Supplementary Material for Fast Bayesian Optimization of Machine Learning Hyperparameters on Large Datasets

A Entropy Search

A.1 Approximations used

Given $p(f)$, the probability that a point is the minimum is defined with suggestive notation as

$$p_{min}(\boldsymbol{x}|\mathcal{D}) = p(\boldsymbol{x} = \underset{\boldsymbol{x}' \in \mathcal{X}}{\arg\min}\, f(\boldsymbol{x}')|D) = \int p(f|\mathcal{D}) \prod_{\substack{\tilde{\boldsymbol{x}} \in [a,b] \\ \tilde{\boldsymbol{x}} \neq \boldsymbol{x}}} \Theta[f(\tilde{\boldsymbol{x}}) - f(\boldsymbol{x})]\, df \quad (1)$$

where Θ is the Heaviside's step function. The product in this equation is over an infinite domain (yet well-defined if $p(f|D)$ is sufficiently regular). In practice, it has to be represented in a finite form. We follow the approach of Hennig and Schuler (2012), who approximate $p(f|D)$ by a finite-dimensional Gaussian over an irregular grid of points $\boldsymbol{r}_1, ..., \boldsymbol{r}_Z$, which are designed heuristically to provide good interpolation resolution on $p_{\min}$. Like Hennig and Schuler (2012), we sample these so called representer points using Expected Improvement. This step reduces $p_{\min}$ to a discrete distribution, and turns the infinite product in Equation 1 into a finite one. That distribution itself is still analytically intractable, but an analytically tractable (in particular, differentiable) approximation $q_{\min}(\boldsymbol{r}_j)$ of good empirical quality can be computed using *Expectation Propagation (EP)* (Minka, 2001), at computational cost in $\mathcal{O}(Z^4)$. EP does not only yield $p_{\min}$, but also the gradient with respect to means and covariances of the model at the representer points allowing efficient computations after an expensive initial calculation of these quantities. This particular application of EP to Gaussian integrals was introduced by Cunningham et al. (2012) where all the details can be found.

A.2 Pseudocode

Algorithm 1 provides pseudocode for our implementation of Entropy Search. Lines 1-12 precompute various quantities that are needed for evaluating the acquisition function, which is optimized in line 13. Specifically, after sampling K hyperparameter settings from the marginal loglikelihood for the GP using MCMC (line 1), for every hyperparameter setting $\boldsymbol{\theta}_i$, the algorithm

- fits a GP (line 4),
- samples representer points with respect to a_{EI} (line 5),
- stores the representer points and their logarithmic EI values (lines 6 and 7),
- computes $\boldsymbol{\mu}$ and Σ for the joint predictive distribution at the representer points (line 8),
- computes p_{min} given $\boldsymbol{\mu}$ and Σ, using EPMGP (line 9),
- draws random points from a normal distribution centered at 0 and unit variance (line 10) for the innovation in Algorithm 3, and stores them (line 11) for later usage.

Algorithm 1 Selection of next point by Entropy Search

Require: $\mathcal{D}_n = (\boldsymbol{x}_j, y_j)_{j=1...n}$

1: Sample K instantiations of the GP hyperparameters $\boldsymbol{\Theta} = [\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_K]$ w.r.t. marginal likelihood
2: $p_{min} \leftarrow [], \Omega \leftarrow [], \boldsymbol{R} \leftarrow [], \boldsymbol{U} \leftarrow []$
3: **for** $i = 1 \ldots K$ **do**
4: Fit GP model $\mathcal{M}^{(i)}$ on $\mathcal{D}_n$ with hyperparameter $\boldsymbol{\theta}_i$
5: $(\boldsymbol{r}_1, a_{EI}(\boldsymbol{r}_1)) \ldots, (\boldsymbol{r}_Z, a_{EI}(\boldsymbol{r}_Z)) \sim a_{EI}(\boldsymbol{x}|\mathcal{M}^{(i)})$ ▷ Sample Z representer points
6: $\boldsymbol{R}[i] \leftarrow \boldsymbol{r}_1, \ldots, \boldsymbol{r}_Z$ ▷ Store representer points $R \in \mathbb{R}^{K \times Z \times D}$
7: $\boldsymbol{U}[i] \leftarrow a_{EI}(\boldsymbol{r}_1), \ldots, a_{EI}(\boldsymbol{r}_Z)$ ▷ Store LogEI values of the representer points $U \in \mathbb{R}^{K \times Z}$
8: Let $\boldsymbol{\mu}, \Sigma$ be the mean and covariance matrix at $\boldsymbol{r}_1, \ldots, \boldsymbol{r}_Z$ based on $\mathcal{M}^{(i)}$
9: $p_{min}[i] \leftarrow$ computePmin $(\boldsymbol{\mu}, \Sigma)$ ▷ Probability of each $\boldsymbol{r}_1, \ldots, \boldsymbol{r}_Z$ to be the minimum.
10: For $p = 1, \ldots, P$: $\boldsymbol{\omega}_p \sim \mathcal{N}(0, \boldsymbol{I}_Z)$ ▷ Stochastic change to hallucinate P values at representer points
11: $\Omega[i] \leftarrow [\boldsymbol{\omega_1}, \ldots, \boldsymbol{\omega}_P]$ ▷ Store stochastic change for the innovations $\Omega \in \mathbb{R}^{K \times Z \times P}$
12: **end for**
13: $\boldsymbol{x}_{n+1} \leftarrow \arg\max_{\boldsymbol{x} \in \mathcal{X}}$ InformationGain$(\boldsymbol{x}, \mathcal{D}_n, \boldsymbol{R}, \boldsymbol{U}, \Omega, \boldsymbol{\Theta})$
14: **return** $\boldsymbol{x}_{n+1}$

Given these quantities, Algorithm 2 then computes the ES acquisition function from Equation (3) in the main paper (repeated here for convenience):

$$a_{\text{ES}}(\boldsymbol{x}) := \mathbb{E}_{p(y|\boldsymbol{x},\mathcal{D})}\left[\int p_{\min}(\boldsymbol{x}' \mid \mathcal{D} \cup \{(\boldsymbol{x}, y)\}) \cdot \log \frac{p_{\min}(\boldsymbol{x}' \mid \mathcal{D} \cup \{(\boldsymbol{x}, y)\})}{u(\boldsymbol{x}')} \, \mathrm{d}\boldsymbol{x}'\right].$$

For each hyperparameter $\boldsymbol{\theta}_i$ of the GP, it then carries out the following steps:

- train a model $\mathcal{M}^{(i)}$ on the data $\mathcal{D}$ by computing the Cholesky decomposition (line 3)
- based on this model $\mathcal{M}^{(i)}$, compute the mean and the variance for the test point $\boldsymbol{x}$ and the mean and covariance for the representer points $\boldsymbol{r}_1, \ldots, \boldsymbol{r}_Z$ (line 4 and 5)
- For each of the P stochastic change vectors $\boldsymbol{\omega}_p$ sampled in Algorithm 1 (line 10) and stored in $\Omega[i, j, :]$,
  - fantasize the change $\Delta\boldsymbol{\mu}, \Delta\Sigma$ of the current posterior $p(f|D)$ (line 7) with Algorithm 3
  - estimate the p_{min} distribution of this updated posterior (line 8)
  - compute the relative change in entropy (line 9)
- take the expectation over $p(y|\boldsymbol{x}, D)$ of Equation (3) (line 10)
- marginalize the acquisition function $a_{\text{ES}}(\boldsymbol{x})$ over all hyperparameters $\boldsymbol{\Theta}$ (line 13)

Algorithm 2 InformationGain

Require: $\boldsymbol{x}, \mathcal{D}, \boldsymbol{R}, \boldsymbol{U}, \Omega, \boldsymbol{\Theta}$

1: $a(\boldsymbol{x}) \leftarrow 0$
2: **for** $i = 1, \ldots K$ **do** ▷ Marginalization over $\boldsymbol{\Theta}$
3: Let $\mathcal{M}^{(i)}$ be the trained model on $\mathcal{D}$ with hyperparameters $\boldsymbol{\theta}_i$
4: Let μ, σ^2 be the predictive mean and variance at $\boldsymbol{x}$ based on $\mathcal{M}^{(i)}$
5: Let $\boldsymbol{\mu}, \Sigma$ be the mean and covariance matrix at $\boldsymbol{r}_1, \ldots, \boldsymbol{r}_Z$ based on $\mathcal{M}^{(i)}$
6: **for** $j = 0, \ldots P$ **do**▷ Averages over all hallucinated values.
7: $\Delta\boldsymbol{\mu}, \Delta\Sigma \leftarrow$ Innovations$(\boldsymbol{x}, \mathcal{M}^{(i)}, \boldsymbol{R}[i, :, :], \sigma^2, \Omega[i, j, :])$ ▷ Change in the posterior believe at $\boldsymbol{r}_1, \ldots, \boldsymbol{r}_Z$ if we would evaluate at $\boldsymbol{x}$
8: $q_{min} \leftarrow$ computePmin $(\boldsymbol{\mu} + \Delta\boldsymbol{\mu}, \Sigma + \Delta\Sigma)$ ▷ New Pmin of the updated posterior
9: $d\mathrm{H} \leftarrow -\sum_j q_{min}\big(\log(q_{min}) + \boldsymbol{U}[i]\big) + \sum_j p_{min}[i]\big(\log(p_{min}[i]) + \boldsymbol{U}[i]\big)$
10: $a(\boldsymbol{x}) \leftarrow a(\boldsymbol{x}) + \frac{1}{P} dH$
11: **end for**
12: **end for**
13: **return** $\frac{1}{K} a(\boldsymbol{x})$

This algorithm in turns makes use of Algorithm 3 to compute the innovations, which

- computes the change in the mean $\Delta\boldsymbol{\mu}$ by first computing the correlation $\Sigma(\boldsymbol{x}, \boldsymbol{r})$ of $\boldsymbol{x}$ and the representer points $\boldsymbol{r}_1, \ldots, \boldsymbol{r}_Z$ and multiplying it with the Cholesky decomposition of the $k(\boldsymbol{x}, \boldsymbol{x})$ and the vector $\omega \in \boldsymbol{\Omega}$. Note that this change is stochastic (line 1).
- computes the change of the covariance (line 2) which is deterministic

Algorithm 3 Innovations

Require: $\boldsymbol{x}, \mathcal{M}, \boldsymbol{r}_1, \ldots, \boldsymbol{r}_Z, \sigma^2, \boldsymbol{\omega}$

1: $\Delta\boldsymbol{\mu}(\boldsymbol{x}) = \Sigma(\boldsymbol{x}, \boldsymbol{r}) * \sigma^2 * C[\sigma^2 + \sigma^2_{noise}]\boldsymbol{\omega}$ ▷ $\Sigma(\boldsymbol{x}, \boldsymbol{x}')$ denotes the correlation between $\boldsymbol{x}$ and $\boldsymbol{x}'$ based on $\mathcal{M}$
2: $\Delta\Sigma(\boldsymbol{x}) = \Sigma(\boldsymbol{x}, \boldsymbol{r}) * \sigma^2 * \Sigma(\boldsymbol{x}, \boldsymbol{r})^T$
3: **return** $\Delta\boldsymbol{\mu}(\boldsymbol{x}), \Delta\Sigma(\boldsymbol{x})$

B Scaling of Loss and Computational Cost With Dataset Size

The runtime of machine learning algorithms usually scales polynomially with the number of data points (N_{sub}), i.e. $\mathcal{O}(N_{\text{sub}}^{\alpha})$ for some positive α. While the computational cost of training grows, the loss of machine learning methods usually decreases with the number of training samples. The computational cost is often largely independent of the hyperparameter values, but the loss depends crucially on the hyperparameter values (which is the reason we want to optimize hyperparameters in the first place).

We invested these trends using our 20×20 grid of SVM configurations for MNIST described in the main text. Figure 1 shows these trends for ten random configurations, evaluated on subsets of different sizes. We note that, as training size increases, the loss of many configurations decreases, but the relative ordering does not change dramatically, such that training on few data points provides information about the full data set. The training time behaves similarly across different configurations. Please note that the complexity of the underlying SVM is not trivial, which is why the curves are not straight lines.

To show that our method, i.e. the kernel we use and our initial design, actually capture these trends, we sampled points from that data as our initial design and predicted loss and cost of unseen configurations, see Figure 2. The cost model already quite accurately captures the growth trends based on this small amount of initial data. The loss model predicts worse (especially for configurations very different from the ones in the initial design), but it also reflects that errors decrease with increasing data set size.

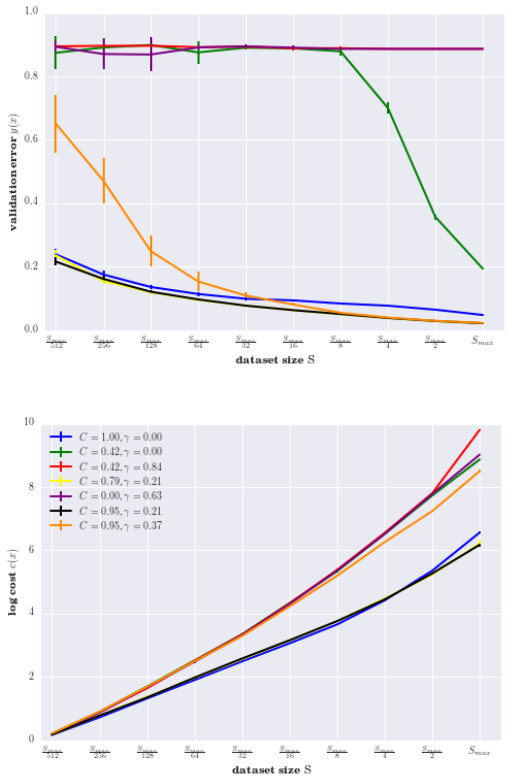

Figure 1: Validation error (left) and training time (left) for ten random configurations training a SVM on the MNIST dataset. For all costs, the cost increases with s whereas the validation error decreases or stays constant.

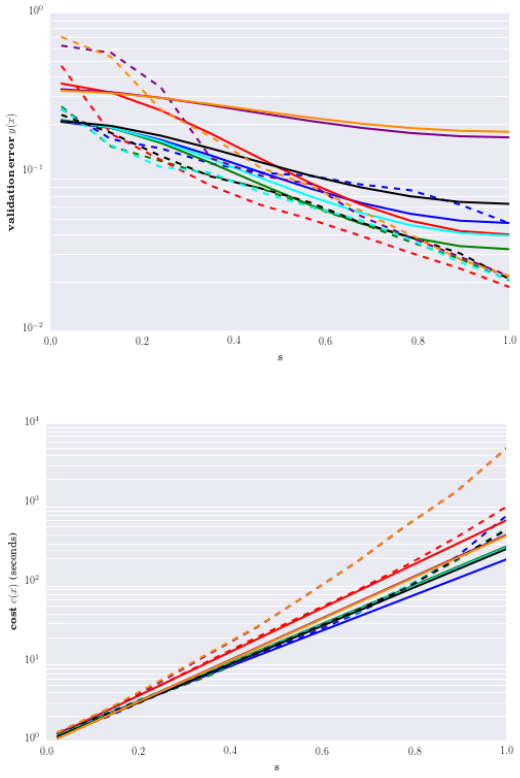

Figure 2: Model (solid line) of the objective function and the cost function (dashed) after the initial design (2 configurations evaluated on $N/16$, $N/32$, $N/64$, $N/128$, $N/256$, $N/512$.

C Modeling the Heteroscedastic Noise

When making the subset size a parameter, we shuffle the data before an evaluation to prevent bias incurred by repeatedly using the same subset. This shuffling introduces additional noise which could be particularly high for small subsets. To investigate this, we again used the SVM grid of 400 configurations from the main paper. We repeated each run with a given subset size $K = 10$ times using different subsets, and estimate the observation noise variance at each point as:

$$\sigma^2_{obs}(\boldsymbol{x}_j, s_i) = \frac{1}{K}\sum_{k=1}^{K}(y_k(\boldsymbol{x}_j, s_i) - \mu_{i,j})^2\,, \qquad (2)$$

where $\mu_{i,j} = K^{-1}\sum_{k=1}^{K} y_k(\boldsymbol{x}_j, s_i)$. The red points in Figure 3 show the mean and standard deviation of $\sigma^2_{obs}(\boldsymbol{x}_j, s_i)$ over all configurations for all s_i values considered. As expected, the noise decreases with an increasing s, to a point where σ^2_{obs} is zero for $s = 1$.

In contrast to this heteroscedastic noise intrinsic to the random subsampling, the commonly used noise hyperparameter σ^2 of a GP (call it σ^2_{GP}) is fixed and typically estimated using MCMC sampling. To compare these two noise values, for each fixed size s, we also trained a GP to predict losses and plotted its estimates σ^2_{GP} as blue markers in Figure 3. To obtain a good estimate of the GP's hyperparameters, we used a relatively long MCMC chain compared to the ones used during Bayesian optimization. Figure 3 clearly shows that the estimated variance σ^2_{GP} is always larger than the observation noise σ^2_{obs}. This might indicate a certain misfit between the true objective and the space of functions the GP can model (Sollich, 2002). Consequently, we believe the heteroscedastic noise from subsampling the data to often be negligible compared to the noise estimated by the MCMC sampling.

Nonetheless, we model the noise decreasing in s by an additive kernel $K_{noise} = diag(r_i)$ with $r_i = \alpha s_i^{\beta}$ for each data point $(\boldsymbol{x}_i, s_i)$. This adds a heteroscedastic noise to the Gram matrix of our kernel k_f for our objective function, and we treat α and β as additional hyperparameters. These are marginalized together with the other GP's hyperparameter by sampling from the marginal loglikelihood (see Section 3.4 in the main paper for details). Figure 4 shows sample functions of this kernel for different α and β sampled from

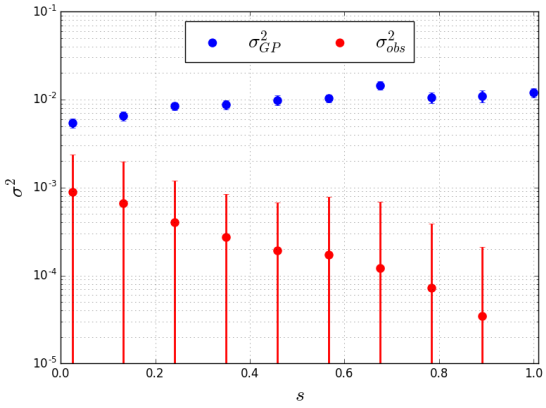

Figure 3: Evaluating a configurations on a shuffled subset of the data induces an additional noise, σ^2_{obs} that depends on the dataset size s. The noise parameter σ^2_{GP} estimated by MCMC sampling for fixed dataset sizes.

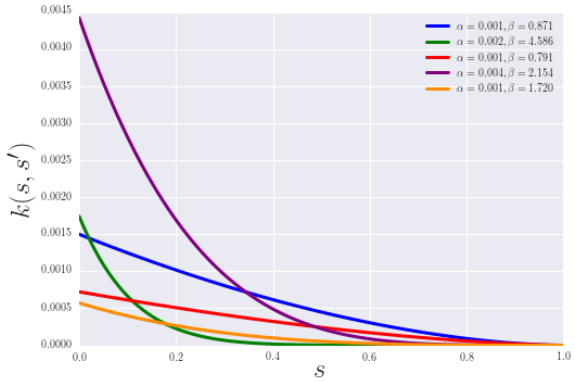

Figure 4: Sample functions of our heteroscedastic noise kernel for different values for its hyperparameters α and β.

their respective priors:

$$\alpha \sim \log \mathcal{N}(-3, 1) \qquad \beta \sim \log \mathcal{N}(3, 1)\,. \tag{3}$$

These are not very strong priors, but they still enforce the MCMC sampler to choose configurations similar to those shown in Figure 4, which resemble the ones in Figure 3. We performed preliminary Bayesian optimization experiments with this kernel, but these did not lead to an additional improvement. Thus we did not use them in our production experiments in order to reduce our model's total number of hyperparameters.

D Optimization ranges for Bayesian optimization experiments

Table 1 shows the ranges of the 2 hyperparameters we optimized in our SVM experiments, Table 2 the ranges of the 5 convolutional neural network hyperparameters we optimized, and Table 3 the ranges of the 4 deep residual network hyperparameters we optimized.

Table 1: Hyperparameters for all support vector machine tasks.

Hyperparameter	lower bound	upper bound	log
Regularization C	e^{-10}	e^{10}	X
Kernel parameter γ	e^{-10}	e^{10}	X

Table 2: Hyperparameters for the convolutional neural network task.

Hyperparameter	lower bound	upper bound	log
Initial learning rate	10^{-6}	10^{0}	X
Batch size	32	512	
# units layer 1	2^4	2^9	X
# units layer 2	2^4	2^9	X
# units layer 3	2^4	2^9	X

Table 3: Hyperparameters for the deep residual network task.

Hyperparameter	lower bound	upper bound	log
Learning rate	10^{-6}	1	X
L_2 regularization	10^{-6}	1	X
Learning rate factor	10^{-4}	1	X
Momentum	0.1	0.999	

References

P. Hennig and C. Schuler. Entropy search for information-efficient global optimization. *JMLR*, (1), 2012.

Thomas P. Minka. Expectation propagation for approximate bayesian inference. In *Proc. of UAI'01*, UAI '01, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

J. Cunningham, P. Hennig, and S. Lacoste-Julien. Approximate gaussian integration using expectation propagation. pages 1–11, January 2012.

Peter Sollich. Gaussian process regression with mismatched models. In T.G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*. MIT Press, 2002.