C2FAR: Coarse-to-Fine Autoregressive Networks for Precise Probabilistic Forecasting

Anonymous Author(s)
Affiliation
Address
email

Abstract

We present coarse-to-fine autoregressive networks (C2FAR), a method for modeling the probability distribution of univariate random variables. In C2FAR, a coarse-to-fine discretization of the variable is generated autoregressively; progressively finer intervals of support are generated, conditioned on previously-generated coarser intervals. Unlike prior binned distributions, C2FAR can represent values with exponentially higher precision, for only a linear increase in complexity. We use C2FAR for probabilistic forecasting via a recurrent neural network, thus modeling time series autoregressively in both space and time. C2FAR is the first method to simultaneously handle discrete and continuous series of arbitrary scale and distribution shape. This flexibility enables a variety of time series use cases, including anomaly detection, interpolation, and compression. C2FAR achieves improvements over the state-of-the-art on several benchmark forecasting datasets.

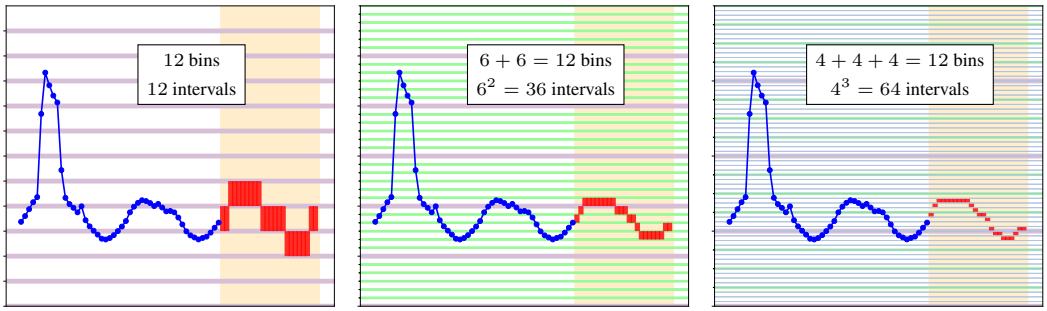
1 Introduction

Probabilistic forecasting is the task of estimating a joint distribution over future values of a time series, given a sequence of historical values. As an important problem with many applications and an abundance of real-world data, probabilistic forecasting has unsurprisingly come to be dominated by deep learning methods [7]. Such methods typically fit a global sequence model to a dataset of related time series. Future values can then be generated from one-step-ahead univariate output distributions. In this paper, we propose coarse-to-fine autoregressive networks (C2FAR), a general method for modeling univariate distributions, and we show how C2FAR can improve output distributions in probabilistic forecasting.

Such improvements are needed because time series offer challenges not present in applications like text and image modeling. First, dynamic range or *scale* of real-world time series can vary widely within a single dataset [44]. The typical solution is to normalize the data based on input historical values, but many time series have heavy tails [17], meaning even after normalization, future values must be modeled via distributions with unbounded support. Second, even within one dataset, time series may be discrete, continuous, or mixed, and each of these may be distributed in arbitrary, multi-modal ways, which vary over time. All of this makes it difficult for practitioners to encode input values and to select appropriate parametric forms for output distributions.

We introduce C2FAR, a method for modeling any univariate output (§3). C2FAR extends prior work using categorical distributions over binned representations [39, 17]. While the precision of prior methods is limited by the number of bins in the discretization, C2FAR achieves an exponential increase in precision through only a linear increase in the number of bin parameters. Like digits in the decimal number system (and unlike, say, roman numerals or simple tally marks), C2FAR represents scalar quantities hierarchically, treating numbers as multidimensional objects, with each successive dimension or *level* representing the original value at a progressively finer resolution (Fig. 1).

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.



(a) Flat binning (b) 2-level C2F binning (c) 3-level C2F binning

Figure 1: Flat vs. coarse-to-fine binnings for a *traff* time series (§5). C2F discretizations specify bin indices from coarse-to-fine, e.g. $3, 2, 2$. For the same total number of bins, the number of unique intervals in a C2F binning is much higher (and thus reconstruction error is much lower).

C2FAR proposes an autoregressive generative model for such representations. Levels are generated from coarse-to-fine. Finer and finer intervals of support are generated from categorical distributions, iteratively conditioned on coarser intervals, until ultimately the final value is generated. The conditional distributions are parameterized by a neural network, and the model is fit by maximizing log-likelihood of observed data. Compared to flat binnings, C2FAR models are more efficient for a given level of precision, and better enable learning of order and distance.

We use C2FAR to model output distributions in a deep forecasting model based on DeepAR [44] (§4). In our experiments, C2FAR-based forecasting models better recover synthetic distributions compared to recent state-of-the-art methods (§5.1). Evaluation on benchmark forecasting datasets shows that, in contrast to prior work, discretization improves accuracy (§5.2). Multi-level C2FAR models further improve over flat binnings across all datasets, including evaluation on a large new dataset of public cloud demand, which we release as a paper supplement.

2 Background and related work

2.1 Probabilistic neural forecasting

Probabilistic forecasting architectures include LSTMs [58, 44], temporal convolutions [46, 12] and transformers [32, 63, 59]. While C2FAR can be used with any such architecture, for simplicity we restrict our discussion in this paper to models that are autoregressive in time [32, 44]. While non-AR models typically output best-guess point forecasts [11, 46, 63] or specific quantiles of interest [58, 18], they can be made to output C2FAR parameters at all horizons (as was done with Gaussians in [12]).

Autoregressive networks must define step-wise input and output. To handle series of very different *scales*, input values are sometimes normalized (e.g., dividing values by their mean [32, 44, 39]); output values are scaled back after prediction. A density over outputs is achieved by mapping network states to location and spread parameters of specific distributions. Practitioners must choose a distribution "to match the statistical properties of the data" [44]. In practice, Gaussian [44, 32], Student's-t [2], and Gaussian mixture distributions [36] have been used for real-valued data, while a negative binomial distribution [44] is employed for discrete. Since output uncertainty is often not well captured by standard parametrics, recent work has investigated more flexible outputs, including spine quantile functions [21] and implicit quantile networks [24], which we compare to below (§5).

Rabanser et al. [39] use a categorical distribution over a binned representation. Categoricals are flexible, but do not encode any underlying concept of bin order or distance (which must be learned). In prior work on pixel modeling, categorical distributions sometimes work better [56], sometimes worse [42] than mixture densities. In [39], the benefits of binning were mixed, and harmed DeepAR accuracy. Ehrlich et al. [17] "splice" a continuous Pareto distribution into a categorical in order to model extreme values; we show how Pareto distributions can also be used when samples are generated autoregressively using C2FAR (§3).

2.2 Challenges in forecasting mixed data

Many time series are actually fully or partly discrete. Counts are natural numbers, while dollar values are usually specified to two decimal places. Even intrinsically continuous series are always processed or serialized to a certain level of precision. For example, one version of *elec* been quantized to whole integers, another to six decimal places (§5). As a cloud provider, we observe many usage series with significant probability of being either 0% or 100%, while otherwise being continuous.

Modeling discrete data can be challenging. The inherent limits of series are rarely known at prediction time, so unbounded distributions such as negative binomials are often used [49]. Since it is "not possible" for such distributions to output values in a normalized domain [44], we cannot normalize series of differing scales. Yet real-valued forecasts are usually acceptable for discrete data, so practitioners often simply normalize discrete data and train continuous models. Gouttes et al. [24] report better results on *wiki* with a Student's-t than a negative binomial. Rabanser et al. [39] normalize *wiki* prior to "discretizing" it again for their categorical. We have also encountered practitioners with the (erroneous) belief that normalizing makes discrete data continuous.

Unfortunately, fitting powerful continuous models to discrete data "can lead to arbitrarily high density values, by locating a narrow high density spike on each of the possible discrete values" [55]. Systems that model densities with splines [21] or flows [41], or spline flows [16], are therefore not able to maximize likelihood of discrete data directly; otherwise they would fit parameters (e.g., spline *knots*) solely to enable arbitrarily-narrow density spikes. This issue can be side-stepped by *dequantizing*, e.g., adding `Uniform[0,1]` noise as in [41]. But this assumes: (1) we know the discrete series a priori, and (2) the resulting loss in precision does not outweigh the benefits of continuous models. For C2FAR, the ability to locate high-density spikes on discrete values is a feature, not a bug. C2FAR does not generate *arbitrarily* narrow spikes, rather it is restricted by the precision of the hierarchical binning. C2FAR can train for likelihood on the actual discrete data, and it can generate realistic samples reflecting the true data distribution.

2.3 Coarse-to-fine representations in time series

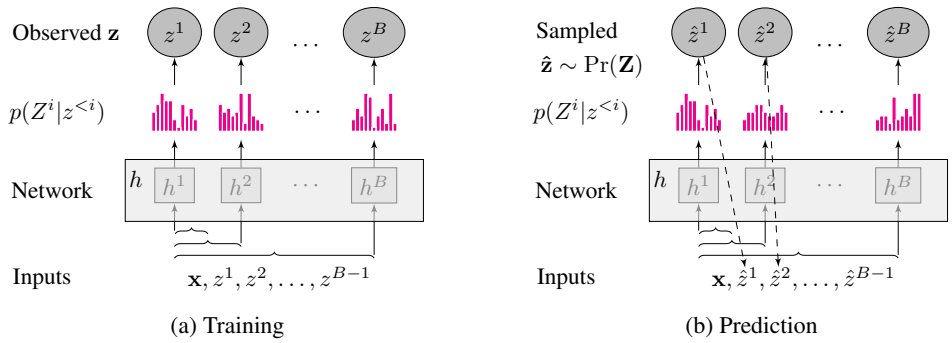
Coarse-to-fine binnings have been used previously in machine learning. A coarse-to-fine representation is induced by algorithms that recursively discretize continuous-valued attributes for decision tree classifiers [19]. On the output side, SSR-Net [60] classifies human age in a coarse-to-fine (but not autoregressive) manner. iSAX [47] uses a coarse-to-fine representation for indexing time series.

Prior autoregressive models of discrete data can be viewed as specific instances of C2FAR. WaveRNN [28] generates 16-bit audio samples by first generating the coarse (high) 8-bits from a categorical distribution, then generating the fine (low) 8-bits, conditioned on the sampled coarse bits. Vipperla et al. [57] found a different split between the coarse and fine bits in WaveRNN resulted in lower cross-entropy. These approaches can be regarded as two-level C2FAR models with classifier conditionals. Meanwhile, the multi-stage count likelihood from [45] can be viewed as a three-level C2FAR model, with binary classification at level 1 and 2, and Poisson regression to generate the final counts. C2FAR unifies and generalizes these approaches, enabling modeling of bounded or unbounded, continuous or discrete data, even within the same dataset.

3 C2FAR networks

The process of discretization typically assumes that the support of a continuous variable can be partitioned into a finite number of distinct *bins*. A value is discretized by mapping it to the index of its corresponding bin. With C2FAR, partitioning is hierarchical: support of the variable is first divided into a number of coarse bins, each of these coarse bins is further divided into a number of finer bins, and so on (recall Fig. 1 in §1). C2FAR thereby discretizes a data point, z, into a vector of B indices for each of the B coarse-to-fine levels, $\mathbf{z} = (z^1, \ldots, z^B)$. More formally, C2FAR assumes a discretization function $d : \mathbb{R} \to (\mathcal{B}_1, \ldots \mathcal{B}_B)$, where each $\mathcal{B}_i$ has one of $|\mathcal{B}_i|$ distinct values. Function $d^{\leftarrow}$ is a kind of inverse of d, mapping a set of bin indices to an *interval* in the original real-valued domain, $d^{\leftarrow} : (\mathcal{B}_1, \ldots \mathcal{B}_B) \to (\mathbb{R}, \mathbb{R})$.

Let $\mathbf{x}$ be a set of optional features that may inform the distribution of z (for example, for mixed data, $\mathbf{x}$ could flag if the bin contains an integer). After obtaining our B-dimensional discretization $\mathbf{z} = d(z)$, we follow prior work in modeling *multivariate* conditional distributions $p(\mathbf{z}|\mathbf{x})$ as autoregressive



(a) Training (b) Prediction

Figure 2: The C2FAR network. We train to minimize NLL of a set of examples; log loss on each example decomposes into the sum of NLL of each z^i under its categorical distribution. In prediction, we sample from the categoricals, and autoregressively use the sampled values as inputs.

generative models via the chain rule [20, 6, 54]:

$$p(\mathbf{z}) = \prod_{i=1}^{B} p(z_i|\mathbf{x}, \mathbf{z}_{<i}) \tag{1}$$

where $\mathbf{z}_{<i} = z^1, \ldots, z^{i-1}$. In the C2FAR case, each of the one-dimensional conditional distributions, $p(z_i|\mathbf{x}, \mathbf{z}_{<i})$, is a categorical distribution over the set of bin indices at the ith granularity, $\mathcal{B}_i$. Intuitively, the ith categorical is a distribution over the support *within* the generated $(i-1)$th bin. We parameterize these distributions using neural networks with a softmax output layer, and train the parameters to minimize negative log-likelihood (NLL) of a set of training examples (Fig. 2a). Inputs may be represented with 1-hot-encodings or embedding layers. As a generative model, we may also synthesize new data by first sampling a coarse bin ($\hat{z}^1$) given $\mathbf{x}$, then iteratively sampling finer and finer bins given the sampled coarser values (Fig. 2b).

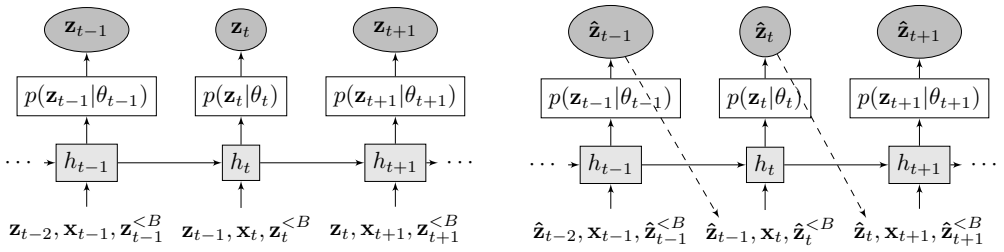
C2FAR supports a variety of neural architectures: following prior work in multivariate distribution modeling, we may use a separate network for each categorical [6] (as in Fig. 2), or one global network with masking to enforce the autoregressive property [23]. Separate networks may share parameters across the levels [31] or with classifications at the same level across time (§4). Having separate networks for each level means the modeling problem at a given level need not change if we add more levels to the hierarchy; we adopt this approach in our forecasting experiments (§4.2).

While not strictly required, we use linear (evenly-spaced) binnings at each level. We hypothesize this enables the finer networks to better generalize learned concepts of order and distance (i.e., regardless of the coarse bins that they are conditioned on). We refer to the span from the first interval to the last interval as the *extent* of the binning. The extent, the number of levels, and the number of bins at each level, are C2FAR hyperparameters. We also consider the extreme high/low intervals to be open-ended, terminating at $\pm\infty$; we discuss the implications of this below.

Consider a C2FAR discretization with B levels and a cardinality of K bins at each level; the original support of z is effectively partitioned into K^B intervals, but modeled using only KB categorical outputs in total. C2FAR models can therefore use exponentially fewer parameters compared to categorical distributions over flat binnings, i.e., categoricals with K^B total classes.

Let (a, b) be the interval defined by $(a, b) = d^{\leftarrow}(\mathbf{z})$. To complete our generative story, we must generate a value within (a, b) (rather than choosing a single "reconstruction value" for each bin as in [39]). Since our extreme intervals are open-ended, we cannot interpret our model as a piecewise uniform distribution [56], as uniforms are only defined on finite intervals. C2FAR solves this by allowing different distributions to be used depending on the interval (a, b). Essentially, we assume a final conditional in (1) (implicitly at level $B+1$) that generates from a parametric distribution. We use distributions of the form:

$$p(Z^{B+1}|a, b) \sim \begin{cases} \text{Uniform}[a, b] & -\infty < a \text{ and } b < \infty \\ \text{Pareto}[a, \alpha_1] & b = \infty \\ -\text{Pareto}[b, \alpha_2] & a = -\infty \end{cases} \tag{2}$$



(a) Training (b) Prediction

Figure 3: High-level C2FAR-RNN model. Compared to DeepAR, C2FAR-RNN uses discretized input/output vectors, $\mathbf{z}_t$, rather than scalars, and adds additional (autoregressive) inputs $\mathbf{z}_t^{<B}$.

Parameters α_1 and α_2 are generated by the h_{B+1} neural network. In this way, we are effectively defining a piecewise-uniform density with Pareto-distributed tails.[1,2]

We use this model for continuous and discrete data. If data is truly discrete and precision is useful, our tuning procedure will choose more bins/levels. If one knows a priori the true discrete support, one may instead use discrete distributions at level $B + 1$, such as negative binomials or discrete uniforms.

4 Forecasting with C2FAR

We now explain how C2FAR can be used for N-step-ahead probabilistic forecasting. We base the forecasting framework on DeepAR [44], which has served as the basis for other recent improvements in forecast distribution modeling [21, 24], and therefore facilitates experimental comparison (§5).

4.1 DeepAR-style probabilistic forecasting

Let z_t be the value of a time series at time t, and $\mathbf{x}_t$ be a vector of time-varying features or *covariates*. Probabilistic forecasting aims to model the conditional distribution of N future values of z_t (the *prediction range*) given the $T + N$ covariates, and T historical values (the *conditioning range*):

$$p(z_{T+1} \dots z_{T+N} | z_1 \dots z_T, \mathbf{x}_1 \dots \mathbf{x}_{T+N}) \stackrel{\text{def}}{=} p(z_{T+1:T+N} | z_{1:T}, \mathbf{x}_{1:T+N}) \tag{3}$$

DeepAR [44] formulates this conditional distribution as an autoregressive generative model:

$$p(z_{T+1:T+N} | z_{1:T}, \mathbf{x}_{1:T+N}) = \prod_{t=T+1}^{T+N} p(z_t | z_{1:t-1}, \mathbf{x}_{1:T+N}) = \prod_{t=T+1}^{T+N} p(z_t | \theta_t = y(h_t))$$

where h_t is the output of an LSTM [26] recurrent neural network:

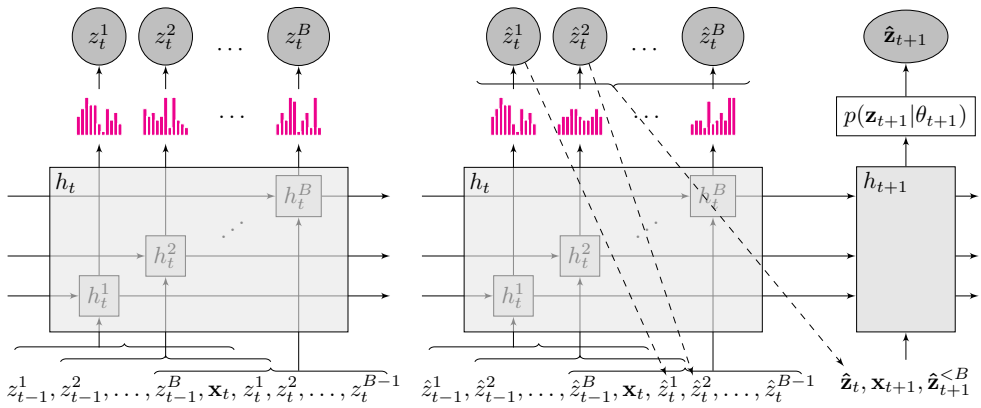
$$h_t = \text{rnn}(h_{t-1}, z_{t-1}, x_t)$$

and $y(\cdot)$ is a function that maps the output of the LSTM to the parameters of a parametric distribution $p(z_t|\theta_t)$. For example, $p(z_t|\theta_t)$ could be a Gaussian and $y(\cdot)$ output the Gaussian's mean and variance as θ_t. Fig. 3a illustrates the overall architecture (with additions for C2FAR noted). Information about the conditioning range $z_1 \dots z_T$ is conveyed through the state of the network at time T (i.e., h_T). From an encoder-decoder perspective [51, 3], DeepAR uses the same network to encode and decode.

For training, each series is sliced into multiple *windows*, i.e., conditioning+prediction ranges at different start points. Windows are normalized using their conditioning ranges, and LSTM and projection

[1]Note full C2FAR densities are never explicitly manifested when training or predicting; we need only compute the top-level distribution, then finer-grained distributions within each observed (or sampled) coarser bin. For plotting (e.g., Fig. 6), we explicitly compute likelihood at tiny increments over a predefined range.

[2]By parameterizing the quantile function with linear splines, SQF-RNN [21] actually enforces piecewise uniformity, over a finite range. Neural spline flows [16] require derivatives of the spline functions to match at knots, to avoid "numerical issues." By parameterizing the PDF directly using classifiers, and only manifesting the portion needed, C2FAR has none of the above restrictions, while being more efficient and well-behaved.



(a) Training (b) Prediction

Figure 4: C2FAR-RNN model detailing high-level unit, h_t, from Fig. 3. We train to minimize sum of NLL at each z_t^i under a categorical with parameters given by a level-specific RNN, h_t^i. We predict by sequentially sampling a discretization at each time step, $\mathbf{z}_t$, which is re-input at step $t+1$.

layer $y(\cdot)$ parameters are learned by minimizing NLL of prediction range outputs. To generate a forecast for a given (normalized) conditioning range, DeepAR draws samples $\hat{z}_{T+1} \dots \hat{z}_{T+N}$ in sequence from $p(z_t|\theta_t)$ (Fig. 3b). Sampled roll-outs are unnormalized, and by repeating this procedure many times, a Monte Carlo estimate of (3) is obtained, from which desired forecast quantiles can be derived (see Fig. 7 for examples).

4.2 Forecasting with C2FAR: C2FAR-RNN

We augment the DeepAR model by first converting all time series to their C2FAR discretization $\mathbf{z}_t = d(z_t)$. Rather than generating a single set of parameters at each time step (θ_t), we generate C2FAR categorical parameters autoregressively *within* each time step (Fig. 4). Each generated bin index, z_t^{i-1}, thus informs the distribution of the next bin index at that time step z_t^i. We also leverage information about discretized values at earlier time steps by replacing C2FAR's classifier-based conditionals (§3, Fig. 2) with LSTMs, one for each level in the C2FAR hierarchy. Intuitively, when generating bin index z_t^i, our inputs comprise both our current position in higher-level bins (e.g. z_t^{i-1}) *and* the ith-level index at the previous time step (z_{t-1}^i), along with the previous LSTM state, h_{t-1}^i.

We call the resulting system C2FAR-RNN. As with DeepAR, C2FAR-RNN is trained to minimize NLL of observed outputs in (normalized) prediction ranges (Fig. 4a). During training, and when evaluating NLL of test sequences, all values are known and all distributions can be computed *in parallel*. When predicting (Fig. 4b), we must sample each bin index z_t^i *sequentially* at each time step.

Recall that C2FAR has an implicit $B+1$ level, where a real value is generated from a parametric distribution (Eqn. (2)). We do not use an RNN for this level; the uniform distributions are fully-specified by the interval endpoints, while we generate Pareto parameters (α_1, α_2) via a simple feed-forward neural network (with a single hidden layer and softplus output transformation). To help inform the Pareto networks, we also encode real-valued z_{t-1} as an additional input/covariate.

While C2FAR is trained for NLL, it is *tuned* for a given target metric. For forecasting, that metric is multi-step-ahead error, and thus we run sampling on our validation set periodically during training in order to evaluate forecasts. For tasks such as anomaly detection, denoising, etc., we would train for NLL and tune for the application-specific metric. While there is no one-size-fits-all metric for evaluating generative models, log-likelihood itself is sometimes regarded as the de facto standard [53]. Ironically, this is the one loss function C2FAR can *not* tune for, at least not on discrete data, because of the density spike issue (§2.2); if we tune directly for NLL, the tuner chooses more and more bins and levels, leading to narrower and higher spikes. If we *must* tune for log-likelihood, we could use discrete distributions in the $B+1$ level. We could also add uniform noise, as other approaches do (but note prior work does this to enable *training*, not *tuning*, on discrete data).

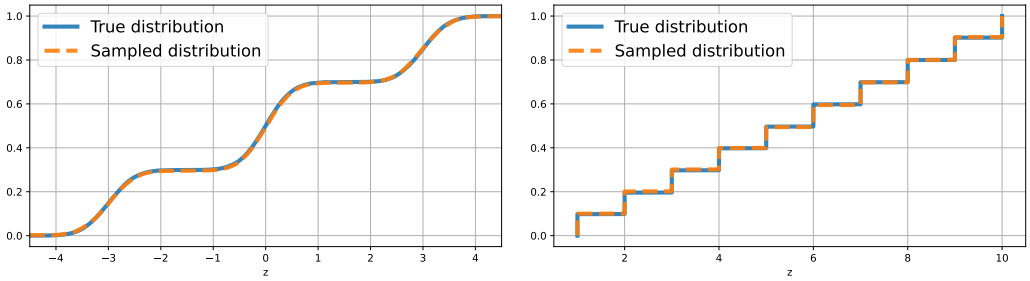


Figure 5: Distribution recovery: CDFs for a Gaussian mixture (left) and discrete uniform over $1 \ldots 10$ (right). After training C2FAR-RNNs on each, we sample paths and plot the *sampled* CDF. C2FAR recovers the true CDF much better than SQF-RNN or IQN-RNN (cf. Fig. 3 in [21], Fig. 2 in [24]).

4.3 Limitations and broader impact of C2FAR forecasting

Every B-level C2FAR model has an equally *expressive* [40] flat counterpart with a single categorical over all the fine-grained intervals; given unlimited training data, C2FAR may therefore not offer modeling benefits over flat binning. Conversely, when there is limited data, simple parametric distributions, with fewer parameters, may generalize better than C2FAR. Also, given it has extra hyperparameters (number of levels/bins-per-level), and given tuning search space grows exponentially with added hyperparameters, C2FAR may not discover optimum settings as quickly. We investigate experimentally whether benefits of C2FAR outweigh these drawbacks on real-world data (§5.2).

C2FAR is more *complex* than flat binning, but whether it is more or less *efficient* will depend on implementation. Compared to their 2-level coarse-to-fine model, WaveRNN [28] found flat binning required "significantly more parameters, memory and compute." In our implementation, C2FAR runs slower, but with less memory (see supplement). If flat binning was truly more efficient, we could always generate a large volume of data from a C2FAR model and *uptrain* [10, 38] a flat model on it.

By enabling simultaneous modeling of discrete and continuous series, without human involvement, C2FAR-RNN is a step toward a universal forecast model. We could train a large C2FAR model on vast quantities of diverse series (similar to efforts in text [14, 33, 61] and vision [48, 52, 25]). Such models could be fine-tuned for new domains and tasks, and help make highly-accurate forecasting systems more widely-used, improving decision making and resource allocation.

There are also risks to this approach. Very large models have environmental and financial costs [4], which may be unnecessary when smaller models suffice. A universal model may be used more easily by those with less expertise and this may lead to misuse; for example, *automation bias* has been shown to disproportionately affect those with less domain expertise [9]. Recommended usage and possible misuse should be documented through artifacts such as model cards [35] and datasheets [22].

5 Experiments

C2FAR is implemented in PyTorch [37], using a 2-layer LSTM [26] with intra-layer dropout [50, 34], trained via Adam [29]. Notation C2FAR-RNN$_i$ refers to an i-level C2FAR model. C2FAR-RNN$_1$ is essentially the standard flat-binning approach (but trained for continuous likelihood).

5.1 Distribution recovery from synthetic data

We first evaluate C2FAR-RNN's ability to recover the distribution of synthetic data, following [21, 24]. We generate data exactly as in [21], creating 500 series with a year's worth of hourly observations, each value sampled independently from a 3-component Gaussian mixture with weights $[0.3, 0.4, 0.3]$, means $[-3, 0, 3]$ and standard deviation of 0.4. To illustrate C2FAR-RNN's ability to model discrete data, we create an equivalent dataset where every element is drawn from a discrete uniform, $\mathcal{U}\{1, 10\}$.

On each dataset we trained a C2FAR-RNN$_3$ model with 20 bins per level and hyperparameters based on *elec* experiments (below). In prior work, SQF-RNN [21] and IQN-RNN [24] struggled to recover the Gaussian mixture, but C2FAR can fit both distributions perfectly (Fig. 5).

Table 1: ND, wQL, Cov80, and Cov99 for our implementation (top), results in [39] (middle, denoted †) and [24] (bottom, denoted ‡), where available. In all cases, flat binned C2FAR-RNN$_1$ improves on DeepAR-Gaussian, while deeper C2FAR-RNN$_2$ likewise improves over C2FAR-RNN$_1$. Results are generally superior to prior state-of-the-art output distributions in RNN-based forecasting.

	ND%				wQL%				Cov80%	Cov99%
	elec	*traff*	*wiki*	*azure*	*elec*	*traff*	*wiki*	*azure*	*azure*	*azure*
Naive	40.8	73.6	35.7	3.49	40.8	73.6	35.7	3.49	-	-
DeepAR-Gaussian	7.05	16.1	43.8	3.60	5.60	13.7	54.7	3.06	89.9/16.9	98.0/37.7
C2FAR-RNN$_1$	6.14	13.0	24.6	2.95	4.87	10.7	21.3	2.41	83.6/**8.3**	98.5/32.2
C2FAR-RNN$_2$	6.09	**12.9**	24.2	2.86	4.83	**10.6**	**21.0**	2.31	**79.0**/8.5	98.4/**29.1**
C2FAR-RNN$_3$	**6.00**	13.3	**24.1**	**2.77**	**4.76**	10.9	**21.0**	**2.27**	86.0/8.9	**98.6**/32.7
DeepAR-Binned†	8.21	23.2	94.6	-	6.47	18.8	84.7	-	-	-
DeepAR-StudentT†	6.95	14.6	26.9	-	5.71	12.2	23.8	-	-	-
IQN-RNN‡	7.40	16.8	**24.1**	-	-	-	-	-	-	-
SQF-RNN‡	9.70	18.6	32.8	-	-	-	-	-	-	-
DeepAR-StudentT‡	7.80	21.6	27.0	-	-	-	-	-	-	-

5.2 Empirical study on real-world data

We evaluate C2FAR forecasting models on the following datasets:

- *elec*: hourly electricity usage of 321 customers [15], using *discretized* version from [30].
- *traff*: real-valued hourly occupancy from 0 to 1 for 862 car lanes [15], using version from [62].
- *wiki*, daily integer number of hits for 9535 pages, first used in [21], using same version as in [24].
- *azure*, (discrete) hourly usage of virtual machine (VM) flavors and flavor groupings (by tenant, deployment type, etc.) in a large public cloud, based on data from [13]. We release this dataset publicly as a paper supplement.

For *azure*, we use 20 days as training, 3 days for validation, and 3 final days for testing. Other splits are the same as in [43]. We forecast 24 hours for hourly series, 30 days for *wiki*, and use rolling evals as in [24]. We evaluate our forecast distributions using *weighted quantile loss* (wQL), evaluated at forecast quantiles $\{0.1, 0.2 \ldots 0.9\}$, as in [39]. For point forecasts, we use the median of our distribution and compute *normalized deviation* (ND) from true values. We also test the *calibration* and *sharpness* [58] of each system, measuring coverage of true values within particular percentiles (e.g. from 10% to 90%, coverage closer to 80% is better) and normalized width of this interval (lower is better). For a target coverage of X%, we refer to these metrics together as *CovX* (e.g. Cov80).

We compare our model to the *Naive* baseline [27], which outputs the last-observed historical value at all horizons, and DeepAR-Gaussian, which is our implementation of DeepAR with a Gaussian output distribution; Gaussian outputs are common in practice, even on discrete data [32, 44, 12].

Note deeper C2FAR models actually include shallower models as special cases. We therefore seek to answer the question: does the improved modeling of C2FAR outweigh the larger amount of tuning required? For fair comparison of models, we establish a *parameter budget* [34] of 1M parameters; after sampling other hyperparameters, we restrict the possible number of LSTM hidden units so that this cap is enforced. We also restrict the total number of tuning evaluations to 100 for each system. We use Optuna [1] for tuning, with the TPESampler [8], and use MedianPruning and early stopping.

Table 1 affirms that deeper C2FAR models do offer value over flat binnings. In only one case (*traff*) did a C2FAR model not improve over flat binning. In this case, C2FAR-RNN$_3$, with a larger tuning search space, also failed to find a superior setting of hyperparameters on the validation set; going forward, using the same number of bins at each level could enable deeper C2FAR models without additional tuning. Contrary to prior work, we also find flat binning much more effective than standard parametric distributions. We attribute this difference to our use of systematic tuning; prior work used a fixed 1024 output bins [39], while our tuner often selected quite fewer bins for C2FAR-RNN$_1$ on our datasets. In terms of both standard and extreme quantiles (Cov88% and Cov99% in Table 1), we find C2FAR models are both better calibrated than DeepAR-Gaussian (being twice as close to the desired coverage) while also having a sharper prediction interval width.

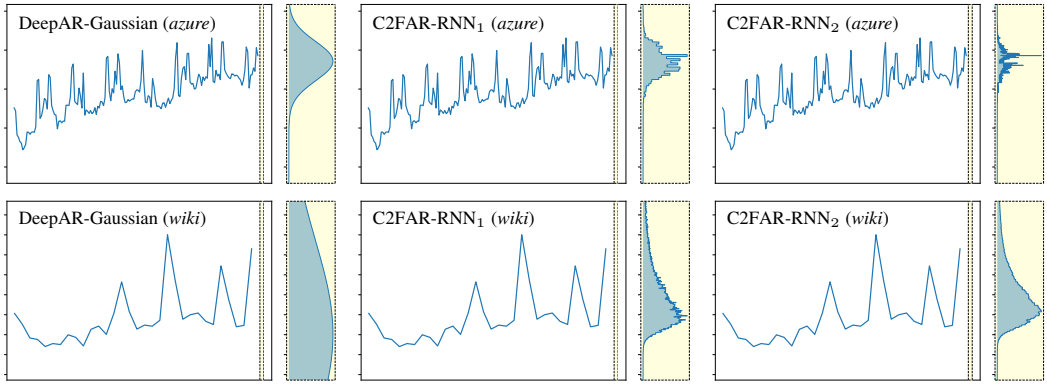


Figure 6: Distributions over possible next values. To cover likely values, Gaussians (left) also cover many unlikely ones. Flat binnings (middle) are optimized to either use fewer bins, but suffer in precision (top), or many bins but suffer in noise (bottom). C2FAR-RNN$_2$ (right) is able to place high probability on a repeat of the previous value (top) or generate smooth distributions (bottom).

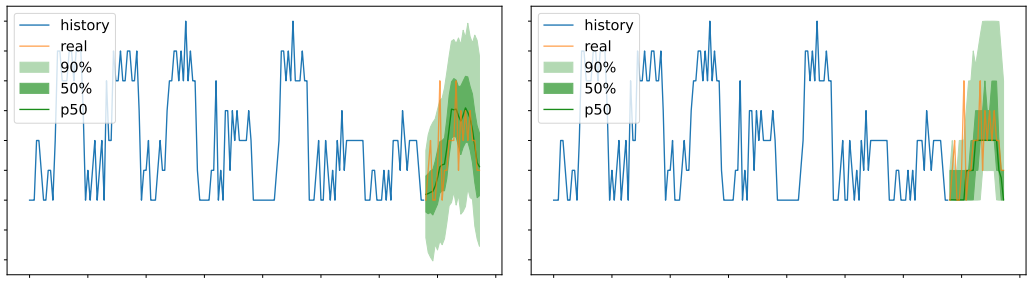


Figure 7: Forecast for an *elec* time series. DeepAR-Gaussian (left) yields a reasonable p50, but its lower percentiles span too low. C2FAR-RNN$_2$ yields better lower quantiles. C2FAR percentiles also suggest high-fidelity roll-outs, i.e., samples that closely mimic the discretized dynamics of the series.

Analyzing the distributions, we find C2FAR models are able to achieve good *precision* (having many intervals), while simultaneously learning smooth distributions (Fig. 6). Flat binnings must first *learn* "that a value of 128 is close to a value of 127 or 129" [42], while for C2FAR models, intervals will implicitly be close in probability because they are in the same coarser bins.

C2FAR generates both better quantiles and higher-fidelity samples (Fig. 7). Autoregressive models have a known disconnect between training, when true values are used as inputs, and generation, when samples are used [5]. This disconnect may be especially problematic when we train on discrete data but use continuous outputs; here C2FAR's realistic samples may be especially advantageous.

Please refer to the supplement for further experiments, analysis, definitions, and experimental details.

6 Conclusion

We presented C2FAR, a new method for distribution modeling. C2FAR reduces the generative process to a sequence of classifications over a discretized representation, with special handling of data outside the binning range. It can be applied to a variety of neural architectures, including RNN-based probabilistic forecasting, where it demonstrates state-of-the-art results for both recovering synthetic distributions and forecasting real-world data. We show binned models (whether flat or, especially, coarse-to-fine) are more effective than parametric distributions — if binning precision is tuned.

Analysis shows that C2FAR can successfully model complex, multi-modal densities, in real or discrete data, without any prior information. This flexibility enables improved modeling for a variety of time series use cases, including forecasting, anomaly detection, interpolation, compression, denoising, and generating high-fidelity samples. It also enables development of large-scale forecasting models trained on diverse datasets, i.e., it is a step toward a universal neural forecaster.

References

[1] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. 2019. Optuna: A next-generation hyperparameter optimization framework. In *ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2623–2631.

[2] Alexander Alexandrov, Konstantinos Benidis, Michael Bohlke-Schneider, Valentin Flunkert, Jan Gasthaus, Tim Januschowski, Danielle C Maddix, Syama Rangapuram, David Salinas, Jasper Schulz, Stella Lorenzo, Ali Caner Türkmen, and Yuyang Wang. 2020. GluonTS: Probabilistic and Neural Time Series Modeling in Python. *Journal of Machine Learning Research* 21, 116 (2020), 1–6.

[3] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. *International Conference on Learning Representation* (2015).

[4] Emily M Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. 2021. On the Dangers of Stochastic Parrots: Can Language Models Be Too Big?. In *ACM Conference on Fairness, Accountability, and Transparency*. 610–623.

[5] Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. *Advances in Neural Information Processing Systems* 28 (2015).

[6] Yoshua Bengio and Samy Bengio. 2000. Modeling High-Dimensional Discrete Data with Multi-Layer Neural Networks. In *Advances in Neural Information Processing Systems*.

[7] Konstantinos Benidis, Syama Sundar Rangapuram, Valentin Flunkert, Bernie Wang, Danielle Maddix, Caner Turkmen, Jan Gasthaus, Michael Bohlke-Schneider, David Salinas, Lorenzo Stella, et al. 2020. Neural forecasting: Introduction and literature overview. *arXiv preprint arXiv:2004.10240* (2020).

[8] James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. 2011. Algorithms for hyper-parameter optimization. *Advances in Neural Information Processing Systems* 24 (2011).

[9] Raymond R Bond, Tomas Novotny, Irena Andrsova, Lumir Koc, Martina Sisakova, Dewar Finlay, Daniel Guldenring, James McLaughlin, Aaron Peace, Victoria McGilligan, Stephen J. Leslie, Hui Wang, and Marek Malik. 2018. Automation bias in medicine: The influence of automated diagnoses on interpreter accuracy and uncertainty when reading electrocardiograms. *Journal of electrocardiology* 51, 6 (2018), S6–S11.

[10] Cristian Buciluǎ, Rich Caruana, and Alexandru Niculescu-Mizil. 2006. Model compression. In *ACM SIGKDD International Conference On Knowledge Discovery & Data Mining*. 535–541.

[11] Defu Cao, Yujing Wang, Juanyong Duan, Ce Zhang, Xia Zhu, Congrui Huang, Yunhai Tong, Bixiong Xu, Jing Bai, Jie Tong, Qi Zhang, et al. 2020. Spectral Temporal Graph Neural Network for Multivariate Time-series Forecasting. *Advances in Neural Information Processing Systems* 33 (2020).

[12] Yitian Chen, Yanfei Kang, Yixiong Chen, and Zizhuo Wang. 2020. Probabilistic forecasting with temporal convolutional neural network. *Neurocomputing* 399 (2020), 491–501.

[13] Eli Cortez, Anand Bonde, Alexandre Muzio, Mark Russinovich, Marcus Fontoura, and Ricardo Bianchini. 2017. Resource Central: Understanding and predicting workloads for improved resource management in large cloud platforms. In *Symposium on Operating Systems Principles*. 153–167.

[14] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

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

[16] Conor Durkan, Artur Bekasov, Iain Murray, and George Papamakarios. 2019. Neural spline flows. *Advances in Neural Information Processing Systems* 32 (2019).

[17] Elena Ehrlich, Laurent Callot, and François-Xavier Aubet. 2021. Spliced Binned-Pareto Distribution for Robust Modeling of Heavy-tailed Time Series. *arXiv preprint arXiv:2106.10952* (2021).

[18] Chenyou Fan, Yuze Zhang, Yi Pan, Xiaoyue Li, Chi Zhang, Rong Yuan, Di Wu, Wensheng Wang, Jian Pei, and Heng Huang. 2019. Multi-horizon time series forecasting with temporal attention learning. In *ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2527–2535.

[19] Usama Fayyad and Keki Irani. 1993. Multi-interval discretization of continuous-valued attributes for classification learning. *International Joint Conference On Artificial Intelligence* 2 (1993), 1022–1027.

[20] Brendan J Frey, Geoffrey E Hinton, and Peter Dayan. 1995. Does the wake-sleep algorithm produce good density estimators? *Advances in Neural Information Processing Systems* 8 (1995).

[21] 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 *International conference on artificial intelligence and statistics*. 1901–1910.

[22] Timnit Gebru, Jamie Morgenstern, Briana Vecchione, Jennifer Wortman Vaughan, Hanna Wallach, Hal Daumé III, and Kate Crawford. 2021. Datasheets for datasets. *Commun. ACM* 64, 12 (2021), 86–92.

[23] Mathieu Germain, Karol Gregor, Iain Murray, and Hugo Larochelle. 2015. MADE: Masked autoencoder for distribution estimation. In *International Conference on Machine Learning*. PMLR, 881–889.

[24] Adèle Gouttes, Kashif Rasul, Mateusz Koren, Johannes Stephan, and Tofigh Naghibi. 2021. Probabilistic Time Series Forecasting with Implicit Quantile Networks. *arXiv preprint arXiv:2107.03743* (2021).

[25] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *IEEE conference on computer vision and pattern recognition*. 770–778.

[26] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[27] Rob J Hyndman and George Athanasopoulos. 2018. *Forecasting: Principles and Practice*. OTexts.

[28] Nal Kalchbrenner, Erich Elsen, Karen Simonyan, Seb Noury, Norman Casagrande, Edward Lockhart, Florian Stimberg, Aaron van den Oord, Sander Dieleman, and Koray Kavukcuoglu. 2018. Efficient neural audio synthesis. In *International Conference on Machine Learning*. PMLR, 2410–2419.

[29] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[30] Guokun Lai, Wei-Cheng Chang, Yiming Yang, and Hanxiao Liu. 2018. Modeling long-and short-term temporal patterns with deep neural networks. In *ACM SIGIR Conference on Research & Development in Information Retrieval*. 95–104.

[31] Hugo Larochelle and Iain Murray. 2011. The neural autoregressive distribution estimator. In *International conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 29–37.

[32] 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. *Advances in Neural Information Processing Systems* 32 (2019).

[33] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized BERT pretraining approach. *arXiv preprint arXiv:1907.11692* (2019).

[34] Gábor Melis, Chris Dyer, and Phil Blunsom. 2017. On the state of the art of evaluation in neural language models. *arXiv preprint arXiv:1707.05589* (2017).

[35] Margaret Mitchell, Simone Wu, Andrew Zaldivar, Parker Barnes, Lucy Vasserman, Ben Hutchinson, Elena Spitzer, Inioluwa Deborah Raji, and Timnit Gebru. 2019. Model cards for model reporting. In *Conference on Fairness, Accountability, and Transparency*. 220–229.

[36] Srayanta Mukherjee, Devashish Shankar, Atin Ghosh, Nilam Tathawadekar, Pramod Kompalli, Sunita Sarawagi, and Krishnendu Chaudhury. 2018. AR-MDN: Associative and recurrent mixture density networks for eretail demand forecasting. *arXiv preprint arXiv:1803.03800* (2018).

[37] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Tejani Alykhan, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems* 32 (2019).

[38] Slav Petrov, Pi-Chuan Chang, Michael Ringgaard, and Hiyan Alshawi. 2010. Uptraining for Accurate Deterministic Question Parsing. In *Empirical Methods in Natural Language Processing*. 705–713.

[39] 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. *arXiv preprint arXiv:2005.10111* (2020).

[40] Maithra Raghu, Ben Poole, Jon Kleinberg, Surya Ganguli, and Jascha Sohl-Dickstein. 2017. On the expressive power of deep neural networks. In *International Conference on Machine Learning*. PMLR, 2847–2854.

[41] Kashif Rasul, Abdul-Saboor Sheikh, Ingmar Schuster, Urs Bergmann, and Roland Vollgraf. 2020. Multi-variate probabilistic time series forecasting via conditioned normalizing flows. *arXiv preprint arXiv:2002.06103* (2020).

[42] Tim Salimans, Andrej Karpathy, Xi Chen, and Diederik P Kingma. 2017. PixelCNN++: Improving the PixelCNN with discretized logistic mixture likelihood and other modifications. *arXiv preprint arXiv:1701.05517* (2017).

[43] David Salinas, Michael Bohlke-Schneider, Laurent Callot, Roberto Medico, and Jan Gasthaus. 2019. High-dimensional multivariate forecasting with low-rank Gaussian copula processes. *Advances in Neural Information Processing Systems* 32 (2019).

[44] David Salinas, Valentin Flunkert, Jan Gasthaus, and Tim Januschowski. 2020. DeepAR: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting* 36, 3 (2020), 1181–1191.

[45] Matthias W Seeger, David Salinas, and Valentin Flunkert. 2016. Bayesian intermittent demand forecasting for large inventories. *Advances in Neural Information Processing Systems* 29 (2016).

[46] Rajat Sen, Hsiang-Fu Yu, and Inderjit Dhillon. 2019. Think globally, act locally: A deep neural network approach to high-dimensional time series forecasting. *arXiv preprint arXiv:1905.03806* (2019).

[47] Jin Shieh and Eamonn Keogh. 2008. iSAX: indexing and mining terabyte sized time series. In *ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 623–631.

[48] Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).

[49] Ralph D Snyder, J Keith Ord, and Adrian Beaumont. 2012. Forecasting the intermittent demand for slow-moving inventories: A modelling approach. *International Journal of Forecasting* 28, 2 (2012), 485–496.

[50] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research* 15, 1 (2014), 1929–1958.

[51] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to Sequence Learning with Neural Networks. In *Advances in Neural Information Processing Systems*, Vol. 27.

[52] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going deeper with convolutions. In *IEEE conference on computer vision and pattern recognition*. 1–9.

[53] Lucas Theis, Aäron van den Oord, and Matthias Bethge. 2015. A note on the evaluation of generative models. *arXiv preprint arXiv:1511.01844* (2015).

[54] Benigno Uria, Marc-Alexandre Côté, Karol Gregor, Iain Murray, and Hugo Larochelle. 2016. Neural autoregressive distribution estimation. *The Journal of Machine Learning Research* 17, 1 (2016), 7184–7220.

[55] Benigno Uria, Iain Murray, and Hugo Larochelle. 2013. RNADE: The real-valued neural autoregressive density-estimator. *Advances in Neural Information Processing Systems* 26 (2013).

[56] Aaron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. 2016. Pixel recurrent neural networks. In *International Conference on Machine Learning*. PMLR, 1747–1756.

[57] Ravichander Vipperla, Sangjun Park, Kihyun Choo, Samin Ishtiaq, Kyoungbo Min, Sourav Bhattacharya, Abhinav Mehrotra, Alberto Gil CP Ramos, and Nicholas D Lane. 2020. Bunched LPCNet: Vocoder for low-cost neural text-to-speech systems. *arXiv preprint arXiv:2008.04574* (2020).

[58] Ruofeng Wen, Kari Torkkola, Balakrishnan Narayanaswamy, and Dhruv Madeka. 2017. A multi-horizon quantile recurrent forecaster. *arXiv preprint arXiv:1711.11053* (2017).

[59] Haixu Wu, Jiehui Xu, Jianmin Wang, Mingsheng Long, et al. 2021. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. *Advances in Neural Information Processing Systems* 34 (2021).

[60] Tsun-Yi Yang, Yi-Hsuan Huang, Yen-Yu Lin, Pi-Cheng Hsiu, and Yung-Yu Chuang. 2018. SSR-Net: A Compact Soft Stagewise Regression Network for Age Estimation.. In *International Joint Conference On Artificial Intelligence*. 1078–1084.

[61] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. XLNet: Generalized autoregressive pretraining for language understanding. *Advances in Neural Information Processing Systems* 32 (2019).

[62] Hsiang-Fu Yu, Nikhil Rao, and Inderjit S Dhillon. 2016. Temporal regularized matrix factorization for high-dimensional time series prediction. *Advances in Neural Information Processing Systems* 29 (2016).

[63] Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. 2021. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] In particular, §4.3
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] §4.3
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [No] The code is proprietary, however we are working to share an open source implementation. All datasets are either shared with the paper or already public.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See §5.2 and paper supplement.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] See paper supplement.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See paper supplement.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] §5.2
   (b) Did you mention the license of the assets? [Yes] For shared data, see supplement
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] See paper supplement.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes] See paper supplement.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [Yes] See paper supplement.
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]