Forecasting Big Time Series: Theory and Practice: Part II

Christos Faloutsos, Valentin Flunkert, **Jan Gasthaus**, Tim Januschowski, **Yuyang Wang**

CMU/Amazon Research and AWS AI Labs

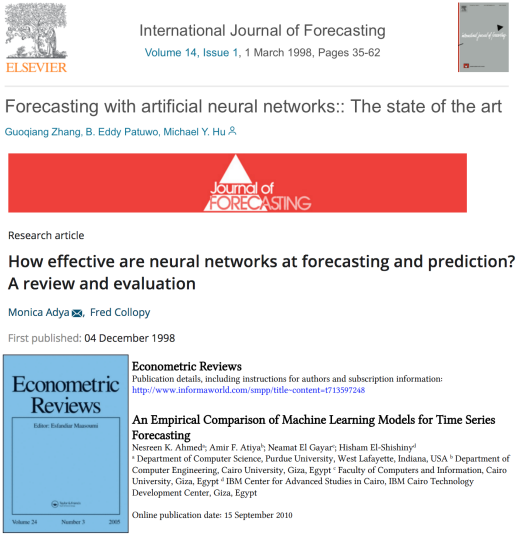




https://lovvge.github.io/Forecasting-Tutorial-KDD-2019/

August 4th, 2019

Forecasting with Neural Networks – Timeline



- 1969 Weather forecasting with adaptive linear neurons (Hu)
- 1986 Backpropagation (Rumelhart et al.)
- 1988 NNs using backpropagation applied to forecasting; positive results (Werbos)
- 199x Many authors applying mostly feed-forward models to various forecasting problem (single time series)
- 1998 Review articles: "The outcome of all of these studies has been somewhat mixed"; "While ANNs provide a great deal of promise, they also embody much uncertainty."
- 2000 M3 competition – simple methods declared the winner
- 200x Less work on NN-based forecasting methods
- 2012 AlexNet wins ImageNet competition – start of the Deep Learning revival (Krizhevsky et al.)
- 2014 Generating Sequences With RNNs (Graves); seq2seq architecture (Sutskever et al.)
- 2014– Modern deep learning techniques (RNNs, CNNs) get applied to forecasting (across time series)
- 2018 M4 competition: combination of NNs and classical techniques wins

Forecasting with Neural Networks: Old and New

"Consensus" in the Forecasting Community: NNs don't work!

This supports the general consensus in forecasting, that neural networks (and other highly non-linear and nonparametric methods) are not well suited to time series forecasting due to the relatively short nature of most time series. The longest series in this competition was only 126 observations long. That is simply not enough data to fit a good neural network model.

– Rob Hyndman on M-challenges, 2018

Forecasting with Neural Networks: Old and New

"Consensus" in the Forecasting Community: NNs don't work!

This supports the general consensus in forecasting, that neural networks (and other highly non-linear and nonparametric methods) are not well suited to time series forecasting due to the relatively short nature of most time series. The longest series in this competition was only 126 observations long. That is simply not enough data to fit a good neural network model.

– Rob Hyndman on M-challenges, 2018

Our View

Neural networks are **great** for learning complex patterns from *many* time series in operational forecasting problems!

Neural Networks win M4 competition

- most influential forecasting competition
- 100,000 series of following frequencies: monthly, quarterly, yearly, daily, weekly, and hourly.

Neural Networks win M4 competition

- most influential forecasting competition
- 100,000 series of following frequencies: monthly, quarterly, yearly, daily, weekly, and hourly.
- *won by a neural network (work by Slawek Smyl, Uber)*

Deep Learning for Forecasting: Shouldn't it just work?

"Time series? Just use an RNN!"

- Is there anything special about forecasting?
- Shouldn't the hugely successful models from areas like NLP and CV *just work*?

Deep Learning for Forecasting: Shouldn't it just work?

"Time series? Just use an RNN!"

- Is there anything special about forecasting?
- Shouldn't the hugely successful models from areas like NLP and CV *just work*?

YES!

Deep Learning for Forecasting: Shouldn't it just work?

"Time series? Just use an RNN!"

- Is there anything special about forecasting?
- Shouldn't the hugely successful models from areas like NLP and CV *just work*?

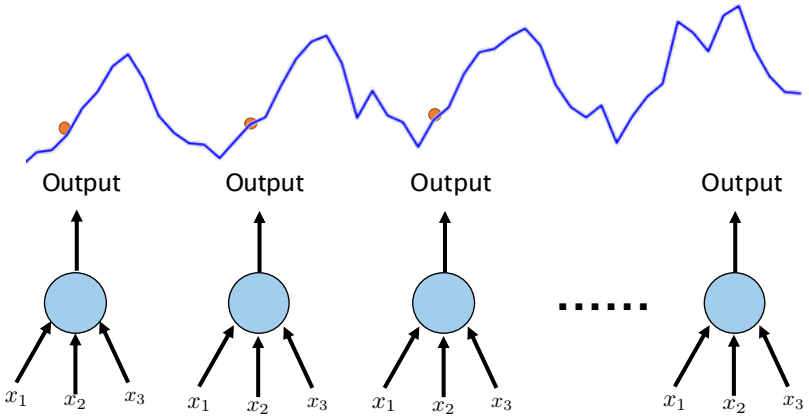
YES!

- ... mostly.
- Challenges: scaling, probability distribution outputs, sample efficiency, incorporating prior knowledge

Deep Learning for Forecasting: Outline

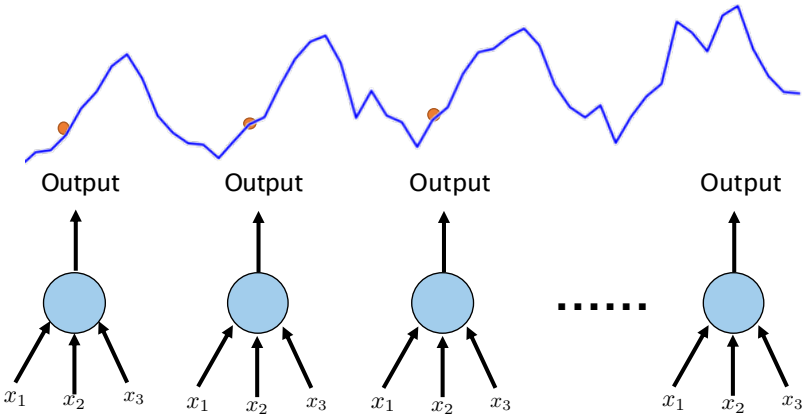
1. Non-linear regression, feed-forward models (aka MLPs)
2. Basic model structure
3. Training neural network models
4. Convolutional Neural Nets
5. Recurrent Neural Nets
6. Probabilistic Forecasting with Neural Nets
7. Deep Probabilistic Models

From Linear Regression to Feed-Forward Neural Networks



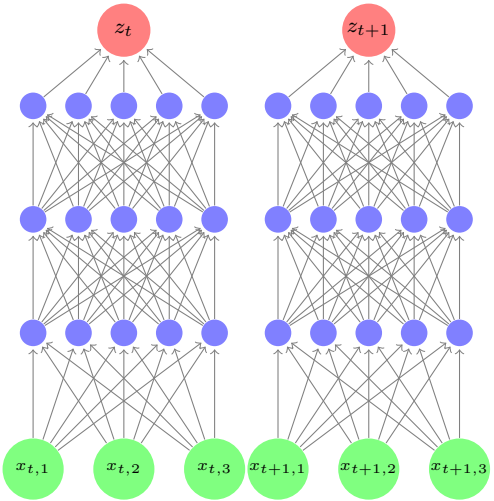
$$z_t = \boldsymbol{w}^T \boldsymbol{x}_t$$

From Linear Regression to Feed-Forward Neural Networks



$$z_t = \sigma(\boldsymbol{w}_l^T(\sigma(W_{l-1}^T(\sigma(W_{l-2}^T(\cdots W_0^T \boldsymbol{x}_t))))) := \text{DEEP-NET}(\boldsymbol{x}_t)$$

Feed-Forward Neural Networks (Multi-layer Perceptron (MLPs))

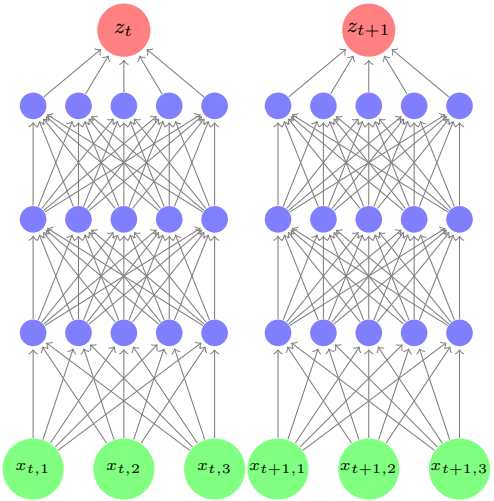


- Linear model + non-linear hidden layers
- Each neuron in a hidden layer computes an affine function of the previous layer, followed by a non-linear *activation* function,

$$h_{l,j} = \sigma\left(\mathbf{w}_{l,j}^{\top}\mathbf{h}_{l-1} + b_{l,j}\right)$$

- FF Neural Networks are flexible general function estimators
- More (and larger) hidden layers $\rightarrow$ more complex functions

Feed-Forward Neural Networks (Multi-layer Perceptron (MLPs))

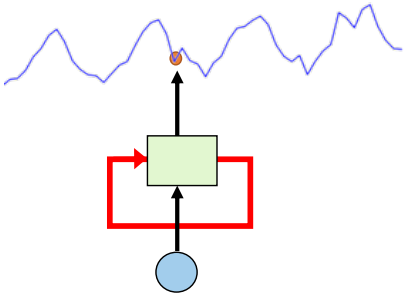


- Main advantage over linear models: Can learn complex input-output relationships

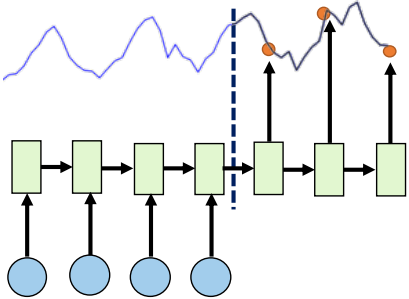
$\Rightarrow$ Less manual feature engineering

- Main disadvantage: more data needed for training
- Careful tuning (e.g. of regularization, learning rate, etc.) might be necessary for good results
- Sensitive to scaling of inputs

Basic Model Structures



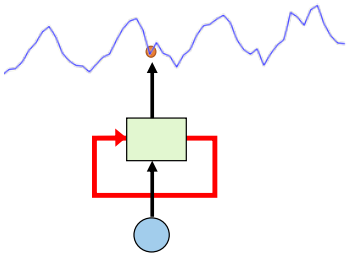
Canonical (One-to-One)



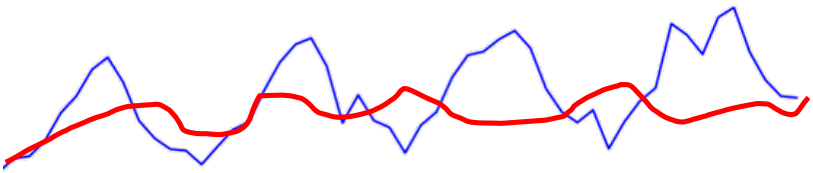
Seq2Seq (Many-to-Many)

One-to-One Structure (Generative Model)

$$f_t : x_t \mapsto z_t$$



Canonical (One-to-One)

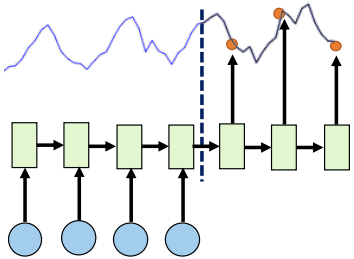


Training sequence

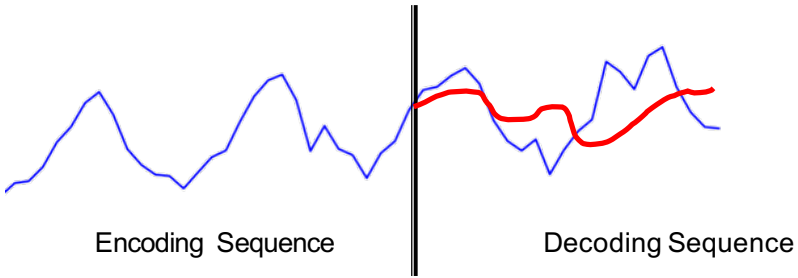
How well does the prediction reconstruct the the observed time series?

Sequence-to-Sequence / Many-to-Many Structure (Discriminative Model)

$$f : \{z_1, \cdots, z_{T_e}\} \mapsto \{z_{T_e+1}, \cdots, z_{T_e+T_d}\}$$



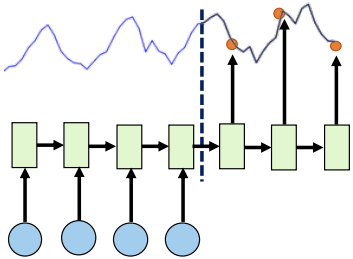
Seq2Seq (Many-to-Many)



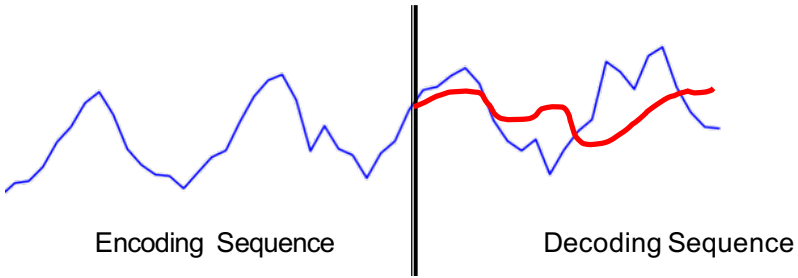
How well does the prediction reconstruct the decoding sequence conditioned on the encoding sequence?

Sequence-to-Sequence / Many-to-Many Structure (Discriminative Model)

$$f : \{z_1, \cdots, z_{T_e}\} \mapsto \{z_{T_e+1}, \cdots, z_{T_e+T_d}\}$$



Seq2Seq (Many-to-Many)



How well does the prediction reconstruct the decoding sequence conditioned on the encoding sequence? Conceptually close to multivariate regression

Training Neural Networks

General recipe

Pick a class of functions $f(\mathbf{x}; \theta)$ and learn the parameters θ by minimizing some notion of error on a *training set*,

$$\theta^\star = \underset{\theta}{\operatorname{argmin}} \sum_i L\left(\mathbf{z}_i, f(\mathbf{x}_i, \theta)\right)$$

- Optimization algorithm of choice: Stochastic Gradient Descent (SGD)
  1. For each iteration $k = 1, 2, 3, \ldots$
  2. Randomly pick a *minibatch* of examples $i_1, i_2, \ldots, i_B$
  3. Compute the batch loss $L_k = \sum_b L\left(\mathbf{z}_{i_b}, f(\mathbf{x}_{i_b}, \theta)\right)$
  4. Compute the gradient of the loss $g_k = \nabla_\theta L_k(\theta)$
  5. Update the parameters $\theta_k = \theta_{k-1} - \eta g_k$
  6. (Optional but recommended: Adjust the learning rate η)

Loss Functions

- In *supervised learning*, one key modelling choice is the *loss function*.
- In the forecasting context, the loss function compares a forecast to the truth (on historical training data where the truth is known).
- For point forecasts, a loss function compares two real numbers, e.g. $\widehat{z}_t$ vs. z_t,

$$e_t = (\widehat{z}_t - z_t)^2$$

- For distribution forecasts, a loss function compares a forecast distribution F_t to a real number z_t, using a so-called *scoring rule*, e.g., negative log likelihood.

Convolutional Neural Networks

Convolutional Neural Networks

- Convolutional Neural Networks (CNNs) = NNs that use *convolutional layers*
- Typical CNN model architectures combine convolutional layers with other layer types
- CNNs with 2D convolutions are extremely successful in computer vision applications

$$\Longrightarrow \text{encode spatial invariance}$$

- 1D convolutions are a promising alternative to RNNs for sequential data

$$\Longrightarrow \text{encode temporal invariance, ``stationarity''}$$

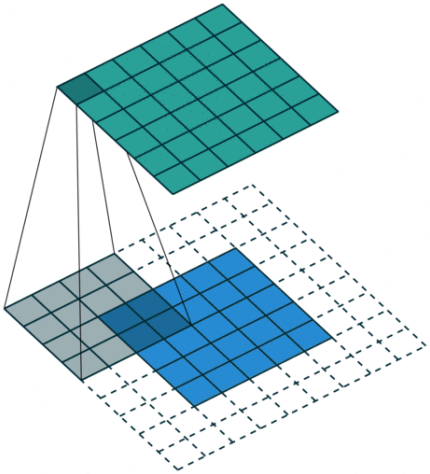
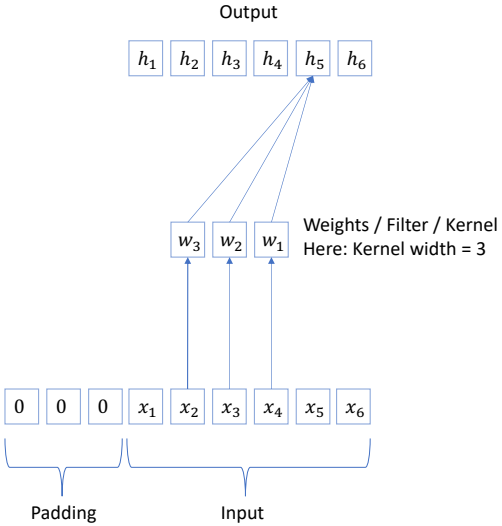


Figure credit: Vincent Dumoulin, Francesco Visin - A guide to convolution arithmetic for deep learning;
https://github.com/vdumoulin/conv_arithmetic

Convolutional Layers

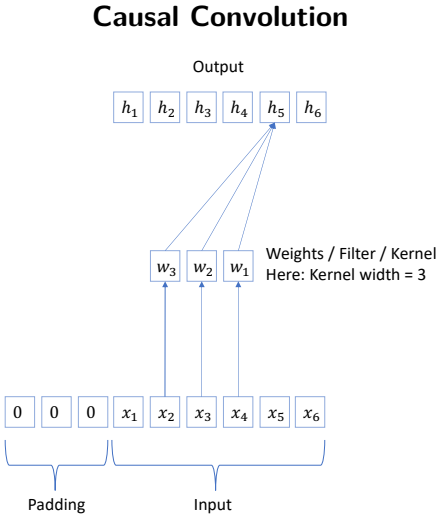


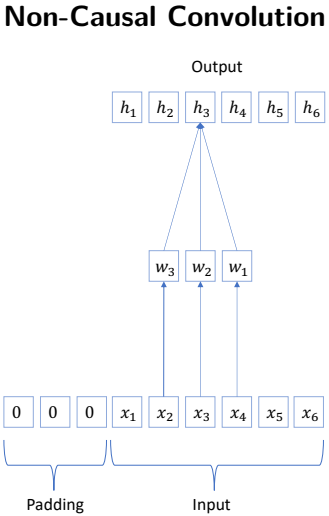
- The output h_j of a neuron j in a convolution layer is a discrete convolution of the inputs $\mathbf{x}$ with the layer's weigths/filter $\mathbf{w}$.
- For a one-dimensional convolution with a kernel with width D we have

$$h_j = \sum_{d=1}^{D} w_d x_{j-d}$$

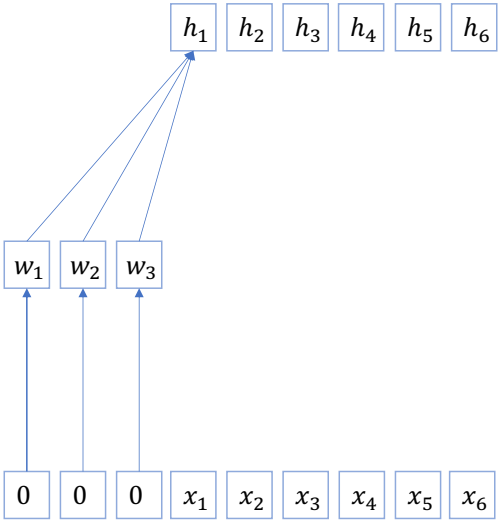
- Padding is used to shift the input relative to the output and change the behavior around the edges (causal vs. non-causal)

Causal vs. Non-Causal Convolution

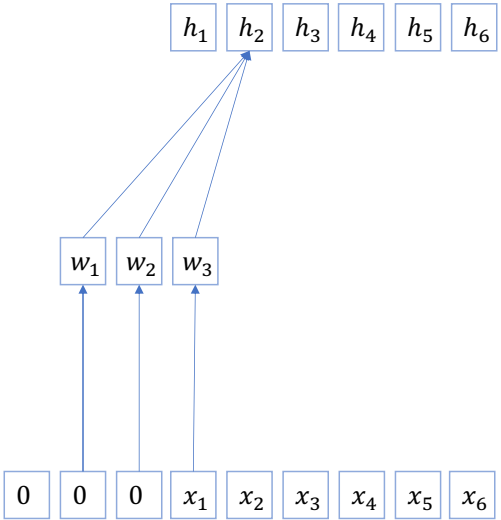




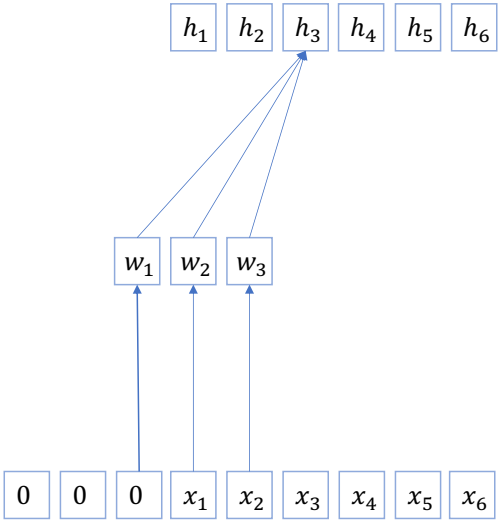
1D Causal Convolution



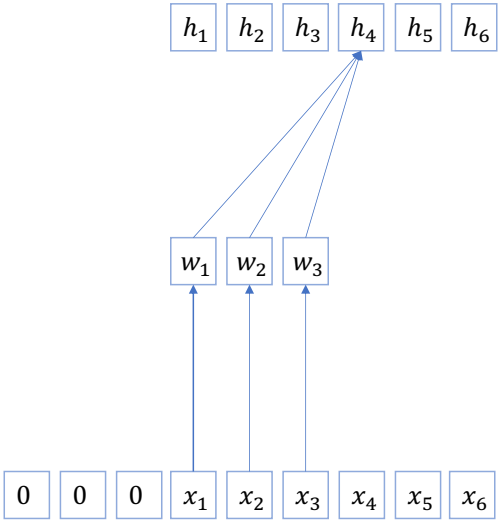
1D Causal Convolution



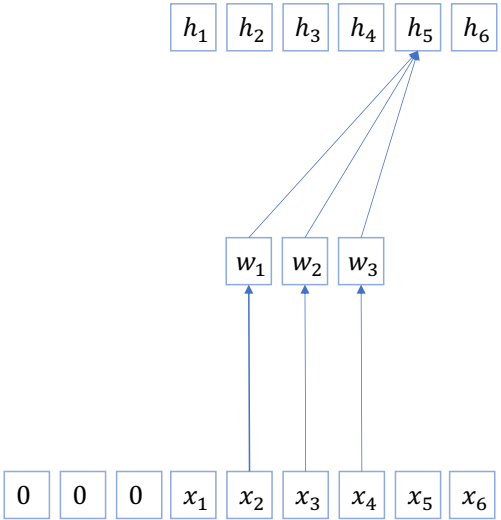
1D Causal Convolution



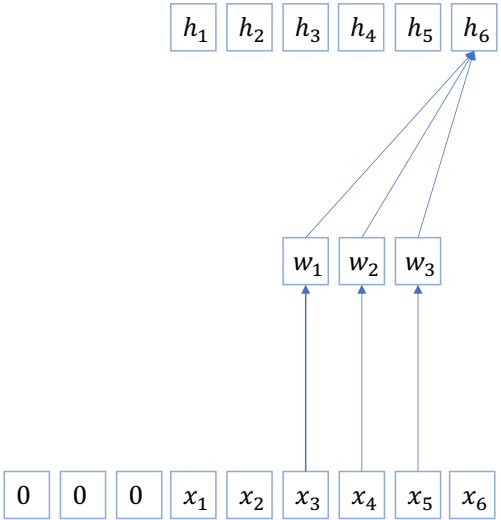
1D Causal Convolution



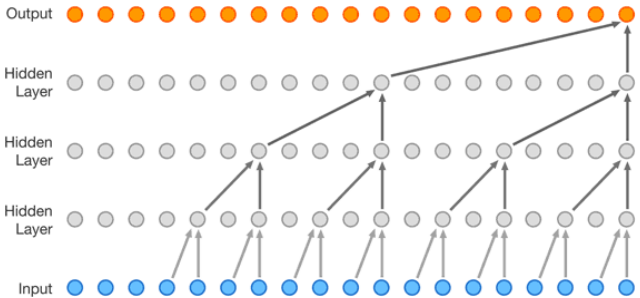
1D Causal Convolution



1D Causal Convolution



Dilated Causal Convolution and WaveNet [Van Den Oord et al., 2016]



- Dilation increases receptive field
- Forecast is generated in an autoregressive fashion
- Can be used as encoder or decoder in sequence-to-sequence (next section)
- More complex structures including gating and residual links [Van Den Oord et al., 2016]

Dilated Causal Convolution and WaveNet [Van Den Oord et al., 2016]

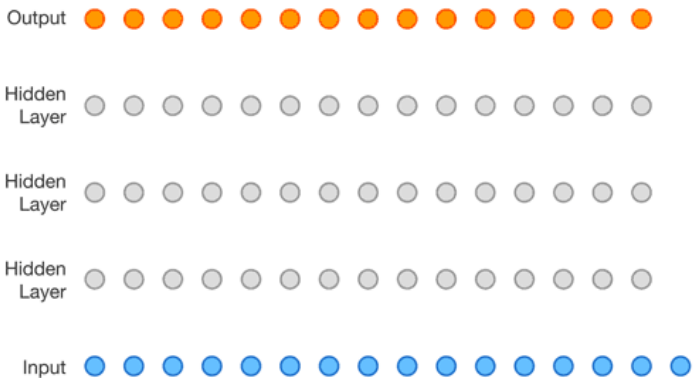
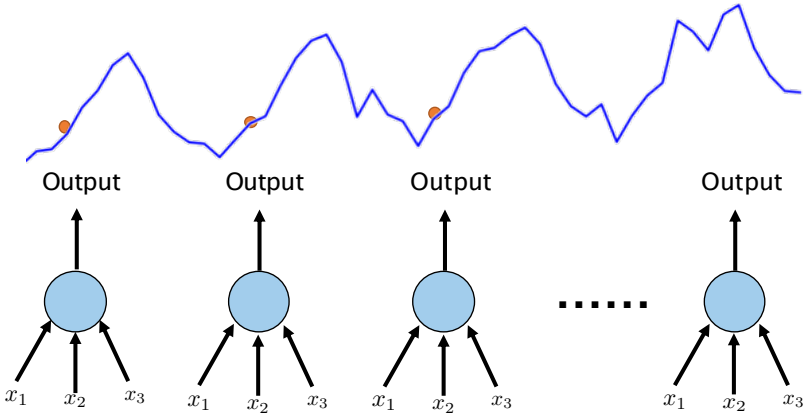


Figure credit: WaveNet: A Generative Model for Raw Audio; https://deepmind.com/blog/wavenet-generative-model-raw-audio/

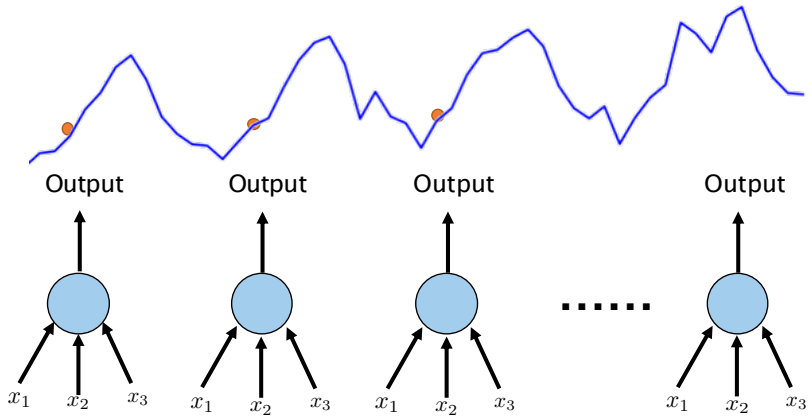
Recurrent Neural Networks

Recap: MLP for Forecasting



$$z_t = \text{DEEP-NET}(\boldsymbol{x}_t)$$

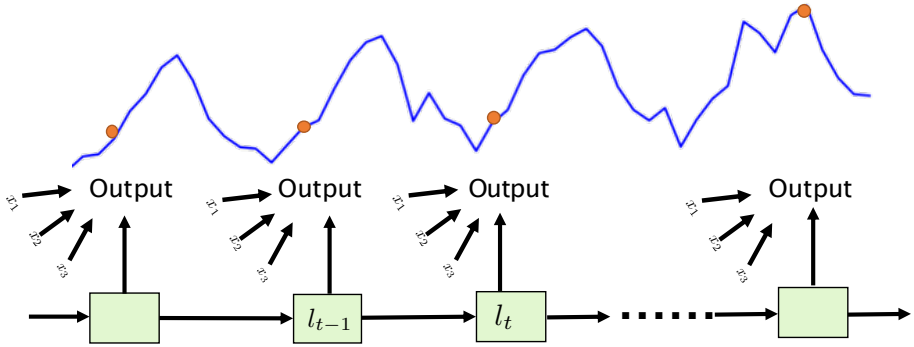
Recap: MLP for Forecasting



$$z_t = \text{DEEP-NET}(\boldsymbol{x}_t)$$

How about the sequential relationship?

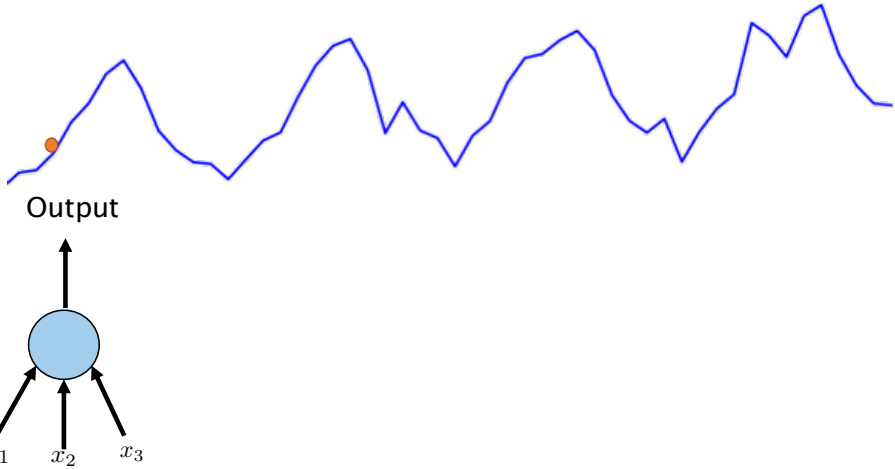
Recap: State-Space Models for Forecasting



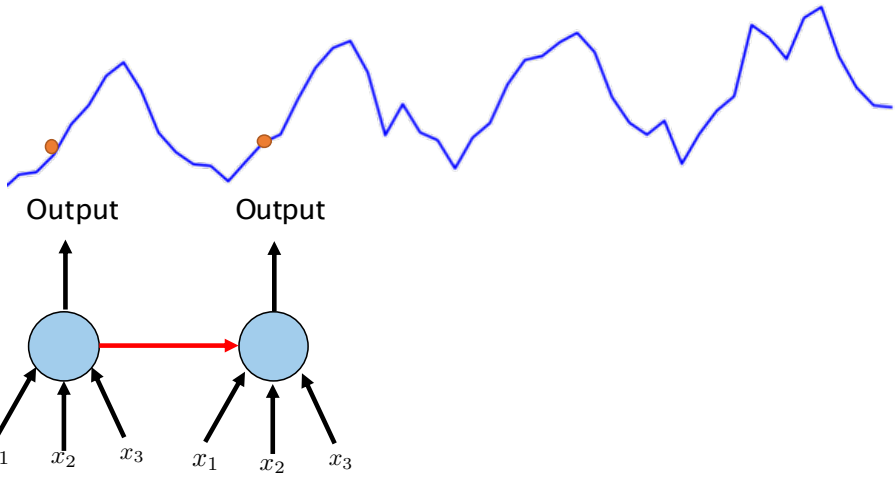
$$\boldsymbol{h}_t = \boldsymbol{l}_{t-1} + \alpha \cdot \epsilon_t$$
$$z_t = \boldsymbol{l}_t + \boldsymbol{w}^T \boldsymbol{x}_t + \epsilon_t$$

Can we do the same with NNs?

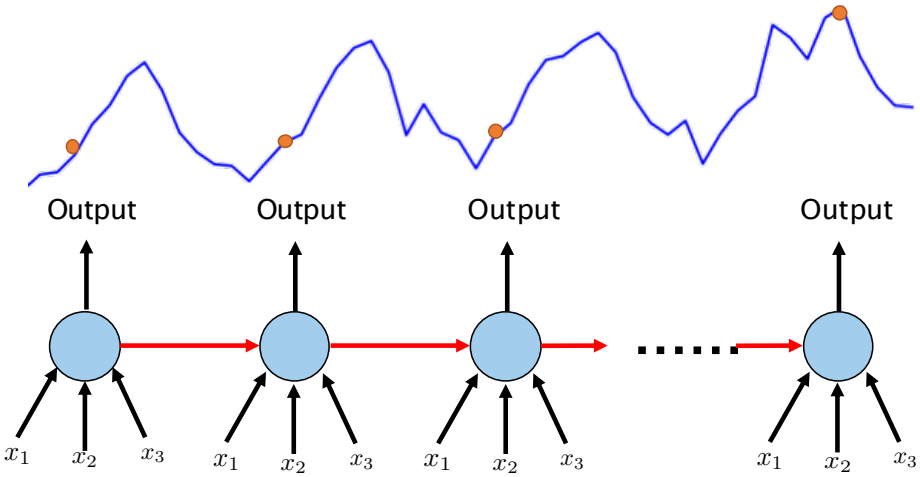
From Feed-forward NN to Recurrent NN



From Feed-forward NN to Recurrent NN



From Feed-forward NN to Recurrent NN

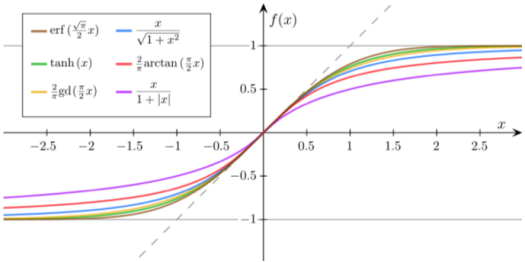


From Latent State (Exponential Smoothing) to Recurrent NN

Current hidden state h_t combines

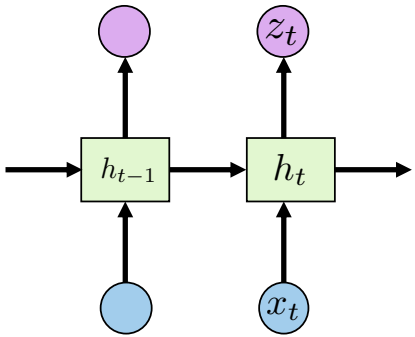
- the previous hidden state h_{t-1}
- input features x_t

and goes into



Source: Wikipedia

RECURRENT NEURAL NETWORK

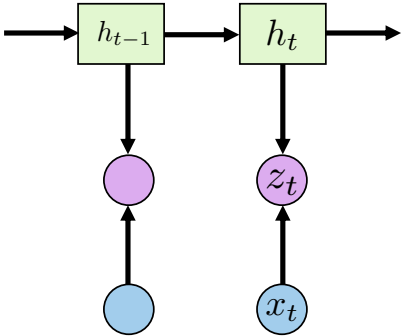


$$\boldsymbol{h}_t = \sigma(\theta_0 h_{t-1} + \theta_1 x_t)$$
$$z_t = \sigma(\theta h_t)$$

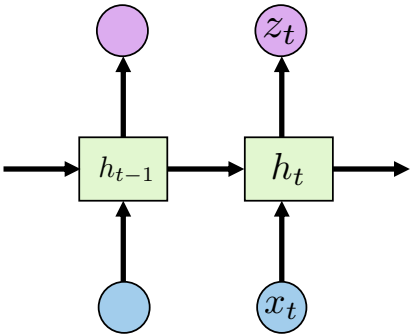
Central idea: Exponential Smoothing

today = yesterday's information + new knowledge

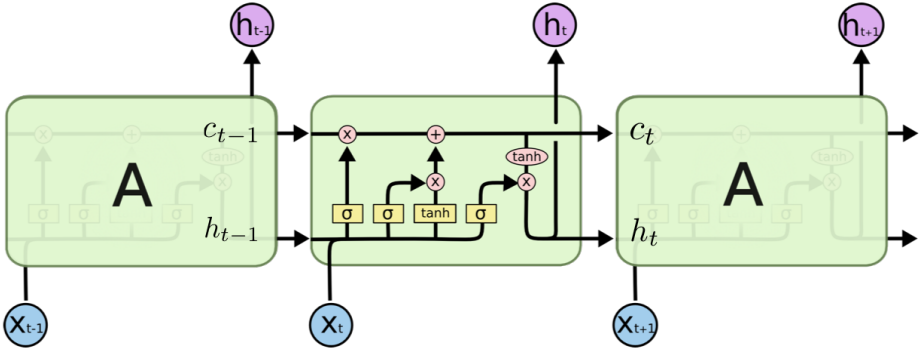
STATE-SPACE MODEL



RECURRENT NEURAL NETWORK



Long Short-Term Memory (LSTM) [Hochreiter and Schmidhuber, 1997]



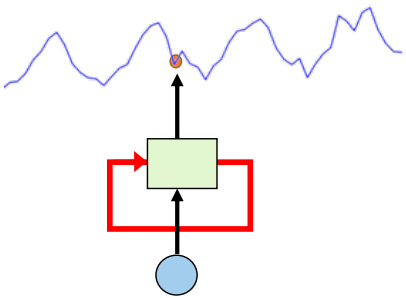
HTTP://COLAH.GITHUB.IO/POSTS/2015-08-UNDERSTANDING-LSTMS/

$$C_t = \alpha_t \cdot C_{t-1} + \beta_t \times \sigma(\theta_0 h_{t-1} + \theta_1 x_t)$$

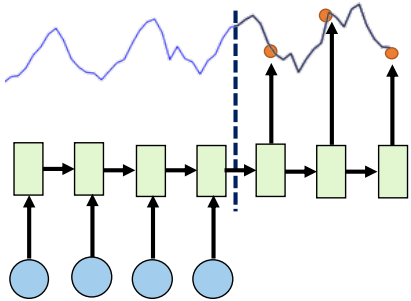
current state = forget gate × old stuff + input gate × new stuff.

The same exponential smoothing idea!

Basic Model Structures (again)



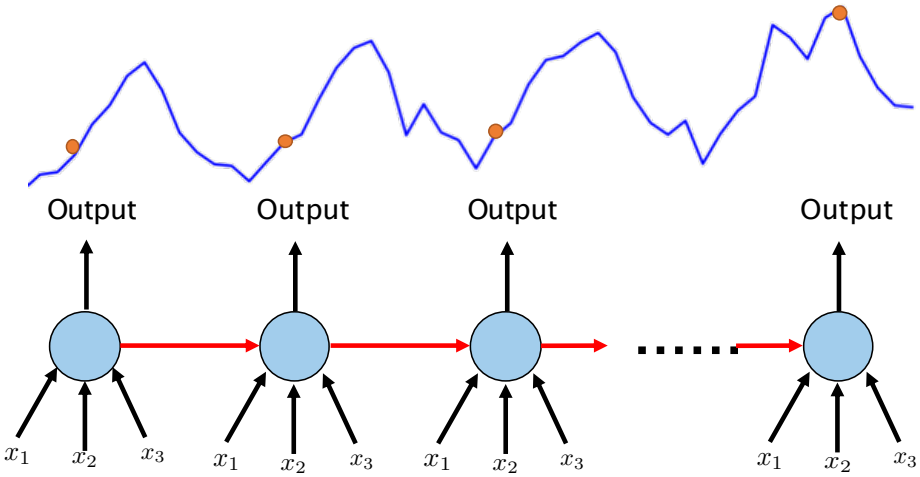
Canonical (One-to-One)



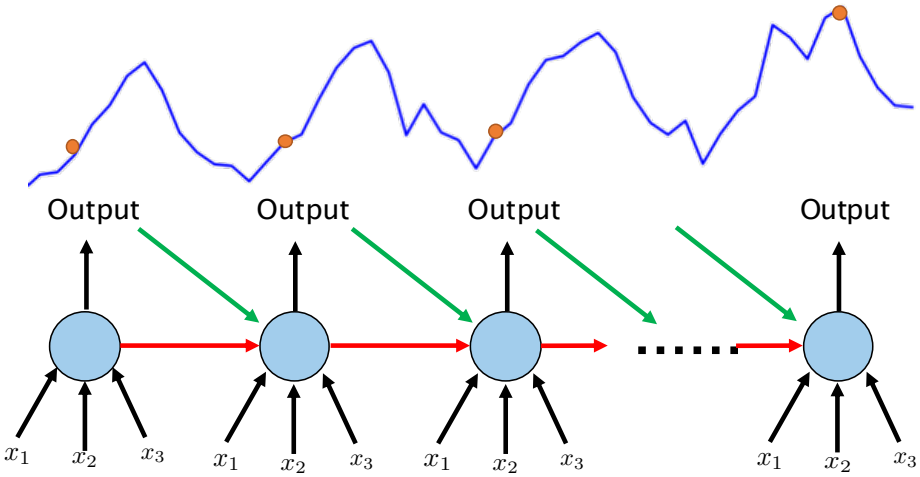
Seq2Seq (Many-to-Many)

- Canonical (One-to-One) RNN: DeepAR [Flunkert et al., 2017], AR-MDN [Mukherjee et al., 2018], Deep LSTM [Yu et al., 2017], ...
- Sequence-to-Sequence (Many-to-Many) models: Diffusion Convolutional RNNs [Li et al., 2018], MQ-RNN [Wen et al., 2017], ...

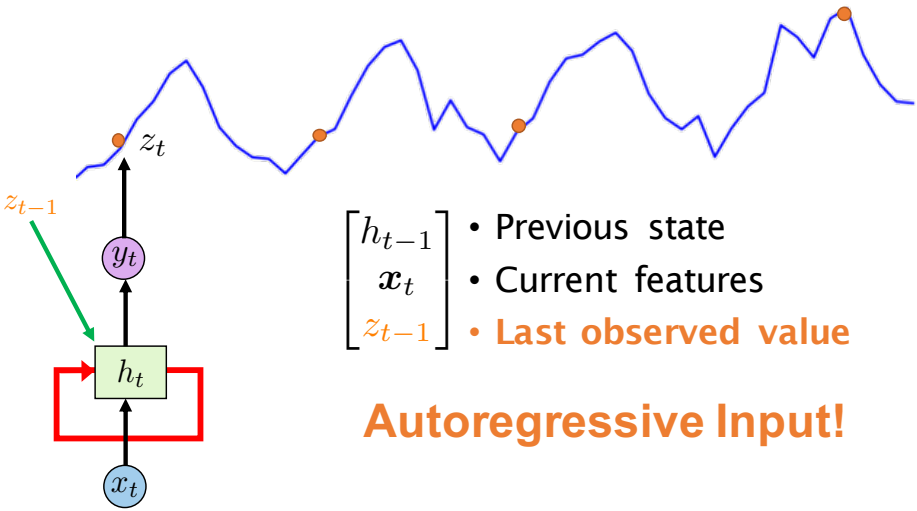
Canonical RNN Structure: DeepAR [Flunkert et al., 2017]



Canonical RNN Structure: DeepAR [Flunkert et al., 2017]



Canonical RNN Structure: DeepAR [Flunkert et al., 2017]

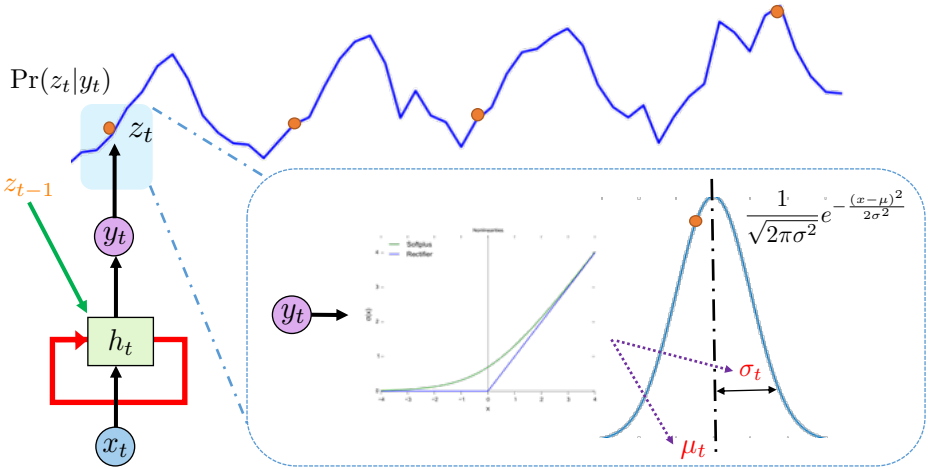


$$\begin{bmatrix} h_{t-1} \\ \boldsymbol{x}_t \\ z_{t-1} \end{bmatrix}$$

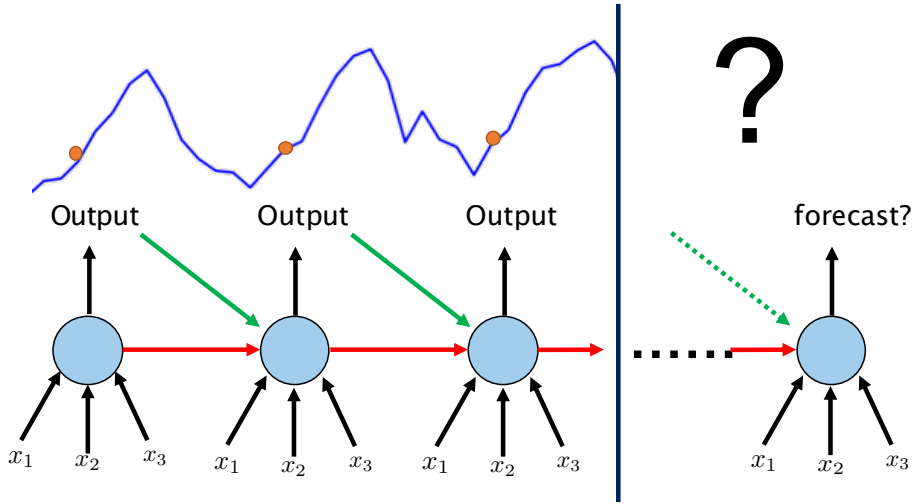
- Previous state
- Current features
- **Last observed value**

Autoregressive Input!

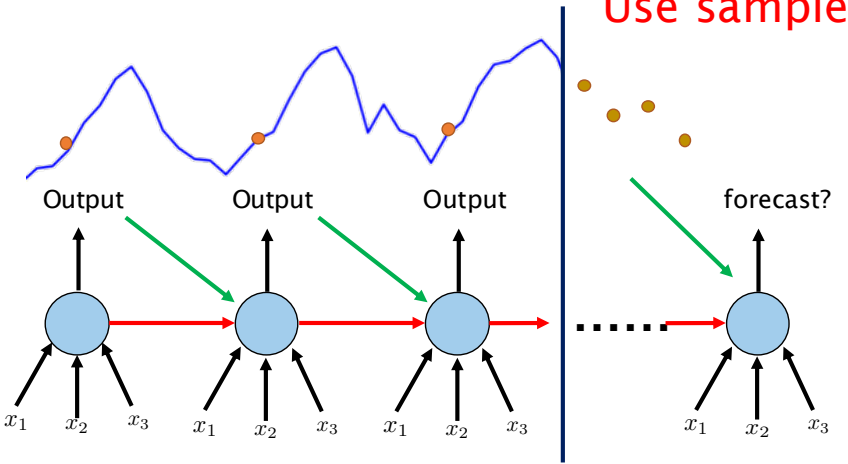
Canonical RNN Structure: DeepAR [Flunkert et al., 2017]



Canonical RNN Structure: How do we do forecast?

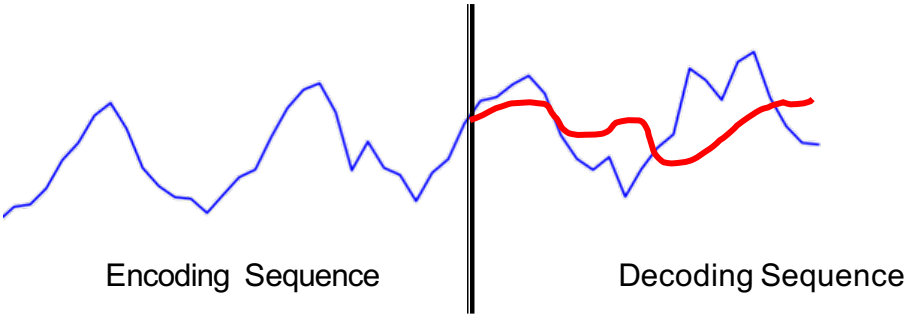


Canonical RNN Structure: How do we do forecast?

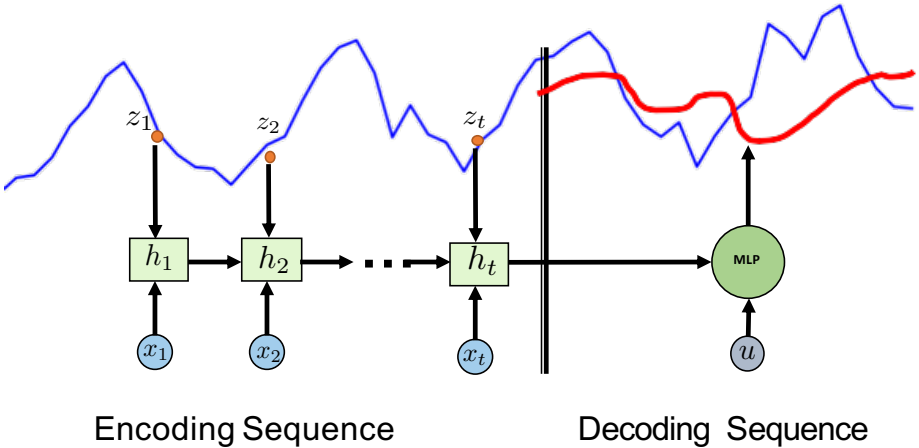


Sequence to Sequence (Seq2Seq) Structure: Many-to-Many

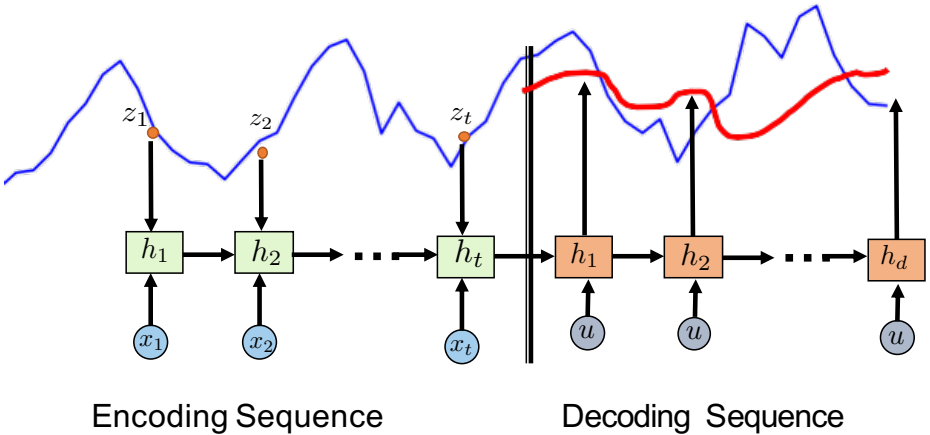
$$f_{encoder} : \{z_1, \cdots, z_{T_e}\} \mapsto \boldsymbol{h}_{T_e}$$
$$f_{decoder} : \boldsymbol{h}_{T_e} \mapsto \{z_{T_e+1}, \cdots, z_{T_e+T_d}\}$$



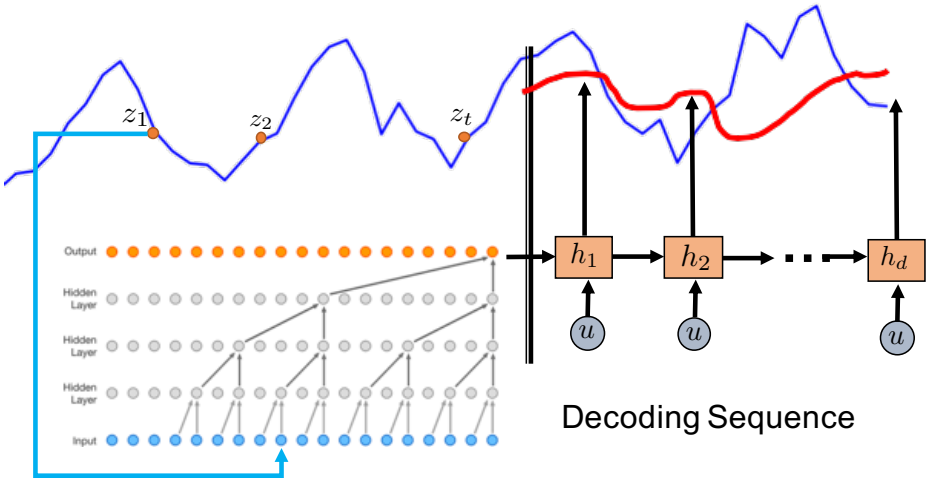
Seq2Seq: RNN-MLP [Wen et al., 2017]



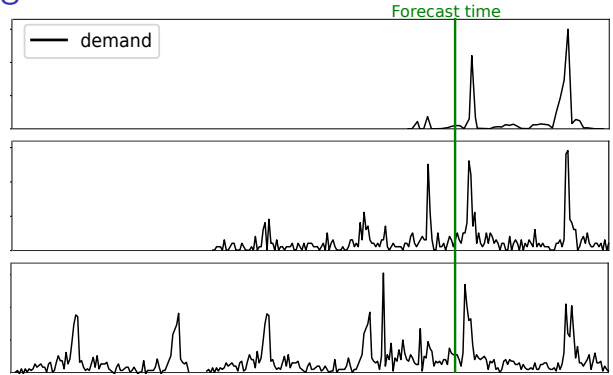
Seq2Seq: RNN-RNN



Seq2Seq: Causal CNN-RNN

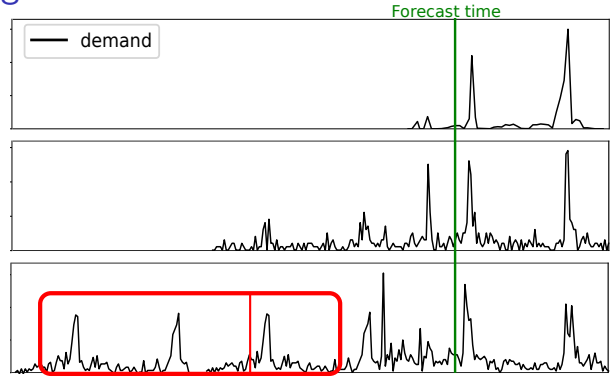


Seq2Seq: Training



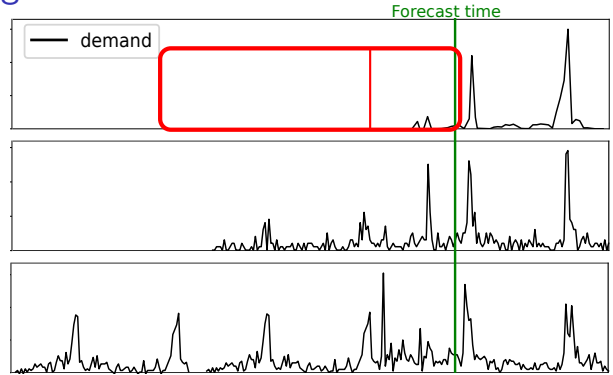
- Input:
  - time series (targets) z_t's: encoding and decoding
  - input features: encoding $\boldsymbol{x}_t$'s and decoding $\boldsymbol{u}$'s

Seq2Seq: Training



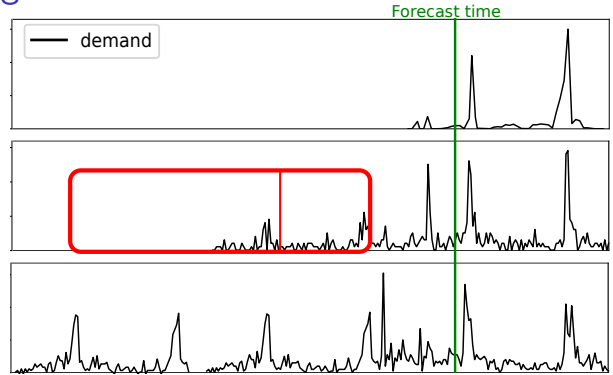
- Input:
  - time series (targets) z_t's: encoding and decoding
  - input features: encoding $\boldsymbol{x}_t$'s and decoding $\boldsymbol{u}$'s
- Slicing windows across item and time (temporal)
- Trained by minimizing certain metrics (negative loglikelihood, L_1/L_2 loss, quantile loss, etc.)

Seq2Seq: Training



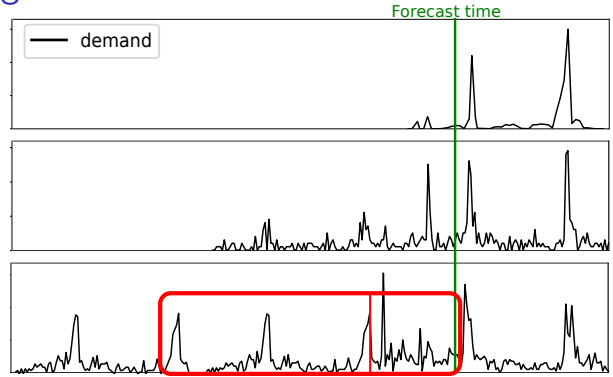
- Input:
  - time series (targets) z_t's: encoding and decoding
  - input features: encoding $\boldsymbol{x}_t$'s and decoding $\boldsymbol{u}$'s
- Slicing windows across item and time (temporal)
- Trained by minimizing certain metrics (negative loglikelihood, L_1/L_2 loss, quantile loss, etc.)

Seq2Seq: Training



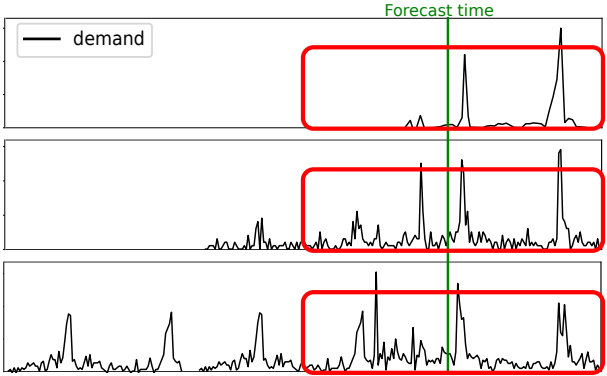
- Input:
  - time series (targets) z_t's: encoding and decoding
  - input features: encoding $\boldsymbol{x}_t$'s and decoding $\boldsymbol{u}$'s
- Slicing windows across item and time (temporal)
- Trained by minimizing certain metrics (negative loglikelihood, L_1/L_2 loss, quantile loss, etc.)

Seq2Seq: Training

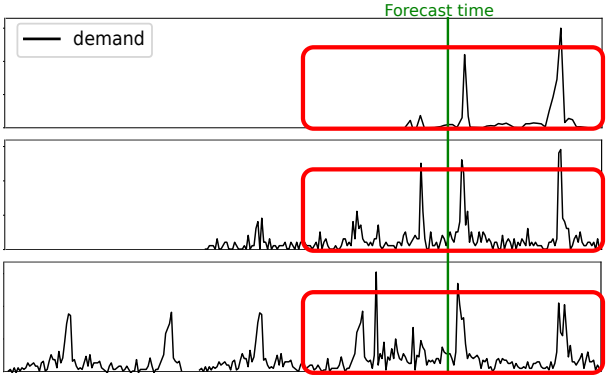


- Input:
  - time series (targets) z_t's: encoding and decoding
  - input features: encoding $\boldsymbol{x}_t$'s and decoding $\boldsymbol{u}$'s
- Slicing windows across item and time (temporal)
- Trained by minimizing certain metrics (negative loglikelihood, L_1/L_2 loss, quantile loss, etc.)

Seq2Seq: Prediction

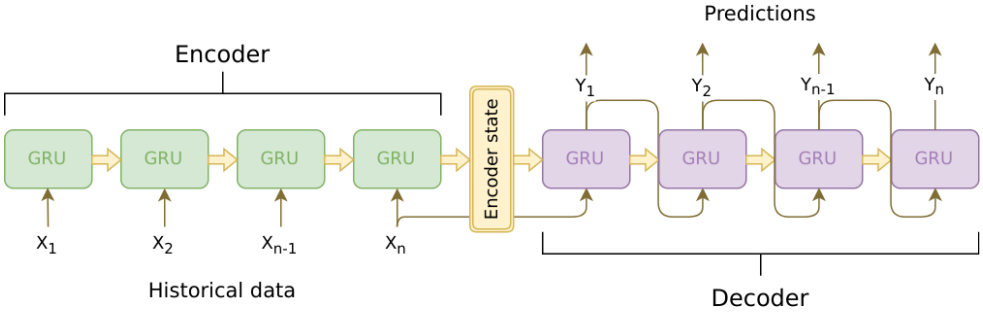


Seq2Seq: Prediction



- target z_t is unobserved after the forecast time

Seq2Seq: Gated Recurrent Units (GRU) [Suilin, 2017]

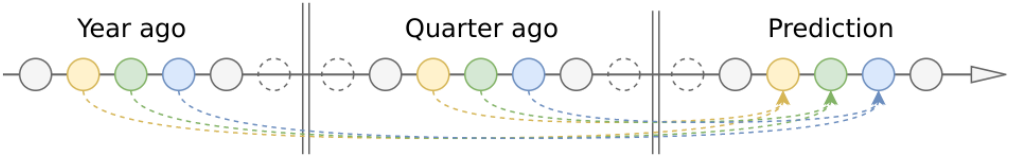


https://github.com/Arturus/kaggle-web-traffic/blob/master/how_it_works.md

- Kaggle Wikipedia winning solution: GRU and GRU decoder

What about ATTENTION?

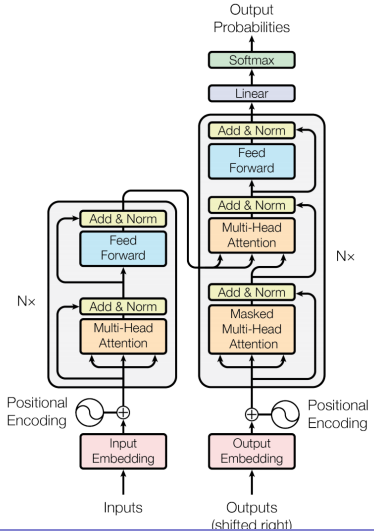
Lag is All You Need!



https://github.com/Arturus/kaggle-web-traffic/blob/master/how_it_works.md

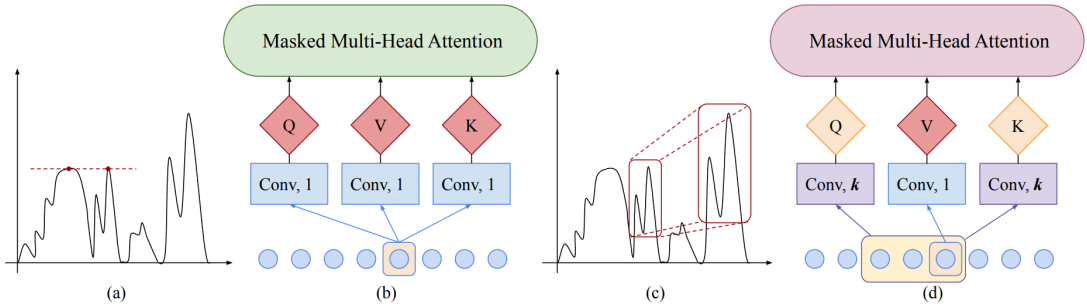
Transformer for Forecasting

Attention is All You Need, Vaswani et al. [2017]

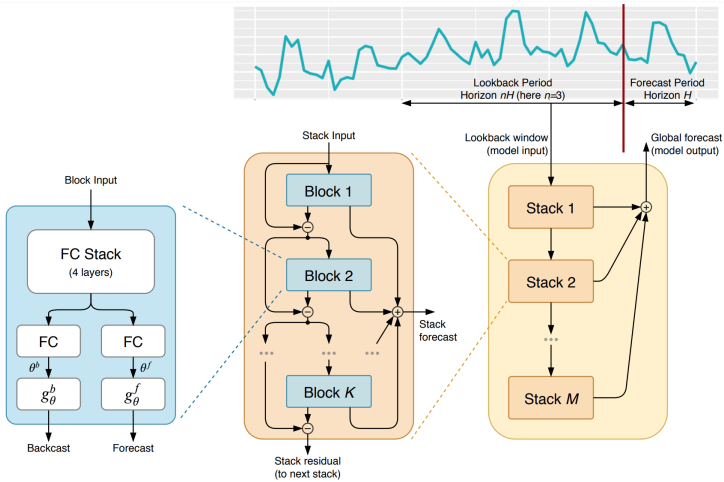


Transformer for Forecasting

- With some tweaks, can be applied to forecasting: Li et al. [2019]

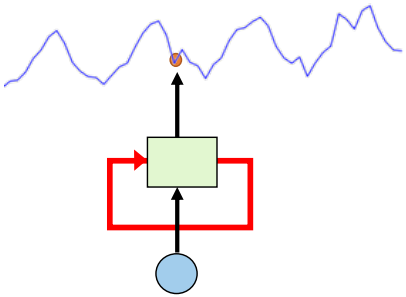


N-BEATS: [Oreshkin et al., 2019]

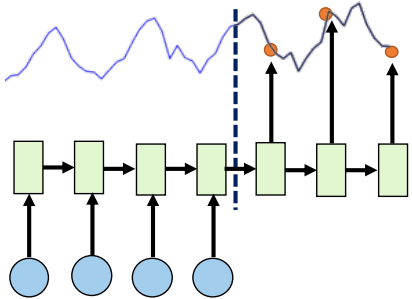


- Generic architecture pure ML model that achieves SOTA on M4

Comparison: Canonical (One-to-One) vs. Seq2Seq (Many-to-Many)



Canonical (One-to-One)



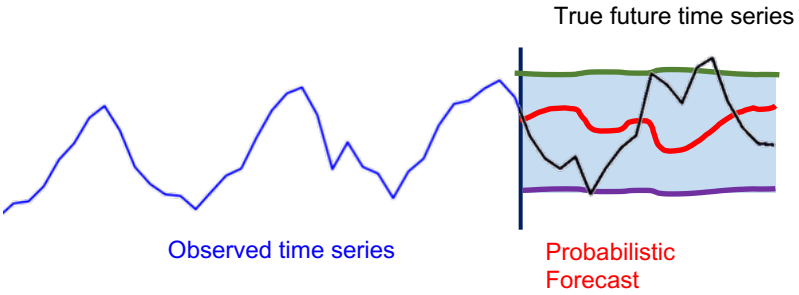
Seq2Seq (Many-to-Many)

Canonical
- input features need to be available during prediction phase
- no need to re-train for different prediction length (forecast horizon)

Seq2Seq
- can have disjoint encoding and decoding features
- needs re-training when changing the decoder length

Probabilistic Forecasts from Neural Nets

Probabilistic Forecasts from Neural Nets



- How can we get probabilistic forecasts from neural network models?

How to Get Probabilistic Forecasts

- Post-hoc: Generate a point forecast first and then use the residuals to estimate a distribution
  - Simple to implement, but more complex to maintain
  - Distribution not taken into account for estimating the model

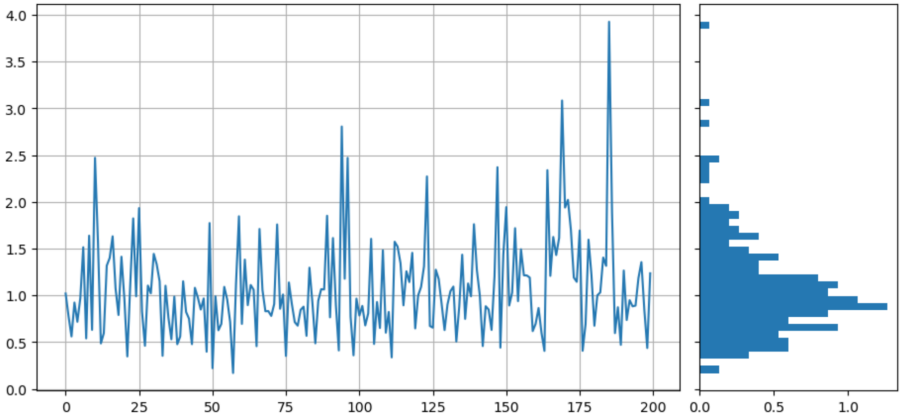
How to Get Probabilistic Forecasts

- Post-hoc: Generate a point forecast first and then use the residuals to estimate a distribution
  - Simple to implement, but more complex to maintain
  - Distribution not taken into account for estimating the model
- Parametric output distributions, e.g. Gaussian
  - Easy to train by maximizing log-likelihood $\sum_t \log p(z_t)$
  - Strong assumptions about the data
  - Cannot directly capture complex distributions
  - Small number of parameters, e.g. μ and σ for a Gaussian distribution

How to Get Probabilistic Forecasts

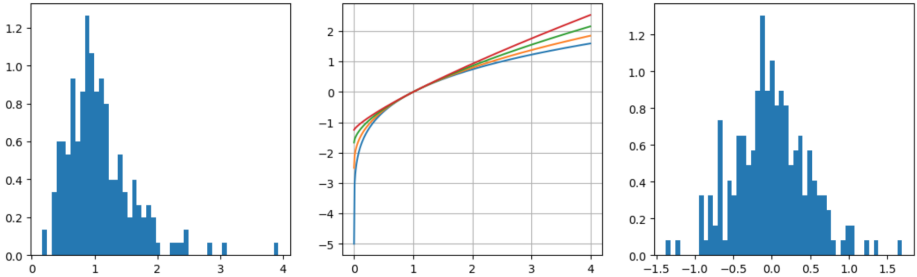
- Post-hoc: Generate a point forecast first and then use the residuals to estimate a distribution
  - Simple to implement, but more complex to maintain
  - Distribution not taken into account for estimating the model
- Parametric output distributions, e.g. Gaussian
  - Easy to train by maximizing log-likelihood $\sum_t \log p(z_t)$
  - Strong assumptions about the data
  - Cannot directly capture complex distributions
  - Small number of parameters, e.g. μ and σ for a Gaussian distribution
- Nonparametric & Semiparametric models
  - No strong assumptions about the data
  - Can capture complex distributions
  - Large number of parameters ($\Longrightarrow$ use a NN)

Input/Output Transformations



- Time series with different magnitudes
- Non-Gaussian data

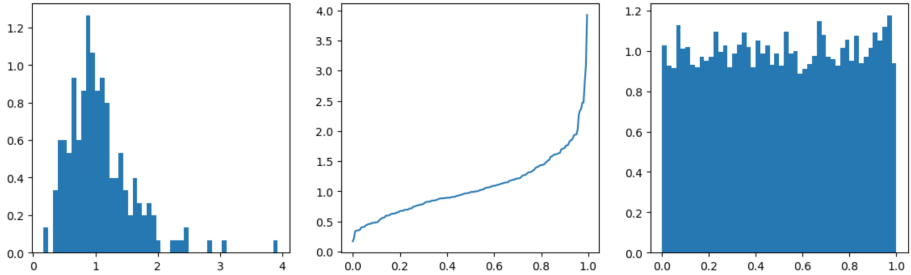
Input/Output Transformations: Box-Cox Transform



- Transform the data to make it closer to a normal distribution

$$y = \frac{x^{\lambda} - 1}{\lambda}$$

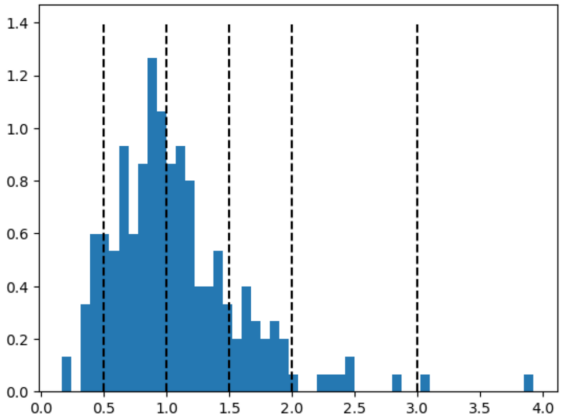
Input/Output Transformations: Probabilty Integral Transform



- Estimate the empirical CDF $\hat{F}$ and use to map the data to the quantile domain

$$y = \hat{F}(x)$$

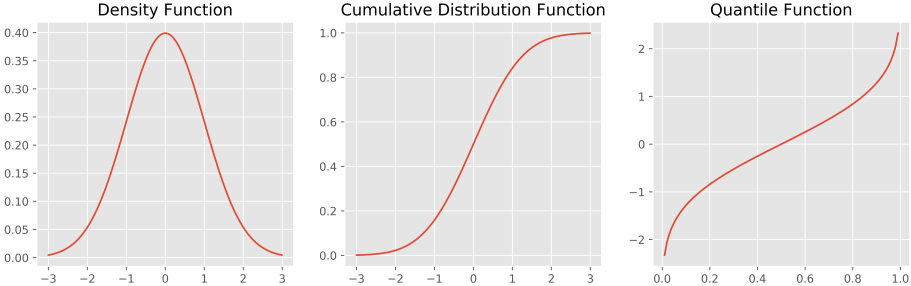
Input/Output Transformations: Binning



- Many variants: global vs. local; equally-spaced vs. quantile binning

Distribution Representations

- Predict a representation of a distribution



- Which representation is more convenient in a nonparametric setting?
  - PDF: should integrate to 1
  - CDF: nondecreasing, bounded in $[0, 1]$
  - Quantile Function: nondecreasing
- The quantile function restrictions are easier to enforce in a NN

Spline Quantile Function, [Gasthaus et al., 2019]

- Approximate Quantile Function with piecewise-linear spline:

$$q(\alpha; \gamma_0, \mathbf{b}, \mathbf{d}) = \gamma_0 + \sum_{l=0}^{L} b_l (\alpha - d_l)_+,$$

$\Rightarrow$ γ_0 is the intercept term

$\Rightarrow$ $\mathbf{b} \in \mathbb{R}^{L+1}$ are weights describing the slopes of the function pieces

$\Rightarrow$ $\mathbf{d} \in \mathbb{R}^{L+1}$ is a vector of "knot" positions

$\Rightarrow$ $(x)_+ = \max(x, 0)$

Spline Quantile Function cont'd

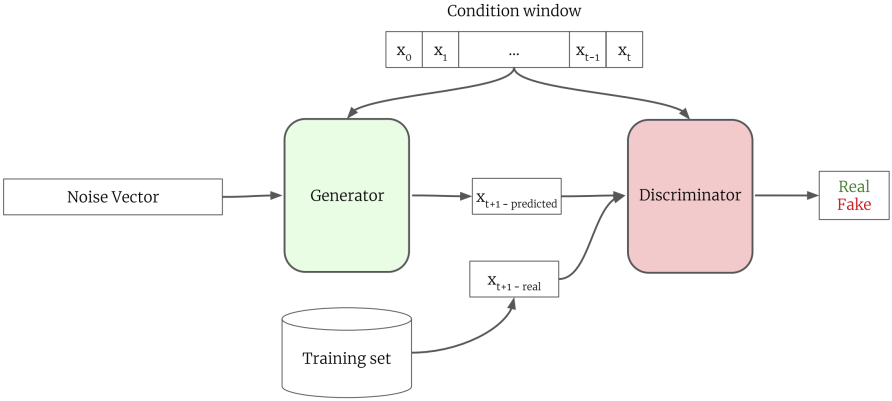
- use Continuous ranked probability score (CRPS) for training:

$$\text{CRPS}(q(\alpha;\theta), z) = \int_0^1 2\Lambda_\alpha(q(\alpha;\theta), z)d\alpha$$

- we can solve this for piecewise-linear splines analytically
- use this on top of a neural forecasting architecture (e.g., WaveNet or DeepAR [Flunkert et al., 2017])

GANs for Forecasting

Generative **A**dversarial **N**etworks, Goodfellow et al. [2014]



Application to forecasting and figure from [Koochali et al., 2019]

So ... Neural Networks, huh?

When to use what for which types of time series?

- MLPs are robust baseline methods, but require heavy feature engineering
- RNNs are, the *de facto* standard model for sequence modeling. Sometimes stability problems in training.
- Recent research [Miller and Hardt, 2018; Bai et al., 2018] advocates that CNNs are as accurate but much more efficient
- But people argue that dilated RNNs [Chang et al., 2017] are just as efficient ...
- Transformer-based methods are also showing promise: [Li et al., 2019]

So ... Neural Networks, huh?

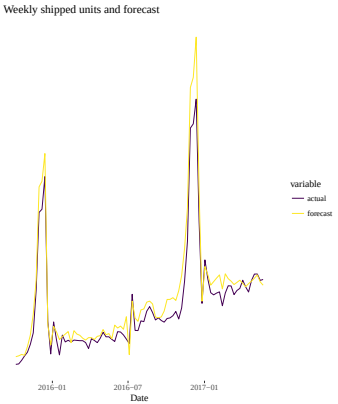
When to use what for which types of time series?

- MLPs are robust baseline methods, but require heavy feature engineering
- RNNs are, the *de facto* standard model for sequence modeling. Sometimes stability problems in training.
- Recent research [Miller and Hardt, 2018; Bai et al., 2018] advocates that CNNs are as accurate but much more efficient
- But people argue that dilated RNNs [Chang et al., 2017] are just as efficient ...
- Transformer-based methods are also showing promise: [Li et al., 2019]

We do not know yet, but we will!

http://quantumfuture.net/quantum_future/homepage.htm

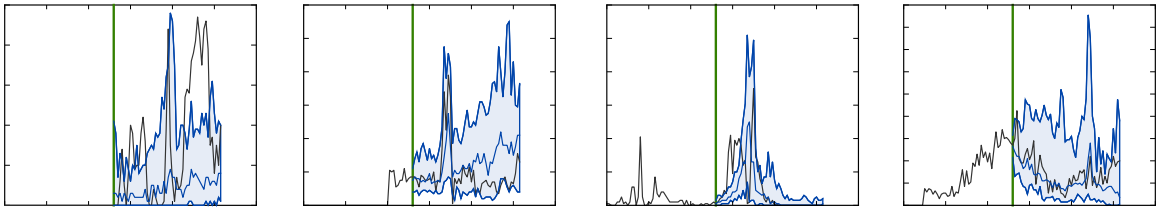
Modern methods struggle with strategic forecasting problems



Predict overall Amazon retail demand years into the future.

Not enough data may be available for training, assumptions on long-term behaviour should be handled properly. Use a classical, local model

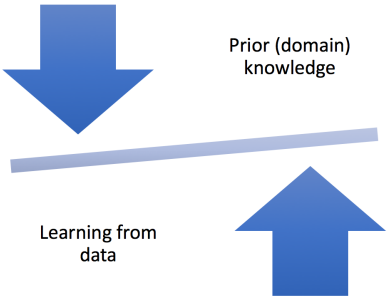
Modern models handle operational forecasting problems well



Predict the demand for a each product available at Amazon

Time series are irregular, only combined to they have enough history and exhibit clear patterns.

Finding the right balance: data vs model driven



Goals:

- increase data efficiency: data efficiency improves sample complexity
- improve interpretability: interpretability facilitates better decision-making
- enforce structure: structure enables fast computation

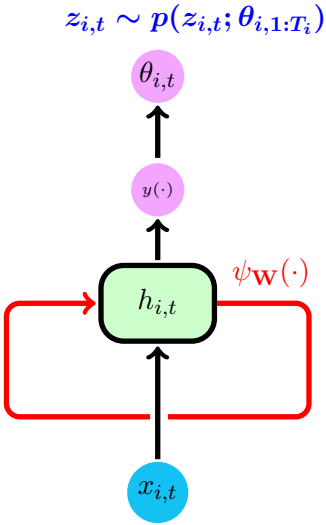
THE BEST OF BOTH WORLDS

HANNAH MONTANA

Deep Probabilistic Models

- Deep State Space Models [Rangapuram et al., 2018]
- Deep Factor Models [Maddix et al., 2018; Wang et al., 2019]

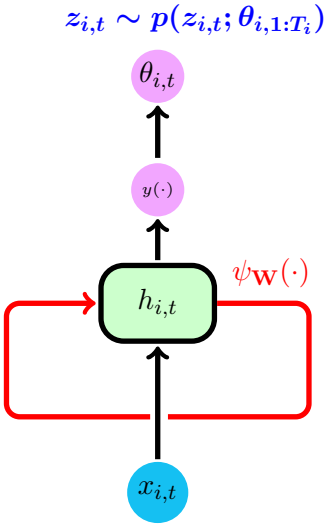
Deep State Space Model in a Nutshell



- State space model as the data generation process

$$z_{i,1:T_i} \sim p_{\mathrm{SSM}}(z_{i,1:T_i}; \Theta_i)$$

Deep State Space Model in a Nutshell



- State space model as the data generation process

$$z_{i,1:T_i} \sim p_{\mathrm{SSM}}(z_{i,1:T_i}; \Theta_i)$$

- Recurrent Neural Network to learn the parameters of the state space model

$$\Theta_i = \mathrm{RNN}(\boldsymbol{x}_i; \mathbf{W})$$

Deep State - Training

- Loss: negative log-likelihood of the data under our model given features

observations $z_{1:T_i}^{(i)}$

$\mathbf{x}_1^{(i)}$ $\cdots$ $\mathbf{x}_t^{(i)}$ $\cdots$ $\mathbf{x}_{T_i}^{(i)}$ features

Deep State - Training

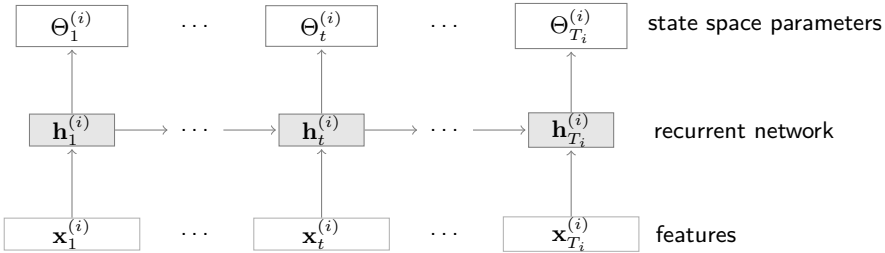
- Loss: negative log-likelihood of the data under our model given features

observations $z_{1:T_i}^{(i)}$

Deep State - Training

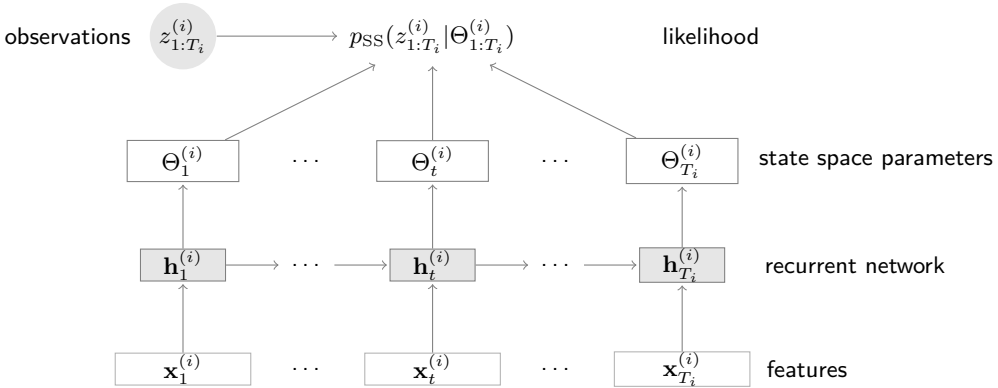
- Loss: negative log-likelihood of the data under our model given features

observations $z_{1:T_i}^{(i)}$



Deep State - Training

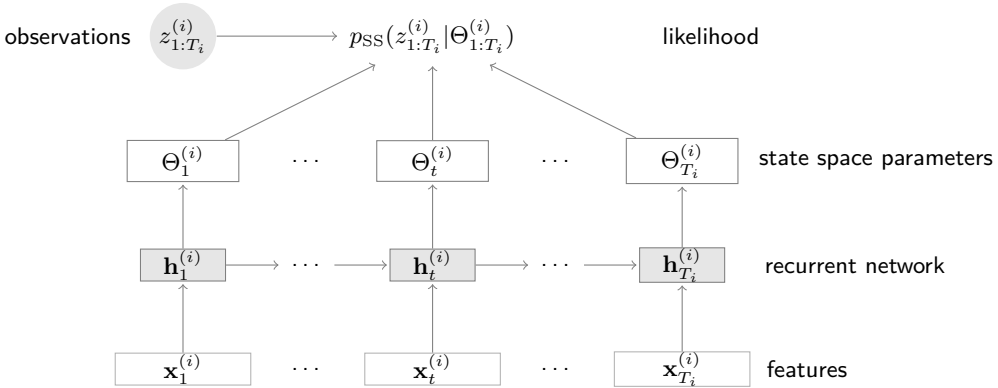
- Loss: negative log-likelihood of the data under our model given features



- Kalman filtering in the case of linear Gaussian model

Deep State - Training

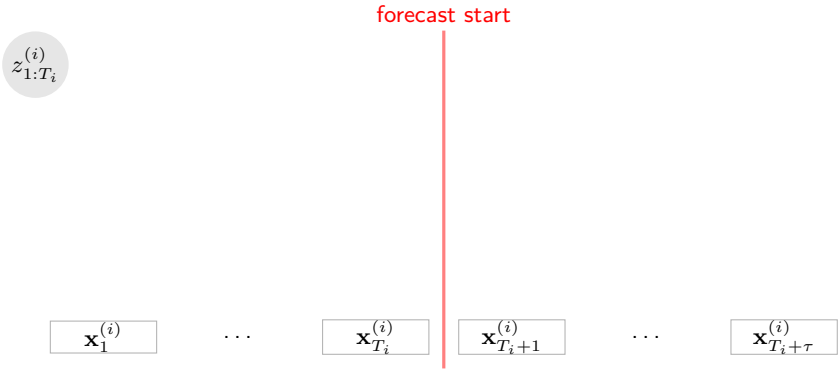
- Loss: negative log-likelihood of the data under our model given features



- Kalman filtering in the case of linear Gaussian model
- Robust to outliers and can handle missing data (z_{t-1} is not fed back as feature for time step t)

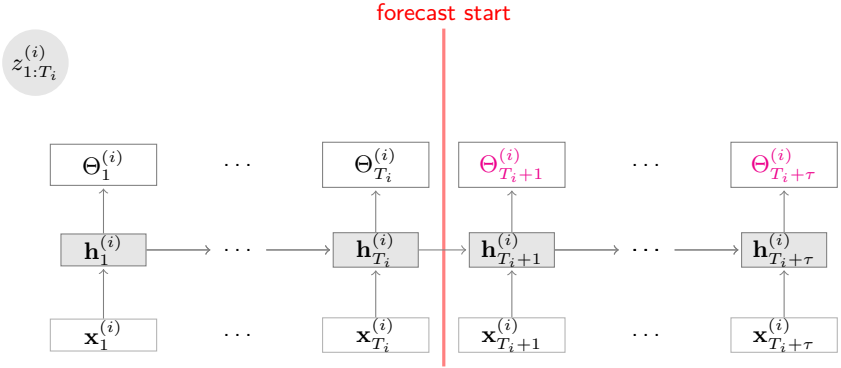
Deep State - Prediction

Test time series (possibly new/unseen) with features in both training and prediction ranges



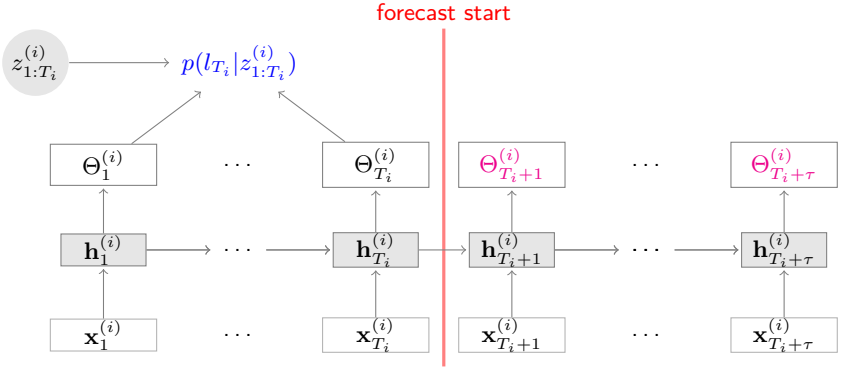
Deep State - Prediction

Test time series (possibly new/unseen) with features in both training and prediction ranges



Deep State - Prediction

Test time series (possibly new/unseen) with features in both training and prediction ranges

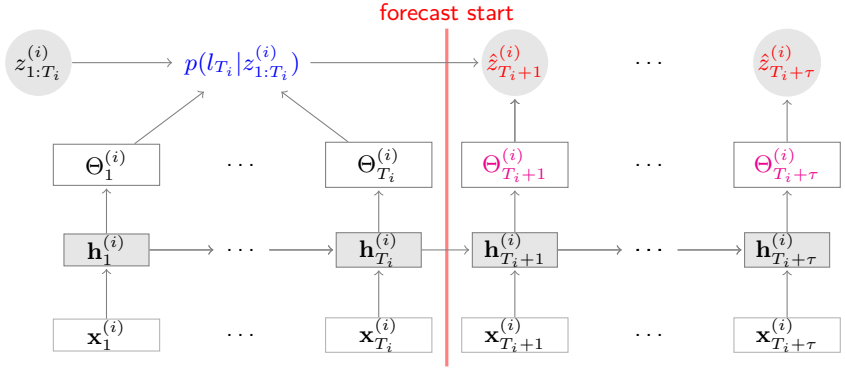


- Given the posterior of the final state and state space parameters, one can obtain the forecast distribution

$$P(z_{T_i+1}, z_{T_i+2}, \ldots, z_{T_i+\tau}|z_1, z_2, \ldots, z_{T_i})$$

Deep State - Prediction

Test time series (possibly new/unseen) with features in both training and prediction ranges



- Given the posterior of the final state and state space parameters, one can obtain the forecast distribution

$$P(z_{T_i+1}, z_{T_i+2}, \ldots, z_{T_i+\tau}|z_1, z_2, \ldots, z_{T_i})$$

Deep Factor Models [Wang et al., 2019]

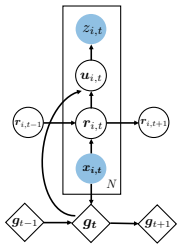
- *Fixed effects*: Linear combinations of K latent deep factors $g_{k,t}$ with their *attention*,

$$f_{i,t} = w_i^\top g_t(x_{i,t})$$

- *Random effects*: Local fluctuations

$$r_i \sim \mathbf{GP}(0, \mathcal{K}_i(\cdot,\cdot)) \quad \text{or} \quad r_i \sim \mathbf{LDS}$$

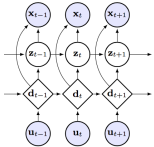
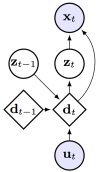
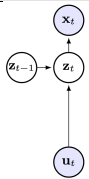
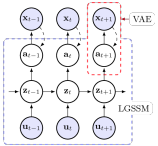
- *Latent function*: $u_{i,t} = f_{i,t} + r_{i,t}$
- *Emission*: $z_{i,t} \sim p(\cdot|u_{i,t})$



Global-local hybrid

- Decompose time series into global and local part
- many options for both parts RNNs, Gaussian Processes
- many open questions
  - which part is more important?
  - why does the optimization work?
  - do we need to regularize more?

We are not the first to realize this...

SRNN [Fraccaro et al., 2016]	VRNN SSL LSTM-LDA [Chung et al., 2015; Zaheer et al., 2017; Zheng et al., 2017]	DMM [Krishnan et al., 2015, 2017]	KVAE DVBF [Fraccaro et al., 2017; Karl et al., 2017]
			

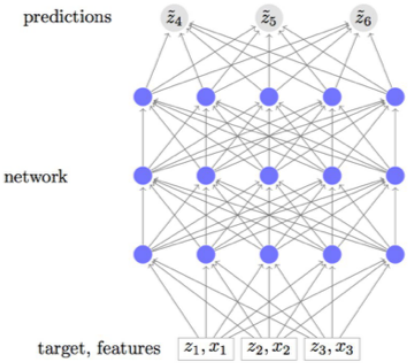
Forecasting Systems

Pecularities of forecasting systems

- Time plays a role: important for backtesting & evaluation
- main primitive type are time series (one dimension more than usual)
- difference between data at train and inference time
- long feedback cycles (e.g., compare with recommender systems)
- complex interaction with downstream decision problems
- traditionally batch system, moving towards on-demand/real-time systems
- B2B not B2C scenario
- users are typically Business Intelligence officers, analysts, data scientists or business functions

Forecasting Systems: Two Extremes

vs

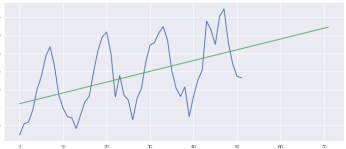


A complex pipeline of simple model vs. a complex model in a simple pipeline.

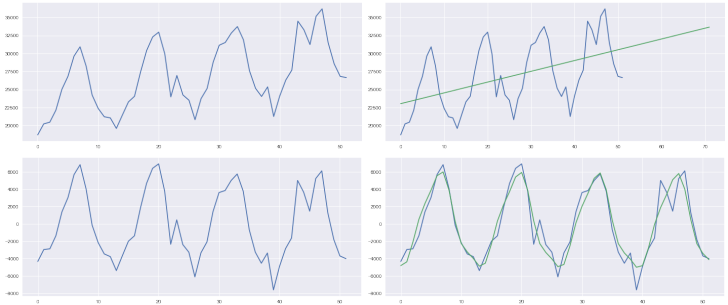
Example: m4 forecasting competition. Won by neural network approach, follow-up by ensemble methods. [Makridakis et al., 2018]

Building Forecasting Systems with Classical Models

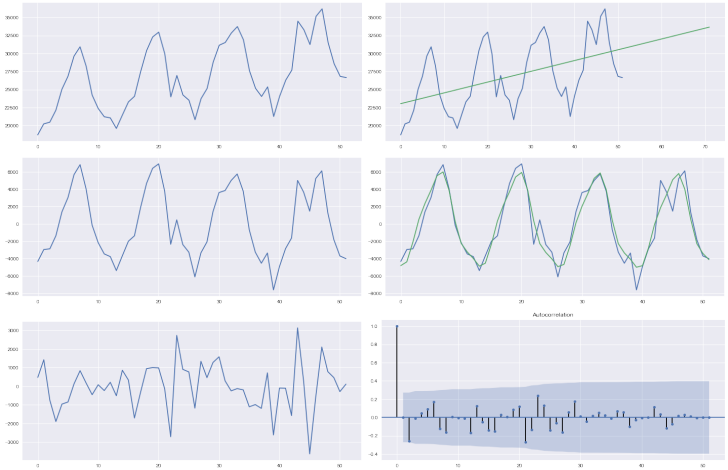
Building Forecasting Systems with Classical Models



Building Forecasting Systems with Classical Models



Building Forecasting Systems with Classical Models



Forecasting Systems with classical models

Image (c) User:Colin / Wikimedia Commons / CC BY-SA 3.0

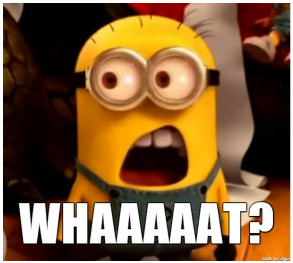
PROS

- models are canonical and relatively easy to understand
- Decomposition $\rightarrow$ decoupling
- White box: explicitly model-based
- Embarassingly parallel

CONS

- Requires lots manual work by experts $\Rightarrow$ hard to tune & maintain
- Cannot learn patterns across time series $\Rightarrow$ pipelines of models must be used
- Cannot handle cold-starts
- Model-based: all effects need to be explicitly modelled

Notable Non-Advantages



interpretability even though each model may be interpretable, the pipeline is not.

running time even though each models runs quickly, entire pipeline does not $\rightarrow$ on-demand forecasting hard to realize.

simple infrastructure forecasting pipeline require complex model combination mechanisms and feature preprocessing.

further things maintainability, tunability, ...

Forecasting System Architecture

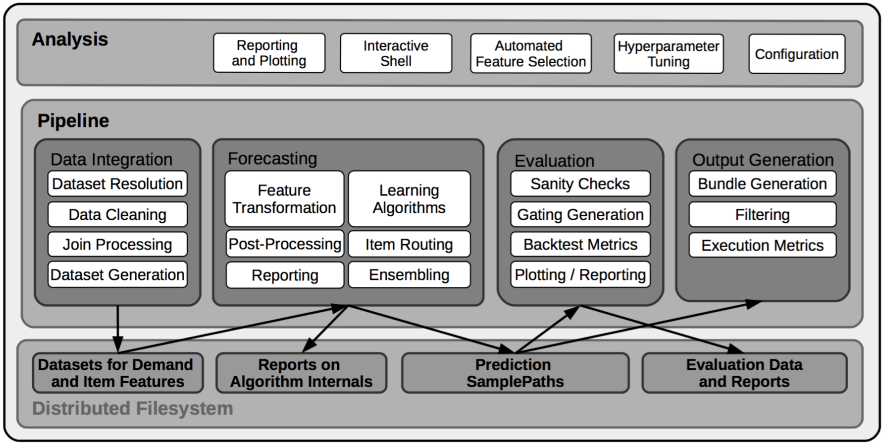
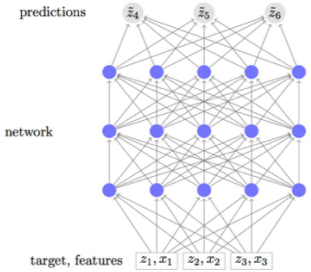


Figure from *Probabilistic Demand Forecasting at Scale* [Böse et al., 2017]

Neural Forecasting Approaches



PROS

- litte feature engineering needed
- learns across time series
- quick at inference
- default settings lead to surprinsingly good results
- state-of-the art performance in competitions

CONS

- little control over predictions
- potentially high-variance in training
- costly to train
- model serving infrastructure needed

Recent public competitions won by neural forecasting approaches: m4 competiton, wikipedia Kaggle competition.

Third Prinicple

Conservation law

Forecasting systems are complex.

Classical forecasting system

Neural forecasting system

Naturally, combinations of both extremes are possible.

Forecasting system are ML systems

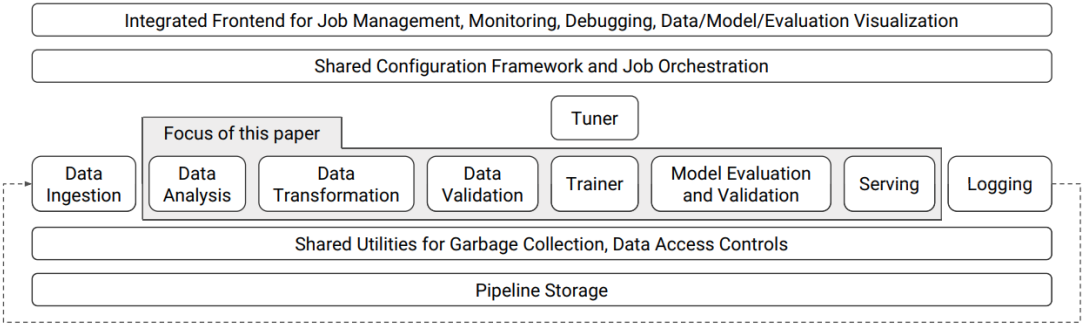


Figure from [Polyzotis et al., 2017]

ML systems: uncomprehensive laundry list of components

- ETL & Data Provenance management
- Data Cleaning, Imputation & monitoring (both for training and inference data)
- feature transformation component
- model training & experiment tracking
- model ensembling
- hyper-parameter optimization & Auto-ML/Meta-Learning
- model serving & management
- model monitoring
- live testing: bandits & A/B tests
- reporting & plotting & notebook
- configuration & orchestration
- ...

Take away: many challenges. See more in [Modi et al., 2017].

Further challenges: Time Series DBs

- many open-source and commercial variants are available
- (relatively) fast-growing market
- active research area (e.g., work by Eamonn Keogh on matrix profile, https://www.cs.ucr.edu/~eamonn/MatrixProfile.html)
- overview article of time series DB Jensen et al. [2017]
- forecasting in DB [Fischer et al., 2012]

Further challenges: Time Series DBs

- many open-source and commercial variants are available
- (relatively) fast-growing market
- active research area (e.g., work by Eamonn Keogh on matrix profile, https://www.cs.ucr.edu/~eamonn/MatrixProfile.html)
- overview article of time series DB Jensen et al. [2017]
- forecasting in DB [Fischer et al., 2012]
- time series models as primitives for time series DBs (e.g., ModelarDB [Jensen et al., 2019], TimeTravel [Khalefa et al., 2012], F^2DB [Fischer et al., 2012].
- natural idea: query these models for future points

Further challenges: Time Series DBs

- many open-source and commercial variants are available
- (relatively) fast-growing market
- active research area (e.g., work by Eamonn Keogh on matrix profile, https://www.cs.ucr.edu/~eamonn/MatrixProfile.html)
- overview article of time series DB Jensen et al. [2017]
- forecasting in DB [Fischer et al., 2012]
- time series models as primitives for time series DBs (e.g., ModelarDB [Jensen et al., 2019], TimeTravel [Khalefa et al., 2012], F^2DB [Fischer et al., 2012].
- natural idea: query these models for future points

Selected References

- TFX: [Modi et al., 2017; Polyzotis et al., 2017]
- Spark-based ML: [Boehm et al., 2016; Meng et al., 2016; Sparks et al., 2017]
- Declarative ML: [Schelter et al., 2016]
- Data verification: [Schelter et al., 2018b]
- Missing data: [Biessmann et al., 2018]
- Model serving: [Crankshaw et al., 2017, 2015]
- Experiment and Meta-Data Tracking: [Schelter et al., 2017]
- Model management: [Schelter et al., 2018a]

Selected References: Forecasting competitions

M4 competition: [Makridakis et al., 2018] (and predecessors)

- Winning entry: https://eng.uber.com/m4-forecasting-competition/

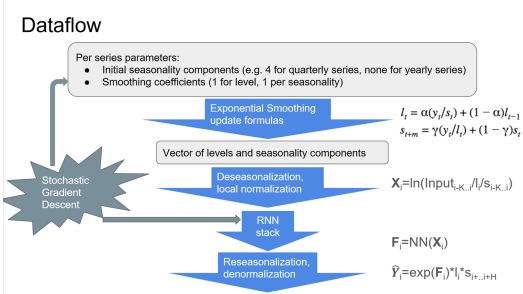
Kaggle competitions on forecasting:

- Rossmann store sales: https://www.kaggle.com/c/rossmann-store-sales
- Wikipedia traffic forecast: https://www.kaggle.com/c/web-traffic-time-series-forecasting

M4 competition

- January-May 2018
- 100,000 series of following frequencies: monthly, quarterly, yearly, daily, weekly, and hourly.
- 95% of series within first 3 categories.
- The forecasting horizons varied, e.g., six for yearly, 18 for monthly, and 48 for hourly series
- point forecasts and prediction intervals were evaluated (95th and 97.5)

Some conclusions



- winning solution: an RNN with integrated exponential smoothing formulae
- ensembles of classical solutions using sophisticated time series feature extraction
- the data set, though critizable, will become very influential

Figure by Slawek Smyl, M4 Forecasting Competition Winning Method, International Symposium on Forecasting, 2019.

Getting Started with Forecasting

Open-source forecasting packages: Classical methods

Rob Hyndman's R package [Hyndman et al., 2007] is among the most popular packages. Contains implementations for many classic methods. Very robust, very hard to beat. You have to like R.

Facebook's Prophet package [Taylor and Letham, 2018] uses Stan [Carpenter et al., 2017] behind the scences. Very flexible but the inference is slow.

Further examples: Tensorflow Probability has Structural Time Series, Bayesian Structural Time Series (BSTS) in R.

Upcoming overview: [Januschowski et al., 2019]

The Amazon ML stack: Broadest & deepest set of capabilities

AI SERVICES

Vision: REKOGNITION IMAGE, REKOGNITION VIDEO

Speech: TEXTRACT, POLLY, TRANSCRIBE

Language: TRANSLATE, COMPREHEND & COMPREHEND MEDICAL

Chatbots: LEX

Forecasting: FORECAST

Recommendations: PERSONALIZE

ML SERVICES

AMAZON SAGEMAKER

BUILD

Pre-built algorithms

Data labeling (GROUND TRUTH)

Algorithms & models (AWS MARKETPLACE FOR MACHINE LEARNING)

Notebook Hosting

DeepAR

TRAIN

One-click model training & tuning

Optimization (NEO)

Hyper Parameter Optimization

Reinforcement learning

DEPLOY

One-click deployment & hosting

Auto-scaling

Virtual Private Cloud

Private Link

Elastic Inference integration

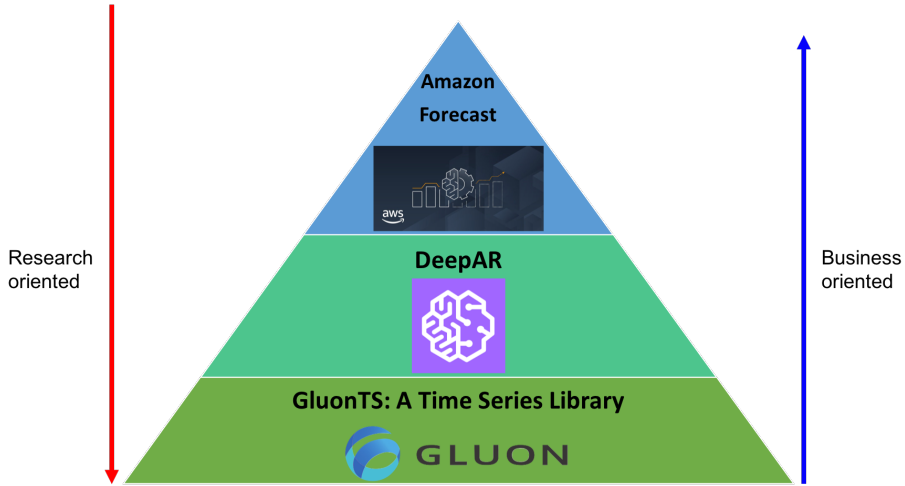
ML FRAMEWORKS & INFRASTRUCTURE

Frameworks: TensorFlow, mxnet, PYTORCH, GluonTS

Interfaces: GLUON, Keras

Infrastructure: EC2 P3 & P3dn, EC2 C5, FPGAs, GREENGRASS, ELASTIC INFERENCE, INFERENTIA

AWS Forecasting Stack

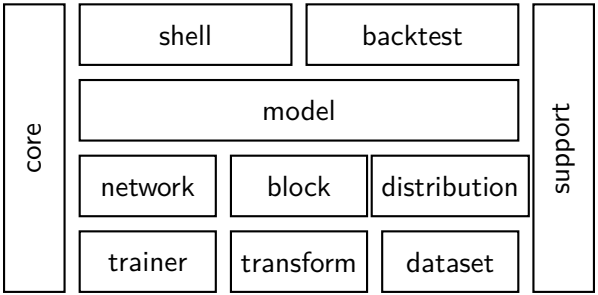


GluonTS

https://github.com/awslabs/gluon-ts and [Alexandrov et al., 2019]

- time series toolkit based on MxNet [Chen et al., 2015]
- comes with pre-built forecasting models
- allows scientists to build new models more quickly

Diagram of the main GluonTS packages



Example Components

Time Series Specific Transformations

- Splitting and padding time series for evaluation splits
- **Box-Cox transformations**
- Marking of special points in time and missing values
- Flexible design to define custom transformations and combine them with existing ones

Distributions

- Real: Gaussian, Student's t, Gamma, ...
- Discrete: Negative Binomial, ...
- Nonparametric: Splines

Metrics

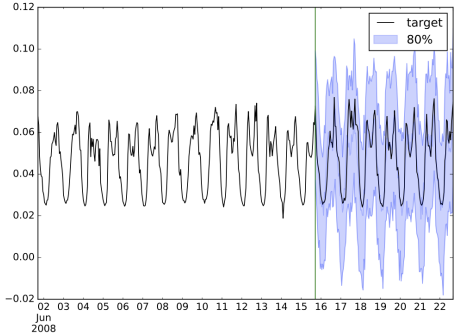
- MASE, sMAPE, MAPE
- Quantile Losses (QL)
- Continuous Ranked Probability Score (CRPS)
- ...

Simple Workflow for Creating, Training and Evaluating Model

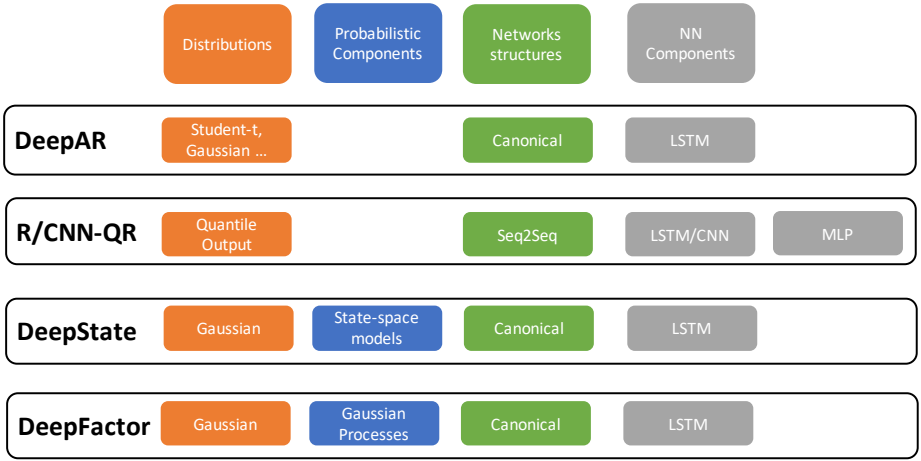
```
from gluonts.dataset.repository.datasets import get_dataset
from gluonts.model.deepar import DeepAREstimator
from gluonts.trainer import Trainer
from gluonts.evaluation import Evaluator
from gluonts.evaluation.backtest import backtest_metrics

meta, train_ds, test_ds = get_dataset("electricity")
estimator = DeepAREstimator(
  freq=meta.time_granularity,
  prediction_length=100,
  trainer=Trainer(epochs=20, batch_size=32)
)
predictor = estimator.train(train_ds)
evaluator = Evaluator(quantiles=(0.1, 0.5, 0.9))
agg_metrics, item_metrics = backtest_metrics(
        train_dataset=train_ds,
        test_dataset=test_ds,
        forecaster=predictor,
        evaluator=evaluator
)
```

Visualization of time series and forecasts using matplotlib

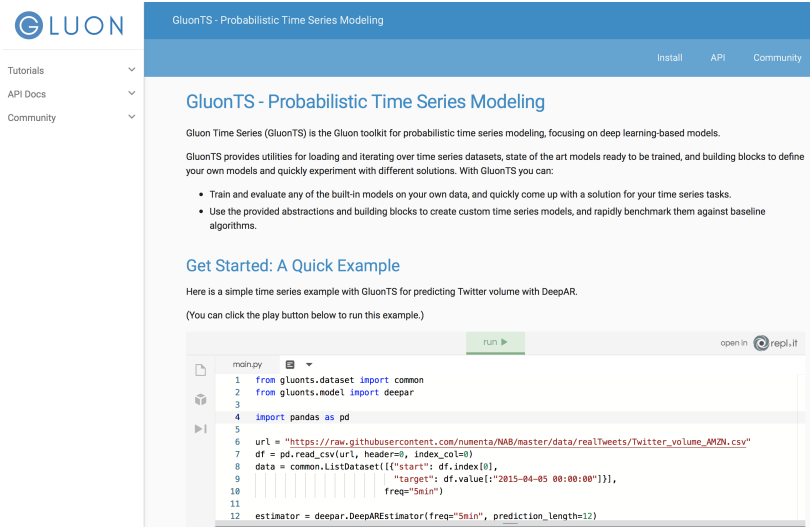
```
pipeline = ... # user-defined
model = ... # user-defined

train_ds, test_ds = split(ds).at('2008-06-16')

estimator = PrototypeEstimator(
    network=network,
    pipeline=pipeline,
    trainer=Trainer(epochs=20)
)
predictor = estimator.train(train_ds)

evaluator = Evaluator(quantiles=(0.1, 0.9))
print(evaluator(predictor, test_ds))

forecasts = predictor.predict(test_ds)
pairs = zip(forecasts, test_ds)
for forecast, ts in islice(pairs, 0, 3):
    plot(forecast, ts).plot()
```



Models Assembled by Different Components



It's Day 1 for GluonTS ...



- more models are coming (e.g., Deep State Space, Deep Factor, etc)
- more flexible backtesting
- more documentation
- more time series applications coming

Forecasting on SageMaker

```
style = {'description_width': 'initial'}
interact_manual(
    plot_interact,
    customer_id=IntSlider(min=0, max=369, value=181, style=style),
    forecast_day=IntSlider(min=0, max=100, value=21, style=style),
    confidence=IntSlider(min=51, max=99, value=80, step=5, style=style),
    show_samples=Checkbox(value=False),
    continuous_update=False
)
```

customer_id 181

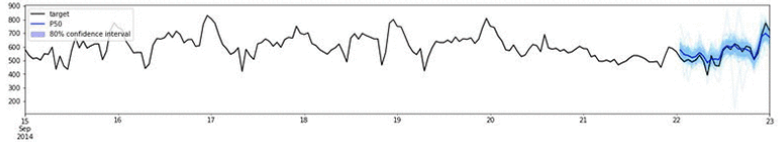
forecast_day 21

confidence 80

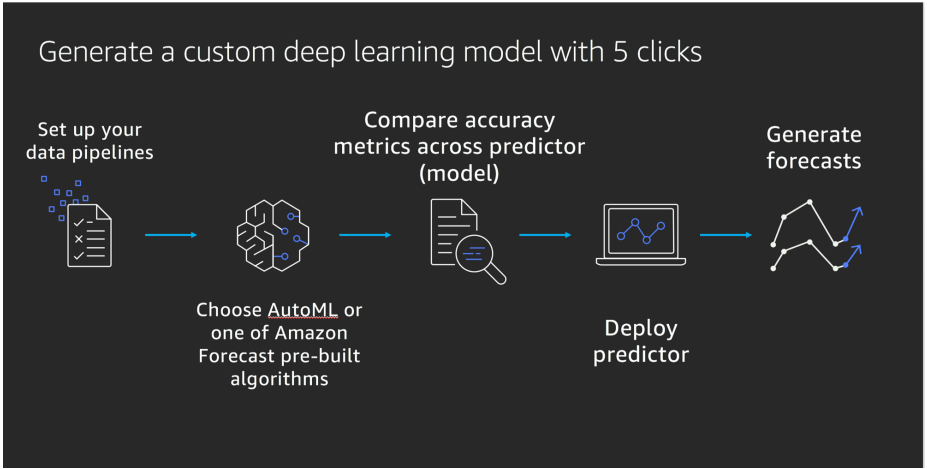
show_samples

Run Interact

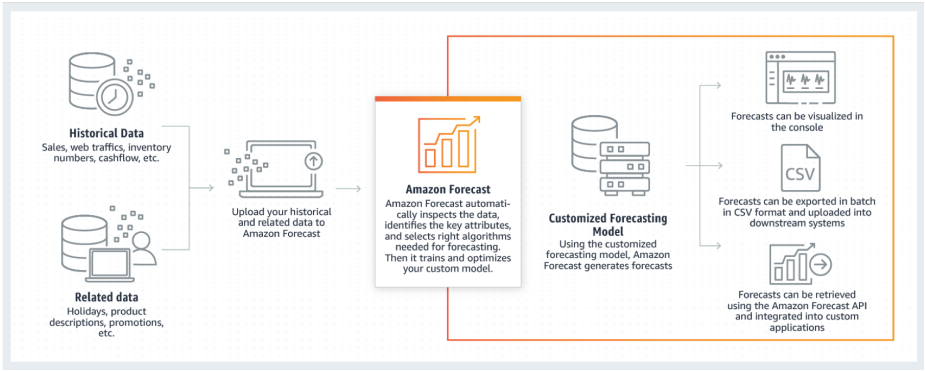
```
calling served model to generate predictions for customer 181 starting at 2014-09-22 00:00:00
done in 1105 ms
```



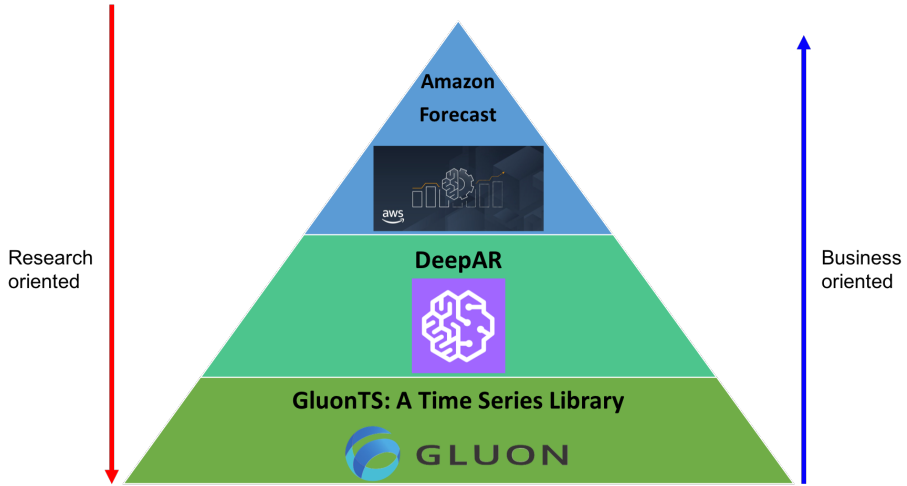
Forecasting as a Service: Amazon Forecast



Forecasting as a Service: Amazon Forecast



Demo: GluonTS, DeepAR (SageMaker) and Amazon Forecast



THANK YOU FOR ATTENDING!





Website: https://lovvge.github.io/Forecasting-Tutorial-KDD-2019/

References

Alexandrov, A., Benidis, K., Bohlke-Schneider, M., Flunkert, V., Gasthaus, J., Januschowski, T., Maddix, D. C., Rangapuram, S., Salinas, D., Schulz, J., Stella, L., TÃijrkmen, A. C., and Wang, Y. (2019). GluonTS: Probabilistic Time Series Modeling in Python. *arXiv preprint arXiv:1906.05264*.

Bai, S., Kolter, J. Z., and Koltun, V. (2018). An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*.

Biessmann, F., Salinas, D., Schelter, S., Schmidt, P., and Lange, D. (2018). Deep learning for missing value imputation in tables with non-numerical data. *CIKM*.

Boehm, M., Dusenberry, M. W., Eriksson, D., Evfimievski, A. V., Manshadi, F. M., Pansare, N., Reinwald, B., Reiss, F. R., Sen, P., Surve, A. C., and Tatikonda, S. (2016). Systemml: Declarative machine learning on spark. *Proc. VLDB Endow.*, 9(13).

Böse, J.-H., Flunkert, V., Gasthaus, J., Januschowski, T., Lange, D., Salinas, D., Schelter, S., Seeger, M., and Wang, Y. (2017). Probabilistic demand forecasting at scale. *PVLDB*, 10(12):1694–1705.

Carpenter, B., Gelman, A., Hoffman, M. D., Lee, D., Goodrich, B., Betancourt, M., Brubaker, M., Guo, J., Li, P., and Riddell, A. (2017). Stan: A probabilistic programming language. *Journal of statistical software*, 76(1).

Chang, S., Zhang, Y., Han, W., Yu, M., Guo, X., Tan, W., Cui, X., Witbrock, M., Hasegawa-Johnson, M. A., and Huang, T. S. (2017). Dilated recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 77–87.

Chen, T., Li, M., Li, Y., Lin, M., Wang, N., Wang, M., Xiao, T., Xu, B., Zhang, C., and Zhang, Z. (2015). Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems. *CoRR*, abs/1512.01274.

References (cont.)

Crankshaw, D., Bailis, P., Gonzalez, J. E., Li, H., Zhang, Z., Franklin, M. J., Ghodsi, A., and Jordan, M. I. (2015). The missing piece in complex analytics: Low latency, scalable model management and serving with velox. In *CIDR 2015, Seventh Biennial Conference on Innovative Data Systems Research, Asilomar, CA, USA, January 4-7, 2015, Online Proceedings*.

Crankshaw, D., Wang, X., Zhou, G., Franklin, M. J., Gonzalez, J. E., and Stoica, I. (2017). Clipper: A low-latency online prediction serving system. In *14th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2017, Boston, MA, USA, March 27-29, 2017*, pages 613–627.

Fischer, U., Rosenthal, F., and Lehner, W. (2012). F2db: The flash-forward database system. In *2012 IEEE 28th International Conference on Data Engineering*.

Flunkert, V., Salinas, D., Gasthaus, J., and Januschowski, T. (2017). Deepar: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting, arXiv:1704.04110*.

Fraccaro, M., Kamronn, S., Paquet, U., and Winther, O. (2017). A disentangled recognition and nonlinear dynamics model for unsupervised learning. In *Advances in Neural Information Processing Systems*, pages 3604–3613.

Fraccaro, M., Sønderby, S. K., Paquet, U., and Winther, O. (2016). Sequential neural models with stochastic layers. In *Advances in neural information processing systems*, pages 2199–2207.

Gasthaus, J., Benidis, K., Wang, Y., Rangapuram, S. S., Salinas, D., Flunkert, V., and Januschowski, T. (2019). Probabilistic forecasting with spline quantile function RNNs. In Chaudhuri, K. and Sugiyama, M., editors, *Proceedings of Machine Learning Research*, volume 89 of *Proceedings of Machine Learning Research*, pages 1901–1910. PMLR.

References (cont.)

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In *NIPS*, pages 2672–2680.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

Hyndman, R. J., Khandakar, Y., et al. (2007). *Automatic time series for forecasting: the forecast package for R*. Number 6/07. Monash University, Department of Econometrics and Business Statistics.

Januschowski, T., Gasthaus, J., and Wang, Y. (2019). Open-source forecasting tools in python. *Foresight. The Applied Forecasting journal*.

Jensen, S. K., Pedersen, T. B., and Thomsen, C. (2017). Time series management systems: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 29(11):2581–2600.

Jensen, S. K., Pedersen, T. B., and Thomsen, C. (2019). Demonstration of modelardb: Model-based management of dimensional time series. In *Proceedings of the 2019 International Conference on Management of Data*, SIGMOD '19, pages 1933–1936, New York, NY, USA. ACM.

Karl, M., Soelch, M., Bayer, J., and van der Smagt, P. (2017). Deep variational bayes filters: Unsupervised learning of state space models from raw data. *ICLR*.

Khalefa, M. E., Fischer, U., Pedersen, T. B., and Lehner, W. (2012). Model-based integration of past & future in timetravel. *Proc. VLDB Endow.*

Koochali, A., Schichtel, P., Ahmed, S., and Dengel, A. (2019). Probabilistic forecasting of sensory data with generative adversarial networks - forgan. *CoRR*, abs/1903.12549.

References (cont.)

Krishnan, R. G., Shalit, U., and Sontag, D. (2015). Deep kalman filters. *arXiv preprint arXiv:1511.05121*.

Krishnan, R. G., Shalit, U., and Sontag, D. (2017). Structured inference networks for nonlinear state space models. In *AAAI*, pages 2101–2109.

Li, S., Jin, X., Xuan, Y., Zhou, X., Chen, W., Wang, Y.-X., and Yan, X. (2019). Enhancing the Locality and Breaking the Memory Bottleneck of Transformer on Time Series Forecasting. *arXiv e-prints*, page arXiv:1907.00235.

Li, Y., Yu, R., Shahabi, C., and Liu, Y. (2018). Diffusion convolutional recurrent neural network: Data-driven traffic forecasting.

Maddix, D. C., Wang, Y., and Smola, A. (2018). Deep factors with gaussian processes for forecasting. In *Workshop on Bayesian Deep Learning, NIPS*.

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

Meng, X., Bradley, J., Yavuz, B., Sparks, E., Venkataraman, S., Liu, D., Freeman, J., Tsai, D., Amde, M., Owen, S., Xin, D., Xin, R., Franklin, M. J., Zadeh, R., Zaharia, M., and Talwalkar, A. (2016). Mllib: Machine learning in apache spark. *Journal of Machine Learning Research*, 17(34):1–7.

Miller, J. and Hardt, M. (2018). When recurrent models don't need to be recurrent. *arXiv preprint arXiv:1805.10369*.

Modi, A. N., Koo, C. Y., Foo, C. Y., Mewald, C., Baylor, D. M., Breck, E., Cheng, H.-T., Wilkiewicz, J., Koc, L., Lew, L., Zinkevich, M. A., Wicke, M., Ispir, M., Polyzotis, N., Fiedel, N., Haykal, S. E., Whang, S., Roy, S., Ramesh, S., Jain, V., Zhang, X., and Haque, Z. (2017). Tfx: A tensorflow-based production-scale machine learning platform. In *KDD 2017*.

References (cont.)

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

Oreshkin, B. N., Carpov, D., Chapados, N., and Bengio, Y. (2019). N-BEATS: Neural basis expansion analysis for interpretable time series forecasting.

Polyzotis, A., Zinkevich, M. A., Whang, S., and Roy, S. (2017). Data management challenges in production machine learning. pages 1723–1726.

Rangapuram, S. S., Seeger, M., Gasthaus, J., Stella, L., Wang, Y., and Januschowski, T. (2018). Deep state space models for time series forecasting. In *Advances in Neural Information Processing Systems*.

Schelter, S., Bießmann, F., Januschowski, T., Salinas, D., Seufert, S., and Szarvas, G. (2018a). On challenges in machine learning model management. *IEEE Data Eng. Bull.*, 41(4):5–15.

Schelter, S., Boese, J.-H., Kirschnick, J., Klein, T., and Seufert, S. (2017). Automatically tracking metadata and provenance of machine learning experiments. In *NIPS Workshop ML Systems*.

Schelter, S., Lange, D., Schmidt, P., Celikel, M., Biessmann, F., and Grafberger, A. (2018b). Automating large-scale data quality verification. *PVLDB*, 11.

Schelter, S., Palumbo, A., Quinn, S., Marthi, S., and Musselman, A. (2016). Samsara: Declarative machine learning on distributed dataflow systems. In *NIPS Workshop MLSystems*.

Sparks, E. R., Venkataraman, S., Kaftan, T., Franklin, M. J., and Recht, B. (2017). Keystoneml: Optimizing pipelines for large-scale advanced analytics. In *2017 IEEE 33rd International Conference on Data Engineering (ICDE)*.

References (cont.)

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

Van Den Oord, A., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A. W., and Kavukcuoglu, K. (2016). Wavenet: A generative model for raw audio. In *SSW*, page 125.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. *CoRR*, abs/1706.03762.

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, Long Beach, California, USA. PMLR.

Wen, R., Torkkola, K., and Narayanaswamy, B. (2017). A multi-horizon quantile recurrent forecaster. *NIPS Workshop on Time Series, arXiv:1711.11053*.

Yu, R., Li, Y., Shahabi, C., Demiryurek, U., and Liu, Y. (2017). Deep learning: A generic approach for extreme condition traffic forecasting. In *Proceedings of the 2017 SIAM International Conference on Data Mining*, pages 777–785. SIAM.