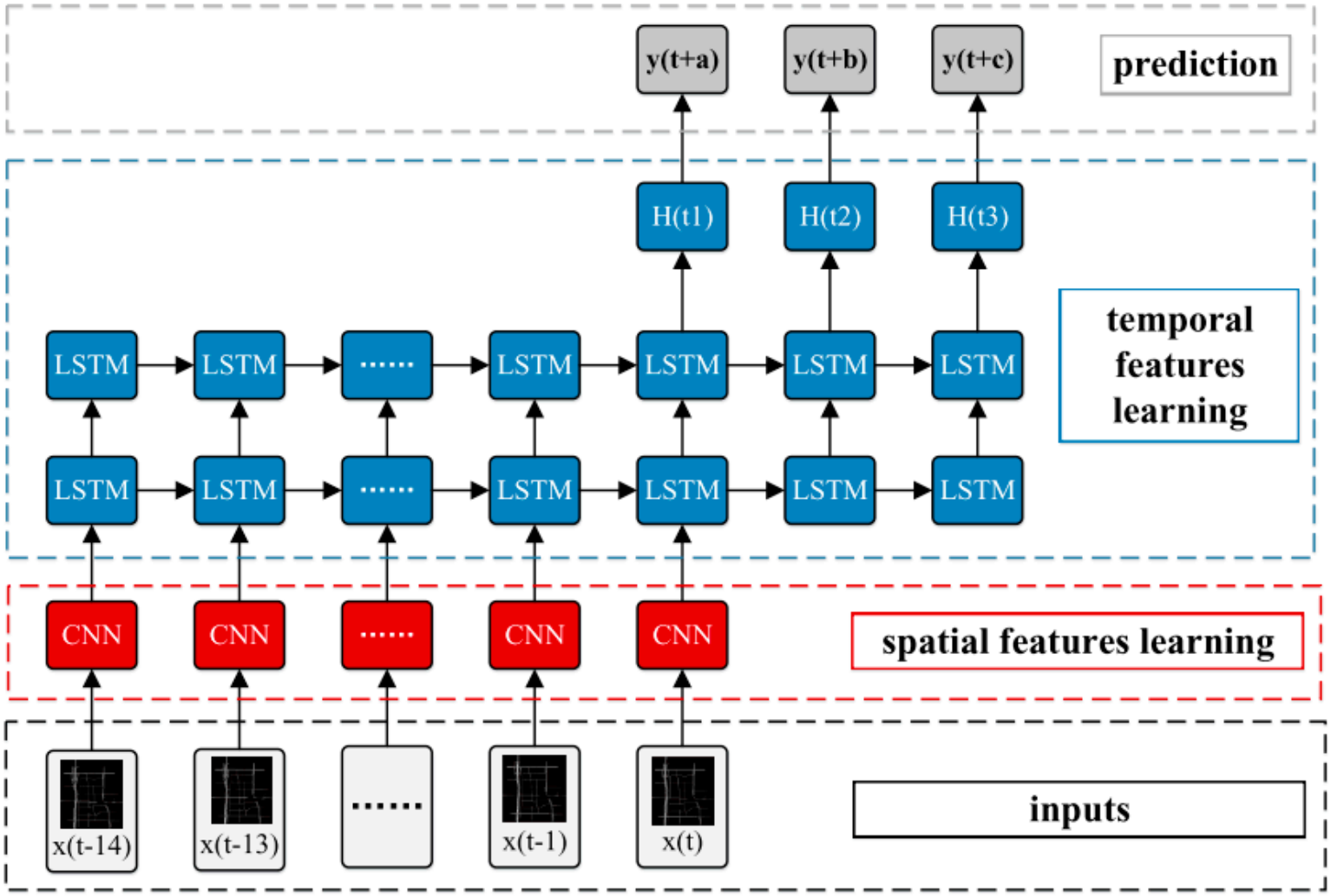
Lecture 4: Embeddings, ConvLSTM, Attention, Graph Neural Networks

Peyman Noursalehi

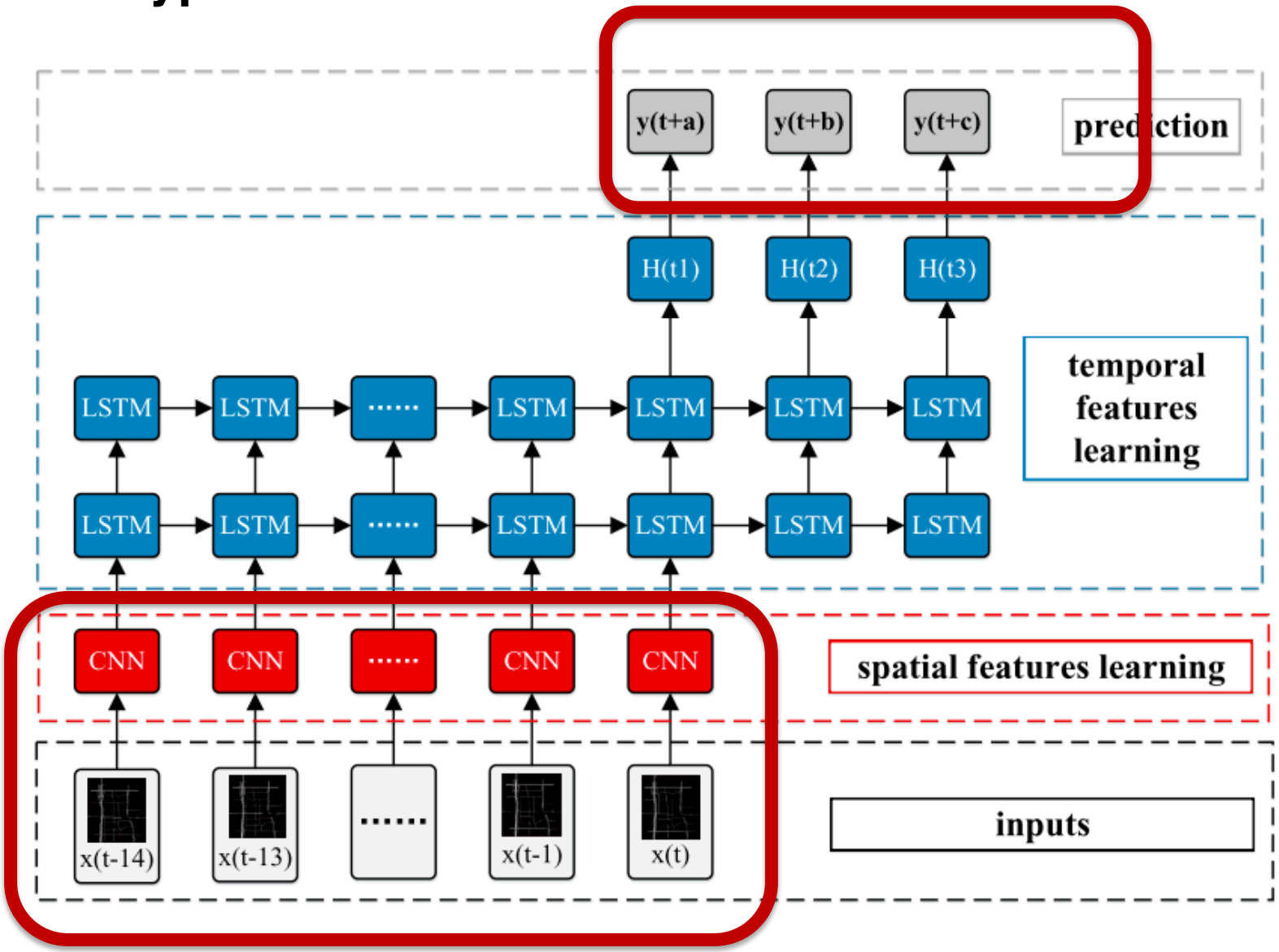
Summary so far

- CNN is the perfect tool for capturing spatial dependencies
- RNN is the perfect tool for capturing temporal dependencies
  - LSTM
  - GRU
- Use CNN in the initial layers to capture spatial dependencies
  - Feature extraction
- Use RNN on features extracted by CNN

A new type of architecture

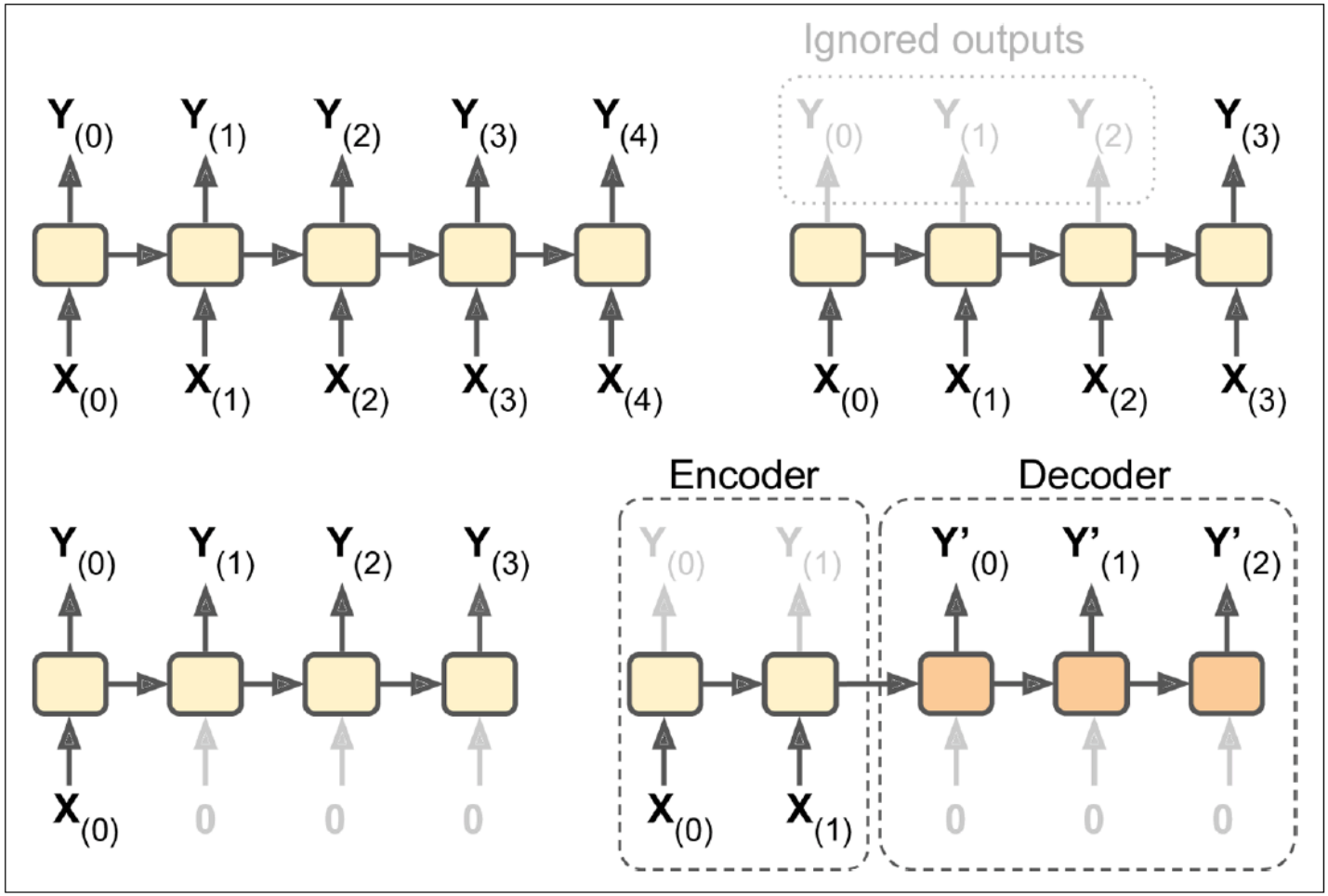


A new type of architecture



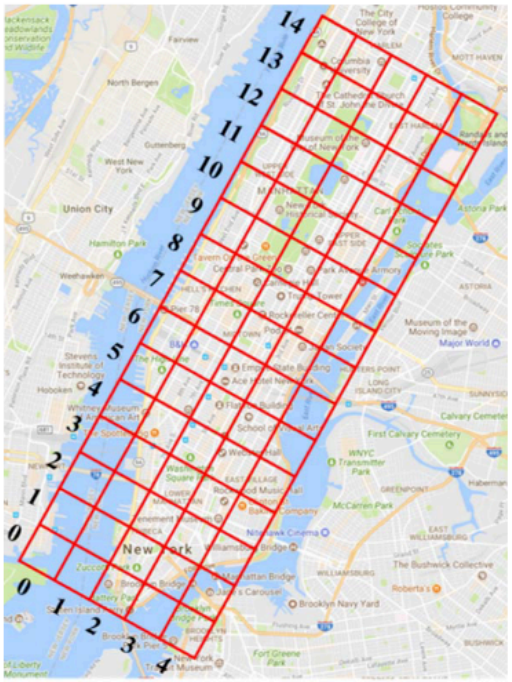
Encoder-decoder architectures

- Many possible formulations
- Encoder-decoder



Taxi demand prediction

- City is divided into N non-overlapping zones
- Predict 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

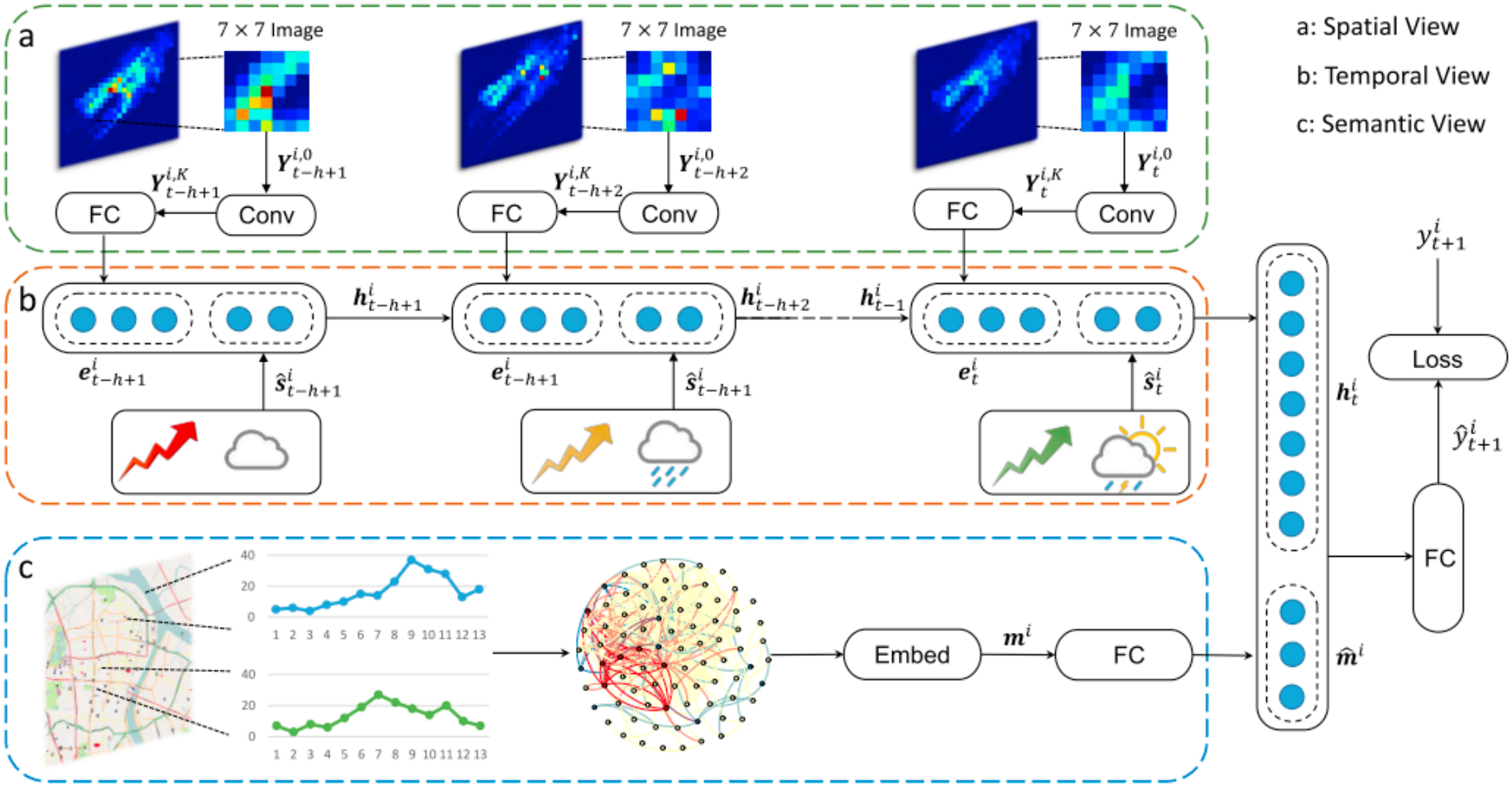
Jintao Ke*
Hong Kong University of Science and Technology
jke@connect.ust.hk

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

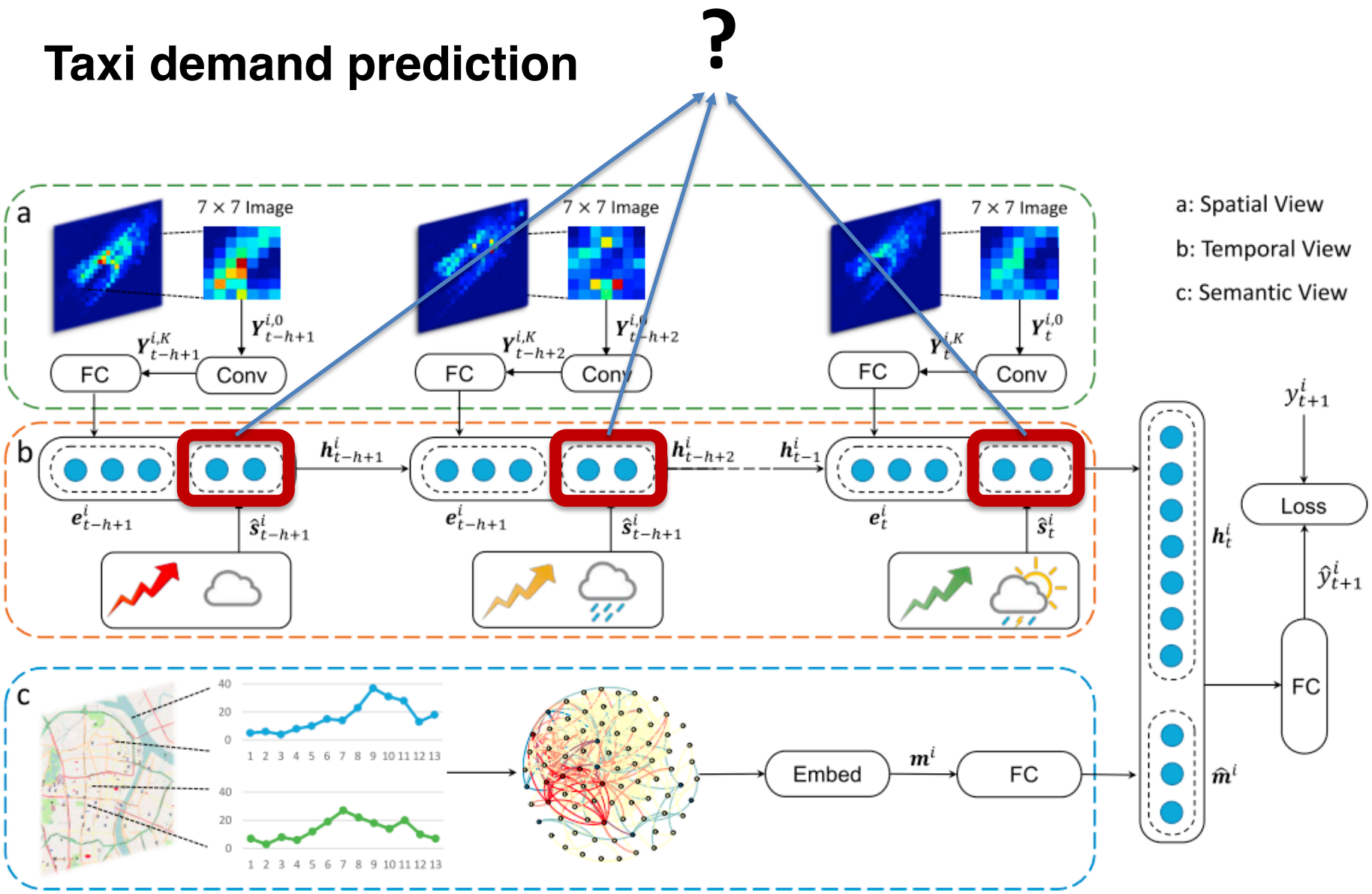
Yitian Jia, Siyu Lu, Pinghua Gong, Jieping Ye
Didi Chuxing
{jiayitian, lusiyu, gongpinghua, yejieping}@didichuxing.com

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

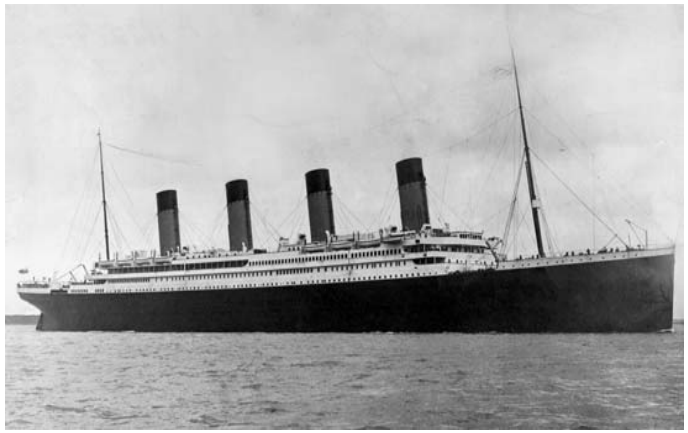
- How should we represent each word?

- Some words are more “similar”

Embeddings

NLP tasks are faced with a similar problem

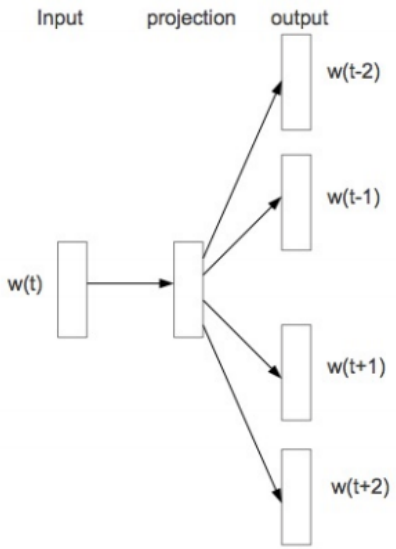
- How should we represent each word?

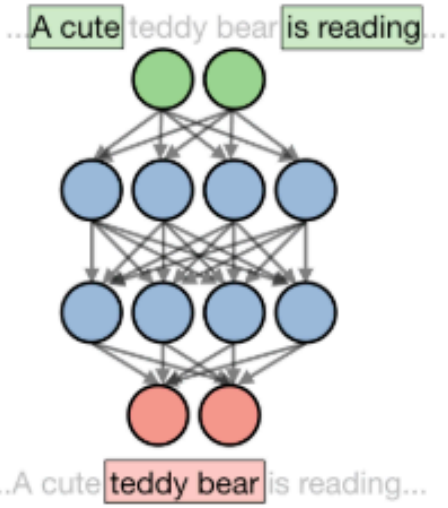


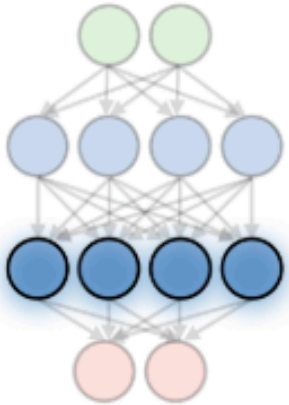
Embeddings

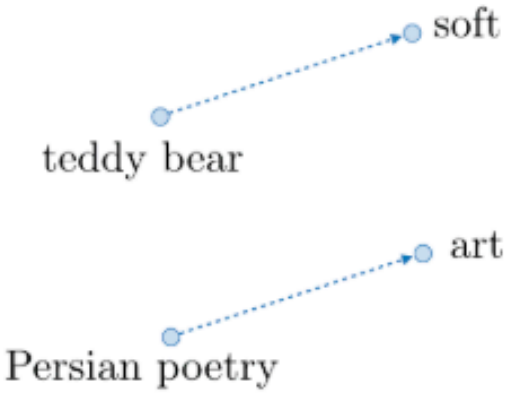
NLP tasks are faced with a similar problem

- Word2Vec





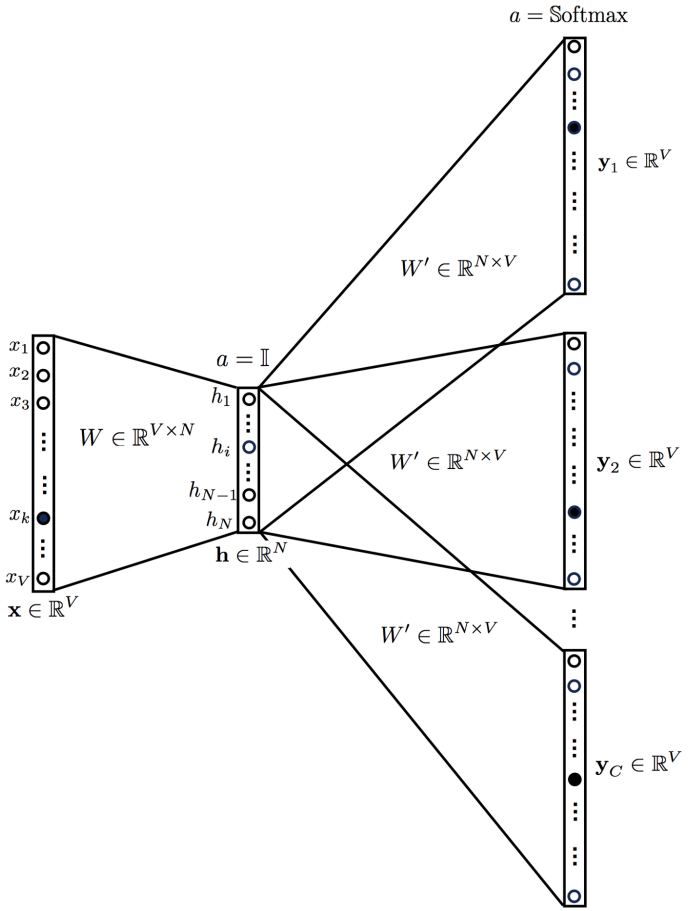




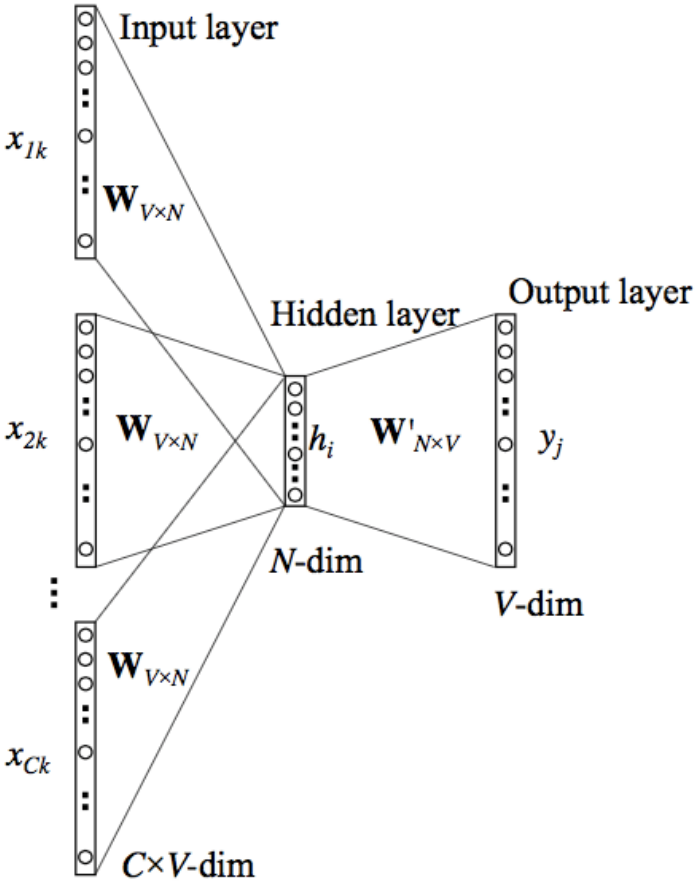
Train network on proxy task → Extract high-level representation → Compute word embeddings

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

Skip Gram



CBOW (Continuous Bag of Words)

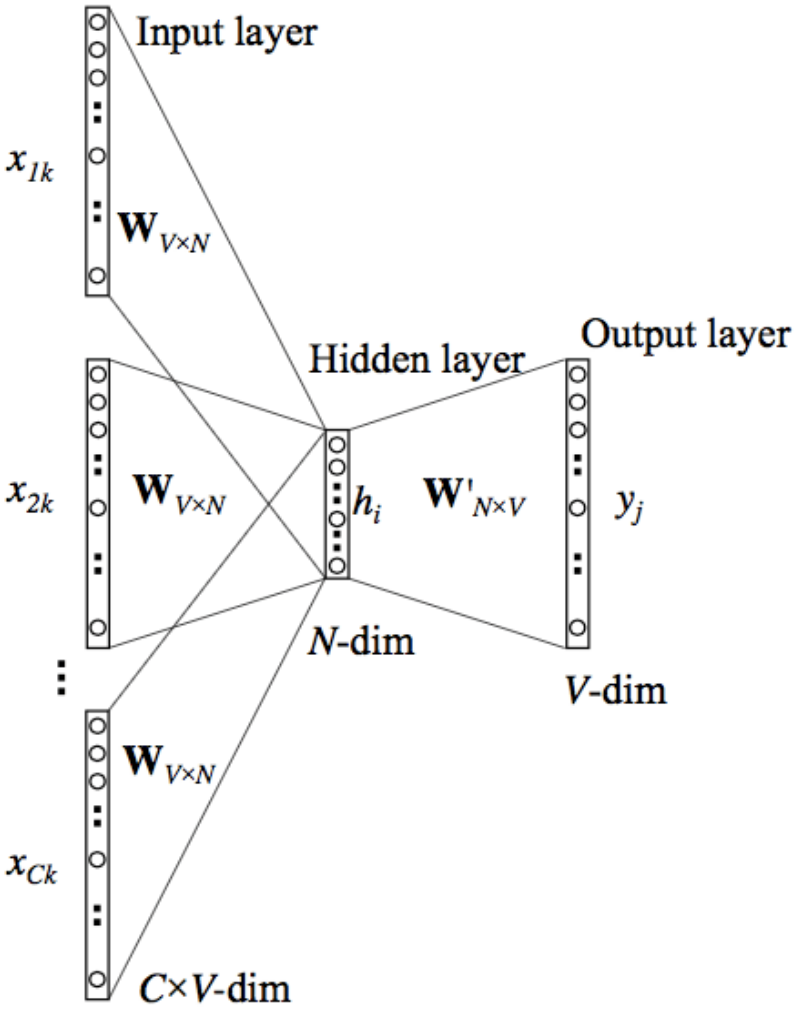


Embeddings

CBOW (Continuous Bag of Words)

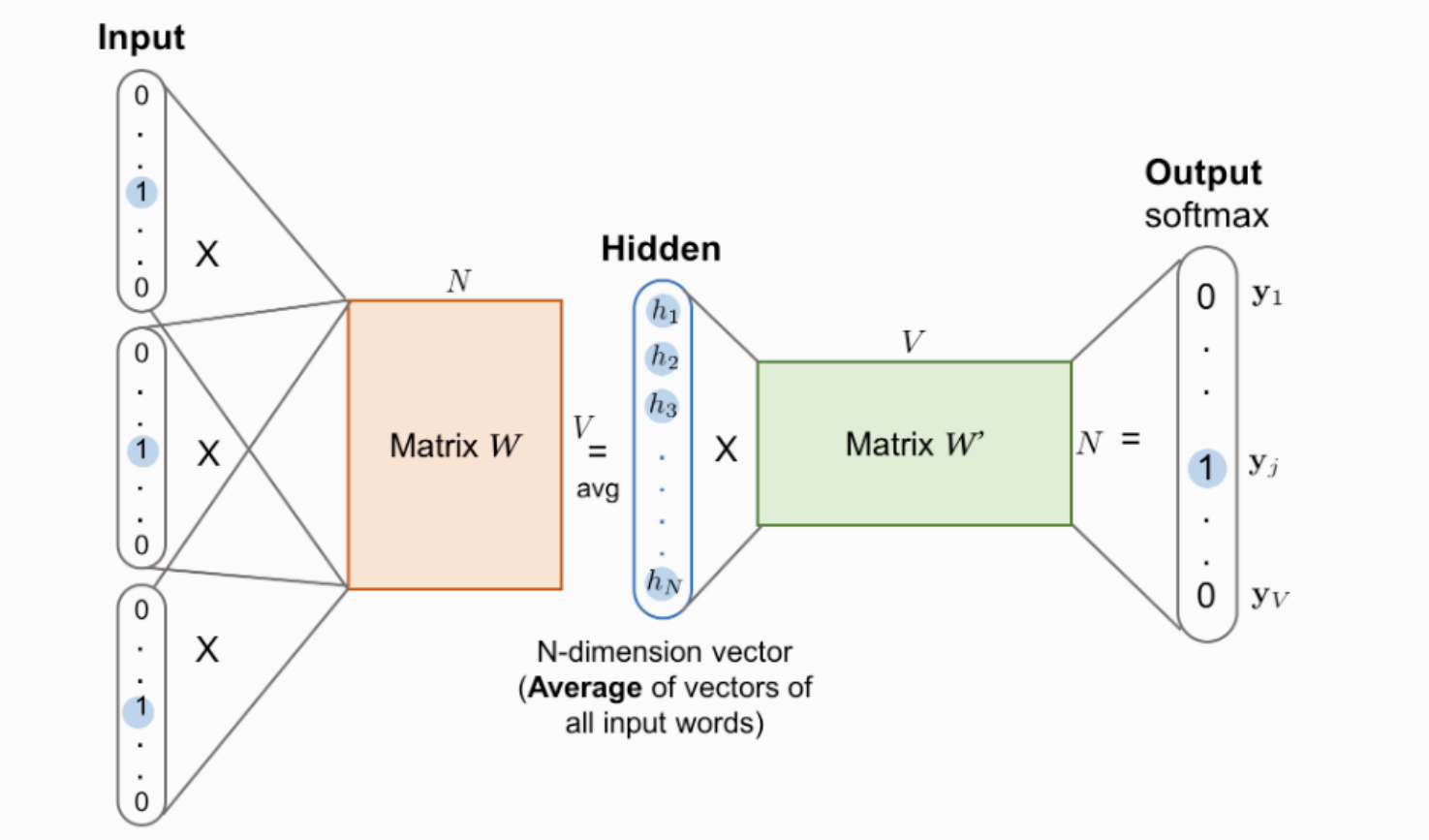
- One-hot each word
- SoftMax layer at the end

- C: Context
- V: Vocab dimension
- N: Embedding size



Embeddings

$$\mathcal{L}_\theta = -\log \frac{\exp({v'_{w_O}}^\top v_{w_I})}{\sum_{i=1}^{V} \exp({v'_{w_i}}^\top v_{w_I})} = -{v'_{w_O}}^\top v_{w_I} + \log \sum_{i=1}^{V} \exp({v'_{w_i}}^\top v_{w_I})$$



https://lilianweng.github.io/lil-log/2017/10/15/learning-word-embedding.html#context-based-continuous-bag-of-words-cbow

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'])
```

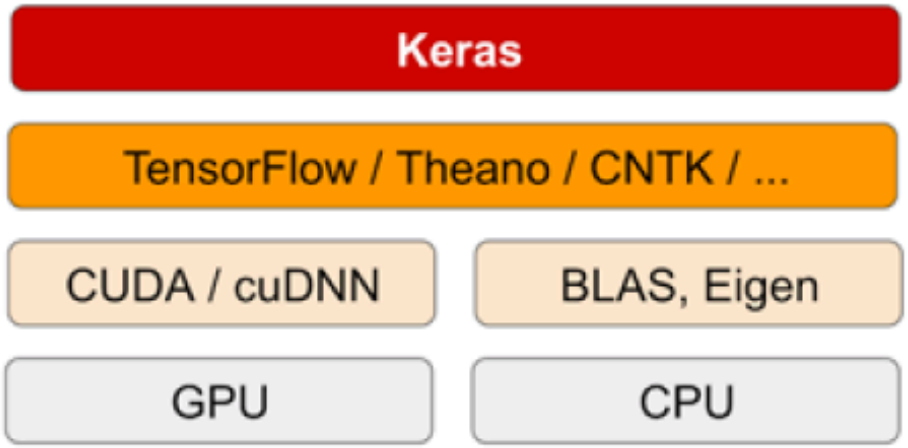
```
Out[16]: [('matter', 0.6234616041183472),
          ('spad', 0.42893028259277344),
          ('issue', 0.4024146795272827),
          ('situation', 0.39529454708099365),
          ('item', 0.3943479061126709),
          ('conversation', 0.39270853996276855),
          ('discussion', 0.39042913913726807),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

Lab 1



Metro passenger flow prediction

- Predict passenger arrival at stations for the next 15 minutes

Contents lists available at ScienceDirect

Transportation Research Part C

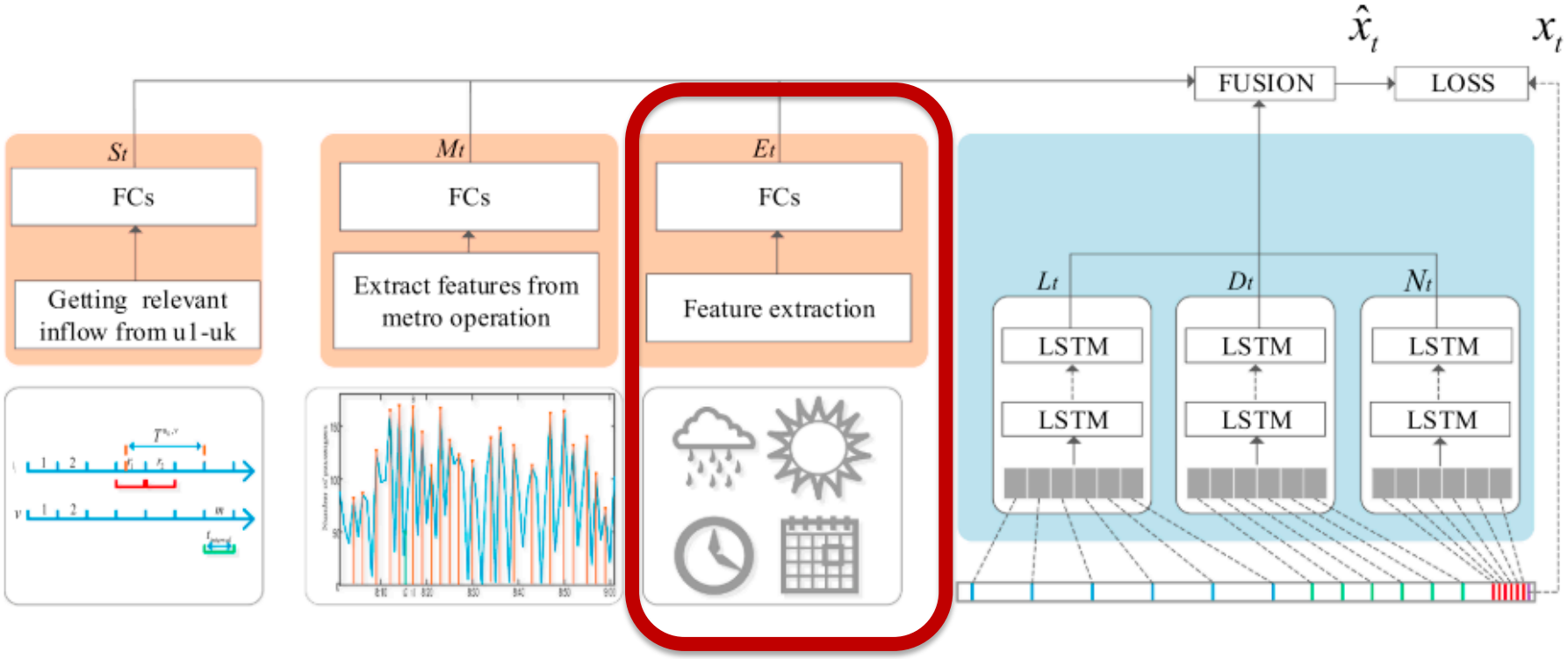
journal homepage: www.elsevier.com/locate/trc

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

Yang Liu, Zhiyuan Liu*, Ruo Jia

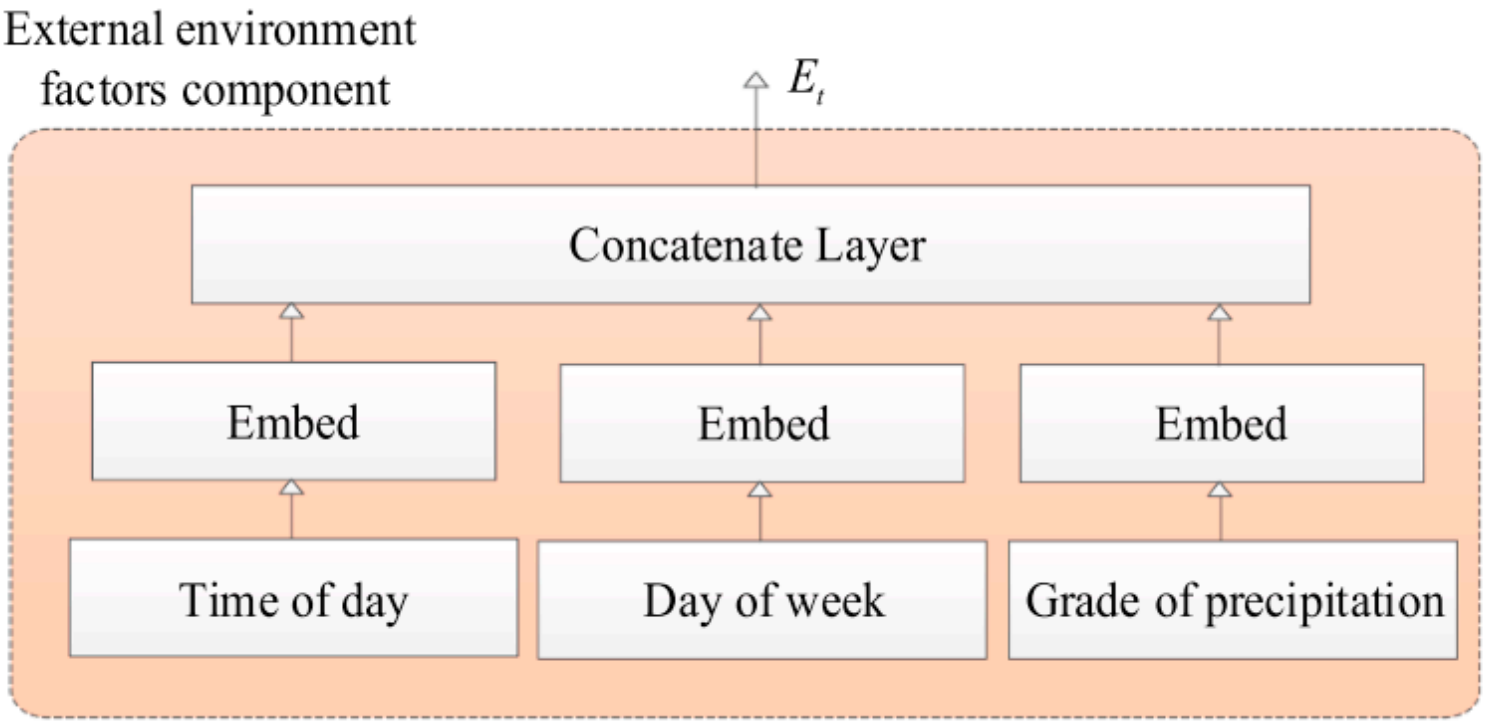
Metro passenger flow prediction

- Predict passenger arrival at stations for the next 15 minutes



Metro passenger flow prediction

- Predict passenger arrival at stations for the next 15 minutes



Demand prediction for bike sharing services

Contents lists available at ScienceDirect

Transportation Research Part C

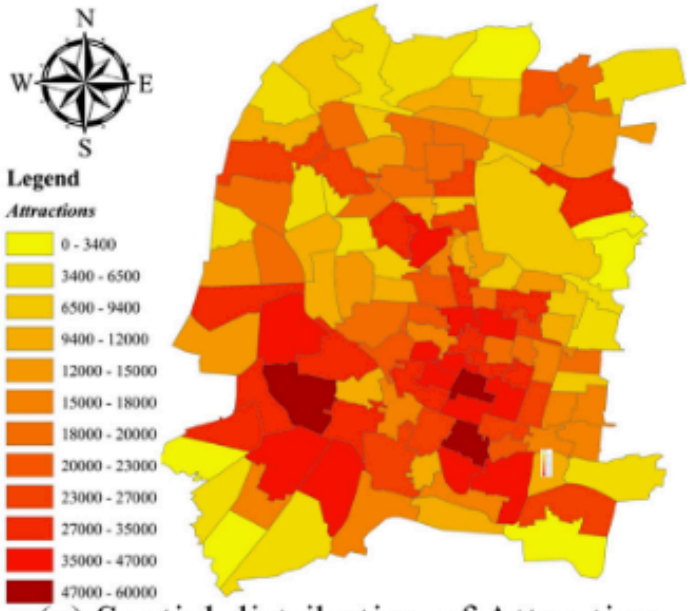
journal homepage: www.elsevier.com/locate/trc

The station-free sharing bike demand forecasting with a deep learning approach and large-scale datasets

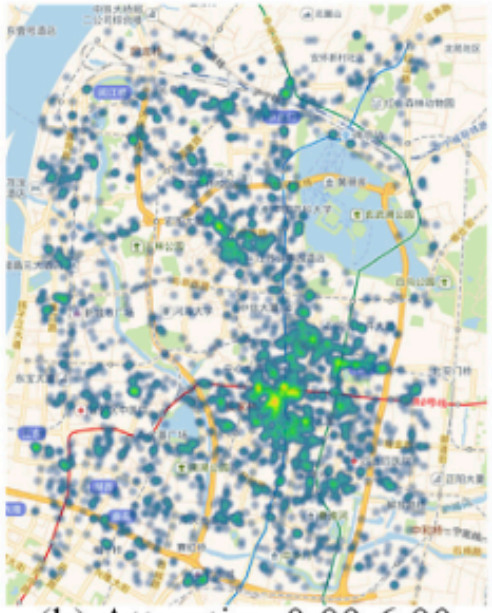
Chengcheng Xu*, Junyi Ji, Pan Liu

(a) Station-free bike sharing

(b) Bike sharing system with docking station



(a) Spatial distribution of Attraction

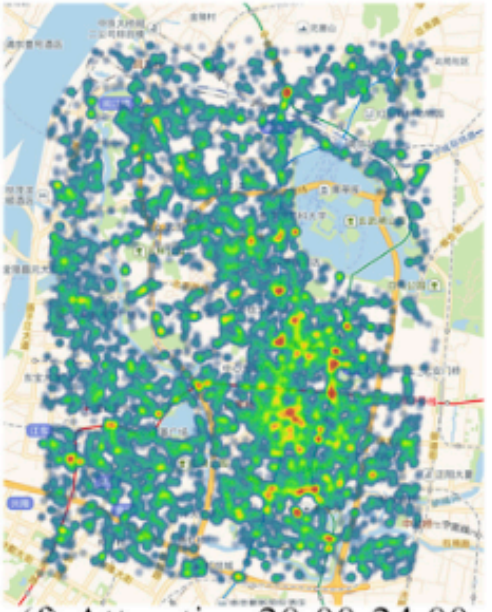


(b) Attraction 0:00-6:00

(c) Attraction 6:00-10:00

(d) Attraction 10:00-15:00

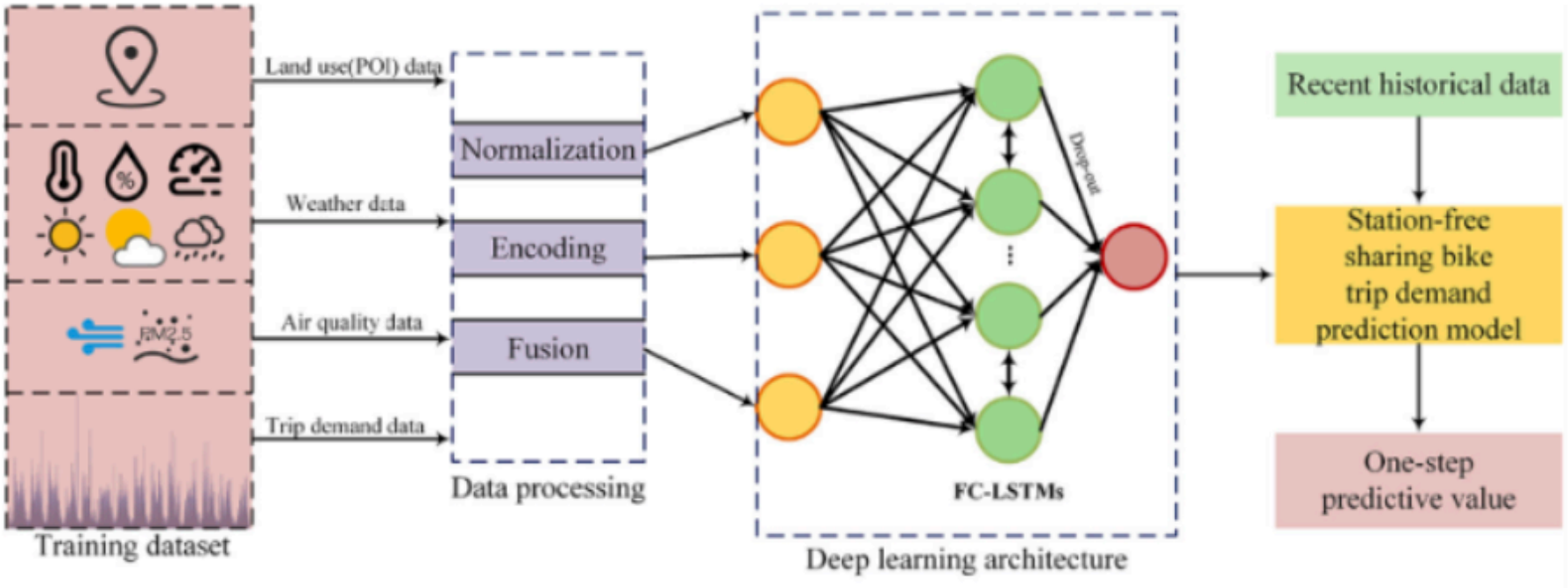
(e) Attraction 15:00-20:00

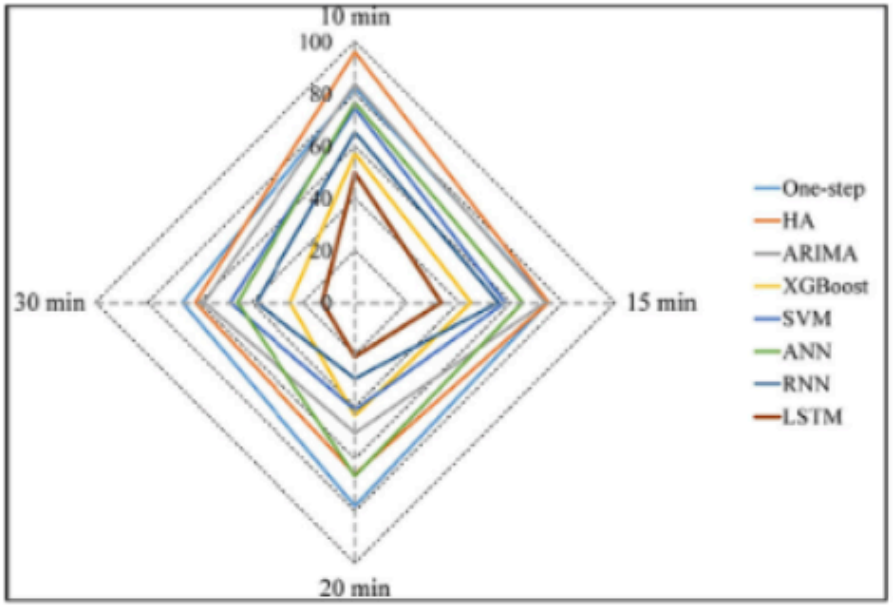


(f) Attraction 20:00-24:00

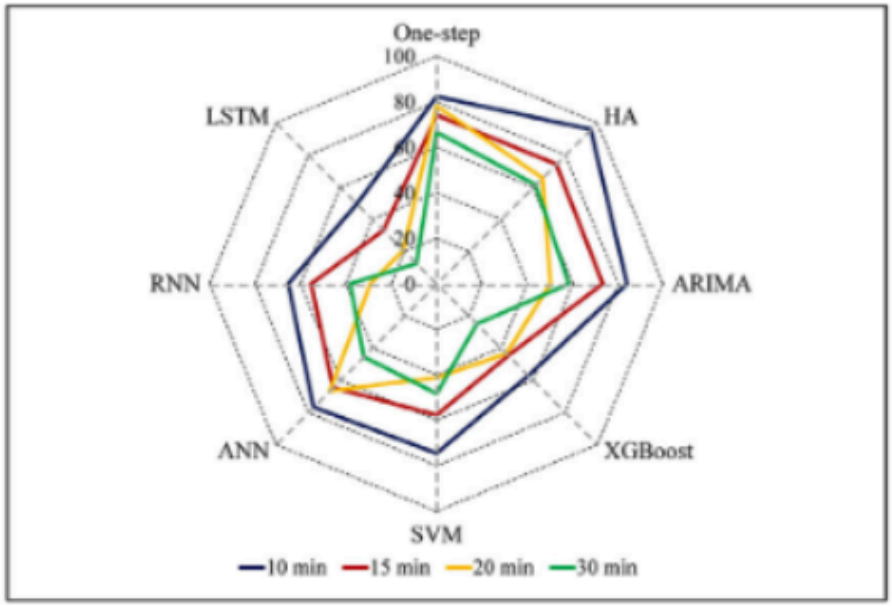
Demand prediction for bike sharing services

- Embedding + LSTM

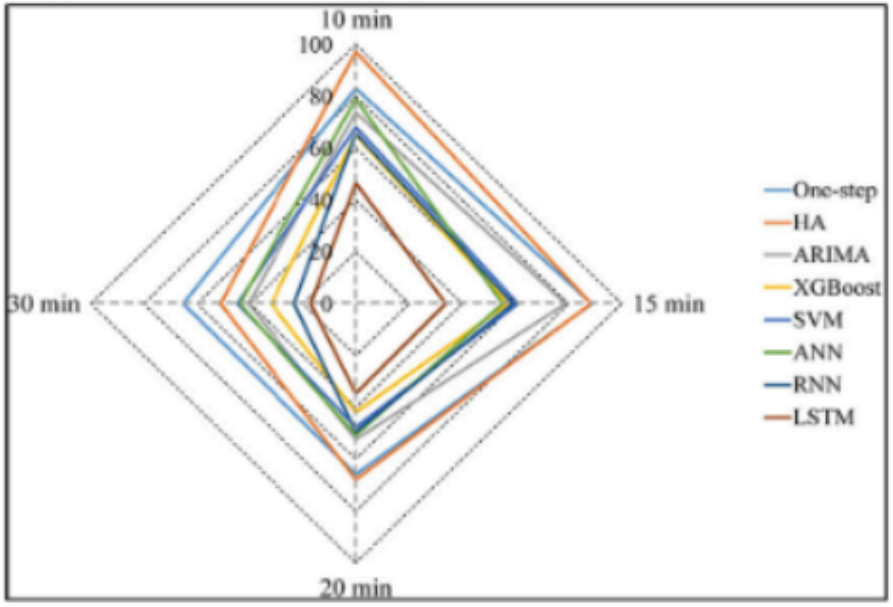




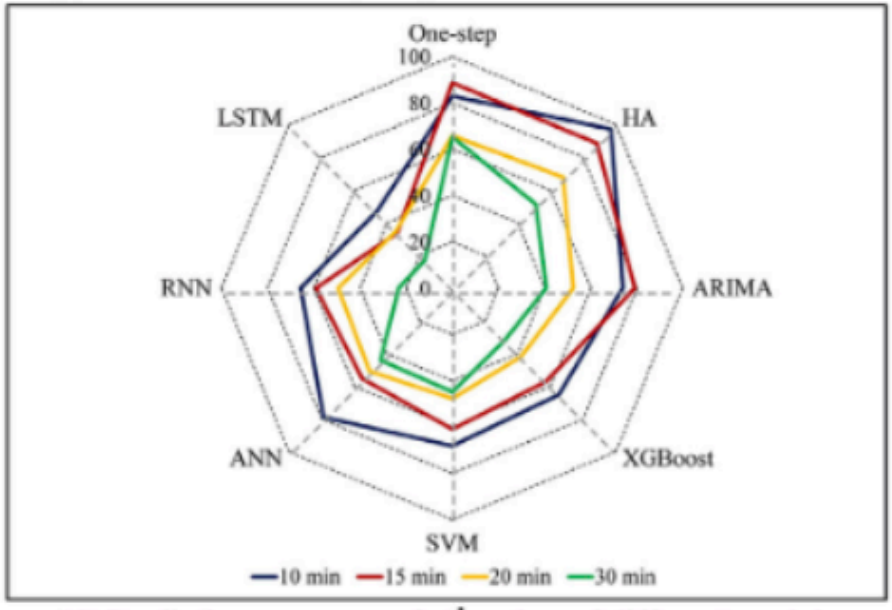
(a) Prediction accuracy of production at different intervals



(b) Prediction accuracy of production of different models



(c) Prediction accuracy of attraction at different intervals



(d) Prediction accuracy of attraction of different models

Mobility Embeddings

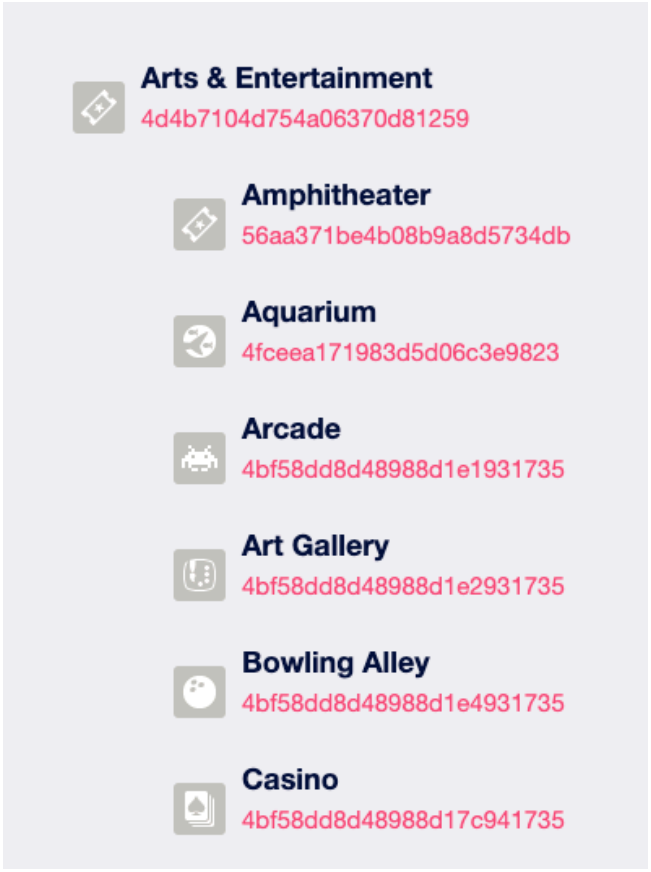
Unsupervised Learning of Parsimonious General-Purpose Embeddings for User and Location Modelling

JING YANG, ETH Zurich, Switzerland
CARSTEN EICKHOFF, ETH Zurich, Switzerland

- Check-in data from Foursquare
- Learn
  - Geographic, temporal, and functional embeddings
- Application
  - Location recommendation
  - Urban functional zone study
  - Crime prediction

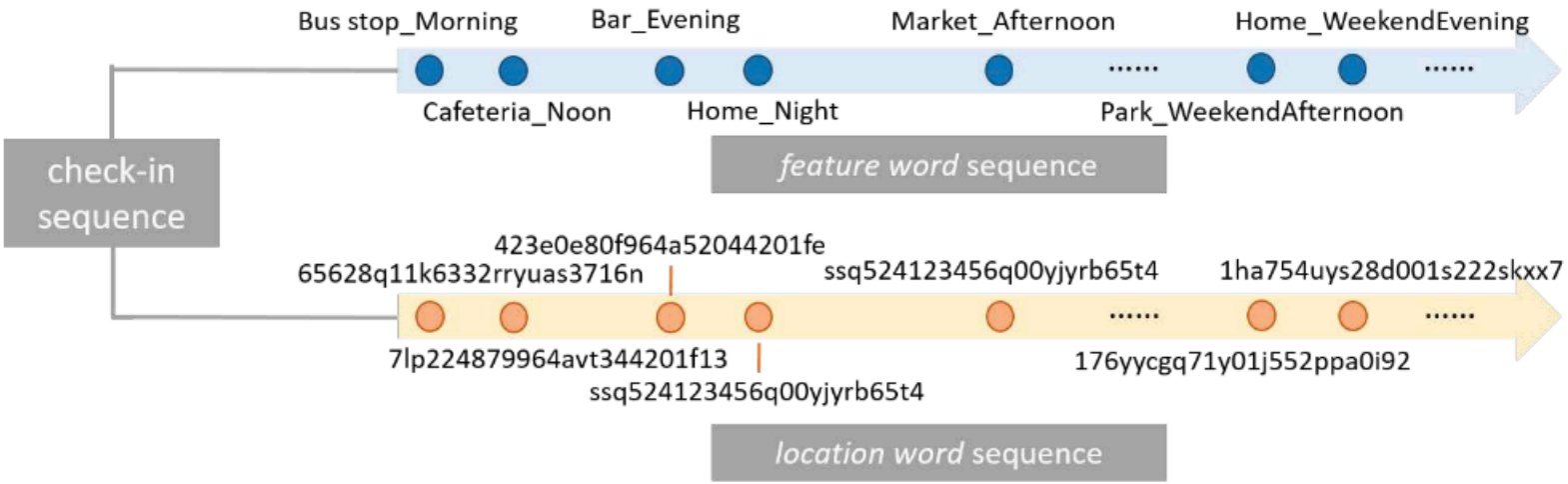
Mobility Embeddings

- Check-in is defines as a tuple $c =< u, f, t, l >$
  - User u
  - Functional role f
    - Restaurant, bar, café, etc.
  - Time
  - Location (ID)



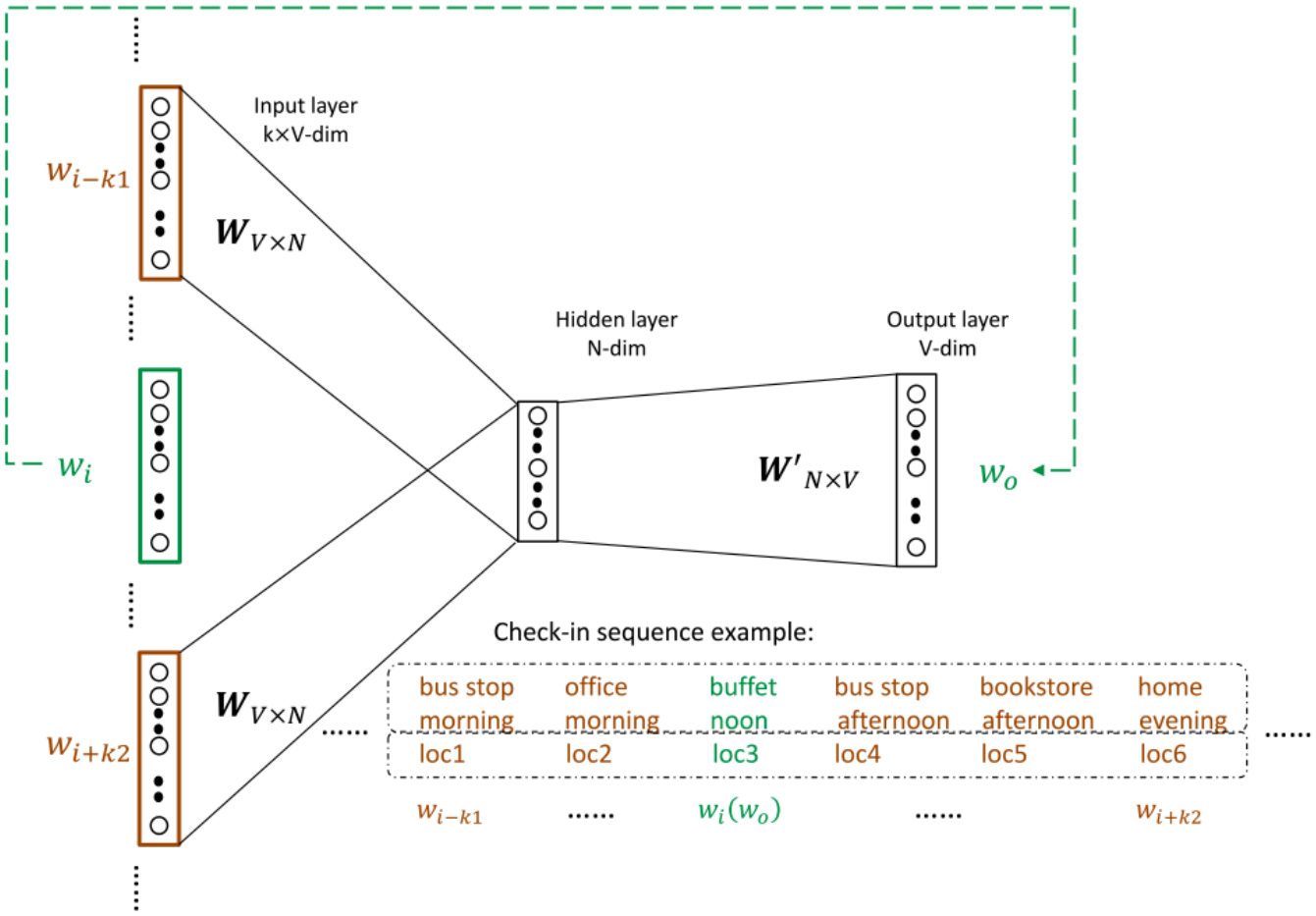
Mobility Embeddings

- Feature-word and Location-word sequences

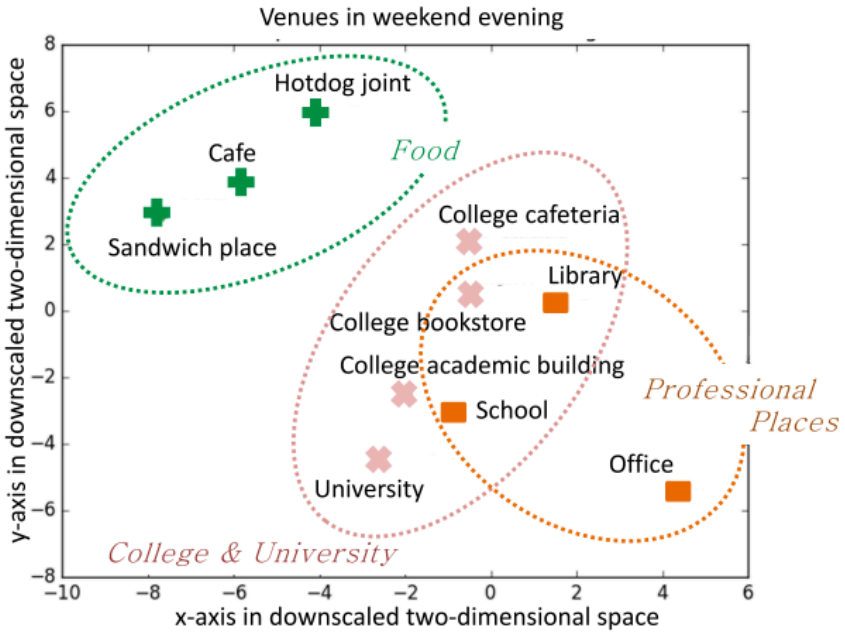


Mobility Embeddings

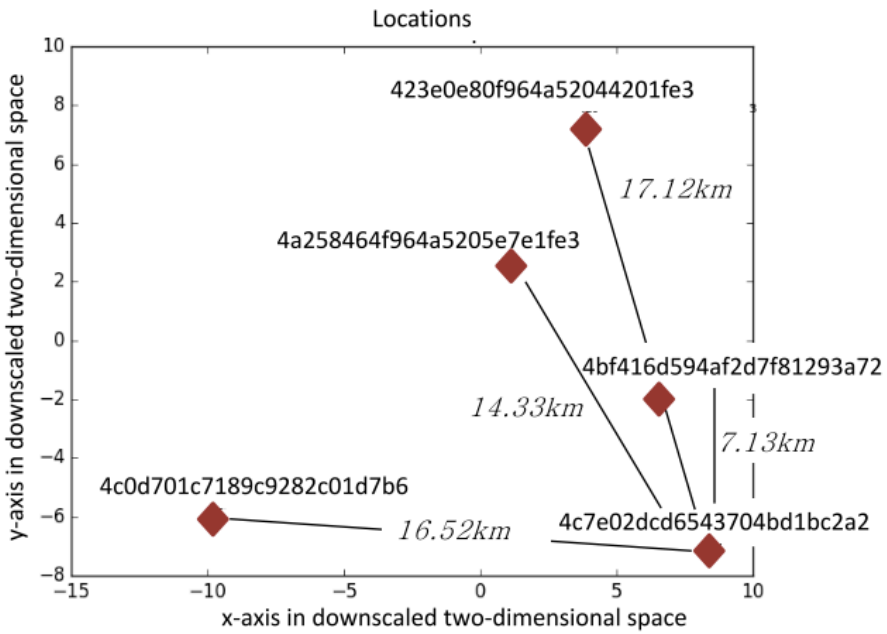
- Feature-word and Location-word sequences



Mobility Embeddings



(a) *feature word* embeddings



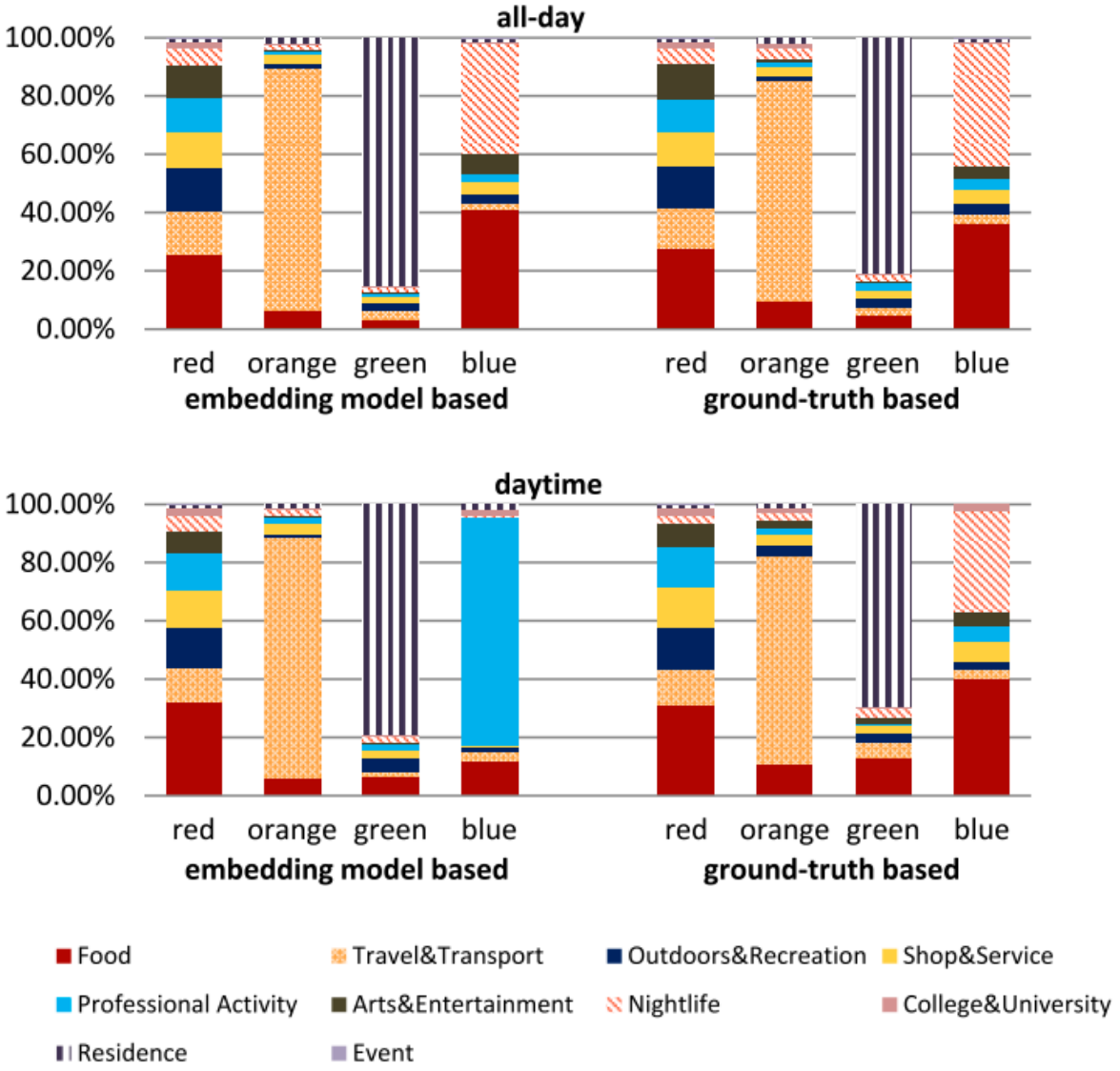
(b) *location word* embeddings

Mobility Embeddings

- Urban functional zone study

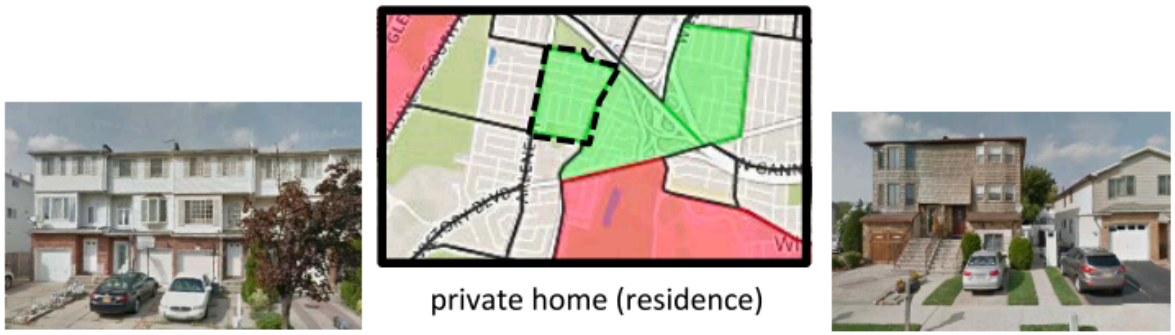
- Partition zones bases on the inhabitants' interactions with urban spaces
- Use feature-word embeddings with second-level venue categories
- Represent each neighborhood using the mean of all contained check-in vectors
- Feed into KNN

Mobility Embeddings



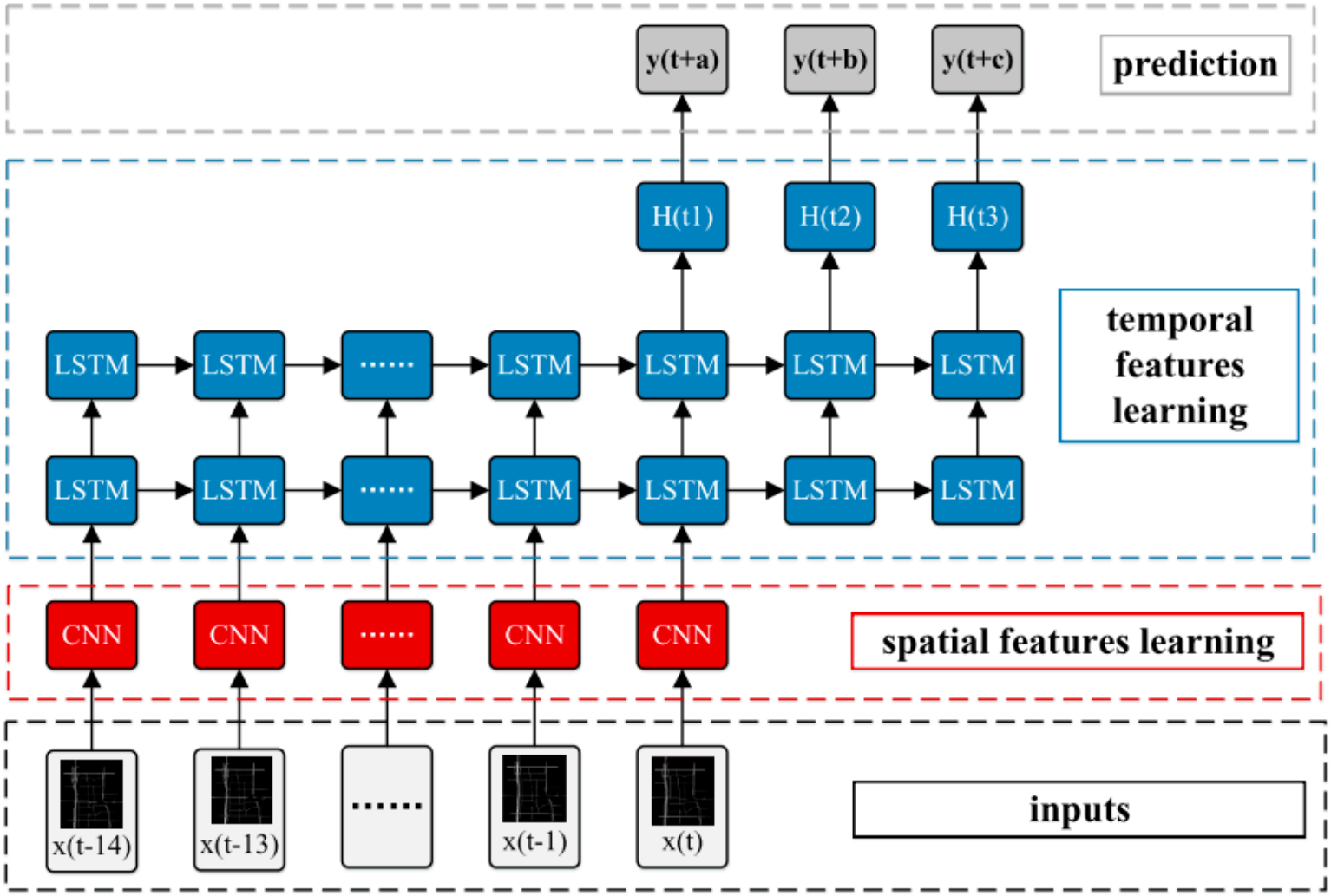
Mobility Embeddings

(a) neighborhood in the day

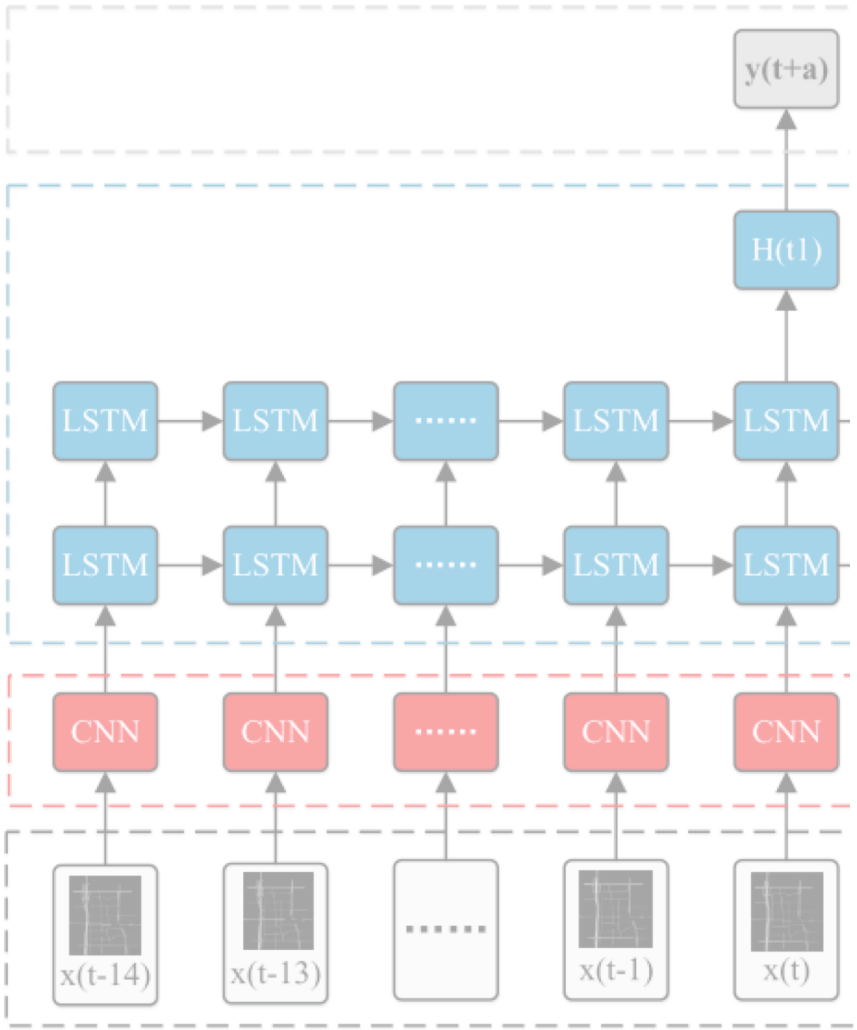


(b) neighborhood in the night

Two issues


y(t+a)
y(t+b)
y(t+c)
prediction
H(t1)
H(t2)
H(t3)
LSTM
temporal features learning
CNN
spatial features learning
x(t-14)
x(t-13)
x(t-1)
x(t)
inputs

Two issues



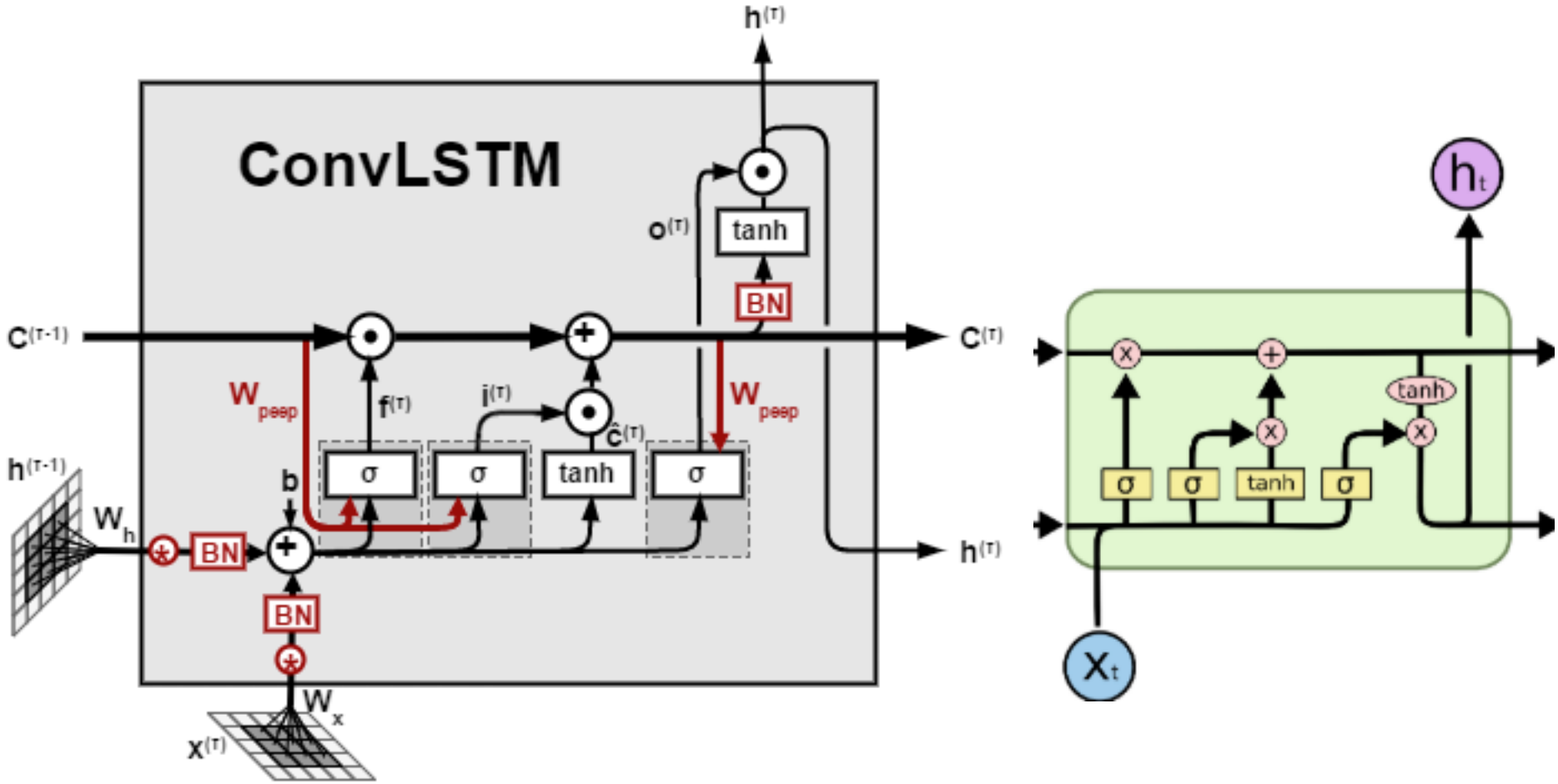
0	Inputs	1	(163,148
1	Convolution	16	(3,3)
	Max-pooling	16	(2,2)
	Activation (relu)	— —	— —
	Batch-normalization	— —	— —
2	Convolution	32	(3,3)
	Max-pooling	32	(2,2)
	Activation (relu)	— —	— —
	Batch-normalization	— —	— —
3	Convolution	64	(3,3)
	Activation (relu)	— —	— —
	Batch-normalization	— —	— —
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.

ConvLSTM


ConvLSTM
h(τ)
o(τ)
tanh
BN
C(τ-1)
C(τ)
W peep
f(τ)
i(τ)
ĉ(τ)
W peep
h(τ-1)
b
σ
σ
tanh
σ
W h
BN
BN
W x
x(τ)
h(τ)
h t
tanh
X t

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.

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

ConvLSTM

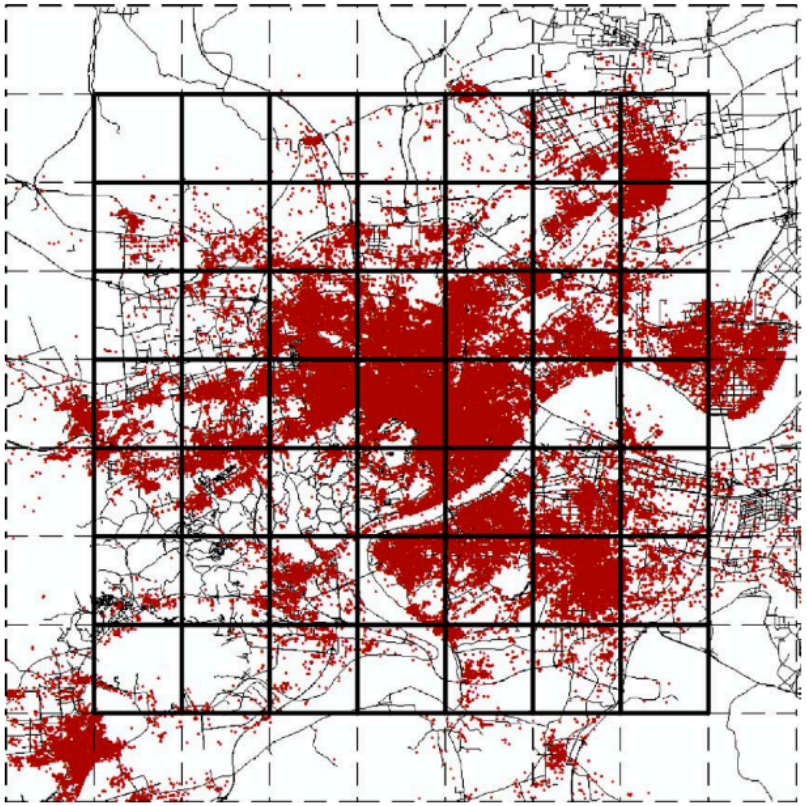
$$
\begin{aligned}
i_t &= \sigma(W_{xi} * X_t + W_{hi} * H_{t-1} + W_{ci} \odot C_{t-1} + b_i) \\
f_t &= \sigma(W_{xf} * X_t + W_{hf} * H_{t-1} + W_{cf} \odot C_{t-1} + b_f) \\
C_t &= f_{tt-1} + i_t \odot tanh(W_{xc} * X_t + + W_{hc} * H_{t-1} + b_c) \\
o_t &= \sigma(W_{xo} * X_t + W_{ho} * H_{t-1} +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

LSTM

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

ConvLSTM

- Demand prediction for ride hailing services
- 1 million requests btw 2015-2016
- 1-hour time intervals
- 4.77 * 4.88 km zones
- Temperature, humidity, weather state, wind speed, and visibility



Contents lists available at ScienceDirect

Transportation Research Part C

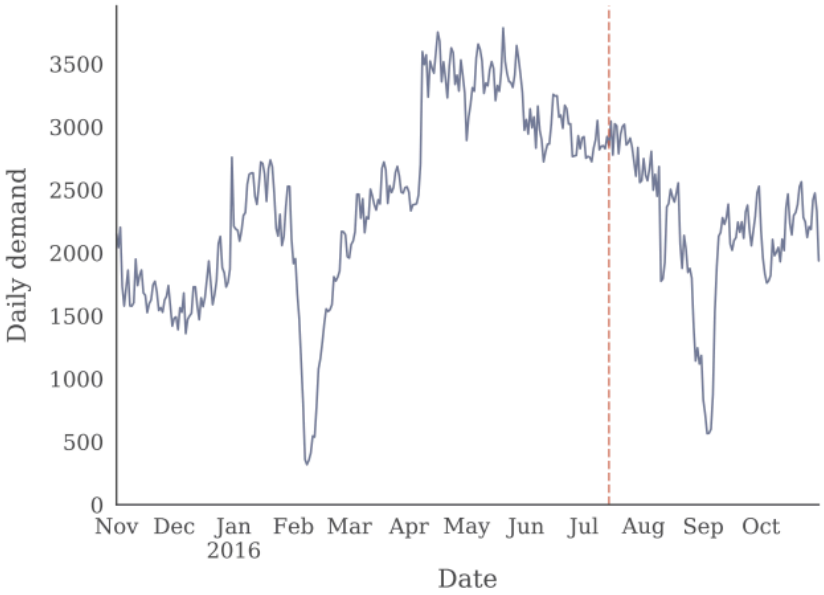
journal homepage: www.elsevier.com/locate/trc

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

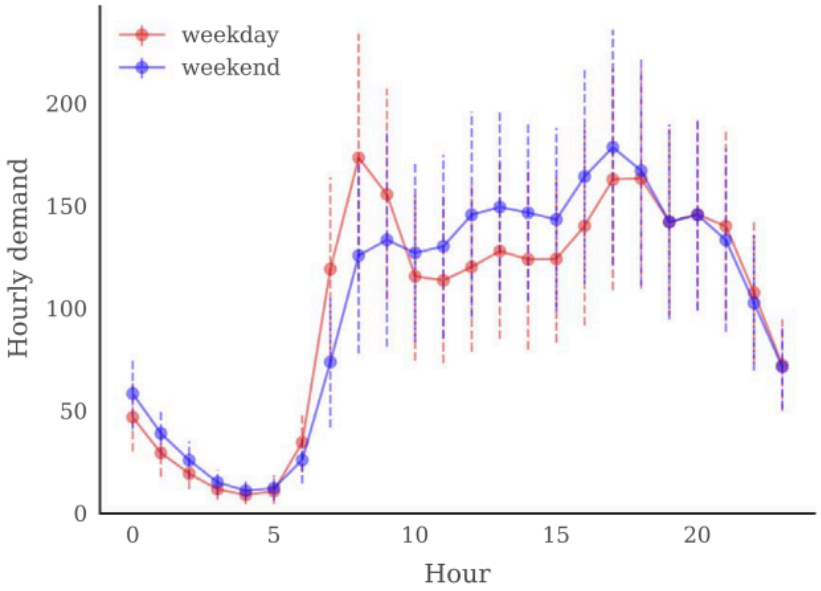
Jintao Ke[a], Hongyu Zheng[b], Hai Yang[a], Xiqun (Michael) Chen[b,*]

ConvLSTM

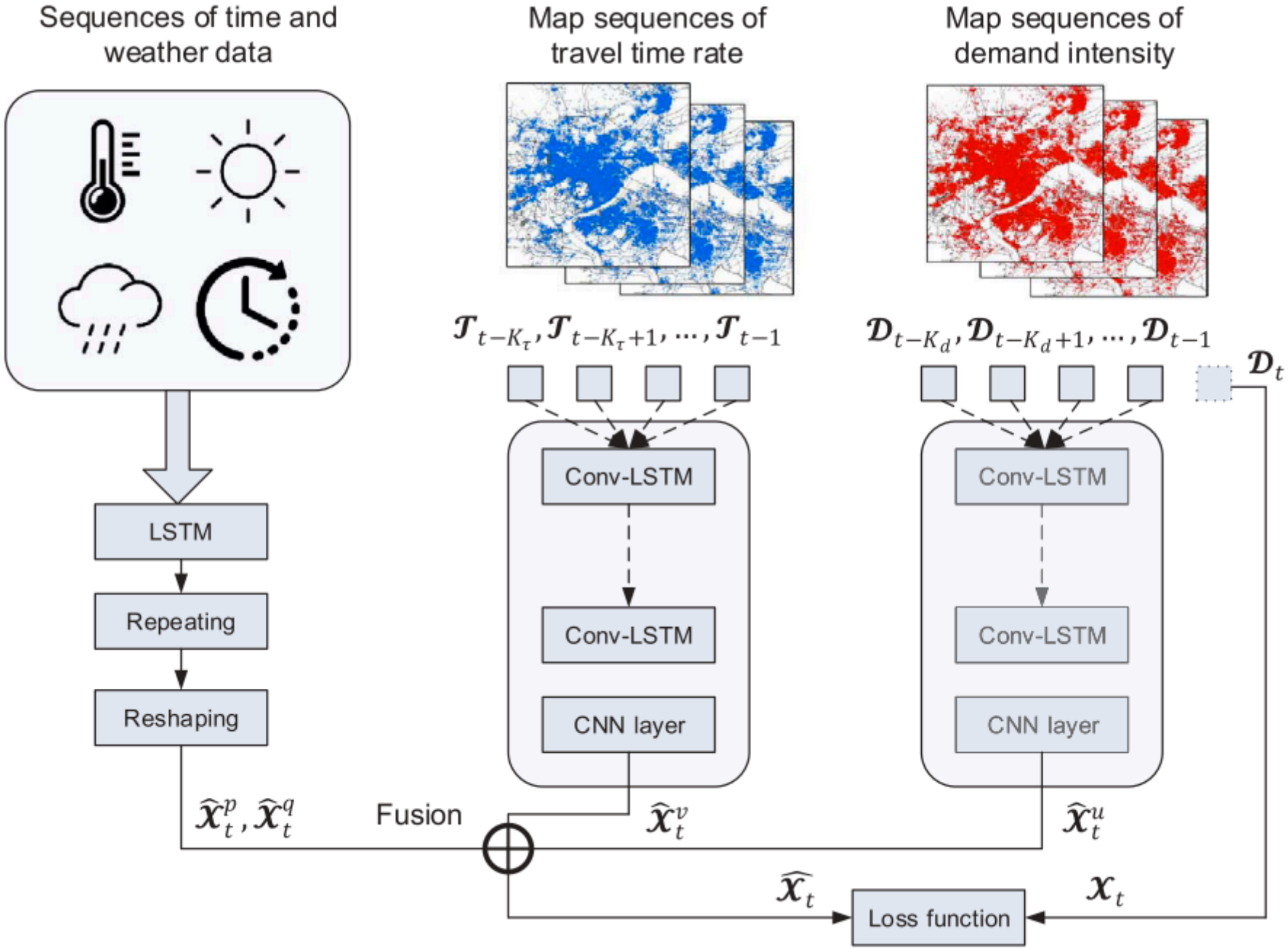
- Demand prediction for ride hailing services



(a) Demand in different days

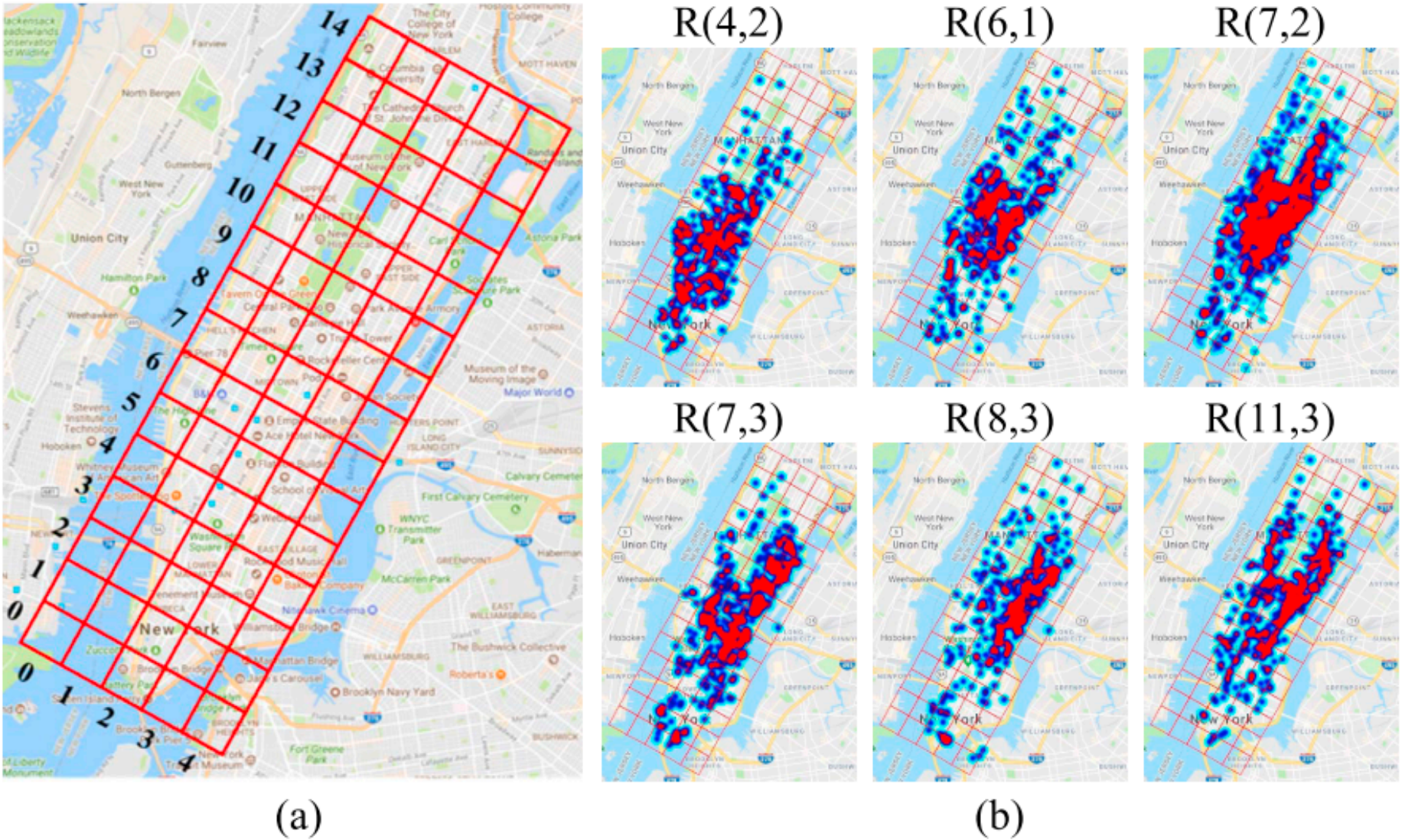


(b) Mean and standard deviation of hourly demand

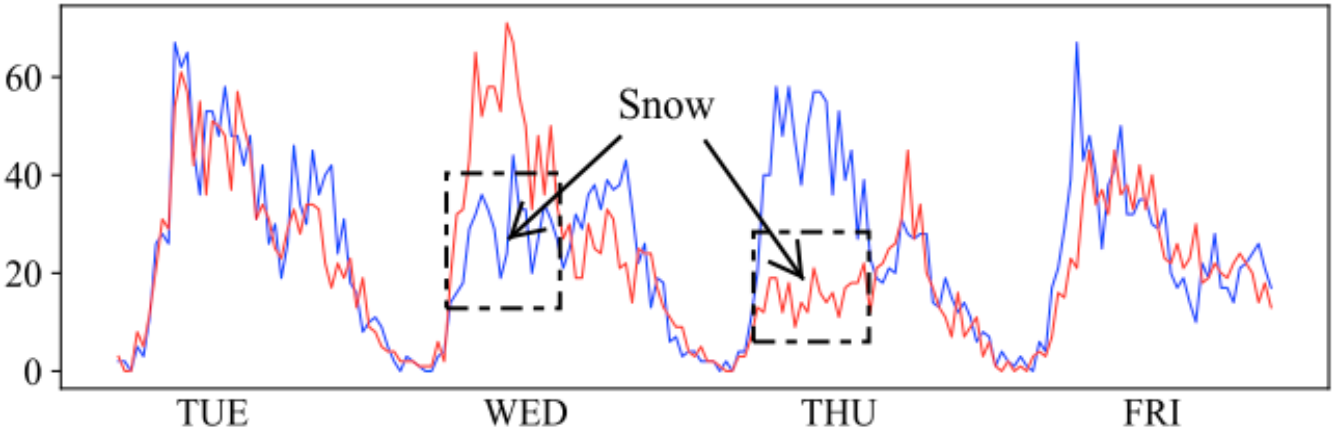

Sequences of time and weather data
Map sequences of travel time rate
Map sequences of demand intensity
$\mathcal{T}_{t-K_\tau}, \mathcal{T}_{t-K_\tau+1}, \ldots, \mathcal{T}_{t-1}$
$\mathcal{D}_{t-K_d}, \mathcal{D}_{t-K_d+1}, \ldots, \mathcal{D}_{t-1}$
$\mathcal{D}_t$
LSTM
Repeating
Reshaping
Conv-LSTM
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

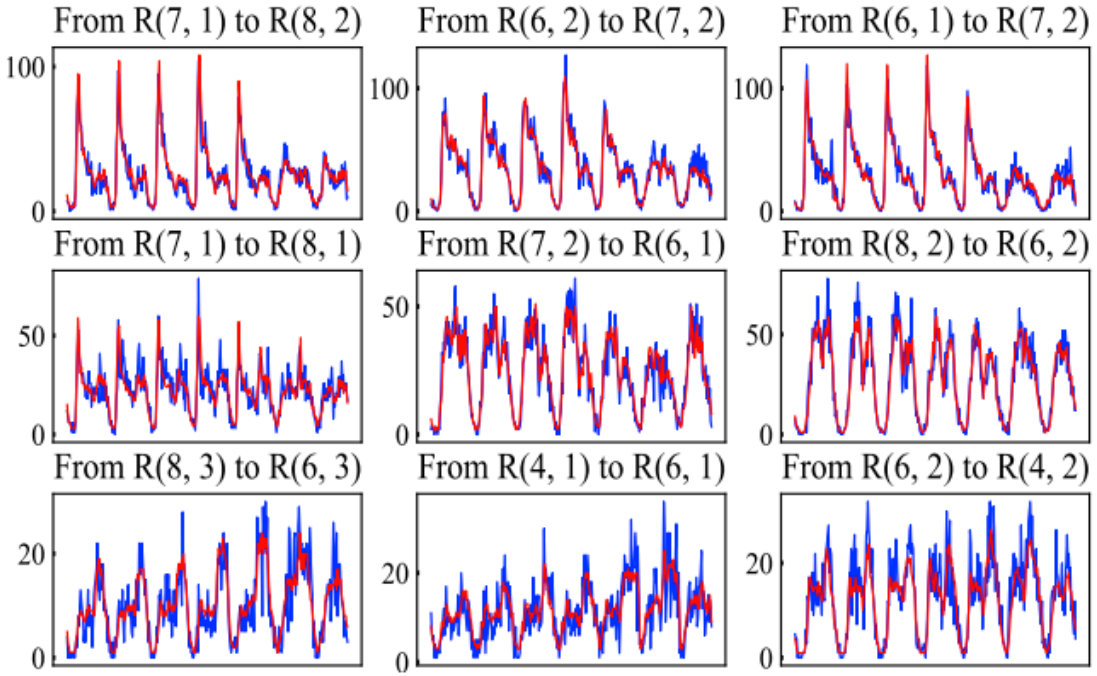
Contextualized Spatial–Temporal Network for Taxi Origin-Destination Demand Prediction

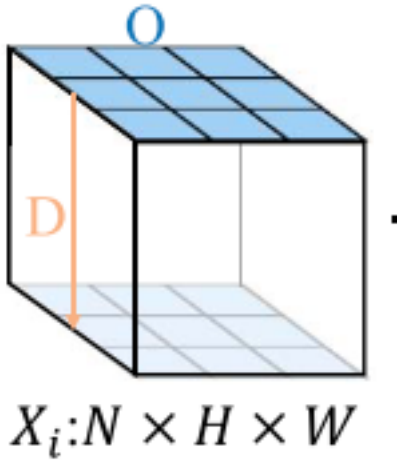
Lingbo Liu, Zhilin Qiu, Guanbin Li, *Member, IEEE*, Qing Wang, Wanli Ouyang, *Senior Member, IEEE*, and Liang Lin, *Senior Member, IEEE*



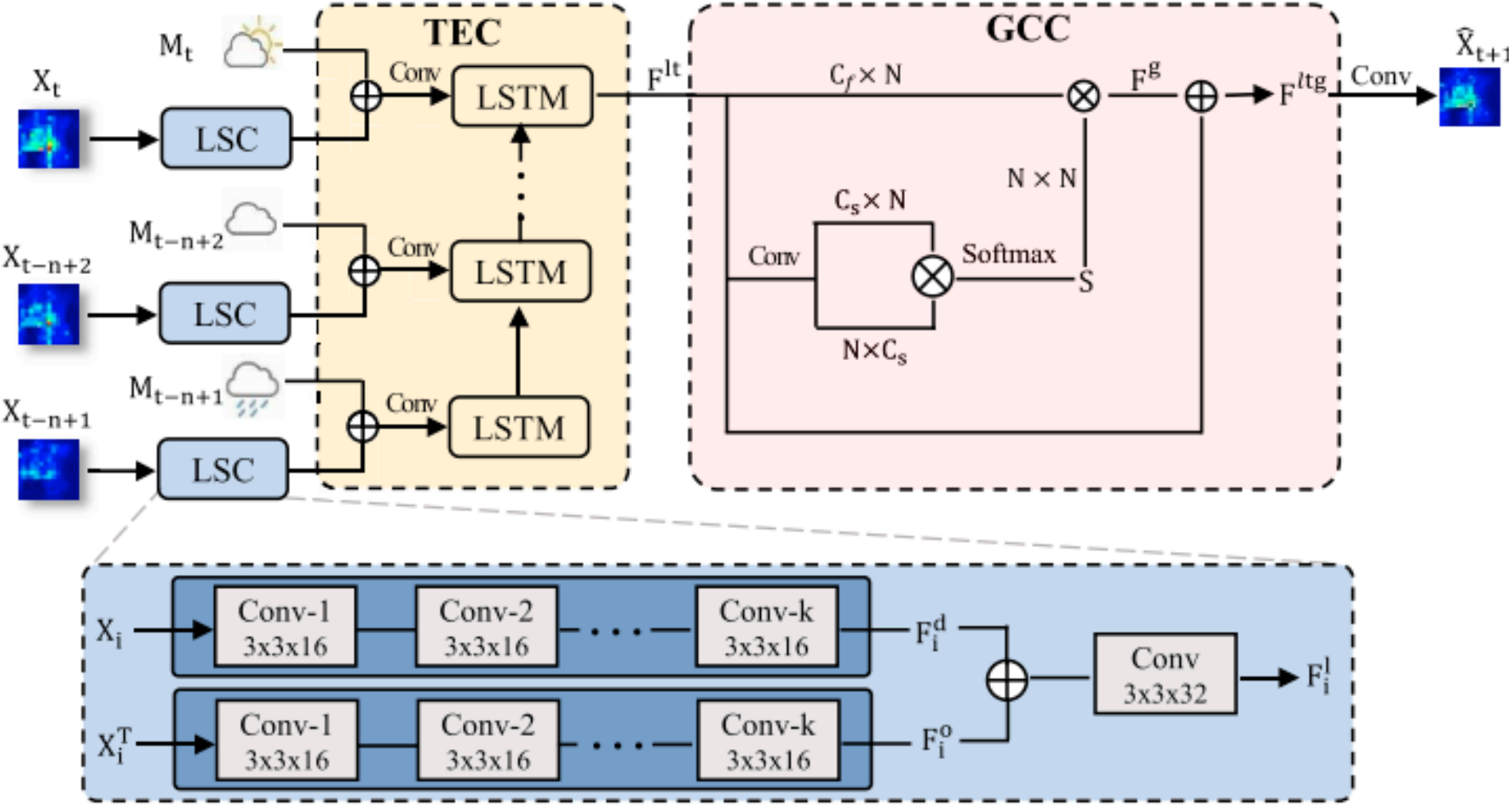
Demand prediction



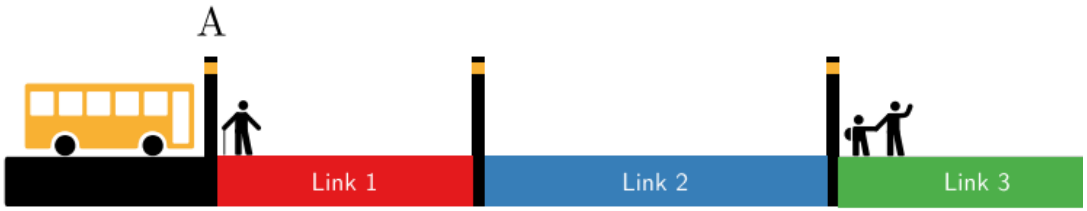




Demand prediction



Bus travel time prediction



- AVL data from buses
- 1,2M travel time observations for the "4A" bus line in the period May to October 2017.
- aggregated into 15-min intervals

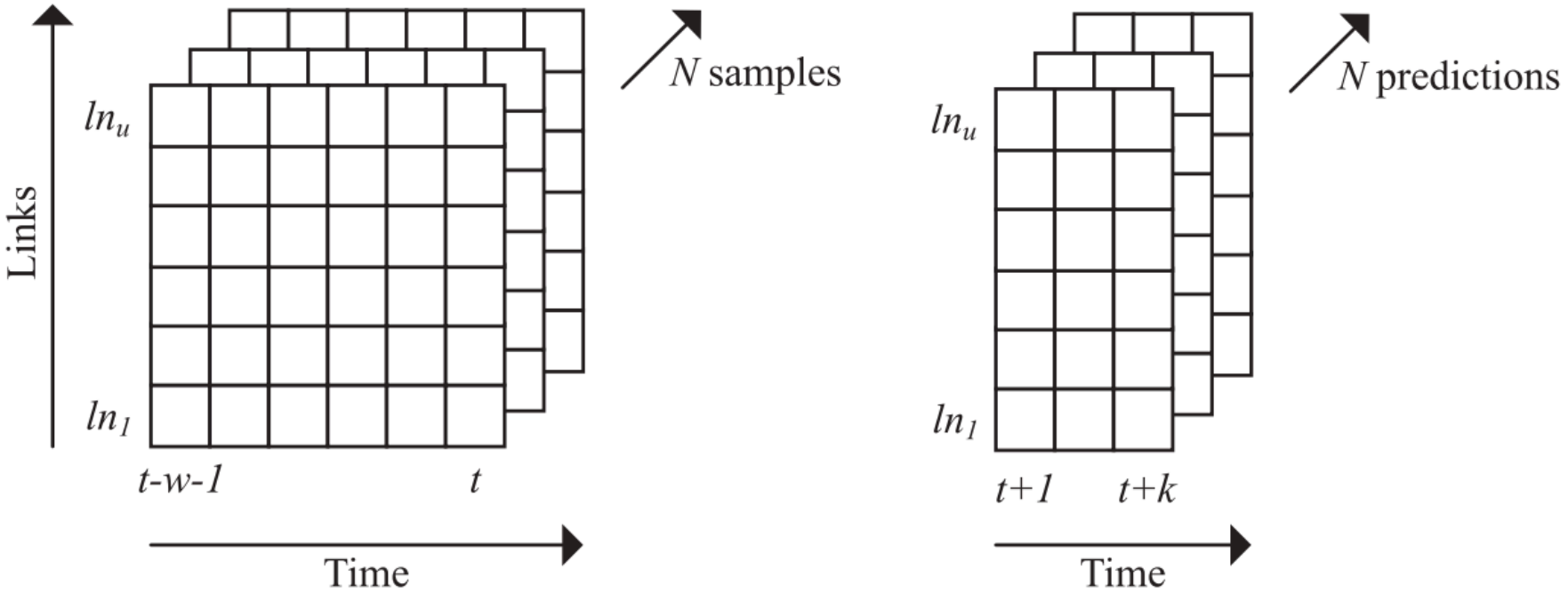
Multi-output bus travel time prediction with convolutional LSTM neural network

Niklas Christoffer Petersen[a,b,*], Filipe Rodrigues[b], Francisco Camara Pereira[b]

Bus travel time prediction

Table 1
Example of raw travel time measurements.

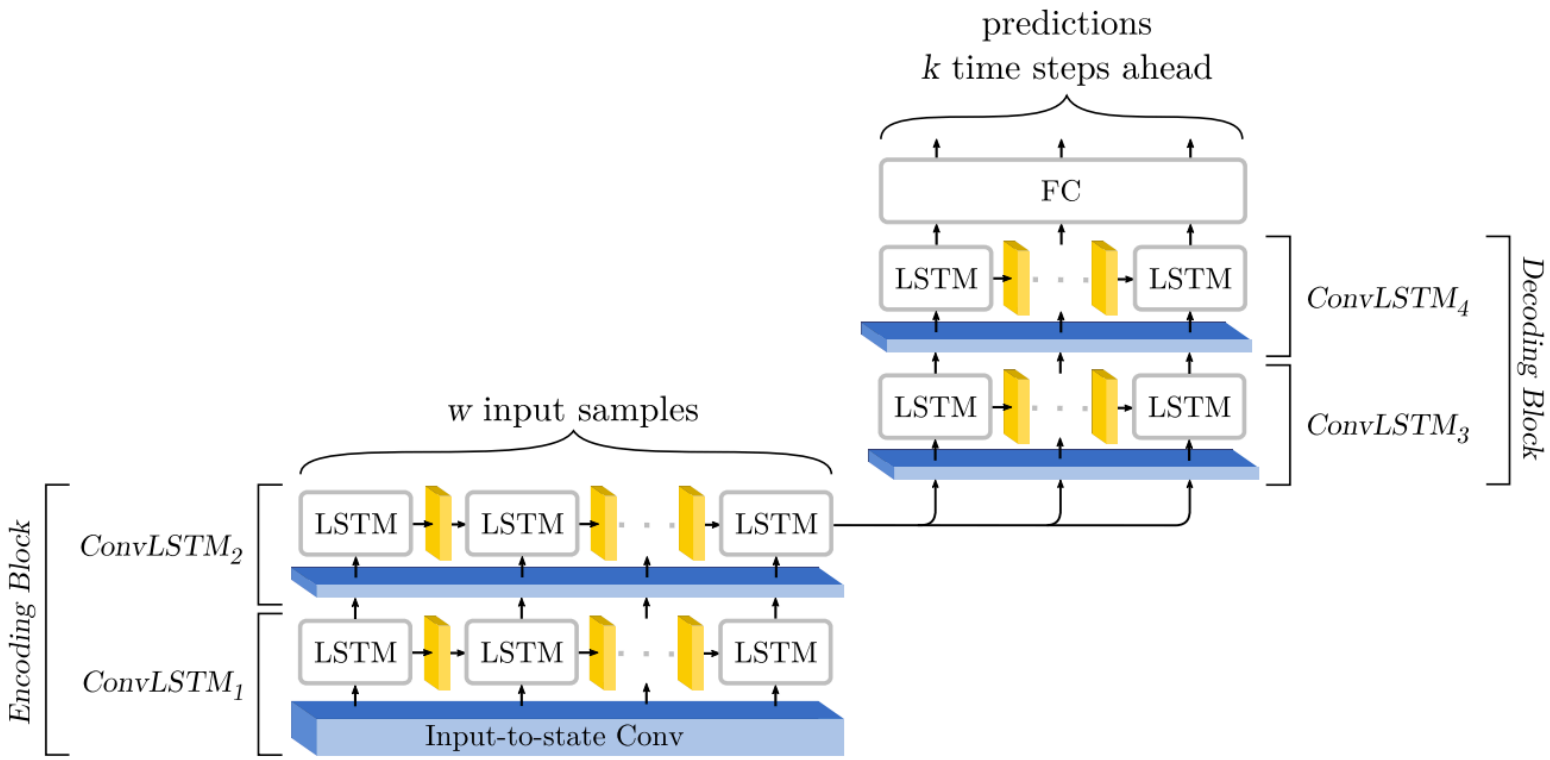
Timestamp	Linkref.	Link travel time (s)
2017-10-10 00:20:02	29848:1254	63
2017-10-10 00:21:07	1254:1255	65
2017-10-10 00:21:51	1255:10115	44
⋮	⋮	⋮



Bus travel time prediction

Table 1
Example of raw travel time measurements.

Timestamp	Linkref.	Link travel time (s)
2017-10-10 00:20:02	29848:1254	63
2017-10-10 00:21:07	1254:1255	65
2017-10-10 00:21:51	1255:10115	44
⋮	⋮	⋮

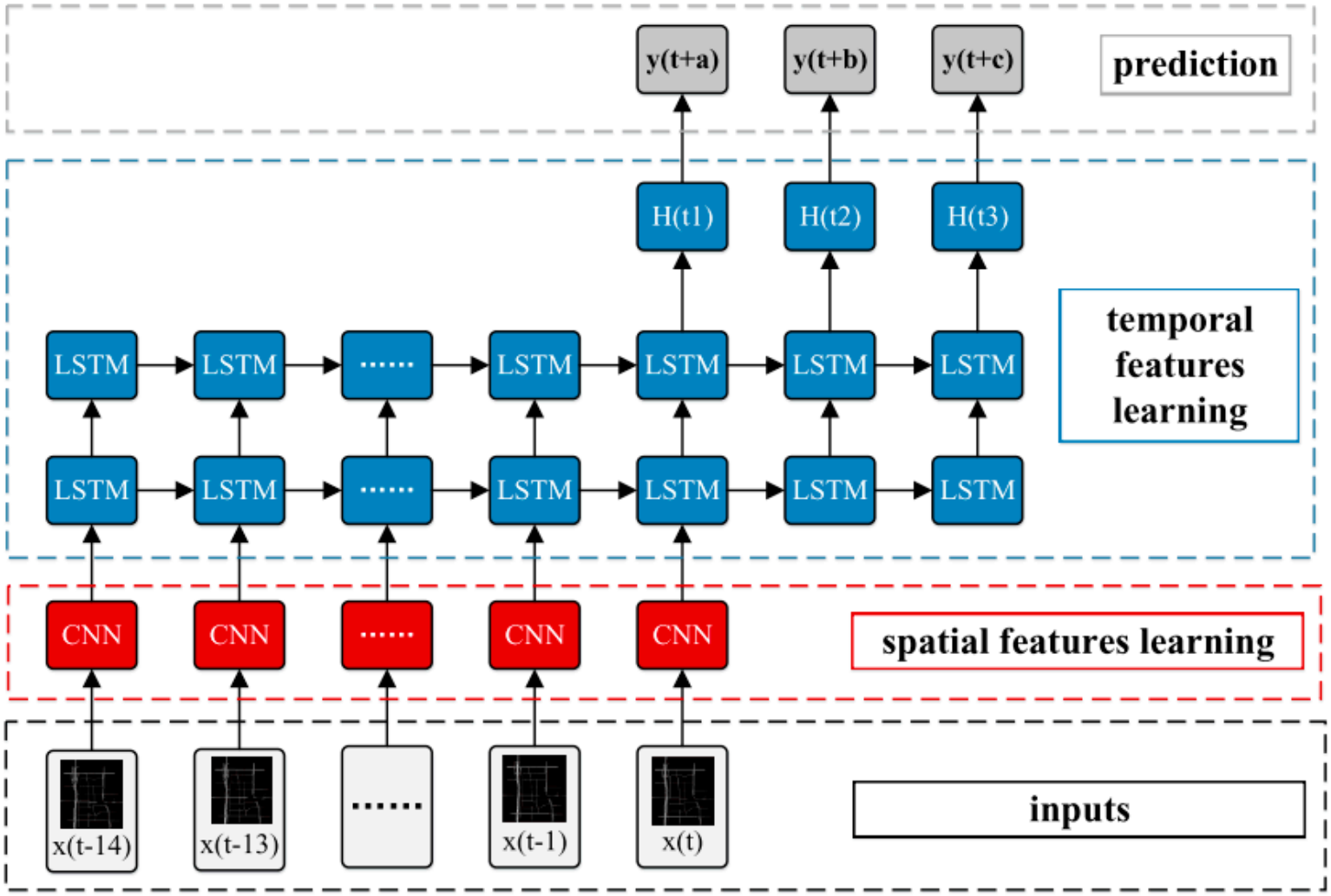


Bus travel time prediction

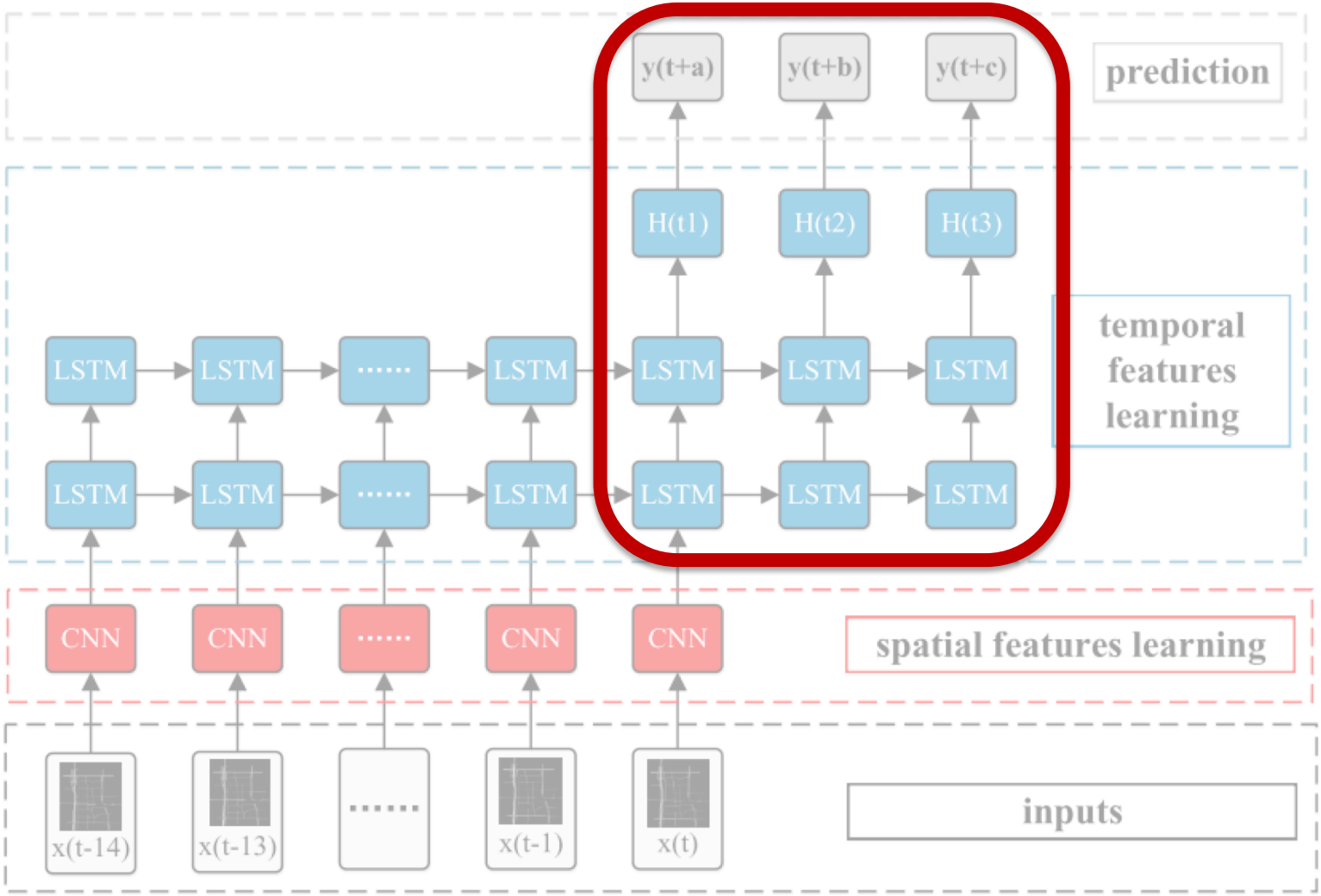
Table 2
Results of the proposed and the baseline models.

Model	Time ahead	RMSE (min)	MAE (min)	MAPE (%)
Historical average		4.35	3.23	6.51%
Current model	$t+1$ (15 min)	4.92	3.90	8.05%
	$t+2$ (30 min)	4.91	3.46	6.82%
	$t+3$ (45 min)	5.47	4.15	8.68%
Pure LSTM	$t+1$ (15 min)	3.48	2.48	5.02%
	$t+2$ (30 min)	3.56	2.51	5.08%
	$t+3$ (45 min)	3.68	2.62	5.34%
Google Traffic	$t+1$ (15 min)	3.67	2.96	6.32%
ConvLSTM	$t+1$ (15 min)	2.66	1.99	4.19%
	$t+2$ (30 min)	2.89	2.11	4.44%
	$t+3$ (45 min)	3.11	2.27	4.75%

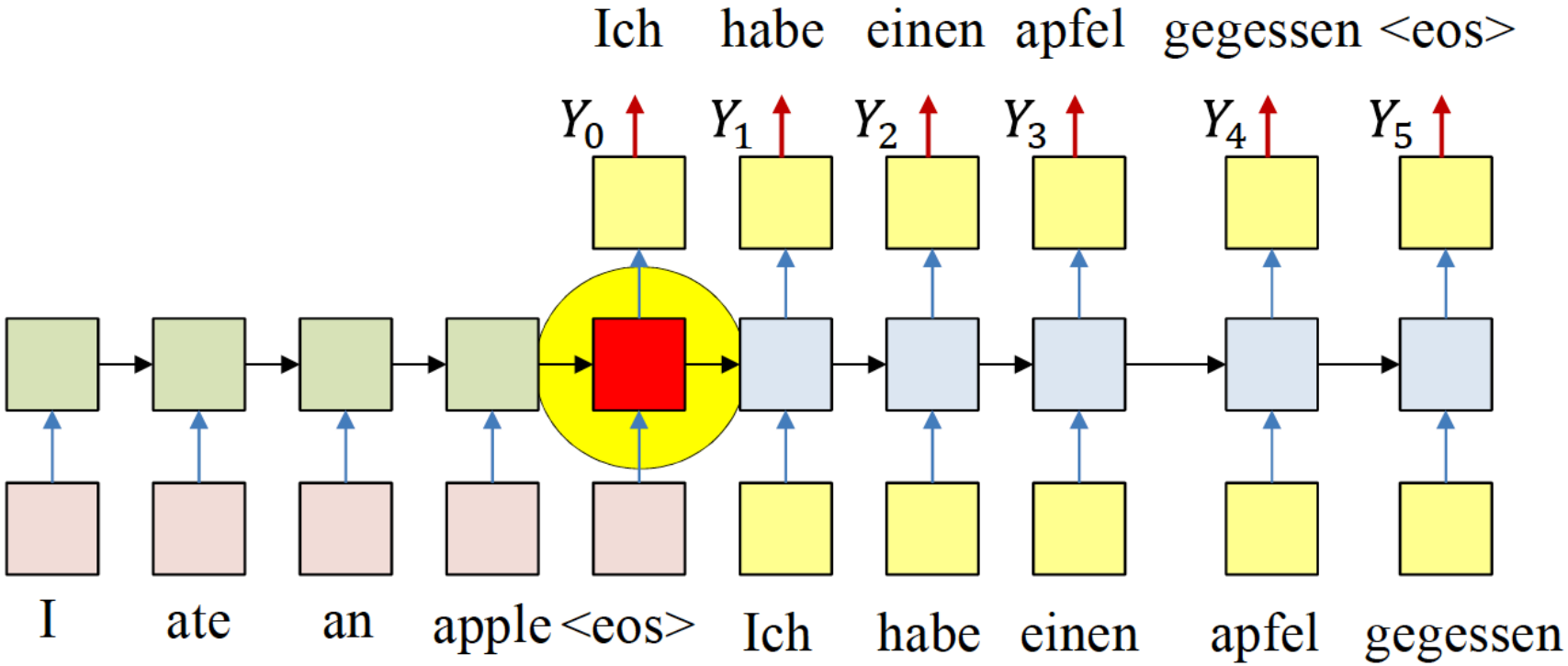
Two issues


y(t+a)
y(t+b)
y(t+c)
prediction
H(t1)
H(t2)
H(t3)
LSTM
temporal features learning
CNN
spatial features learning
x(t-14)
x(t-13)
x(t-1)
x(t)
inputs

Two issues

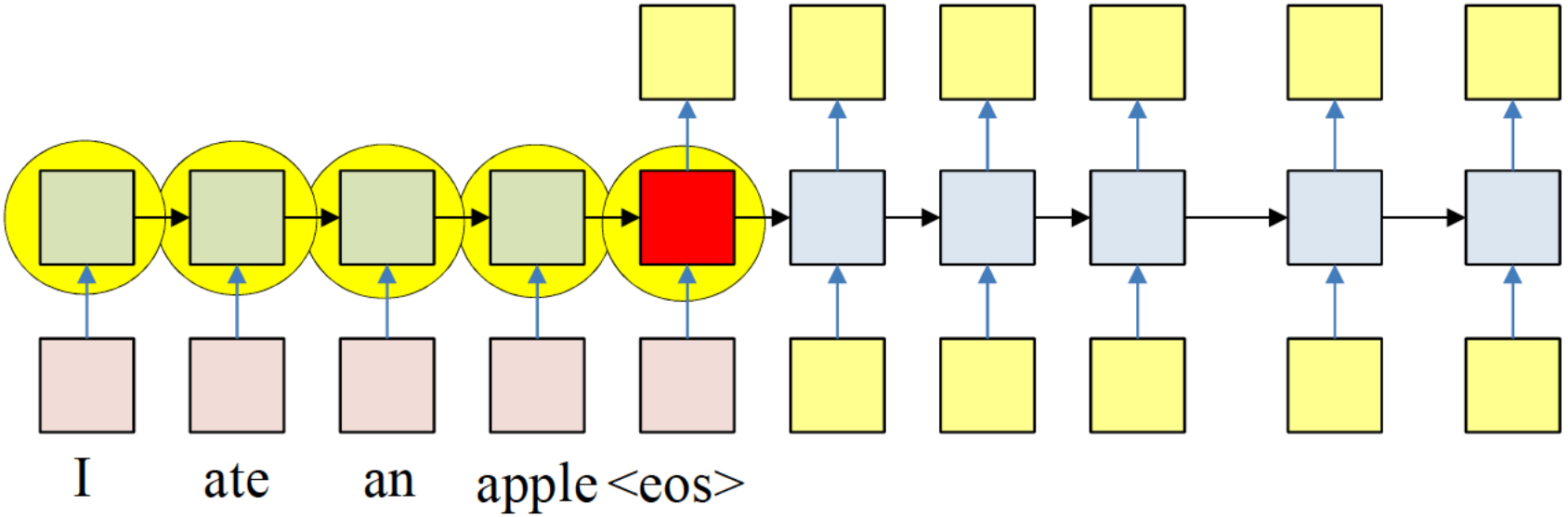

y(t+a)
y(t+b)
y(t+c)
prediction
H(t1)
H(t2)
H(t3)
LSTM
LSTM
.....
LSTM
LSTM
LSTM
LSTM
temporal features learning
LSTM
LSTM
.....
LSTM
LSTM
LSTM
LSTM
CNN
CNN
.....
CNN
CNN
spatial features learning
x(t-14)
x(t-13)
.....
x(t-1)
x(t)
inputs

Neural machine translation



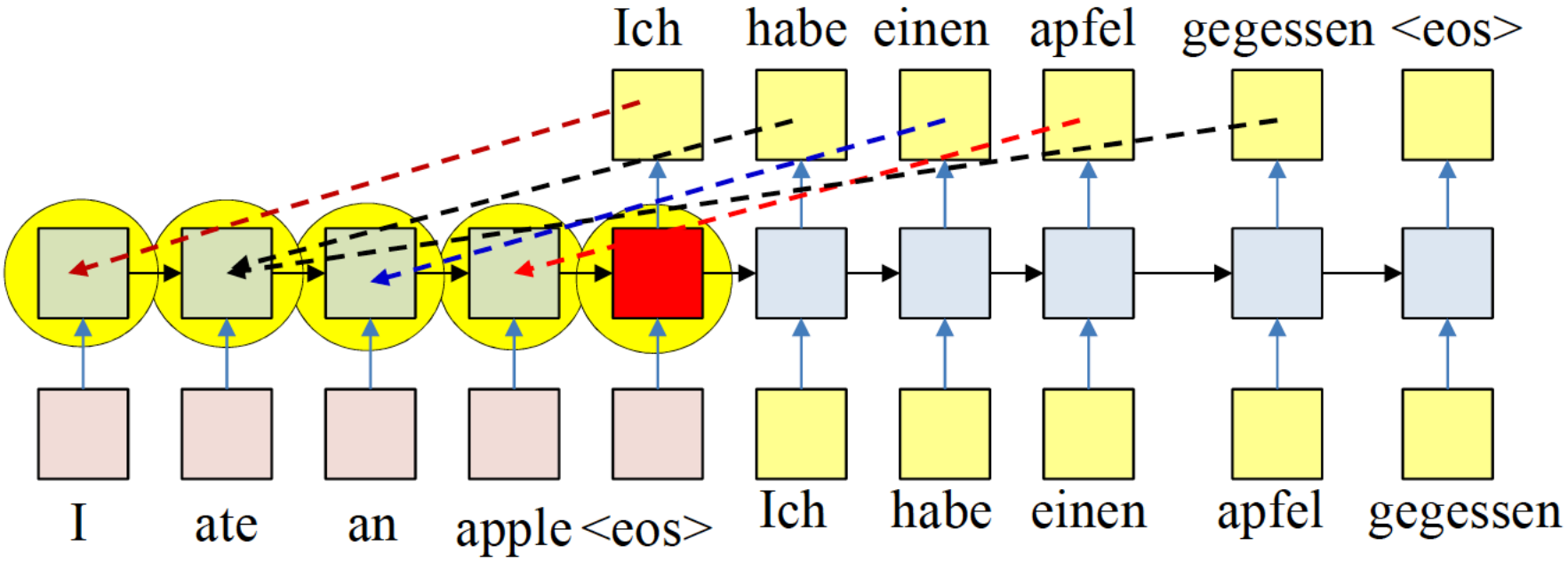
Source: Bhiksha Raj

Neural machine translation



Source: Bhiksha Raj

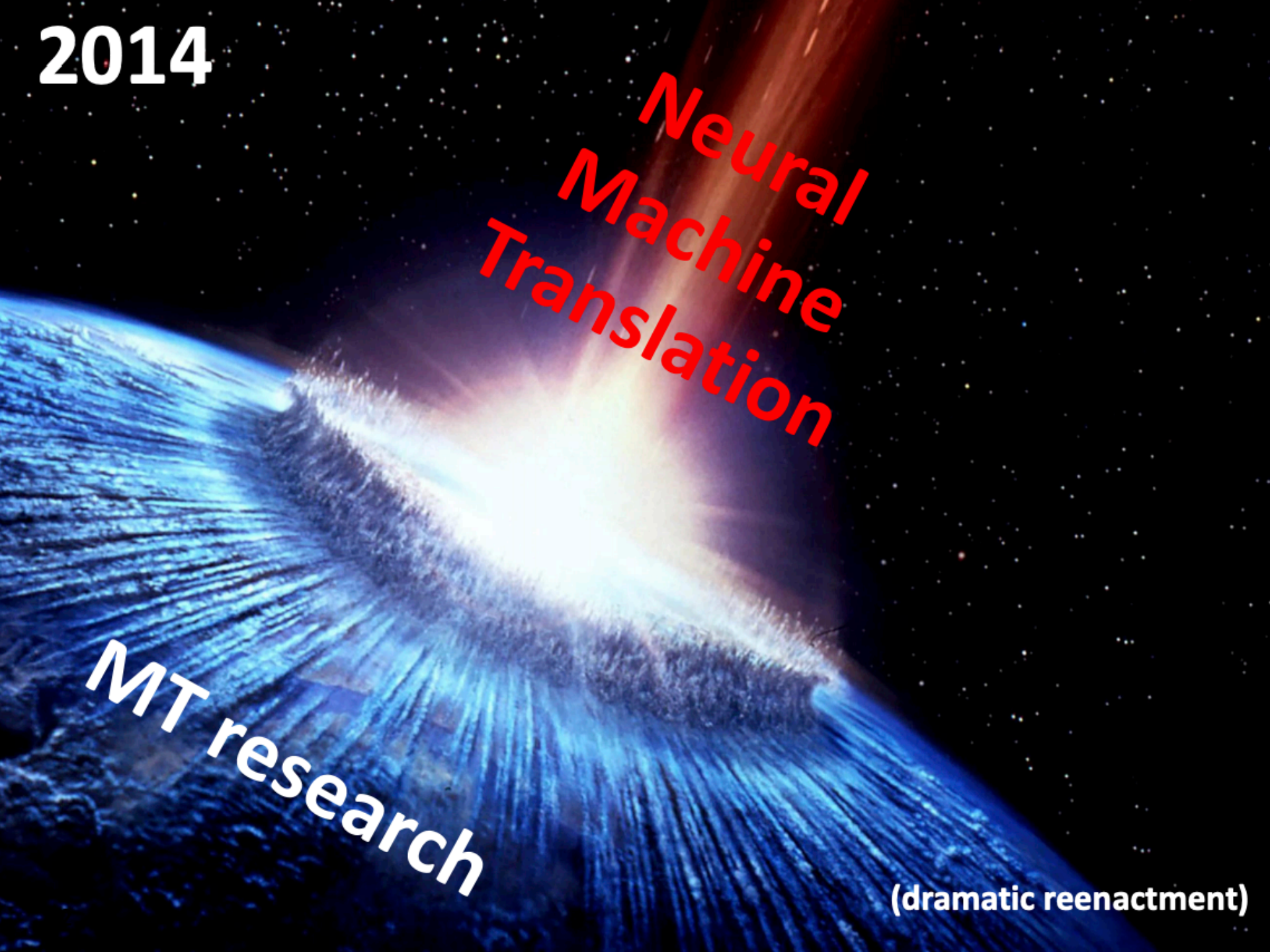
Neural machine translation



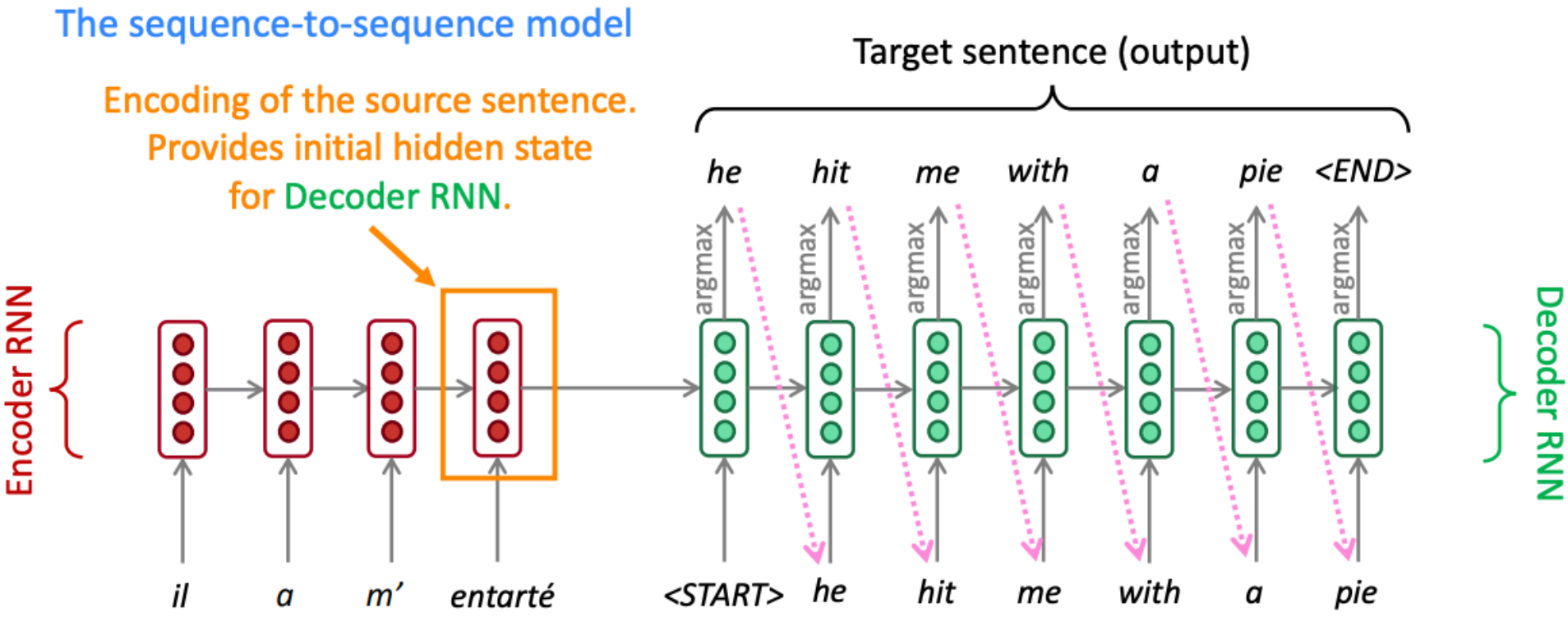
Source: Bhiksha Raj

2014

(dramatic reenactment)


2014
Neural
Machine
Translation
MT research
(dramatic reenactment)

Neural machine translation



Source: Abigail See

Attention

- Attend to different parts of the input based on the output

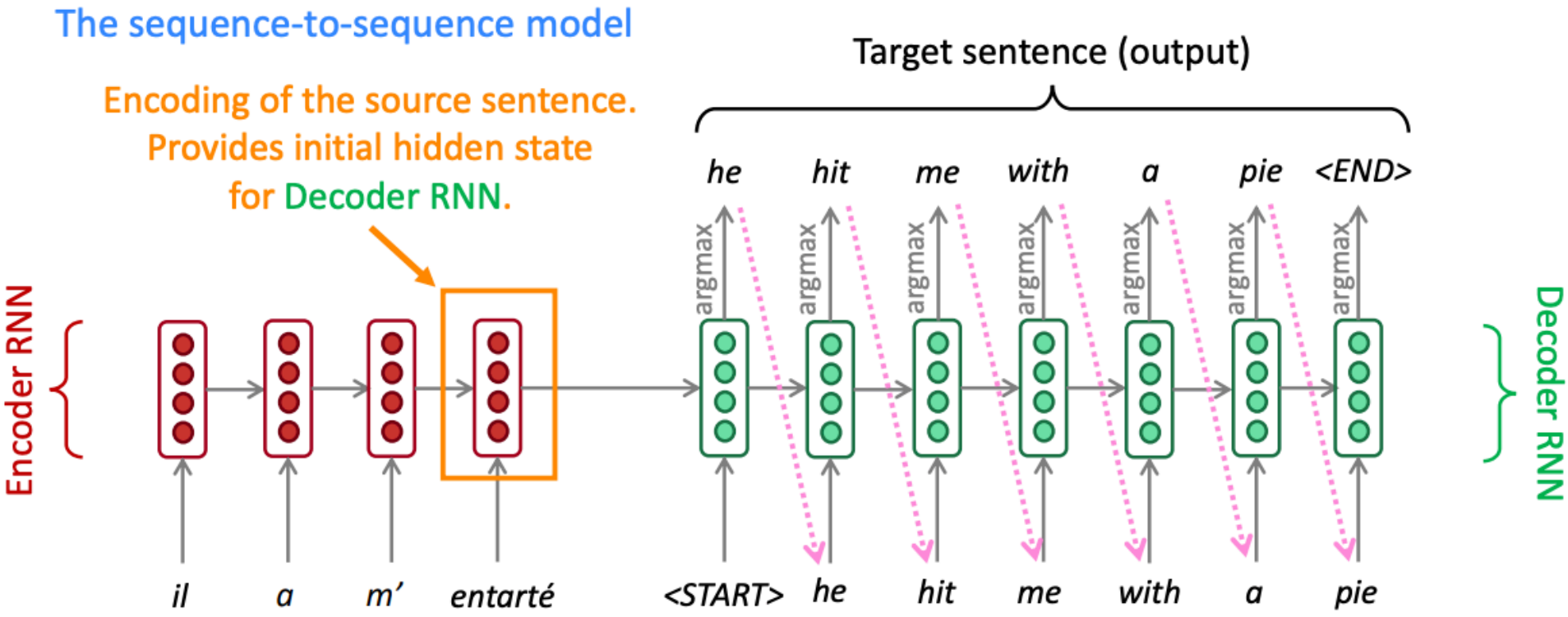
$$\alpha_{ts} = \frac{\exp\left(\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s)\right)}{\sum_{s'=1}^{S} \exp\left(\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_{s'})\right)} \quad \text{[Attention weights]} \quad (1)$$

$$\boldsymbol{c}_t = \sum_s \alpha_{ts} \bar{\boldsymbol{h}}_s \quad \text{[Context vector]} \quad (2)$$

$$\boldsymbol{a}_t = f(\boldsymbol{c}_t, \boldsymbol{h}_t) = \tanh(\boldsymbol{W_c}[\boldsymbol{c}_t; \boldsymbol{h}_t]) \quad \text{[Attention vector]} \quad (3)$$

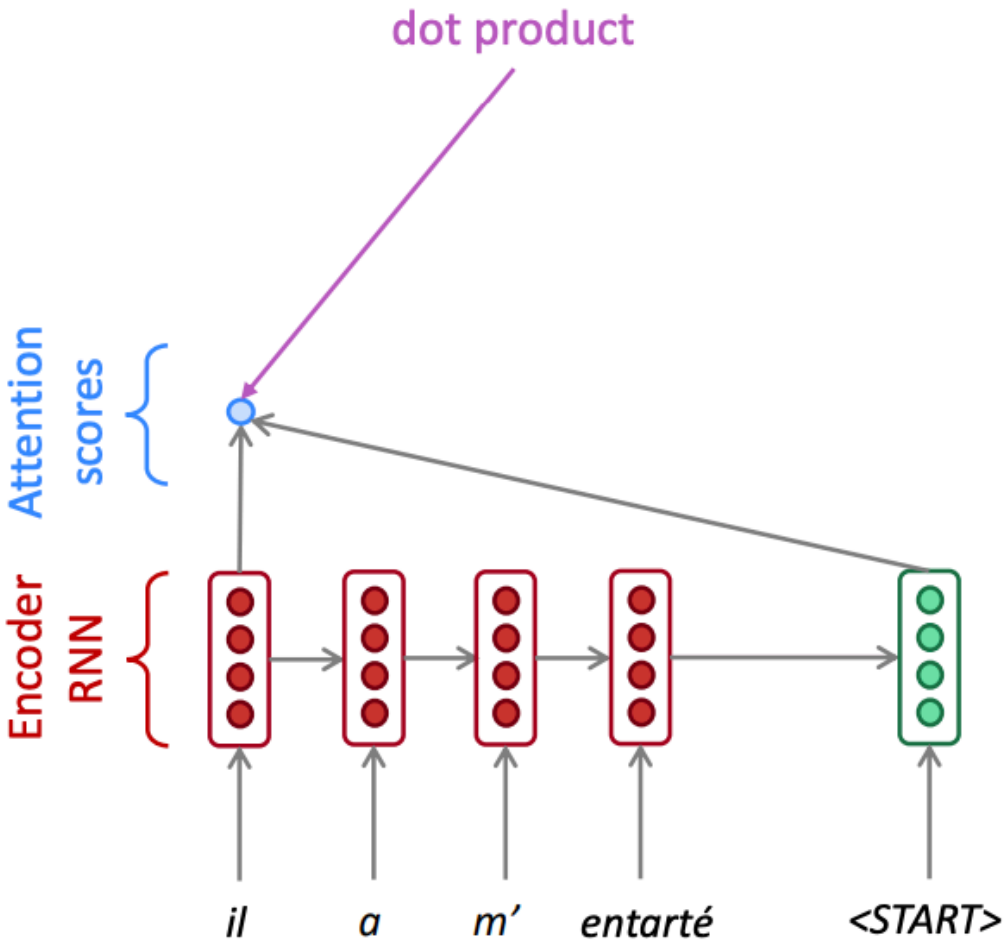
$$\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s) = \begin{cases} \boldsymbol{h}_t^\top \boldsymbol{W} \bar{\boldsymbol{h}}_s & \text{[Luong's multiplicative style]} \\ \boldsymbol{v}_a^\top \tanh\left(\boldsymbol{W_1} \boldsymbol{h}_t + \boldsymbol{W_2} \bar{\boldsymbol{h}}_s\right) & \text{[Bahdanau's additive style]} \end{cases} \quad (4)$$

Neural machine translation



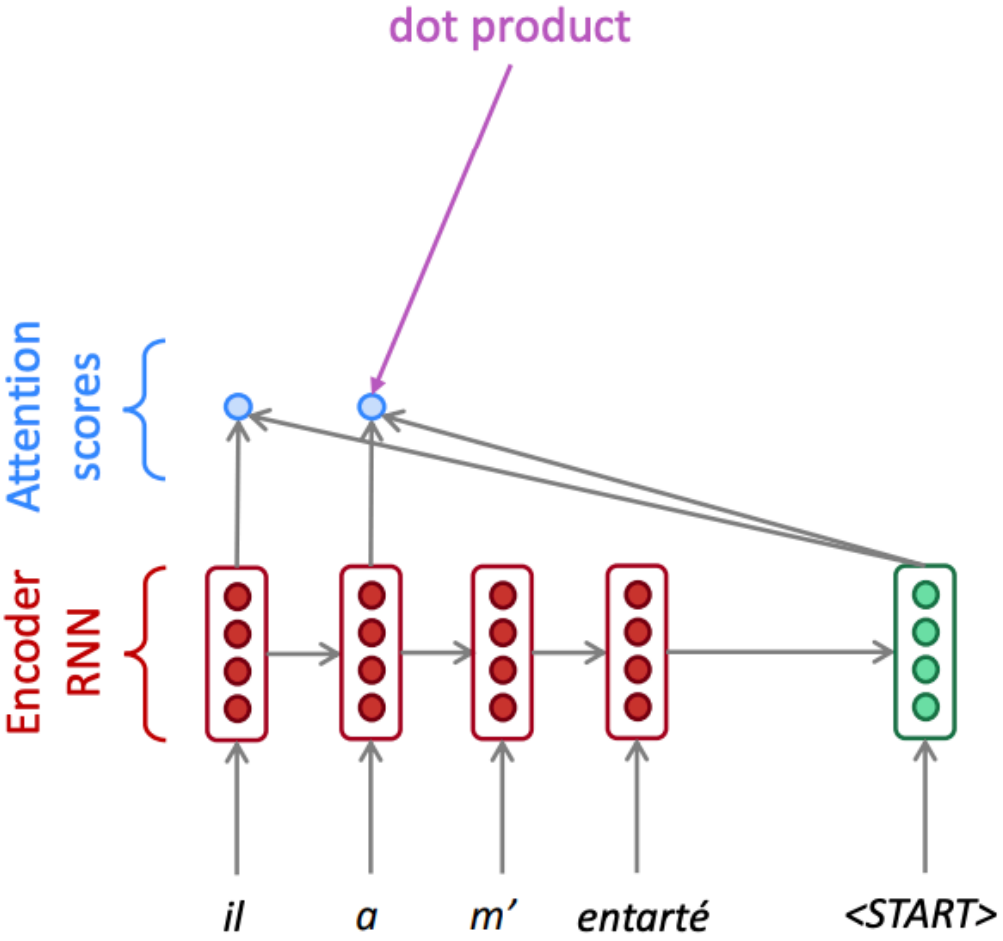
Source: Abigail See

Neural machine translation



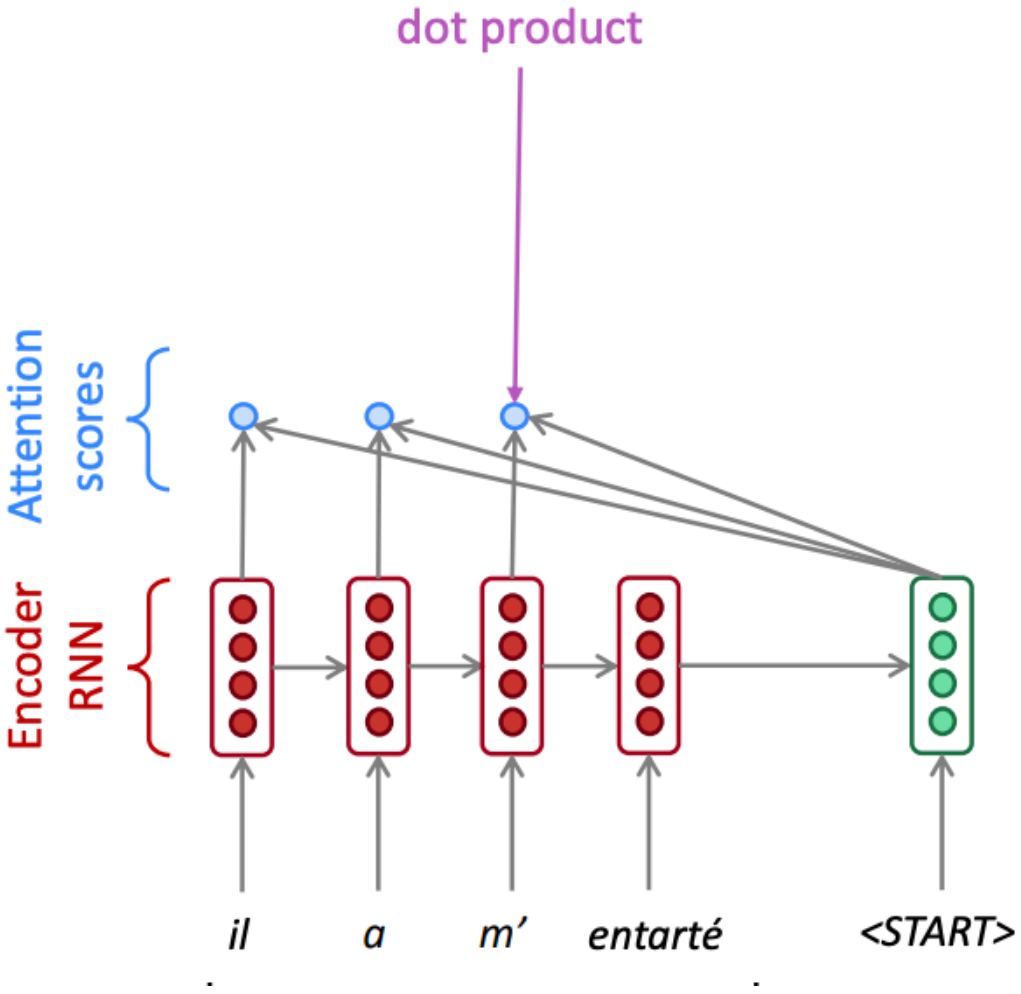
Source: Abigail See

Neural machine translation



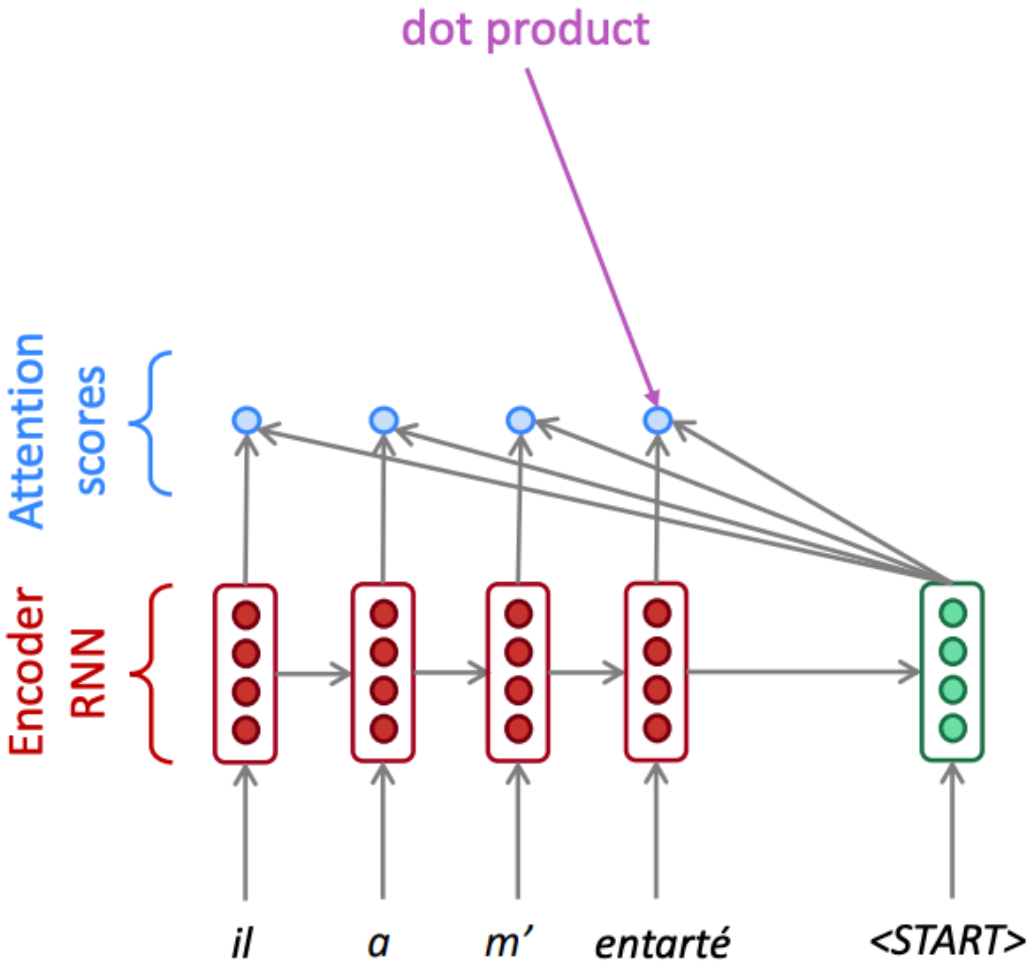
Source: Abigail See

Neural machine translation



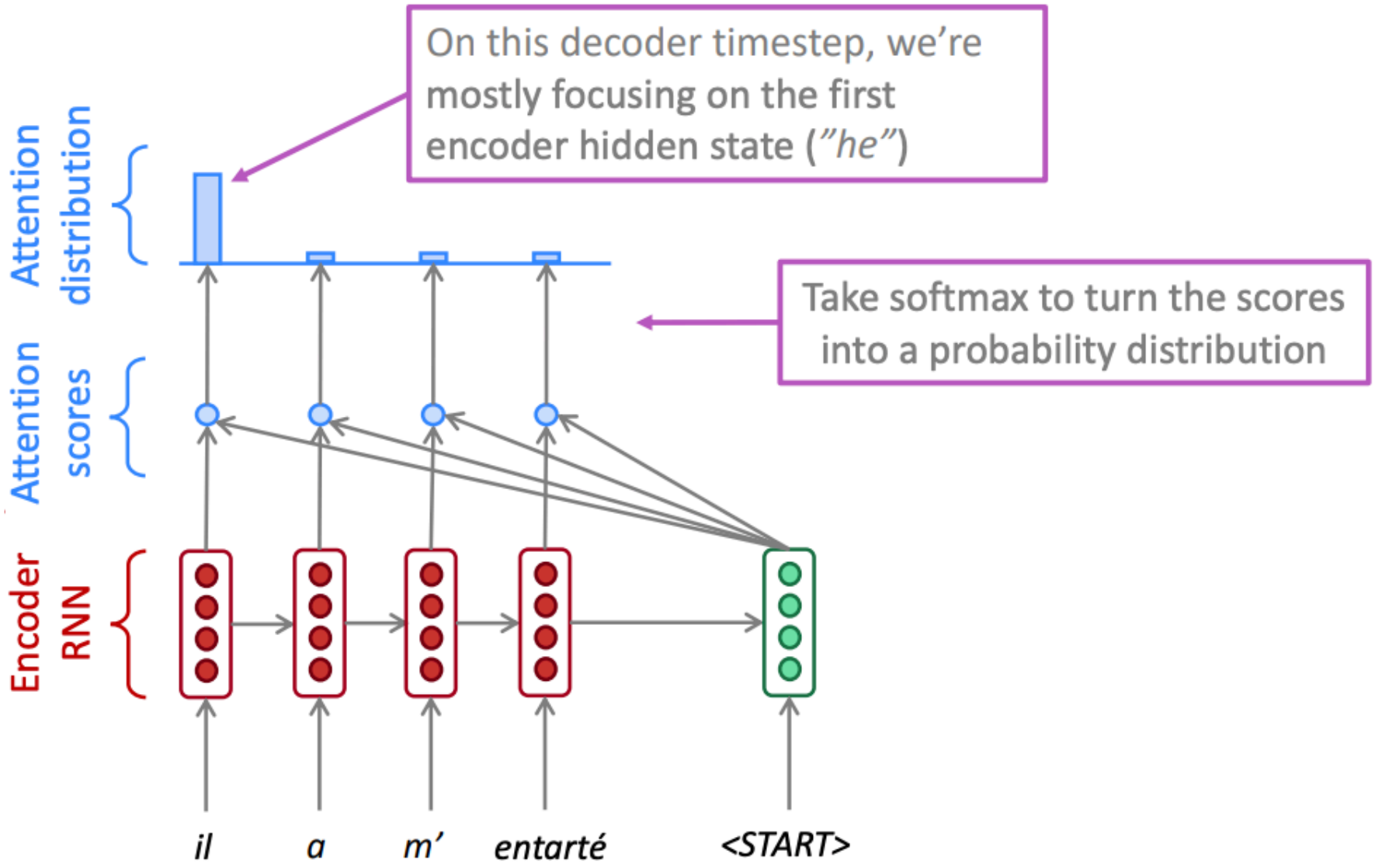
Source: Abigail See

Neural machine translation

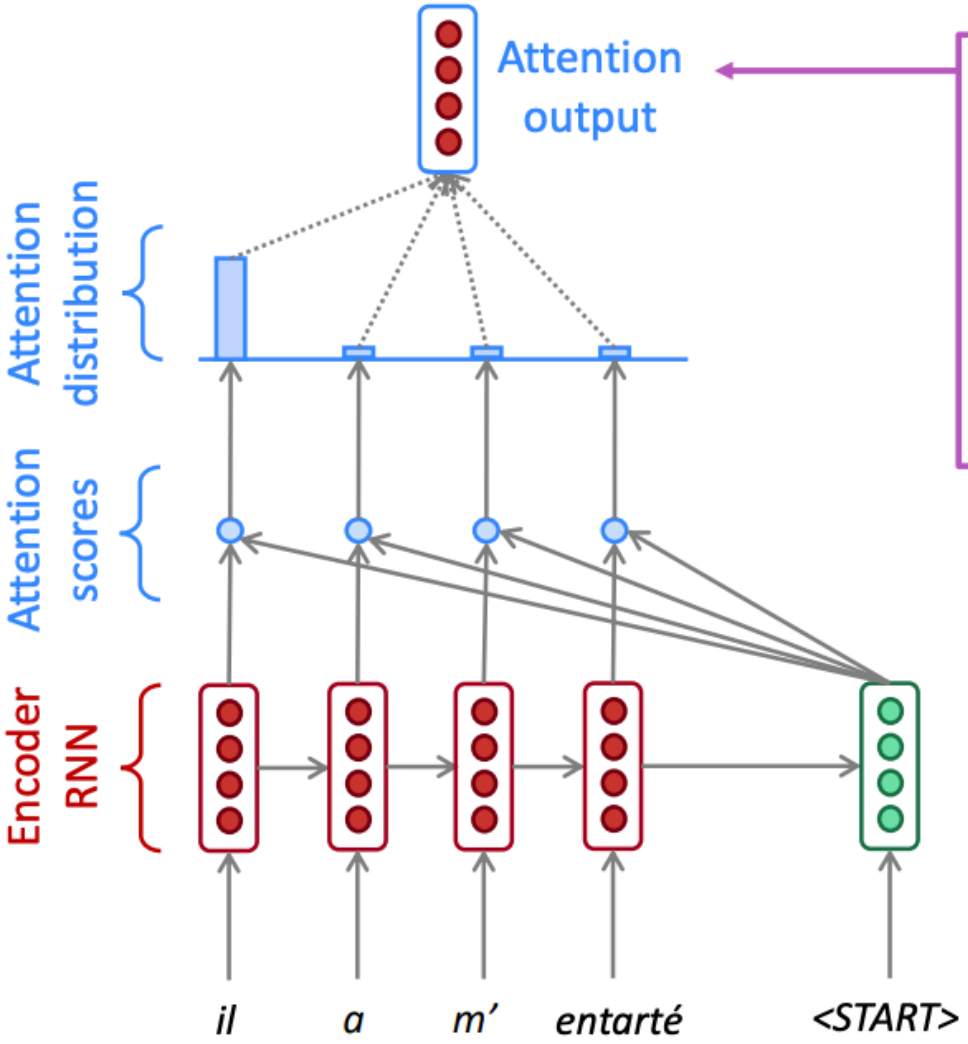


Source: Abigail See

Neural machine translation



Neural machine translation

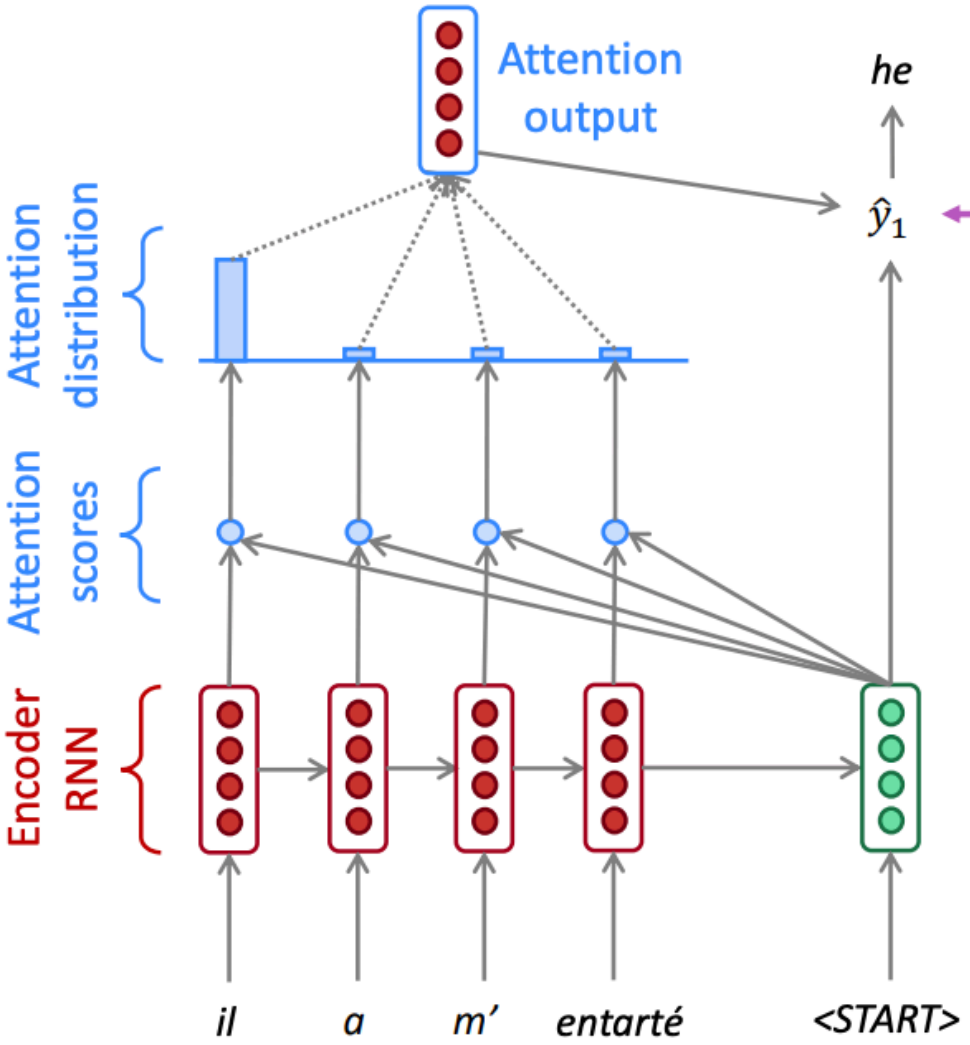


Use the attention distribution to take a **weighted sum** of the encoder hidden states.

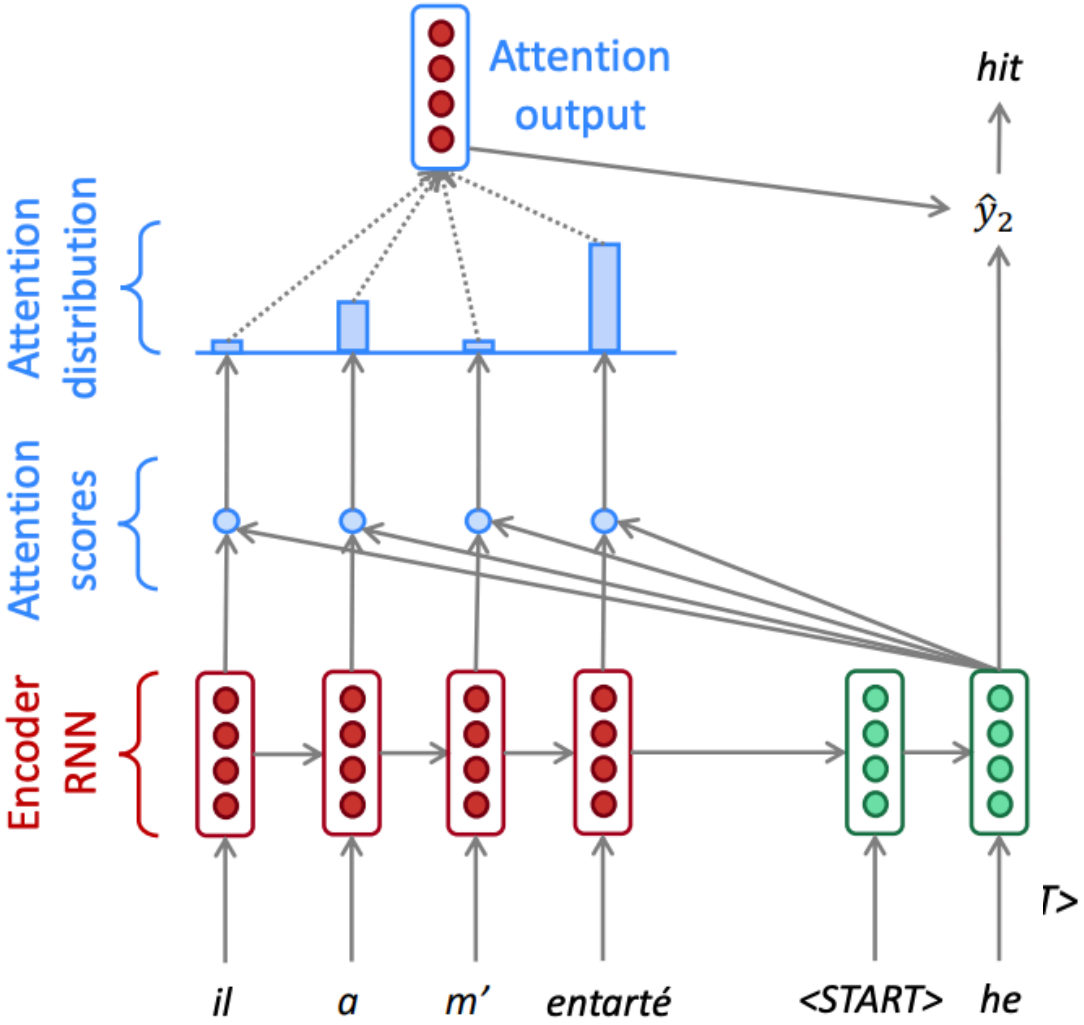
The attention output mostly contains information from the hidden states that received high attention.

Decoder RNN

Neural machine translation

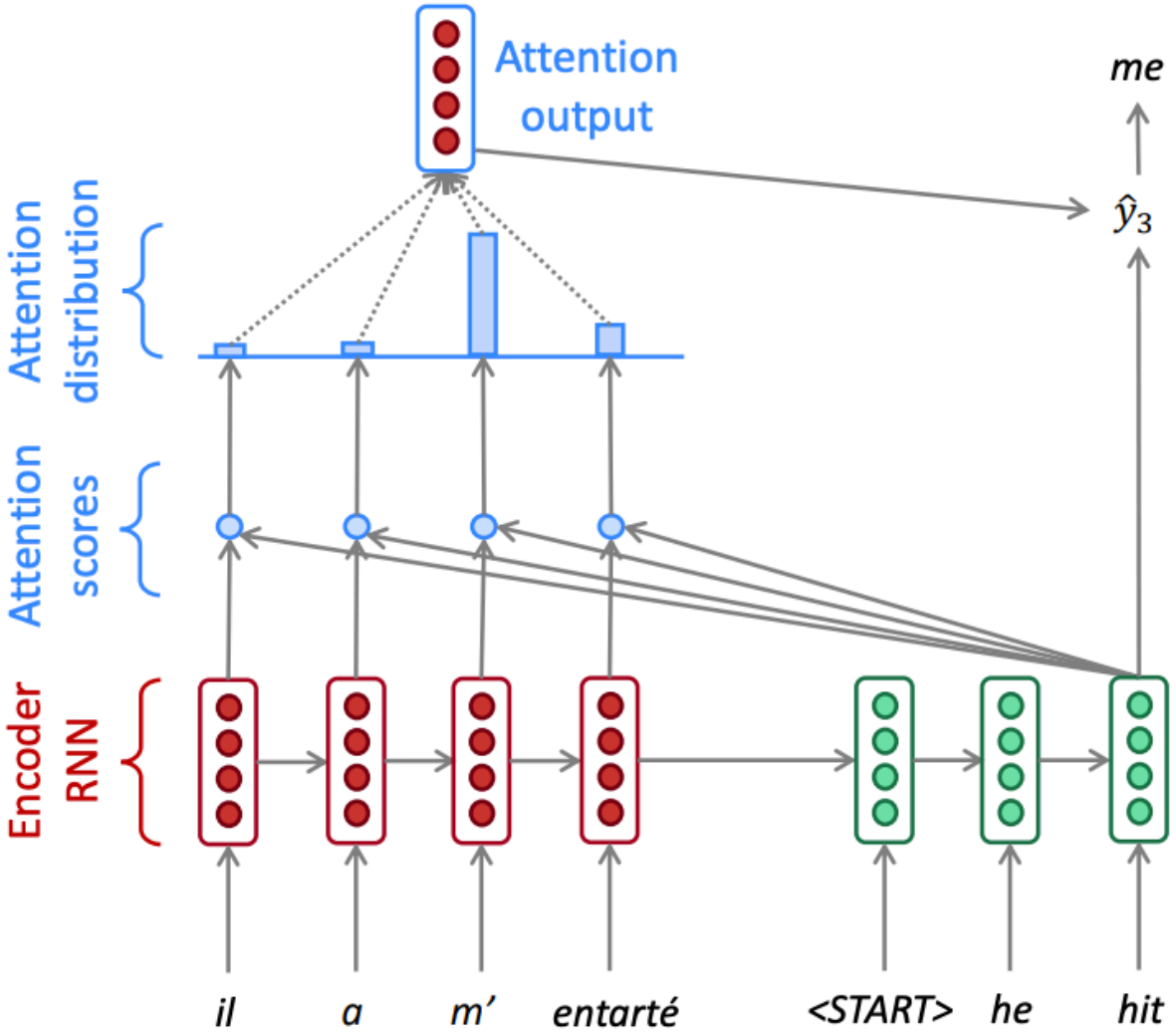


Neural machine translation



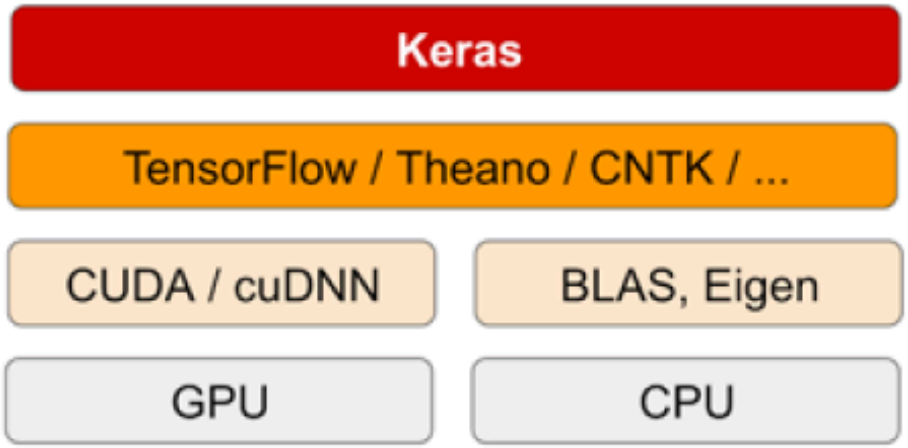
Source: Abigail See

Neural machine translation



Source: Abigail See

Lab 2



Individual mobility prediction

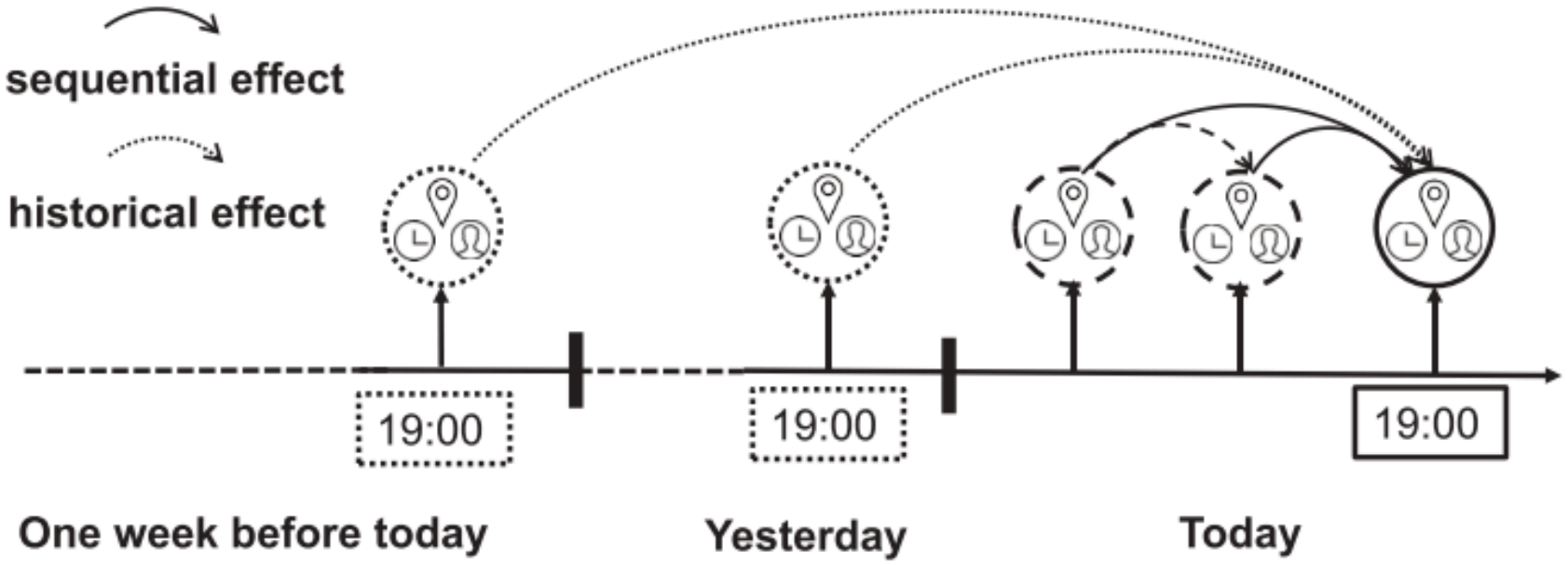
DEFINITION 1 (TRAJECTORY SEQUENCE). *Spatiotemporal point q is a tuple of time stamp t and location identification l, i.e., $q = (t, l)$. Given a user identification u, trajectory sequence S is a spatiotemporal point sequence, i.e., $S^u = q_1 q_2 \ldots q_n$.*

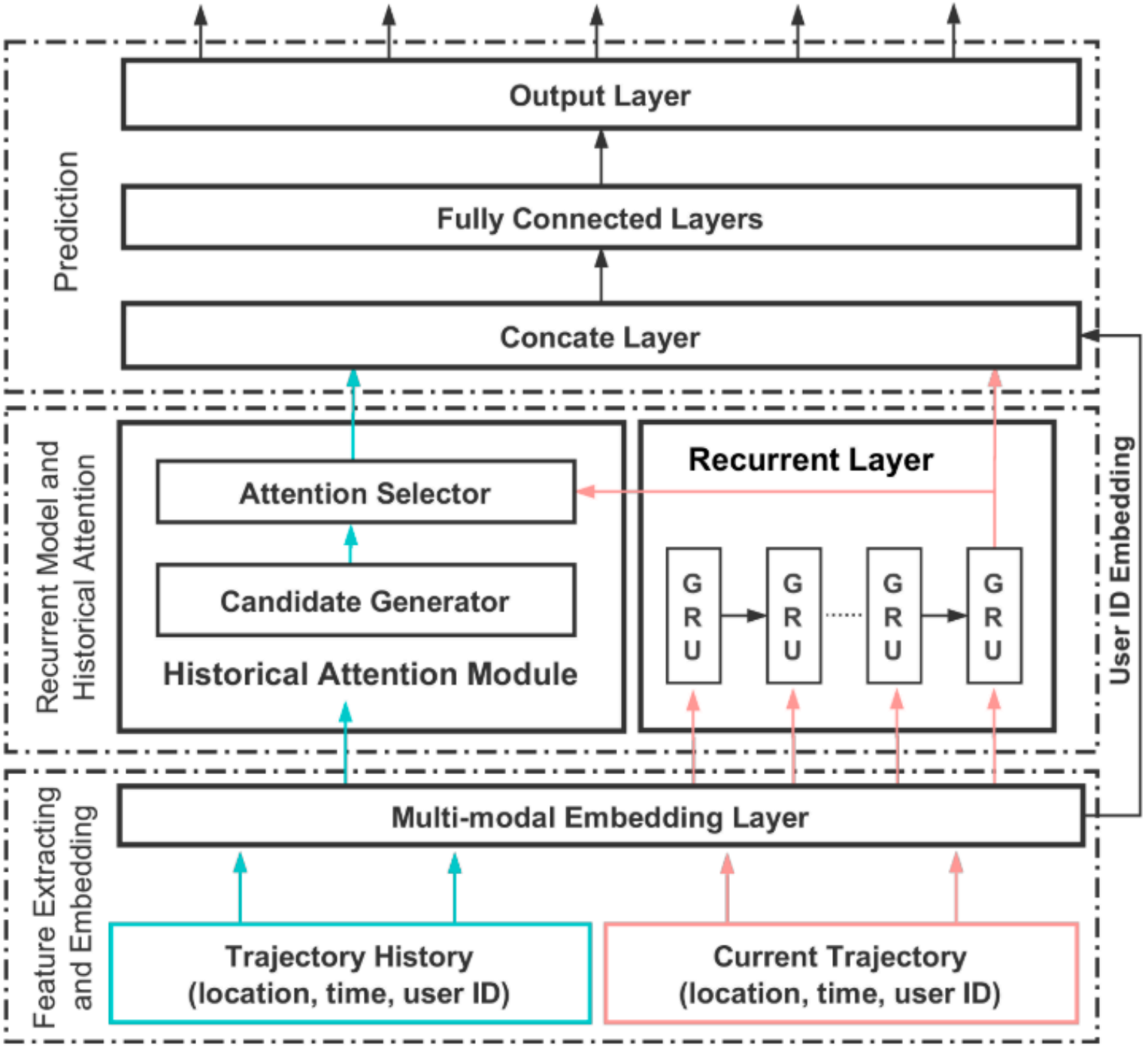
DEFINITION 2 (TRAJECTORY). *Given a trajectory sequence S^u and a time window t_w, trajectory is a subsequence $S^u_{t_w} = q_i q_{i+1} .. q_{i+k}$ of S^u in the time window t_w, if $\forall\ 1 < j \leq k$, t_{q_j} belongs to t_w.*

DeepMove: Predicting Human Mobility with Attentional Recurrent Networks

Individual mobility prediction

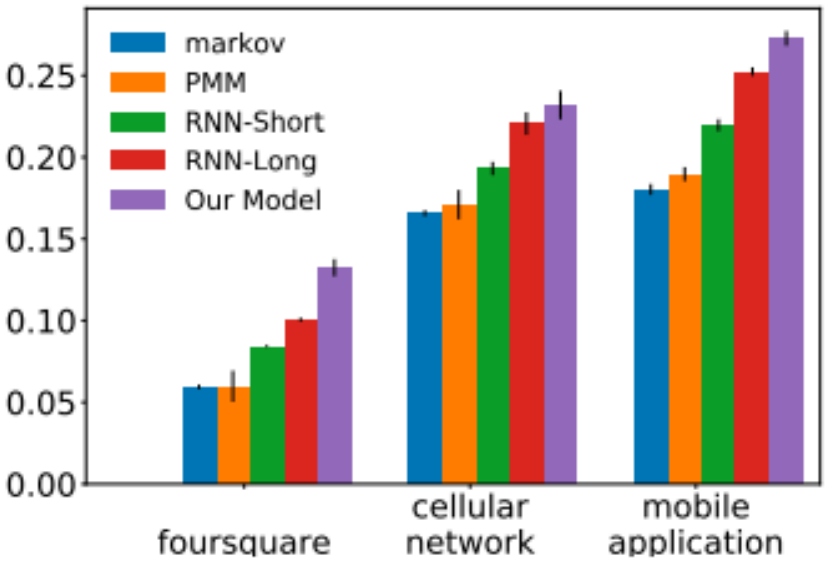
PROBLEM 1 (MOBILITY PREDICTION). *Given the current trajectory* $S^u_{t_{wm}} = q_1 q_2 .. q_n$ *and the corresponding trajectory history* $S^u_{t_{w_1}} S^u_{t_{w_2}} \ldots S^u_{t_{w_{m-1}}}$, *predict the next spatiotemporal point* q_{n+1} *in the trajectory.*



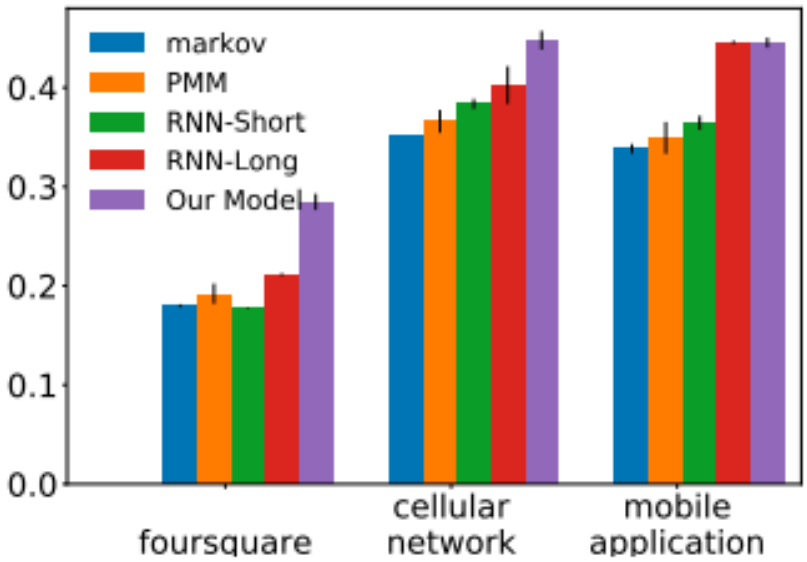

Prediction
Output Layer
Fully Connected Layers
Concate Layer
Recurrent Model and Historical Attention
Attention Selector
Candidate Generator
Historical Attention Module
Recurrent Layer
G R U
G R U
G R U
G R U
User ID Embedding
Feature Extracting and Embedding
Multi-modal Embedding Layer
Trajectory History (location, time, user ID)
Current Trajectory (location, time, user ID)

Individual mobility prediction

- Rank the candidate locations by the probabilities generated from the model, and check whether the ground-truth location v appears in the top- k candidate locations.

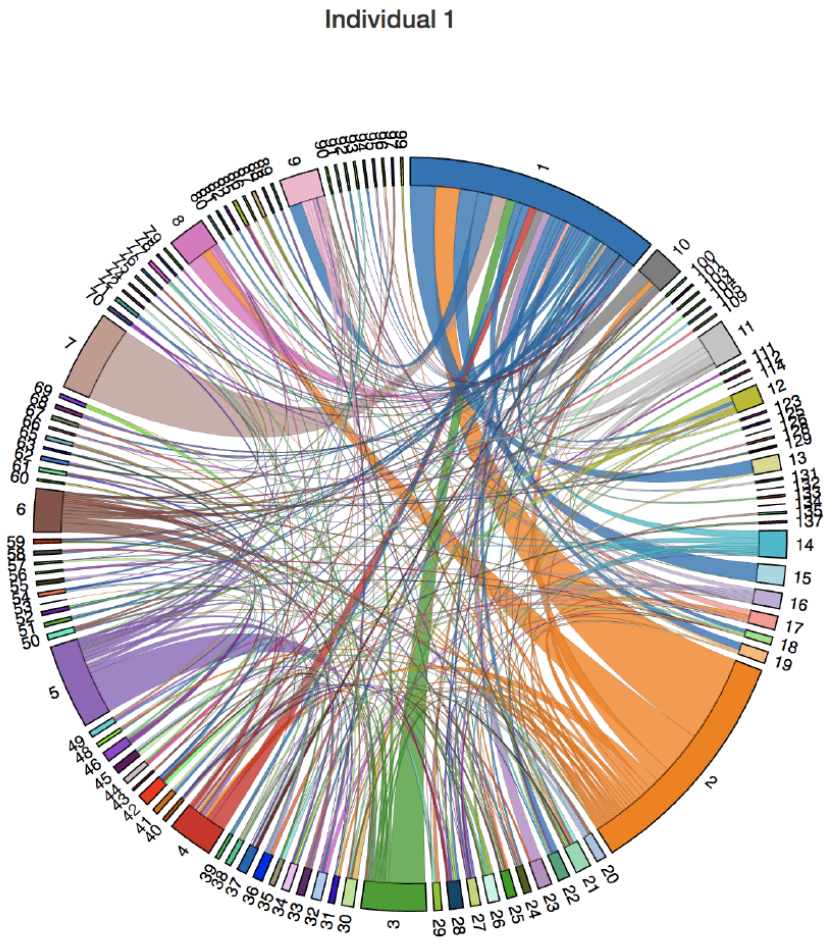


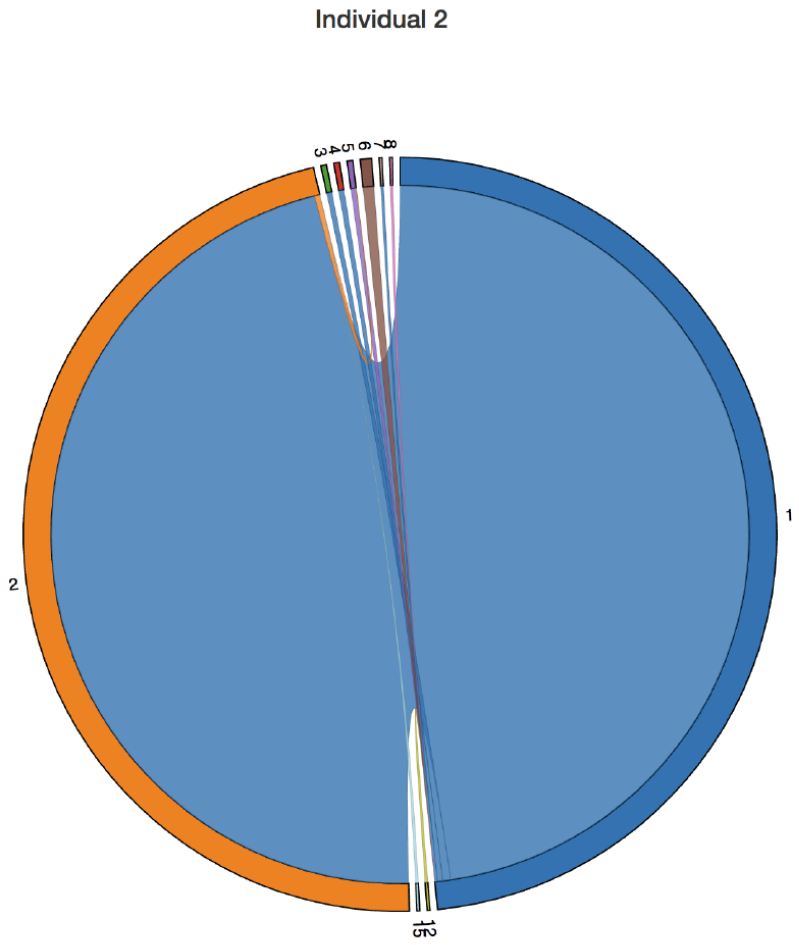
(a) top-1 prediction accuracy



(b) top-5 prediction accuracy

Individual mobility prediction





Graph Neural Networks

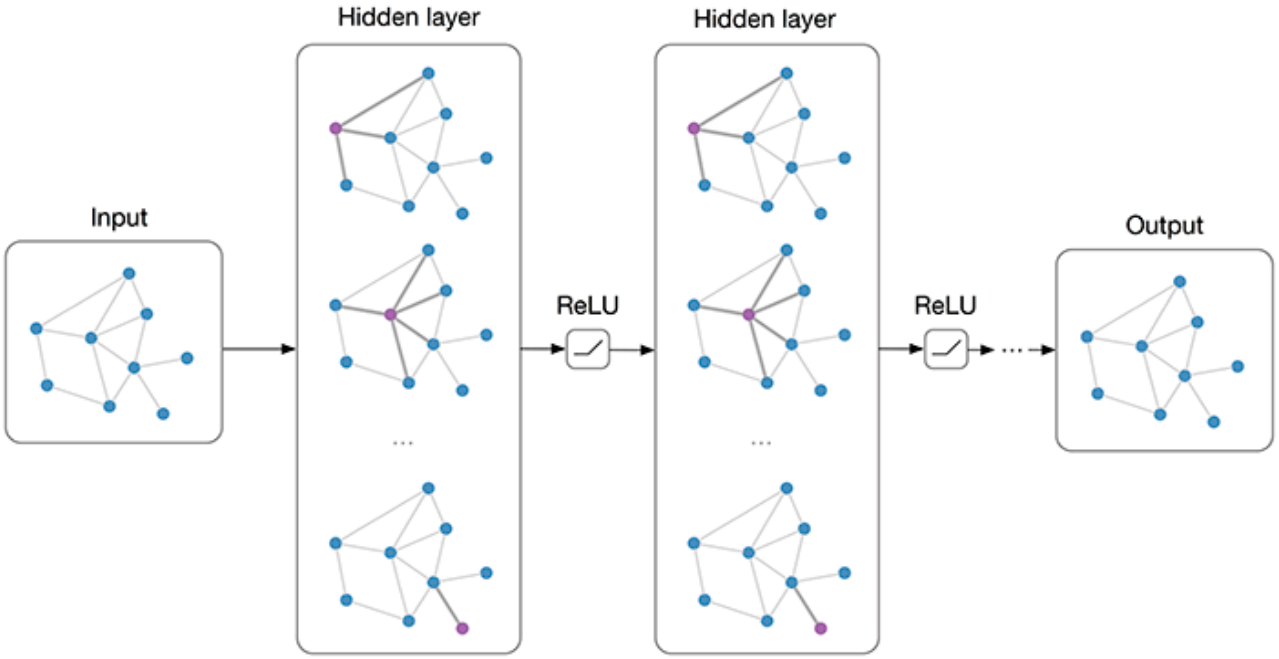
Graph Neural Networks

- CNN only works for regular grids

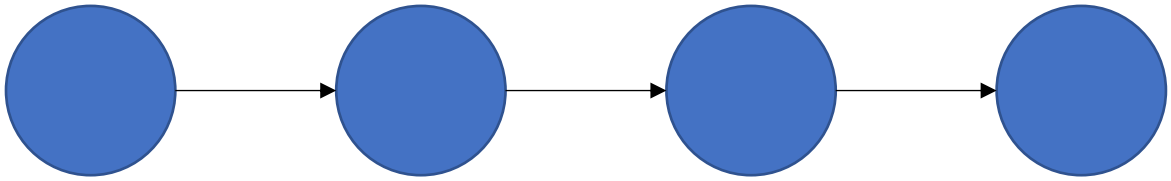
Graph Neural Networks

- CNN only works for regular grids

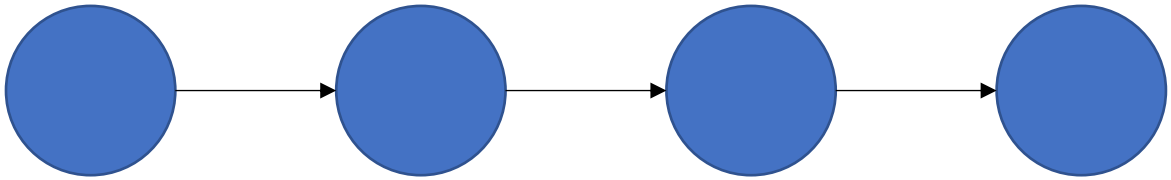
- We want something similar, but for irregular grids
  - Capture local dependencies
  - Translation invariant
  - Parameter sharing



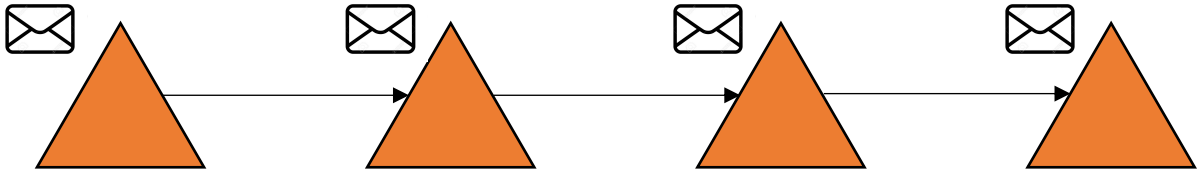
Neural message passing networks

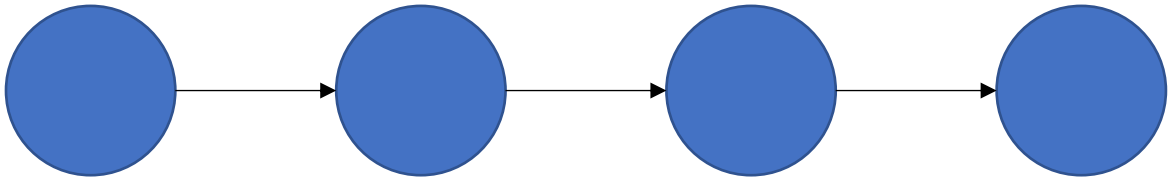


Embed('word') = 

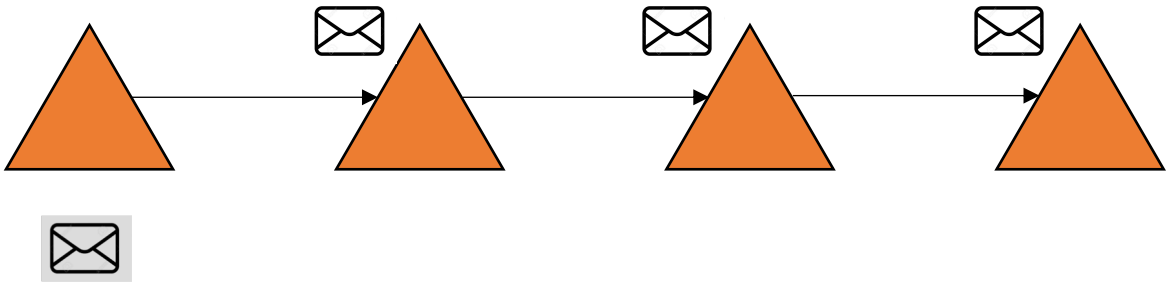


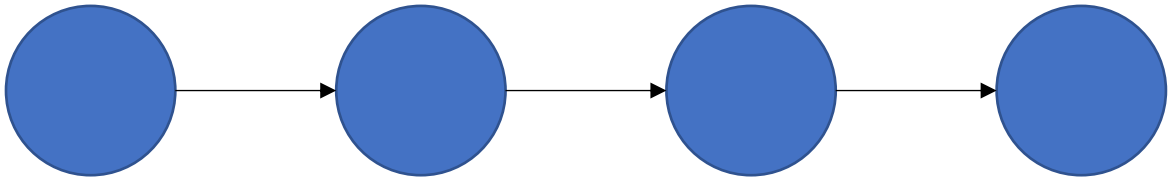
Embed('word') = 



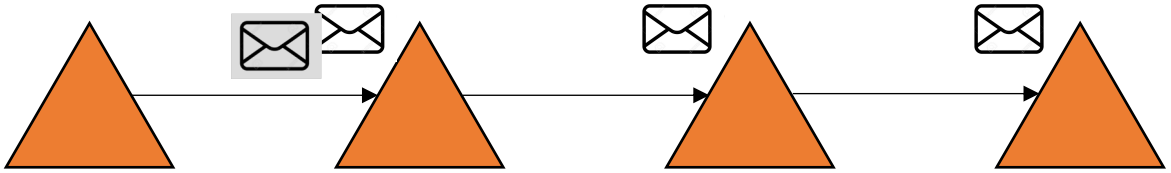


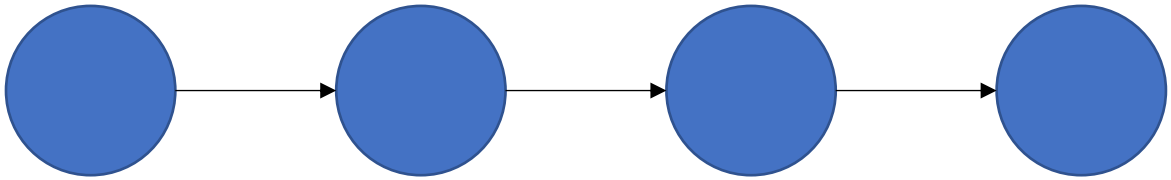
Embed('word') = 



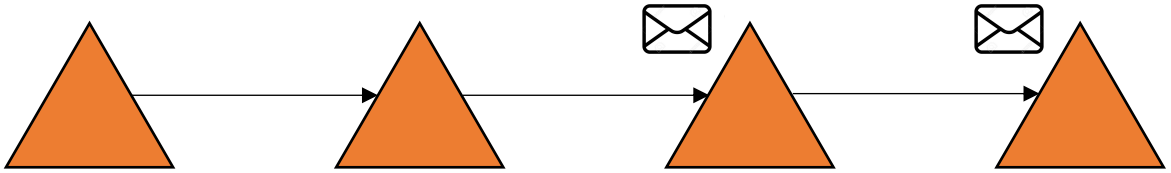


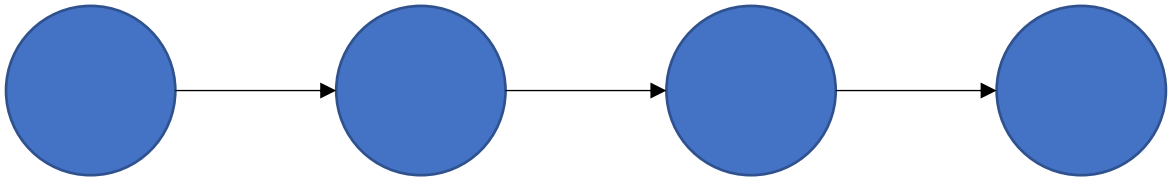
Embed('word') = ✉



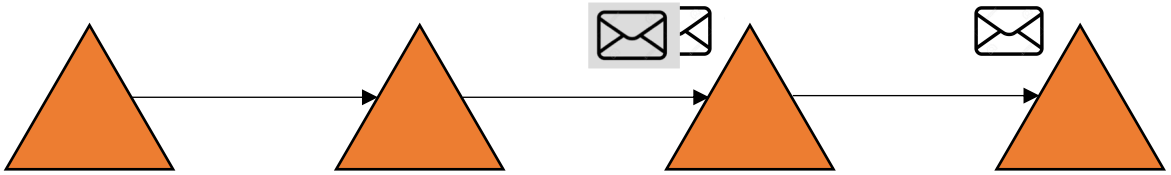


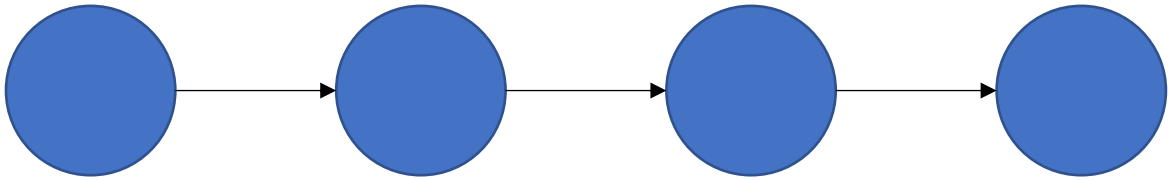
Embed('word') = 



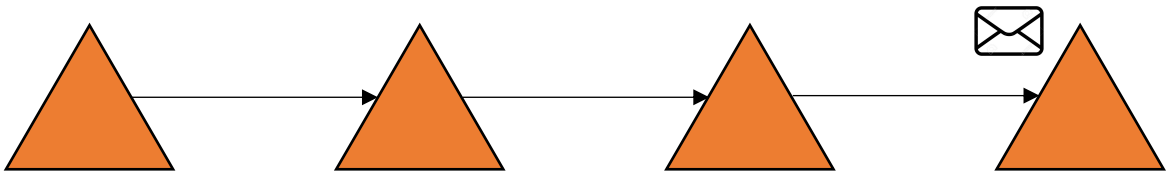


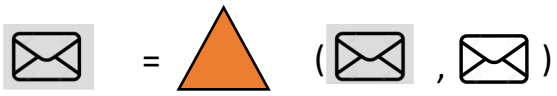
Embed('word') = 

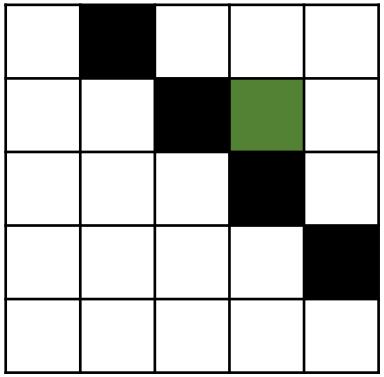


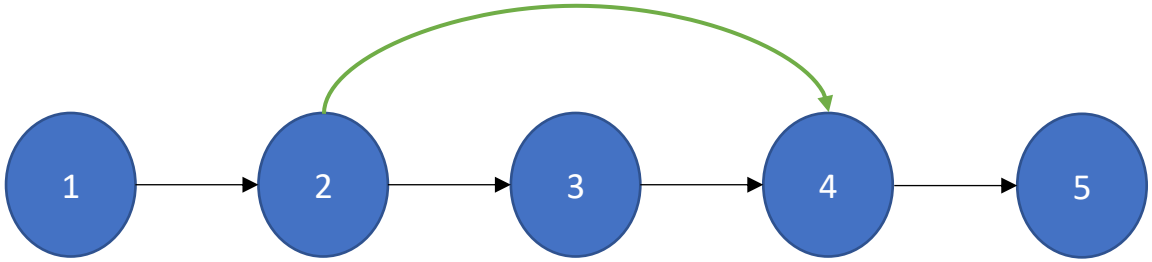


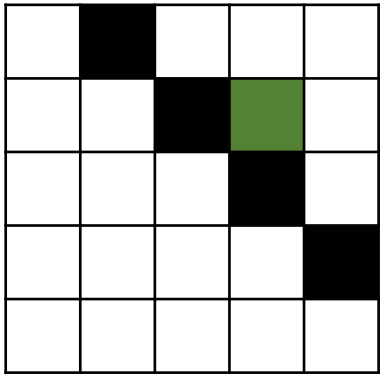
Embed('word') = 

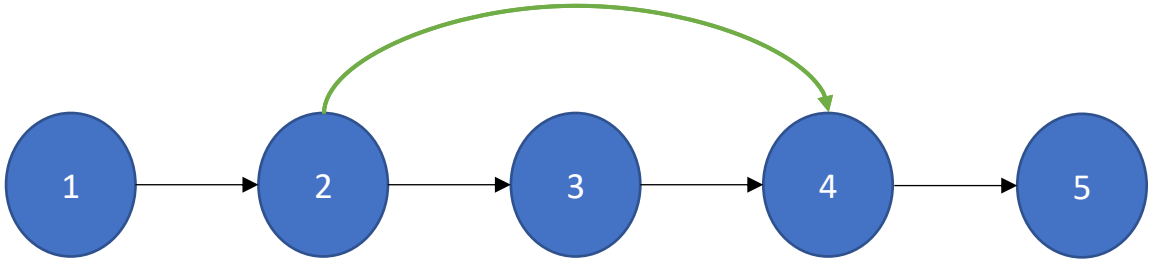


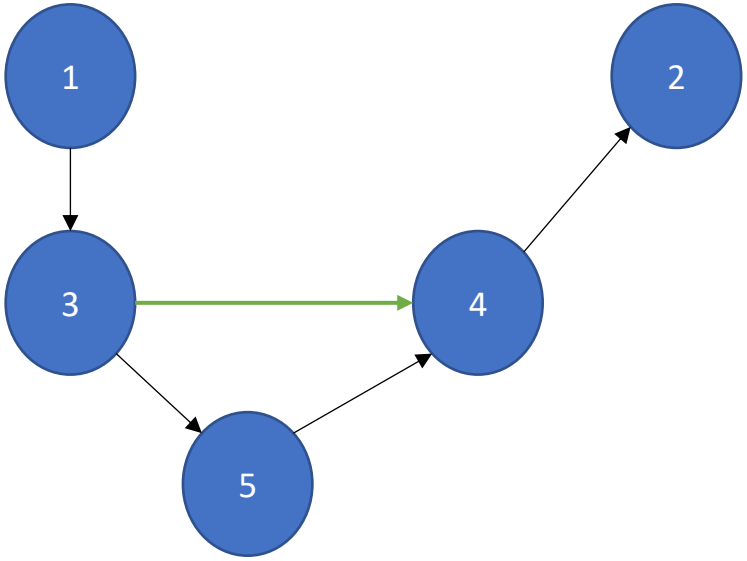


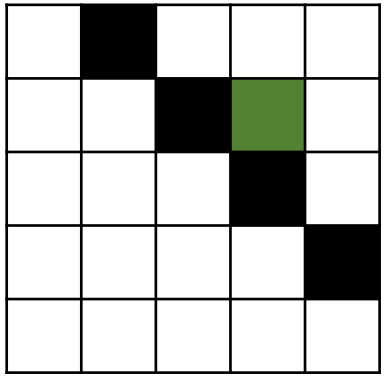


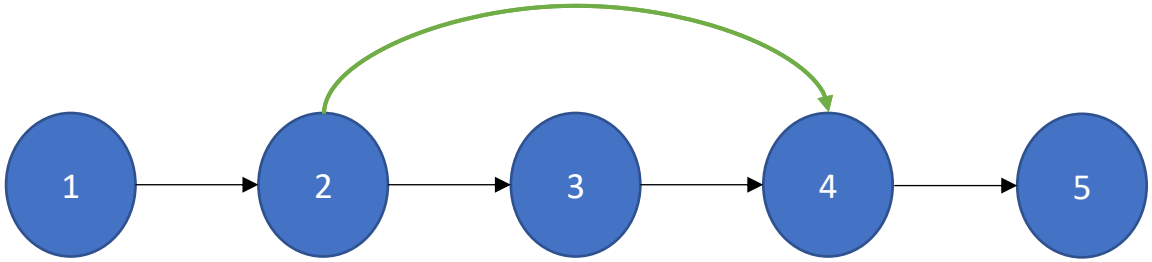

1
2
3
4
5

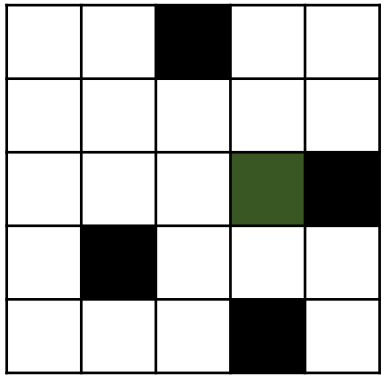


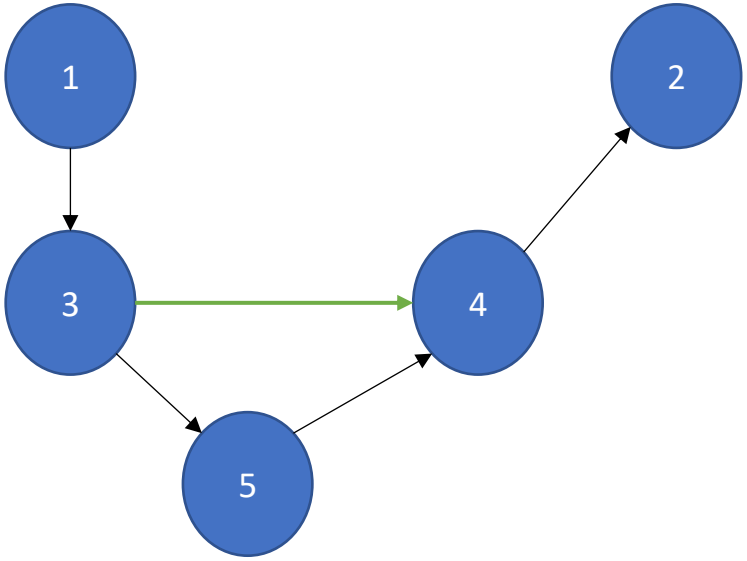

1
2
3
4
5

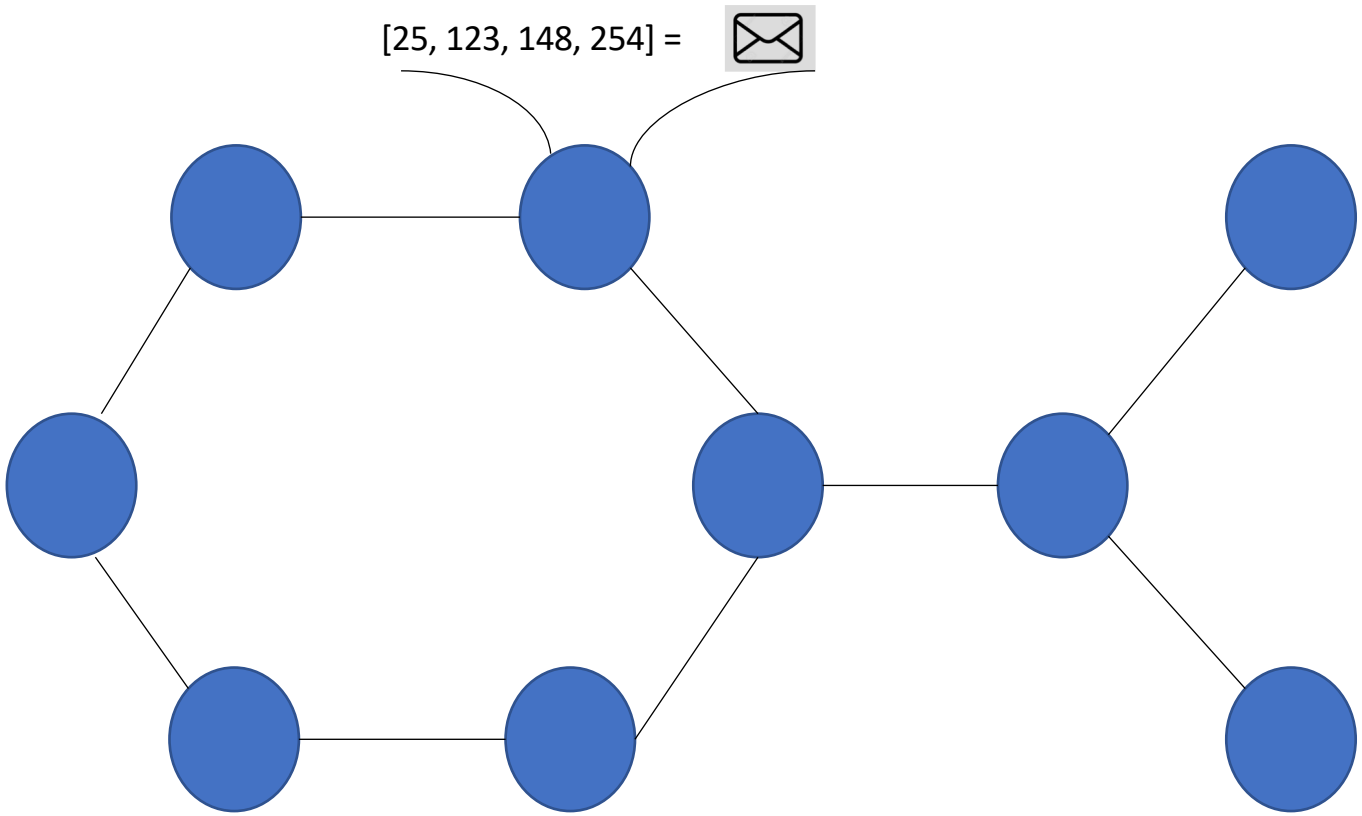

1
2
3
4
5

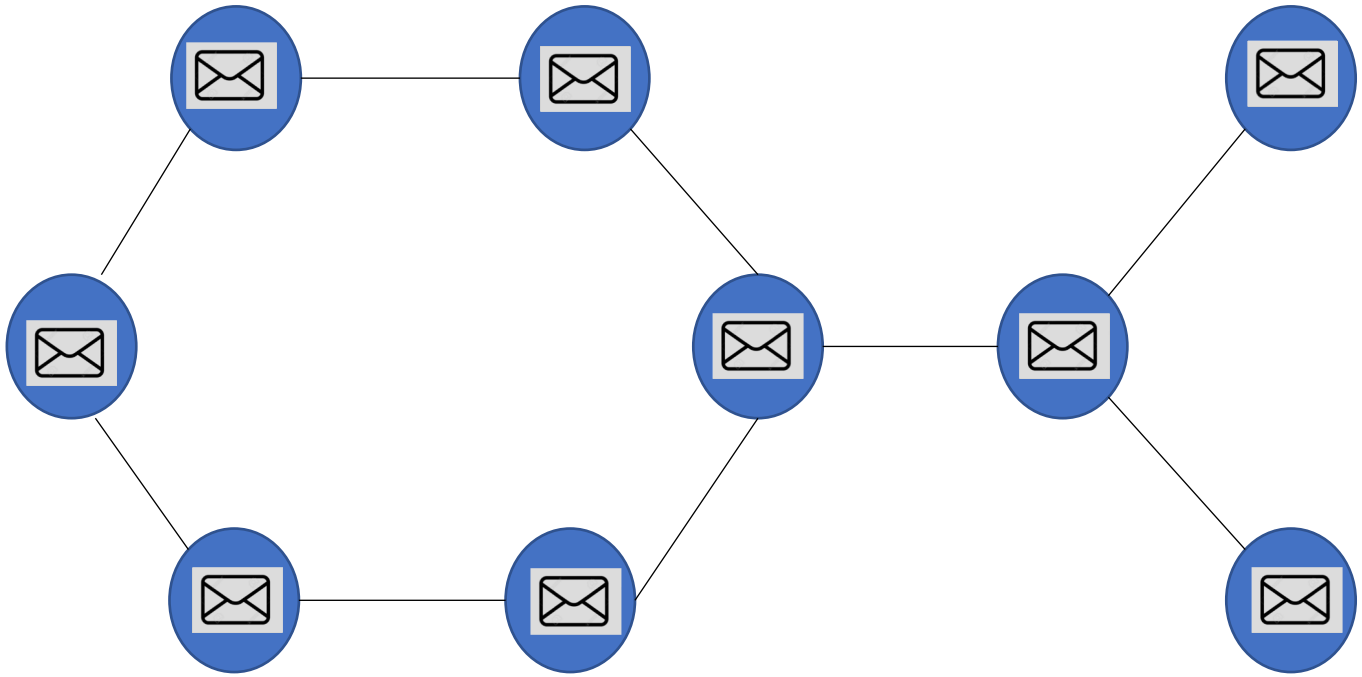


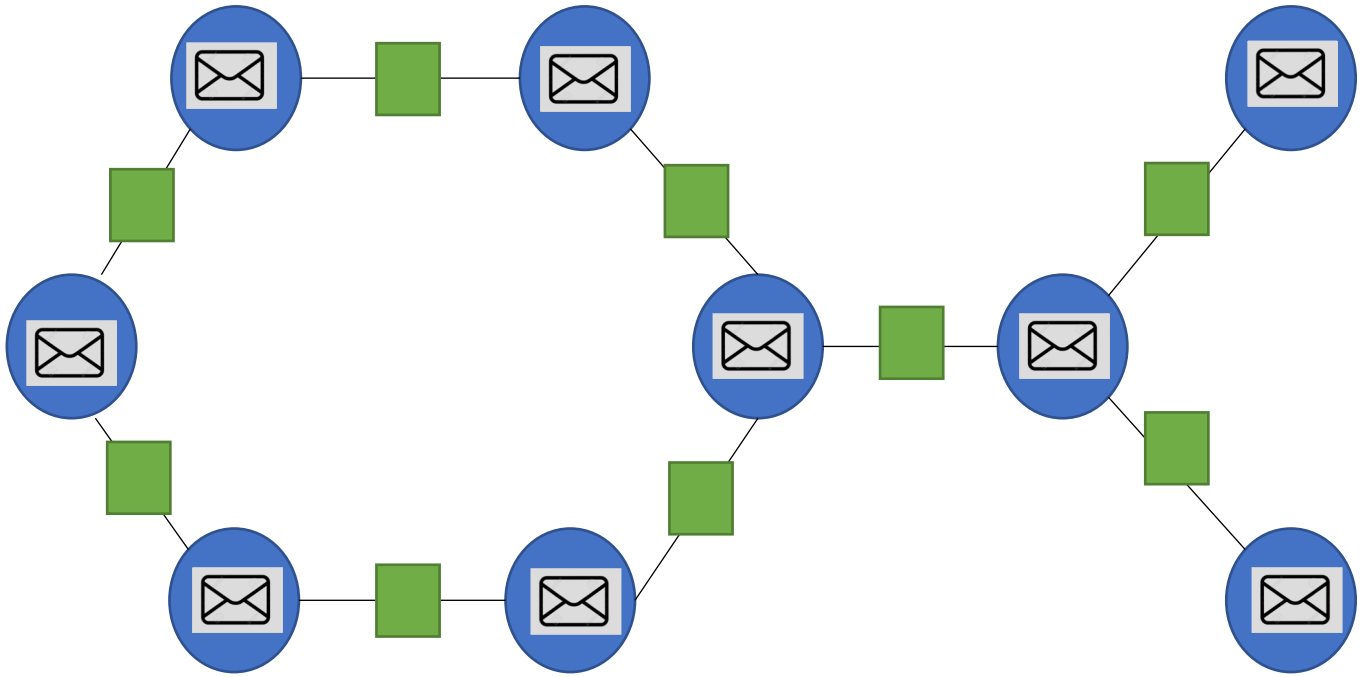

1
2
3
4
5

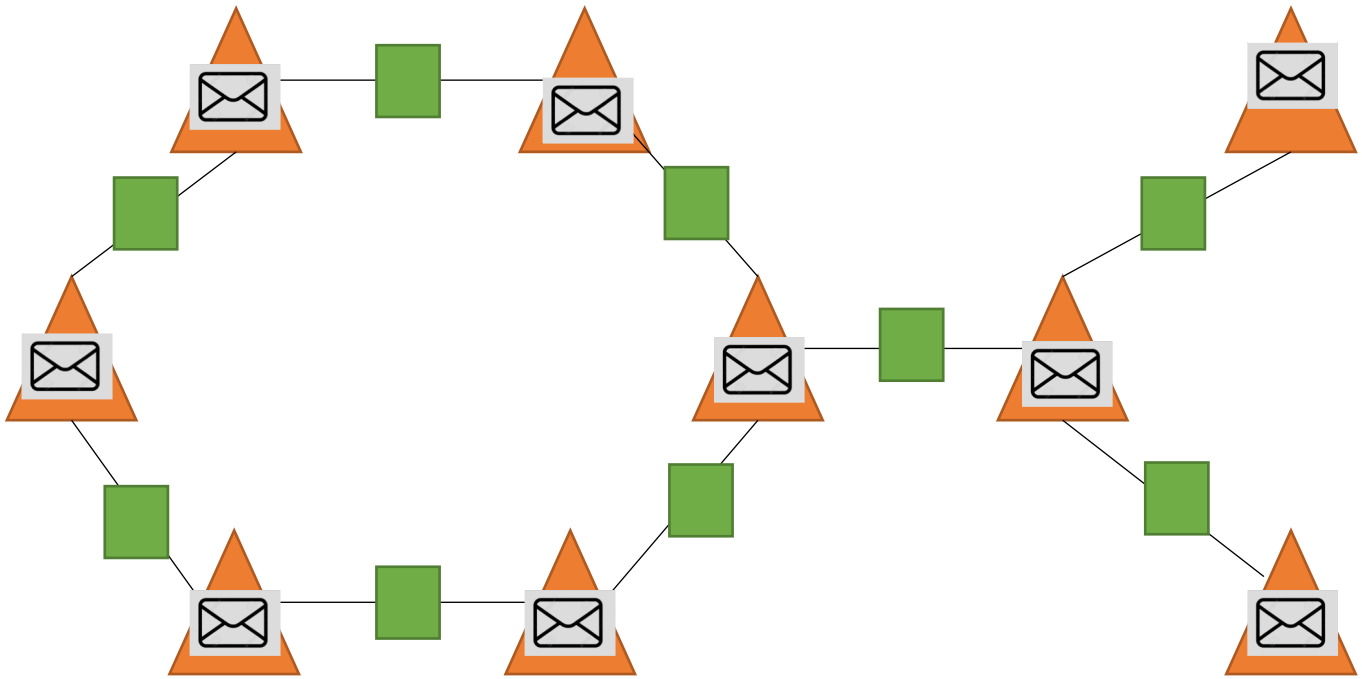


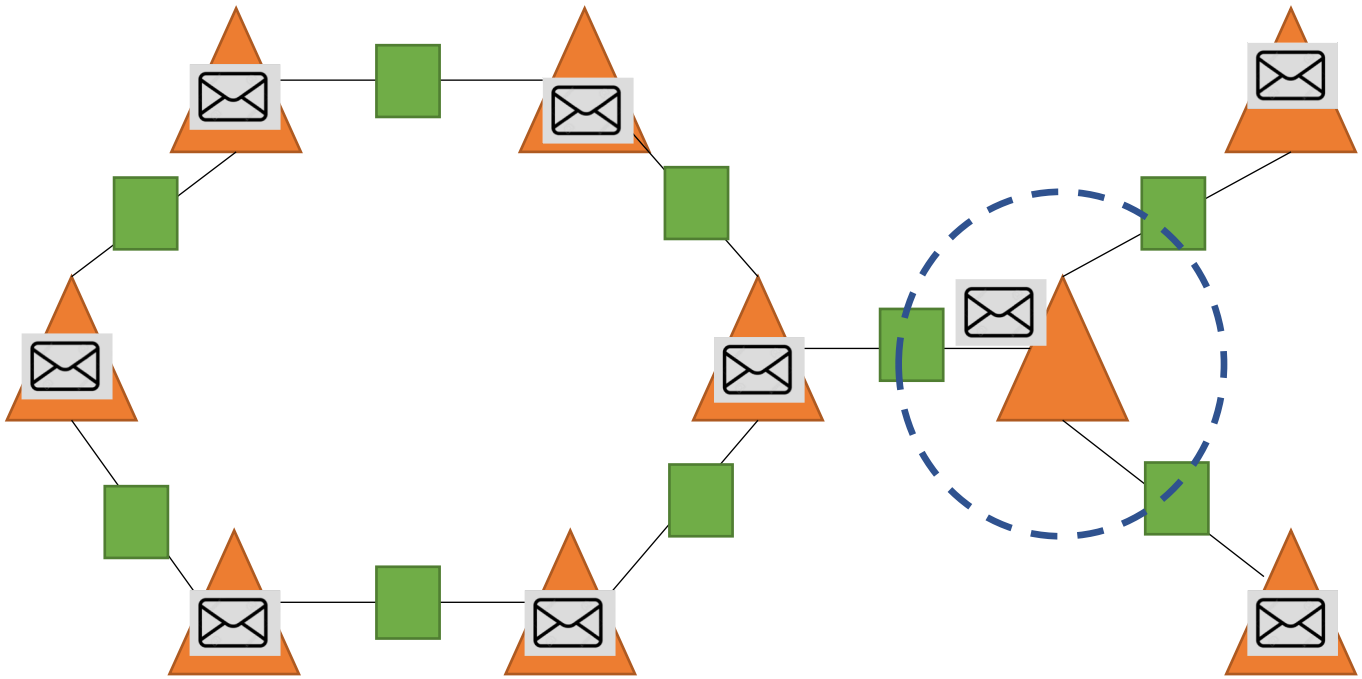

1
2
3
4
5

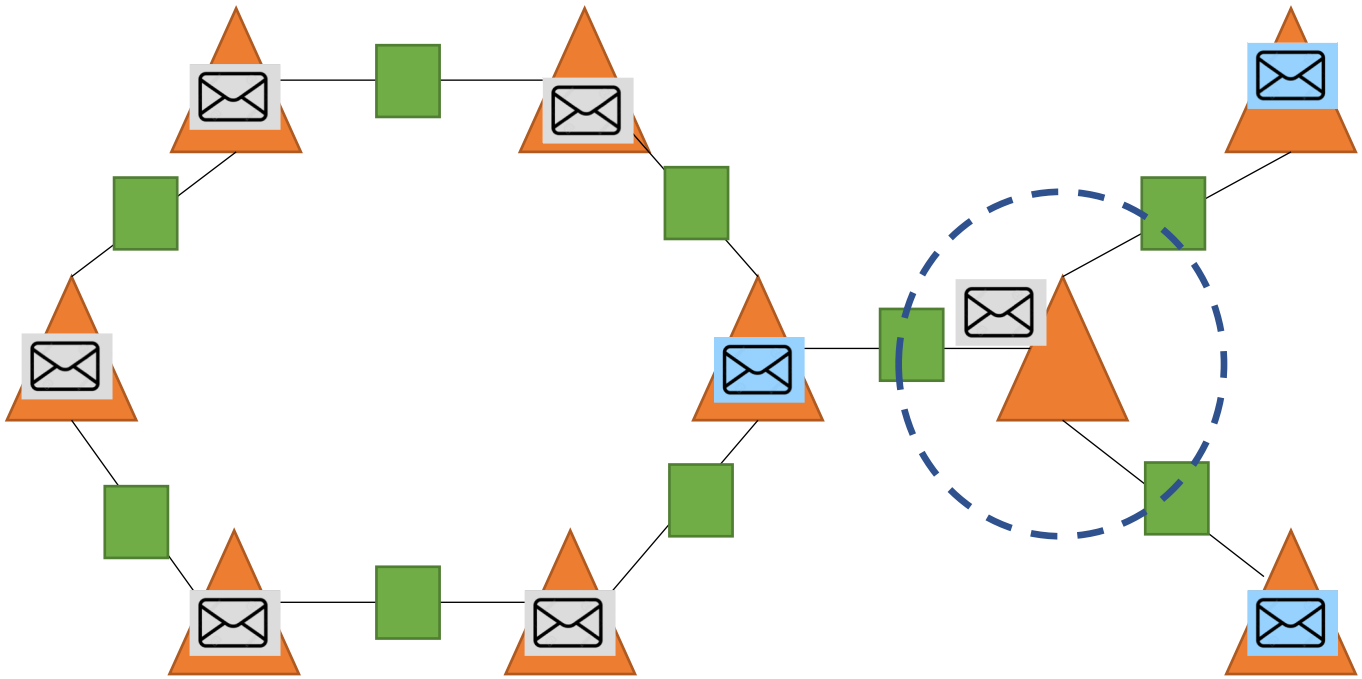

[25, 123, 148, 254] =











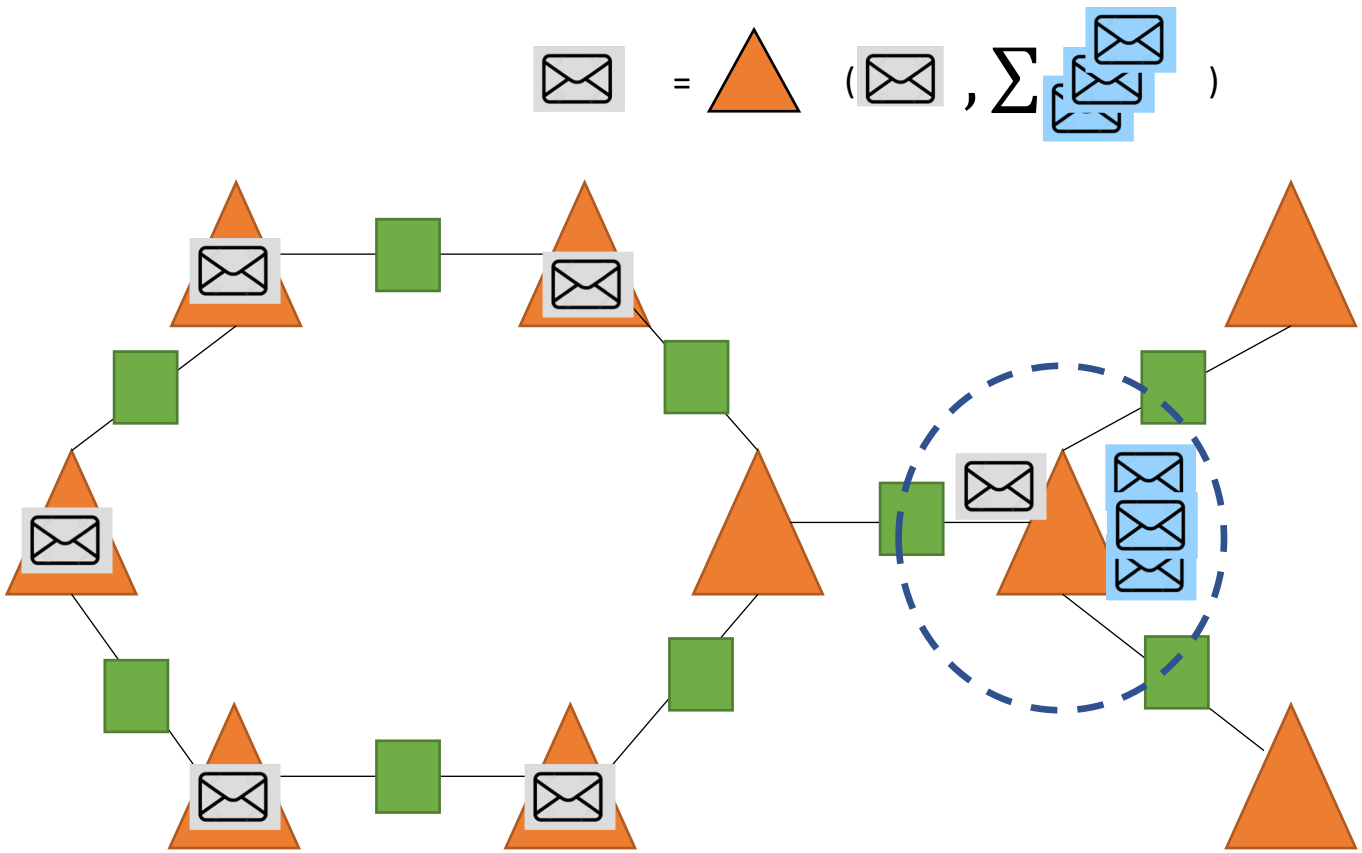


TABLE 2
Different variants of graph neural networks.

Name	Variant	Aggregator	Updater
Spectral Methods	ChebNet	$\mathbf{N}_k = \mathbf{T}_k(\tilde{\mathbf{L}})\mathbf{X}$	$\mathbf{H} = \sum_{k=0}^{K} \mathbf{N}_k \mathbf{\Theta}_k$
	1^{st}-order model	$\mathbf{N}_0 = \mathbf{X}$ $\mathbf{N}_1 = \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}\mathbf{X}$	$\mathbf{H} = \mathbf{N}_0\mathbf{\Theta}_0 + \mathbf{N}_1\mathbf{\Theta}_1$
	Single parameter	$\mathbf{N} = (\mathbf{I}_N + \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}})\mathbf{X}$	$\mathbf{H} = \mathbf{N}\mathbf{\Theta}$
	GCN	$\mathbf{N} = \tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{X}$	$\mathbf{H} = \mathbf{N}\mathbf{\Theta}$
Non-spectral Methods	Neural FPs	$\mathbf{h}^t_{\mathcal{N}_v} = \mathbf{h}^{t-1}_v + \sum_{k=1}^{\mathcal{N}_v} \mathbf{h}^{t-1}_k$	$\mathbf{h}^t_v = \sigma(\mathbf{h}^t_{\mathcal{N}_v}\mathbf{W}^{\mathcal{N}_v}_L)$
	DCNN	Node classification: $\mathbf{N} = \mathbf{P}^*\mathbf{X}$ Graph classification: $\mathbf{N} = 1^T_N\mathbf{P}^*\mathbf{X}/N$	$\mathbf{H} = f(\mathbf{W}^c \odot \mathbf{N})$
	GraphSAGE	$\mathbf{h}^t_{\mathcal{N}_v} = \text{AGGREGATE}_t(\{\mathbf{h}^{t-1}_u, \forall u \in \mathcal{N}_v\})$	$\mathbf{h}^t_v = \sigma(\mathbf{W}^t \cdot [\mathbf{h}^{t-1}_v \| \mathbf{h}^t_{\mathcal{N}_v}])$
Graph Attention Networks	GAT	$\alpha_{vk} = \frac{\exp(\text{LeakyReLU}(\mathbf{a}^T[\mathbf{W}\mathbf{h}_v \| \mathbf{W}\mathbf{h}_k]))}{\sum_{j\in\mathcal{N}_v}\exp(\text{LeakyReLU}(\mathbf{a}^T[\mathbf{W}\mathbf{h}_v \| \mathbf{W}\mathbf{h}_j]))}$ $\mathbf{h}^t_{\mathcal{N}_v} = \sigma\left(\sum_{k\in\mathcal{N}_v}\alpha_{vk}\mathbf{W}\mathbf{h}_k\right)$ Multi-head concatenation: $\mathbf{h}^t_{\mathcal{N}_v} = \Big\|_{m=1}^{M}\sigma\left(\sum_{k\in\mathcal{N}_v}\alpha^m_{vk}\mathbf{W}^m\mathbf{h}_k\right)$ Multi-head average: $\mathbf{h}^t_{\mathcal{N}_v} = \sigma\left(\frac{1}{M}\sum_{m=1}^{M}\sum_{k\in\mathcal{N}_v}\alpha^m_{vk}\mathbf{W}^m\mathbf{h}_k\right)$	$\mathbf{h}^t_v = \mathbf{h}^t_{\mathcal{N}_v}$
Gated Graph Neural Networks	GGNN	$\mathbf{h}^t_{\mathcal{N}_v} = \sum_{k\in\mathcal{N}_v}\mathbf{h}^{t-1}_k + \mathbf{b}$	$\mathbf{z}^t_v = \sigma(\mathbf{W}^z\mathbf{h}^t_{\mathcal{N}_v} + \mathbf{U}^z\mathbf{h}^{t-1}_v)$ $\mathbf{r}^t_v = \sigma(\mathbf{W}^r\mathbf{h}^t_{\mathcal{N}_v} + \mathbf{U}^r\mathbf{h}^{t-1}_v)$ $\widetilde{\mathbf{h}}^t_v = \tanh(\mathbf{W}\mathbf{h}^t_{\mathcal{N}_v} + \mathbf{U}(\mathbf{r}^t_v \odot \mathbf{h}^{t-1}_v))$ $\mathbf{h}^t_v = (1-\mathbf{z}^t_v)\odot\mathbf{h}^{t-1}_v + \mathbf{z}^t_v \odot \widetilde{\mathbf{h}}^t_v$
Graph LSTM	Tree LSTM (Child sum)	$\mathbf{h}^t_{\mathcal{N}_v} = \sum_{k\in\mathcal{N}_v}\mathbf{h}^{t-1}_k$	$\mathbf{i}^t_v = \sigma(\mathbf{W}^i\mathbf{x}^t_v + \mathbf{U}^i\mathbf{h}^t_{\mathcal{N}_v} + \mathbf{b}^i)$ $\mathbf{f}^t_{vk} = \sigma(\mathbf{W}^f\mathbf{x}^t_v + \mathbf{U}^f\mathbf{h}^{t-1}_k + \mathbf{b}^f)$ $\mathbf{o}^t_v = \sigma(\mathbf{W}^o\mathbf{x}^t_v + \mathbf{U}^o\mathbf{h}^t_{\mathcal{N}_v} + \mathbf{b}^o)$ $\mathbf{u}^t_v = \tanh(\mathbf{W}^u\mathbf{x}^t_v + \mathbf{U}^u\mathbf{h}^t_{\mathcal{N}_v} + \mathbf{b}^u)$ $\mathbf{c}^t_v = \mathbf{i}^t_v \odot \mathbf{u}^t_v + \sum_{k\in\mathcal{N}_v}\mathbf{f}^t_{vk}\odot\mathbf{c}^{t-1}_k$ $\mathbf{h}^t_v = \mathbf{o}^t_v \odot \tanh(\mathbf{c}^t_v)$
	Tree LSTM (N-ary)	$\mathbf{h}^{ti}_{\mathcal{N}_v} = \sum_{l=1}^{K}\mathbf{U}^i_l\mathbf{h}^{t-1}_{vl}$ $\mathbf{h}^{tf}_{\mathcal{N}_v k} = \sum_{l=1}^{K}\mathbf{U}^f_{kl}\mathbf{h}^{t-1}_{vl}$ $\mathbf{h}^{to}_{\mathcal{N}_v} = \sum_{l=1}^{K}\mathbf{U}^o_l\mathbf{h}^{t-1}_{vl}$ $\mathbf{h}^{tu}_{\mathcal{N}_v} = \sum_{l=1}^{K}\mathbf{U}^u_l\mathbf{h}^{t-1}_{vl}$	$\mathbf{i}^t_v = \sigma(\mathbf{W}^i\mathbf{x}^t_v + \mathbf{h}^{ti}_{\mathcal{N}_v} + \mathbf{b}^i)$ $\mathbf{f}^t_{vk} = \sigma(\mathbf{W}^f\mathbf{x}^t_v + \mathbf{h}^{tf}_{\mathcal{N}_v k} + \mathbf{b}^f)$ $\mathbf{o}^t_v = \sigma(\mathbf{W}^o\mathbf{x}^t_v + \mathbf{h}^{to}_{\mathcal{N}_v} + \mathbf{b}^o)$ $\mathbf{u}^t_v = \tanh(\mathbf{W}^u\mathbf{x}^t_v + \mathbf{h}^{tu}_{\mathcal{N}_v} + \mathbf{b}^u)$ $\mathbf{c}^t_v = \mathbf{i}^t_v \odot \mathbf{u}^t_v + \sum_{l=1}^{K}\mathbf{f}^t_{vl}\odot\mathbf{c}^{t-1}_{el}$ $\mathbf{h}^t_v = \mathbf{o}^t_v \odot \tanh(\mathbf{c}^t_v)$
	Graph LSTM in [44]	$\mathbf{h}^{ti}_{\mathcal{N}_v} = \sum_{k\in\mathcal{N}_v}\mathbf{U}^i_{m(v,k)}\mathbf{h}^{t-1}_k$ $\mathbf{h}^{to}_{\mathcal{N}_v} = \sum_{k\in\mathcal{N}_v}\mathbf{U}^o_{m(v,k)}\mathbf{h}^{t-1}_k$ $\mathbf{h}^{tu}_{\mathcal{N}_v} = \sum_{k\in\mathcal{N}_v}\mathbf{U}^u_{m(v,k)}\mathbf{h}^{t-1}_k$	$\mathbf{i}^t_v = \sigma(\mathbf{W}^i\mathbf{x}^t_v + \mathbf{h}^{ti}_{\mathcal{N}_v} + \mathbf{b}^i)$ $\mathbf{f}^t_{vk} = \sigma(\mathbf{W}^f\mathbf{x}^t_v + \mathbf{U}^f_{m(v,k)}\mathbf{h}^{t-1}_k + \mathbf{b}^f)$ $\mathbf{o}^t_v = \sigma(\mathbf{W}^o\mathbf{x}^t_v + \mathbf{h}^{to}_{\mathcal{N}_v} + \mathbf{b}^o)$ $\mathbf{u}^t_v = \tanh(\mathbf{W}^u\mathbf{x}^t_v + \mathbf{h}^{tu}_{\mathcal{N}_v} + \mathbf{b}^u)$ $\mathbf{c}^t_v = \mathbf{i}^t_v \odot \mathbf{u}^t_v + \sum_{k\in\mathcal{N}_v}\mathbf{f}^t_{vk}\odot\mathbf{c}^{t-1}_k$ $\mathbf{h}^t_v = \mathbf{o}^t_v \odot \tanh(\mathbf{c}^t_v)$

Two view of graph neural nets

- Spectral networks
  - Rely on the eigen-decomposition of the Laplacian matrix
  - Any perturbation to a graph results in a change of eigenbasis
  - Need to load the whole graph into the memory to perform graph convolution
- Spatial networks
  - Take the aggregation of the central node representation and its neighbors representation

Spectral networks

- What the heck is “spectral” now?
  - It simply means *decomposing* a signal/audio/image/graph into a combination (usually, a sum) of simple elements (wavelets, graphlets)
  - these simple elements are usually orthogonal, i.e. mutually linearly independent, and therefore form a basis

Spectral networks

- “Spectral” in signal processing
  - Usually means applying the Fourier Transform to a signal
  - Decompose it into elementary sine and cosine waves of different frequencies
- “Spectral” in Graph signal processing
  - Eigen-decomposition of the graph Laplacian L
  - Decompose it into communities

Spectral Networks

- Given a graph, G = (V, E)
  - X, a feature description for every node (N * D)
  - Adjacency matrix (A)
- Output Z, node level output (N * F)

Spectral Networks

- Every neural network layer can be written as

$$H^{(l+1)} = f(H^{(l)}, A)$$
$$H^{(0)} = X$$

- For example:

$$f(H^{(l)}, A) = \sigma(AH^{(l)}W^{(l)})$$

- For every node, sum up all the feature vectors of all neighboring nodes <u>but not the node itself</u>

Spectral Networks

- Fix it by adding the identity matrix

$$\hat{A} = A + I$$

- If we want to add multiple layers to one another, they have to have similar scales
- Need to normalize the adjacency matrix, such that all rows of the adjacency matrix must add up to one

$$\hat{D}^{-0.5}\hat{A}\hat{D}^{-0.5}$$

- $\hat{D}$ is the diagonal node degree matrix of $\hat{A}$

Spectral Networks

- Putting it all together

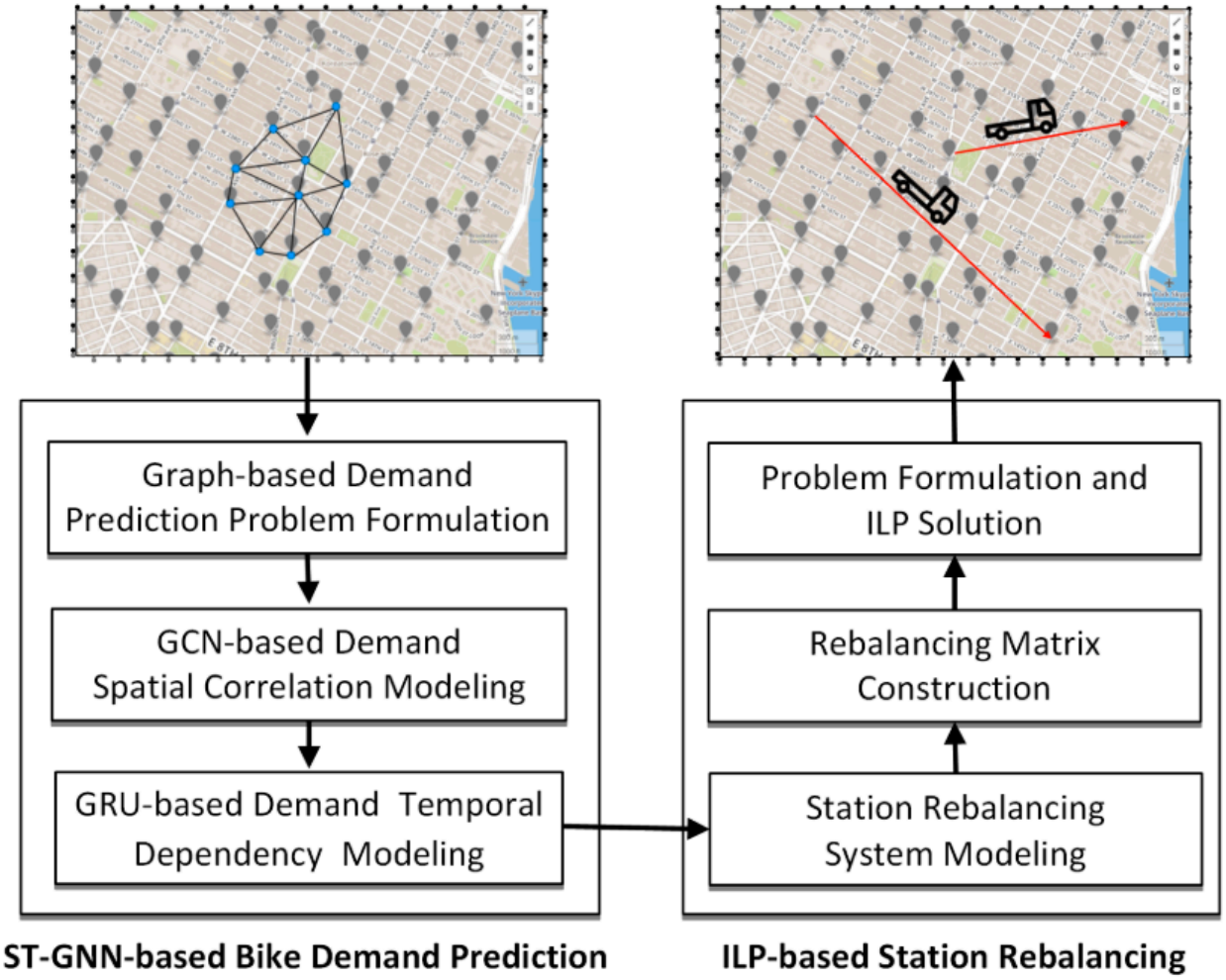
$$f(H^{(l)}, A) = \sigma(\hat{D}^{-0.5}\hat{A}\hat{D}^{-0.5}H^{(l)}W^{(l)})$$

- Compare with;

$$f(H^{(l)}, A) = \sigma(AH^{(l)}W^{(l)})$$

BikeNet: Accurate Bike Demand Prediction Using Graph Neural Networks for Station Rebalancing

Ruiying Guo[1], Zhihan Jiang[1], Jingchun Huang[1], Jianrong Tao[2], Cheng Wang[1], Jonathan Li[1,3], Longbiao Chen*[1]

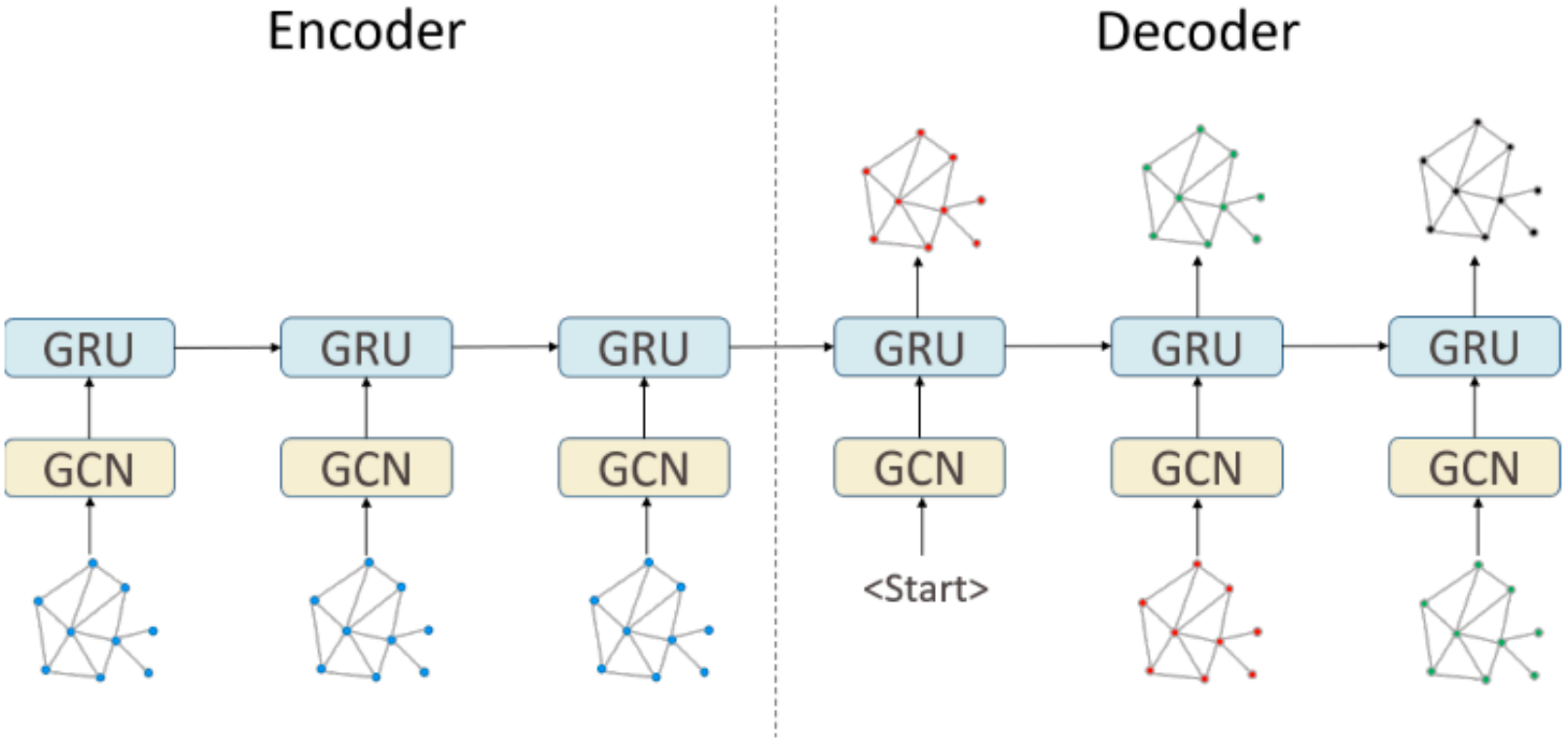


Demand prediction for rebalancing

Demand prediction for rebalancing

- Represent the bike sharing system as a weighted undirected spatiotemporal graph
- Node v's message is bike demand v during [t,t +Δt]
- Edges: distance between stations
  - Could use correlations
  - Mutual information
  - Etc.

Demand prediction for rebalancing



Demand prediction for rebalancing

$$\text{minimize} \quad C^T x \tag{3}$$

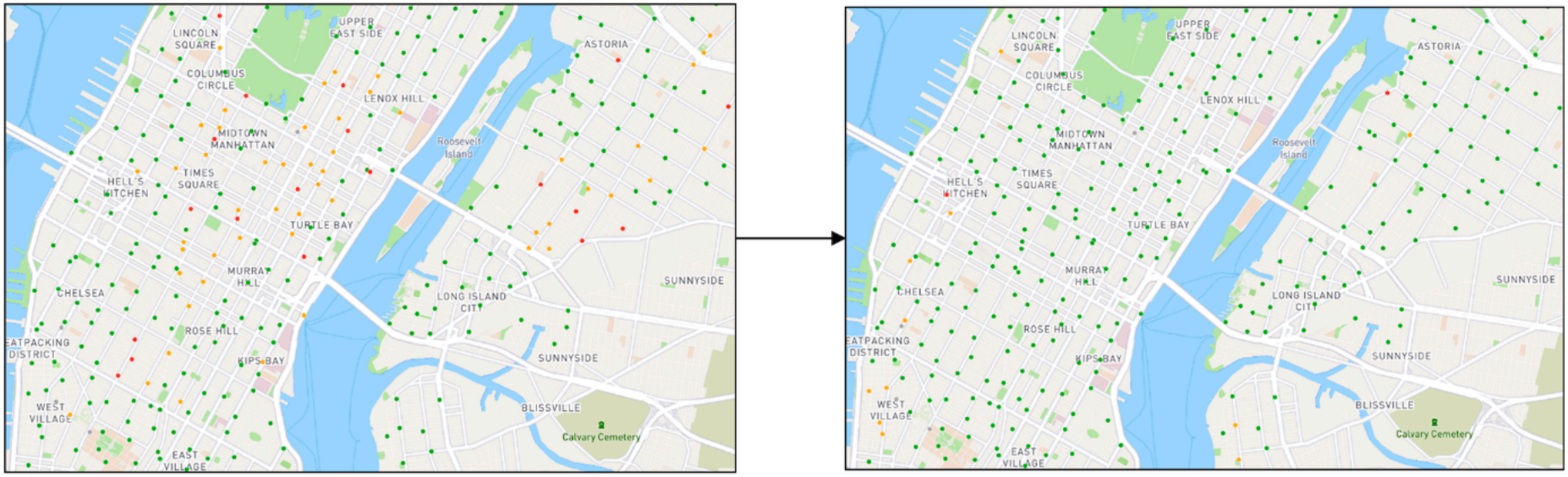
subject to

$$\sum_{j=1}^{l} a_{ij} x_j \leq 0 (i = 1, ..., n+m) \tag{4}$$

$$x_j (j = 1, ..., n+m) \geq 0 \tag{5}$$

$$\sum_{i=1}^{n+m} \sum_{j=1}^{l} a_{ij} x_j = 2min(n, m) \tag{6}$$

Demand prediction for rebalancing



References

CMU Deep learning course 2019
Bhiksha Raj