ROBUST NONPARAMETRIC DISTRIBUTION FORECAST WITH BACKTEST-BASED BOOTSTRAP AND ADAPTIVE RESIDUAL SELECTION

Longshaokan Wang[⋆], *Lingda Wang*[1,†], *Mina Georgieva*[⋆], *Paulo Machado*[⋆], *Abinaya Ulagappa*[⋆], *Safwan Ahmed*[⋆], *Yan Lu*[⋆], *Arjun Bakshi*[⋆], *Farhad Ghassemi*[⋆]
[⋆]Amazon [†] University of Illinois at Urbana-Champaign
longsha@amazon.com

ABSTRACT

Distribution forecast can quantify forecast uncertainty and provide various forecast scenarios with their corresponding estimated probabilities. Accurate distribution forecast is crucial for planning – for example when making production capacity or inventory allocation decisions. We propose a practical and robust distribution forecast framework that relies on backtest-based bootstrap and adaptive residual selection. The proposed approach is robust to the choice of the underlying forecasting model, accounts for uncertainty around the input covariates, and relaxes the independence between residuals and covariates assumption. It reduces the Absolute Coverage Error by more than 63% compared to the classic bootstrap approaches and by $2\% - 32\%$ compared to a variety of State-of-the-Art deep learning approaches on in-house product sales data and M4-hourly competition data.

Index Terms— forecasting, time series, bootstrap

1. INTRODUCTION

Time series forecasting is crucial in many industrial applications and enables data-driven planning [1–3], such as making production capacity or inventory allocation decisions based on demand forecast [4]. Planners or optimization systems that consume the forecast often require the estimated distribution of the response variable (referred to as distribution forecast, or DF) instead of only the estimated mean/median (referred to as point forecast, or PF) to make informed and nuanced decisions. An accurate DF method should ideally factor in different sources of forecast uncertainty, including uncertainty associated with parameter estimates and model misspecification [2]. Furthermore, when deploying a DF method in industrial applications, there are other important practical considerations such as ease of adoption, latency, interpretability, and robustness to model misspecification. To this end, we propose a practical and robust DF framework that uses backtesting [5] to build a collection of predictive residuals and an adaptive residual selector to pick the relevant residuals for bootstrapping DF. The proposed framework incorporates different sources of forecast uncertainty by design, integrates well with an arbitrary PF model to produce DF, is robust to model misspecification, has negligible inference time latency, retains interpretability for model diagnostics, and achieves more accurate coverage than the current State-of-the-Art DF methods in our experiments.

The contributions of this paper are as follows:

1. We propose a flexible plug-and-play framework that can extend an arbitrary PF model to produce DF.
2. We extend the existing approach of bootstrapping predictive residuals [6, 7] by using backtest and covariate sampling to improve efficiency and account for uncertainty in input covariates.
3. We propose an adaptive residual selector, which relaxes the independence between residuals and covariates assumption and boosts model performance.
4. We propose a new formula on how bootstrapped residuals are applied during forecasting, which scales the residuals w.r.t. the PF.
5. Lastly, we empirically evaluate the performance of various DF approaches on our in-house product sales data and the M4-hourly [8,9] competition data. The proposed DF approach reduces the Absolute Coverage Error by more than 63% compared to the classic bootstrap approaches and by $2\% - 32\%$ compared to a variety of State-of-the-Art deep learning approaches.

2. RELATED WORK

The existing approaches for DF include the following categories: 1. Using models that make parametric distribution assumptions around the response variable and estimate the associated parameters, such as state space models [4, 10–12] and estimating model uncertainty through posterior distributions in a Bayesian setting [13–18]; 2. Using models that explicitly minimize quantile loss and generate quantile forecast, such as Quantile Gradient Boosting [19], MQ-R(C)NN [20], and Temporal Fusion Transformers [21]; and 3. Using variations of bootstrap methods that sample residuals to generate multiple bootstrap forecasts and then compute sample quantiles of the bootstrap forecasts [6, 7, 22–25]. The bootstrap methods have the practical advantage of being able to integrate with an arbitrary PF model to obtain DF, but the classic bootstrap methods are usually designed under strong assumptions around the PF model and dataset, which might not work well with complex real-world data or modern machine learning PF models. The "delete-$\mathbf{x}^t$" approach of bootstrapping predictive residuals [6, 7] by design has less assumptions on

[1]Work done during internship at Amazon.

the data and is more robust to the choice of PF model; however, it only computes one predictive residual for each model training, ignores the uncertainty in covariates, and assumes that the residuals are independent from the covariates and PF. Our proposed DF framework is a significant generalization of the "delete-$\mathbf{x}^t$" approach and addresses each of the aforementioned limitations. Another closely related approach is the Level Set Forecaster [26], which is similar to the special case of our approach where the in-sample forecast is used for computing and selecting residuals.

3. METHOD

The proposed DF framework is composed of a backtester, a residual selector, and a PF model (Figure 1). To summarize how it works: During training: 1. Backtest [5] is performed on the training data with the PF model to build a collection of predictive residuals (Figure 2); for covariates that need to be estimated for future time points (e.g., future price of a product), their values can be sampled from estimated distributions during backtest to account for the uncertainty in covariates. 2. The residual selector is pre-specified or learned from the training data as a set of rules or separate machine learning model that selects the most relevant subset of predictive residuals given a future data point based on their meta information. 3. Lastly, the PF model is trained on the entire training data. During forecasting: 1. For each future data point of interest, the trained PF model generates the PF. 2. The residual selector selects a subset of residuals. 3. Lastly, random samples of residuals are drawn from the subset and applied to the PF to obtain multiple bootstrap forecasts that provide the empirical distribution, and sample quantiles of the bootstrap forecasts provide the quantile forecasts for arbitrary target quantiles. Essentially, we use the empirical distribution of the selected predictive residuals from backtest to estimate the distribution of the future predictive residuals and thus the distribution of the future response variable.

3.1. Backtesting

Let $\mathcal{D} = \{(\mathbf{X}_i^t, Y_i^t)\}_{i=1,2,\ldots,n}^{t=s_i,s_i+1,\ldots,d_i}$ be the training data, where $\mathbf{X}_i^t$ is the covariates matrix at time t, Y_i^t is the response variable at time t, s_i is the first time point, and d_i is the last time point for time series i. For nonparametric distribution forecast, it suffices to estimate the conditional quantiles $\widehat{Q}_{Y_i^{d_i+k_i}|Y_i^{s_i:d_i},\mathbf{X}_i^{s_i:d_i},\mathbf{X}_i^{(d_i+1):(d_i+k_i)}}(\tau)$ for arbitrary target quantile $\tau \in (0,1)$, where k_i is the number of time points into the future. Backtest is essentially a move-forward cross-validation that preserves the order in time for time series data, where the test split is always further in time than the training split. Let the backtest start time and step size be a and l respectively. For each split point $j = a, a+l, a+2l, \ldots, \max_i(d_i) - 1$, the training data are divided into a training split $\mathcal{A}_j = \{(\mathbf{X}_i^t, Y_i^t) \in \mathcal{D} \,|\, t \leq j\}$ and a test split $\mathcal{B}_j = \{(\mathbf{X}_i^t, Y_i^t) \in \mathcal{D} \,|\, t > j\}$; the PF model $\widehat{f}_j$ is trained on $\mathcal{A}_j$, and predictive residuals are computed as $\{Y_i^t - \widehat{f}_j(Y_i^{s_i:j}, \mathbf{X}_i^{s_i:j}, \mathbf{X}_i^{(j+1):t}) | (\mathbf{X}_i^t, Y_i^t) \in \mathcal{B}_j\}$. This process generates a collection of predictive residuals $\mathcal{E} = \{\varepsilon_{i,j}^t\}_{i,j,t}$. For those covariates that are not available in the future and need to be estimated, we can use their historic estimates, sample from their estimated distributions, or add simulated noise to create $\widetilde{\mathbf{X}}_i^t$ to replace $\mathbf{X}_i^t$ during backtest to account for uncertainty in covariates.

3.2. Selecting Residuals

Common PF models typically assume that the residuals are i.i.d. and independent from the covariates and the PF itself [2]. However, such assumption doesn't always hold in practice for the predictive residuals. E.g., the variance of residuals can increase as we forecast further into the future or as the magnitude of PF increases. To relax the commonly imposed independence assumption between residuals and covariates (or more generally any meta information which can include the PF or other variables not in the original covariates), an adaptive residual selector can be learned from the training data to select a subset of residuals based on the meta information of the predictive residuals from backtest and the future data point, $\widehat{g}(\mathcal{E}, \mathcal{M}, \mathcal{M}^{\text{future}})$, so that the selected residuals are conditionally i.i.d.. The residual selector should ideally be based on the meta information that has a non-negligible impact on the predictive residuals. We mention two options for learning the residual selector here: 1. Compute distance correlation (which can detect both linear and non-linear dependence) [27,28] between the predictive residuals from backtest and their corresponding meta information to identify variables with the highest distance correlation to the residuals. Then design rules (e.g., set simple thresholds) around these variables to select residuals that have a different distribution from the distribution of the entire collection of residuals, which can be verified by the Kolmogorov-Smirnov test [29]. Note that if the residual selector has no impact, the selected residuals should have the same distribution as the entire collection. 2. Fit a machine learning model, such as a regression decision tree, to predict residuals from their meta information, then apply the model to the meta information of future data points to select the corresponding residuals. The performance of this model can also be used to check dependence between meta information and residuals – if the residuals are already independent from the meta information pre-selection, then the model should perform poorly.

3.3. Bootstrapping

We describe two formulae of generating bootstrap forecasts, *Backtest-Additive* and *Backtest-Multiplicative*. They can be applied to either iterative or direct PF models (an iterative model recursively consumes the forecast from the previous time point to forecast for the next, whereas a direct model generates forecast for a future time point directly from covariates [30]). For Backtest-Additive, to generate bootstrap forecasts for the next time point $d_i + 1$, after obtaining the PF $\widehat{Y}_i^{d_i+1} = \widehat{f}(Y_i^{s_i:d_i}, \mathbf{X}_i^{s_i:d_i}, \mathbf{X}_i^{d_i+1})$ and the selected predictive residuals from backtest $\mathcal{G} = \widehat{g}(\mathcal{E}, \mathcal{M}, \mathcal{M}_i^{d_i+1})$, random samples are drawn from the selected residuals $\varepsilon_b \in \mathcal{G}$ for

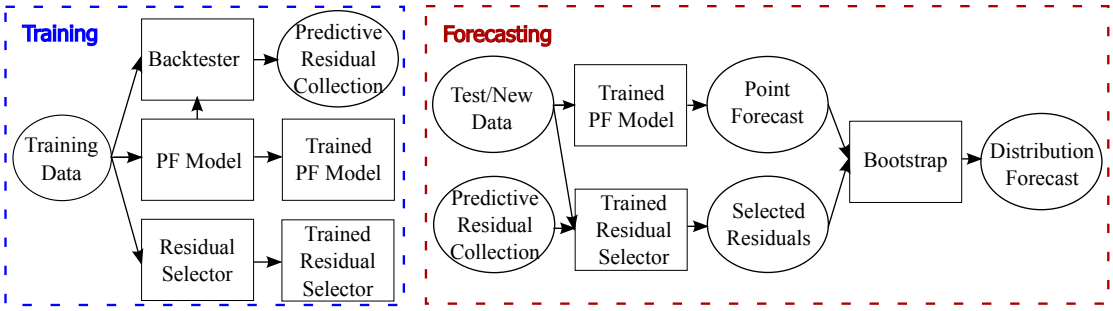

Fig. 1: Overview of the proposed DF framework. The backtester generates a collection of predictive residuals; the residual selector selects a subset of residuals for each future data point; the bootstrapping step combines the PF and selected residuals to generate DF.

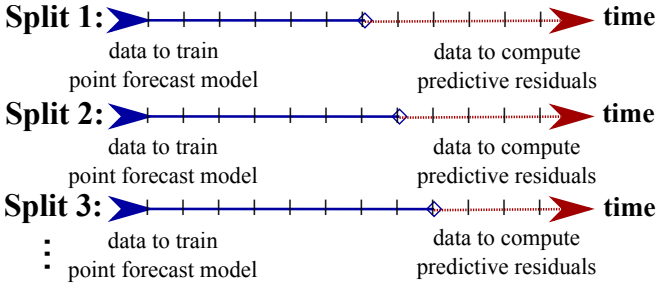

Fig. 2: Illustration of building a collection of predictive residuals with backtest. The training split is used to train the PF model, and the test split is used to compute the predictive residuals.

$b = 1, 2, \ldots, B$, then the bootstrap forecasts are given by $\widehat{Y}^{d_i+1}_{i,b,\text{Add.}} = \widehat{Y}^{d_i+1}_i + \varepsilon_b$. Quantile forecasts are obtained by taking sample quantiles of the bootstrap forecasts. Generalizing to arbitrary future time point $d_i + k_i$, for an iterative PF model, bootstrap forecasts are recursively generated for the next time point until $d_i + k_i$; for a direct PF model, the calculation remains the same as 1-step forecast with $d_i + 1$ replaced by $d_i + k_i$. Note that for a direct PF model, quantile forecasts can be obtained by skipping the residual sampling step and adding the sample quantiles of the selected residuals to the PF. The formula for Backtest-Additive is similar to the existing approach to bootstrapping predictive residuals [6, 7] (while the backtest and residual selection steps are novel in Backtest-Additive). The performance of Backtest-Additive can degrade if the variance of residuals increases with the magnitude of the PF, or if the magnitude of the future PF is very different from the magnitude of the response variable seen during backtest. Hence we also propose Backtest-Multiplicative, which scales the residuals based on the PF: After obtaining the PF and the selected residuals in the same way as Backtest-Additive, the error ratios are computed by dividing the extracted residuals over their corresponding forecast (or response variable) during backtest, $\mathcal{R} = \{\varepsilon^t_{h,j}/\widehat{Y}^t_{h,j} \,|\, \varepsilon^t_{h,j} \in \mathcal{G}\}$; then the bootstrap forecasts for the next time point are given by sampling $r_b \in \mathcal{R}$ and $\widehat{Y}^{d_i+1}_{i,b,\text{Multi.}} = \widehat{Y}^{d_i+1}_i \cdot (1 + r_b)$. The rest remains the same.

3.4. Practical Considerations

Both the backtest step and the residual selection step can be efficiently parallelized across multiple CPU's/GPU's. The backtest step requires multiple model training, but it is more efficient than the previous "delete-$\mathbf{x}^t$" approach of bootstrapping predictive residuals [6, 7] and can be done offline at a lower frequency than updating the PF model. The only computational overhead during inference time is the (optional) residual selection given the PF, so the additional latency of obtaining DF is negligible. Furthermore, once a residual collection from backtest is built, quantile forecast for any target quantile can be generated without re-running backtest or re-training the PF model, whereas DF methods that explicitly minimize quantile loss typically require the target quantile to be specified before model training. The backtest-based methods are also relatively interpretable: They retain the interpretability of the underlying PF model if the PF model is interpretable; even with a less interpretable PF model, one can check the predictive residual distribution and model performance on the test split (and model coefficients if applicable) during the backtest step to help identify which data points or covariates tend to contribute to large predictive residuals and whether the model has systematic bias during out-of-sample forecasting. The choices of bootstrap formula (Backtest-Additive vs Backtest-Multiplicative), denominator of error ratios (backtest forecast vs observed response variable), residual selector variation, and PF model can be tuned as hyperparameters.

4. EXPERIMENTS

We conduct experiments on two real-world time-series datasets: an in-house product sales dataset and the M4-hourly competition dataset [8, 9]. The product sales dataset consists of daily sales of 76 products between 01/01/2017 and 01/10/2021 and 147 covariates capturing information on pricing, supply constraints, trend, seasonality, special events, and product attributes. The standard Absolute Coverage Error (ACE) is used to evaluate the DF performance: The coverage (CO) of quantile forecast $\widehat{Y}^t_{i(\tau)}$ for target quantile τ over the test set $\mathcal{D}_{\text{test}}$ is defined as $\text{CO}(\mathcal{D}_{\text{test}}; \tau) = \frac{1}{|\mathcal{D}_{\text{test}}|} \sum_{\mathcal{D}_{\text{test}}} I\{Y^t_i \le \widehat{Y}^t_{i(\tau)}\}$; and ACE is defined as $\text{ACE}(\mathcal{D}_{\text{test}}; \tau) = |\text{CO}(\mathcal{D}_{\text{test}}; \tau) - \tau|$. (We also track other metrics such as quantile loss, weighted quantile loss, and weighted prediction interval width; the conclusions from different metrics are overall consistent.) A 100-fold backtest is used for evaluation, which is separate

from the backtest used for computing predictive residuals – in each training-test split for evaluation, the latter half of the training split is used to perform a separate backtest to build the predictive residual collection without using information from the test split for a fair evaluation. The reported ACE is averaged across all training-test splits, 24-week forecast horizon for product sales and 48-hour horizon for M4-hourly, and the following range of target quantiles: $\tau = 0.1, 0.2, \ldots, 0.9$. For experiments with deep learning models, the reported ACE is also averaged across 10 trials due to the fluctuation in model performance.

Compared to other DF approaches, bootstrap approaches have the advantage of extending any PF model to produce DF, which makes them easy to adopt and able to potentially retain desired properties of the PF model. Thus, the first experiment focuses on comparing the proposed Backtest-Additive (BA) and Backtest-Multiplicative (BM) against classic bootstrap approaches for DF: bootstrap with fitted residuals (FR) [2] and bootstrap with fitted models (FM) [7, 22]. This experiment is performed on the product sales dataset, as it contains covariates which can accommodate the use of standard Machine Learning models as direct PF models. A variety of PF models are used to assess the bootstrap approaches' robustness to the choice of PF model, including Ridge Regression [19], Support Vector Regression (SVR) [19], Random Forest (RF) [19], and Feed-forward Neural Networks (NN) [19]. The proposed bootstrap approaches outperform the classic approaches for all PF models (Table 1).

The second experiment compares against other State-of-the-Art DF approaches, including Quantile Lasso (QLasso) [19], Quantile Gradient Boosting (QGB) [19], DeepAR [3, 9], Deep Factors (DFact) [9, 31], MQ-CNN [9, 20], Deep State Space Models (DSSM) [4, 9], and Temporal Fusion Transformers (TFT) [9, 21]. Because the bootstrap approaches require an underlying PF model, for a fair comparison we use the median forecast from each of the aforementioned benchmarks as the PF models to be integrated with the backtest-based bootstrap, so they share the same model architecture and hyperparameters. The comparison against QGB and QLasso is performed on the product sales data and the comparison against the deep learning models is performed on the M4-hourly data[1]. The proposed bootstrap approaches integrated with the median forecast outperform the default DF from the benchmarks (Table 2).

The third experiment assesses the robustness of the proposed approaches to model assumptions/hyperparameters. DeepAR requires the output distribution to be specified prior to the model learning its parameters. In this experiment the backtest-based bootstrap approaches integrated with the median forecast are compared against the default DF from DeepAR under a variety of output distribution assumptions on the M4-hourly data. The proposed approaches outperform the default DF in 5 out of 6 distribution settings (Table 3).

The median or mean forecast from the bootstrap approaches can be viewed as the updated PF through Bootstrap Aggregating (Bagging). As an ensemble output, the Bagging PF can be potentially more accurate than the original PF. The fourth experiment evaluates the relative change in Mean Absolute Percentage Error (MAPE) of the Bagging PF compared to the original PF on the product sales data. The Bagging PF from the proposed approaches achieves the greatest reduction in MAPE (Table 4) for all PF models; i.e., in addition to providing DF, the proposed approaches can also provide more accurate PF. One explanation is that if a PF model has systematic bias during backtest, its predictive residual distribution will reflect such bias, so by design the median forecast will correct for the bias from backtest when bootstrapping DF (Section 3.3).

Table 1: ACE comparison of different bootstrap DF approaches integrated with different PF models.

Bootstrap\PF	Ridge	SVR	RF	NN
FR	0.102(−0%)	0.195(−0%)	0.207(−0%)	0.176(−0%)
FM	0.095(−7%)	0.218(+12%)	0.171(−17%)	0.125(−29%)
BA	0.069(−32%)	0.065(−67%)	0.055(−73%)	0.077(−56%)
BM	**0.038**(−**63**%)	**0.061**(−**69**%)	**0.027**(−**87**%)	**0.048**(−**73**%)

Table 2: ACE comparison of backtest-based bootstrap integrated with the median forecast vs the default DF.

DF\Model	QLasso	QGB	DeepAR	DFact	MQCNN	DSSM	TFT
Default	0.188	0.119	0.102	0.098	0.092	0.136	0.067
Median + BA	0.114	0.078	**0.100**	**0.067**	0.078	0.124	**0.058**
Median + BM	**0.039**	**0.036**	0.104	0.070	**0.071**	**0.112**	0.060

Table 3: ACE comparison of backtest-based bootstrap integrated with the median forecast vs the default DF from DeepAR under different pre-specified output distributions.

DF\Output Dist.	Neg. Bin.	Student's t	Normal	Gamma	Laplace	Poisson
Default	0.102	0.192	0.162	**0.138**	0.114	0.134
Median + BA	**0.100**	0.169	0.116	0.157	0.094	0.128
Median + BM	0.104	**0.165**	**0.111**	0.156	**0.088**	**0.125**

Table 4: Relative change in MAPE for Bagging PF compared to the original PF.

Bootstrap\PF Model	Ridge	SVR	RF	NN
FR	+0.8%	+6.5%	+0.2%	+0.7%
FM	+0.4%	+6.6%	−3.8%	+2.6%
BA	−12.3%	−21.0%	−**10.0**%	+1.5%
BM	−**22.1**%	−**31.8**%	−5.3%	−**13.4**%

5. CONCLUSION

This paper proposes a robust DF framework with backtest-based bootstrap and adaptive residual selection. It can efficiently extend an arbitrary PF model to generate DF, is robust to the choice of model, and outperforms a variety of benchmark DF methods on real-world data, making the proposed framework well-suited for industrial applications.

[1] Note on the data: QGB and QLasso are not traditional time series models and require engineered covariates available in the product sales data but not in M4-hourly, while the deep learning models implemented in GluonTS [9] package require the conditioning and forecast horizons fixed for all time series and the product sales data contain time series of varying lengths.

6. REFERENCES

[1] Paul D Larson, "Designing and managing the supply chain: concepts, strategies, and case studies," *Journal of Business Logistics*, vol. 22, no. 1, pp. 259, 2001.

[2] Rob J Hyndman and George Athanasopoulos, *Forecasting: principles and practice*, OTexts: Melbourne, Australia, 2 edition, 2018.

[3] David Salinas, Valentin Flunkert, Jan Gasthaus, and Tim Januschowski, "DeepAR: Probabilistic forecasting with autoregressive recurrent networks," *International Journal of Forecasting*, vol. 36, no. 3, pp. 1181–1191, 2020.

[4] Syama Sundar Rangapuram, Matthias W Seeger, Jan Gasthaus, Lorenzo Stella, Yuyang Wang, and Tim Januschowski, "Deep state space models for time series forecasting," in *Advances in Neural Information Processing Systems*. 2018, vol. 31, Curran Associates, Inc.

[5] David H Bailey, Jonathan Borwein, Marcos Lopez de Prado, and Qiji Jim Zhu, "The probability of backtest overfitting," *Journal of Computational Finance, forthcoming*, 2016.

[6] Dimitris N. Politis, "Model-free model-fitting and predictive distributions," *TEST*, vol. 22, no. 2, pp. 183–221, 2013.

[7] Li Pan and Dimitris N Politis, "Bootstrap prediction intervals for linear, nonlinear and nonparametric autoregressions," *Journal of Statistical Planning and Inference*, vol. 177, pp. 1–27, 2016.

[8] Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos, "The M4 competition: Results, findings, conclusion and way forward," *International Journal of Forecasting*, vol. 34, no. 4, pp. 802–808, 2018.

[9] Alexander Alexandrov, Konstantinos Benidis, Michael Bohlke-Schneider, Valentin Flunkert, Jan Gasthaus, Tim Januschowski, Danielle C. Maddix, Syama Rangapuram, David Salinas, Jasper Schulz, Lorenzo Stella, Ali Caner Turkmen, and Yuyang Wang, "GluonTS: Probabilistic and neural time series modeling in python," *Journal of Machine Learning Research*, vol. 21, no. 116, pp. 1–6, 2020.

[10] Rob Hyndman, Anne B Koehler, J Keith Ord, and Ralph D Snyder, *Forecasting with exponential smoothing: the state space approach*, Springer Science & Business Media, 2008.

[11] James Durbin and Siem Jan Koopman, *Time series analysis by state space methods*, Oxford university press, 2012.

[12] Matthias Seeger, David Salinas, and Valentin Flunkert, "Bayesian intermittent demand forecasting for large inventories," in *Proceedings of the 30th International Conference on Neural Information Processing Systems*, 2016, pp. 4653–4661.

[13] David B Dunson and Jack A Taylor, "Approximate bayesian inference for quantiles," *Journal of Nonparametric Statistics*, vol. 17, no. 3, pp. 385–400, 2005.

[14] Brian J Reich, Howard D Bondell, and Huixia J Wang, "Flexible bayesian quantile regression for independent and clustered data," *Biostatistics*, vol. 11, no. 2, pp. 337–352, 2010.

[15] Yunwen Yang and Xuming He, "Bayesian empirical likelihood for quantile regression," *The Annals of Statistics*, vol. 40, no. 2, pp. 1102–1131, 2012.

[16] Yunwen Yang, Huixia Judy Wang, and Xuming He, "Posterior inference in bayesian quantile regression with asymmetric laplace likelihood," *International Statistical Review*, vol. 84, no. 3, pp. 327–344, 2016.

[17] Yarin Gal and Zoubin Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," in *Proceedings of the 33rd International Conference on Machine Learning*. 2016, ICML'16, pp. 1050–1059, JMLR.org.

[18] Lingxue Zhu and Nikolay Laptev, "Deep and confident prediction for time series at Uber," in *2017 IEEE International Conference on Data Mining Workshops (ICDMW)*. IEEE, 2017, pp. 103–110.

[19] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[20] Ruofeng Wen, Kari Torkkola, Balakrishnan Narayanaswamy, and Dhruv Madeka, "A multi-horizon quantile recurrent forecaster," *NeurIPS, Time Series Workshop*, 2017.

[21] Bryan Lim, Sercan Ö. Arık, Nicolas Loeff, and Tomas Pfister, "Temporal fusion transformers for interpretable multi-horizon time series forecasting," *International Journal of Forecasting*, vol. 37, no. 4, pp. 1748–1764, 2021.

[22] Jeremy Berkowitz and Lutz Kilian, "Recent developments in bootstrapping time series," *Econometric Reviews*, vol. 19, no. 1, pp. 1–48, 2000.

[23] Tata Subba Rao, Suhasini Subba Rao, and C.R. Rao, *Time Series Analysis: Methods and Applications*, North Holland, 1 edition, 2012.

[24] Gerard Keogh, "The splice bootstrap," *arXiv preprint arXiv:1311.5828*, 2013.

[25] Todd A. Kuffner, Stephen M. S. Lee, and G. Alastair Young, "Optimal hybrid block bootstrap for sample quantiles under weak dependence," *arXiv preprint arXiv:1710.02537*, 2017.

[26] Hilaf Hasson, Yuyang (Bernie) Wang, Tim Januschowski, and Jan Gasthaus, "Probabilistic forecasting: A level-set approach," *NeurIPS*, 2021.

[27] Gábor J. Székely and Maria L. Rizzo, "Brownian distance covariance," *The Annals of Applied Statistics*, vol. 3, no. 4, pp. 1236 – 1265, 2009.

[28] Longshaokan Wang, *Sufficient Markov Decision Processes*, Doctoral dissertation, North Carolina State University, 2018.

[29] A. Kolmogorov, "Sulla determinazione empirica di una lgge di distribuzione," *Inst. Ital. Attuari, Giorn.*, 1933.

[30] Bryan Lim and Stefan Zohren, "Time-series forecasting with deep learning: a survey," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 379, no. 2194, 2021.

[31] Yuyang Wang, Alex Smola, Danielle C. Maddix, Jan Gasthaus, Dean Foster, and Tim Januschowski, "Deep factors for forecasting," *ICML*, 2019.