Neural time series models with GluonTS
Time Series Workshop ICML 2019

Alexander Alexandrov [* 1] **Konstantinos Benidis** [* 1] **Michael Bohlke-Schneider** [* 1] **Valentin Flunkert** [* 1] **Jan Gasthaus** [* 1] **Tim Januschowski** [* 1] **Danielle Maddix** [* 2] **Syama Rangapuram** [* 1] **David Salinas** [* 1] **Jasper Schulz** [* 1] **Lorenzo Stella** [* 1] **Ali Caner Türkmen** [* 2] **Yuyang Wang** [* 2]

Abstract

We introduce GluonTS, the Gluon Time Series Toolkit, a library for deep learning based time series modeling. GluonTS simplifies the development of and experimentation with time series models for common tasks such as forecasting or anomaly detection. It provides all necessary components and tools that scientists need for quickly building new models, for efficiently running and analyzing experiments and for evaluating model accuracy.

1. Introduction

Large collections of time series are ubiquitous and occur in areas as different as natural and social sciences, internet of things applications, cloud computing, supply chains and many more. Traditionally, time series modeling has focused (mostly) on individual time series via *local* models.[1] A number of commercial and open-source toolkits exist for this (Hyndman & Khandakar, 2008; Taylor & Letham, 2017; Scott & Varian, 2014).

In recent years, advances in deep learning have led to accuracy improvements over the local approach by utilizing the large amounts of data available for estimating parameters of a single *global* model over the entire collection of time series (Flunkert et al., to appear; Wen et al., 2017; Laptev et al., 2017). Deep learning frameworks, such as (Chen et al., 2015; Paszke et al., 2017; Abadi et al., 2016) are growing in popularity. In recent years, more application-specific toolkits have appeared (Hieber et al., 2018; Dai et al., 2018; Bingham et al., 2018). For time series modeling, however, there exists, to the best of our knowledge, currently no dedicated deep learning toolkit.

We fill this gap with GluonTS (`https://gluon-ts.mxnet.io/index.html`) – a deep learning based library that bundles components, models and tools for time series applications such as forecasting or anomaly detection. GluonTS simplifies all aspects of scientific experiments with time series models. It includes components such as distributions, neural network architectures for sequences, and feature processing steps which can be used to quickly assemble and train new models. Apart from supporting pure deep learning based models, GluonTS also includes probabilistic models and components such as State-Space Models and Gaussian Processes (Rangapuram et al., 2018; Krishnan et al., 2017; Fraccaro et al., 2016). The library is based on the *Gluon* API[2] of the MxNet deep learning framework (Chen et al., 2015).

2. Library design and components

GluonTS supports researchers in their experiments through modular design and well tested scalable components. Experiments are reproducibly, since all details of the configuration, such as parameter values, can be logged and the experiment can be re-created from the log.

Listing 1 shows a simple workflow for creating and training a pre-built forecasting model, and evaluating the model in a backtest. GluonTS contains a set of time series specific transformations that include splitting and padding of time series (e.g. for evaluation splits), common time series transformation such as Box-Cox transformations or marking of special points in time and missing values. A user can easily include custom transformations for specific purposes, and combine them with existing transformations in a pipeline.

GluonTS also provides a flexible abstraction for probabil-

[*]Equal contribution [1]Amazon Research, Berlin, Germany [2]Amazon Research, Palo Alto, USA. Correspondence to: Valentin Flunkert <flunkert@amazon.com>, Tim Januschowski <tjnsch@amazon.com>.

Proceedings of the 36th International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019. Copyright 2019 by the author(s).

[1]In local models, the free parameters of the time series model are estimated per individual time series in the collection of time series.

[2]`https://mxnet.apache.org/versions/master/gluon/index.html`

Listing 1 Model training and evaluation in GluonTS

```
from gluonts.dataset import load_dataset
from gluonts.model.deepar import DeepAREstimator
from gluonts.trainer import Trainer
from gluonts.evaluation import Evaluator
from gluonts.evaluation.backtest import backtest_metrics

meta, train_ds, test_ds = load_dataset('./mydataset')
estimator = DeepAREstimator(
  freq=meta.freq,
  prediction_length=100,
  batch_size=32,
  trainer=Trainer(epochs=20)
)
predictor = estimator.train(train_ds)
evaluator = Evaluator(quantiles=(0.1, 0.5, 0.9))
agg_metrics, item_metrics = backtest_metrics(
        train_dataset=train_ds,
        test_dataset=test_ds,
        estimator=predictor,
        evaluator=evaluator
)
```

ity distributions (and densities), which are common building blocks in probabilistic time series modeling. Concrete implementations include common parametric distributions, such as Gaussian, Student's t, gamma, and negative binomial as well as more complex transformed distributions (Dillon et al., 2017; Rezende & Mohamed, 2015).

Trained forecast models (i.e. predictors) return `Forecast` objects as predictions, which contain a time index (start, end and time granularity) and a representation of the probability distribution of values over the time index. Different models may use different techniques for estimating and representing this joint probability distribution, such as sample paths for auto-regressive models (Seeger et al., 2016; Flunkert et al., to appear) or a collection of quantiles (Wen et al., 2017; Gasthaus et al., 2019).

Forecast objects in GluonTS have a common interface that allows the `Evaluation` component to compute accuracy metrics such as quantile loss, Mean Absolute Scale Error (MASE), Mean Absolute Percent Error (MAPE) and Scaled Mean Absolute Percent Error (sMAPE) using simple or complex backtest scenarios (e.g. rolling evaluations) For qualitatively assessing the accuracy of time series models, GluonTS contains methods that visualize time series and forecasts using matplotlib (Hunter, 2007).

While GluonTS can be used directly on a laptop, training and prediction can also be scaled up and out through integration with Amazon SageMaker. SageMaker is a managed machine learning service on AWS that handles model training as well as predictions.

3. Time Series Problems & Models

GluonTS addresses probabilistic modeling of uni- or multivariate sequences of (large) collections of time series. Important applications include forecasting, smoothing and anomaly detection. More formally, let $Z = \{z_{i,1:T_i}\}_{i=1}^{N}$ be a set of N univariate time series, where $z_{i,1:T_i} := (z_{i,1}, z_{i,2}, \ldots, z_{i,T_i})$, and $z_{i,t} \in \mathbb{R}$ denotes the value of the i-th time series at time t. We mainly consider time series where the time points are equally spaced but the time units across different sets can be arbitrary (e.g. hours, days, months). Furthermore, let $X = \{\mathbf{x}_{i,1:T_i+\tau}\}_{i=1}^{N}$ be a set of associated, time-varying covariate vectors with $\mathbf{x}_{i,t} \in \mathbb{R}^D$.

The goal of *forecasting* (Hyndman & Athanasopoulos, 2017) is to predict the probability distribution $p\left(z_{i,T_i+1:T_i+\tau} \mid z_{i,1:T_i}, \mathbf{x}_{i,1:T_i+\tau}; \Phi\right)$ of future values $z_{i,T_{i+1}:T_{i+\tau}}$ given the past values $z_{i,1:T_i}$, the covariates $\mathbf{x}_{i,1:T_{i+\tau}}$, and the model parameters Φ:

Smoothing or *missing value imputation* in time series can leverage the sequence structure, and therefore allow for more sophisticated approaches compared to the general missing value imputation setting (e.g., (Biessmann et al., 2018)). Smoothing is similar to forecasting, except that the time points that we want to predict do not lie in the future. Instead, for a set of series and *arbitrary* time points there are missing or unobserved values, and the goal is to estimate the (joint) probability distribution over these missing values.

In *anomaly* or *outlier detection* we want to identify time points, where the observed values are unlikely to have occurred. While we may have additional labels that annotate anomalous behavior, it is in many applications not feasible to directly train a classifier on these labels, because the labels are too sparse – after all, anomalies are often rare. In this unsupervised case, anomaly detection is similar to forecasting, except that all values are observed and we want to know how likely they were. A probabilistic forecasting model can be converted into an anomaly detection model in different ways. For univariate models where the cumulative distribution function (CDF) of the marginal predicted distribution can be evaluated, one can directly calculate the p-value (tail probability) of a newly observed data point via the CDF (Shipmon et al., 2017). This allows us to mark unlikely points or unlikely sequences of observations as anomalies.

All these tasks from have at their core the estimation of a joint probability distribution over time series values at different time points. We model this as a supervised learning problem, by fixing a model structure upfront and learning the model parameters Φ using a statistical optimization method, such as maximum likelihood estimation, and the sets Z and X as training data.

Classical models that were developed for these tasks (Hyndman & Athanasopoulos, 2017), are, with some exceptions (Chapados, 2014), local models that learn the parameters Φ for each time series individually. Recently, however, several neural time series models have been proposed (Flunkert et al., to appear; Gasthaus et al., 2019; Rangapuram et al., 2018; Laptev et al., 2017; Wen et al., 2017) where a single global model is learned for all time series in the dataset by sharing the parameters Φ.

Time series models can be broadly categorized as generative and discriminative, depending on how the target Z is modeled (Ng & Jordan, 2002).[3] Generative models assume that the given time series are generated from an unknown stochastic process $p(Z|X;\Phi)$ given the covariates X. Prominent examples include classical models such as ARIMA and ETS (Hyndman et al., 2008), Bayesian structural time series (BSTS) (Scott & Varian, 2014) and the recently proposed deep state space model (Rangapuram et al., 2018). The unknown parameters Φ of this stochastic process are typically estimated by maximizing the likelihood, which is the probability of the observed time series, $\{z_{i,1:T_i}\}$, under the model $p(Z|X;\Phi)$, given the covariates $\{\mathbf{x}_{i,1:T_i}\}$. Once the parameters Φ are learned, the forecast distribution can be obtained from $p(Z|X;\Phi)$. In contrast to ETS and ARIMA, which learn Φ per time series individually, neural generative models like (Rangapuram et al., 2018) further express Φ as a function of a neural network whose weights are shared among all time series and learned from the whole training data.

Discriminative models such as (Flunkert et al., to appear; Gasthaus et al., 2019; Wen et al., 2017), model the conditional distribution (for a fixed τ) directly via a neural network. Compared to generative models, conditional models are more flexible, since they make less structural assumptions, and hence are also applicable to a broader class of application domains.

We describe the generative model forecasting methods that implemented in GluonTS: State Space Models, Deep State Space Models and Gaussian Processes. Further examples in this family include Deep Factor models (Maddix et al., 2018), which we omit due to space restrictions.

State Space Models (SSMs) provide a principled framework for modeling and learning time series patterns (Hyndman et al., 2008; Durbin & Koopman, 2012; Seeger et al., 2016). In particular, SSMs model the temporal structure of the data via a latent state $\boldsymbol{l}_t \in \mathbb{R}^D$ that can be used to encode time series components, such as level, trend, and seasonality patterns. A general SSM is described by the so-called state-transition equation, defining the stochastic transition dynamics $p(\boldsymbol{l}_t|\boldsymbol{l}_{t-1})$ by which the latent state evolves over time, and an observation model specifying the conditional probability $p(z_t|\boldsymbol{l}_t)$ of observations given the latent state. Several classical methods (e.g., ETS, ARIMA) can be cast under this framework by choosing appropriate transition dynamics and observation model (Hyndman et al., 2008).

Deep State Space Models (Rangapuram et al., 2018) (referred as DeepState here) is a probabilistic time series forecasting approach that combines state space models with deep learning. The main idea is to parametrize the linear SSM using a recurrent neural network (RNN) whose weights are learned jointly from a dataset of raw time series and associated covariates.

Gaussian Processes (GPs) are popular non-parametric Bayesian methods for time series modeling (Girard et al., 2003). A prior is specified on the stochastic process of the time series $f(\cdot)$, using a user-defined mean function $m(\cdot)$ and covariance structure $\mathcal{K}(\cdot,\cdot)$. Given the target values, one performs inference on the function space specified by the covariance function, and obtains the posterior distribution of the underlying function that is assumed to generate the time series. In GluonTS, GPs (exact inference) with radial basis function (RBF) and periodic kernels are included.

Inspired by the seq-to-seq approach presented in (Sutskever et al., 2014), several forecasting methods of this type have been proposed (Wen et al., 2017). GluonTS contains a flexible seq-to-seq framework that makes it possible to combine generic encoder and decoder networks to create custom sequence-to-sequence models. Moreover, GluonTS also has example implementations of specific seq-to-seq models (Wen et al., 2017) as well as generic models (Vaswani et al., 2017).

Neural Quantile Regression Models. Quantile regression (Koenker, 2005) is a technique for directly predicting a particular quantile of a dependent variable. These techniques have been combined with deep learning and employed in the context of time series forecasting (Xu et al., 2016; Wen et al., 2017). We have implemented variants of such quantile decoder models in GluonTS following (Wen et al., 2017) and combined them with RNN and Dilated Causal Convolution (CNN) encoders, resulting in models dubbed RNN-QR and CNN-QR, respectively.

Transformer. GluonTS also contains an implementation of the Transformer architecture (Vaswani et al., 2017) that has been successful in natural language processing. The Transformer model captures the dependencies of a sequence by relying entirely on attention mechanisms. The elimina-

[3]Note that our categorization overloads the distinction used in Machine Learning. Technically, neither of the category defined here jointly models the covariates X and the target Z and thus both belong to "discriminative" models in the traditional sense.

tion of the sequential computation makes the representation of each time step independent of all other time steps and therefore allows the parallelization of the computation.

Auto-regressive models reduce the sequence prediction task fully to a one-step-ahead problem. The model is trained on a sequence by sequential one-step-ahead predictions, and the error is aggregated over the sequence for the model update. For prediction, the model is propagated forward in time by feeding in samples, through multiple simulations, a set of sample-paths representing the joint probability distribution over the future of the sequence is obtained.

NPTS. The Non-Parametric Time Series forecaster (NPTS) (Gasthaus, 2016) falls into the class of simple forecasters that use one of the past observed targets as the forecast for the current time step. Unlike the naive or seasonal naive forecasters that use a fixed time index (the previous index $T-1$ or the past season $T-\tau$) as the prediction for the time step T, NPTS randomly samples a time index $t \in \{0, \ldots, T-1\}$ in the past to generate a prediction sample for the current time step T. By sampling multiple times, one obtains a Monte Carlo sample from the predictive distribution, which can be used e.g. to compute prediction intervals.

DeepAR. GluonTS contains an auto-regressive RNN time series model, similar to the architectures described in (Flunkert et al., to appear; Gasthaus et al., 2019). DeepAR consists of a RNN (either using LSTM or GRU cells) that takes the previous time points and co-variates as input. DeepAR then either estimates parameters of a parametric distribution or a highly flexible parameterization of the quantile function.

Wavenet (van den Oord et al., 2016) is an auto-regressive neural network with dilated causal convolutions at its core. In the set of GluonTS models, it represents the archetypical auto-regressive Convolutional Neural Network (CNN) models. While it was developed for speech synthesis tasks, it is in essence a time series model that can be used for time series modeling in other problem domains. In the text-to-speech application, the output is a bounded signal and in many implementations the value range is often quantized into discrete bins. This discretized signal is then modeled using a flexible softmax distribution that can represent arbitrary distributions over the discrete values, at the cost of discarding ordinal information.

4. Experiments

In this section we demonstrate the practical effectiveness of a subset of the forecast and anomaly detection models. Table 1 shows the CRPS (Gasthaus et al., 2019) accuracy of different forecast methods in GluonTS on the following 11 public datasets: daily difference time series of log-return of stocks from S&P500; hourly time series of the electricity consumption of 370 customers (Dheeru & Karra Taniskidou, 2017); 6 datasets from the M4 competition (Makridakis et al., 2018) with daily, hourly, weekly, monthly, quaterly and yearly frequencies; monthly demand for car parts used in (Seeger et al., 2016); hourly occupancy rate, between 0 and 1, of 963 car lanes of San Francisco bay area freeways (Dheeru & Karra Taniskidou, 2017); and daily traffic of 10K Wikipedia pages.

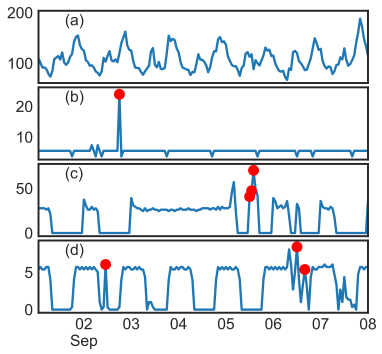

Figure 1. Examples of anomalies detected using a trained DeepAR forecast model on the electricity dataset.

The hyperparameters of each methods are kept constant across all datasets and we train with 5000 gradient updates the neural networks models.

Regarding accuracy, there is no overall dominating method. Hence, the experiment illustrates the need for a flexible modeling toolkit, such as GluonTS, that allows to assemble models quickly for the dataset and application at hand.

Fig. 1 demonstrate how a GluonTS forecast model (in this case DeepAR) can be used to detect anomalies (Shipmon et al., 2017) – in this case points that do not match the series historical behavior such as seasonality or noise level.

5. Conclusion

We introduced GluonTS, a toolkit for building time series models based on deep learning and probabilistic modeling techniques. By offering tooling and abstractions such as probabilistic models, basic neural building blocks, human-readable model logging for increased reproducability and unified I/O & evaluation, GluonTS allows scientists to rapidly develop new time series models for common tasks such as forecasting or anomaly detection.

GluonTS's pre-bundled implementations of state-of-the-art models allow easy benchmarking of new algorithms. We demonstrated this in a large scale experiment of running the pre-bundled models on different datasets and comparing their accuracy with classical approaches. Such experiments

estimator dataset	**Auto-ARIMA**	**Auto-ETS**	**Prophet**	**NPTS**	**Transformer**	**CNN-QR**	**DeepAR**	**DeepAR-Spl**	**GP**
SP500-returns	0.975±0.000	0.982±0.001	0.985±0.001	**0.832±0.000**	0.836±0.001	0.906±0.006	0.838±0.003	0.836±0.005	0.858±0.000
electricity	0.056±0.000	0.067±0.000	0.094±0.000	**0.055±0.000**	0.062±0.001	0.081±0.003	0.065±0.007	0.065±0.009	0.111±0.001
m4-Daily	0.024±0.000	**0.023±0.000**	0.090±0.000	0.145±0.000	0.028±0.000	0.026±0.001	0.028±0.000	0.025±0.002	0.074±0.000
m4-Hourly	0.040±0.001	0.044±0.000	0.043±0.000	0.048±0.000	0.042±0.010	0.064±0.006	**0.035±0.006**	0.124±0.038	0.132±0.000
m4-Monthly	**0.097±0.000**	0.099±0.000	0.132±0.000	0.233±0.000	0.134±0.002	0.127±0.002	0.136±0.002	0.106±0.001	0.242±0.000
m4-Quarterly	0.080±0.000	**0.078±0.000**	0.123±0.000	0.255±0.000	0.095±0.003	0.091±0.000	0.091±0.001	0.081±0.002	0.335±0.000
m4-Weekly	0.050±0.000	0.051±0.000	0.108±0.000	0.296±0.001	0.075±0.005	0.056±0.000	0.072±0.001	**0.043±0.001**	0.137±0.000
m4-Yearly	0.124±0.000	0.126±0.000	0.156±0.000	0.355±0.000	0.127±0.004	0.121±0.000	0.121±0.001	**0.111±0.002**	0.455±0.000
parts	1.401±0.002	1.342±0.002	1.637±0.002	1.356±0.002	1.000±0.003	**0.901±0.000**	0.970±0.005	0.943±0.025	1.591±0.000
traffic	-	0.462±0.000	0.273±0.000	0.162±0.000	0.132±0.010	0.186±0.002	0.127±0.004	**0.087±0.001**	0.152±0.000
wiki10k	0.610±0.001	0.841±0.001	0.681±0.000	0.452±0.000	0.294±0.008	0.314±0.002	0.295±0.028	**0.273±0.007**	0.452±0.000

Table 1. CRPS error for all methods (hyperparameters are fixed for each method between datasets). Ten runs are done for each method the mean and std are computed over 10 runs. Missing value indicates the method did not complete in 24 hours. DeepAR and Transformer use a Student's-t distribution, DeepAR-Spl uses a quantile regression spline output. The GP uses the RBF kernel.

are a first step towards a more thorough understanding of neural architectures for time series modelling.

References

Abadi, M. et al. Tensorflow: A system for large-scale machine learning. In *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation*, OSDI'16, pp. 265–283, Berkeley, CA, USA, 2016. USENIX Association. ISBN 978-1-931971-33-1.

Biessmann, F., Salinas, D., Schelter, S., Schmidt, P., and Lange, D. "deep" learning for missing value imputationin tables with non-numerical data. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, CIKM '18, pp. 2017–2025, New York, NY, USA, 2018. ACM. ISBN 978-1-4503-6014-2.

Bingham, E. et al. Pyro: Deep Universal Probabilistic Programming. *Journal of Machine Learning Research*, 2018.

Chapados, N. Effective Bayesian modeling of groups of related count time series. In *Proceedings of The 31st International Conference on Machine Learning*, pp. 1395–1403, 2014.

Chen, T. et al. Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems. *arXiv preprint arXiv:1512.01274*, 2015.

Dai, Z., Meissner, E., and Lawrence, N. D. Mxfusion: A modular deep probabilistic programming library. In *NIPS Workshop MLOSS (Machine Learning Open Source Software)*, 2018.

Dheeru, D. and Karra Taniskidou, E. UCI machine learning repository, 2017. URL `http://archive.ics.uci.edu/ml`.

Dillon, J. V. et al. Tensorflow distributions. *CoRR*, abs/1711.10604, 2017.

Durbin, J. and Koopman, S. J. *Time series analysis by state space methods*, volume 38. OUP Oxford, 2012.

Flunkert, V., Salinas, D., Gasthaus, J., and Januschowski, T. Deepar: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting*, to appear.

Fraccaro, M., Sønderby, S. K., Paquet, U., and Winther, O. Sequential neural models with stochastic layers. In *Advances in neural information processing systems*, pp. 2199–2207, 2016.

Gasthaus, J. Non parametric time series forecaster. Technical report, Amazon, 2016.

Gasthaus, J. et al. Probabilistic forecasting with spline quantile function rnns. In *AISTATS*, 2019.

Girard, A., Rasmussen, C. E., Candela, J. Q., and Murray-Smith, R. Gaussian process priors with uncertain inputs application to multiple-step ahead time series forecasting. In *Advances in neural information processing systems*, pp. 545–552, 2003.

Hieber, F. et al. The sockeye neural machine translation toolkit at AMTA 2018. In *AMTA (1)*, pp. 200–207. Association for Machine Translation in the Americas, 2018.

Hunter, J. D. Matplotlib: A 2d graphics environment. *Computing In Science & Engineering*, 9(3):90–95, 2007.

Hyndman, R., Koehler, A. B., Ord, J. K., and Snyder, R. D. *Forecasting with Exponential Smoothing: The State Space Approach*. Springer Series in Statistics. Springer, 2008. ISBN 9783540719182.

Hyndman, R. J. and Athanasopoulos, G. Forecasting: Principles and practice. *www. otexts. org/fpp.*, 987507109, 2017.

Hyndman, R. J. and Khandakar, Y. Automatic time series forecasting: the forecast package for r. *Journal of Statistical Software*, 2008.

Koenker, R. *Quantile Regression*. Econometric Society Monographs. Cambridge University Press, 2005.

Krishnan, R. G., Shalit, U., and Sontag, D. Structured inference networks for nonlinear state space models. In *AAAI*, pp. 2101–2109, 2017.

Laptev, N., Yosinsk, J., Li Erran, L., and Smyl, S. Time-series extreme event forecasting with neural networks at uber. In *ICML Time Series Workshop*. 2017.

Maddix, D. C., Wang, Y., and Smola, A. Deep factors with gaussian processes for forecasting. *arXiv preprint arXiv:1812.00098*, 2018.

Makridakis, S., Spiliotis, E., and Assimakopoulos, V. The m4 competition: Results, findings, conclusion and way forward. *International Journal of Forecasting*, 34(4):802 – 808, 2018. ISSN 0169-2070. doi: https://doi.org/10.1016/j.ijforecast.2018.06.001.

Ng, A. Y. and Jordan, M. I. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In Dietterich, T. G., Becker, S., and Ghahramani, Z. (eds.), *Advances in Neural Information Processing Systems 14*, pp. 841–848. MIT Press, 2002.

Paszke, A. et al. Automatic differentiation in pytorch. In *NIPS-W*, 2017.

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

Rezende, D. J. and Mohamed, S. Variational inference with normalizing flows. In *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pp. 1530–1538. JMLR.org, 2015.

Scott, S. L. and Varian, H. R. Predicting the present with bayesian structural time series. *IJMNO*, 5:4–23, 2014.

Seeger, M. W., Salinas, D., and Flunkert, V. Bayesian intermittent demand forecasting for large inventories. In *Advances in Neural Information Processing Systems*, pp. 4646–4654, 2016.

Shipmon, D. T., Gurevitch, J. M., Piselli, P. M., and Edwards, S. Time Series Anomaly Detection. *arXiv:1708.03665 [stat.ML]*, pp. 9, 2017.

Sutskever, I., Vinyals, O., and Le, Q. V. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, pp. 3104–3112, 2014.

Taylor, S. J. and Letham, B. Forecasting at scale. *PeerJ Preprints*, 5:e3190v2, September 2017. ISSN 2167-9843. doi: 10.7287/peerj.preprints.3190v2.

van den Oord, A. et al. Wavenet: A generative model for raw audio. *CoRR*, abs/1609.03499, 2016.

Vaswani, A. et al. Attention is all you need. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 30*, pp. 5998–6008. Curran Associates, Inc., 2017.

Wen, R. W., Torkkola, K., and Narayanaswamy, B. A multi-horizon quantile recurrent forecaster. In *NIPS Time Series Workshop*. 2017.

Xu, Q., Liu, X., Jiang, C., and Yu, K. Quantile autoregression neural network model with applications to evaluating value at risk. *Applied Soft Computing*, 49:1–12, 2016.