Network of Tensor Time Series

Presenter: Baoyu Jing
Contact: baoyuj2@Illinois.edu

Baoyu Jing
(UIUC)

Hanghang Tong
(UIUC)

Yada Zhu
(IBM)

Outline

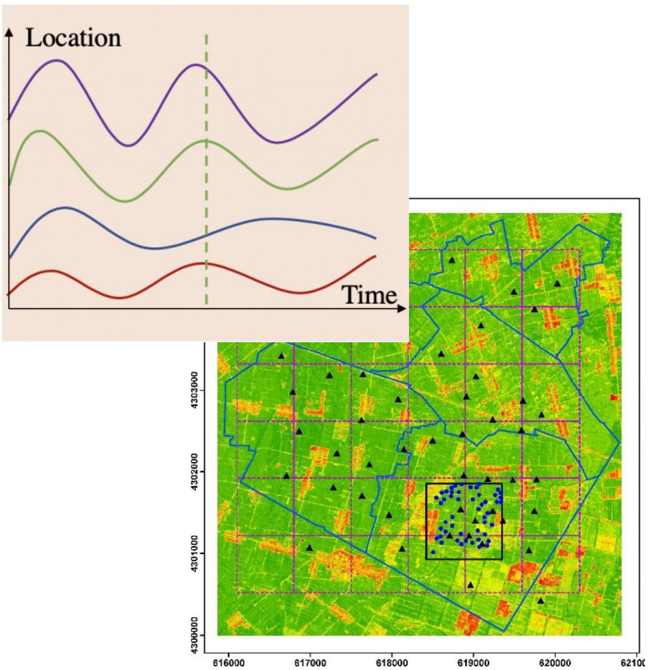
- Introduction
- Preliminaries
- Methodology
- Experiments
- Conclusion

Outline

- ➢ **Introduction**
- Preliminaries
- Methodology
- Experiments
- Conclusion

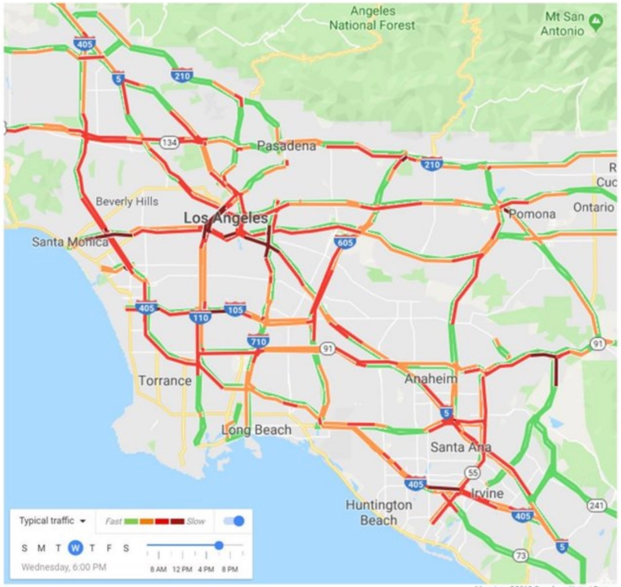
Ubiquity of Co-evolving Time Series

- Co-evolving time series is ubiquitous.
  - Each time series is related to each other.



Environmental Monitoring

Financial Analysis



Smart Transportation

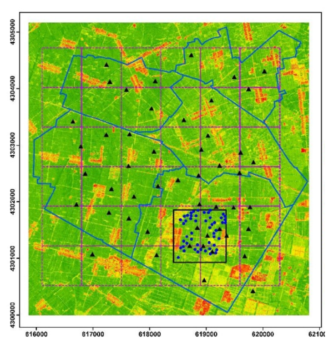
Properties of Co-evolving Time Series

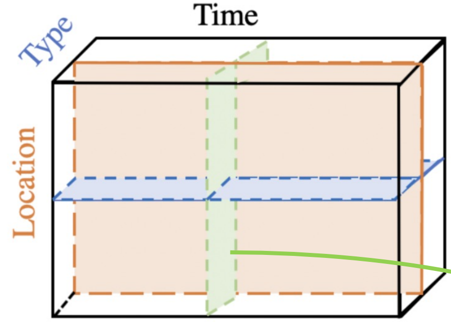
- Take environmental monitoring as an example. (Left Figure)
- It is a tensor. (Middle Figure)
- Each temporal snapshot is a tensor. (The green slice)
- Network constraint for each dimension.

1. We have some monitoring sites/locations.
2. Each site has multiple types of sensors.

Coefficients

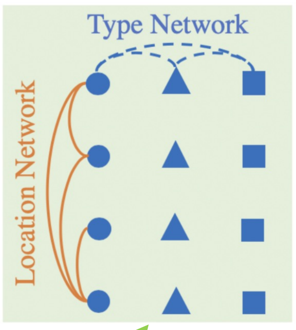
Temperature	1.0	0.7	0.9
Humidity	0.7	1.0	0.5
Pressure	0.9	0.5	1.0





Distance

Site 1	0.0	15	20
Site 2	15	0.0	99
Site 3	20	99	0.0

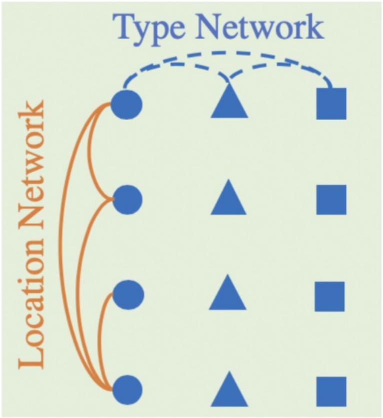


Challenge #1: Model Explicit Relations

- Network constraints.
  - Distance between the sensors.
  - Correlation between temperature, humidity, pressure etc.

×Existing methods are designed for **flat** graphs (e.g., GCN).
  - Either location network or type network, but not all.

✓We introduce
  - Spectral Convolution for Tensor Graphs
  - Tensor Graph Convolutional Network (TGCN)

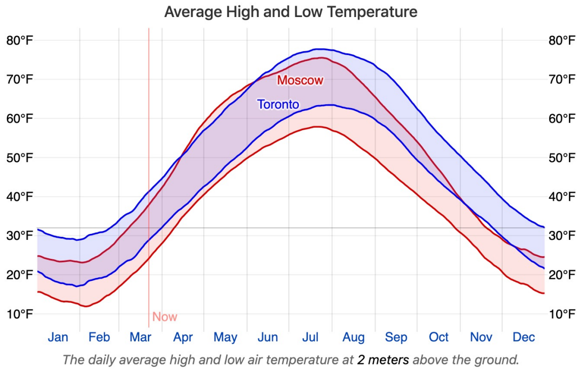


Kipf, Thomas N., and Max Welling. "Semi-supervised classification with graph convolutional networks." *arXiv preprint arXiv:1609.02907* (2016).

Challenge #2: Model Implicit Relations

- Different time series might have similar patterns.
  - E.g., air temperatures of Toronto and Mosco.
  - Explicit distance network constraint cannot capture this relation.

×Existing methods use
  - the **same** model for all time series
  - an **individual** model for each time series



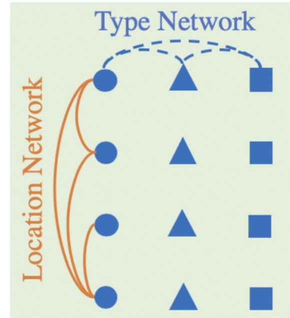
✓We introduce:
  - Tensor Recurrent Neural Network (TRNN)
  - Implement RNN with LSTM: TLSTM

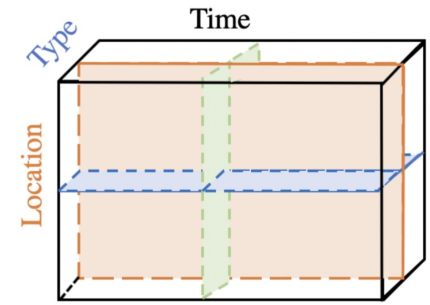
https://weatherspark.com/compare/y/100524~19863/Comparison-of-the-Average-Weather-in-Moscow-and-Toronto

Outline

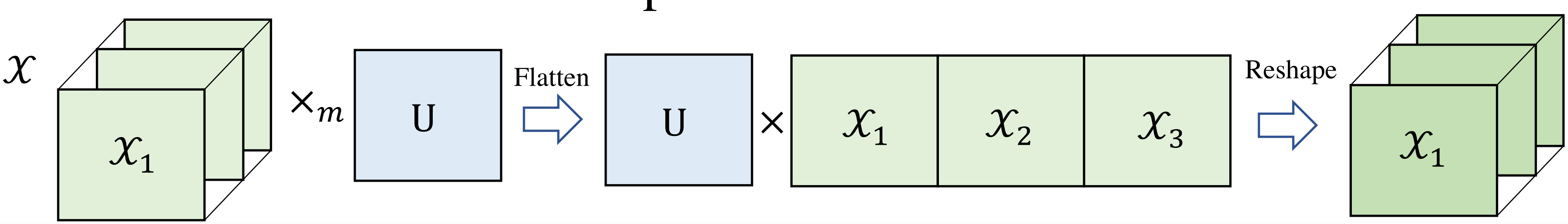
- Introduction
- ➢ **Preliminaries**
- Methodology
- Experiments
- Conclusion

Preliminaries





- Tensor Graph $\mathcal{G}$:
  - A tensor graph is comprised of (1) a M-dimensional tensor $\mathcal{X} \in \mathbb{R}^{N_1 \times \cdots \times N_M}$ and (2) adjacency matrices $\mathrm{A_m} \in \mathbb{R}^{N_m \times N_m}$.
- Network of Tensor Time Series:
  - It is comprised of a (1) tensor time series $\mathcal{S} \in \mathbb{R}^{N_1 \times \cdots \times N_M \times T}$, and (2) adjacency matrices $\mathrm{A_m} \in \mathbb{R}^{N_m \times N_m}$.
- Mode-m product between tensor $\mathcal{X}$ and matrix U: $\mathcal{X} \times_{\mathrm{m}} \mathrm{U}$
  - Generalization of the product between matrices.

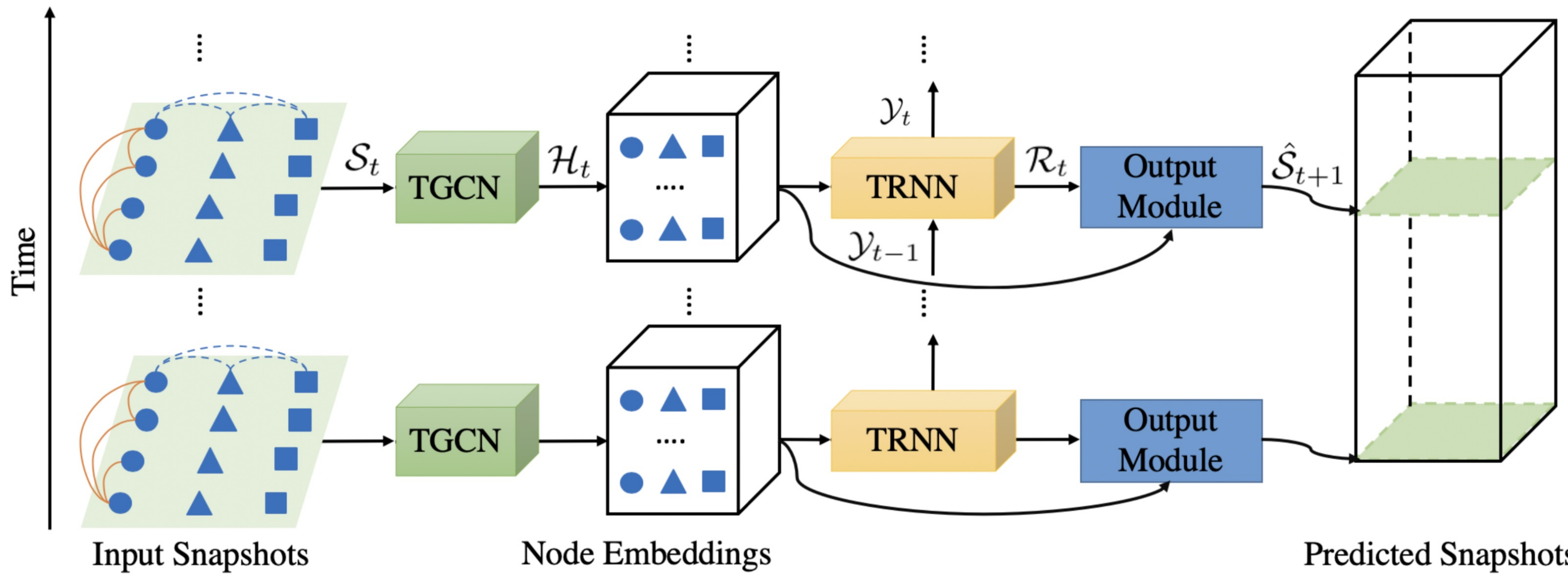


Outline

- Introduction
- Preliminaries
- ➢ **Methodology**
- Experiments
- Conclusion

Overview

- Problem: given the ω historical snapshots, predict the next τ snapshots.
- Challenge 1: Explicit Relations - TGCN
- Challenge 2: Implicit Relations - TRNN



Tensor Graph Convolutional Network

Φ: Eigenvector matrix of the graph Laplacian $\mathrm{L} = \mathrm{I} - \mathrm{D}^{1/2}\mathrm{A}\mathrm{D}^{1/2} = \Phi^T \Lambda \Phi$

$$\tilde{\mathbf{x}} = \Phi^T \mathbf{x}$$

$$\tilde{x} = \phi^T x \xrightarrow{1} \tilde{\mathcal{X}} = \phi_1^T \, \mathcal{X} \, \phi_2^T$$

1. Graph Fourier Transformation -> Tensor Graph Fourier Transformation

$\times_m$: mode-m product

$$\tilde{\mathcal{X}} = \mathcal{X} \prod_{m=1}^{M} \times_m \Phi_m$$

$$\Downarrow 2$$

$$\mathcal{G} \star \mathcal{X} = \phi_1^T \, \cancel{\tilde{g}_1} \, \phi_1 \, \mathcal{X} \, \phi_2^T \, \cancel{\tilde{g}_2} \, \phi_2$$

2. Spectral Convolution

$\bigstar$: convolution operation $\quad \mathcal{G}$: parameter tensor

$$\mathcal{G} \star \mathcal{X} = \mathcal{X} \prod_{m=1}^{M} \times_m \Phi_m^T \mathrm{diag}(\tilde{\mathbf{g}}_m) \Phi_m$$

$$\Downarrow 3$$

$$\mathcal{G}_\theta \star \mathcal{X} = \sum_p \theta_p T_p(\tilde{L}_1) \, \mathcal{X} \, \sum_p \theta_p T_p(\tilde{L}_2)$$

3. Approximation via Chebyshev Polynomials

T_p: Chebyshev Polynomial with degree p

$$\mathcal{G}_\theta \star \mathcal{X} = \mathcal{X} \prod_{m=1}^{M} \times_m \sum_{p_m=0}^{P} \theta_{m,p_m} T_{p_m}(\tilde{\mathbf{L}}_m)$$

$\tilde{L}_m$ normalized Laplacian matrix

4. Simplification

$$\mathcal{G}_\theta \star \mathcal{X} = \sum_{\exists p_m=1} \theta_{p_1,\cdots,p_M} \mathcal{X} \prod_{p_m=1} \times_m \tilde{\mathrm{A}}_m + \theta_{0,\cdots,0} \mathcal{X}$$

$\tilde{A}_m$: normalized adjacency matrix $\quad \theta$: parameters $\quad \sigma$: activation function

5. Updating function for each layer

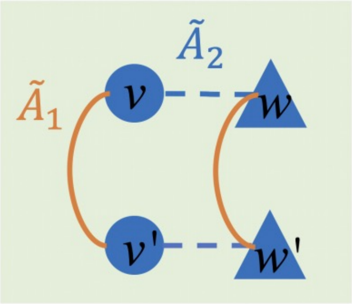
$$\mathrm{TGCL}(\mathcal{X}, \{\mathrm{A}_m\}_{m=1}^{M}) = \sigma\Big(\sum_{\exists p_m=1} \mathcal{X} \prod_{p_m=1} \times_m \tilde{\mathrm{A}}_m \times_{M+1} \Theta_{p_1,\cdots,p_M} + \mathcal{X} \times_{M+1} \Theta_0\Big)$$

TGCN: Detailed Analysis

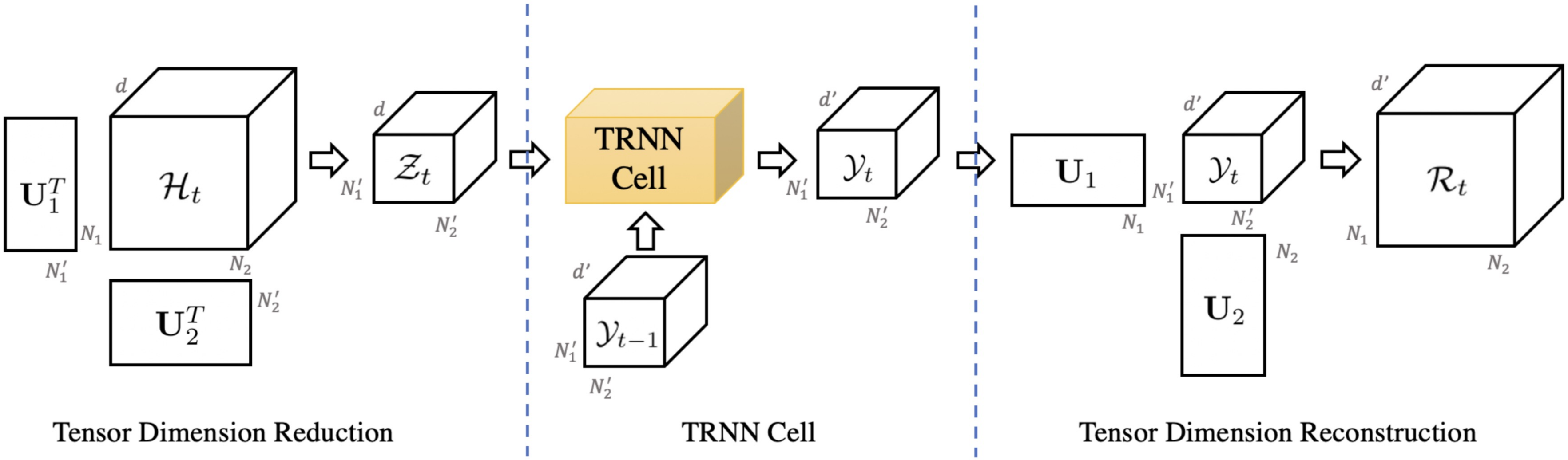
- Let $m = 2$:

$$\mathcal{G}_\theta \star \mathcal{X} = \underbrace{\boxed{\theta_{1,1}\mathcal{X} \times_1 \tilde{\mathrm{A}}_1 \times_2 \tilde{\mathrm{A}}_2}}_{\text{Synergy of } A_1 \text{ and } A_2} + \underbrace{\boxed{\theta_{1,0}\mathcal{X} \times_1 \tilde{\mathrm{A}}_1}}_{\text{Only } A_1} + \underbrace{\boxed{\theta_{0,1}\mathcal{X} \times_2 \tilde{\mathrm{A}}_2}}_{\text{Only } A_2} + \underbrace{\boxed{\theta_{0,0}\mathcal{X}}}_{\text{Self-convolution or Residue connection}}$$

  - Capture the synergy.
  - Capture each network separately.
  - Have a self-convolution/residue connection.
  - Illustration of synergy:
    - node v could gather information from w'.



Tensor Recurrent Neural Network



Tucker decomposition
U_m is **orthonormal**

$$\mathcal{Z}_t = \mathcal{H}_t \prod_{m=1}^{M} \times_m \mathrm{U}_m^T$$

Replace **linear** operations in RNN by **multi-linear** operations

$$\mathrm{Linear}(x) = xw + b$$

$$\mathrm{TLL}(\mathcal{X}) = \mathcal{X} \prod_{m=1}^{M+1} \times_m \mathbf{W}_m + \mathbf{b}$$

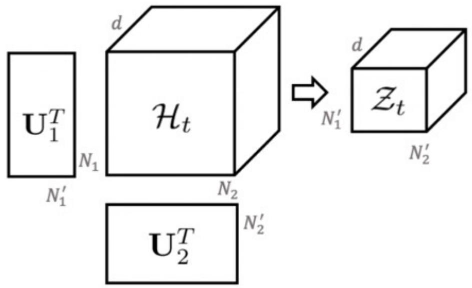
Reuse U_m since it is **orthonormal**.

$$\mathcal{R}_t = \mathcal{Y}_t \prod_{m=1}^{M} \times_m \mathrm{U}_m$$

We use LSTM to implement RNN.

The Implicit Relationship

- The Tucker decomposition is a high-order PCA or SVD.
  - $\mathrm{U_m}$ extracts the eigenvectors of the m-th dimension.
  - Each element in $\mathcal{Z}$ indicates the interaction of the eigenvectors.
    - The degree of implicit relation.
- Let ρ be the interaction degree: $N'_m = \rho N_m \; (\forall m \in [1, \cdots, M])$
    - $\rho \in (0, 1)$: ideal range
    - $\rho > 1$: $\mathrm{U_m}$ is over-complete and have redundant information
    - $\rho = 0$: no interaction



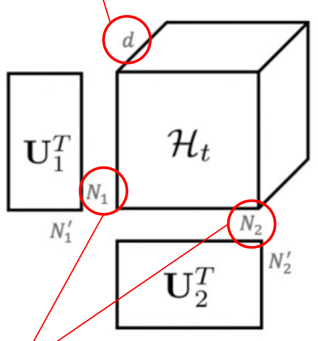
Parameter Reduction

- Parameter Comparison: We use LSTM to implement TRNN -> TLSTM
  - TLSTM cell < multiple LSTM cells
  - Tucker decomposition introduces new parameters U_m
- TLSTM uses less parameters than multiple LSTM if:

LEMMA 3.5 (UPPER-BOUND FOR ρ). *Let N_m and N'_m be the dimensions of* U_m *in Equation* (24), *and let $d \in \mathbb{R}$ and $d' \in \mathbb{R}$ be the hidden dimensions of the inputs and outputs of TLSTM. TLSTM uses less parameters than multiple separate LSTMs, as long as the following condition holds:*

$$\rho \leq \sqrt{\frac{(\prod_{m=1}^{M} N_m - 1)d'(d + d' + 1)}{2\sum_{m=1}^{M} N_m^2} + \frac{1}{256}} - \sqrt{\frac{1}{256}} \tag{33}$$

d: the hidden dimension of $\mathcal{H}$



ρ be the interaction degree: $N'_m = \rho N_m \ (\forall m \in [1, \cdots, M])$

d': the hidden dimension of LSTM/TLTSM

N_m: the dimension of the m-th mode of $\mathcal{H}_t$

Outline

- Introduction
- Preliminaries
- Methodology
- ➢ **Experiments**
- Conclusion

Experimental Setup

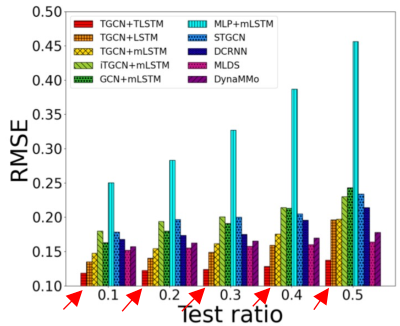
- Datasets:

Dataset	Shape	# Nodes	Modes with A
Motes	$54 \times 4 \times 2880$	216	1, 2
Soil	$42 \times 5 \times 2 \times 365$	420	1, 2, 3
Revenue	$410 \times 3 \times 62$	1,230	1, 2
Traffic	$1000 \times 2 \times 1440$	2,000	1
20CR	$30 \times 30 \times 20 \times 6 \times 180$	108,000	1, 2, 3, 4

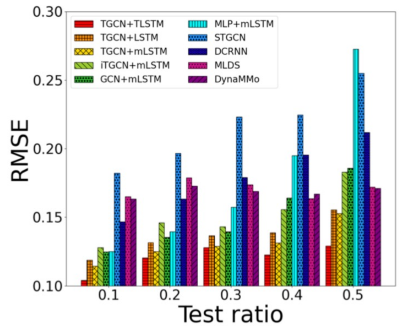
- Tasks:
  - Missing Value Recovery
  - Future Value Prediction
- Metric: RMSE (the lower the better)
- Preprocessing:
  - Normalize each time series by z-scores.
  - Missing value recovery: use $[0.1, 0.2, 0.3, 0.4, 0.5]$ for test
  - Future value prediction: use $[0.02, 0.04, 0.06, 0.08, 0.1]$ for test
- Questions:
  - How accurate is NET^3 for missing value recovery and future value prediction?
  - How will synergy improve the performance?
  - How does the interaction degree ρ impact the performance?
  - How efficient and scalable is NET^3?

Missing value recovery & Future value prediction

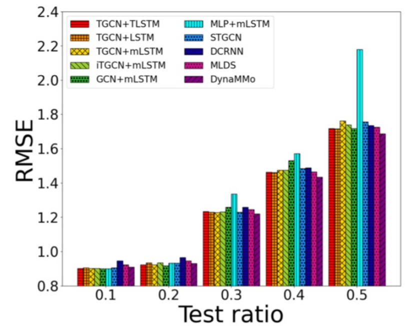
- The red arrows point (or the left-most bars) to NET^3.
- NET^3 performs the best: lowest RMSE.



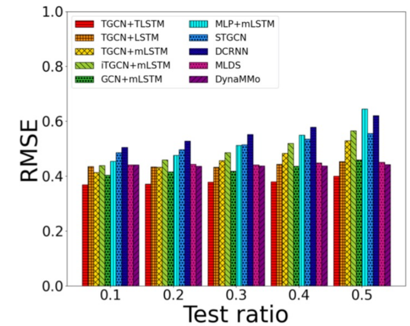
(a) Motes-Missing



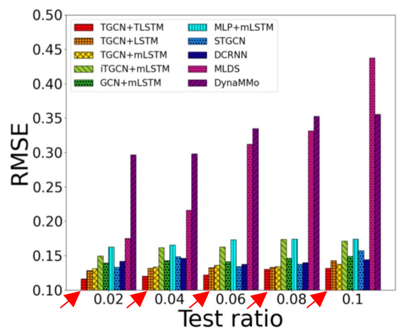
(b) Soil-Missing



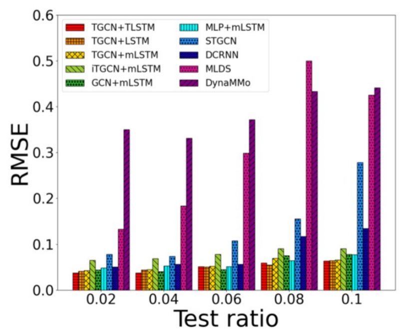
(c) Revenue-Missing



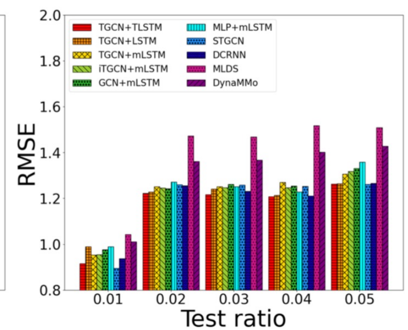
(d) Traffic-Missing



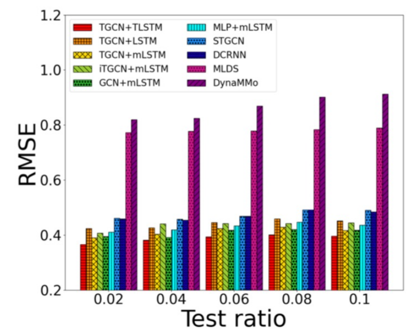
(e) Motes-Future



(f) Soil-Future



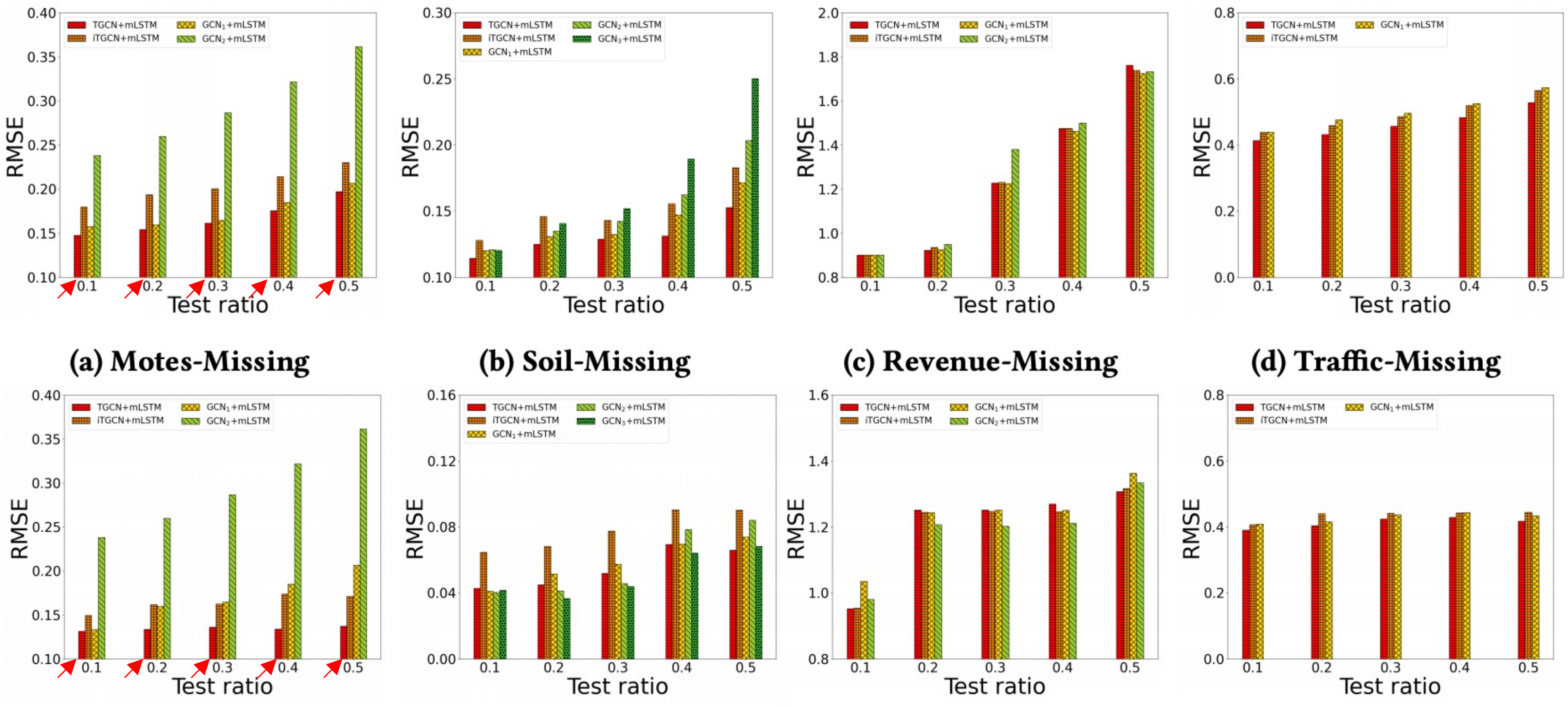
(g) Revenue-Future



(h) Traffic-Future

Synergy Analysis

- Comparison methods:
  - (1) GCN with one network, (2) iTGCN: ~multiple GCNs, (3) TGCN: Full model
- TGCN (red arrows) performs the best.

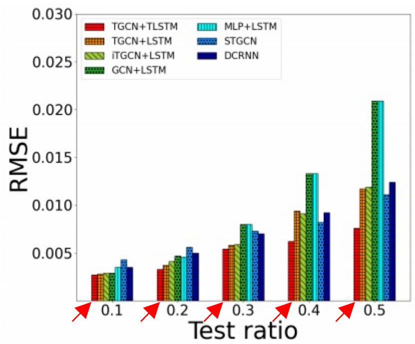


(a) Motes-Missing (b) Soil-Missing (c) Revenue-Missing (d) Traffic-Missing

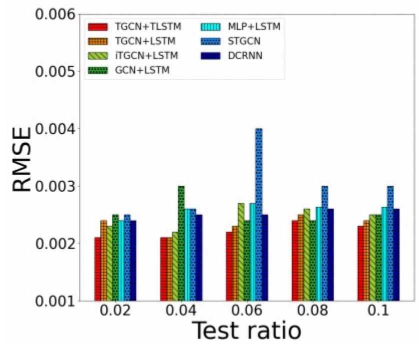
(e) Motes-Future (f) Soil-Future (g) Revenue-Future (h) Traffic-Future

Experiments: 20CR dataset

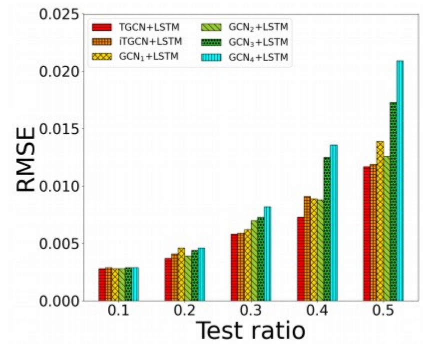
- The red arrows point (or the left-most bars) to the full model NET^3.
- NET^3 performs the best: lowest RMSE.



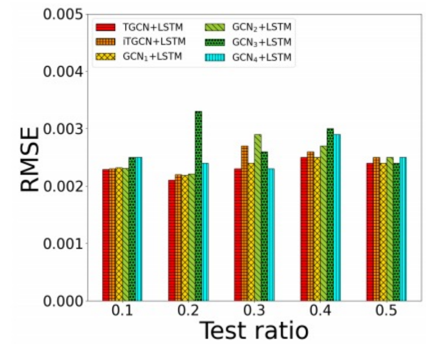
(a) Missing Value Recovery



(b) Future Value Prediction



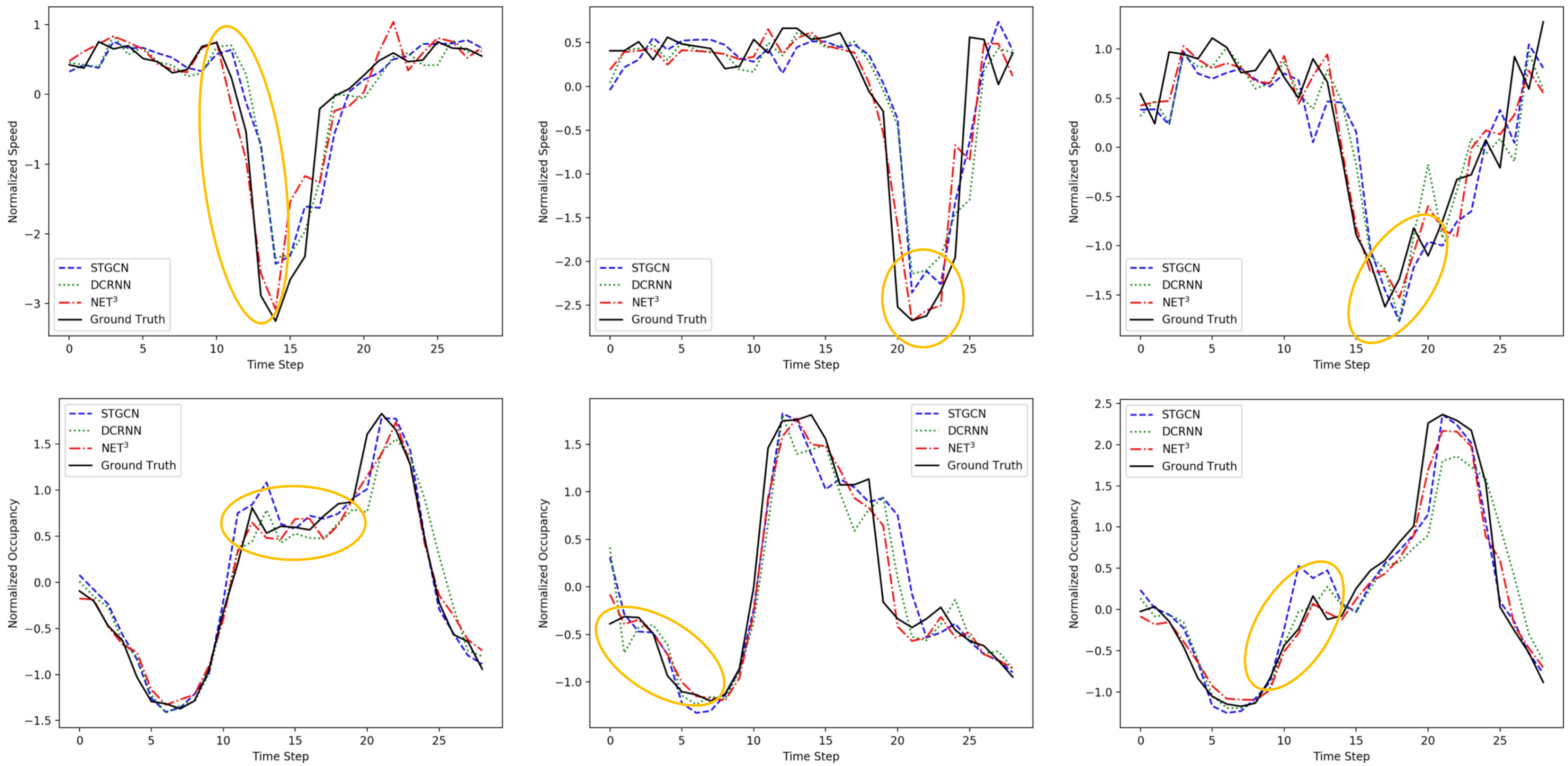
(c) Synergy-Missing



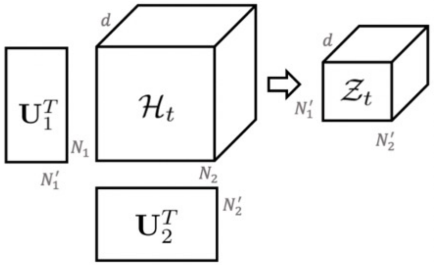
(d) Synergy-Future

Visualization on the Traffic Dataset

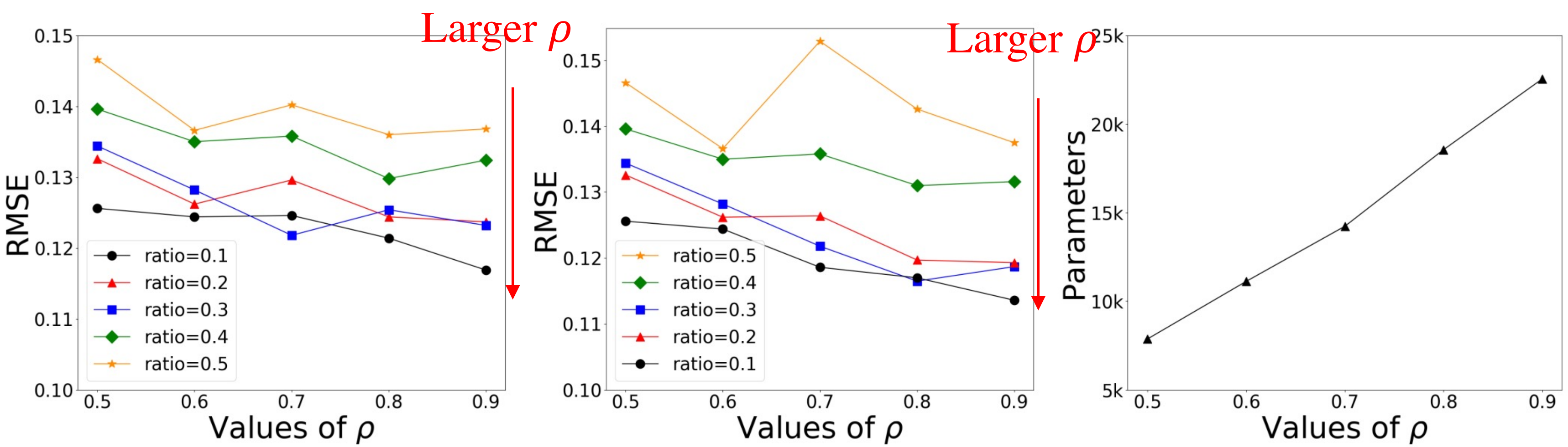
- NET^3 (red), Ground truth (black), Baselines (green, blue)
- NET^3 performs the best: closest to the ground truth (see yellow circles).



Sensitivity



- As ρ increases, the model performs better in general.
  - U_m contains more eigenvectors: more information.
- # parameters of TLSTM grows linearly with ρ.

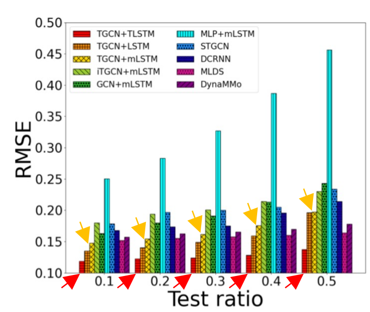


(a) Missing value recovery **(b) Future value prediction** **(c) Number of parameters**

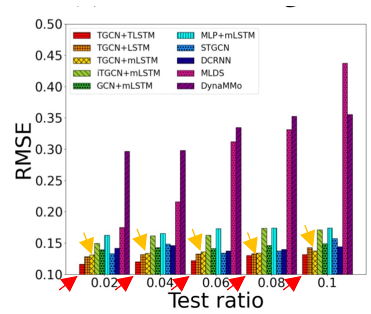
Memory Efficiency

- TLSTM can significantly reduce # parameters.
- TLSTM achieves lower RMSE than mLSTM.

		Motes	*Soil*	*Revenue*	*Traffic*	*20CR*
Upper bound	ρ_{max}	2.17	2.43	0.64	0.31	57.25
Values in experiments	ρ_{exp}	0.80	0.80	0.20	0.10	0.90
# parameters	TLSTM	18,552	10,996	87,967	180,554	16,696
	mLSTM	117,504	57,120	669,120	1,088,000	58,752,000
Reduction ratio	Reduce	84.21%	80.75%	86.85%	83.40%	99.97%



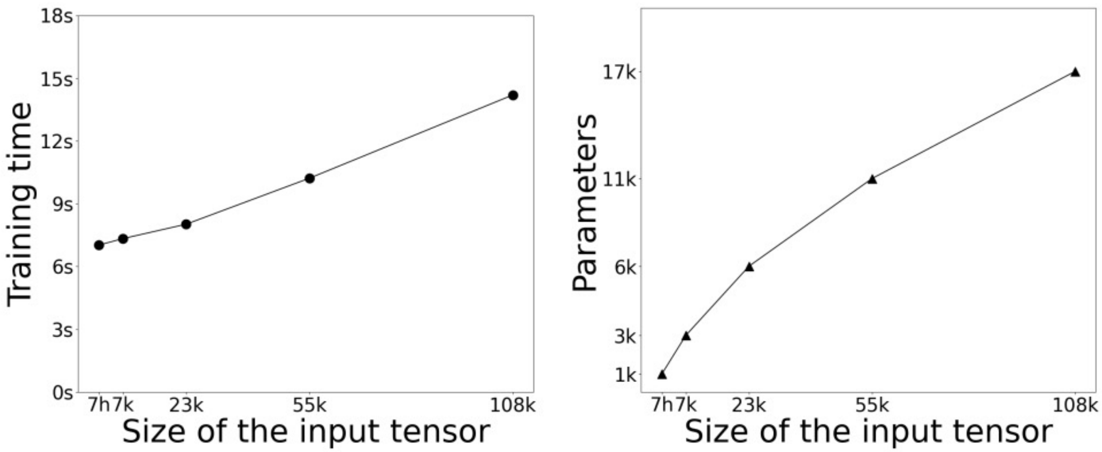
(a) Motes-Missing



(e) Motes-Future

Scalability

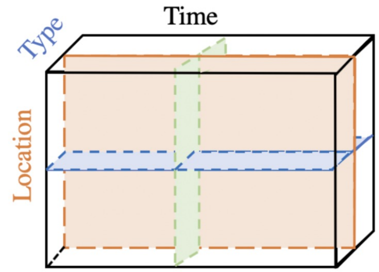
- The training time V.S. size of input tensor: almost linear
- # parameters V.S. size of input tensor: almost linear.

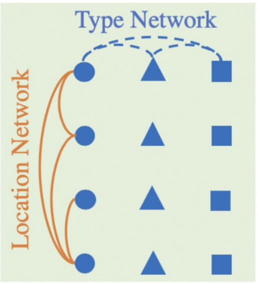


Outline

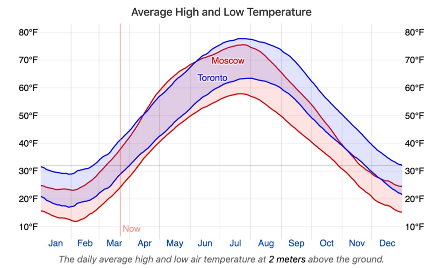
- Introduction
- Preliminaries
- Methodology
- Experiments
- ➢ **Conclusion**

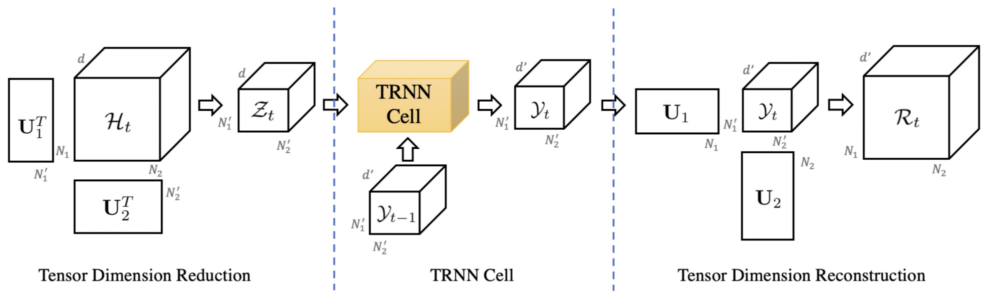
Conclusion



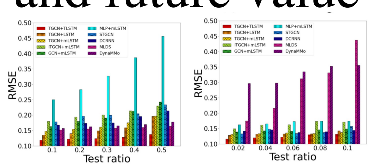


- Model Co-evolving time series
  - Challenge 1: Explicit Relationship
  - Solution 1: Tensor Graph Convolutional Network (TGCN)
  - Challenge 2: Implicit Relationship
  - Solution 2: Tensor Recurrent Neural Network (TRNN)

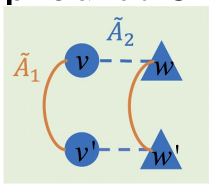




- Results:
  - NET[3] performs the best for missing value recovery and future value prediction.
  - TGCN captures the synergy among networks.
  - TRNN reduces # parameters and performs better.



(a) Motes-Missing (e) Motes-Future



Thank you!