Bayesian Neural Networks for Predicting Learning Curves

Aaron Klein Stefan Falkner Jost Tobias Springenberg Frank Hutter
Department of Computer Science
University of Freiburg
{kleinaa,sfalkner,springj,fh}@cs.uni-freiburg.de

Abstract

The performance of deep neural networks (DNNs) crucially relies on good hyperparameter settings. Since the computational expense of training DNNs renders traditional blackbox optimization infeasible, recent advances in Bayesian optimization model the performance of iterative methods as a function of time to adaptively allocate more resources to promising hyperparameter settings. Here, we propose a specialized Bayesian neural network to model DNN learning curves jointly across hyperparameter configurations.

1 Introduction

Deep learning has celebrated many successes, but its performance relies crucially on good hyperparameter settings. Bayesian optimization (e.g, [1, 2, 3]) is a powerful method for optimizing deep neural networks (DNNs), but its traditional treatment of DNN performance as a black box poses fundamental limitations for large and computationally expensive data sets, for which training a single model can take weeks. Recent extensions move beyond this blackbox notion to exploit cheaper signals about which hyperparameter settings work well, such as the performance after a few epochs of an iterative training method [4, 5], performance on small subsets of the data [6], and performance on other, related data sets [7, 8]. All of these extensions yield many cheap data points that need to be modelled, requiring scalable probabilistic models with well-calibrated uncertainty estimates for Bayesian optimization.

While traditional solutions for scalable Bayesian optimization include approximate Gaussian process models (e.g., [9, 4]) and random forests [10], a recent trend is to exploit the flexible model class of neural networks for this purpose [11, 12]. In this paper, we develop a specialized model for learning curves based on Bayesian neural networks that combines MCMC sampling using SGHMC (as in [12]) with a flexible parametric learning curve model (as in Domhan et al. [5]). Preliminary experiments show that our method can predict learning curves of different hyperparameter settings better than standard Bayesian neural networks.

2 Probabilistic prediction of learning curves with Bayesian neural networks

Probabilistic extrapolation of learning curves. Domhan et al. [5] proposed a model that uses a set of k different parametric functions $\phi_i(\boldsymbol{\theta}_i, t) \in \{\phi_1(\boldsymbol{\theta}_1, t), ...\phi_k(\boldsymbol{\theta}_k, t)\}$ to extrapolate learning curves $y_{1:n}$ from their first n time steps. Each parametric function ϕ_i depends on a time step $t \in [1, T]$ and on a parameter vector $\boldsymbol{\theta}_i$. The individual functions are combined to a single predictive model by a weighted linear combination $f(t, \boldsymbol{\xi}) = \sum_{i=1}^{k} w_i \phi_i(t, \boldsymbol{\theta}_i)$, where $\boldsymbol{\xi} = (w_1, \ldots, w_k, \boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_k, \sigma^2)$ denotes the combined vector of all parameters $\boldsymbol{\theta}_1, ..., \boldsymbol{\theta}_k$, weights $w_1, ..., w_k$ and the noise variance σ^2 of the observation noise $\mathcal{N}(0, \sigma^2)$ such that $y_t \sim \mathcal{N}(f(t|\boldsymbol{\xi}), \sigma^2)$. Domhan et al. [5] define a prior

$P(\boldsymbol{\xi})$ and use MCMC to obtain S samples $\boldsymbol{\xi}_1, ..., \boldsymbol{\xi}_S$ from the posterior $P(\boldsymbol{\xi}|y_{1:n}) \propto P(y_{1:n}|\boldsymbol{\xi})P(\boldsymbol{\xi})$, where the data likelihood is given as $P(y_{1:n}|\boldsymbol{\xi}) = \prod_{t=1}^{n} \mathcal{N}(y_t; f(t, \boldsymbol{\xi}), \sigma^2)$. These samples then yield probabilistic extrapolations of the learning curve for future time steps m, with mean predictions $\mathbb{E}[y_m|y_{1:n}] \approx \frac{1}{S}\sum_{s=1}^{S} f(m, \boldsymbol{\xi}_s)$. The ability to include arbitrary parametric functions make this model very flexible, and Domhan et al. [5] used it successfully to terminate evaluations of poorly-performing hyperparameters early for various different network architectures (thereby speeding up Bayesian optimization by a factor of two). However, the model's major disadvantage is that it does not take into account the used hyperparameters at all and therefore can only make useful predictions after observing a substantial initial fraction of the learning curve.

Predicting learning curves with Bayesian neural networks. Intuitively, similar hyperparameter configurations should lead to similar learning curves, and modelling this dependence would allow us to predict learning curves for new configurations without the need to observe their initial performance. Swersky et al. [4] followed this approach based on an approximate Gaussian process model; here, we formulate the problem using Bayesian neural networks. We aim to model the validation loss $f(\boldsymbol{x}, t)$ of a configuration $\boldsymbol{x} \in \mathcal{X} \subset \mathbb{R}^d$ at time step $t \in [1, T]$ based on noisy observations $y(\boldsymbol{x}, t) \sim \mathcal{N}(f(\boldsymbol{x}, t), \sigma^2)$. Here, we approximate f by the output of a neural network $\hat{f}(\boldsymbol{x}, t, W)$ with parameters W. For each configuration $\boldsymbol{x}$ trained for $T_{\boldsymbol{x}}$ time steps, we obtain $T_{\boldsymbol{x}}$ data points for our model; denoting the combined data by $\mathcal{D} = \{(\boldsymbol{x}_1, t_1, y_1), \ldots, (\boldsymbol{x}_n, t_n, y_n)\}$, we can then write the joint probability of the data and the model parameters as $p(\mathcal{D}, W) = p(W)p(\sigma^2)\prod_{i=1}^{|\mathcal{D}|} \mathcal{N}(y_i|\hat{f}(\boldsymbol{x}_i, t_i, W), \sigma^2)$. It is intractable to compute the posterior weight distribution $p(W|\mathcal{D})$, but we can use MCMC to sample it, in particular stochastic gradient MCMC methods, such as SGLD [13] or SGHMC [14]. Given M samples $W^1, \ldots, W^M$, we can then obtain the mean and variance of the predictive distribution as $\mu(\hat{y}(\boldsymbol{x}, t)|\mathcal{D}) = \frac{1}{M}\sum_{i=1}^{M} \hat{y}(\boldsymbol{x}, t; W^i)$ and $\sigma^2(\hat{y}(\boldsymbol{x}, t)|\mathcal{D}) = \frac{1}{M}\sum_{i=1}^{M} \left(\hat{y}(\boldsymbol{x}, t; W^i) - \mu(\hat{y}(\boldsymbol{x}, t)|\mathcal{D})\right)^2$, respectively. This is exactly the model that Springenberg et al. [12] used for (blackbox) Bayesian optimization with Bayesian neural networks; the only difference is in the input to the model: here, there is a data point for every time step of the curve, whereas Springenberg et al. [12] only used a single data point per curve (for its final time step).

Combining the two approaches. We can combine the approach of Domhan et al. [5] with Bayesian neural networks to not only take similarities between hyperparameter configurations into account but to also exploit the prior knowledge of the shape of learning curves. Instead of obtaining the parameters $\boldsymbol{\xi}$ by sampling from the posterior, we use a Bayesian neural network to learn a mapping $f : \mathcal{X} \to \boldsymbol{\Theta}$ from a hyperparameter configuration $\boldsymbol{x}$ to the parameters $\boldsymbol{\theta}$ of a parametric function model. The prediction for a hyperparameter configuration $\boldsymbol{x}$ at time t is then given by the convex combination $\hat{y}(\boldsymbol{x}, t) = \sum_{i=1}^{k} w_i \phi_i(\boldsymbol{\theta}_i(\boldsymbol{x}), t)$. These parametric functions are implemented as an additional layer in our neural network and we can sample the weights $w_1, ..., w_k$ and the noise σ^2 together with all other parameters of our neural network via stochastic gradient MCMC. Our architecture is illustrated in Figure 1.

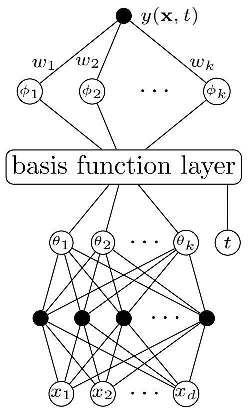

Figure 1: Our neural network architecture for learning curve prediction.

3 Preliminary Experiments

For our empirical evaluation we sampled 40 random configurations of a convolutional neural network (CNN) where each configuration was trained for 40 epochs on the CIFAR10 benchmark [15]. We also sampled 200 configurations for each of a fully connected network (FCNet), logistic regression (LR) and a variational autoencoder (VAE) all trained on MNIST [16]. The fully connected network and the logistic regression were trained for 100 epochs while the variation autoencoder was trained for 300 epochs.

We compare our new learning curve Bayesian neural network (denoted with LC in Figure 2) to a standard Bayesian neural network without our learning curve layer. For both networks we used a

Method	CNN		FCNet		LR		VAE	
	MSE	ALL	MSE	ALL	MSE	ALL	MSE	ALL
Sgld	0.039 ± 0.038	-16.231 ± 10.2	0.054 ± 0.031	-11.153 ± 3.9	0.009 ± 0.005	-0.590 ± 1.4	0.0004 ± 0.0003	1.901 ± 1.1
Sgld-LC	0.032 ± 0.038	-5.720 ± 8.5	0.032 ± 0.022	-2.165 ± 2.5	0.009 ± 0.006	0.807 ± 0.6	0.0003 ± 0.0003	2.692 ± 0.4
Sghmc	0.047 ± 0.058	-0.585 ± 2.9	0.038 ± 0.022	-0.816 ± 1.6	0.011 ± 0.009	0.883 ± 0.5	0.0004 ± 0.0003	2.233 ± 0.4
Sghmc-LC	0.017 ± 0.021	-0.001 ± 2.6	0.031 ± 0.026	-0.657 ± 2.0	0.008 ± 0.005	1.146 ± 0.4	0.0003 ± 0.0002	2.701 ± 0.4

Table 1: In each column we report the mean squared error (MSE) and the average loglikelihood (ALL) of the 10 fold CV learning curve prediction for a neural network without learning curve prediction layer (Sgld and Sghmc) and with our new layer (Sgld-LC and Sghmc-LC).

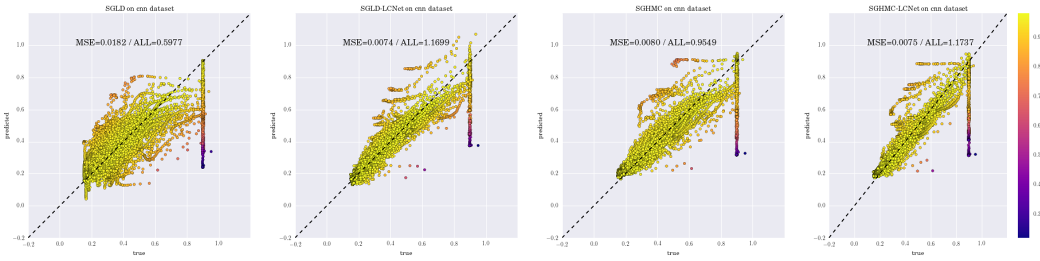

Figure 2: On the horizontal axis we plot the true value and on the vertical axis the predicted values. Each point is colored by its loglikelihood (the brighter the higher).

3 layer architecture with tanh activations and 50 units per layer. We compared the networks based on two different sampling methods: stochastic gradient Langevin dynamics (SGLD) and stochastic gradient Hamiltonian MCMC (SGHMC), following the approach of Springenberg et al. [12] to automatically adapt the noise estimate and the preconditoning of the gradients.

Table 1 shows the mean squared error and the average log-likelihood averaged over the 10 folds. We make two observations: firstly, our learning curve layer improved both mean predictions and likelihoods, for both SGLD and SGHMC. Secondly, SGHMC performed better than SGLD; the latter predicted too small variances, which we attribute to its random walk behavior. Figure 2 visualizes the results for our CNN dataset in more detail.

4 Conclusion

We introduced a new method for using Bayesian neural networks to model learning curves of iterative machine learning methods, such as convolutional neural networks (CNNs). In future work, we will evaluate our method on other types of learning curves and evaluate how well it estimates the asymptotic values of partially-observed learning curves. Ultimately, we aim to develop a reliable probabilistic model for predicting the performance of a given hyperparameter configuration, trained for a given number of steps on a subset of the training data of given size, and to use this model inside of a parallel Bayesian optimization method to find good CNN hyperparameter settings automatically and orders of magnitude faster than possible to date. We believe that this is an exciting challenge for the area of Bayesian (deep) neural networks.

References

[1] E. Brochu, V. Cora, and N. de Freitas. A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *CoRR*, 2010.

[2] J. Snoek, H. Larochelle, and R. P. Adams. Practical Bayesian optimization of machine learning algorithms. In *Proc. of NIPS'12*, 2012.

[3] B. Shahriari, K. Swersky, Z. Wang, R. Adams, and N. de Freitas. Taking the human out of the loop: A Review of Bayesian Optimization. *Proc. of the IEEE*, (1), 12/2015 2016.

[4] K. Swersky, J. Snoek, and R. Adams. Freeze-thaw Bayesian optimization. *CoRR*, 2014.

[5] T. Domhan, J. T. Springenberg, and F. Hutter. Speeding up automatic hyperparameter optimization of deep neural networks by extrapolation of learning curves. In Q. Yang and M. Wooldridge, editors, *Proc. of IJCAI'15*, 2015.

[6] A. Klein, S. Falkner, S. Bartels, P. Hennig, and F. Hutter. Fast bayesian optimization of machine learning hyperparameters on large datasets. *CoRR*, 2016.

[7] K. Swersky, J. Snoek, and R. Adams. Multi-task Bayesian optimization. In *Proc. of NIPS'13*, 2013.

[8] M. Feurer, T. Springenberg, and F. Hutter. Initializing Bayesian hyperparameter optimization via meta-learning. In *Proc. of AAAI'15*, 2015.

[9] F. Hutter, H. Hoos, K. Leyton-Brown, and K. Murphy. Time-bounded sequential parameter optimization.

[10] F. Hutter, H. Hoos, and K. Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In *LION'11*, 2011.

[11] J. Snoek, O. Rippel, K. Swersky, R. Kiros, N. Satish, N. Sundaram, M. M. A. Patwary, Prabhat, and R. P. Adams. Scalable Bayesian optimization using deep neural networks. In *Proc. of ICML'15*, 2015.

[12] J. Springenberg, A. Klein, S. Falkner, and F. Hutter. Bayesian optimization with robust bayesian neural networks. In *Proc. of NIPS'16*, 2016.

[13] M. Welling and Y. Teh. Bayesian learning via stochastic gradient Langevin dynamics. 2011.

[14] T. Chen, E.B. Fox, and C. Guestrin. Stochastic gradient Hamiltonian Monte Carlo. In *Proc. of ICML'14*, 2014.

[15] A. Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

[16] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. In S. Haykin and B. Kosko, editors, *Intelligent Signal Processing*. IEEE Press, 2001. URL `http://www.iro.umontreal.ca/~lisa/pointeurs/lecun-01a.pdf`.