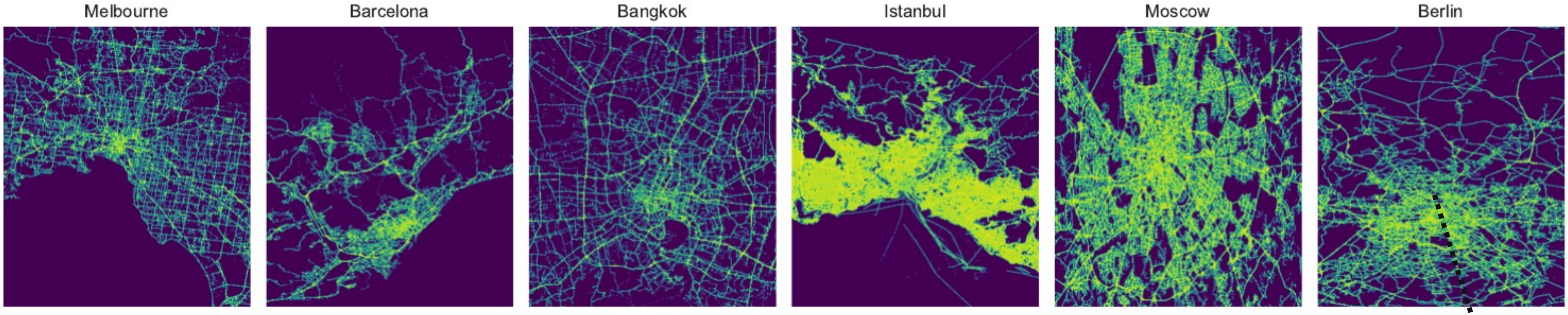
Traffic4Cast 2021

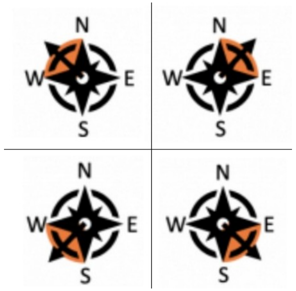
A Graph-based U-Net Model for Predicting Traffic in unseen Cities

Authors:
Malte Schilling, Andrew Melnik, Markus Vieth,
Riza Velioglu, Luca Hermes

Traffic4Cast - Data Format



- Traffic movies from GPS data recorded in 8 different cities
- **Directional speed and volume information**
  - Directions quantized: NE, SE, SW, NW



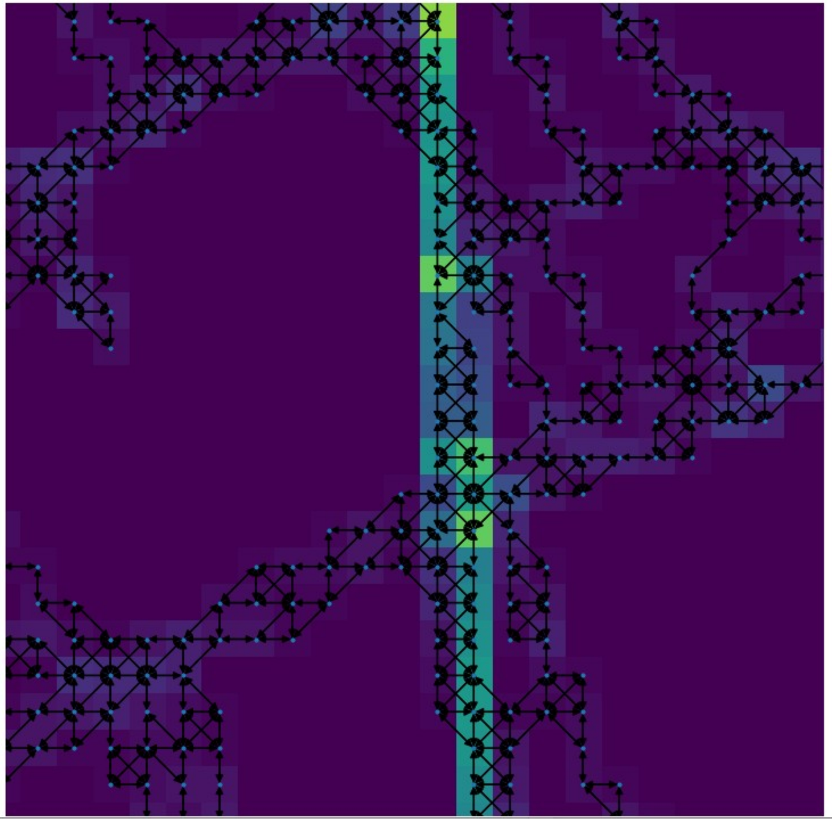
directionality of the traffic speed and volume features

$$\begin{bmatrix} \text{volume}_{NW} \\ \text{speed}_{NW} \\ \text{volume}_{NE} \\ \text{speed}_{NE} \\ \text{volume}_{SE} \\ \text{speed}_{SE} \\ \text{volume}_{SW} \\ \text{speed}_{SW} \end{bmatrix}$$

single pixel feature vector

Traffic4Cast – Graph Data

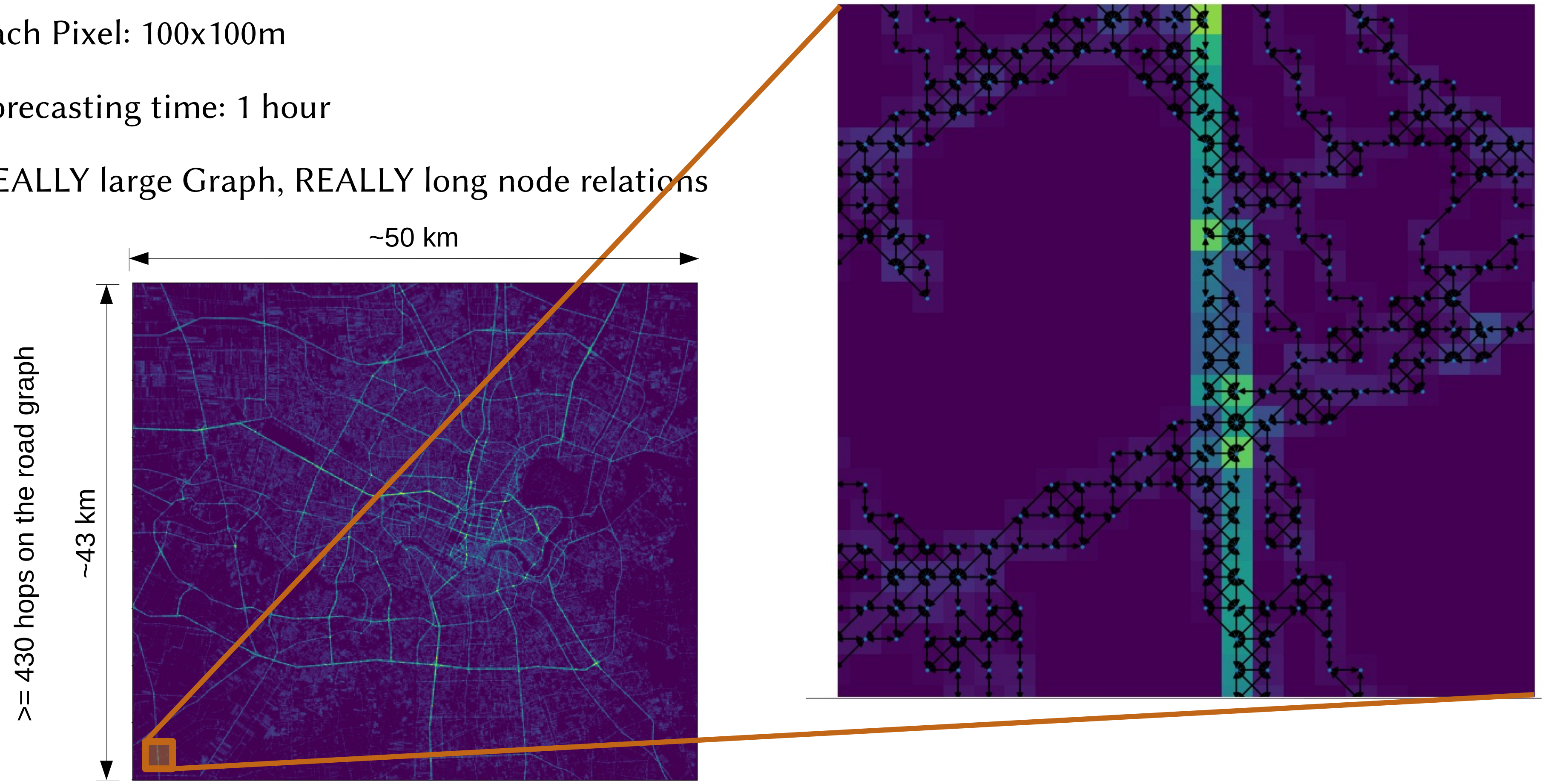
- Graph data:
  - **Nodes**: Pixel information
  - **Edges**: Traffic flow information
- Challenges of this graph for GNNs
  - Long-range interactions (high graph diameter)
  - Encoding full Graph requires hierarchical representations



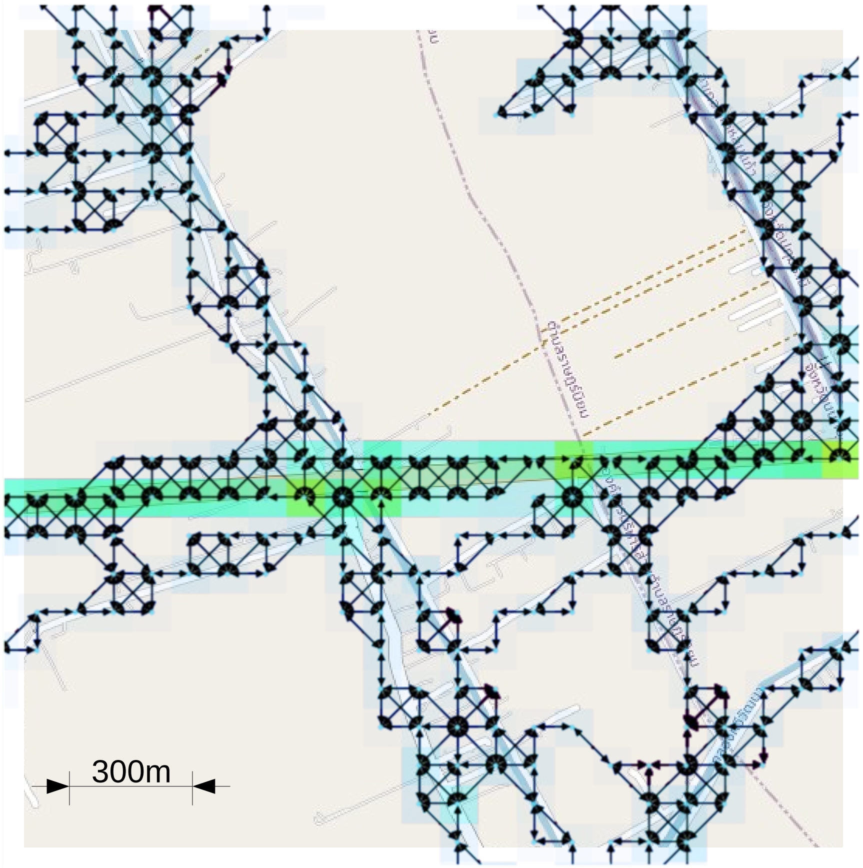
Small window of the Graph of Bangkok

Traffic4Cast – Graph Data

- Each Pixel: 100x100m
- Forecasting time: 1 hour
- REALLY large Graph, REALLY long node relations

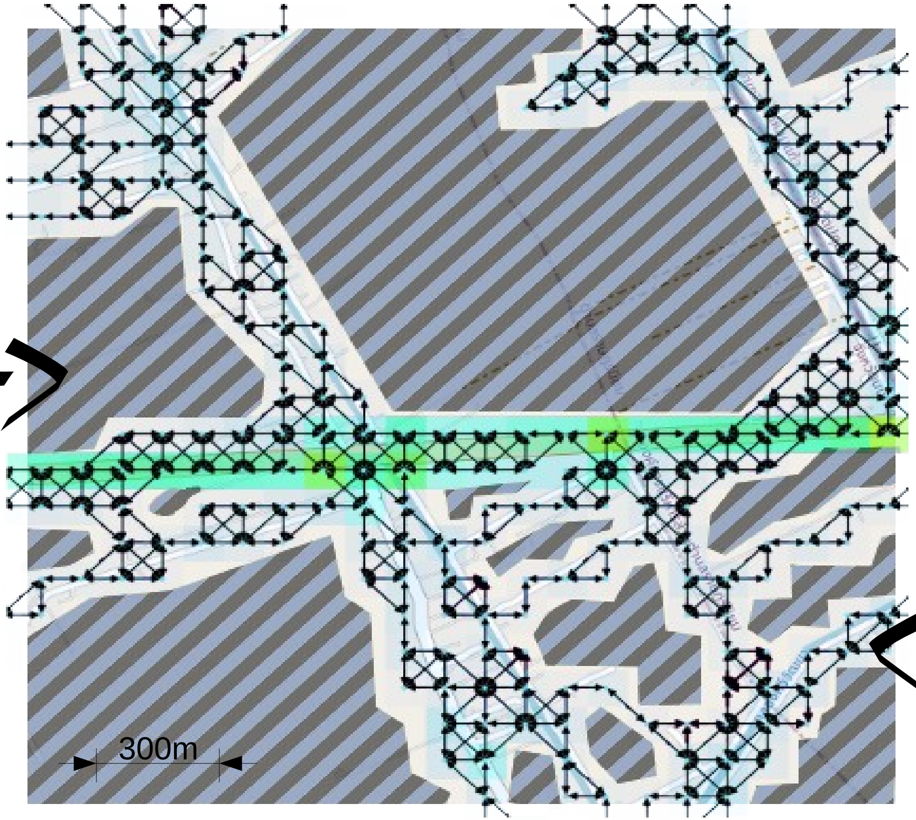



300m


300m

An Advantage of Graphs over Images?

Empty areas directly influence the predictions
of a vision-based model but contain no explicit traffic information



Graph-Based models learn traffic development based on structure and local measurements, which seems closer to the way streets work

Our Approach

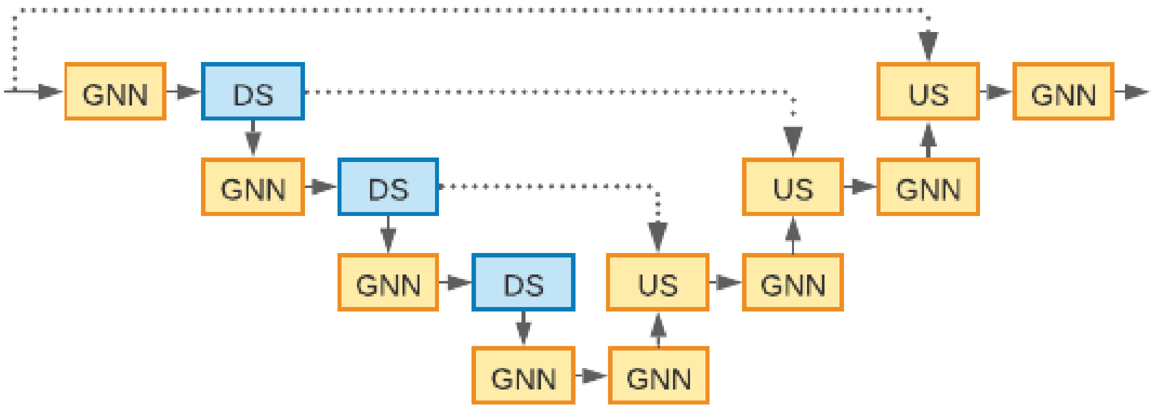
- **Goal**
  - Spatial generalization ⟶ Generalize to unseen cities
- **Observations**
  - U-Net models are amongst the best performing models
  - Visual convolutions (CNN) have limited spatial generalization capacity, but have shown very effective in recent Traffic4Cast challenges on known cities
  - Graph neural networks (GNN) generalize well to unseen cities, but have shown not as effective on known cities as CNN [1]
- **Hypotheses**
  - CNNs encode traffic and **empty spaces, which are city specific**

    ⟶ bad impact on generalization to unseen cities?
  - GNNs only encode traffic and thus learn traffic flow pattern on the underlying road network

    ⟶ This might lead to better generalization to unseen cities

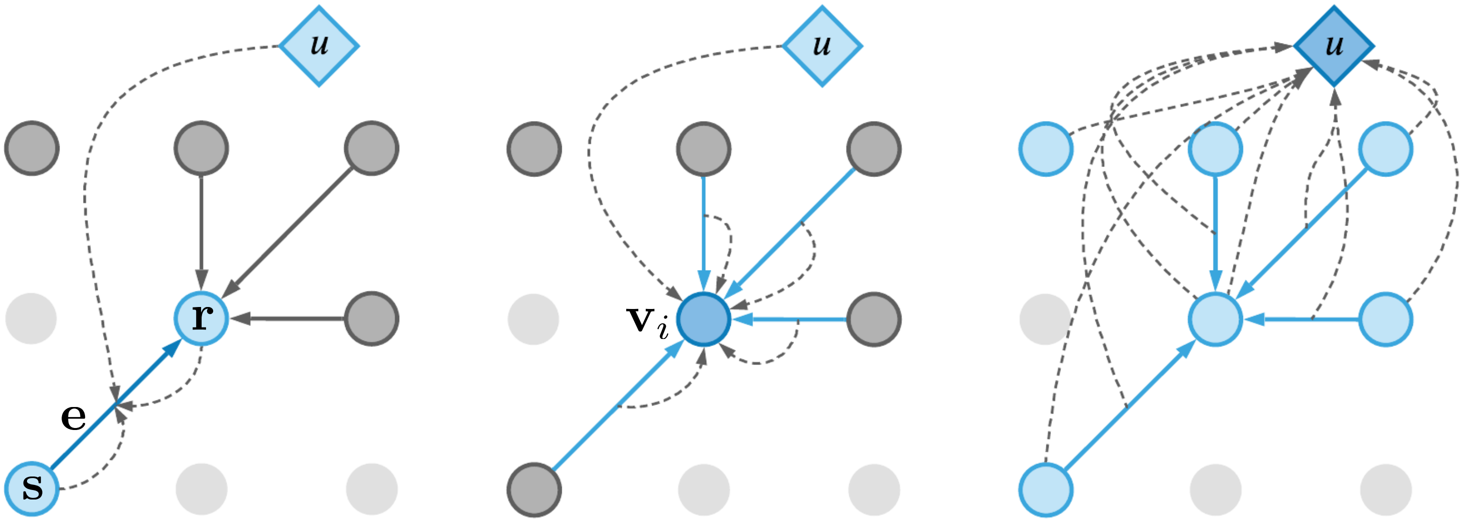
[1] Martin et. al. 2019 [Arxiv 1910.13824]

Our Approach: U-Net-style Architecture



- U-Net style model with GNN layers instead of CNN layers
- Downsampling (DS) / Upsampling (US) were adapted to be applicable to graphs
  - We leverage the **2D position of the pixels** for these operations
  - Up- and Downsampling operations increase the receptive field

Summary: Graph Operation [2]



1. Edge Update:

$$\mathbf{e}'_k = \phi^e([\mathbf{s}, \mathbf{r}, \mathbf{e}_k, \mathbf{u}])$$

2. Node Update:

$$\bar{\mathbf{e}}'_i = \sum_{\forall \mathbf{e}_k \in \mathcal{N}_i} \mathbf{e}'_k$$

$$\mathbf{v}'_i = \phi^v([\mathbf{v}_i, \bar{\mathbf{e}}'_i, \mathbf{u}])$$

3. Global State Update:

$$\bar{\mathbf{v}}' = \sum_{\forall \mathbf{v}_i \in V} \mathbf{v}'_i$$

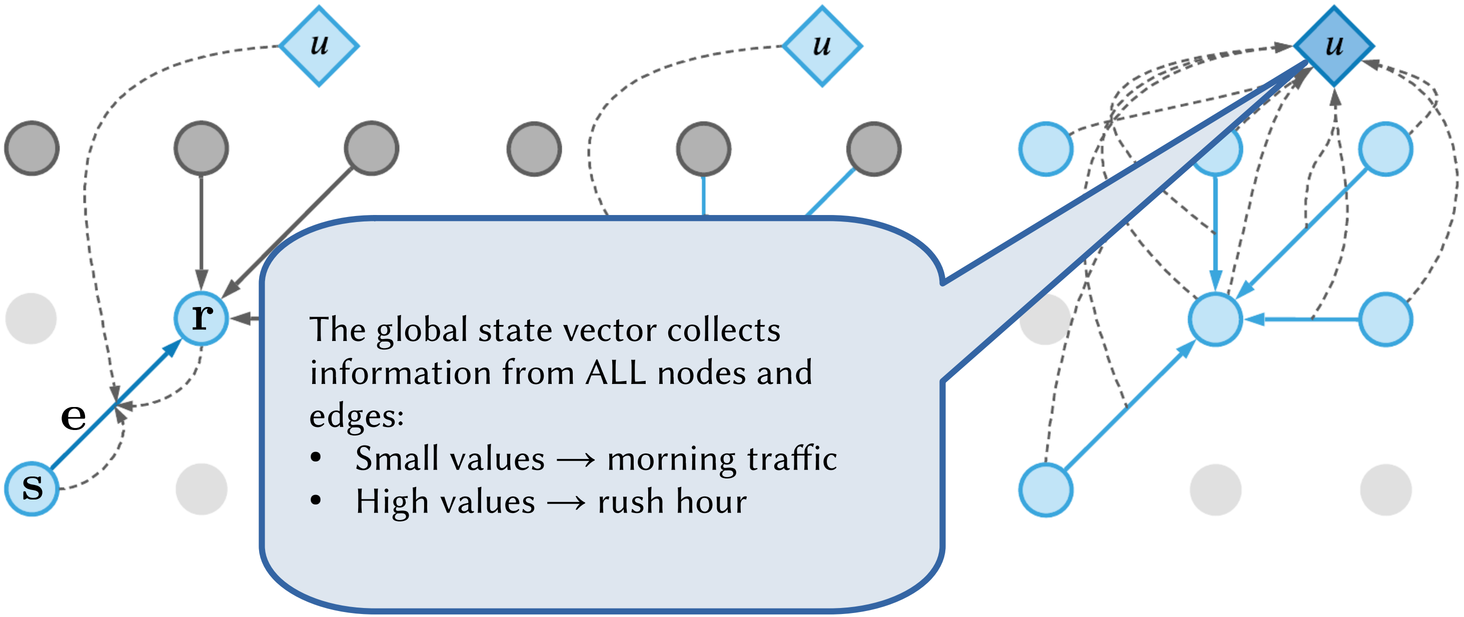
$$\bar{\mathbf{e}}' = \sum_{\forall \mathbf{e}_k \in E} \mathbf{e}'_k$$

$$\mathbf{u}' = \phi^u([\mathbf{u}, \bar{\mathbf{v}}', \bar{\mathbf{e}}'])$$

ϕ = 1-Layer MLPs

[2] Battaglia et. al. 2018 [Arxiv 1806.01261]

Summary: Graph Operation [1]



1. Edge Update:

$$\mathbf{e}'_k = \phi^e([\mathbf{s}, \mathbf{r}, \mathbf{e}_k, \mathbf{u}])$$

2. Node Update:

$$\bar{\mathbf{e}}'_i = \sum_{\forall \mathbf{e}_k \in \mathcal{N}_i} \mathbf{e}'_k$$

$$\mathbf{v}'_i = \phi^v([\mathbf{v}_i, \bar{\mathbf{e}}'_i, \mathbf{u}])$$

3. Global State Update:

$$\bar{\mathbf{v}}' = \sum_{\forall \mathbf{v}_i \in V} \mathbf{v}'_i$$

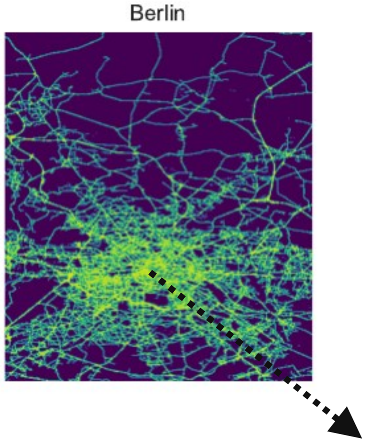
$$\bar{\mathbf{e}}' = \sum_{\forall \mathbf{e}_k \in E} \mathbf{e}'_k$$

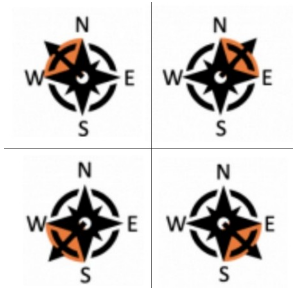
$$\mathbf{u}' = \phi^u([\mathbf{u}, \bar{\mathbf{v}}', \bar{\mathbf{e}}'])$$

ϕ = 1-Layer MLPs

[1] Battaglia et. al. 2018 [Arxiv 1806.01261]

A Problem of Traffic4Cast with GNNs



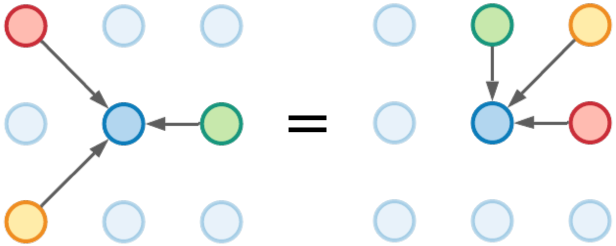


$$\begin{bmatrix} \text{volume}_{NW} \\ \text{speed}_{NW} \\ \text{volume}_{NE} \\ \text{speed}_{NE} \\ \text{volume}_{SE} \\ \text{speed}_{SE} \\ \text{volume}_{SW} \\ \text{speed}_{SW} \end{bmatrix}$$

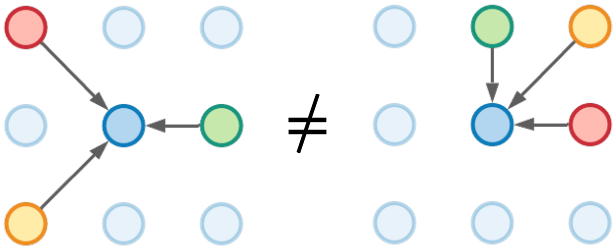
The provided information is **partitioned by global directionality**

GNN
invariant to global directionality
$\longrightarrow$ Fully **Permutation Invariant** Kernel

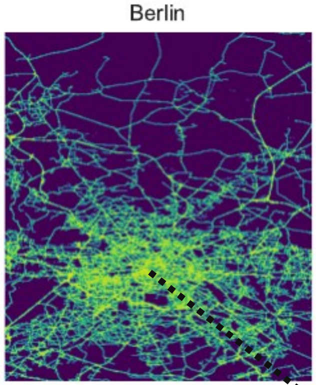
Indistinguishable

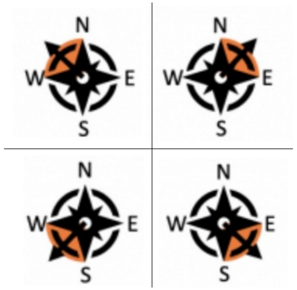


CNN
captures global directionality
$\longrightarrow$ Fully **Permutation Sensitive** Kernel



A Problem of Traffic4Cast with GNNs



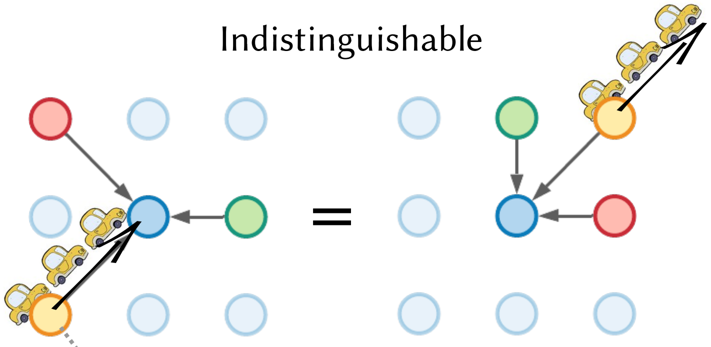


$$\begin{bmatrix} \text{volume}_{NW} \\ \text{speed}_{NW} \\ \text{volume}_{NE} \\ \text{speed}_{NE} \\ \text{volume}_{SE} \\ \text{speed}_{SE} \\ \text{volume}_{SW} \\ \text{speed}_{SW} \end{bmatrix}$$

The provided information is **partitioned by global directionality**

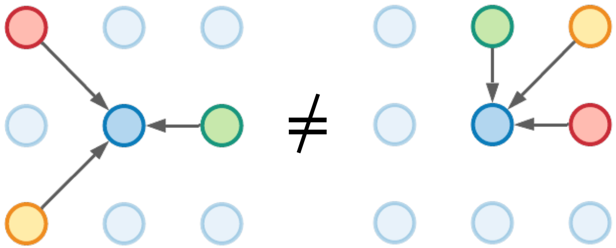
GNN
invariant to global directionality
$\longrightarrow$ Fully **Permutation Invariant** Kernel

Indistinguishable

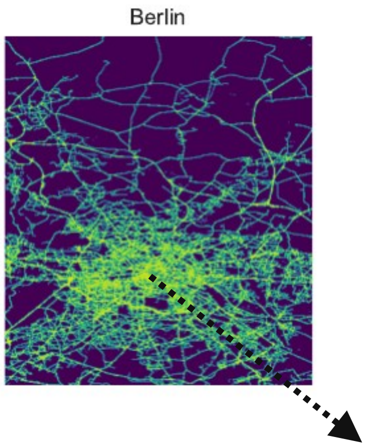


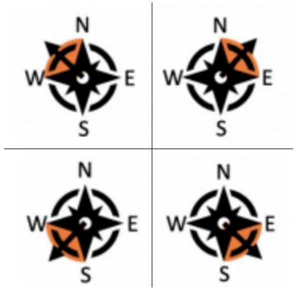
Features of Yellow Node

CNN
captures global directionality
$\longrightarrow$ Fully **Permutation Sensitive** Kernel



A Problem of Traffic4Cast with GNNs

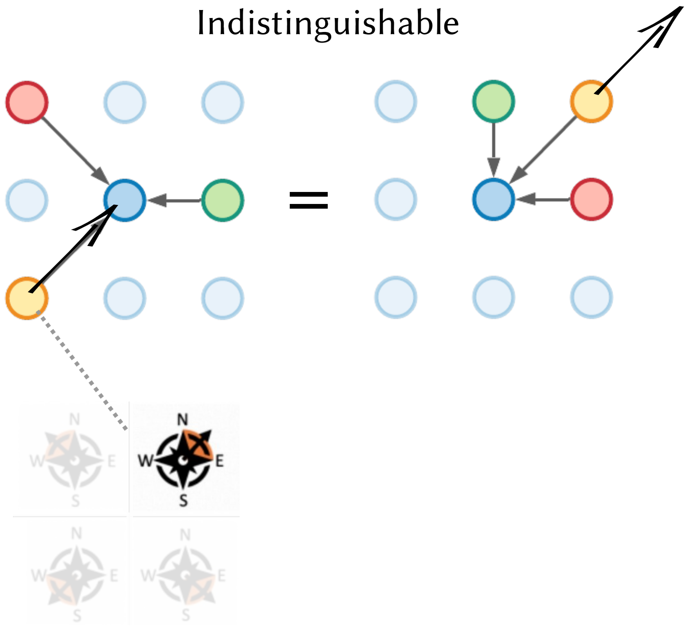




$$\begin{bmatrix} \text{volume}_{NW} \\ \text{speed}_{NW} \\ \text{volume}_{NE} \\ \text{speed}_{NE} \\ \text{volume}_{SE} \\ \text{speed}_{SE} \\ \text{volume}_{SW} \\ \text{speed}_{SW} \end{bmatrix}$$

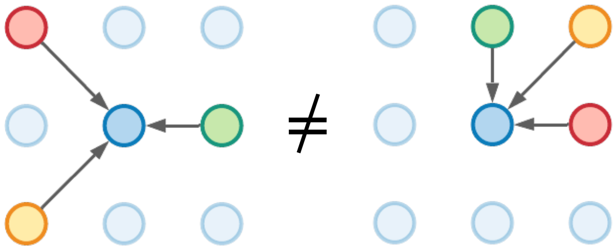
The provided information is **partitioned by global directionality**

GNN
invariant to global directionality
$\longrightarrow$ Fully **Permutation Invariant** Kernel



Features of Yellow Node

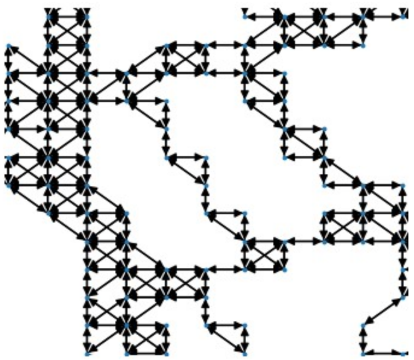
CNN
captures global directionality
$\longrightarrow$ Fully **Permutation Sensitive** Kernel



This is intuitively problematic for graph-based models

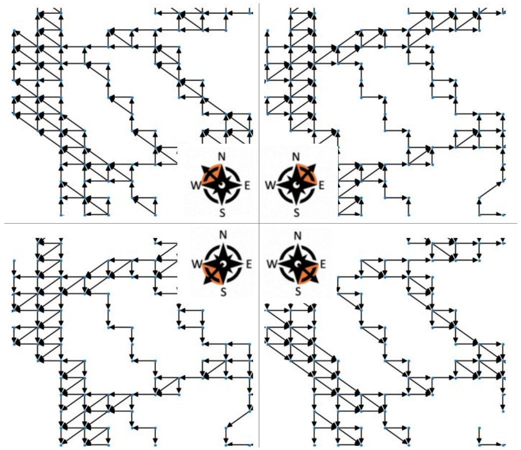
Our Solution: Graph Partitioning

Full Graph



direction-based

subgraphing



Resulting Node Vector

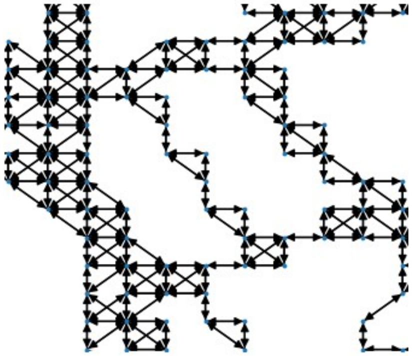
We first split edge set of the graph into four directional subsets

To each subset we apply a separate edge update layer

And accumulate them in the node features depending on the subgraph it belongs to

Our Solution: Graph Partitioning

Full Graph



direction-based

subgraphing

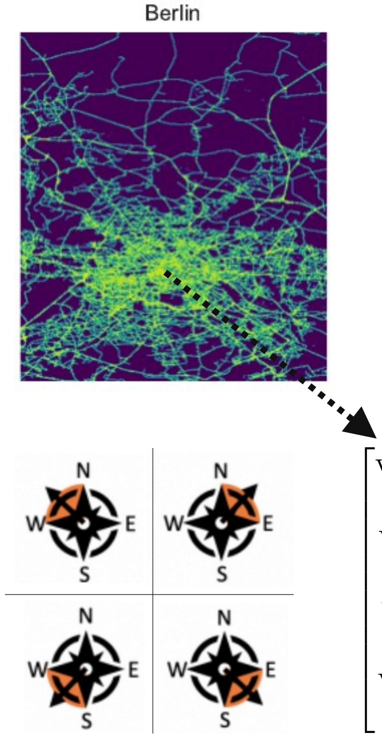
Resulting Node Vector

We first split edge set of the graph into four directional subsets

The node update is then sensitive to the global direction of neighbors

And accumulate them in the node features depending on the subgraph it belongs to

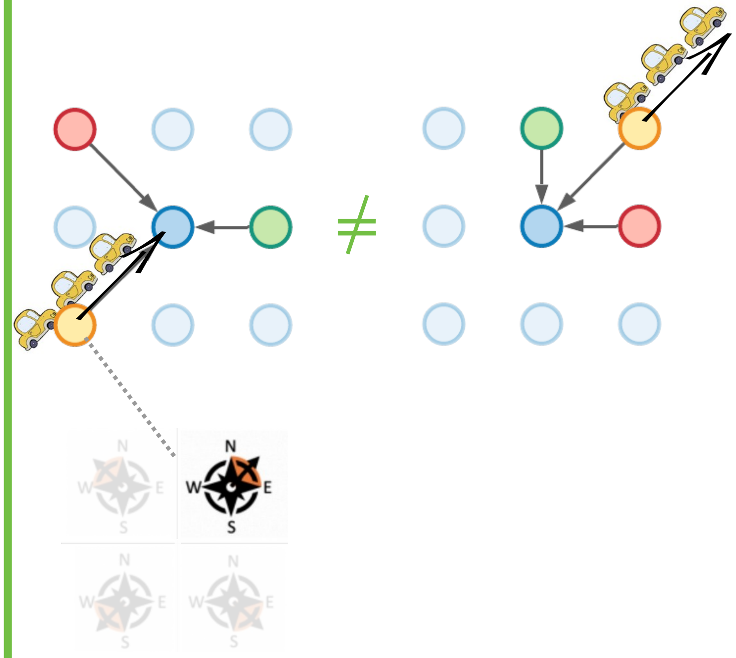
Traffic4Cast with GNN + Subgraphing



$$\begin{bmatrix} \text{volume}_{NW} \\ \text{speed}_{NW} \\ \text{volume}_{NE} \\ \text{speed}_{NE} \\ \text{volume}_{SE} \\ \text{speed}_{SE} \\ \text{volume}_{SW} \\ \text{speed}_{SW} \end{bmatrix}$$

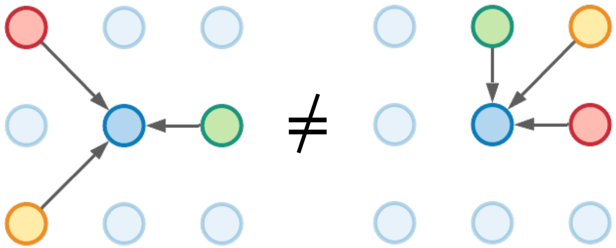
The provided information is **partitioned by global directionality**

GNN + Subgraphing
sensitive to global directionality



Features of Yellow Node

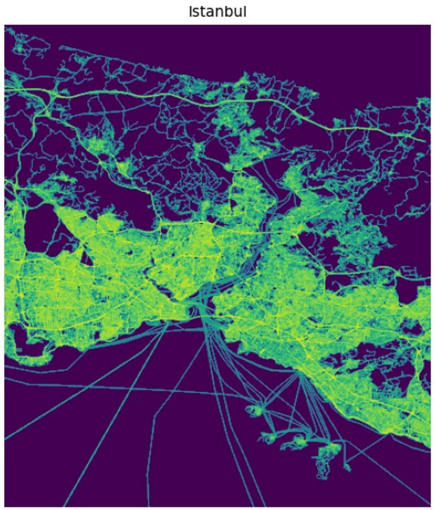
CNN
captures global directionality
→ Fully **Permutation Sensitive** Kernel



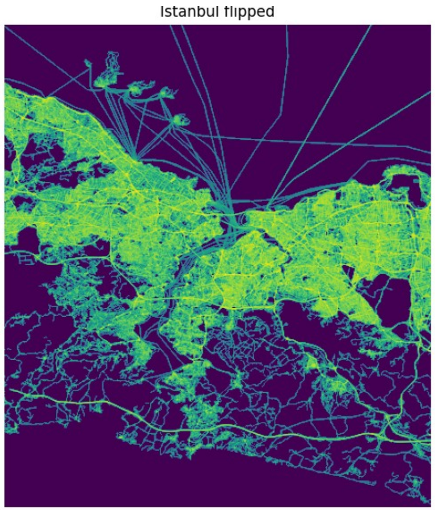
Now the node features are sensitive to neighborhood permutations

Evaluation

Structure included in the training set
Traffic data excluded from the training set



Istanbul from evaluation set S1

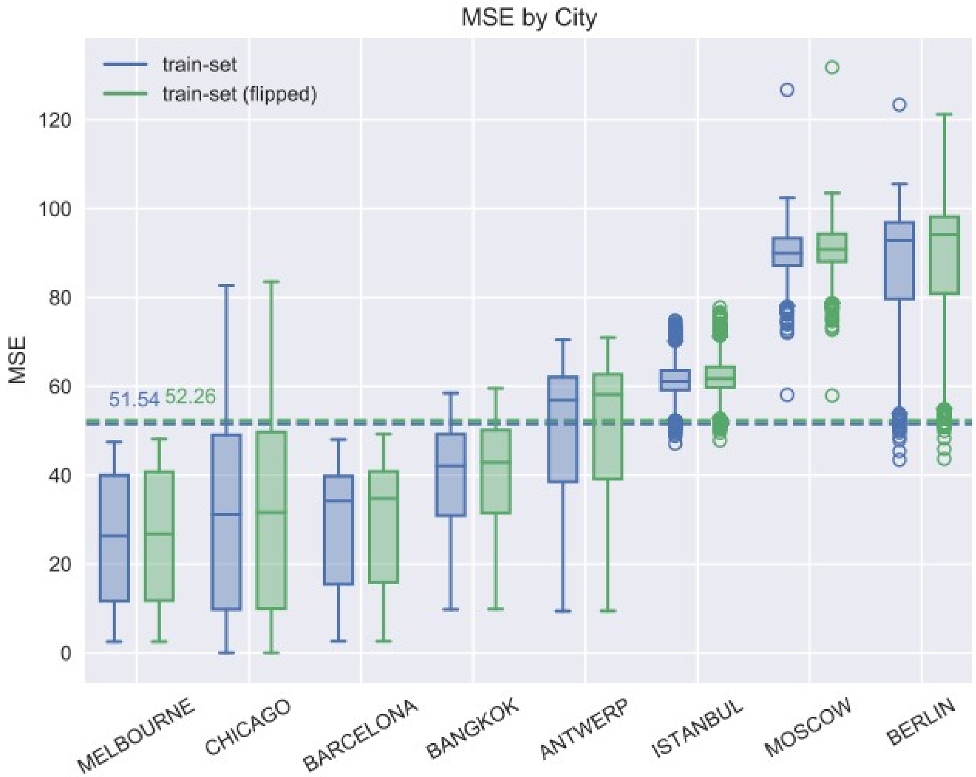


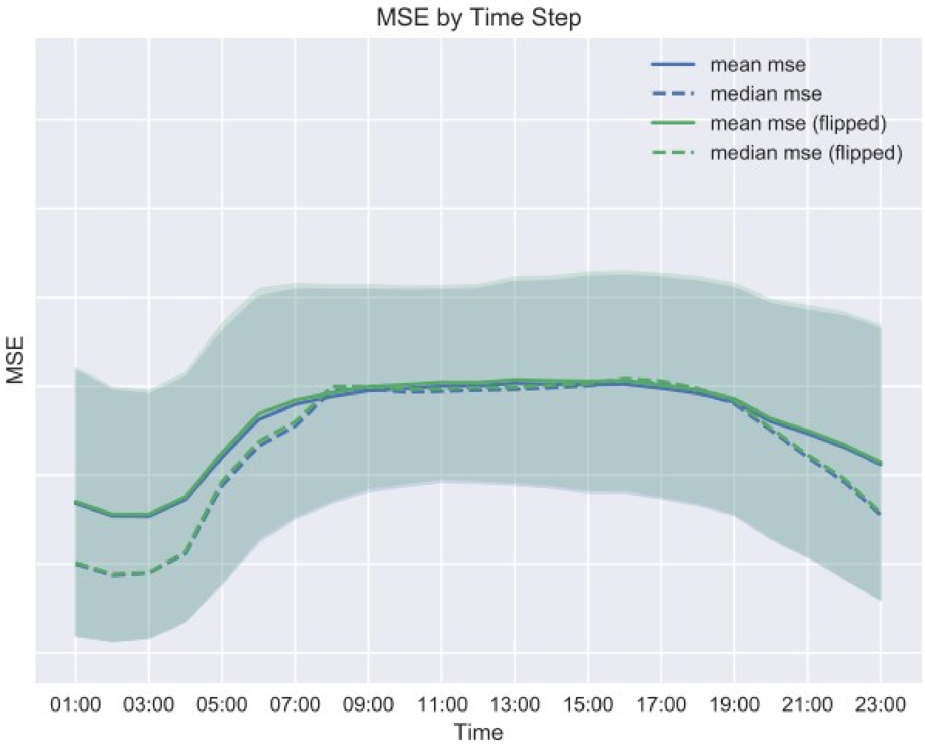
Istanbul from evaluation set S2

Structure and traffic data are excluded from the training set

- The focus is on **spatial generalization**
- Our evaluation setup involves two evaluation datasets to test spatial generalization
  - S1: Subset of the original data (Wed 2019-03-20; all cities)
  - S2: Vertically and horizontally flipped version of the evaluation set S1

Quantitative Results



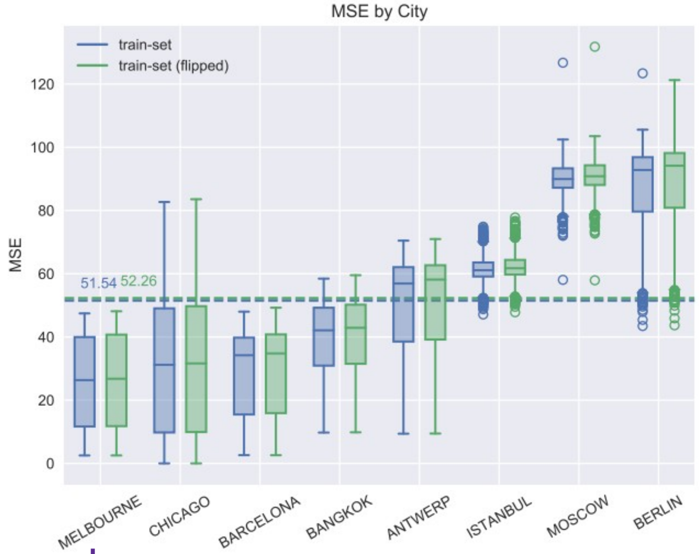


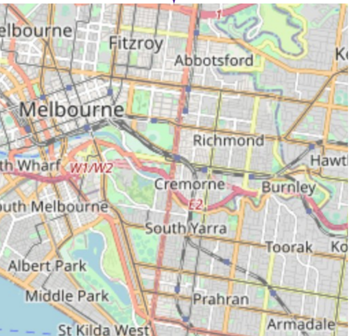
The MSE on both evaluation sets is very similar

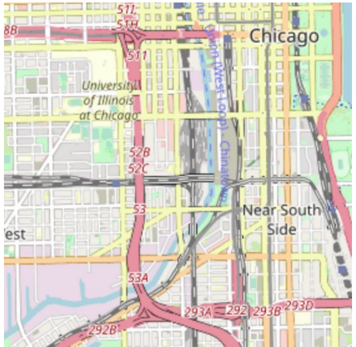
⟶ Indicates good spatial generalization

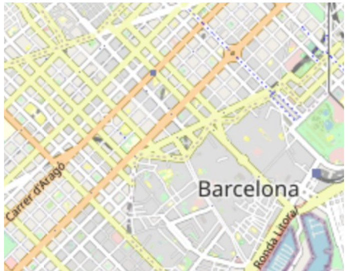
~~Quantitative~~ Results

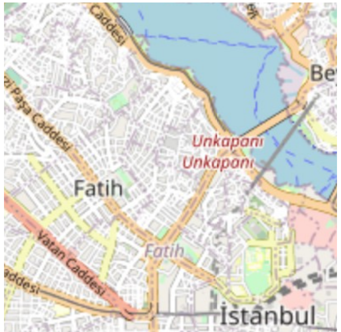
- What determines the model performance?
  - Population Density?
  - *Squareness* of the road network relevant for performance?

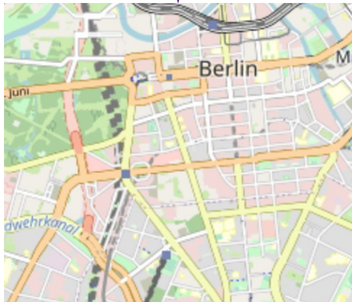




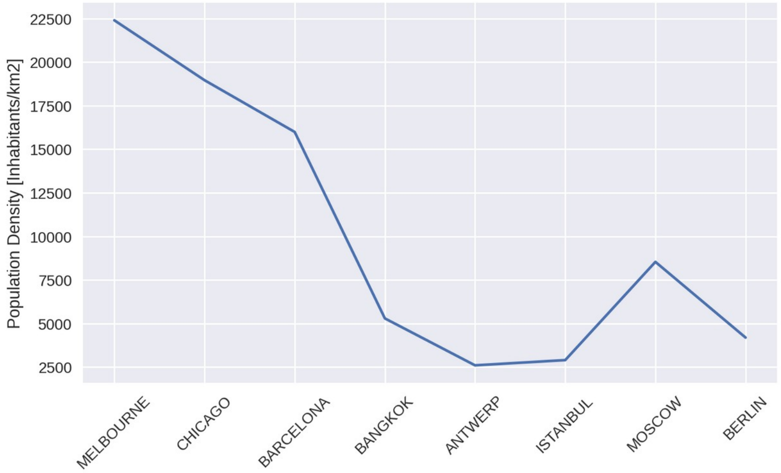


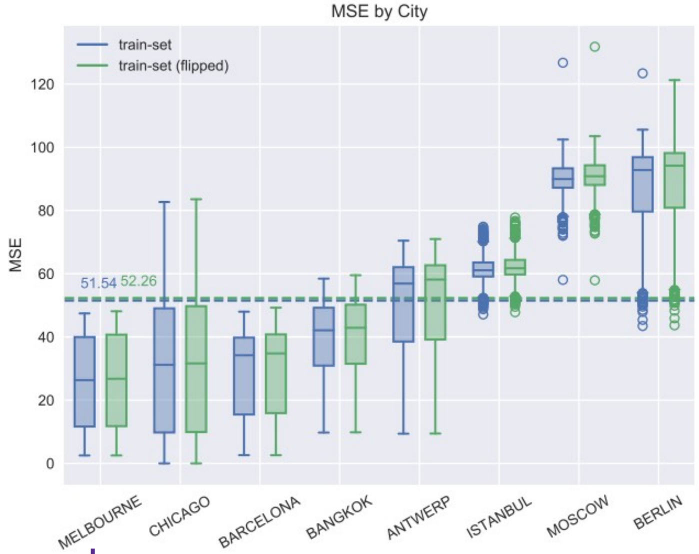


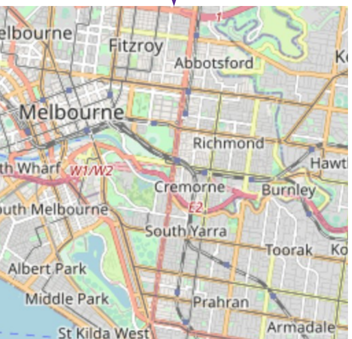


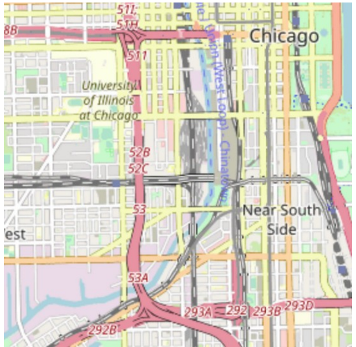


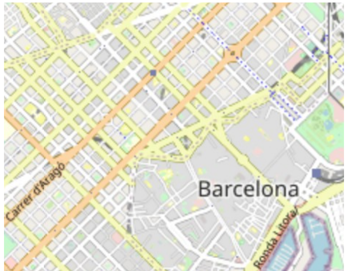
~~Quantitative~~ Results

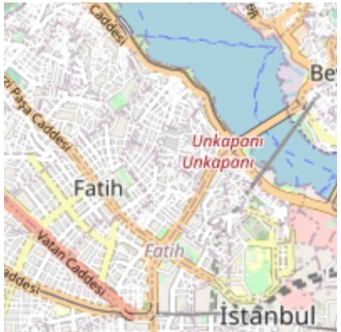


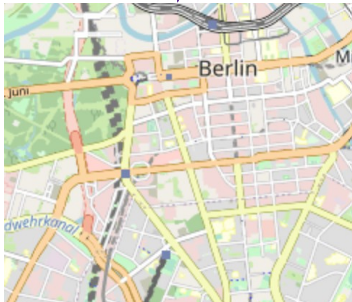












Ablations and comparison to Vanilla U-Net

Presented Model
(+ Subgraphing)

Presented Model
(NO Subgraphing)

	Hybrid UNet			Graph UNet			Vanilla UNet		
	MSE	MSE*	rel. MSE	MSE	MSE*	rel. MSE	MSE	MSE*	rel. MSE
ANTWERP	48.35	**49.034**	0.986	48.819	49.186	0.993	**48.193**	50.712	0.95
BANGKOK	39.466	40.338	0.978	39.729	**40.045**	0.992	**39.444**	40.908	0.964
BARCELONA	28.742	29.502	0.974	28.968	**29.284**	0.989	**28.609**	29.663	0.964
BERLIN	87.047	88.41	0.985	87.798	**88.388**	0.993	**86.95**	91.068	0.955
CHICAGO	**32.147**	32.593	0.986	32.451	**32.526**	0.998	32.228	32.939	0.978
ISTANBUL	**61.237**	**62.028**	0.987	61.98	62.262	0.995	61.588	64.3	0.958
MELBOURNE	**25.325**	25.74	0.984	25.626	**25.709**	0.997	25.393	26.091	0.973
MOSCOW	**89.628**	**90.587**	0.989	90.44	90.855	0.995	89.846	93.752	0.958
average	**51.493**	**52.279**	0.985	51.976	52.282	0.994	51.531	53.679	0.96

Ablations and comparison to Vanilla U-Net

Presented Model (+ Subgraphing)

Presented Model (NO Subgraphing)

	Hybrid UNet			Graph UNet			Vanilla UNet		
	MSE	MSE*	rel. MSE	MSE	MSE*	rel. MSE	MSE	MSE*	rel. MSE
ANTWERP	48.35	**49.034**	0.986	48.819	49.186	0.993	**48.193**	50.712	0.95
BANGKOK	39.466	40.338	0.978	39.729	**40.045**	0.992	**39.444**	40.908	0.964
BARCELONA	28.742	29.502	0.974	28.968	**29.284**	0.989	**28.609**	29.663	0.964
BERLIN	87.047	88.41	0.985	87.798	**88.388**	0.993	**86.95**	91.068	0.955
CHICAGO	**32.147**	32.593	0.986	32.451	**32.526**	0.998	32.228	32.939	0.978
ISTANBUL	**61.237**	**62.028**	0.987	61.98	62.262	0.995	61.588	64.3	0.958
MELBOURNE	**25.325**	25.74	0.984	25.626	**25.709**	0.997	25.393	26.091	0.973
MOSCOW	**89.628**	**90.587**	0.989	90.44	90.855	0.995	89.846	93.752	0.958
average	**51.493**	**52.279**	0.985	51.976	52.282	0.994	51.531	53.679	0.96

On four of the ‘known’ cities, U-Net outperforms our model,
the average difference is very small

Ablations and comparison to Vanilla U-Net

Presented Model (+ Subgraphing)

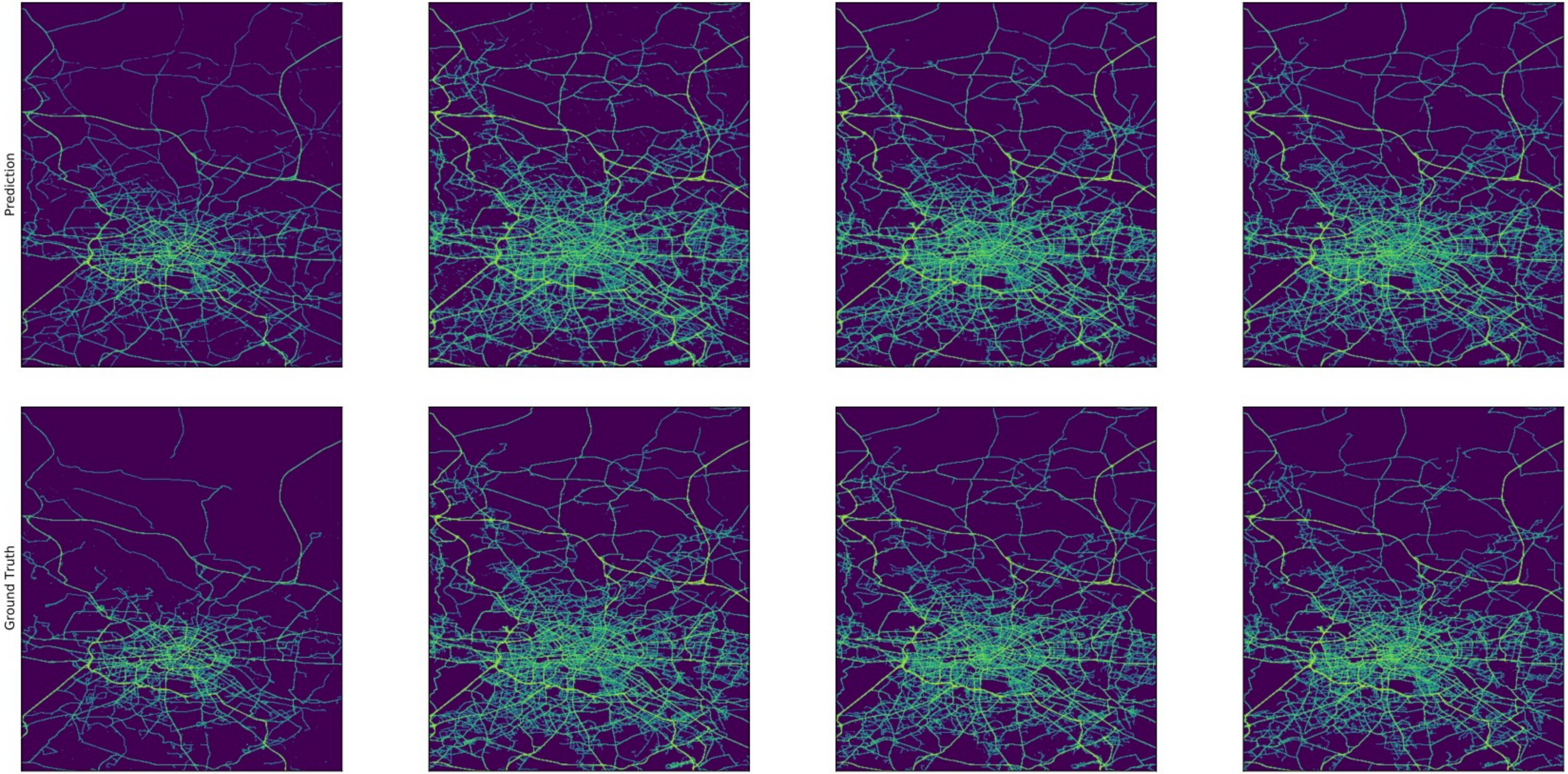
Presented Model (NO Subgraphing)

	Hybrid UNet			Graph UNet			Vanilla UNet		
	MSE	MSE*	rel. MSE	MSE	MSE*	rel. MSE	MSE	MSE*	rel. MSE
ANTWERP	48.35	**49.034**	0.986	48.819	49.186	0.993	**48.193**	50.712	0.95
BANGKOK	39.466	40.338	0.978	39.729	**40.045**	0.992	**39.444**	40.908	0.964
BARCELONA	28.742	29.502	0.974	28.968	**29.284**	0.989	**28.609**	29.663	0.964
BERLIN	87.047	88.41	0.985	87.798	**88.388**	0.993	**86.95**	91.068	0.955
CHICAGO	**32.147**	32.593	0.986	32.451	**32.526**	0.998	32.228	32.939	0.978
ISTANBUL	**61.237**	**62.028**	0.987	61.98	62.262	0.995	61.588	64.3	0.958
MELBOURNE	**25.325**	25.74	0.984	25.626	**25.709**	0.997	25.393	26.091	0.973
MOSCOW	**89.628**	**90.587**	0.989	90.44	90.855	0.995	89.846	93.752	0.958
average	**51.493**	**52.279**	0.985	51.976	52.282	0.994	51.531	53.679	0.96

Hybrid U-Net generalizes better to the unseen flipped cities

Berlin
Wed , 20.03.2019

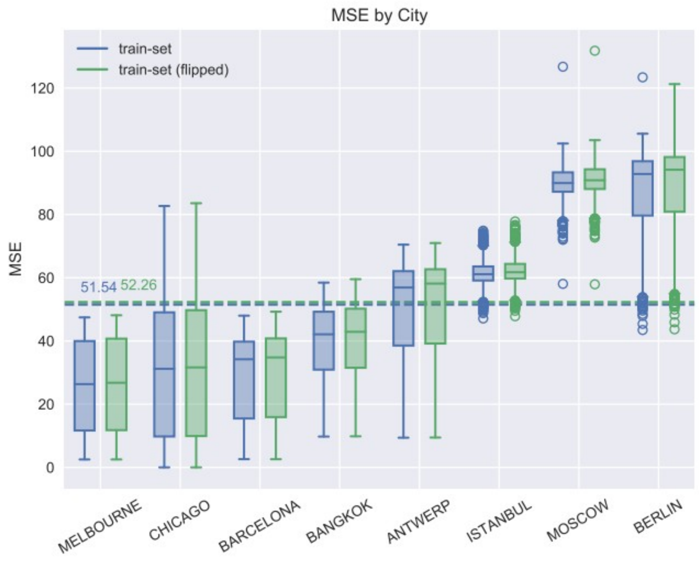
Qualitative Results



Time: 00:00 06:00 12:00 18:00

Thank You!

Any Questions?



Code on GitHub:
https://github.com/LucaHermes/graph-unet-traffic-prediction

Link To the Paper:
https://rebrand.ly/nobii5z