Lecture 3: RNN, CNN+RNN, encoder-decoder, ConvLSTM

Peyman Noursalehi

Recap

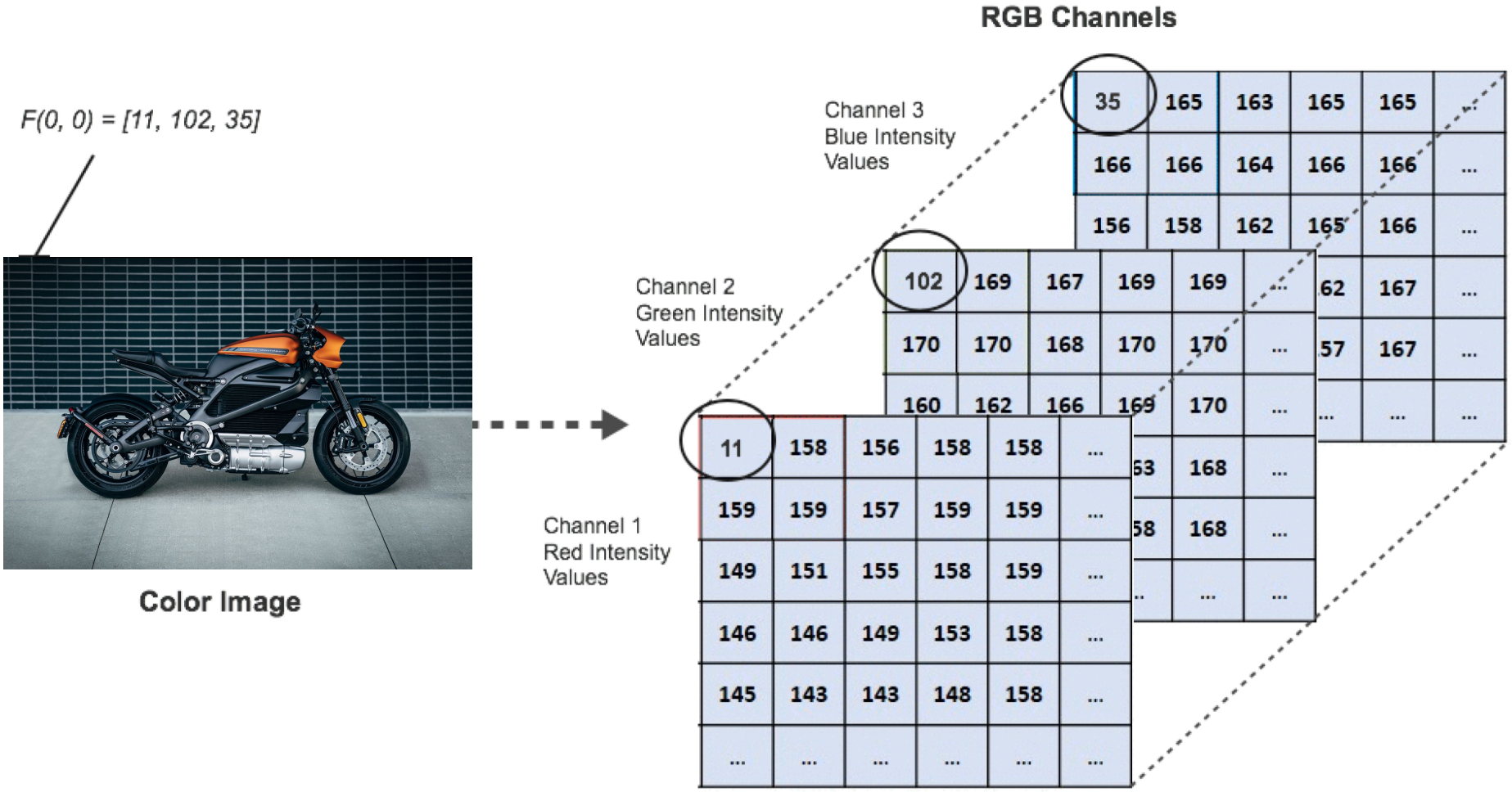
- Fundamentals of Deep learning
  - Architecture
  - Activation functions
  - Loss function
  - Regularization
  - Software frameworks
- Autoencoders
- Multilayer perceptron (MLP)
- Convolutional neural nets (CNN)

An image is

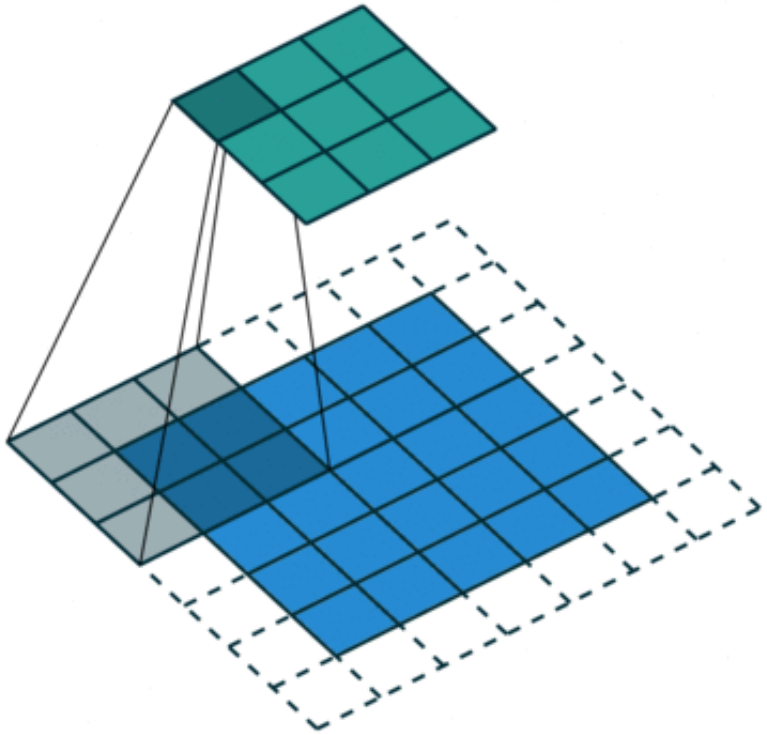
28 x 28
= 784 pixels

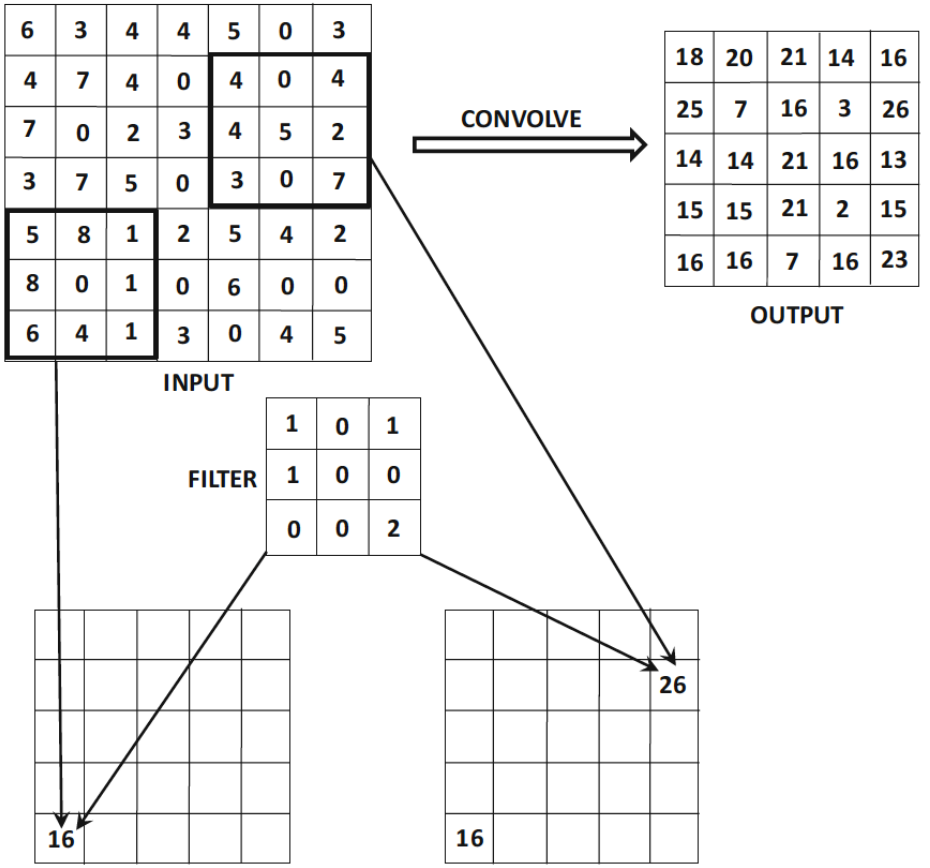
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	55	87	157	156	187	215	81	0	0	0	0	0
0	0	0	0	0	0	0	0	5	25	54	25	96	140	215	215	251	254	254	254	254	254	163	0	0	0	0	0
0	0	0	0	0	0	0	0	157	254	254	254	0	254	254	254	254	254	254	254	254	241	100	0	0	0	0	0
0	0	0	0	0	0	0	0	156	117	184	214	214	156	117	49	19	19	141	254	241	158	23	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	16	173	254	246	70	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	90	247	254	219	58	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	43	203	254	250	121	15	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	43	145	254	254	229	111	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	4	137	245	254	254	238	40	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	130	254	254	254	254	254	235	106	23	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	55	196	196	114	194	223	254	254	216	23	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	11	75	245	253	55	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	150	254	134	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	14	200	254	134	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	11	246	253	59	0	0	0	0	0	0	0	0	0
0	0	0	17	11	18	0	0	0	0	0	0	0	0	67	254	254	170	0	0	0	0	0	0	0	0	0	0
0	0	0	128	254	159	0	0	0	0	0	0	20	140	248	254	254	9	0	0	0	0	0	0	0	0	0	0
0	0	0	154	254	229	80	79	79	161	176	218	254	254	254	104	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	29	203	254	254	254	215	254	254	254	250	95	11	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	5	80	155	155	156	111	58	58	36	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

An image is


F(0, 0) = [11, 102, 35]
Color Image
RGB Channels
Channel 3
Blue Intensity
Values
Channel 2
Green Intensity
Values
Channel 1
Red Intensity
Values
35 165 163 165 165 ...
166 166 164 166 166 ...
156 158 162 165 166 ...
102 169 167 169 169 ...
170 170 168 170 170 ...
160 162 166 169 170 ...
11 158 156 158 158 ...
159 159 157 159 159 ...
149 151 155 158 159 ...
146 146 149 153 158 ...
145 143 143 148 158 ...
... ...

Convolutional Neural Net





A typical CNN architecture

Contents lists available at ScienceDirect

Transportation Research Part C

journal homepage: www.elsevier.com/locate/trc

Review

Enhancing transportation systems via deep learning: A survey

Yuan Wang[a], Dongxiang Zhang[b,*], Ying Liu[b], Bo Dai[b], Loo Hay Lee[a]

[a] Department of Industrial Systems Engineering and Management, National University of Singapore, Singapore
[b] School of Computer Science and Engineering, University of Electronic Science and Technology of China, China

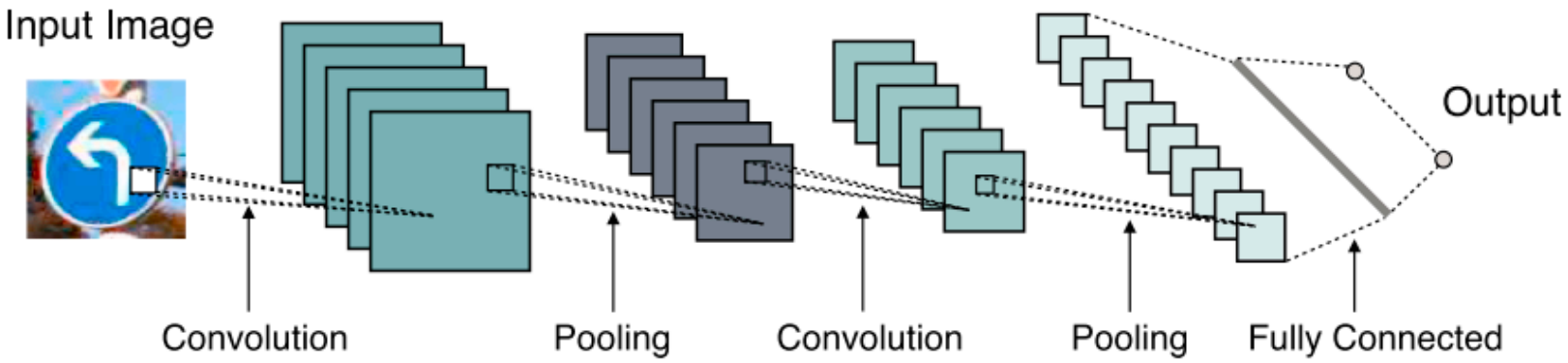


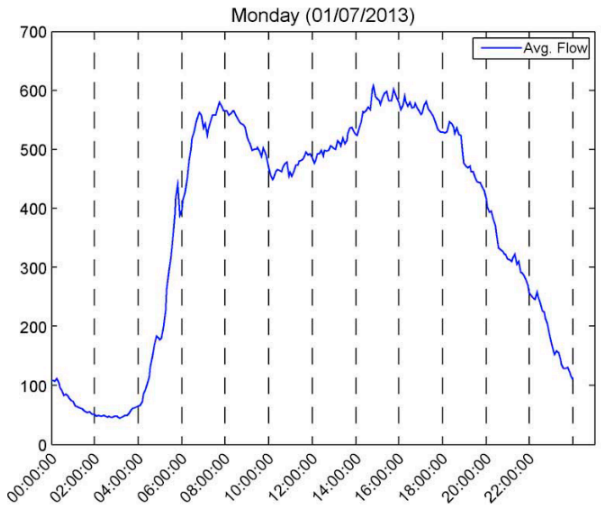
Fig. 3. Model architecture of convolutional neural network.

Our first DL model

- $D = \{x_t, x_{t-1}, \ldots, x_1\}$
- Let's turn it into a supervised learning problem
  - Choose a lookback horizon h

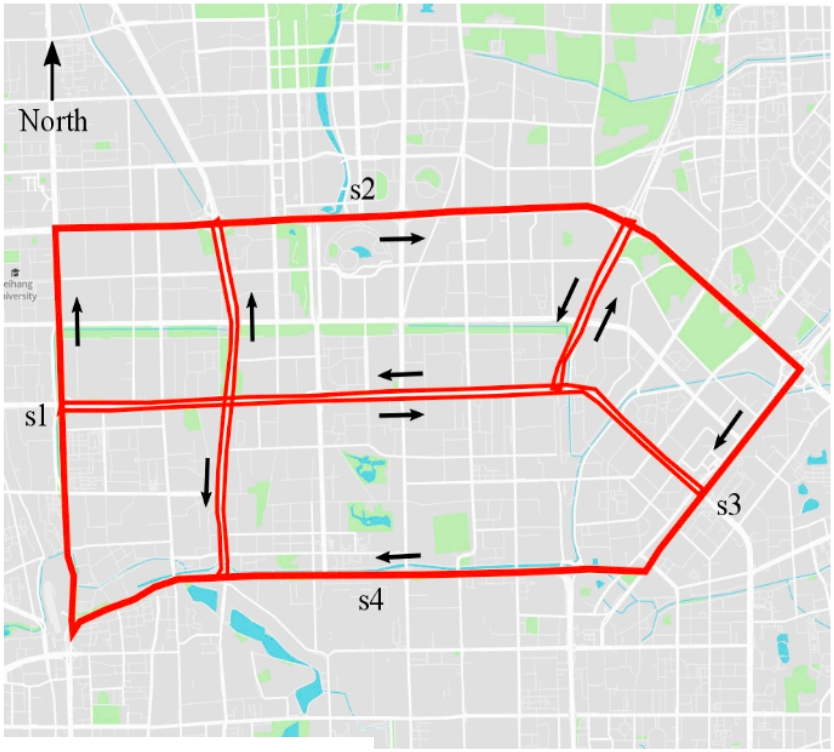
$$\{x_t\} \quad \{x_{t-1}, x_{t-2}, \ldots, x_{t-h}\}$$
$$\{x_{t-1}\} \quad \{x_{t-2}, x_{t-1}, \ldots, x_{t-h-1}\}$$

- Note: fixed input size



Predict traffic speed for an entire city with CNN

- 10-min ahead traffic prediction
  - Using last 40-min traffic speeds
  - Aggregated into 2-min intervals
- GPS data from 10,000 taxis
  - 37 days



sensors

MDPI

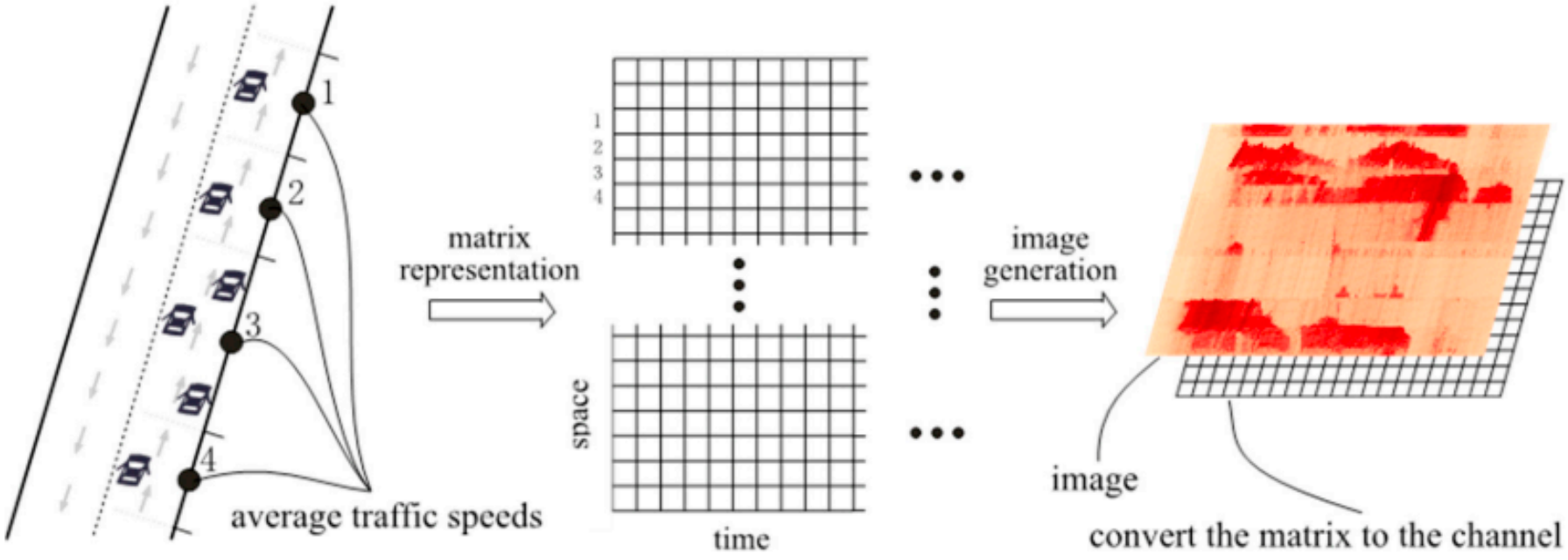
Article

Learning Traffic as Images: A Deep Convolutional Neural Network for Large-Scale Transportation Network Speed Prediction

Xiaolei Ma [1], Zhuang Dai [1], Zhengbing He [2], Jihui Ma [2,*], Yong Wang [3] and Yunpeng Wang [1]

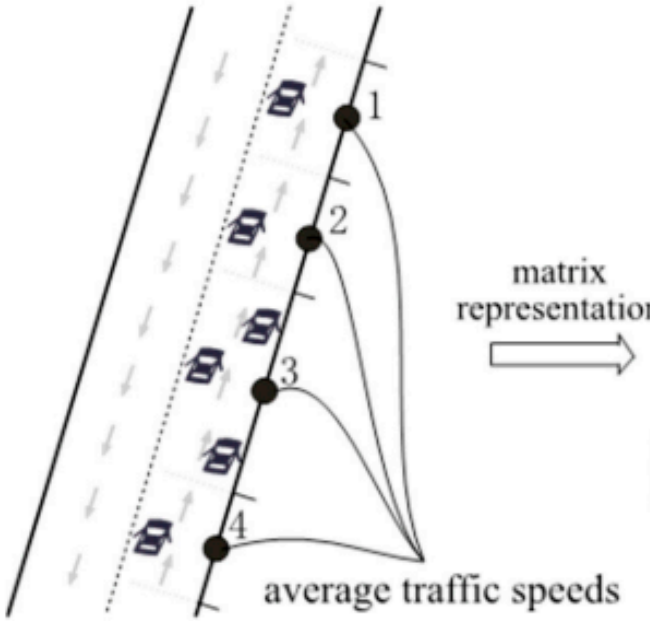
Predict traffic speed for an entire city with CNN

- We need to represent the input as a multi-dimensional image



Predict traffic speed for an entire city with CNN

- Q sections, N time periods



matrix representation

$$M = \begin{bmatrix} m_{11}, m_{12}, \cdots, m_{1N} \\ m_{21}, m_{22}, \cdots, m_{2N} \\ \vdots \quad \vdots \quad \cdots \quad \vdots \\ m_{Q1}, m_{Q2}, \cdots, m_{QN} \end{bmatrix}$$

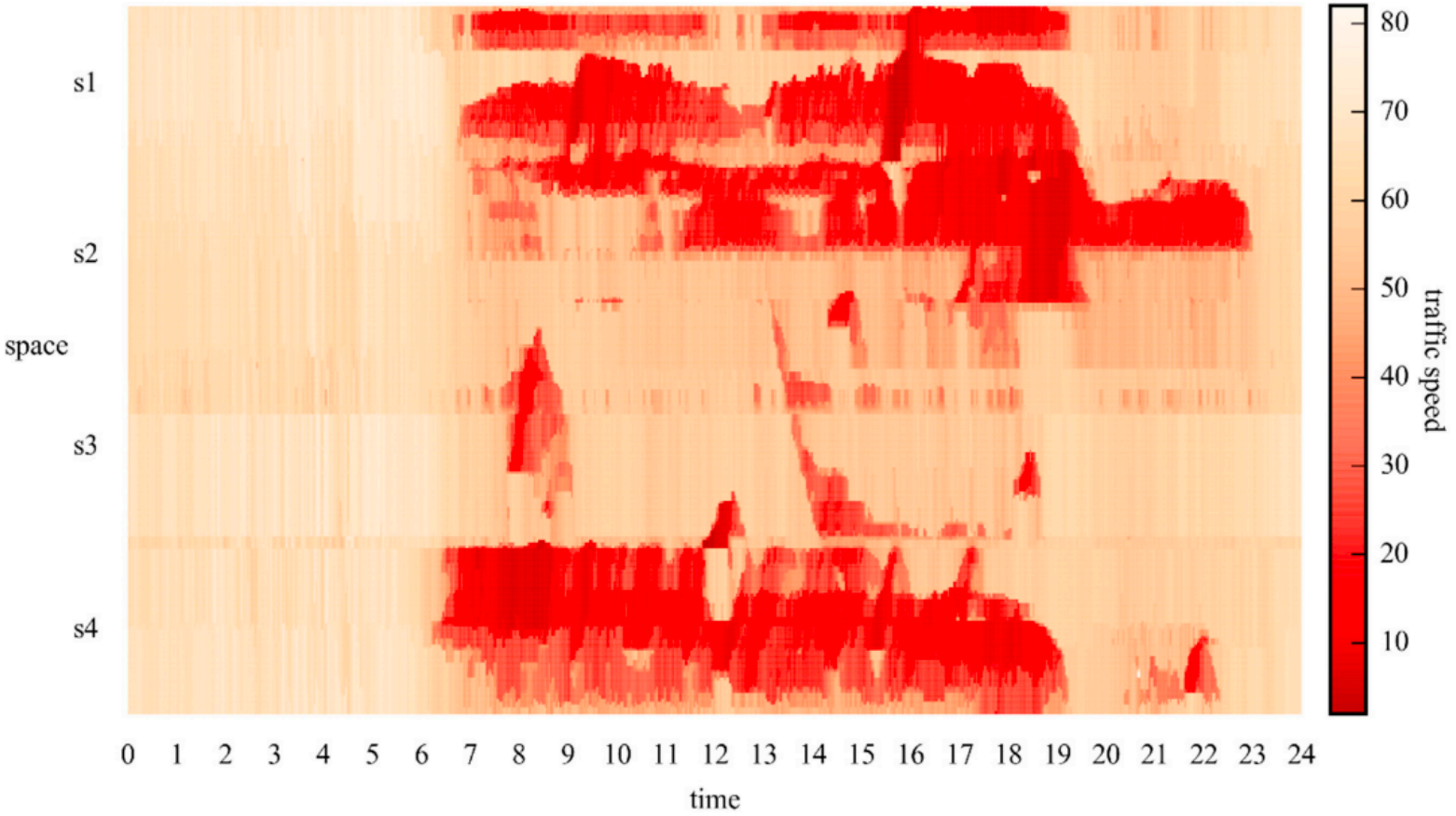
Predict traffic speed for an entire city with CNN

- 236 sections
- Using last 40-min traffic speeds
- Aggregated into 2-min intervals

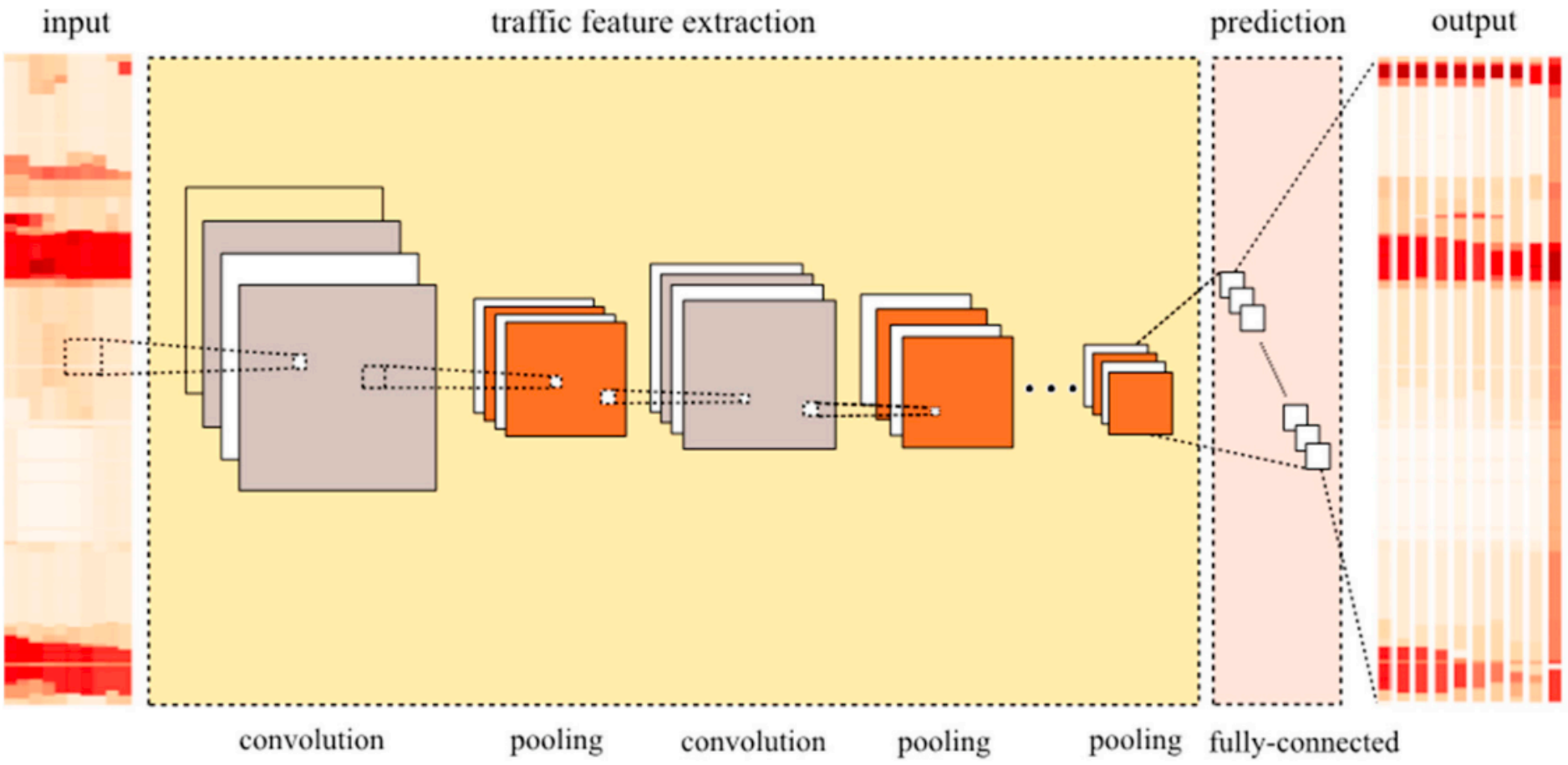
$$M = \begin{bmatrix} m_{11}, m_{12}, \cdots, m_{1N} \\ m_{21}, m_{22}, \cdots, m_{2N} \\ \vdots \quad \vdots \quad \cdots \quad \vdots \\ m_{Q1}, m_{Q2}, \cdots, m_{QN} \end{bmatrix}$$

Predict traffic speed for an entire city with CNN

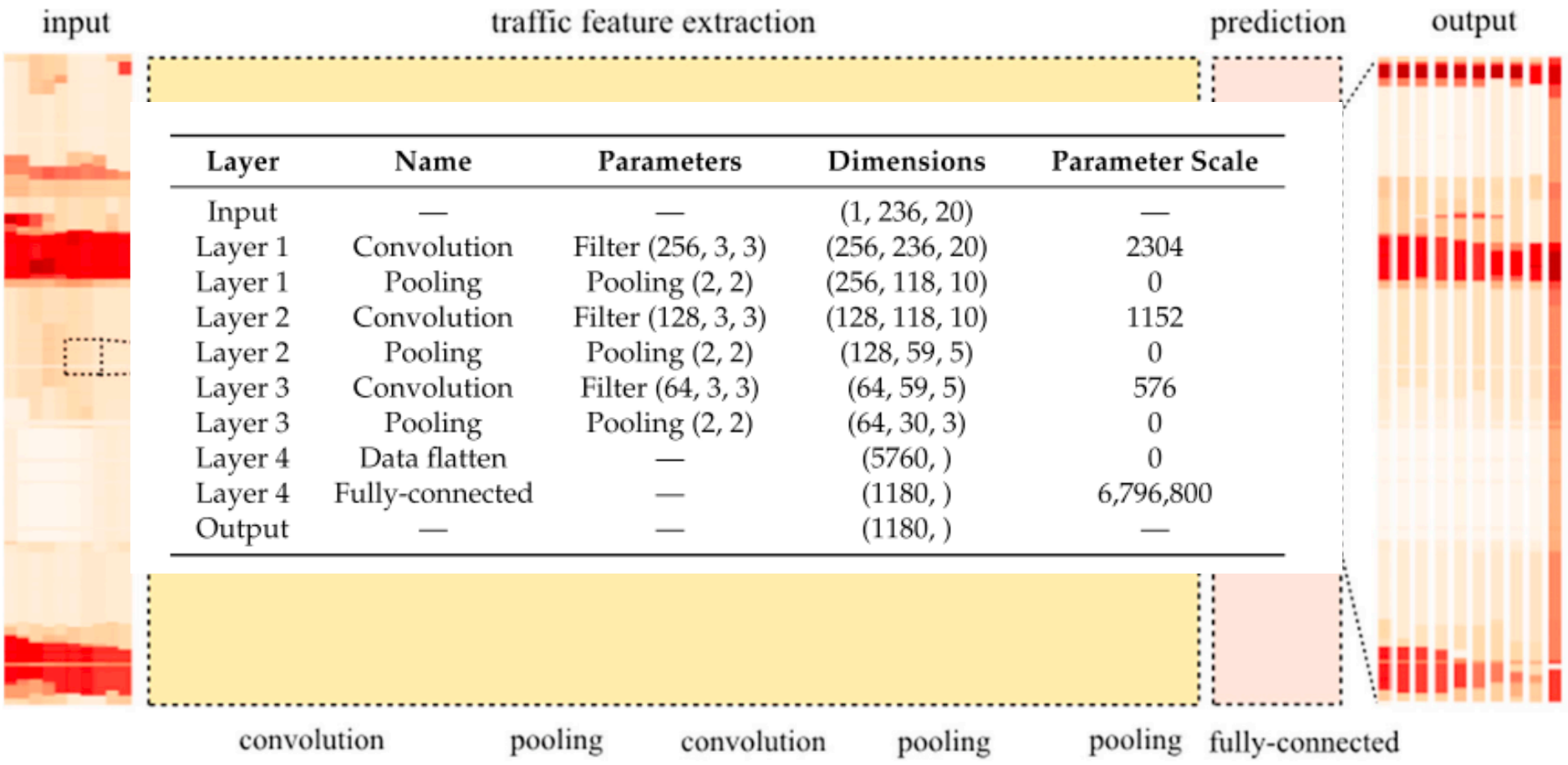
- Q sections, N time periods



Predict traffic speed for an entire city with CNN



Predict traffic speed for an entire city with CNN

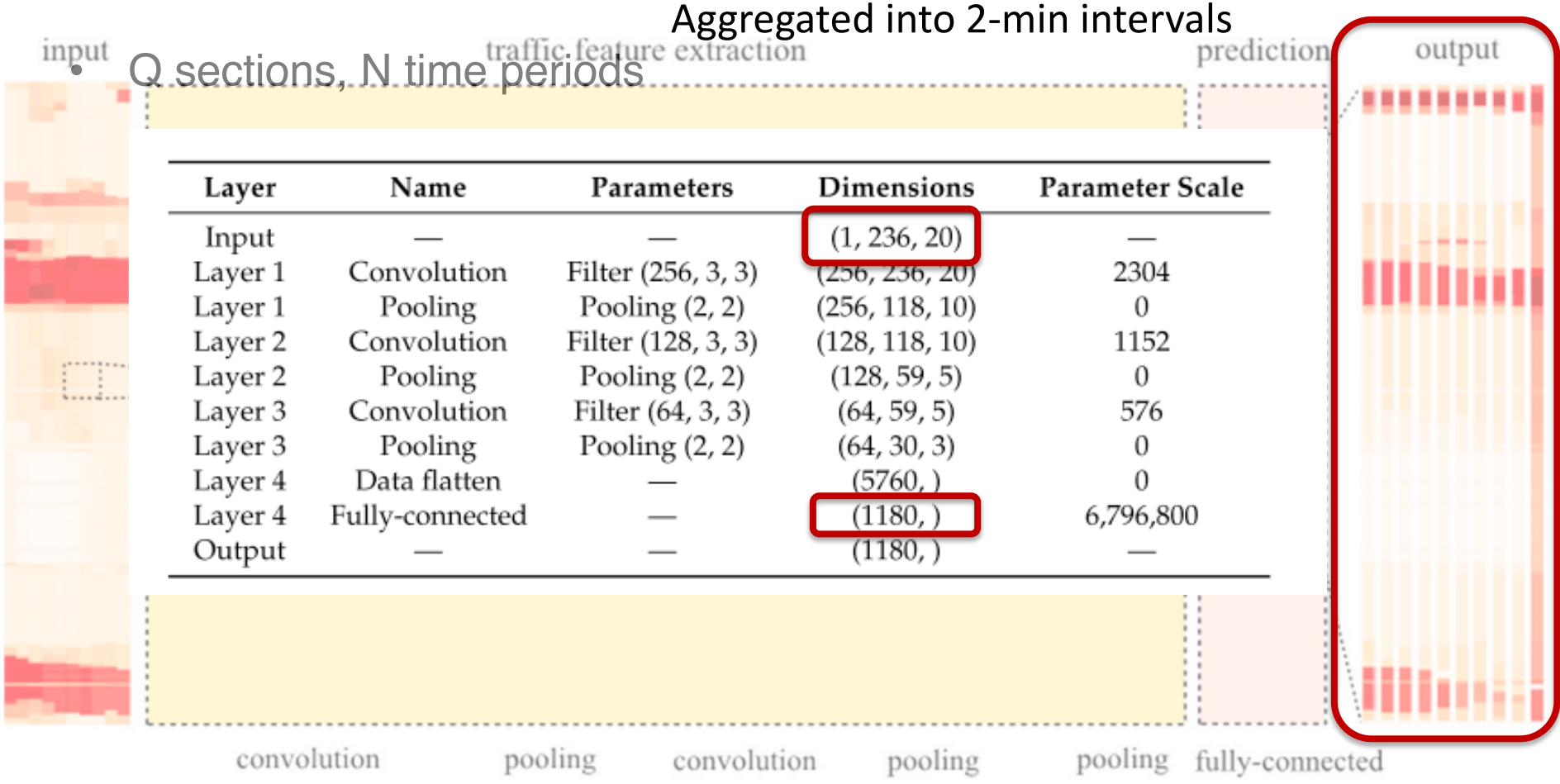


Layer	Name	Parameters	Dimensions	Parameter Scale
Input	—	—	(1, 236, 20)	—
Layer 1	Convolution	Filter (256, 3, 3)	(256, 236, 20)	2304
Layer 1	Pooling	Pooling (2, 2)	(256, 118, 10)	0
Layer 2	Convolution	Filter (128, 3, 3)	(128, 118, 10)	1152
Layer 2	Pooling	Pooling (2, 2)	(128, 59, 5)	0
Layer 3	Convolution	Filter (64, 3, 3)	(64, 59, 5)	576
Layer 3	Pooling	Pooling (2, 2)	(64, 30, 3)	0
Layer 4	Data flatten	—	(5760,)	0
Layer 4	Fully-connected	—	(1180,)	6,796,800
Output	—	—	(1180,)	—

Predict traffic speed for an entire city with CNN

Using last 40-min traffic speeds
Aggregated into 2-min intervals

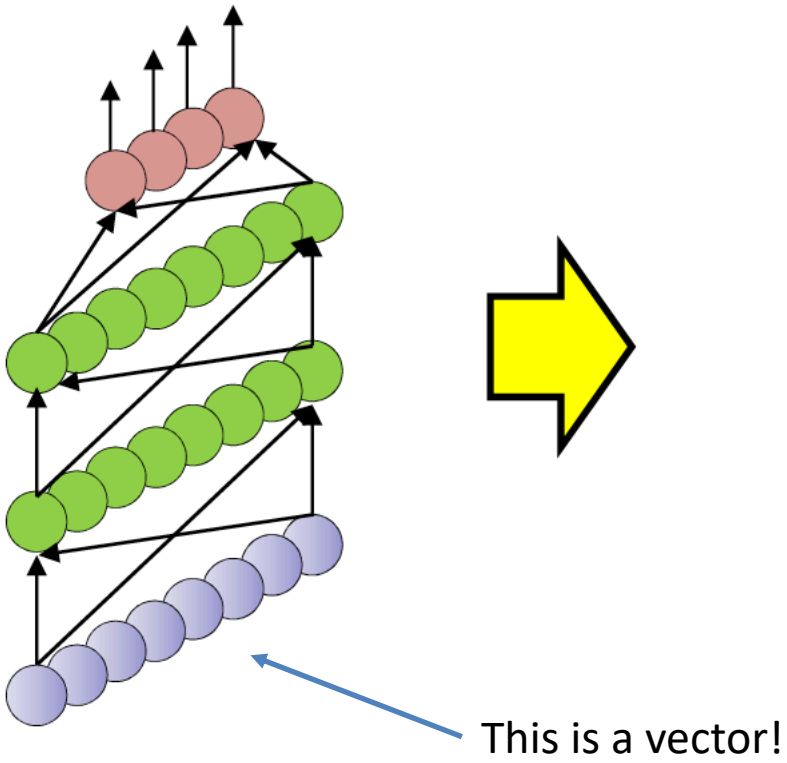
- Q sections, N time periods



Layer	Name	Parameters	Dimensions	Parameter Scale
Input	—	—	(1, 236, 20)	—
Layer 1	Convolution	Filter (256, 3, 3)	(256, 236, 20)	2304
Layer 1	Pooling	Pooling (2, 2)	(256, 118, 10)	0
Layer 2	Convolution	Filter (128, 3, 3)	(128, 118, 10)	1152
Layer 2	Pooling	Pooling (2, 2)	(128, 59, 5)	0
Layer 3	Convolution	Filter (64, 3, 3)	(64, 59, 5)	576
Layer 3	Pooling	Pooling (2, 2)	(64, 30, 3)	0
Layer 4	Data flatten	—	(5760,)	0
Layer 4	Fully-connected	—	(1180,)	6,796,800
Output	—	—	(1180,)	—

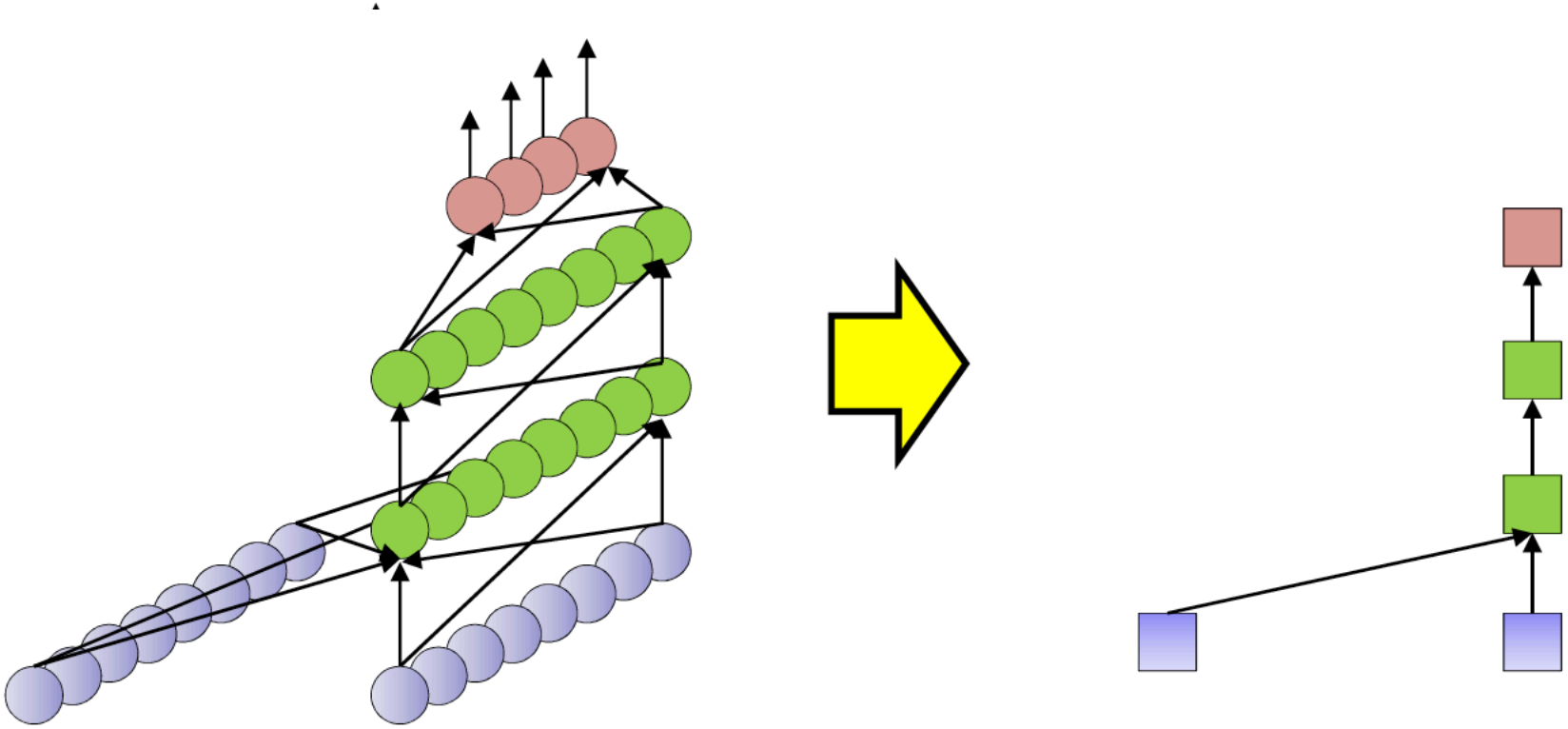
CNN for timeseries data

Clarifying the notation



Source: Bhiksha Raj

Clarifying the notation

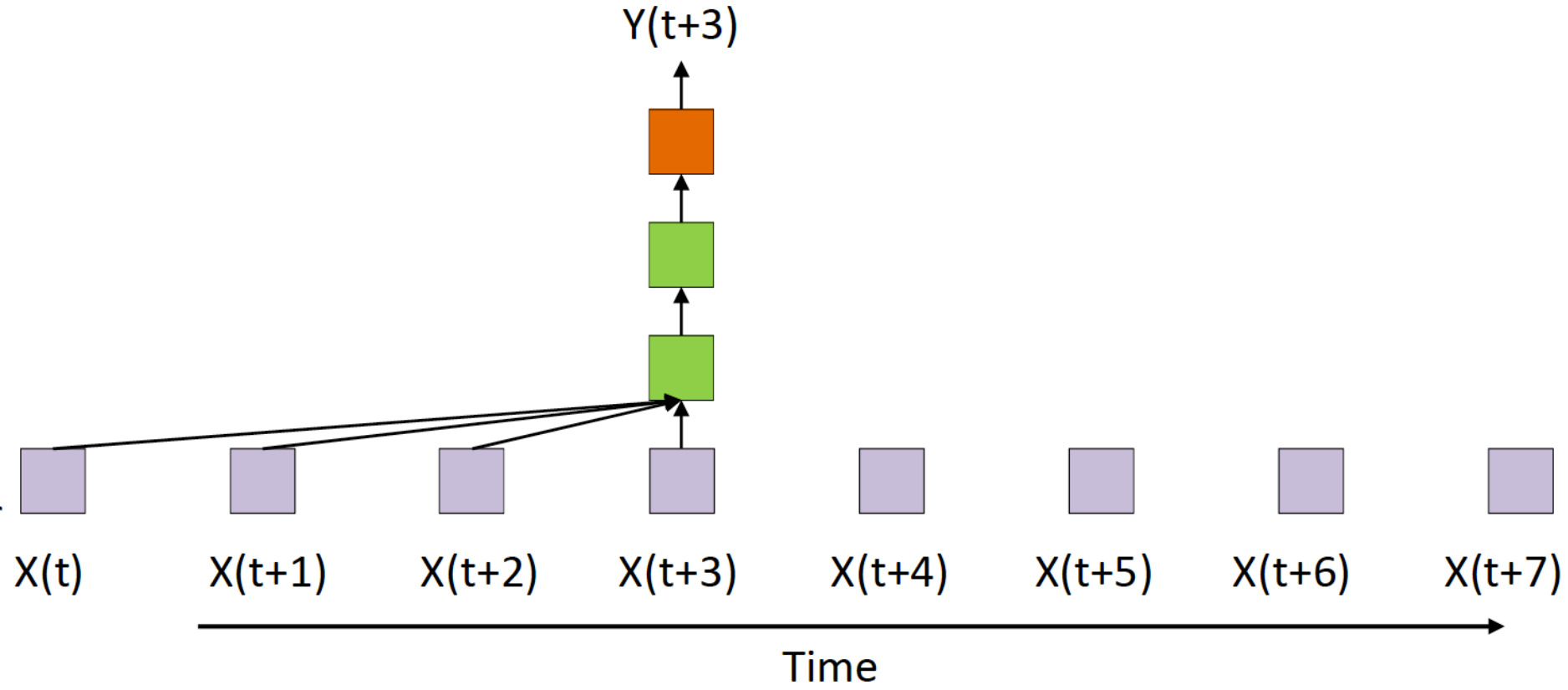


Source: Bhiksha Raj

CNN for timeseries data

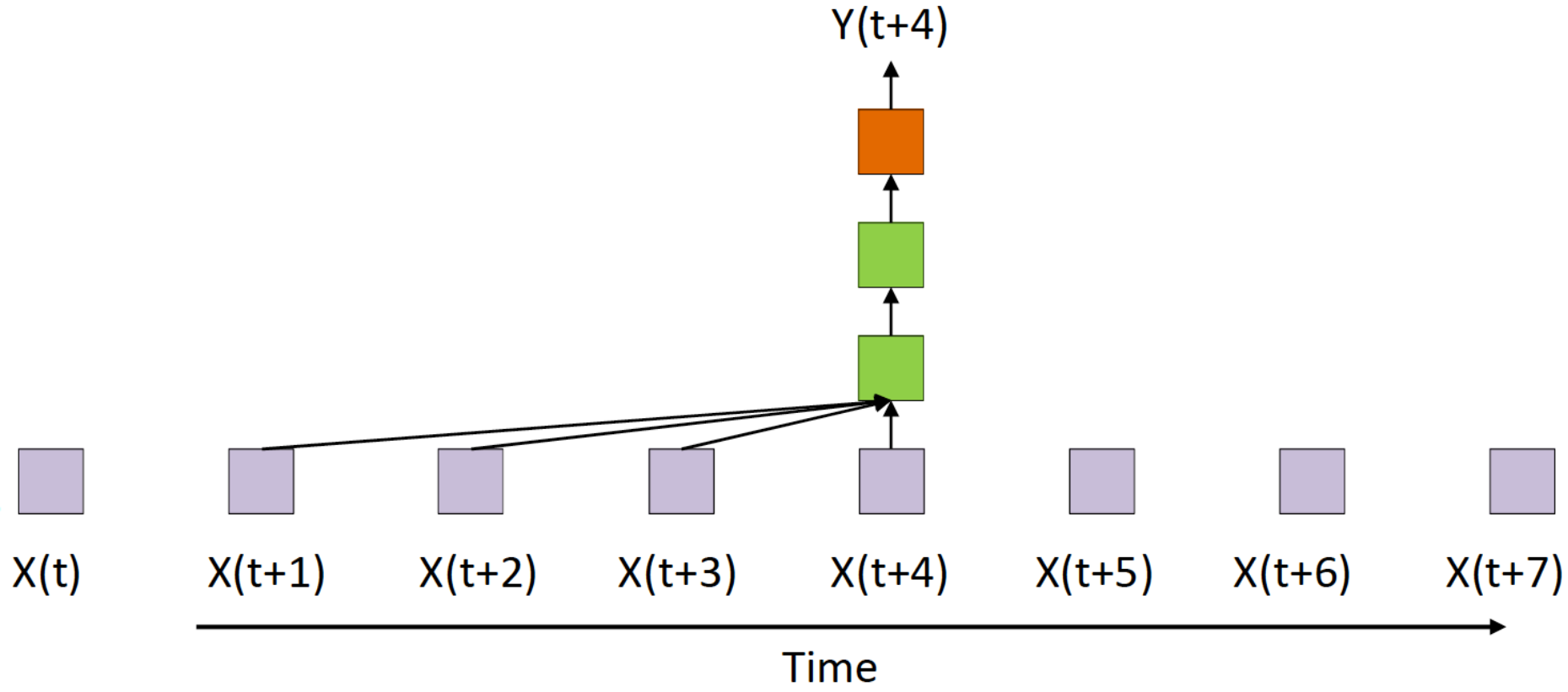
- CNNs are great at capturing spatial dependencies
- But in many situations one must consider a series of inputs to produce an output
  - Timeseries data
- The output might also be a time series
  - Traffic in 15, 30, 45 mins

Sliding window



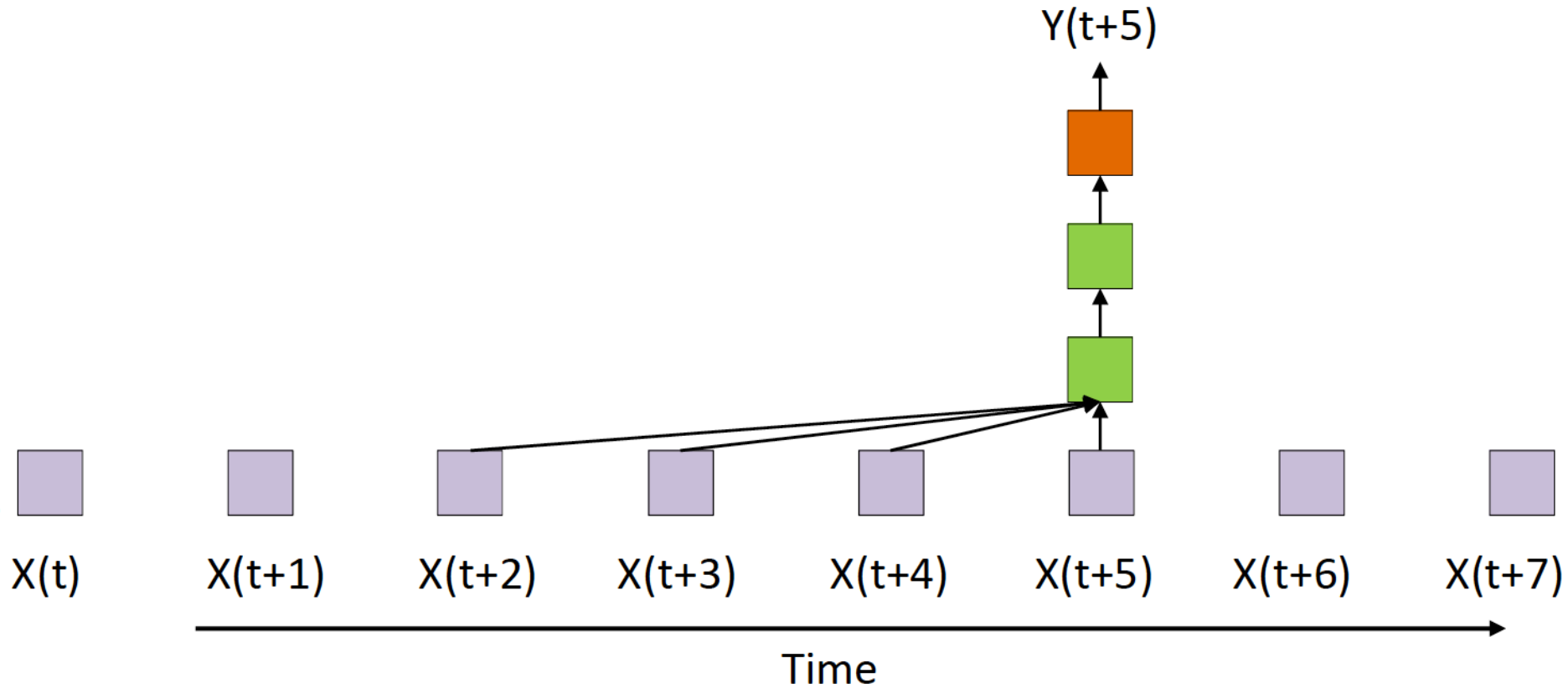
Source: Bhiksha Raj

Sliding window



Source: Bhiksha Raj

Sliding window

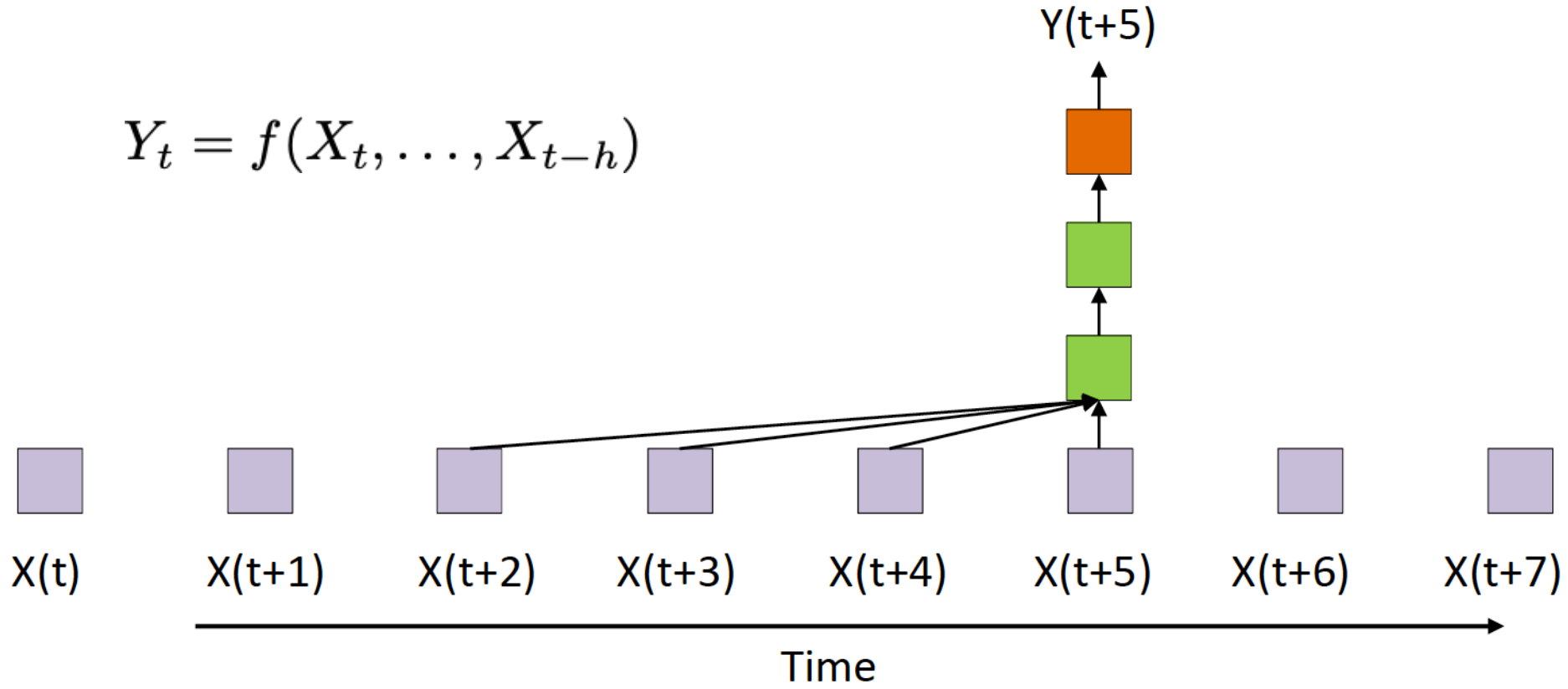


Source: Bhiksha Raj

Sliding window

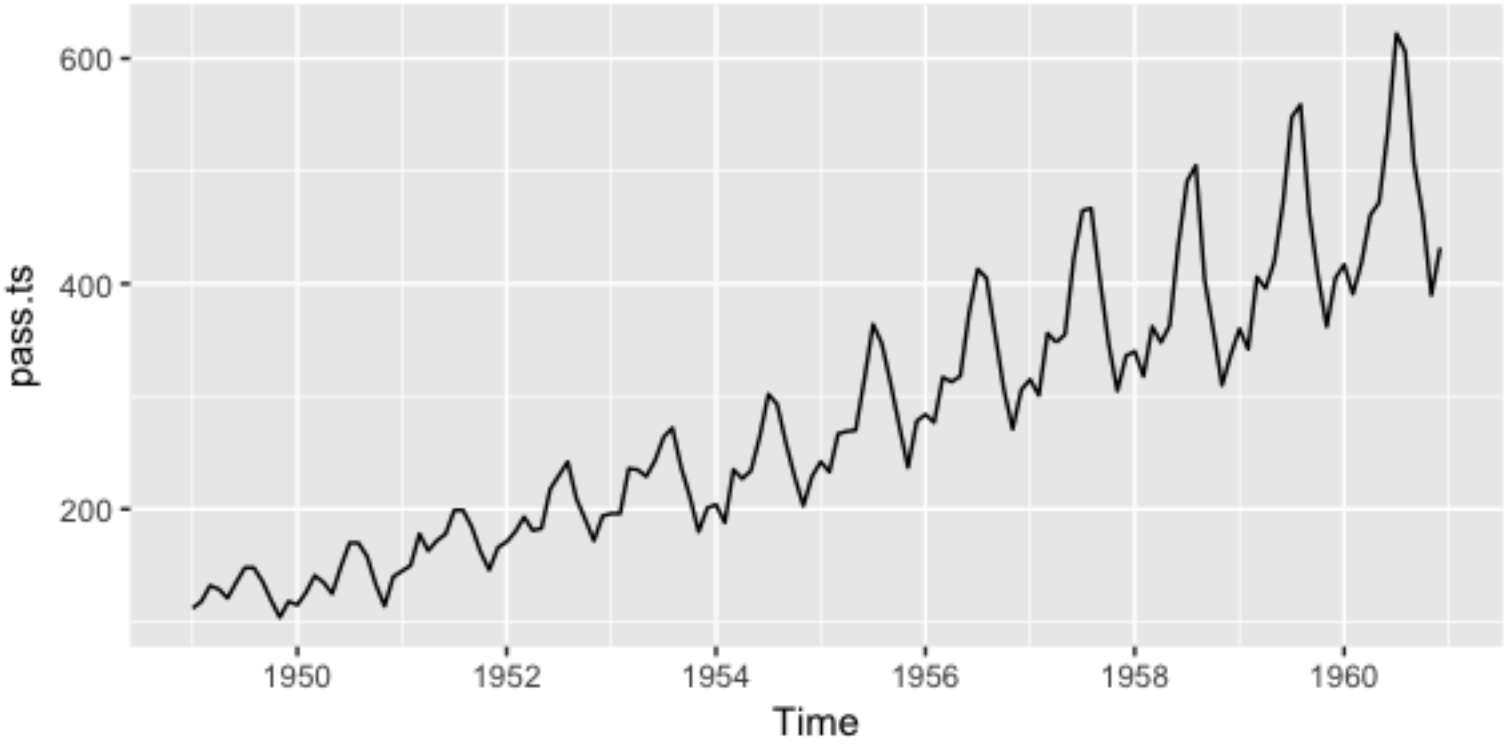
This is just a 1 dimensional CNN

$$Y_t = f(X_t, \ldots, X_{t-h})$$

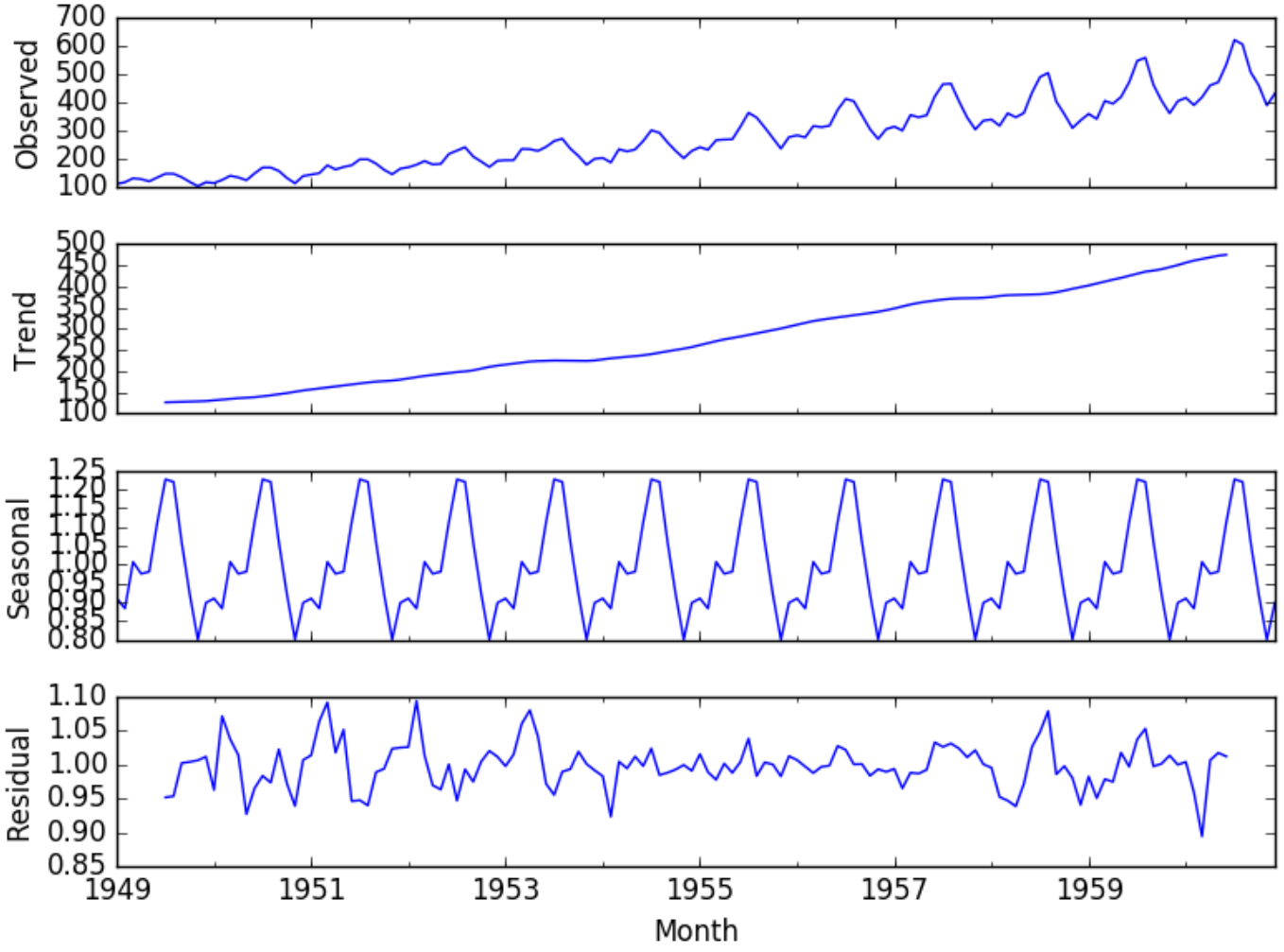


Source: Bhiksha Raj

A typical timeseries

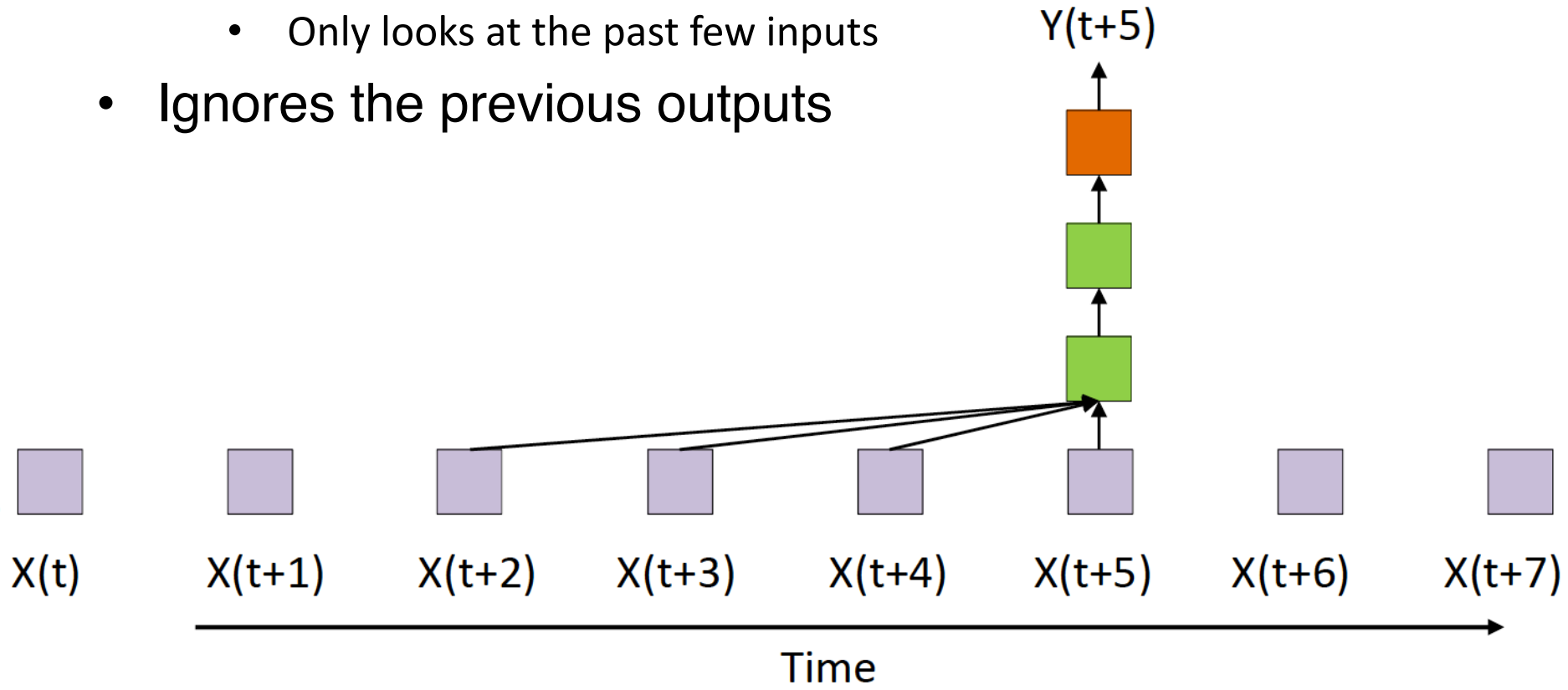


A typical timeseries



Sliding window

- Limited memory (from inputs)
  - Only looks at the past few inputs
- Ignores the previous outputs

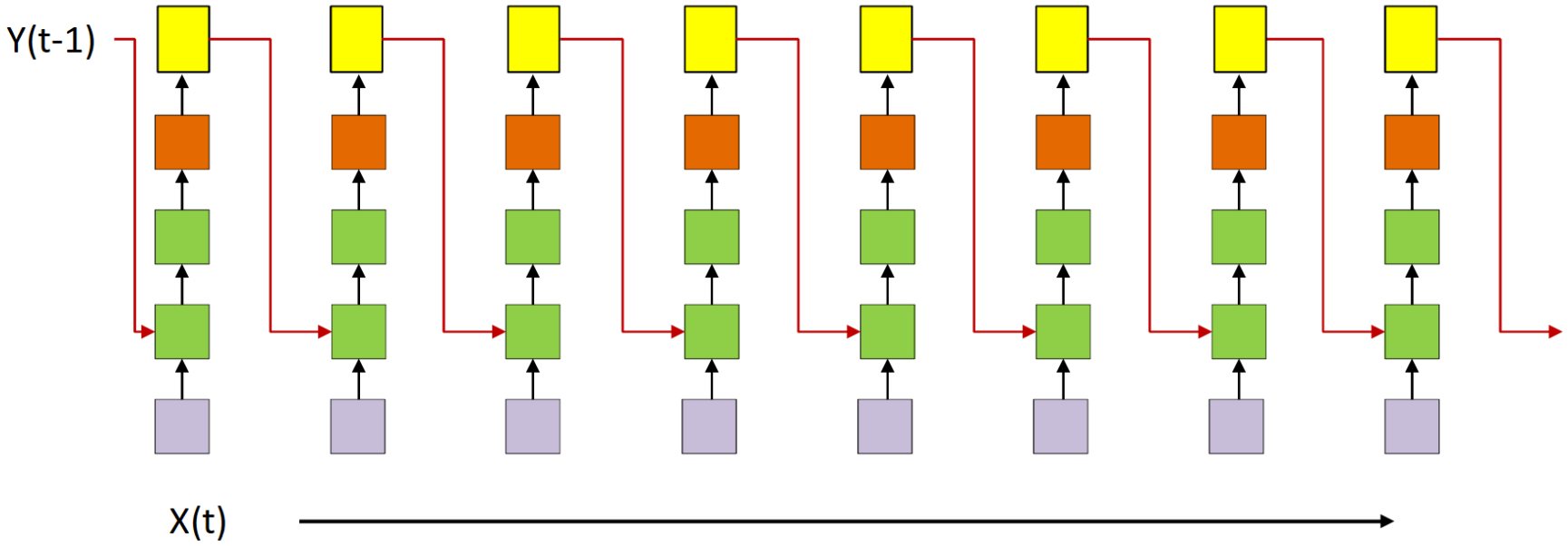


Source: Bhiksha Raj

NARX networks

- Nonlinear autoregressive network with exogenous inputs
- Contrast with ARIMA

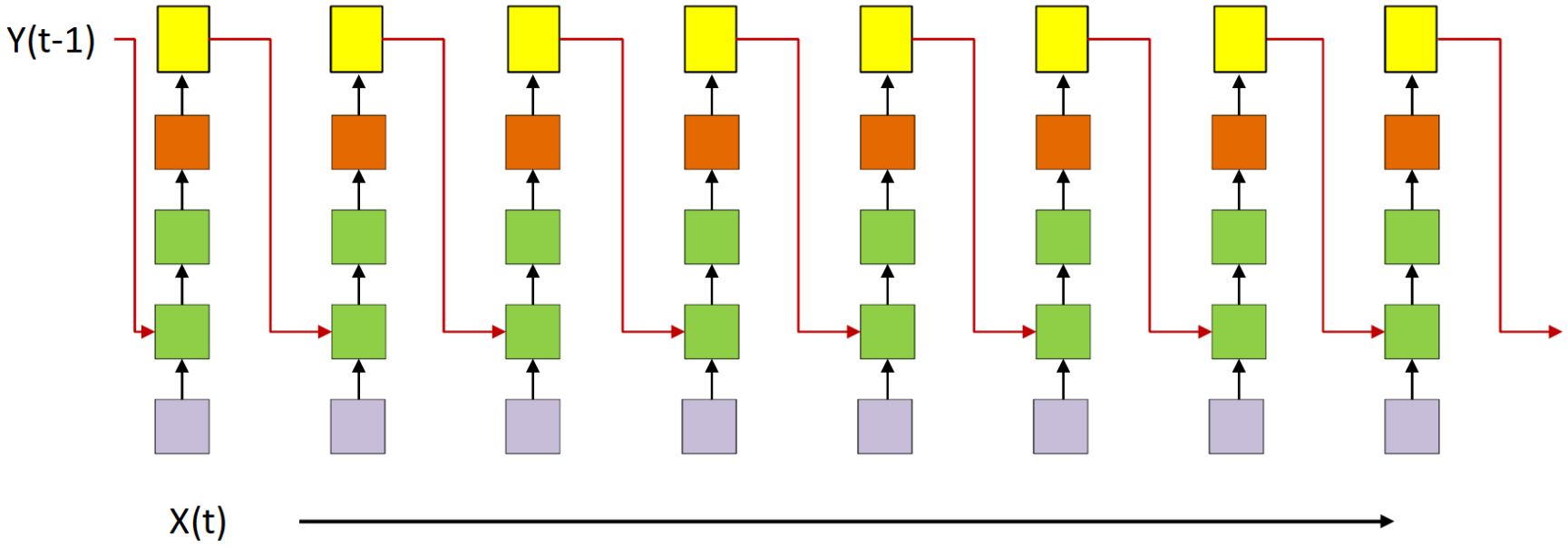
$$y_t = \sum_{j=1}^{r} \phi_j y_{t-j} + \sum_{j=1}^{r-1} \psi_j \epsilon_{t-j} + \epsilon_t$$



Source: Bhiksha Raj

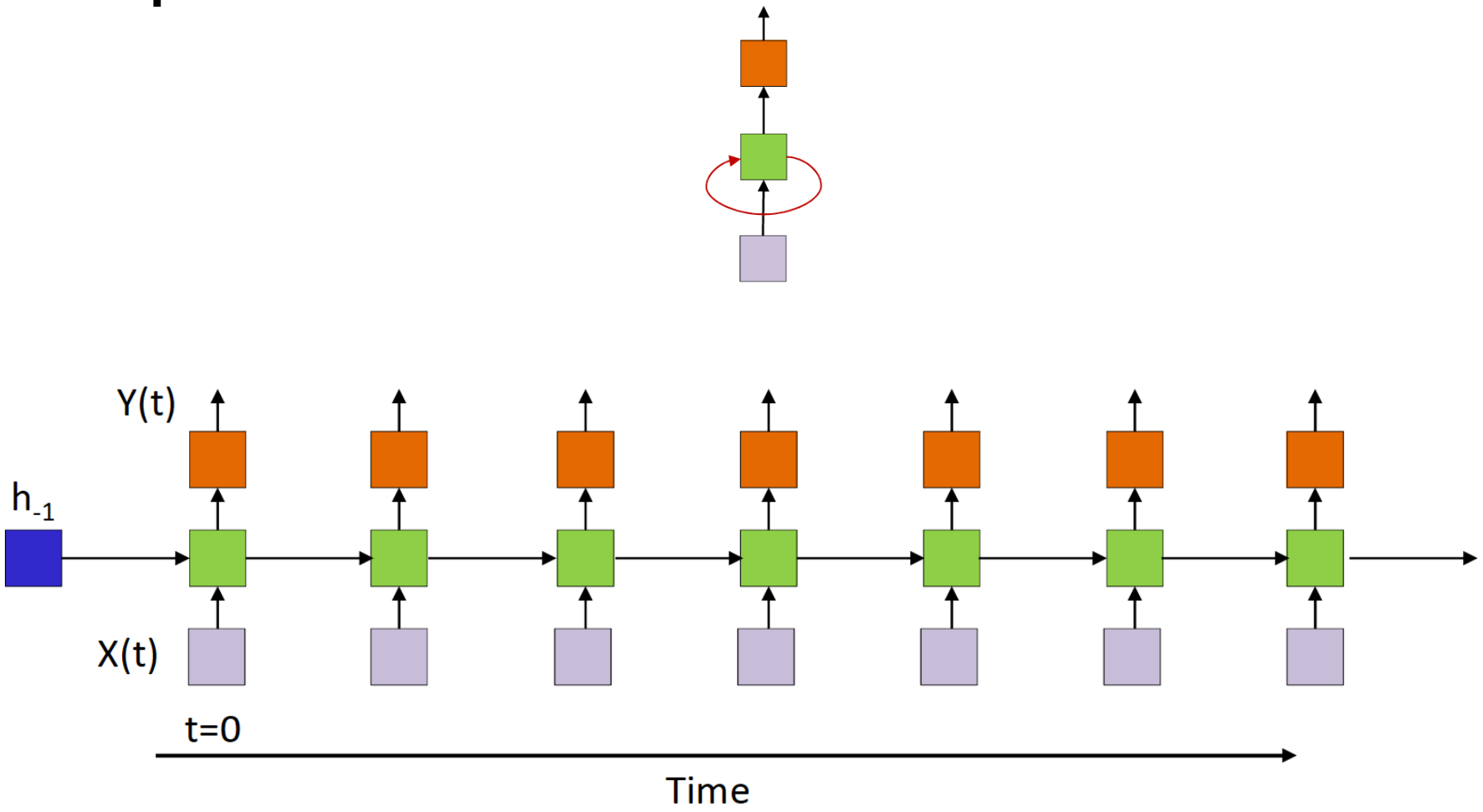
NARX networks

- The memory is in the output, not in the network
- The network itself does not remember its previous state



Source: Bhiksha Raj

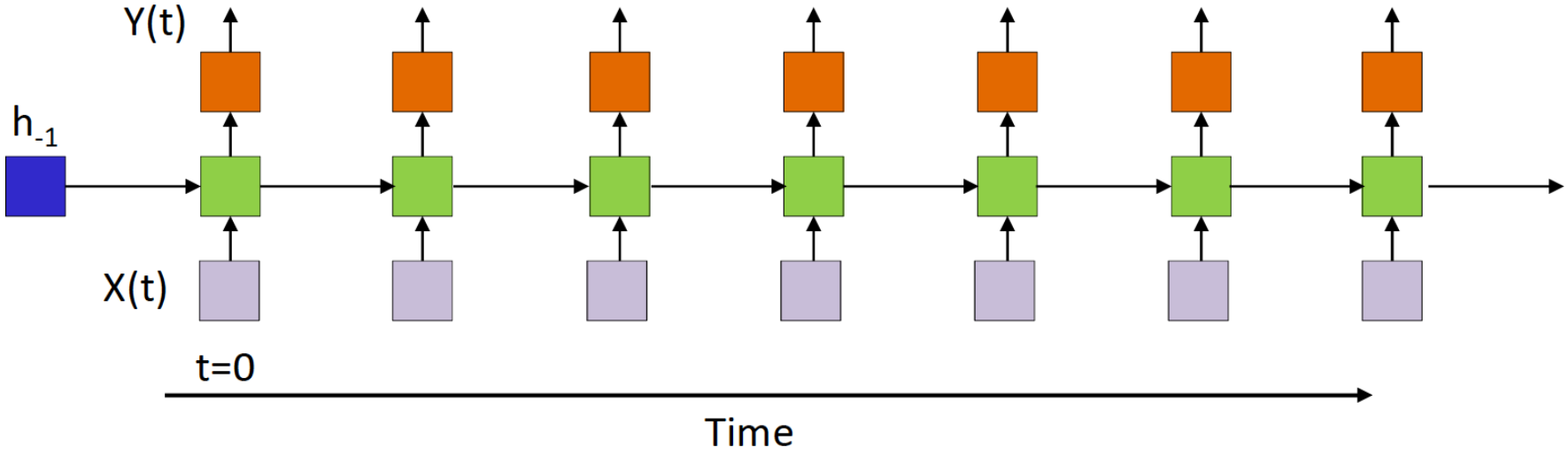
Simple RNN



Source: Bhiksha Raj

Simple RNN

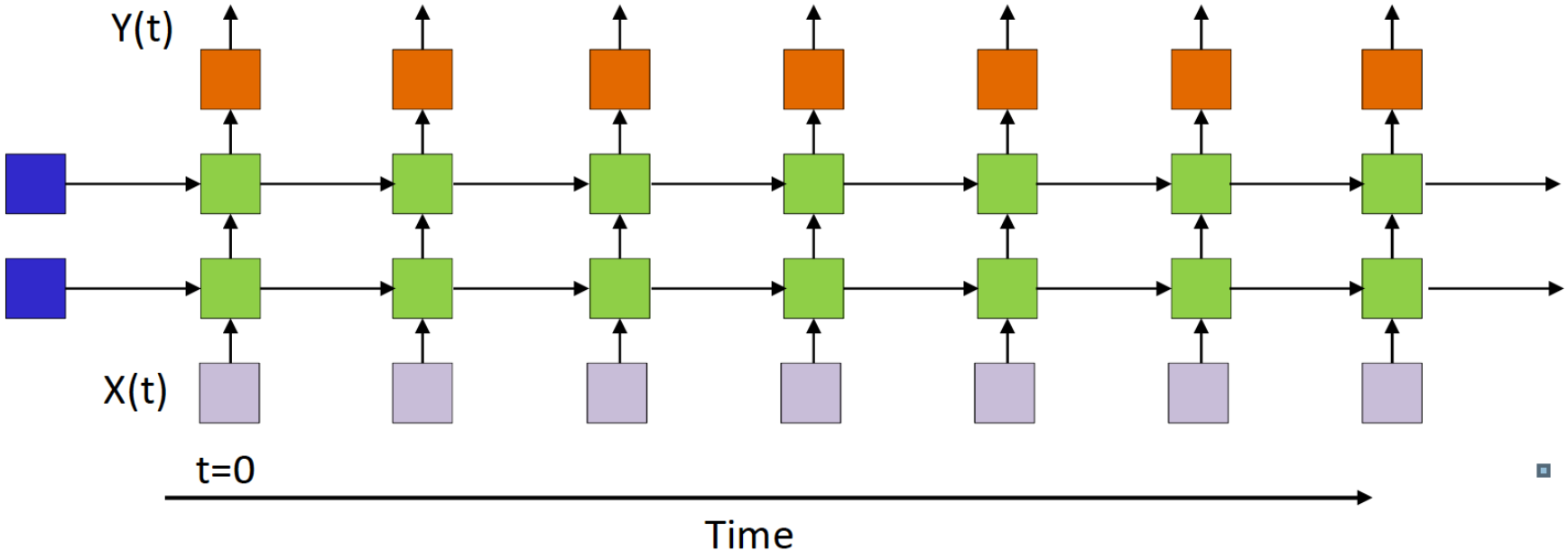
- An input at t=0 affects outputs forever
- Can capture long-term dependencies



Source: Bhiksha Raj

Multilayer RNN

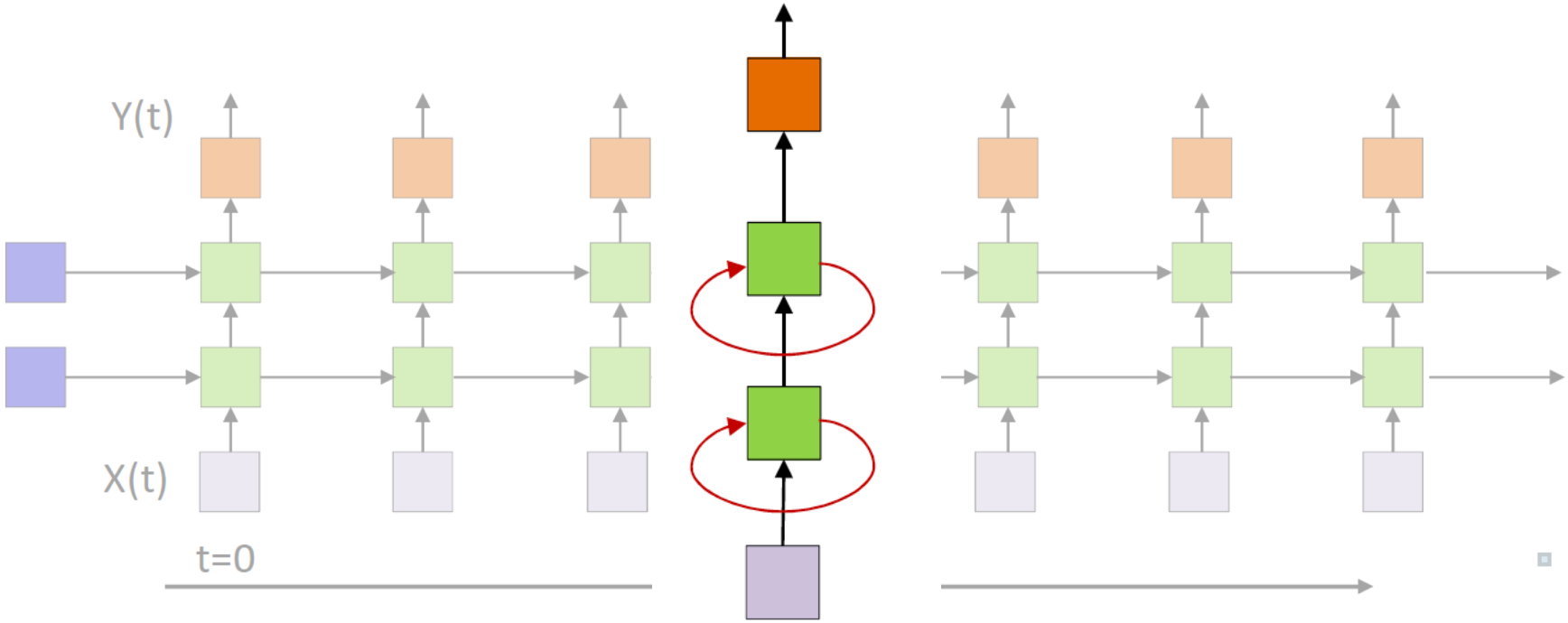
- Note: an input at t=0 affects outputs forever



Source: Bhiksha Raj

Multilayer RNN

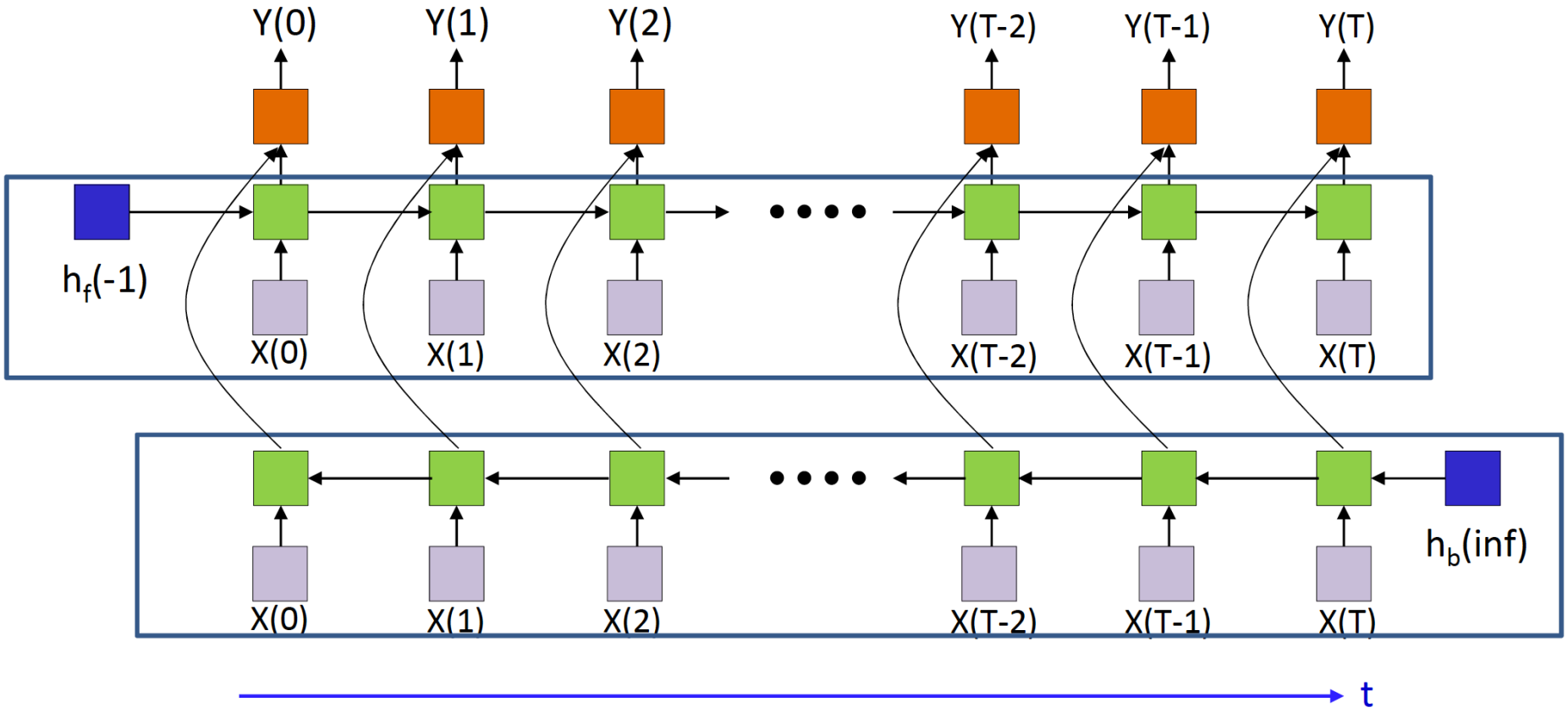
- Note: an input at t=0 affects outputs forever



Source: Bhiksha Raj

Bidirectional RNN

- Not a causal model



Source: Bhiksha Raj

Simple RNN

- RNNs are trained using *backpropagation through time*
  - Calculate loss
  - Optimize with respect to the weights
  - Change the weights all the way back to the first layer

- Suffer from the vanishing gradient problem
  - Gradients are very small by the time they get to the earlier layers
  - Their weights don't change
  - Model doesn't learn

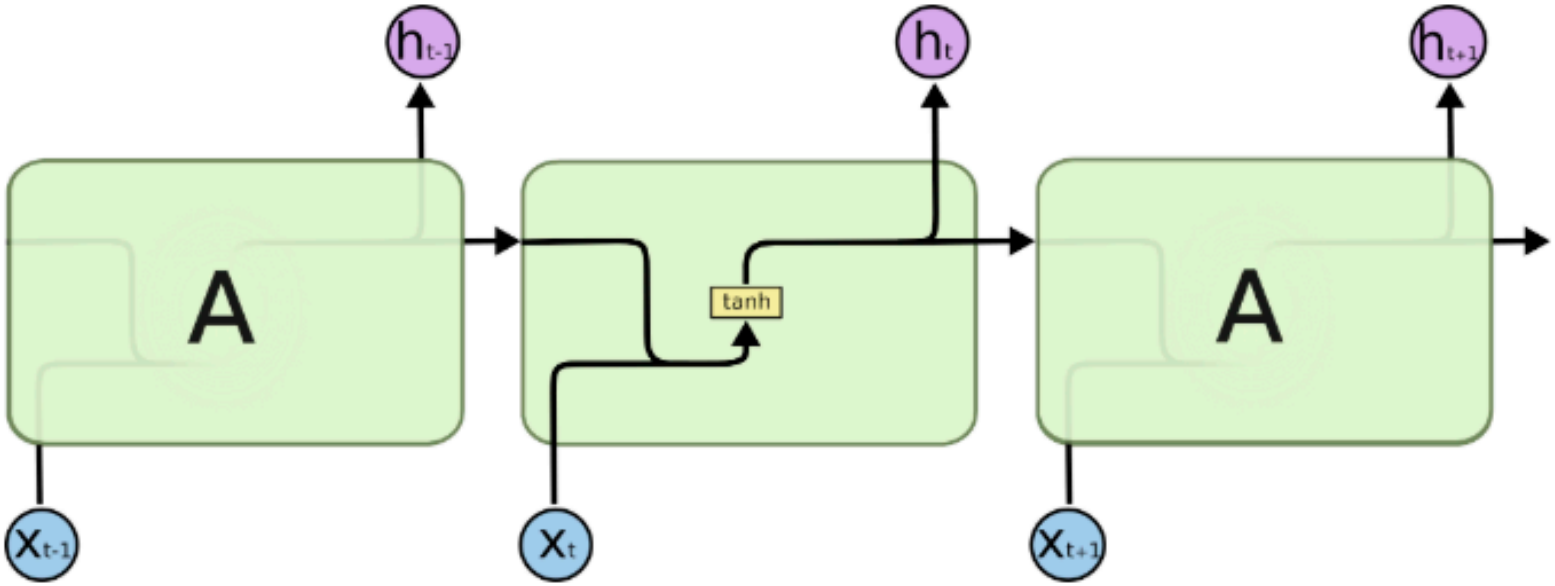
Simple RNN

- RNNs are trained using *backpropagation through time*
  - Calculate loss
  - Optimize with respect to the weights
  - Change the weights all the way back to the first layer

- Suffer from the vanishing gradient problem
  - Gradients are very small by the time they get to the earlier layers
  - Their weights don't change
  - Model doesn't learn

- "Gating models"
  - Long Short-Term Memory (LSTM)
  - Gated Recurrent Unit (GRU)

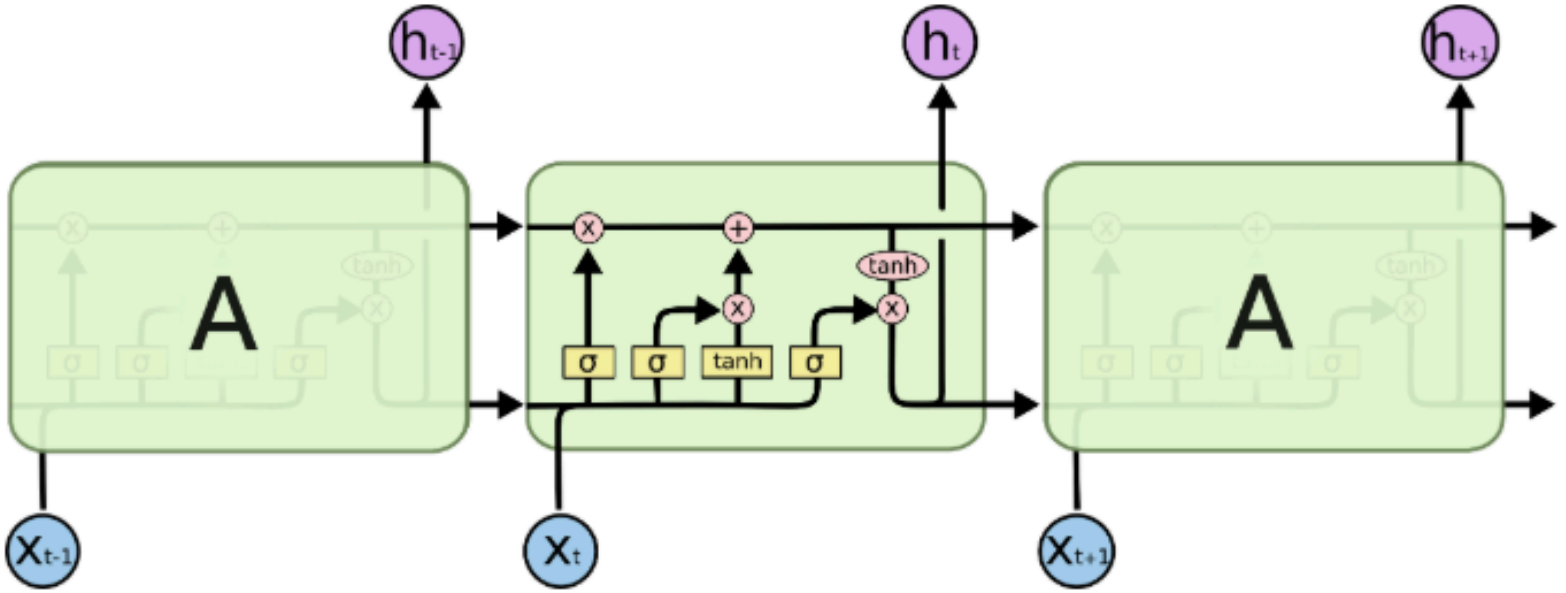
Simple RNN



http://colah.github.io/posts/2015-08-Understanding-LSTMs/

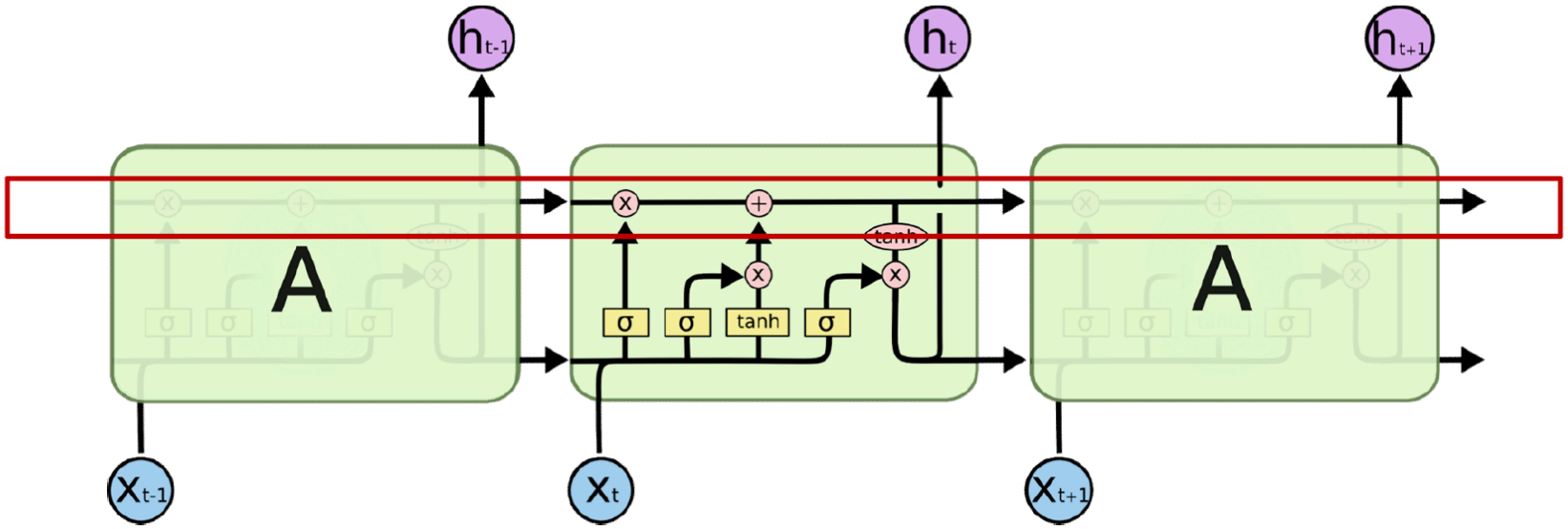
LSTM

- Learns to “forget” some of its history
- Learns to weight its input’s importance



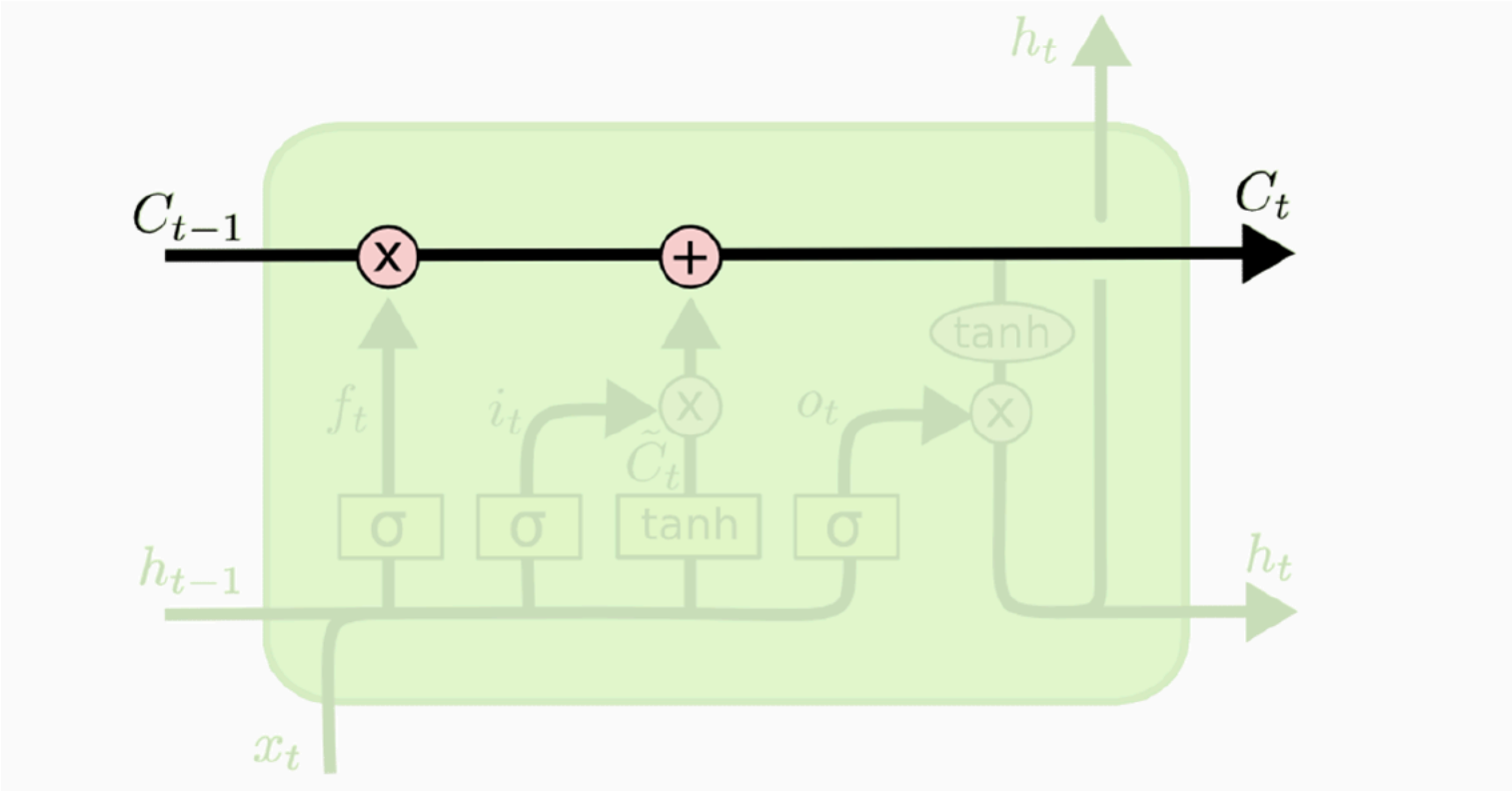
LSTM

- Learns to “forget” some of its history



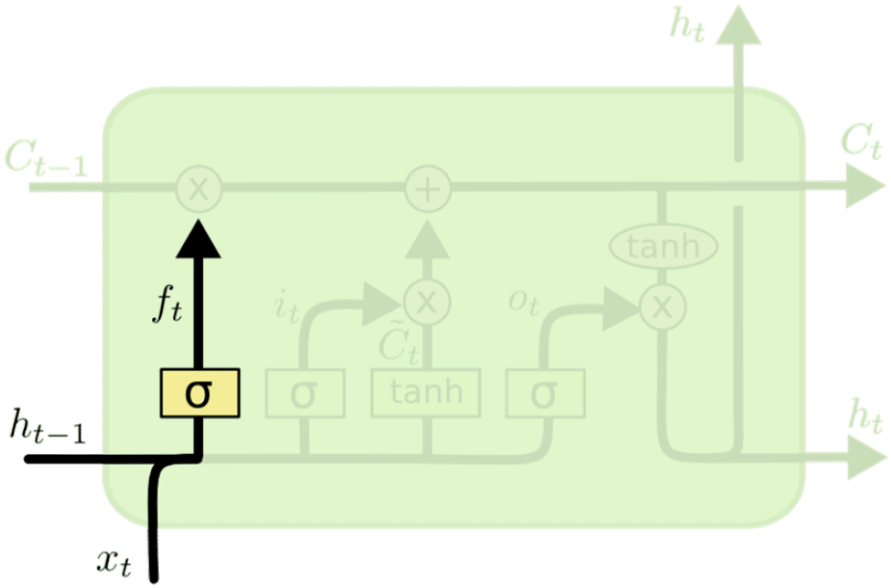
LSTM

- Cell state
- Acts like a conveyor belt, carrying the history of the network



LSTM

- Sigmoid layer with outputs in range (0,1)
- Decides what information it should ”forget”
  - Outputs a number in [0,1] for each number in the cell state

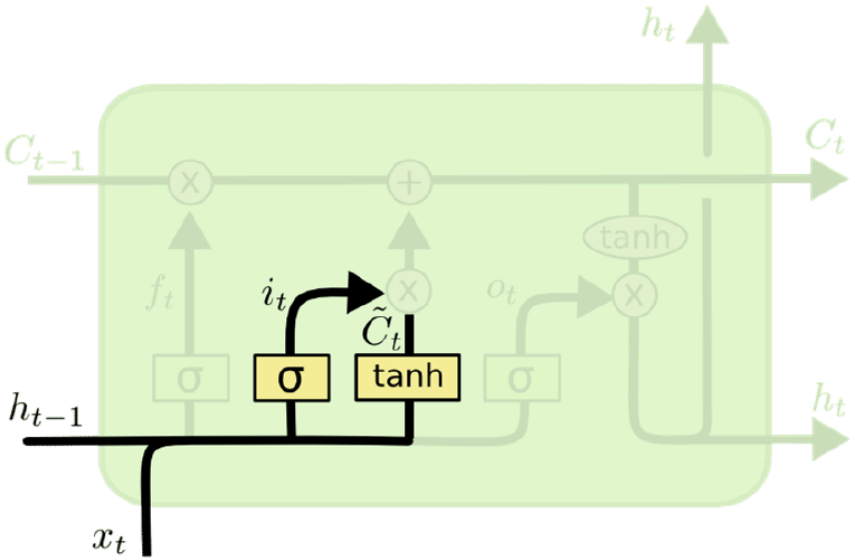


$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

Concatenate state and input

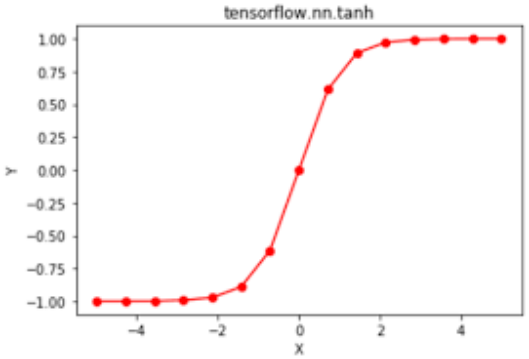
LSTM

- Input gate
- How much new information is there
- How much it should influence the old memory



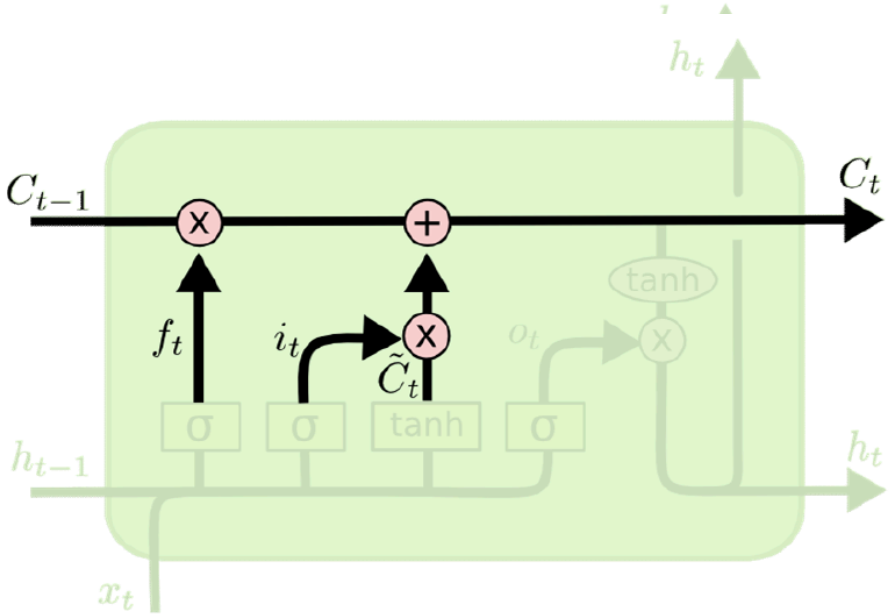
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$



LSTM

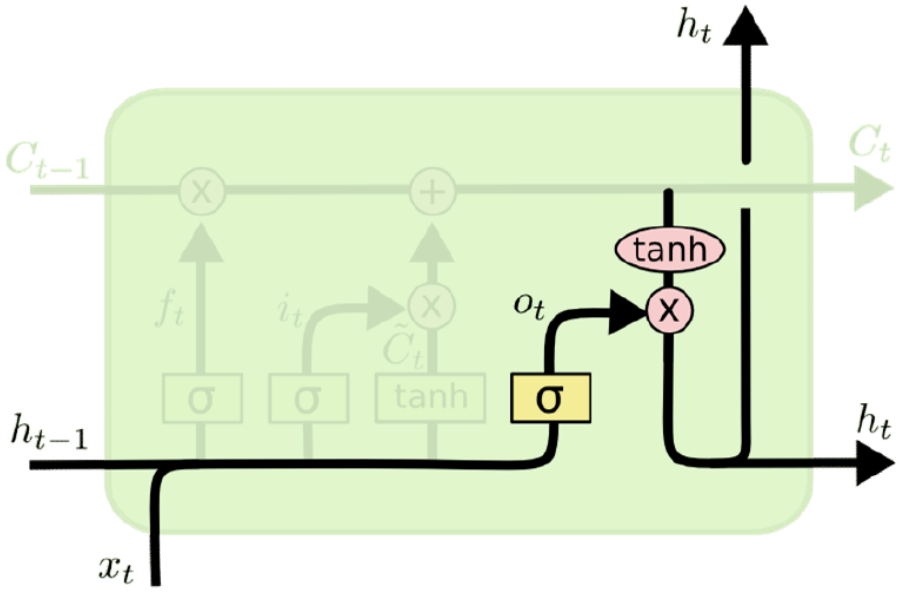
- Compute the new cell state
- i.e., update the memory



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

LSTM

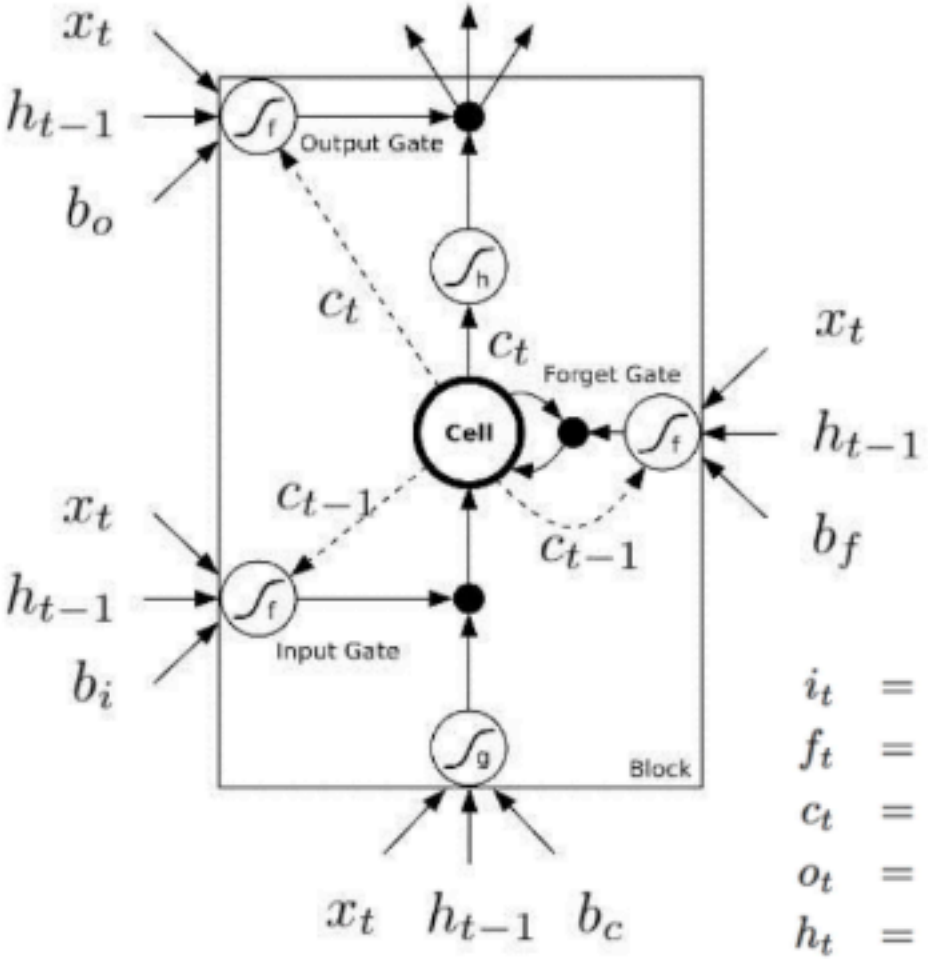
- Output gate: how much of the memory is worth reporting at this time



$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

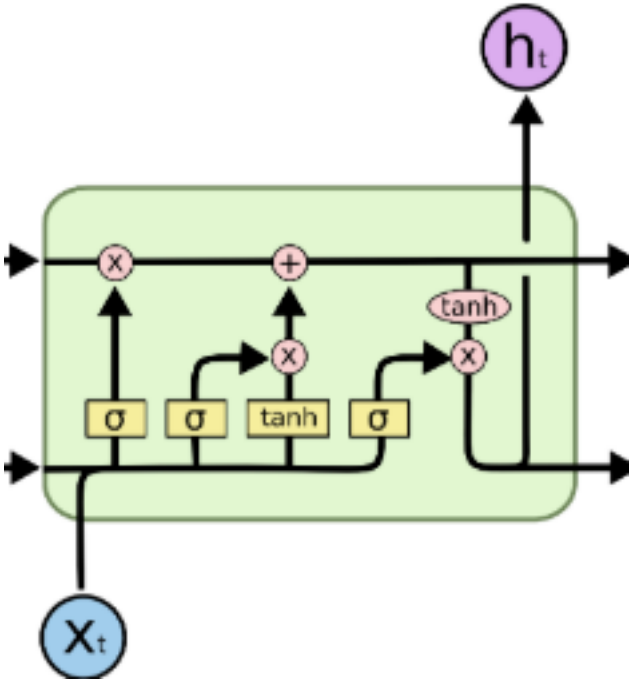
What you'll see in papers

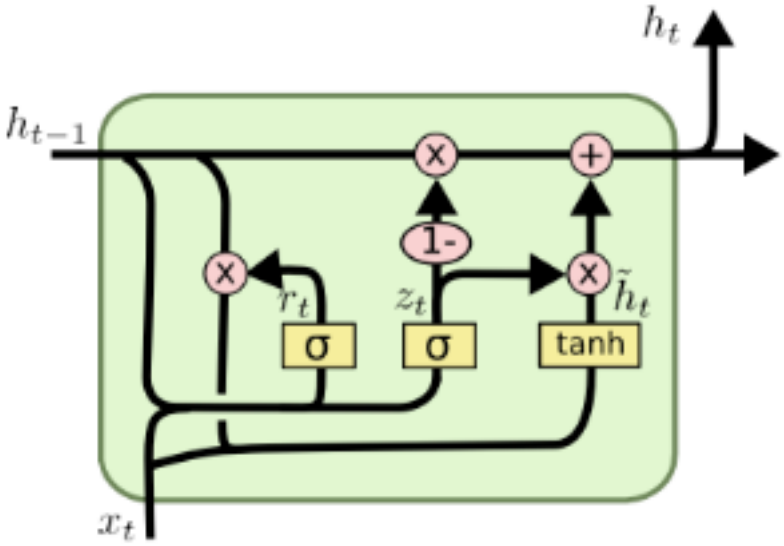


$$
\begin{aligned}
i_t &= \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i) \\
f_t &= \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f) \\
c_t &= f_t c_{t-1} + i_t tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \\
o_t &= \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o) \\
h_t &= o_t tanh(c_t)
\end{aligned}
$$

LSTM vs GRU

- Gated Recurrent Unit
- Simplified version of LSTM

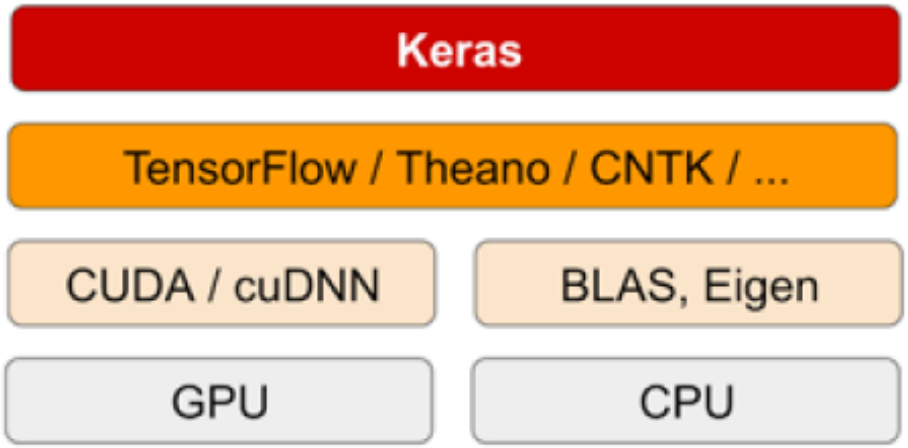




Applications of RNN

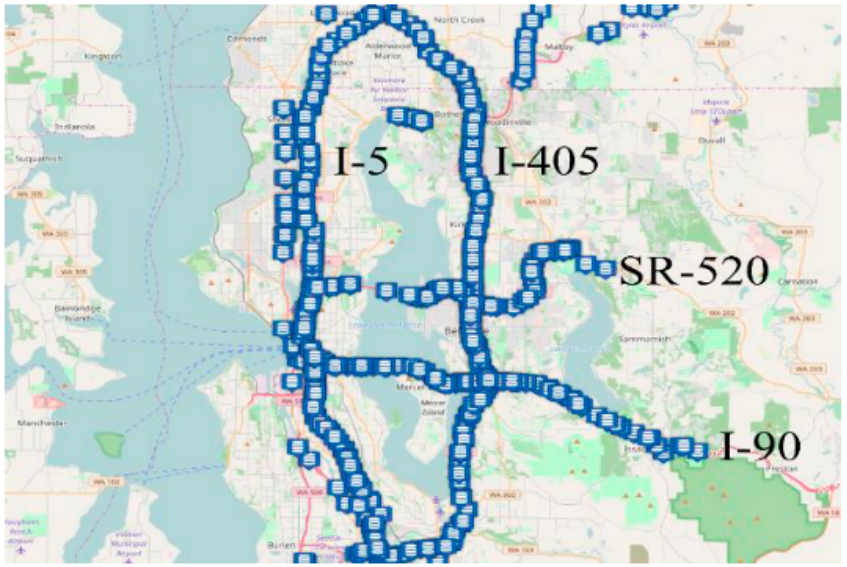
- Timeseries prediction
- Timeseries classification
  - Mode detection
  - Anomaly detection
- Sequence prediction

Lab 1



Traffic speed prediction with RNN

- 323 loop detectors
  - 4 freeways, 85 miles
  - Using last 50-min traffic speeds
  - Aggregated into 5-min intervals



Deep Stacked Bidirectional and Unidirectional LSTM Recurrent Neural Network for Network-wide Traffic Speed Prediction

Zhiyong Cui
University of Washington
Seattle, WA 98195, USA
zhiyongc@uw.edu

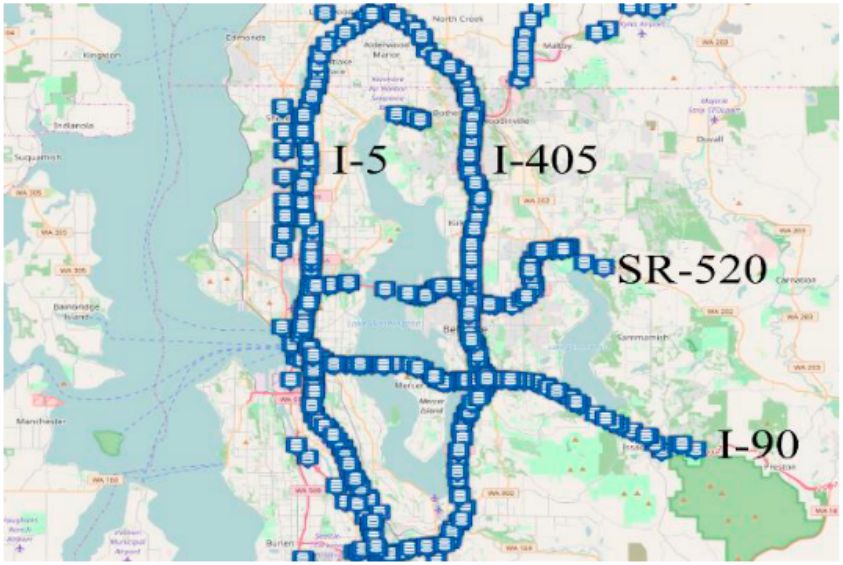
Ruimin Ke
University of Washington
Seattle, WA 98195, USA
ker27@uw.edu

Yinhai Wang*
University of Washington
Seattle, WA 98195, USA
yinhai@uw.edu

Traffic speed prediction with RNN

- 323 loop detectors

$$\boldsymbol{X}_T^P = \begin{bmatrix} \boldsymbol{x}^1 \\ \boldsymbol{x}^2 \\ \vdots \\ \boldsymbol{x}^P \end{bmatrix} = \begin{bmatrix} x_{T-n}^1 & x_{T-n+1}^1 & \cdots & x_{T-2}^1 & x_{T-1}^1 \\ x_{T-n}^2 & x_{T-n+1}^2 & & x_{T-2}^2 & x_{T-1}^2 \\ & \vdots & \ddots & \vdots & \vdots \\ x_{T-n}^P & x_{T-n+1}^P & \cdots & x_{T-2}^P & x_{T-1}^P \end{bmatrix}$$

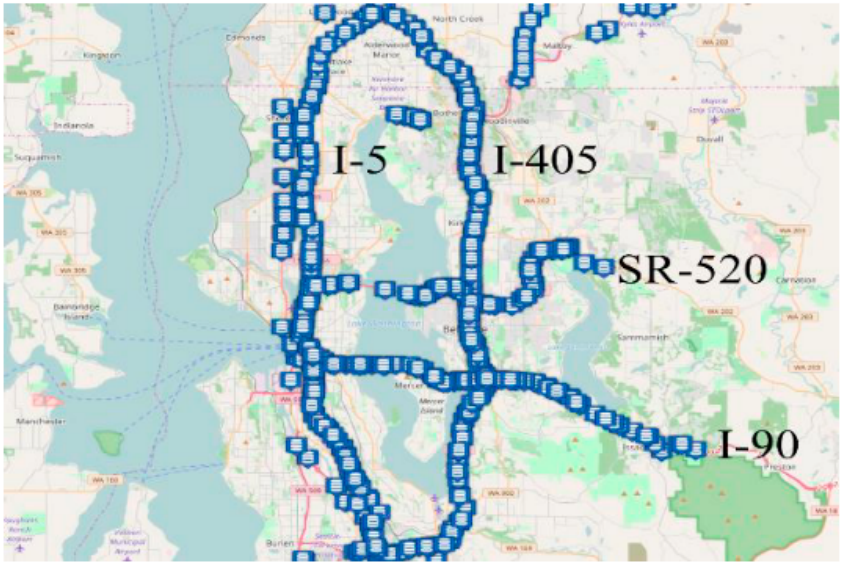


Traffic speed prediction with RNN

Feature engineering is done manually!

- 323 loop detectors

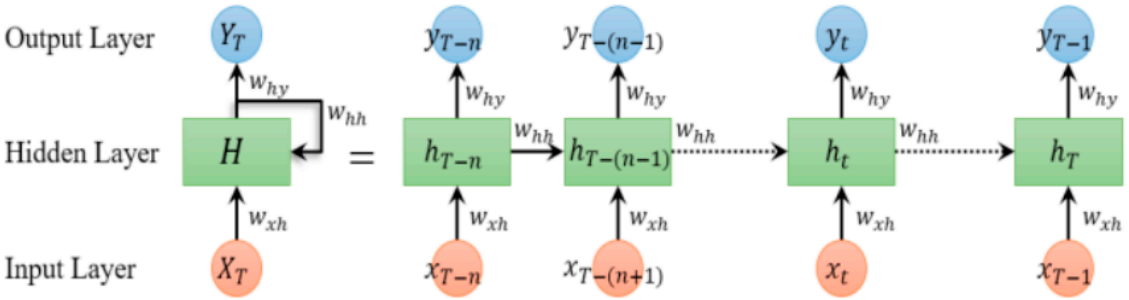
$$\boldsymbol{X}_T^P = \begin{bmatrix} \boldsymbol{x}^1 \\ \boldsymbol{x}^2 \\ \vdots \\ \boldsymbol{x}^P \end{bmatrix} = \begin{bmatrix} x_{T-n}^1 & x_{T-n+1}^1 & \cdots & x_{T-2}^1 & x_{T-1}^1 \\ x_{T-n}^2 & x_{T-n+1}^2 & & x_{T-2}^2 & x_{T-1}^2 \\ & \vdots & \ddots & \vdots & \vdots \\ x_{T-n}^P & x_{T-n+1}^P & \cdots & x_{T-2}^P & x_{T-1}^P \end{bmatrix}$$



Traffic speed prediction with RNN

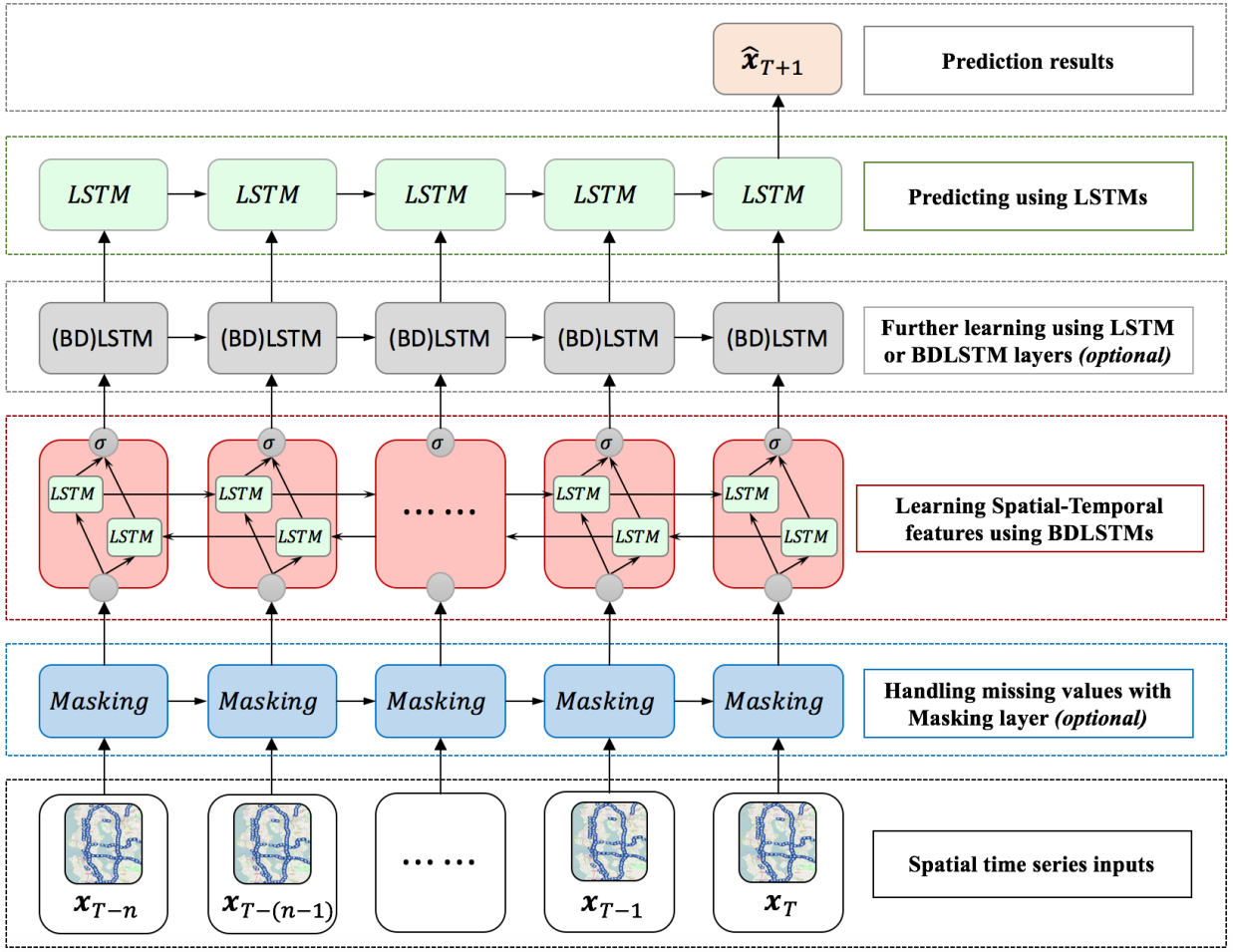
- 323 loop detectors

$$\boldsymbol{X}_T^P = \begin{bmatrix} \boldsymbol{x}^1 \\ \boldsymbol{x}^2 \\ \vdots \\ \boldsymbol{x}^P \end{bmatrix} = \begin{bmatrix} x_{T-n}^1 & x_{T-n+1}^1 & \cdots & x_{T-2}^1 & x_{T-1}^1 \\ x_{T-n}^2 & x_{T-n+1}^2 & & x_{T-2}^2 & x_{T-1}^2 \\ & \vdots & \ddots & \vdots & \vdots \\ x_{T-n}^P & x_{T-n+1}^P & \cdots & x_{T-2}^P & x_{T-1}^P \end{bmatrix}$$



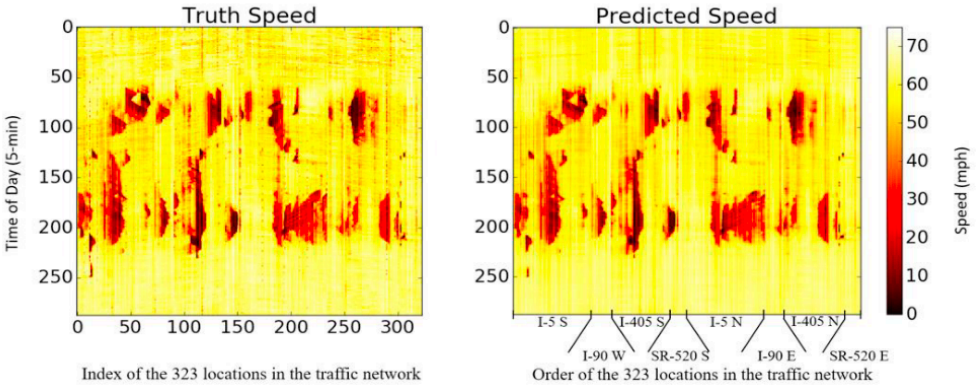
Traffic speed prediction with RNN

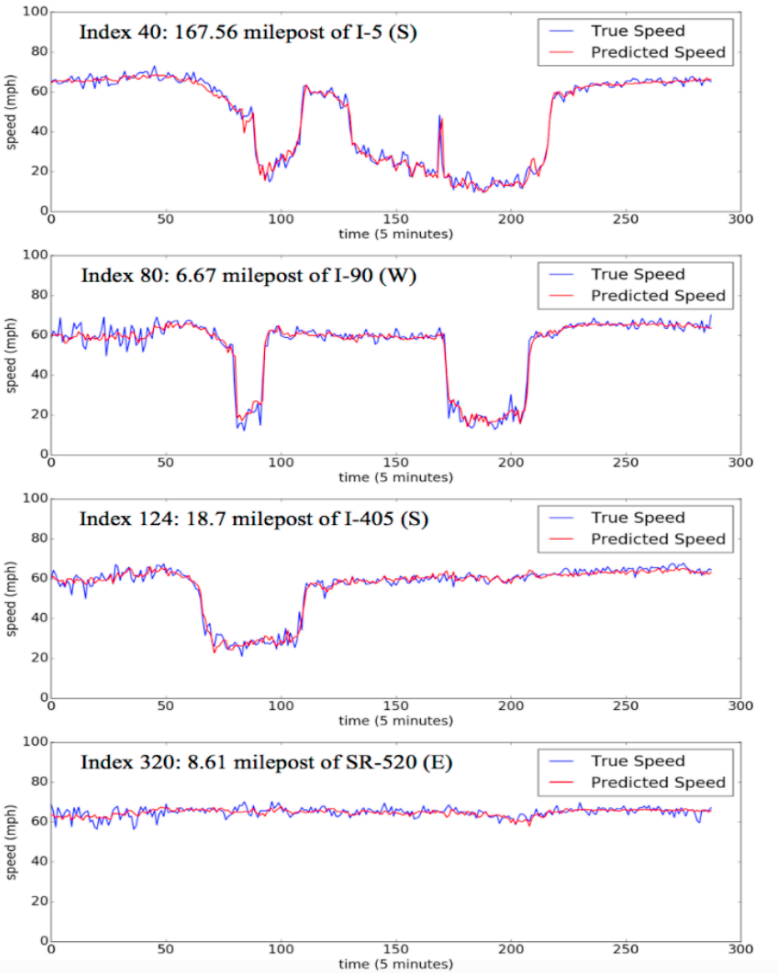
- 323 loop detectors



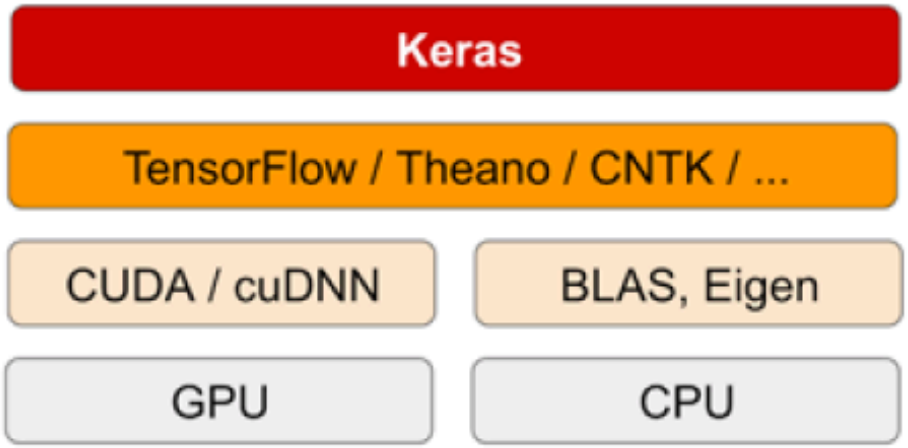
Traffic speed prediction with RNN

- 323 loop detectors





Lab 2



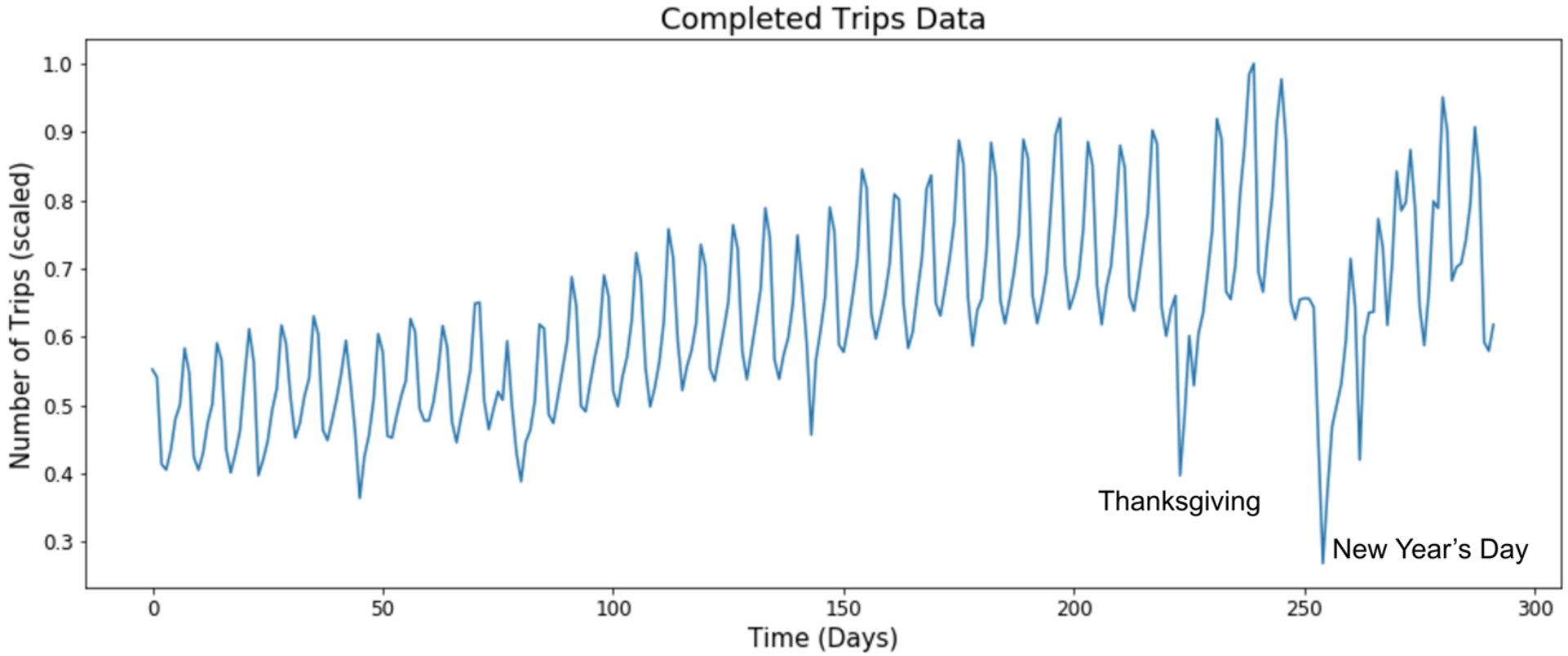
Uber Data

Engineering Extreme Event Forecasting at Uber with Recurrent Neural Networks

Problem 1: Event forecasting at UBER

- Extreme events—peak travel times such as holidays, concerts, inclement weather, and sporting events.
- Even though they have plethora of data, it is sparse for new cities and special events
- Calculating demand time series forecasting during extreme events is a critical component of anomaly detection, optimal resource allocation, and budgeting.

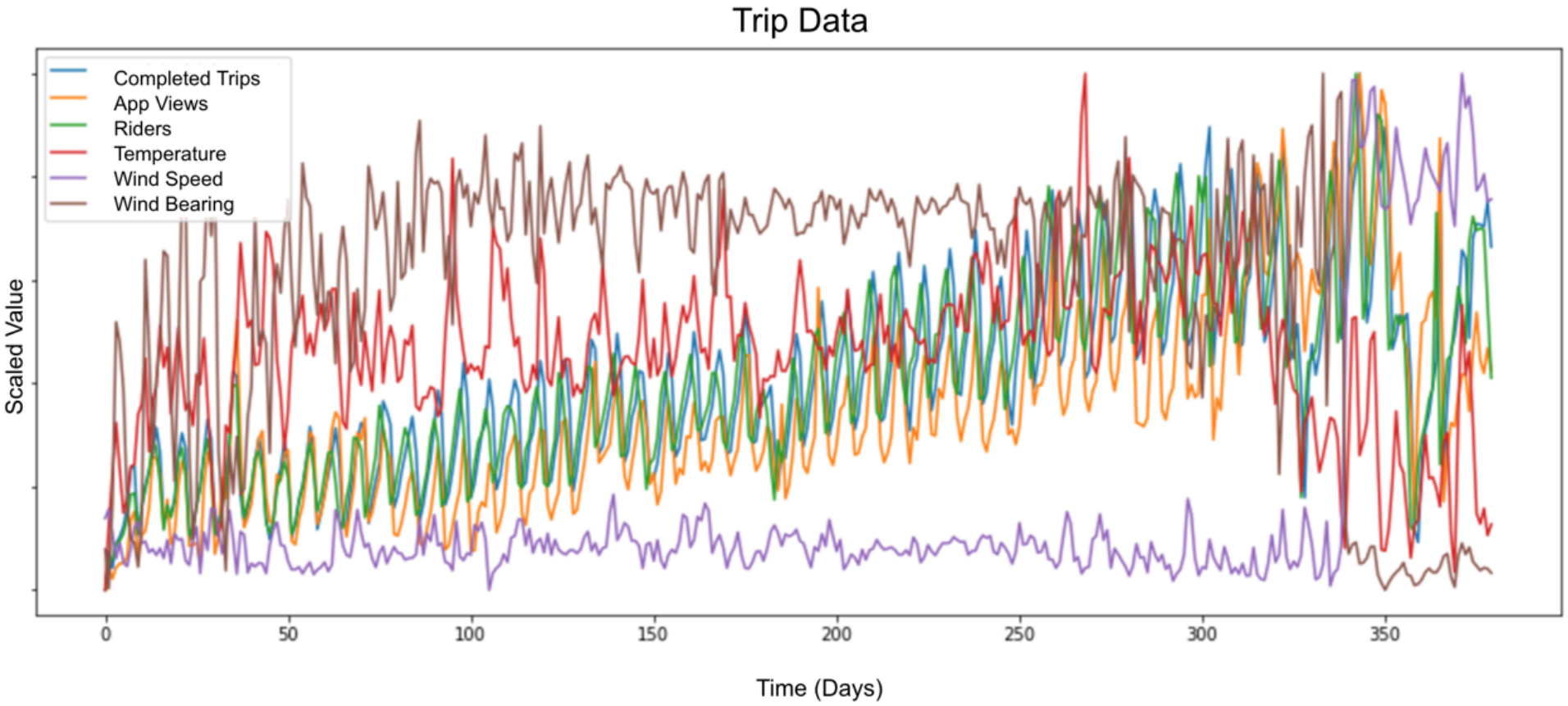
Event forecasting at UBER



Problem 2: Forecasting at UBER

- Uber tracks thousands of metrics per time interval
- It's impractical to train one LSTM model per metric
- Simple LSTM failed miserably
- Come up with a generic architecture that can be used for any of the metrics

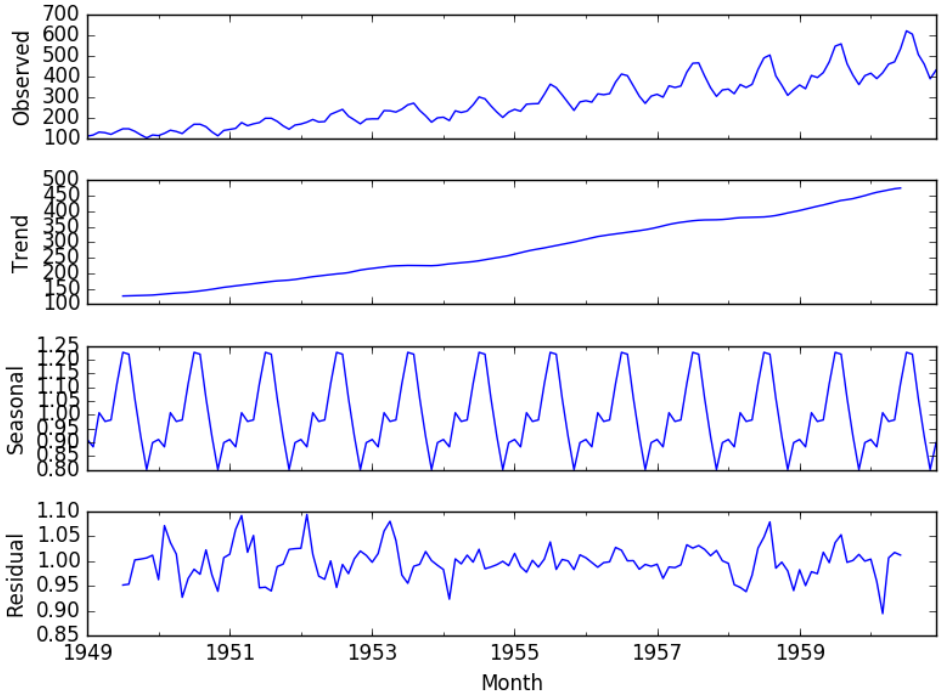
Problem 2: Forecasting at UBER



Problem 2: Forecasting at UBER

- Classical feature engineering approach

Feature	Description
Mean	Mean.
Var	Variance.
ACF1	First order of autocorrelation.
Trend	Strength of trend.
Linearity	Strength of linearity.
Curvature	Strength of curvature
Season	Strength of seasonality.
Peak	Strength of peaks.
Trough	Strength of trough.
Entropy	Spectral entropy.
Lumpiness	Changing variance in remainder.
Spikiness	Strength of spikiness
Lshift	Level shift using rolling window.
Vchange	Variance change.
Fspots	Flat spots using disretization.
Cpoints	The number of crossing points.
KLscore	Kullback-Leibler score.
Change.idx	Index of the maximum KL score.

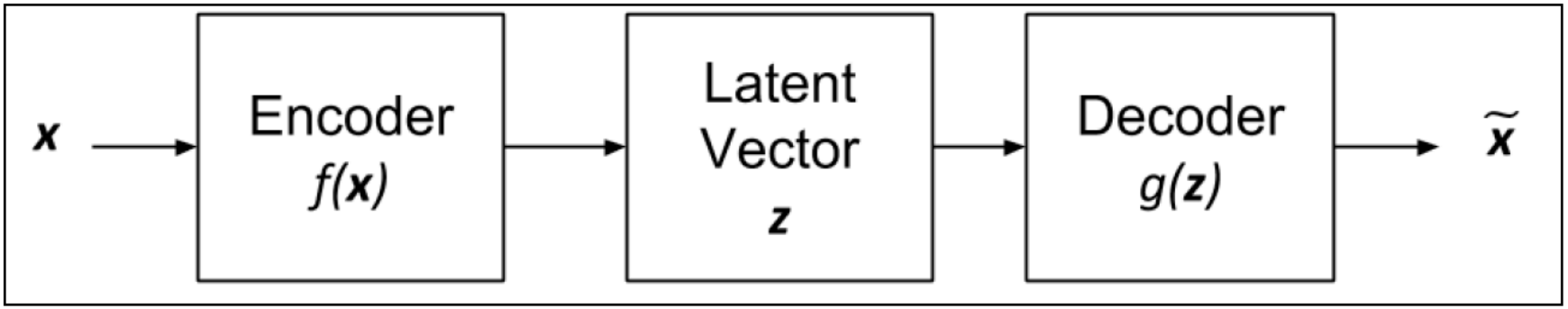


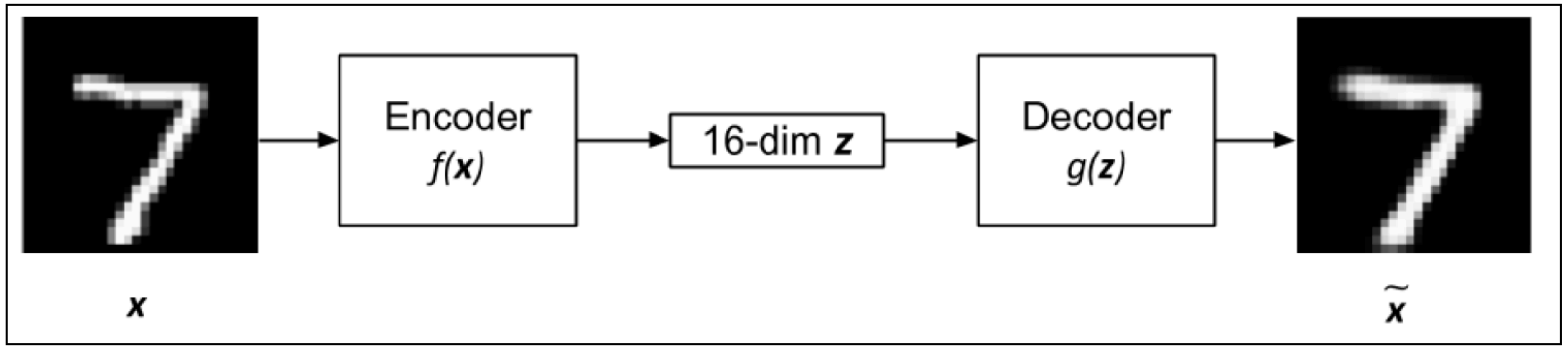
Problem 2: Forecasting at UBER

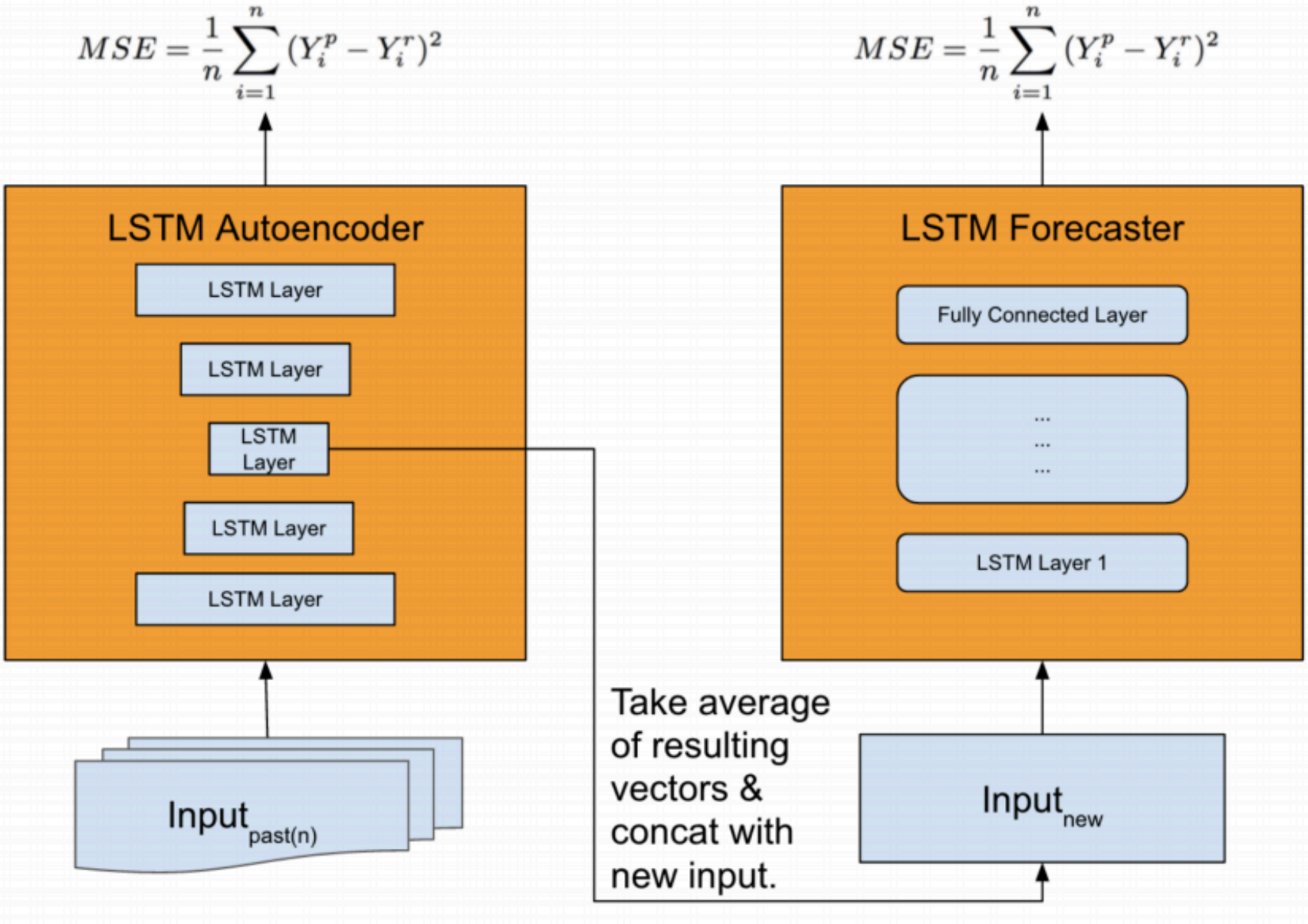
- Uber tracks thousands of metrics per time interval
- Come up with a generic architecture that can be used for any of the metrics

Problem 2: Forecasting at UBER

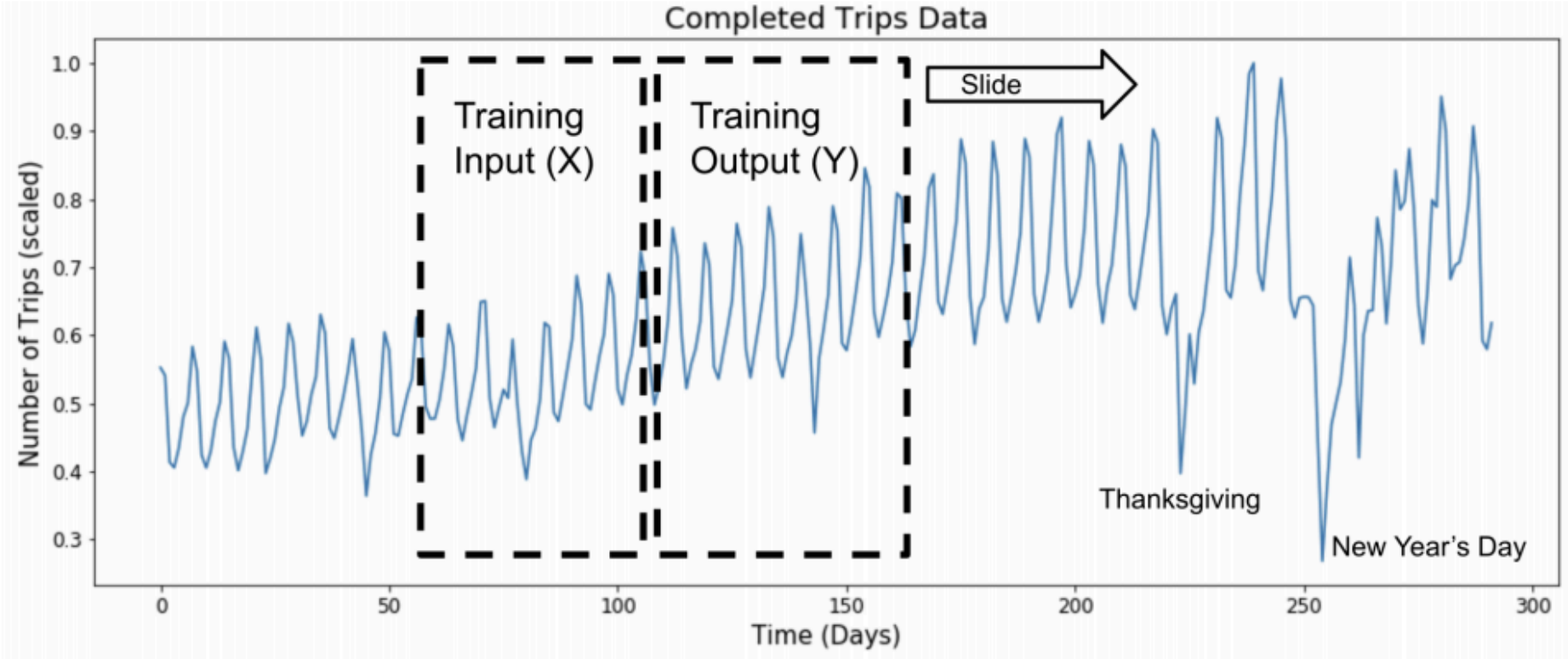
- Uber tracks thousands of metrics per time interval






$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i^p - Y_i^r)^2$
$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i^p - Y_i^r)^2$
LSTM Autoencoder
LSTM Layer
LSTM Layer
LSTM Layer
LSTM Layer
LSTM Layer
LSTM Forecaster
Fully Connected Layer
...
...
...
LSTM Layer 1
Input past(n)
Take average of resulting vectors & concat with new input.
Input new

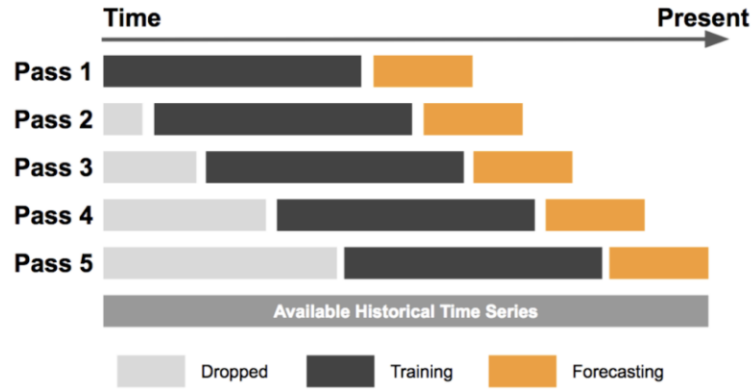
Event forecasting at UBER



Event forecasting at UBER

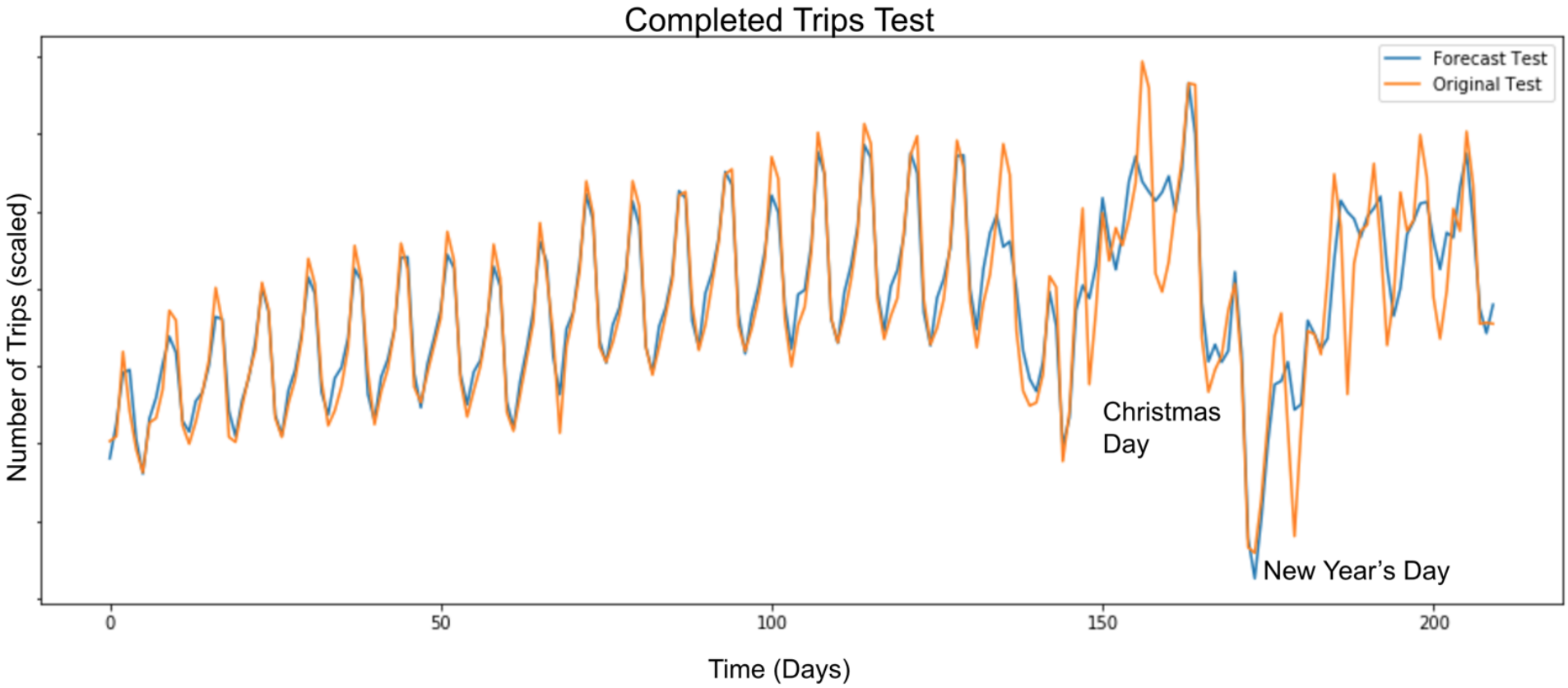
- Other options

Sliding Window



Expanding Window

Event forecasting at UBER



Event forecasting at UBER

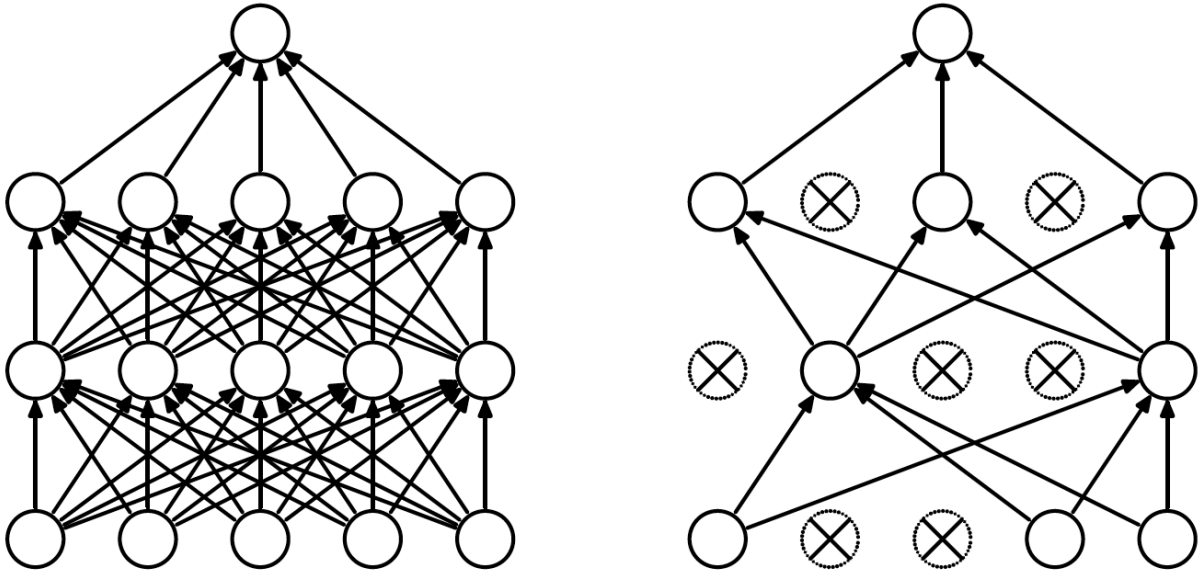
- Achieved a 14.09 percent symmetric mean absolute percentage error (SMAPE) improvement over the base LSTM architecture.

- 25 percent improvement over the classical time series models.

Holiday	Described Neural Network Architecture Error	Previous Model Error
Christmas Day	**11.1**	29.2
Martin Luther King, Jr. Day	**8.7**	20.2
Independence Day	**2.8**	17.6
Labor Day	**2.9**	6.9
New Year's Day	**6.8**	7.8
Veteran's Day	**4.7**	8.9

Predicting forecasting Uncertainty at UBER

- Remember Dropout?
- Used during **training** to avoid overfitting



(a) Standard Neural Net

(b) After applying dropout.

Predicting forecasting Uncertainty at UBER

- Turns out, if applied during testing, it has a Bayesian interpretation and can be used for estimating the prediction uncertainty

---

Dropout as a Bayesian Approximation:
Representing Model Uncertainty in Deep Learning

Yarin Gal YG279@CAM.AC.UK
Zoubin Ghahramani ZG201@CAM.AC.UK
University of Cambridge

Predicting forecasting Uncertainty at UBER

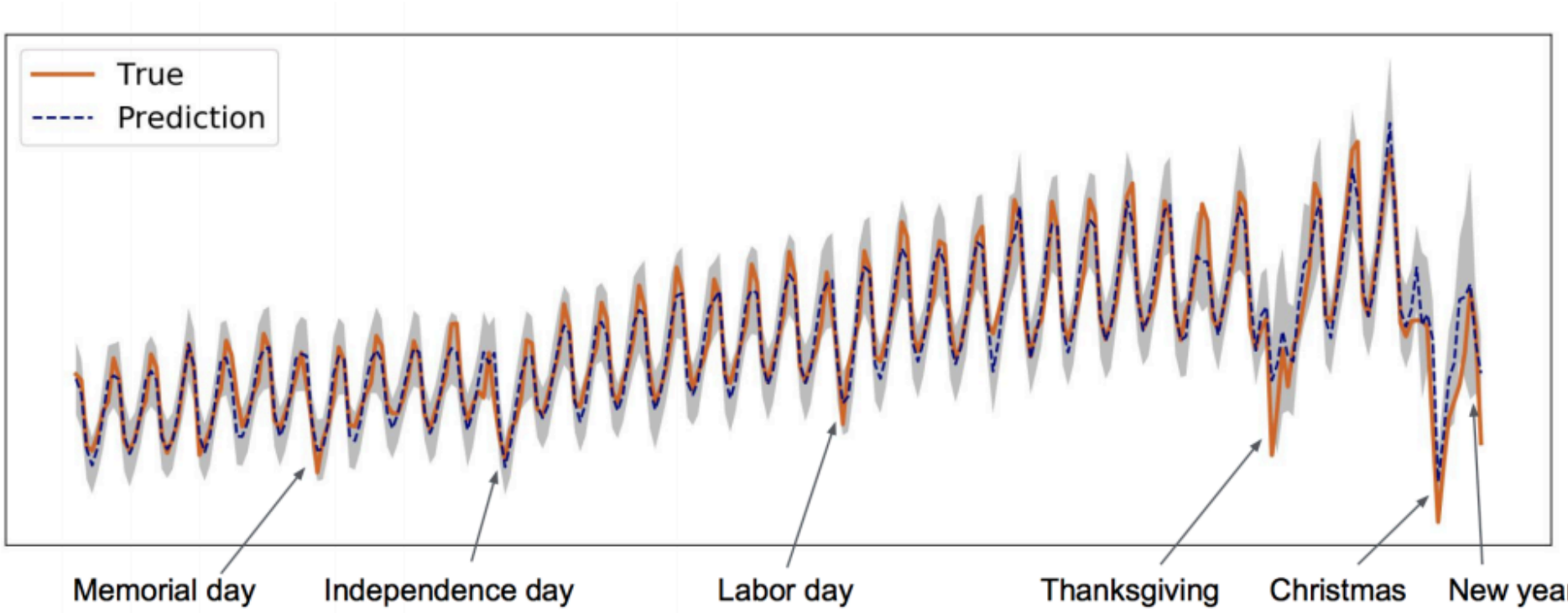
- Turns out, if applied during testing, it has a Bayesian interpretation and can be used for estimating the prediction uncertainty

Listing 1. Practical implementation of estimating the uncertainty bound

```
vals = []
for r in range(100):
  vals.append(model.eval(input,
                         dropout = random(0,1)))
mean = np.mean(vals)
var = np.var(vals)
```

Predicting forecasting Uncertainty at UBER

- Prediction uncertainty intervals



Summary so far

- CNN is the perfect tool for capturing spatial dependencies
- RNN is the perfect tool for capturing temporal dependencies
  - LSTM
  - GRU

CNN RNN for spatiotemporal data

Modeling spatiotemporal data

- Use CNN in the initial layers to capture spatial dependencies
    - Feature extraction

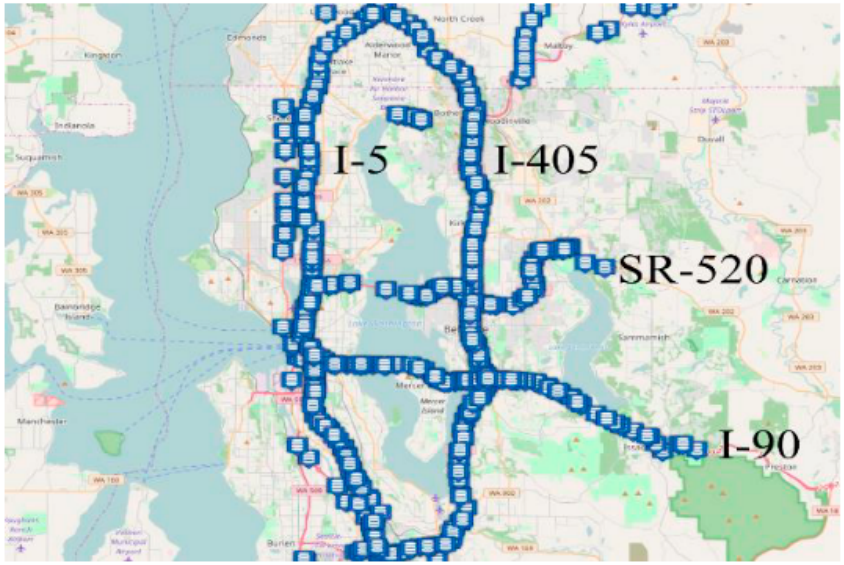
- Use RNN on features extracted by CNN

Traffic speed prediction with RNN

Feature engineering is done manually!

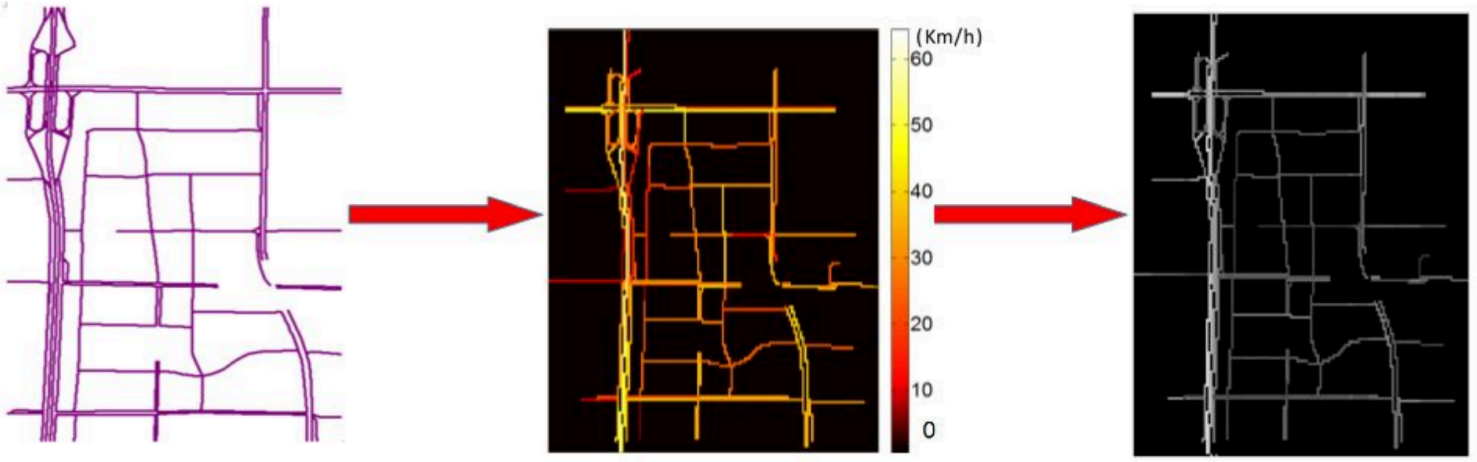
- 323 loop detectors

$$\boldsymbol{X}_T^P = \begin{bmatrix} \boldsymbol{x}^1 \\ \boldsymbol{x}^2 \\ \vdots \\ \boldsymbol{x}^P \end{bmatrix} = \begin{bmatrix} x_{T-n}^1 & x_{T-n+1}^1 & \cdots & x_{T-2}^1 & x_{T-1}^1 \\ x_{T-n}^2 & x_{T-n+1}^2 & & x_{T-2}^2 & x_{T-1}^2 \\ & \vdots & \ddots & \vdots & \vdots \\ x_{T-n}^P & x_{T-n+1}^P & \cdots & x_{T-2}^P & x_{T-1}^P \end{bmatrix}$$



Revisiting our example

- Traffic Prediction in Transportation Networks



Network

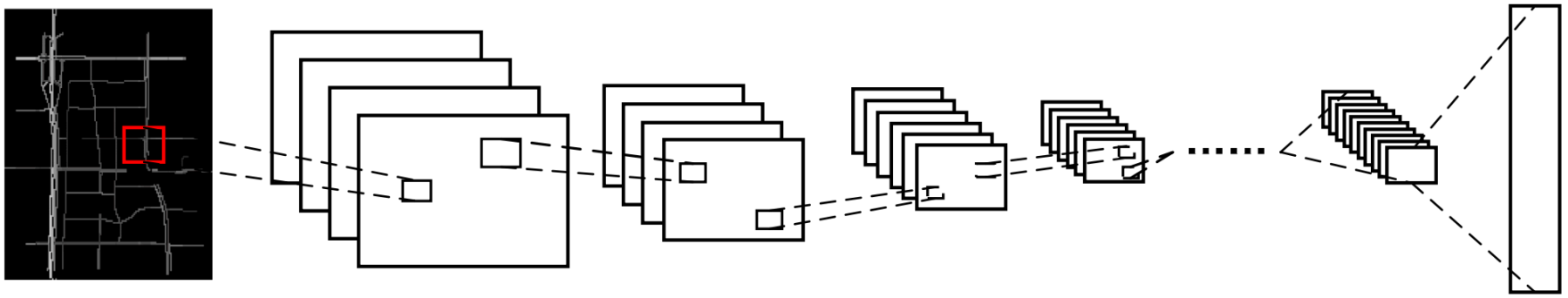
Speeds

Normalized Speeds

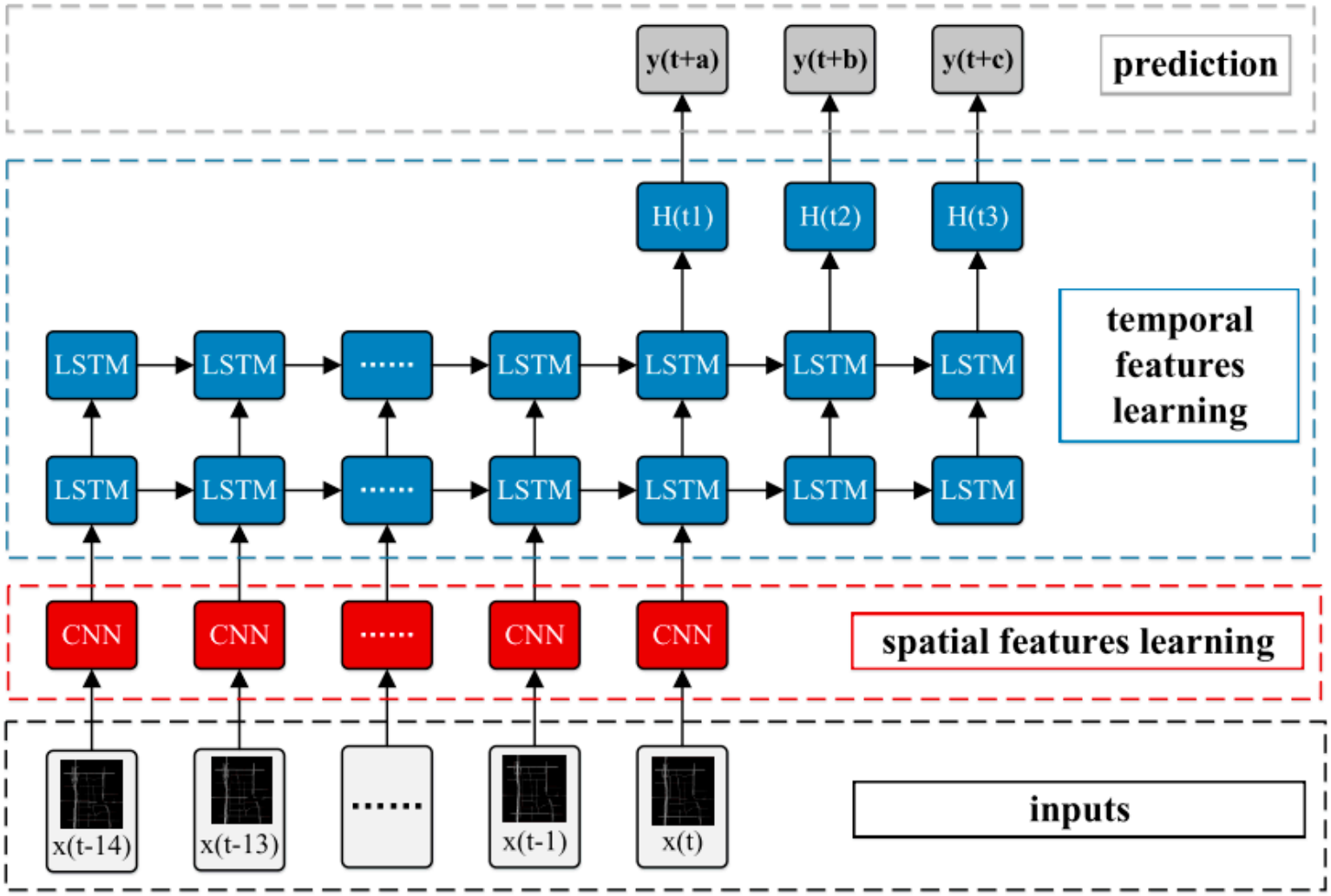
"Spatiotemporal recurrent convolutional networks for traffic prediction in transportation networks."
Sensors 17, no. 7 (2017)

Traffic Prediction in Transportation Networks

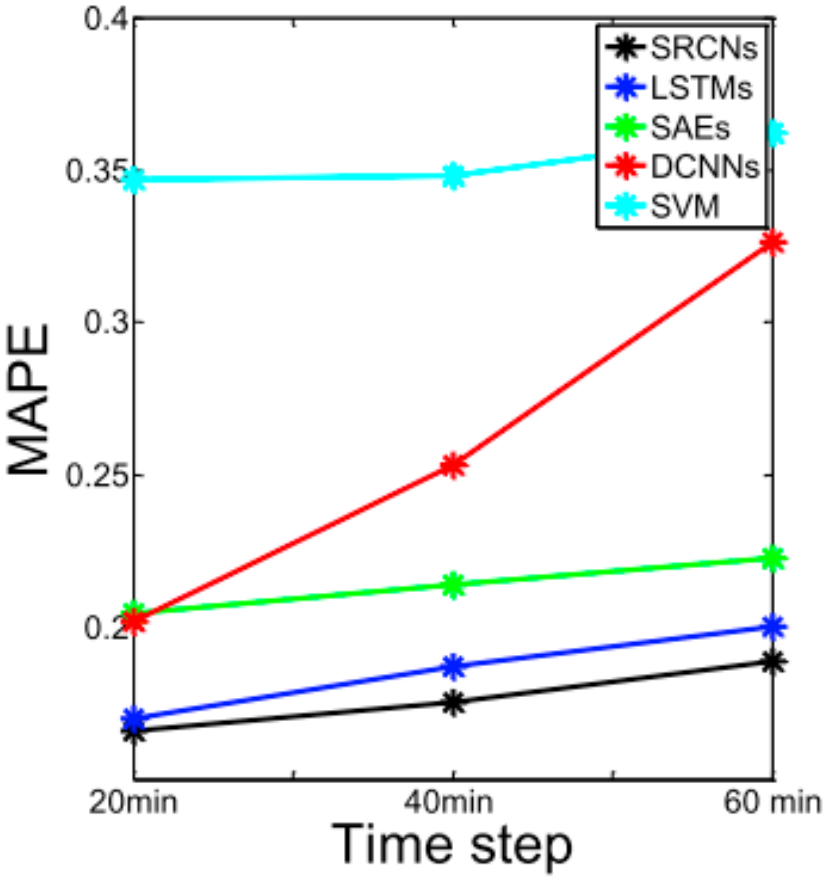
- Traffic Prediction in Transportation Networks

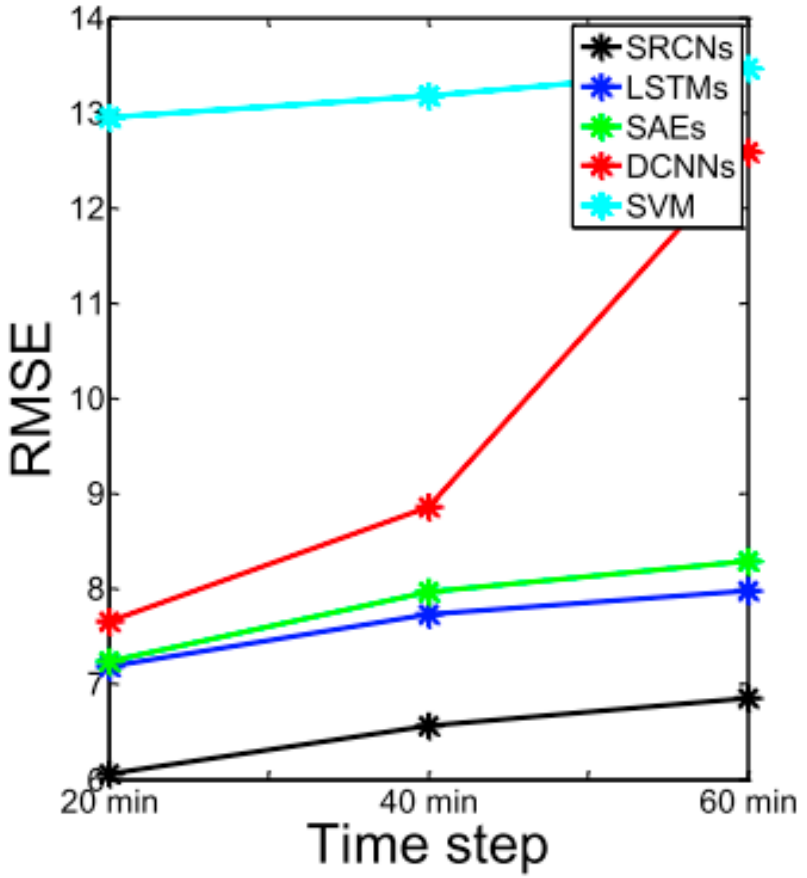


Traffic Prediction in Transportation Networks

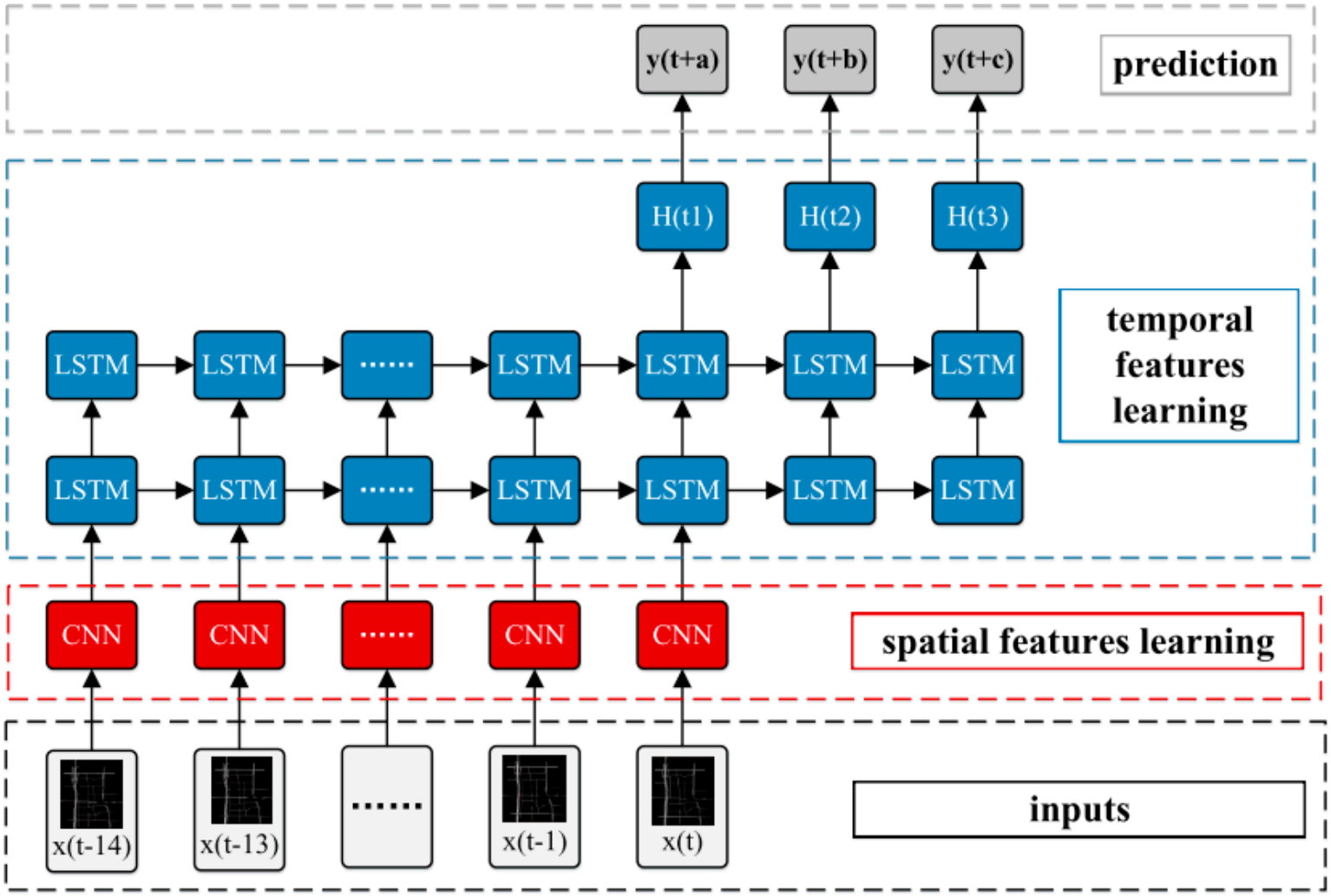


Traffic Prediction in Transportation Networks

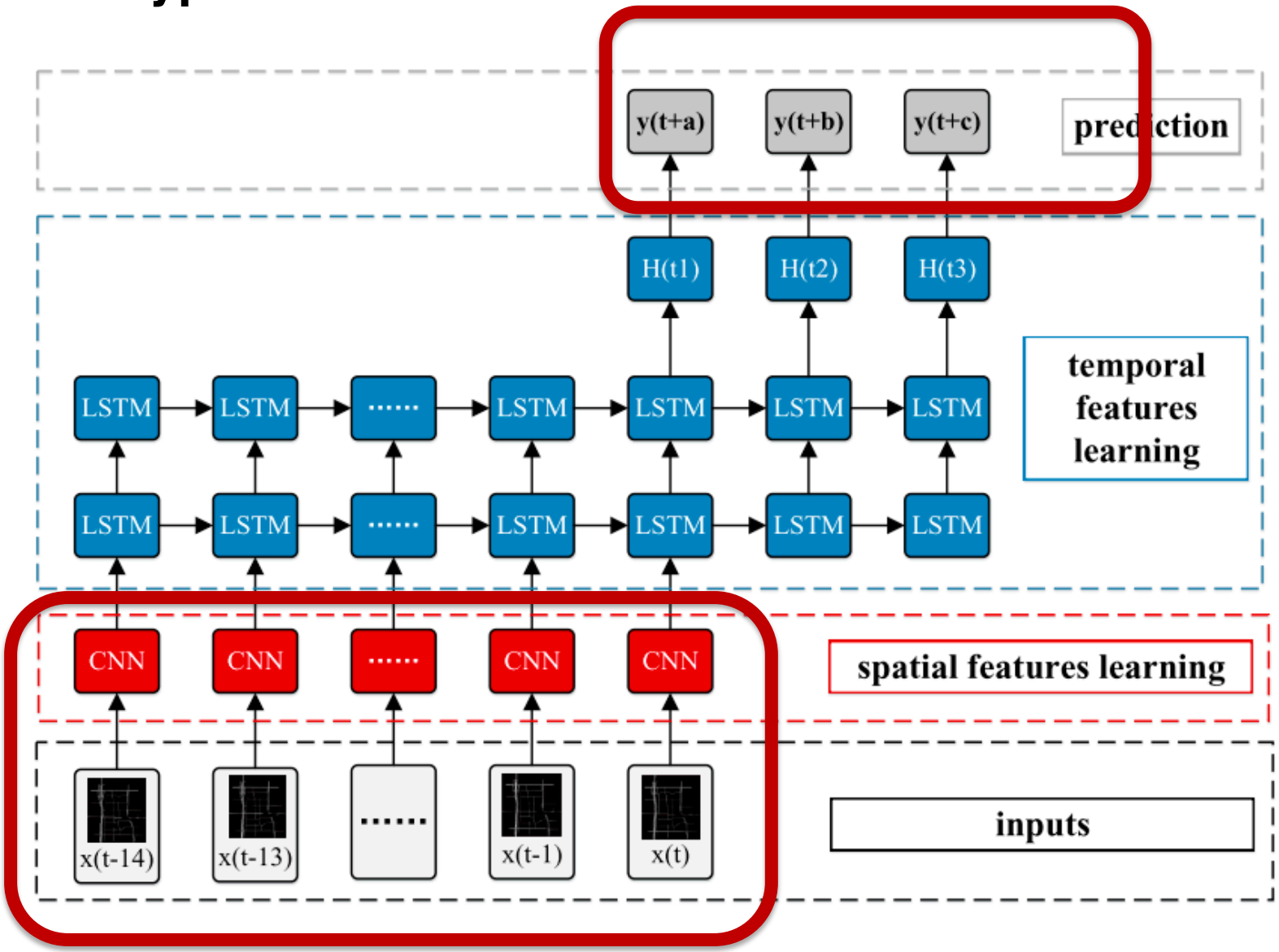




A new type of architecture

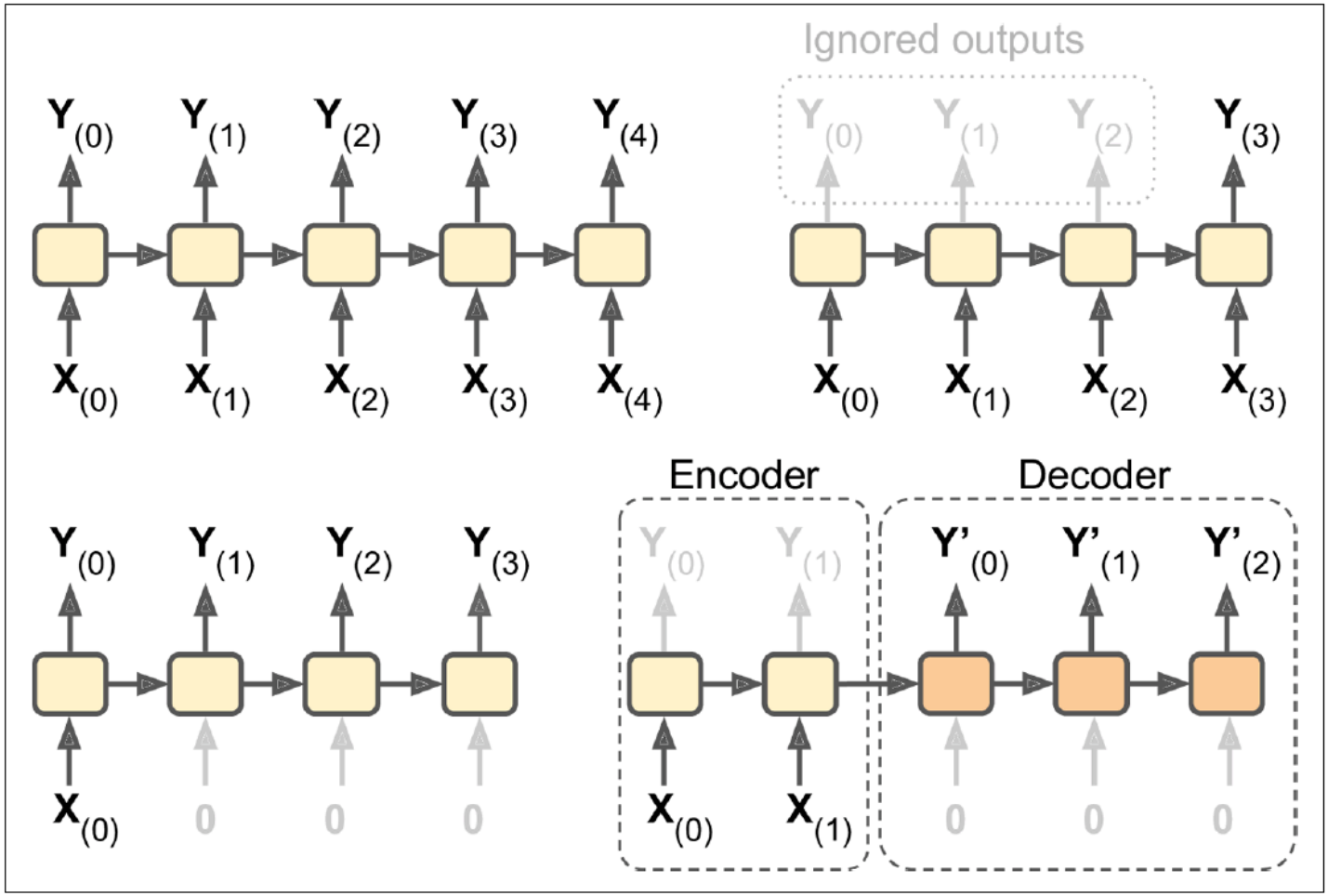


A new type of architecture



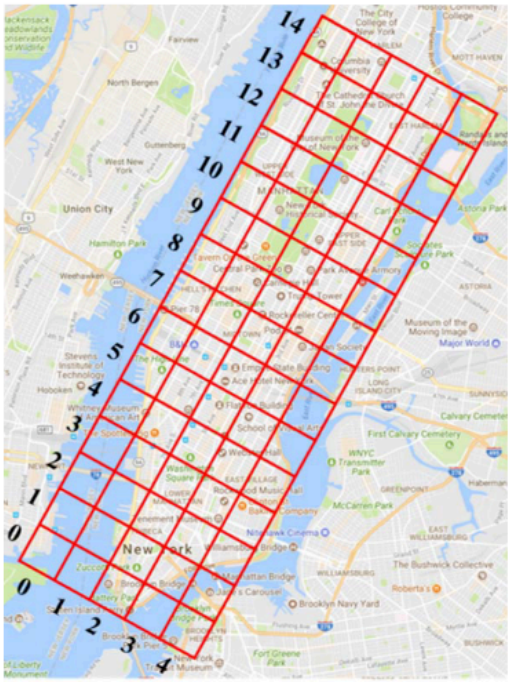
Encoder-decoder architectures

- Many possible formulations
- Encoder-decoder



Taxi demand prediction

- City is divided into N non-overlapping zones
- Record the number of taxi requests per zone



Deep Multi-View Spatial-Temporal Network for Taxi Demand Prediction

Huaxiu Yao*, Fei Wu
Pennsylvania State University
{huaxiuyao, fxw133}@ist.psu.edu

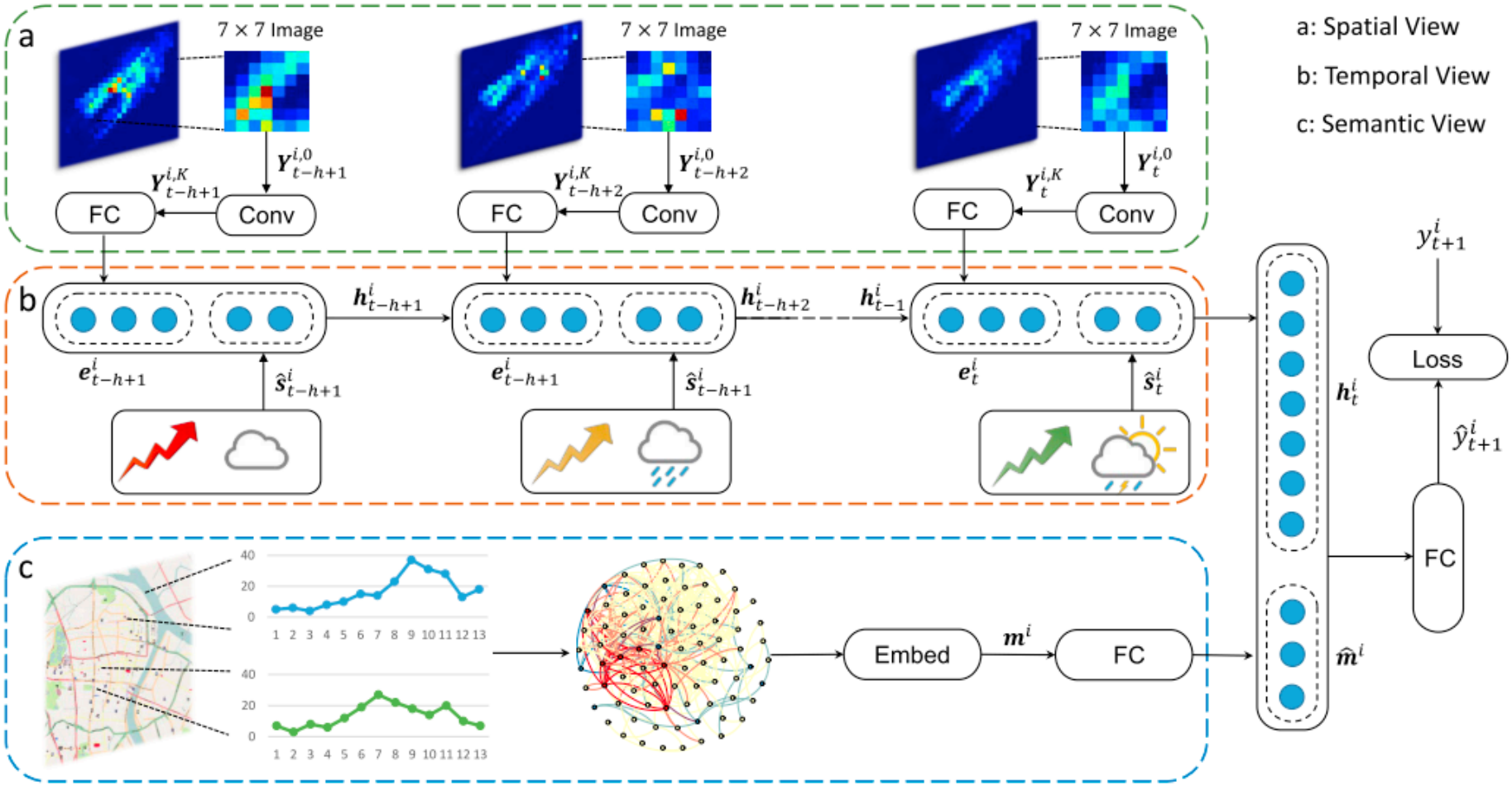
Jintao Ke*
Hong Kong University of Science and Technology
jke@connect.ust.hk

Xianfeng Tang
Pennsylvania State University
xianfeng@ist.psu.edu

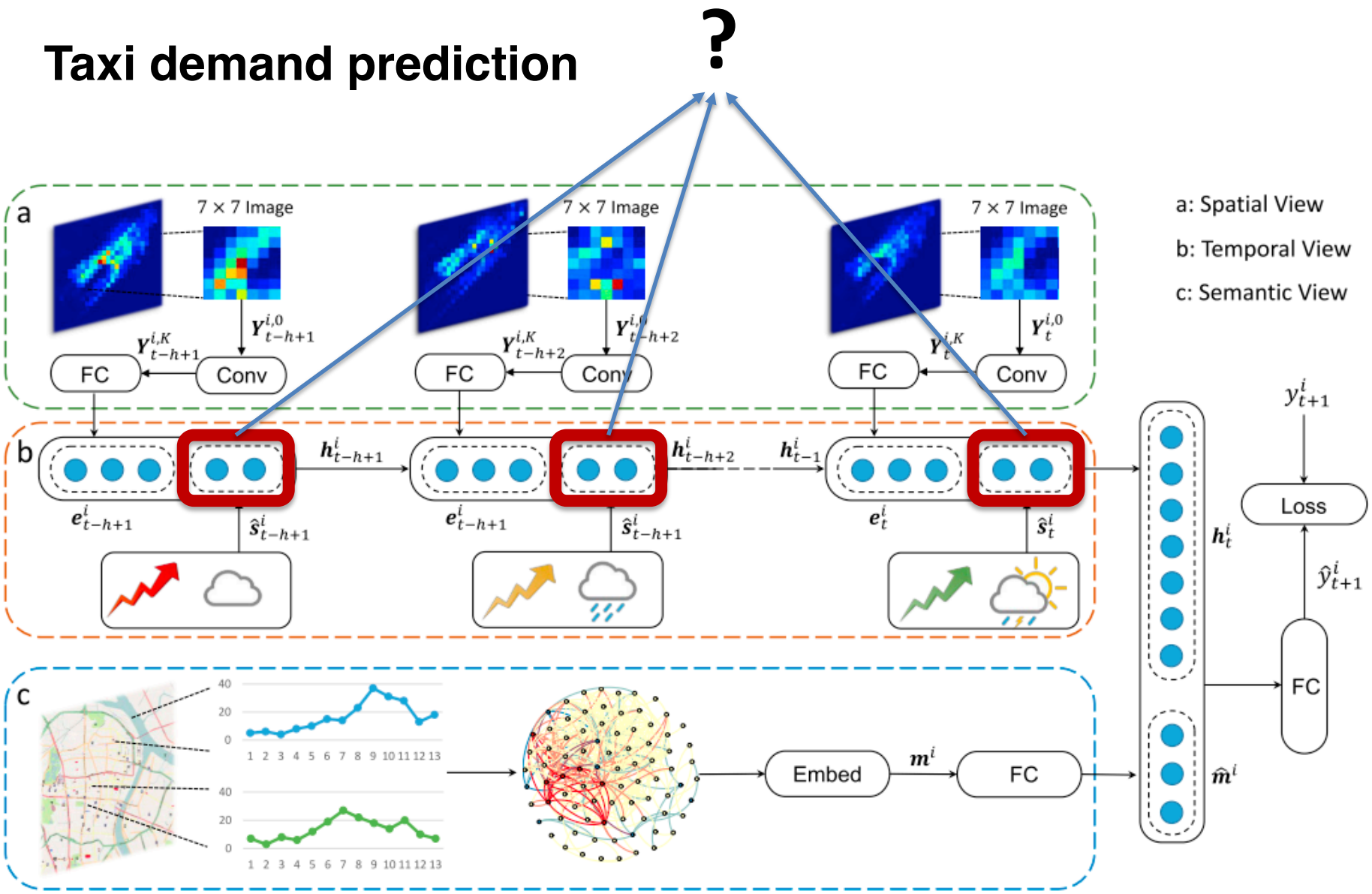
Yitian Jia, Siyu Lu, Pinghua Gong, Jieping Ye
Didi Chuxing
{jiayitian, lusiyu, gongpinghua, yejieping}@didichuxing.com

Zhenhui Li
Pennsylvania State University
jessieli@ist.psu.edu

Taxi demand prediction



Taxi demand prediction



Embeddings

- We often use categorical variables in our models
  - Weather:
    - Rain, snow, sunny, cloudy
  - Day of the week
  - Type of event
    - Concert, soccer game, basketball game
  - Service status
    - Normal, minor disruptions, major disruptions
- How to represent this type of data in our model?

Embeddings

- We often use categorical variables in our models
- Typically use one-hot encoding
  - Rain: $[0, 1, 0, 0, 0, 0]$
  - Snow $[1, 0, 0, 0, 0, 0]$
  - Sunny $[0, 0, 1, 0, 0, 0]$

Embeddings

- We often use categorical variables in our models
- Typically use one-hot encoding
  - Rain: $[0, 1, 0, 0, 0, 0]$
  - Snow $[1, 0, 0, 0, 0, 0]$
  - Sunny $[0, 0, 1, 0, 0, 0]$

Vocabulary size = 6

Embeddings

Issues with one-hot encoding

- For large vocabularies, it becomes very high dimensional
- Extremely sparse
  - DL methods don't work well with sparse data
- The mapping is completely uninformed: "similar" categories are not placed closer to each other in embedding space.

Embeddings

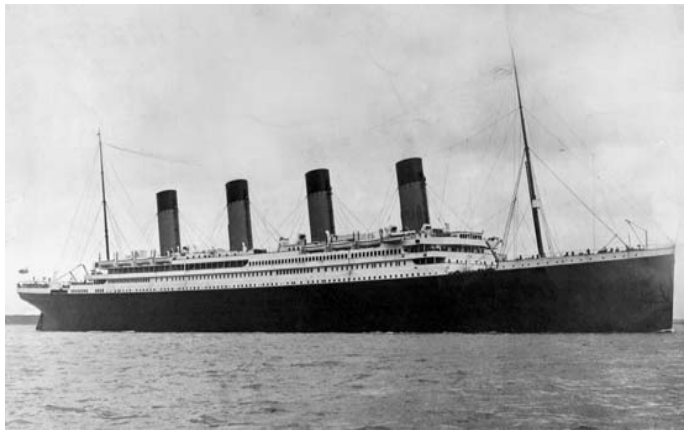
NLP tasks are faced with a similar problem

- How should we represent each word?
- Extremely sparse
  - DL methods don't work well with sparse data
- The mapping is completely uninformed: "similar" categories are not placed closer to each other in embedding space.

Embeddings

NLP tasks are faced with a similar problem

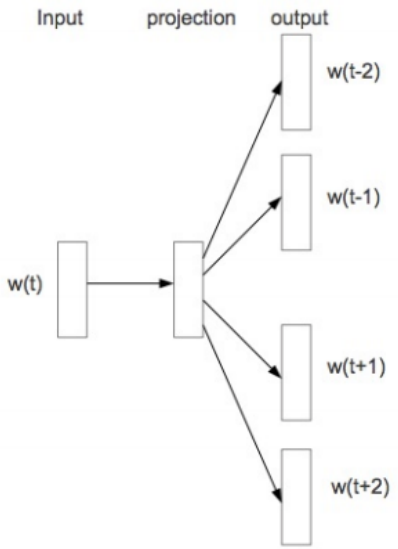
- How should we represent each word?

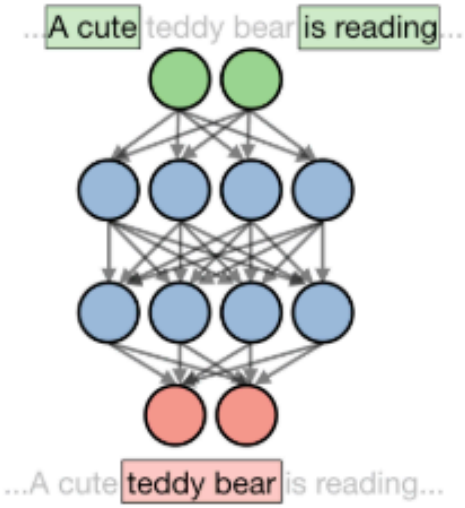


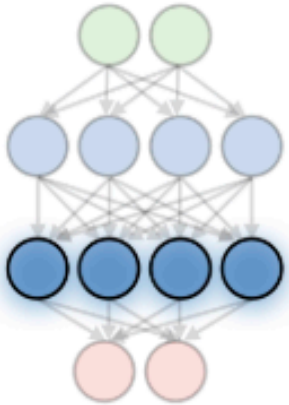
Embeddings

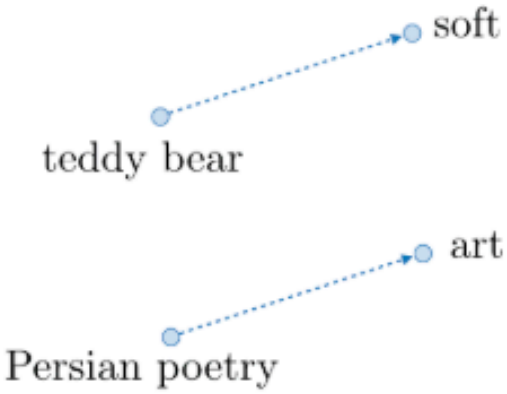
NLP tasks are faced with a similar problem

- Word2Vec









Train network on proxy task → Extract high-level representation → Compute word embeddings

https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks

Embeddings

- Word2vec vs one-hot encoding
  - Rain $[0, 1, 0, 0, 0, 0]$
  - Rain $[0.23, 0.051, 0.674, 0.599, 0.111, 0.672]$

Embeddings

```
In [1]: import gensim
```

```
In [3]: model=gensim.models.Word2Vec.load("gensimModel")
```

```
In [4]: model.wv.most_similar(positive=['customer'])
```

```
Out[4]: [('passenger', 0.855195164680481),
         ('lady', 0.6993716955184937),
         ('person', 0.6990634202957153),
         ('woman', 0.6973735690116882),
         ('vagrant', 0.6603719592094421),
         ('man', 0.6498541831970215),
         ('child', 0.6353968977928162),
         ('male', 0.6159298419952393),
         ('passengers', 0.606838583946228),
         ('female', 0.5832453966140747)]
```

```
In [14]: model.wv.most_similar(positive=['oxford'], negative=['incident'])
```

```
Out[14]: [('nhg', 0.484191358089447),
          ('vic', 0.45740655064582825),
          ('leicester', 0.4572640061378479),
          ('n241', 0.4489314556121826),
          ('pimlico', 0.4430849552154541),
          ('s220', 0.4392928183078766),
          ('stockwell', 0.43596023321151733),
          ('s221', 0.4301430583000183),
          ('s250', 0.42746976017951965),
          ('n244', 0.4272845387458801)]
```

```
In [16]: model.wv.most_similar(positive=['incident'])
```

```
Out[16]: [('matter', 0.6234616041183472),
          ('spad', 0.42893028259277344),
          ('issue', 0.4024146795272827),
          ('situation', 0.39529454708099365),
          ('item', 0.3943479061126709),
          ('conversation', 0.39270853996276855),
          ('discussion', 0.39042913913726807),
          ('sm', 0.38687431812286377),
          ('soo', 0.3864406943321228),
```

Embeddings

- Word vectors demonstrated the usefulness of dense vector representation of the inputs

- In other domains, we simply initialize each categorical variable as a list of random numbers, then train it with other parameters in the model

Metro passenger flow prediction

- Predict passenger arrival at stations for the next 15 minutes

Contents lists available at ScienceDirect

Transportation Research Part C

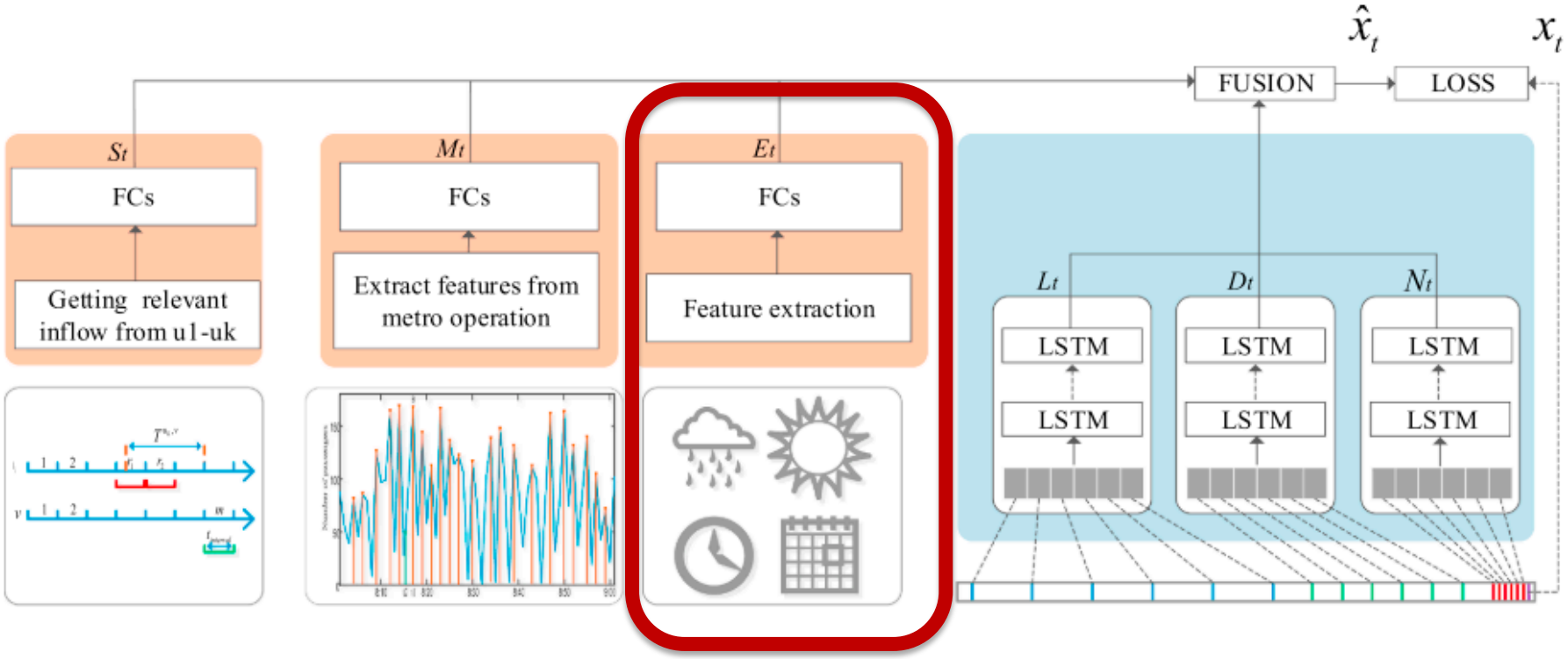
journal homepage: www.elsevier.com/locate/trc

DeepPF: A deep learning based architecture for metro passenger flow prediction

Yang Liu, Zhiyuan Liu*, Ruo Jia

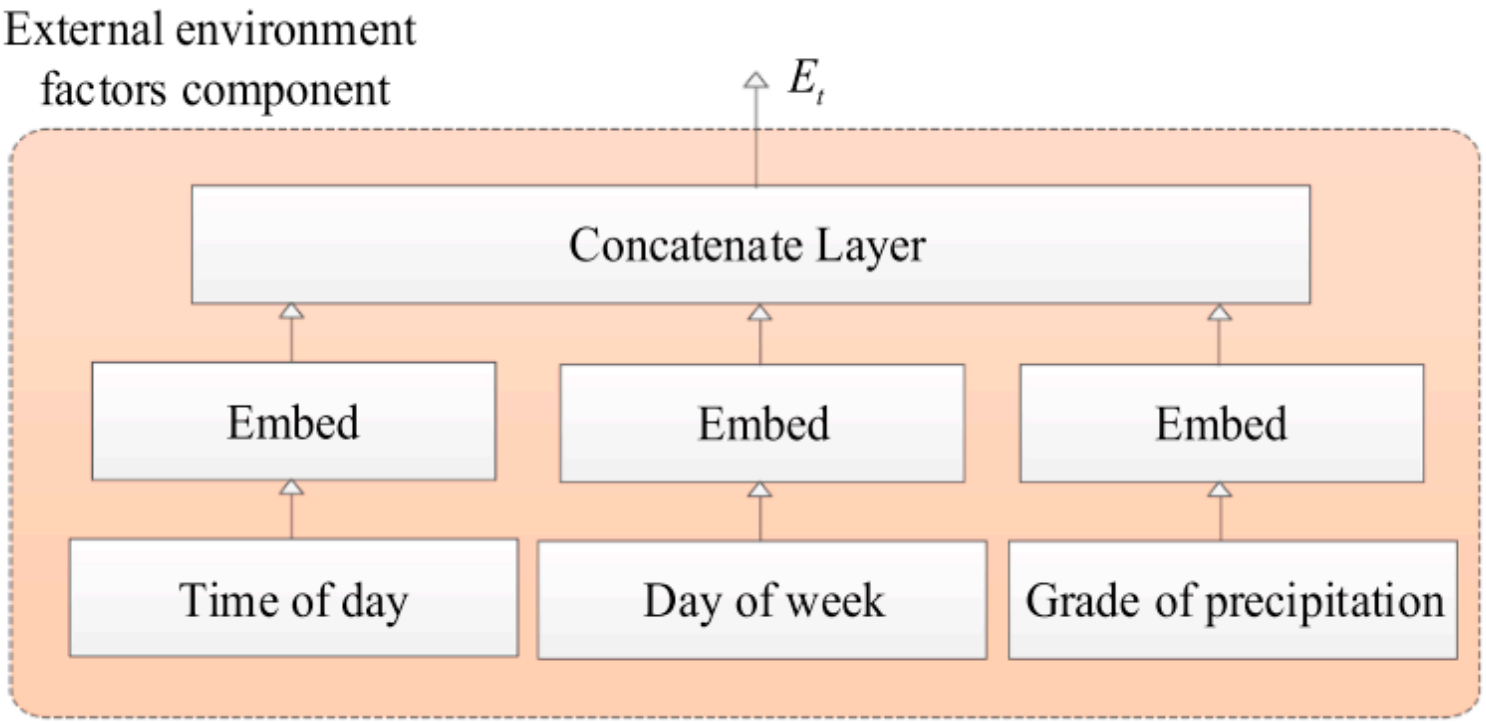
Metro passenger flow prediction

- Predict passenger arrival at stations for the next 15 minutes



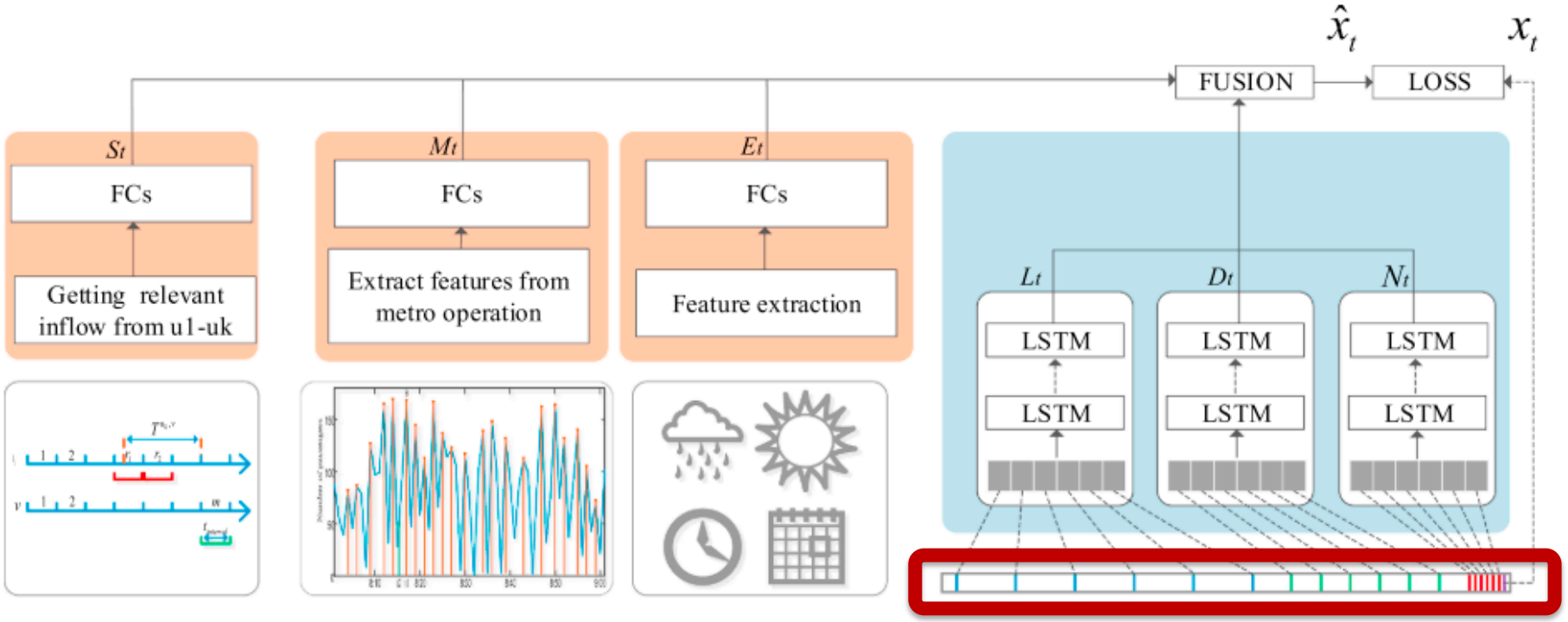
Metro passenger flow prediction

- Predict passenger arrival at stations for the next 15 minutes



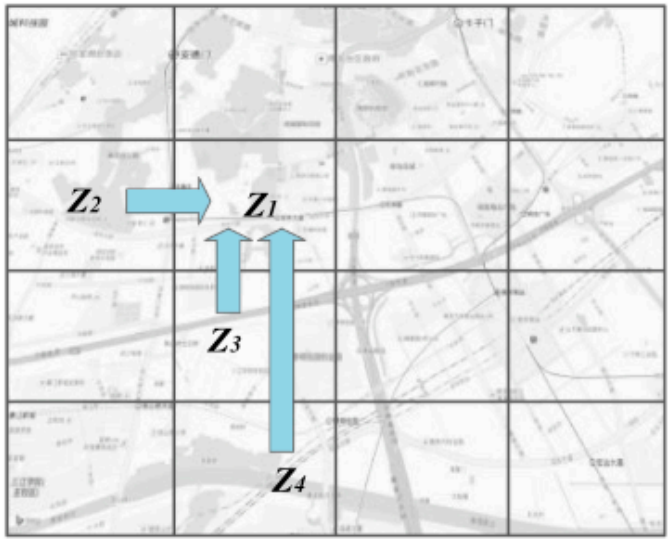
Metro passenger flow prediction

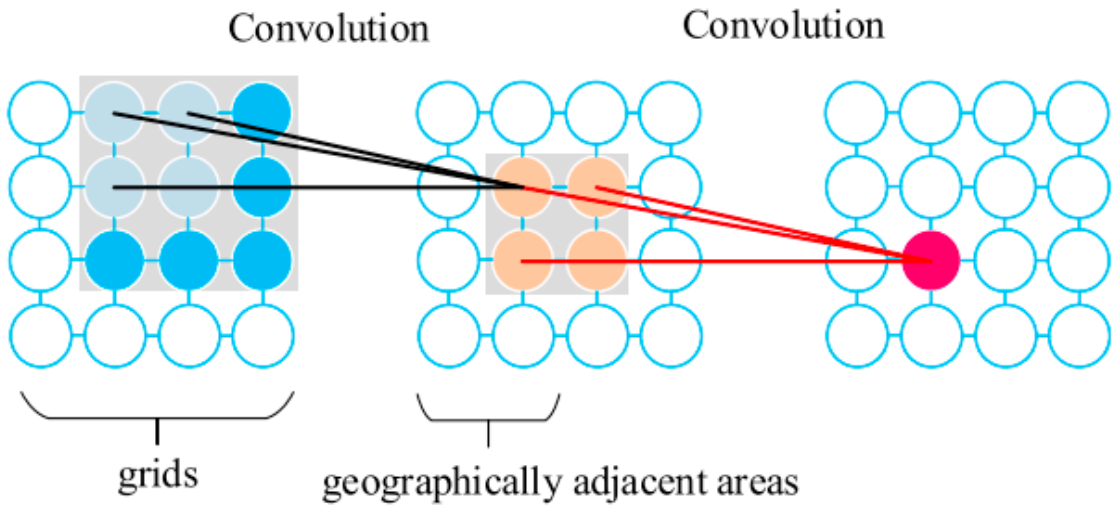
- Predict passenger arrival at stations for the next 15 minutes



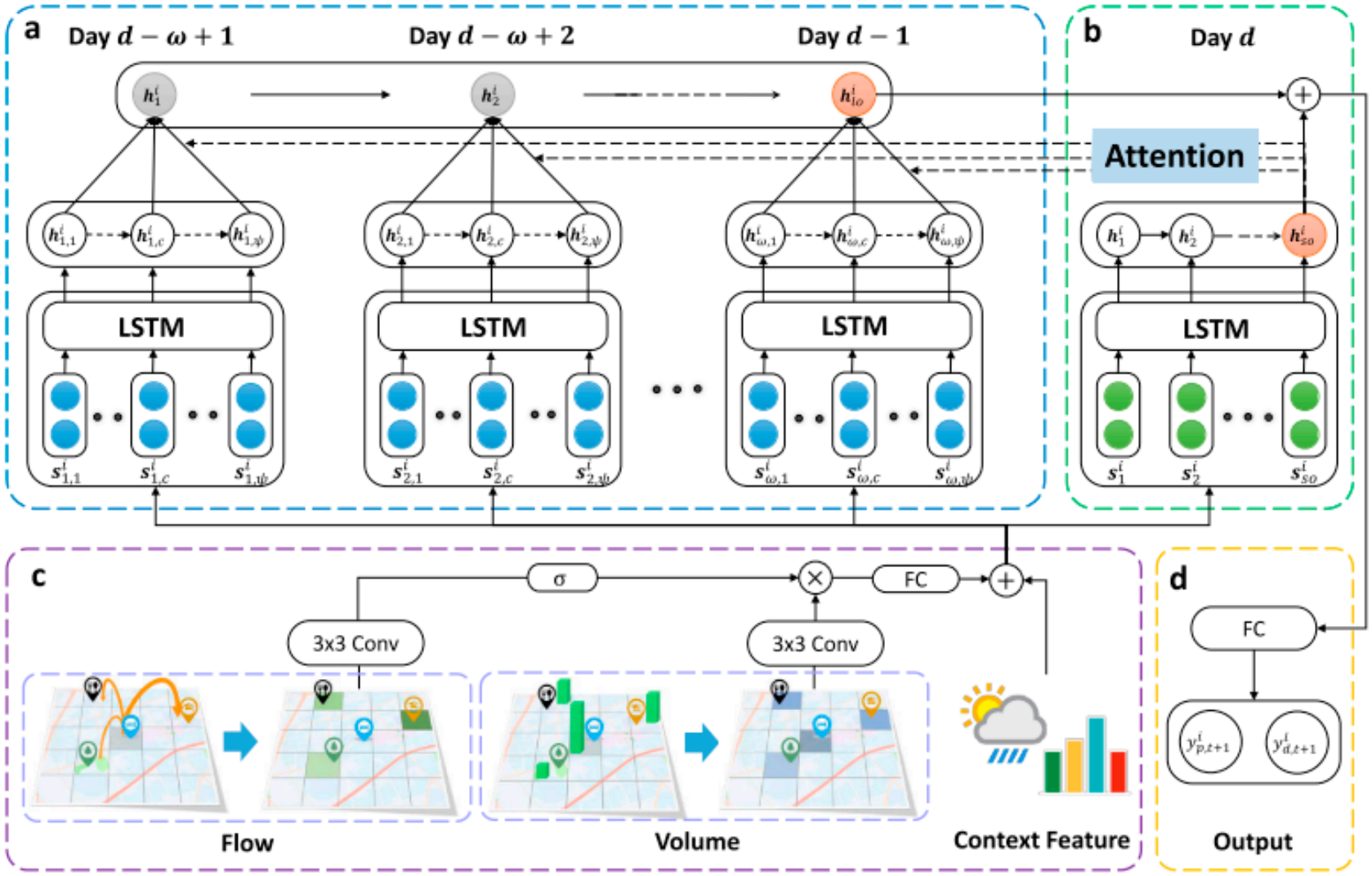
Metro passenger flow prediction

- Predict passenger arrival at stations for the next 15 minutes





Traffic flow prediction

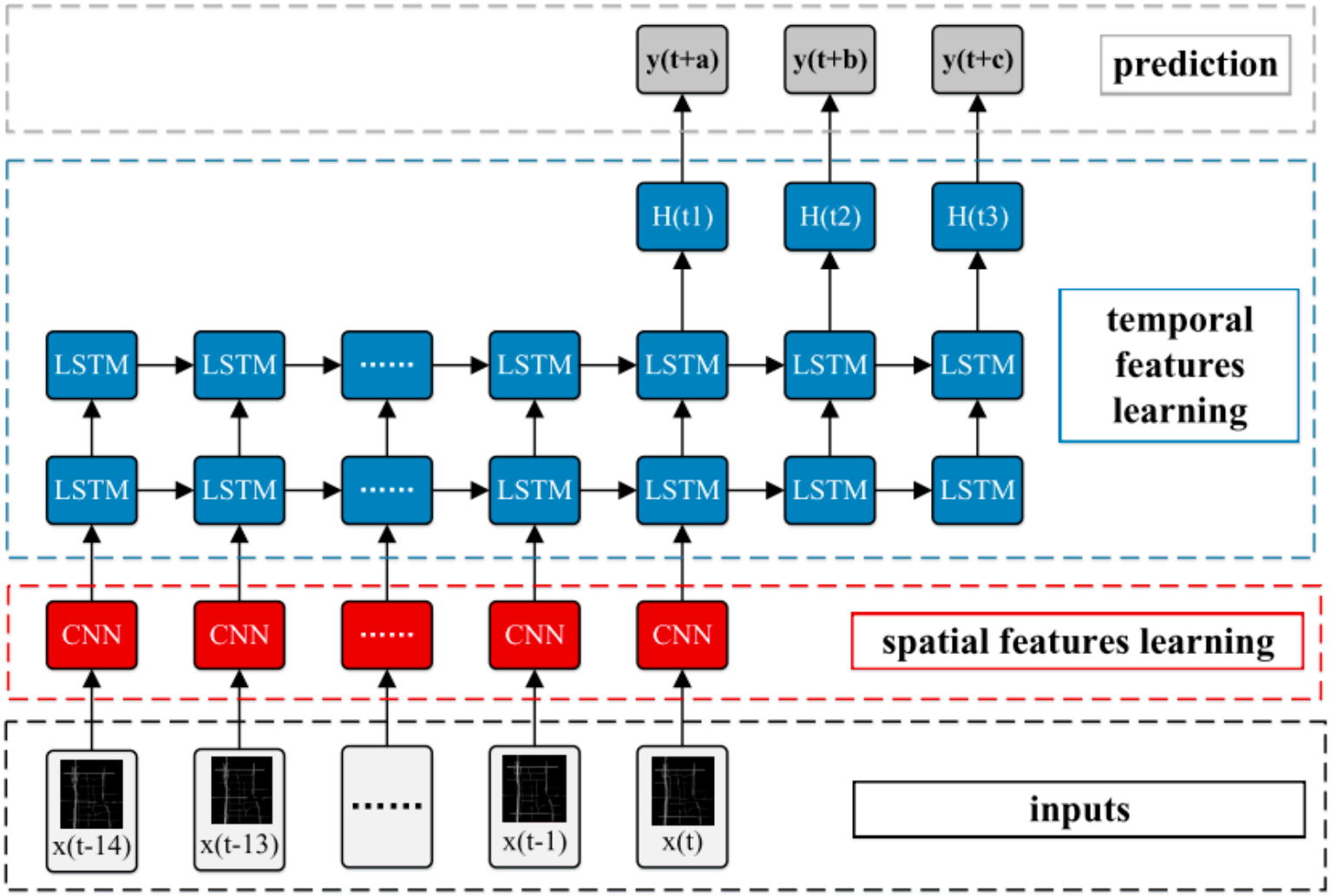


Modeling spatiotemporal data

- Use CNN in the initial layers to capture spatial dependencies
  - Feature extraction

- Use RNN on features extracted by CNN

A new type of architecture



A new type of architecture

0	Inputs	1	(163,148)
1	Convolution	16	(3,3)
	Max-pooling	16	(2,2)
	Activation (relu)	— —	— —
	Batch-normalization	— —	— —
2	Convolution	32	(3,3)
	Max-pooling	32	(2,2)
	Activation (relu)	— —	— —
	Batch-normalization	— —	— —
3	Convolution	64	(3,3)
	Activation (relu)	— —	— —
	Batch-normalization	— —	— —
4	Convolution	64	(3,3)
	Activation (relu)	— —	— —
	Batch-normalization	— —	— —
5	Convolution	128	(3,3)
	Max-pooling	128	(2,2)
	Activation (relu)	— —	— —
	Batch-normalization	— —	— —
6	Flatten	— —	— —
7	Fully connected	— —	278
8	Lstm1	— —	800
	Activation (tanh)	— —	
9	Lstm2	— —	800
	Activation (tanh)	— —	— —
10	Dropout (0.2)	— —	— —
11	Fully connected	— —	278

A new type of architecture

0	Inputs	1	(163,148)
1	Convolution	16	(3,3)
	Max-pooling	16	(2,2)
	Activation (relu)	— —	— —
	Batch-normalization	— —	— —
2	Convolution	32	(3,3)
	Max-pooling	32	(2,2)
	Activation (relu)	— —	— —
	Batch-normalization	— —	— —
3	Convolution	64	(3,3)
	Activation (relu)	— —	— —
	Batch-normalization	— —	— —
4	Convolution	64	(3,3)
	Activation (relu)	— —	— —
	Batch-normalization	— —	— —
5	Convolution	128	(3,3)
	Max-pooling	128	(2,2)
	Activation (relu)	— —	— —
	Batch-normalization	— —	— —
6	Flatten	— —	— —
7	Fully connected	— —	278
8	Lstm1	— —	800
	Activation (tanh)	— —	
9	Lstm2	— —	800
	Activation (tanh)	— —	— —
10	Dropout (0.2)	— —	— —
11	Fully connected	— —	278

ConvLSTM

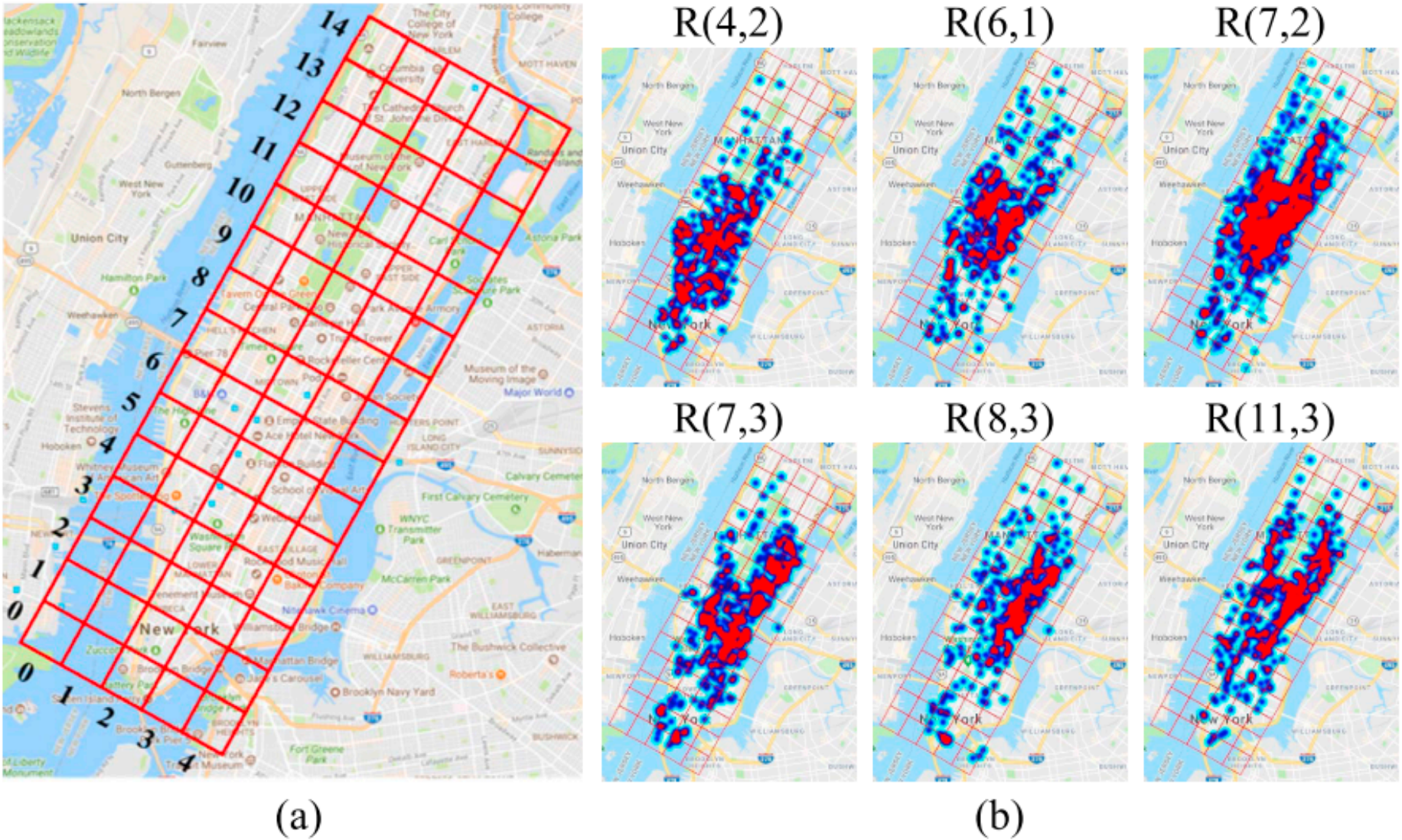
- The downside to using LSTMs for capturing the temporal dependencies is their inability in using the spatial information encoded in the input

- ConvLSTMs address this problem by having convolutional structures in both the input-to-state and state-to-state transitions.

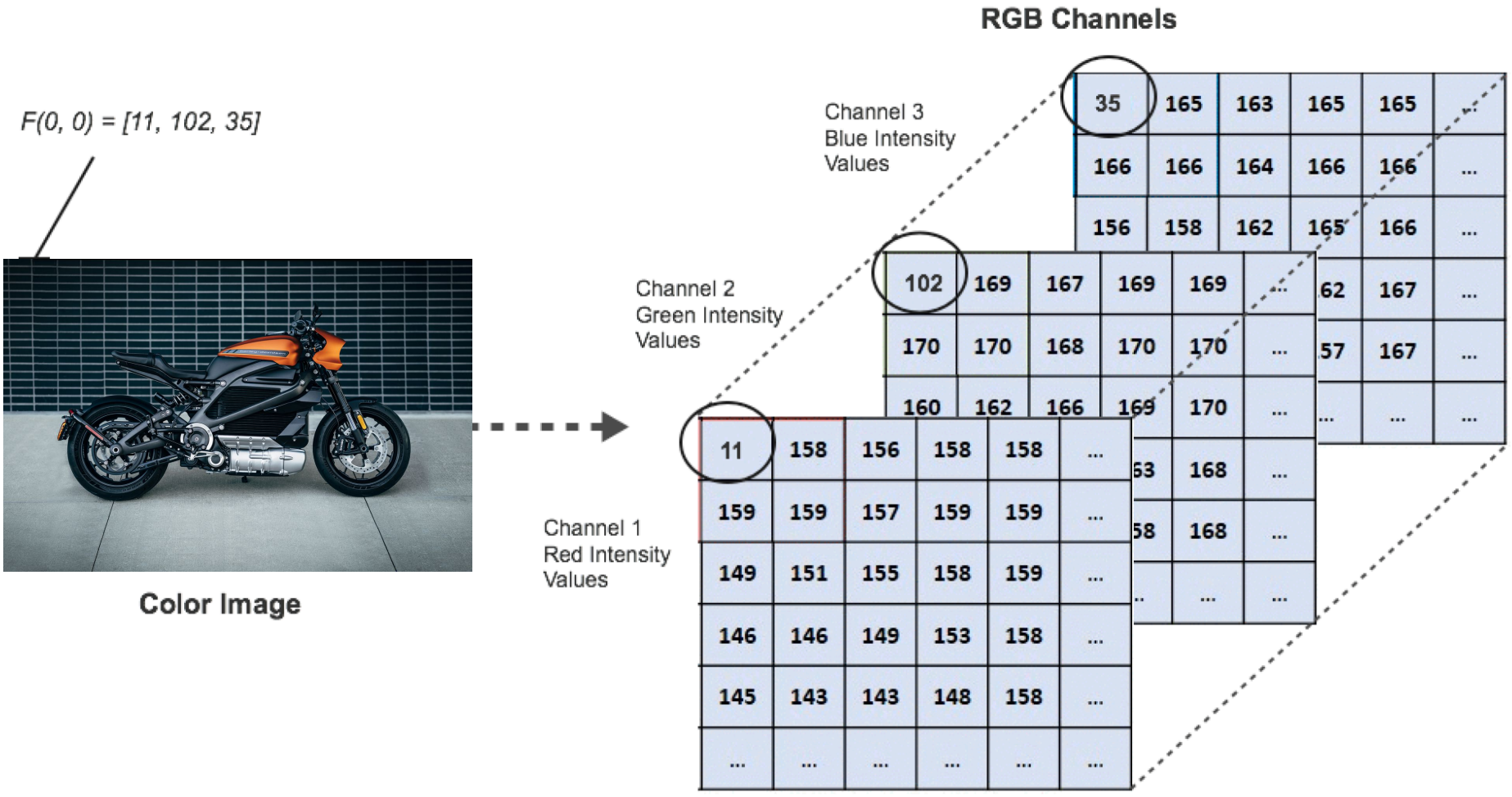
$$
\begin{aligned}
i_t &= \sigma(W_{xi} * X_t + W_{hi} * H_{t-1} + W_{ci} \odot C_{t-1} + b_i) \\
f_t &= \sigma(W_{xf} * X_t + W_{hf} * H_{t-1} + W_{cf} \odot C_{t-1} + b_f) \\
C_t &= f_t t_{-1} + i_t \odot tanh(W_{xc} * X_t + + W_{hc} * H_{t-1} + b_c) \\
o_t &= \sigma(W_{xo} * X_t + W_{ho} * H_{t-1} + W_{co} \odot C_t + b_o) \\
H_t &= o_t \odot tanh(C_t)
\end{aligned}
$$

Contextualized Spatial–Temporal Network for Taxi Origin-Destination Demand Prediction

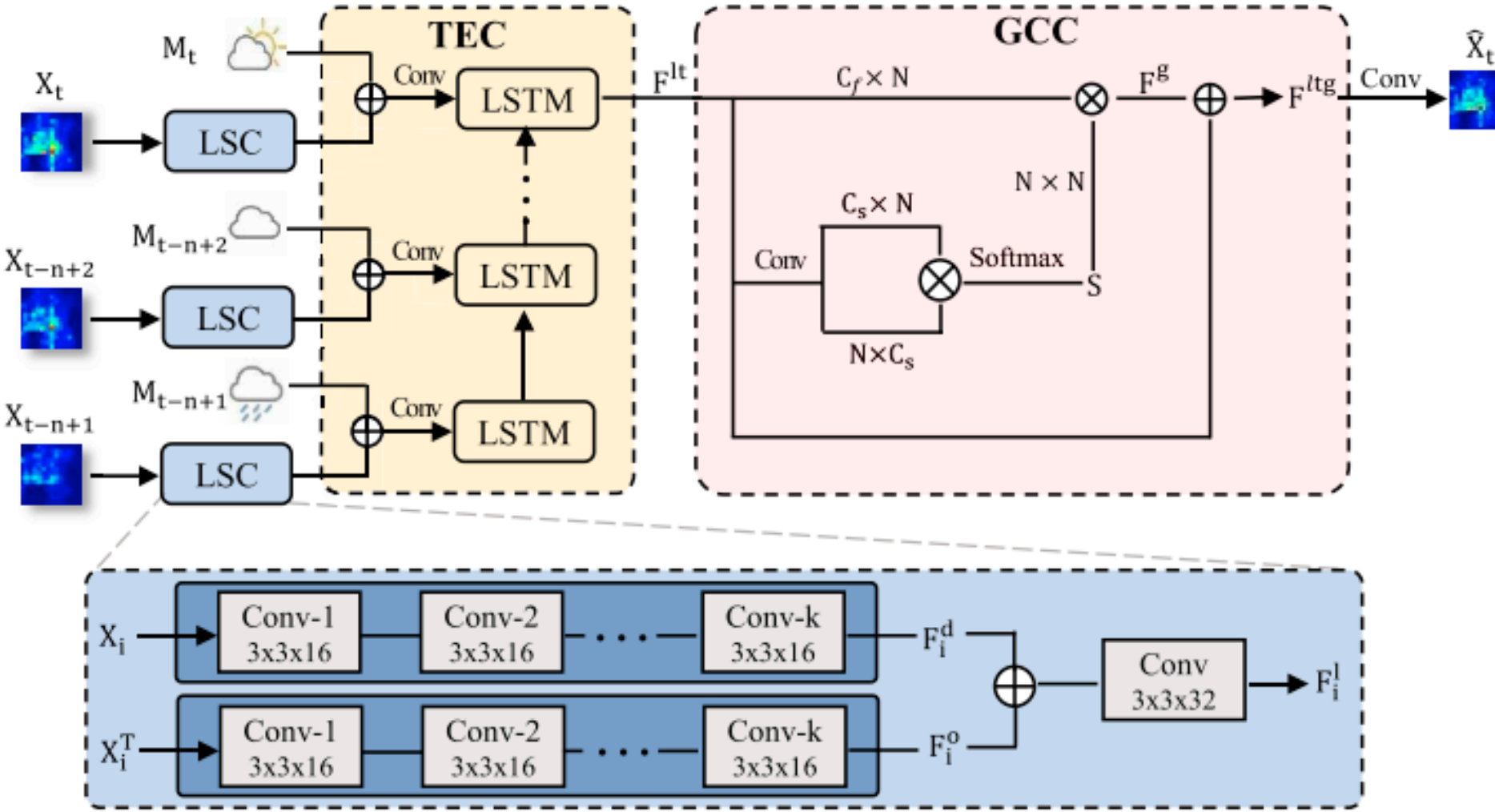
Lingbo Liu, Zhilin Qiu, Guanbin Li, *Member, IEEE*, Qing Wang, Wanli Ouyang, *Senior Member, IEEE*, and Liang Lin, *Senior Member, IEEE*



Let's clarify what an image is



ConvLSTM

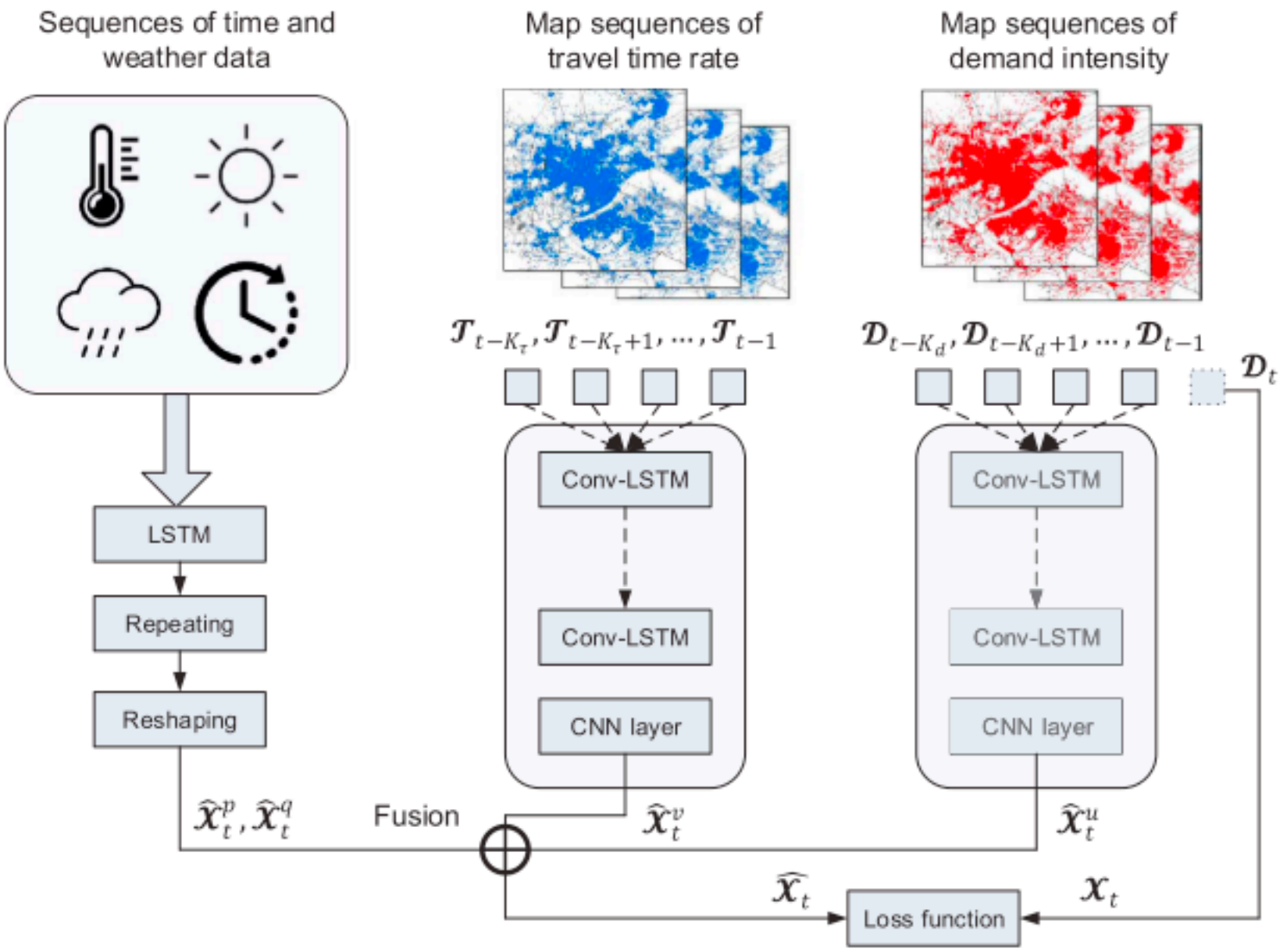

TEC
GCC
LSC
LSTM
Conv
Softmax
Conv-1
3x3x16
Conv-2
3x3x16
Conv-k
3x3x16
Conv
3x3x32

ConvLSTM

Short-term forecasting of passenger demand under on-demand ride services: A spatio-temporal deep learning approach

Jintao Ke[a], Hongyu Zheng[b], Hai Yang[a], Xiqun (Michael) Chen[b,*]

[a] *Department of Civil and Environmental Engineering, Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong, China*
[b] *College of Civil Engineering and Architecture, Zhejiang University, Hangzhou, China*


Sequences of time and weather data
Map sequences of travel time rate
Map sequences of demand intensity
$\mathcal{T}_{t-K_\tau}, \mathcal{T}_{t-K_\tau+1}, \cdots, \mathcal{T}_{t-1}$
$\mathcal{D}_{t-K_d}, \mathcal{D}_{t-K_d+1}, \cdots, \mathcal{D}_{t-1}$
$\mathcal{D}_t$
LSTM
Repeating
Reshaping
Conv-LSTM
Conv-LSTM
CNN layer
Conv-LSTM
Conv-LSTM
CNN layer
$\widehat{\mathcal{X}}_t^p, \widehat{\mathcal{X}}_t^q$
Fusion
$\widehat{\mathcal{X}}_t^v$
$\widehat{\mathcal{X}}_t^u$
$\widehat{\mathcal{X}}_t$
$\mathcal{X}_t$
Loss function

References

CMU Deep learning course 2019
Bhiksha Raj