# Data Seminar

07/01/2021

# Outline

1. Learning to Simulate Complex Physics with Graph Networks. Sanchez-Gonzalez A, Godwin J, Pfaff T, et al. ICML 2020.

2. Connecting the dots: Multivariate time series forecasting with graph neural networks. Wu Z, Pan S, Long G, et al. SIGKDD 2020.

# Outline

1. **Learning to Simulate Complex Physics with Graph Networks. Sanchez-Gonzalez A, Godwin J, Pfaff T, et al. ICML 2020.**

2. Connecting the dots: Multivariate time series forecasting with graph neural networks. Wu Z, Pan S, Long G, et al. SIGKDD 2020.

# Motivation & Idea

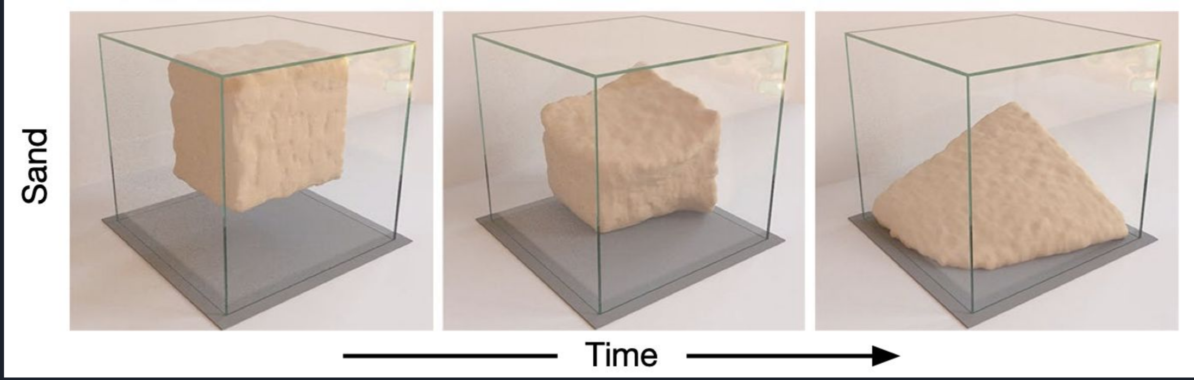
Motivation:

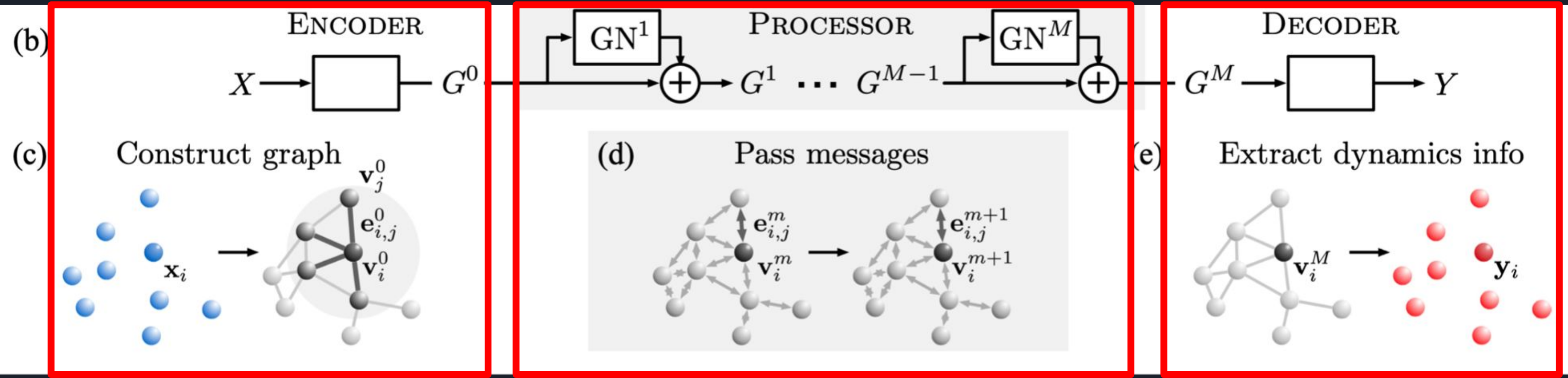
- Traditional engineering simulators: Expensive to create and use, trade off generality for accuracy.
- Machine learning simulators: Difficulty in overcoming large state spaces & complex dynamics.

Ideas:

- Interacting nodes to represent rich physical states.
- Message-passing among nodes to represent complex dynamics.

# Framework




Encoder: Embed the particle-based state representation X as a latent graph G_{0}.

Processor: Update latent graphs as G_{m+1} = GN^{m+1}(G_{m}). GN: Graph Network.

Decoder: Extracts dynamics information from the nodes in G_{m}.

# Mathematical setup

Time: t_{0}, t_{1}, …, t_{K}

K timesteps, N particles.

Dynamics: X^{t_{0:K}} = X^{t_{0}}, X^{t_{1}},..., X^{t_{K}}

Particles: X^{t_{0}} = X_{0}^{t_{0}}, X_{1}^{t_{0}},..., X_{N}^{t_{0}}

$$\dot{\mathbf{p}}^{t_{k+1}} = \dot{\mathbf{p}}^{t_k} + \Delta t \cdot \ddot{\mathbf{p}}^{t_k}$$
$$\mathbf{p}^{t_{k+1}} = \mathbf{p}^{t_k} + \Delta t \cdot \dot{\mathbf{p}}^{t_{k+1}}$$

No dot: Position at time t_{k}
One dot: Velocity at time t_{k}
Two dots: Acceleration at time t_{k}

State of particle i at time t_{k}:

$$\mathbf{x}_i^{t_k} = [\mathbf{p}_i^{t_k}, \dot{\mathbf{p}}_i^{t_{k-C+1}}, \ldots, \dot{\mathbf{p}}_i^{t_k}, \mathbf{f}_i]$$

Position Past C-step Velocity Particle Feature: fluids, solids...

# Processor

Learned simulator: s (parameters: s_{theta})

The simulator gives an simulation from X^{t_{k}} to X^{t_{k+1}}:

$$\tilde{X}^{t_{k+1}} = s(\tilde{X}^{t_k})$$

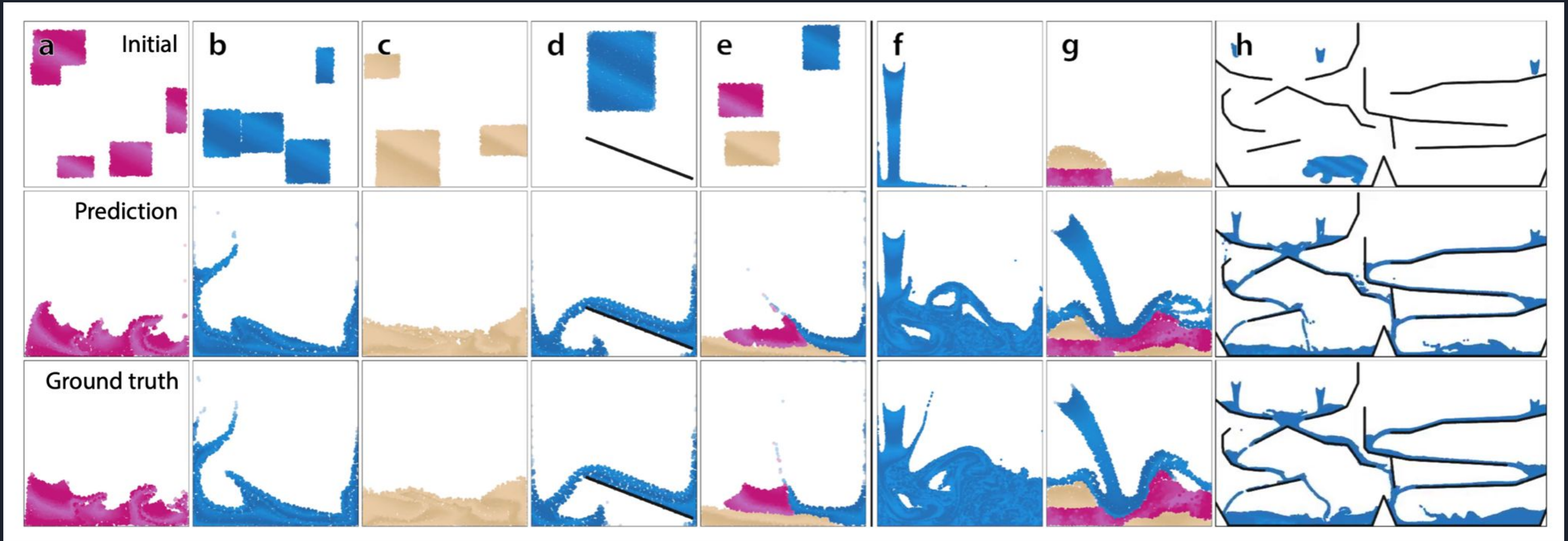
For each step, they train a set of parameters theta

$$\theta^*_{\text{1-step}} \leftarrow \arg_\theta \min \mathbb{E}_{\mathbb{P}(\mathbf{X}^{t_{k:k+1}})} L_{\text{1-step}}(X^{t_{k+1}}, s_\theta(X^{t_k}))\,.$$

Loss measures the RMSE between **acceleration** between X_{t^{k}} and s_{theta}(X_{t^{k}}).

# Experiments : Multiple Materials/Objects.

Experiments showed that this general simulator could simulate multiple materials and objects.




Video: https://sites.google.com/view/learning-to-simulate

# Summary

Use interacting nodes to represent rich physical states.

- Divide the physical system X into N particles.
- Each particle corresponds to one node in the graph.

Message-passing among nodes to represent complex dynamics.

- Edges corresponds to pairwise properties of the corresponding particles.
- One-step loss function to train the learned simulator.

# Outline

1. Learning to Simulate Complex Physics with Graph Networks. Sanchez-Gonzalez A, Godwin J, Pfaff T, et al. ICML 2020.

2. **Connecting the dots: Multivariate time series forecasting with graph neural networks. Wu Z, Pan S, Long G, et al. SIGKDD 2020.**

# Motivation & Idea

Motivation:

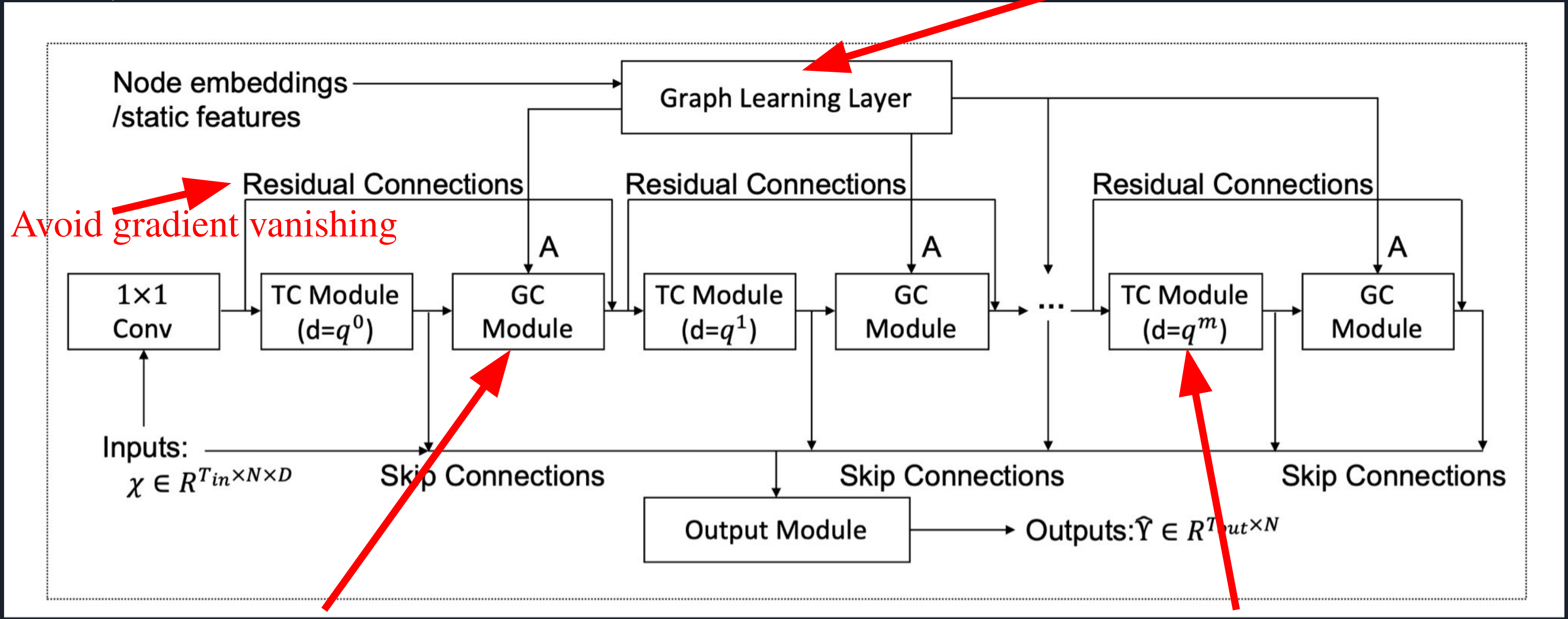
- Existing methods fail to fully exploit latent spatial dependencies between variables.
- GNN have shown high capability in handling relational dependencies.

Ideas:

- Graph learning layer to extract a sparse graph adjacency matrix adaptively based on data.
- Graph convolution model to address the spatial dependencies among variables.
- Temporal convolution module to capture temporal patterns.

# Framework

Learn the graph adjacency matrix: Find hidden association




Graph convolution to learn spatial dependencies

Temporal convolution to capture temporal patterns

# Mathematical setup

Observed multivariate variable X from t_{1} to t_{P}

$$\mathbf{X} = \{\mathbf{z}_{t_1}, \mathbf{z}_{t_2}, \cdots, \mathbf{z}_{t_P}\}$$

The goal is to predict future values from t_{P+1} to t_{P+Q}

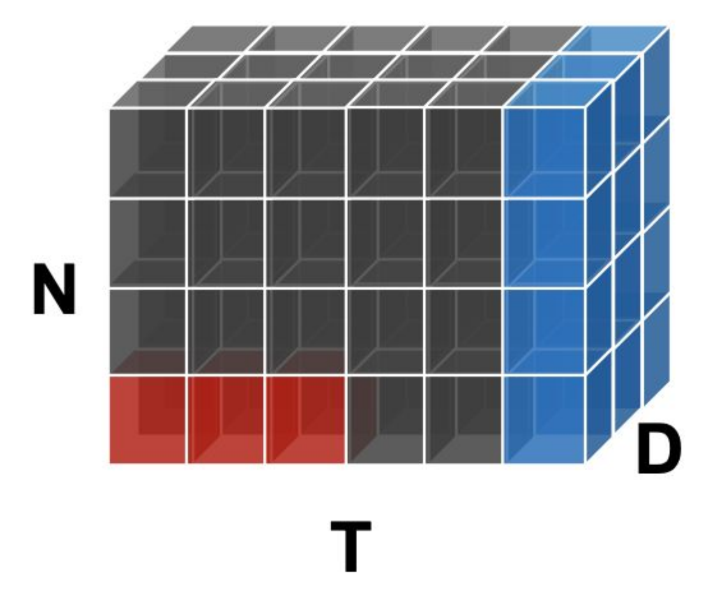
$$\mathbf{Y} = \{\mathbf{z}_{t_{P+1}}, \mathbf{z}_{t_{P+2}}, \cdots, \mathbf{z}_{t_{P+Q}}\}$$

Here, the inputs are in the from

$$\mathcal{X} = \{\mathbf{S}_{t_1}, \mathbf{S}_{t_2}, \cdots, \mathbf{S}_{t_P}\}$$

Where for each t_{i}, we also have D-1 auxiliary features

$$\mathbf{S}_{t_i} \in \mathbf{R}^{N \times D}$$




N: Dimension of the variable
T: Observed time length.
D: Dimension of features

# Graph learning layer

For graph learning layer, we want to learn an adjacency matrix A adaptively to capture the hidden relationships.

- Baseline similarity measure between all node pairs: O(n^2) complexity

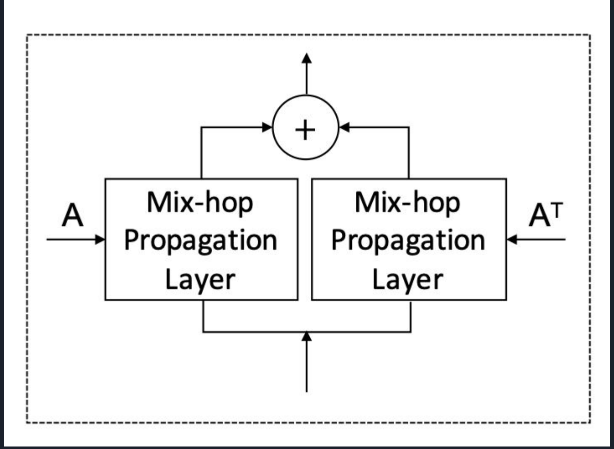
- Only calculate a subset of nodes in each step to reduce the complexity

Put the top-k similarity node pairs as connected in A, others are not connected.

# GC module and TC module

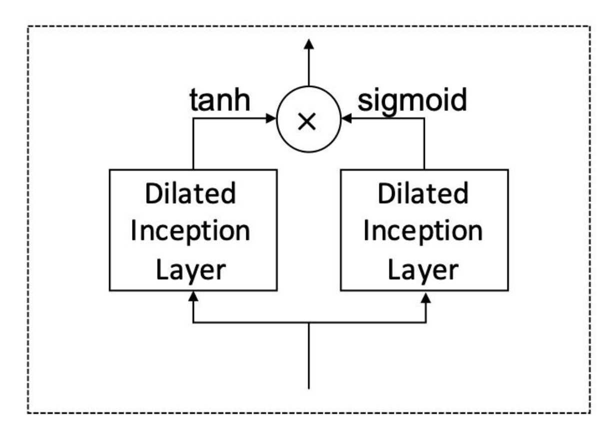
Graph convolution layer are designed to learn spatial dependencies.

- A & A^{T}: A can be asymmetric.
- Mix-hop propagation layer: Handle information flow over spatially dependent nodes.




Temporal convolution module to capture temporal patterns.

- Two gates: tanh and sigmoid as gate to control the amount of information.
- Dilated inception to handle the signals with inherent periods.



# Dilated inception

Filters with multiple sizes:

- Convolutional networks for 2-D: 1x1, 3x3, 5x5
- For 1-D: 1x2, 1x3, 1x6, 1x7… as filters for different periods.

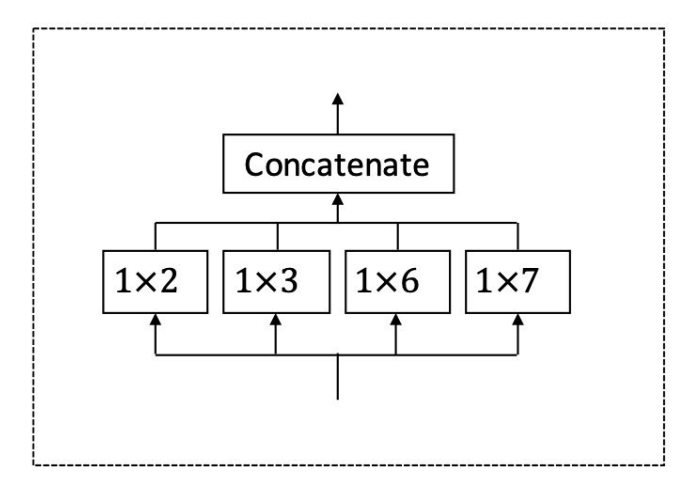
Dilated convolution:

- Without dilation, receptive size for time sequences:

  Length = m(c-1)+1

- With dilation at rate q, receptive size increase exponentially:

  Length = 1+(c-1)(q^m-1)/(q-1)



# Experiments : Single Step Forecasting

Horizon: Target future step (3/6/12/24 minutes later)

Single step: Forecast one output value for future time-step

| Dataset | | Solar-Energy | | | | Traffic | | | | Electricity | | | | Exchange-Rate | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Horizon | | | | Horizon | | | | Horizon | | | | Horizon | | | |
| Methods | Metrics | 3 | 6 | 12 | 24 | 3 | 6 | 12 | 24 | 3 | 6 | 12 | 24 | 3 | 6 | 12 | 24 |
| AR | RSE | 0.2435 | 0.3790 | 0.5911 | 0.8699 | 0.5991 | 0.6218 | 0.6252 | 0.63 | 0.0995 | 0.1035 | 0.1050 | 0.1054 | 0.0228 | 0.0279 | 0.0353 | 0.0445 |
| | CORR | 0.9710 | 0.9263 | 0.8107 | 0.5314 | 0.7752 | 0.7568 | 0.7544 | 0.7519 | 0.8845 | 0.8632 | 0.8591 | 0.8595 | 0.9734 | 0.9656 | 0.9526 | 0.9357 |
| VARMLP | RSE | 0.1922 | 0.2679 | 0.4244 | 0.6841 | 0.5582 | 0.6579 | 0.6023 | 0.6146 | 0.1393 | 0.1620 | 0.1557 | 0.1274 | 0.0265 | 0.0394 | 0.0407 | 0.0578 |
| | CORR | 0.9829 | 0.9655 | 0.9058 | 0.7149 | 0.8245 | 0.7695 | 0.7929 | 0.7891 | 0.8708 | 0.8389 | 0.8192 | 0.8679 | 0.8609 | 0.8725 | 0.8280 | 0.7675 |
| GP | RSE | 0.2259 | 0.3286 | 0.5200 | 0.7973 | 0.6082 | 0.6772 | 0.6406 | 0.5995 | 0.1500 | 0.1907 | 0.1621 | 0.1273 | 0.0239 | 0.0272 | 0.0394 | 0.0580 |
| | CORR | 0.9751 | 0.9448 | 0.8518 | 0.5971 | 0.7831 | 0.7406 | 0.7671 | 0.7909 | 0.8670 | 0.8334 | 0.8394 | 0.8818 | 0.8713 | 0.8193 | 0.8484 | 0.8278 |
| RNN-GRU | RSE | 0.1932 | 0.2628 | 0.4163 | 0.4852 | 0.5358 | 0.5522 | 0.5562 | 0.5633 | 0.1102 | 0.1144 | 0.1183 | 0.1295 | 0.0192 | 0.0264 | 0.0408 | 0.0626 |
| | CORR | 0.9823 | 0.9675 | 0.9150 | 0.8823 | 0.8511 | 0.8405 | 0.8345 | 0.8300 | 0.8597 | 0.8623 | 0.8472 | 0.8651 | 0.9786 | **0.9712** | 0.9531 | 0.9223 |
| LSTNet-skip | RSE | 0.1843 | 0.2559 | 0.3254 | 0.4643 | 0.4777 | 0.4893 | 0.4950 | 0.4973 | 0.0864 | 0.0931 | 0.1007 | 0.1007 | 0.0226 | 0.0280 | 0.0356 | 0.0449 |
| | CORR | 0.9843 | 0.9690 | 0.9467 | 0.8870 | 0.8721 | 0.8690 | 0.8614 | 0.8588 | 0.9283 | 0.9135 | 0.9077 | 0.9119 | 0.9735 | 0.9658 | 0.9511 | 0.9354 |
| TPA-LSTM | RSE | 0.1803 | **0.2347** | 0.3234 | 0.4389 | 0.4487 | 0.4658 | 0.4641 | 0.4765 | 0.0823 | 0.0916 | 0.0964 | 0.1006 | **0.0174** | **0.0241** | **0.0341** | **0.0444** |
| | CORR | 0.9850 | **0.9742** | 0.9487 | 0.9081 | 0.8812 | 0.8717 | 0.8717 | 0.8629 | 0.9439 | 0.9337 | 0.9250 | 0.9133 | **0.9790** | 0.9709 | 0.9564 | 0.9381 |
| MTGNN | RSE | **0.1778** | 0.2348 | **0.3109** | **0.4270** | **0.4162** | 0.4754 | **0.4461** | **0.4535** | **0.0745** | 0.0878 | **0.0916** | **0.0953** | 0.0194 | 0.0259 | 0.0349 | 0.0456 |
| | CORR | **0.9852** | 0.9726 | **0.9509** | 0.9031 | **0.8963** | 0.8667 | **0.8794** | **0.8810** | **0.9474** | 0.9316 | **0.9278** | **0.9234** | 0.9786 | 0.9708 | 0.9551 | 0.9372 |
| MTGNN+sampling | RSE | 0.1875 | 0.2521 | 0.3347 | 0.4386 | 0.4170 | **0.4435** | 0.4469 | 0.4537 | 0.0762 | **0.0862** | 0.0938 | 0.0976 | 0.0212 | 0.0271 | 0.0350 | 0.0454 |
| | CORR | 0.9834 | 0.9687 | 0.9440 | 0.8990 | 0.8960 | **0.8815** | 0.8793 | 0.8758 | 0.9467 | **0.9354** | 0.9261 | 0.9219 | 0.9788 | 0.9704 | **0.9574** | **0.9382** |

MTGNN+sampling: MTGNN but the network input is sampled a subset of graph each iteration.

# Experiments : Multi-step Forecasting

Horizon: Target future step (3/6/12/24 minutes later)

Multi step: Forecast multiple output values future time-step

| | Horizon 3 | | | Horizon 6 | | | Horizon 12 | | |
|---|---|---|---|---|---|---|---|---|---|
| | MAE | RMSE | MAPE | MAE | RMSE | MAPE | MAE | RMSE | MAPE |
| METR-LA | | | | | | | | | |
| DCRNN | 2.77 | 5.38 | 7.30% | 3.15 | 6.45 | 8.80% | 3.60 | 7.60 | 10.50% |
| STGCN | 2.88 | 5.74 | 7.62% | 3.47 | 7.24 | 9.57% | 4.59 | 9.40 | 12.70% |
| Graph WaveNet | 2.69 | 5.15 | 6.90% | 3.07 | 6.22 | 8.37% | 3.53 | 7.37 | 10.01% |
| ST-MetaNet | 2.69 | 5.17 | 6.91% | 3.10 | 6.28 | 8.57% | 3.59 | 7.52 | 10.63% |
| MRA-BGCN | **2.67** | **5.12** | 6.80% | 3.06 | 6.17 | 8.30% | 3.49 | 7.30 | 10.00% |
| GMAN | 2.77 | 5.48 | 7.25% | 3.07 | 6.34 | 8.35% | **3.40** | **7.21** | **9.72%** |
| MTGNN | 2.69 | 5.18 | 6.86% | **3.05** | **6.17** | **8.19%** | 3.49 | 7.23 | 9.87% |
| MTGNN+sampling | 2.76 | 5.34 | **5.18%** | 3.11 | 6.32 | 8.47% | 3.54 | 7.38 | 10.05% |
| PEMS-BAY | | | | | | | | | |
| DCRNN | 1.38 | 2.95 | 2.90% | 1.74 | 3.97 | 3.90% | 2.07 | 4.74 | 4.90% |
| STGCN | 1.36 | 2.96 | 2.90% | 1.81 | 4.27 | 4.17% | 2.49 | 5.69 | 5.79% |
| Graph WaveNet | 1.30 | 2.74 | **2.73%** | 1.63 | 3.70 | 3.67% | 1.95 | 4.52 | 4.63% |
| ST-MetaNet | 1.36 | 2.90 | 2.82% | 1.76 | 4.02 | 4.00% | 2.20 | 5.06 | 5.45% |
| MRA-BGCN | **1.29** | **2.72** | 2.90% | **1.61** | **3.67** | 3.80% | 1.91 | 4.46 | 4.60% |
| GMAN | 1.34 | 2.82 | 2.81% | 1.62 | 3.72 | **3.63%** | **1.86** | **4.32** | **4.31%** |
| MTGNN | 1.32 | 2.79 | 2.77% | 1.65 | 3.74 | 3.69% | 1.94 | 4.49 | 4.53% |
| MTGNN+sampling | 1.34 | 2.83 | 2.83% | 1.67 | 3.79 | 3.78% | 1.95 | 4.49 | 4.62% |

MTGNN+sampling: MTGNN but the network input is sampled a subset of graph each iteration.

# Summary

Use GNN to learn spatial dependencies between variables.

Learn the relationships between the node pairs (variables) to build networks for GNN.

- Graph learning layer to extract a sparse graph adjacency matrix adaptively based on data.
- Graph convolution model to address the spatial dependencies among variables.
- Temporal convolution module to capture temporal patterns.