|x)$ (one for each row). The first column shows the PDF and CDF of the true distribution and the model distribution in blue and black, respectively. Each of the subsequent columns shows the estimated calibration curves as a function of the number validation samples N. We repeat this experiment one hundred times and display the mean and confidence intervals corresponding to ± 2 standard deviations.

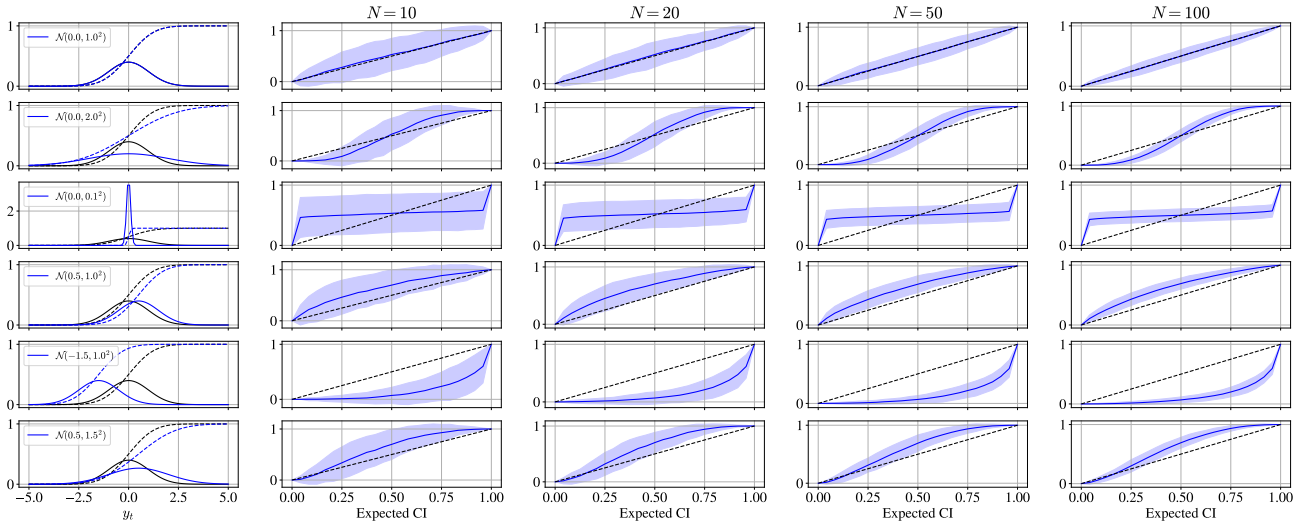

Figure 1: Examples of calibration curves computed on various number of test examples N, when the true data comes from a standard Gaussian and the model (left plots) varies (each row). Even in the best case scenario when samples are i.i.d., a large sample-to-sample variance can be expected in the ranges of N for which BO normally operates. Calibration curve distributions are made from 100 random seeds, and the intervals corresponds to two times the standard deviation.

6 EMPIRICAL CONFIRMATION OF PROPOSITION 1

To validate proposition 1, we now expand the experiment from Figure 1. In Figure 2, we have conducted a numerical experiment, where we sample 100 models of the form $p_t(y|x) = \mathcal{N}(y|\mu, \sigma)$, where $\mu \sim \mathcal{N}(0, 1)$ and $\sigma \sim \text{LogNormal}(1, 1)$. For each model, we compute 100 calibration curves for each sample size $N \in [5, 1000]$ and subsequently estimate the variance of those curves. Figure 2 shows the maximum standard deviation as a function of the sample size N.

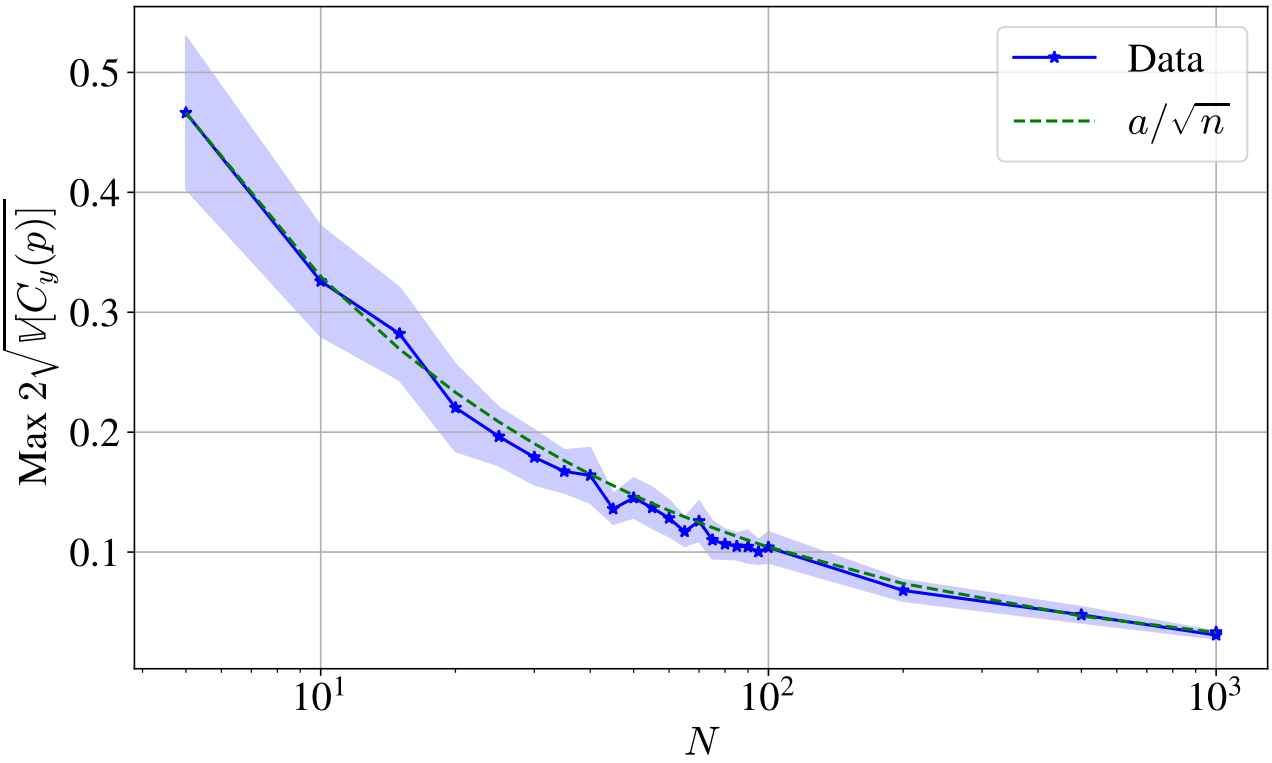

Figure 2: Maximum uncertainty across p for calibration distribution $C_p(y)$ when N samples of y is given for computing the individual calibration curves. We sample 100 models (normal distributions) each with arguments $\mu_i \sim \text{Normal}(0, 1)$ and $\sigma_i \sim \text{LogNormal}(1, 1)$ each modelling data coming from a standard normal. For each experiment 100 calibration curves, that is 100 independent samples of size N from the true model, constitutes the mean and std. We also plot the function $f(N) = a/\sqrt{N}$ for $a \approx 1.05$.

References

M. Balandat, B. Karrer, D. R. Jiang, S. Daulton, B. Letham, A. G. Wilson, and E. Bakshy. BoTorch: A Framework for Efficient Monte-Carlo Bayesian Optimization. In *Advances in Neural Information Processing Systems 33*, 2020. URL http://arxiv.org/abs/1910.06403.

J. R. Gardner, G. Pleiss, D. Bindel, K. Q. Weinberger, and A. G. Wilson. Gpytorch: Blackbox matrix-matrix gaussian process inference with gpu acceleration. In *Advances in Neural Information Processing Systems*, 2018.

S. Lee, H. Kim, and J. Lee. Graddiv: Adversarial robustness of randomized neural networks via gradient diversity regularization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

S. Seabold and J. Perktold. statsmodels: Econometric and statistical modeling with python. In *9th Python in Science Conference*, 2010.