A MODEL DETAILS

A.1 BP-GNN

In this section, we write down the equations of the BP-GNN model in its full form. Given the bipartite graph $\mathcal{G} = (\mathcal{Y}, \mathcal{U}, \mathcal{E})$ defined in Section 4, where $\mathcal{Y}$ is a set of N nodes corresponding to the N time series and $\mathcal{U}$ is a set of K auxiliary nodes. Nodes $\mathcal{Y}$ have associated the time series embeddings $\mathbf{z} = \{\mathbf{z}_1, \dots \mathbf{z}_N\}$ and auxiliary nodes $\mathcal{U}$ have associated the embeddings $\mathbf{u} = \{\mathbf{u}_1, \dots \mathbf{u}_K\}$ which are free learnable parameters.

Then, the total amount of nodes $\mathcal{V} = \mathcal{Y} \cup \mathcal{U}$ in the graph is $N + K$. And nodes in $\mathcal{V}$ have associated the node embeddings $\mathbf{h} = \{\mathbf{h}_1, \dots, \mathbf{h}_{N+K}\}$, which at layer $l = 0$ are initialized as $\mathbf{h}^0 = \mathbf{z}||\mathbf{u}$. Then, equations of the Bipartite Graph Neural Network can be defined as:

Step 1 $\mathcal{Y} \to \mathcal{U}$	Step 2 $\mathcal{U} \to \mathcal{Y}$
$i \in \mathcal{U} \quad \& \quad j \in \mathcal{Y}$	$i \in \mathcal{Y} \quad \& \quad j \in \mathcal{U}$
$\mathbf{m}_{ij} = \phi_{e_1}(\mathbf{h}_i^l, \mathbf{h}_j^l, a_{ij})$	$\mathbf{m}_{ij} = \phi_{e_2}(\mathbf{h}_i^l, \mathbf{h}_j^l, a_{ij})$
$\mathbf{m}_i = \sum_{j \in \mathcal{Y}} \phi_{\alpha_1}(\mathbf{m}_{ij}) \mathbf{m}_{ij}$	$\mathbf{m}_i = \sum_{j \in \mathcal{U}} \phi_{\alpha_2}(\mathbf{m}_{ij}) \mathbf{m}_{ij}$
$\mathbf{h}_i^{l+1} = \phi_{h_1}(\mathbf{h}_i^l, \mathbf{m}_i)$	$\mathbf{h}_i^{l+1} = \phi_{h_2}(\mathbf{h}_i^l, \mathbf{m}_i)$

Table 7: BP-GNN formulation.

Where Step 1 propagates messages from nodes $\mathcal{Y}$ (time series) to nodes $\mathcal{U}$ (auxiliary nodes) and Step 2 propagates messages the other way around.

A.2 ARCHITECTURE CHOICES ϕ_e, ϕ_α, ϕ_h, f_{enc} AND f_{dec}

In this section we will use the shortcut MLP_{res} for a one layer MLP with a residual connection which we define as:

Input $\to$ {LinearLayer(nf, nf) $\to$ Swish() $\to$ LinearLayer(nf, nf) $\to$ Addition(Input) } $\to$ Output

Where "nf" represents the number of features.

Decoder f_{dec}

Given the above MLP_{res} definition, the decoder consists of one residual MLP followed by a linear layer:

$\hat{\mathbf{z}}_i \to$ { $\text{MLP}_{res} \to$ LinearLayer(nf, out_dim)} $\to \hat{\mathbf{x}}_{i,t+1:T}$

Encoder f_{enc}

The encoder for METR-LA and PEMS-BAY consists of a linear layer and two consecutive MLP_{res}. Notice the overall structure can be considered an MLP with residual connections among some of its layers.

$\mathbf{x}_{i,t_0:t} \to$ { LinearLayer(in_dim, nf) $\to \text{MLP}_{res} \to \text{MLP}_{res} \to$ concatenate($\mathbf{c}_i$)} $\to \mathbf{z}_i$

In the other datasets (Solar-Energy, Traffic, Electricity and Exchange-Rate) we used the following Convolutional Neural Network as an encoder:

$\mathbf{x}_{i,t_0:t} \to$ {conv1d(in_dim, nf, kernel_size, stride) $\to \text{CNN}_{res}$(nf)

$\to$ conv1d(nf, 2nf, kernel_size, stride) $\to \text{CNN}_{res}$(2nf)

$\to$ conv1d(2nf, 4nf, kernel_size, stride) $\to \text{CNN}_{res}$(4nf)

$\to$ conv1d(4nf, out_dim, 1, 1) $\to$ concatenate($\mathbf{c}_i$) } $\to \mathbf{z}_i$

Where CNN_{res} follows the same architecture than MLP_{res} but replacing the Linear Layers by Conv1d layers with stride=1 and kernel_size=1. In other words, both architectures are equivalent since a CNN

with kernel size and stride 1 is equivalent to an MLP that broadcasts over the batch size and sequence length. Formally, we write CNN_{res} as:

Input $\rightarrow$ {conv1d(nf, nf/2, 1, 1) $\rightarrow$ Swish() $\rightarrow$ conv1d(nf/2, nf, 1, 1) $\rightarrow$ Addition(Input) } $\rightarrow$ Output

Edge update ϕ_e

Consists of two layers MLP. We divide the number of features by two in the intermediate layer for efficiency

$[\mathbf{h}_i, \mathbf{h}_j] \rightarrow$ {LinearLayer(2·nf, nf/2) $\rightarrow$ Swish() $\rightarrow$ LinearLayer(nf/2, nf) $\rightarrow$ Swish() } $\rightarrow \mathbf{m}_{ij}$.

Node update ϕ_h

$[\mathbf{h}_i^l, \mathbf{m}_i] \rightarrow$ {LinearLayer(2· nf, nf) $\rightarrow$ Swish() $\rightarrow$ LinearLayer(nf, nf) $\rightarrow$ Addition($\mathbf{h}_i^l$) } $\rightarrow \mathbf{h}_i^{l+1}$

Edge inference ϕ_α

$[\mathbf{m}_{ij}] \rightarrow$ {LinearLayer(nf, 1) $\rightarrow$ Sigmoid() } $\rightarrow \alpha_{ij}$

B METR-LA AND PEMS-BAY

METR-LA is a traffic dataset collected from the highway of Los Angeles. It contains 207 nodes, a sampling resolution of 5 minutes and 34,272 samples per node (i.e. length of each time series).

PEMS-BAY is also a traffic dataset with 325 nodes located in Bay Area. The sampling resolution is also 5 minutes and it contains 52,116 samples per time series / node.

Both datasets METR-LA and PEMS-BAY can be download from the following link: https://github.com/liyaguang/DCRNN

	(N) #Nodes	# Samples	Resolution	Context length	Pred. length
METR-LA	207	34,272	5 min	12	12
PEMS-BAY	325	52,116	5 min	12	12

Table 8: METR-LA and PEMS-BAY specifications.

B.1 IMPLEMENTATION DETAILS

In this experiment (METR-LA and PEMS-BAY datasets) we used the exact same training configuration as (Shang et al., 2021). Given the whole training time series panel of dimensionality (N $\times$ Length), where Length is the number of samples of the training partition, we construct the input to the network by uniformly slicing windows $\mathbf{x}_{t_0:t} \in \mathbb{R}^{N\times 12}$ of length 12. The ground truth labels are obtained in the same way, by slicing the next 12 time steps $\mathbf{x}_{t:t+12} \in \mathbb{R}^{N\times 12}$. The same slicing process is done in validaton and test with their respective partitions. The sliced windows are uniformly distributed through the whole time series panel.

All our models (FC-GNN, BP-GNN, NE-GNN) have been trained with Adam optimizer, batch size 16 and 2 layers in the GNN module. The number of features "nf" in the hidden layers is 64, The learning rates and number of epochs are provided in the following Table 9.

		Learning Rate	Decay at epochs	Decay factor	Epochs
METR-LA	NE-GNN	$2 \cdot 10^{-3}$	[20, 30, 40]	10	200
	FC-GNN	$5 \cdot 10^{-3}$	[20, 30, 40]	10	200
	BP-GNN	$2 \cdot 10^{-3}$	[20, 30, 40]	10	200
PEMS-BAY	NE-GNN	$2 \cdot 10^{-4}$	[50]	10	300
	FC-GNN	$2 \cdot 10^{-4}$	[50]	10	300
	BP-GNN	$2 \cdot 10^{-4}$	[50]	10	300

Table 9: Learning rates for METR-LA and PEMS-BAY datasets

Time results have been run for batch size 16 in a Tesla V100-SXM GPU. All hyperparameters have been tuned in the validation partition using the following search spaces. The learning rate search space was lr $\in \{1 \cdot 10^{-4}, 2 \cdot 10^{-4}, 5 \cdot 10^{-4}, 1 \cdot 10^{-3}, 2 \cdot 10^{-3}, 5 \cdot 10^{-3}\}$. The number of features per layer "nf" was chosen among nf $\in \{32, 64, 128\}$. The number of layers was chosen among $\{1, 2, 3, 4\}$. The learning rate decay and the decay factor was not modified and left the same as in the original code from (Shang et al., 2021). The number of epochs was chosen large enough to do early stopping in the validation partition when the validation loss stops decreasing.

B.2 BASELINES

Regarding the baselines reported in this experiment, LDS (Franceschi et al., 2019) and STGCN (Yu et al., 2017) are not evaluated on METR-LA and PEMS-BAY in their original papers, therefore, for STGCN we used the results provided in the WaveNet paper (Wu et al., 2019) which were computed by the same WaveNet authors and for LDS we used the results provided in the GTS work (Shang et al., 2021). For the DCRNN timing results we used the unofficial Pytorch implementation https://github.com/chnsh/DCRNN_PyTorch.

B.3 METR-LA AND PEMS-BAY RESULTS WITH STANDARD DEVITATION

In this subsection we report the standard deviations for METR-LA and PEMS-BAY results.

	METR-LA								
	15 min			30 min			60 min		
	MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE
Ablation study									
NE-GNN *(w/o id)*	2.80 ± .01	5.73 ± .02	7.50% ± .03	3.40 ± .00	7.15 ± .02	9.74% ± .12	4.22 ± .01	8.79 ± .04	13.06% ± .26
FC-GNN *(w/o id)*	2.77 ± .00	5.65 ± .01	7.39% ± .06	3.36 ± .00	7.02 ± .02	9.59% ± .06	4.14 ± .01	8.64 ± .06	12.70% ± .11
NE-GNN	2.69 ± .01	5.57 ± .04	7.21% ± .01	3.14 ± .01	6.74 ± .05	9.01% ± .09	3.62 ± .01	7.88 ± .05	10.94% ± .07
Our models									
FC-GNN	2.60 ± .02	5.19 ± .06	6.78% ± .12	2.95 ± .02	6.15 ± .08	8.14% ± .16	3.35 ± .03	7.14 ± .09	9.73% ± .27
BP-GNN (K=4)	2.64 ± .01	5.37 ± .03	7.07% ± .08	3.02 ± .02	6.42 ± .05	8.46% ± .08	3.40 ± .02	7.32 ± .05	9.91% ± .19

Table 10: METR-LA results including standard deviations.

	PEMS-BAY								
	15 min			30 min			60 min		
	MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE
Ablation study									
NE-GNN *(w/o id)*	1.40 ± .00	3.03 ± .01	2.92% ± .02	1.85 ± .00	4.25 ± .02	4.21% ± .04	2.39 ± .01	5.50 ± .03	5.93% ± .07
FC-GNN *(w/o id)*	1.39 ± .00	3.00 ± .01	2.88% ± .01	1.82 ± .00	4.18 ± .01	4.11% ± .01	2.32 ± .01	5.35 ± .03	5.71% ± .02
NE-GNN	1.36 ± .00	2.88 ± .01	2.86% ± .02	1.72 ± .01	3.91 ± .03	3.93 % ± .05	2.07 ± .01	4.79 ± .04	5.04% ± .06
Our models									
FC-GNN	1.33 ± .00	2.82 ± .01	2.79% ± .03	1.65 ± .00	3.75 ± .01	3.72% ± .06	1.93 ± .01	4.46 ± .02	4.53% ± .06
BP-GNN (K=4)	1.33 ± .00	2.82 ± .01	2.80% ± .02	1.66 ± .01	3.78 ± .01	3.75% ± .03	1.94 ± .01	4.46 ± .02	4.57% ± .04

Table 11: PEMS-BAY results including standard deviations.

C SINGLE STEP FORECASTING

Electricity, Solar-Energy, Electricity and Exchange Rate datasets can be downloaded from the following link: https://github.com/laiguokun/multivariate-time-series-data

	#Nodes	# Samples	Resolution	Context length	Pred. length
Solar-Energy	137	52,560	10 min	168	1
Traffic	862	17,544	1 hour	168	1
Electricity	321	26,304	1 hour	168	1
Exchange-Rate	8	7,588	1 day	168	1

Table 12: Dataset specifications.

C.1 IMPLEMENTATION DETAILS

In this experiment we used the exact same training process as (Wu et al., 2020). Similarly to METR-LA and PEMS-BAY, the sample construction process, consists of windows uniformly sliced from the time series panel and inputted to the network. Specifically, in this experiment, given the training time series panel of dimensionality (N $\times$ Length), where Length is the number of samples of the training partition, we slice windows $\mathbf{x}_{t_0:t} \in \mathbb{R}^{N\times 168}$ of length 168 which are the input to our network. In this experiment, we only forecast one time step into the future such that the length of the forecasts and ground truth labels is 1.

All our models (FC-GNN, BP-GNN, NE-GNN) have been trained with Adam optimizer. The number of features "nf" in the hidden layers is 128, the number of layers in the GNN is 2. The encoder network in this experiment is a Convolutional Neural Networks previously described in Appendix A. Following, we provide a table with the the different learning rates, number of epochs and batch sizes for all datasets. All models NE-GNN, FC-GNN and BP-GNN were trained with the same training parameters.

	Learning Rate	Batch Size	Epochs
Solar-Energy	$5 \cdot 10^{-4}$	4	30
Traffic	$2 \cdot 10^{-4}$	2	50
Electricity	$2 \cdot 10^{-4}$	4	80
Exchange-Rate	$1 \cdot 10^{-4}$	4	100

Table 13: Table of hyperparameters for Solar-Energy, Traffic, Electricity and Exchange-Rate.

All hyperparameters have been tuned in the validation partition using the following search spaces. The learning rate search space was lr $\in \{5 \cdot 10^{-5}, 1 \cdot 10^{-4}, 2 \cdot 10^{-4}, 5 \cdot 10^{-4}, 1 \cdot 10^{-3}, 2 \cdot 10^{-3}\}$. The number of features nf was chosen among {32, 64, 128}, the number of layers chosen among {1, 2, 3, 4, 8} by choosing between a trade-off of accuracy and efficiency. The number of epochs was chosen large enough such that the validation loss would stop decreasing. The batch size was set to 4, except for Traffic where it was reduced to 2 in order to fit in memory the more expensive explored configurations.

C.2 EVALUATION METRICS

The evaluation metrics for this experiments are exactly the same as for Lai et al. (2018); Shih et al. (2019); Wu et al. (2020). The following equations are extracted from Lai et al. (2018). The Root Relative Squared Error (RSE) is defined as:

$$RSE = \frac{\sqrt{\sum_{(i,t)\in\Omega_{Test}}(\mathbf{x}_{it}-\hat{\mathbf{x}}_{it})^2}}{\sqrt{\sum_{(i,t)\in\Omega_{Test}}(\mathbf{x}_{it}-\text{mean}(\mathbf{x}))^2}}$$

The Empirical Correlation Coefficient (CORR) is defined as:

$$\text{CORR} = \frac{1}{N}\sum_{i=1}^{N}\frac{\sum_t (\mathbf{x}_{it}-\text{mean}(\mathbf{x}_i))(\hat{\mathbf{x}}_{it}-\text{mean}(\hat{\mathbf{x}}_i))}{\sqrt{\sum_t (\mathbf{x}_{it}-\text{mean}(\mathbf{x}_i))^2 (\hat{\mathbf{x}}_{it}-\text{mean}(\hat{\mathbf{x}}_i))^2}}$$

Where $\mathbf{x}, \hat{\mathbf{x}} \in \mathbb{R}^{N\times T}$.

C.3 SINGLE STEP RESULTS WITH STANDARD DEVITATION

In this subsection we report the standard deviations for Solar-Energy, Traffic, Electricity and Exchange-Rate results.

Dataset		Solar-Energy				Traffic			
		Horizon				Horizon			
Methods	Metrics	3	6	12	24	3	6	12	24
NE-GNN	RSE	.1898 ± .0018	.2580 ± .0013	.3472 ± .0059	.4441 ± .0083	.4212 ± .0007	.4586 ± .0017	.4679 ± .0031	.4743 ± .0036
	CORR	.9829 ± .0003	.9663 ± .0003	.9367 ± .0025	.8905 ± .0052	.8951 ± .0005	.8748 ± .0007	.8700 ± .0015	.8670 ± .0013
FC-GNN	RSE	.1651 ± .0006	.2202 ± .0020	.2981 ± .0035	.3997 ± .0047	.4057 ± .0012	.4395 ± .0049	.4624 ± .0021	.4620 ± .0033
	CORR	.9876 ± .0002	.9765 ±.0003	.9551 ± .0011	.9148 ± .0024	.9024 ± .0006	.8850 ± .0004	.8764 ± .0015	.8751 ± .0007
BP-GNN (K=4)	RSE	.1704 ± .0017	.2257 ± .0020	.3072 ± .0095	.4050 ± .0082	.4095 ± .0012	.4470 ± .0062	.4640 ± .0033	.4641 ± .0024
	CORR	.9865 ± .0002	.9751 ± .0005	.9522 ± .0033	.9138 ± .0024	.8999 ± .0005	.8820 ± .0007	.8744 ± .0015	.8723 ± .0013

Table 14: Solar-Energy and Traffic results including standard deviations.

Dataset		Electricity				Exchange-Rate			
		Horizon				Horizon			
Methods	Metrics	3	6	12	24	3	6	12	24
NE-GNN	RSE	.0762 ± .0007	.0917 ± .0029	.0966 ± .0018	.0994 ± .0019	.0175 ± .0002	.0244 ± .0002	.0338 ± .0006	.0447 ± .0008
	CORR	.9494 ± .0006	.9362 ± .0009	.9308 ± .0006	.9262 ± .0006	.9769 ± .0001	.9686 ± .0005	.9535 ± .0003	.9352 ± .0003
FC-GNN	RSE	.0732 ± .0007	.0907 ± .0041	.0915 ± .0026	.0979 ± .0030	.0174 ± .0001	.0245 ± .0002	.0344 ± .0010	.0450 ± .0013
	CORR	.9521 ± .0008	.9404 ± .0008	.9351 ± .0007	.9294 ± .0006	.9772 ± .0001	.9685 ± .0004	.9538 ± .0008	.9349 ± .0007
BP-GNN (K=4)	RSE	.0740 ± .0010	.0898 ± .0041	.0940 ± .0025	.0980 ± .0019	.0175 ± .0001	.0244 ± .0003	.0339 ± .0004	.0442 ± .0005
	CORR	.9519 ± .0004	.9396 ± .0007	.9345 ± .0007	.9288 ± .0001	.9769 ± .0003	.9684 ± .0002	.9530 ± .0007	.9360 ± .0011

Table 15: Electricity and Exchange-Rate results including standard deviations.

D Inferred Graph Analysis

D.1 Datasets

In the Inferred graph analysis experiment we presented two synthetic datasets, "Cycle Graph Gaussian" and "Noisy Sinusoids". Next, we provide a more detailed explanation on how these datasets have been generated:

- Cycle Graph Gaussians: This dataset consists of a panel of N=10 time series of length T=10.000, where each time series i at time step t has been sampled from the past $t-5$ of another time series $i-1 \mod N$ from the same panel. The resulting multivariate time series adjacency matrix is a Cyclic Directed Graph where each variable in the panel depends on the previously indexed one. Formally the generation process is written as

$$\mathbf{x}_{i,t} \sim \mathcal{N}(\beta \mathbf{x}_{(i-1 \mod N),t-5}; \sigma^2) \tag{8}$$

  Where $\beta = 0.9$ and $\sigma = 0.5$.

- Noisy Sinusoids: Motivated by the Discrete Sine Transformation. This dataset consists of $N = 10$ time series of length $T = 10.000$ generated as the weighted sum of different sinusoids with different frequencies and amplitudes. More formally we define a time series $\mathbf{x}_i$ as:

$$\mathbf{x}_i = \sum_{k=1}^{M} B_{i,m} \sin(2\pi w_{i,m} t) + \epsilon_{i,t} \tag{9}$$

  Where $B_{i,k}$ is sampled from a Uniform distribution $\tilde{B}_{i,m} \sim \mathcal{U}(0,1)$ and further normalized $B_{i,m} = \frac{\tilde{B}_{i,m}}{\sum_k \tilde{B}_{i,m}}$. $w_{i,m}$ is also sampled from a uniform distribution $w_{i,m} \sim \mathcal{U}(0, 0.2)$ and $\epsilon_{i,t}$ is sampled from a Gaussian distribution $\epsilon_{i,t} \sim \mathcal{N}(0, 0.2^2)$. Finally, $B_{i,m}$ and $w_{i,m}$ are shared among the first half signals $1 \leq i \leq 5$ of the panel and among the second half $6 \leq i \leq 10$. We choose $M = 3$.

The time series in both datasets have been splitted in train/val/test as 6K/2K2K. In figures 4 and 5 we plot the first 200 timesteps of the training set of Cycle Graph Gaussians and Noisy Sinusoids datasets respectively.

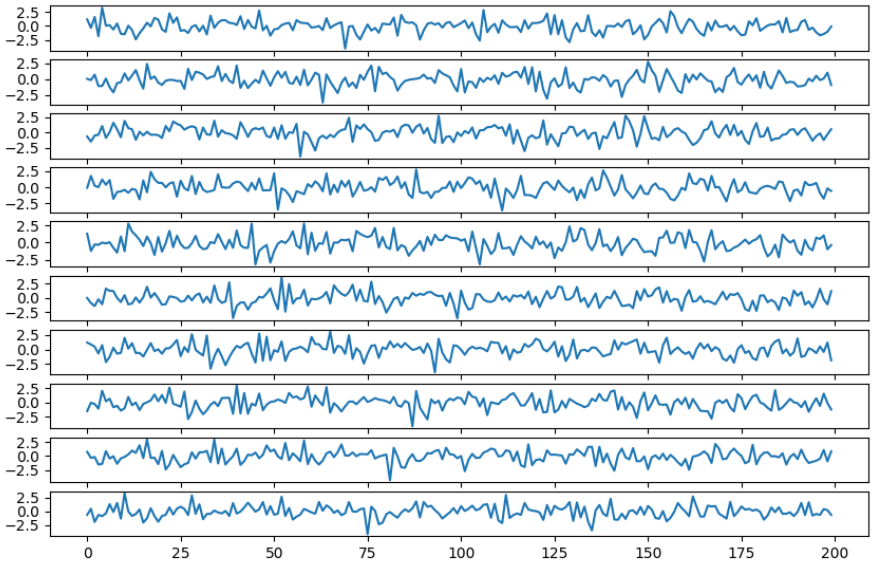


Figure 4: First 200 timesteps of the Cycle Graph Gaussian dataset.

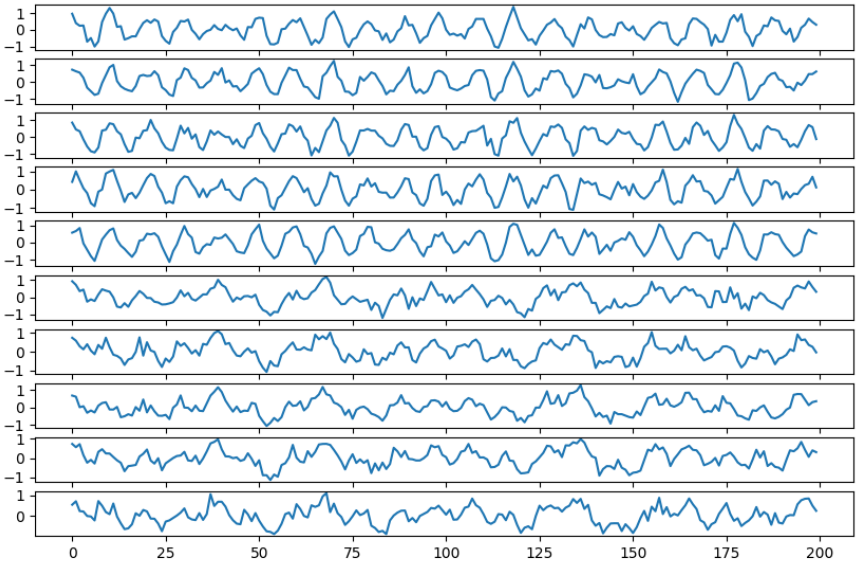


Figure 5: First 200 timesteps of Noisy Sinusoids dataset.

D.2 Implementation details

In the Inferred Graph Analysis, all models have been trained for 100 epochs, learning rate of $2 \cdot 10^{-3}$, weight decay 10^{-14}. The context length (input length of the sequence) is 6, the prediction length (output length of the sequence) is 1. The Mean Absolute Error between prediction and ground

truth has been minimized for training. We also added a small regularization value to the loss $R = \frac{10^{-6}}{\#edges} \sum_{i,j} abs(A_{i,j})$ that pushes unnecessary edges closer to 0 for sharper visualizations.

In both BP-GNN and FC-GNN we used the same MLP encoder than the one used METR-LA and PEMS-BAY experiments defined in the Appendix section A.2. The Graph Convolutional layer consists of only 1 layer. Since the edge inference is dynamic we obtained the reported Adjacency matrices by averaging over 10 different t.

D.3 FURTHER RESULTS

In this section we report further inferred adjacency matrices as in Experiment 5.4 when modifying the random seed. As mentioned in 5.4, different random seeds can produce different adjacency matrices.

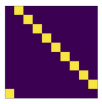
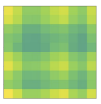
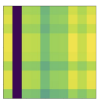
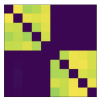
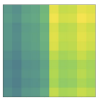
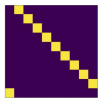
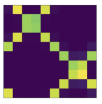
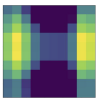
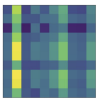
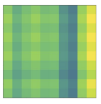
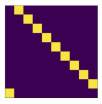
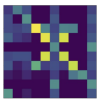
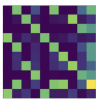
			Expected Adjacency	NE-GNN	FC-GNN	BP-GNN K=10	BP-GNN K=5	BP-GNN K=2	BP-GNN K=1
Seed 0	Cycle Graph Gaussian	Adjacency							
		MAE	-	0.9291	0.4023	0.4098	0.4149	0.4295	0.4955
	Noisy Sinusoids	Adjacency							
		MAE	-	0.2149	0.1868	0.1873	0.1878	0.1881	1.892
			Expected Adjacency	NE-GNN	FC-GNN	BP-GNN K=10	BP-GNN K=5	BP-GNN K=2	BP-GNN K=1
Seed 1	Cycle Graph Gaussian	Adjacency							
		MAE	-	0.9291	0.4020	0.4082	0.4175	0.4241	0.4363
	Noisy Sinusoids	Adjacency							
		MAE	-	0.2141	0.1876	0.1877	0.1895	0.1866	0.1977
			Expected Adjacency	NE-GNN	FC-GNN	BP-GNN K=10	BP-GNN K=5	BP-GNN K=2	BP-GNN K=1
Seed 2	Cycle Graph Gaussian	Adjacency							
		MAE	-	0.9287	0.4024	0.4113	0.4121	0.4249	0.5511
	Noisy Sinusoids	Adjacency							
		MAE	-	0.2143	0.1866	0.1886	0.1918	0.1871	0.1879
			Expected Adjacency	NE-GNN	FC-GNN	BP-GNN K=10	BP-GNN K=5	BP-GNN K=2	BP-GNN K=1
Seed 3	Cycle Graph Gaussian	Adjacency							
		MAE	-	0.9283	0.4009	0.4109	0.4116	0.4313	0.4862
	Noisy Sinusoids	Adjacency							
		MAE	-	0.2159	0.1859	0.1876	0.1873	0.1879	0.1878

Figure 6: Inferred adjacency matrices and MAE losses in the proposed synthetic datasets for seeds $\{0, 1, 2, 3\}$.

In the previous Figure 6 we presented the inferred adjacency matrices for FC-GNN and BP-GNN in a synthetic dataset. In Experiment 5.4 we mentioned that we constructed the BP-GNN matrices $\tilde{A} = A_2 A_1$ as the product of two bipartite graph matrices A_2 and A_1. Following we provide the visualization of the BP-GNN A_2 and A_1.

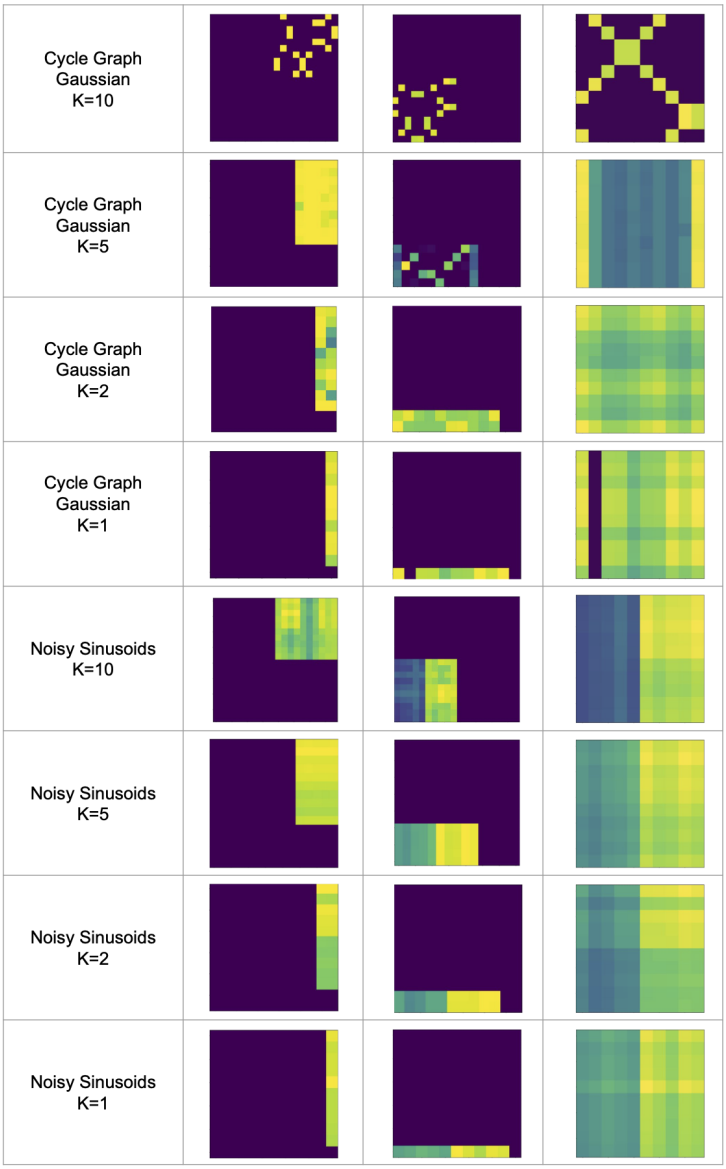


Figure 7: Inferred adjacency matrices in BP-GNN.

D.4 INFERRED GRAPH FOR METR-LA

In this section we analyze the adjacency matrices inferred by our FC-GNN method in METR-LA. Notice that in BP-GNN, matrices $\tilde{A} = A_2 A_1$ are constrained to be very dense for $K << N$ values when nodes communicate. This is because N nodes have to share the same K auxiliary nodes to communicate. METR-LA contains 207 nodes, and we chose BP-GNN to be $K = 4$. This resulted in very competitive performance and very cheap computation time, but the obtained adjacencies are very dense and not as meaningful as the ones obtained from FC-GNN. Therefore, in the following plots, we only report the matrices A inferred by the FC-GNN method. As in the previous synthetic experiment, we build the adjacency matrices in the FC-GNN case from the inferred values $A_{ij} = \alpha_{ij}$. We used the exact same training settings as in the main experiment section 5.2, but this time we only used one graph layer in the FC-GNN module from which we obtained the α_{ij} values. The provided adjacency has been averaged over 10 different t values.

In METR-LA, sensors are spatially located around a city. Therefore, just for comparison, we also report a matrix built from the distance between each pair of sensors. We call it A_{sim}, where each input $A_{sim}[i, j]$ is proportional to the negative distance between the two sensors $A_{sim}[i, j] = \text{bias} - \text{dist}(i, j)$. We can expect to see some correlations between this matrix and the inferred one, but this does not have to be the case. Close sensors can be correlated, but also far away sensors can be correlated (e.g. we can expect far away residential areas to have a simultaneous increase in traffic right before working hours). Matrices are presented in the following figure, A_{sim} and the inferred adjacency A.

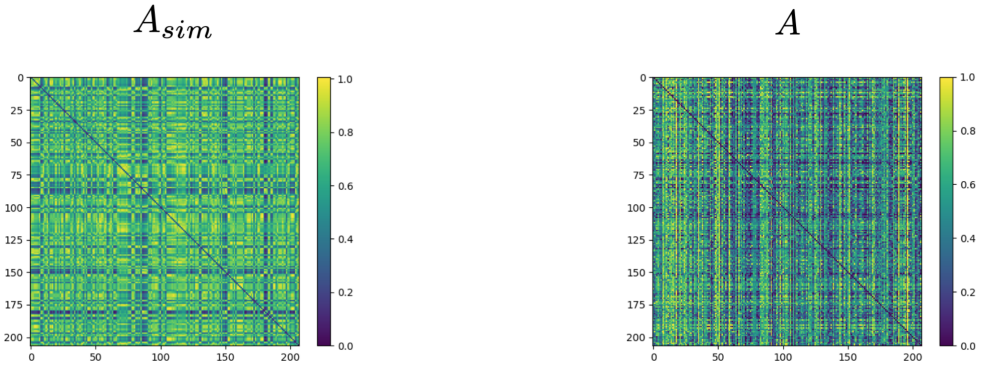


Figure 8: $A_{sim} \rightarrow$ matrix obtained from the negative distance among sensors and $A \rightarrow$ Inferred adjacency matrix by FC-GNN.

Additionally, our model infers matrices dynamically for each t. This means that the inferred adjacency matrix can change depending on the input $\mathbf{x}_{t0:t}$ for different t values. Following, we plot three different matrices for different t values and we see that despite the overall adjacencies share similarities for different t, some components differ as we change t.

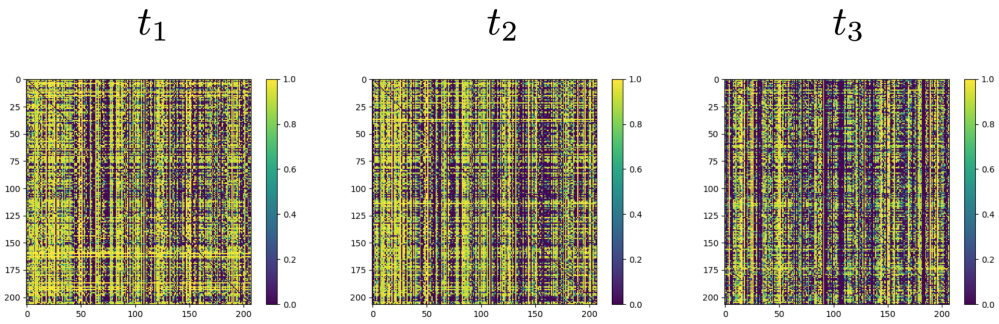


Figure 9: Ajacency matrices for different time steps t.