Deeper Connections between Neural Networks and Gaussian Processes Speed-up Active Learning

Evgenii Tsymbalov [1] **Sergei Makarychev** [1] **Alexander Shapeev** [1] **Maxim Panov** [1]

Abstract

Active learning methods for neural networks are usually based on greedy criteria, which require retraining the neural network after adding each data batch. This is computationally inefficient; moreover, uncertainty estimates for neural networks sometimes are overconfident for points lying far from the training sample. In this work, we propose to approximate Bayesian neural networks (BNN) by Gaussian processes, which allows us to update the uncertainty estimates of predictions efficiently without retraining the neural network while avoiding overconfident uncertainty prediction for out-of-sample points. In a series of experiments on real-world data including large-scale problems of chemical and physical modeling, we show the superiority of the proposed approach over the state-of-the-art methods.

1. Introduction

In the modern applications of machine learning, especially in engineering design, physics, chemistry, molecular and materials modeling, the datasets are usually of limited size due to the expensive cost of computations. On the other side, such applications typically allow for the calculation of additional data at points chosen by the experimentalist. Thus, there is a need to get the highest possible approximation quality with as few function evaluations as possible. *Active learning* methods help to achieve this goal by trying to select the best candidates for further target function computation using the already existing data and machine learning models based on these data.

Modern active learning methods are usually based either on ensemble-based methods (Settles, 2012) or on probabilistic models such as Gaussian process (GP) regression (Sacks et al., 1989; Burnaev & Panov, 2015). However, the models from these classes often do not give state-of-the-art results in the downstream tasks. For example, GP-based models have very high computational complexity, which in many cases prohibits their usage even for moderate-sized datasets. Moreover, GP training is usually done for stationary covariance functions, which is a very limiting assumption in many real-world scenarios. Thus, active learning methods applicable to the broader classes of models are needed.

In this work, we aim at developing efficient active learning strategies for neural network models. Unlike Gaussian processes, neural networks are known for relatively good scalability with the dataset size and are very flexible in terms of dependencies they can model. However, active learning for neural networks is currently not very well developed, especially for tasks other than classification. Recently some uncertainty estimation and active learning approaches were introduced for Bayesian neural networks (Gal et al., 2017; Hafner et al., 2018). However, still one of the main complications is in designing active learning approaches which allow points sampling without retraining the neural network too often. Recently, (Matthews et al., 2018) proved that deep neural networks with random weights converge to Gaussian processes in the infinite layer width limit. Our approach aims to approximate trained Bayesian neural networks and show that obtained uncertainty estimates enable to improve active learning performance significantly.

We summarize the main contributions of the paper as follows. (i) We propose to compute the approximation of trained Bayesian neural network with Gaussian process, which allows using the Gaussian process machinery to calculate uncertainty estimates for neural networks. (ii) We propose an active learning strategy based on obtained uncertainty estimates which significantly speed-up the active learning with NNs by improving the quality of selected samples and decreasing the number of times the NN is retrained. The proposed framework shows significant improvement over competing approaches in a range of real-world problems, including the cutting edge applications in chemoinformatics.

In the next section, we discuss the problem statement and the proposed approach in detail.

[1]Skolkovo Institute of Science and Technology (Skoltech), Moscow, Russia. Correspondence to: Evgenii Tsymbalov <e.tsymbalov@skoltech.ru>.

Presented at the ICML 2019 Workshop on Uncertainty and Robustness in Deep Learning.

2. Active learning with Bayesian neural networks and beyond

2.1. Problem statement

We consider a regression problem with an unknown function $f(\mathbf{x})$ defined on a subset of Euclidean space $\mathcal{X} \subset \mathbb{R}^d$, where an approximation function $\hat{f}(\mathbf{x})$ should be constructed based on noisy observations: $y = f(\mathbf{x}) + \epsilon$ with ϵ being some random noise.

We focus on *active learning* scenario which allows to iteratively enrich training set by computing the target function in design points specified by the experimenter. More specifically, we assume that the initial training set $D_{\text{init}} = \{\mathbf{x}_i, y_i = f(\mathbf{x}_i) + \epsilon_i\}_{i=1}^{N}$ with precomputed function values is given. On top of it, we are given another set of points called "pool" $\mathcal{P} = \{\mathbf{x}_j \in \mathcal{X}\}_{j=1}^{N_{\text{pool}}}$, which represents unlabelled data. Each point $\mathbf{x} \in \mathcal{P}$ can be annotated by computing $y = f(\mathbf{x}) + \epsilon$ so that the pair $\{\mathbf{x}, y\}$ is added to the training set.

The standard active learning approaches rely on the greedy point selection

$$\mathbf{x}_{new} = \arg\max_{\mathbf{x} \in \mathcal{P}} A(\mathbf{x} \,|\, \hat{f}, D), \tag{1}$$

where $A(\mathbf{x} \,|\, \hat{f}, D)$ is so-called *acquisition function*, which is usually constructed based on the current training set D and corresponding approximation function $\hat{f}$.

The most popular choice of acquisition function is the variance $\hat{\sigma}^2(\mathbf{x} \,|\, \hat{f}, D)$ of the prediction $\hat{f}(\mathbf{x})$, which can be easily estimated in some machine learning models such as Random forest (Mentch & Hooker, 2016) or Gaussian process regression (Rasmussen, 2004). However, for many types of models (e.g., neural networks) the computation of prediction uncertainty becomes a nontrivial problem.

2.2. Uncertainty estimation and active learning with Bayesian neural networks

In this work, we will stick to the Bayesian approach which treats neural networks as probabilistic models $p(y \,|\, \mathbf{x}, \mathbf{w})$. Vector of neural network weights $\mathbf{w}$ is assumed to be a random variable with some prior distribution $p(\mathbf{w})$. The likelihood $p(y \,|\, \mathbf{x}, \mathbf{w})$ determines the distribution of network output at a point $\mathbf{x}$ given specific values of parameters $\mathbf{w}$. There is vast literature on training Bayesian networks (see (Graves, 2011) and (Paisley et al., 2012) among many others), which mostly targets the so-called variational approximation of the intractable posterior distribution $p(\mathbf{w} \,|\, D)$ by some easily computable distribution $q(\mathbf{w})$.

The approximate posterior predictive distribution reads as:

$$q(y \,|\, \mathbf{x}) = \int p(y \,|\, \mathbf{x}, \mathbf{w})\, q(\mathbf{w})\, d\mathbf{w}.$$

The simple way to generate random values from this distribution is to use the Monte-Carlo approach, which allows estimating the mean:

$$\mathbb{E}_{q(y \,|\, \mathbf{x})}\, y \approx \frac{1}{T} \sum_{t=1}^{T} \hat{f}(\mathbf{x}, \mathbf{w}_t),$$

where the weight values $\mathbf{w}_t$ are i.i.d. random variables from distribution $q(\mathbf{w})$. Similarly, one can use Monte-Carlo to estimate the approximate posterior variance $\hat{\sigma}^2(\mathbf{x} \,|\, \hat{f})$ of the prediction y at a point $\mathbf{x}$ and use it as an acquisition function:

$$A(\mathbf{x} \,|\, \hat{f}, D) = \hat{\sigma}^2(\mathbf{x} \,|\, \hat{f}).$$

We note that the considered acquisition function formally doesn't depend on the dataset D except for the fact that D was used for training the neural network $\hat{f}$.

Let us note that the general greedy active learning approach (1) by design gives one candidate point per active learning iteration. If one tries to use (1) to obtain several samples with the same acquisition function, it usually results in obtaining several nearby points from the same region of design space.Such behaviour is typically undesirable as nearby points are likely to have very similar information about the target function. Moreover, neural network uncertainty predictions are sometimes overconfident in out-of-sample regions of design space.

There are several classic approaches to overcome these issues each having its drawbacks, see Appendix A for details. In the next section, we propose the approach to elude similar problems by considering the Gaussian process approximation of the neural network.

2.3. Gaussian process approximation of Bayesian neural network

Effectively, the random function $\hat{f}(\mathbf{x}, \mathbf{w}) = \mathbb{E}_{p(y \,|\, \mathbf{x}, \mathbf{w})}\, y$ is the stochastic process indexed by $\mathbf{x}$. The covariance function of the process $\hat{f}(\mathbf{x}, \mathbf{w})$ is given by $k(\mathbf{x}, \mathbf{x}') = \mathbb{E}_{q(\mathbf{w})}\big(\hat{f}(\mathbf{x}, \mathbf{w}) - m(\mathbf{x})\big)\big(\hat{f}(\mathbf{x}', \mathbf{w}) - m(\mathbf{x}')\big)$, where $m(\mathbf{x}) = \mathbb{E}_{q(\mathbf{w})}\hat{f}(\mathbf{x}, \mathbf{w})$.

As was shown in (Matthews et al., 2018; Lee et al., 2017) neural networks with random weights converge to Gaussian processes in the infinite layer width limit. However, one is not limited to asymptotic properties of purely random networks as Bayesian neural networks trained on real-world

data exhibit near Gaussian behaviour, see the example on Appendix B.

We aim to make the Gaussian process approximation $\hat{g}(\mathbf{x} \mid \hat{f})$ of the stochastic process $\hat{f}(\mathbf{x}, \mathbf{w})$ and compute its posterior variance $\hat{\sigma}^2(\mathbf{x} \mid \hat{f}, X)$ given the set of anchor points $X = \{\mathbf{x}_i\}_{i=1}^N$. Typically, X is a subset of the training sample. Given X, Monte-Carlo estimates $\hat{k}(\mathbf{x}', \mathbf{x}'')$ of the covariance function $k(\mathbf{x}', \mathbf{x}'')$ for every pair of points $\mathbf{x}', \mathbf{x}'' \in X \cup \mathbf{x}$ allow computing

$$\hat{\sigma}^2(\mathbf{x} \mid \hat{f}, X) = \hat{k}(\mathbf{x}, \mathbf{x}) - \hat{\mathbf{k}}^{\mathrm{T}}(\mathbf{x}) \hat{K}^{-1} \hat{\mathbf{k}}(\mathbf{x}), \qquad (2)$$

where $\hat{K} = \left[\hat{k}(\mathbf{x}_i, \mathbf{x}_j)\right]_{i,j=1}^N$ and $\hat{\mathbf{k}}(\mathbf{x}) = \left(\hat{k}(\mathbf{x}_1, \mathbf{x}), \ldots, \hat{k}(\mathbf{x}_N, \mathbf{x})\right)^{\mathrm{T}}$.

We note that only the trained neural network $\hat{f}(\mathbf{x}, \mathbf{w})$ and the ability to sample from the distribution $q(\mathbf{w})$ are needed to compute $\hat{\sigma}^2(\mathbf{x} \mid \hat{f}, X)$.

2.4. Active learning strategies

The benefits of the Gaussian process approximation and the usage of the formula (2) are not evident as one might directly estimate the variance of neural network prediction $\hat{f}(\mathbf{x}, \mathbf{w})$ at any point $\mathbf{x}$ by sampling from $q(\mathbf{w})$ and use it as acquisition function. However, the approximate posterior variance $\hat{\sigma}^2(\mathbf{x} \mid \hat{f}, X)$ of Gaussian process $\hat{g}(\mathbf{x} \mid \hat{f})$ has an important property of having large values for points $\mathbf{x}$ lying far from the points from the training set X. Thus, out-of-sample points are likely to be selected by the active learning procedure.

Moreover, the function $\hat{\sigma}^2(\mathbf{x} \mid \hat{f}, X)$ depends solely on covariance function values for points from a set X (and not on the output function values). Such property allows updating uncertainty predictions by just adding sample points to the set X. More specifically, if we decide to sample some point $\mathbf{x}'$, then the updated posterior variance $\hat{\sigma}^2(\mathbf{x} \mid \hat{f}, X')$ for $X' = X \cup \mathbf{x}'$ can be easily computed:

$$\hat{\sigma}^2(\mathbf{x} \mid \hat{f}, X') = \hat{\sigma}^2(\mathbf{x} \mid \hat{f}, X) - \frac{\hat{k}^2(\mathbf{x}, \mathbf{x}' \mid \hat{f}, X)}{\hat{\sigma}^2(\mathbf{x}' \mid \hat{f}, X)},$$

where $\hat{k}(\mathbf{x}, \mathbf{x}' \mid \hat{f}, X) = \hat{k}(\mathbf{x}, \mathbf{x}') - \hat{\mathbf{k}}^{\mathrm{T}}(\mathbf{x}) \hat{K}^{-1} \hat{\mathbf{k}}(\mathbf{x}')$ is the posterior covariance function of the process $\hat{g}(\mathbf{x} \mid \hat{f})$ given X.

Importantly, $\hat{\sigma}^2(\mathbf{x} \mid \hat{f}, X')$ for points $\mathbf{x}$ in some vicinity of $\mathbf{x}'$ will have low values, which guarantees that further sampled points will not lie too close to $\mathbf{x}'$ and other points from the training set X. The resulting NNGP active learning procedure is depicted in Figure 1.

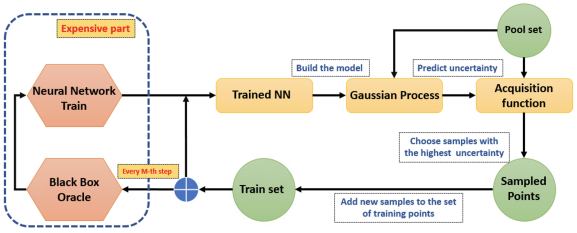

Figure 1. Schematic representation of the NNGP approach to active learning. GP is fitted on the data from a stochastic output of NN, and the posterior variance of GP is used as an acquisition function for sampling. The most computationally expensive part (function evaluation at sampled points and neural network retraining) is done only every M steps of sampling, while all the intermediate iterations are based solely on trained neural network and corresponding GP approximation.

3. Experiments

Our experimental study is focused on real-world data to ensure that the algorithms are indeed useful in real-world scenarios. We compare the following algorithms (i) pure random sampling, (ii) sampling based on the variance of NN stochastic output from $\hat{f}$, which we refer to as MCDUE (see (Gal et al., 2017; Tsymbalov et al., 2018)), and (iii) proposed GP-based approaches (NNGP).

Following (Gal & Ghahramani, 2016) we use the Bayesian NN with the Bernoulli distribution on the weights, which is equivalent to using the dropout on the inference stage.

For details of the experiments, please refer to Appendix D.

3.1. Airline delays dataset and NCP comparison

We start the experiments from comparing the proposed approach to active learning with the one based on uncertainty estimates obtained from a Bayesian neural network with *Noise Contrastive Prior (NCP)*, see (Hafner et al., 2018). Following this paper, we use the airline delays dataset (see (Hensman et al., 2013)) and NN consisting of two layers with 50 neurons each, leaky ReLU activation function, and trained with respect to NCP-based loss function.

The results for the test set are shown in Figure 2. The proposed NNGP approach demonstrates a comparable error with respect to the previous results and outperforms (on average) other methods in the continuous active learning scenario.

3.2. Experiments on UCI datasets

We conducted a series of experiments with active learning performed on the data from the UCI ML repository (Dua & Taniskidou, 2017). All the datasets represent real-world regression problems with 15+ dimensions and 30000+ sam-

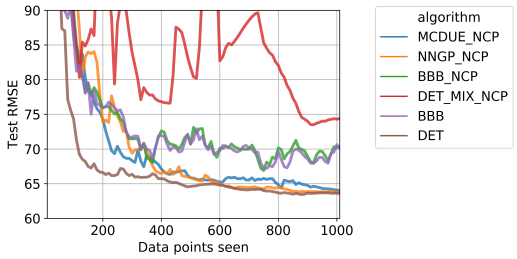

Figure 2. Root mean squared errors as functions of active learning iteration for different methods on the Airline delays data set. Plots show the median of the errors over 25 runs.

ples, see Table 1 for details. To compare the performance of the algorithms across the different datasets and different choices of training samples, we will use so-called *Dolan-More curves*, see Appendix E for details.

The Dolan-More curves for the errors of approximation for considered problems after the 16th iteration of the active learning procedure are presented in Figure 3. We see that the NNGP procedure is superior in terms of RMSE compared to MCDUE and random sampling.

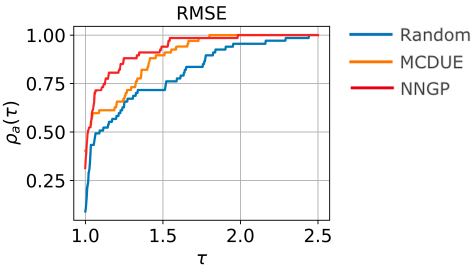

Figure 3. Dolan-More curves for UCI datasets and different active learning algorithms after 16 active learning iterations. Root mean squared error (RMSE) on the independent test set is considered. NNGP-based algorithm shows better performance compared to MCDUE and random sampling.

3.3. SchNet training

To demonstrate the power of our approach, we conducted a series of numerical experiments with the state-of-the-art neural network architecture in the field of chemoinformatics "SchNet" (Schütt et al., 2017). This network takes information about an organic molecule as an input, and, after special preprocessing and complicated training procedure, outputs some properties of the molecule (like energy). Despite its complex structure, SchNet contains fully connected layers, so it is possible to use a dropout in between them.

We tested our approach on the problem of predicting the internal energy of the molecule at 0K from the QM9 data set (Ramakrishnan et al., 2014). The results are shown in Figure 4. The NNGP approach demonstrates the most steady decrease in error, with the 25% accuracy increase in RMSE. Such improvement is very significant in terms of the time savings for the computationally expensive quantum-mechanical calculations. For example, to reach the RMSE of 2 kcal/mol starting from the SchNet trained on 10 000 molecules, one need to additionally sample 15 000 molecules in case of random sampling or just 7 500 molecules using the NNGP uncertainty estimation procedure.

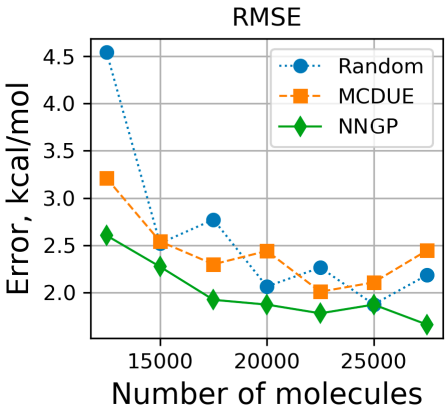

Figure 4. Training curves for the active learning scenario for SchNet: starting from 10 000 random molecules pick 2 500 based on the uncertainty estimate. NNGP-based algorithm results in 25% decrease in RMSE. Simple dropout-based approach (MCDUE) does not demonstrate a difference from the random sampling in terms of accuracy.

4. Summary and discussion

We have proposed a novel dropout-based method for the uncertainty estimation for deep neural networks, which uses the approximation of a neural network by Gaussian process. Experiments on different architectures and real-world problems show that the proposed estimate achieves state-of-the-art results in the context of active learning. Importantly, the proposed approach works for any neural network architecture involving dropout (as well as other Bayesian neural networks), so it can be applied to a very wide range of networks and problems without a need to change neural network architecture.

It is of interest whether the proposed approach can be efficient for other applications areas, such as image classification. We also plan to study the applicability of modern methods for GP speed-up in order to improve the scalability of proposed approach.

ACKNOWLEDGEMENTS

E.T. and A.S. were supported by the Skoltech NGP Program No. 2016-7/NGP (a Skoltech-MIT joint project).

References

Bertin-Mahieux, T., Ellis, D. P., Whitman, B., and Lamere, P. The million song dataset. In *Ismir*, volume 2, pp. 10, 2011.

Burnaev, E. and Panov, M. Adaptive design of experiments based on gaussian processes. In *International Symposium on Statistical Learning and Data Sciences*, pp. 116–125. Springer, 2015.

Buza, K. Feedback prediction for blogs. In *Data analysis, machine learning and knowledge discovery*, pp. 145–152. Springer, 2014.

Dua, D. and Taniskidou, E. K. Uci machine learning repository [http://archive. ics. uci. edu/ml]. 2017.

Fernandes, K., Vinagre, P., and Cortez, P. A proactive intelligent decision support system for predicting the popularity of online news. In *Portuguese Conference on Artificial Intelligence*, pp. 535–546. Springer, 2015.

Gal, Y. and Ghahramani, Z. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *Proc. ICML'16*, pp. 1050–1059, 2016.

Gal, Y., Islam, R., and Ghahramani, Z. Deep bayesian active learning with image data. *arXiv:1703.02910*, 2017.

Graf, F., Kriegel, H.-P., Schubert, M., Pölsterl, S., and Cavallaro, A. 2d image registration in ct images using radial image descriptors. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pp. 607–614. Springer, 2011.

Graves, A. Practical variational inference for neural networks. In *Advances in neural information processing systems*, pp. 2348–2356, 2011.

Hafner, D., Tran, D., Irpan, A., Lillicrap, T., and Davidson, J. Reliable uncertainty estimates in deep neural networks using noise contrastive priors. *arXiv:1807.09289*, 2018.

Hensman, J., Fusi, N., and Lawrence, N. D. Gaussian processes for big data. *arXiv:1309.6835*, 2013.

Lee, J., Bahri, Y., Novak, R., Schoenholz, S. S., Pennington, J., and Sohl-Dickstein, J. Deep neural networks as gaussian processes. *arXiv:1711.00165*, 2017.

Matthews, A. G. d. G., Rowland, M., Hron, J., Turner, R. E., and Ghahramani, Z. Gaussian process behaviour in wide deep neural networks. *arXiv:1804.11271*, 2018.

Mentch, L. and Hooker, G. Quantifying uncertainty in random forests via confidence intervals and hypothesis tests. *The Journal of Machine Learning Research*, 17(1): 841–881, 2016.

Nugteren, C. and Codreanu, V. Cltune: A generic auto-tuner for opencl kernels. In *Embedded Multicore/Many-core Systems-on-Chip (MCSoC), 2015 IEEE 9th International Symposium on*, pp. 195–202. IEEE, 2015.

Paisley, J., Blei, D. M., and Jordan, M. I. Variational bayesian inference with stochastic search. In *Proc. ICML'12*, pp. 1363–1370, 2012. ISBN 978-1-4503-1285-1. URL http://dl.acm.org/citation.cfm?id=3042573.3042748.

Ramakrishnan, R., Dral, P. O., Rupp, M., and Von Lilienfeld, O. A. Quantum chemistry structures and properties of 134 kilo molecules. *Scientific data*, 1:140022, 2014.

Rasmussen, C. E. Gaussian processes in machine learning. In *Advanced lectures on machine learning*, pp. 63–71. Springer, 2004.

Sacks, J., Welch, W. J., Mitchell, T. J., and Wynn, H. P. Design and analysis of computer experiments. *Statistical science*, pp. 409–423, 1989.

Schütt, K. T., Arbabzadah, F., Chmiela, S., Müller, K. R., and Tkatchenko, A. Quantum-chemical insights from deep tensor neural networks. *Nature communications*, 8: 13890, 2017.

Settles, B. Active learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 6(1):1–114, 2012.

Shannon, P., Markiel, A., Ozier, O., Baliga, N. S., Wang, J. T., Ramage, D., Amin, N., Schwikowski, B., and Ideker, T. Cytoscape: a software environment for integrated models of biomolecular interaction networks. *Genome research*, 13(11):2498–2504, 2003.

Tsymbalov, E., Panov, M., and Shapeev, A. Dropout-based active learning for regression. In *International Conference on Analysis of Images, Social Networks and Texts*, pp. 247–258. Springer, 2018.

A. Classic solutions to greedy sampling problem

- One may retrain the model after each point addition which may result in a significant change of the acquisition function and lead to the selection of a more diverse set of points. However, such an approach is usually very computationally expensive, especially for neural network-based models.

- One may try to add distance-based heuristic, which explicitly prohibits sampling points which are very close to each other and increase values of acquisition function for points positioned far from the training sample. Such an approach may give satisfactory results in some cases, however usually requires fine-tuning towards particular application (like the selection of specific distance function or choice the parameter value which determines whether two points are near or not), while its performance may degrade in high dimensional problems.

- One may treat specially normalized vector of acquisition function values at points from the pool as a probability distribution and sample the desired number of points based on their probabilities (the higher acquisition function value, the point is more likely to be selected). This approach usually improves over greedy baseline procedure. However, it still gives many nearby points.

B. Near-Gaussian behaviour of stochastic NN estimates

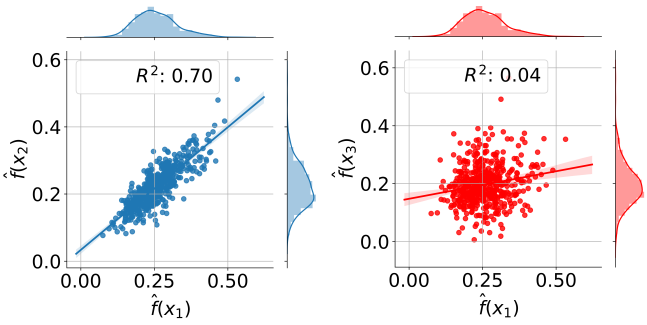

Figure 5. Bivariate distribution plots for the stochastic NN output at points $\mathbf{x}_1, \mathbf{x}_2$ and $\mathbf{x}_3$, where $\mathbf{x}_1$ is much closer to $\mathbf{x}_2$ in feature space than to $\mathbf{x}_3$. Both univariate and bivariate distributions are Gaussian-like, while the correlation between function values is much higher for closer points.

C. Details on active learning approaches

Algorithm 1 MCDUE

Require: Number of samples to generate N_s, pool $\mathcal{P}$, neural network model $\hat{f}(\mathbf{x}, \mathbf{w})$ and dropout probability π.
Ensure: Set of points $X_s \subset \mathcal{P}$ with $|X_s| = N_s$.
for each sample $\mathbf{x}_j$ from the pool $\mathcal{P}$ **do**
  for $t = 1, \ldots, T$ **do**
    $\omega_t \sim \text{Bern}(\pi)$.
    $\mathbf{w}_t = \hat{\mathbf{w}}(\omega_t)$.
    $y_t = \hat{f}_t(\mathbf{x}_j) = \hat{f}(\mathbf{x}_j, \mathbf{w}_t)$.
  end for
  Calculate the variance:

$$\hat{\sigma}_j^2 = \frac{1}{T-1}\sum_{t=1}^{T}(y_t - \bar{y})^2, \; \bar{y} = \frac{1}{T}\sum_{t=1}^{T} y_t.$$

end for
Return N_s points from pool $\mathcal{P}$ with largest values of the variance $\hat{\sigma}_j^2$.

Algorithm 2 NNGP

Require: Number of samples to generate N_s, pool $\mathcal{P}$, the set X_* of inducing points for Gaussian process model, neural network model $\hat{f}(\mathbf{x}, \mathbf{w})$, dropout probability π and regularization parameter λ.
Ensure: Set of points $X_s \subset \mathcal{P}$ with $|X_s| = N_s$.
for $t = 1, \ldots, T$ **do**
  $\omega_t \sim \text{Bern}(\pi)$.
  $\mathbf{w}_t = \hat{\mathbf{w}}(\omega_t)$.
  $y_t^i = \hat{f}(\mathbf{x}_i, \mathbf{w}_t)$ for each $\mathbf{x}_i \in \mathcal{P}$.
  $z_t^j = \hat{f}(\mathbf{x}_j, \mathbf{w}_t)$ for each $\mathbf{x}_j \in X_*$.
end for
Calculate the covariance matrix $\hat{K} = \left[cov(z^i, z^j)\right]_{i,j=1}^{N}$.
for each $\mathbf{x}_j \in \mathcal{P}$ **do**
  $\hat{\mathbf{k}}_j = \left[cov(z^i, y^j)\right]_{i=1}^{N}$.
  $v_j = var(y^j)$.
  $\hat{\sigma}_j^2 = v_j - \hat{\mathbf{k}}_j^{\mathrm{T}}(\hat{K} + \lambda I)^{-1}\hat{\mathbf{k}}_j$.
end for
Return N_s points from pool $\mathcal{P}$ with largest values of the variance $\hat{\sigma}_j^2$.

Algorithm 3 M-step NNGP

Require: Number of samples to generate N_s, number of samples per active learning iteration M, pool $\mathcal{P}$, the set X_* of inducing points for Gaussian process model, neural network model $\hat{f}(\mathbf{x}, \mathbf{w})$, dropout probability π and regularization parameter λ.
Ensure: Set of points $X_s \subset \mathcal{P}$ with $|X_s| = N_s$.
1: Initialize sets $X_s := \emptyset, \mathcal{P}^* := \mathcal{P}$.
2: **for** $m = 1, \ldots, M$ **do**
3: Run NNGP procedure with parameters $N_s/M, \mathcal{P}^*$, $X_* \cup X_s, \hat{f}, \pi, \lambda$, which returns a set of points X_o.
4: $X_s = X_s \cup X_o$.
5: $\mathcal{P}^* = \mathcal{P}^* \setminus X_o$.
6: **end for**

D. Experiment and data details

D.1. Airline delays dataset

We took a random subset of 50,000 data samples from the data available on the January – April of 2008 as a training set and we chose 100,000 random data samples from May of 2008 as a test set. We used the following variables as input features *PlaneAge, Distance, CRSDepTime, AirTime, CRSArrTime, DayOfWeek, DayofMonth, Month* and *ArrDelay + DepDelay* as a target.

D.2. UCI datasets

For every experiment, data are shuffled and split in the following proportions: 10% for the training set D_{train}, 5% for the test set D_{test}, 5% for the validation set D_{val} needed for early-stopping and 80% for the pool $\mathcal{P}$.

We used a simple neural network with three hidden layers of sizes 256, 128 and 128. Learning rate started at 10^{-3}, its decay was set to 0.97 and changed every 50000 epochs. Minimal learning rate was set to 10^{-5}. We reset the learning rate for each active learning algorithm in the hope of beating the local minima problem. The training dropout rate was set to 0.1. L_2 regularization was set to 10^{-4}. Batch size set to 200. For each active learning iteration, we used an early stopping with patience equal to 3 and the number of mandatory initial epochs equal to 10000.

We performed 16 active learning iterations with 200 points picked at each iteration.

The summary of the datasets used in these experiment in provided in 1.

D.3. QM9 data (SchNet training)

In our experiment, we separate the whole dataset of 133 885 molecules into the initial set of 10 000 molecules, the test set of 5 000 molecules, and the rest of the data allocated as the pool. On each active learning iteration, we perform 100 000 training epochs and then calculate the uncertainty estimates using either MCDUE or NNGP approach. We then select 2 500 molecules with the highest uncertainty from the pool, add them to the training set and perform another active learning iteration.

We used a Tensorflow implementation of a SchNet with the same architecture as in original paper except for an increased size of hidden layers (from 64 and 32 units to 256 and 128 units, respectively) and dropout layer placed in between of them and turned on during an inference only.

Table 1. Summary of the datasets used in experiments with UCI data.

Dataset name	Samples	Attributes	Target
BlogFeedback (Buza, 2014)	60021	281	Number of comments
SGEMM GPU (Nugteren & Codreanu, 2015)	241600	18	Calculation time (median)
YearPredictionMSD (Bertin-Mahieux et al., 2011)	515345	90	Year
Relative location of CT slices (Graf et al., 2011)	53500	386	Relative location
Online News Popularity (Fernandes et al., 2015)	39797	61	Number of shares
KEGG Network (Shannon et al., 2003)	53414	24	Clustering coefficient

E. Dolan-More curves

Let q_a^p be an error measure of the a-th algorithm on the $\mathcal{P}$-th problem. Then, determining the performance ratio $r_a^p = \frac{q_a^p}{\min_x (q_x^p)}$, we can define the Dolan-More curve as a function of the performance ratio factor τ:

$$\rho_a(\tau) = \frac{\#(p\colon r_a^p \leq \tau)}{n_p},$$

where n_p is the total number of evaluations for the problem p. Thus, $\rho_a(\tau)$ defines the fraction of problems in which the a-th algorithm has the error not more than τ times bigger than the best competitor in the chosen performance metric. Note that $\rho_a(1)$ is the ratio of problems on which the a-th algorithm performance was the best, while in general, the higher curve means the better performance of the algorithm.