Learning to Estimate Point-Prediction Uncertainty and Correct Output in Neural Networks

Xin Qiu
Cognizant
qiuxin.nju@gmail.com

Elliot Meyerson
Cognizant
elliot.meyerson@cognizant.com

Risto Miikkulainen
Cognizant
The University of Texas at Austin
risto@cognizant.com

Abstract

Neural Networks (NNs) have been extensively used for a wide spectrum of real-world regression tasks, where the goal is to predict a numerical outcome such as revenue, effectiveness, or a quantitative result. In many such tasks, the point prediction is not enough, but also the uncertainty (i.e. risk, or confidence) of that prediction must be estimated. Standard NNs, which are most often used in such tasks, do not provide any such information. Existing approaches try to solve this issue by combining Bayesian models with NNs, but these models are hard to implement, more expensive to train, and usually do not perform as well as standard NNs. In this paper, a new framework called RIO is developed that makes it possible to estimate uncertainty in any pretrained standard NN. RIO models prediction residuals using Gaussian Process with a composite input/output kernel. The residual prediction and I/O kernel are theoretically motivated and the framework is evaluated in twelve real-world datasets. It is found to provide reliable estimates of the uncertainty, reduce the error of the point predictions, and scale well to large datasets. Given that RIO can be applied as a meta-step to any standard NN without modifications to model architecture or training pipeline, it provides an important ingredient in building real-world applications of NNs.

1 Introduction

Nowadays, Neural Networks (NNs) are arguably the most popular machine learning tool among Artificial Intelligence (AI) community. Researchers and practitioners have applied NNs to a wide variety of fields, including manufacturing [3], bioinformatics [15], physics [2], finance [19], chemistry [1], healthcare [22], etc. Although standard NNs are good at making a point prediction (a single outcome) for supervised learning tasks, they are unable to provide the uncertainty information about predictions. For real-world decision makers, representing the model uncertainty is of crucial importance [6, 9, 13]. For example, in the case of regression, providing a 95% confidence interval of the prediction allows the decision maker to anticipate the possible outcomes with explicit probability. In contrast, simply returning a single point prediction imposes increased risks on the decision making, e.g., a predictively good but actually risky medical treatment may be overconfidently interpreted without uncertainty information.

Conventional Bayesian models such as Gaussian Process (GP) [21] offer a mathematically grounded approach to reason about the predictive uncertainty, but usually come with a prohibitive computational cost and lessened performance compared to NNs [6]. As a potential solution, considerable research has been devoted to the combination of Bayesian models and NNs (see "Related Work" section for a detailed review of these approaches), aiming to overcome the downsides of both. However, from the classical Bayesian Neural Network [18], in which a distribution of weights is learned, to the most recent Neural Processes [7, 8, 12], in which a distribution over functions is defined, all these methods require significant modifications to the model infrastructure and training pipeline. Compared to standard (non-Bayesian) NNs, these new models are often computationally slower to train and harder to implement [14, 6], creating tremendous difficulty for practical uses. In [6], a theoretical tool is derived to extract uncertainty information from dropout training, however, the method can only be applied to dropout models, and it also requires to change the internal inference pipeline of dropout NNs. Quantifying point-prediction uncertainty in standard NNs, which are overwhelmingly popular in practical applications, still remains a challenging problem with significant potential impact.

To circumvent the above issues, this paper presents a new framework that can quantitatively estimate the point-prediction uncertainty of standard NNs without any modifications to the model structure or training pipeline. The new method can be directly applied to any pretrained NNs without retraining them. The main idea is to estimate the prediction residuals of NNs using a modified GP, which introduces a new composite kernel that utilizes both inputs and outputs of the NNs. The framework is referred to as RIO (for Residual Input/Output), and the new kernel as an I/O kernel. In addition to uncertainty estimation, RIO has an interesting and unexpected side-effect: It also provides a way to reduce the error of the NN predictions. Thus, RIO is a metalearning process that learns to characterize the behavior of existing learners, and improve their performance. Moreover, with the help of recent sparse GP models, RIO is well scalable to large datasets. Since classification problems can also be treated as regression on class labels [16], this paper will focus on regression tasks. A theoretical foundation is provided to prove the effectiveness of both residual estimation and I/O kernel. Twelve real-world datasets are tested in empirical studies, in which RIO exhibits reliable uncertainty estimations, more accurate point predictions, and better scalability compared to existing approaches.

2 The RIO Framework

In this section, the general problem statement will be given, the RIO framework will be developed, and justified theoretically, focusing on the two main contributions: estimating the residuals with GP and using an I/O kernel. For background introductions of NNs, GPs, and its more efficient approximation, Stochastic Variational Gaussian Processes (SVGPs) [10, 11], see section S2 in appendix.

Consider a training dataset $\mathcal{D} = (\mathcal{X}, \mathcal{Y}) = \{(\mathbf{x}_i, y_i) \mid i = 1, 2, \dots, n\}$, and a pre-trained standard NN that outputs a point prediction $\hat{y}_i$ given $\mathbf{x}_i$. The problem is two-fold: (1) Quantify the uncertainty in the predictions of the NN, and (2) calibrate the point predictions of the NN (i.e. make them more accurate).

The main idea of the proposed framework, RIO (for Residual estimation with an I/O kernel), is to predict the residuals between observed outcomes y and NN predictions $\hat{y}$ using GP with a composite kernel. The framework can be divided into two phases: training phase and deployment phase.

In the training phase, the residuals between observed outcomes and NN predictions on the training dataset are calculated as

$$r_i = y_i - \hat{y}_i, \;\text{ for } i = 1, 2, \dots, n\,. \tag{1}$$

Let $\mathbf{r}$ denote the vector of all residuals and $\hat{\mathbf{y}}$ denote the vector of all NN predictions. A GP with a composite kernel is trained assuming $\mathbf{r} \sim \mathcal{N}(0, \mathbf{K}_c((\mathcal{X}, \hat{\mathbf{y}}), (\mathcal{X}, \hat{\mathbf{y}})) + \sigma_n^2\mathbf{I})$, where $\mathbf{K}_c((\mathcal{X}, \hat{\mathbf{y}}), (\mathcal{X}, \hat{\mathbf{y}}))$ denotes an $n \times n$ covariance matrix at all pairs of training points based on a composite kernel

$$k_c((\mathbf{x}_i, \hat{y}_i), (\mathbf{x}_j, \hat{y}_j)) = k_{\text{in}}(\mathbf{x}_i, \mathbf{x}_j) + k_{\text{out}}(\hat{y}_i, \hat{y}_j), \;\text{ for } i, j = 1, 2, \dots, n\,. \tag{2}$$

Suppose a radial basis function (RBF) kernel is used for both k_{in} and k_{out}. Then,

$$k_c((\mathbf{x}_i, \hat{y}_i), (\mathbf{x}_j, \hat{y}_j)) = \sigma_{\text{in}}^2 \exp(-\frac{1}{2l_{\text{in}}^2}\|\mathbf{x}_i - \mathbf{x}_j\|^2) + \sigma_{\text{out}}^2 \exp(-\frac{1}{2l_{\text{out}}^2}\|\hat{y}_i - \hat{y}_j\|^2)\,. \tag{3}$$

The training process of GP learns the hyperparameters σ_{in}^2, l_{in}, σ_{out}^2, l_{out}, and σ_n^2 by maximizing the log marginal likelihood $\log p(\mathbf{r}|\mathcal{X}, \hat{\mathbf{y}})$ given by

$$\log p(\mathbf{r}|\mathcal{X}, \hat{\mathbf{y}}) = -\frac{1}{2}\mathbf{r}^\top(\mathbf{K}_c((\mathcal{X}, \hat{\mathbf{y}}), (\mathcal{X}, \hat{\mathbf{y}})) + \sigma_n^2\mathbf{I})\mathbf{r} - \frac{1}{2}\log|\mathbf{K}_c((\mathcal{X}, \hat{\mathbf{y}}), (\mathcal{X}, \hat{\mathbf{y}})) + \sigma_n^2\mathbf{I}| - \frac{n}{2}\log 2\pi\,. \tag{4}$$

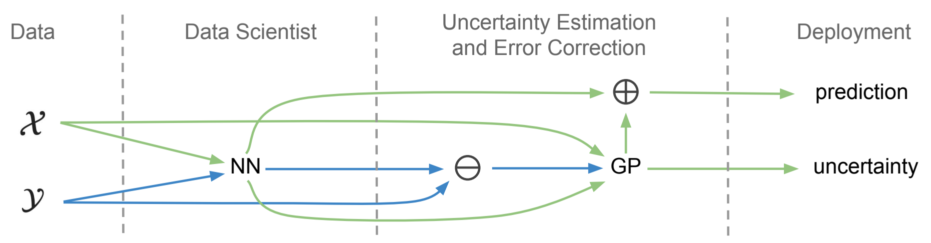

Figure 1: **Complete model-building process.** Given a dataset, first a standard NN model is constructed and trained by a data scientist. The RIO method takes this pre-trained model and trains a GP to estimate the residuals of the NN using both the output of the NN and the original input. Blue pathways are only active during the training phase. In the deployment phase, the GP provides uncertainty estimates for the predictions, while calibrating them, i.e., making point predictions more accurate.

In the deployment phase, a test point $\mathbf{x}_*$ is input to the NN to get an output $\hat{y}_*$. The trained GP predicts the distribution of the residual as $\hat{r}_*|\mathcal{X}, \hat{\mathbf{y}}, \mathbf{r}, \mathbf{x}_*, \hat{y}_* \sim \mathcal{N}(\bar{\hat{r}}_*, \text{var}(\hat{r}_*))$, where

$$\bar{\hat{r}} = \mathbf{k}_*^\top (\mathbf{K}_c((\mathcal{X}, \hat{\mathbf{y}}), (\mathcal{X}, \hat{\mathbf{y}})) + \sigma_n^2 \mathbf{I})^{-1} \mathbf{r} \ , \tag{5}$$

$$\text{var}(\hat{r}) = k_c((\mathbf{x}_*, \hat{y}_*), (\mathbf{x}_*, \hat{y}_*)) - \mathbf{k}_*^\top (\mathbf{K}_c((\mathcal{X}, \hat{\mathbf{y}}), (\mathcal{X}, \hat{\mathbf{y}})) + \sigma_n^2 \mathbf{I})^{-1} \mathbf{k}_* \ , \tag{6}$$

where $\mathbf{k}_*$ denotes the vector of kernel-based covariances (i.e., $k_c((\mathbf{x}_*, \hat{y}_*), (\mathbf{x}_i, \hat{y}_i))$) between $(\mathbf{x}_*, \hat{y}_*)$ and the training points.

Interestingly, note that the predicted residuals can also be used to calibrate the point predictions of the NN. Finally, the calibrated prediction with uncertainty information is given by

$$\hat{y}'_* \sim \mathcal{N}(\hat{y}_* + \bar{\hat{r}}_*, \text{var}(\hat{r}_*)) \ . \tag{7}$$

In other words, RIO not only adds the uncertainty estimation to a standard NN—it also makes their output more accurate, without any modification of its architecture or training. Figure 1 shows the overall procedure when applying the proposed framework in real-world applications.

Section S3.1 in appendix gives a theoretical justification for why residual prediction helps both error correction and uncertainty estimation of an NN. Specifically, such prediction (1) leads to output that is more accurate than that of GP alone, (2) leads to output that is more accurate than that of the NN alone, and (3) leads to uncertainty estimates that are positively correlated with variance of NN residuals. Section S3.2 in appendix also justifies theoretically why a GP using the proposed I/O kernel is more robust than the standard GP, i.e., using the input kernel alone. All these conclusions are confirmed in experiments, as will be described next.

RIO is scalable to large datasets by applying sparse GP methods, e.g., SVGP [10, 11]. All the conclusions above-mentioned still remain valid since sparse GP is simply an approximation of the original GP. In the case of applying SVGP with a traditional optimizer, e.g., L-BFGS-B [4, 24], the computational complexity is $\mathcal{O}(nm^2)$, and space complexity is $\mathcal{O}(nm)$, where n is the number of data points and m is the number of inducing variables. Experiments show that the computational cost of this implementation is significantly cheaper than other state-of-the-art approaches.

3 Empirical Evaluation

Experiments in this section compare nine algorithms on twelve real-world datasets. The algorithms include standard NN, the proposed RIO framework, four ablated variants of RIO, and three SOTA models that can provide predictive uncertainty, namely, SVGP [10], NNGP [16], and ANP [12]. In naming the RIO variants, "R" means estimating NN residuals then correcting NN outputs, "Y" means directly estimating outcomes, "I" means only using input kernel, "O" means only using output kernel, and "IO" means using I/O kernel. For all RIO variants (including full RIO), SVGP is used as the GP component, but using the appropriate kernel and prediction target. Therefore, "Y+I" amounts to original SVGP, and it is denoted as "SVGP" in all the experimental results. All twelve datasets are real-world regression problems [5], and cover a wide variety of dataset sizes and feature

Table 1: Summary of experimental results

Dataset n × d	Method	RMSE mean±std	NLPD mean±std	Noise Variance	Time (sec)	Dataset n × d	Method	RMSE mean±std	NLPD mean±std	Noise Variance	Time (sec)
yacht	NN	2.30±0.93†‡	-	-	4.02	ENB/h	NN	1.03±0.51†‡	-	-	6.65
	RIO	**1.46±0.49**	2.039±0.762	0.82	7.16		RIO	**0.70±0.38**	**1.038±0.355**	0.46	8.18
	R+I	2.03±0.73†‡	2.341±0.516†‡	2.54	4.30		R+I	0.79±0.46†‡	1.147±0.405†‡	0.63	7.52
308	R+O	1.88±0.66†‡	2.305±0.614†‡	1.60	6.27	768	R+O	0.80±0.43†‡	1.169±0.388†‡	0.59	7.61
×	Y+O	1.86±0.64†‡	2.305±0.639†‡	1.89	9.93	×	Y+O	0.88±0.48†‡	1.248±0.405†‡	0.75	11.06
6	Y+IO	1.58±0.52†‡	2.160±0.773†‡	1.18	9.44	8	Y+IO	0.76±0.41†‡	1.124±0.368†‡	0.56	10.64
	SVGP	4.42±0.62†‡	2.888±0.102†‡	18.56	8.96		SVGP	2.13±0.18†‡	2.200±0.074†‡	4.70	10.16
	NNGP	12.40±1.45†‡	35.18±0.534†‡	-	7347		NNGP	4.97±0.29†‡	32.40±0.638†‡	-	7374
	ANP	7.59±3.20†‡	**1.793±0.887**†‡	-	40.82		ANP	4.08±2.27†‡	2.475±0.559†‡	-	102.3
ENB/c	NN	1.88±0.44†‡	-	-	6.45	airfoil	NN	4.82±0.43†‡	-	-	6.48
	RIO	**1.48±0.33**	**1.816±0.191**	1.58	8.07		RIO	**3.07±0.18**	**2.554±0.053**	9.48	17.63
	R+I	1.71±0.44†‡	1.969±0.236†‡	2.22	5.02		R+I	3.16±0.18†‡	2.583±0.051†‡	10.07	15.90
768	R+O	1.75±0.43†‡	2.000±0.229†‡	2.25	4.57	1505	R+O	4.17±0.26†‡	2.849±0.066†‡	16.64	9.97
×	Y+O	1.76±0.43†‡	2.000±0.231†‡	2.32	10.99	×	Y+O	4.24±0.28†‡	2.869±0.075†‡	17.81	22.72
8	Y+IO	1.64±0.36†‡	1.936±0.210†‡	1.96	10.56	5	Y+IO	3.64±0.53†‡	2.712±0.150†‡	14.40	24.51
	SVGP	2.63±0.23†‡	2.403±0.078†‡	6.81	10.28		SVGP	3.59±0.20†‡	2.699±0.053†‡	12.67	21.74
	NNGP	4.91±0.32†‡	30.14±0.886†‡	-	7704		NNGP	6.54±0.23†‡	33.60±0.420†‡	-	3355
	ANP	4.81±2.15†‡	2.698±0.548†‡	-	64.11		ANP	21.17±30.72†‡	5.399±6.316†‡	-	231.7
CCS	NN	6.23±0.53†‡	-	-	9.46	wine/r	NN	0.691±0.041†‡	-	-	3.61
	RIO	**5.97±0.48**	**3.241±0.109**	24.74	13.71		RIO	0.672±0.036	1.094±0.100	0.28	9.25
1030	R+I	6.01±0.50†‡	3.248±0.112†‡	25.40	9.52	1599	R+I	0.669±0.036†‡	1.085±0.097†‡	0.28	8.34
×	R+O	6.17±0.54†‡	3.283±0.120†‡	26.31	9.54	×	R+O	0.676±0.035†‡	1.099±0.094‡	0.29	5.02
8	Y+O	6.15±0.52†‡	3.279±0.117†‡	26.53	21.35	11	Y+O	0.676±0.034†‡	1.096±0.092	0.29	12.71
	Y+IO	6.06±0.49†‡	3.261±0.110†‡	25.82	23.15		Y+IO	0.672±0.036†‡	1.094±0.098	0.28	12.48
	SVGP	6.87±0.39†‡	3.336±0.048†‡	44.55	19.85		SVGP	**0.642±0.028**†‡	**0.974±0.042**†‡	0.40	12.17
wine/w	NN	0.721±0.023†‡	-	-	7.17	CCPP	NN	4.96±0.53†‡	-	-	14.52
	RIO	0.704±0.018	1.090±0.038	0.37	16.74		RIO	**4.05±0.128**	**2.818±0.031**	16.30	42.65
4898	R+I	**0.699±0.018**†‡	**1.081±0.037**†‡	0.38	13.5	9568	R+I	4.06±0.13†‡	2.822±0.031†‡	16.39	39.88
×	R+O	0.710±0.019†‡	1.098±0.038†‡	0.39	6.19	×	R+O	4.32±0.15†‡	2.883±0.035†‡	18.50	18.48
11	Y+O	0.710±0.019†‡	1.096±0.038†‡	0.39	18.39	4	Y+O	4.37±0.20†‡	2.914±0.122†‡	23.98	48.27
	Y+IO	0.705±0.019†‡	1.090±0.038	0.38	20.06		Y+IO	4.56±1.00†‡	2.958±0.216†‡	31.06	46.8
	SVGP	0.719±0.018†‡	**1.081±0.022**†‡	0.50	18.18		SVGP	4.36±0.13†‡	2.893±0.031†‡	19.04	46.43
protein	NN	4.21±0.07†‡	-	-	151.8	SC	NN	12.23±0.77†‡	-	-	51.9
	RIO	**4.08±0.06**	**2.826±0.014**	15.71	149.4		RIO	**11.28±0.46**	**3.853±0.042**	105.83	53.39
45730	R+I	4.11±0.06†‡	2.834±0.037†‡	15.99	141.2	21263	R+I	11.33±0.45†‡	3.858±0.041†‡	107.35	47.72
×	R+O	4.14±0.06†‡	2.840±0.015†‡	16.18	115.1	×	R+O	11.63±0.52†‡	3.881±0.046†‡	112.91	30.47
9	Y+O	4.14±0.06†‡	2.840±0.015†‡	16.17	138.4	81	Y+O	11.64±0.53†‡	3.882±0.046†‡	113.61	45.35
	Y+IO	**4.08±0.06**	**2.826±0.014**	15.72	155.5		Y+IO	11.32±0.45†‡	3.856±0.041†‡	106.93	57.74
	SVGP	4.68±0.04†‡	2.963±0.007†‡	22.54	149.5		SVGP	14.66±0.25†‡	4.136±0.014†‡	239.28	50.89
CT	NN	1.17±0.34†‡	-	-	194.5	MSD	NN	12.42±2.97†‡	-	-	1136
	RIO	**0.88±0.15**	1.284±0.219	1.02	516.4		RIO	**9.26±0.21**	**3.639±0.022**	84.28	1993
53500	R+I	1.17±0.34†‡	1.538±0.289†‡	1.71	19.80	515345	R+I	10.92±1.30†‡	3.811±0.128†‡	135.34	282.0
×	R+O	**0.88±0.15**	1.283±0.219†‡	1.02	159.4	×	R+O	**9.25±0.20**	**3.638±0.021**	84.05	1518
384	Y+O	0.99±0.42†‡	1.365±0.385†‡	2.45	168.2	90	Y+O	10.00±0.86†‡	3.768±0.148†‡	169.90	1080
	Y+IO	0.91±0.16†‡	**1.280±0.177**‡	0.62	578.6		Y+IO	9.43±0.52‡	3.644±0.025†‡	85.66	2605
	SVGP	52.07±0.19†‡	5.372±0.004†‡	2712	27.56		SVGP	9.57±0.00†‡	3.677±0.000†‡	92.21	2276

The symbols † and ‡ indicate that the difference between the marked entry and RIO is statistically significant at the 5% significance level using paired t-test and Wilcoxon test, respectively. The best entries that are significantly better than all the others under at least one statistical test are marked in boldface (ties are allowed).

dimensionalities. All datasets are tested for 100 independent runs. During each run, the dataset is randomly split into training set, validation set, and test set, and all algorithms are trained on the same split. All RIO variants that involve an output kernel or residual estimation are based on the trained NN in the same run. See section S4.1 in appendix for the details of experimental setups.

From Table 1, in terms of point-prediction errors ("RMSE" column), RIO consistently improves over the original NNs, and performs the best or equals the best (based on statistical tests) in 10 out of 12 datasets. To measure the quality of uncertainty estimation quantitatively, average negative log predictive density (NLPD) [20] is used as the performance metric, and RIO performs the best or equals the best method (based on statistical tests) in 8 out of 12 datasets.

4 Conclusion and Future Directions

The RIO framework both provides estimates of predictive uncertainty of NNs, and reduces their point-prediction errors. Remarkably, it can be applied directly to any pretrained NNs without modifications to model architecture or training pipeline. Interesting future directions includes: (1) applying RIO to reinforcement learning (RL) algorithms, which usually use standard NNs for reward predictions. This allows uncertainty estimation of the future rewards. Agents can then employ more mathematically efficient exploration strategies, e.g., bandit algorithms [23], rather than traditional epsilon greedy methods. (2) applying RIO to Bayesian optimization (BO) [17]. This allows the usage of NNs in BO, and it can potentially improve the expressivity of surrogate models and scalability of BO. (3) metalearning with pretrained models. RIO learns to exploit a single pretrained model. A more general metalearning process would learn to optimally integrate a collection of pretrained models. By avoiding training base learners from scratch, such methods can scale well to real world applications.

References

[1] O. Anjos, C. Iglesias, F. Peres, J. MartÃnez, A. Garcia, and J. Taboada. Neural networks applied to discriminate botanical origin of honeys. *Food Chemistry*, 175:128–136, 05 2015.

[2] P. Baldi, P. Sadowski, and D. Whiteson. Searching for exotic particles in high-energy physics with deep learning. *Nature communications*, 5:4308, 07 2014.

[3] S. Bergmann, S. Stelzer, and S. Strassburger. On the use of artificial neural networks in simulation-based manufacturing control. *Journal of Simulation*, 8(1):76–90, Feb 2014.

[4] R. Byrd, P. Lu, J. Nocedal, and C. Zhu. A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing*, 16(5):1190–1208, 1995.

[5] D. Dua and C. Graff. UCI machine learning repository, 2017.

[6] Y. Gal and Z. Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, pages 1050–1059. JMLR.org, 2016.

[7] M. Garnelo, D. Rosenbaum, C. Maddison, T. Ramalho, D. Saxton, M. Shanahan, Y. W. Teh, D. Rezende, and S. M. A. Eslami. Conditional neural processes. In J. Dy and A. Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1704–1713. PMLR, 10–15 Jul 2018.

[8] M. Garnelo, J. Schwarz, D. Rosenbaum, F. Viola, D. J. Rezende, S. M. A. Eslami, and Y. W. Teh. Neural processes. *CoRR*, abs/1807.01622, 2018.

[9] Z. Ghahramani. Probabilistic machine learning and artificial intelligence. *Nature*, 521:452 EP –, 05 2015.

[10] J. Hensman, N. Fusi, and N. D. Lawrence. Gaussian processes for big data. In *Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence*, UAI'13, pages 282–290, Arlington, Virginia, United States, 2013. AUAI Press.

[11] J. Hensman, A. Matthews, and Z. Ghahramani. Scalable Variational Gaussian Process Classification. In G. Lebanon and S. V. N. Vishwanathan, editors, *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*, volume 38 of *Proceedings of Machine Learning Research*, pages 351–360, San Diego, California, USA, 09–12 May 2015. PMLR.

[12] H. Kim, A. Mnih, J. Schwarz, M. Garnelo, S. M. A. Eslami, D. Rosenbaum, O. Vinyals, and Y. W. Teh. Attentive neural processes. *CoRR*, abs/1901.05761, 2019.

[13] M. Krzywinski and N. Altman. Importance of being uncertain. *Nature Methods*, 10:809 EP –, 08 2013.

[14] B. Lakshminarayanan, A. Pritzel, and C. Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 6402–6413. Curran Associates, Inc., 2017.

[15] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521:436 EP –, 05 2015.

[16] J. Lee, Y. Bahri, R. Novak, S. Schoenholz, J. Pennington, and J. Sohl-dickstein. Deep neural networks as gaussian processes. *International Conference on Learning Representations*, 2018.

[17] J. Močkus. On bayesian methods for seeking the extremum. In G. I. Marchuk, editor, *Optimization Techniques IFIP Technical Conference Novosibirsk, July 1–7, 1974*, pages 400–404, Berlin, Heidelberg, 1975. Springer Berlin Heidelberg.

[18] R. M. Neal. *Bayesian Learning for Neural Networks*. Springer-Verlag, Berlin, Heidelberg, 1996.

[19] S. T. A. Niaki and S. Hoseinzade. Forecasting s&p 500 index using artificial neural networks and design of experiments. *Journal of Industrial Engineering International*, 9(1):1, Feb 2013.

[20] J. Quiñonero-Candela, C. E. Rasmussen, F. Sinz, O. Bousquet, and B. Schölkopf. Evaluating predictive uncertainty challenge. In J. Quiñonero-Candela, I. Dagan, B. Magnini, and F. d'Alché Buc, editors, *Machine Learning Challenges. Evaluating Predictive Uncertainty, Visual Object Classification, and Recognising Tectual Entailment*, pages 1–27, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[21] C. Rasmussen and C. Williams. *Gaussian Processes for Machine Learning*. Adaptive Computation and Machine Learning. MIT Press, Jan. 2006.

[22] N. Shahid, T. Rappon, and W. Berta. Applications of artificial neural networks in health care organizational decision-making: A scoping review. *PLOS ONE*, 14(2):1–22, 02 2019.

[23] W. R. Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3/4):285–294, 1933.

[24] C. Zhu, R. H. Byrd, P. Lu, and J. Nocedal. Algorithm 778: L-bfgs-b: Fortran subroutines for large-scale bound-constrained optimization. *ACM Trans. Math. Softw.*, 23(4):550–560, Dec. 1997.