

# Advanced Time Series

## Lecture 2: **Forecasting - II**

Gleb Ivashkevich

# Today

## TS forecasting II:

- **probabilistic** forecasting: DeepAR model
- considerations for endo + exog
- AR connections

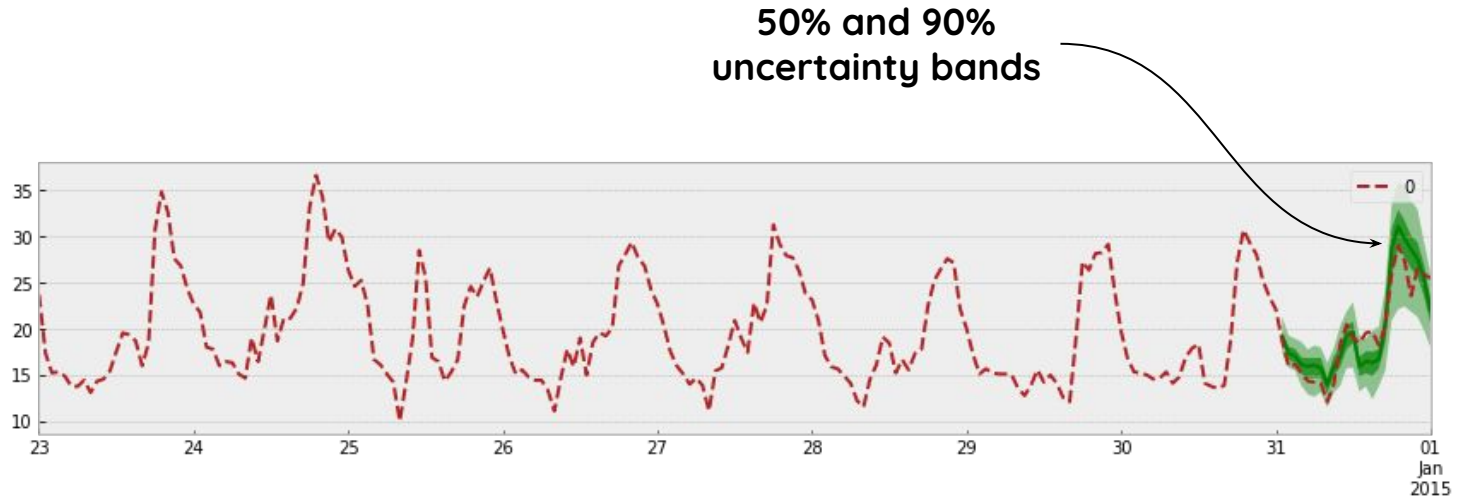
# Probabilistic forecasting

# Probabilistic forecasting

**Point forecasts** provide only one sample from a **random process**. Very often it is **not useful**.

**Probabilistic forecasts** provide entire distribution over future values. Very plausible feature, as it allows for uncertainty estimation and fine-grained management of extreme scenarios.

# Probabilistic forecasting



# Two flavors of probabilistic forecasting

- **Bayesian**
- forecast **probability distribution** directly

# Bayesian

The diagram illustrates the Bayesian inference equation:  $p(\theta|D) = \frac{p(D|\theta)p(\theta)}{p(D)}$ . It includes several labels and arrows: 'likelihood' points to  $p(D|\theta)$ ; 'prior distribution over model parameters' points to  $p(\theta)$ ; 'posterior distribution over model parameters' points to  $p(\theta|D)$ ; 'family of models' points to the entire equation; and an arrow points from 'family of models' to 'inference'.

**likelihood**

**prior**  
distribution  
over model  
parameters

$$p(\theta|D) = \frac{p(D|\theta)p(\theta)}{p(D)}$$

**posterior**  
distribution  
over model  
parameters

**family**  
of models

**inference**

# Bayesian

Bayesian ML is a **domain on its own**. We need only **likelihood**.

To compute likelihood, we need to set **probability distribution**.



# Likelihood

Gaussian distribution:

$$p(x|\mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

$p(D|\theta)$

# Direct

Predict **probability distribution parameters**:

- impose a **proper** distribution (say, Gaussian for continuous, Poisson for counts)
- **predict parameters** of that distribution with your model
- **maximize likelihood** of your data given the predicted parameters
- at inference time: **sample** from the predicted distribution

# DeepAR model

## DeepAR: Probabilistic Forecasting with Autoregressive Recurrent Networks

**DeepAR** model implements the idea:

- first introduced in 2017
- better handling of multiple related t. s.
- improves over the benchmarks

# DeepAR model

## DeepAR:

- despite “autoregressive” in its name, it’s an **encoder-decoder** architecture with **shared weights**
- trained on **electricity load** dataset and others
- **pre-processing** to handle **dramatically different ranges** for different time series

# DeepAR model

## Pre-processing, training sampling, features:

- **scale** inputs by **per-item average** (works for continuous, but not counts)
- sample time series with large scale **more frequently**
- calendar features + age
- **embeddings** for categoricals (product category for sales/ **identity** for others)

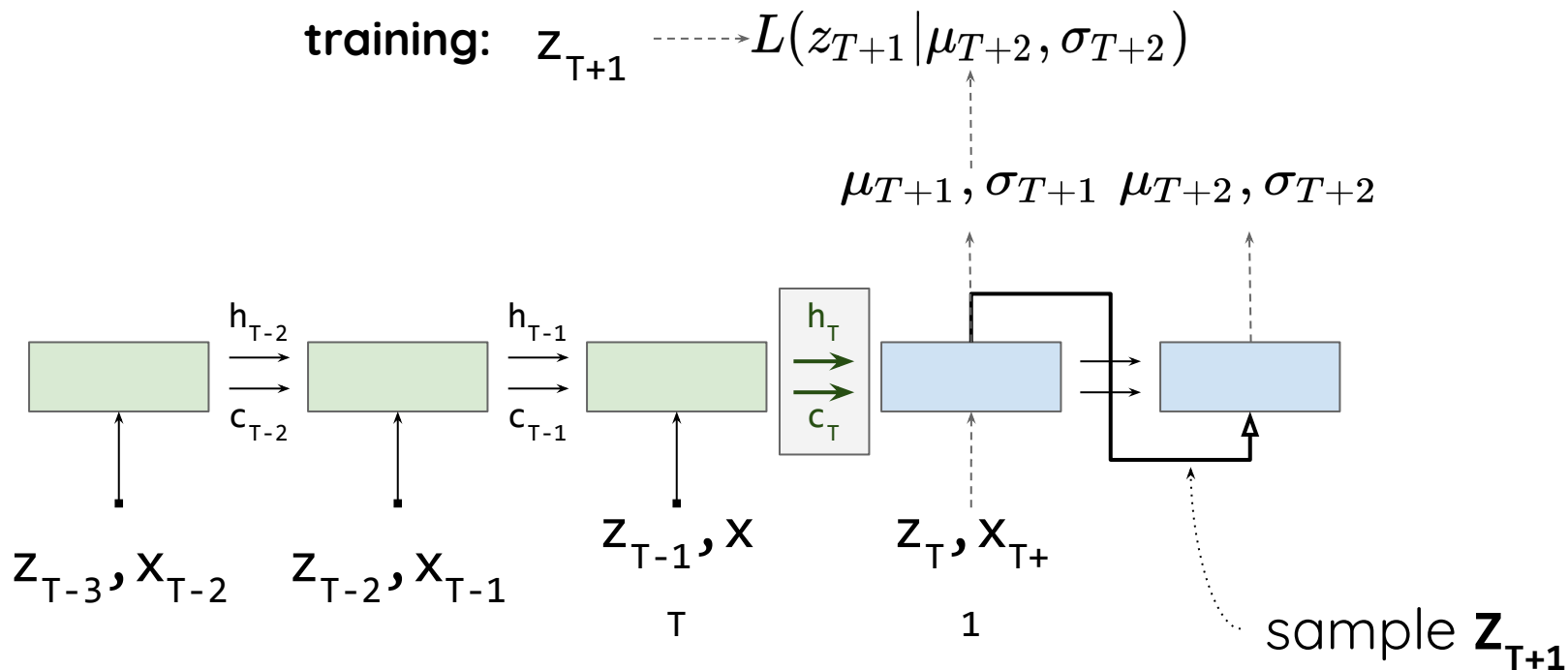
# DeepAR model

Design allows for:

- **leveraging information** from related time series: cold start problem resolved (to some extent)
- straightforward **probabilistic forecasting**
- **additional covariates** are not a problem

Implementations: [GluonTS](#) - MXNet-based

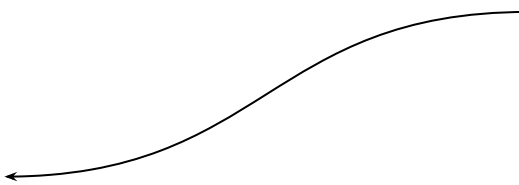
# DeepAR design in details



# DeepAR model

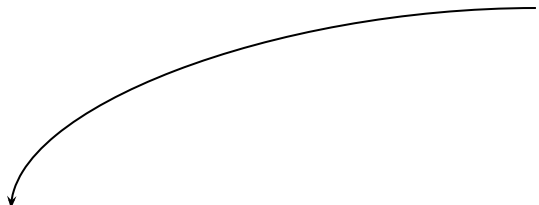
$$\mu(h_{i,t}) = W_{\mu} h_{i,t} + b_{\mu}$$

linear dense layer



$$\sigma(h_{i,t}) = \log(1 + \exp(W_{\sigma} h_{i,t} + b_{\sigma}))$$

softplus dense layer



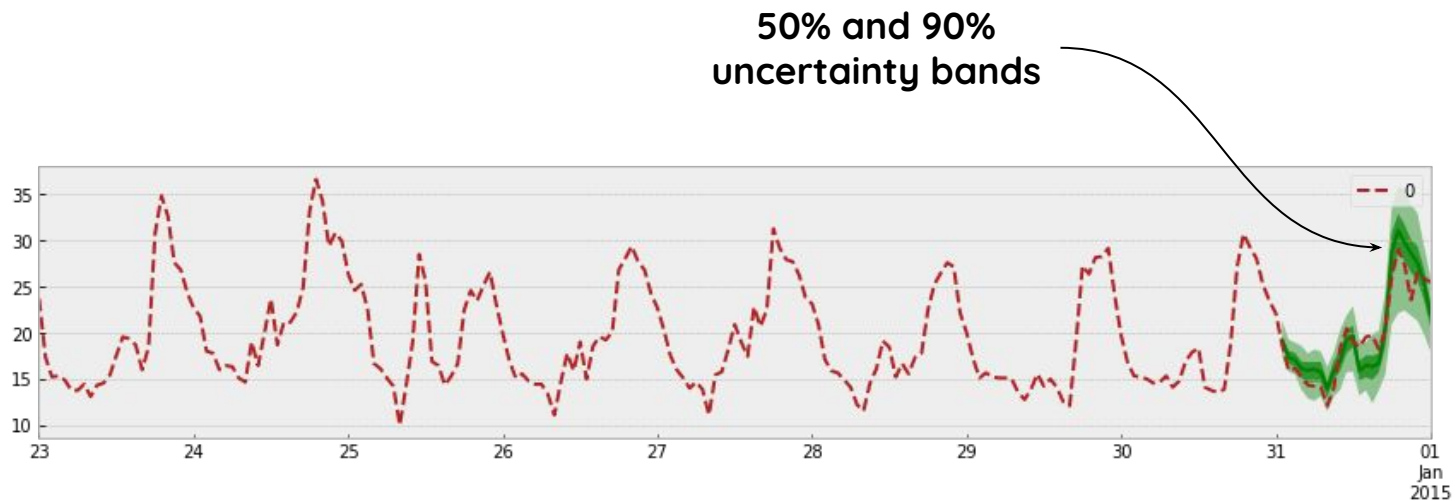


# DeepAR model

## Design technicalities:

- hourly data, 168 hours of historical data, 24 hours forecasting horizon
- 3 LSTM layers, 40 hidden units
- batch size: 64

# DeepAR model

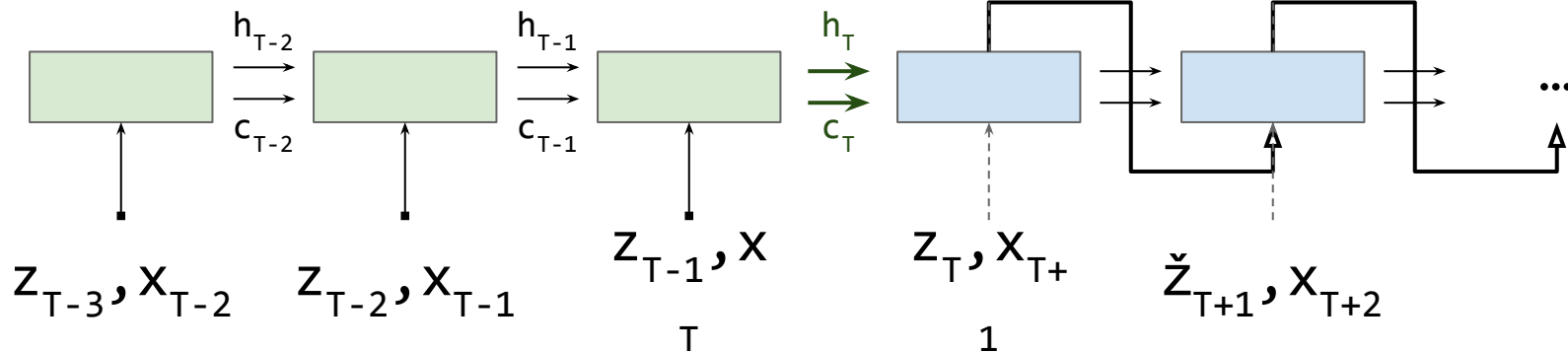


# Endogenous + exogenous considerations

# Shared weights

## Design technicalities:

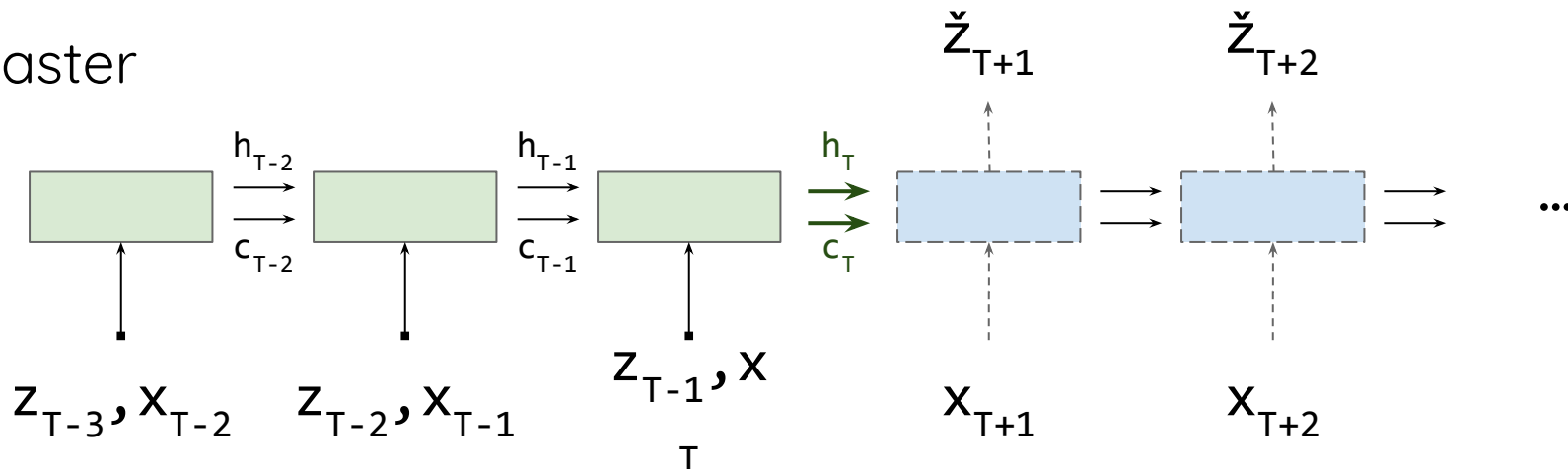
- re-feed forecasts in decoder
- slower



# Non-shared weights

## Design technicalities:

- use only exogenous in decoder
- faster



# Shared vs. non-shared

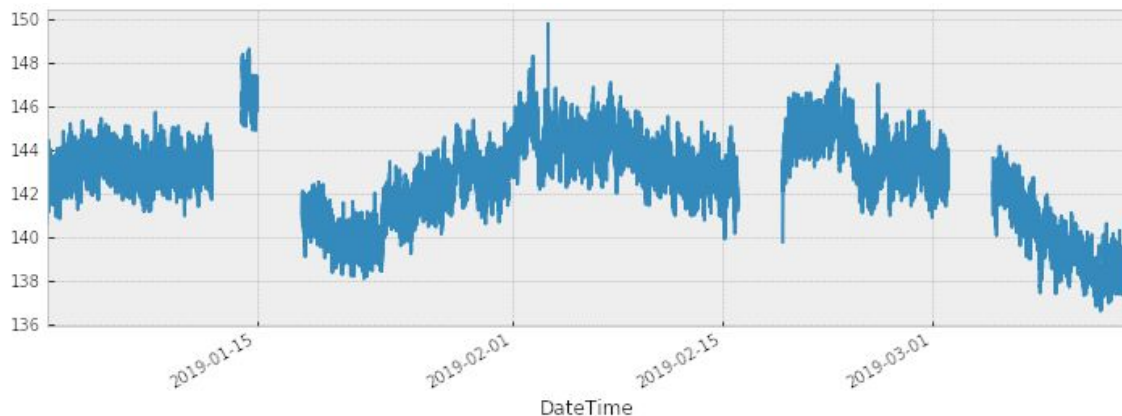
## Choice:

- strong **AR** dependence in the target series: **shared**
- target is **mainly** driven by exogenous: **non-shared**
- target is **fully** driven by exogenous: **shared with exogenous only**

Explicit autoregressive  
component

# Time series scales

For **non-stationary** time series, **scale** may possess a problem: non-linear elements of deep learning models do not scale. **Variability** may be, in contrast, **nearly constant**.





# Time series scales

Linear elements, in contrast, are **resilient** to scale changes. We may want to add a **direct AR component**:

$$\tilde{y}_{T+1} = \underbrace{y_{T+1}^{AR}}_{\text{autoregressive part}} + \underbrace{y_{T+1}^{ED}}_{\text{encoder-decoder part}}$$

$$y_{T+1}^{AR} = \sum_{k=0}^{k=h} W_k y_{T-k} + b$$

# AR component

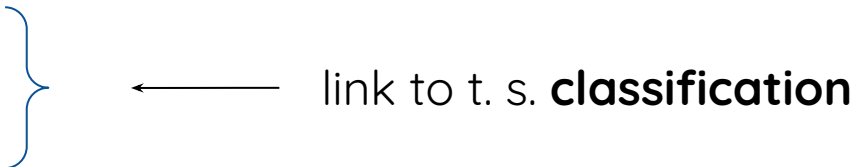
## Direct AR component:

- handles changing <sup>(non-seasonal!)</sup> scale
- works nicely with other elements, objective functions and training procedures <sup>(it's just a dense layer after all)</sup>

Next steps

# Hybrid architectures

Usable components:

- recurrent
  - AR
  - convolutional
  - attention
- 
- A blue right-facing curly bracket groups the last three items of the list: 'convolutional', 'attention', and 'link to t. s. **classification**'. A horizontal arrow points from the text 'link to t. s. **classification**' to the middle of the bracket.

[Modeling Long- and Short-Term Temporal Patterns with Deep Neural Networks](#)

# HW 1

## A Multi-Horizon Quantile Recurrent Forecaster

<https://arxiv.org/abs/1711.11053>

### Paper review:

- deadline is June 3 24:00
- see instructions in Google Classroom
- **compare with DeepAR**

questions?