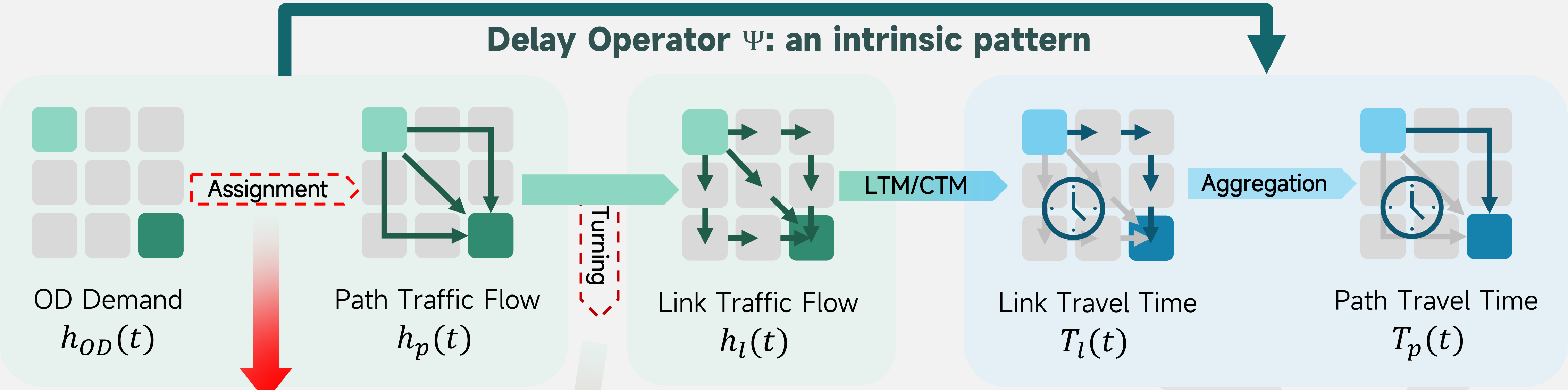
A Neural Network Surrogate Model of Delay Operator Using Link-to-link Segment

Dinghan Liu
Tsinghua Univ.

1 Introduction

The delay operator Ψ is a mapping that relates
Departure Rates to **Travel Times**

$$\boldsymbol{\Psi}(h) \approx \left(\boldsymbol{\Psi}_{r,i}(h) : r \in \mathbf{R}, \mathrm{i} = 1, \dots, n\right) \in \boldsymbol{R}_{+}^{n\times|\boldsymbol{P}|}$$



Bottleneck 1
Hard to **capture driver's decision set** $\Delta_i(t)$ **and apply BPR Function** in dynamic scenarios.

Bottleneck 2
Only using link departure rates would **lose "turning" information.**

Insights

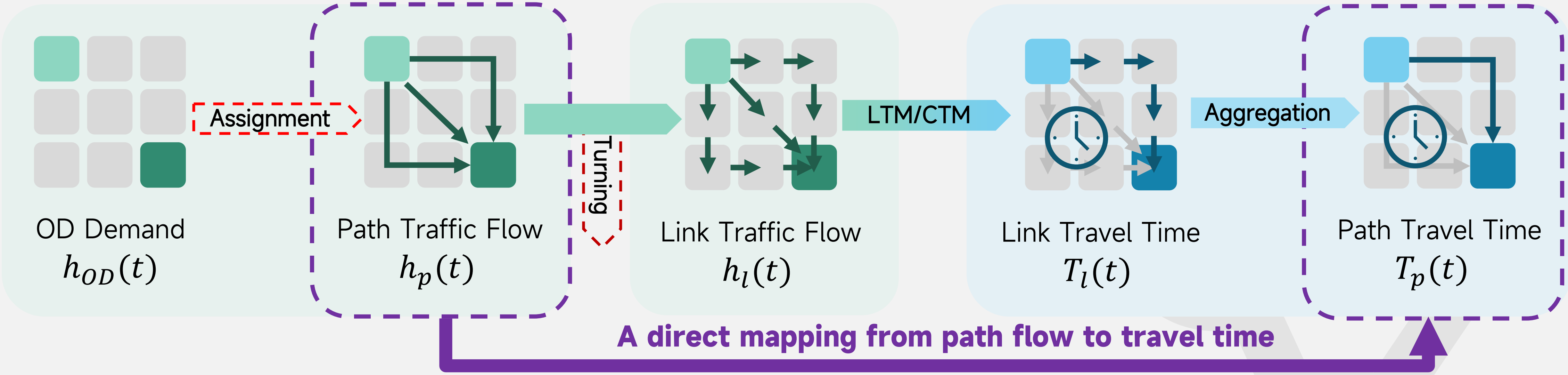
- Create a network's **intrinsic pattern between Departure Rates to Travel Times** (Independent from decision set $\Delta_i(t)$)

Delay Operator

1 Introduction

The delay operator Ψ is a mapping that relates **Departure Rates** to **Travel Times**

$$\boldsymbol{\Psi}(h) \approx \left(\boldsymbol{\Psi}_{r,i}(h) : r \in \mathbf{R}, \mathrm{i} = 1, \dots, n\right) \in \boldsymbol{R}_{+}^{n \times |\boldsymbol{P}|}$$



Drawbacks of State-of-arts

- A direct mapping from **path** flow to travel time is **computationally prohibitive** as there're too many paths.
- **Bad performances** are shown in current NN Model.

Method	RMSE	MAPE
NN	122.62	75.30
NN	105.48	69.18
NN	92.95	60.26

Insights

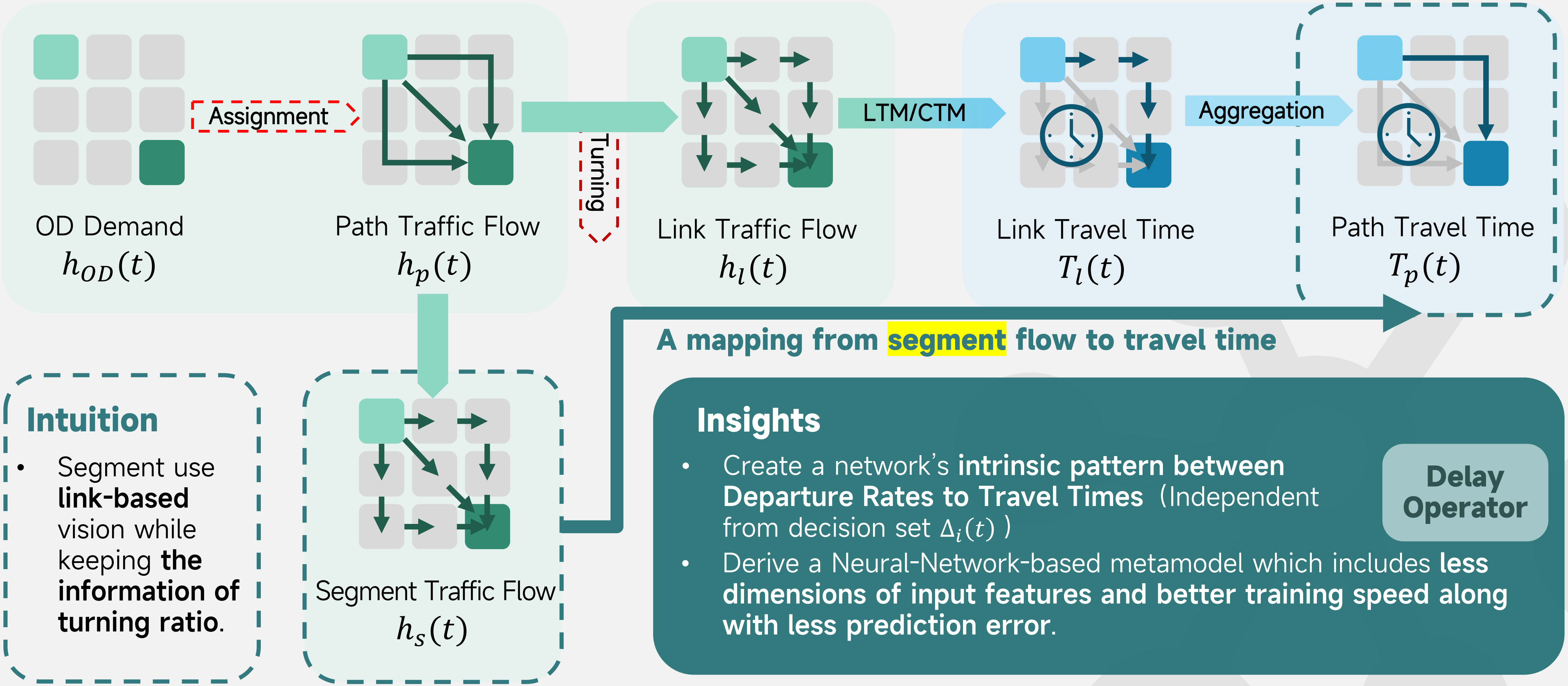
- Create a network's **intrinsic pattern between Departure Rates to Travel Times** (Independent from decision set $\Delta_i(t)$)
- Derive a Neural-Network-based metamodel which includes **less dimensions of input features and better training speed along with less prediction error**.

Delay Operator

1 Introduction

The delay operator Ψ is a mapping that relates **Departure Rates** to **Travel Times**

$$\boldsymbol{\Psi}(h) \approx \left(\boldsymbol{\Psi}_{r,i}(h) : r \in \mathbf{R}, \mathrm{i} = 1, \dots, n\right) \in \boldsymbol{R}_{+}^{n\times|\boldsymbol{P}|}$$



Intuition

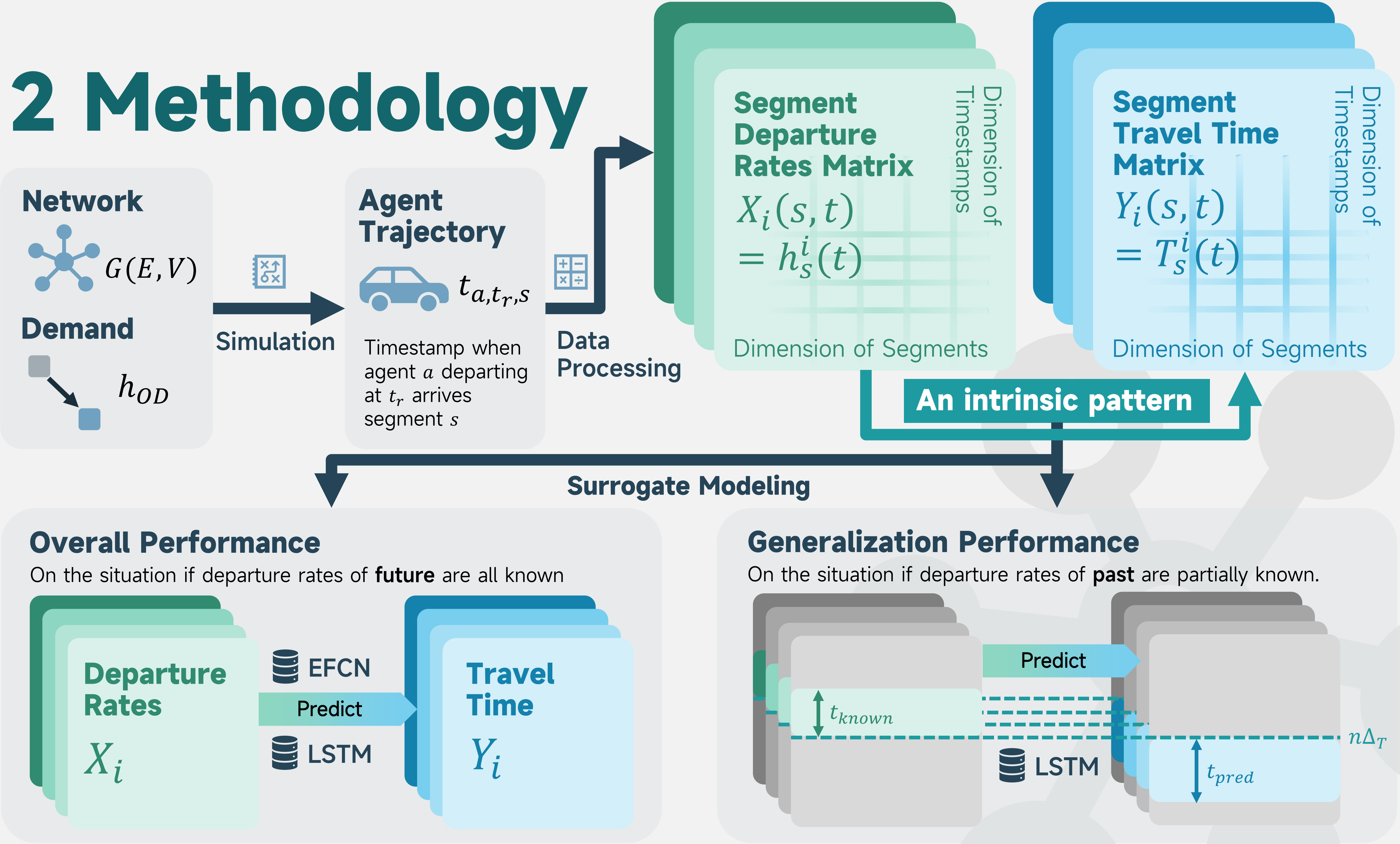
- Segment use **link-based** vision while keeping **the information of turning ratio.**

Insights

- Create a network's **intrinsic pattern between Departure Rates to Travel Times** (Independent from decision set $\Delta_i(t)$)
- Derive a Neural-Network-based metamodel which includes **less dimensions of input features and better training speed along with less prediction error.**

Delay Operator

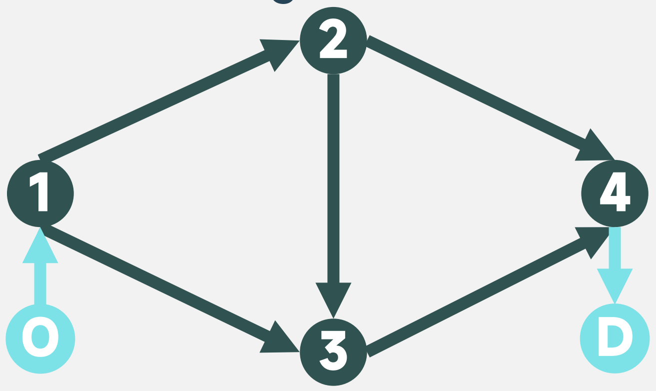
2 Methodology



2 Methodology

2.1 Link-to-link Segment

Considering OD from 1 to 4



Segment = "A certain link" + "Turn"
(Denoted as [from_node_id, to_node_id, direction])

Path	1→2→3→4
Link	1→2; 2→3; 3→4
Segment	1→2→4=(O,1,2); (1,2,3); (2,3,4); (3,4,D)

Intuition

- In a $n \times n$ network, the number of paths is proportional to $n!$ while that of links to n^2, which **greatly reduces input amounts.**
- Keeping "turn" information can **keep the information of driver's decision.**

$$x_{\lambda_0}^{t} = \sum_{r \in R_\lambda} \sum_{t_r \in I_t} q_r^{t_r} \delta_{r,t,0}^{\lambda,t_r}$$

$\delta_{r,t,0}^{\lambda,t_r} = 1 \text{ if } g(g^{-1}(t_r) + \sum_{v \in \Lambda} T_v^0) = t); o.w. = 0$

$$T_r^{t_r} = \sum_{\lambda \in \Lambda_r} T_\lambda^{h_{\lambda,r}^{t_r}}$$

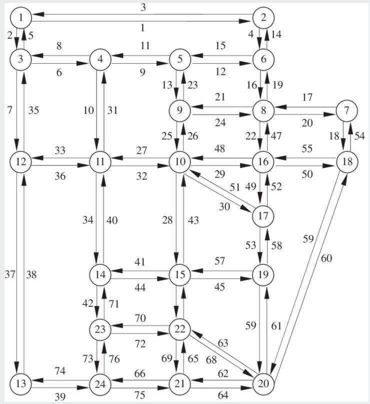
$$h_{\lambda,r}^{t_r} = g(g^{-1}(t_r) + \sum_{\lambda' \in \Lambda_{r,\lambda}} T_{\lambda'}^{h_{\lambda',r}^{t_r}})$$

$g(T_{index}) = t_r$, as $g(x)$ connects time index and real timestamp.

2 Methodology

2.2 Dataset Generation

Network Definition



Sioux-Falls Network

24 Nodes
76 Links
330 Segments
528 OD-Pairs
6,180 Paths

Timespan
08:00-10:00
08:30 set to be peak

1

Demand Generation

(Step 1) Generate OD Demand

- Set basic demand as $h = 700\ vec/h$
- Set lower bound c_u
- Set sample number n
- Use LHS Sampling sample n coefficients between $[c_u, \frac{1}{c_u}]$, get co-array C
- Calculate OD array H as hC

(Step 2) Generate OD Pairs

- Use LHS Sampling sample n OD pairs in all 528 OD-pairs

(Step 3) Match OD Demand

for each epoch in n
- Match one OD-pair with one OD-demand together
- Modify .csv file as input

2

Simulation

- Based on C++based **DTALite simulator**
- Based on **LTM method**
- Automatically simulate DTA process and DNL process
- Minimum time unit: **1min**

My story with DTALite

During my research, I found some bugs with this simulator. I emailed with developers from ASU and assisted them debug with it.

3

2 Methodology

2.3 Neural Network Structure

Baseline — **An FCN Neural Network used in reference with mild performance**

Intuition 1 — **Reducing the number of parameters: Reduce layers & hidden units**

Intuition 2 — **Preventing overfitting: Introduce Regularization, Learning rate descending, smooth criterion**

Intuition 3 — **Considering time sequences: Introduce LSTM & 2dCNN**

Network	Number of Layers	Number of hidden units per layer	Number of Parameters	L2 regularization	Epochs	Learning Rate	Learning Rate Descending	Optimizer	Criterion
Baseline-FCN	3	2500	164,937,980	0.000001	1000	0.0008	\	ADAM	MSE
Enhanced-FCN	3	1,024,512,256	39,705,616	0.000001	1000	0.001	\	ADAM	logMSE
LSTM	2	128	361,710	0.0001	1000	0.001	10% per 50 epoch	ADAM	logMSE
LSTM-2dCNN	1+1	64	163,059	0.0001	1000	0.001	10% per 50 epoch	ADAM	logMSE

3 Numerical analysis

3.1 Efficiency on Link-to-link Segment based model

*: Network and training process requires computationally-prohibited memory.

Method	Parameters(#)	Datasize(#)	RMSE(standardized)	MAPE(%)	Training Time(s)	Epochs(#)	Speed-up Index
Ori-FCN	164937980	128	0.7186	86.99	361.10	54	*
		256	0.6880	88.23	524.66	48	*
		512	0.6165	94.19	1539.21	75	*
EFCN	39705616	128	0.4661	78.35	166.69	118	201.04
		256	0.4262	73.88	420.84	165	167.23
		512	0.4620	73.63	526.23	110	140.00
LSTM	361710	128	0.1471	6.81	181.14	473	9.30
		256	0.0944	6.89	255.85	344	9.45
		512	0.0891	6.94	386.85	259	8.87
LSTM+2DCNN	163059	128	0.1301	7.72	233.62	1000	61.51
		256	0.1154	7.81	391.75	844	59.35
		512	0.0840	7.51	391.95	433	60.44

$$Speed\_up\ index = \frac{t^i_{path}}{t^i_{seg}}$$

i stands for a certain scenario of a training model

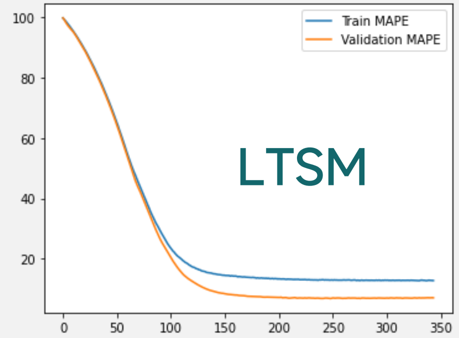
LTSM Models

60x Speed

- By decomposing path to link-to-link segments, calculation and training speed can be around **10^1~10^2** times faster.

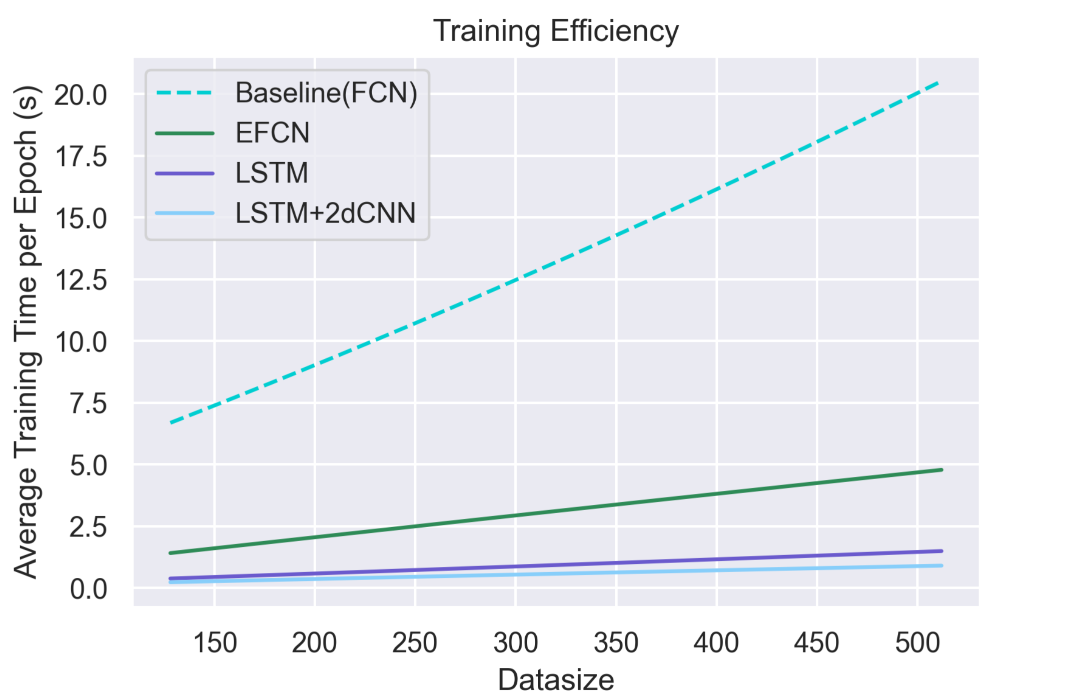
3 Numerical analysis

Baseline



3.2 Training time and Convergence

*: Network and training process requires computationally-prohibited memory.



Method	Training Time(s)	Epochs(#)	Speed-up Index
Ori-FCN	361.10	54	*
	524.66	48	*
	1539.21	75	*
EFCN	166.69	118	201.04
	420.84	165	167.23
	526.23	110	140.00
LSTM	181.14	473	9.30
	255.85	344	9.45
	386.85	259	8.87
LSTM+2DCNN	233.62	1000	61.51
	391.75	844	59.35
	391.95	433	60.44

Training Time

Enhanced FCN **5 Times Faster**

LTSM Models **20 Times Faster**

Conver-gence

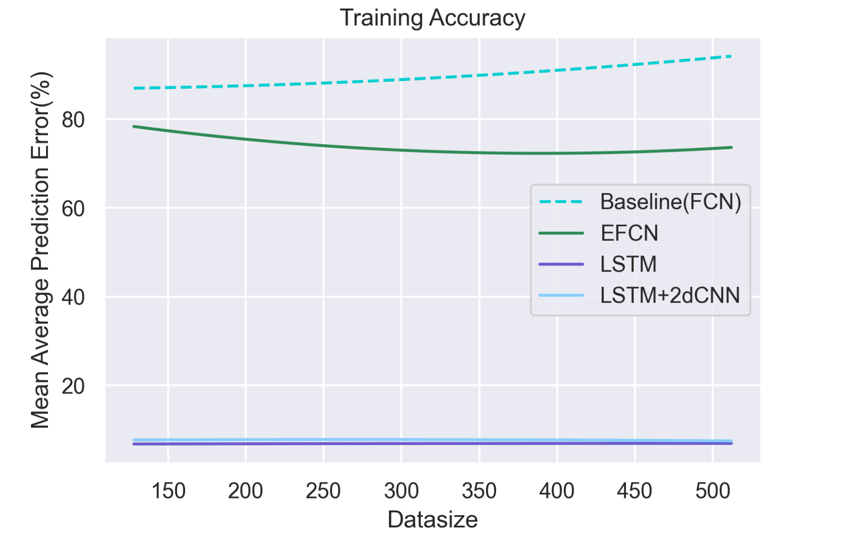
Shows better convergence but slower convergence speed

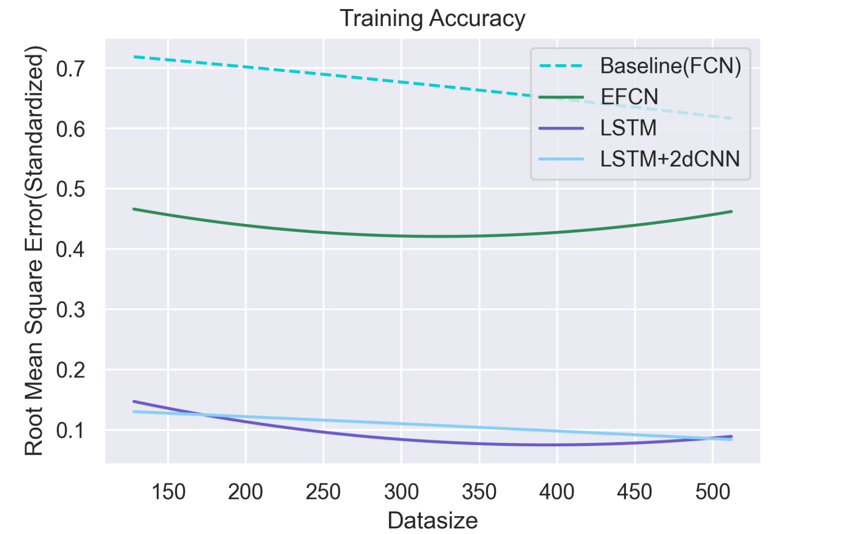
3 Numerical analysis

3.3 Training Accuracy

*: Network and training process requires computationally-prohibited memory.

Method	Parameters(#)	Datasize(#)	RMSE(standardized)	MAPE(%)
Ori-FCN	164937980	128	0.7186	86.99
		256	0.6880	88.23
		512	0.6165	94.19
EFCN	39705616	128	0.4661	78.35
		256	0.4262	73.88
		512	0.4620	73.63
LSTM	361710	128	0.1471	6.81
		256	0.0944	6.89
		512	0.0891	6.94
LSTM+2DCNN	163059	128	0.1301	7.72
		256	0.1154	7.81
		512	0.0840	7.51





$$RMSE = \sqrt{\frac{(y_i - \hat{y}_i)^2}{n}}$$

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right|$$

LTSM Models

7% Error

- EFCN possesses a slightly better accuracy as a result of less parameters.
- LSTM Models possesses **a significantly better accuracy** as result of consideration of time sequences.

3 Numerical analysis

3.4 Generalization Performance

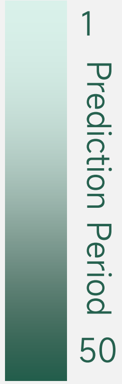
Known Departure rates for a period before a timestamp

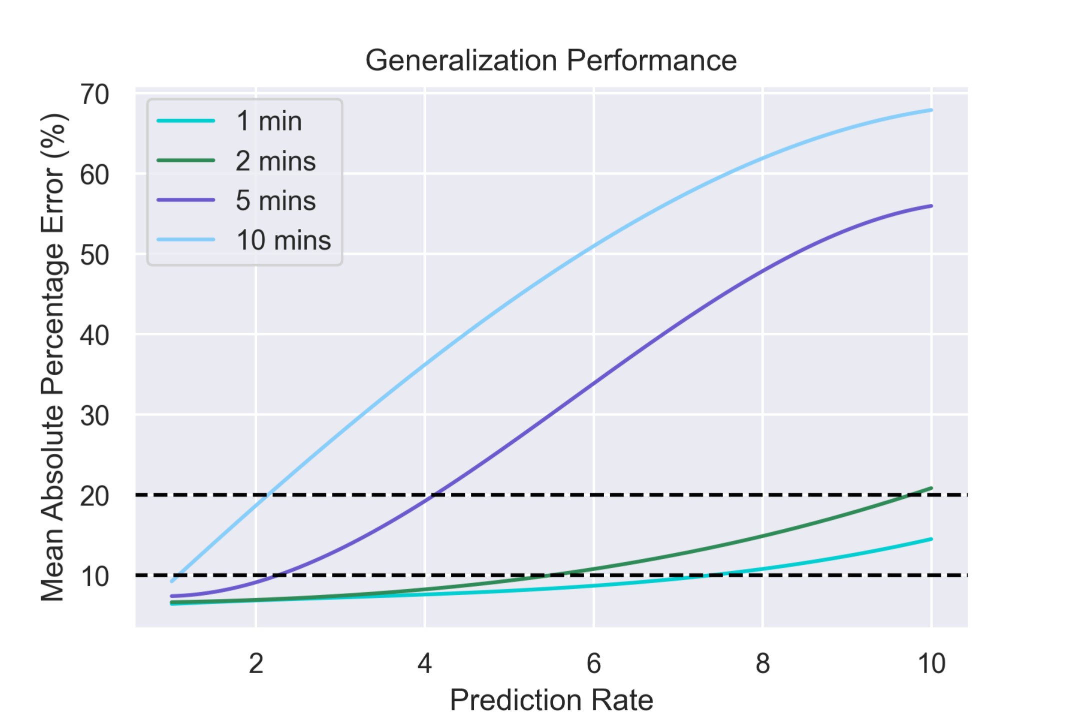
Predicting Departure rates for a period before a timestamp

$$Prediction\ Rate = \frac{t_{prediction}}{t_{known}}$$

* Time offset Selected as 5mins.
** LSTM-Model Applied

MAPE(%)	Known Time Sequence(min)			
Prediction Rate	1	2	5	10
1	6.43	6.64	7.40	9.24
2	6.86	6.93	9.12	18.70
5	8.06	9.33	26.31	56.02
10	14.50	20.84	55.98	77.92





- Using only a period of time to **predict future is feasible**.
- The longer we know, the shorter the prediction time period, and the higher the accuracy.
- Taking 20% as threshold, it can **predict 20 minutes**. Even if the predicted time period is **5 times** the known time period, the model's prediction accuracy is still around 20%.
- This also shows that known traffic data of 1-2 minutes can actually make good DTA predictions.

4 Conclusions

Network Decomposition Breakthrough

- Revolutionized network decomposition into "link + turn"(segment) units, reducing complexity from $o(n!)$ to $o(n^2)$ and achieving a **100x speed increase** in simulation.

Optimized Fully-Connected Layer

- Refined FCN neural network reduces MAE by 10% with a **5x faster** training speed.

LSTM Network Implementation

- Integrated a unidirectional LSTM network, decreasing test set MAE to **7%** and outpacing fully-connected network training speed by **20x**.

Realistic Traffic Time Prediction

- Adapted LSTM to predict future travel times with an accuracy rate within **10%**, effectively using **1-2 minutes** of traffic data for reliable DTA forecasting.

Subsequent Works

- Apply sensor data collected in real urban areas for training and validation.
- Derive a metamodel which can tolerate network changes in topology and capacity.
- Summarize work above and publish in 2024 Spring.

Thanks!

Dinghan Liu
Tsinghua Univ.

References

[1] Song, Wenjing, et al. "Statistical metamodeling of dynamic network loading." Transportation research procedia 23 (2017): 263-282.
[2] Patwary, Ashraf Uz Zaman, Wei Huang, and Hong K. Lo. "A link-to-link segment based metamodel for dynamic network loading." Transportation Research Part C: Emerging Technologies 130 (2021): 103286.
[3] Hamilton, Will, Zhitao Ying, and Jure Leskovec. "Inductive representation learning on large graphs." Advances in neural information processing systems 30 (2017).
[4] Wu, Yuankai, et al. "Spatial aggregation and temporal convolution networks for real-time kriging." arXiv preprint arXiv:2109.12144 (2021).
[5] Yperman, Isaak. "The link transmission model for dynamic network loading." Katholieke Universiteit Leuven 481 (2007): 482.
[6] Fan, Wenbo, et al. "Deep Learning-Based Dynamic Traffic Assignment With Incomplete Origin–Destination Data." Transportation Research Record 2677.3 (2023): 1340-1356.
[7] Zhou, Xuesong, and Jeffrey Taylor. "DTALite: A queue-based mesoscopic traffic simulator for fast model evaluation and calibration." (2014): 961345.