Abstract

Probabilistic forecasting of time series is an important matter in many applications and research fields. In order to draw conclusions from a probabilistic forecast, we must ensure that the model class used to approximate the true forecasting distribution is expressive enough. Yet, characteristics of the model itself, such as its uncertainty or its general functioning are not of lesser importance. In this paper, we propose Autoregressive Transformation Models (ATMs), a model class inspired from various research directions such as normalizing flows and autoregressive models. ATMs unite expressive distributional forecasts using a semi-parametric distribution assumption with an interpretable model specification and allow for uncertainty quantification based on (asymptotic) Maximum Likelihood theory. We demonstrate the properties of ATMs both theoretically and through empirical evaluation on several simulated and real-world forecasting datasets.

1 INTRODUCTION

In prediction tasks, a common notion of uncertainty is the distinction between aleatoric uncertainty and epistemic uncertainty (Hüllermeier and Waegeman, 2021). Aleatoric uncertainty stems from the inherent randomness of the data generating process characterized by the cumulative distribution function (CDF) \( F_{Y \mid x}(y \mid x) \), relating the outcome \( y \) to features \( x \), and is thus irreducible. Epistemic uncertainty, in contrast, is a model characteristic that can be reduced to zero with growing data size and appropriate model complexity. Approaches that can explicitly model or quantify both uncertainties are, however, less common or restricted in their expressiveness. Standard Gaussian processes (GP), for example, model both types of uncertainty, but with the aleatoric uncertainty (the observation noise term) being restricted to homoscedasticity. A heteroscedastic GP (Lázaro-Gredilla and Titsias, 2011) or a GP conditional density estimator (Durdorj et al., 2018) address this issue, but are still based on a pre-specified parametric distribution assumption. Bayesian approaches (e.g., Blundell et al., 2015) induce uncertainties by imposing distributions for model weights \( \theta \) and rely on a (predictive) posterior for inference statements that also accounts for...
the epistemic uncertainty. Yet, the resulting inference is still based on an (approximate) parametric distribution that is in many cases too restrictive (Kingma et al., 2016) and influenced by prior assumptions.

A recent, popular solution to the limited expressiveness of parametric posterior distributions are normalizing flows (Papamakarios et al., 2019). Flows turn a simple distribution into a potentially complex one using (multiple) parameterized transformations and thereby allow to minimize the discrepancy between the model’s CDF \( F_{Y|X}(y|x, \theta) \) and the true CDF \( F_{Y}(y|x) \). This discrepancy, the so-called structural uncertainty, is only one part of the epistemic uncertainty. Another source of epistemic uncertainty, the parametric uncertainty (PU), stems from estimating the usually unknown parameters \( \theta \) (given a fixed model structure) and is not accounted for by flow-based methods (Liu et al., 2019). Other approaches for epistemic uncertainty quantification (UQ) in machine learning such as deep ensembles (Lakshminarayanan et al., 2017) quantify for model uncertainty, but are not well studied for time series or do not allow for specific treatment of the aleatoric uncertainty. On top of that, these methods are also difficult to interpret. While approaches like ensembles perform well in quantifying predictive uncertainty, it is not immediately clear what parts of the model cause uncertainty. Ideally, we would like to conduct probabilistic forecasting of time series that is both more expressive than parametric (deep) autoregressive models and also allow for interpretation as well as parametric UQ (cf. Figure 1).

**Our contributions** In order to achieve these goals, we propose a new and general class of semi-parametric autoregressive models for time series analysis called *autoregressive transformation models* (ATMs; Section 3) that learn expressive distributions based on interpretable parametric transformations. We also derive asymptotic results for estimated parameters for a special class of ATMs (Section 4.2) and thereby allow for parametric UQ in a non-Bayesian manner, i.e., without the need of specifying prior distributions. Finally, we provide evidence for the efficacy of our proposal both with numerical experiments based on simulated data and by comparing ATMs against other existing state-of-the-art methods in benchmarks.

2 BACKGROUND

Approaches that model the conditional density can be distinguished by their underlying distribution assumption. Approaches can be parametric, such as mixture density networks or GPs for conditional density estimation and then learn the parameters of a pre-specified parametric distribution, or non-parametric such as Bayesian non-parametrics (Dunson, 2010). A third line of research that we describe as semi-parametric or *parametric transforming*, are approaches that start with a simple parametric distribution assumption \( F_Z \) and end up with a far more flexible distribution \( F_{Y|X} \) by transforming \( F_Z \) (multiple times). Such approaches have sparked great interest in recent years, triggered by research ideas such as density estimation using non-linear independent components estimation or real-valued non-volume preserving transformations (Dinh et al., 2015, 2017). A general notion of such transformations is known as normalizing flow (NF; Papamakarios et al., 2019), where realizations \( z \sim F_Z \) of an error distribution \( F_z \) are transformed to observations \( y \) via

\[
y = h_k \circ h_{k-1} \circ \cdots \circ h_1(z)
\]

using \( k \) transformation functions. Many different approaches exist to define expressive flows. These are often defined as a chain of several transformations or an expressive neural network and allow for universal representation of \( F_{Y|X} \) (Papamakarios et al., 2019). Autoregressive models (e.g., Bengio and Bengio, 1999; Uria et al., 2010) for distribution estimation of continuous variables are a special case of NFs, more precisely autoregressive flows (AFs; Kingma et al., 2016, Papamakarios et al., 2017), with a single transformation. Similar, suitably parameterized transformation models (TMs; Hothorn et al., 2014) are an alternative to NFs using also only a single transformation function. The transformation in TMs is chosen to be expressive enough on its own and comes with desirable approximation guarantees. Instead of a transformation from \( z \) to \( y \), TMs define an inverse flow \( h(y) = z \). The key idea of TMs is that many well-known statistical regression models can be represented by a base distribution \( F_Z \) and some transformation \( h \). Prominent examples include linear regression or the Cox proportional hazards model (Cox, 1972), which can both be seen as a special case of TMs (Hothorn et al., 2014). Various authors have noted the connection between autoregressive models and NFs (e.g., Papamakarios et al., 2019) and between TMs and NFs (e.g., Sick et al., 2021). Advantages of TMs and conditional TMs (CTMs) are their parsimony in terms of parameters, interpretability of the input-output relationship and existing theoretical results on epistemic uncertainty (Hothorn et al., 2018). This also led to various recent TM advancements both in the field of machine learning (see, e.g., Van Belle et al., 2011) and in deep learning (see, e.g., Baumann et al., 2021, Kook et al., 2021).

\[^1\text{The equivalent to the error distribution in flows.}\]
2.1 Transformation models

Parametrized transformation models as proposed by [Hothorn et al., 2014, 2018] are likelihood-based approaches to estimate the CDF $F_Y$ of $Y$. The main ingredient of TMs is a monoton increasing transformation function $h$ to convert a simple error distribution $F_Z$ to a more complex and appropriate CDF $F_Y$. Conditional TMs (CTMs) work analogously for the conditional distribution of $Y$ given features $x \in \chi$ from feature space $\chi$:

$$F_{Y|x}(y) = \mathbb{P}(Y \leq y|x) = F_Z(h(y|x)).$$  \hspace{5cm} (2)

CTMs learn $h(y|x)$ from the data. A convenient parameterization of $h$ for continuous $Y$ are Bernstein polynomials (BSPs; Farouki, 2012) with order $M$ (usually $M \ll 50$). BSPs are motivated by the Bernstein approximation (Bernstein, 1912) with uniform convergence guarantees for $M \to \infty$, while also being easily invertible and computationally attractive with only $M+1$ parameters. BSPs further have easy and analytically accessible derivatives, which makes them a particularly interesting choice for the change of random variables. We denote the BSP basis by $a_M : \Xi \mapsto \mathbb{R}^{M+1}$ with sample space $\Xi$. The transformation function $h$ is then defined as $h(y|x) = a_M(y)^	op \vartheta(x)$ with feature-dependent basis coefficients $\vartheta$. This can be seen as an evaluation of $y$ based on a mixture of Beta densities $f_{Be}(\kappa,\mu)$ with different distribution parameters $\kappa, \mu$ and weights $\vartheta(x)$:

$$a_M(y)^	op \vartheta(x) = \sum_{m=0}^{M} \vartheta_m(x)f_{Be(m+1,M-m+1)}(y) \frac{1}{M+1},$$  \hspace{5cm} (3)

where $\tilde{y}$ is a rescaled version of $y$ to ensure $\tilde{y} \in [0,1]$. Restricting $\vartheta_m > \vartheta_{m-1}$ for $m = 1, \ldots, M+1$ guarantees monotonicity of $h$ and thus of the estimated CDF. Roughly speaking, using BSPs of order $M$, allows to model the polynomials of degree $M$ of $y$.

2.2 Model definition and interpretability

The transformation function $h$ can include different data dependencies. One common choice [Hothorn, 2020; Baumann et al., 2021] is to split the transformation function into two parts

$$h(y|x) = h_1(y,x) + h_2(x) = a(y)^	op \vartheta(x) + \beta(x),$$  \hspace{5cm} (4)

where $a(y)$ is a pre-defined basis function such as the BSP basis (omitting $M$ for readability in the following), $\vartheta : \chi \mapsto \mathbb{R}^{M+1}$ a conditional parameter function defined on $\chi_\vartheta \subseteq \chi$ and $\beta(x)$ models a feature-induced shift in the target distribution. The flexibility and interpretability of TMs stems from the parameterization

$$\vartheta(x) = \sum_{j=1}^J \Gamma_j \beta_j(x),$$  \hspace{5cm} (5)

where the matrix $\Gamma_j \in \mathbb{R}^{(M+1) \times O_j}, O_j \geq 1$, subsumes all trainable parameters and represents the effect of the interaction between the basis functions in $a$ and the chosen predictor terms $b_j : \chi_b \mapsto \mathbb{R}^{O_j}, \chi_b \subseteq \chi$. The predictor terms $b_j$ have a role similar to base learners in boosting and represent simple learnable functions. For example, a predictor term can be the $j$th feature, $b_j(x) = x_j$, and $\Gamma_j \in \mathbb{R}^{(M+1) \times 1}$ describes the linear effect of this feature on the $M+1$ basis coefficients, i.e., how the feature $x_j$ relates to the density transformation from $Z$ to $Y|x$. Other structured non-linear terms such as splines allow for interpretable lower-dimensional non-linear relationships. Various authors also proposed neural network predictors to allow potentially multidimensional feature effects or to incorporate unstructured data sources (Sick et al., 2021; Baumann et al., 2021; Kook et al., 2021). In a similar fashion, $\beta(x)$ can be defined using various predictors.

Relating features and their effect to the coefficients $a$ in an additive fashion allows to directly assess the impact of each feature on the transformation and also whether changes in the feature just shift the distribution in its location ($\beta(x)$) or if the relationship also transforms other distribution characteristics such as variability or skewness (see, e.g., Baumann et al., 2021).

Relationship with autoregressive flows In the notation of AFs, $h^{-1}(\cdot)$ is known as transformer, a parameterized and bijective function. By the definition of (4), the transformer in the case of TMs is represented by the basis function $a(\cdot)$ and parameters $\vartheta$. In AFs, these transformer parameters are learned by a conditioner, which in the case of TMs are the functions $b_j$. In line with the assumptions made for AFs, these conditioners in TMs do not need to be bijective functions themselves.

3 AUTOREGRESSIVE TRANSFORMATIONS

Inspired by TMs and AFs, we propose autoregressive transformation models (ATMs). The basic idea is to use a parameter-free error distribution $F_Z$ and transform this distribution in an interpretable fashion to obtain $F_{Y|x}$. One of the assumptions of TMs is the stochastic independence of observations, i.e., $Y_i|x_i \perp Y_j|x_j, i \neq j$. When $Y$ is a time-series, this assumption does clearly not hold. In contrast, this assumption is not required for AFs.

Let $t \in T \subseteq \mathbb{N}_0$ be a time index for the time series $(Y_t)_{t \in T}$. Assume

$$Y_t|F_{t-1} \sim G(\vartheta, Y_{t-1}, \ldots, Y_{t-p})$$  \hspace{5cm} (6)
for some $p \in \{1, \ldots, t\}$, distribution $G$, parameter $\theta \in \Theta$ with compact parameter space $\Theta \subset \mathbb{R}^n$ and filtration $\mathcal{F}_s$, $s \in \mathcal{T}$, $s < t$, on the underlying probability space. Assume that the joint distribution of $Y_t, Y_{t-1}, \ldots, Y_1$ possesses the Markov property of order $p$, i.e., the joint distribution, expressed through its absolutely continuous density $f$, can be rewritten as product of its conditionals with $p$ lags:

$$f(y_t, \ldots, y_1 | x) = \prod_{s=2}^{t} f(y_s | y_{s-1}, \ldots, y_{s-p}, x). \quad (7)$$

The time-dependency of $x$ is omitted for better readability here and in the following. Given this autoregressive structure, we propose a time-dependent transformation $h_t$ that extends (C)TMs to account for filtration and time-varying feature information.

**Definition 1 Autoregressive Transformation Models** Let $h_t, t \in \mathcal{T}$, be a time-dependent monotonic transformation function and $F_Z$ the parameter-free error distribution as in Definition 3 in the Supplementary Material. We define autoregressive transformation models as follows:

$$P(Y_t \leq y_t | \mathcal{F}_{t-1}, x) = F_{Y_t | \mathcal{F}_{t-1}, x}(y_t) = F_Z(h_t(y_t | \mathcal{F}_{t-1}, x)). \quad (8)$$

This can be seen as the natural extension of [2] for time-series data with autoregressive property and time-varying transformation function $h_t$. In other words, [8] says that after transforming $y_t$ with $h_t$, its conditional distribution follows the error distribution $F_Z$, or vice versa, a random variable $Z \sim F_Z$ can be transformed to follow the distribution $Y_t | x$ using $h_t^{-1}$.

**Relationship with autoregressive models and autoregressive flows** Autoregressive models (AMs; Bengio and Bengio [1999]) and AFs both rely on the factorization of the joint distribution into conditionals as in (7). Using the CDF of each conditional in (7) as transformer in an AF, we obtain the class of AMs (Papamakarios et al., 2019). AMs and ATMs are thus both (inverse) flows using a single transformation, but with different transformers and, as we will outline in Section 3.2, also with different conditioners.

### 3.1 Likelihood-based estimation

Based on (7), (8) and the change of variable theorem, the likelihood contribution of the $t$th observation $y_t$ in ATMs is given by $f_{Y_t | x}(y_t | \mathcal{F}_{t-1}, x) = f_Z(h_t(y_t | \mathcal{F}_{t-1}, x)) \cdot \left| \frac{\partial h_t(y_t | \mathcal{F}_{t-1}, x)}{\partial y_t} \right|$ and the full likelihood for $T$ observations thus by

$$f_{Y_T | x}(Y_T, \ldots, Y_1 | y_0, x) = \prod_{t=1}^{T} \left\{ f_Z(h_t(y_t | \mathcal{F}_{t-1}, x)) \cdot \left| \frac{\partial h_t(y_t | \mathcal{F}_{t-1}, x)}{\partial y_t} \right| \right\}. \quad (9)$$

where $y_0$ is a known finite starting value and $\mathcal{F}_0$ only contains $y_0$. Based on (9), we define the loss of all model parameters $\theta$ as negative log-likelihood

$$-\ell(\theta) := -\log f_{Y_T | x}(Y_T, \ldots, Y_1 | y_0, x)$$

given by

$$-\sum_{t=1}^{T} \left\{ \log f_Z(h_t(y_t | \mathcal{F}_{t-1}, x)) + \log \left| \frac{\partial h_t(y_t | \mathcal{F}_{t-1}, x)}{\partial y_t} \right| \right\}, \quad (10)$$

and use (10) to train the model.

As for AFs, many special cases can be defined from the above definition and more concrete structural assumptions for $h_t$ make ATMs an interesting alternative to other methods in practice. We will elaborate on meaningful structural assumptions in the following.

### 3.2 Structural assumptions

In CTMs, the transformation function $h$ is usually decomposed as $h(y | x) = h_1(y | x) + h_2(x)$, where $h_1$ is a function depending on $y$ and $h_2$ is a distribution-shift function depending only on $x$. For time-varying transformations $h_t$ our fundamental idea is that the outcome $y_t$ shares the same transformation with its filtration $\mathcal{F}_{t-1}$, i.e., the lags $\mathcal{Y}_t = \{y_{t-1}, \ldots, y_{t-p}\}$. In other words, a transformation applied to the outcome must be equally applied to its predecessor in time to make
sense of the autoregressive structural assumption. An appropriate transformation structure can thus be described by

\[ h_t(y_t|T_{t-1}, x) = h_{1t}(y_t|x) + h_{2t}((h_{1t} \circ Y_t|T_{t-1}, x)|x) \]

\[ =: \lambda_{1t} + \lambda_{2t}, \]

for \( t \in T \), where \( \circ \) indicates the element-wise application of \( h_{1t} \) to all lags in \( Y_t \). In other words, ATMs first apply the same transformation \( h_{1t} \) to all lags in \( y_t \) and individually to \( y_{t-1}, y_{t-2}, \ldots \), and then further consider a transformation function \( h_{2t} \) to shift the distribution based on the transformed filtration. While the additivity assumption of \( \lambda_{1t} \) and \( \lambda_{2t} \) seems restrictive at first glance, the imposed relationship between \( y_t \) and \( Y_t \) only needs to hold in the transformed probability space. For example, \( h_{1t} \) can compensate for a multiplicative autoregressive effect between the filtration and \( y_t \) by implicitly learning a log-transformation (cf. Section 6.1). At the same time, the additivity assumption offers a nice interpretation of the model, also described in Figure 2. After transforming \( y_t \) and \( Y_t \), (11) implies that training an ATM is equal to a regression model of the form \( \lambda_{1t} = \lambda_{2t} + \epsilon \), with additive error term \( \epsilon \sim F_Z \) (cf. Proposition 1 in the Supplementary Material). This also helps explaining why only \( \lambda_{2t} \) depends on \( T_{t-1} \): if \( \lambda_{1t} \) also involves \( T_{t-1} \), ATMs would effectively model the joint distribution of the current time point and the filtration, which in turn contradicts the Markov assumption.

Specifying \( h_{1t} \) very flexible clearly results in overfitting. As for CTMs, we use a feature-driven basis function representation \( h_{1t}(y_t|x) = a(y_t) \phi(x) \) with BSPs \( a \) and specify their weights as in (9). The additional transformation \( h_{2t} \) ensures enough flexibility for the relationship of transformed response and the transformed filtration, e.g., by using a non-linear model or neural network. An interesting special case arises for linear transformations in \( h_{2t} \), which we elaborate in Section 4 in more detail.

Interpretability The three main properties that make ATMs interpretable are 1) their additive predictor structure as outlined in (5); 2) the clear relationship between features and the outcome through the BSP basis, and 3) ATM’s structural assumption as given in (11). As for (generalized) linear models, the additivity assumption in the predictor allows to interpret feature influences through their partial effect ceteris paribus. On the other hand, choices of \( M \) and \( F_Z \) will influence the relationship of features and outcome by inducing different types of models. A normal distribution assumption for \( F_Z \) and \( M = 1 \) will turn ATMs into an additive regression model with Gaussian error distribution (see also Section 4). For \( M > 1 \), features in \( h_1 \) will also influence higher moments of \( Y|x \) and allow more flexibility in modeling \( F_{Y|x} \). For example, a (smooth) monotonously increasing feature effect will induce rising moments of \( Y|x \) with increasing feature values. Other choices for \( F_Z \) such as the logistic distribution also allow for easy interpretation of feature effects (e.g., on the log-odds ratio scale; see [Kook et al. 2021]). Finally, the structural assumption of ATMs enforces that the two previous interpretability aspects are consistent over time. We will provide additional explanation as well as an illustrative example in the Supplementary Material and refer to [Hothorn et al. 2014] for more details on interpretability of CTMs.

Implementation In order to allow for a flexible choice of transformation functions and predictors \( b_j \), we propose to implement ATMs in a neural network. While this allows for complex model definitions, there are also several computational advantages. In a network, weight sharing for \( h_{1t} \) across time points is straightforward to implement and common optimization routines such as Adam ([Kingma and Ba, 2014]) prove to work well for ATMs despite the monotonicity constraints required for the BSP basis. Furthermore, as basis evaluations for a large number of outcome lags in \( T_{t-1} \) can be computationally expensive and add \( M \) additional columns per lag to the feature matrix, an additional advantage is the dynamic nature of mini-batch training. It allows to evaluate the bases only during training and separately in each mini-batch.

4 AT(p) Transformations

A particular interesting special case of ATMs is the AT(p) model. This model class is a direct extension of the well-known autoregressive model of order \( p \) (short AR(p) model) to transformation models.

Definition 2 AT(p) transformations We define AT(p) transformations, a special class of ATMs, by setting \( h_{1t}(y_t|x) = a(y_t) \phi(x) \), and \( h_{2t}(T_{t-1}, x) = \sum_{j=1}^p \phi_j h_{1t}(y_{t-j}) + r(x) \), i.e., an autoregressive shift term with optional exogenous remainder term \( r(x) \).

4.1 Model Details

The AT(p) model is a very powerful and interesting model class for itself, as it allows to recover the classical time series AR(p) model when setting \( M = 1 \), \( \phi(x) \equiv \phi \) and \( r(x) \equiv 0 \) (see Proposition 2 in the Supplementary Material for a proof of equivalence). But it can also be extended to more flexible autoregressive models in various directions. We can increase
$M$ to get a more flexible density, allowing to deviate from the error distribution assumption $F_Z$, e.g., to relax the normal distribution assumption of AR models. Alternatively, incorporating exogenous effects into $h_{1t}$ allows to estimate the density data-driven or to introduce exogenous shifts in time series using features $x$ in $r(x)$. ATMs can also recover well-known transformed autoregressive models such as the multiplicative autoregressive model (Wong and Li [2000]) as demonstrated in Section 6.1. When specifying $h_{1t}$ flexible enough, an AT($p$) model will, e.g., learn the log-transformation function required to transform a multiplicative autoregressive time series to an additive autoregressive time series on the log-scale. In general, this allows the user to learn autoregressive models without the need to find an appropriate transformation before applying the time series model. This means that the uncertainty about preprocessing steps (e.g., a Box-Cox transformation; Sakia [1992]) is incorporated into the model estimation, making parts of the pre-processing obsolete for the modeler and its uncertainty automatically available.

Non-linear extensions of AT($p$) models can be constructed by modeling $\lambda_t$ in $h_{2t}$ non-linear, allowing ATMs to resemble model classes such as non-linear AR models with exogenous terms (e.g., Lin et al. [1996]).

4.2 Asymptotic theory and parametric inference

As elaborated in the introduction, an important yet often neglected aspect of probabilistic forecasts is the epistemic uncertainty, more specifically the PU. Based on general asymptotic theory for time series models (Ling and McAleer [2010]), we derive theoretical PU properties for AT($p$)/s in this section. Corresponding proofs follow directly from theorems in Ling and McAleer [2010] together with common assumptions for time series modeling described in the Supplementary Material. In particular, we assume that time series are strictly stationary and ergodic, a common assumption in (deep) time series models. The other assumptions made in Ling and McAleer [2010] can be directly transferred to our use case since AT($p$)/s and non-linear extensions are fully-parameterized time series models with parameter estimator $\hat{\theta}_T = \text{arg\,min}_\theta  - \ell(\theta)$ based on Maximum-Likelihood estimation (MLE).

Let $\theta^*$ be the true value of $\theta$ and interior point of $\Theta$. We define the following quantities involved in standard asymptotic MLE theory: $\nabla_T(\theta) = \partial \ell_T(\theta)/\partial \theta$, $J_T(\theta) = -\partial^2 \ell_T(\theta)/\partial \theta \partial \theta^T$, $I = \text{E}_G(J_T(\theta^*))$ and $J = \text{E}_G(\nabla_T(\theta^*)\nabla_T(\theta^*))$.

Theorem 1 (Consistency) If $y_0$ is finite and Assumption [4] holds, then $\hat{\theta}_T \xrightarrow{a.s.} \theta^*$ for $T \to \infty$.

As stated in Hothorn et al. (2018), Assumption [1(i)] holds if $a$ is not arbitrarily ill-posed. In practice, both a finite $y_0$ and Assumption [1(i)] are realistic assumptions. Making two additional and also rather weak assumptions (1(ii)-(iii)) allows to derive the asymptotic normal distribution for $\hat{\theta}$.

Theorem 2 (Asymptotic Normality) If $y_0$ is finite and Assumptions [2] hold, then for $T \to \infty$,

$$\hat{\theta} = \theta^* + O(\sqrt{\log \log T}/T)$$

and

$$\sqrt{T}(\hat{\theta} - \theta^*) \xrightarrow{D} \mathcal{N}(0, I^{-1}J(\theta^*)I^{-1}).$$

Based on the same assumptions, a consistent estimator for the covariance can be derived.

Theorem 3 (Consistent Covariance Estimator) For finite $y_0$ and under Assumptions [3]

$$\hat{I}_T = \frac{1}{T} \sum_{t=1}^{T} J_T(\theta_t)$$

and

$$\hat{J}_T = \frac{1}{T} \sum_{t=1}^{T} \nabla_T(\theta_t)\nabla_T^{\top}(\theta_t)$$

are consistent estimators for $I$ and $J$, respectively.
Using the above results, we can derive statistically valid UQ. An example is depicted in Figure 3. Since \( h \) is parameterized through \( \theta \), it is also possible to derive the structural uncertainty of ATMs based on the PU. More specifically, \( h \) can be represented using a linear transformation of \( \theta \), \( h = \mathbf{Y}\theta \), implying the (co-)variance \( \mathbf{Y} \mathbf{I}^{-1}(\theta^\top \mathbf{I}^{-1}) \mathbf{Y}^\top \) for \( h \).

**Practical application** ATM define the distribution \( F_{Y_1|Y_1,...,Y_{t-1},X} \) via \( F_{Y_1|Y_1,...,Y_{t-1},X} = F_X \circ h_t \), where \( h_t \) is parameterized by \( \theta \). In order to assess PU in the estimated density as, e.g. visualized in Figure 1 and 3 we propose to use a parametric Bootstrap described in detail in Supplementary Material [C].

5 RELATED LITERATURE

As outlined in Section 2.2 and 3 NFs and AFs are directly linked to ATMs and motivated by CTMs. Next to flow-based models that focus on a generative task, various approaches have been proposed for CDE [Dinh et al., 2017] Papamakarios et al., 2019 and combined with sequential modeling such as masked autoregressive flows [Papamakarios et al., 2017]. One recent approach combining flows and time series forecasts is Rasul et al. (2021). While their work is motivated for multivariate time series applications, Rasul et al. also show competitive results when using their method for univariate time series forecasting.

Among other approaches for time series forecasting, classical linear or state-space models and neural network-based approaches [Shi et al., 2015; Oord et al., 2016; Li et al., 2017; Wen et al., 2017; Mukherjee et al., 2018; Yu et al., 2019] are most popular [Faloutsos et al., 2018]. One of the most common classical time series approaches is Prophet [Taylor and Letham, 2018], a Bayesian additive model that decomposes the time series into a trend, a seasonal component, additional features effects and an error term. For neural network-based models, a prominent architecture is DeepAR [Salinas et al., 2020], which often outperforms other approaches in probabilistic forecasts. DeepAR is based on a recurrent neural network architecture learned by maximizing a parametric likelihood. An alternative network, DeepFactor, was proposed by Wang et al. (2019) and explores the structures of a large amount of time series by decomposing time series into a global and a local part. Another commonly used approach is DeepState [Rangapuram et al., 2018], a recurrent neural network to learn the time-varying parameters of a state-space model.

Figure 4: Empirical evidence for the correctness of our theoretical results on PU: Expected vs. observed quantiles of the transformation function \( h_t \) (left; one line per dataset) and model parameters \( \theta \) for the different (lagged) transformed outcomes (right; one cross per dataset) based on 1000 simulation replications. The ideal angle bisector is plotted in red.

6 EXPERIMENTS

We will first investigate theoretical properties of ATMs as well as their epistemic uncertainty in simulation studies. We then compare our approach against other state-of-the-art methods described in the previous section on probabilistic forecasting tasks in a benchmark study. Additional results can be found in the Supplementary Material [D].

6.1 Simulation Study

| Dataset | \( T \) | \( p = 1 \) Mean (SD) | \( p = 2 \) Mean (SD) | \( p = 4 \) Mean (SD) |
|---------|-----|-------------------|-------------------|-------------------|
| Oracle  | 400 | 0.33 (0.31)       | 0.22 (0.19)       | 0.25 (0.13)       |
| AT(\(p\)) | 400 | 0.52 (0.46)       | 0.33 (0.3)        | 0.34 (0.23)       |
| Oracle  | 800 | 0.27 (0.34)       | 0.13 (0.12)       | 0.13 (0.085)      |
| AT(\(p\)) | 800 | 0.26 (0.36)       | 0.17 (0.17)       | 0.18 (0.12)       |

**Equivalence and consistency** We first demonstrate Theorem 1 and Proposition 2 in the Supplementary Material, i.e., for growing number of observations \( AT(\(p\)) \) models can recover \( AR(\(p\)) \) models when equally specified. We therefore simulate various AR models using lags \( p = 1, 2, 4 \), \( n = 200, 400, 800 \) and estimate both a classical \( AR(p) \) model and an \( AT(p) \) model for 20 replications. For the latter, we use the mapping derived in Proposition 2 to obtain the estimated AR coefficients from the \( AT(p) \) model. In Table 3 in the Supplementary Material [D] we compare both models based on their estimated coefficients against the
ground truth using the mean squared error (MSE). Results show that the AT(\(p\)) model can empirically recover the AR(\(p\)) model very well.

**Flexibility** Next, we demonstrate how the AT(\(p\)) model with \(M = 30\) can recover a multiplicative autoregressive process. We therefore generate data using an AR model with different lags \(p\) and observations \(n\) as before. This time, however, we provide the AT(\(p\)) model only with the exponentiated data \(\hat{y}_t = \exp(y_t)\). This means the model needs to learn the inverse transformation back to \(y_t\) itself. Despite having to estimate the log-transformation in addition, the AT(\(p\)) model recovers the true model well and, for larger \(n\), is even competitive to the ground truth model (Oracle) that has access to the original non-exponentiated data (cf. Table 1 for an excerpt of the results).

**Epistemic Uncertainty** In this experiment we validate our theoretical results proposed in Section 4.2. As in the previous experiment, we try to learn the log-transformed AR model using an AT(\(p = 3\)) model with coefficients \((0.3, 0.2, 0.1)\). After estimation, we check the empirical distribution of \(\hat{\theta}\) and \(\hat{h}\) against their respective theoretical one in 1000 simulation replications. Figure 4 depicts a quantile-quantile plot of the empirical and theoretical distribution for both \(\hat{h}\) and all 4 parameters (intercept and three lag coefficients). The empirical distributions are well aligned with their theoretical distribution as derived in Section 4.2 confirming our theoretical results.

### 6.2 Benchmarks

Finally, we compare our approach to conditional NPs for (multivariate) time series (MCNF: Rasul et al., 2021) as well as different state-of-the-art forecasting methods previously introduced (DeepAR, DeepFactor, DeepState, Prophet) and an ARIMA baseline. We compare these approaches on commonly used benchmark datasets electricity (elec: Yu et al., 2016), traffic (Yu et al., 2016), monthly tourism (Athanasopoulous et al., 2011), the hourly m4 dataset (Makridakis et al., 2018) and stock exchange (Lai et al., 2018). A short summary of these datasets can be found in Table 5 in the Supplementary Material. For each neural network-based algorithm a grid-search is used on a rolling window to find the optimal set of hyperparameters. For electricity and traffic we use both the 24 hours and 72 hours forecast horizon. For m4 and tourism the test sets are already pre-defined with 48 hours and 24 months forecast windows, respectively. For each proposed method and dataset, we report continuous ranked probability scores (CRPS; Gneiting et al., 2007; Jordan et al., 2019) and average results across time series and time points. Our model uses only linear effects of the day, hour, month and/or household for \(\theta\). These effects change higher moments of the distribution in an additive manner allowing to relate individual influences of features to the outcome distribution. The (transformed) lags of the outcome only change the distribution’s location. Further details can be found in the Supplementary Material D.

**Results** Table 2 shows the results of the comparison. Interpretability naturally comes at the cost of decreased prediction performance. DeepAR outperforms all models in most cases. However, our approach often yields competitive and consistently good results while its inner workings are straightforward to understand.

### 7 CONCLUSION AND OUTLOOK

We have proposed ATMs, a flexible and comprehensible model class combining and extending various existing modeling approaches. ATMs allow for expressive probabilistic forecasts using a base distribution and a single transformation modeled by Bernstein polynomials. Additionally, a parametric inference paradigm based on MLE allows for epistemic UQ. ATMs can be based on interpretable additive predictors or deep neural networks, empirically and theoretically recover well-known models and demonstrate competitive per-
formance on real-world datasets.

ATMs are the first adaption of (deep) transformation models to time series applications. Although our approach can be easily extended to incorporate deep architectures, PU derivations no longer hold (e.g., because uniqueness of $\theta^*$ cannot be guaranteed). The derived results are still valuable, as they will help derivations of PU for more complex models in the future and further advance the presented methods.

References

Alexandrov, A., Benidis, K., Bohlke-Schneider, M., Flunkert, V., Gasthaus, J., Januschowski, T., Mad-dix, D. C., Rangapuram, S., Salinas, D., Schulz, J., Stella, L., Türkmen, A. C., and Wang, Y. (2020). GhuonTS: Probabilistic and Neural Time Series Modeling in Python. Journal of Machine Learning Research, 21(116):1–6.

Athanasiopoulos, G., Hyndman, R. J., Song, H., and Wu, D. C. (2011). The tourism forecasting competition. International Journal of Forecasting, 27(3):822–844.

Baumann, P. F. M., Hothorn, T., and Rügamer, D. (2021). Deep Conditional Transformation Models. In Machine Learning and Knowledge Discovery in Databases. Research Track, pages 3–18, Cham. Springer International Publishing.

Bender, A., Rügamer, D., Scheipl, F., and Bischl, B. (2021). A general machine learning framework for survival analysis. In Hutter, F., Kersting, K., Lijffijt, J., and Valera, I., editors, Machine Learning and Knowledge Discovery in Databases, pages 158–173, Cham. Springer International Publishing.

Bengio, Y. and Bengio, S. (1999). Modeling high-dimensional discrete data with multi-layer neural networks. NIPS’99, page 400–406. MIT Press.

Bernstein, S. (1912). Démonstration du théorème de weierstrass fondée sur le calcul des probabilités. 13(1):1–2.

Bishop, C. M. (1994). Mixture density networks.

Blundell, C., Cornebise, J., Kavukcuoglu, K., and Wierstra, D. (2015). Weight uncertainty in neural networks. Proceedings of the 32nd International Conference on Machine Learning, 37:1613–1622.

Chernozhukov, V., Fernández-Val, I., and Melly, B. (2013). Inference on counterfactual distributions. Econometrica, 81(6):2205–2268.

Cox, D. R. (1972). Regression models and life-tables. Journal of the Royal Statistical Society: Series B (Methodological), 34(2):187–202.

Dinh, L., Krueger, D., and Bengio, Y. (2015). Nice: Non-linear independent components estimation.

Dinh, L., Sohl-Dickstein, J., and Bengio, S. (2017). Density estimation using real nvp.

Dunson, D. B. (2010). Nonparametric Bayes applications to biostatistics. Bayesian nonparametrics, 28:223–273.

Dutordoir, V., Salimbeni, H., Deisenroth, M. P., and Hensman, J. (2018). Gaussian process conditional density estimation. In Proceedings of the 32nd International Conference on Neural Information Processing Systems, NIPS’18, page 2391–2401. Curran Associates Inc.

Faloutsos, C., Gasthaus, J., Januschowski, T., and Wang, Y. (2018). Forecasting big time series: Old and new. Proceedings of the VLDB Endowment, 11(12).

Farouki, R. T. (2012). The Bernstein polynomial basis: A centennial retrospective. Computer Aided Geometric Design, 29(6):379–419.

Foresi, S. and Peracchi, F. (1995). The conditional distribution of excess returns: An empirical analysis. Journal of the American Statistical Association, 90(430):451–466.

Gneiting, T., Balabdaoui, F., and Raftery, A. E. (2007). Probabilistic forecasts, calibration and sharpness. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 69(2):243–268.

Hothorn, T. (2020). Transformation boosting machines. Statistics and Computing, 30(1):141–152.

Hothorn, T., Kneib, T., and Bühlmann, P. (2014). Conditional transformation models. Journal of the Royal Statistical Society: Series B: Statistical Methodology, pages 3–27.

Hothorn, T., Möst, L., and Bühlmann, P. (2018). Most likely transformations. Scandinavian Journal of Statistics, 45(1):110–134.

Hüllermeier, E. and Waegeman, W. (2021). Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods. Machine Learning, 110(3):457–506.

Hyndman, R., Athanasopoulos, G., Bergmeir, C., Caceres, G., Chhay, L., O’Hara-Wild, M., Petropoulos, F., Razbash, S., Wang, E., and Yasmeen, F. (2021). forecast: Forecasting functions for time series and linear models. R package version 8.15.

Jordan, A. et al. (2002). On discriminative vs. generative classifiers: A comparison of logistic regression and naive Bayes. Advances in neural information processing systems, 14(2002):841.

Jordan, A., Krüger, F., and Lerch, S. (2019). Evaluating probabilistic forecasts with scoring rules. Journal of Statistical Software, Articles, 90(12):1–37.
Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

Kingma, D. P., Salimans, T., Jozefowicz, R., Chen, X., Sutskever, I., and Welling, M. (2016). Improved variational inference with inverse autoregressive flow. In Lee, D., Sugiyama, M., Luxburg, U., Guyon, I., and Garnett, R., editors, Advances in Neural Information Processing Systems, volume 29. Curran Associates, Inc.

Koenker, R. (2005). Quantile Regression, volume Economic Society Monographs. Cambridge University Press.

Kook, L., Herzog, L., Hothorn, T., Dürr, O., and Sick, B. (2021). Deep and interpretable regression models for ordinal outcomes. Pattern Recognition.

Lai, G., Chang, W.-C., Yang, Y., and Liu, H. (2018). Modeling long-and short-term temporal patterns with deep neural networks. In The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, pages 95–104.

Lázaro-Gredilla, M. and Titsias, M. K. (2011). Variational heteroscedastic gaussian process regression. In International Conference on Machine Learning, pages 841–848. PMLR.

Li, Y., Yu, R., Shahabi, C., and Liu, Y. (2017). Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. arXiv preprint arXiv:1707.01926.

Lin, T., Horne, B. G., Tino, P., and Giles, C. L. (1996). Learning long-term dependencies in narn recurrent neural networks. IEEE Transactions on Neural Networks, 7(6):1329–1338.

Ling, S. and McAleer, M. (2010). A general asymptotic theory for time-series models. Statistica Neerlandica, 64(1):97–111.

Liu, J., Paisley, J., Kioumourtzoglou, M.-A., and Coull, B. (2019). Accurate uncertainty estimation and decomposition in ensemble learning. In Wallach, H., Larochelle, H., Beygelzimer, A., d’Alché-Buc, F., Fox, E., and Garnett, R., editors, Advances in Neural Information Processing Systems, volume 32. Curran Associates, Inc.

Makridakis, S., Spiliotis, E., and Assimakopoulos, V. (2018). The m4 competition: Results, findings, conclusion and way forward. International Journal of Forecasting, 34(4):802–808.

Mukherjee, S., Shankar, D., Ghosh, A., Tathawadekar, N., Kompalli, P., Sarawagi, S., and Chaudhury, K. (2018). Armdn: Associative and recurrent mixture density networks for retail demand forecasting. arXiv preprint arXiv:1803.03800.

Murphy, K. P. (2012). Machine learning: a probabilistic perspective. MIT press.

Oord, A. v. d., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., and Kavukcuoglu, K. (2016). WaveNet: A generative model for raw audio. arXiv preprint arXiv:1609.03499.

Papamakarios, G., Nalisnick, E., Rezende, D. J., Mohamed, S., and Lakshminarayanan, B. (2019). Normalizing flows for probabilistic modeling and inference.

Papamakarios, G., Pavlakou, T., and Murray, I. (2017). Masked autoregressive flow for density estimation. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, Advances in Neural Information Processing Systems, volume 30. Curran Associates, Inc.

Rangapuram, S. S., Seeger, M. W., Gasthaus, J., Stella, L., Wang, Y., and Januschowski, T. (2018). Deep state space models for time series forecasting. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R., editors, Advances in Neural Information Processing Systems, volume 31. Curran Associates, Inc.

Rasul, K., Sheikh, A.-S., Schuster, I., Bergmann, U., and Vollgraf, R. (2021). Multivariate Probabilistic Time Series Forecasting via Conditioned Normalizing Flows. In International Conference on Learning Representations 2021.

Rothfuss, J., Ferreira, F., Walther, S., and Ulrich, M. (2019). Conditional density estimation with neural networks: Best practices and benchmarks. arXiv preprint arXiv:1903.00954.

Rügamer, D., Kolb, C., Fritz, C., Pfisterer, F., Bischl, B., Shen, R., Bukas, C., de Andrade e Sousa, L. B., Thalmeier, D., Baumann, P., Klein, N., and Müller, C. L. (2021). deepregression: a Flexible Neural Network Framework for Semi-Structured Deep Distributional Regression.

Rügamer, D., Kolb, C., and Klein, N. (2020). Semi-Structured Deep Distributional Regression: A Combination of Additive Models and Deep Learning. arXiv preprint arXiv:2002.05777.

Rügamer, D., Pfisterer, F., and Bischl, B. (2020). Neural mixture distributional regression.
Sakia, R. M. (1992). The box-cox transformation technique: a review. *Journal of the Royal Statistical Society: Series D (The Statistician)*, 41(2):169–178.

Salinas, D., Flunkert, V., Gasthaus, J., and Januschowski, T. (2020). Deepar: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting*, 36(3):1181–1191.

Shi, X., Chen, Z., Wang, H., Yeung, D.-Y., Wong, W.-K., and Woo, W.-c. (2015). Convolutional lstm network: A machine learning approach for precipitation nowcasting. *Advances in neural information processing systems*, 28.

Shumway, R. H., Stoffer, D. S., and Stoffer, D. S. (2000). *Time series analysis and its applications*, volume 3. Springer.

Sick, B., Hothorn, T., and Dürr, O. (2021). Deep transformation models: Tackling complex regression problems with neural network based transformation models. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 2476–2481. IEEE.

Taylor, S. and Letham, B. (2021). *prophet: Automatic Forecasting Procedure*. R package version 1.0.

Taylor, S. J. and Letham, B. (2018). Forecasting at scale. *The American Statistician*, 72(1):37–45.

Uria, B., Côté, M.-A., Gregor, K., Murray, I., and Larochelle, H. (2016). Neural autoregressive distribution estimation. *Journal of Machine Learning Research*, 17(205):1–37.

Van Belle, V., Pelckmans, K., Suykens, J. A., and Van Huffel, S. (2011). Learning transformation models for ranking and survival analysis. *Journal of machine learning research*, 12(3).

Wang, Y., Smola, A., Maddix, D., Gasthaus, J., Foster, D., and Januschowski, T. (2019). Deep factors for forecasting. In Chaudhuri, K. and Salakhutdinov, R., editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 6607–6617. PMLR.

Wen, R., Torkkola, K., Narayanaswamy, B., and Madeka, D. (2017). A multi-horizon quantile recurrent forecaster. *arXiv preprint arXiv:1711.11053*.

Wong, C. S. and Li, W. K. (2000). On a mixture autoregressive model. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 62(1):95–115.

Wu, C. O. and Tian, X. (2013). Nonparametric estimation of conditional distributions and rank-tracking probabilities with time-varying transformation models in longitudinal studies. *Journal of the American Statistical Association*, 108(503):971–982.
A RELATED LITERATURE

Conditional models describe the conditional distribution $F_Y \mid x$ of an outcome $Y$ conditional on observed features $x$ (see, e.g., Jordan et al., 2002). Instead of modeling the complete distribution of $Y \mid x$, many approaches focus on modeling a single characteristic of this conditional distribution. Predictive models, for example, often focus on predicting the average outcome value, i.e., the mean of the conditional distribution. Quantile regression (Koenker, 2005) can model a certain quantile of $Y \mid x$ and is thus more flexible in explaining the conditional distribution and its aleatoric uncertainty – the non-deterministic nature of the input-output relationship (Hüllermeier and Waegeman, 2021). Various approaches in machine and deep learning allow for an even richer explanation by, e.g., directly modeling the distribution’s density $f_Y \mid x$ and thus the whole distribution $F_Y \mid x$. Examples include mixture density networks (Bishop, 1994), survival models (Bender et al., 2021) or, in general, probabilistic approaches (Chernozhukov et al., 2013; Foresi and Peracchi, 1995; Rügamer et al., 2020; Wu and Tian, 2013). These approaches can also be regarded as conditional density estimation (CDE) models. DR or CDE is a challenging task that requires balancing the representational capacity of the model and its risk for overfitting. We refer to Rothfuss et al. (2019) for a recent work of best practices in CDE.

B DEFINITIONS, ASSUMPTIONS, THEORETICAL ANALYSIS

The following definition of the error distribution follows Hothorn et al. (2018).

Definition 3 Error Distributions  Let $Z : \Omega \rightarrow \mathbb{R}$ be a $\mathcal{U} - \mathcal{B}$ measurable function from $(\Omega, \mathcal{U})$ to the Euclidian space with Borel $\sigma$-algebra $\mathcal{B}$ with absolutely continuous distribution $P_Z = F_Z \circ \mu_L$ on the probability space $(\mathbb{R}, \mathcal{B}, P_Z)$ and $\mu_L$ the Lebesque measure. We define $F_Z$ and $F_Z^{-1}$ as the corresponding distributions and assume $F_Z(-\infty) = 0$, $F_Z(\infty) = 1$. $0 < f_Z(z) < \infty \forall z \in \mathbb{R}$ with log-concave, twice-differentiable density $f_Z$ with bounded first and second derivatives.

Proposition 1 (Interpretation of $h$) The ATM as defined in (8) and further specified in (11) can be seen as an additive regression model with outcome $h_{1t}(y_t)$, predictor $h_{2t}((h_{1t} \circ \mathcal{Y}_1|F_{t-1}, x)|x)$ and error term $\varepsilon \sim F_Z$.

Proof We first define an additive regression model with outcome $\lambda_1 := h_{1t}(y_t)$, predictor $\hat{\lambda}_2 := -h_{2t}((h_{1t} \circ \mathcal{Y}_1|F_{t-1}, x)|x)$ and error term $\varepsilon \sim F_Z$, i.e.,

$$\lambda_1 = \hat{\lambda}_2 + \varepsilon, \varepsilon \sim F_Z,$$

where we use $\hat{\lambda}_2$ instead of $\lambda_2$ for convenience without limitation of generality. This implies that $\lambda_1 - \hat{\lambda}_2 = \lambda_1 + \lambda_2 = \varepsilon$ or equally $\lambda_1 + \lambda_2 \sim F_Z$. Optimizing this model is equal to fitting an ATM as defined in (8) with structural assumption as defined in (11).

Proposition 2 (Equivalence of AR(p) and AT(p) models) An autoregressive model of order $p$ (AR(p)) with independent white noise following the distribution $F_Z$ in the location-scale family is equivalent to an AT(p) model for $M = 1$, $\theta(x) \equiv \theta$, $r(x) \equiv 0$ and error distribution $F_Z$.

Proof The transformation function of an AT(p) model with BSPs of order $M$ defined on an interval $[t_l, t_u]$, $\theta(x) \equiv \theta$ and $r(x) \equiv 0$ is given by

$$h_{1t} + h_{2t} = a(y_t)^\top \theta + \sum_{j=1}^{p} \phi_j a(y_{t-j})^\top \theta.$$

We can further simplify the model by making $a(y_t)$ more explicit:

$$a(y_t) = (M + 1)^{-1}(f_{BE(1, M+1)}(\tilde{y}_t), \ldots, f_{BE(M, M-M+1)}(\tilde{y}_t), \ldots, f_{BE(M+1, 1)}(\tilde{y}_t)) \in \mathbb{R}^{1 \times (M+1)}$$

with $\tilde{y}_t = (y - t_l)/(t_u - t_l)$ and Beta distribution density $f_{BE(\kappa, \mu)}$ with parameters $\kappa, \mu$. For simplicity and w.l.o.g. assume that $y_t \equiv \tilde{y}_t$. Setting $M$ to 1, we get

$$h_{1t} = (\vartheta_0 f_{BE(1, 2)} + \vartheta_1 f_{BE(2, 1)})/2 = \vartheta_0 (1 - y_t) + \vartheta_1 y_t = \vartheta_0 + (\vartheta_1 - \vartheta_0) y_t = \vartheta_0 + \tilde{\vartheta}_1 y_t.$$
The transformation of the AT(p) model is thus given by

\[ h_t(y_t|F_{t-1}, x) = \tilde{\vartheta}_0 + \tilde{\vartheta}_1 y_t + \tilde{\varphi}_j \vartheta_0 + \vartheta_1 y_{t-j} \]  

(12)

with \( \tilde{\vartheta}_0 = (\vartheta_0 (1 + \sum \varphi_j))/\tilde{\varphi}_1 \) and \( \tilde{\varphi}_j = \varphi_j \vartheta_1/\tilde{\vartheta}_1 \). From (8) we know

\[ P(Y_t \leq y_t|F_{t-1}, x) = F_Z(h_t(y_t|F_{t-1}, x)). \]  

(13)

The AR(p) model is given by

\[ y_t = \varphi_0 + \sum_{j=1}^{p} \varphi_j y_{t-j} + \sigma \varepsilon_t, \varepsilon_t \sim F_Z \]  

\[ \Leftrightarrow Z = y_t - \varphi_0 - \sum_{j=1}^{p} \varphi_j y_{t-j} \sim F_Z. \]  

(14)

The equivalence of (13) in combination (12) with (14) is then given when setting \( \tilde{\vartheta}_0 = -\varphi_0, \tilde{\varphi}_j = -\varphi_j \forall j \in \{1, \ldots, p\} \) and \( \sigma = \tilde{\vartheta}_1^{-1} \). Since both models find their parameters using Maximum Likelihood and it holds \( \tilde{\vartheta}_1 > 0 \) (as required for \( \sigma \)) by the monotonicity restriction on the BSPs coefficient, the models are identical up to different parameterization.

**Assumption 1**

Assumptions following Ling and McAleer (2010):

(i) \( \mathbb{E}_G \{ \sup_{\theta \in \Theta} [\ell_T(\theta)] \} < \infty \) and \( \theta^* \) is unique.

(ii) \( \nabla_T(\theta^*) \) is a martingale difference w.r.t. \( F_{T-1} \) with \( 0 < J < \infty \).

(iii) \( I \) is positive-definite and \( \mathbb{E}_G \{ \sup_{\theta \in \Theta} ||\theta - \theta^*|| < \xi \} < \infty \) for some \( \xi > 0 \)

**C PARAMETRIC UNCERTAINTY AND PRACTICAL APPLICATION**

To assess the PU included in the estimated density, we propose to use a parametric Bootstrap (similar to the one suggested in Hothorn et al., 2018) that is based on the following steps:

1. Generate \( \hat{\theta}^{(v)}, v = 1, \ldots, N \) from the limiting distribution (Theorem 2 and 3);
2. Draw samples \( Z_i \sim F_Z, i \in T \) and calculate \( Y_{i,v} = \inf\{y \in \Xi | h_i(y, \hat{\theta}^{(v)}) \geq Z_i \} \);
3. Refit the model for each data set \( \{Y_{i,v} \}_{i \in T}, v = 1, \ldots, N \);
4. Calculate the \( N \) model densities.

Based on these \( N \) model densities, uncertainty in the originally estimated density can be analyzed, e.g., visually by plotting all densities together as done in Figure 1 and 3.

**D EXPERIMENTAL SETUP**

**D.1 Simulations**

In this subsection, we describe the details of the data generating process used in Figure 1 (Section D.1.1) and provide results on experiments for the equivalence and consistency paragraph of Section 6.1 in Section D.1.2.

**D.1.1 Data Generating Process Toy Example**

For Figure 1 we simulate \( T = 1000 \) time points \( y_1, \ldots, y_T \) that exhibit two modes as follows:

1. Set \( y_0 = 0 \);
2. Define a shift \( \varrho = 2 \) and sample \( x_1, \ldots, x_T \) from \((-\varrho, \varrho)\) with equal probability;
Table 3: Average MSE in percent (with standard deviation in brackets) of estimated coefficients by the AR($p$) and AT($p$) model (rows) for different simulation settings (columns) over 100 replications.

|          | $p = 1$       | $p = 2$       | $p = 5$       |
|----------|---------------|---------------|---------------|
| $T = 200$ | AR($p$) 0.54 (0.73) | AR($p$) 0.73 (1) | AR($p$) 0.68 (0.6) |
|          | AT($p$) 0.12 (0.16) | AT($p$) 0.17 (0.25) | AT($p$) 0.15 (0.16) |
| $T = 1000$ | AR($p$) 0.019 (0.03) | AR($p$) 0.06 (0.09) | AR($p$) 0.05 (0.05) |
| $T = 5000$ | AR($p$) 0.12 (0.16) | AR($p$) 0.17 (0.25) | AR($p$) 0.15 (0.16) |

3. Define a autoregressive coefficient $\phi_1 = 0.1$

4. For $t = 1, \ldots, T$, sample $y_t \sim N(\phi_1 y_{t-1} + x_t, 1)$

When providing the model with the marginal distribution of $y_t$ and defining $x_t$ as latent, unobserved variable, $y_t$ will exhibit two modes centered around $\pm \rho$.

D.1.2 AR($p$) comparison

The data generating process for the simulation of Section 6.1 is an AR model with the $p$ first coefficients 0.4, 0.2, 0.1, 0.05, 0.025. A standard implementation for the AR model was used. For the AT model we use the implementation provided in [Rügamer et al., 2021] using 2500 epochs, batch size of 50, and early stopping based on 10% of the training data.

Table 4: Mean and standard deviation (brackets) of the mean squared error ($\times 10^2$ for better readability) between estimated and true coefficients in an AR($p$) model using our approach on the tampered data (bottom row) and the corresponding oracle based on the true data (Oracle).

|          | $p = 1$       | $p = 2$       | $p = 4$       |
|----------|---------------|---------------|---------------|
| $T = 200$ | Oracle 0.65 (0.84) | Oracle 0.49 (0.62) | Oracle 0.65 (0.45) |
|          | AR($p$) 0.12 (0.16) | AR($p$) 0.22 (0.19) | AR($p$) 0.23 (0.23) |
| $T = 400$ | Oracle 0.33 (0.31) | Oracle 0.33 (0.3) | Oracle 0.34 (0.23) |
|          | AR($p$) 0.27 (0.34) | AR($p$) 0.17 (0.17) | AR($p$) 0.18 (0.12) |

D.2 Details on the benchmark study

D.2.1 Datasets

Table 5 summarizes the characteristics of the data sets used.

|          | electricity | exchange | traffic | tourism | m4 |
|----------|-------------|----------|---------|---------|----|
| # time series | 370         | 8        | 963     | 366     | 414 |
| frequency   | hourly      | daily    | hourly  | monthly | hourly |
| forecast horizon | 24/72      | 1219     | 24/72   | 24      | 48  |
| # training samples | 97065840 | 6094     | 10446624 | 24522   | 289800 |
| last day in training set | 2014-12-28 | 2008-09-16 | 2009-03-27 | varying | varying |

D.2.2 Software

We used MxNet and the gluon-ts [Alexandrov et al., 2020] implementation of DeepAR, DeepState and DeepFactor for our comparison as well as the conditional flow implementation given in pytorch-ts [Rasul et al., 2021]. For ATMs we extended the software deepregression [Rügamer et al., 2021] by including an additional additive component for lags and used optimization techniques considered in [Rügamer et al., 2020].
For ARIMA, we use the `forecast` R package (Hyndman et al., 2021) and for Prophet the `prophet` R package (Taylor and Letham, 2021).

### D.2.3 Hyperparameter Setup

For DeepAR, DeepState and DeepFactor we tuned batch size, context length and epochs on a grid in line with the recommendations (see, e.g., the DeepAR Documentation) for each forecasting horizon and dataset. For ATMs we used the same lags as for DeepAR and applied a linear or neural network on top of the transformed lags as $h_{2t}$. We performed a grid search over other additive predictor components in $h_{1t}$ and $h_{2t}$ with possible options: 1) intercept only; 2) linear effect for the time series dimension indicator (i.e., the household); 3) linear hour effect and 2; 4) linear day effect and 2; 5) linear day and 3; 6) linear indicator-day effect; 7) linear indicator-hour effect; 8) linear indicator-month effect. We further searched over the number of BSPs $M \in \{10, 20, 30\}$. Network training was done using 50 epochs with early stopping and a batch size of 64 or 128. For MCNF we used the an LSTM as recurrent neural network, trained the network with batch size 64 for 1, 20 or 40 epochs as suggested in Rasul et al. (2021) and further compared the performance of 1, 3 or 5 RealNVP flows. For ARIMA we used the `auto.arima` implementation (Hyndman et al., 2021) and performed a stepwise search via the aicc with different starting values for the order of the AR and the MA term. For the AR term possible parameter values were 6, 24, 72 for hourly and monthly data when a 24 hour, 72 hour or 24 month forecast had to be performed, and 6 and 24 when a 48h forecast was performed. The search space for the MA term started either with 0 or 3. We chose the ARIMA model with the lowest aicc on the validation set. For Prophet (Taylor and Letham, 2021), we tuned the parameter modulating the flexibility of the automatic changepoint selection (0.001, 0.05, 0.5), the parameter modulating the strength of the seasonality model (0.01, 0.5, 10) and the number of Fourier components (3, 5) on a grid in line with the recommendations (see, e.g., the Prophet Documentation). As competitor to Prophet’s automatic seasonality detection, we manually added seasonal components in the additive predictor where the corresponding periodicity was determined based on the estimated spectral density for the frequency domain of the observed time series. For the exchange data, we did not tune the ARIMA and the Prophet model due to the small sample size compared to the models’ complexity.

For the exchange data, the ARIMA model for the test data was found by choosing the model with the lowest in-sample aicc after starting with AR order 0 or 3 and with MA order 0 or 3. For the Prophet model, we took the default values.

For all models, we evaluate the CRPS for each hyperparameter set on a separate validation set which has the same size as the corresponding test set. The set of hyperparameter with the lowest CRPS on the validation set for each forecasting window we evaluate the (same) final model multiple times (over each window) and average results.

### D.2.4 Computational Setup

- All models were run on a server with 90GB RAM, 20 vCPUs from type Intel Xeon Processor (Skylake, IBRS), and a server with 64GB RAM, 32 vCPUs from type Intel(R) Xeon(R) CPU E5-2650 v2 @ 2.60GHz;
- DeepAR, DeepState and DeepFactor were additionally run on a high-performance cluster with Intel(R) Xeon(R) CPU E3-1284L v4 @ 2.90GHz cores.