Supplementary Material for "Frequentist Uncertainty in Recurrent Neural Networks via *Blockwise* Influence Functions"

Ahmed M. Alaa [1] **Mihaela van der Schaar** [1 2]

Appendix

Appendix A: Literature Survey

Driven by the desire to suppress the "overconfidence" exhibited by poorly calibrated — but highly discriminative — deep learning models, various methods for uncertainty estimation in standard feed-forward neural networks have been recently proposed (Hernández-Lobato & Adams, 2015; Gal & Ghahramani, 2016a; Lakshminarayanan et al., 2017; Malinin & Gales, 2018). However, the literature on uncertainty quantification in RNNs is rather scarce. In what follows, we provide a detailed overview of uncertainty quantification methods in RNNs, other methods that have been developed for feed-forward networks, background on jackknife resampling, and the connections between our work and modern applications of influence functions.

Uncertainty quantification in RNNs

We identified the following three strands of literature relating to uncertainty quantification in RNNs.

Bayesian RNNs. The most prevalent approaches for RNN uncertainty estimation hinge on Bayesian modeling (Fortunato et al., 2017; Chien & Ku, 2015; Zhu & Laptev, 2017; Mirikitani & Nikolaev, 2009). By specifying a prior over the RNN parameters, these approaches estimate a model's uncertainty via its posterior credible intervals. However, exact inference in Bayesian deep learning is generally intractable or computationally infeasible. Hence, practical Bayesian RNNs rely on dropout-based approximate inference (Gal & Ghahramani, 2016a;b), which is often hard to calibrate, and is generally ill-posed as its posterior distributions do not concentrate asymptotically (Osband, 2016; Hron et al., 2017). Moreover, Bayesian methods require significant modifications to model architecture and training which limits their versatility. Our paper addresses these issues by developing a frequentist post-hoc alternative to Bayesian methods.

Quantile RNNs. Similar to quantile regression (Koenker & Hallock, 2001), quantile RNNs (Q-RNNs) learn prediction intervals by explicitly modeling the cumulative distribution function of the prediction targets (Taylor, 2000; Wen et al., 2017; Gasthaus et al., 2019). The key difference between these approaches and ours is that Q-RNNs learn uncertainty jointly with the main prediction task, which as we have shown in Section **??**, can compromise their predictive accuracy. Moreover, Q-RNNs are bespoke models that must be retailored for each new prediction task, whereas our approach can be applied in a post-hoc manner to any RNN architecture.

Probabilistic RNNs. This class of models capture the structural uncertainty within the stochastic dynamics of sequential data. Existing variants of probabilistic RNNs are mainly based on deep state-space models (Krishnan et al., 2017; Rangapuram et al., 2018; Alaa & van der Schaar, 2019). Probabilistic models are bespoke to the application at hand, and may not be appropriate for some application domains. Moreover, most probabilistic models entail restrictive assumptions on sequence dynamics, such as state Markovianity (Alaa & Van Der Schaar, 2018).

Uncertainty quantification in feed-forward networks

Bayesian models are the dominant tools for uncertainty quantification in feed-forward neural networks (Welling & Teh, 2011; Hernández-Lobato & Adams, 2015; Ritter et al., 2018; Maddox et al., 2019). Post-hoc application of Bayesian methods is not possible since Bayesian models require major modifications to the underlying predictive models (by specifying priors

[1]University of California, Los Angeles [2]Cambridge University. Correspondence to: Ahmed M. Alaa <ahmedmalaa@ucla.edu>.

Proceedings of the 37^{th} International Conference on Machine Learning, Vienna, Austria, PMLR 119, 2020.

over model parameters). Moreover, Bayesian credible intervals do not guarantee frequentist coverage, and more crucially, the achieved coverage can be very sensitive to hyper-parameter tuning (Bayarri & Berger, 2004). Finally, a major limitation to exact Bayesian inference is that they are computationally prohibitive, and their alternative approximations — e.g., (Gal & Ghahramani, 2016a) — may induced posteriors that do not concentrate asymptotically (Osband, 2016; Hron et al., 2017).

Deep ensembles (Lakshminarayanan et al., 2017) are regarded as the most competitive alternative non-Bayesian benchmark for uncertainty estimation (Ovadia et al., 2019). These methods repeatedly re-trains the model on sub-samples of the data (using adversarial training), and then estimates uncertainty through the variance of the aggregate predictions. However, while applying deep ensemble methods may be feasible in feed-forward neural networks, creating large ensembles of RNN models is computationally infeasible, especially for long time series.

Jackknife resampling

Jackknife resampling, originally developed by Quenouille in (Quenouille, 1956) and refined by Tukey in (Tukey, 1958), is a general methodology for estimating sampling distributions. Our work builds on this literature by applying the jackknife to RNN predictions in order to estimate its predictive uncertainty through the sampling variance. The key difference between our setup and that of the standard jackknife is that RNNs do not operate on i.i.d data — they operate on temporally correlated sequences. In (Kunsch, 1989), Künsch proposed a variant of the jackknife for general stationary observations rather than i.i.d data. As we have shown in Section **??**, the sampling procedure therein is a special case of ours when the RNN is trained with a single data sequence (i.e., a single realization of the underlying stochastic process).

The (infinitesimal) jackknife method was previously used for quantifying the predictive uncertainty in random forests (Wager et al., 2014; Mentch & Hooker, 2016; Wager & Athey, 2018). In these works, however, the developed jackknife estimators are bespoke to bagging predictors, and cannot be straightforwardly extended to deep neural networks. More recently, general-purpose jackknife estimators were developed in (Barber et al., 2019b), where two exhaustive leave-one-out procedures: the *jackknife+* and the *jackknife-minmax* where shown to have assumption-free worst-case coverage guarantees of $(1-2\alpha)$ and $(1-\alpha)$, respectively. However, our work is the first to develop such methods for time-series data.

Previous application of influence functions has been limited to model interpretability by learning sample importance (Koh & Liang, 2017; Koh et al., 2019). Our work uses the same machinery of influence function computation via Hessian-vector products, but for the purpose of estimating leave-one-out model parameters.

Appendix B: Stochastic Estimation of HVPs via Blockwise Batch Sampling

The bottleneck in computing the influence function in (**??**) is the inversion of the Hessian matrix $H_{\boldsymbol{w}}$; for P RNN parameters in $\hat{\boldsymbol{\theta}}$, the complexity of computing $H_{\boldsymbol{w}}^{-1}$ is $\mathcal{O}(P^3)$. Moreover, because the RNN parameters are shared across time steps, the complexity of gradient computation using backpropagation through time (BPTT) is $\mathcal{O}(PT)$. This computational burden can be significant when performing interval block deletion on long sequences, since deleting the j-th interval block entails altering the RNN hidden states in time steps $t \geq K \cdot (j+1)$ and re-computing all gradients for those time steps.

We start by the following well-known fact about the inverse of a matrix H with $\|H\| \leq 1$ and $H \succ 0$. The inverse of such matrix can be evaluated through the following power series expansion:

$$H^{-1} = \sum_{i=0}^{\infty} (I - H)^i. \tag{1}$$

Following the approaches in (Pearlmutter, 1994; Agarwal et al., 2016; Koh & Liang, 2017), we address the computational challenges above by developing a stochastic estimation approach with blockwise batch sampling to estimate the hessian-vector product (HVP) $H_{\boldsymbol{w}}^{-1}\, g_{\boldsymbol{w}}(\boldsymbol{w}')$, without explicitly inverting or forming the Hessian. The pseudo-code for BLOCKWISEINFLUENCE, which computes $\mathcal{I}_{\boldsymbol{w}}(\boldsymbol{w}_{-j[K]}^{-i})$, is:

- Randomly sample $v \leq n-1$ sub-sequences $m_1, \ldots, m_v$, each represented as a tuple $\{\mathcal{S}_{1:Kj}^{(m_s)}, \mathcal{S}_{K(j+1):T}^{(m_s)}\}_{s=1}^{v}$, with the j-th interval block deleted, then from each tuple select $\mathcal{S}_{1:Kj}^{(m_s)}$. In doing so, we only include temporally contiguous data in the sampled batch, hence retaining the RNN hidden states computed in training. (See Figure **??** for an illustration.)
- For all j and $s \in \{1, \ldots, v\}$, recursively compute

$$\widetilde{H}_{s,\boldsymbol{w}}^{-1}\, g = g + (I - \nabla_\theta^2\, L(\mathcal{S}_{1:Kj}^{(m_s)}; \hat{\boldsymbol{\theta}}(\boldsymbol{w}), \boldsymbol{w}'))\, \widetilde{H}_{s-1,\boldsymbol{w}}^{-1}\, g,$$

where $g = g_{\boldsymbol{w}}(\boldsymbol{w}^{-i}_{-j[K]})$ is the loss gradient after j-th interval and i-th sequence deletion, $\widetilde{H}^{-1}_{s,\boldsymbol{w}} = \sum_{v=0}^{s}(I - \widetilde{H}_{s,\boldsymbol{w}})^v$, and $\widetilde{H}$ is the stochastic estimate of the Hessian H. The recursive procedure is initialized with $\widetilde{H}^{-1}_{0,\boldsymbol{w}}\, g = g$, and our final estimate of the HVP is given by $\widetilde{H}^{-1}_{v,\boldsymbol{w}}\, g$, which converges to the true inverse for $v \to \infty$.

Since restricting our sampled sub-sequences to the interval $[1, Kj]$ enables reusing the loss gradients evaluated in training, the overall complexity of our procedure is $\mathcal{O}(nP + vP)$, this can be broken down as follows. The complexity of multiplying $\nabla^2_\theta\, L(\mathcal{S}^{(m_s)}_{1:Kj}; \hat{\boldsymbol{\theta}}(\boldsymbol{w}), \boldsymbol{w}'))$ with $\widetilde{H}^{-1}_{s-1,\boldsymbol{w}}\, g$ is $\mathcal{O}(P)$ per iteration which renders the overall complexity of computing this Hessian-product term as $\mathcal{O}(vP)$. The complexity of computing the initial gradient terms is $\mathcal{O}(nP)$.

The procedure above converges only if the norm of the Hessian satisfies $\|H\| \leq 1$. Thus, if the largest eigenvalue σ_{max} of H is greater than 1, we normalize the loss to keep the largest eigenvalue less than 1. Moreover, we add a dampening term λ to the Hessian in order to keep it invertible (positive definite). This results in the following recursion:

$$\widetilde{H}^{-1}_{s,\boldsymbol{w}}\, g = g + (1 - \lambda)\, \widetilde{H}^{-1}_{s-1,\boldsymbol{w}}\, g - \frac{1}{\sigma_{max}} \nabla^2_\theta\, L(\mathcal{S}^{(m_s)}_{1:Kj}; \hat{\boldsymbol{\theta}}(\boldsymbol{w}), \boldsymbol{w}')\, \widetilde{H}^{-1}_{s-1,\boldsymbol{w}}\, g,$$

which converges with the appropriate selection of σ_{max} and λ.

Appendix C: Proof of Theorem 1

Given a mixing-time of T_0, we can segment the data set $\mathcal{D}_n = \{(\boldsymbol{x}^{(i)}, y^{(i)})\}_{i=1}^n$ into evenly spaced segments of data

$$\widetilde{\mathcal{D}}_n = \{(\boldsymbol{x}^{(i)}_{j\cdot T_0 : j\cdot T_0 + T_0}, y^{(i)}_{j\cdot T_0 : j\cdot T_0 + T_0})\}_{i=1}^n,$$

which we can re-write as a new data set with $\tilde{n} = n \cdot T/T_0$ samples each with $d \cdot T_0$ dimensional features as follows:

$$\widetilde{\mathcal{D}}_n = \{(\tilde{\boldsymbol{x}}^{(j)}, \tilde{y}^{(j)})\}_{j=1}^{\tilde{n}},$$

where all the data points $\widetilde{\mathcal{D}}_n$ are independent. By treating each RNN prediction at time step t as a new regression problem, it follows that the exact BJ procedure achieves coverage by directly applying Theorem 1 in (Barber et al., 2019a) to the modified data set $\widetilde{\mathcal{D}}_n$.

Appendix D: Experiments

Experimental Setup

Data. We used data from the Medical Information Mart for Intensive Care (MIMIC-III) (Johnson et al., 2016) database, which comprises the electronic health records for critically-ill patients admitted to an intensive care unit (ICU). From MIMIC-III, we extracted sequential data for patients on antibiotics, with trajectories up to $T = 10$ time steps — this resulted in a data set with $n = 5,833$ sequences. For each patient, we extracted 30 variables (lab tests and vital signs) listed in Table 1, based on which predictions are issued.

The follow-up times were varying from one patient to the other, ranging from 4 days to 10 days. This results in a learning problem with variable-length sequences. Clinical care teams decide *when* to administer an antibiotic for each patient on a daily basis — the frequency of antibiotic treatments over time for the entire population is provided in Figure 1. The patient outcomes under consideration are the white blood cell count (WBCC) — a high white blood cell count is associated with severe illness and poor outcome for ICU patients (Shuman et al., 2012). The WBCC averaged for all patients over time are depicted in Figure 2.

Clinical setup. The WBCC levels are medically categories into four levels as follows (Waheed et al., 2003):

- Leucopenic ($< 4 \times 10^9/L$).
- Normal ($4 - 10 \times 10^9/L$).
- Leucemoid ($10 - 25 \times 10^9/L$).
- Exaggerated Leucemoid ($> 25 \times 10^9/L$).

Variable Names	
Age	Weight
Temperature	Heart rate
Systolic blood pressure	Diastolic blood pressure
Mean blood pressure	SpO2
FiO2	Respiratory rate
Glucose	Bicarbonate (high)
Bicarbonate (low)	Creatinine (high)
Creatinine (low)	Hematocrit (high)
Hematocrit (low)	Hemoglobin (high)
Hemoglobin (low)	Platelet (high)
Platelet (low)	Potassium (high)
Potassium (low)	Billirubun (high)
Billirubun (low)	Weight blood cell (high)
Weight blood cell (low)	Antibiotics
Norepinephrine	Mechanical ventilator

Table 1. Vital signs and lab tests in MIMIC-III.

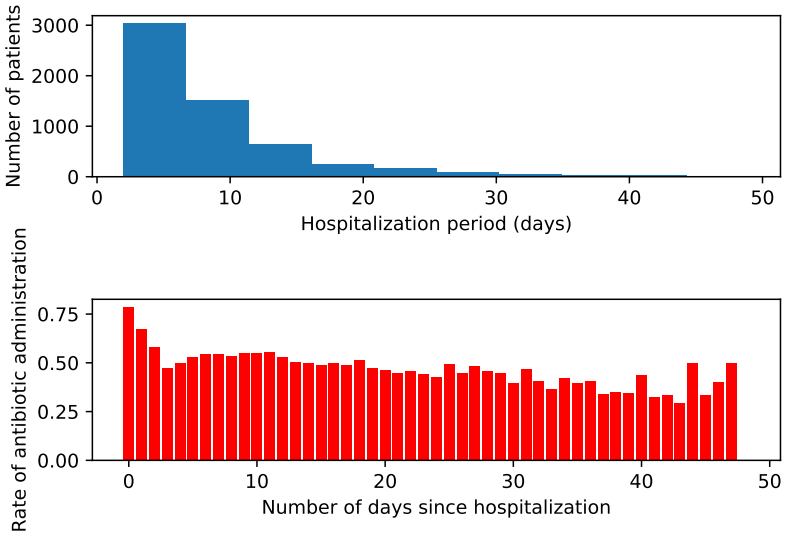

Figure 1. Statistics of hospitalization and treatment data.

Antibiotic administration in the ICU aims at keeping the WBCC within the normal range. However, the extent by which an antibiotic affects a patient's WBCC — if the WBCC reduces significantly and enters the Leucopenic level, then the patient might encounter further health risks. Thus, we need to predict a patient's outcome with confidence intervals; if those intervals overlaps with the Leucopenic and Leucemoid regions, then we should abstain from administering an antibiotic.

We link this setup with our formulation as follows. Each patient's feature sequence $\boldsymbol{x}$ is a collection of 30 clinical variables measured repeatedly over T time steps ($T \leq 10$). We selected the most relevant 5 among these features, and trained an RNN model to predict the next-day WBCC for every patient, and use our procedure to construct confidence intervals on these predictions.

Baselines. We conduct our main experiment with the simple RNN architecture, in addition to LSTMs and GRUs. Further architectures are also explored later in this Section. We compared our method with 5 baselines for predictive uncertainty estimation in RNNs that cover the different modeling categories presented in Section **??**. The modeling approaches under consideration and their respective baselines are:

(a) Bayesian RNNs. We implemented the Monte Carlo dropout-based Bayesian RNNs (DP-RNNs) proposed in (Gal & Ghahramani, 2016b) using the approximate variational inference scheme described in Sections 3 and 4 in (Gal & Ghahramani,

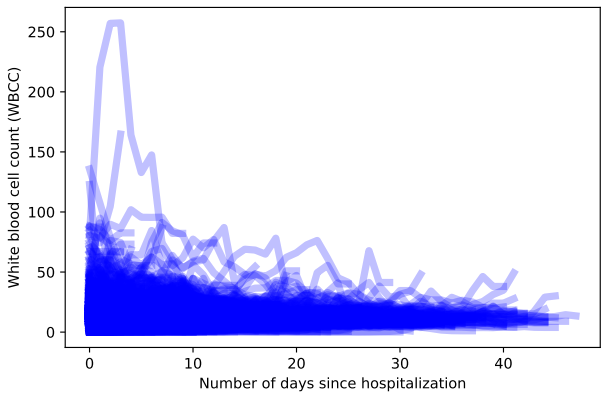

Figure 2. Distribution of white blood cell counts over time.

2016b). Dropout is applied to all RNN layers during training and then Monte Carlo dropout is applied for each new test sequence to collect samples of the model predictions. We compute the confidence intervals by estimating the posterior variance of a Gaussian distribution through the Monte Carlo samples and then constructing confidence intervals through the tails of the Gaussian.

(b) Quantile RNNs. We implemented an quantile RNN forecaster similar to the one proposed in (Wen et al., 2017) using the quantile loss:

$$\ell(y, f) = q\,(y - f^{(q)})_+ + (1 - q)\,(f^{(q)} - y)_+,$$

where $(.)_+ = \max(0, .)$ and $y^{(q)}$ is the q-th quantile of the prediction target. The parameter q is selected so that the learned prediction intervals achieve a coverage probability of $(1 - \alpha)$. In order to construct confidence intervals with α-level coverage, we train the RNN decoder to predict $f^{(1-\alpha/2)}$ and $f^{(1-\alpha/2)}$ in order to recover the lower and upper limits of the confidence intervals, respectively.

Hyperparameter Tuning and Uncertainty Calibration

To ensure a fair comparison, we fix the hyperparameters of the underlying RNNs in all baselines to the following:

- Number of layers: 1
- Number of hidden units: 20
- Learning rate: 0.01
- Batch size: 150
- Number of epochs: 10
- Number of steps: 1000

The hyperparameters above where tuned to optimize the RMSE performance of the vanilla RNN model. There was no improvement in the out-of-sample RMSE or discriminative accuracy of all baselines by tweaking these hyperparameters. For the DP-RNN model, we tuned the dropout rate to optimize its RMSE given the values above for the other hyperparameters, the optimal dropout value was 0.45.

The $(1 - \alpha)$ confidence intervals for all baselines were computed as follows:

- **MQ-RNN:** The confidence intervals are obtained straightforwardly as the interval bounded by the model's quantile outputs $[y^{(\alpha)}, y^{(1-\alpha)}]$. Note that the MQ-RNN is explicitly trained to predict these values.

- **DP-RNN:** Because DP-based inference approximates a (deep) Gaussian process prosterior, we construct the $(1-\alpha)$ confidence intervals through the tails of the Gaussian, i.e., the confidence interval is given by $[y^{(\alpha)}, y^{(1-\alpha)}]$, where the limits of the interval are constructed as $y^{(q)} = z_c(q) \cdot \sqrt{\mathbb{V}_n[\{y_i\}_i]}$, where $z_c(q)$ is the critical value of the Gaussian at the q-th quantile, and $\mathbb{V}_n[\{y_i\}_i]$ is the empirical variance of the Monte Carlo samples of the model.

References

Agarwal, N., Bullins, B., and Hazan, E. Second-order stochastic optimization in linear time. *stat*, 1050:15, 2016.

Alaa, A. M. and Van Der Schaar, M. A hidden absorbing semi-markov model for informatively censored temporal data: Learning and inference. *The Journal of Machine Learning Research*, 19(1):108–169, 2018.

Alaa, A. M. and van der Schaar, M. Attentive state-space modeling of disease progression. In *Advances in Neural Information Processing Systems*, pp. 11334–11344, 2019.

Barber, R. F., Candes, E. J., Ramdas, A., and Tibshirani, R. J. Conformal prediction under covariate shift. *arXiv preprint arXiv:1904.06019*, 2019a.

Barber, R. F., Candes, E. J., Ramdas, A., and Tibshirani, R. J. Predictive inference with the jackknife+. *arXiv preprint arXiv:1905.02928*, 2019b.

Bayarri, M. J. and Berger, J. O. The interplay of bayesian and frequentist analysis. *Statistical Science*, pp. 58–80, 2004.

Chien, J.-T. and Ku, Y.-C. Bayesian recurrent neural network for language modeling. *IEEE transactions on neural networks and learning systems*, 27(2):361–374, 2015.

Fortunato, M., Blundell, C., and Vinyals, O. Bayesian recurrent neural networks. *arXiv preprint arXiv:1704.02798*, 2017.

Gal, Y. and Ghahramani, Z. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *International Conference on Machine Learning (ICML)*, pp. 1050–1059, 2016a.

Gal, Y. and Ghahramani, Z. A theoretically grounded application of dropout in recurrent neural networks. In *Advances in neural information processing systems*, pp. 1019–1027, 2016b.

Gasthaus, J., Benidis, K., Wang, Y., Rangapuram, S. S., Salinas, D., Flunkert, V., and Januschowski, T. Probabilistic forecasting with spline quantile function rnns. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pp. 1901–1910, 2019.

Hernández-Lobato, J. M. and Adams, R. Probabilistic backpropagation for scalable learning of bayesian neural networks. In *International Conference on Machine Learning (ICML)*, pp. 1861–1869, 2015.

Hron, J., Matthews, A. G. d. G., and Ghahramani, Z. Variational gaussian dropout is not bayesian. *arXiv preprint arXiv:1711.02989*, 2017.

Johnson, A. E., Pollard, T. J., Shen, L., Li-wei, H. L., Feng, M., Ghassemi, M., Moody, B., Szolovits, P., Celi, L. A., and Mark, R. G. Mimic-iii, a freely accessible critical care database. *Scientific data*, 3:160035, 2016.

Koenker, R. and Hallock, K. F. Quantile regression. *Journal of economic perspectives*, 15(4):143–156, 2001.

Koh, P. W. and Liang, P. Understanding black-box predictions via influence functions. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1885–1894. JMLR. org, 2017.

Koh, P. W., Ang, K.-S., Teo, H. H., and Liang, P. On the accuracy of influence functions for measuring group effects. *arXiv preprint arXiv:1905.13289*, 2019.

Krishnan, R. G., Shalit, U., and Sontag, D. Structured inference networks for nonlinear state space models. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

Kunsch, H. R. The jackknife and the bootstrap for general stationary observations. *The annals of Statistics*, pp. 1217–1241, 1989.

Lakshminarayanan, B., Pritzel, A., and Blundell, C. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 6402–6413, 2017.

Maddox, W., Garipov, T., Izmailov, P., Vetrov, D., and Wilson, A. G. A simple baseline for bayesian uncertainty in deep learning. *arXiv preprint arXiv:1902.02476*, 2019.

Malinin, A. and Gales, M. Predictive uncertainty estimation via prior networks. In *Advances in Neural Information Processing Systems*, pp. 7047–7058, 2018.

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

Mirikitani, D. T. and Nikolaev, N. Recursive bayesian recurrent neural networks for time-series modeling. *IEEE Transactions on Neural Networks*, 21(2):262–274, 2009.

Osband, I. Risk versus uncertainty in deep learning: Bayes, bootstrap and the dangers of dropout. In *NIPS Workshop on Bayesian Deep Learning*, 2016.

Ovadia, Y., Fertig, E., Ren, J., Nado, Z., Sculley, D., Nowozin, S., Dillon, J. V., Lakshminarayanan, B., and Snoek, J. Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. *arXiv preprint arXiv:1906.02530*, 2019.

Pearlmutter, B. A. Fast exact multiplication by the hessian. *Neural computation*, 6(1):147–160, 1994.

Quenouille, M. H. Notes on bias in estimation. *Biometrika*, 43(3/4):353–360, 1956.

Rangapuram, S. S., Seeger, M. W., Gasthaus, J., Stella, L., Wang, Y., and Januschowski, T. Deep state space models for time series forecasting. In *Advances in Neural Information Processing Systems*, pp. 7785–7794, 2018.

Ritter, H., Botev, A., and Barber, D. A scalable laplace approximation for neural networks. 2018.

Shuman, M., Demler, T. L., Trigoboff, E., and Opler, L. A. Hematologic impact of antibiotic administration on patients taking clozapine. *Innovations in clinical neuroscience*, 9(11-12):18, 2012.

Taylor, J. W. A quantile regression neural network approach to estimating the conditional density of multiperiod returns. *Journal of Forecasting*, 19(4):299–311, 2000.

Tukey, J. Bias and confidence in not quite large samples. *Ann. Math. Statist.*, 29:614, 1958.

Wager, S. and Athey, S. Estimation and inference of heterogeneous treatment effects using random forests. *Journal of the American Statistical Association*, 113(523):1228–1242, 2018.

Wager, S., Hastie, T., and Efron, B. Confidence intervals for random forests: The jackknife and the infinitesimal jackknife. *The Journal of Machine Learning Research (JMLR)*, 15(1):1625–1651, 2014.

Waheed, U., Williams, P., Brett, S., Baldock, G., and Soni, N. White cell count and intensive care unit outcome. *Anaesthesia*, 58(2):180–182, 2003.

Welling, M. and Teh, Y. W. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pp. 681–688, 2011.

Wen, R., Torkkola, K., Narayanaswamy, B., and Madeka, D. A multi-horizon quantile recurrent forecaster. *arXiv preprint arXiv:1711.11053*, 2017.

Zhu, L. and Laptev, N. Deep and confident prediction for time series at uber. In *2017 IEEE International Conference on Data Mining Workshops (ICDMW)*, pp. 103–110. IEEE, 2017.