The Effectiveness of Discretization in Forecasting: An Empirical Study on Neural Time Series Models

Stephan Rabanser*
University of Toronto
stephan.rabanser@mail.utoronto.ca

Tim Januschowski
AWS AI Labs
tjnsch@amazon.com

Valentin Flunkert
AWS AI Labs
flunkert@amazon.com

David Salinas
NAVER LABS Europe
david.salinas@naverlabs.com

Jan Gasthaus
AWS AI Labs
gasthaus@amazon.com

ABSTRACT

Time series modeling techniques based on deep learning have seen many advancements in recent years, especially in data-abundant settings, and with the central aim of learning global models that can extract patterns across multiple time series. While the crucial importance of appropriate data pre-processing and scaling has often been noted in prior work, most published work focusses on improved model architectures. In this paper we empirically investigate the effect of data input and output transformations on the predictive performance of several neural forecasting architectures. In particular, we investigate the impact of several forms of data *binning*, i.e. converting real-valued time series into categorical ones, on prediction performance, when combined with feed-forward, recurrent, and convolution-based models. We find that binning can significantly improve performance when combined with certain model architectures (compared to scaling techniques), but that the particular type of binning chosen is of lesser importance.

CCS CONCEPTS

• **Mathematics of computing → Time series analysis**; • **Applied computing → Forecasting**; • **Computing methodologies → Neural networks**.

ACM Reference Format:
Stephan Rabanser, Tim Januschowski, Valentin Flunkert, David Salinas, and Jan Gasthaus. 2020. The Effectiveness of Discretization in Forecasting: An Empirical Study on Neural Time Series Models. In *MileTS '20: 6th KDD Workshop on Mining and Learning from Time Series, August 24th, 2020, San Diego, California, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Neural forecasting methods have seen many advancements in recent years, especially in data-abundant settings and with the central aim of learning a single *global* model over a panel of time series to then extract patterns across multiple time series. Global models produce univariate forecasts, i.e. they forecast each time series member of the panel of time series independently, but parameters of the model are estimated over the entire panels.

Many deep learning architectures that have seen success in other domains (e.g. computer vision or natural language processing) have been adapted to and evaluated in the global forecasting setting, ranging from simple feed forward models, convolutional neural networks (CNNs), in particular using 1-dimensional dilated causal convolutions [2, 3, 16, 25], recurrent neural networks (RNNs) [15, 19, 21], and attention-based models [11, 12, 23].

While some prior work has only considered the point forecasting setting, we focus on models that can produce probabilistic forecasts, i.e. forecasts that quantify the uncertainty over future events by estimating a probability distribution over future trajectories. Such probabilistic forecasts can be used for decision making under uncertainty, which is typically the ultimate goal in practical applications. To that end, the various aforementioned deep learning architectures have been combined with techniques for modeling probabilistic outputs. These techniques range from parametric distributions and parametric mixtures [15, 19], over quantile regression-based techniques like quantile grids [25], to parametric quantile functions models [6], semi-parametric probability integral transform / copula based techniques [18, 24], and approaches based on discretization/bucketing [16].

Recent developments in neural time series forecasting have mostly focused on improving model architectures [10–12, 17, 20], and developing strategies for modeling the probabilistic outputs in these models [6, 18, 24] (see Faloutsos et al. [5] for a recent overview). The work on global neural models for time series forecasting [19] has often hinted at (but not explored in detail) the importance of careful data pre-processing for learning across time series. To our knowledge, a thorough and systematic empirical evaluation of the impact on predictive performance and training stability that input and output representations have relative to the core forecasting model has not been performed to date. The study presented in this paper shines some light on this question by performing an empirical comparison of multiple different input and output transformation techniques—with a particular focus on discretizing transformations—when combined with commonly-used neural forecasting architectures. It complements empirical studies evaluating the impact of other architectural choices, e.g.

*Work done at AWS AI Labs.

MileTS '20, August 24th, 2020, San Diego, California, USA

ACM ISBN 978-1-4503-9999-9/18/06…$15.00
https://doi.org/10.1145/1122445.1122456

the extensive study of RNN models for forecasting conducted by Hewamalage et al. [8].

The main finding and core contribution of our empirical study is the effectiveness of discretization of inputs and outputs as a general technique for neural forecasting models. Our experimental results show that such binning techniques can improve the accuracy of forecasting models across multiple architectures.

2 PRELIMINARIES

Our study considers the following setting: We are given a set $Z = \{z_{i,1:T_i}\}_{i=1}^{N}$ of N univariate time series. Each time series $z_{i,1:T_i} = (z_{i,1}, z_{i,2}, \ldots, z_{i,T_i})$ is composed of T_i consecutive values $z_{i,t} \in \mathbb{R}$ which are assumed to be equally spaced. In addition to the *target* time series $z_{i,t}$ (i.e. the ones we are trying to predict the future of), the methods we consider can optionally make use of a set of associated covariates $X = \{x_{i,1:T_i+\tau}\}$ with $x_{i,t} \in \mathbb{R}^D$, which are required to be available until time point $T_i + \tau$ with τ being the prediction horizon of the forecast.[1]

Our goal is to model the joint conditional probability distribution over $z_{i,T_i+1:T_i+\tau}$ for each time series i, given its past values $z_{i,1:T_i}$ and the observed additional covariates $x_{i,1:T_i+\tau}$. *Global* neural forecasting models achieve this by parametrizing this conditional distribution using a neural network $\mathcal{M}_\Theta$, whose parameters Θ are learned *jointly* from the entire data set (Z, X). In particular, for each time series i we have,

$$p(z_{i,T_i+1:T_i+\tau} \mid z_{i,1:T_i}, x_{i,1:T_i+\tau}) = \mathcal{M}_\Theta(z_{i,1:T_i}, x_{i,1:T_i+\tau}), \quad (1)$$

and the parameters Θ are learned by optimizing some scoring rule $\mathcal{L}$ (often negative log-likelihood) measuring the compatibility of the model with the observed data over the training data set, i.e. $\Theta^\star = \text{argmin}_\Theta \sum_i \mathcal{L}(\mathcal{M}_\Theta, (z_{i,1:T_i}, x_{i,1:T_i}))$.

Note that in Eq. (1) the values of the target time series z_i appear in both the conditioning set and the predicted variables. We refer to a transformation that is applied to the variables in the conditioning set as an *input transformation* $\phi(\cdot)$, and to a transformation that affects the predicted distribution as an *output transformation* $\psi(\cdot)$. The resulting transformed values are *input* and *output representations*, respectively. Also note that while using the same transformation for both input and output is commonly done (e.g. by pre-processing the data before applying the model), this is not necessary. In particular, the input transformation is not required to be invertible in general. See Figure 1 for a setup overview.

3 METHODS

Next, we introduce the main objects in our study, the input and output transformations, and the neural network models.

3.1 Transformations

The transformations of data that we apply in our empirical study range for schemes to rescale the time series in the panel to discretization and other transformations. We describe these next in detail.

[1] In this paper we exclusively focus on applying transformations to the target time series values $z_{i,t}$, considering the covariates $x_{i,t}$ given and fixed. In practice, the covariates are often synthetically constructed (e.g. date-dependent dummy variables) and require no further processing or similar normalization techniques.

3.1.1 *Scaling.*

When training global forecasting models on datasets with heterogeneous scales, accounting for the difference in scales between time series in some way is of critical importance for obtaining good predictive performance. Firstly, it is desirable for the models to learn scale-invariant patterns, especially seasonal behavior. Secondly, neural network models with saturating non-linearities are very sensitive to the scale of their inputs, leading to numerical issues and slow convergence (or convergence to undesirable optima) if the scale of their inputs is not carefully controlled.

A common approach for addressing the challenge of heterogeneous scales is to apply an affine transformation to each time series, i.e. $z'_{i,t} = (z_{i,t} - b_i)/a_i$, where the parameters for the transformation are chosen for each time series independently. Here, as a representative member of this family of transformations, we use the mean scaling (`ms`) scheme employed e.g. by DeepAR [19], which seems to be effective in many practical settings. In particular, we set $a_i = \frac{1}{T_i}\sum_{t=1}^{T_i} |z_{i,t}|$ and $b_i = 0$.

3.1.2 *Probability Integral Transform.*

The probability integral transform (PIT) is the transformation that maps a random variable X through its cumulative distribution function, i.e. $Y = F_X(X)$, resulting in a transformed variable Y with uniform distribution. An (approximate) probability integral transform (`pit`) can be used as an effective pre-processing technique to make the empirical marginal distribution of values in each time series (approximately) uniform [18]. In our comparison we consider the transformation $z'_{i,t} = \hat{F}_i(z_{i,t})$, where $\hat{F}_i$ is the empirical cumulative distribution function estimated from $z_{i,1:T_i}$.

3.1.3 *Discretizing Transformations.*

Binning is a form of data discretization (also called *quantization*) into a set of buckets with disjoint support, that is widely used in machine learning as a feature engineering technique. Formally, we define a function $b : \mathbb{R} \to \{1, 2, \ldots, B\}$ which maps a real-valued input to a discrete output with $B \in \mathbb{N}$ distinct bin values. Each of the possible output values $b \in \{1, \ldots, B\}$ is tied to a specific interval ("bucket") $S_b = [l_{b-1}, l_b)$ into which real-valued inputs can fall, with edge cases $l_0 = -\infty$ and $l_B = \infty$. The quantization transform b then maps a real-valued input to its bucket index, i.e. $b(x) = b$ iff $x \in S_b$. If the input domain happens to be a subset of $\mathbb{R}$ we can adjust these edges accordingly.

In order to also define a reconstruction function $s : \{1, 2, \ldots, B\} \to \mathbb{R}$ which transforms a discrete bucket value back to the original real-valued domain, we associate each bucket $b \in 1, \ldots, B$ with a value *reconstruction value* $c_b \in \mathbb{R}$ and set $s(b) = c_b$. Given such reconstruction values $c_1 \leq c_2 \leq \ldots \leq c_B$ and assuming squared error as the loss function, it can be shown that the reconstruction error is minimized by choosing the bin edges as $l_b = (c_b + c_{b+1})/2$.

We consider two strategies for selecting appropriate bin edges in this paper: equally-spaced binning and quantile binning. In equally-spaced (linear) binning, a part of the input $I = [x_{\min}, x_{\max}] \subset \mathbb{R}$ is divided into $B-2$ intervals with equal width, i.e. for $b \in \{1, \ldots, B-1\}$, $l_b = x_{\min} + (b-1)(x_{\max} - x_{\min})/(B-2)$. In contrast to linear binning, quantile binning makes use of the underlying cumulative distribution function (CDF) F_Z to construct a binning such that the number of data points falling into each bin is (approximately) equal. In quantile binning, we first create a list of equally spaced quantiles $q = (q_1, q_2, \ldots, q_B)$ where $q_j - q_{j+1} = \frac{1}{B}$ and

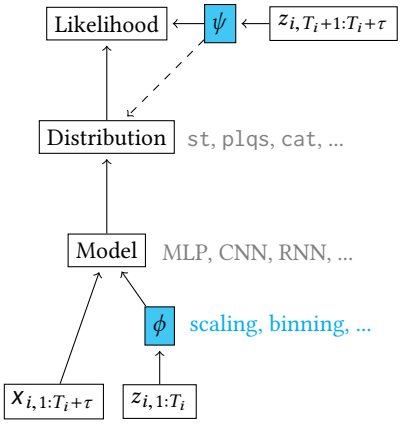
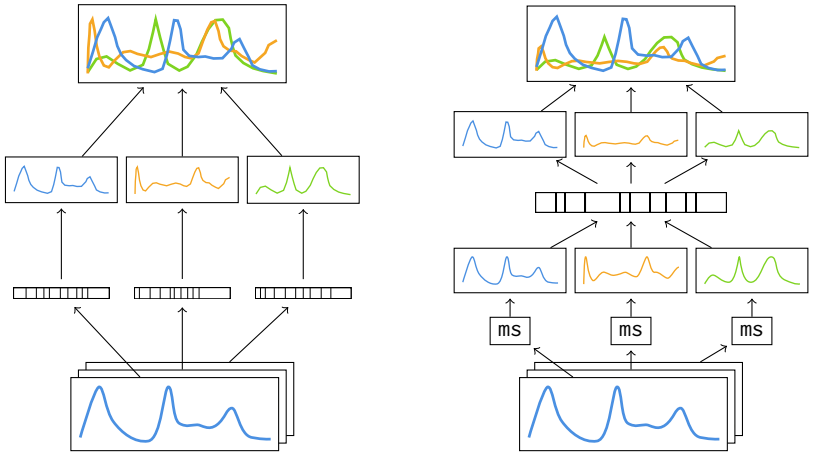


Figure 1: General setup graph with ϕ and ψ (the input and output transformation) being the core elements of this study.



(a) Local Absolute Binning **(b) Global Relative Binning**

Figure 2: Conceptual depictions of the two main binning strategies we use.

$q_j \in [0,1]\ \forall j \in \{1,2,\ldots,B\}$. Then, we can obtain the quantile-based bin representations c_b by evaluating the quantile function F_Z^{-1} for each q_b, i.e. $c_b = F_Z^{-1}(q_b)\ \forall b \in \{1,2,\ldots,B\}$, ensuring that all buckets contain approximately the same number of samples.

The resulting categorical values are fed through an embedding layer as the first layer in the model, which maps each categorical input $b \in \{1,2,\ldots,B\}$ to a vector $e_b \in \mathbb{R}^E$ that is learned together with the weights of the network using gradient descent. We consider two different binning strategies (conceptually shown in Figure 2):

Local Absolute Binning (`lab`*).* In local absolute binning, each time series is binned independently. This involves computing the the bin edges $l_i = (l_{i,1}, l_{i,2}, \ldots, l_{i,B})$ for each time series and then binning each time series $z'_{i,t} = b(z_{i,t}|l_i)$. Since each time series is binned using its own set of bin edges and each time series is mapped to the same set of bin identifiers $\{1,2,\ldots,B\}$, local absolute binning effectively acts as a scaling mechanism.

Global Relative Binning (`grb`*).* In global relative binning, all time series are first rescaled and then binned with one global binning. In particular, we use the mean scaling approach described before to scale each time series. We can then estimate one single set of bin edges $l = (l_1, l_2, \ldots, l_B)$ over the entire collection of scaled time series and bin every time series according to these bin edges $z''_{i,t} = b(z'_{i,t}|l)$ where $z'_{i,t}$ corresponds to the scaled time series. A visual example of the transformations performed by global relative binning is shown in Figure 3.

Hybrid Binning (`hyb`*$([\cdots])$).* We also consider composing multiple binnings by concatenating the resulting embeddings before passing them to the model. This allows us to combine local and global binnings, and enables us to provide the model with multi-scale inputs by combining binnings with varying bin sizes and bin edges.

3.2 Output Distributions

We compare three different approaches for modeling the output distribution $p(z_t|h_t)$: a parametric distribution, in particular a scaled Student-t distribution (`st`) applied to the raw target values $z_{i,t}$ [19]); the piecewise-linear spline quantile function approach of Gasthaus et al. [6] (`plqs`); and a categorical distribution applied to the binned values obtained through one of the described binning strategies, where the forecasts are obtained by applying the reconstruction function $s(\cdot)$ to samples from the predictive distribution.

3.3 Models

We consider three different models which we combine with the aforementioned input/output transformations:

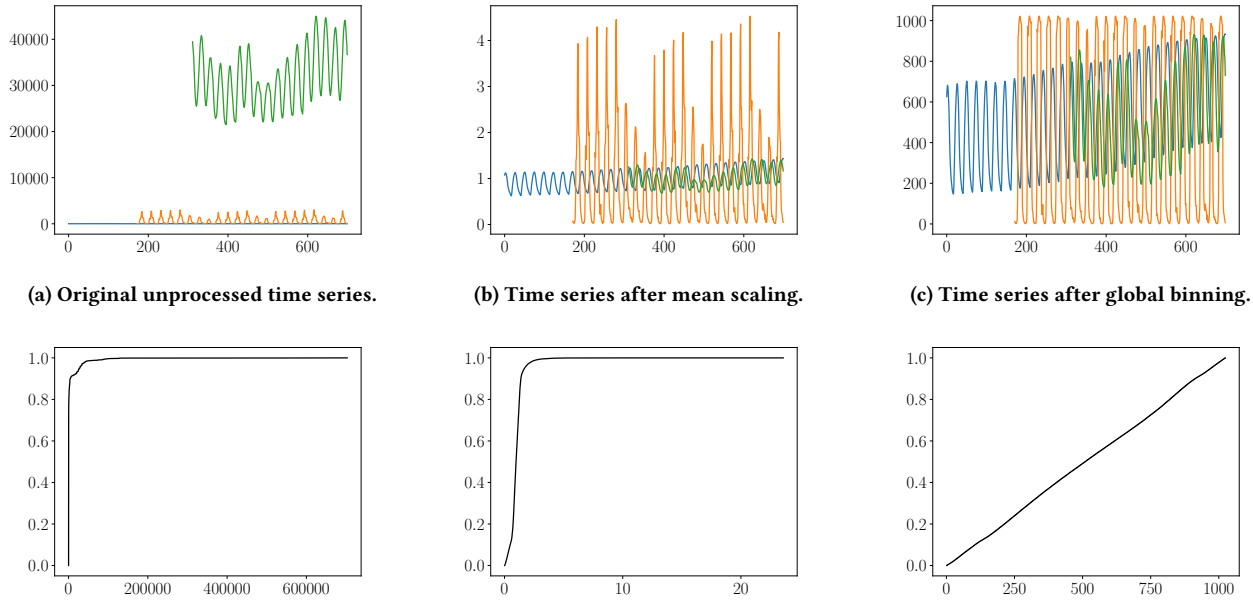
WaveNet (CNN). The WaveNet [16] architecture is an auto-regressive convolutional neural network which uses 1-dimensional dilated causal convolutions. The specific model used in the experiments alters the original architecture by using only a single stack of dilated convolutions with exponentially increasing dilation factor.

DeepAR (RNN). DeepAR [19] is an auto-regressive recurrent neural network architecture designed for time series forecasting. At its core, DeepAR is a RNN consisting of LSTM cells, which additionally receive auto-regressive inputs in the form of lagged target values.

Simple Deep Neural Network (Feed-Forward). The simple feed-forward model is a deep neural network which directly maps the past input sequence to the parameters of a multi-step output distribution without any feedback loops or memory. The model used in the experiments is a simple, plain two-layer model with 40 hidden units per layer with ReLU activations and no additional regularization.

4 EXPERIMENTS

We evaluated the representation-model combinations on data sets from the `m4` forecasting competition [13], on the `electricity` and `traffic` datasets [4], and on a sample of daily page hits of Wikipedia subpages, `wiki10k` and report mean error metrics and standard deviation over 10 random runs per configuration. Specifically, we investigated the performance effect of fixing the input representation while varying the output representation and vice versa and also examined the effects of the binning and embedding resolutions in detail. Although we do report results for `m4_hourly`,



(a) Original unprocessed time series.

(b) Time series after mean scaling.

(c) Time series after global binning.

(d) Original CDF. We can clearly see that the distribution mostly consists of series with small scales but contains a few time series with very large scales.

(e) CDF after mean scaling. While mean scaling lessens the effect that large time series have on the CDF, outliers remain.

(f) CDF after global binning. As quantile binning uses the quantile function to transform the series, the resulting CDF is uniform over the bins.

Figure 3: Global relative binning example on `m4_hourly` with 1024 bins and quantile binning. Plots (a)-(c) show how 3 randomly picked time series pass through the scaling and binning transformations; plots (d)-(f) show the CDFs over the full training set.

we note that we specifically used this dataset for tuning hyper-parameters and for generating deeper insights on representation performance. Details on hyper-parameter tuning and accuracy metrics are discussed in the appendix.

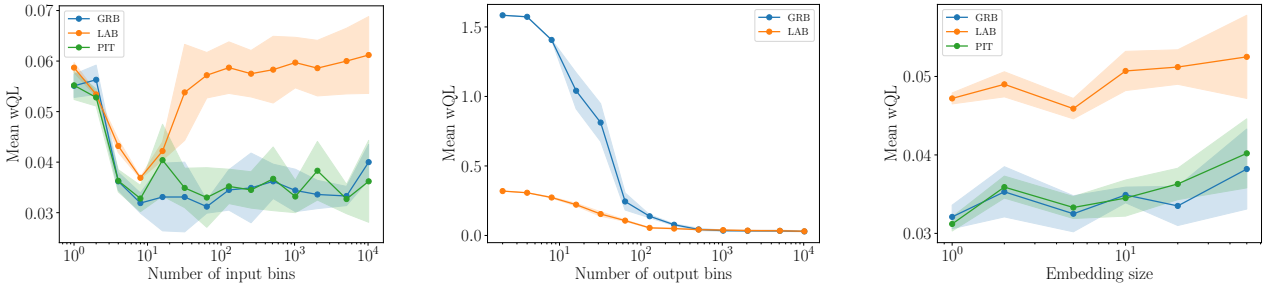
The main results are shown in Table 1 (showing the effect of varying the output representation but keeping the input fixed), Table 2 (different discretizing input transformations while keeping the output fixed), and Table 3 (performance with scaled input/output). We performed additional experiments using the WaveNet model on the `m4_hourly` data set to better understand the effect of the various transformation hyper-parameters (`grb` vs. `lab` vs. `pit`; number of bins used; embedding size). See Figure 4 for these results.

5 DISCUSSION

In the following we summarize the observations and conclusions from our experiments.

Output Scaling vs. Binning. Our main results show (cf. first column in Table 1) that in particular the `WaveNet` model substantially benefits from binned output representations when compared to real-valued, scaled outputs modeled through a parametric Student-t distribution (`ms`) or using the quantile spline output (`ms-plqs`). In fact, WaveNet combined with global relative (quantile) binning for the input and output transformation (`grb(bin1024)`) almost always (`m4_y` being the exception) outperforms all other combinations in our comparison across datasets. Interestingly, for `DeepAR` this effect is reversed (cf. Table 1, col. 2) and mean-scaled, non-binned output representations substantially outperform the discretized ones. `FeedForw` shows no clear advantage for either of the representations, but generally performs worse than either of the other models in their best configuration. These results underline our claim that input/output representations in general and output representations in particular can be equally (or even more) important for obtaining good predictive performance than choosing a particular model class, and more powerful models like WaveNet can be outperformed by simpler models like the feed forward model if the representations are not carefully chosen (e.g. on `m4_h`, `FeedForw` with mean-scaled Student-t output (`ms`) outperforms WaveNet with the same output, but is in turn outperformed by WaveNet with `grb`).

Input Scaling vs. Binning. Table 2 shows the performance of the models when the input representation is varied while the output representation is fixed (`grb`). While the multi-resolution hybrid binning (`hyb(16,128,1024)`) often performs well, there is no clear dominant strategy that outperforms the others across datasets and/or models. However, the impact of the input transformation on the performance is also less pronounced than for the output. One notable exception is local-absolute binning (`lab`) which often performs significantly worse than the other strategies in this setting. As expected, the `pit` strategy, being the continuous analogue of `grb`, performs on par with it, though `grb` appears to have a slight edge.



(a) Performance effects of varying input resolutions with respect to a fixed global relative output with 1024 quantile bins. Although LAB does first improve and then deteriorate in performance, we see that the number of input bins does play a lesser role than the specified output distribution.

(b) Performance effects of varying output resolutions with respect to a fixed global relative input with 1024 quantile bins. It is clearly visible that the chosen output representation plays a key role and that increasing the number of output bins improves performance.

(c) Performance effects of varying embedding sizes given a fixed global relative output with 1024 quantile bins and fixed 1024 input bins across different input representations. Similar to the number of input bins, the embedding size does not play a major role w.r.t model performance.

Figure 4: Insights into the performance effects incurred by altering the input/output bins, as well as the embedding size.

Binning resolution effects. Interestingly, we found that (cf. Figure 4), given a fixed global relative binning on the *output* with 1024 quantile bins, a surprisingly small number of input bins already suffices to achieve good predictive accuracy and, more so, that increasing the number of input bins does not significantly improve performance. In contrast, given a fixed global relative binning on the *input* with 1024 quantile bins, increasing the number of bins on the output leads to steady improvements in performance. While the latter effect mostly expected due to the reconstruction loss incurred with a discretized output with less bins (cf. Figure 5), the former effect is more surprising and hints at the fact the the models learn to focus on coarse-grained effects in the input, rather than focussing on fine details (that would be lost with a smaller number of bins).

Embedding size effects. Since the embedding size, which is governed by a heuristic described in Section 4, is dependent on the number of bins, we also explicitly assess the performance impact of varying the embedding size in isolation, keeping the other parameters fixed (Figure 4 c)). Similar to the results reported in Figure 4 a), we found that altering the embedding size while keeping the number of bins fixed does not significantly impact performance, and that a relatively small embedding size is sufficient.

Global vs. Local Binning. The global relative binning strategy tends to outperform local absolute binning for the output (especially e.g. WaveNet on `m4_m`, `m4_q`, and `m4_y` in Table 1). Note that (`grb`) is used for the input transformation here, so that there is a "mismatch" between the input and the output binning, which seems to be responsible for part of this effect. However, we performed additional experiment with (`lab`) input transformation (not shown) where this effect is somewhat alleviated, but does not vanish.

Hybrid vs. Single Binning. We also analyzed whether hybrid binning strategies used as an input transformation can improve performance over a single binning. Specifically, we considered two different kinds of hybrid binnings: `hyb(16,128,1024)` which includes multiple global relative binnings at different resolutions and `hyb(grb,lab)` which combines a global relative and a local absolute binning. Our results show that the multi-scale hybrid binning does indeed improve performance in many instances and is in fact the best-performing method reported for many datasets if used in conjunction with the WaveNet. However, combining both local and global information does not consistently lead to improvements over the best performing method, but rather averages results reported for global relative inputs and local absolute inputs.

Models. Overall, WaveNet does profit the most from the proposed binning strategies, while the FeedForw model does not show any meaningful gains from using binning. As already hinted at, while DeepAR can make effective use of input binnings, it demonstrates significantly worse performance when combined with a binned output representation. The reason for this is not yet clear and would benefit from further investigation.

6 CONCLUSIONS AND FUTURE WORK

We have conducted an empirical study which shines light on the question to which extent input and output transformations affect the predictive performance of different model architectures, with the overarching conclusion that carefully choosing and tuning the input and output transformations is important, as it has a large impact on the models' predictive performance, potentially larger than the performance difference between model architectures.
Directions for future work include exploring additional kinds of input and output transformations, e.g. hybrid binnings using multiple scales, and using hybrid binnings also at the output (e.g. using a multi-resolution approach similar to the "dual softmax" used in [9]). On the methodological side, extensive and principled hyperparameter tuning would allow us to make stronger conclusions about the effectiveness of particular model classes when combined with different input/output representations.

REFERENCES

[1] Alexander Alexandrov, Konstantinos Benidis, Michael Bohlke-Schneider, Valentin Flunkert, Jan Gasthaus, Tim Januschowski, Danielle C Maddix, Syama Rangapuram, David Salinas, Jasper Schulz, et al. 2019. GluonTS: Probabilistic Time Series Models in Python. *arXiv preprint arXiv:1906.05264* (2019).

[2] Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. 2018. An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling. *CoRR* abs/1803.01271 (2018). arXiv:1803.01271 http://arxiv.org/abs/1803.01271

[3] Anastasia Borovykh, Sander Bohte, and Cornelis W. Oosterlee. 2017. Conditional Time Series Forecasting with Convolutional Neural Networks. *arXiv e-prints*, Article arXiv:1703.04691 (Mar 2017), arXiv:1703.04691 pages. arXiv:stat.ML/1703.04691

[4] Dheeru Dua and Casey Graff. 2017. UCI Machine Learning Repository. http://archive.ics.uci.edu/ml

[5] Christos Faloutsos, Valentin Flunkert, Jan Gasthaus, Tim Januschowski, and Yuyang Wang. 2019. Forecasting Big Time Series: Theory and Practice. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019, Anchorage, AK, USA, August 4-8, 2019.*

[6] Jan Gasthaus, Konstantinos Benidis, Yuyang Wang, Syama Sundar Rangapuram, David Salinas, Valentin Flunkert, and Tim Januschowski. 2019. Probabilistic Forecasting with Spline Quantile Function RNNs. In *The 22nd International Conference on Artificial Intelligence and Statistics.*

[7] Tilmann Gneiting and Adrian E Raftery. 2007. Strictly proper scoring rules, prediction, and estimation. *J. Amer. Statist. Assoc.* 102, 477 (2007), 359–378.

[8] Hansika Hewamalage, Christoph Bergmeir, and Kasun Bandara. 2019. Recurrent neural networks for time series forecasting: Current status and future directions. *arXiv preprint arXiv:1909.00590* (2019).

[9] Nal Kalchbrenner, Erich Elsen, Karen Simonyan, Seb Noury, Norman Casagrande, Edward Lockhart, Florian Stimberg, Aäron van den Oord, Sander Dieleman, and Koray Kavukcuoglu. 2018. Efficient Neural Audio Synthesis. *arXiv e-prints* (2018). arXiv:1802.08435

[10] Guokun Lai, Wei-Cheng Chang, Yiming Yang, and Hanxiao Liu. 2017. Modeling Long- and Short-Term Temporal Patterns with Deep Neural Networks. *CoRR* abs/1703.07015 (2017). arXiv:1703.07015 http://arxiv.org/abs/1703.07015

[11] Shiyang Li, Xiaoyong Jin, Yao Xuan, Xiyou Zhou, Wenhu Chen, Yu-Xiang Wang, and Xifeng Yan. 2019. Enhancing the Locality and Breaking the Memory Bottleneck of Transformer on Time Series Forecasting. In *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché Buc, E. Fox, and R. Garnett (Eds.). Curran Associates, Inc., 5244–5254.

[12] Bryan Lim, Sercan Arik, Nicolas Loeff, and Tomas Pfister. 2020. Temporal Fusion Transformers for Interpretable Multi-horizon Time Series Forecasting. In *arXiv*.

[13] Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos. 2018. The M4 Competition: Results, findings, conclusion and way forward. *International Journal of Forecasting* 34, 4 (2018), 802–808.

[14] James E Matheson and Robert L Winkler. 1976. Scoring rules for continuous probability distributions. *Management science* 22, 10 (1976), 1087–1096.

[15] Srayanta Mukherjee, Devashish Shankar, Atin Ghosh, Nilam Tathawadekar, Pramod Kompalli, Sunita Sarawagi, and Krishnendu Chaudhury. 2018. ARMDN: Associative and Recurrent Mixture Density Networks for eRetail Demand Forecasting. *CoRR* abs/1803.03800 (2018). arXiv:1803.03800 http://arxiv.org/abs/1803.03800

[16] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. 2016. Wavenet: A Generative Model for Raw Audio. *arXiv preprint arXiv:1609.03499* (2016).

[17] Boris N. Oreshkin, Dmitri Carpov, Nicolas Chapados, and Yoshua Bengio. 2019. N-BEATS: Neural basis expansion analysis for interpretable time series forecasting. *arXiv e-prints* (2019). arXiv:stat.ML/1905.10437

[18] David Salinas, Michael Bohlke-Schneider, Laurent Callot, Roberto Medico, and Jan Gasthaus. 2019. High-dimensional multivariate forecasting with low-rank Gaussian Copula Processes. In *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché Buc, E. Fox, and R. Garnett (Eds.). Curran Associates, Inc., 6824–6834.

[19] David Salinas, Valentin Flunkert, Jan Gasthaus, and Tim Januschowski. 2019. DeepAR: Probabilistic Forecasting with Autoregressive Recurrent Networks. *International Journal of Forecasting* (2019).

[20] Rajat Sen, Hsiang-Fu Yu, and Inderjit S Dhillon. 2019. Think globally, act locally: A deep neural network approach to high-dimensional time series forecasting. In *Advances in Neural Information Processing Systems*. 4838–4847.

[21] Slawek Smyl. 2020. A hybrid method of exponential smoothing and recurrent neural networks for time series forecasting. *International Journal of Forecasting* 36, 1 (2020), 75–85.

[22] TensorFlow Team. 2017. Introducing TensorFlow Feature Columns. https://developers.googleblog.com/2017/11/introducing-tensorflow-feature-columns.html

[23] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. In *Advances in Neural Information Processing Systems*.

[24] Ruofeng Wen and Kari Torkkola. 2019. Deep Generative Quantile-Copula Models for Probabilistic Forecasting. *arXiv e-prints* (Jul 2019). arXiv:stat.ML/1907.10697

[25] Ruofeng Wen, Kari Torkkola, Balakrishnan Narayanaswamy, and Dhruv Madeka. 2017. A Multi-Horizon Quantile Recurrent Forecaster. *arXiv e-prints* (Nov 2017). arXiv:stat.ML/1711.11053

A HYPER-PARAMETER & ACCURACY METRIC DETAILS

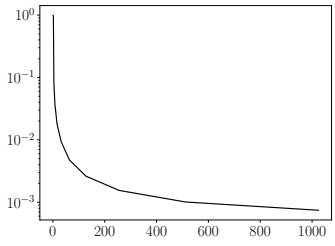
Both the `FeedForw` and `WaveNet` model were trained using the Adam optimizer with a decaying initial learning rate of 10^{-2} (decay rate 1/2) in batches of 32 samples over 150 epochs (where one epoch consists of 50 batches) on a `p2.xlarge` instance (NVIDIA K80 GPU) on Amazon Web Services. `DeepAR` follows the same setting but starts with an initial learning rate of 10^{-3} and is trained over 100 epochs. All models with the exception of `FeedForw` make use of supplementary covariates x encoding date-dependent features in the form of dummy variables in addition to time series target values z. Moreover, `DeepAR` further utilizes lagged values at varying frequencies (hourly, daily, weekly, etc.) for quicker convergence as it allows the model to pick up highly periodic patterns more easily.

By default, binnings are assumed to be quantile-based, utilize $B = 1024$ bins, and are embedded in a $E = \sqrt[4]{B}$ dimensional space [22] before being fed into the model. Initially, we also experimented with linear binning, but found the quantile binnings to be generally more reliable. When using quantile splines on the output, we default to a resolution of 20 knots.

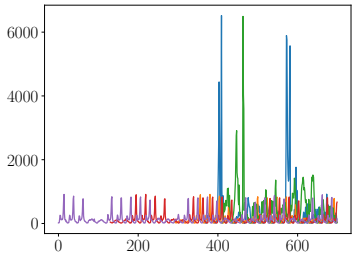
We report predictive performance in the form of two commonly-used accuracy metrics: To evaluate the quality of the predictive distributions we measure mean weighted quantile loss, which is an approximation to the continuous ranked probability score [7, 14]. In particular, we compute,

$$\text{mean wQL} = \frac{\sum_{i,t} \frac{2}{|A|} \sum_{\alpha \in A} (\alpha - I[z_{i,t} < q_{i,t}(\alpha)])(z_{i,t} - q_{i,t}(\alpha))}{\sum_{i,t} |z_{i,t}|},$$

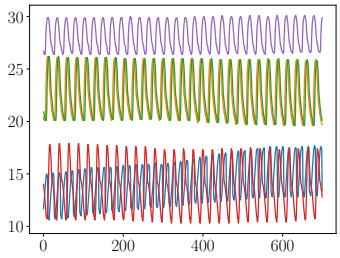
where $q_{i,t}(\alpha)$ is the α-quantile of the predictive distribution for $z_{i,t}$, and $A = \{0.1, 0.2, \ldots, 0.9\}$ is the set of quantile levels we evaluate. To evaluate the point forecasting performance, we evaluate the normalized deviation (ND), which is equivalent to wQL evaluated only at the median, i.e. with $A = \{0.5\}$.



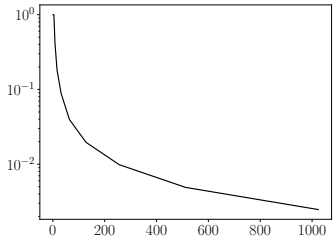
(a) Reconstruction loss over all series decreases with increasing bins.



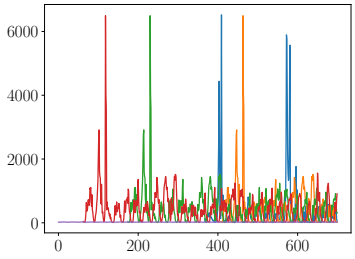
(b) Top-5 series with largest reconstruction loss (major outliers).



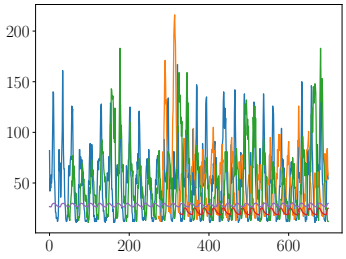
(c) Top-5 series with smallest reconstruction loss (highly regular and periodic).



(d) Reconstruction loss over all series decreases with increasing bins.



(e) Top-5 series with largest reconstruction loss (major outliers).



(f) Top-5 series with smallest reconstruction loss.

Figure 5: Time series reconstruction using global relative binning on `m4_hourly` with varying bin sizes. Plots (a)-(c) show quantile binning results, plots (d)-(f) linear binning results.

Table 1: Results with a fixed input global relative binning with 1024 quantile bins and varying output representations.

		WaveNet		DeepAR		FeedForw	
Dataset	Output	Mean wQL	ND	Mean wQL	ND	Mean wQL	ND
m4_h	ms	0.0988 (± 0.0871)	0.1135 (± 0.0940)	0.0566 (± 0.0096)	0.0676 (± 0.0102)	0.0407 (± 0.0028)	0.0519 (± 0.0015)
	ms-plqs	0.0453 (± 0.0110)	0.0557 (± 0.0106)	0.1462 (± 0.0257)	0.1618 (± 0.0289)	NaN (± NaN)	NaN (± NaN)
	grb(bin1024)	**0.0371** (± 0.0092)	0.0487 (± 0.0132)	0.0953 (± 0.0176)	0.1071 (± 0.0152)	0.0428 (± 0.0006)	0.0539 (± 0.0010)
	grb(bin1024,iqF)	0.1292 (± 0.0083)	0.1518 (± 0.0170)	0.0779 (± 0.0155)	0.0890 (± 0.0120)	0.0468 (± 0.0007)	0.0588 (± 0.0009)
	lab(bin1024)	0.0372 (± 0.0029)	**0.0463** (± 0.0028)	0.0979 (± 0.0134)	0.1123 (± 0.0177)	0.0419 (± 0.0004)	0.0528 (± 0.0004)
m4_d	ms	0.0260 (± 0.0030)	0.0321 (± 0.0040)	0.0282 (± 0.0009)	0.0338 (± 0.0012)	0.0298 (± 0.0001)	0.0304 (± 0.0001)
	ms-plqs	0.0237 (± 0.0013)	0.0289 (± 0.0019)	0.0300 (± 0.0021)	0.0363 (± 0.0030)	NaN (± NaN)	NaN (± NaN)
	grb(bin1024)	**0.0228** (± 0.0004)	**0.0280** (± 0.0005)	0.2134 (± 0.0181)	0.2235 (± 0.0214)	0.0307 (± 0.0001)	0.0353 (± 0.0000)
	grb(bin1024,iqF)	0.0530 (± 0.0009)	0.0629 (± 0.0012)	0.2103 (± 0.0184)	0.2187 (± 0.0195)	0.0283 (± 0.0000)	0.0330 (± 0.0001)
	lab(bin1024)	0.0359 (± 0.0003)	0.0412 (± 0.0002)	0.1675 (± 0.0033)	0.2132 (± 0.0017)	0.0316 (± 0.0000)	0.0367 (± 0.0001)
m4_w	ms	0.0547 (± 0.0039)	0.0686 (± 0.0049)	0.0455 (± 0.0016)	**0.0565** (± 0.0026)	0.0705 (± 0.0002)	0.0811 (± 0.0002)
	ms-plqs	0.0502 (± 0.0038)	0.0626 (± 0.0045)	0.0477 (± 0.0031)	0.0570 (± 0.0056)	NaN (± NaN)	NaN (± NaN)
	grb(bin1024)	**0.0447** (± 0.0016)	0.0569 (± 0.0021)	0.1746 (± 0.0142)	0.1947 (± 0.0133)	0.0725 (± 0.0002)	0.0851 (± 0.0002)
	grb(bin1024,iqF)	0.0641 (± 0.0024)	0.0799 (± 0.0029)	0.1724 (± 0.0150)	0.1899 (± 0.0133)	0.0707 (± 0.0002)	0.0832 (± 0.0001)
	lab(bin1024)	0.0623 (± 0.0013)	0.0770 (± 0.0018)	0.2140 (± 0.0036)	0.2446 (± 0.0028)	0.0764 (± 0.0002)	0.0885 (± 0.0002)
m4_m	ms	0.1313 (± 0.0046)	0.1576 (± 0.0049)	0.1376 (± 0.0123)	0.1639 (± 0.028)	0.1227 (± 0.0009)	0.1589 (± 0.0007)
	ms-plqs	0.1378 (± 0.0013)	0.1595 (± 0.0028)	0.1471 (± 0.0149)	0.1648 (± 0.0062)	NaN (± NaN)	NaN (± NaN)
	grb(bin1024)	**0.1177** (± 0.0031)	**0.1447** (± 0.0030)	0.1755 (± 0.0223)	0.2046 (± 0.0048)	0.1273 (± 0.0004)	0.1466 (± 0.0003)
	grb(bin1024,iqF)	0.1429 (± 0.0034)	0.1749 (± 0.0054)	0.1727 (± 0.0220)	0.2049 (± 0.0020)	0.1268 (± 0.009)	0.1454 (± 0.0007)
	lab(bin1024)	0.1507 (± 0.0003)	0.1819 (± 0.0022)	0.1931 (± 0.0254)	0.2257 (± 0.0089)	0.1231 (± 0.0006)	0.1470 (± 0.0006)
m4_q	ms	0.0936 (± 0.0032)	0.1148 (± 0.0036)	0.1067 (± 0.0039)	0.1299 (± 0.0047)	0.1097 (± 0.0007)	0.1299 (± 0.0002)
	ms-plqs	0.0987 (± 0.0028)	0.1188 (± 0.0035)	0.1267 (± 0.0117)	0.1439 (± 0.0108)	NaN (± NaN)	NaN (± NaN)
	grb(bin1024)	**0.0908** (± 0.0015)	**0.1126** (± 0.0018)	0.1673 (± 0.0060)	0.1903 (± 0.0044)	0.1146 (± 0.0011)	0.1318 (± 0.0003)
	grb(bin1024,iqF)	0.0998 (± 0.0014)	0.1237 (± 0.0017)	0.1591 (± 0.0092)	0.1819 (± 0.0073)	0.1131 (± 0.0013)	0.1291 (± 0.0003)
	lab(bin1024)	0.1195 (± 0.0019)	0.1412 (± 0.0022)	0.1647 (± 0.0103)	0.1980 (± 0.0087)	0.1197 (± 0.0004)	0.1374 (± 0.0002)
m4_y	ms	**0.1235** (± 0.0030)	**0.1476** (± 0.0032)	0.1733 (± 0.0073)	0.1940 (± 0.0072)	0.1262 (± 0.0014)	0.1497 (± 0.0014)
	ms-plqs	0.1271 (± 0.0033)	0.1486 (± 0.0039)	0.1758 (± 0.0200)	0.1973 (± 0.0206)	NaN (± NaN)	NaN (± NaN)
	grb(bin1024)	0.1538 (± 0.0112)	0.1860 (± 0.0140)	0.2765 (± 0.0076)	0.3073 (± 0.0110)	0.2131 (± 0.0008)	0.2295 (± 0.0002)
	grb(bin1024,iqF)	0.1407 (± 0.0116)	0.1712 (± 0.0122)	0.3001 (± 0.0148)	0.3264 (± 0.0124)	0.2100 (± 0.0009)	0.2254 (± 0.0002)
	lab(bin1024)	0.2024 (± 0.0110)	0.2237 (± 0.0169)	0.2596 (± 0.0081)	0.3034 (± 0.0135)	0.2300 (± 0.0003)	0.2399 (± 0.0006)
elec	ms	0.0610 (± 0.0018)	0.0774 (± 0.0028)	0.0551 (± 0.0011)	0.0678 (± 0.0016)	0.0668 (± 0.0010)	0.0826 (± 0.0017)
	ms-plqs	0.0540 (± 0.0028)	0.0681 (± 0.0036)	0.0582 (± 0.0029)	0.0707 (± 0.0030)	NaN (± NaN)	NaN (± NaN)
	grb(bin1024)	**0.0475** (± 0.0016)	**0.0588** (± 0.0024)	0.0647 (± 0.0018)	0.0821 (± 0.0020)	0.0677 (± 0.0013)	0.0841 (± 0.0020)
	grb(bin1024,iqF)	0.0512 (± 0.0015)	0.0641 (± 0.0013)	0.0712 (± 0.0064)	0.0905 (± 0.0088)	0.0661 (± 0.0007)	0.0817 (± 0.0008)
	lab(bin1024)	0.0527 (± 0.0009)	0.0647 (± 0.0007)	0.0791 (± 0.0001)	0.1013 (± 0.0002)	0.0671 (± 0.0010)	0.0793 (± 0.0012)
traff	ms	0.1437 (± 0.0044)	0.1721 (± 0.0050)	**0.1185** (± 0.0089)	**0.1401** (± 0.0014)	0.2111 (± 0.0008)	0.2527 (± 0.0008)
	ms-plqs	0.1237 (± 0.0034)	0.1510 (± 0.0040)	0.1369 (± 0.0045)	0.1680 (± 0.0016)	0.3185 (± 0.1331)	0.3849 (± 0.1646)
	grb(bin1024)	0.1209 (± 0.0016)	0.1455 (± 0.0019)	0.1883 (± 0.0012)	0.2323 (± 0.0005)	0.2218 (± 0.0010)	0.0252 (± 0.0015)
	grb(bin1024,iqF)	0.1229 (± 0.0017)	0.1495 (± 0.0022)	0.1835 (± 0.0034)	0.2245 (± 0.0012)	0.2210 (± 0.0017)	0.0341 (± 0.0050)
	lab(bin1024)	0.1261 (± 0.0006)	0.1536 (± 0.0007)	0.1632 (± 0.0051)	0.2005 (± 0.0028)	0.3820 (± 0.0832)	0.0206 (± 0.0058)
wiki	ms	0.2204 (± 0.0021)	0.2480 (± 0.0027)	0.2284 (± 0.0012)	0.2577 (± 0.0012)	0.2721 (± 0.0015)	0.3349 (± 0.0014)
	ms-plqs	0.2336 (± 0.0097)	0.2654 (± 0.0118)	0.2305 (± 0.0040)	0.2561 (± 0.0033)	NaN (± NaN)	NaN (± NaN)
	grb(bin1024)	**0.2156** (± 0.0020)	**0.2439** (± 0.0021)	0.8465 (± 0.0199)	0.9459 (± 0.0270)	0.2930 (± 0.0013)	0.3316 (± 0.0010)
	grb(bin1024,iqF)	0.2224 (± 0.0037)	0.2524 (± 0.0044)	0.7919 (± 0.0019)	0.9086 (± 0.0076)	0.2961 (± 0.0015)	0.3346 (± 0.0009)
	lab(bin1024)	0.2564 (± 0.0016)	0.2901 (± 0.0021)	0.6996 (± 0.0009)	0.8200 (± 0.0007)	0.2540 (± 0.0004)	0.2858 (± 0.0005)

Table 2: Results with a fixed output global relative binning with 1024 quantile bins and varying input representations.

		WaveNet		DeepAR		FeedForw	
Dataset	Input	Mean wQL	ND	Mean wQL	ND	Mean wQL	ND
m4_h	ms	0.0391 (± 0.0057)	0.0506 (± 0.0083)	0.0931 (± 0.0093)	0.1066 (± 0.0090)	0.0463 (± 0.0005)	0.0588 (± 0.0011)
	lab(bin1024)	0.0577 (± 0.0075)	0.0736 (± 0.0099)	0.1114 (± 0.0078)	0.1255 (± 0.0101)	0.0517 (± 0.0035)	0.0643 (± 0.0034)
	pit(bin1024)	**0.0296** (± 0.0001)	**0.0370** (± 0.0002)	0.0902 (± 0.0089)	0.1120 (± 0.0095)	0.0721 (± 0.0392)	0.0912 (± 0.0491)
	hyb(16,128,1024)	0.0375 (± 0.0009)	0.0504 (± 0.0003)	0.1020 (± 0.0057)	0.1189 (± 0.0109)	0.0435 (± 0.0004)	0.0549 (± 0.0006)
	hyb(grb,lab)	0.0369 (± 0.0061)	0.0475 (± 0.0089)	0.1057 (± 0.0088)	0.1201 (± 0.0110)	0.0421 (± 0.0017)	0.0537 (± 0.0021)
m4_d	ms	0.0315 (± 0.0057)	0.0378 (± 0.0065)	0.2128 (± 0.0182)	0.2216 (± 0.0188)	0.0305 (± 0.0000)	0.0352 (± 0.0000)
	lab(bin1024)	0.0317 (± 0.0007)	0.0369 (± 0.0008)	0.2189 (± 0.0124)	0.2244 (± 0.0130)	0.0305 (± 0.0000)	0.0352 (± 0.0001)
	pit(bin1024)	0.0286 (± 0.0053)	0.0345 (± 0.0061)	0.2204 (± 0.0144)	0.2283 (± 0.0141)	0.0305 (± 0.0002)	0.0352 (± 0.0000)
	hyb(16,128,1024)	**0.0227** (± 0.0003)	**0.0278** (± 0.0004)	0.2196 (± 0.0137)	0.2267 (± 0.0135)	0.0306 (± 0.0001)	0.0353 (± 0.0001)
	hyb(grb,lab)	0.0272 (± 0.0004)	0.0318 (± 0.0003)	0.2222 (± 0.0156)	0.2301 (± 0.0157)	0.0307 (± 0.0001)	0.0353 (± 0.0000)
m4_w	ms	0.0848 (± 0.0327)	0.1026 (± 0.0371)	0.1651 (± 0.0113)	0.1830 (± 0.0111)	0.0750 (± 0.0005)	0.0839 (± 0.0001)
	lab(bin1024)	0.1061 (± 0.0023)	0.1244 (± 0.0033)	0.1838 (± 0.0070)	0.1995 (± 0.0083)	0.0760 (± 0.0002)	0.0834 (± 0.0001)
	pit(bin1024)	0.0467 (± 0.0022)	0.0585 (± 0.0028)	0.1884 (± 0.0099)	0.2082 (± 0.0103)	0.0724 (± 0.0005)	0.0848 (± 0.0004)
	hyb(16,128,1024)	**0.0443** (± 0.0010)	**0.0561** (± 0.0014)	0.1792 (± 0.0047)	0.1980 (± 0.0042)	0.0723 (± 0.0002)	0.0854 (± 0.0002)
	hyb(grb,lab)	0.0500 (± 0.0012)	0.0627 (± 0.0015)	0.1815 (± 0.0072)	0.1975 (± 0.0074)	0.0719 (± 0.0003)	0.0849 (± 0.0002)
m4_m	ms	0.1373 (± 0.0143)	0.1655 (± 0.0137)	0.2080 (± 0.0102)	0.2412 (± 0.0098)	0.1392 (± 0.0009)	0.1470 (± 0.0000)
	lab(bin1024)	0.2055 (± 0.0021)	0.2136 (± 0.0012)	0.2395 (± 0.0154)	0.2891 (± 0.0101)	0.1396 (± 0.0005)	0.1463 (± 0.0001)
	pit(bin1024)	0.1213 (± 0.0024)	0.1481 (± 0.0029)	0.1921 (± 0.0097)	0.2287 (± 0.0084)	0.1332 (± 0.0049)	0.1462 (± 0.0009)
	hyb(16,128,1024)	**0.1187** (± 0.0037)	0.1463 (± 0.0046)	0.1944 (± 0.0098)	0.2294 (± 0.0057)	0.1267 (± 0.0023)	0.1459 (± 0.0001)
	hyb(grb,lab)	0.1206 (± 0.0010)	0.1468 (± 0.0008)	0.2018 (± 0.0105)	0.2388 (± 0.0083)	0.1264 (± 0.0014)	**0.1454** (± 0.0002)
m4_q	ms	0.1272 (± 0.0006)	0.1488 (± 0.0003)	0.1507 (± 0.0037)	0.1698 (± 0.0021)	0.1256 (± 0.0009)	0.1501 (± 0.0008)
	lab(bin1024)	0.1299 (± 0.0017)	0.1486 (± 0.0013)	0.1689 (± 0.0025)	0.1861 (± 0.0016)	0.1174 (± 0.0011)	0.1320 (± 0.0004)
	pit(bin1024)	0.1278 (± 0.0014)	0.1488 (± 0.0002)	0.1748 (± 0.0028)	0.1958 (± 0.0021)	0.1180 (± 0.0011)	0.1324 (± 0.0002)
	hyb(16,128,1024)	**0.0893** (± 0.0011)	**0.1108** (± 0.0012)	0.1743 (± 0.0052)	0.1972 (± 0.0035)	0.1152 (± 0.0019)	0.1314 (± 0.0007)
	hyb(grb,lab)	0.1137 (± 0.0032)	0.1372 (± 0.0034)	0.1722 (± 0.0029)	0.1974 (± 0.0008)	0.1152 (± 0.0021)	0.1308 (± 0.0007)
m4_y	ms	**0.1308** (± 0.0039)	**0.1562** (± 0.0034)	0.2663 (± 0.0177)	0.2907 (± 0.0123)	0.2162 (± 0.0016)	0.2326 (± 0.0008)
	lab(bin1024)	0.2812 (± 0.0144)	0.3171 (± 0.0094)	0.3062 (± 0.0140)	0.3248 (± 0.0085)	0.2143 (± 0.0004)	0.2309 (± 0.0002)
	pit(bin1024)	0.1844 (± 0.0523)	0.2202 (± 0.0621)	0.3058 (± 0.0077)	0.3280 (± 0.0086)	0.2151 (± 0.0019)	0.2324 (± 0.0022)
	hyb(16,128,1024)	0.1337 (± 0.0033)	0.1618 (± 0.0045)	0.2925 (± 0.0028)	0.3219 (± 0.0024)	0.2129 (± 0.0006)	0.2295 (± 0.0001)
	hyb(grb,lab)	0.2065 (± 0.0149)	0.2505 (± 0.0195)	0.3184 (± 0.0050)	0.3576 (± 0.0052)	0.2235 (± 0.0068)	0.2388 (± 0.0020)
elec	ms	0.0501 (± 0.0010)	0.0607 (± 0.0017)	0.0732 (± 0.0007)	0.0923 (± 0.0004)	0.0800 (± 0.0038)	0.1004 (± 0.0056)
	lab(bin1024)	0.1389 (± 0.0070)	0.1677 (± 0.0096)	0.0986 (± 0.0023)	0.1107 (± 0.0067)	0.1269 (± 0.0033)	0.1632 (± 0.0034)
	pit(bin1024)	0.0484 (± 0.0010)	0.0598 (± 0.0015)	0.4210 (± 0.1192)	0.4924 (± 0.1078)	0.0705 (± 0.0026)	0.0875 (± 0.0041)
	hyb(16,128,1024)	0.0495 (± 0.0004)	0.0612 (± 0.0007)	0.1143 (± 0.0028)	0.1339 (± 0.0055)	0.0678 (± 0.0018)	0.0843 (± 0.0026)
	hyb(grb,lab)	**0.0472** (± 0.0005)	**0.0585** (± 0.0005)	0.1528 (± 0.0072)	0.1801 (± 0.0089)	0.0687 (± 0.0009)	0.0856 (± 0.0011)
traff	ms	0.1251 (± 0.0013)	0.1507 (± 0.0013)	0.1974 (± 0.0088)	0.2423 (± 0.0339)	0.2280 (± 0.0005)	0.0287 (± 0.0021)
	lab(bin1024)	0.2571 (± 0.0174)	0.3200 (± 0.0246)	0.2535 (± 0.0246)	0.3131 (± 0.0632)	0.2456 (± 0.0010)	0.0070 (± 0.0021)
	pit(bin1024)	0.1275 (± 0.0008)	0.1539 (± 0.0010)	0.5953 (± 0.1299)	0.7266 (± 0.2040)	0.2258 (± 0.0017)	0.0254 (± 0.0037)
	hyb(16,128,1024)	**0.1242** (± 0.0008)	**0.1498** (± 0.0009)	0.1886 (± 0.0072)	0.2316 (± 0.0290)	0.2184 (± 0.0011)	0.0249 (± 0.0016)
	hyb(grb,lab)	0.1245 (± 0.0011)	0.1505 (± 0.0018)	0.1885 (± 0.0091)	0.2315 (± 0.0387)	0.2182 (± 0.0007)	0.0217 (± 0.0028)
wiki	ms	0.2183 (± 0.0028)	0.2472 (± 0.0033)	0.8156 (± 0.0176)	0.9170 (± 0.0234)	0.3027 (± 0.0007)	0.3381 (± 0.0008)
	lab(bin1024)	0.3071 (± 0.0030)	0.3478 (± 0.0026)	0.8143 (± 0.0112)	0.9115 (± 0.0130)	0.3066 (± 0.0005)	0.3408 (± 0.0004)
	pit(bin1024)	0.2177 (± 0.0043)	0.2465 (± 0.0047)	0.9238 (± 0.2095)	0.9981 (± 0.3049)	0.2935 (± 0.0014)	0.3304 (± 0.0014)
	hyb(16,128,1024)	**0.2163** (± 0.0017)	**0.2447** (± 0.0019)	0.8140 (± 0.0129)	0.9075 (± 0.0090)	0.2927 (± 0.0012)	0.3311 (± 0.0008)
	hyb(grb,lab)	0.2342 (± 0.0035)	0.2631 (± 0.0037)	0.8191 (± 0.0150)	0.9238 (± 0.0198)	0.2931 (± 0.0012)	0.3316 (± 0.0011)

Table 3: Results with mean scaling on both inputs and outputs. This is the standard scaling setting in GluonTS [1].

	WaveNet		DeepAR		FeedForw	
Dataset	Mean wQL	ND	Mean wQL	ND	Mean wQL	ND
m4_h	0.1517 (± 0.0904)	0.2008 (± 0.1334)	0.0533 (± 0.0012)	0.0645 (± 0.0009)	**0.0463** (± 0.0010)	**0.0580** (± 0.0012)
m4_d	0.0334 (± 0.0088)	0.0401 (± 0.0102)	0.0318 (± 0.0029)	0.0384 (± 0.0036)	**0.0247** (± 0.0005)	**0.0296** (± 0.0008)
m4_w	0.0574 (± 0.0036)	0.0716 (± 0.0042)	**0.0460** (± 0.0011)	**0.0565** (± 0.0012)	0.0521 (± 0.0006)	0.0614 (± 0.0006)
m4_m	0.1481 (± 0.0170)	0.1674 (± 0.0152)	0.1362 (± 0.0089)	0.1480 (± 0.0083)	**0.1159** (± 0.0011)	**0.1260** (± 0.0023)
m4_q	0.0983 (± 0.0019)	0.1196 (± 0.0017)	0.1030 (± 0.0031)	0.1176 (± 0.0027)	**0.0869** (± 0.0010)	**0.1030** (± 0.0010)
m4_y	**0.1236** (± 0.0055)	**0.1458** (± 0.0057)	0.1570 (± 0.0088)	0.1757 (± 0.0085)	0.1262 (± 0.0014)	0.1497 (± 0.0014)
elec	0.0724 (± 0.0151)	0.0923 (± 0.0194)	**0.0571** (± 0.0012)	**0.0695** (± 0.0018)	0.0649 (± 0.0011)	0.0793 (± 0.0015)
traff	0.1450 (± 0.0065)	0.1720 (± 0.0073)	**0.1222** (± 0.0077)	**0.1456** (± 0.0082)	0.2144 (± 0.0008)	0.2558 (± 0.0009)
wiki	**0.2295** (± 0.0063)	**0.2601** (± 0.0072)	0.2378 (± 0.0070)	0.2694 (± 0.0091)	0.2594 (± 0.0026)	0.3030 (± 0.0036)