SUPPLEMENTARY MATERIAL FOR "PROBABILISTIC META-LEARNING FOR BAYESIAN OPTIMIZATION"

The appendix contains additional plots of visualizations and additional results, detailed descriptions of the synthetic function ensembles and meta-learning benchmarks, and a section about the hyperparameters of BaNNER and their optimization.

A ADDITIONAL PLOTS

A.1 VISUALIZATION OF THE REGULARIZATION LOSS

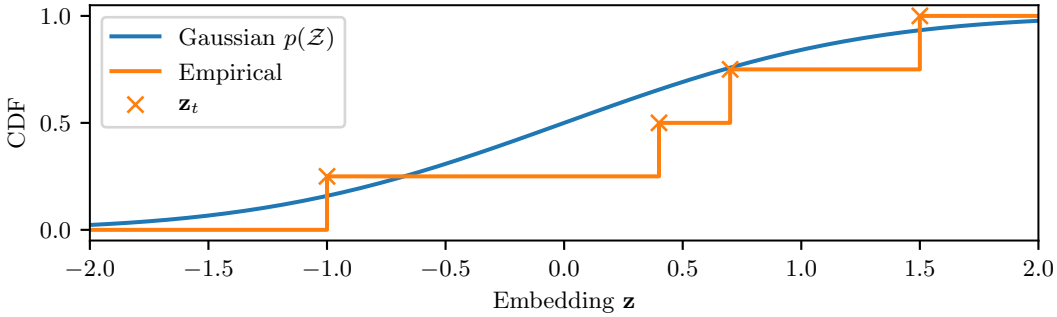

Figure 5: Example comparison between the Gaussian prior $p(\mathcal{Z})$ and the empirical CDF based on the embeddings $\mathbf{z}_1, \ldots, \mathbf{z}_T$. We compare the two distributions to regularize the task embeddings to conform with the prior $p(\mathcal{Z})$.

A.2 POSTERIOR VISUALIZATION

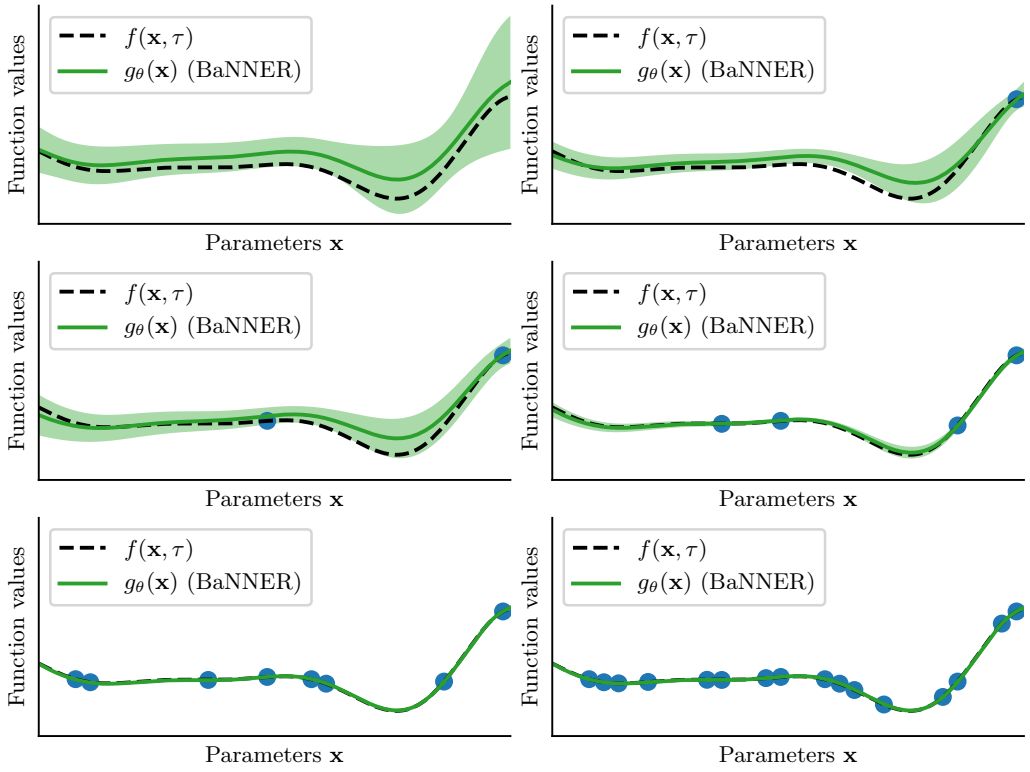

Figure 6: Visualization of the increasingly more accurate prediction (mean plus standard deviation in green) for the unknown task (dashed black) for various number of context points (0, 1, 2, 4, 8, 16). The points originate from the same distribution as used during meta-training.

A.3 Additional Experimental Results

Now, we show additional plots comparing all methods on all benchmarks using the mean and the standard error of the mean (95% confidence) and the median and the inter quantile range to visualize different aspects of the methods. The first one highlights the average expected regret when run many times, while the latter presents a picture of the variability of the runs, which might be more interesting when a robust result with only a few runs is desirable.

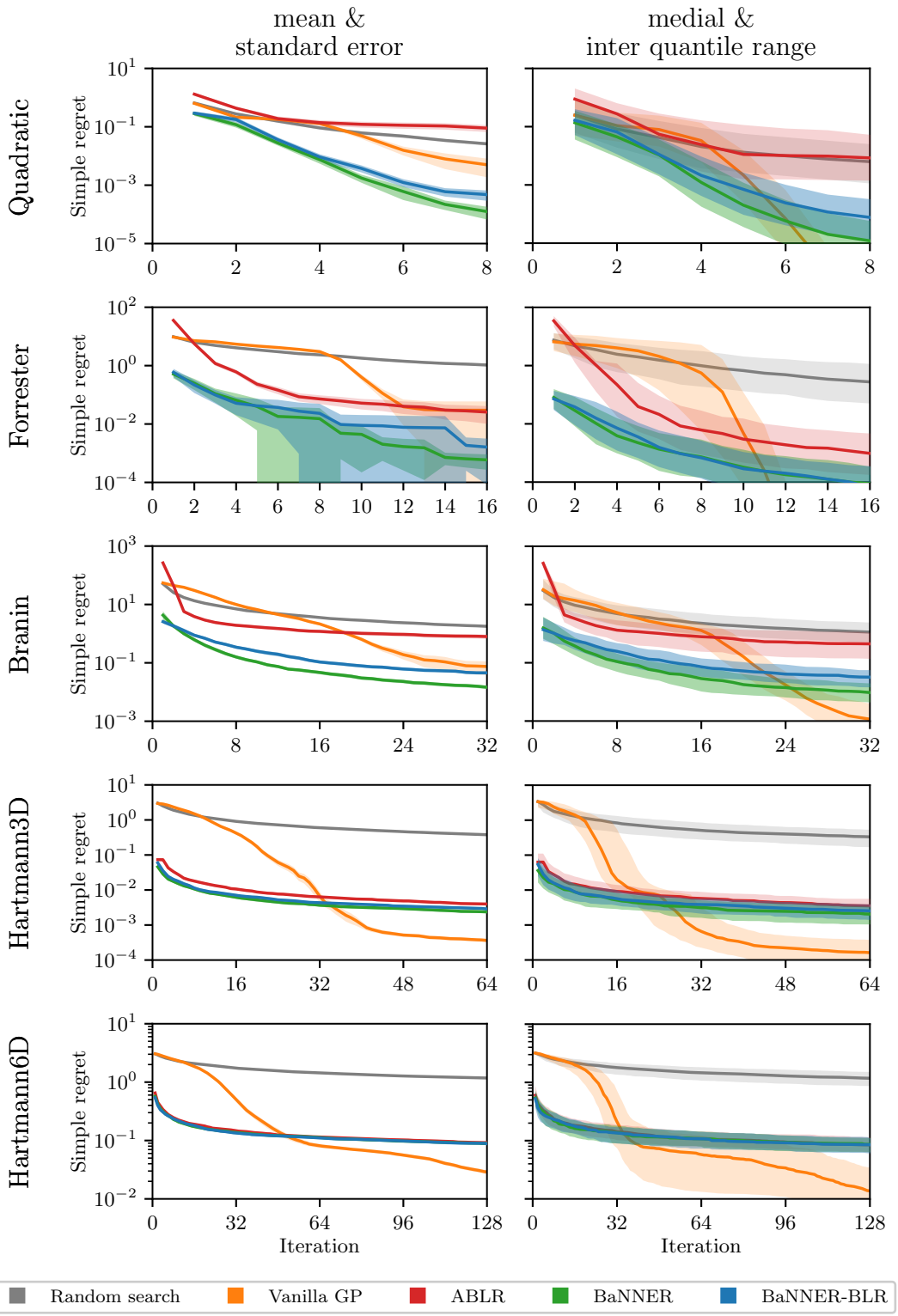

Figure 7: Performance on the ensembles of synthetic functions. Every row depicts a different synthetic function benchmark. Left panel shows Mean $\pm$ standard error of the mean, while the right depicts the medial and inter quantile range.

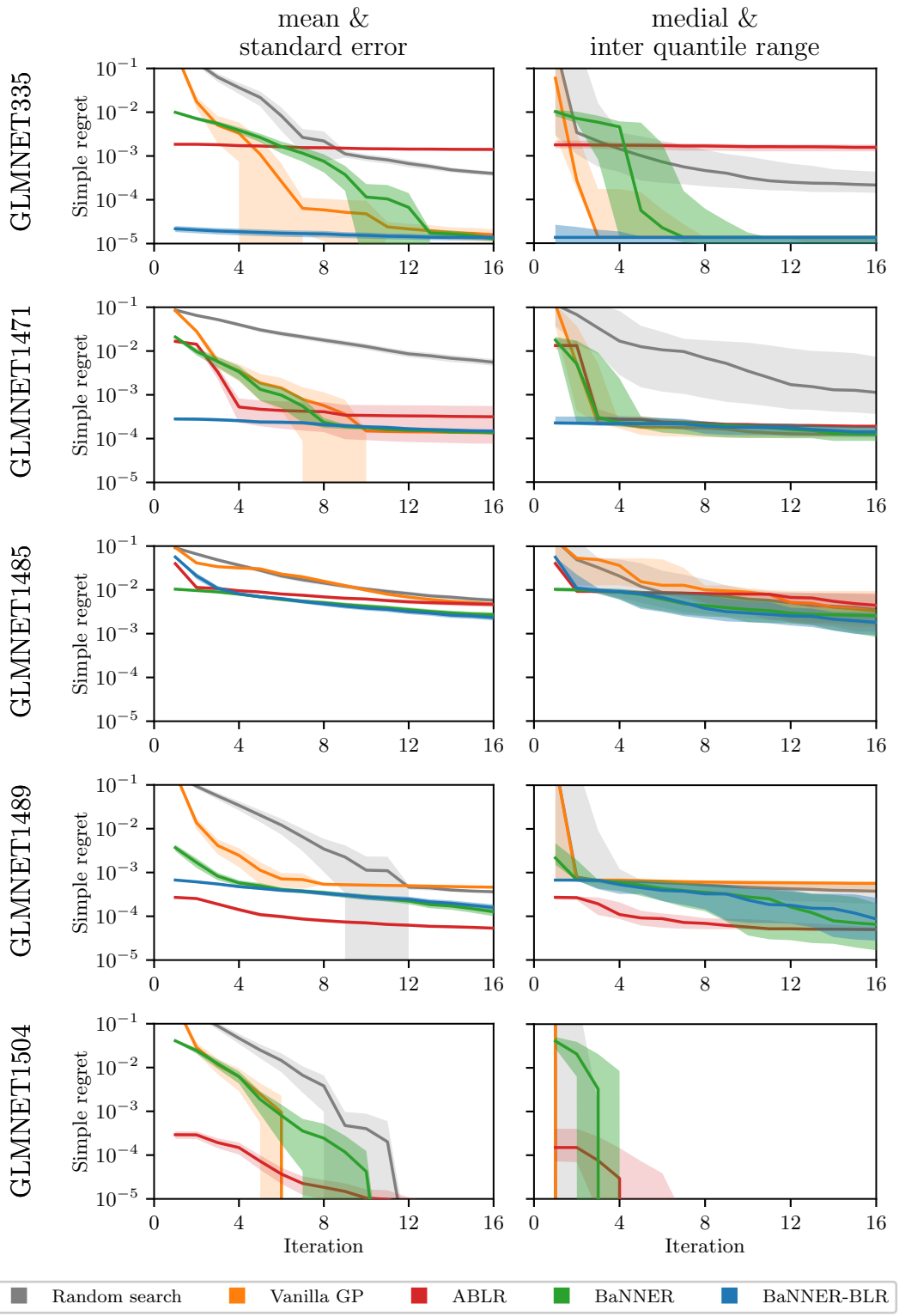

Figure 8: Performance on the GLMNET benchmark. Each row represents a different target task dataset. Left panel shows Mean $\pm$ standard error of the mean, while the right depicts the medial and inter quantile range.

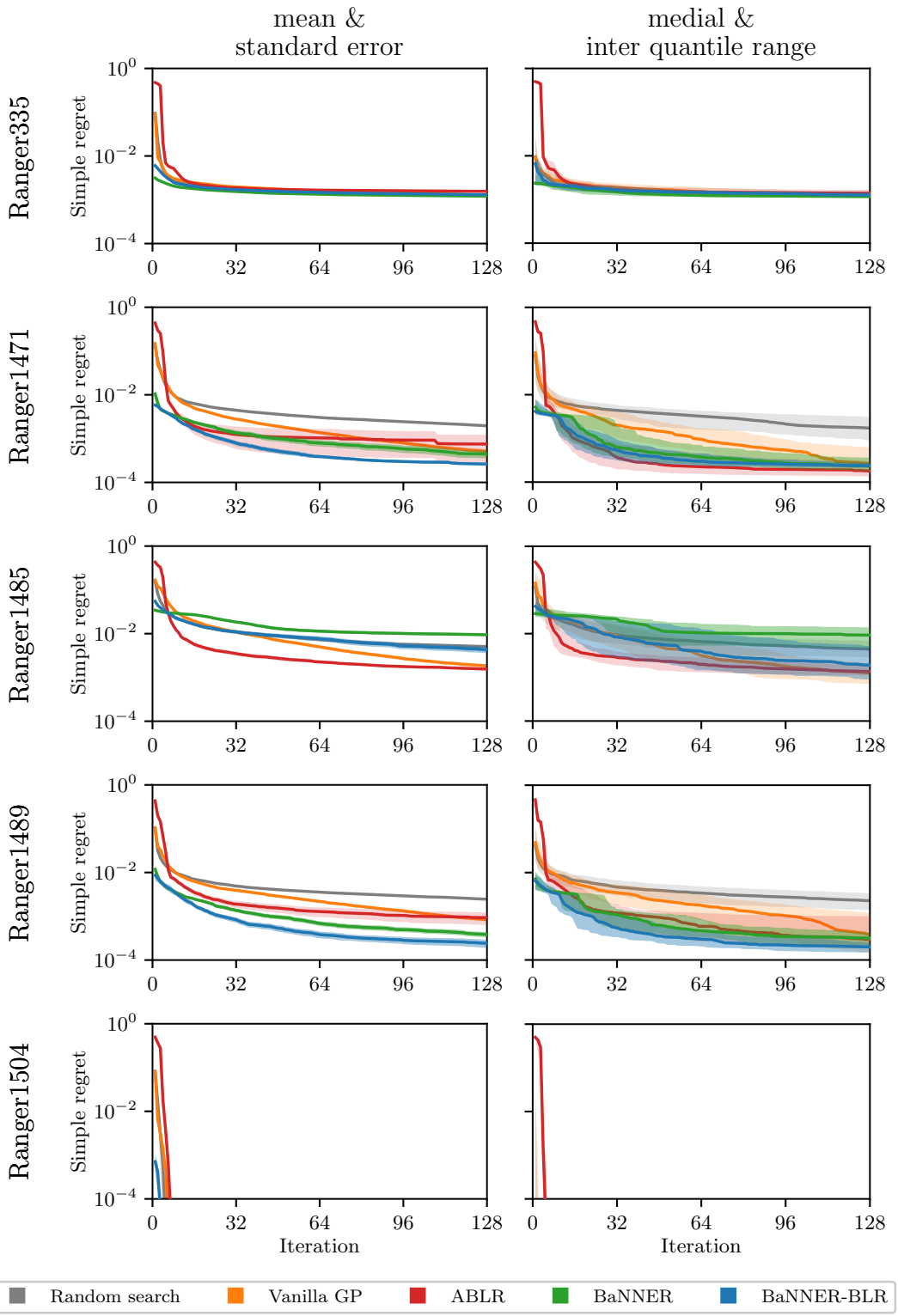

Figure 9: Performance on the Ranger benchmark. Each row represents a different target task dataset. Left panel shows Mean $\pm$ standard error of the mean, while the right depicts the medial and inter quantile range.

B DETAILS ON THE SYNTHETIC BENCHMARKS

Table 1: Overview over the synthetic function ensemble and their properties: the search space dimension d_x, the dimensionality of the unknown task parameters d_τ, the number of meta tasks used for training T, and the number of points per tasks N_t.

Ensemble	d_x	d_τ	T	N_t
Forrester	1	3	256	32
Quadratic	1	3	256	16
Branin	2	6	256	128
Hartmann3	3	4	256	1024
Hartmann6	6	4	256	1024

B.1 THE FORRESTER ENSEMBLE

The original Forrester function is defined as

$$f(x, a, b, c) = a \cdot (6x - 2)^2 \cdot \sin(12x - 4) + b \cdot (x - 0.5) - c \qquad x \in [0, 1] \tag{4}$$

The function has one global minimum, one local minimum, and a zero-gradient inflection point in the domain $x \in [0, 1]$. To form an ensemble of functions, we chose the following distributions:

$$a \sim \mathcal{U}(0.2, 3) \qquad b \sim \mathcal{U}(-5, 15) \qquad c \sim \mathcal{U}(-5, 5)\,. \tag{5}$$

Here $\tau = \{a, b, c\}$ and $p(\tau)$ is a three-dimensional uniform distribution. For the meta training, we used 256 randomly drawn tasks and sampled 32 points per task using a Sobol sequence, which we started at different sequence indices to ensure variety in the x values.

This ensemble with the chosen number of tasks and points per task constitutes an ensemble where the exact location of the optimum varies, but there are only two main regions where where it occurs. Meta-learning algorithm could benefit from that and show strong early performance.

B.2 THE QUADRATIC ENSEMBLE

The function for this ensemble reads

$$f(x, a, b, c) = (a \cdot (x - b))^2 - c \qquad x \in [-1, 1] \tag{6}$$

We chose the following distributions for the parameters:

$$a \sim \mathcal{U}(0.5, 1.5) \qquad b \sim \mathcal{U}(-0.9, 0.9) \qquad c \sim \mathcal{U}(-1, 1)\,. \tag{7}$$

This parametrization ensures that search interval always contains the minimum of the parabola at $x^* = b$ with $f(x^*) = c$. Here $\tau = \{a, b, c\}$ and $p(\tau)$ is a three-dimensional uniform distribution. For the meta training, we used 256 randomly drawn tasks and sampled 32 points per task using a Sobol sequence, which we started at different sequence indices to ensure variety in the x values.

This ensemble was designed to have a broader distribution over the location of the optimum to highlight algorithms that learn the global structure of the benchmark rather than focusing on a small area of interest.

B.3 THE BRANIN ENSEMBLE

The function for this ensemble reads

$$f(x, a, b, c) = a(x_2 - b \cdot x_1^2 + c \cdot x_1 - r) + s \cdot (1 - t) \cdot \cos(x_1) + s \qquad x_1 \in [-5, 10], x_2 \in [0, 15] \tag{8}$$

We chose the following distributions for the parameters:

$$\begin{aligned} &a \sim \mathcal{U}(0.5, 1.5) \quad b \sim \mathcal{U}(0.1, 0.15) \quad c \sim \mathcal{U}(1, 2) \\ &r \sim \mathcal{U}(5, 7) \qquad s \sim \mathcal{U}(8, 12) \qquad t \sim \mathcal{U}(0.03, 0.05) \end{aligned} \tag{9}$$

Here $\tau = \{a, b, c, r, s, t\}$ and $p(\tau)$ is a six-dimensional uniform distribution. The ranges where chosen around the usually used fixed values for the parameters. For the meta training, we used 256 randomly drawn tasks and sampled 128 points per task using a Sobol sequence, which we started at different sequence indices to ensure variety in the x values.

B.4 THE HARTMANN3 ENSEMBLE

The function for this ensemble reads

$$f(x, \alpha_1, \alpha_2, \alpha_3, \alpha_4) = -\sum_{i=1}^{4} \alpha_i \exp\left(-\sum_{j=1}^{3} A_{i,j}\left(x_j - P_{i,j}\right)^2\right) \qquad x \in [0,1]^3 \tag{10}$$

with

$$\mathbf{A} = \begin{bmatrix} 3.0 & 10 & 30 \\ 0.1 & 10 & 35 \\ 3.0 & 10 & 30 \\ 0.1 & 10 & 35 \end{bmatrix} \qquad \text{and} \qquad \mathbf{P} = 10^{-4} \cdot \begin{bmatrix} 3689 & 1170 & 2673 \\ 4699 & 4387 & 7470 \\ 1091 & 8732 & 5547 \\ 381 & 5743 & 8828 \end{bmatrix}. \tag{11}$$

We chose the following distributions for the parameters:

$$\begin{aligned} \alpha_1 &\sim \mathcal{U}(1.00, 1.02) \quad \alpha_2 \sim \mathcal{U}(1.18, 1.20) \\ \alpha_3 &\sim \mathcal{U}(2.8, 3.0) \quad \alpha_2 \sim \mathcal{U}(3.2, 3.4) \end{aligned} \tag{12}$$

Here $\tau = \{\alpha_1, \alpha_2, \alpha_3, \alpha_4\}$ and $p(\tau)$ is a four-dimensional uniform distribution. The ranges where chosen close to the usually used fixed values for the parameters. For the meta training, we used 256 randomly drawn tasks and sampled 1024 points per task using a Sobol sequence, which we started at different sequence indices to ensure variety in the x values. We increased the number of points per task to cover the three-dimensional space better.

B.5 THE HARTMANN6 ENSEMBLE

The function for this ensemble reads

$$f(x, \alpha_1, \alpha_2, \alpha_3, \alpha_4) = -\sum_{i=1}^{4} \alpha_i \exp\left(-\sum_{j=1}^{3} A_{i,j}\left(x_j - P_{i,j}\right)^2\right) \qquad x \in [0,1]^6 \tag{13}$$

with

$$\begin{aligned} \mathbf{A} &= \begin{bmatrix} 10 & 3 & 17 & 3.5 & 1.7 & 8 \\ 0.05 & 10 & 17 & 0.1 & 8 & 14 \\ 3 & 3.5 & 1.7 & 10 & 17 & 8 \\ 17 & 8 & 0.05 & 10 & 0.1 & 14 \end{bmatrix} \qquad \text{and} \\ \mathbf{P} &= 10^{-4} \cdot \begin{bmatrix} 1312 & 1696 & 5569 & 124 & 8283 & 5886 \\ 2329 & 4135 & 8307 & 3736 & 1004 & 9991 \\ 2348 & 1451 & 3522 & 2883 & 3047 & 6650 \\ 4047 & 8828 & 8732 & 5743 & 1091 & 381 \end{bmatrix}. \end{aligned} \tag{14}$$

We chose the following distributions for the parameters:

$$\begin{aligned} \alpha_1 &\sim \mathcal{U}(1.00, 1.02) \quad \alpha_2 \sim \mathcal{U}(1.18, 1.20) \\ \alpha_3 &\sim \mathcal{U}(2.8, 3.0) \quad \alpha_2 \sim \mathcal{U}(3.2, 3.4) \end{aligned} \tag{15}$$

Here $\tau = \{\alpha_1, \alpha_2, \alpha_3, \alpha_4\}$ and $p(\tau)$ is a four-dimensional uniform distribution. The ranges where chosen close to the usually used fixed values for the parameters. For the meta training, we used 256 randomly drawn tasks and sampled 1024 points per task using a Sobol sequence, which we started at different sequence indices to ensure variety in the x values. We increased the number of points per task to cover the three-dimensional space better.

C DETAILS ON THE SURROGATE META-LEARNING BENCHMARKS

Data_id	Task_id	Name	n	p	majPerc	numFeat	catFeat
335	3494	monks-problems-3	554	7	0.52	0	7
1471	9983	eeg-eye-state	14980	15	0.55	14	1
1485	9976	madelon	2600	501	0.50	500	1
1489	9952	phoneme	5404	6	0.71	5	1
1504	9967	steel-plates-fault	1941	34	0.65	33	1

Table 2: OpenML Random Bot surrogate benchmark test dataset characteristics, selected from the full repertoire of 37 datasets (cf Table 2 in Kühn et al. (2018a)). n are the number of observations, p the number of features, $\mathrm{majPerc}$ the percentage of observations in the largest class, $\mathrm{numFeat}$ the number of numeric features and $\mathrm{catFeat}$ the number of categorical features. Dataset 335 (monks-problems-3) is one of the six cases where only categorical and no numerical features are present. It is one of the smaller datasets with only 5 out of 37 having fewer features, while all of these five have more observations available. Dataset 1471 (eeg-eye-state) has average characteristics and, like the majority of datasets, has only one categorical feature. Dataset 1485 (madelon) has the third largest number of features with all except one being numeric, while containing a number observations below average. Dataset 1489 (phoneme) is one of the five with the fewest features but contains an average amount of observations. Dataset 1504 (steel-plates-fault) has average characteristics. These five datasets were chosen randomly.

D DETAILS ON THE HYPERPARAMETER OPTIMIZATION

Here we provide details of the hyperparameter optimization (HPO) used to find the best configuration of BaNNER based on the meta data for each task.

For HPO, we used BOHB by Falkner et al. (2018) to run concurrently on 20 CPUs (Intel® Xeon® CPU E5-2695 v4) for optimization. The optimization budget for each task was limited to two days, which depending on the dimensionality yielded slightly different numbers of evaluated configurations. To speed up the optimization over the 15 dimensional search space (see Table 3), we used the multi-fidelity feature of BOHB, which allows to quickly evaluate an approximation to the true validation loss. We scaled both, the number of training epochs and the number of validation tasks used to compute the average log-likelihood (ALL) of unseen points given test data. This allowed to quickly discard configurations with unreasonable parameters and focuses the search in the promising parts of the search space. The exact number of training iterations for the benchmarks can be found in the code that we will release upon acceptance of the paper. They where chosen such that training and evaluation of a single model on the highest fidelity took approximately 1 hour. As the smallest fidelity, we used $1/32$ of the training iterations and validation tasks, which corresponds to a training of about 2 minutes and an evaluation on 4 validation tasks.

We fixed a few other hyperparameters for different reasons: We focuse on ADAM by Kingma & Ba (2014), as we saw no difference between SGD and ADAM in early evaluations. We fixed the batch size to 128 to limit the variation in training time because extremely small and large batch sizes lead to increased run time. We fixed the number of HMC burin-steps to 512 and the number of collected HMC samples to 1024 during training. With this choice, BaNNER finishes inference on the benchmarks in approximately 5-10 seconds (depending on d_z, the dimensionality of the latent representation). We fixed the initialization of z for the sampling to zero, which is the mode of our prior and also showed stable results in early evaluations compared to random samples or MAP estimates of the latent embedding. We also normalize the input domain via Min-Max-scaling to the unit hypercube and apply a standardization to the y values, choices one could potentially also optimize, but was not necessary on the benchmarks presented in the paper.

Besides the goal of maximizing the ALL, we also rejected configurations where the HMC step did not finish in a reasonable amount of time. We observed that some configurations slowed down the HPO by orders of magnitude because the HMC sampling rejected most of the samples resulting in a much larger computational cost. We actively canceled runs where the sampling on any of the

Table 3: Search space for the hyperparameter optimization for BaNNER. We optimize over training parameters (e.g. the learning rate schedule), architectural choices (e.g. number of layers and units per layers of the multi layer perceptron (MLP)), inference parameters (e.g. HMC step size), and unknown properties of the tasks (e.g. the latest dimensionality d_z). The "Log" transformation indicates parameters we model on a logarithmic scale rather than a linear one. We placed a uniform prior over each parameter. A description of the parameters can be found in Table 4.

Parameter Name	Type	Values	Transformation
LR schedule	Categorical	[ExponentialDecay, PolynomialDecay]	
LR exponent	Float	$[0.1, \cdot 10^1]$	Log
initial LR	Float	$[\cdot 10^{-6}, 1.0]$	Log
final LR / inital LR	Float	$[0.0001, 1.0]$	Log
d_z	Integer	$[1, 16]$	
$p_0(z)$	Categorical	[$\mathcal{N}$, δ]	
ϵ_z	Float	$[\cdot 10^{-6}, 1.0]$	Log
λ^{-1}	Float	$[\cdot 10^2, \cdot 10^6]$	Log
MLP depth	Integer	$[1, 3]$	
MLP width	Integer	$[16, 256]$	Log
# BLR Features	Integer	$[4, 64]$	
Hidden Activation	Categorical	[ReLU, Tanh, Sigmoid, RBF]	
HMC Step Size	Float	$[0.001, 1.0]$	Log
#HMC Steps	Integer	$[1, 64]$	Log
HMC acceptance prob.	Float	$[0.4, 1.0]$	

validation tasks took longer than 100 HMC iterations per second (on average plus) on our hardware. This means, that the HPO will also select configurations with a stable and efficient inference.

We show the best found configurations for all benchmarks in Table 4. The results suggest that most parameters are adapted to achieve the best generalization on the different benchmarks. We see different complexity of the MLP from very small on the Quadratic and Ranger benchmark to deeper and wider on, e.g., Forrester and Hartmann3D. Also the learning rate schedule, and the embedding noise ϵ_z vary between the different tasks. The HMC acceptance probability seems to be an oddity in the results. The optimizer consistently favors configurations where the value is lower than the default of 0.8. We suspect this could be an artifact of our short sampling for inference, which stabilize the the predictions by rejecting MCMC proposals more often than usual.

Table 4: Best found configuration of BaNNER on the different benchmarks. The learning rate schedule parameters (all containing a LR) control the meta-training. The latent representation for the task has d_z dimensions and the values are initialized via $p_0(z)$ either as all zeros or randomly according to the prior. During the training, Gaussian noise with isotropic scale ϵ_z. The network consists of "MLP depth" hidden layers of width "MLP width", all using the same "Hidden Activation". This network is followed by the last hidden layer with $\#BLRFeatures$ units. The remaining parameters affect the HMC sampling by setting the discretization step length, the number of those steps between MCMC proposals and the target acceptance probability for sampling.

Parameter	Forrester	Quadratic	Branin	Hartmann3D	Hartmann6D	GLMNET	Ranger
LR schedule	Exponential	Polynomial	Exponential	Polynomial	Polynomial	Polynomial	Polynomial
LR exponent	2.0	4.13	None	0.139	1.64	0.294	4.19
initial LR	0.00246	0.00205	0.000715	$5.37 \cdot 10^{-6}$	0.000375	0.000121	0.000909
final LR / inital LR	0.000862	0.525	0.13	0.501	0.0148	0.000277	0.00123
d_z	4.0	10.0	10.0	16.0	16.0	15.0	14.0
$p_0(z)$	zeroes	zeroes	zeroes	zeroes	zeroes	random	zeroes
ϵ_z	$5.94 \cdot 10^{-5}$	0.000811	0.00336	0.0402	0.122	0.017	0.328
Λ_R	$2.46 \cdot 10^3$	$3.27 \cdot 10^2$	$6.31 \cdot 10^4$	$6.28 \cdot 10^4$	$8.09 \cdot 10^4$	$1.53 \cdot 10^4$	$1.39 \cdot 10^4$
MLP width	152	18	123	47	124	142	21
MLP depth	2	1	2	3	1	1	1
# BLR Features	29.0	61.0	58.0	42.0	14.0	50.0	51.0
Hidden Activation	Sigmoid	Tanh	ReLU	ReLU	ReLU	ReLU	RBF
HMC Step Size	0.00109	0.00324	0.00135	0.00156	0.00104	0.00375	0.00122
#HMC Steps	10.0	24.0	1.0	1.0	9.0	2.0	23.0
HMC acceptance prob.	0.487	0.582	0.668	0.611	0.525	0.663	0.565