MSBA 434: Advanced Workshop on Machine Learning

Lecture 3: Recurrent Neural Network

UCLA Anderson

2019

Danylo Vashchilenko

Agenda

1. Sequences
2. Recurrent Connection
3. Gated Recurrent Unit
4. Long Short-term Memory

5. Case Study: DeepAR
6. Case Study: Weather Forecasting
7. Assignment #3
8. Case Study: MQ-RNN

Tasks with Sequences

Task	Example of Input	Example of Output
Speech recognition	Audio recording	Text in English
Language translation	Text in English	Text in Chinese
Image captioning	Image pixels	Short description in English
Music generation	Seed number	Music notes
Sentiment classification	Product review	Positivity score
Stock price prediction	Historical stock prices	Future stock price

Non-sequences	Temporal observations	Natural language text

Character Sequences

Text	Hello world!											
Character	`H`	`e`	`l`	`l`	`o`		`w`	`o`	`r`	`l`	`d`	`!`
Vector	72	101	108	108	111	32	119	111	114	108	100	33

Pros

1. Simple to convert to and from integer representation.
2. Any language, including programming, markup, math, etc.

Cons

1. Requires to learn words are useful higher-level concept.
2. Vector differences are meaningless.

Case Study: http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Word Sequences

Text	Hello world!			
Token	**`hello`**		**`world`**	**`!`**
Vector	6,543	10	9,876	4

Pros

1. Speeds up training due to operating on a higher level concept than characters.

Cons

1. Requires a word dictionary, and a tokenization algorithm to deal with capitalization, spelling errors, etc.
2. Vector differences are meaningless.

Embedding Sequences

Text	Hello world!			
Token	**`hello`**		**`world`**	**`!`**
Vector	[0.1, 0.33, 0.5, 0.2, … , 0.2]	[...]	[0.2, 0.1, 0.01, 0.8, … , 0]	[...]

Pros

1. Vector differences are meaningfully representing relationships between words.

Cons

1. Requires embedding dictionary, and a tokenization algorithm to deal with capitalization, spelling errors, etc.
2. Words have multiple meanings.

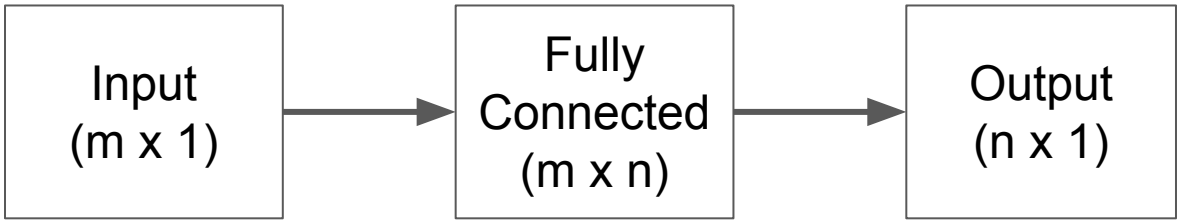
Case Study: https://projector.tensorflow.org/

Temporal Sequences

Measurement	Frequency	Duration	Sequence Shape
Temperature in LA	1 day	10 years	~3.6K x 1
Audio recording	44 kHz	1 minute	~2.6M x 1
Earth vibrations	4 MHz	2 minutes	480M x 1
Price of every stock in S&P 500	1 Hz	1 week	117K x 500 (= ~60M)
Purchase activity by 1M customers	1 day	1 year	365 x 1M (= 365M)

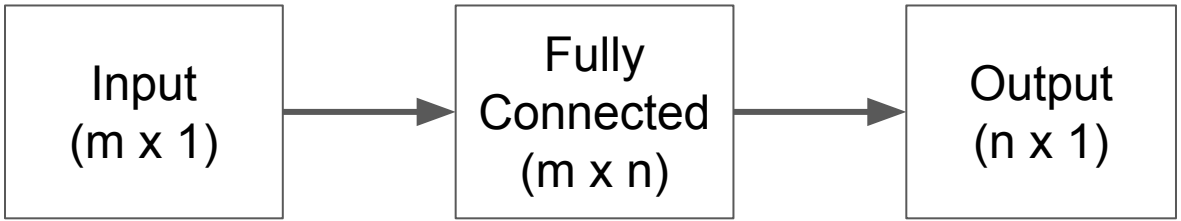
1-dimensional sequence	2-dimensional sequence

What about Fully Connected Layer?



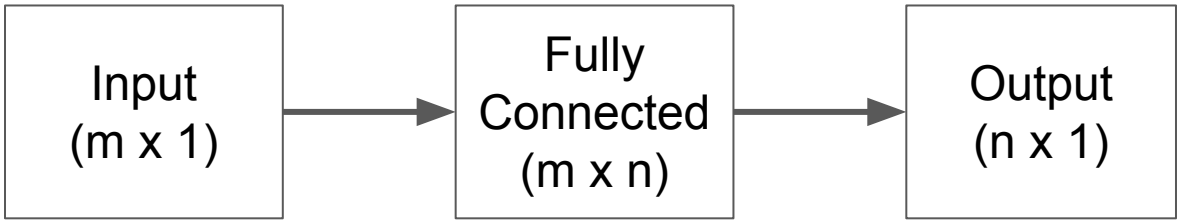
Problem 1
Count of parameters grows proportionally to the sequence length. Density of information might be too low to give the model such capacity.

What about Fully Connected Layer?



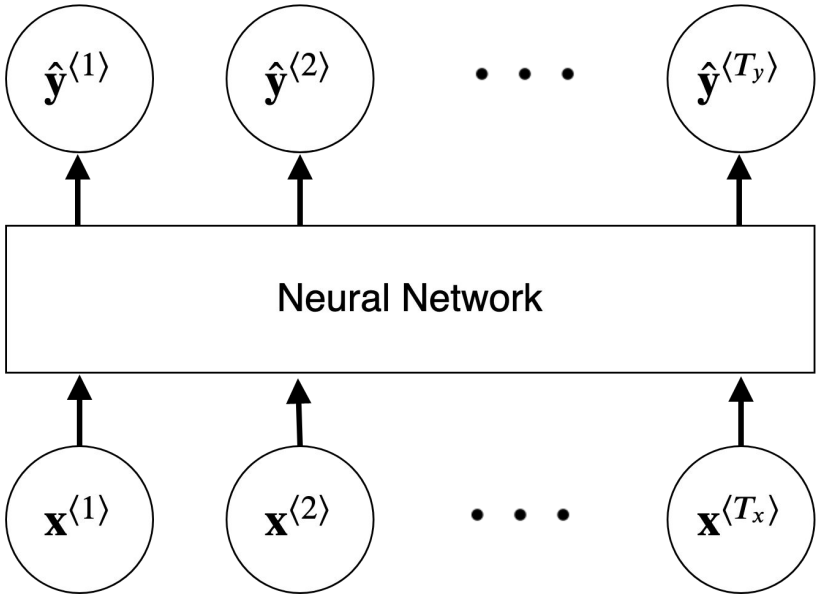
Problem 2
In any sequence, nearby items are related, and their relationship is location-invariant. Fully connected layer treats every item independently, until it learns otherwise.

What about Fully Connected Layer?



Problem 3
Sequence length can vary across examples within the same task. For fully connected layers, shape of input and output is frozen at design time.

Sequence to Sequence

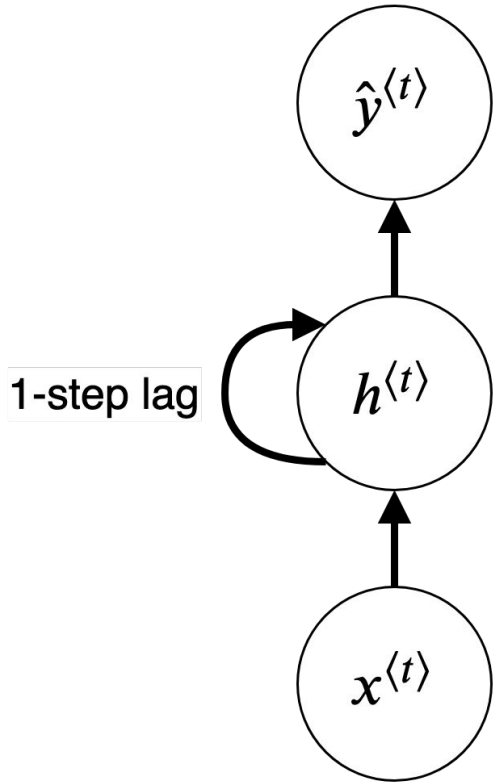


$$\mathbf{x}^{\langle t \rangle} \in \mathbb{R}^{n}, t \in \{1, 2, \ldots, T_x\}$$

$$\hat{\mathbf{y}}^{\langle t \rangle} \in \mathbb{R}^{m}, t \in \{1, 2, \ldots, T_y\}$$

Recurrent Connection

$$\mathbf{h}^{\langle t \rangle} = f(\mathbf{h}^{\langle t-1 \rangle}, \mathbf{x}^{\langle t \rangle})$$

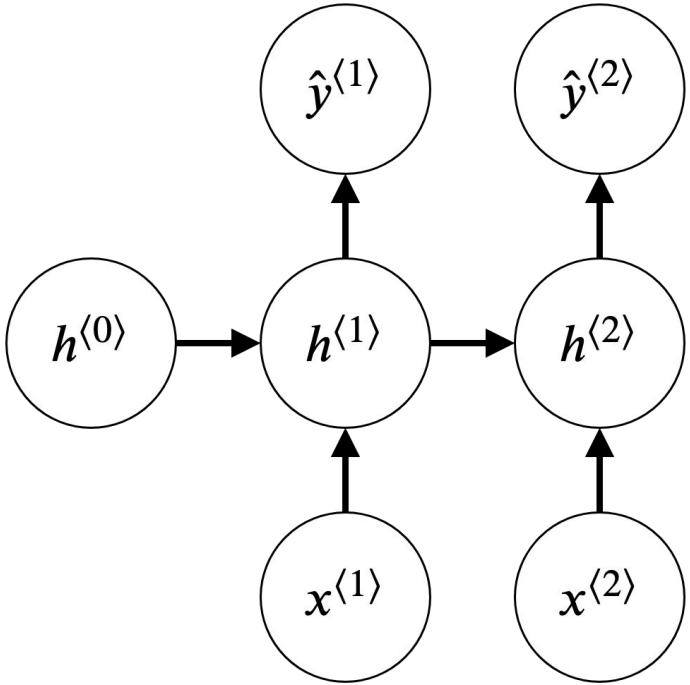


Unrolled Recurrent Connection

$$\mathbf{h}^{\langle 0 \rangle} = \mathbf{0}$$

$$\mathbf{h}^{\langle 1 \rangle} = f(\mathbf{h}^{\langle 0 \rangle}, \mathbf{x}^{\langle 1 \rangle})$$

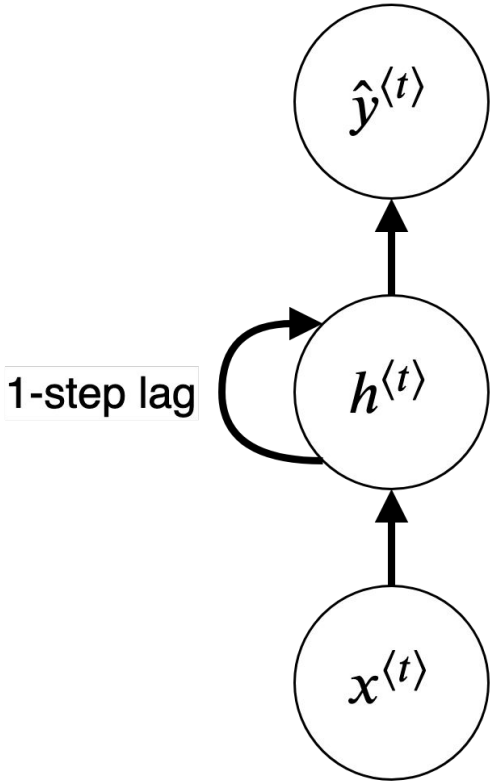
$$\mathbf{h}^{\langle 2 \rangle} = f(\mathbf{h}^{\langle 1 \rangle}, \mathbf{x}^{\langle 2 \rangle})$$



Example: Dot Product / Tanh

$$\mathbf{z}^{\langle t \rangle} = \mathbf{V}\mathbf{h}^{\langle t-1 \rangle} + \mathbf{U}\mathbf{x}^{\langle t \rangle} + \mathbf{b}$$

$$\mathbf{h}^{\langle t \rangle} = \tanh(\mathbf{z}^{\langle t \rangle})$$



Dot Product / Tanh: Unrolled

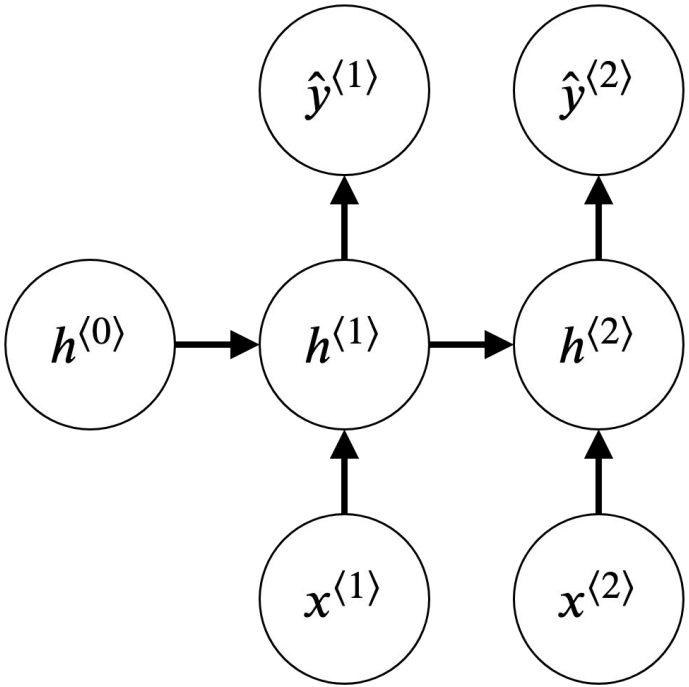
$$\mathbf{h}^{\langle 0 \rangle} = \mathbf{0}$$

$$\mathbf{z}^{\langle 1 \rangle} = \mathbf{V}\mathbf{h}^{\langle 0 \rangle} + \mathbf{U}\mathbf{x}^{\langle 1 \rangle} + \mathbf{b}$$

$$\mathbf{h}^{\langle 1 \rangle} = \tanh(\mathbf{z}^{\langle 1 \rangle})$$

$$\mathbf{z}^{\langle 2 \rangle} = \mathbf{V}\mathbf{h}^{\langle 1 \rangle} + \mathbf{U}\mathbf{x}^{\langle 2 \rangle} + \mathbf{b}$$

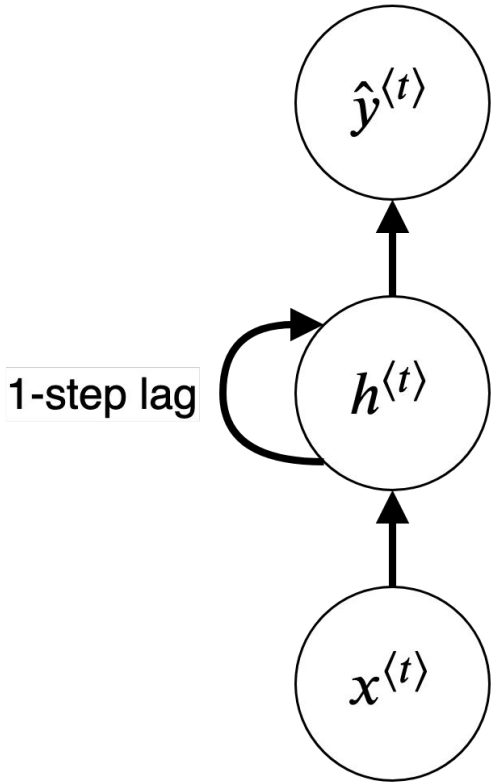
$$\mathbf{h}^{\langle 2 \rangle} = \tanh(\mathbf{z}^{\langle 2 \rangle})$$



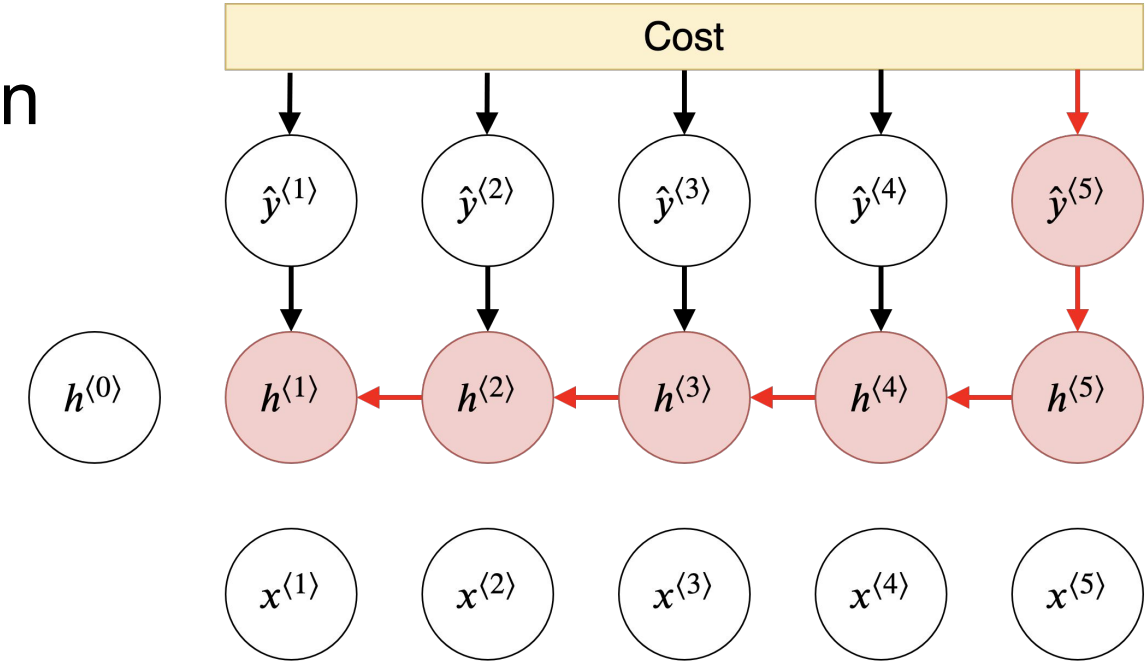
Stacked Parameters Notation

$$\mathbf{W} = \begin{pmatrix} \mathbf{V} & \mathbf{U} \end{pmatrix}$$

$$\mathbf{h}^{\langle t \rangle} = \tanh\left(\mathbf{W} \begin{pmatrix} \mathbf{h}^{\langle t-1 \rangle} \\ \mathbf{x}^{\langle t \rangle} \end{pmatrix} + \mathbf{b} \right)$$



Backpropagation Through Time



$$\frac{\partial C}{\partial \mathbf{W}} = \cdots + \frac{\partial C}{\partial \hat{\mathbf{y}}^{\langle 5 \rangle}} \frac{\partial \hat{\mathbf{y}}^{\langle 5 \rangle}}{\partial \mathbf{h}^{\langle 5 \rangle}} \frac{\partial \mathbf{h}^{\langle 5 \rangle}}{\partial \mathbf{h}^{\langle 4 \rangle}} \frac{\partial \mathbf{h}^{\langle 4 \rangle}}{\partial \mathbf{h}^{\langle 3 \rangle}} \frac{\partial \mathbf{h}^{\langle 3 \rangle}}{\partial \mathbf{h}^{\langle 2 \rangle}} \frac{\partial \mathbf{h}^{\langle 2 \rangle}}{\partial \mathbf{h}^{\langle 1 \rangle}} \frac{\partial \mathbf{h}^{\langle 1 \rangle}}{\partial \mathbf{W}} + \cdots$$

Exploding Gradient

Problem

Repetitive matrix multiplication makes gradient very large.

Solution 1

If the gradient's norm is larger than threshold, element-wise rescale gradient by (threshold / norm).

$$\frac{dC}{dW} = \nabla, n = ||\nabla||_2$$

$$\nabla_i \leftarrow \begin{cases} \nabla_i \cdot \frac{t}{n} & \text{if } n > t \\ \nabla_i & \text{otherwise} \end{cases}$$

Exploding Gradient

Problem

Repetitive matrix multiplication makes gradient very large.

Solution 2

Simply element-wise clip gradient to a reasonable interval.

$$\frac{dC}{dW} = \nabla, m > 0$$

$$\nabla_i \leftarrow \begin{cases} min(\nabla_i, m) & \text{if } \nabla_i > 0 \\ max(\nabla_i, -m) & \text{otherwise} \end{cases}$$

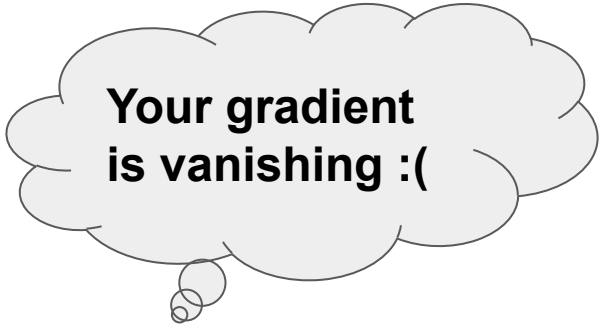
Vanishing Gradient

Problem

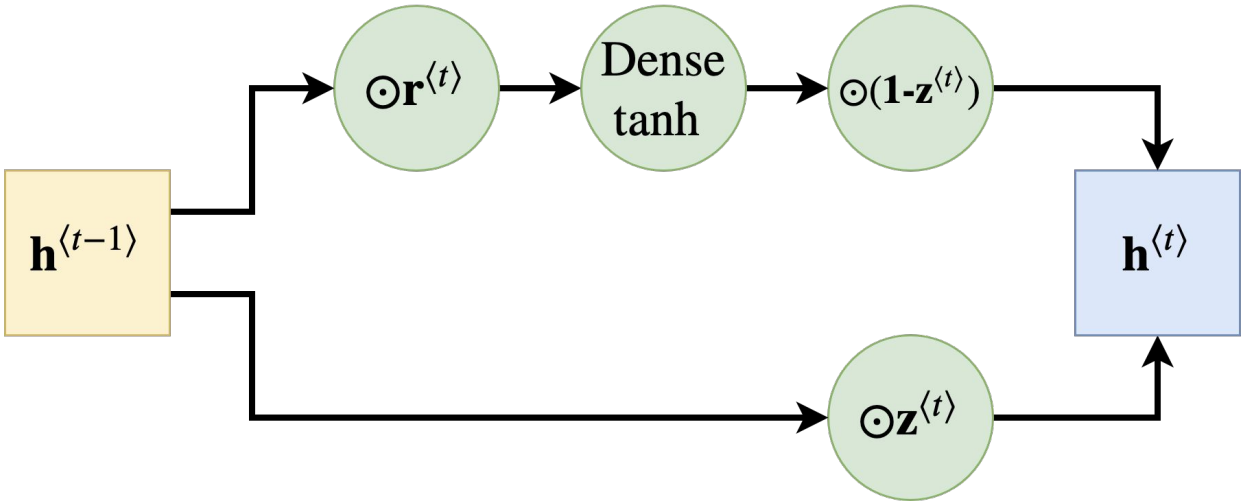
Repetitive matrix multiplication makes gradient very small.

Solution

Need a different cell architecture, where gradient is not repetitively multiplied by the same matrix.

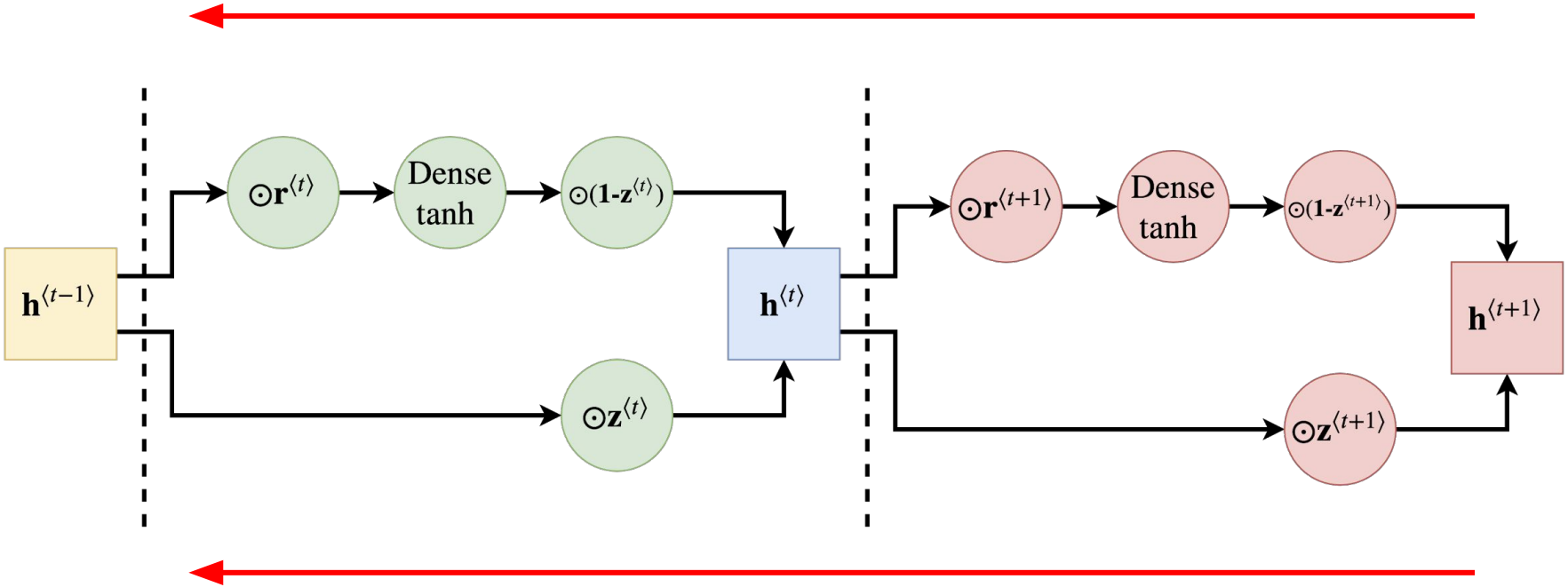


Gated Recurrent Unit



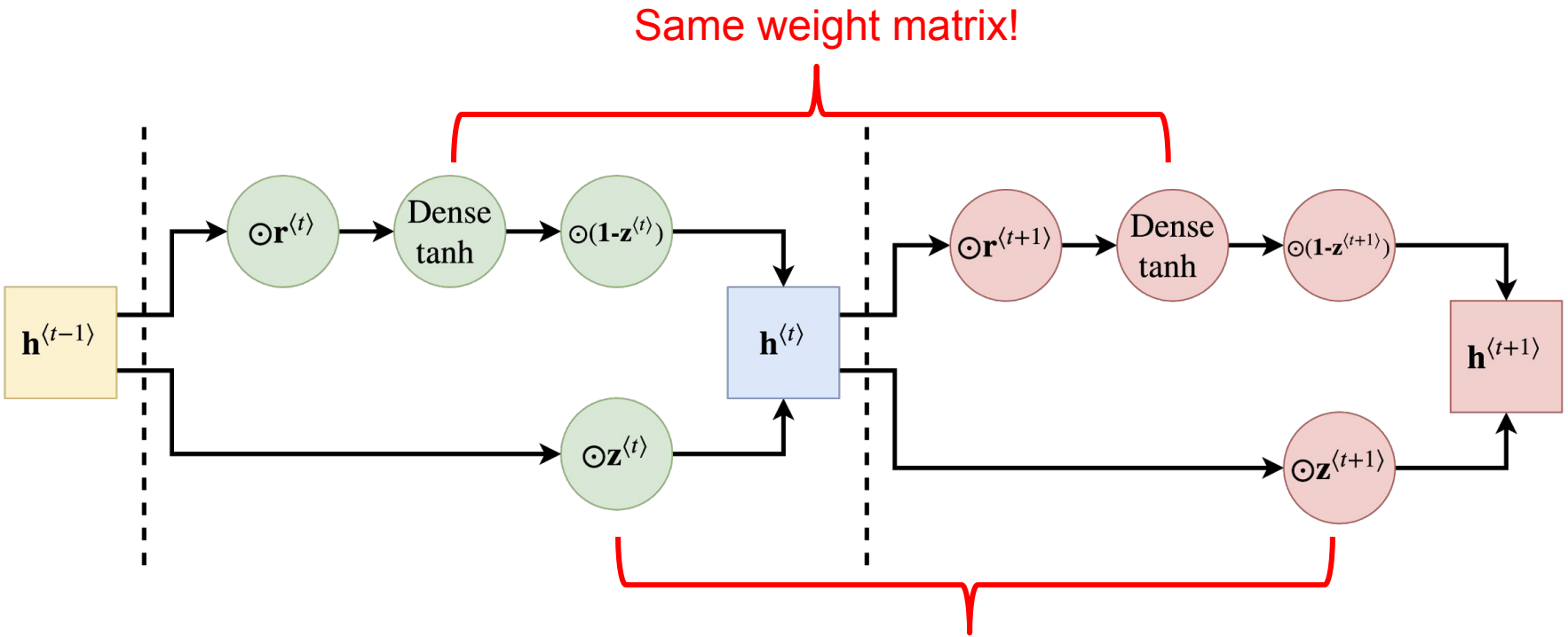
Cho, K., et al. (2014)

GRU: Gradient Flow



Cho, K., et al. (2014)

GRU: Gradient Flow



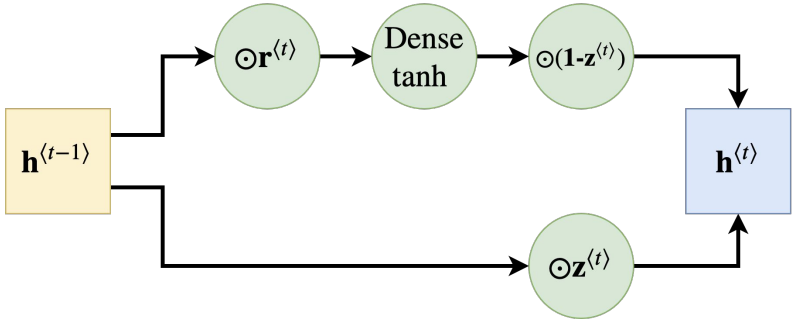
Cho, K., et al. (2014)

GRU: Equations

$$\begin{pmatrix} \mathbf{r}^{\langle t \rangle} \\ \mathbf{z}^{\langle t \rangle} \end{pmatrix} = \text{sigmoid}\left[\mathbf{U}\begin{pmatrix} \mathbf{h}^{\langle t-1 \rangle} \\ \mathbf{x}^{\langle t \rangle} \end{pmatrix} + \mathbf{b}\right]$$

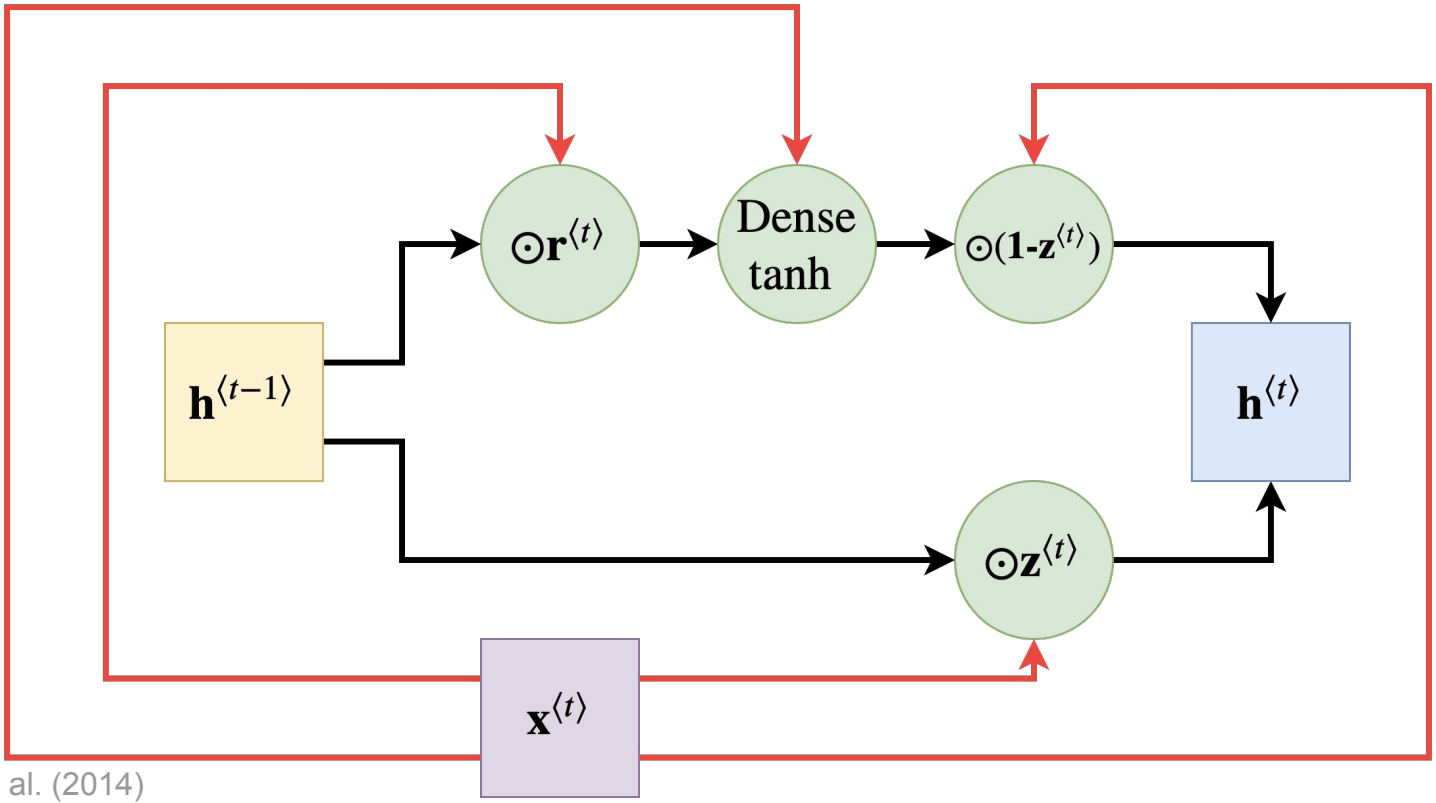
$$\tilde{\mathbf{h}}^{\langle t \rangle} = \tanh\left[\mathbf{V}\begin{pmatrix} \mathbf{r}^{\langle t \rangle} \odot \mathbf{h}^{\langle t-1 \rangle} \\ \mathbf{x}^{\langle t \rangle} \end{pmatrix} + \mathbf{c}\right]$$

$$\mathbf{h}^{\langle t \rangle} = \left[\mathbf{z}^{\langle t \rangle} \odot \mathbf{h}^{\langle t-1 \rangle}\right] + \left[(1 - \mathbf{z}^{\langle t \rangle}) \odot \tilde{\mathbf{h}}^{\langle t \rangle}\right]$$



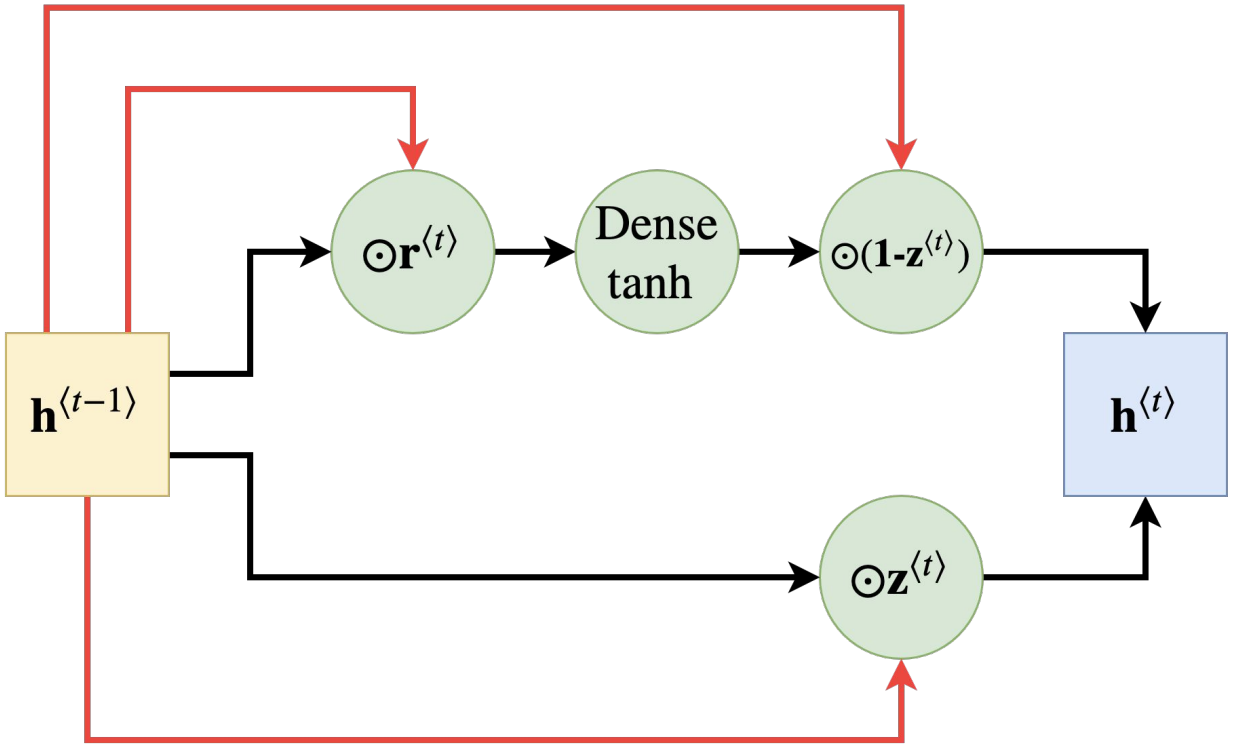
Cho, K., et al. (2014)

GRU: Input Consumption



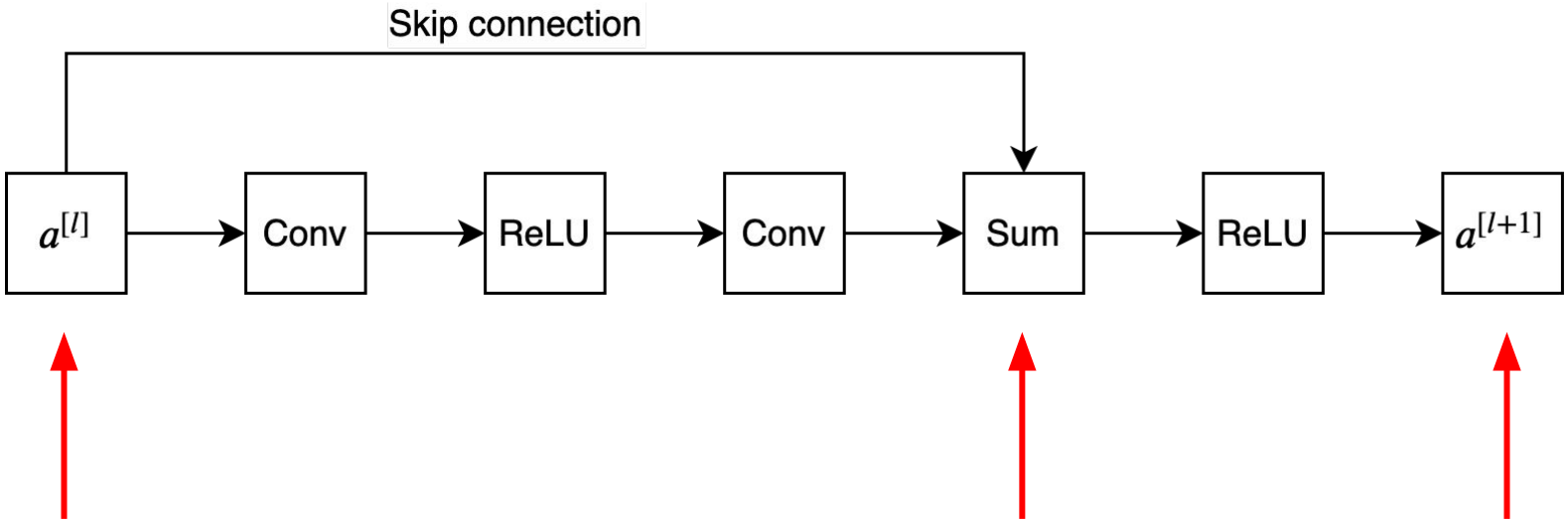
Cho, K., et al. (2014)

GRU: Recurrent Connection



Cho, K., et al. (2014)

Analogy with Residual Layer

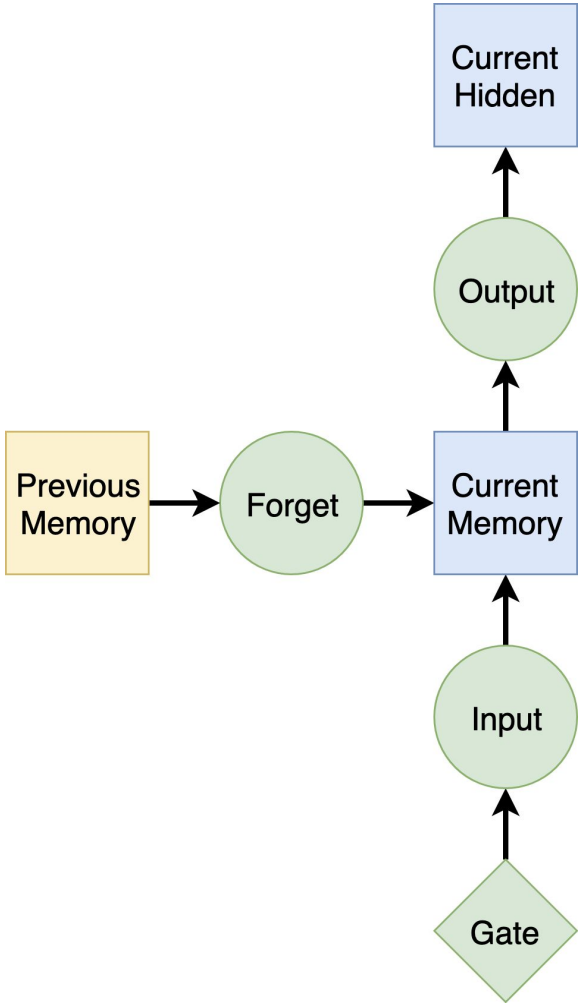


Long Short-Term Memory

$$\begin{pmatrix} \mathbf{i}^{\langle t \rangle} \\ \mathbf{f}^{\langle t \rangle} \\ \mathbf{o}^{\langle t \rangle} \\ \mathbf{g}^{\langle t \rangle} \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} \mathbf{W} \begin{pmatrix} \mathbf{h}^{\langle t-1 \rangle} \\ \mathbf{x}^{\langle t \rangle} \end{pmatrix} + \mathbf{b}$$

$$\mathbf{c}^{\langle t \rangle} = \mathbf{f} \odot \mathbf{c}^{\langle t-1 \rangle} + \mathbf{i}^{\langle t \rangle} \odot \mathbf{g}^{\langle t \rangle}$$

$$\mathbf{h}^{\langle t \rangle} = \mathbf{o} \odot \tanh(\mathbf{c}^{\langle t \rangle})$$



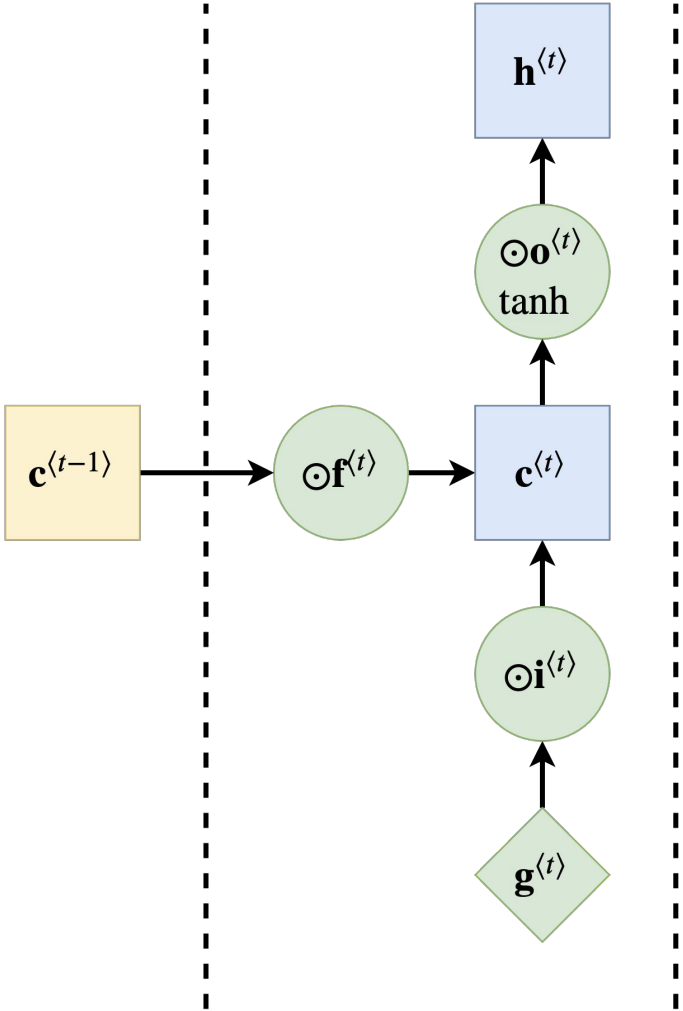
Hochreiter, S, and Schmidhuber, J. (1997)

Long Short-Term Memory

$$\begin{pmatrix} \mathbf{i}^{\langle t \rangle} \\ \mathbf{f}^{\langle t \rangle} \\ \mathbf{o}^{\langle t \rangle} \\ \mathbf{g}^{\langle t \rangle} \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} \mathbf{W} \begin{pmatrix} \mathbf{h}^{\langle t-1 \rangle} \\ \mathbf{x}^{\langle t \rangle} \end{pmatrix} + \mathbf{b}$$

$$\mathbf{c}^{\langle t \rangle} = \mathbf{f} \odot \mathbf{c}^{\langle t-1 \rangle} + \mathbf{i}^{\langle t \rangle} \odot \mathbf{g}^{\langle t \rangle}$$

$$\mathbf{h}^{\langle t \rangle} = \mathbf{o} \odot \tanh(\mathbf{c}^{\langle t \rangle})$$



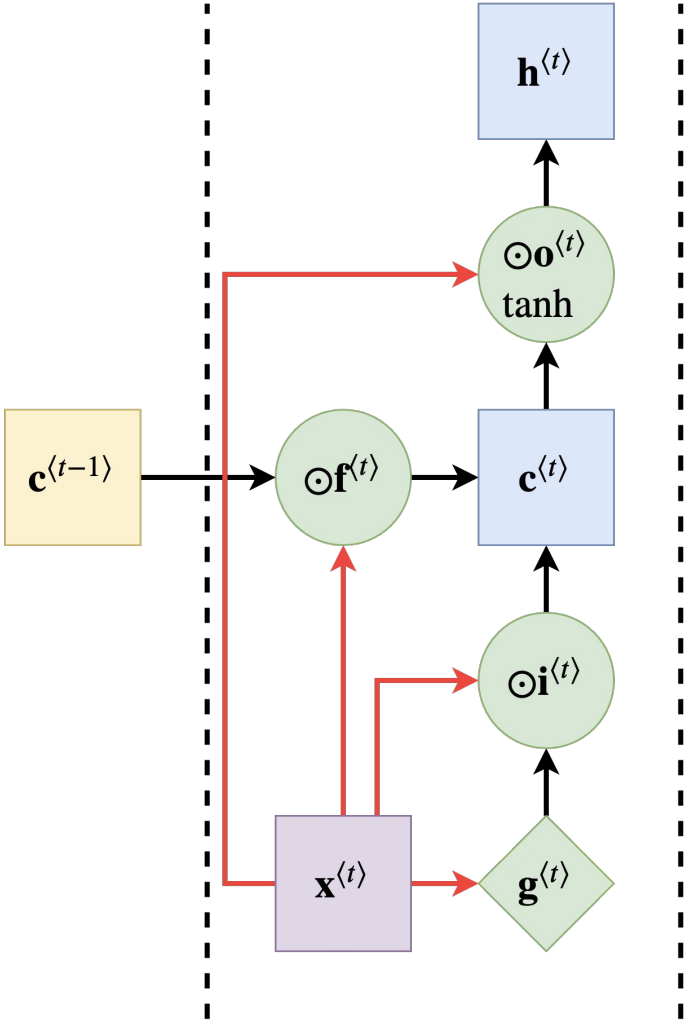
Hochreiter, S, and Schmidhuber, J. (1997)

Long Short-Term Memory

$$\begin{pmatrix} \mathbf{i}^{\langle t \rangle} \\ \mathbf{f}^{\langle t \rangle} \\ \mathbf{o}^{\langle t \rangle} \\ \mathbf{g}^{\langle t \rangle} \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} \mathbf{W} \begin{pmatrix} \mathbf{h}^{\langle t-1 \rangle} \\ \mathbf{x}^{\langle t \rangle} \end{pmatrix} + \mathbf{b}$$

$$\mathbf{c}^{\langle t \rangle} = \mathbf{f} \odot \mathbf{c}^{\langle t-1 \rangle} + \mathbf{i}^{\langle t \rangle} \odot \mathbf{g}^{\langle t \rangle}$$

$$\mathbf{h}^{\langle t \rangle} = \mathbf{o} \odot \tanh(\mathbf{c}^{\langle t \rangle})$$



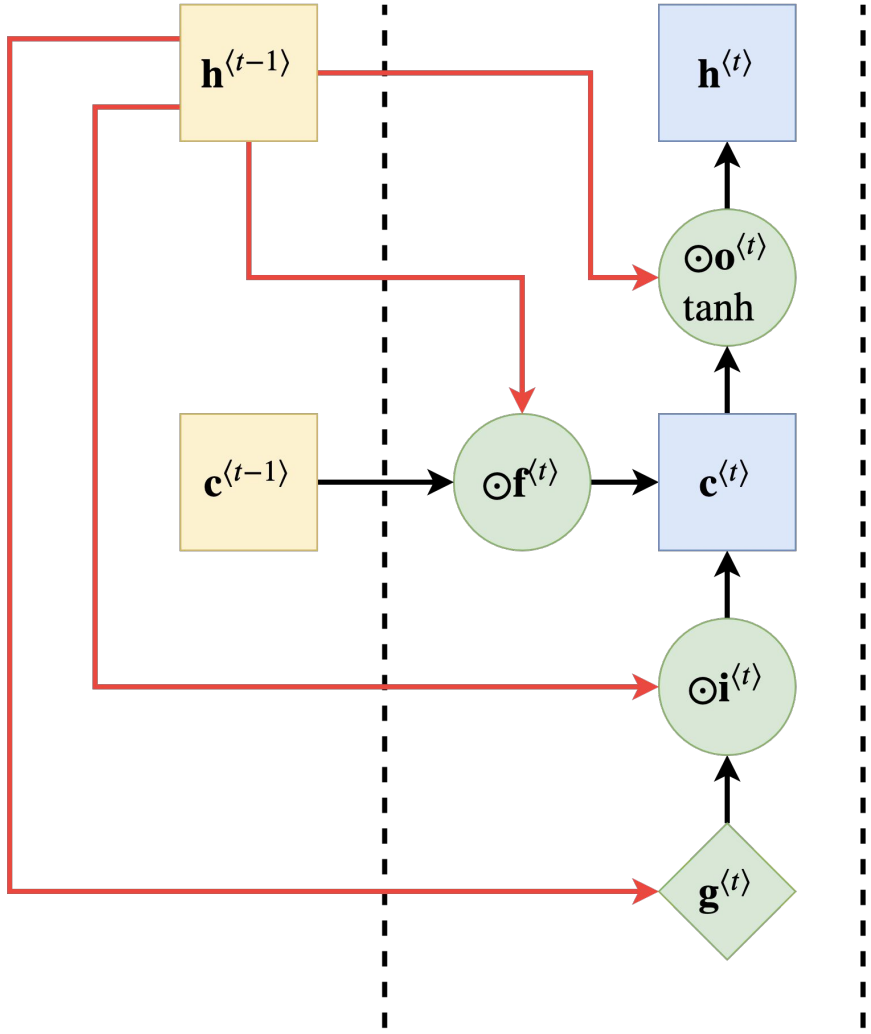
Hochreiter, S, and Schmidhuber, J. (1997)

Long Short-Term Memory

$$\begin{pmatrix} \mathbf{i}^{\langle t \rangle} \\ \mathbf{f}^{\langle t \rangle} \\ \mathbf{o}^{\langle t \rangle} \\ \mathbf{g}^{\langle t \rangle} \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} \mathbf{W} \begin{pmatrix} \mathbf{h}^{\langle t-1 \rangle} \\ \mathbf{x}^{\langle t \rangle} \end{pmatrix} + \mathbf{b}$$

$$\mathbf{c}^{\langle t \rangle} = \mathbf{f} \odot \mathbf{c}^{\langle t-1 \rangle} + \mathbf{i}^{\langle t \rangle} \odot \mathbf{g}^{\langle t \rangle}$$

$$\mathbf{h}^{\langle t \rangle} = \mathbf{o} \odot \tanh(\mathbf{c}^{\langle t \rangle})$$

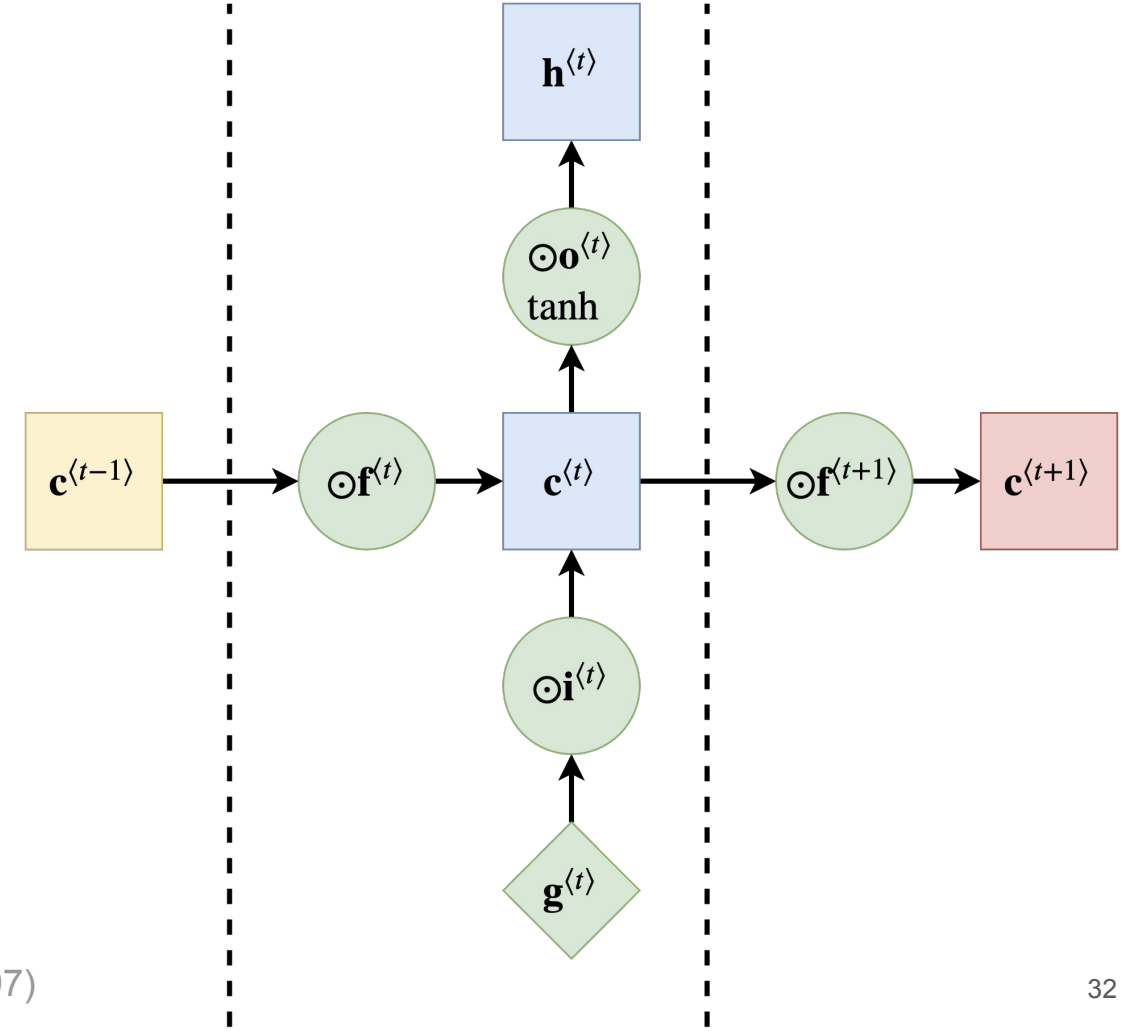


Hochreiter, S, and Schmidhuber, J. (1997)

Gradients flows further in LSTM than in the vanilla architecture due to:

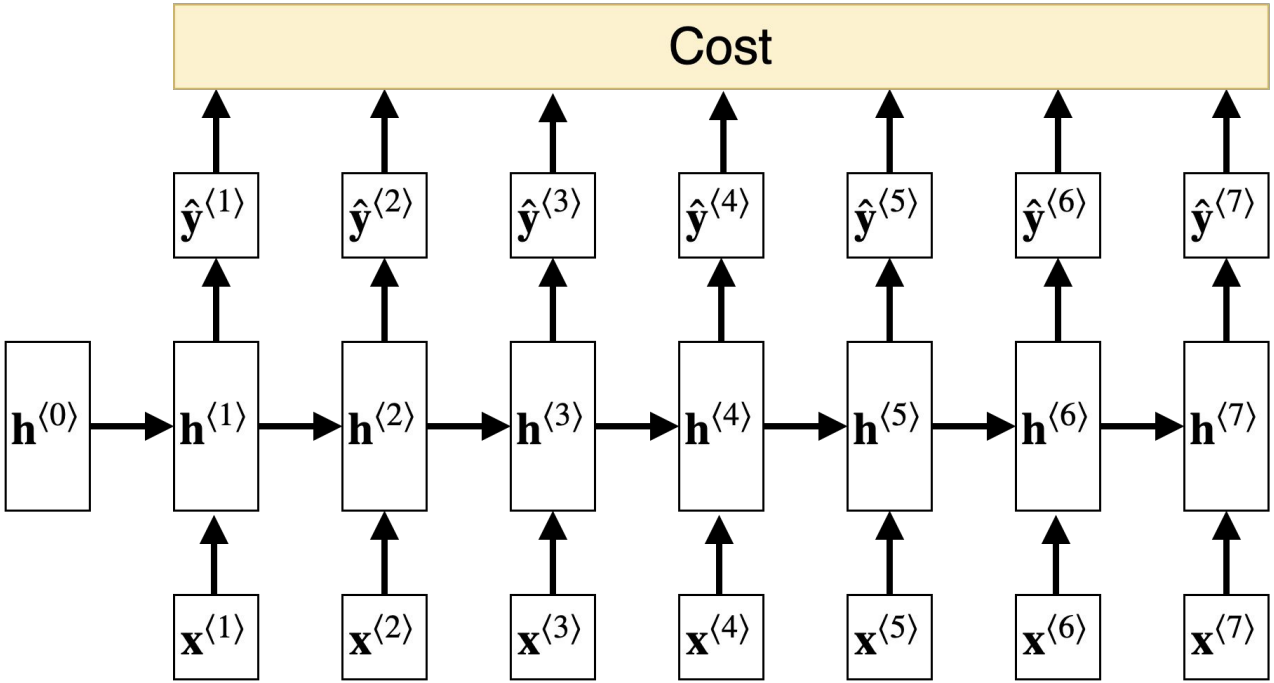
1) Between time steps, memory connected through an element-wise product rather than a dot product.

2) At each step, forget mask varies, while the weight matrix does not.

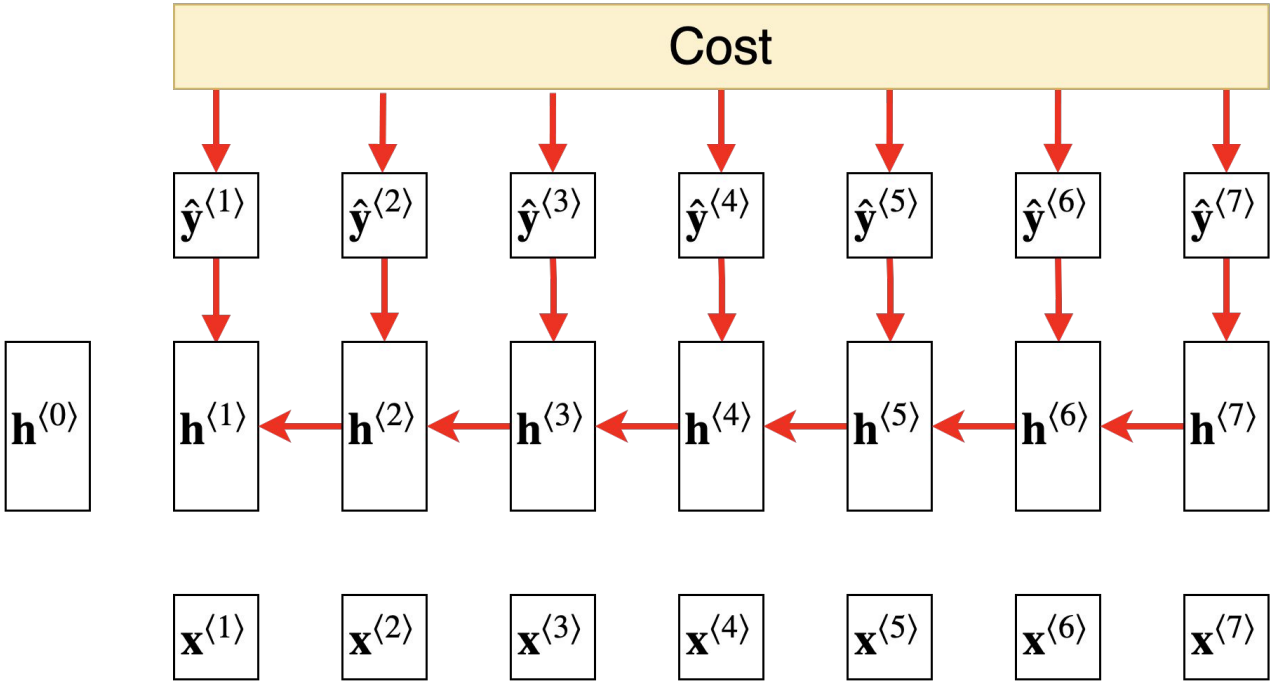


Hochreiter, S, and Schmidhuber, J. (1997)

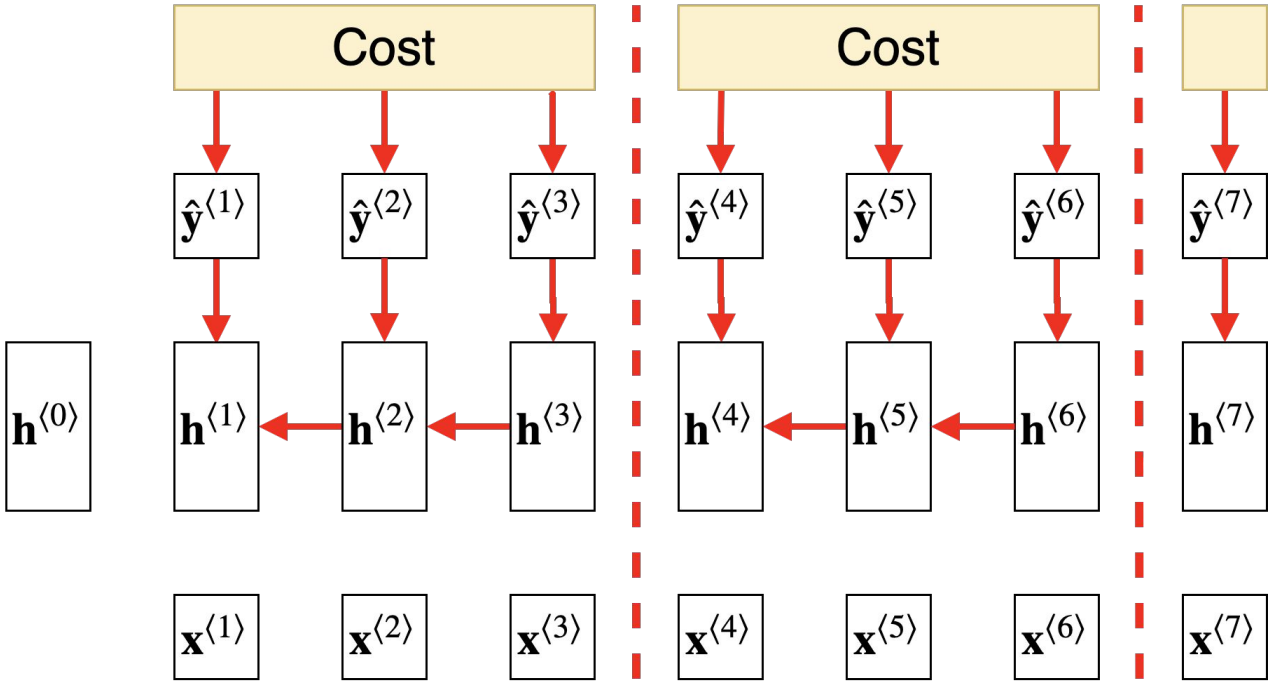
Long Sequences: Forward Pass



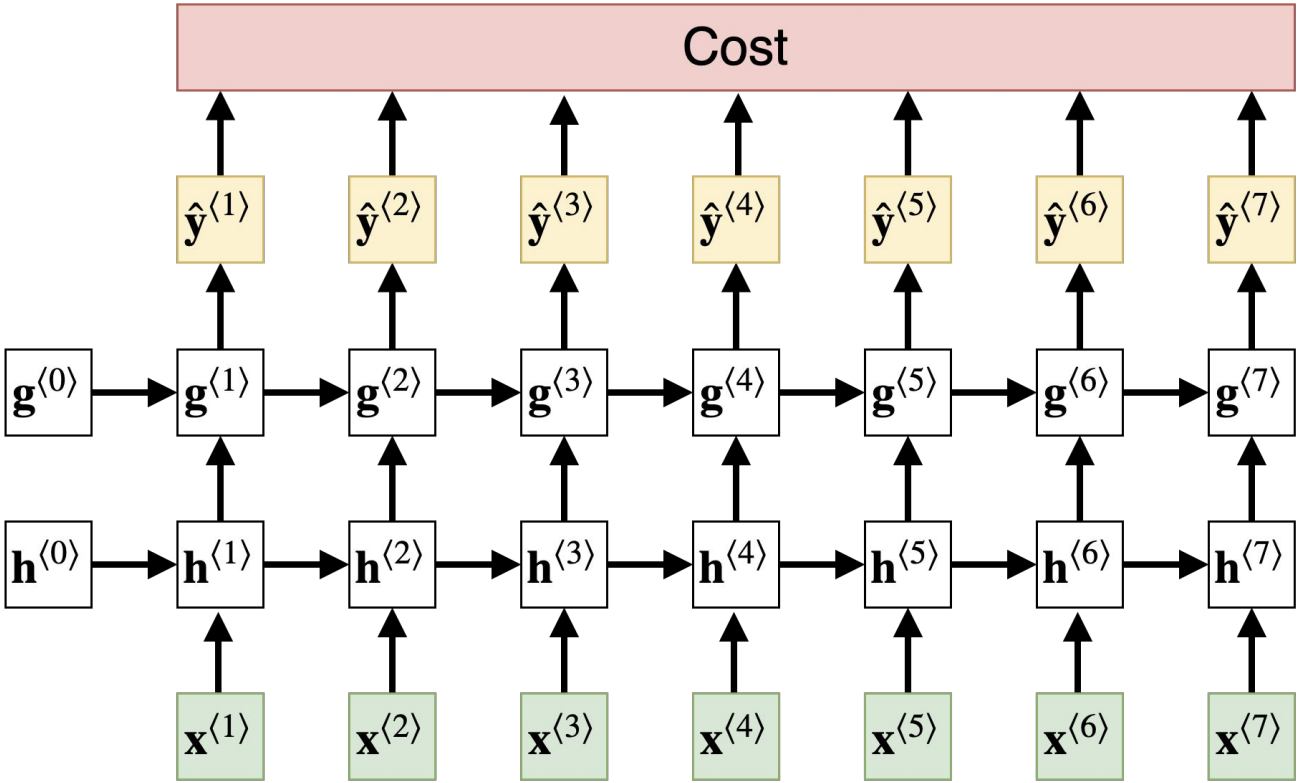
Long Sequences: Backward Pass



Long Sequences: Truncated Backprop



Stacking Recurrent Layers

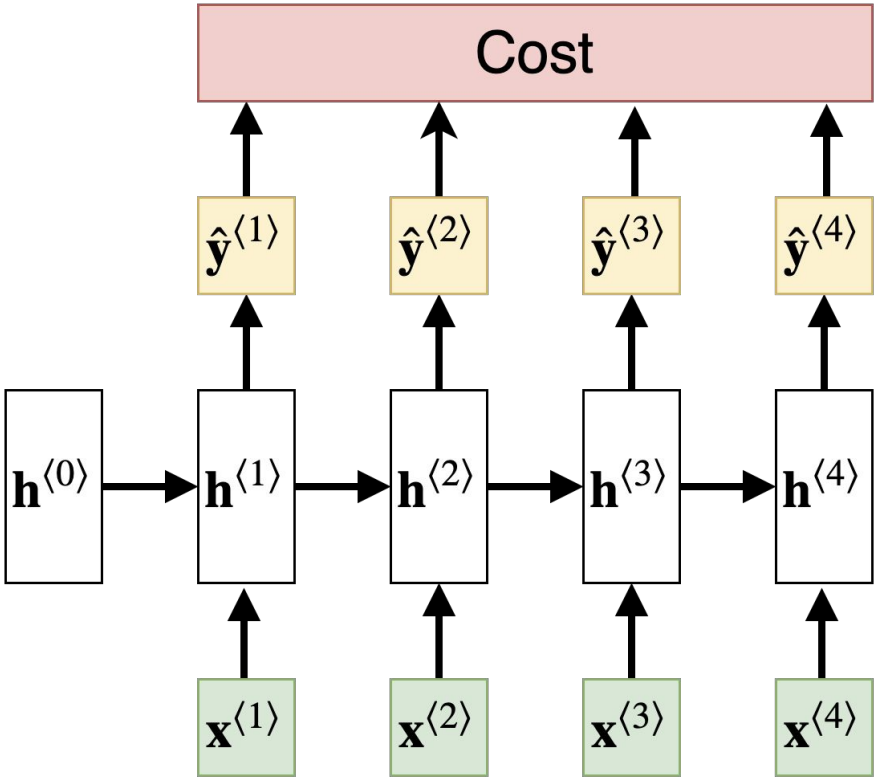


Sequence to Sequence: Sync / Training

Requires to produce output at each time step, without seeing the following inputs.

Examples:

- Speech recognition
- Single-horizon forecasting



Case Study: DeepAR

Task

Time-series forecasting

Key Attributes

- Autoregressive model
- LSTM encoder/decoder
- Probabilistic output
- Multiple time series

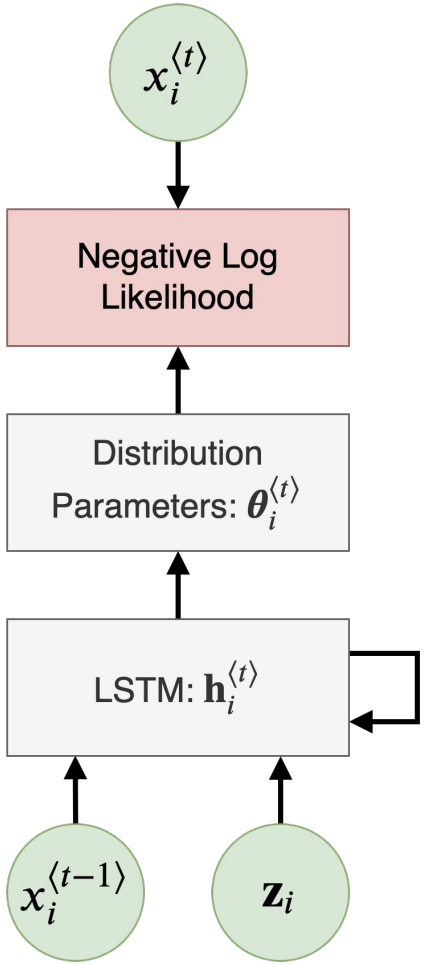
Available via AWS SageMaker and AWS Forecast.

Salinas, D., et al. (2019)

DeepAR: Model

In training, the model learns to output distribution parameters that maximize the likelihood of the next item in the sequence's prediction window.

$$C = \sum_{i=1}^{N} \sum_{t=t_0}^{T_i} -\log \ell\left(x_i^{\langle t \rangle} \mid \theta(\mathbf{h}_i^{\langle t \rangle})\right)$$



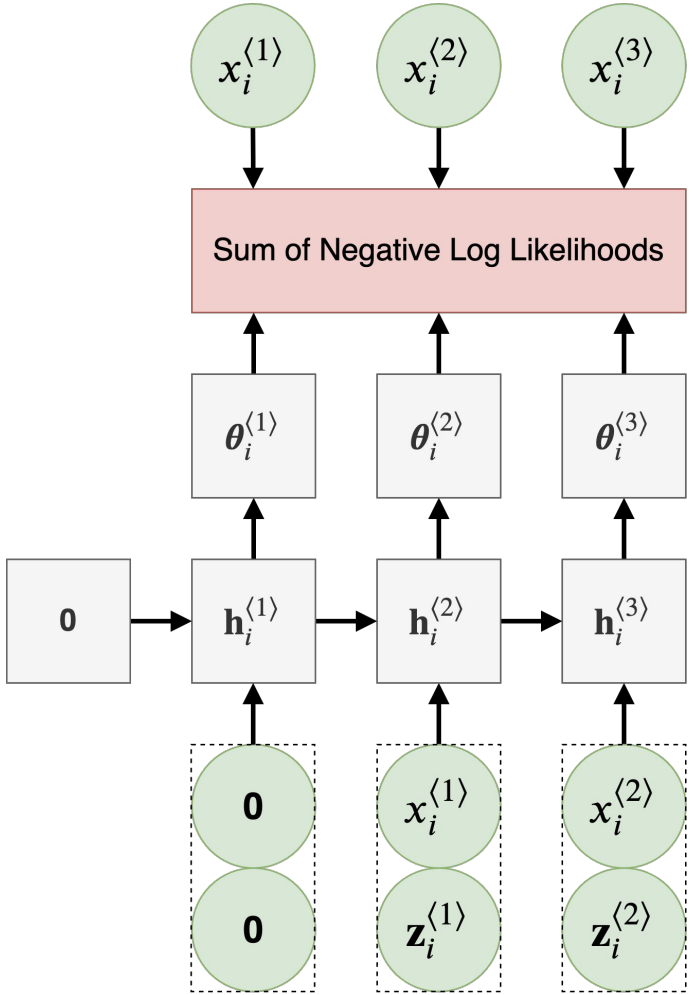
Salinas, D., et al. (2019)

DeepAR: Training

For each of N time series, multiple start times are randomly chosen (cutting sequences).

Some start times are negative, simulating the cold start problem during the inference.

All N time series are trained simultaneously across mini-batches.

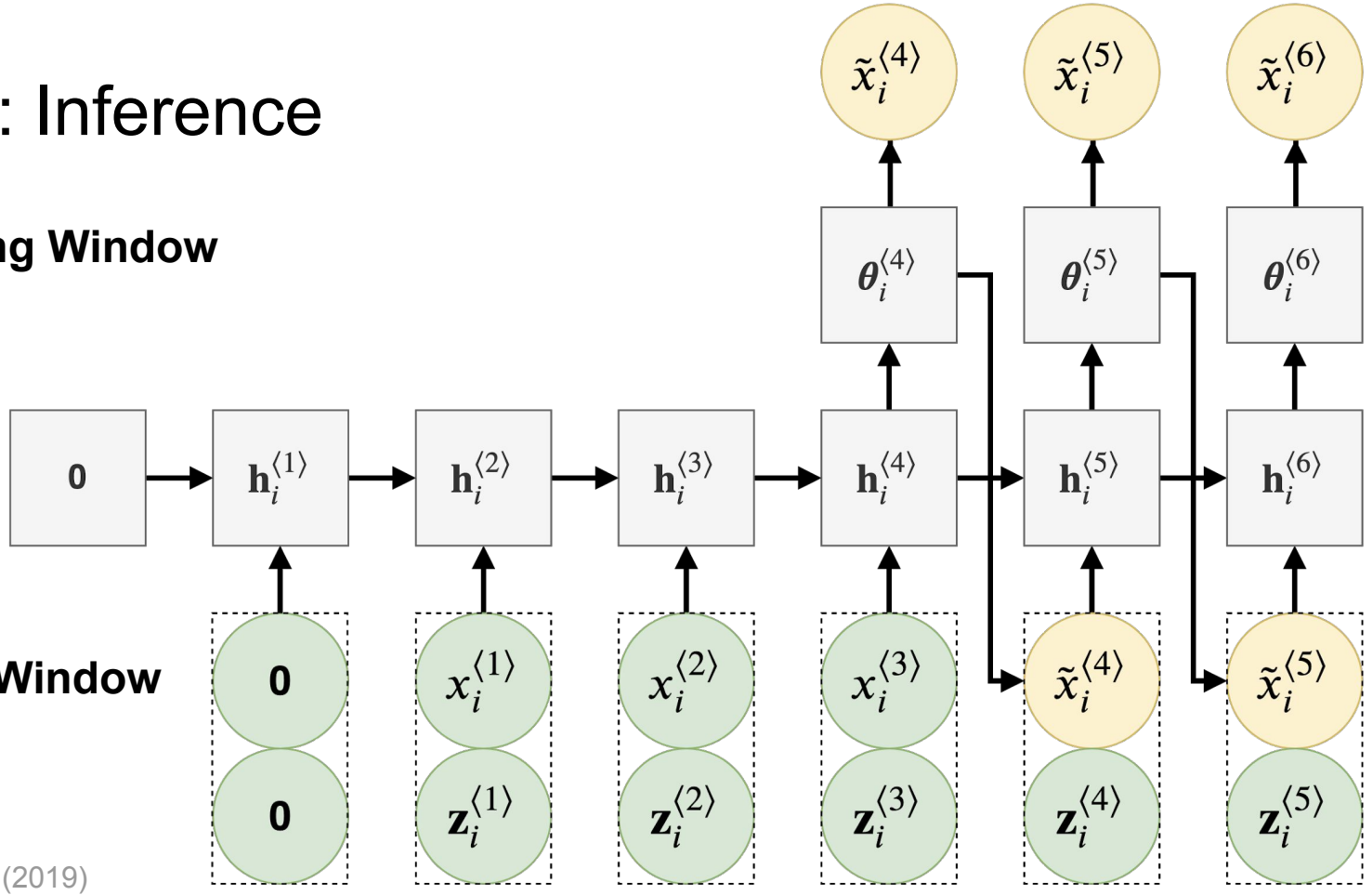


Salinas, D., et al. (2019)

DeepAR: Inference

Conditioning Window
3 periods

Prediction Window
3 periods



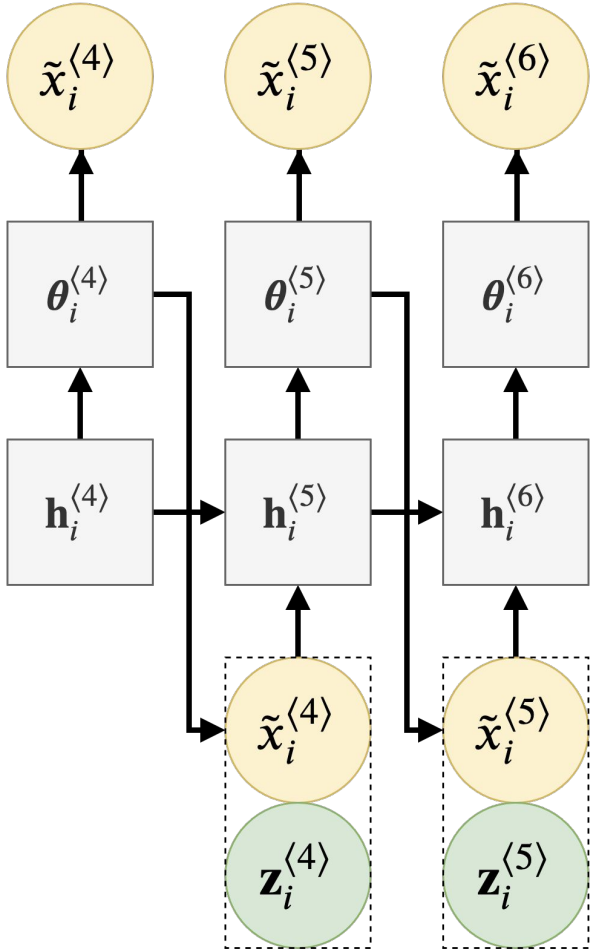
Salinas, D., et al. (2019)

DeepAR: Ancestral Sampling

Ancestral Sampling for prediction window N:

1. Randomly sample the next unknown item
2. Feed into the network as input
3. Repeat for N time steps

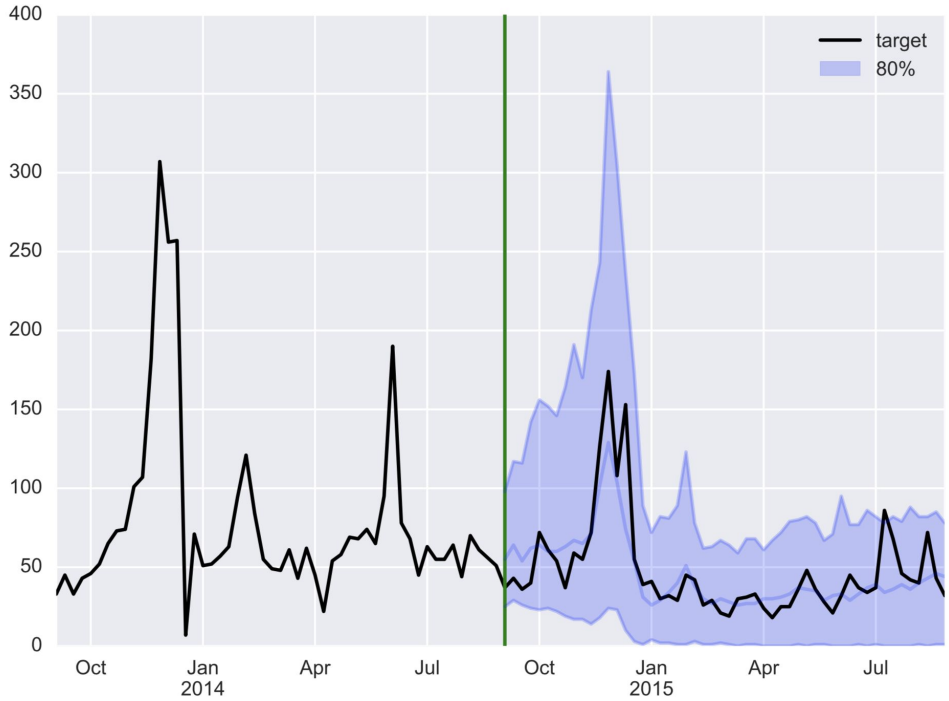
To estimate confidence intervals, sample many prediction sequences.



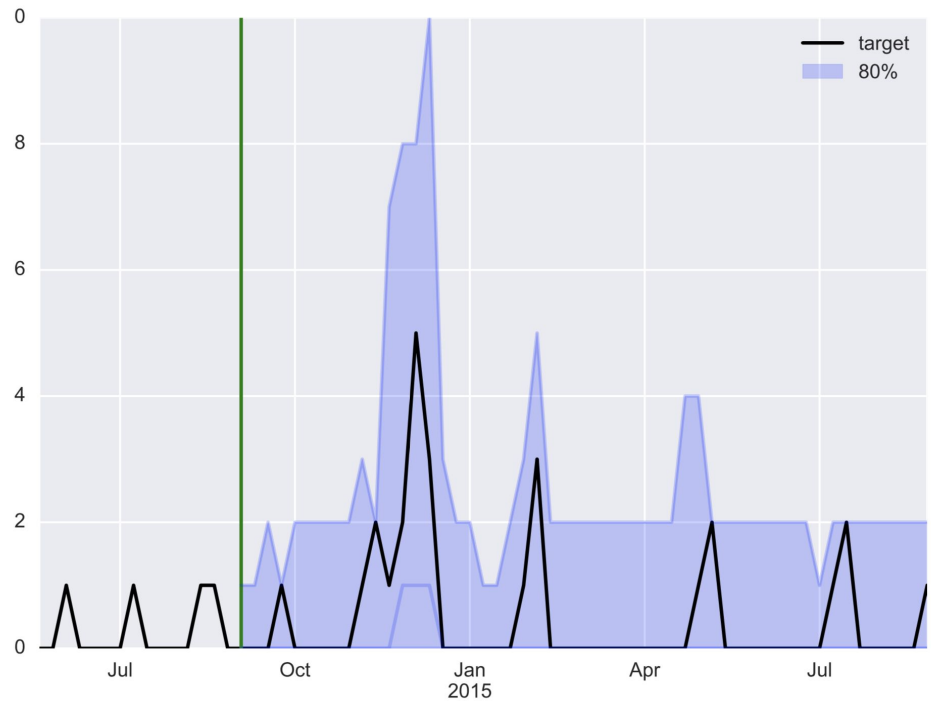
Salinas, D., et al. (2019)

DeepAR: Performance

Salinas, D., et al. (2019)

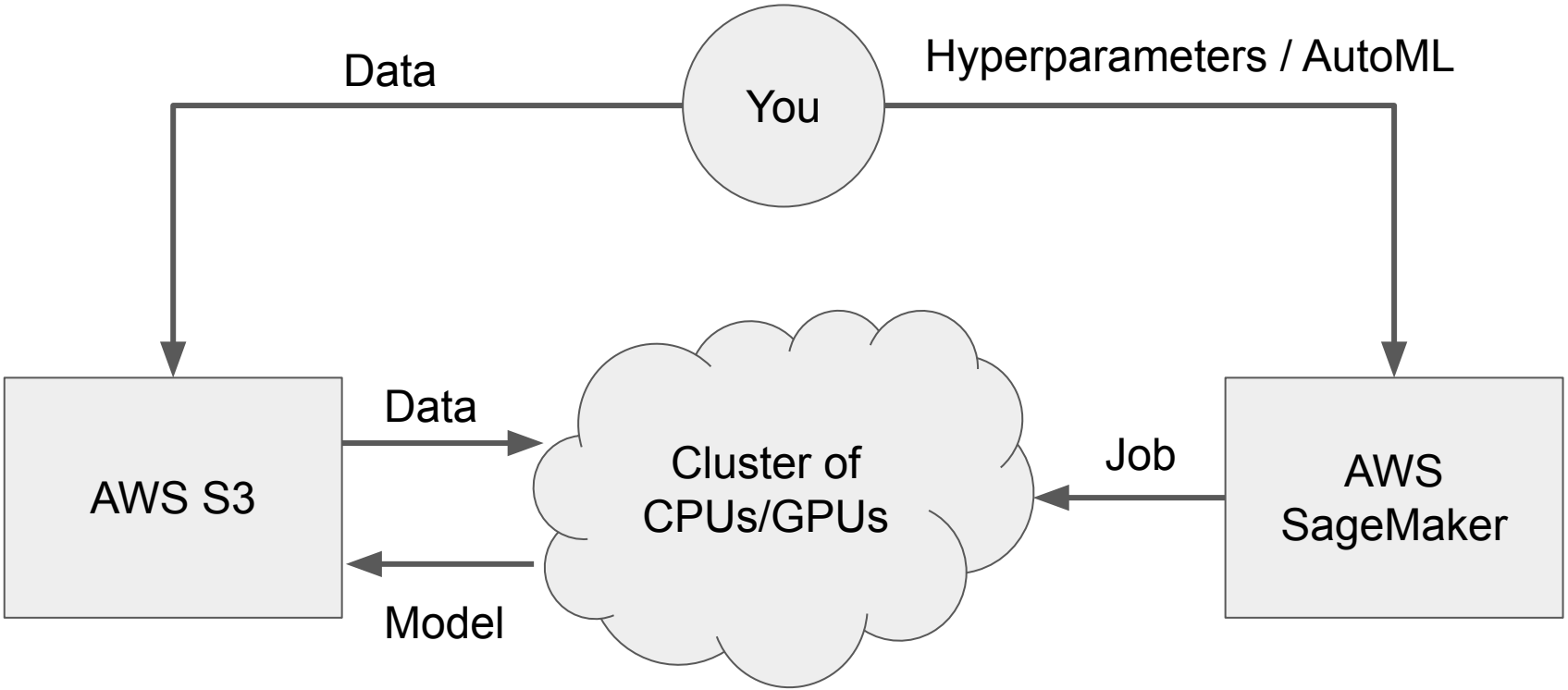


High-demand / warm-start

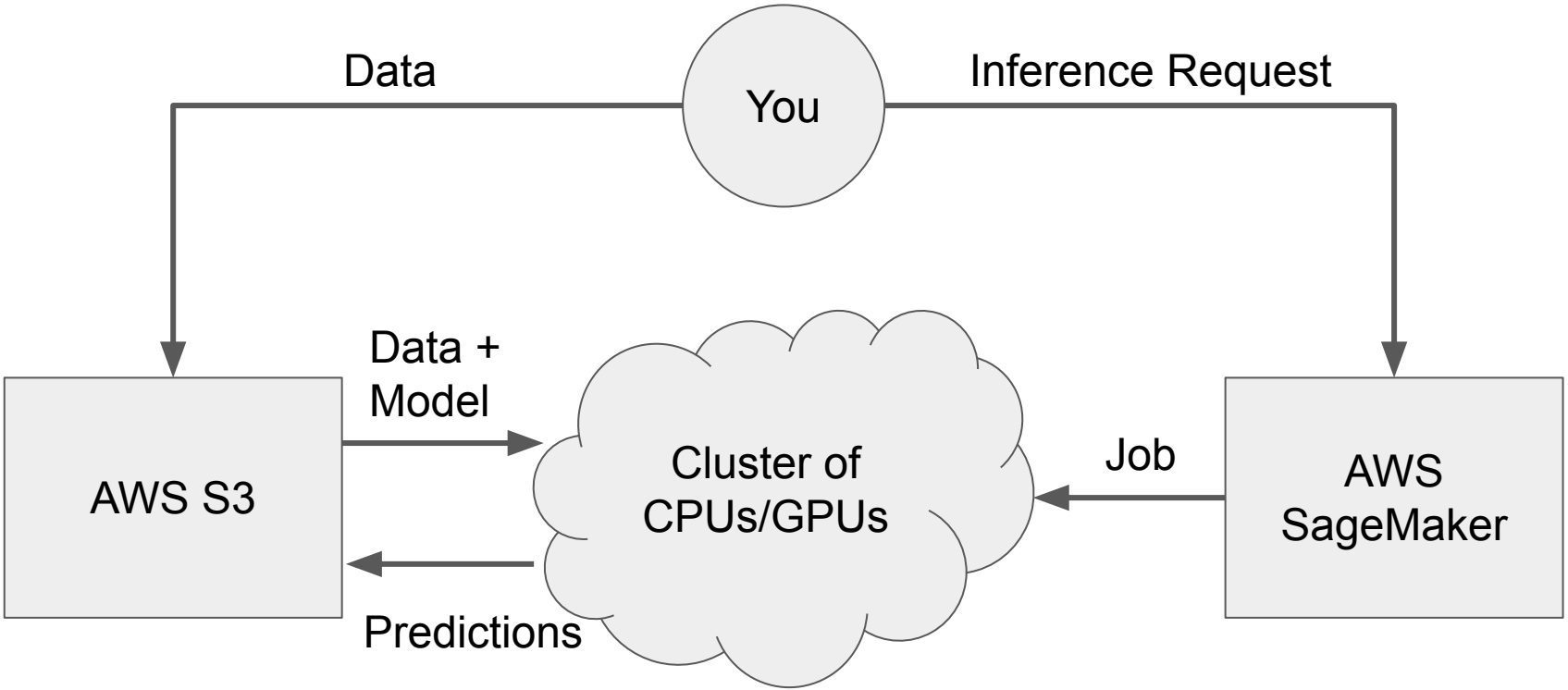


Low-demand / cold-start

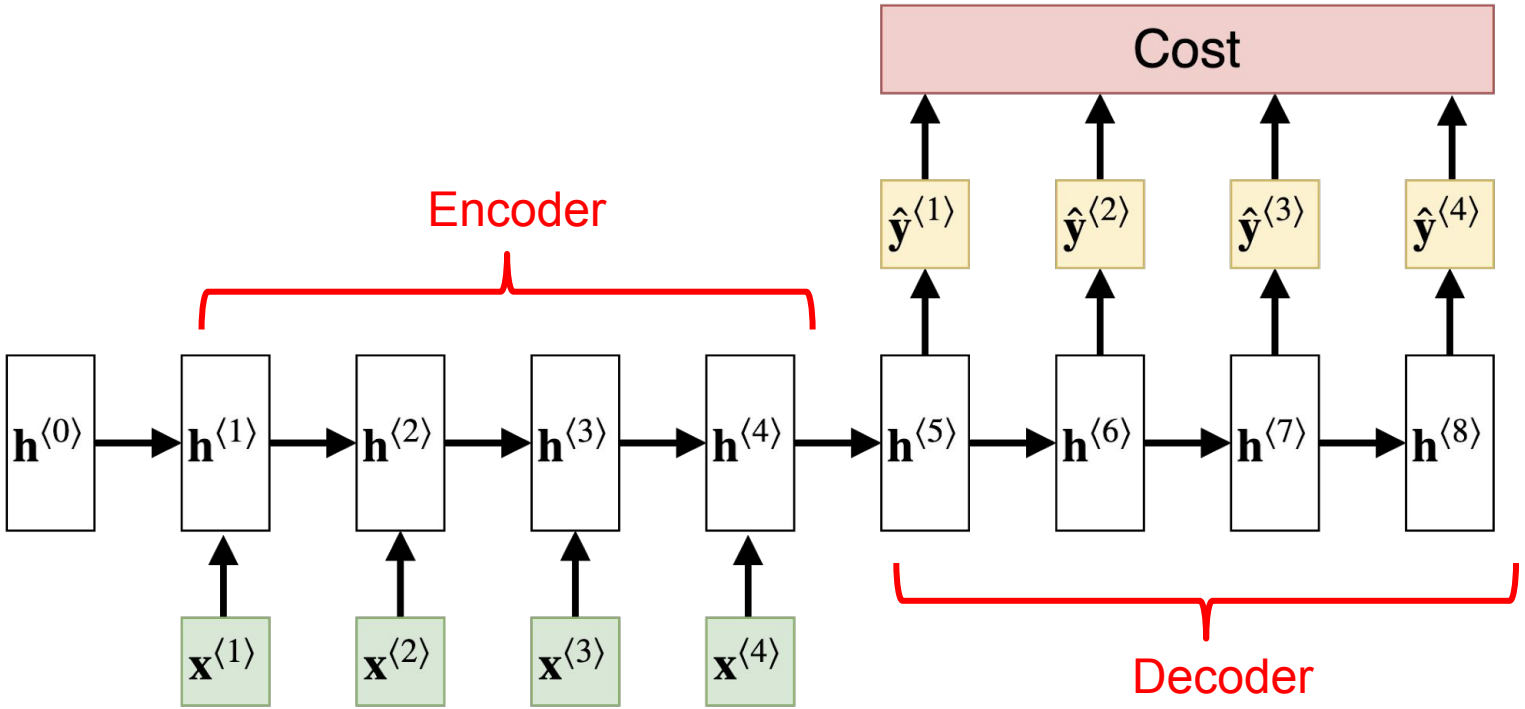
DeepAR: Training in AWS



DeepAR: Batch Inference in AWS



Sequence to Sequence: Async



Case Study: Weather Forecasts

Dataset

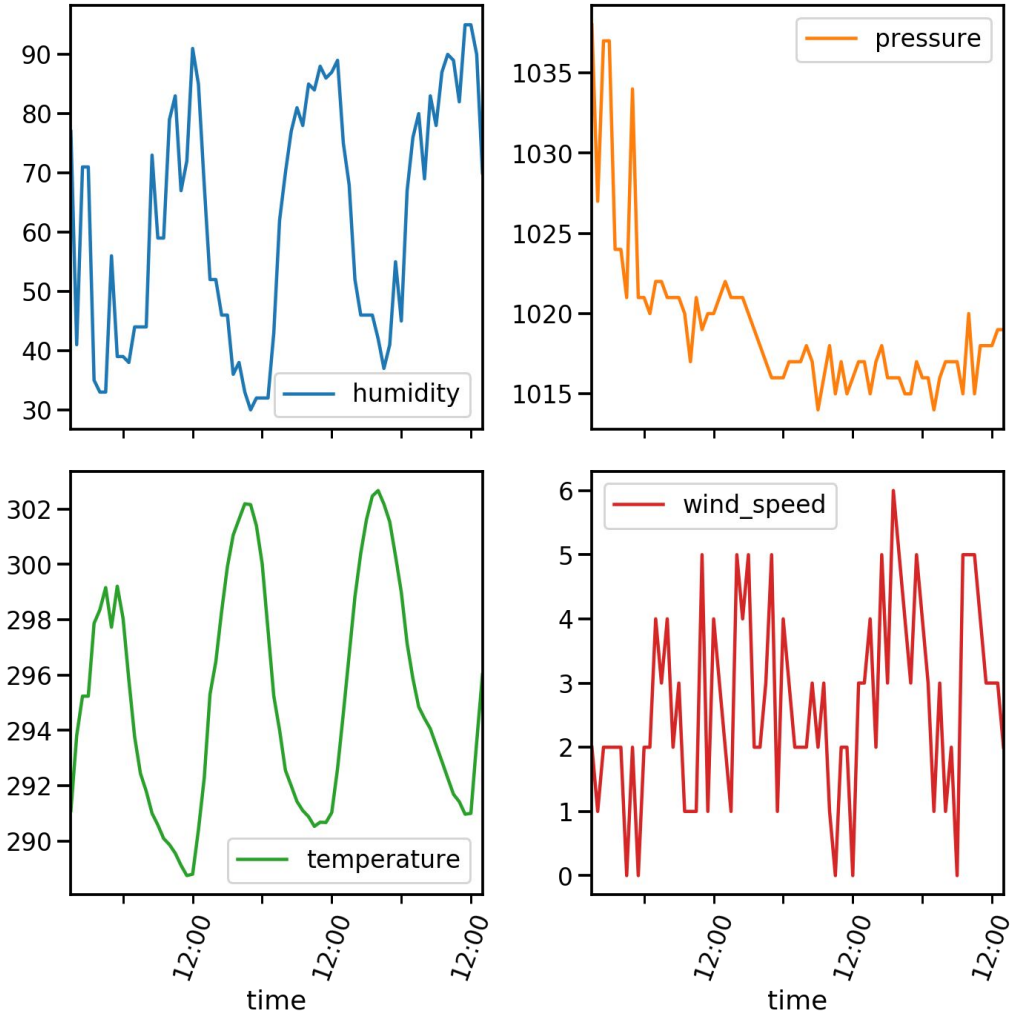
Hourly frequency

36 cities across the world

5 years of non-continuous recordings

Features

Temperature, humidity, atmospheric pressure, wind speed



Weather: Data Quality

1. Marked no-change observation as N/A
2. Forward-padded N/A observations up to 3 hours
3. Removed remaining N/A observations

	feature	humidity	pressure	temperature	wind_speed
sample	time				
Albuquerque/1108/0	2015-02-16 09:00:00	47.0	1031.0	276.323000	3.0
	2015-02-16 10:00:00	45.0	1032.0	275.097667	3.0
	2015-02-16 11:00:00	48.0	1032.0	273.931333	2.0
	2015-02-16 12:00:00	46.0	1033.0	274.380000	2.0
	2015-02-16 13:00:00	51.0	1033.0	272.675333	1.0

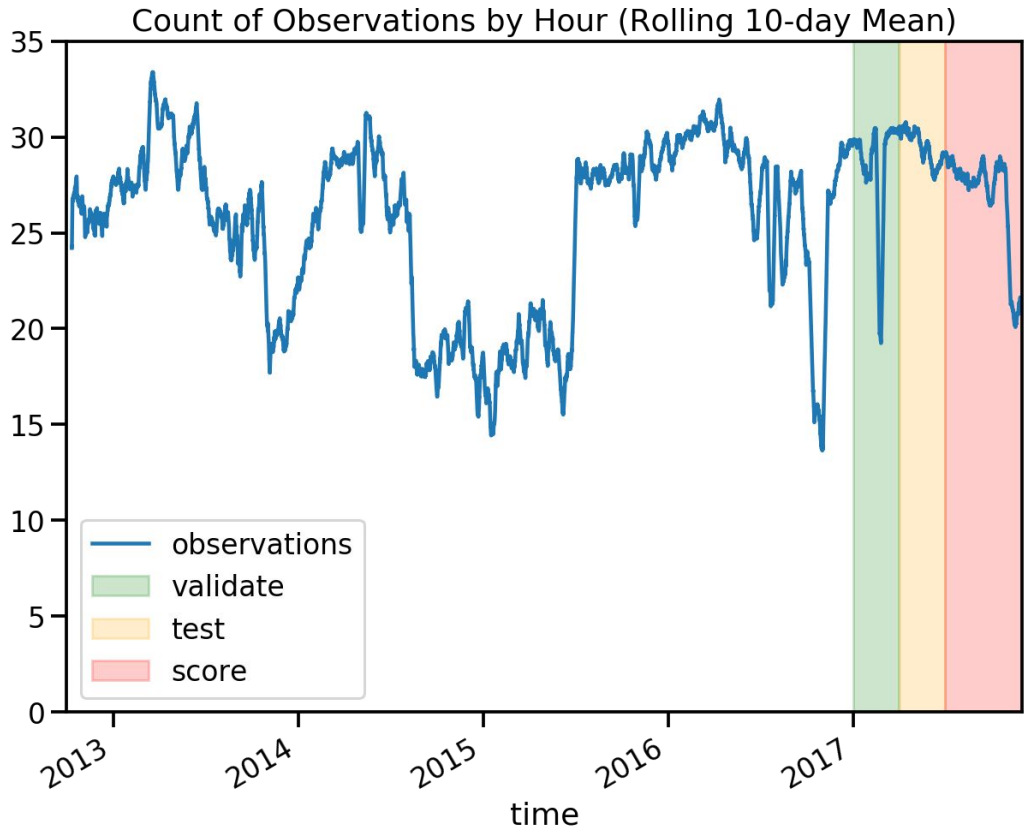
Weather: Data Splits

Simple time-based split

Approximate ratios

80% / 5% / 5% / 10%

	observations	%	date_min	date_max
train	939344	82	2012-10-01	2016-12-31
validate	60785	5	2017-01-01	2017-03-31
test	63502	5	2017-04-01	2017-06-30
score	96635	8	2017-07-01	2017-11-30

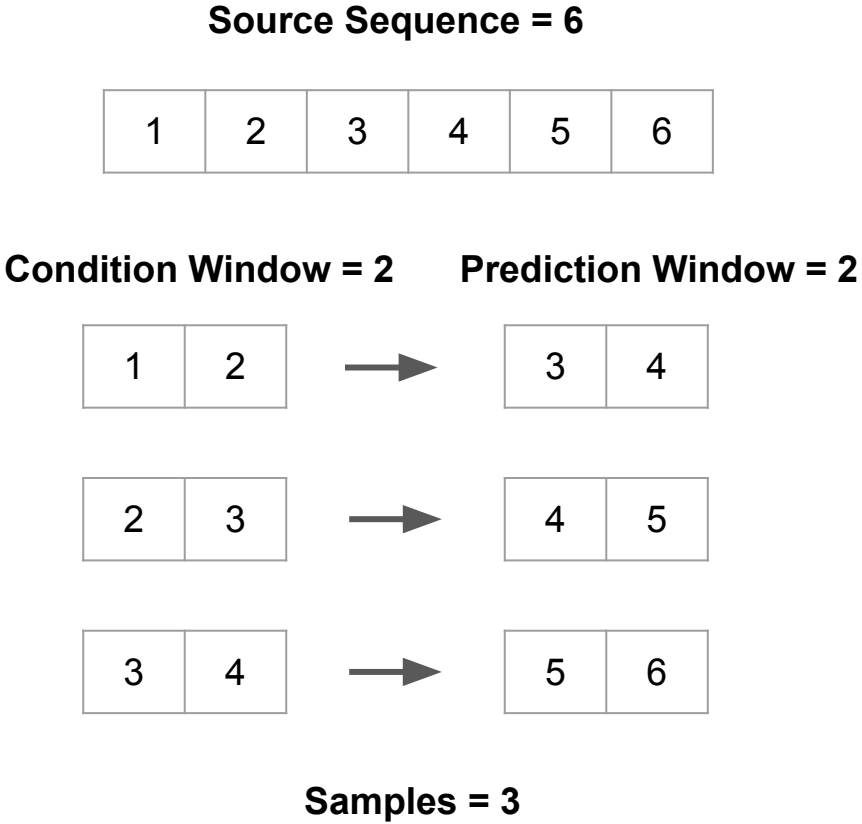


Weather: Batches

Rolling Window

Condition Features
Temperature, pressure, humidity, wind speed

Prediction Variables
Temperature



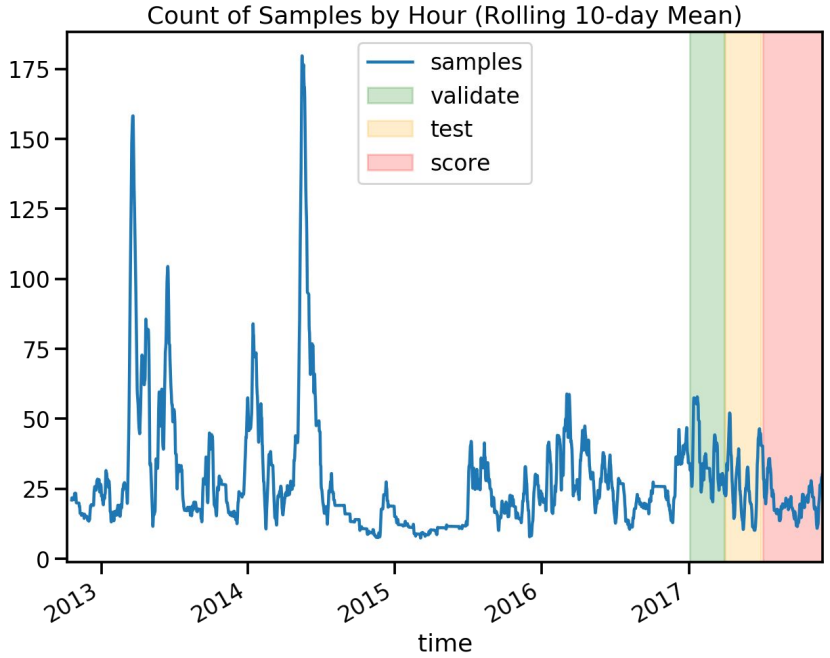
Weather: Batches

Condition Window
24 hours

Prediction Window
24 hours

Minimum Sequence Length
48 hours

Fixed Sample Length
48 hours



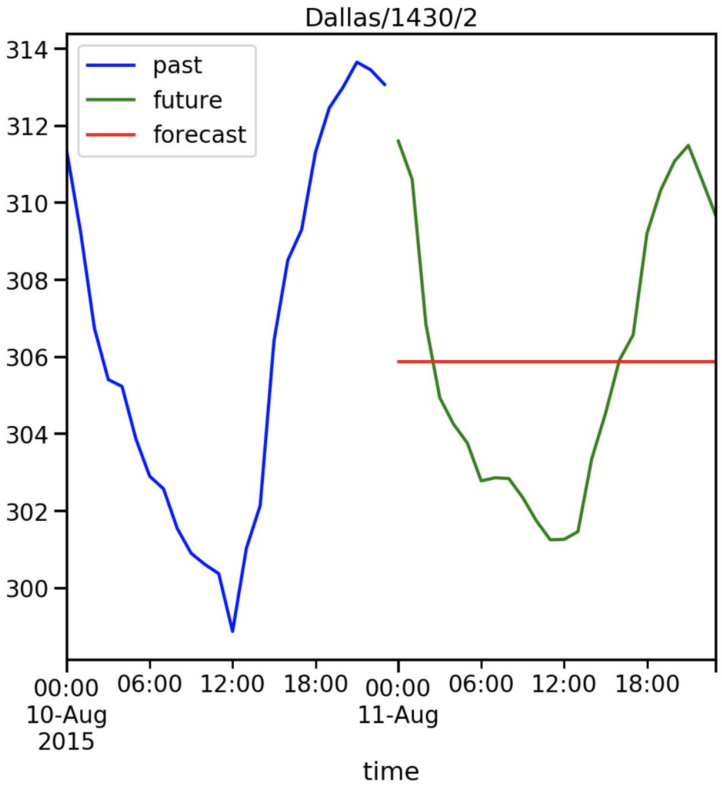
	samples	%	date_min	date_max
train	34436	83	2012-10-08	2016-12-30
validate	2288	6	2017-01-02	2017-03-30
test	2177	5	2017-04-01	2017-06-29
score	2399	6	2017-07-02	2017-11-29

Weather: Baseline Models

Mean Model

Forecast = mean of the past 24 hours

	MAE	ME	MSE	R2
train	4.00	-0.49	26.77	0.80
validate	4.62	-0.98	37.23	0.58
test	4.22	-1.07	28.52	0.61
score	4.30	-1.43	28.70	0.78

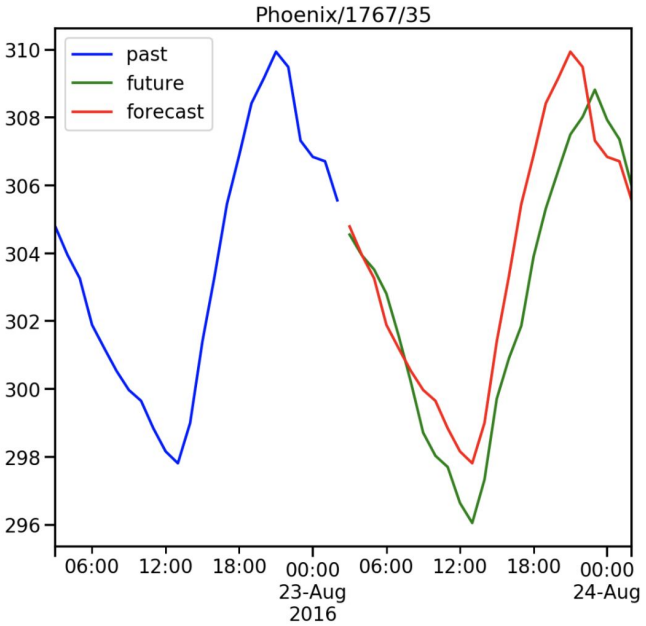


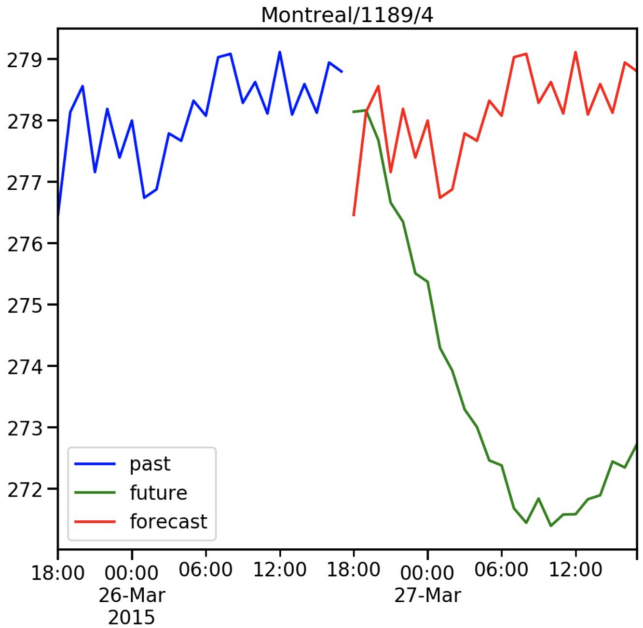
Weather: Baseline Models

	MAE	ME	MSE	R2
train	3.26	-0.49	21.82	0.84
validate	4.76	-0.98	40.25	0.54
test	3.20	-1.07	20.20	0.72
score	3.29	-1.43	22.31	0.83

Constant Model

Forecast = replay of the past 24 hours



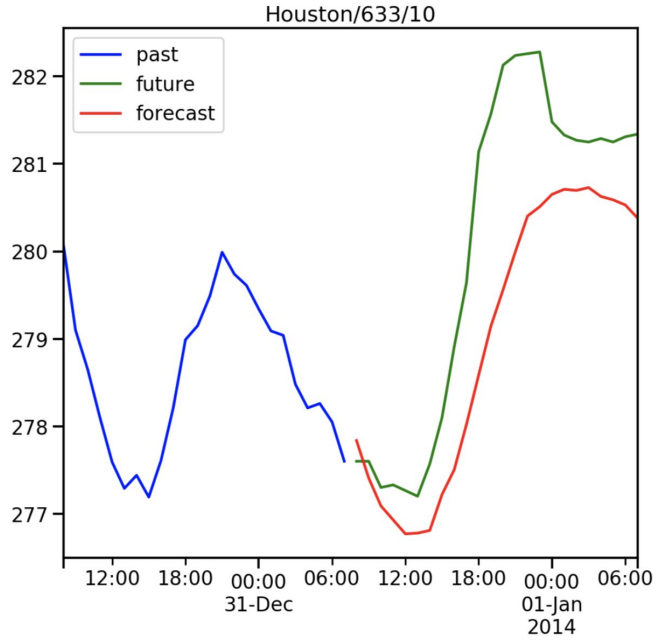


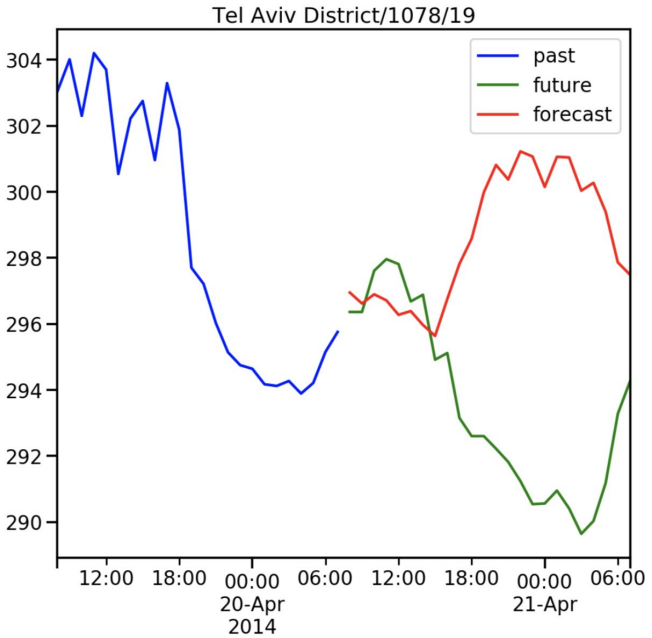
Weather: Baseline Models

Regression Model

24 independent linear regressions

	MAE	ME	MSE	R2
train	4.21	0.00	30.53	0.78
validate	4.04	-0.24	29.94	0.66
test	4.51	0.15	33.64	0.54
score	4.49	-0.26	33.45	0.75





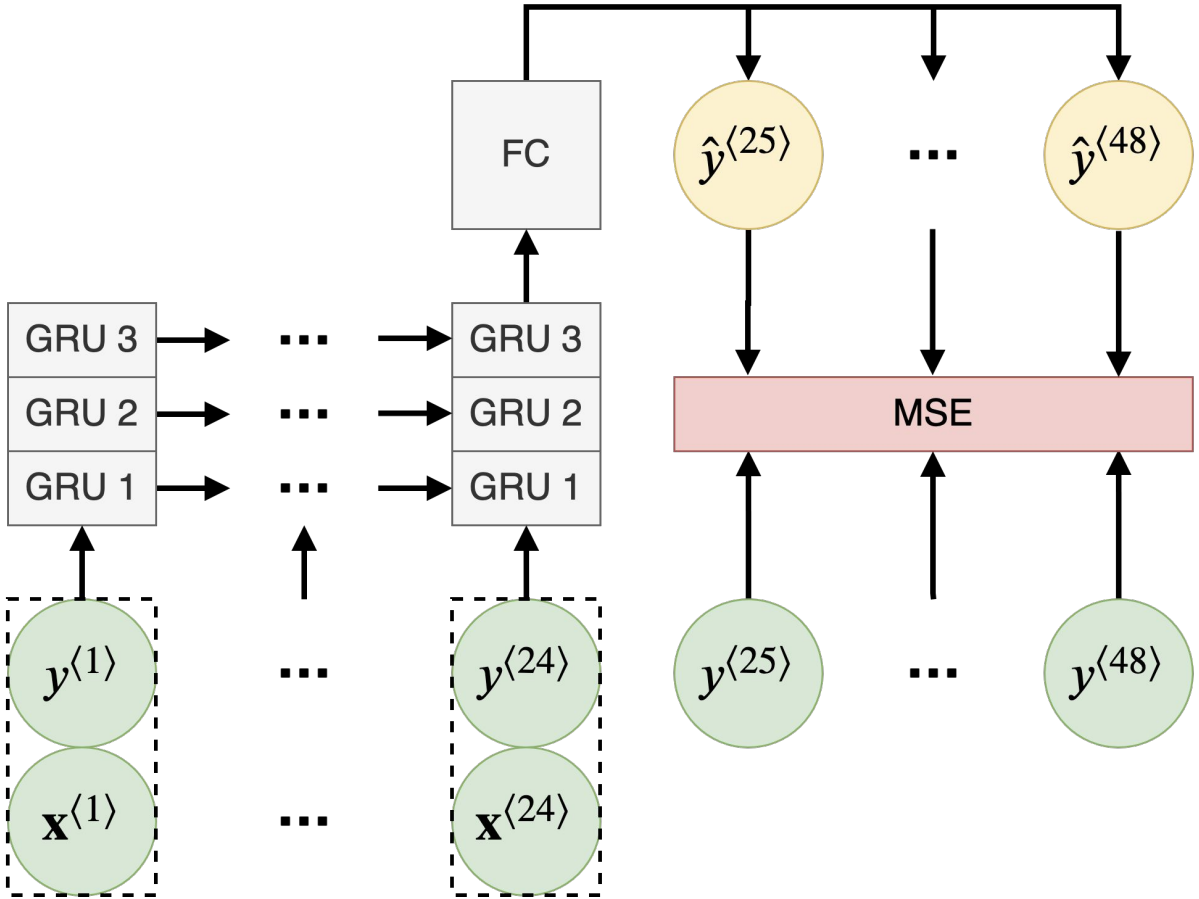
Weather: RNN

Encoder
Stacked GRUs
100 / 50 / 25 units

Decoder
Fully Connected

Parameters
~60K

Loss
Mean Squared Error

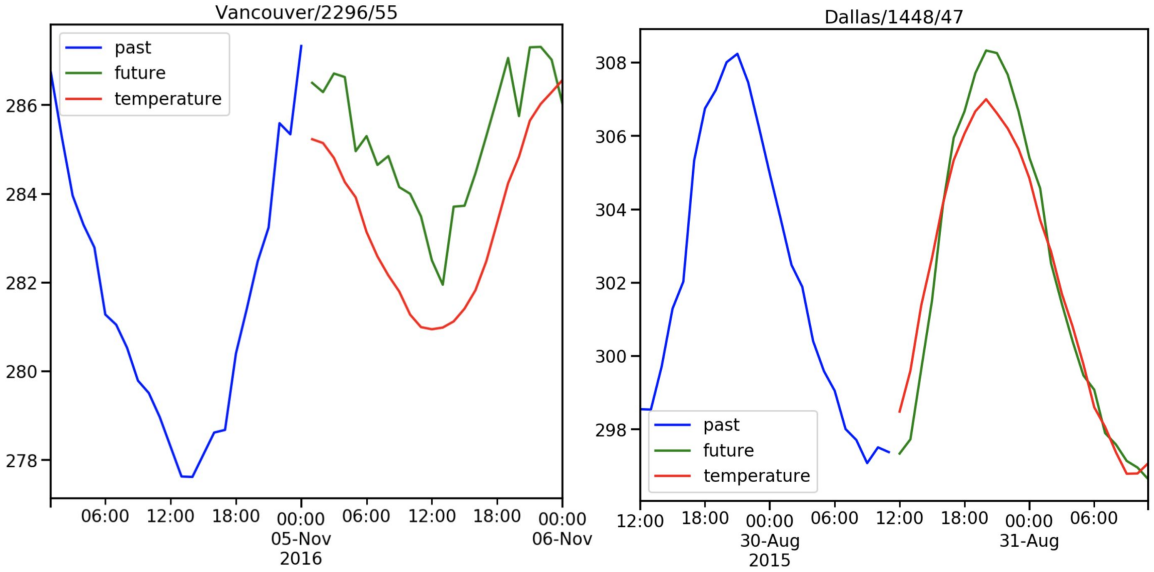


Weather: RNN

Inference

1. RNN model beats all baseline models.

2. The model is overfitting on the train segment.



	train		validate		test		score	
	MSE	R2	MSE	R2	MSE	R2	MSE	R2
mean	26.77	0.80	37.23	0.58	28.52	0.61	28.70	0.78
constant	21.82	0.84	40.25	0.54	20.20	0.72	22.31	0.83
regression	30.53	0.78	29.94	0.66	33.64	0.54	33.45	0.75
rnn	2.08	0.98	19.78	0.78	15.80	0.78	14.80	0.89

Assignment #3

Methodology

- Build and train RNN model on the weather forecasting task
- Try LSTM/GRU cell types
- Try deeper encoder / decoder
- Try L1 / L2 regularization
- Try adding calendar / location features
- Can’t submit exactly the same model as the example
- Must document changes to the training code

Grading

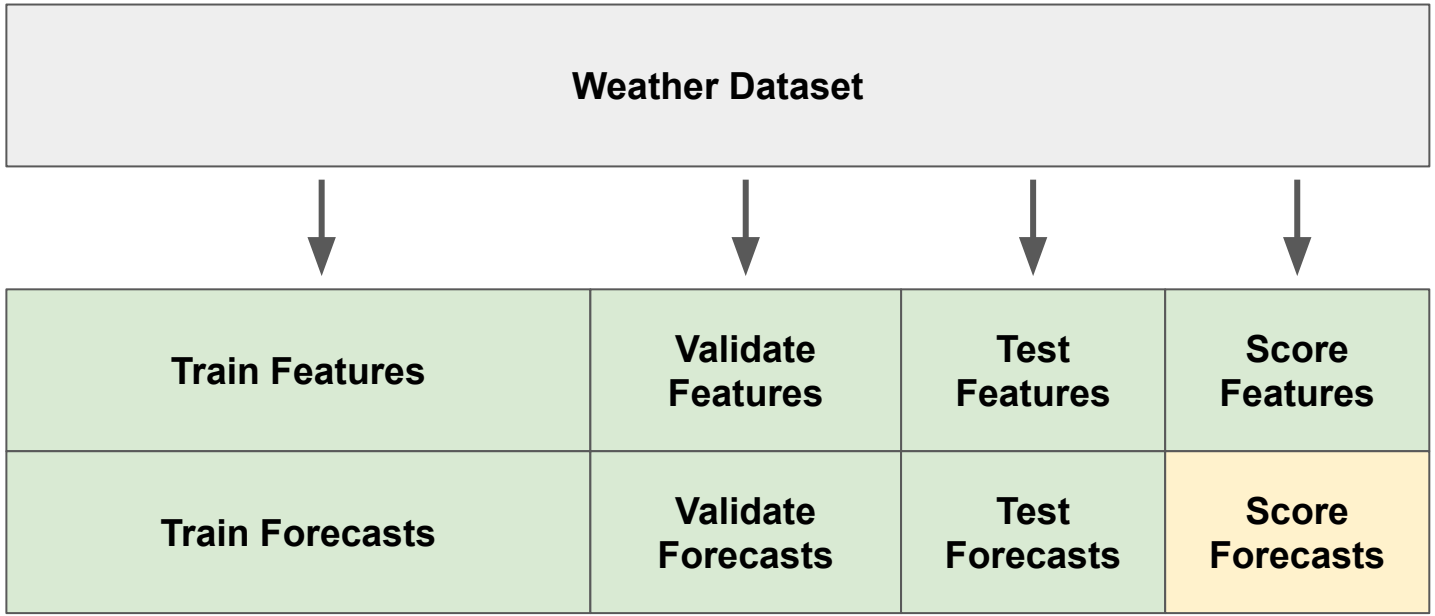
- Deadline - EOD Tuesday 10/27
- 10 points for R2 on Score segment > 0.85
- 10 points proportionally to the percentile versus other submissions

Assignment #3

Deliverables

1. Jupyter notebook and/or Python script
    - Data preprocessing
    - Construction and training of the final model
    - Inference on the Score dataset
2. Final model definition
    - Keras serialization in JSON format
3. Final model parameters
    - Keras serialization in H5 format
4. Temperature forecasts for the Score segment
    - Shape = 57576 x 1
    - Data Type = float32
    - File Type = Parquet

Assignment Dataset



Demo

WeatherForecast.ipynb

Case Study: MQ-RNN

Task

Time-series forecasting

Key Attributes

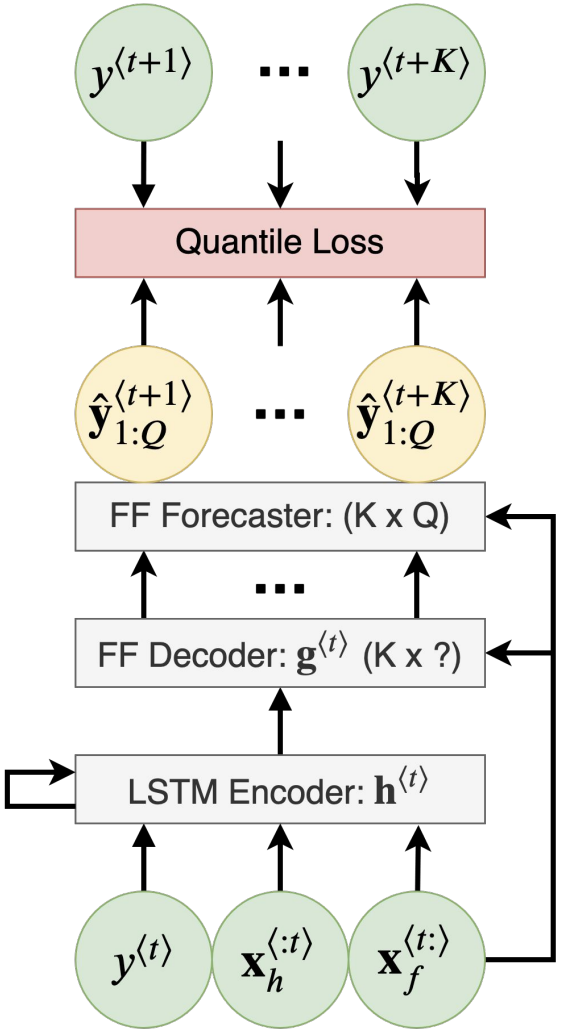
- Autoregressive model
- LSTM encoder
- Feed-forward decoder
- Quantile output
- Multiple time series

Time	**1**	**2**	**3**	**4**	**5**	**6**
Window	Conditioning			Prediction		
Horizon				1	2	3
Target						
q = 25						
q = 50						
q = 75						

Wen, R, et al. (2018)

MQ-RNN: Model

- Decoder summarizes horizon-specific context vector for each of K horizons
- Forecaster outputs prediction for each of K horizons and Q quantiles
- Forecaster shares parameters across horizons, but not across quantiles



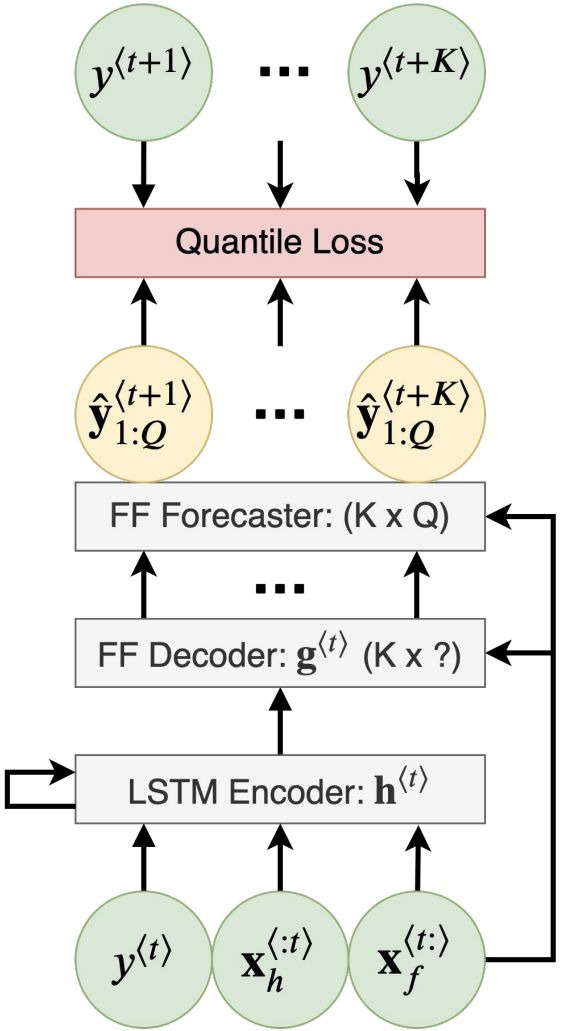
Wen, R, et al. (2018)

MQ-RNN: Model

$$\mathbf{g}^{\langle t \rangle} = (\mathbf{c}^{\langle t+1 \rangle}, \ldots, \mathbf{c}^{\langle t+K \rangle}, \mathbf{c}_a^{\langle t \rangle}) = g(\mathbf{h}^{\langle t \rangle}, \mathbf{x}_f^{\langle t: \rangle})$$

$$\hat{\mathbf{y}}^{\langle t+k \rangle} = (\hat{y}_1^{\langle t+k \rangle}, \ldots, \hat{y}_Q^{\langle t+k \rangle}) = f(\mathbf{c}^{\langle t+k \rangle}, \mathbf{c}_a^{\langle t \rangle}, x_f^{\langle t+k \rangle})$$

FC decoder looks at all horizons, while FC forecaster looks at one horizon at a time.



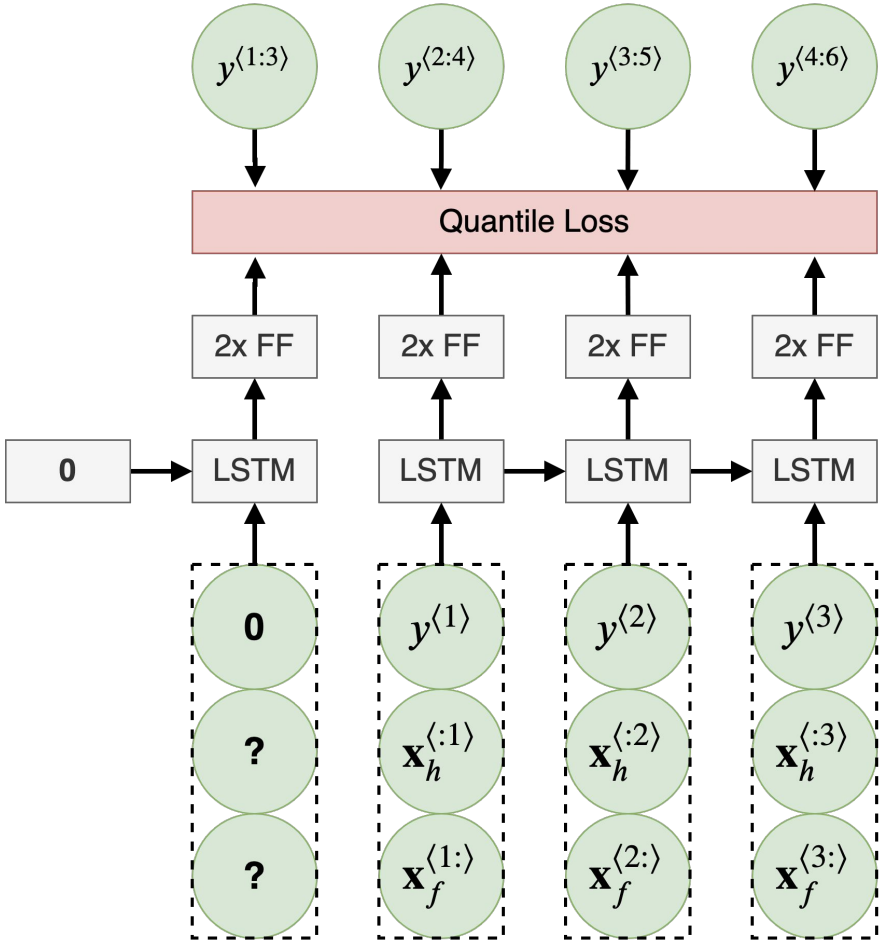
Wen, R, et al. (2018)

MQ-RNN: Training

Conditioning Window
Full trailing history

Prediction Window
3 periods

Forking Sequences
Unlike cutting sequences approach of DeepAR, the model outputs multi-horizon prediction at each step.

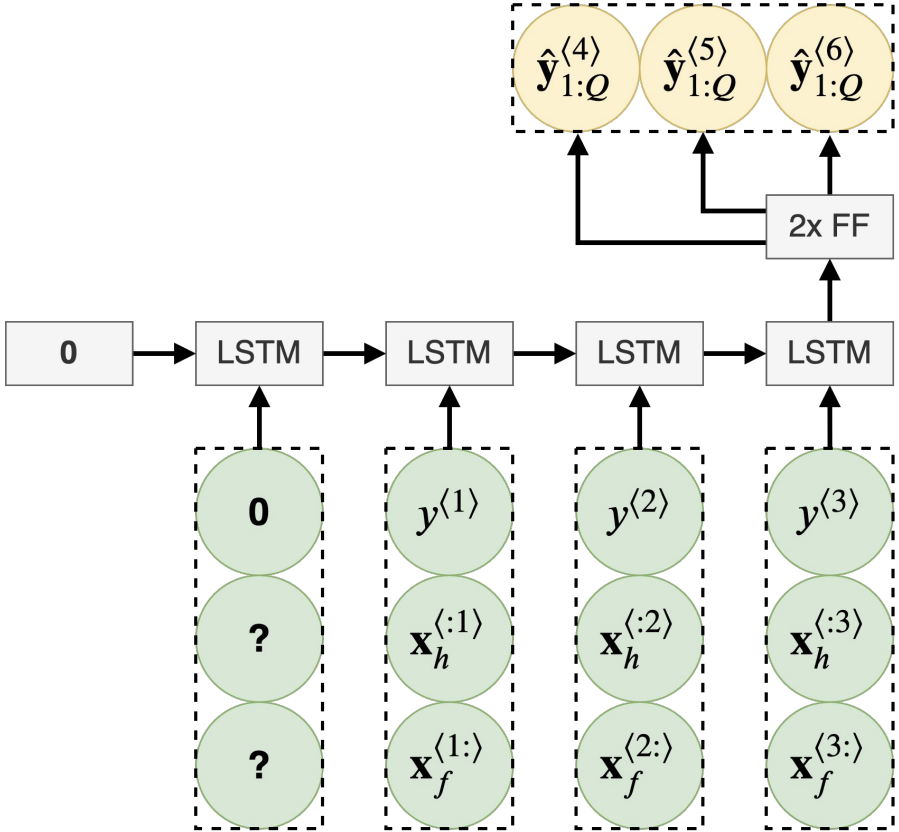


Wen, R, et al. (2018)

MQ-RNN: Inference

Conditioning Window
Full trailing history

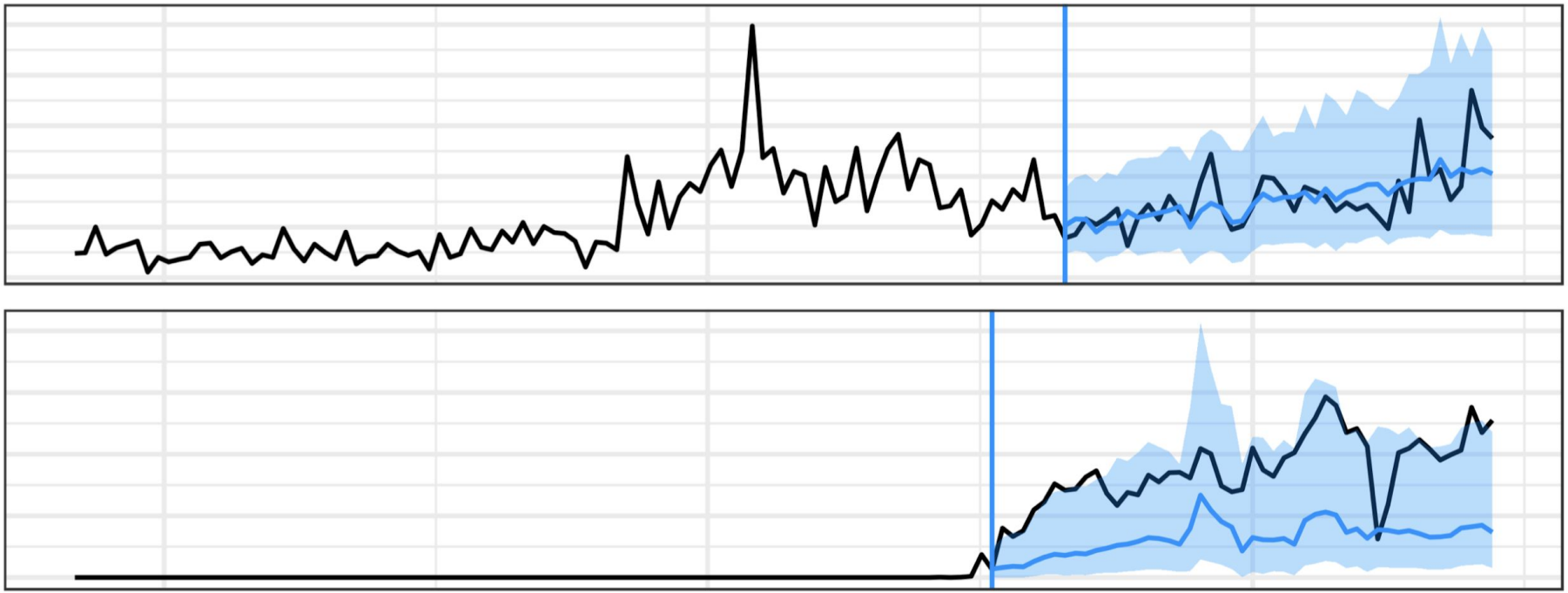
Prediction Window
3 periods



Wen, R, et al. (2018)

MQ-RNN: Performance

Warm start



Cold start

Wen, R, et al. (2018)

References

1. A Multi-Horizon Quantile Recurrent Forecaster (Wen, R, et al., 2018)
2. DeepAR: Probabilistic Forecasting with Autoregressive Recurrent Networks (Salinas, D., et al., 2019)
3. **Deep Learning (Goodfellow, I., Bengio, Y, and Courville, A., 2016)**
4. Long Short-Term Memory (Hochreiter, S, and Schmidhuber, J., 1997)
5. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation (Cho, K., et al., 2014)

Thanks