



Amazon AI

Deep Learning for Forecasting

ISF: Deep Learning Workshop

Lorenzo Stella, Tim Januschowski

AWS AI Labs



<https://github.com/awslabs/gluon-ts>

October 29, 2020



Acknowledgement



Konstantinos Benidis



Syama Rangapuram



Michael Bohlke-Schneider



Laurent Callot



Peihong Jiang

Acknowledgement



Konstantinos Benidis



Syama Rangapuram



Michael Bohlke-Schneider



Laurent Callot



Peihong Jiang

and many more!

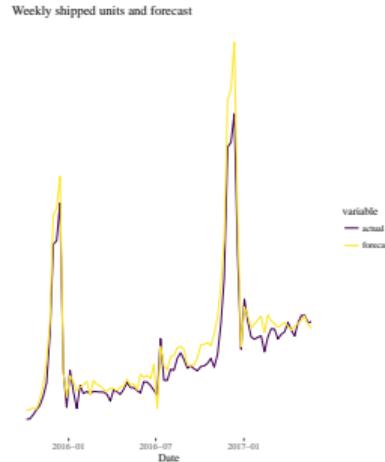
Forecasting at AWS AI Labs

We look after ...

- DeepAR in Amazon SageMaker
<https://docs.aws.amazon.com/sagemaker/latest/dg/deepar.html>
- Amazon Forecast <https://aws.amazon.com/forecast/>
- GluonTS <https://github.com/awslabs/gluon-ts>
- Amazon internal applications like Labor Planning for Fulfillment Centers
<https://dl.acm.org/doi/abs/10.1145/3399579.3399869>
- and more!

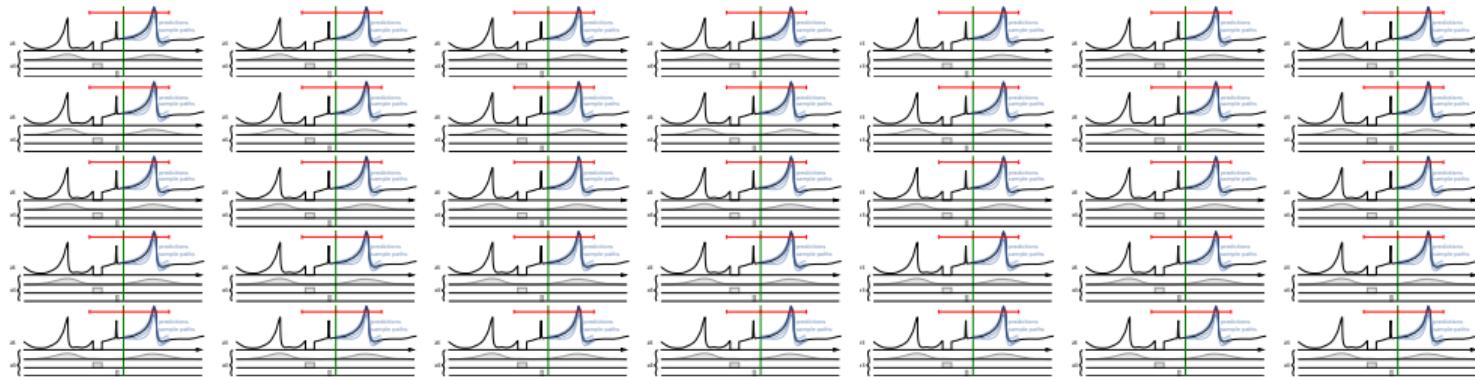
```
pip install --upgrade mxnet==1.6.0 torch==1.6.0
pip install git+https://github.com/awslabs/gluon-ts.git
```

Neural Networks: no silver bullet



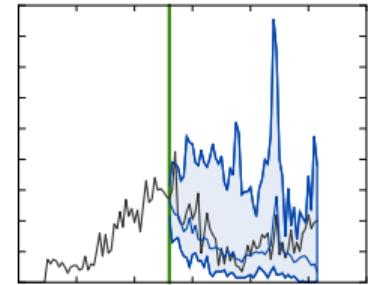
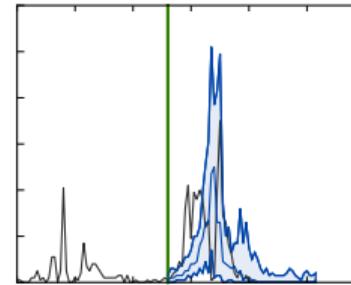
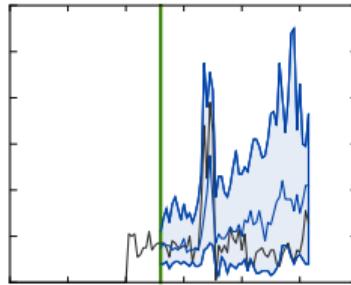
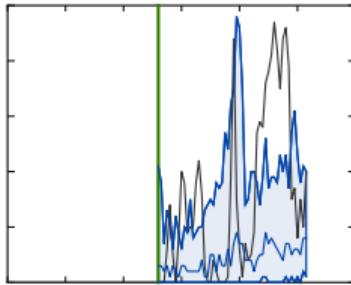
- Forecasting Problem: predict overall Amazon retail demand years into the future.
- Decision Problems: topology planning, market entry/segment analyses
- ⇒ we call them *strategic* forecasting problems
- neural networks probably won't be the first/best/appropriate/... method: not enough "control", over fitting, hard to fit in domain knowledge, etc.

Contemporary business forecasting problems



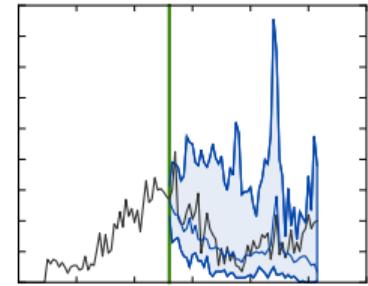
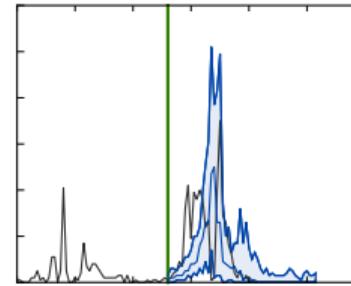
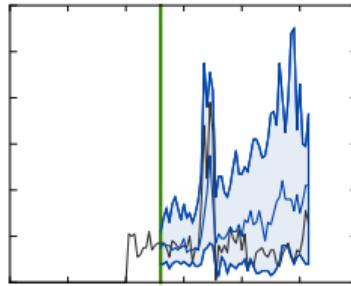
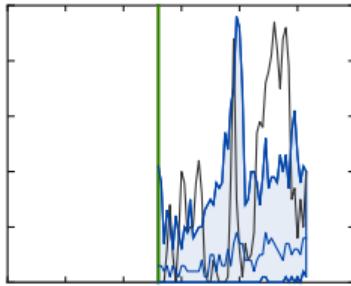
- Examples: demand for products of a retailer, workforce cohorts of a company in its locations, compute capacity needs per region and server type
- ⇒ we call them *operational* forecasting problems
- neural networks probably a strong contender: patterns are complex but enough data is available to learn them, so no over fitting concerns; too many time series and edge cases to design specific methods for, etc.

Operational forecasting problems: characteristics & approaches



- cold start/new items
- short cycles
- burstiness, sparsity
- high degree of automation of downstream systems such as inventory management or scheduling systems
- ratio of people/time series $\ll 1$

Operational forecasting problems: characteristics & approaches



- cold start/new items
- short cycles
- burstiness, sparsity
- high degree of automation of downstream systems such as inventory management or scheduling systems
- ratio of people/time series $\ll 1$

Why are neural networks good?

- easy to learn across multiple time series (*global* models)
- little manual feature engineering needed
- simplify (software) system

Structure of the Workshop (3h)

- Deep Learning (50 mins)
 - Gentle Intro: from linear methods to deep learning
 - Optimization challenges
 - Different architectures for different problem types
 - Probabilistic prediction
 - Q&A (10 mins)
- break (10 mins)
- Deep Learning for Forecasting (50 mins)
 - Discriminative vs Generative Models
 - Deep dive into DeepAR
 - Categorization of state-of-the-art
 - Results
 - Q&A (10 mins)
- break (10 mins)
- Practical session (30 mins)

Structure of the Workshop (3h)

- Deep Learning (50 mins)
 - Gentle Intro: from linear methods to deep learning
 - Optimization challenges
 - Different architectures for different problem types
 - Probabilistic prediction
 - Q&A (10 mins)
- break (10 mins)
- Deep Learning for Forecasting (50 mins)
 - Discriminative vs Generative Models
 - Deep dive into DeepAR
 - Categorization of state-of-the-art
 - Results
 - Q&A (10 mins)
- break (10 mins)
- Practical session (30 mins)
- this doesn't add up to 3h! → overflow

Goals

- understand about the basics of deep learning [first part]
- if you know deep learning, understand how to get probabilistic predictions [end of first part]
- in-depth discussion of two simple neural networks architectures
- some hands-on experience [practical part & self study]

what we won't do here:

- broad survey of the state-of-the-art →
<https://lovvge.github.io/Forecasting-Tutorial-WWW-2020/>
- discuss other machine learning techniques like gradient boosted trees

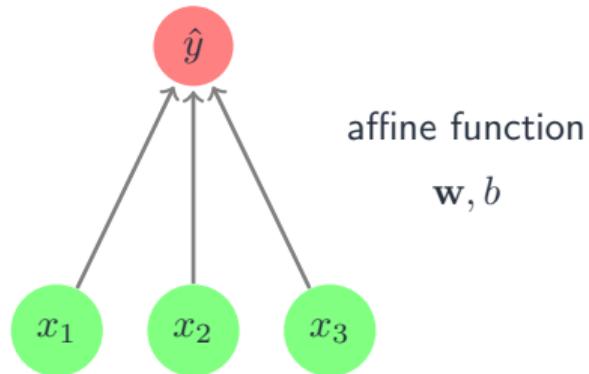
Part I: Deep Learning

From Linear Methods to Deep Learning

Linear methods: $D = \{\mathbf{x}_i, y_i\}_{i=1}^n, \quad \hat{y} = \mathbf{w}^T \mathbf{x} + b$

From Linear Methods to Deep Learning

Linear methods: $D = \{\mathbf{x}_i, y_i\}_{i=1}^n, \hat{y} = \mathbf{w}^T \mathbf{x} + b$



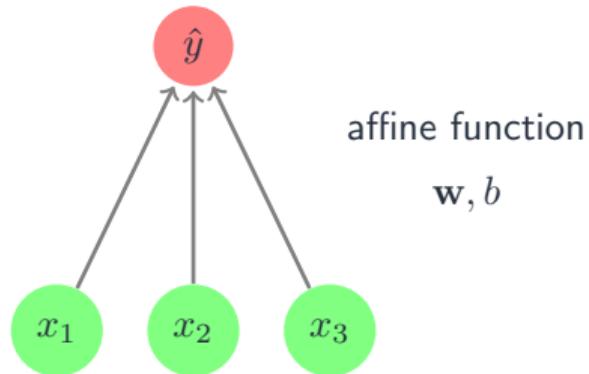
Output: Linear in both inputs \mathbf{x} and weights w (modulo shift b)

$$\hat{y} = \mathbf{w}^T \mathbf{x} + b$$

Inputs: $\mathbf{x} = \{x_1, x_2, x_3\}$

From Linear Methods to Deep Learning

Linear methods: $D = \{\mathbf{x}_i, y_i\}_{i=1}^n, \hat{y} = \mathbf{w}^T \mathbf{x} + b$



Output: Linear in both inputs \mathbf{x} and weights w (modulo shift b)

$$\hat{y} = \mathbf{w}^T \mathbf{x} + b$$

Inputs: $\mathbf{x} = \{x_1, x_2, x_3\}$

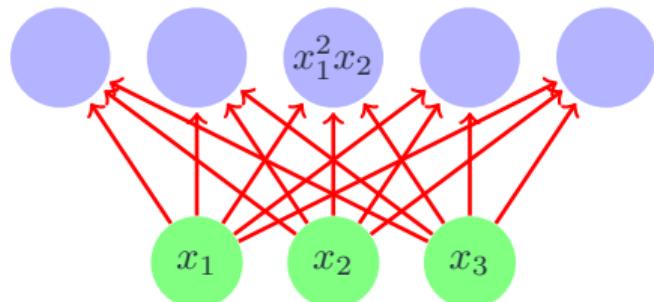
Estimation/Optimization: Analytical solution for the unknown w depending on the loss

Linear Methods with Non-Linear Features



Inputs: $\mathbf{x} = \{x_1, x_2, x_3\}$

Linear Methods with Non-Linear Features

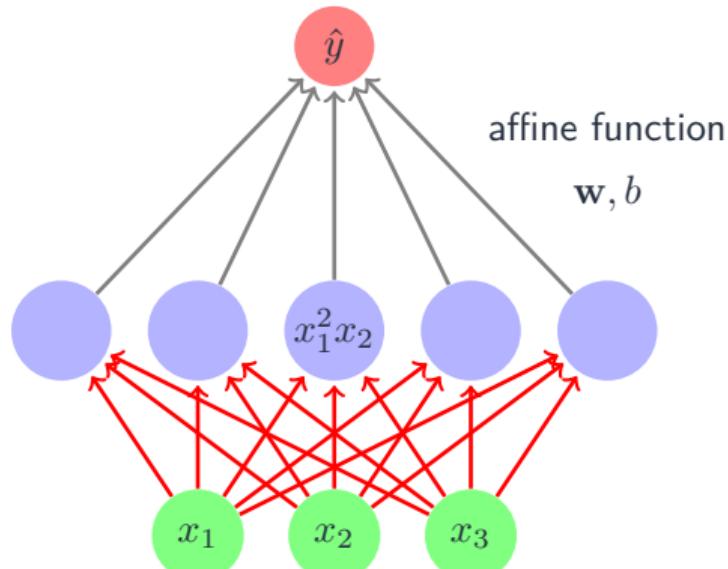


Manually engineered features:

$$\Phi(x) = [x_1, x_2, x_1^2 x_2, x_1 x_3, x_3]$$

Inputs: $\mathbf{x} = \{x_1, x_2, x_3\}$

Linear Methods with Non-Linear Features



Output: Linear model of non-linear features Φ

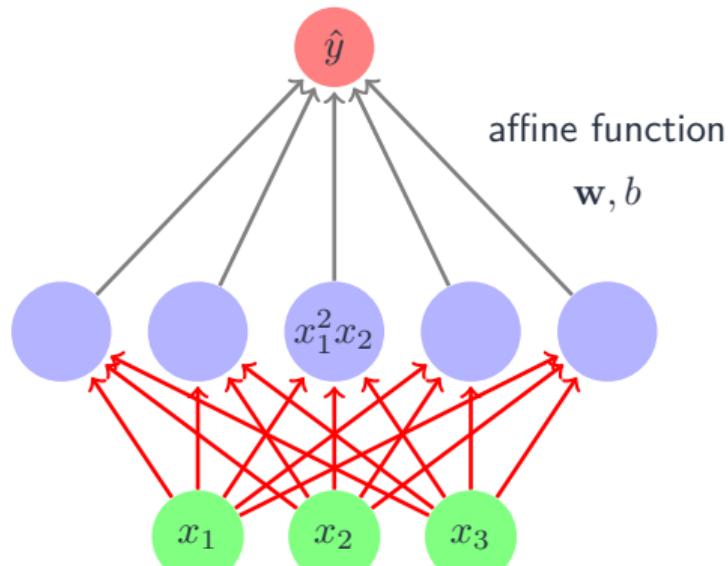
$$\hat{y} = \mathbf{w}^T \Phi(\mathbf{x}) + b$$

Manually engineered features:

$$\Phi(\mathbf{x}) = [x_1, x_2, x_1^2 x_2, x_1 x_3, x_3]$$

Inputs: $\mathbf{x} = \{x_1, x_2, x_3\}$

Linear Methods with Non-Linear Features



Output: Linear model of non-linear features Φ

$$\hat{y} = \mathbf{w}^T \Phi(\mathbf{x}) + b$$

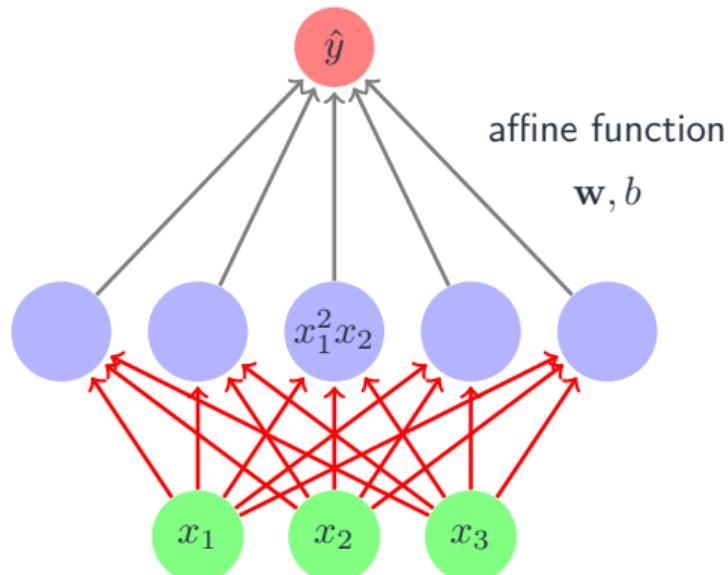
Manually engineered features:

$$\Phi(\mathbf{x}) = [x_1, x_2, x_1^2 x_2, x_1 x_3, x_3]$$

Inputs: $\mathbf{x} = \{x_1, x_2, x_3\}$

Estimation/Optimization: still the same as linear models

Linear Methods with Non-Linear Features



Output: Linear model of non-linear features Φ

$$\hat{y} = \mathbf{w}^T \Phi(\mathbf{x}) + b$$

Manually engineered features:

$$\Phi(\mathbf{x}) = [x_1, x_2, x_1^2x_2, x_1x_3, x_3]$$

Inputs: $\mathbf{x} = \{x_1, x_2, x_3\}$

Estimation/Optimization: still the same as linear models

Kernel Methods: Non-linear feature map via a positive definite kernel

Neural Networks (Non-Linear Methods)

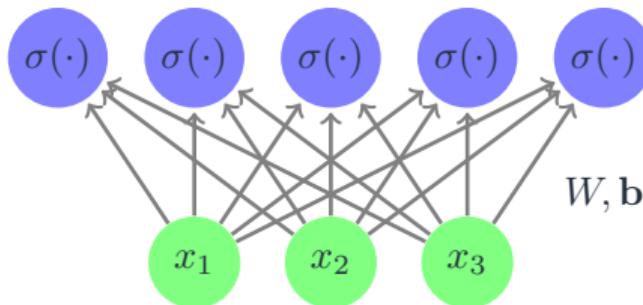
Why not learn the non-linear features?



Inputs: $\mathbf{x} = \{x_1, x_2, x_3\}$

Neural Networks (Non-Linear Methods)

Why not learn the non-linear features?



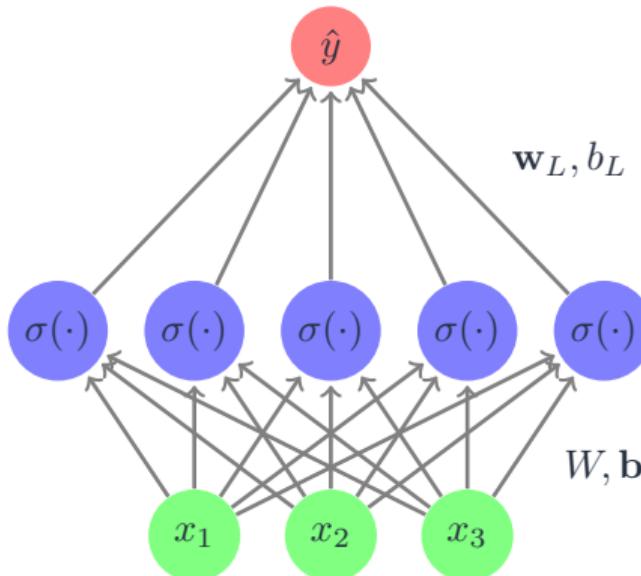
Learned features: Affine functions of inputs followed by a non-linear activation

$$\phi_j(\mathbf{x}; W_j, b_j) = \sigma(W_j^\top \mathbf{x} + b_j)$$

Inputs: $\mathbf{x} = \{x_1, x_2, x_3\}$

Neural Networks (Non-Linear Methods)

Why not learn the non-linear features?



Output: Linear model of (learned) non-linear features Φ

$$\hat{y} = \mathbf{w}_L^T \Phi(\mathbf{x}; W, \mathbf{b}) + b_L$$

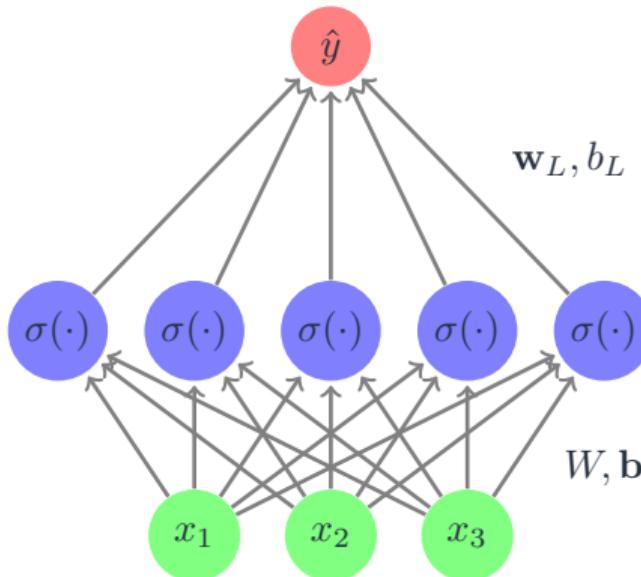
Learned features: Affine functions of inputs followed by a non-linear activation

$$\phi_j(\mathbf{x}; W_j, b_j) = \sigma(W_j^\top \mathbf{x} + b_j)$$

Inputs: $\mathbf{x} = \{x_1, x_2, x_3\}$

Neural Networks (Non-Linear Methods)

Why not learn the non-linear features?



Output: Linear model of (learned) non-linear features Φ

$$\hat{y} = \mathbf{w}_L^T \Phi(\mathbf{x}; W, \mathbf{b}) + b_L$$

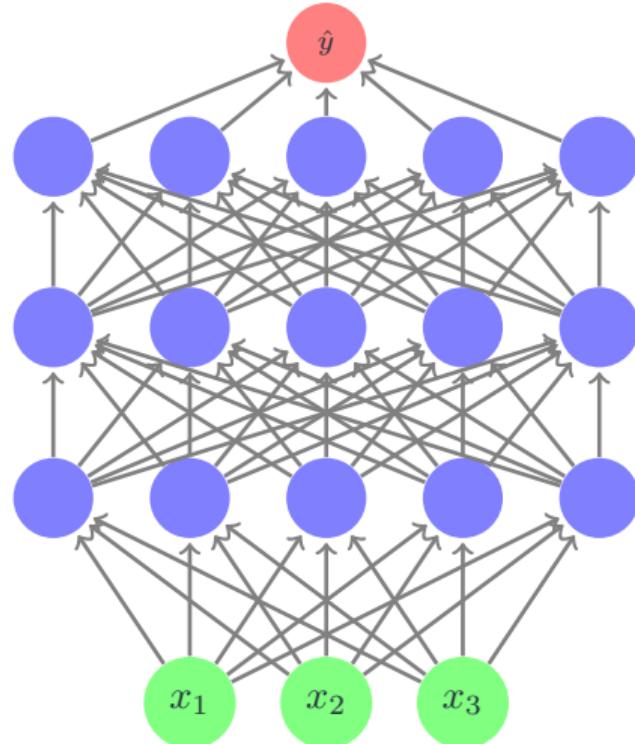
Learned features: Affine functions of inputs followed by a non-linear activation

$$\phi_j(\mathbf{x}; W_j, b_j) = \sigma(W_j^\top \mathbf{x} + b_j)$$

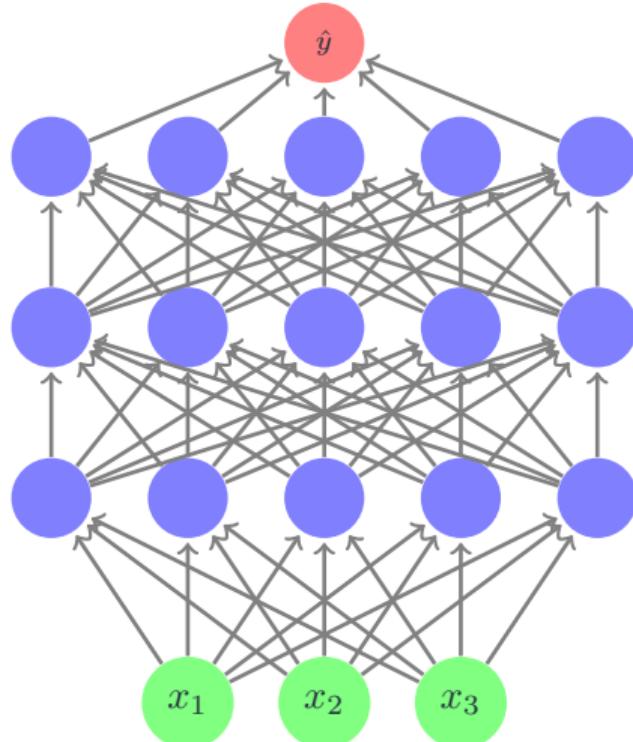
Inputs: $\mathbf{x} = \{x_1, x_2, x_3\}$

Estimation/Optimization: Non-convex!

Feed-Forward Neural Networks/Multi-layer Perceptrons (MLPs)

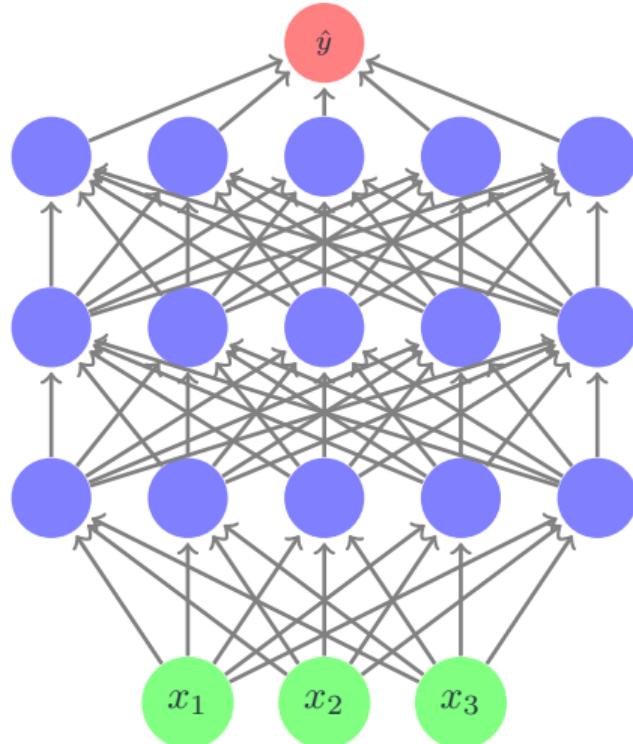


Feed-Forward Neural Networks/Multi-layer Perceptrons (MLPs)



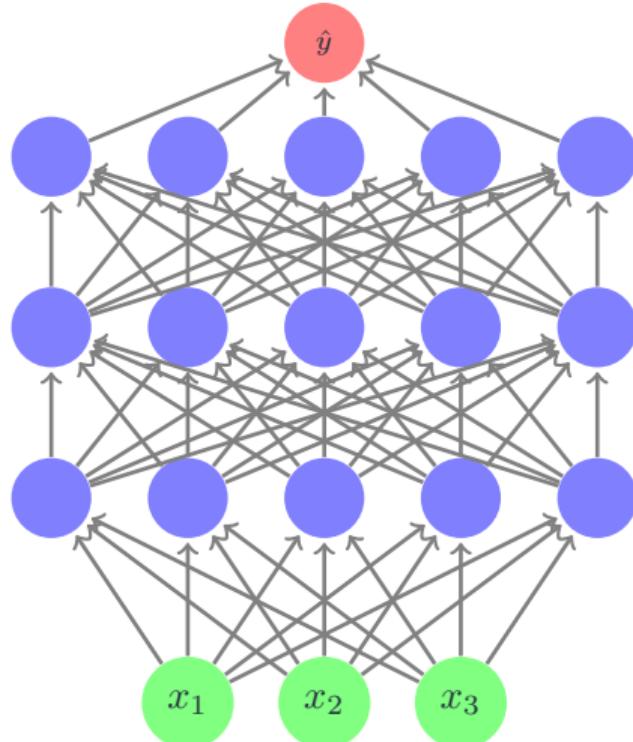
- Each neuron in a hidden layer computes an affine function of the previous layer, followed by a non-linear activation function
- Last hidden layer provides “**feature embeddings**” useful for transfer learning

Feed-Forward Neural Networks/Multi-layer Perceptrons (MLPs)



- Each neuron in a hidden layer computes an affine function of the previous layer, followed by a non-linear activation function
- Last hidden layer provides “**feature embeddings**” useful for transfer learning
- Flexible general function estimators
- More (and larger) hidden layers → more complex functions

Feed-Forward Neural Networks/Multi-layer Perceptrons (MLPs)



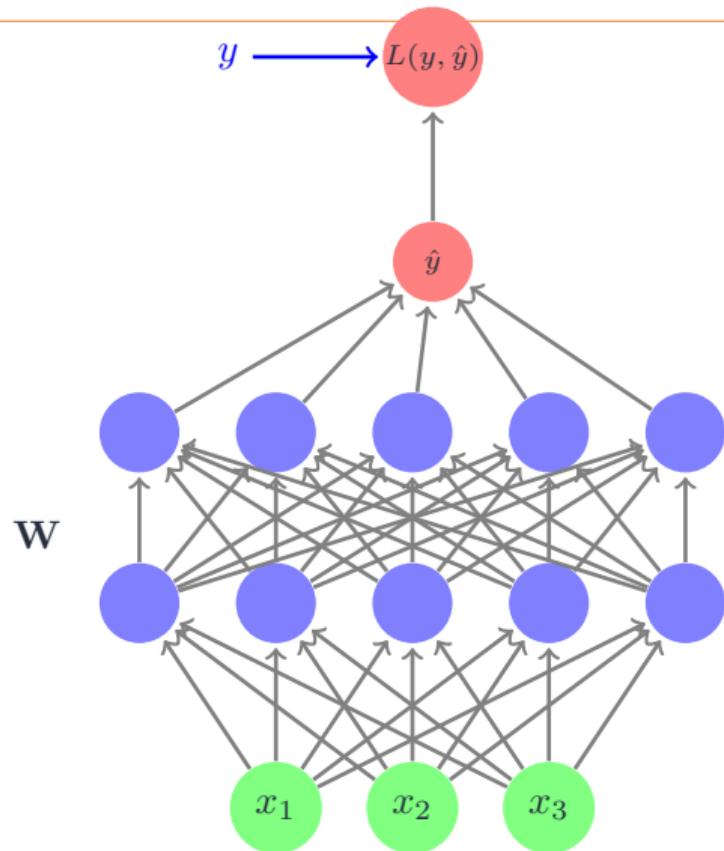
Pros

- Can learn complex input-output relationships
- Less manual feature engineering

Cons

- More data needed for training
- Careful tuning (e.g., regularization, learning rate)
- Sensitive to scaling of inputs

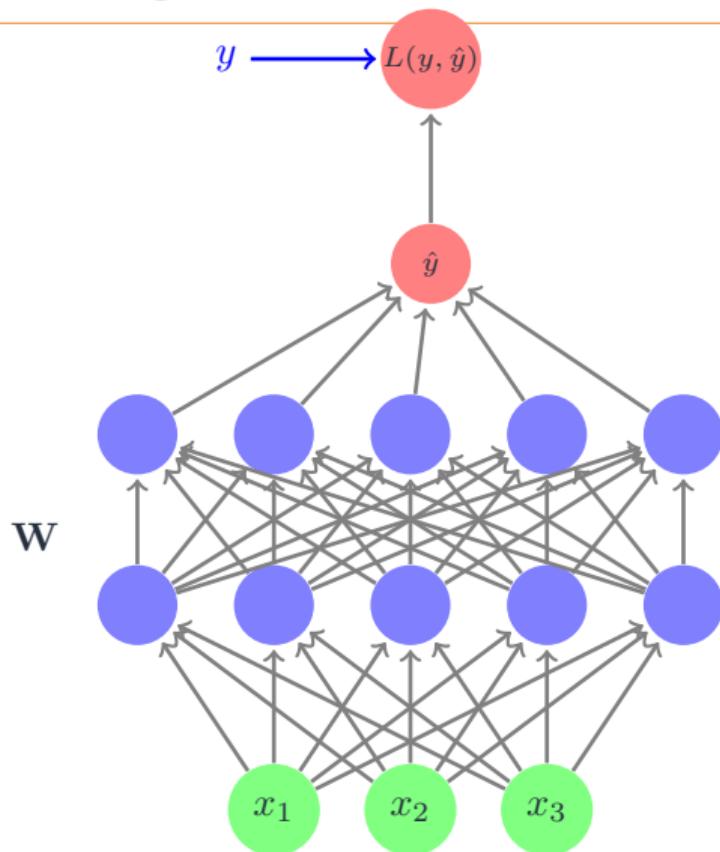
Training Neural Networks



- **Loss function:** Minimize some notion of error $L(y, \hat{y})$ on a training set $D = \{\mathbf{x}_i, y_i\}$

$$\mathbf{W}^* = \underset{\mathbf{W}}{\operatorname{argmin}} \sum_i L(y_i, \hat{y}_i), \quad \hat{y}_i = f(\mathbf{x}_i; \mathbf{W})$$

Training Neural Networks

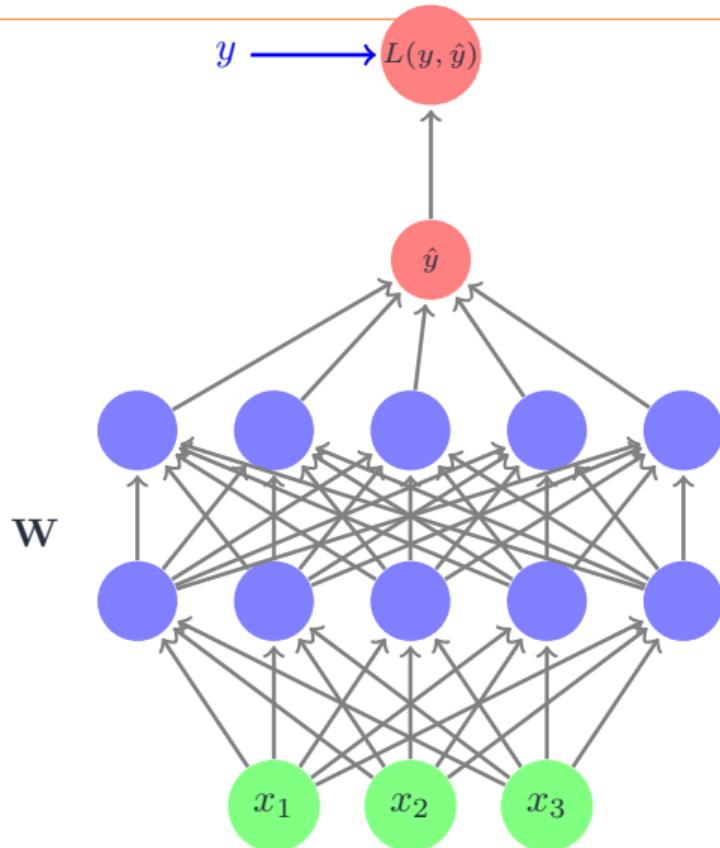


- **Loss function:** Minimize some notion of error $L(y, \hat{y})$ on a training set $D = \{\mathbf{x}_i, y_i\}$

$$\mathbf{W}^* = \underset{\mathbf{W}}{\operatorname{argmin}} \sum_i L(y_i, \hat{y}_i), \quad \hat{y}_i = f(\mathbf{x}_i; \mathbf{W})$$

- No analytical (closed-form) solution

Training Neural Networks



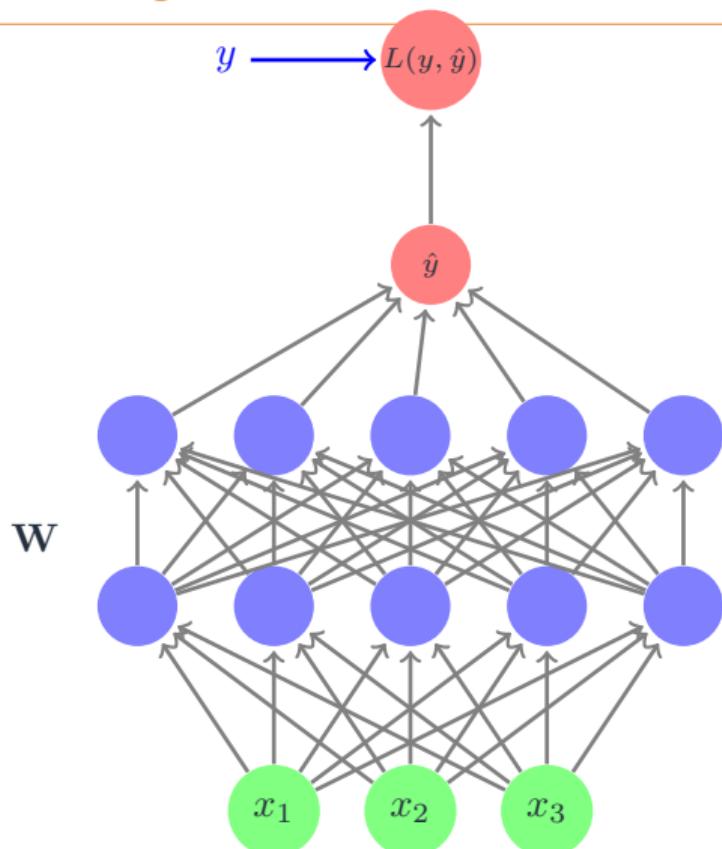
- **Loss function:** Minimize some notion of error $L(y, \hat{y})$ on a training set $D = \{\mathbf{x}_i, y_i\}$

$$\mathbf{W}^* = \underset{\mathbf{W}}{\operatorname{argmin}} \sum_i L(y_i, \hat{y}_i), \quad \hat{y}_i = f(\mathbf{x}_i; \mathbf{W})$$

- No analytical (closed-form) solution
- Numerical optimization, e.g., (stochastic) gradient descent

$$\mathbf{W}^t = \mathbf{W}^{t-1} - \eta \nabla \sum_i L(y_i, f(\mathbf{x}_i; \mathbf{W}^{t-1}))$$

Training Neural Networks



- **Loss function:** Minimize some notion of error $L(y, \hat{y})$ on a training set $D = \{\mathbf{x}_i, y_i\}$

$$\mathbf{W}^* = \underset{\mathbf{W}}{\operatorname{argmin}} \sum_i L(y_i, \hat{y}_i), \quad \hat{y}_i = f(\mathbf{x}_i; \mathbf{W})$$

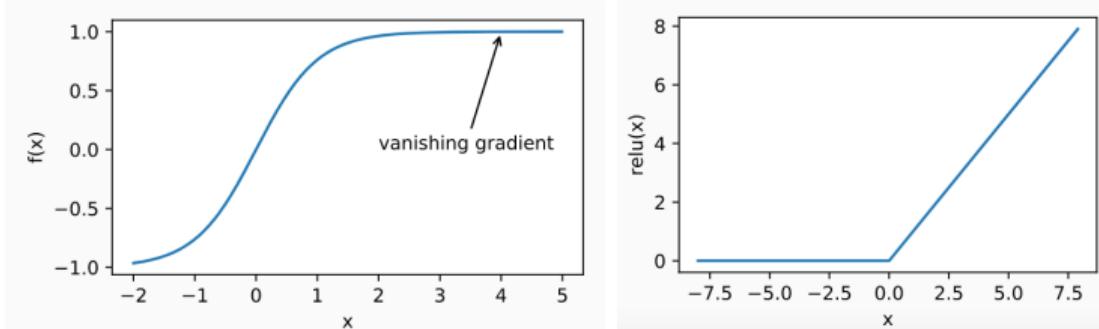
- No analytical (closed-form) solution
- Numerical optimization, e.g., (stochastic) gradient descent

$$\mathbf{W}^t = \mathbf{W}^{t-1} - \eta \nabla \sum_i L(y_i, f(\mathbf{x}_i; \mathbf{W}^{t-1}))$$

- **Backpropagation:** computes gradients efficiently (automatic differentiation via chain rule)

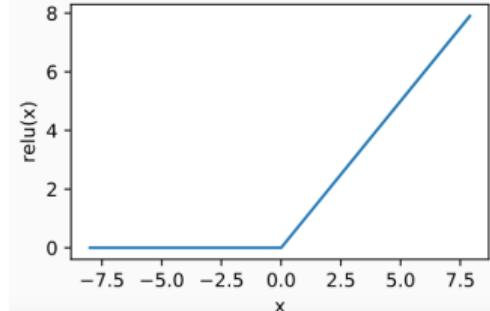
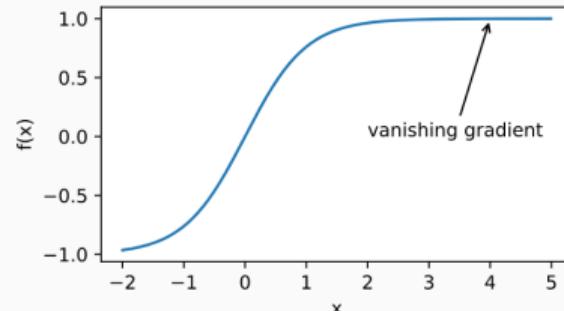
Optimization Challenges in Deep Learning

Vanishing gradient: Choice of non-linearities and architectures (RNN)

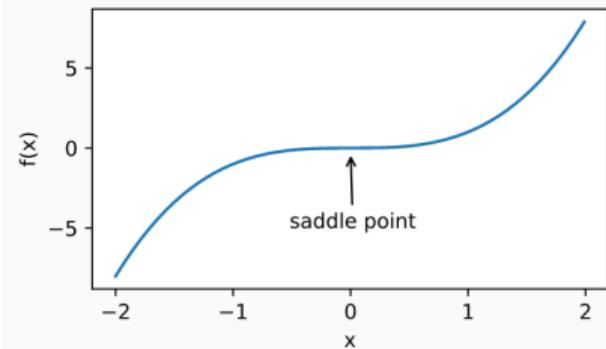
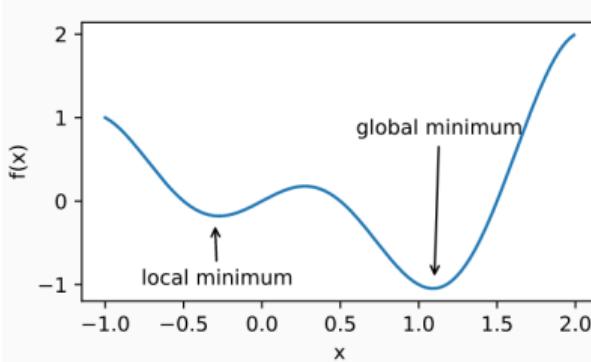


Optimization Challenges in Deep Learning

Vanishing gradient: Choice of non-linearities and architectures (RNN)



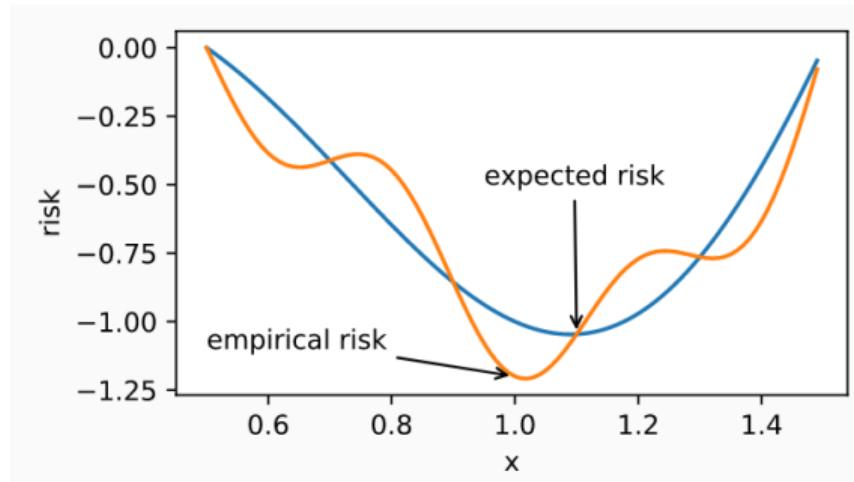
Non-convexity



What do you want to optimize?

Goal of machine learning: Minimize generalization error / expected risk

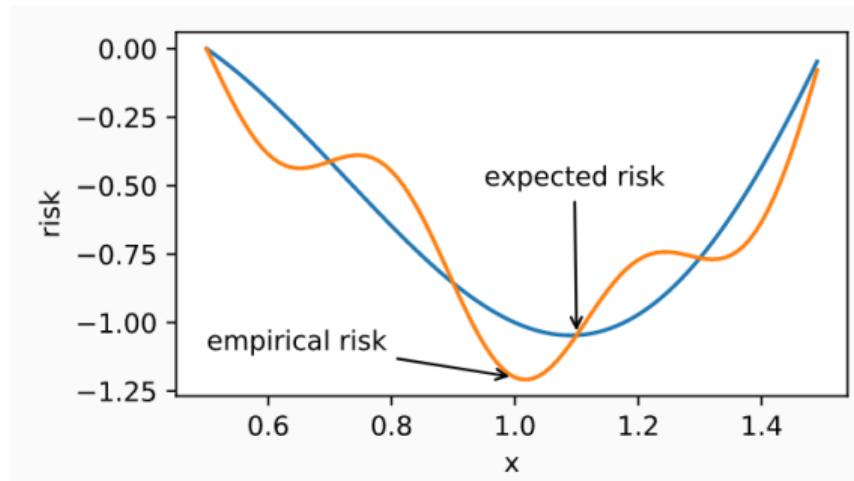
Practice: Minimize a surrogate: empirical risk on a *training set*



What do you want to optimize?

Goal of machine learning: Minimize generalization error / expected risk

Practice: Minimize a surrogate: empirical risk on a *training set*



Regularization: Explicit penalty in a convex model vs Approximate solution in a non-convex model

Stochastic Gradient Descent

Idea: Use a randomly sampled training example to update the weights

$$\mathbf{W}^{t+1} = \mathbf{W}^t - \eta \nabla L(y_i, f(\mathbf{x}_i; \mathbf{W}^t))$$

Motivation:

- Reduce computational cost: error in estimation proportional to $\frac{1}{\sqrt{n}}$
- Better generalization performance
- Avoiding local minima

Stochastic Gradient Descent

Idea: Use a randomly sampled training example to update the weights

$$\mathbf{W}^{t+1} = \mathbf{W}^t - \eta \nabla L(y_i, f(\mathbf{x}_i; \mathbf{W}^t))$$

Motivation:

- Reduce computational cost: error in estimation proportional to $\frac{1}{\sqrt{n}}$
- Better generalization performance
- Avoiding local minima

Practice:

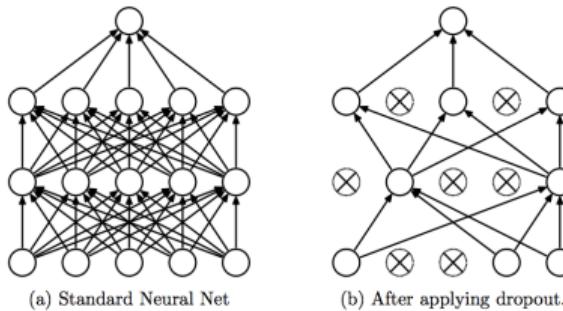
- Mini-batch stochastic gradient descent

$$\mathbf{W}^{t+1} = \mathbf{W}^t - \eta_t \frac{1}{|\mathbf{B}^t|} \sum_{i \in \mathbf{B}^t} \nabla L(y_i, f(\mathbf{x}_i; \mathbf{W}^t))$$

- How to choose size of the batch?
- Popular algorithm: Adam

Further practical questions

Regularization via Dropout



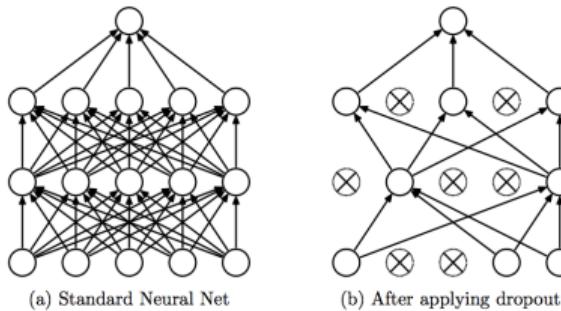
Randomly drop neurons (and their connections) during training [Srivastava et al., 2014]

Initialization of weights, how to adapt the step size, how to tune parameters, . . . Many more!

Big question: which neural architecture is best?

Further practical questions

Regularization via Dropout



Randomly drop neurons (and their connections) during training [[Srivastava et al., 2014](#)]

Initialization of weights, how to adapt the step size, how to tune parameters, . . . Many more!

Big question: which neural architecture is best? Next: Convolutional and recurrent neural networks as two examples.

Convolutional Neural Networks

Convolutional Neural Networks [Le Cun, 1989]

- Convolutional Neural Networks (CNNs) = NNs that use *convolutional layers*
- CNNs with 2D convolutions are extremely successful in computer vision applications
 \implies encode spatial invariance

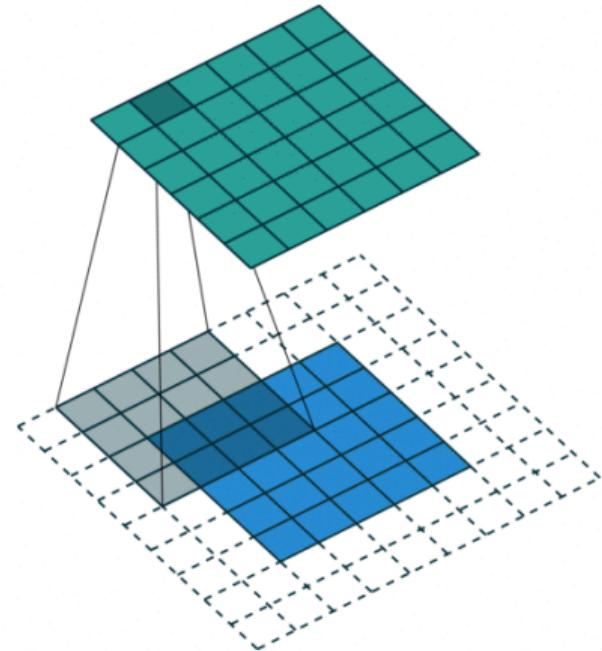


Figure credit: Vincent Dumoulin, Francesco Visin - A guide to convolution arithmetic for deep learning;
https://github.com/vdumoulin/conv_arithmetic

Convolutional Neural Networks [Le Cun, 1989]

- Convolutional Neural Networks (CNNs) = NNs that use *convolutional layers*
- CNNs with 2D convolutions are extremely successful in computer vision applications
 \implies encode spatial invariance

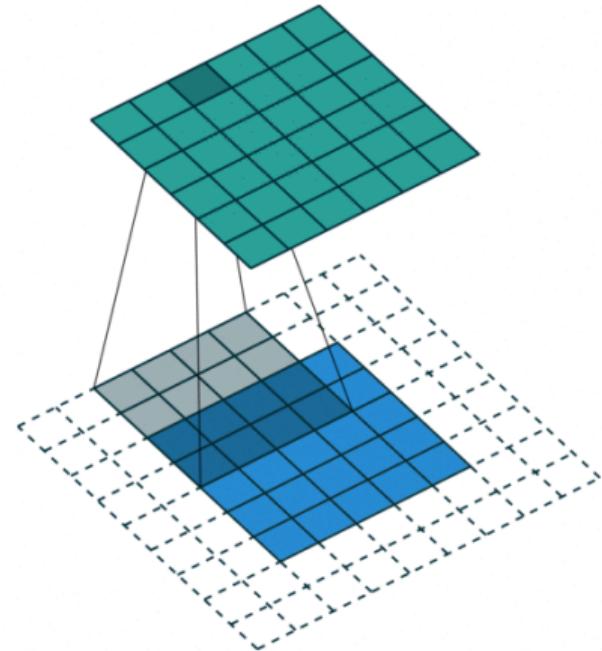


Figure credit: Vincent Dumoulin, Francesco Visin - A guide to convolution arithmetic for deep learning;
https://github.com/vdumoulin/conv_arithmetic

Convolutional Neural Networks [Le Cun, 1989]

- Convolutional Neural Networks (CNNs) = NNs that use *convolutional layers*
- CNNs with 2D convolutions are extremely successful in computer vision applications
 \implies encode spatial invariance

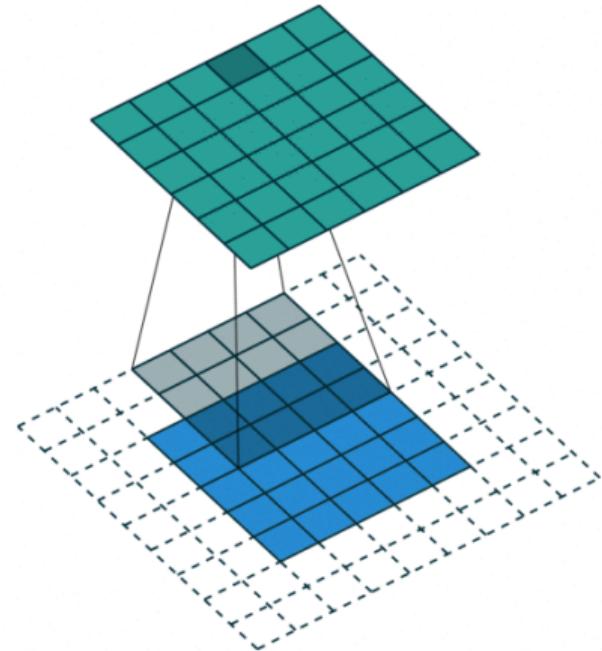


Figure credit: Vincent Dumoulin, Francesco Visin - A guide to convolution arithmetic for deep learning;
https://github.com/vdumoulin/conv_arithmetic

Convolutional Neural Networks [Le Cun, 1989]

- Convolutional Neural Networks (CNNs) = NNs that use *convolutional layers*
- CNNs with 2D convolutions are extremely successful in computer vision applications
 \implies encode spatial invariance

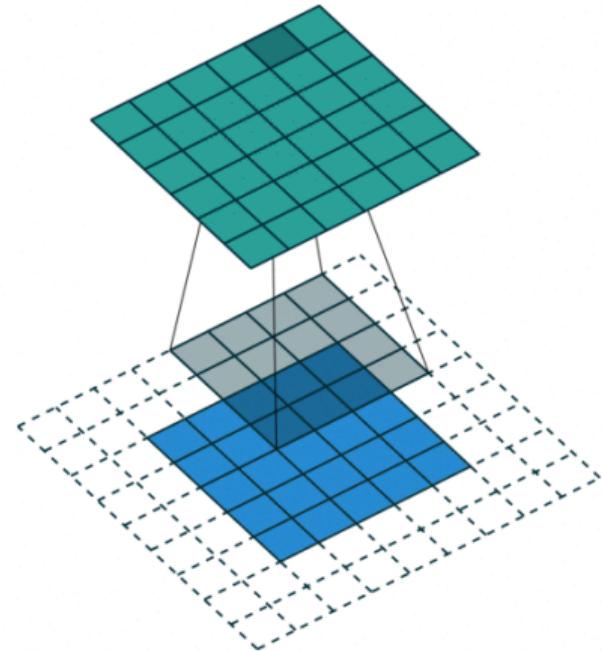


Figure credit: Vincent Dumoulin, Francesco Visin - A guide to convolution arithmetic for deep learning;
https://github.com/vdumoulin/conv_arithmetic

Convolutional Neural Networks [Le Cun, 1989]

- Convolutional Neural Networks (CNNs) = NNs that use *convolutional layers*
- CNNs with 2D convolutions are extremely successful in computer vision applications
 \implies encode spatial invariance

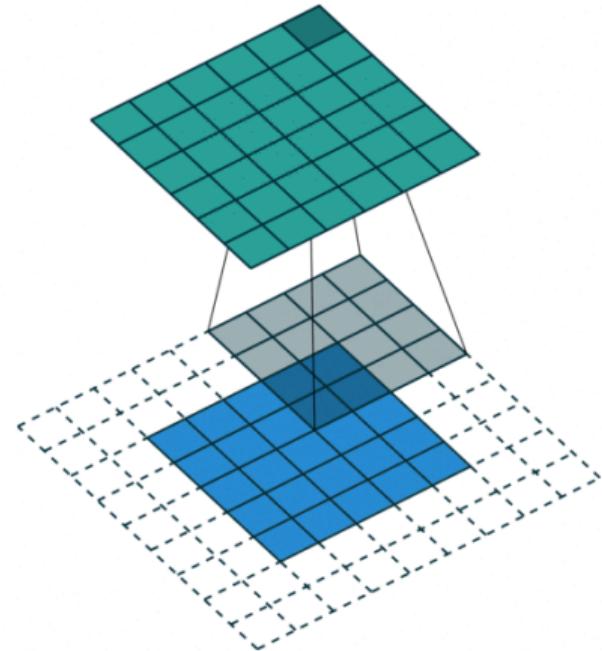


Figure credit: Vincent Dumoulin, Francesco Visin - A guide to convolution arithmetic for deep learning;
https://github.com/vdumoulin/conv_arithmetic

Convolutional Neural Networks [Le Cun, 1989]

- Convolutional Neural Networks (CNNs) = NNs that use *convolutional layers*
- CNNs with 2D convolutions are extremely successful in computer vision applications
 \implies encode spatial invariance

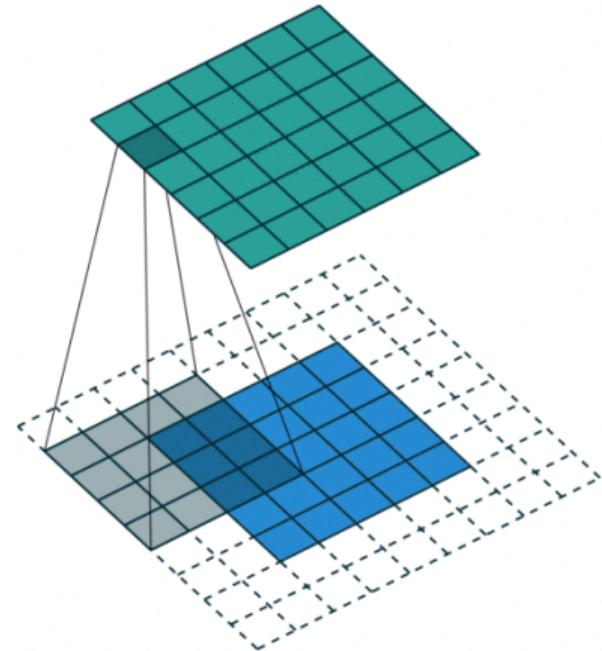


Figure credit: Vincent Dumoulin, Francesco Visin - A guide to convolution arithmetic for deep learning;
https://github.com/vdumoulin/conv_arithmetic

Convolutional Neural Networks [Le Cun, 1989]

- Convolutional Neural Networks (CNNs) = NNs that use *convolutional layers*
- CNNs with 2D convolutions are extremely successful in computer vision applications
 \implies encode spatial invariance

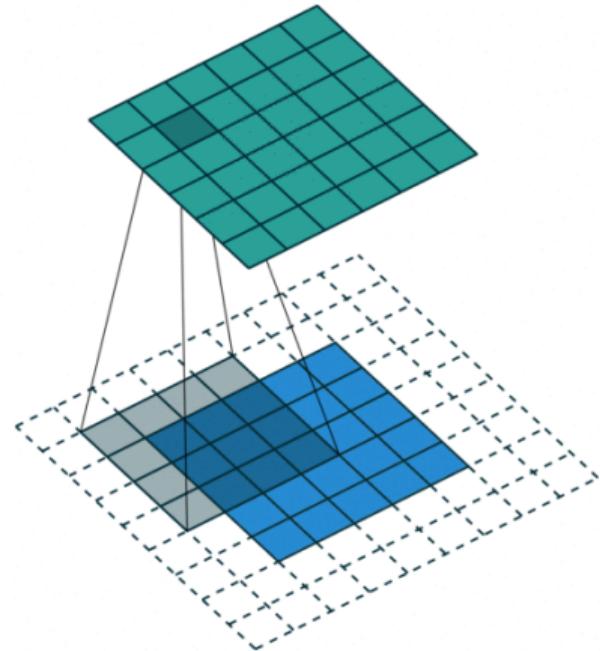


Figure credit: Vincent Dumoulin, Francesco Visin - A guide to convolution arithmetic for deep learning;
https://github.com/vdumoulin/conv_arithmetic

Convolutional Neural Networks [Le Cun, 1989]

- Convolutional Neural Networks (CNNs) = NNs that use *convolutional layers*
- CNNs with 2D convolutions are extremely successful in computer vision applications
 \implies encode spatial invariance

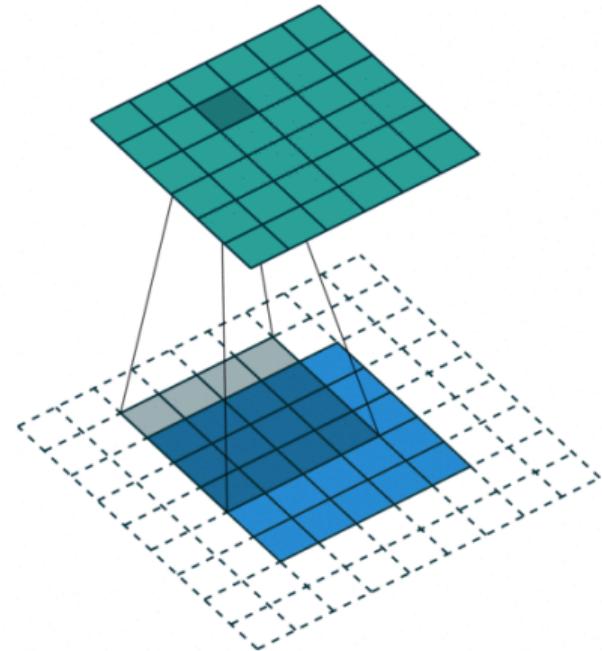


Figure credit: Vincent Dumoulin, Francesco Visin - A guide to convolution arithmetic for deep learning;
https://github.com/vdumoulin/conv_arithmetic

Convolutional Neural Networks [Le Cun, 1989]

- Convolutional Neural Networks (CNNs) = NNs that use *convolutional layers*
- CNNs with 2D convolutions are extremely successful in computer vision applications
 \implies encode spatial invariance

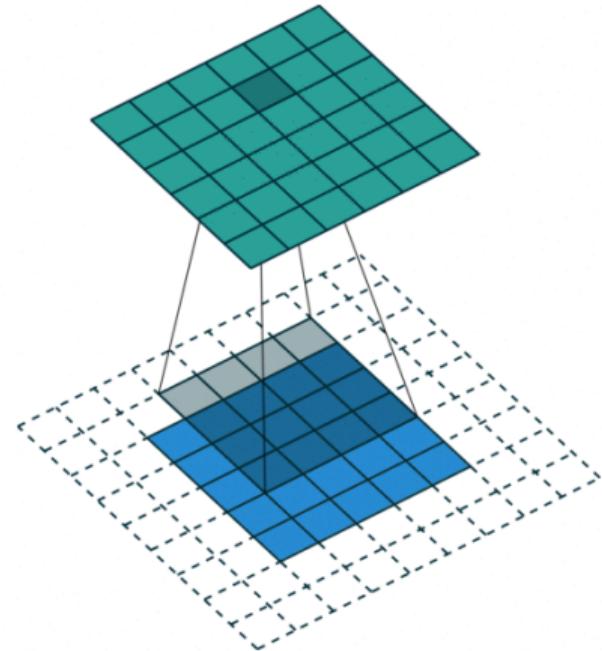
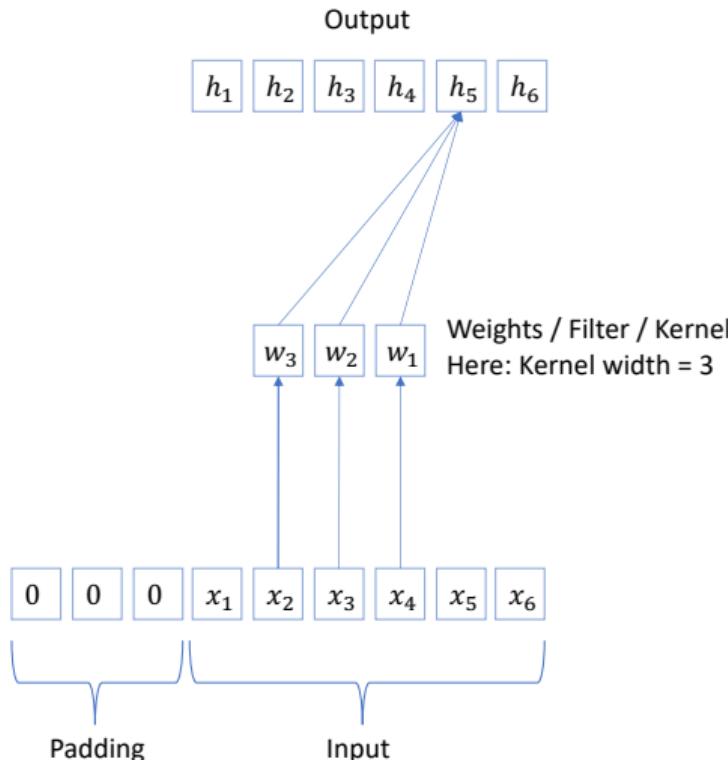


Figure credit: Vincent Dumoulin, Francesco Visin - A guide to convolution arithmetic for deep learning;
https://github.com/vdumoulin/conv_arithmetic

Convolutional Layers



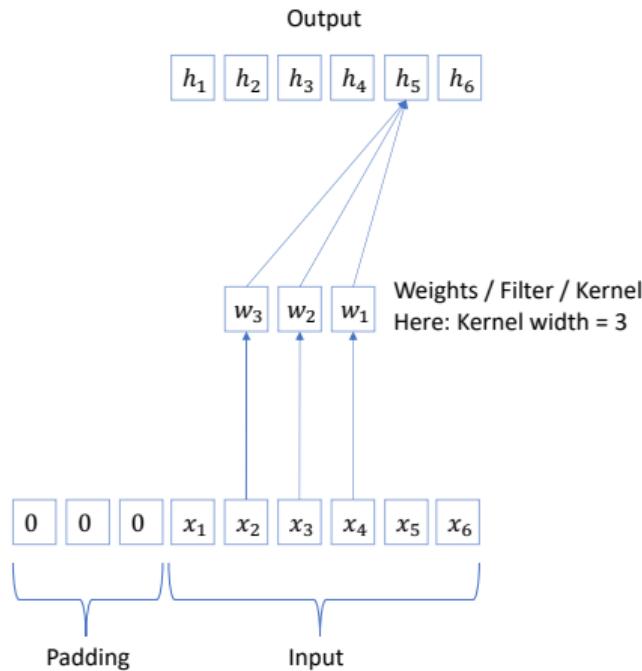
- The output h_j of a neuron j in a convolution layer is a discrete convolution of the inputs \mathbf{x} with the layer's weights/filter \mathbf{w} .
- For a one-dimensional convolution with a kernel with width D we have

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

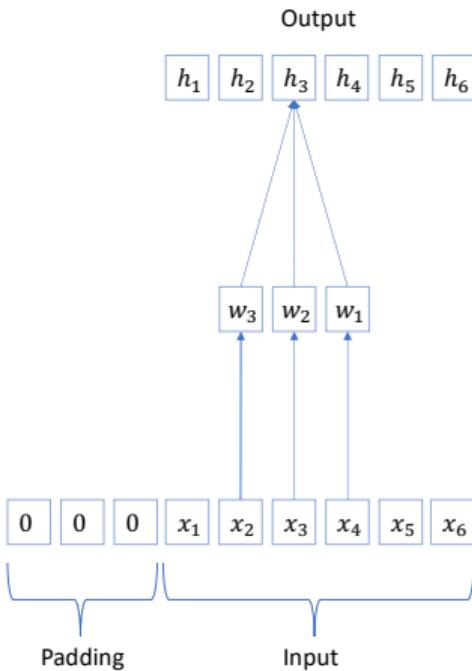
- Padding is used to shift the input relative to the output and change the behavior around the edges (causal vs. non-causal)

Causal vs. Non-Causal Convolution

Causal Convolution



Non-Causal Convolution



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

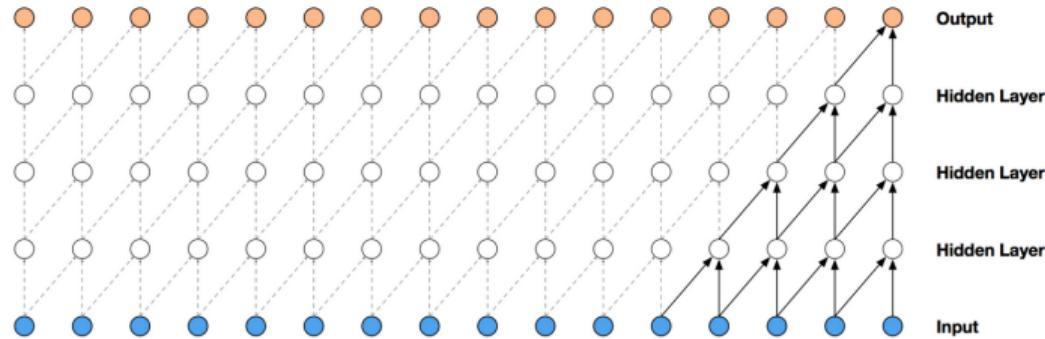


Figure credit: WaveNet: A Generative Model for Raw Audio; <https://deepmind.com/blog/wavenet-generative-model-raw-audio/>

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

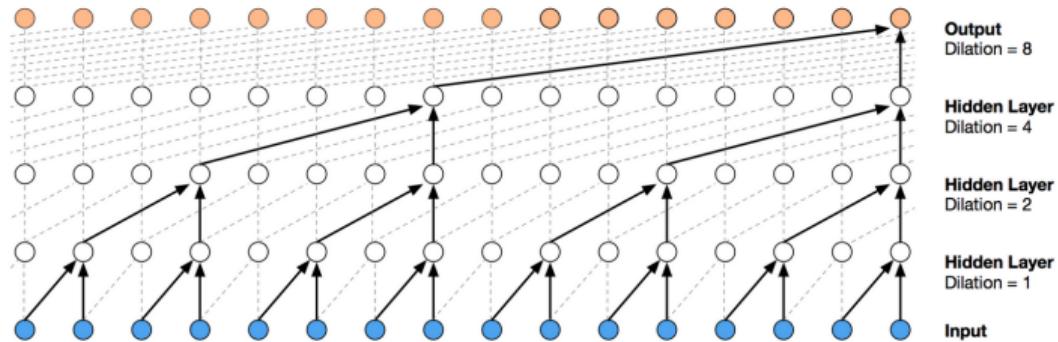
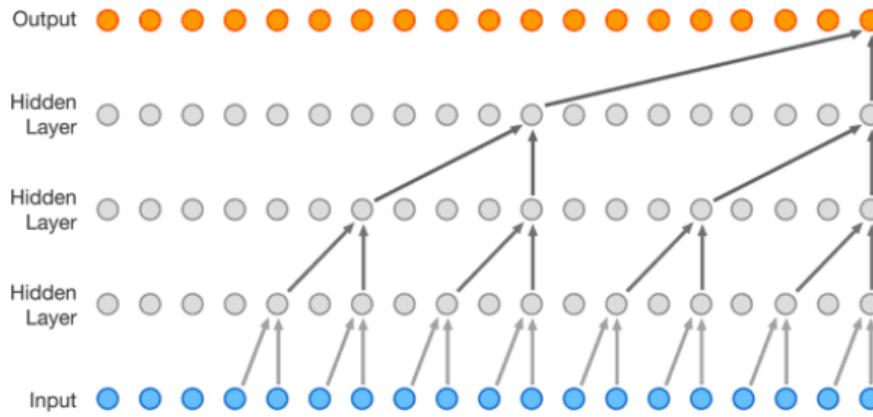


Figure credit: WaveNet: A Generative Model for Raw Audio; <https://deepmind.com/blog/wavenet-generative-model-raw-audio/>

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]

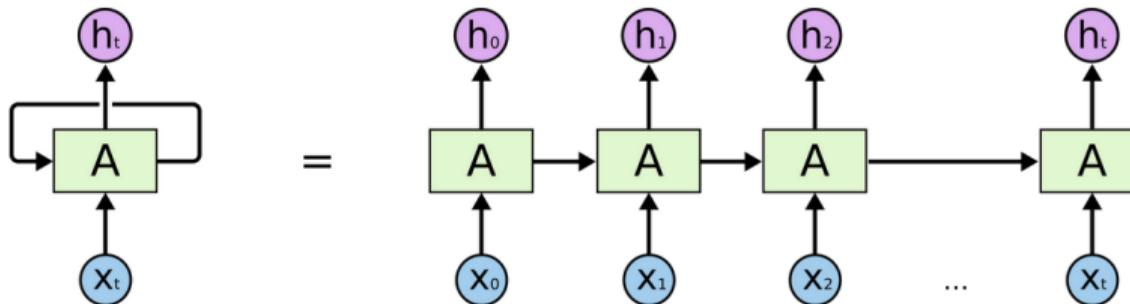
Dilated Causal Convolution and WaveNet [van den Oord et al., 2016]

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

Convolutional Neural Networks

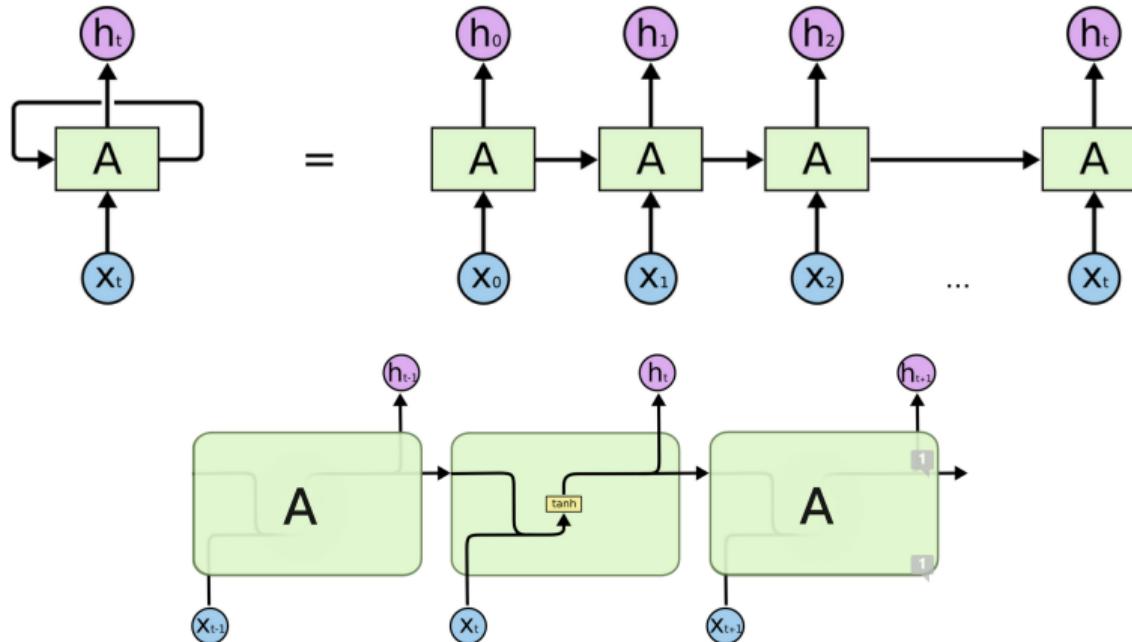
Recurrent Neural Networks (RNN)

Source: <https://colah.github.io/posts/2015-08-Understanding-LSTMs/>



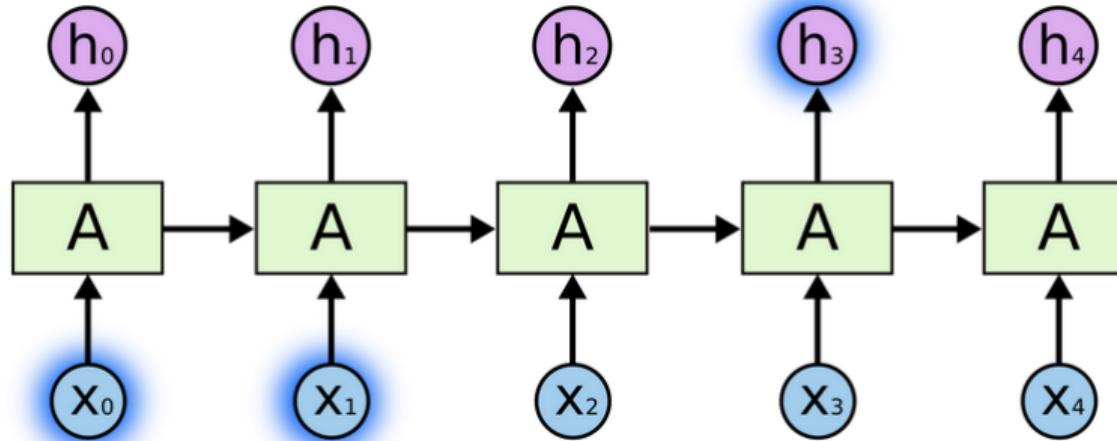
Recurrent Neural Networks (RNN)

Source: <https://colah.github.io/posts/2015-08-Understanding-LSTMs/>



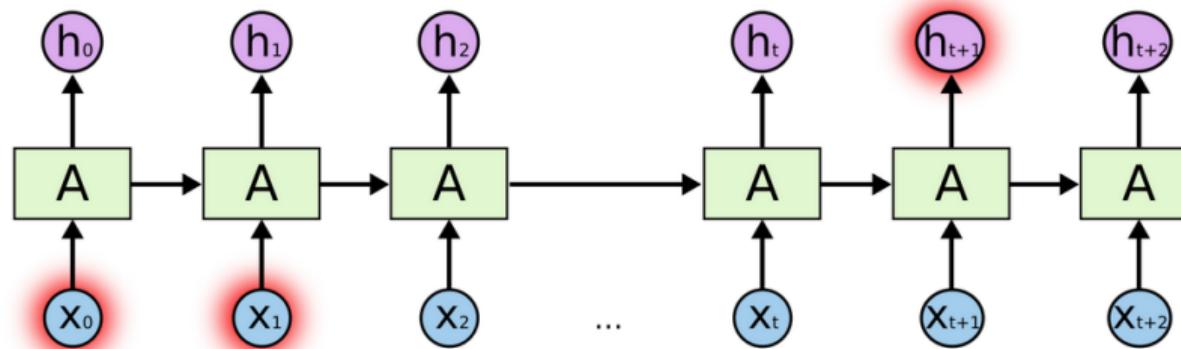
Problem of Long Term Dependencies

Source: <https://colah.github.io/posts/2015-08-Understanding-LSTMs/>

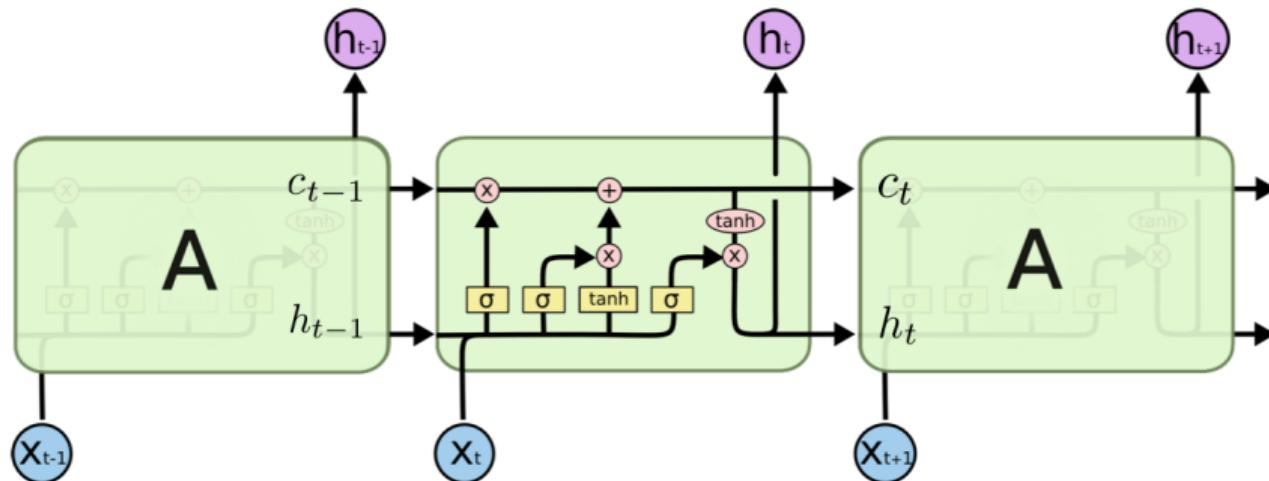


Problem of Long Term Dependencies

Source: <https://colah.github.io/posts/2015-08-Understanding-LSTMs/>

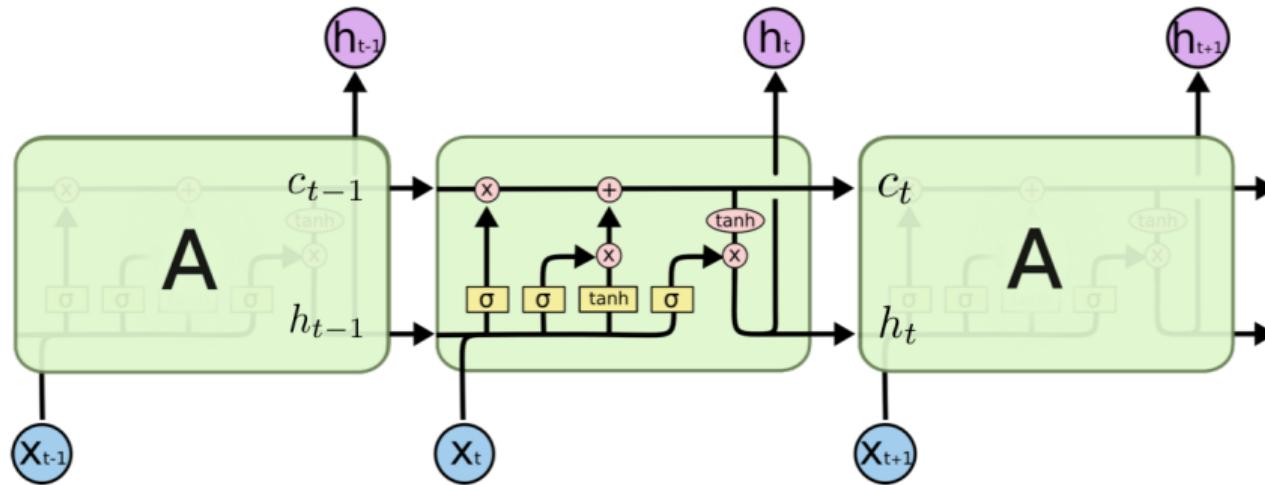


Long Short-Term Memory (LSTM) [Hochreiter and Schmidhuber, 1997]



<HTTP://COLAH.GITHUB.IO/POSTS/2015-08-UNDERSTANDING-LSTMs/>

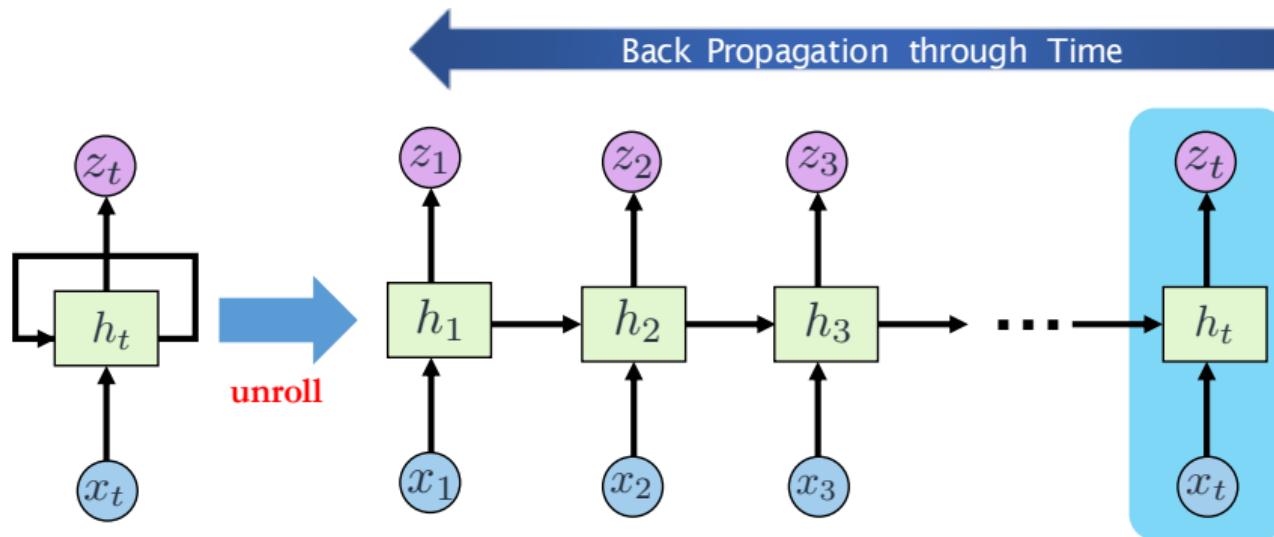
Long Short-Term Memory (LSTM): What?



current state = **forgot gate** \times old stuff + **input gate** \times new stuff

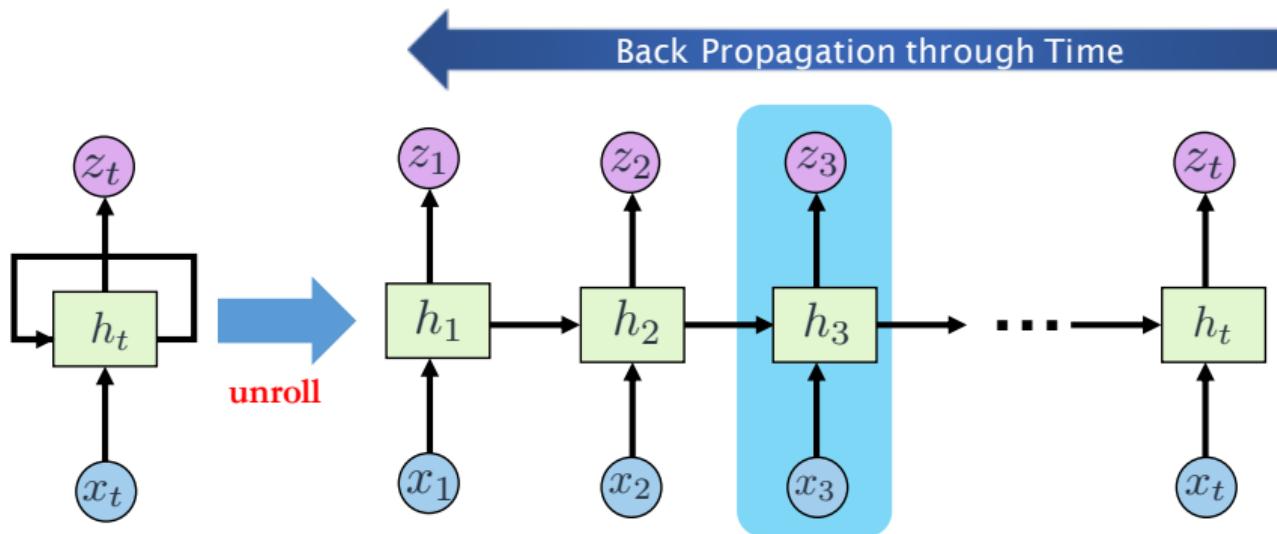
RNN: Vanishing Gradient Problem

WHAT DO YOU WISH TO HAPPEN



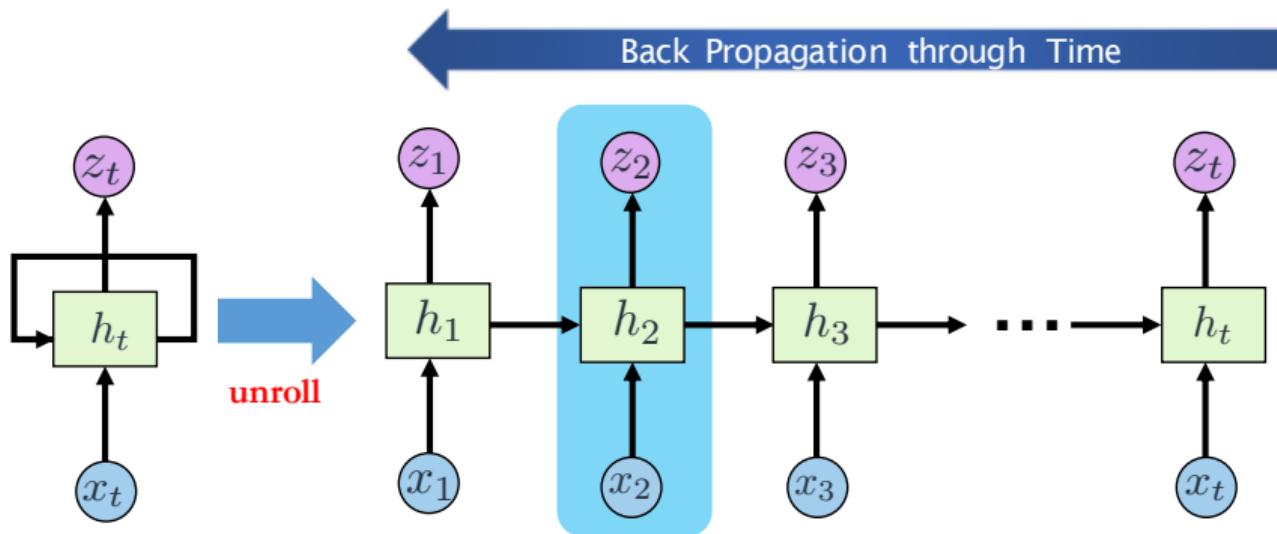
RNN: Vanishing Gradient Problem

WHAT DO YOU WISH TO HAPPEN



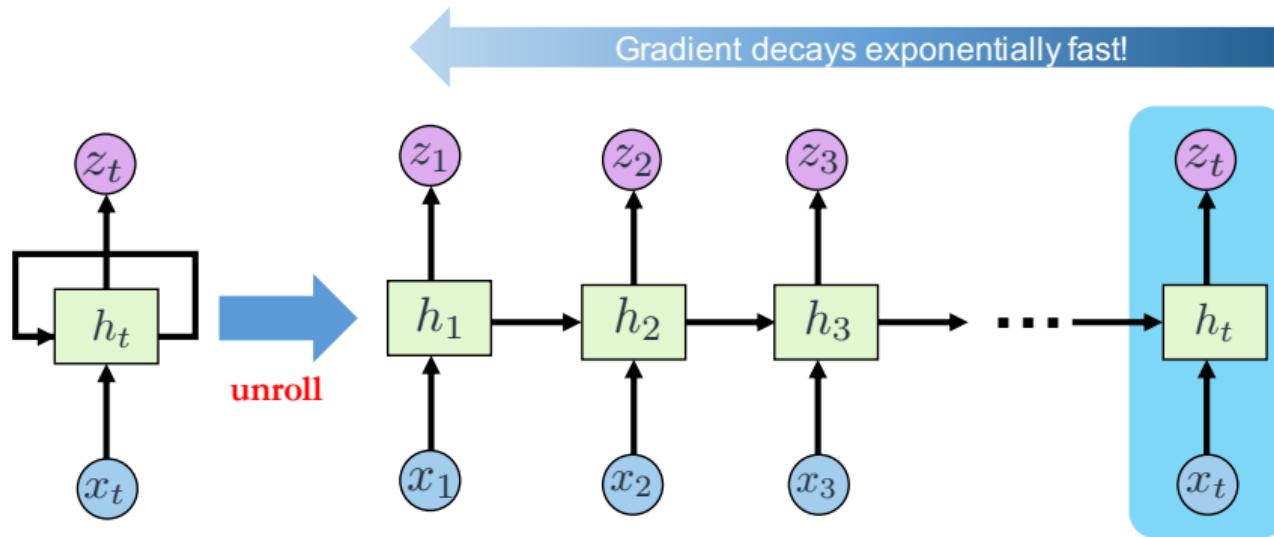
RNN: Vanishing Gradient Problem

WHAT DO YOU WISH TO HAPPEN

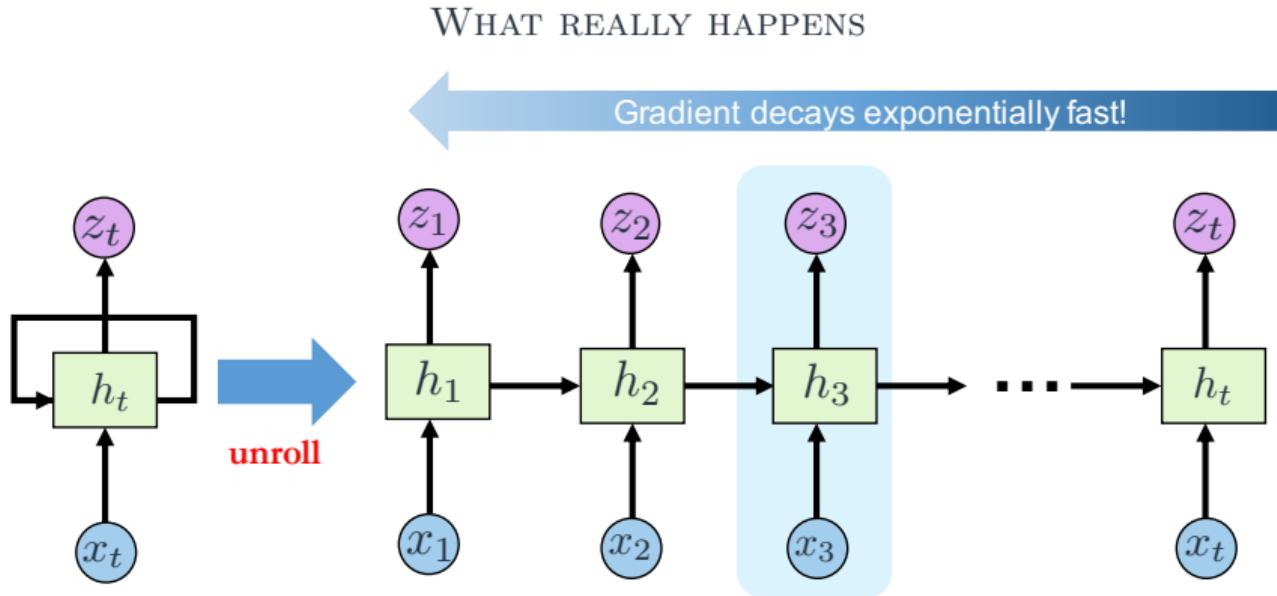


RNN: Vanishing Gradient Problem

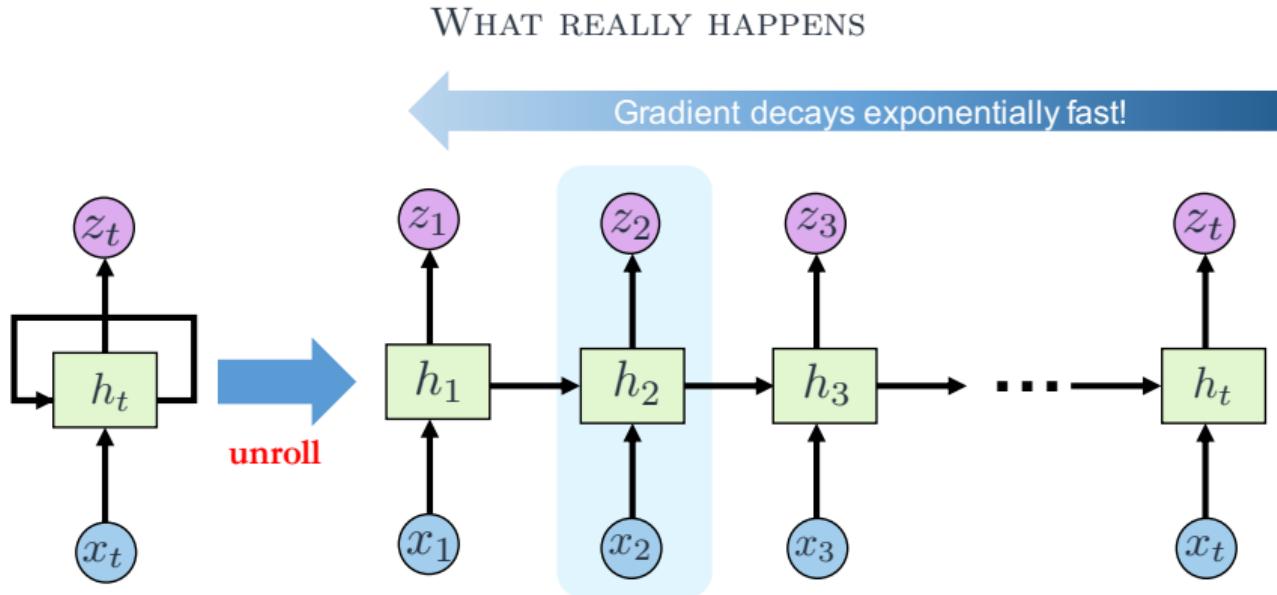
WHAT REALLY HAPPENS



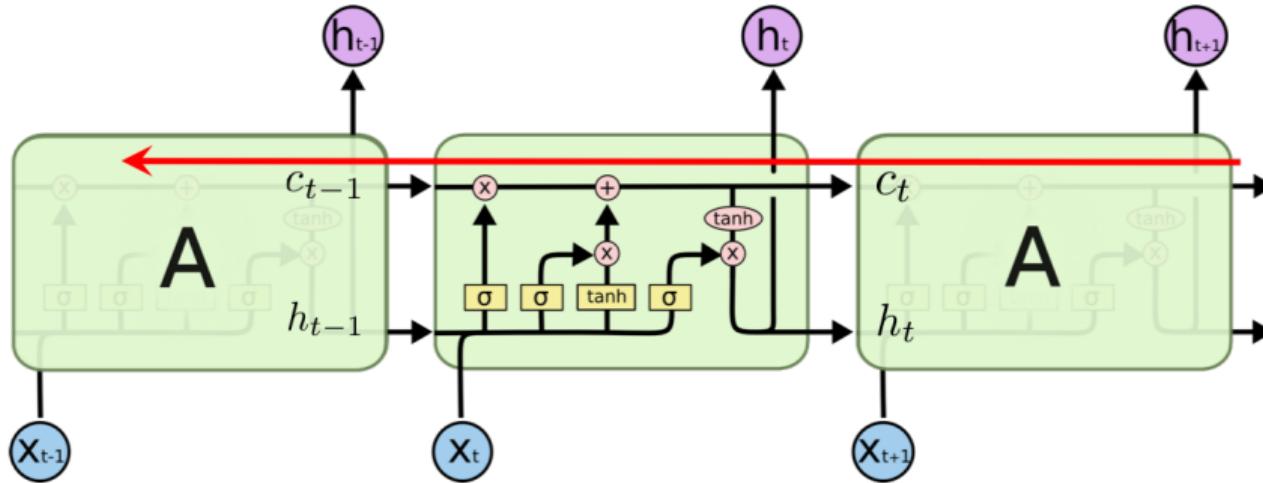
RNN: Vanishing Gradient Problem



RNN: Vanishing Gradient Problem



LSTMs Prevent Vanishing Gradient!



<HTTP://COLAH.GITHUB.IO/POSTS/2015-08-UNDERSTANDING-LSTMs/>

There is at least one path where the gradient does not vanish!

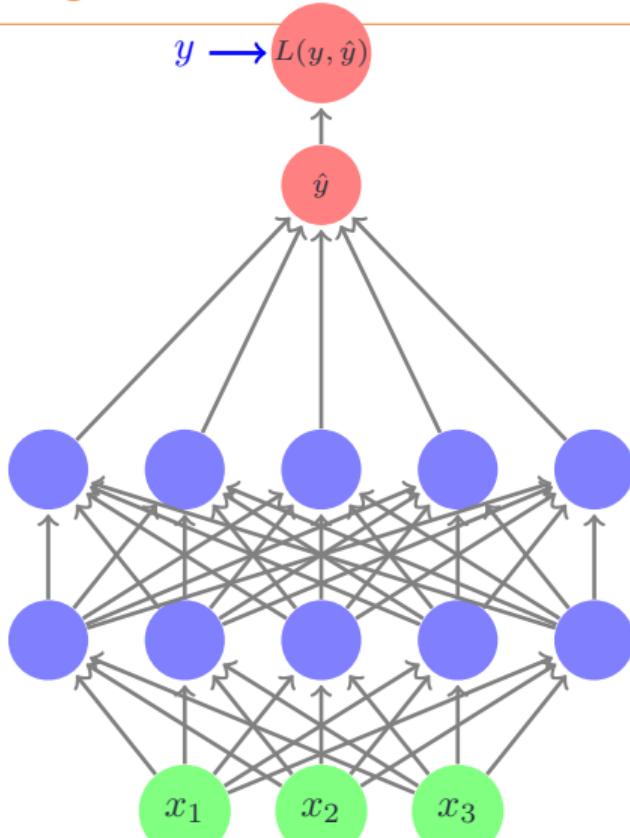
Probabilistic Predictions

Probabilistic Prediction with Deep Learning

Point prediction: $D = \{x_i, y_i\}_{i=1}^n$

$$\hat{y} = f(\mathbf{x}; \theta)$$

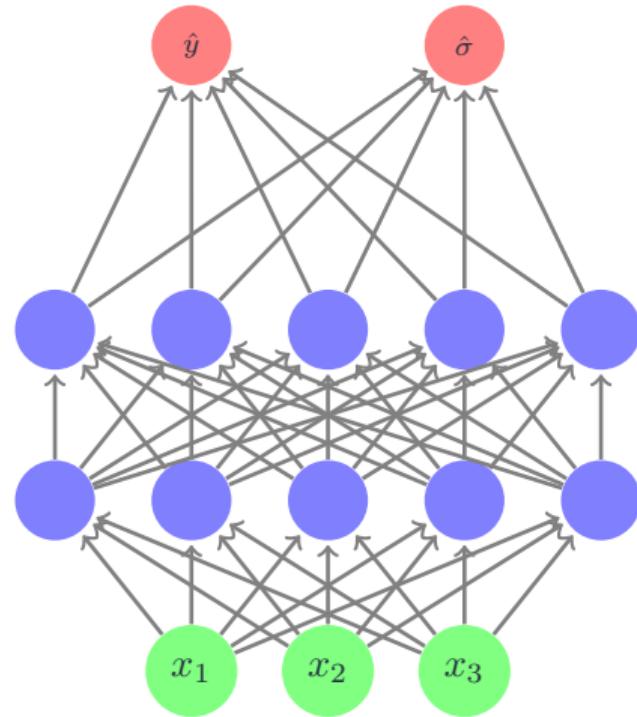
$$\theta^* = \operatorname{argmax}_{\theta} \sum_i L(y_i, \hat{y}_i)$$



Probabilistic Prediction with Deep Learning

Probabilistic prediction: $D = \{x_i, y_i\}_{i=1}^n$

$$y = f(\mathbf{x}; \theta) + \epsilon, \quad \epsilon \sim \mathcal{N}(0, \sigma_x^2),$$
$$p(y|\mathbf{x}; \theta) = \mathcal{N}(y | \underbrace{f(\mathbf{x}; \theta)}_{\text{mean}}, \underbrace{\sigma_x^2}_{\text{variance}}),$$



Probabilistic Prediction with Deep Learning

Probabilistic prediction: $D = \{\mathbf{x}_i, y_i\}_{i=1}^n$

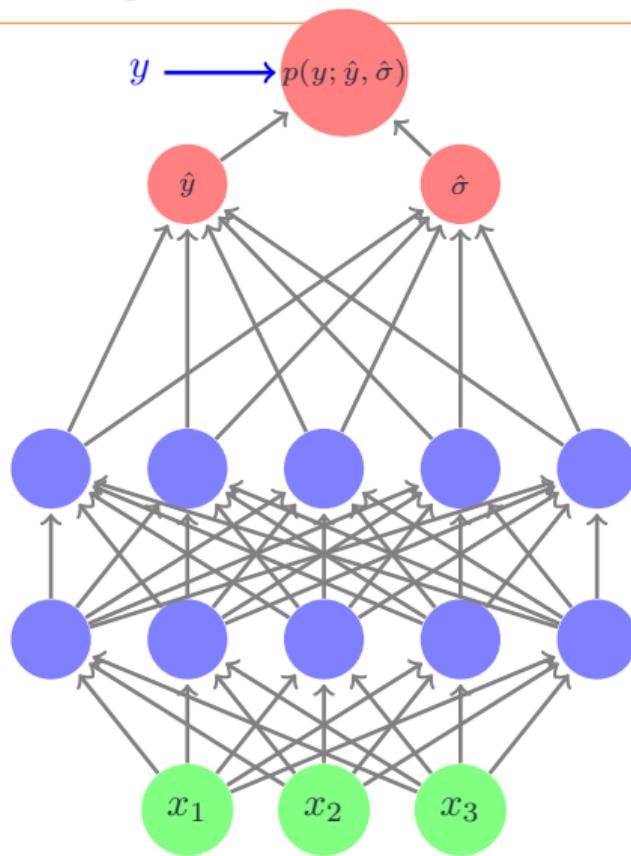
$$y = f(\mathbf{x}; \theta) + \epsilon, \quad \epsilon \sim \mathcal{N}(0, \sigma_x^2),$$

$$p(y|\mathbf{x}; \theta) = \mathcal{N}(y | \underbrace{f(\mathbf{x}; \theta)}_{\text{mean}}, \underbrace{\sigma_x^2}_{\text{variance}}),$$

Maximum likelihood principle:

$$\theta^* = \operatorname{argmax}_{\theta} \sum_i \log p(y_i | \mathbf{x}_i; \theta)$$

(Auto differentiation if the likelihood is differentiable)



Recipe for Probabilistic Prediction with Deep Learning

- Choose appropriate probabilistic model for the targets y : $p(y; \theta)$ with parameters θ
 1. **Regression:** Gaussian, Poisson, Zero-inflated distributions etc.
 2. **Classification:** Bernoulli, Categorical
 3. **Structured prediction:** e.g., for sequences: State space model, Auto-regressive model etc.

Recipe for Probabilistic Prediction with Deep Learning

- Choose appropriate probabilistic model for the targets y : $p(y; \theta)$ with parameters θ
 1. **Regression**: Gaussian, Poisson, Zero-inflated distributions etc.
 2. **Classification**: Bernoulli, Categorical
 3. **Structured prediction**: e.g., for sequences: State space model, Auto-regressive model etc.
- Make θ (can have more than one parameter) the output of the neural network:

$$\theta = \text{NN}(\mathbf{x}; W)$$

Here the type of the neural network NN is chosen according to the input features \mathbf{x}

Recipe for Probabilistic Prediction with Deep Learning

- Choose appropriate probabilistic model for the targets y : $p(y; \theta)$ with parameters θ
 1. **Regression:** Gaussian, Poisson, Zero-inflated distributions etc.
 2. **Classification:** Bernoulli, Categorical
 3. **Structured prediction:** e.g., for sequences: State space model, Auto-regressive model etc.
- Make θ (can have more than one parameter) the output of the neural network:

$$\theta = \text{NN}(\mathbf{x}; W)$$

Here the type of the neural network NN is chosen according to the input features \mathbf{x}

- **Training:** Learn the neural network weights W by maximizing the log-likelihood of θ given the targets y and features \mathbf{x}

$$W^* = \underset{W}{\operatorname{argmax}} \sum_i \log p(y_i | \mathbf{x}_i; \underbrace{\text{NN}(\mathbf{x}_i; W)}_{\theta})$$

COFFEE BREAK!



GluonTS github repository: <https://github.com/awslabs/gluon-ts>



Part II: Deep Learning for Forecasting

Neural Network Forecasting: Old and New – Timeline



International Journal of Forecasting
Volume 14, Issue 1, 1 March 1998, Pages 35-62



Forecasting with artificial neural networks:: The state of the art
Guoqiang Zhang, B. Eddy Patuwo, Michael Y. Hu



Research article

**How effective are neural networks at forecasting and prediction?
A review and evaluation**

Monica Adya Fred Collopy

First published: 04 December 1998



Econometric Reviews
Publication details, including instructions for authors and subscription information:
<http://www.informaworld.com/supp/title-content-073597248>

An Empirical Comparison of Machine Learning Models for Time Series Forecasting
Neoson K. Ahmed^a; Amrit P. Attiyah^b; Neamat El Gayar^c; Hisham El-Shishiny^d
^a Department of Computer Science, Purdue University, West Lafayette, Indiana, USA ^b Department of Computer Engineering, Cairo University, Giza, Egypt ^c Faculty of Computers and Information, Cairo University, Giza, Egypt ^d IBM Center for Advanced Studies in Cairo, IBM Cairo Technology Development Center, Giza, Egypt

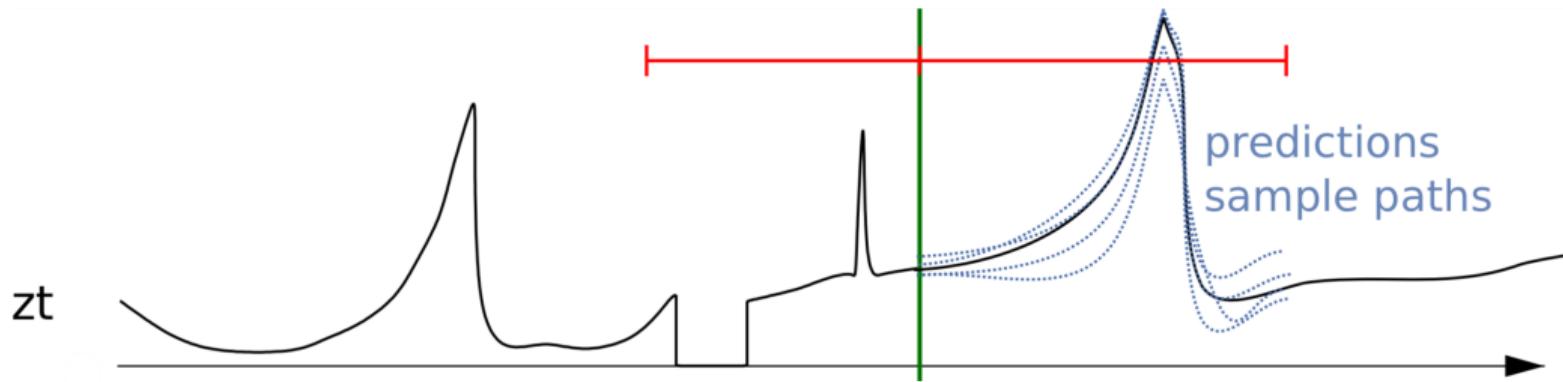
Online publication date: 15 September 2010

- 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, *local* models)
- 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)
- 2017 Transformer models (BERT) take over NLP
- 2018 M4 competition: combination of NNs and classical techniques wins
- 2020 M5 competition: "ML" techniques win. NBEATS in 2nd place, DeepAR in 3rd place

General Setup

Predict the future behaviour of a time series given its past

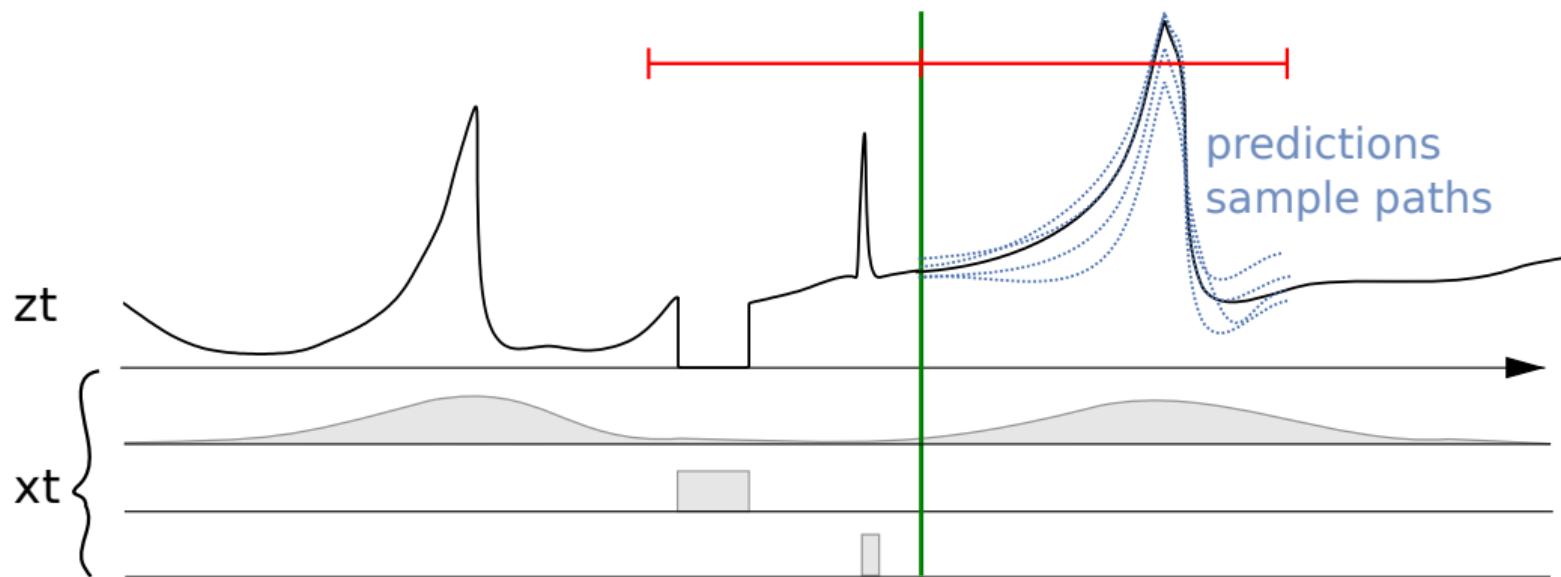
$$z_1, z_2, \dots, z_T \implies P(z_{T+1}, z_{T+2}, \dots z_{T+\tau})$$



General Setup

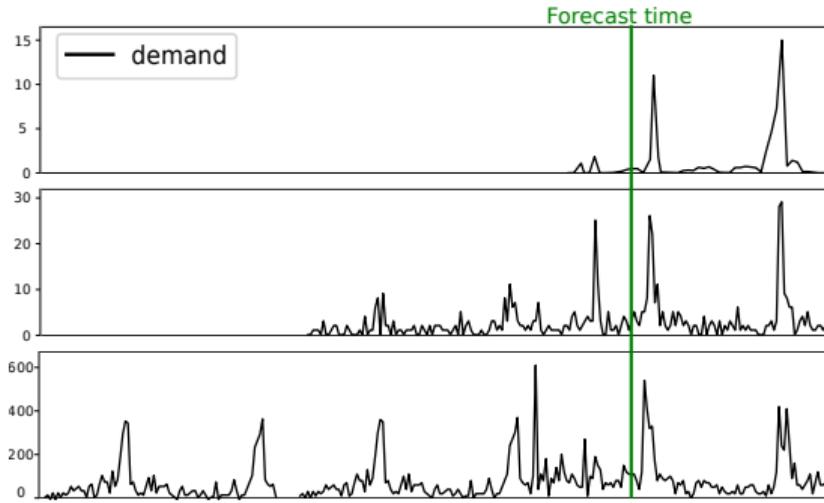
Predict the future behaviour of a time series given its past

$$z_1, z_2, \dots, z_T \implies P(z_{T+1}, z_{T+2}, \dots z_{T+\tau})$$



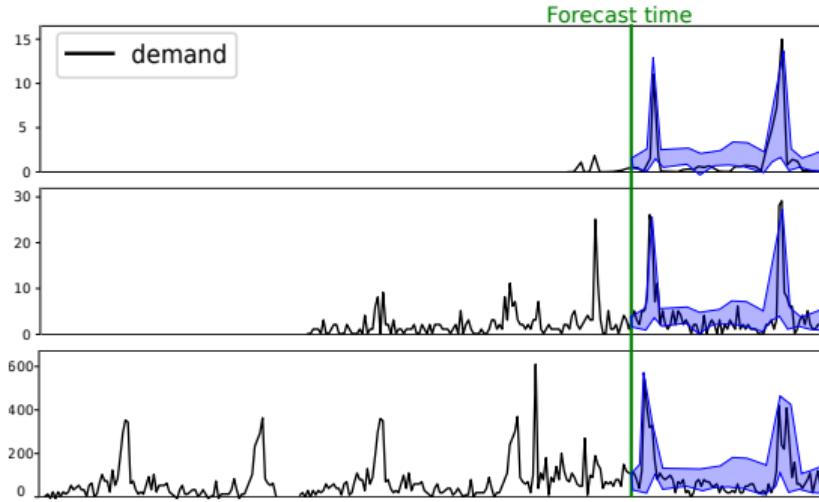
Deep Learning for Forecasting

- Not a single item (time series) but a large collection of related time series
- Exploit information across related time series by sharing parameters *globally*



Deep Learning for Forecasting

- Not a single item (time series) but a large collection of related time series
- Exploit information across related time series by sharing parameters *globally*
- Probabilistic prediction



Neural Networks: use as *global* models

- confusing terminology!
- let Z be a panel of time series, so $Z = \{\mathbf{z}_i = \{z_1, z_2, \dots, z_t\} \in \mathbb{R}^t\}_{i=1, \dots, n}$ all of which we want to forecast. X are co-variates/features.
- *local model*: $P(z_{i,t+1}|z_{i,1}, \dots, z_{i,t}; X_i, \theta_i)$. Important is i in θ_i . A single *local* model per time series (e.g., ETS, ARIMA)
- *global model*: $P(z_{i,t+1}|z_{i,1}, \dots, z_{i,t}; X_i, \theta_i), \quad \theta_i = \Psi(X_i, W)$

Neural Networks: use as *global* models

- confusing terminology!
- let Z be a panel of time series, so $Z = \{\mathbf{z}_i = \{z_1, z_2, \dots, z_t\} \in \mathbb{R}^t\}_{i=1,\dots,n}$ all of which we want to forecast. X are co-variates/features.
- *local model*: $P(z_{i,t+1}|z_{i,1}, \dots, z_{i,t}; X_i, \theta_i)$. Important is i in θ_i . A single *local* model per time series (e.g., ETS, ARIMA)
- *global model*: $P(z_{i,t+1}|z_{i,1}, \dots, z_{i,t}; X_i, \theta_i), \quad \theta_i = \Psi(X_i, W)$
- it gets more complicated!
- we are actually interested in $P(Z_{t+1}|Z_1, \dots, Z_t; X_i, \theta_i), \quad \theta_i = \Psi(X_i, W)$ or *multi-variate models*. We don't cover them today.

Neural Networks: use as *global* models

- each time series z is one instance in the training set
- the weights of the neural network are trained over all time series
- this is the natural way to handle neural networks in other sequence tasks such as natural language processing
- until recently somewhat foreign to forecasters: e.g., 1998 overview article [[Zhang et al., 1998](#)] considered local neural networks, similar to [[Makridakis et al., 2018](#)]
- known under many names: cross-learning, multi-task learning, panel data models,...

Generative vs Discriminative Models for Forecasting

Given a time series: z_1, z_2, \dots, z_T

Generative Model: Assume an underlying data generation process

$$p(z_{1:T}; \theta)$$

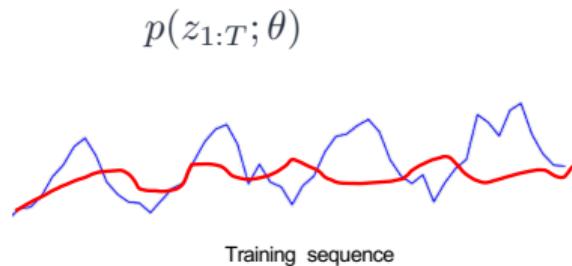
Discriminative Model: Model directly the conditional distribution of future given past

$$p(z_{K+1:T} | z_{1:K}; \theta)$$

Generative vs Discriminative Models for Forecasting

Given a time series: z_1, z_2, \dots, z_T

Generative Model: Assume an underlying data generation process



Discriminative Model: Model directly the conditional distribution of future given past

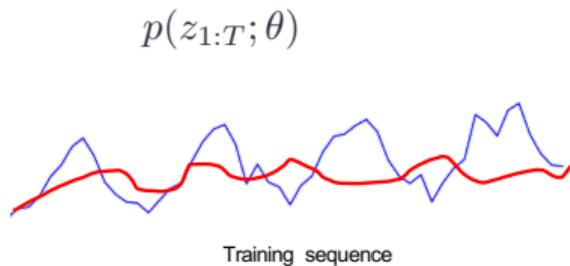
$$p(z_{K+1:T} | z_{1:K}; \theta)$$

Training: find θ that reconstructs observations

Generative vs Discriminative Models for Forecasting

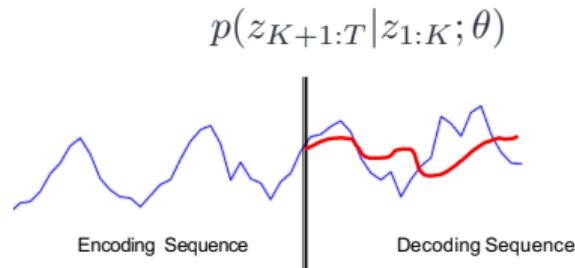
Given a time series: z_1, z_2, \dots, z_T

Generative Model: Assume an underlying data generation process



Training: find θ that reconstructs observations

Discriminative Model: Model directly the conditional distribution of future given past



Training: find θ such that prediction matches future given past

In this workshop...

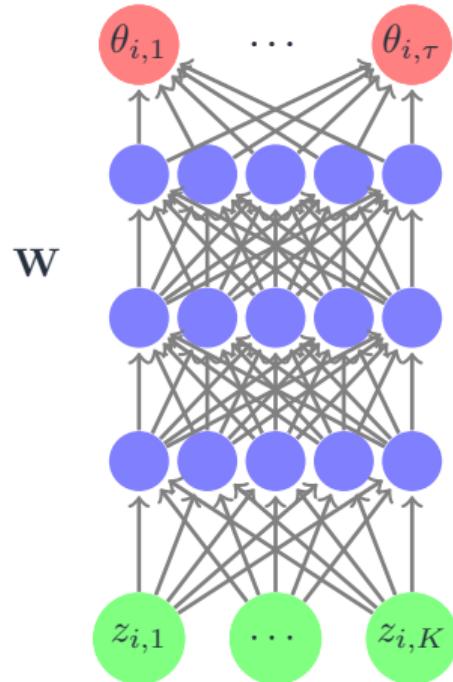
- A simple feed forward neural network model (discriminative)
- DeepAR: Deep Autoregressive Neural network model (generative) [Flunkert et al., 2017]

Topics we will not really cover:

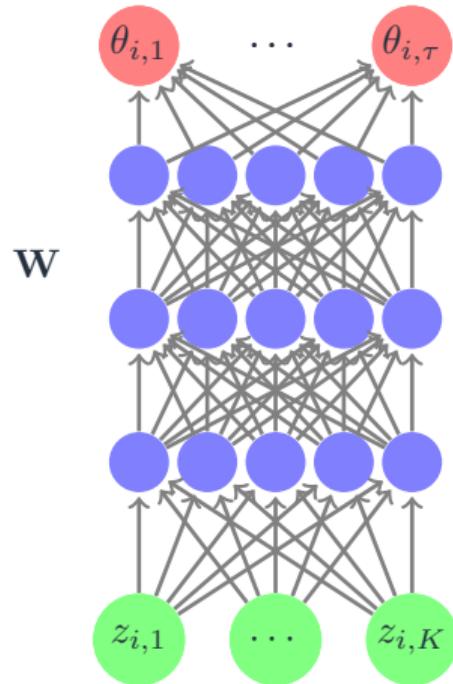
- NBEATS [Oreshkin et al., 2019] (only briefly)
- Transformers [Vaswani et al., 2017] and applications in forecasting [Li et al., 2019; Lim et al., 2019]
- Spline Quantile Forecaster [Gasthaus et al., 2019a]
- Multi-variate models (but there's interesting work [Salinas et al., 2019])
- More Deep Probabilistic models
 - Deep State Space Models for Time Series Forecasting [Rangapuram et al., 2018]
 - Deep Renewal Process for Intermittent Demand Forecasting [Turkmen et al., 2019]
 - Deep Factor Models for Forecasting [Wang et al., 2019]

Simple Feed-Forward Networks for Forecasting

- **Discriminative model:** given past target of length K , predict future distribution for τ time steps



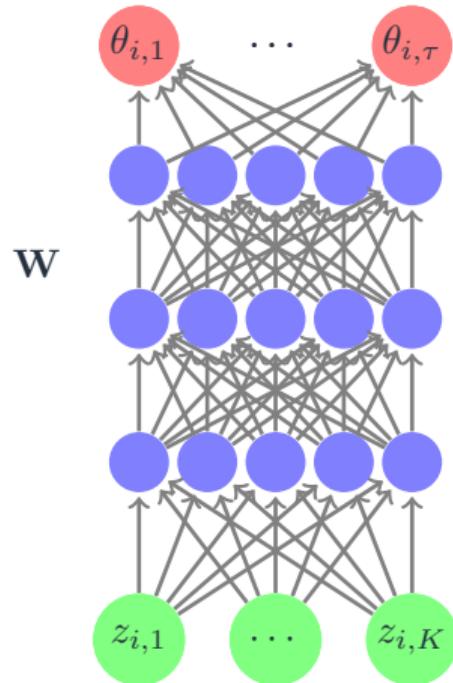
Simple Feed-Forward Networks for Forecasting



- **Discriminative model:** given past target of length K , predict future distribution for τ time steps
- **Output:** parameters of the probabilistic model $\theta_{i,t}$
In case of Gaussian, $\theta_{i,t} = \{\mu_{i,t}, \sigma_{i,t}\}$

Simple Feed-Forward Networks for Forecasting

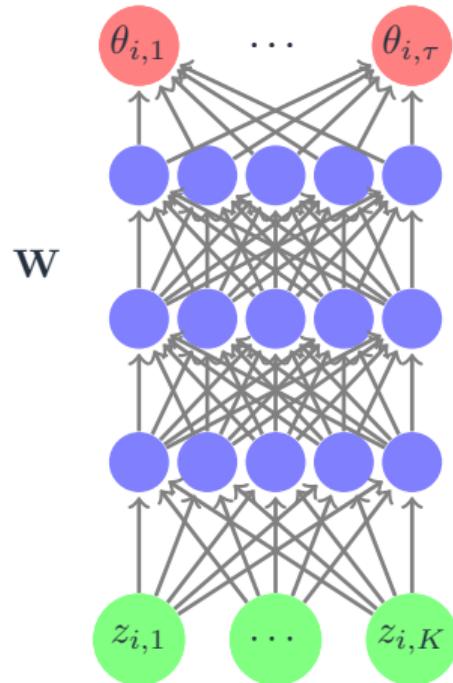
$$\text{Loss: } -\sum_{t=1}^{\tau} \log p(z_{i,K+t}; \theta_{i,t})$$



- **Discriminative model:** given past target of length K , predict future distribution for τ time steps
- **Output:** parameters of the probabilistic model $\theta_{i,t}$
In case of Gaussian, $\theta_{i,t} = \{\mu_{i,t}, \sigma_{i,t}\}$
- **Modelling assumption:** Joint distribution over future is independent over time

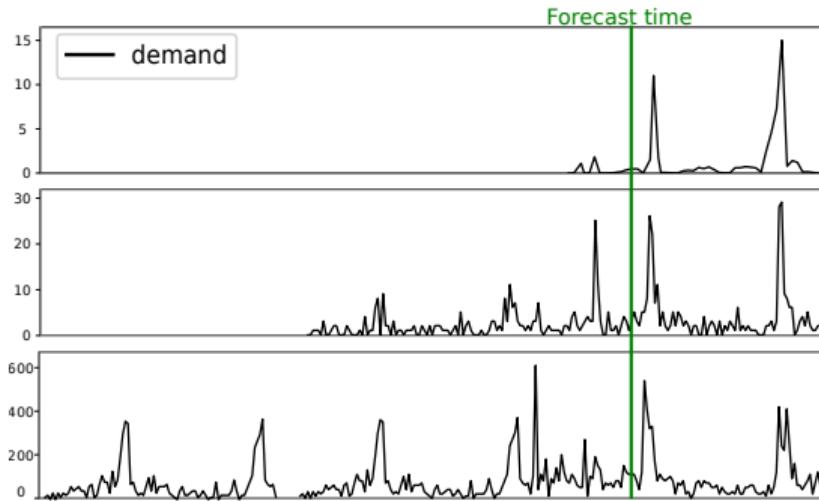
Simple Feed-Forward Networks for Forecasting

$$\text{Loss: } -\sum_{t=1}^{\tau} \log p(z_{i,K+t}; \theta_{i,t})$$

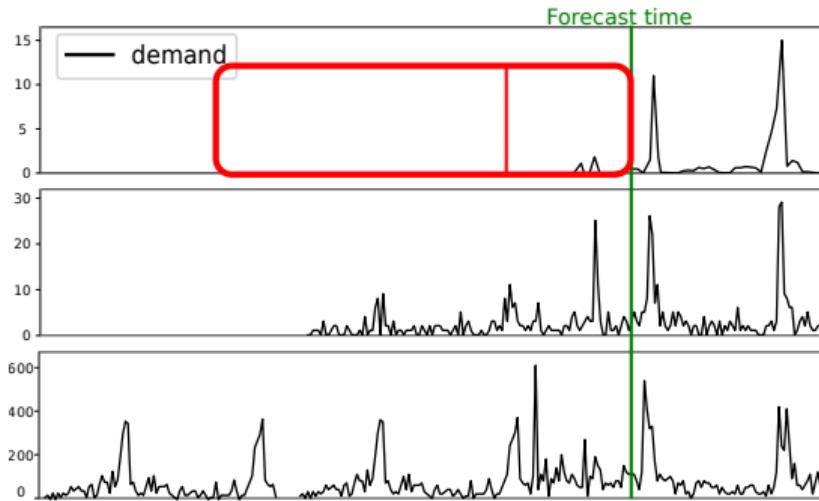


- **Discriminative model:** given past target of length K , predict future distribution for τ time steps
- **Output:** parameters of the probabilistic model $\theta_{i,t}$
In case of Gaussian, $\theta_{i,t} = \{\mu_{i,t}, \sigma_{i,t}\}$
- **Modelling assumption:** Joint distribution over future is independent over time
- **Global training:** Same network with weights \mathbf{W} is used for all time series in the dataset

Training: Augmented Data



Training: Augmented Data

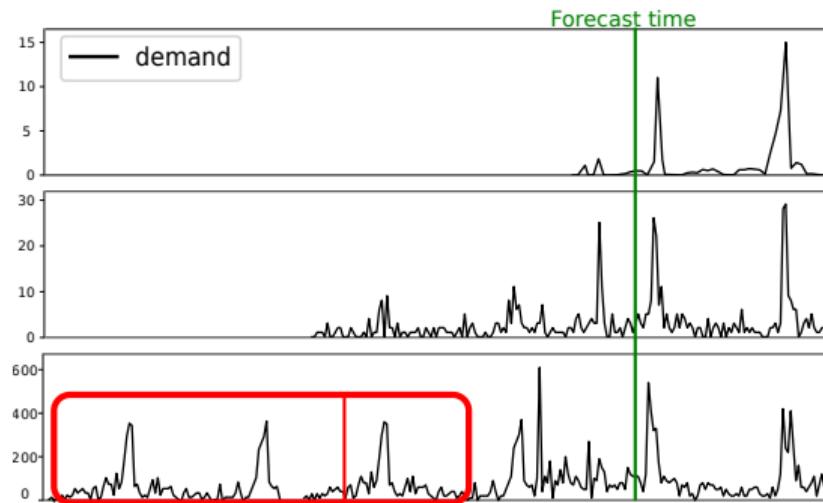


Training: Augmented Data

Training windows: Train on slicing windows across items and time steps

Context versus Prediction: Context target is fed as input to the network; loss is computed on the prediction target

Window size: larger than prediction length (typically at least twice the forecast horizon)

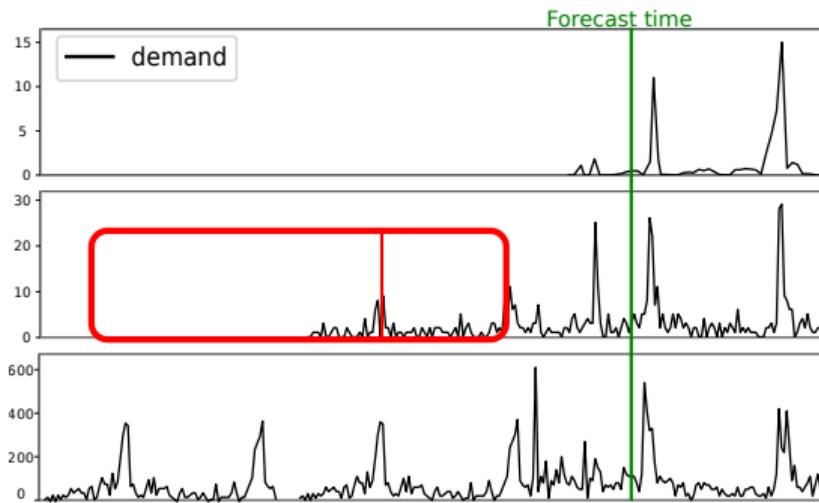


Training: Augmented Data

Training windows: Train on slicing windows across items and time steps

Context versus Prediction: Context target is fed as input to the network; loss is computed on the prediction target

Window size: larger than prediction length (typically at least twice the forecast horizon)

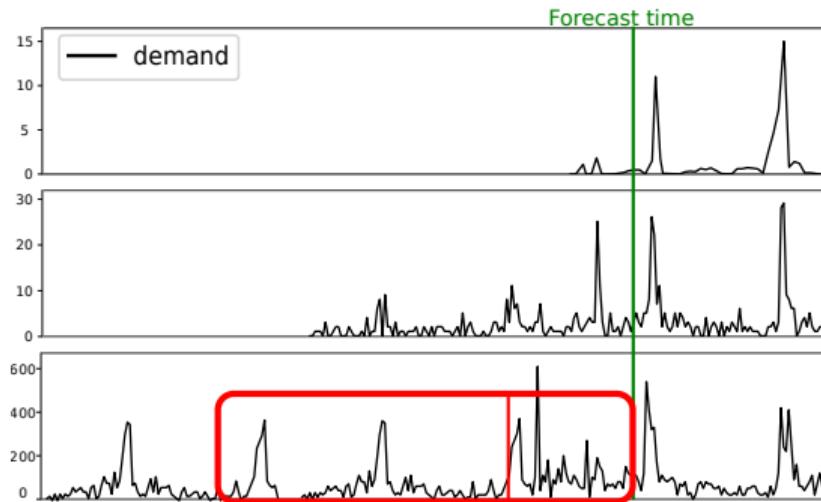


Training: Augmented Data

Training windows: Train on slicing windows across items and time steps

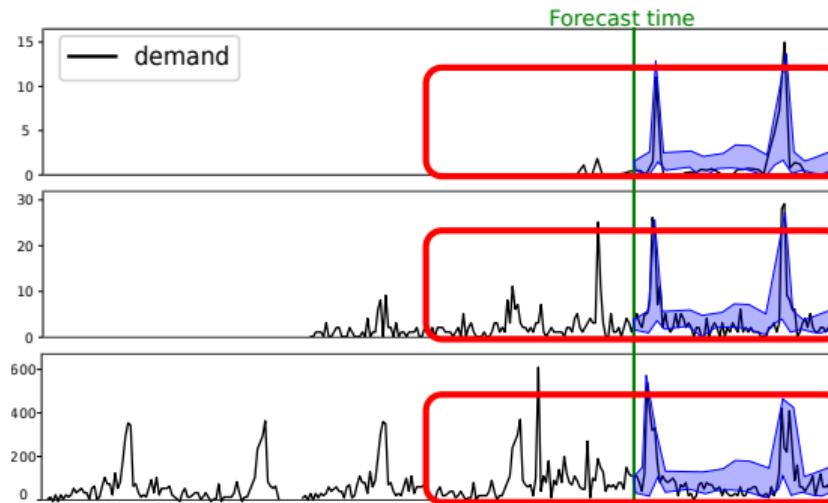
Context versus Prediction: Context target is fed as input to the network; loss is computed on the prediction target

Window size: larger than prediction length (typically at least twice the forecast horizon)



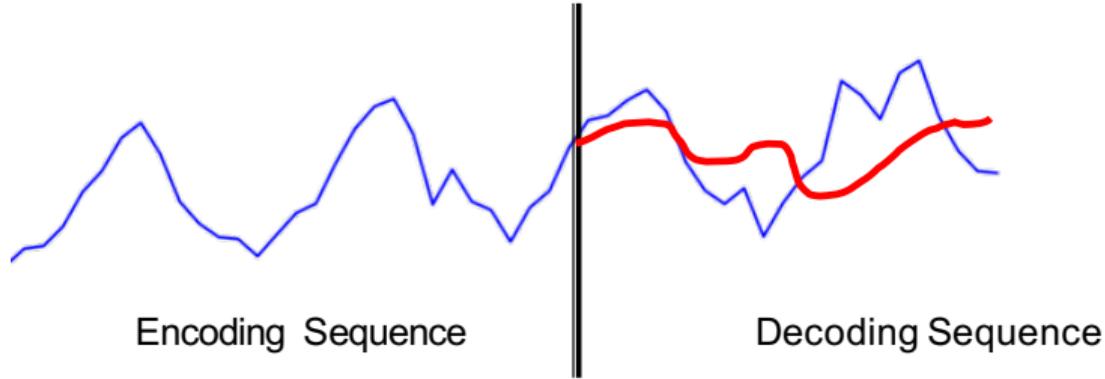
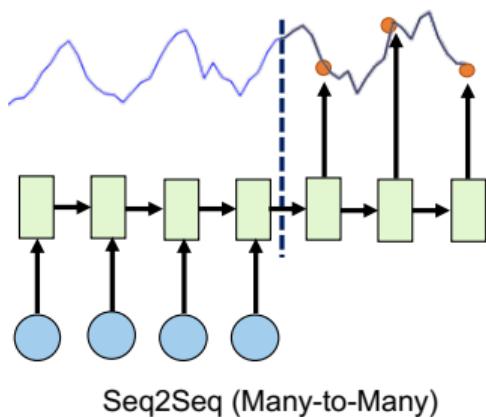
Prediction: Augmented Data

Prediction: Align the window so that start of prediction range coincides with forecast start time

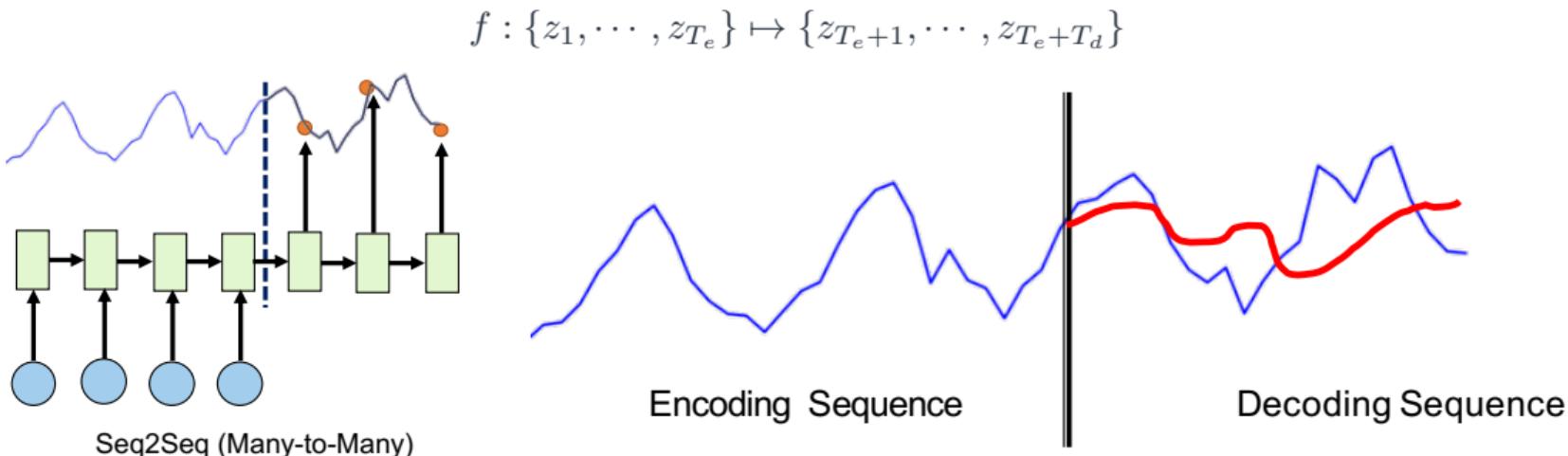


Discriminative Model

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



Discriminative Model

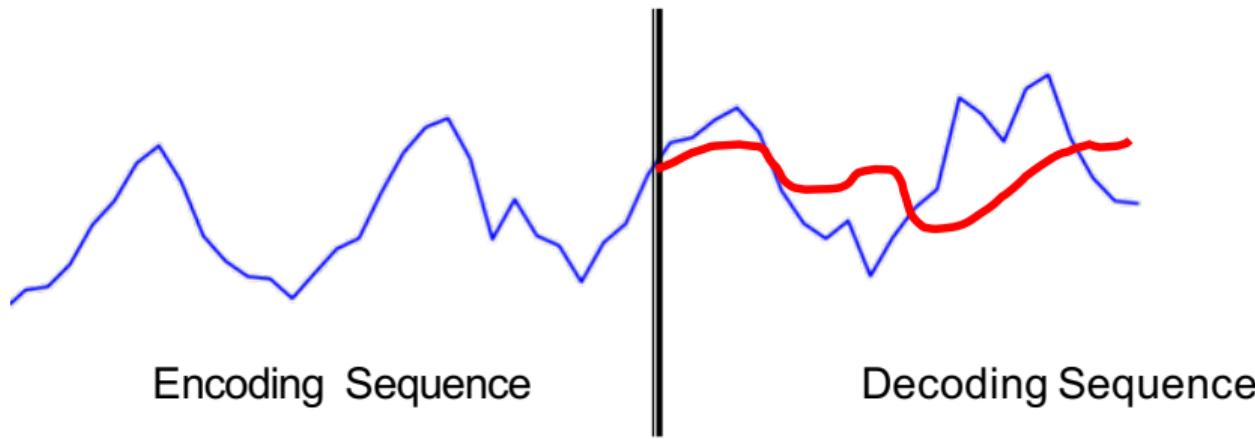


- How well does the *prediction* reconstruct the *decoding sequence* conditioned on the *encoding sequence*?
- Conceptually close to *multivariate regression*
- basic idea: encoder network summarizes encoding sequence in a high-dimensional vector; decoder network takes the high-dimensional vector as input and decodes it into the forecast

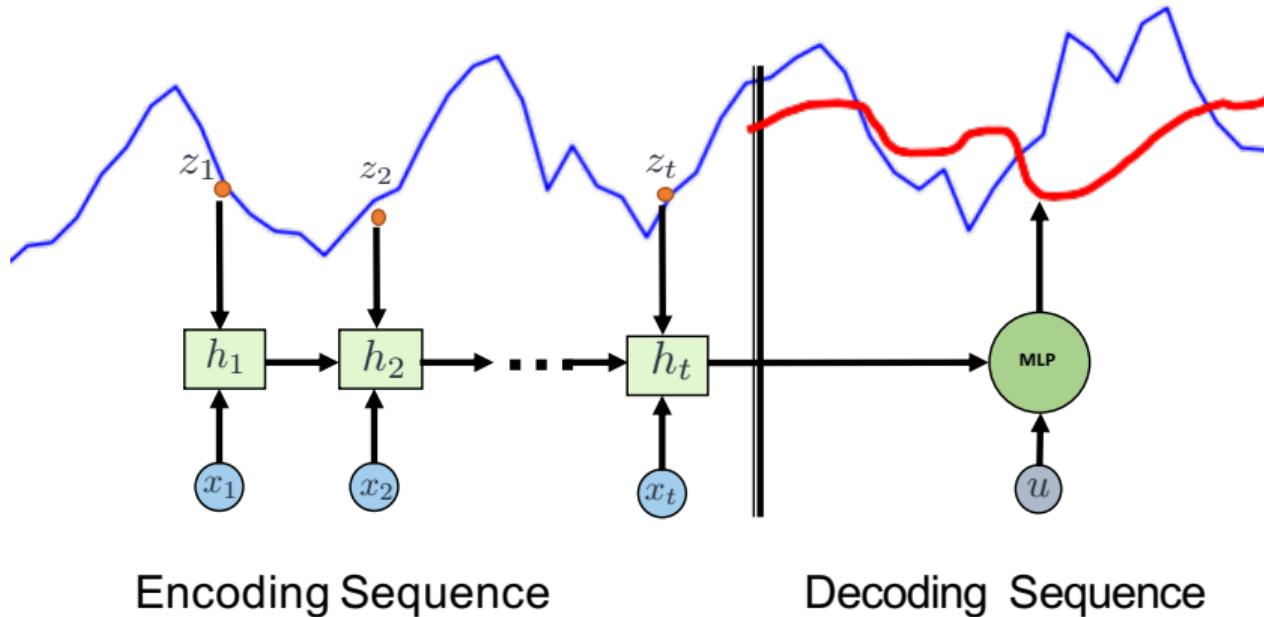
Sequence to Sequence (Seq2Seq) Structure: Many-to-Many

$$f_{encoder} : \{z_1, \dots, z_{T_e}\} \mapsto \mathbf{h}_{T_e}$$

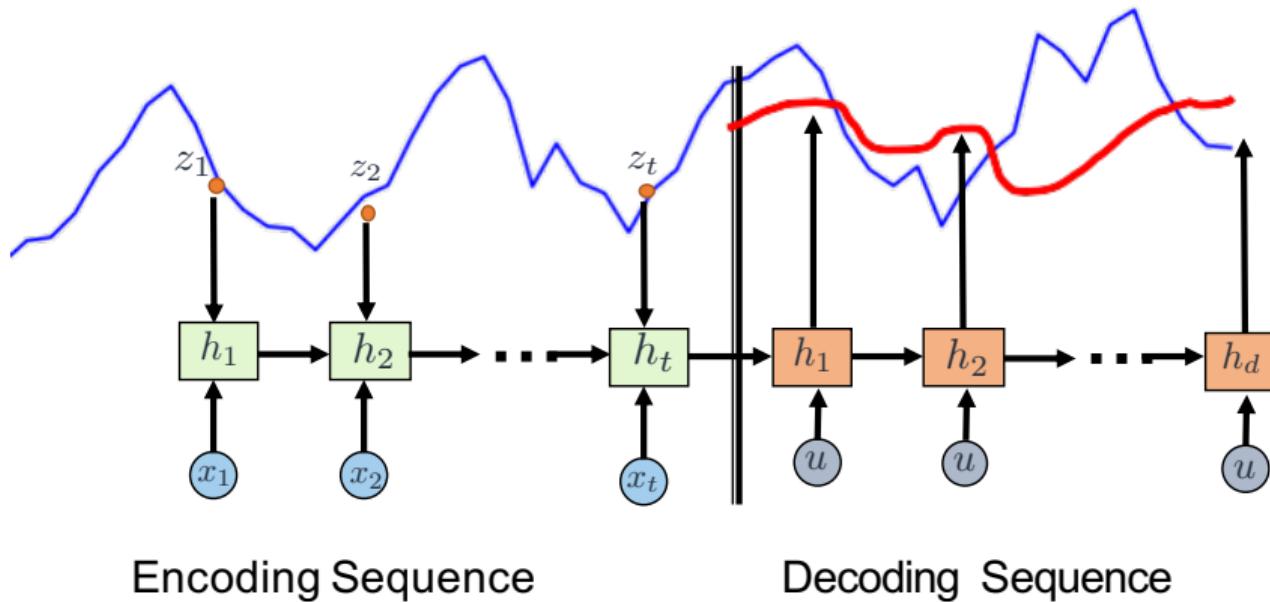
$$f_{decoder} : \mathbf{h}_{T_e} \mapsto \{z_{T_e+1}, \dots, z_{T_e+T_d}\}$$



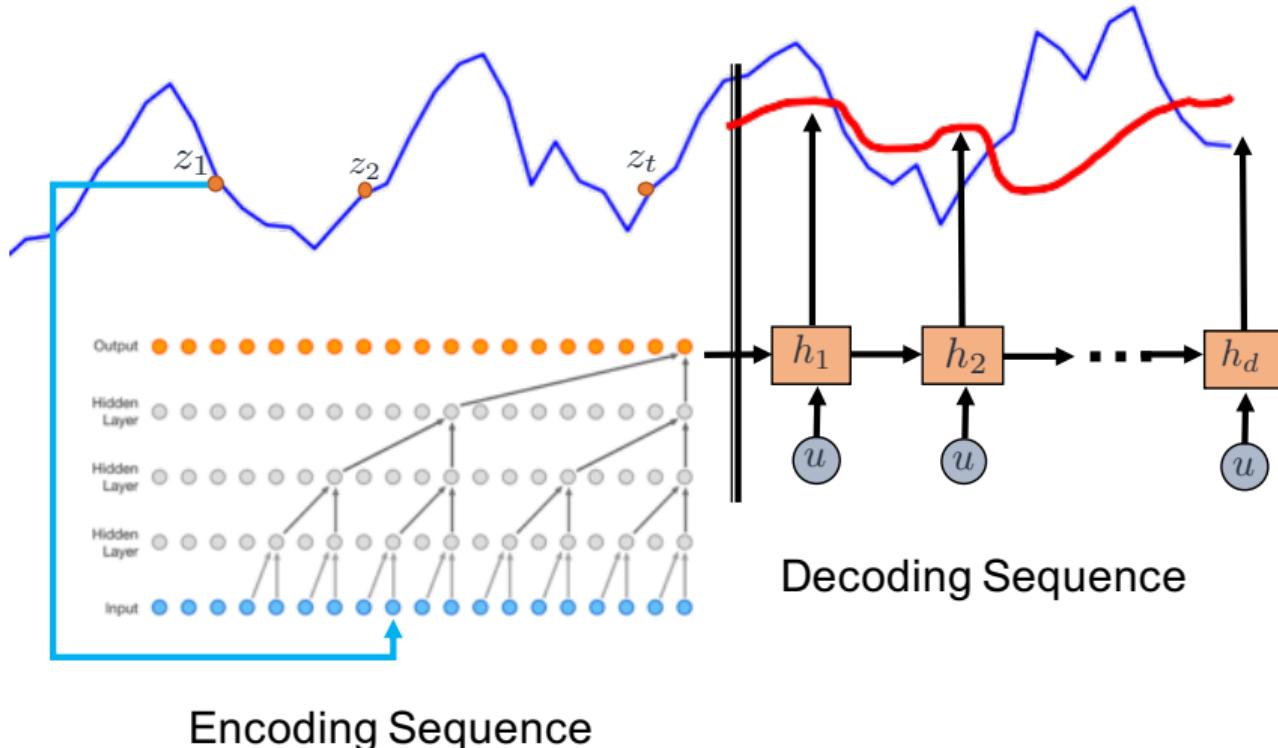
Seq2Seq: RNN-MLP [Wen et al., 2017]



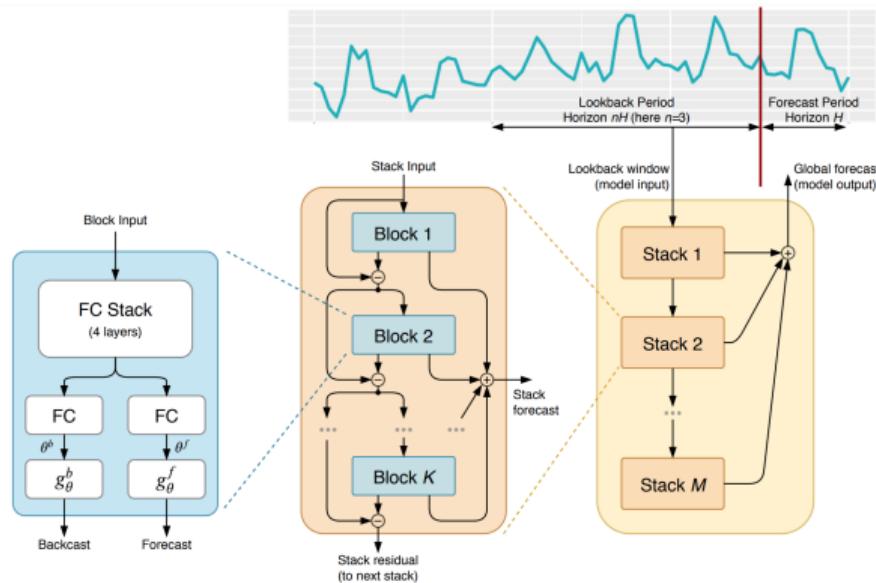
Seq2Seq: RNN-RNN



Seq2Seq: Causal CNN-RNN



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



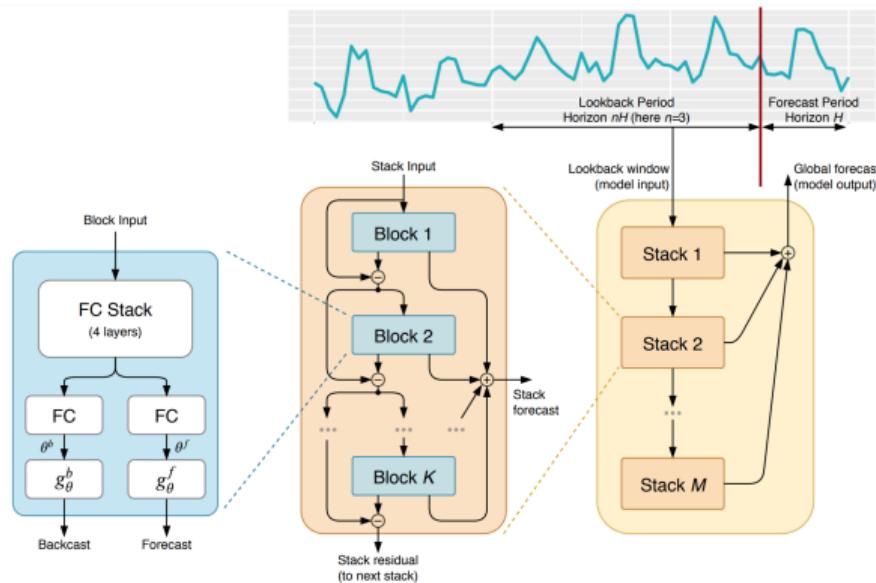
key idea (i) : decompose time series into basis functions

key idea (ii) : residual stacking

key idea (iii) : forecast & backcast at the same time

Generic neural network model (no time series specific ingredients), achieves SOTA on M4.

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



key idea (i) : decompose time series into basis functions

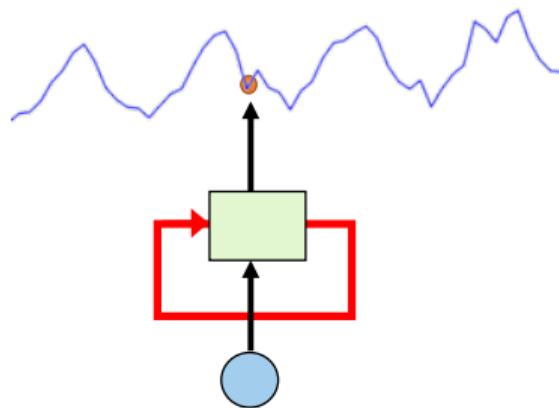
key idea (ii) : residual stacking

key idea (iii) : forecast & backcast at the same time

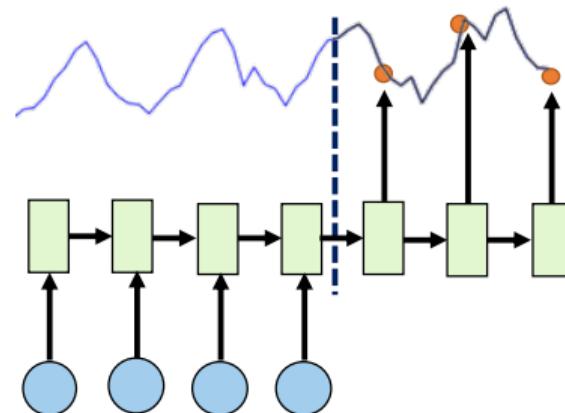
Also some wild ensembling with 180 models with clever tricks (using multiple loss functions).

Generic neural network model (no time series specific ingredients), achieves SOTA on M4.

Generative vs Discriminative (cont'd)



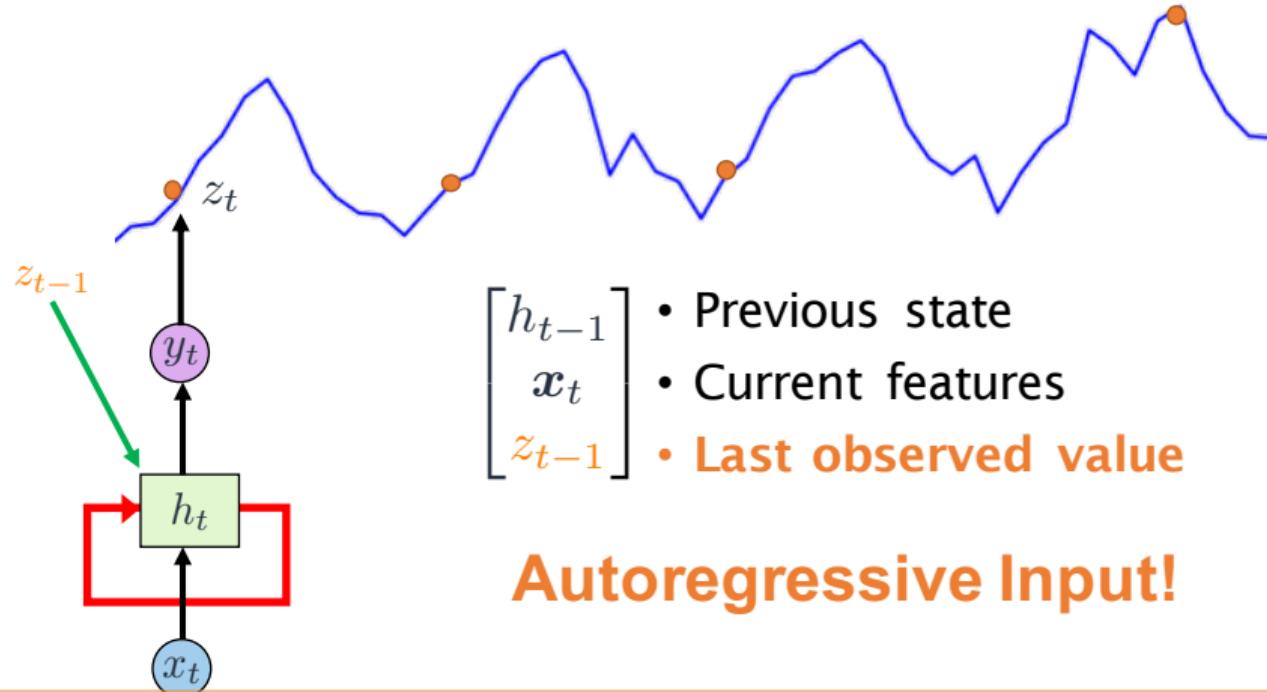
Canonical (One-to-One)



Seq2Seq (Many-to-Many)

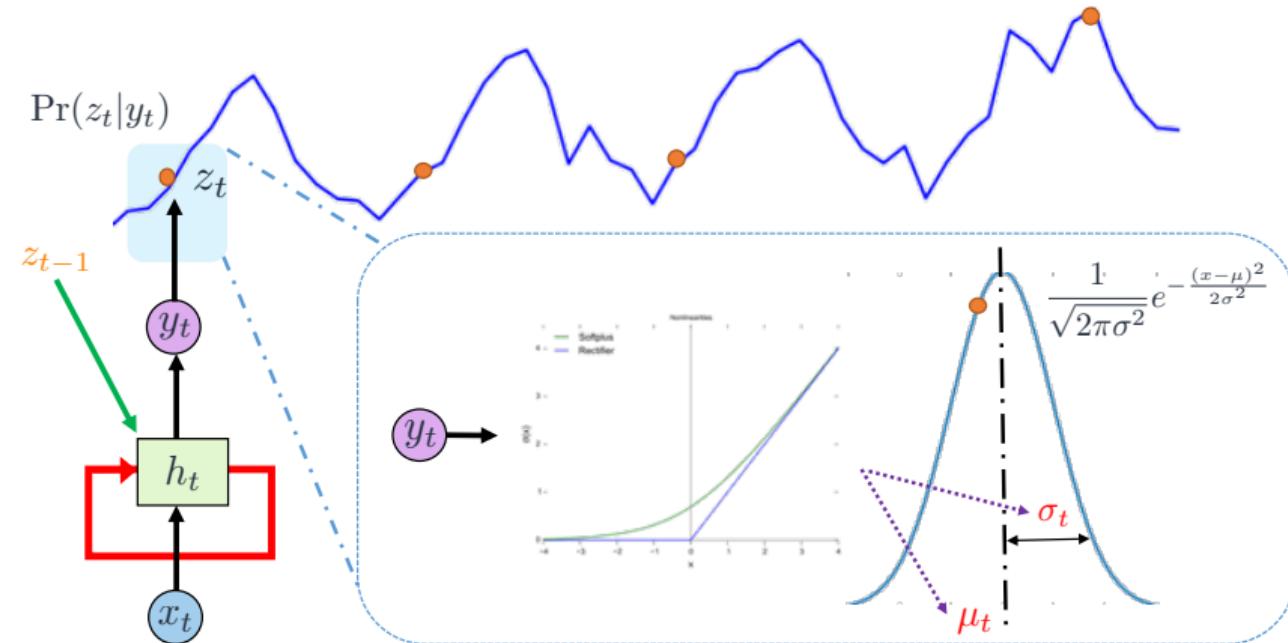
- Generative Models: DeepAR [Flunkert et al., 2017], AR-MDN [Mukherjee et al., 2018], Deep LSTM [Yu et al., 2017], ...
- Sequence-to-Sequence (Many-to-Many) models: Diffusion Convolutional RNNs [Li et al., 2018], MQ-RNN [Wen et al., 2017], ...

DeepAR in a nutshell: A generative model for the time series



Deep Autoregressive Recurrent Neural Network [Flunkert et al., 2017]

DeepAR in a nutshell: A generative model for the time series

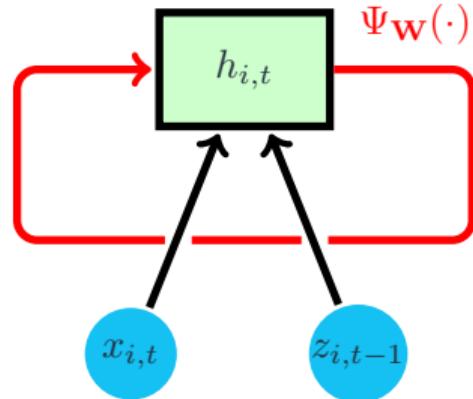


DeepAR: In Detail...,

Generative model for the whole data: $D = \{z_{i,1}, z_{i,2}, \dots, z_{i,T_i}\}_{i=1}^n$

DeepAR: In Detail...,

Generative model for the whole data: $D = \{z_{i,1}, z_{i,2}, \dots, z_{i,T_i}\}_{i=1}^n$



- Transition dynamics:

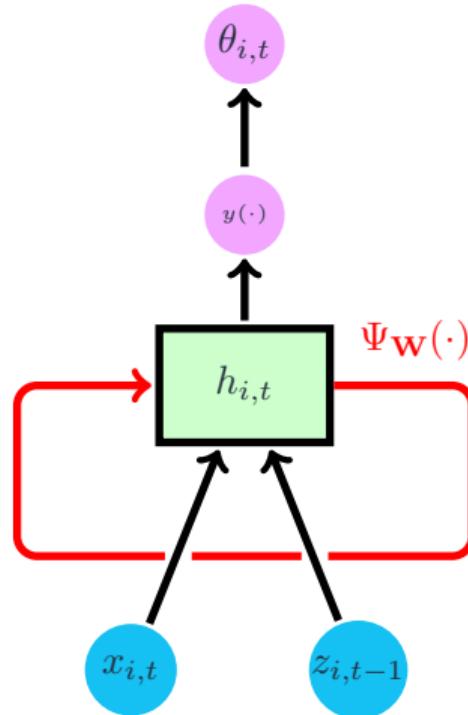
$$h_{i,t} = \Psi_{\mathbf{W}}(x_{it}, z_{i,t-1}, h_{t-1})$$

RNN weights \mathbf{W} are independent of items and time i, t .

DeepAR: In Detail...,

Generative model for the whole data: $D = \{z_{i,1}, z_{i,2}, \dots, z_{i,T_i}\}_{i=1}^n$

$$z_{i,t} \sim p(z_{i,t}; \theta_{i,t})$$



- Transition dynamics:

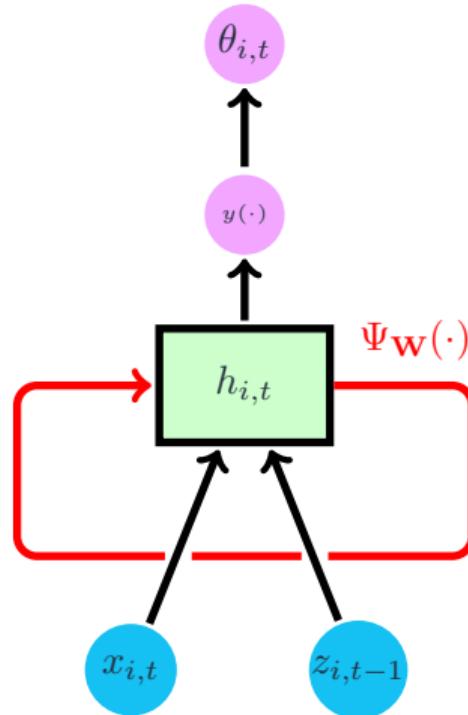
$$h_{i,t} = \Psi_W(x_{it}, z_{i,t-1}, h_{t-1})$$

RNN weights \mathbf{W} are independent of items and time i, t .

DeepAR: In Detail...,

Generative model for the whole data: $D = \{z_{i,1}, z_{i,2}, \dots, z_{i,T_i}\}_{i=1}^n$

$$z_{i,t} \sim p(z_{i,t}; \theta_{i,t})$$



- Observation model:

$$z_{i,t} \sim p(z_{i,t}; \theta_{i,t}) \quad t = 1, 2, \dots, T_i$$

$$\theta_{i,t} = y(h_{i,t})$$

- Transition dynamics:

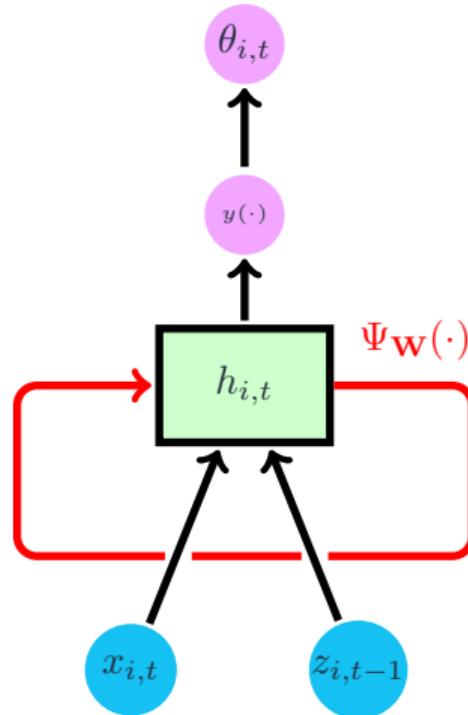
$$h_{i,t} = \Psi_{\mathbf{W}}(x_{it}, z_{i,t-1}, h_{t-1})$$

RNN weights \mathbf{W} are independent of items and time i, t .

DeepAR: In Detail...,

Generative model for the whole data: $D = \{z_{i,1}, z_{i,2}, \dots, z_{i,T_i}\}_{i=1}^n$

$$z_{i,t} \sim p(z_{i,t}; \theta_{i,t})$$



- Observation model:

$$z_{i,t} \sim p(z_{i,t}; \theta_{i,t}) \quad t = 1, 2, \dots, T_i$$

$$\theta_{i,t} = y(h_{i,t})$$

- Transition dynamics:

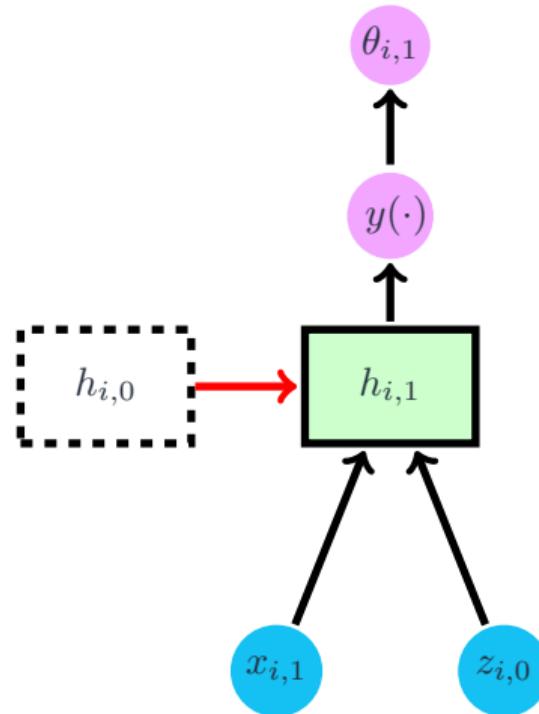
$$h_{i,t} = \Psi_{\mathbf{W}}(x_{it}, z_{i,t-1}, h_{t-1})$$

RNN weights \mathbf{W} are independent of items and time i, t .

- More (seasonal) lags $z_{i,\bar{t}}$, $\bar{t} \in L_t$ as features at each time step t

DeepAR: Training

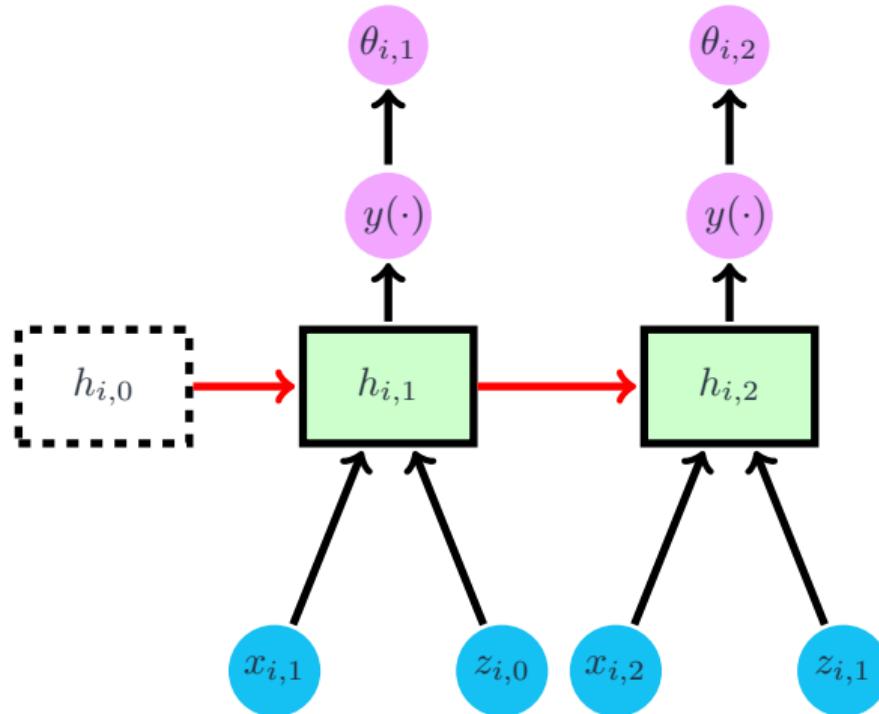
$$-\log p(z_{i,1}; \theta_{i,1})$$



DeepAR: Training

$$-\log p(z_{i,1}; \theta_{i,1})$$

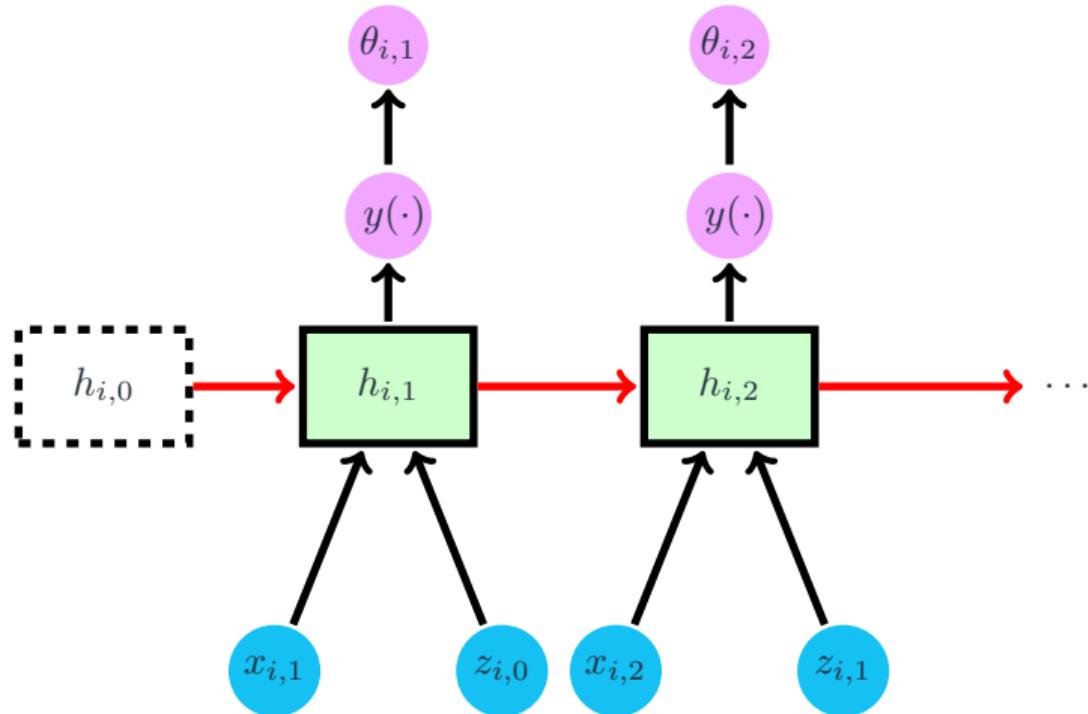
$$-\log p(z_{i,2}; \theta_{i,2})$$



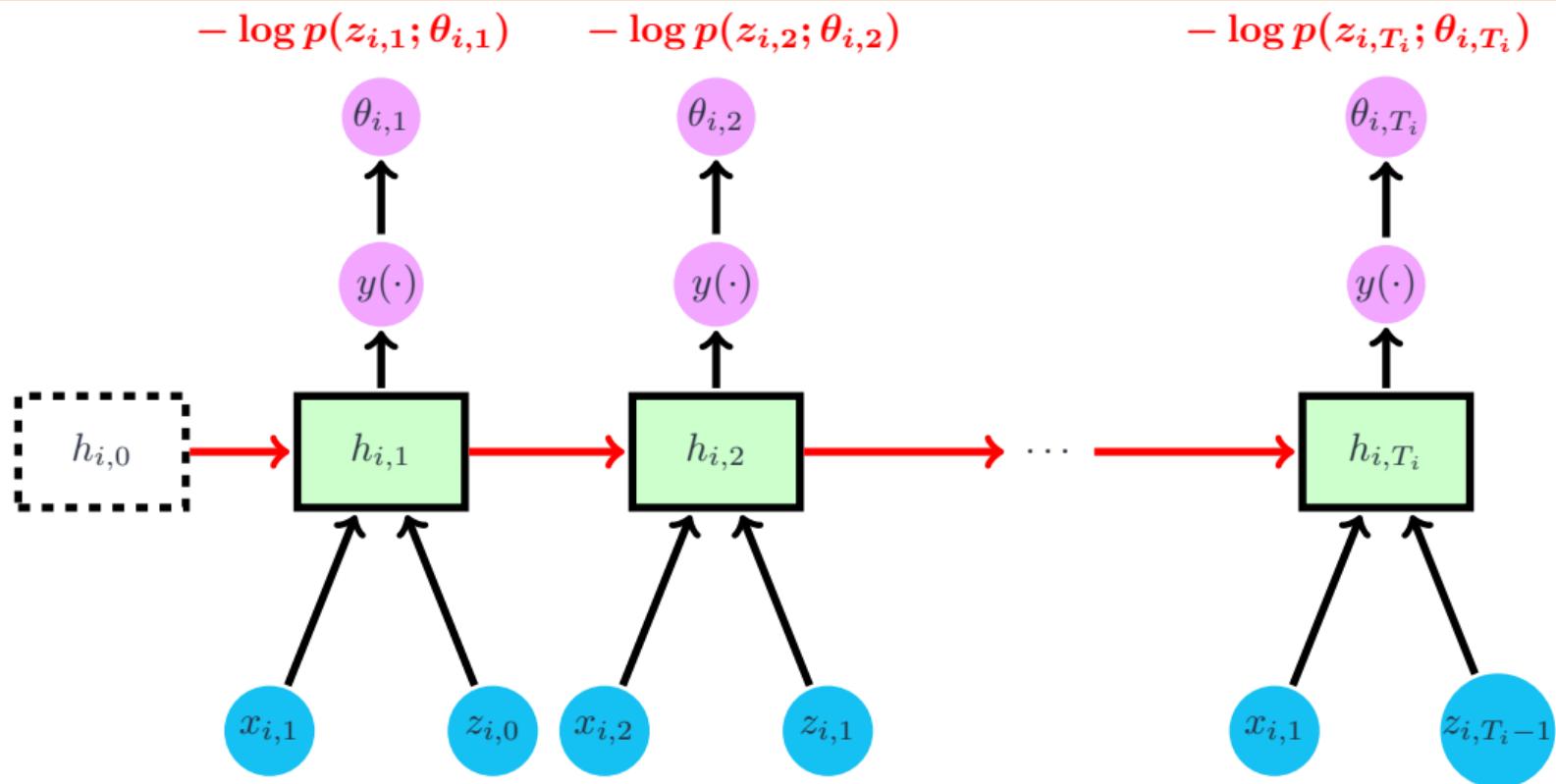
DeepAR: Training

$$-\log p(z_{i,1}; \theta_{i,1})$$

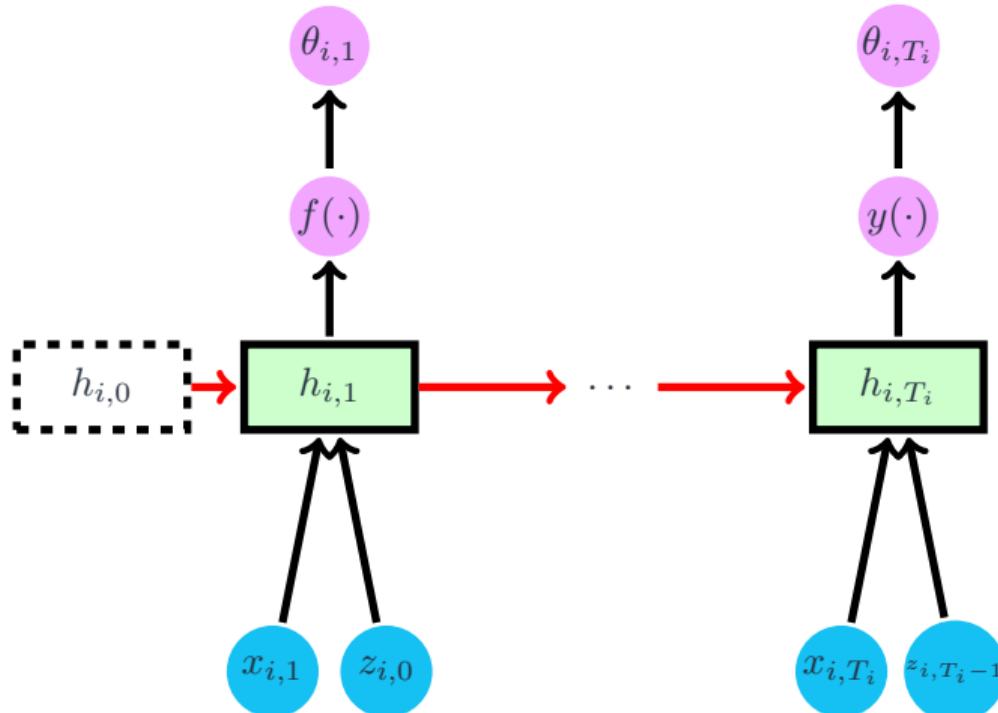
$$-\log p(z_{i,2}; \theta_{i,2})$$



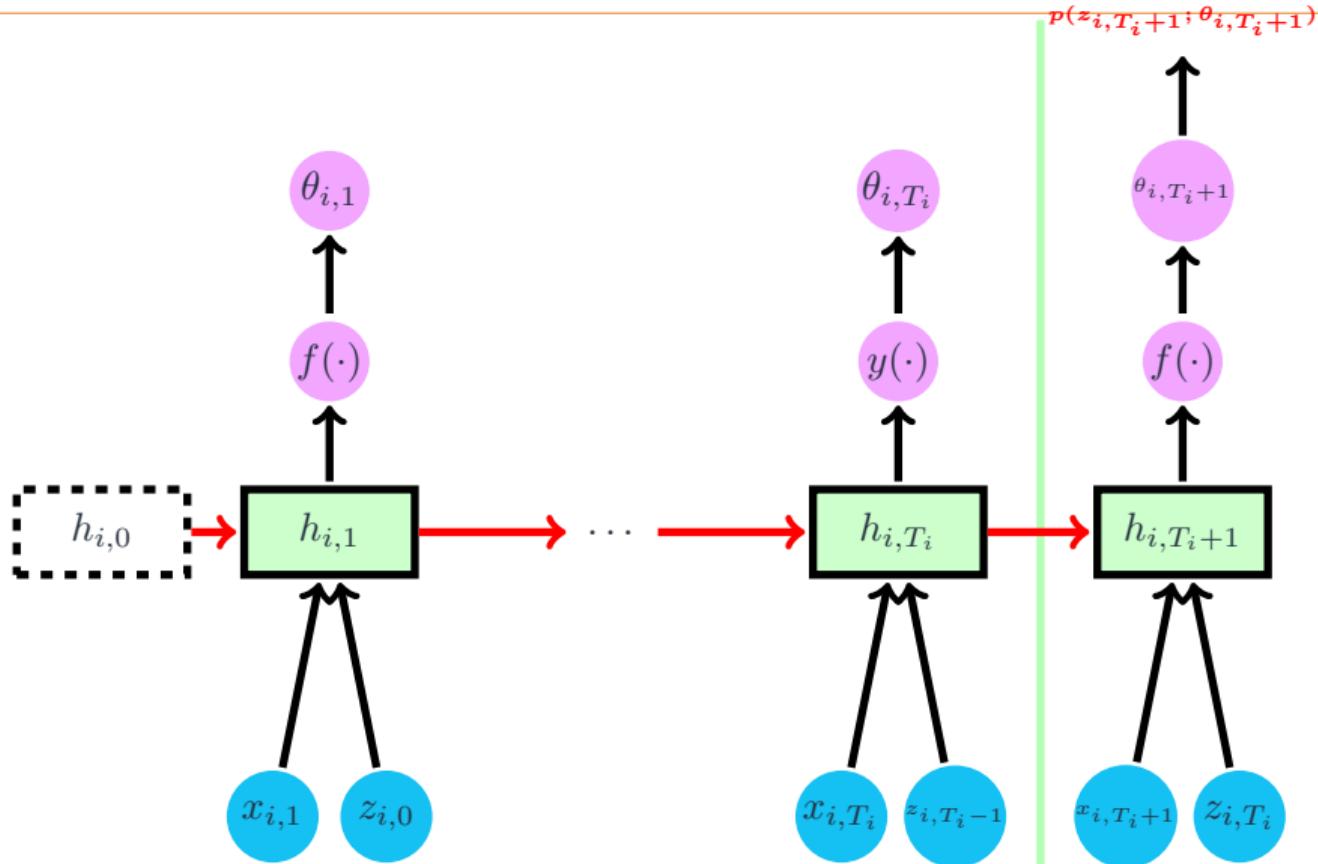
DeepAR: Training



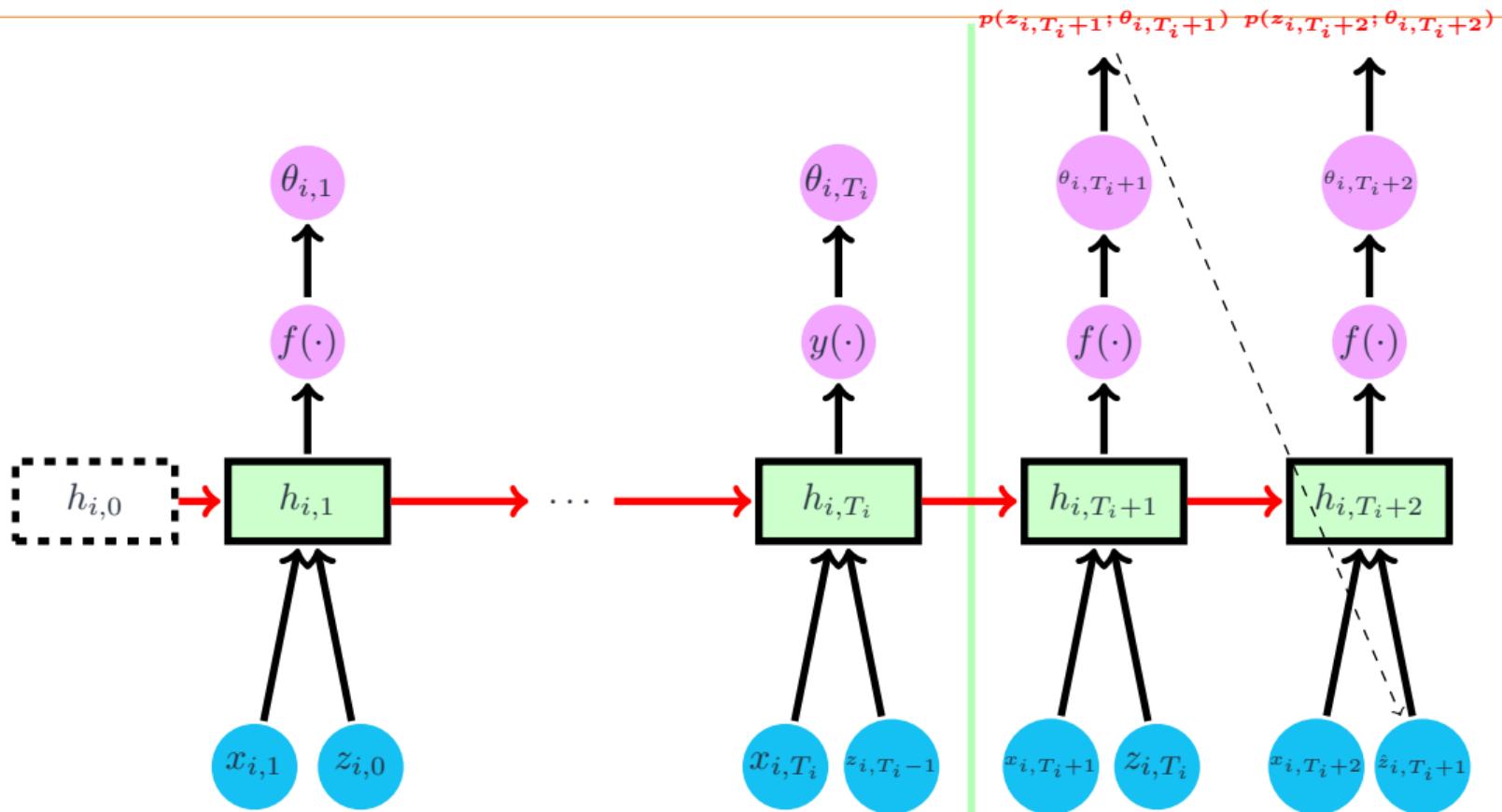
DeepAR: Prediction



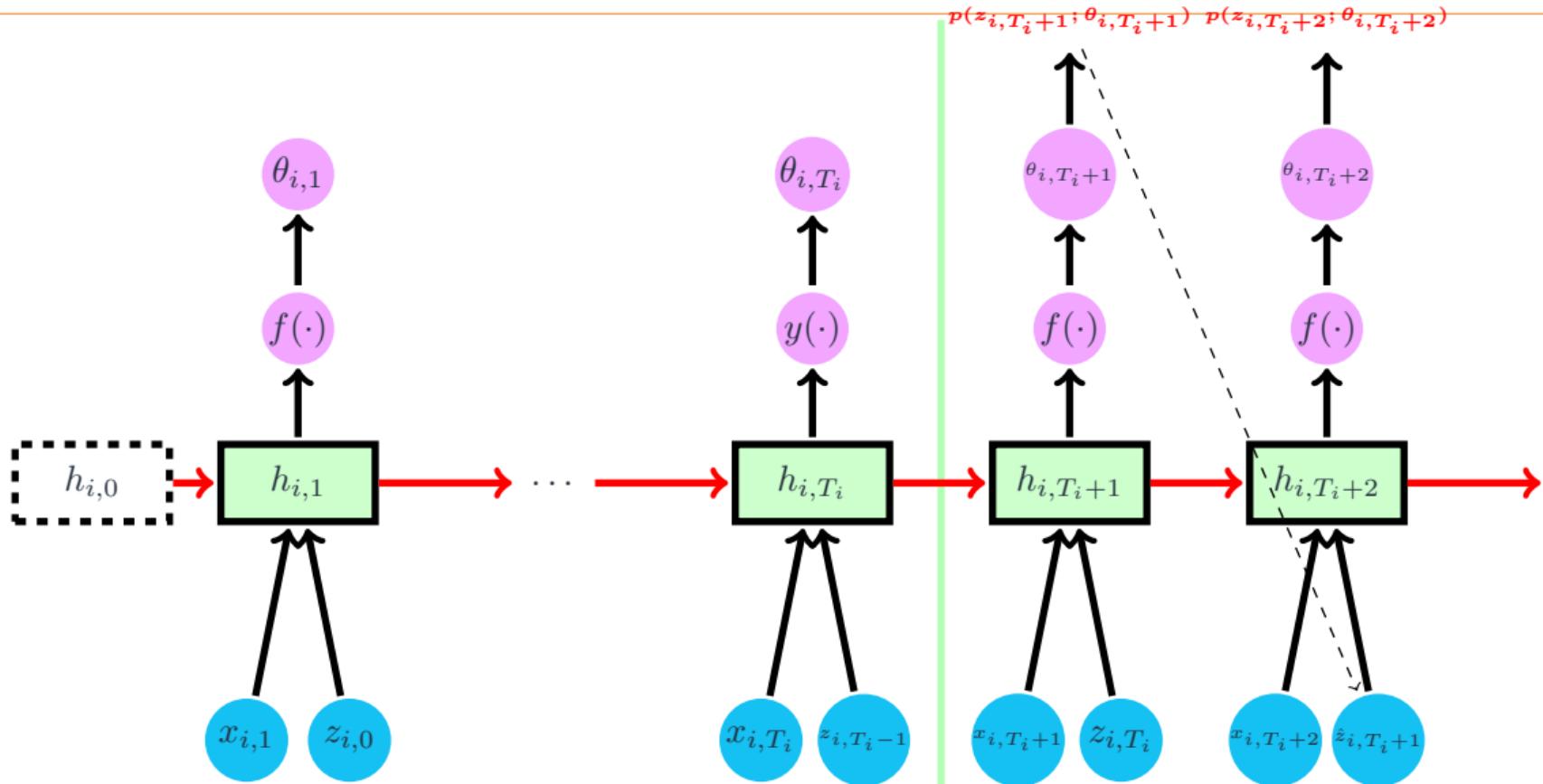
DeepAR: Prediction



DeepAR: Prediction



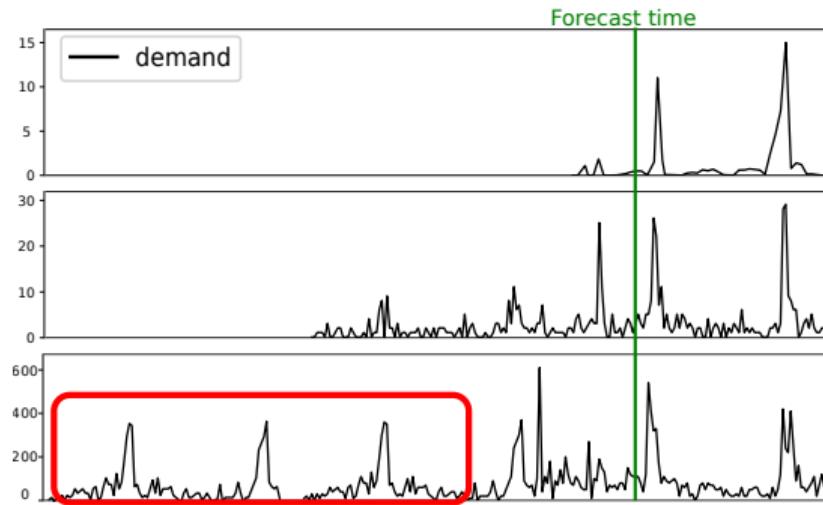
DeepAR: Prediction



DeepAR: Augmented Training

Training windows: Train on slicing windows across items and time steps (RNN is unrolled only on the window)

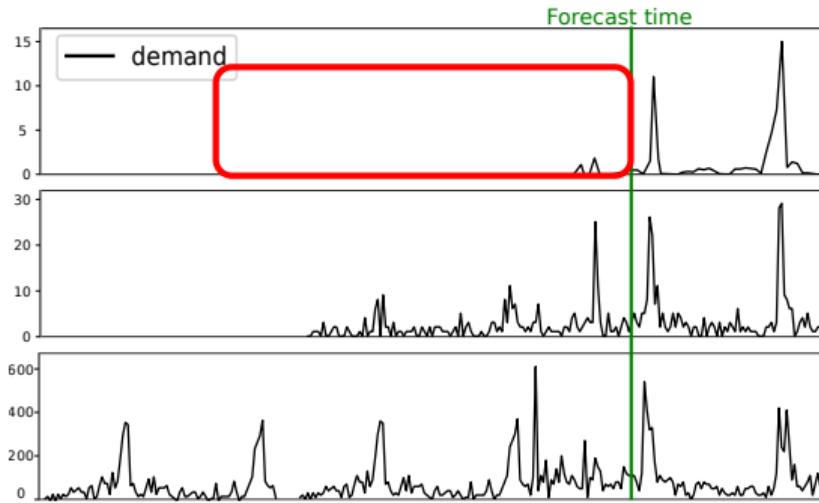
Window size: larger than prediction length (typically at least twice the forecast horizon)



DeepAR: Augmented Training

Training windows: Train on slicing windows across items and time steps (RNN is unrolled only on the window)

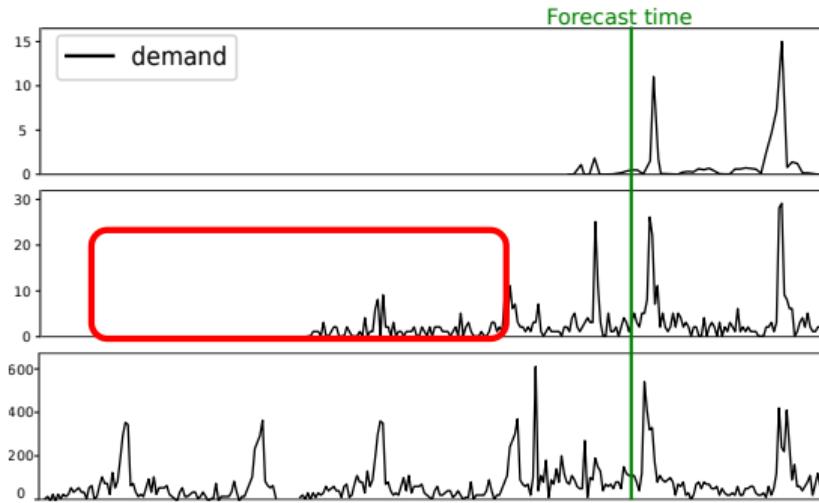
Window size: larger than prediction length (typically at least twice the forecast horizon)



DeepAR: Augmented Training

Training windows: Train on slicing windows across items and time steps (RNN is unrolled only on the window)

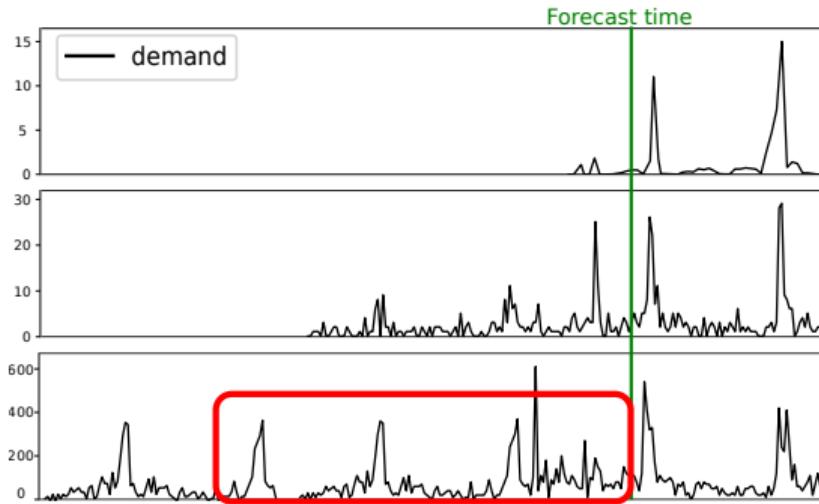
Window size: larger than prediction length (typically at least twice the forecast horizon)



DeepAR: Augmented Training

Training windows: Train on slicing windows across items and time steps (RNN is unrolled only on the window)

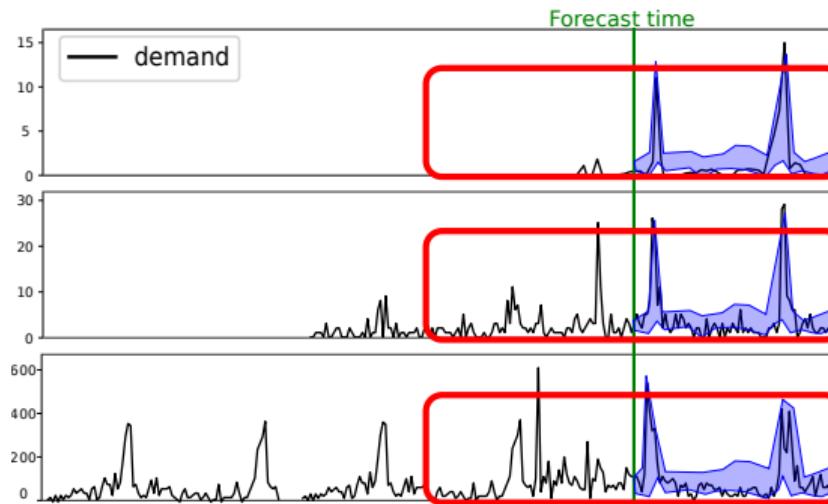
Window size: larger than prediction length (typically at least twice the forecast horizon)



DeepAR: Prediction

Prediction: Align the window to the end of the forecast horizon

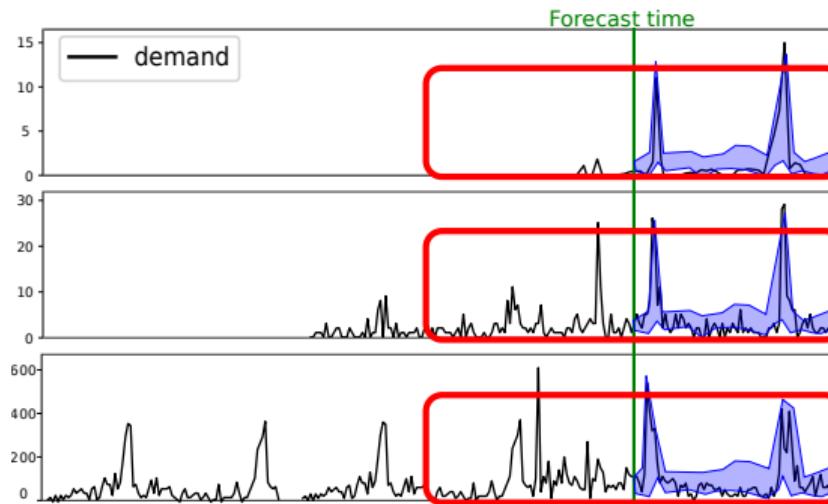
Unroll the RNN before the forecast time usually and start making predictions from the forecast start date



DeepAR: Prediction

Prediction: Align the window to the end of the forecast horizon

Unroll the RNN before the forecast time usually and start making predictions from the forecast start date



(RNN always unrolled for same number of time steps both during training and prediction!)

Classification of Forecasting Approaches in Deep Learning

$$z_1, z_2, \dots, z_T \implies P(z_{T+1}, z_{T+2}, \dots z_{T+\tau})$$

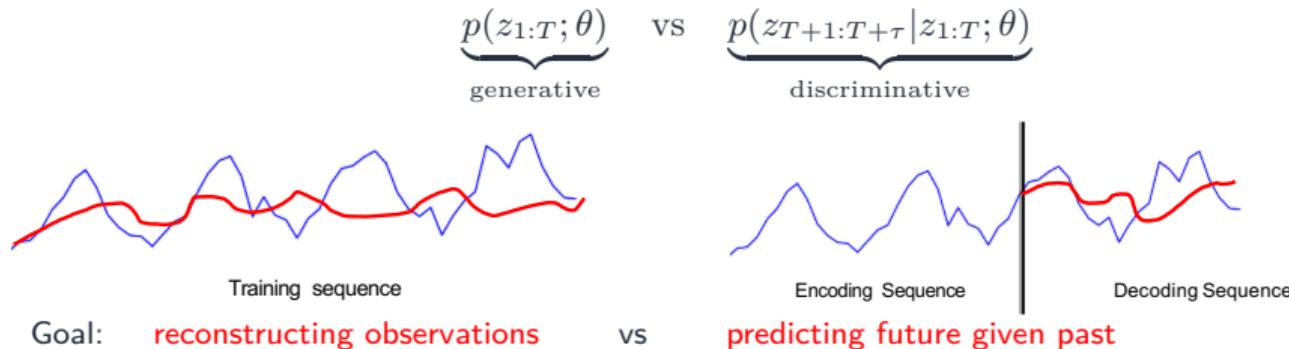
- What choices do I have for modelling observations?

Classification of Forecasting Approaches in Deep Learning

$$z_1, z_2, \dots, z_T \implies P(z_{T+1}, z_{T+2}, \dots z_{T+\tau})$$

- What choices do I have for modelling observations?

1. **What do you model?** Joint distribution of past and future vs conditional distribution of future given past

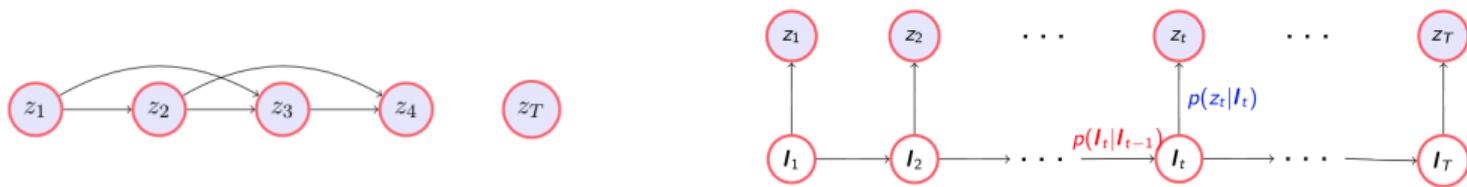


Classification of Forecasting Approaches in Deep Learning

$$z_1, z_2, \dots, z_T \implies P(z_{T+1}, z_{T+2}, \dots, z_{T+\tau})$$

- What choices do I have for modelling observations?

1. **What do you model?** Joint distribution of past and future vs conditional distribution of future given past
2. **How do you model the time series?** Independent, Autoregressive process or State space model



Classification of Forecasting Approaches in Deep Learning

$$z_1, z_2, \dots, z_T \implies P(z_{T+1}, z_{T+2}, \dots z_{T+\tau})$$

- What choices do I have for modelling observations?
 1. **What do you model?** Joint distribution of past and future vs conditional distribution of future given past
 2. **How do you model the time series?** Independent, Autoregressive process or State space model
- Modelling choices for the feature map/architecture:

Classification of Forecasting Approaches in Deep Learning

$$z_1, z_2, \dots, z_T \implies P(z_{T+1}, z_{T+2}, \dots z_{T+\tau})$$

- What choices do I have for modelling observations?
 1. **What do you model?** Joint distribution of past and future vs conditional distribution of future given past
 2. **How do you model the time series?** Independent, Autoregressive process or State space model
- Modelling choices for the feature map/architecture:
 1. MLP
 2. Recurrent neural network
 3. Convolutional neural network/Wavenet
 4. Transformers

Benchmark Open-Source Dataset Results

- GluonTS github repository: <https://github.com/awslabs/gluon-ts>
- Evaluation criteria:

$$\text{CRPS} = \int_0^1 \text{QUANTILE LOSS}(q) dq$$

(More evaluation metrics available from the open source code! as well as in [Alexandrov and other, 2019])

- DeepAR-Spl Gasthaus et al. [2019b] DeepAR that learns to predict the quantile function directly at each time step; the quantile function is parametrized by piecewise linear splines.

estimator dataset	Auto-ARIMA	Auto-ETS	Prophet	NPTS	Transformer	CNN-QR	DeepAR	DeepAR-Spl	GP
SP500-returns	0.975±0.000	0.982±0.001	0.985±0.001	0.832±0.000	0.836±0.001	0.906±0.006	0.838±0.003	0.836±0.005	0.858±0.000
electricity	0.056±0.000	0.067±0.000	0.094±0.000	0.055±0.000	0.062±0.001	0.081±0.003	0.065±0.007	0.065±0.009	0.111±0.001
m4-Daily	0.024±0.000	0.023±0.000	0.090±0.000	0.145±0.000	0.028±0.000	0.026±0.001	0.028±0.000	0.025±0.002	0.074±0.000
m4-Hourly	0.040±0.001	0.044±0.000	0.043±0.000	0.048±0.000	0.042±0.010	0.064±0.006	0.035±0.006	0.124±0.038	0.132±0.000
m4-Monthly	0.097±0.000	0.099±0.000	0.132±0.000	0.233±0.000	0.134±0.002	0.127±0.002	0.136±0.002	0.106±0.001	0.242±0.000
m4-Quarterly	0.080±0.000	0.078±0.000	0.123±0.000	0.255±0.000	0.095±0.003	0.091±0.000	0.091±0.001	0.081±0.002	0.335±0.000
m4-Weekly	0.050±0.000	0.051±0.000	0.108±0.000	0.296±0.001	0.075±0.005	0.056±0.000	0.072±0.001	0.043±0.001	0.137±0.000
m4-Yearly	0.124±0.000	0.126±0.000	0.156±0.000	0.355±0.000	0.127±0.004	0.121±0.000	0.121±0.001	0.111±0.002	0.455±0.000
parts	1.401±0.002	1.342±0.002	1.637±0.002	1.356±0.002	1.000±0.003	0.901±0.000	0.970±0.005	0.943±0.025	1.591±0.000
traffic	-	0.462±0.000	0.273±0.000	0.162±0.000	0.132±0.010	0.186±0.002	0.127±0.004	0.087±0.001	0.152±0.000
wiki10k	0.610±0.001	0.841±0.001	0.681±0.000	0.452±0.000	0.294±0.008	0.314±0.002	0.295±0.028	0.273±0.007	0.452±0.000

Mean and std CRPS error over 10 runs

What's next?

- Transformer/Attention-based models (unless I'm very mistaken)
- Graph Neural Networks
- GANs (maybe, I'm not yet convinced for time series because we don't need the "discriminator")
- Neural ODEs
- more deep probabilistic models
- much more...

What's next?

- Transformer/Attention-based models (unless I'm very mistaken)
- Graph Neural Networks
- GANs (maybe, I'm not yet convinced for time series because we don't need the "discriminator")
- Neural ODEs
- more deep probabilistic models
- much more...

For an overview of further models, check

out <https://lovvge.github.io/Forecasting-Tutorial-WWW-2020/> and [Benidis et al., 2020]

COFFEE BREAK!



GluonTS github repository: <https://github.com/awslabs/gluon-ts>



References

- Alexandrov, A. and other (2019). GluonTS: Probabilistic Time Series Models in Python. *arXiv preprint arXiv:1906.05264*.
- Benidis, K., Rangapuram, S. S., Flunkert, V., Wang, B., Maddix, D., Turkmen, C., Gasthaus, J., Bohlke-Schneider, M., Salinas, D., Stella, L., Callot, L., and Januschowski, T. (2020). Neural forecasting: Introduction and literature overview.
- Flunkert, V., Salinas, D., and Gasthaus, J. (2017). Deepar: Probabilistic forecasting with autoregressive recurrent networks. *arXiv preprint arXiv:1704.04110*.
- Gasthaus, J., Benidis, K., Flunkert, V., Salinas, D., Wang, Y., and Januschowski, T. (2019a). Probabilistic forecasting with spline quantile function RNNs. In *Proceedings of AISTATS*, volume 89, pages 1901–1910.
- Gasthaus, J., Benidis, K., Wang, Y., Rangapuram, S. S., Salinas, D., Flunkert, V., and Januschowski, T. (2019b). Probabilistic forecasting with spline quantile function rnns. In *Proceedings of Machine Learning Research*, volume 89 of *Proceedings of Machine Learning Research*, pages 1901–1910. PMLR.
- Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.
- 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. In *Advances in Neural Information Processing Systems 32*, pages 5244–5254. Curran Associates, Inc.
- Li, Y., Yu, R., Shahabi, C., and Liu, Y. (2018). Diffusion convolutional recurrent neural network: Data-driven traffic forecasting.
- Lim, B., Arik, S. O., Loeff, N., and Pfister, T. (2019). Temporal fusion transformers for interpretable multi-horizon time series forecasting.
- Makridakis, S., Spiliotis, E., and Assimakopoulos, V. (2018). Statistical and machine learning forecasting methods: Concerns and ways forward. *PLoS one*.
- 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*.

References (cont.)

- Oreshkin, B. N., Carpov, D., Chapados, N., and Bengio, Y. (2019). N-beats: Neural basis expansion analysis for interpretable time series forecasting. *arXiv preprint arXiv:1905.10437*.
- 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*, pages 7785–7794.
- Salinas, D., Bohlke-Schneider, M., Callot, L., Medico, R., and Gasthaus, J. (2019). High-dimensional multivariate forecasting with low-rank gaussian copula processes. In *Advances in Neural Information Processing Systems 32*.
- Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958.
- Turkmen, A. C., Wang, Y., and Januschowski, T. (2019). Intermittent demand forecasting with deep renewal processes.
- van den Oord, A., Dieleman, Sanderand 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 *The 9th ISCA Speech Synthesis Workshop*.
- 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, Ł., and Polosukhin, I. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.
- Wang, Y., Smola, A., Maddix, D., Gasthaus, J., Foster, D., and Januschowski, T. (2019). Deep factors for forecasting. In *International Conference on Machine Learning*, pages 6607–6617.
- Wen, R., Torkkola, K., and Narayanaswamy, B. (2017). A multi-horizon quantile recurrent forecaster. *arXiv preprint 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.
- Zhang, G., Patuwo, B. E., and Hu, M. Y. (1998). Forecasting with artificial neural networks:: The state of the art. *International journal of forecasting*, 14(1):35–62.