Probabilistic Time Series Forecasting with Implicit Quantile Networks

Adèle Gouttes [* 1] **Kashif Rasul** [* 1] **Mateusz Koren** [1] **Johannes Stephan** [1] **Tofigh Naghibi** [1]

Abstract

Here, we propose a general method for probabilistic time series forecasting. We combine an autoregressive recurrent neural network to model temporal dynamics with Implicit Quantile Networks to learn a large class of distributions over a time-series target. When compared to other probabilistic neural forecasting models on real- and simulated data, our approach is favorable in terms of point-wise prediction accuracy as well as on estimating the underlying temporal distribution.

1. Introduction

Despite being versatile and omnipresent, traditional time series forecasting methods such as those in (Hyndman & Athanasopoulos, 2018), typically provide univariate point forecasts. Training in such frameworks requires to learn one model per individual time series, which might not scale for large data. Global deep learning-based time series models are typically recurrent neural networks (RNN), like LSTM (Hochreiter & Schmidhuber, 1997). These methods have become popular due to their end-to-end training, the ease of incorporating exogenous covariates, and their automatic feature extraction abilities, which are the hallmarks of deep learning.

It is often desirable for the outputs to be probability distributions, in which case forecasts typically provide uncertainty bounds. In the deep learning setting the two main approaches to estimate uncertainty have been to either model the data distribution explicitly or to use Bayesian Neural Networks as in (Zhu & Laptev, 2017). The former methods rely on some parametric density function, such as that of a Gaussian distribution, which is often based on computational convenience rather than on the underlying distribution of the data.

In this paper, we propose `IQN-RNN`, a deep-learning-based

univariate time series method that learns an implicit distribution over outputs. Importantly, our approach does not make any a-priori assumptions on the underlying distribution of our data. The probabilistic output of our model is generated via Implicit Quantile Networks (Dabney et al., 2018) (IQN) and is trained by minimizing the integrand of the Continuous Ranked Probability Score (CRPS) (Matheson & Winkler, 1976).

The major contributions of this paper are:

1. model the data distribution using IQNs which allows the use of a broad class of datasets;

2. model the time series via an autoregressive RNN where the emission distribution is given by an IQN;

3. demonstrate competitive results on real-world datasets in particular when compared to RNN-based probabilistic univariate time series models.

2. Background

2.1. Quantile Regression

The quantile function corresponding to a cumulative distribution function (c.d.f.) $F \colon \mathbb{R} \to [0, 1]$ is defined as:

$$Q(\tau) = \inf\{x \in \mathbb{R} : \tau \leq F(x)\}.$$

For continuous and strictly monotonic c.d.f. one can simply write $Q = F^{-1}$.

In order to find the quantile for a given τ one can use quantile regression (Koenker, 2005), which minimizes the quantile loss:

$$L_\tau(y, \hat{y}) = \tau(y - \hat{y})_+ + (1 - \tau)(\hat{y} - y)_+, \quad (1)$$

where $()_+$ is non-zero iff the value in the parentheses is positive.

2.2. CRPS

Continuous Ranked Probability Score (Matheson & Winkler, 1976; Gneiting & Raftery, 2007) is a *proper* scoring rule, described by a c.d.f. F given the observation y:

$$\mathrm{CRPS}(F, y) = \int_{-\infty}^{\infty} (F(x) - \mathbb{1}\{y \leq x\})^2 \, dx,$$

---
*Equal contribution [1]Zalando SE, Valeska-Gert-Straße 5, 10243 Berlin, Germany. Correspondence to: Adele Gouttes <adele.gouttes@zalando.de>.

where $\mathbb{1}$ is the indicator function. The formula can be rewritten (Laio & Tamea, 2007) as an integral over the quantile loss:

$$\text{CRPS}(F, y) = 2 \int_0^1 L_\tau(y, Q(\tau)) \, d\tau,$$

where Q is the quantile function corresponding to F.

3. Related work

Over the last years, deep learning models have shown impressive results over classical methods in many fields (Schmidhuber, 2015) like computer vision, speech recognition, natural language processing (NLP), and also time series forecasting, which is related to sequence modeling in NLP (Sutskever et al., 2014). Modern univariate forecast methods like `N-BEATS` (Oreshkin et al., 2020) share parameters, are interpretive and fast to train on many target domains.

To estimate the underlying temporal distribution we can learn the parameters of some target distribution as in the `DeepAR` method (Salinas et al., 2019b) or use mixture density models (McLachlan & Basford, 1988) operating on neural network outputs, called mixture density networks (MDN) (Bishop, 1971). One prominent example is `MDRNN` (Graves, 2013) that uses a mixture-density RNN to model handwriting. These approaches assume some parametric distribution, based on the data being modeled, for example a Negative Binomial distribution for count data. Models are trained by maximizing the likelihood of these distributions with respect to their predicted parameters and ground truth data.

Our approach is closely related to the `SQF-RNN` (Gasthaus et al., 2019) which models the conditional quantile function using isotonic splines. We utilize an IQN (Dabney et al., 2018; Yang et al., 2019) instead, as we will detail, which has been used in the context of Distributional Reinforcement Learning (Bellemare et al., 2017), as well as for Generative Modelling (Ostrovski et al., 2018).

4. Forecasting with Implicit Quantile Networks

In an univariate time series forecasting setting, we typically aim at forecasting a subseries $(y_{T-h}, y_{T-h+1}, \ldots, y_T)$ of length h from a series $y = (y_0, y_1, \ldots, y_T)$ of length $T + 1$, generated by an auto-regressive process $Y = (Y_0, Y_1, \ldots, Y_T)$.

Let $\tau_0 = \mathbb{P}[Y_0 \leq y_0]$ and $\tau_t = \mathbb{P}[Y_t \leq y_t | Y_0 \leq y_0, \ldots, Y_{t-1} \leq y_{t-1}]$ for each integer $t \in [\![1, T]\!]$. We can rewrite y as $(F_{Y_0}^{-1}(\tau_0), F_{Y_1|Y_0}^{-1}(\tau_1), \ldots, F_{Y_T|Y_0,\ldots,Y_T}^{-1}(\tau_T))$, where F is the c.d.f..

In probabilistic time series forecasting, it is typically as-

sumed that a unique function g can represent the distribution of all Y_t, given input covariates X_t and previous observations $y_0, y_1, \ldots, y_{t-1}$. When using IQNs, we additionally parameterize this function with τ_t: IQNs learn the mapping from $\tau_t \sim U([0, 1])$ to y_t. In other words, they are deterministic parametric functions trained to reparameterize samples from the uniform distribution to respective quantile values of a target distribution. Our `IQN-RNN` model should then learn $y_t = g(X_t, \tau_t, y_0, y_1, \ldots, y_{t-1})$ for $t \in [\![T - h, T]\!]$ and can be written as $q \circ [\psi_t \odot (1 + \phi)]$, where $\odot$ is the Hadamard element-wise product and:

- X_t are typically time-dependent features, known for all time steps;

- ψ_t is the state of an RNN that takes the concatenation of X_t, y_{t-1} as well as the previous state ψ_{t-1} as input;

- ϕ embeds a τ_t as described by (Dabney et al., 2018), with $n = 64$:

$$\phi(\tau_t) = \text{ReLU}(\sum_{i=0}^{n-1} \cos(\pi i \tau_t) w_i + b_i);$$

- q is an additional generator layer, which in our case is a simple two-layer feed-forward neural network, with a domain relevant activation function.

We perform the Hadamard operation on $(1 + \phi)$, which is one of the forms considered by (Dabney et al., 2018).

At training time, quantiles are sampled for each observation at each time step and passed to both network and quantile loss (1) (see Figure 1).
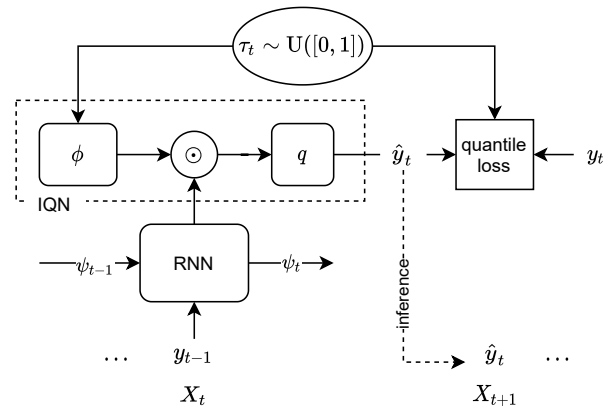

Figure 1. `IQN-RNN` schematic at time t where during training we minimize the quantile loss with respect to the ground truth.

During inference, we analogously sample a new quantile for each time step of our autoregressive loop. Thus, a full

single forward pass follows an *ancestral sampling* scheme along the graph of our probabilistic network. This approach guarantees to produce valid samples from the underlying model. Sampling a larger number of trajectories this way, allows us to estimate statistics over the distribution of each observation such as mean, quantiles, and standard deviation. For instance, the mean of Y_{T-h} can be estimated using the average over the first step of all sampled trajectories. In our experiments, we choose 100 samples (in parallel via the batch dimension) when calculating metrics and empirical quantiles. This strategy also addresses potential quantile-crossing issues, since nothing in the `IQN-RNN` architecture guaranties monotonicity with respect to τ: we simply compute the quantiles from sampled values.

We note that this method would work equally well using a masked Transformer (Vaswani et al., 2017) to model the temporal dynamics or a fixed horizon non-autoregessive model like in (Oreshkin et al., 2020).

5. Experiments

We evaluate our `IQN-RNN` model on synthetic and open datasets and follow the recommendations of the M4 competition (Makridakis et al., 2020) regarding performance metrics. We report the mean scale interval score (MSIS[1]) for a 95% prediction interval, the 50-th and 90-th quantile percentile loss, and the CRPS. The point-wise performance of models is measured by the normalized root mean square error (NRMSE), the mean absolute scaled error (MASE) (Hyndman & Koehler, 2006), and the symmetric mean absolute percentage error (sMAPE) (Makridakis, 1993). For pointwise metrics, we use sampled medians with the exception of NRMSE, where we take the mean over our prediction sample.

The code for our model is available in the `PyTorchTS` (Rasul, 2021) library.

5.1. Results on synthetic data

We firstly evaluate our `IQN-RNN` on synthetic data and compare its performance with another non-parametric probabilistic forecast model: `SQF-RNN` (Gasthaus et al., 2019). In order to minimize the MSIS and CRPS, we use this model with 50 linear pieces. Both models have the same `RNN` architecture and the same hyperparameters for training. Only the probabilistic head is distinct.

In a similar fashion to (Gasthaus et al., 2019), we generate 10,000 time series of 48 points, where each time step is iid and follows a Gaussian Mixture Model: $GMM(\pi, \mu, \sigma)$ with $\pi = [.3, .4, .3]^T$, $\mu = [-3, 0, 3]^T$, $\sigma = [.4, .4, .4]^T$. The models are trained 5 times each, with a context length
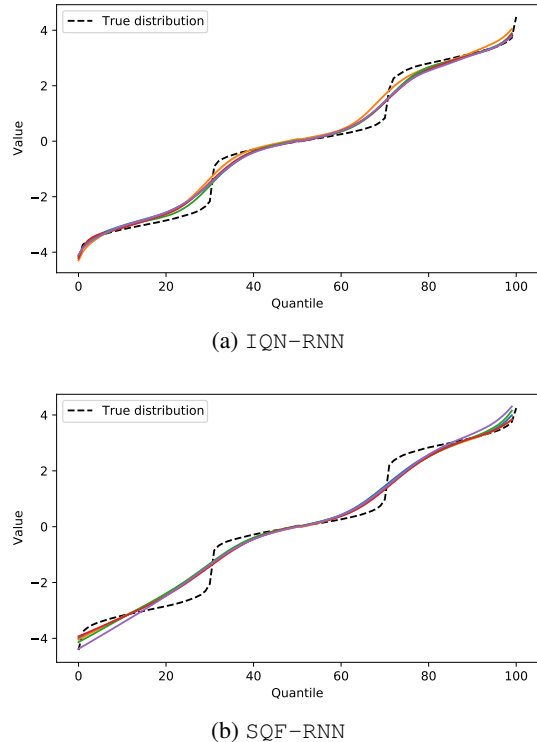
[1]https://bit.ly/3c7ffmS

(a) `IQN-RNN`

(b) `SQF-RNN`

Figure 2. Estimated quantile functions for five training on time series following a Gaussian Mixture using (a) `IQN-RNN` and (b) `SQF-RNN` model.

Table 1. Performance of `IQN-RNN` and `SQF-RNN` in fitting a three-component Gaussian Mixture Model.

METHOD	CRPS	MSIS	sMAPE	MASE
SQF-RNN-50	0.780	3.213	1.756	0.747
IQN-RNN	**0.776**	**3.027**	1.754	0.740

of 15, a prediction window of 2 for 20 epochs.

We show the estimated quantile functions in Figure 2 and report the average metrics in Table 1. While both methods have a similar point-wise performance, `IQN-RNN` is better at estimating the entire probability distribution.

5.2. Results on empirical data

We next evaluate our model on open source datasets for univariate time series: `Electricity`[2], `Traffic`[3], and `Wikipedia`[4], preprocessed exactly as in (Salinas et al., 2019a), with their properties listed in Table 3. Our model is

[2]https://archive.ics.uci.edu/ml/datasets/ElectricityLoadDiagrams20112014
[3]https://archive.ics.uci.edu/ml/datasets/PEMS-SF
[4]https://github.com/mbohlkeschneider/gluon-ts/tree/mv_release/datasets

Table 2. Comparison against different methods: `SQF-RNN` with 50 nodes, `DeepAR` with Student-T (`-t`) or Negative Binomial (`-nb`) output, `ETS` and `IQN-RNN` on the 3 datasets.

DATA SET	METHOD	CRPS	QL50	QL90	MSIS	NRMSE	sMAPE	MASE
ELECTRICITY	SQF-RNN-50	0.078	0.097	0.044	8.66	0.632	0.144	1.051
	DeepAR-t	0.062	0.078	0.046	**6.79**	0.687	0.117	**0.849**
	ETS	0.076	0.100	0.050	9.992	0.838	0.156	1.247
	IQN-RNN	**0.060**	**0.074**	**0.040**	8.74	**0.543**	**0.138**	0.897
TRAFFIC	SQF-RNN-50	0.153	0.186	**0.117**	8.40	**0.401**	0.243	0.76
	DeepAR-t	0.172	0.216	**0.117**	8.027	0.472	0.244	0.89
	ETS	0.427	0.488	0.325	20.856	0.872	0.594	1.881
	IQN-RNN	**0.139**	**0.168**	**0.117**	**7.11**	0.433	**0.171**	**0.656**
WIKIPEDIA	SQF-RNN-50	0.283	0.328	0.321	23.71	2.24	0.261	1.44
	DeepAR-nb	0.452	0.572	0.526	46.79	2.25	0.751	2.94
	DeepAR-t	0.235	0.27	0.267	23.77	2.15	0.21	1.23
	ETS	0.788	0.440	0.836	61.685	3.261	0.301	2.214
	IQN-RNN	**0.207**	**0.241**	**0.238**	**19.61**	**2.074**	**0.179**	**1.141**

Table 3. Number of time series, domain, frequency, total training time steps and prediction length properties of the training datasets used in the experiments.

DATA SET	NUM.	DOM.	FREQ.	TIME STEPS	PRED. STEPS
ELEC.	321	$\mathbb{R}^+$	HOUR	21,044	24
TRAFFIC	862	$(0,1)$	HOUR	14,036	24
WIKI.	9,535	$\mathbb{N}$	DAY	762	30

Table 4. Common hyperparmeters for `SQF-RNN`, `DeepAR` and `IQN-RNN` models.

HYPERPARAMETER	VALUE
RNN_CELL_TYPE	GRU(CHUNG ET AL., 2014)
RNN_HIDDEN_SIZE	64
RNN_NUM_LAYERS	3
RNN_DROPOUT_RATE	0.2
CONTEXT_LENGTH	2 * PRED_STEPS
EPOCHS	10
LEARNING_RATE	0.001
BATCH_SIZE	256
BATCHES_PER_EPOCH	120
NUM_SAMPLES	100
OPTIM	ADAM (KINGMA & BA, 2015)

trained on the training split of each dataset. For testing, we use a rolling-window prediction starting from the last point seen in the training dataset and compare it to the test set.

For comparison, we again use `SQF-RNN` (Gasthaus et al., 2019) with 50 linear pieces. We also evaluate `DeepAR` (Salinas et al., 2019b) with a Student-T or a Negative Binomial distribution depending on the domain of the dataset. Since `IQN-RNN`, `SQF-RNN` and `DeepAR` share the same RNN architecture we compare these models using the same *untuned*, but recommended, hyperparameters (see Table 4)

for training: only the probabilistic heads differ. Thus, deviations from performance reported in the original publications are solely due to the number of epochs used for training. Alternative models are trained on the same instances, consume a similar amount of memory, and need similar training time. We also use `ETS` (Hyndman & Khandakar, 2008) as a comparison, which is an exponential smoothing method using weighted averages of past observations with exponentially decaying weights as the observations get older together with Gaussian additive errors (E) modeling trend (T) and seasonality (S) effects separately.

In Table 2 we report probabilistic and point-wise performance metrics of all models. We found that using `IQN-RNN` often leads to the best performance on both probabilistic and point-wise metrics while being fully non-parametric and without significantly increasing the parameters of the RNN model. We also note, that the resulting performance on point-forecasting metrics does not result in higher errors for our probabilistic measures (unlike e.g. `DeepAR`). We did not incorporate per time series embeddings as covariates in any of our experiments.

6. Conclusion

In this work, we proposed a general method of probabilistic time series forecasting by using IQNs to learn the quantile function of the next time point. We demonstrated the performance of our approach against competitive probabilistic methods on real-world datasets.

Our framework can be easily extended to multivariate time series, under the rather restrictive hypothesis that we observe the same quantile for individual univariate series. This is equivalent to assuming *comonotonicity* of the processes for each time step. Relaxing this assumption is left to future research.

Software

We wish to acknowledge and thank the authors and contributors of the following open source libraries that were used in this work: GluonTS (Alexandrov et al., 2020), NumPy (Harris et al., 2020), Pandas (Pandas development team, 2020), Matplotlib (Hunter, 2007) and PyTorch (Paszke et al., 2019).

Acknowledgements

References

Alexandrov, A., Benidis, K., Bohlke-Schneider, M., Flunkert, V., Gasthaus, J., Januschowski, T., Maddix, D. C., Rangapuram, S., Salinas, D., Schulz, J., Stella, L., Türkmen, A. C., and Wang, Y. GluonTS: Probabilistic and Neural Time Series Modeling in Python. *Journal of Machine Learning Research*, 21(116):1–6, 2020. URL http://jmlr.org/papers/v21/19-820.html.

Bellemare, M. G., Dabney, W., and Munos, R. A distributional perspective on reinforcement learning. volume 70 of *Proceedings of Machine Learning Research*, pp. 449–458, International Convention Centre, Sydney, Australia, 2017. PMLR. URL http://proceedings.mlr.press/v70/bellemare17a.html.

Bishop, C. M. *Pattern Recognition and Machine Learning*. Springer US, 1971. ISBN 9781461575689, 9781461575665. doi: 10.1007/978-1-4615-7566-5. URL https://doi.org/10.1007/978-1-4615-7566-5.

Chung, J., Gulcehre, C., Cho, K., and Bengio, Y. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *NIPS 2014 Workshop on Deep Learning, December 2014*, 2014.

Dabney, W., Ostrovski, G., Silver, D., and Munos, R. Implicit Quantile Networks for Distributional Reinforcement Learning. volume 80 of *Proceedings of Machine Learning Research*, pp. 1096–1105, Stockholmsmässan, Stockholm Sweden, 2018. PMLR. URL http://proceedings.mlr.press/v80/dabney18a.html.

Gasthaus, J., Benidis, K., Wang, Y., Rangapuram, S. S., Salinas, D., Flunkert, V., and Januschowski,

T. Probabilistic Forecasting with Spline Quantile Function RNNs. volume 89 of *Proceedings of Machine Learning Research*, pp. 1901–1910. PMLR, 2019. URL http://proceedings.mlr.press/v89/gasthaus19a.html.

Gneiting, T. and Raftery, A. E. Strictly proper scoring rules, prediction, and estimation. *Journal of the American Statistical Association*, 102(477):359–378, 2007. doi: 10.1198/016214506000001437. URL https://doi.org/10.1198/016214506000001437.

Graves, A. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013.

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del R'ıo, J. F., Wiebe, M., Peterson, P., G'erard-Marchant, P., Sheppard, K., Reddy, T., Weckesser, W., Abbasi, H., Gohlke, C., and Oliphant, T. E. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020. doi: 10.1038/s41586-020-2649-2. URL https://doi.org/10.1038/s41586-020-2649-2.

Hochreiter, S. and Schmidhuber, J. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November 1997. ISSN 0899-7667, 1530-888X. doi: 10.1162/neco.1997.9.8.1735. URL https://doi.org/10.1162/neco.1997.9.8.1735.

Hunter, J. D. Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007. doi: 10.1109/MCSE.2007.55.

Hyndman, R. and Athanasopoulos, G. *Forecasting: Principles and practice*. OTexts, 2018. ISBN 9780987507112.

Hyndman, R. J. and Khandakar, Y. Automatic time series forecasting: Theforecastpackage forR. *J. Stat. Soft.*, 27(3):1–22, 2008. ISSN 1548-7660. doi: 10.18637/jss.v027.i03. URL https://doi.org/10.18637/jss.v027.i03.

Hyndman, R. J. and Koehler, A. B. Another look at measures of forecast accuracy. *International Journal of Forecasting*, 22(4):679–688, 2006. ISSN 0169-2070. doi: https://doi.org/10.1016/j.ijforecast.2006.03.001. URL https://www.sciencedirect.com/science/article/pii/S0169207006000239.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

Koenker, R. *Quantile Regression.* Econometric Society Monographs. Cambridge University Press, 2005. ISBN 9780511754098. doi: 10.1017/cbo9780511754098. URL https://doi.org/10.1017/cbo9780511754098.

Laio, F. and Tamea, S. Verification tools for probabilistic forecasts of continuous hydrological variables. *Hydrology and Earth System Sciences*, 11(4):1267–1277, 2007. doi: 10.5194/hess-11-1267-2007. URL https://hess.copernicus.org/articles/11/1267/2007/.

Makridakis, S. Accuracy measures: theoretical and practical concerns. *International Journal of Forecasting*, 9(4):527–529, 1993. ISSN 0169-2070. doi: https://doi.org/10.1016/0169-2070(93)90079-3. URL https://www.sciencedirect.com/science/article/pii/0169207093900793.

Makridakis, S., Spiliotis, E., and Assimakopoulos, V. The M4 Competition: 100,000 time series and 61 forecasting methods. *International Journal of Forecasting*, 36(1):54–74, 2020. ISSN 0169-2070. doi: https://doi.org/10.1016/j.ijforecast.2019.04.014. URL https://www.sciencedirect.com/science/article/pii/S0169207019301128. M4 Competition.

Matheson, J. E. and Winkler, R. L. Scoring rules for continuous probability distributions. *Manage. Sci.*, 22(10):1087–1096, June 1976. ISSN 0025-1909, 1526-5501. doi: 10.1287/mnsc.22.10.1087. URL https://doi.org/10.1287/mnsc.22.10.1087.

McLachlan, G. and Basford, K. *Mixture models: Inference and applications to clustering.* Marcel Dekker, New York, 1988.

Oreshkin, B. N., Carpov, D., Chapados, N., and Bengio, Y. N-BEATS: Neural basis expansion analysis for interpretable time series forecasting. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=r1ecqn4YwB.

Ostrovski, G., Dabney, W., and Munos, R. Autoregressive quantile networks for generative modeling. In Dy, J. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 3936–3945. PMLR, 2018. URL http://proceedings.mlr.press/v80/ostrovski18a.html.

Pandas development team, T. pandas-dev/pandas: Pandas, February 2020. URL https://doi.org/10.5281/zenodo.3509134.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. PyTorch: An imperative style, high-performance deep learning library. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 32*, pp. 8026–8037. Curran Associates, Inc., 2019.

Rasul, K. PytorchTS, 2021. URL https://github.com/zalandoresearch/pytorch-ts.

Salinas, D., Bohlke-Schneider, M., Callot, L., Medico, R., and Gasthaus, J. High-dimensional multivariate forecasting with low-rank Gaussian copula processes. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 32*, pp. 6824–6834. Curran Associates, Inc., 2019a.

Salinas, D., Flunkert, V., Gasthaus, J., and Januschowski, T. DeepAR: Probabilistic forecasting with autoregressive recurrent networks. *Int. J. Forecasting*, 2019b. ISSN 0169-2070. URL http://www.sciencedirect.com/science/article/pii/S0169207019301888.

Schmidhuber, J. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117, January 2015. ISSN 0893-6080. doi: 10.1016/j.neunet.2014.09.003. URL https://doi.org/10.1016/j.neunet.2014.09.003.

Sutskever, I., Vinyals, O., and Le, Q. V. Sequence to sequence learning with neural networks. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N., and Weinberger, K. (eds.), *Advances in Neural Information Processing Systems 27*, pp. 3104–3112. Curran Associates, Inc., 2014.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. Attention is all you need. In Guyon, I., Luxburg, U., 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. URL http://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf.

Yang, D., Zhao, L., Lin, Z., Qin, T., Bian, J., and Liu, T.-Y. Fully Parameterized Quantile Function for Distributional Reinforcement Learning. In *Advances in Neural Information Processing Systems*, pp. 6190–6199, 2019.

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, November 2017. ISBN 9781538638002. doi: 10.1109/icdmw.2017.19. URL https://doi.org/ 10.1109/icdmw.2017.19.