Journal of Machine Learning Research 21 (2020) 1-6 Submitted 9/19; Published 4/20

GluonTS: Probabilistic and Neural Time Series Modeling in Python

Alexander Alexandrov ALXALE@AMAZON.COM
Konstantinos Benidis KBENIDIS@AMAZON.COM
Michael Bohlke-Schneider BOHLKEM@AMAZON.COM
Valentin Flunkert FLUNKERT@AMAZON.COM
Jan Gasthaus GASTHAUS@AMAZON.COM
Tim Januschowski TJNSCH@AMAZON.COM
Danielle C. Maddix DMMADDIX@AMAZON.COM
Syama Rangapuram RANGAPUR@AMAZON.COM
David Salinas DSALINA@AMAZON.COM
Jasper Schulz SCHJASPE@AMAZON.COM
Lorenzo Stella STELLALO@AMAZON.COM
Ali Caner Türkmen ATTURKM@AMAZON.COM
Yuyang Wang YUYAWANG@AMAZON.COM
Amazon Research
Charlottenstrasse 4, 10969, Berlin, Germany
1900 University Ave., East Palo Alto, CA 94303, US

Editor: Balazs Kegl

Abstract

We introduce the Gluon Time Series Toolkit (GluonTS), a Python library for deep learning based time series modeling for ubiquitous tasks, such as forecasting and anomaly detection. GluonTS simplifies the time series modeling pipeline by providing the necessary components and tools for quick model development, efficient experimentation and evaluation. In addition, it contains reference implementations of state-of-the-art time series models that enable simple benchmarking of new algorithms.

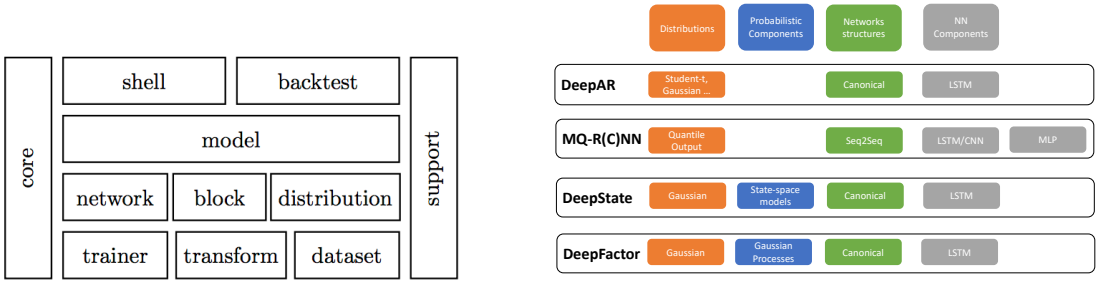
Keywords: time series, deep learning, Python, scientific toolkit, benchmarking

1. Introduction

Traditionally, time series modeling has focused on individual time series via local models, where free parameters are estimated for each time series separately. A number of commercial and open-source toolkits exist for these local models (Hyndman and Khandakar, 2008; Taylor and Letham, 2018; Lning et al., 2019). We refer to (Januschowski et al., 2019) for a survey of open-source forecasting packages in Python. In recent years, advances in deep learning have led to accuracy improvements over the local approach by utilizing the large amounts of available data for estimating parameters of a single global model over an entire collection of time series (Flunkert et al., 2019; Wen et al., 2017). Deep learning frameworks (Chen et al., 2015; Paszke et al., 2017; Abadi et al., 2016) are growing in popularity, and have led to the emergence of more application-specific toolkits (Hieber et al., 2018; Dai et al., 2018; Bingham et al., 2018). To the best of our knowledge, no dedicated deep learning toolkit for

time series modeling currently exists. We fill this gap with GluonTS[1], a deep learning based library based on the *Gluon* API[2] of the MXNet deep learning framework.

GluonTS bundles components such as neural network architectures for sequences, feature processing steps, and experimentation and evaluation mechanisms. These components can be used to quickly assemble, train, and evaluate new models for time series applications such as forecasting and anomaly detection. In addition, GluonTS includes a number of pre-built state-of-the-art deep learning based models and probabilistic models and components, including state-space models and Gaussian processes (Girard et al., 2003), for direct use or benchmarking of new algorithms.



(a) Diagram of packages.

(b) DNN models assembled via components.

Figure 1: Packages available in GluonTS and how their components are used to form the pre-built DNN models.

2. Library design and components

The main design principles in GluonTS are modularity, scalability, and reproducibility. The code is modular, since it is decoupled into small components with clear interfaces that enable users to build models by combining and/or extending these components in new ways. All components scale from small to large data by using Python iterators to enable streaming in the data-processing APIs, so that the datasets do not need to be fully loaded in memory. Experiments are reproducible as all configuration details, such as parameter values, can be logged, and the experiment can be re-created.

Figure 1a illustrates the overall design of GluonTS. The core and support packages provide cross-cutting functionalities, e.g., object serialization and hyperparameter validation. The dataset package contains utilities for reading and writing datasets, calculating dataset statistics, and a dataset repository abstraction. The transform package includes components for constructing feature processing pipelines, such as feature generation, splitting and padding of time series, and marking of special points in time or missing values. A user can easily include custom transformations for specific purposes, and combine them with existing transformations in a pipeline. The trainer package inherits from the Gluon Trainer class, and performs the optimization.

1. `https://github.com/awslabs/gluon-ts`
2. `https://mxnet.incubator.apache.org`

The distribution package provides a flexible abstraction for probability distributions, which are common building blocks in probabilistic time series modeling. Components include common parametric distributions, such as Gaussian, Student-t, gamma, and negative binomial, as well as non-parametric splines. The network and block packages, which define network architectures and reusable Gluon blocks, respectively, provide reusable components for the construction of deep neural network (DNN) models.

The model package implements a minimalistic Estimator interface, which features a train method that returns a Predictor, as in scikit-learn (Pedregosa et al., 2011). Trained models (Predictors) return Forecast objects as predictions, which contain a time index and a representation of the probability distribution of values over that index. The backtest package splits all time series in the dataset at a certain point in time, and uses the first part for training the model and the second part for evaluating the accuracy. Forecast objects have a common interface that allows the evaluation component to compute accuracy metrics, such as quantile losses. To qualitatively assess the model accuracy, GluonTS contains visualization methods that use matplotlib (Hunter, 2007).

While GluonTS can be run directly on a local machine, the shell package enables training and prediction that can be scaled up through integration with Amazon SageMaker, the managed machine learning service offered by Amazon Web Services.

3. Time series modeling and benchmarking

GluonTS enables probabilistic modeling of (large) collections of time series. More formally, let $Z = \{z_{i,1:T_i}\}_{i=1}^{N}$ be a set of N univariate time series, where $z_{i,1:T_i} := (z_{i,1}, z_{i,2}, \ldots, z_{i,T_i})$, and $z_{i,t} \in \mathbb{R}$ denotes the value of the i-th time series at time t. Let $X = \{\mathbf{x}_{i,1:T_i+\tau}\}_{i=1}^{N}$ be a set of associated, time-varying covariate vectors with $\mathbf{x}_{i,t} \in \mathbb{R}^D$. Given a probabilistic model, the goal of forecasting is to predict the probability distribution $p\left(z_{i,T_i+1:T_i+\tau} \mid z_{i,1:T_i}, \mathbf{x}_{i,1:T_i+\tau}; \Phi\right)$ of future values $z_{i,T_{i+1}:T_{i+\tau}}$, with $\tau > 0$, given the past values $z_{i,1:T_i}$, the covariates $\mathbf{x}_{i,1:T_{i+\tau}}$, and the model parameters Φ. In anomaly detection, the goal is to score the likelihood of a point subject to the estimated model, i.e., the lower the likelihood of a point, the more abnormal it is.

GluonTS provides a wide variety of pre-built neural network based models. Recently, Benidis et al. (2020) provided a literature overview of such models. Figure 1b shows how these models are comprised of various GluonTS components. DeepAR uses a recurrent neural network (RNN) with LSTM or GRU cells, and estimates parameters of a parametric distribution or uses a parameterization of the quantile function (Flunkert et al., 2019; Gasthaus et al., 2019). MQ-RNN and MQ-CNN combine RNN and dilatied causal convolution (CNN) encoders, respectively, with a quantile decoder (Wen et al., 2017), using the flexible sequence-to-sequence framework provided in GluonTS. Deep State (Rangapuram et al., 2018) parameterizes a linear state-space model, implemented via a Kalman filter, using an RNN whose weights are learned jointly across all time series (Rangapuram et al., 2018). Deep Factor combines a local method with a global neural network, so that the weights can be learned across all time series (Wang et al., 2019). GluonTS also provides implementations of Transformer (Vaswani et al., 2017) and Wavenet (van den Oord et al., 2016) architectures, that have been successful in natural language processing, adjusted to the time series domain.

	electricity	exchange rate	m4-daily	m4-hourly	m4-monthly	m4-quarterly	m4-weekly	m4-yearly	solar energy	traffic
DeepAR	**0.050**	0.023	0.025	**0.033**	0.115	0.087	**0.048**	0.128	**0.398**	**0.126**
CNN-QR	0.083	0.016	0.027	0.065	0.124	0.089	0.059	**0.122**	0.551	0.272
Transformer	0.066	0.009	0.027	0.035	0.136	0.105	0.083	0.160	0.432	0.132
Auto-ETS	0.121	**0.008**	**0.023**	0.043	**0.099**	**0.079**	0.051	0.126	1.778	0.373
Auto-ARIMA	-	**0.008**	0.024	0.040	-	0.080	0.050	0.124	1.153	-
Seasonal Naive	0.070	0.011	0.028	0.048	0.146	0.119	0.063	0.161	1.000	0.251

Table 1: Mean quantile loss of some of the pre-built models: these are grouped into neural network based global, classical local, and naive benchmark categories. The dash indicates non-termination in 24 hours.

In addition to the global neural network based models, GluonTS provides classical local models, such as ARIMA and ETS (Hyndman and Khandakar, 2008), Prophet (Taylor and Letham, 2018), Gaussian processes with exact inference, radial basis function (RBF) and periodic kernels, and linear dynamical systems (LDS). Additional local naive models are included to serve as benchmarks, such as the seasonal naive, which generates forecasts as the exact values of the previous season.

Table 1 displays the mean quantile loss of various pre-built models in GluonTS on 10 open-source datasets: hourly electricity consumption of 370 customers (Dheeru and Karra Taniskidou, 2017), daily exchange rate between 8 currencies used in (Lai et al., 2017), 6 datasets from the M4 competition (Makridakis et al., 2018), hourly photo-voltaic production of 137 stations in Alabama State used in (Lai et al., 2017), and hourly occupancy rate between 0 and 1 of 963 car lanes of San Francisco bay area freeways (Dheeru and Karra Taniskidou, 2017). The table is not intended to serve as an exhaustive comparison between the models, therefore the hyperparameters the hyperparameters of each method are fixed across all datasets and the training is limited to 5000 gradient updates. Table 1 highlights the wide range of available models that can serve as low effort modeling solutions, and the ease provided by GluonTS of benchmarking a new model against existing state-of-the-art ones on a variety of datasets.

4. Conclusion

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

GluonTS's pre-bundled implementations of state-of-the-art models allow easy benchmarking of new algorithms. We demonstrate this in a large scale experiment of running pre-bundled models on different, publicly available datasets, and comparing their accuracy with classical approaches. These experiments facilitated via GluonTS are a first step towards a more thorough understanding of neural architectures for time series modeling.

References

Martín Abadi et al. TensorFlow: A system for large-scale machine learning. In *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation*, OSDI'16, pages 265–283, Berkeley, CA, USA, 2016. USENIX Association.

Konstantinos Benidis et al. Neural forecasting: Introduction and literature overview. *arXiv preprint arXiv:2004.10240*, 2020.

Eli Bingham et al. Pyro: Deep universal probabilistic programming. *Journal of Machine Learning Research*, 2018.

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

Zhenwen Dai et al. MXFusion: A modular deep probabilistic programming library. In *NIPS Workshop MLOSS (Machine Learning Open Source Software)*, 2018.

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

Valentin Flunkert et al. DeepAR: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting*, 2019.

Jan Gasthaus et al. Probabilistic forecasting with spline quantile function RNNs. In *AISTATS*, 2019.

Agathe Girard et al. Gaussian process priors with uncertain inputs application to multiple-step ahead time series forecasting. In *Advances in Neural Information Processing Systems*, pages 545–552, 2003.

Felix Hieber et al. The sockeye neural machine translation toolkit at AMTA 2018. In *AMTA*, pages 200–207. Association for Machine Translation in the Americas, 2018.

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

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

Tim Januschowski, Jan Gasthaus, and Yuyang Wang. Open-source forecasting tools in Python. *Foresight: The International Journal of Applied Forecasting*, (55):20–26, Fall 2019.

Guokun Lai et al. Modeling long- and short-term temporal patterns with deep neural networks. *CoRR*, abs/1703.07015, 2017. URL `http://arxiv.org/abs/1703.07015`.

Markus Lning et al. sktime: A unified interface for machine learning with time series. *arXiv preprint arXiv:1909.07872*, 2019.

Spyros Makridakis et al. The M4 competition: Results, findings, conclusion and way forward. *International Journal of Forecasting*, 34(4):802 – 808, 2018.

Adam Paszke et al. Automatic differentiation in PyTorch. In *NIPS-W*, 2017.

Fabian Pedregosa et al. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, Nov. 2011.

Syama Sundar Rangapuram et al. Deep state space models for time series forecasting. In *Advances in Neural Information Processing Systems*, 2018.

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

Aäron van den Oord et al. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

Ashish Vaswani et al. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017.

Yuyang Wang et al. Deep factors for forecasting. In *International Conference on Machine Learning*, pages 6607–6617, 2019.

Ruofeng Wen et al. A multi-horizon quantile recurrent forecaster. In *NIPS Time Series Workshop*. 2017.