1. Data preparation:

The traffic data files are available at Google Drive (For Google Drive) or One Drive (For Microsoft One Drive), and should be put into the main/ folder.

2. Create a conda environment for CLGSDN：

1.1 Create an environment.

Create an environment.

```
conda create -n CLGSDN_envs python=3.11
    Proceed ([y]/n)? y
```

Notes：

You can specify another version of Python，but Python>=3.9.

1.2 Activate the environment.

Activate the encironment.

```
conda activate CLGSDN_envs
```

1.3 Install required package.

- Pytorch.

Open the Link: https://pytorch.org/get-started/locally/, find your device, and install Pytorch.

- Other packages.

Install packages.

```
Pip install -r requirement.txt
```

- A special Package.

Pytables cannot be installed via command “pip install”， please use “conda install”.

Install packages.

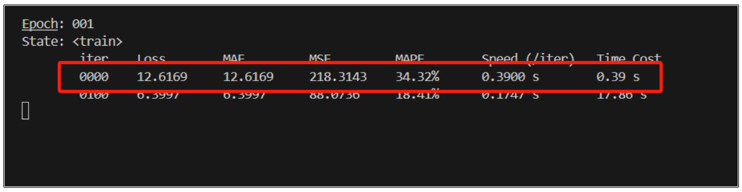
```
Pip install pytables
    Proceed ([y]/n)? y
```

1.4 Test your environment.

A Tesing on environment.

```
Python exp.py
```

The following content indicates that the program ran successfully.



```
Epoch: 001
State: <train>
        iter     Loss       MAE        MSE        MAPE        Speed (/iter)   Time Cost
        0000     12.6169    12.6169    218.3143   34.32%      0.3900 s        0.39 s
        0100     6.3997     6.3997     88.0736    18.41%      0.1747 s        17.86 s
```

3. Conduct the Experiment

3.1 Specify parameters (model, dataset, etc.).

You can experiment with the following command.

Run the code.

```
python exp.py --<argument1> <parameter1> --<argument2> <parameter2>…
```

For example:

Run the code.

```
python exp.py --model_name dcrnn --dataset metr_la --graphgen_name CLGSDN
```

This command indicates that <CLGSDN> is used to generate graphs on the <Metr-la> dataset and the <DCRNN> model is used for prediction.

- Select a model.

  --model_name <model name>

  The options for <model name> are: agcrn, astgcn, tgcn, astgcn, gw, agcrn, dstagnn, lightcts, megacrn and ddgcrn.

- Select a dataset.

Dataset	Data channel	Command
Metr-la	[0]: speed	--dataset metr_la --choise_channels [0]
Pems-bay	[0]: speed	--dataset pems_bay --choise_channels [0]
Pems04	[2]: speed	--dataset pems04 --choise_channels [2]
Pems08	[2]: speed	--dataset pems08 --choise_channels [2]
Taxibj13	[1]: In flow	--dataset taxibj13 --choise_channels [1]
Taxibj13	[-1] or [0,1]: In and out flow	--dataset taxibj13 --choise_channels [-1] --dataset taxibj13 --choise_channels [0,1]
Pems04/08	[0]: flow	--dataset pems04 --choise_channels [0]

When the selected dataset is Metr-la or Pems-bay, the parameter <--choise_channels> does not need to be specified. Its default value is [-1], which means all the channels are selected; However, there is only one channel: vehicle speed.

- Running CLGSDN (as a Graph Generator)

--graphgen_name CLGSDN

This command indicates that the <model> will use the graph generated by <CLGSDN> and will be optimized simultaneously with CLGDSN.

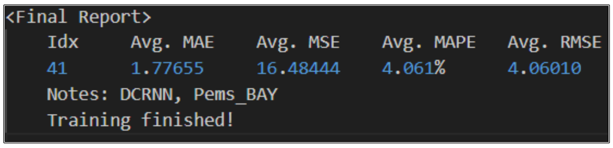
--graphgen_name None

This command means that only the <graph provided by the dataset> is used. If the dataset does not provide any graph, it is the <identity matrix>.

4. Details.

- Final Report

After the program ends, the following result will be output.

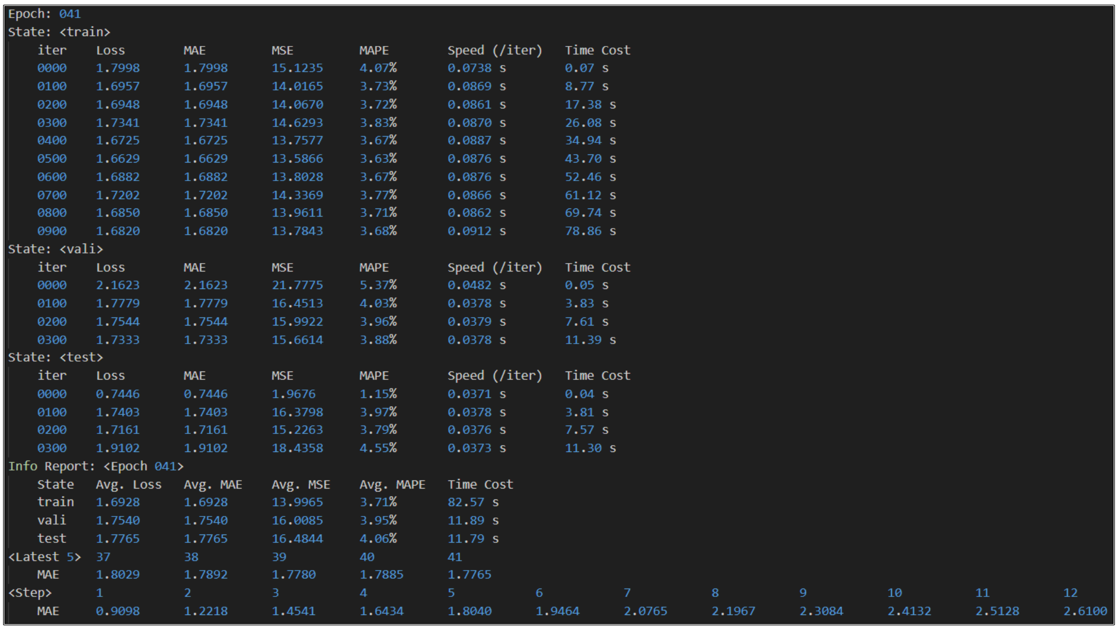


```
<Final Report>
    Idx      Avg. MAE     Avg. MSE     Avg. MAPE     Avg. RMSE
    41       1.77655      16.48444     4.061%        4.06010
    Notes: DCRNN, Pems_BAY
    Training finished!
```

Idx: The epoch with the minimum loss on validation set. (Results below refers to its testing errors.)

- Logs

More detailed information can be found in the log. For example, the log for Epoch 41 is



```
Epoch: 041
State: <train>
    iter    Loss      MAE       MSE       MAPE      Speed (/iter)   Time Cost
    0000    1.7998    1.7998    15.1235   4.07%     0.0738 s        0.07 s
    0100    1.6957    1.6957    14.0165   3.73%     0.0869 s        8.77 s
    0200    1.6948    1.6948    14.0670   3.72%     0.0861 s        17.38 s
    0300    1.7341    1.7341    14.6293   3.83%     0.0870 s        26.08 s
    0400    1.6725    1.6725    13.7577   3.67%     0.0887 s        34.94 s
    0500    1.6629    1.6629    13.5866   3.63%     0.0876 s        43.70 s
    0600    1.6882    1.6882    13.8028   3.67%     0.0876 s        52.46 s
    0700    1.7202    1.7202    14.3369   3.77%     0.0866 s        61.12 s
    0800    1.6850    1.6850    13.9611   3.71%     0.0862 s        69.74 s
    0900    1.6820    1.6820    13.7843   3.68%     0.0912 s        78.86 s
State: <vali>
    iter    Loss      MAE       MSE       MAPE      Speed (/iter)   Time Cost
    0000    2.1623    2.1623    21.7775   5.37%     0.0482 s        0.05 s
    0100    1.7779    1.7779    16.4513   4.03%     0.0378 s        3.83 s
    0200    1.7544    1.7544    15.9922   3.96%     0.0379 s        7.61 s
    0300    1.7333    1.7333    15.6614   3.88%     0.0378 s        11.39 s
State: <test>
    iter    Loss      MAE       MSE       MAPE      Speed (/iter)   Time Cost
    0000    0.7446    0.7446    1.9676    1.15%     0.0371 s        0.04 s
    0100    1.7403    1.7403    16.3798   3.97%     0.0378 s        3.81 s
    0200    1.7161    1.7161    15.2263   3.79%     0.0376 s        7.57 s
    0300    1.9102    1.9102    18.4358   4.55%     0.0373 s        11.30 s
Info Report: <Epoch 041>
    State   Avg. Loss Avg. MAE  Avg. MSE  Avg. MAPE Time Cost
    train   1.6928    1.6928    13.9965   3.71%     82.57 s
    vali    1.7540    1.7540    16.0085   3.95%     11.89 s
    test    1.7765    1.7765    16.4844   4.06%     11.79 s
<Latest 5>  37        38        39        40        41
    MAE     1.8029    1.7892    1.7780    1.7885    1.7765
<Step>      1         2         3         4         5         6         7         8         9         10        11        12
    MAE     0.9098    1.2218    1.4541    1.6434    1.8040    1.9464    2.0765    2.1967    2.3084    2.4132    2.5128    2.6100
```

where:

<Info Report> is the result on the training/validation/test set.

<Step 1-12> is the MAE result of 12 steps (on test set). For example, the results in the paper DCRNN or Graph Wavenet show the errors at the 3rd step (15 min), the 6th step (30 min), and the 12th step (1 hour).

- Reproduce the results (Baseline)

ATTENTION PLEASE: Our experiment focuses on performance changes (whether to use CLGSDN as the Graph Generator). Thus, **All BASELINE SETTINGS ARE THE SAME,** AND **THEIR PERFORMACE IS NOT ALWAYS THE BEST (Compared to the original paper).**

If you want to fully reproduce the results of certain models (of certain published paper) through this program, please ensure the consistency of parameters. For example, if you want to reproduce the results of <DCRNN> on the <Metr-la> dataset, use the following command:

DCRNN.

```
python -exp.py --model_name dcrnn --graphgen_name None --dataset metr_la
    --dataset_prob [0.7,0.1,0.2] --epochs 100
```

The results are:

Source	MAE			
	15 min	30 min	1 hour	Average
Original	2.77	3.15	3.60	\
This Implementation	2.63	3.04	3.62	3.02
With CLGSDN	2.605	2.98	3.54	2.97

Table: MAE Results of DCRNN

However, the parameters required for each baseline are different.

- Reproduce the results (Ours)

Using the command <--graphgen_name CLGSDN>. For example:

DCRNN.

```
python -exp.py --model_name dcrnn --graphgen_name CLGSDN --dataset
    metr_la --dataset_prob [0.7,0.1,0.2] --epochs 100
```