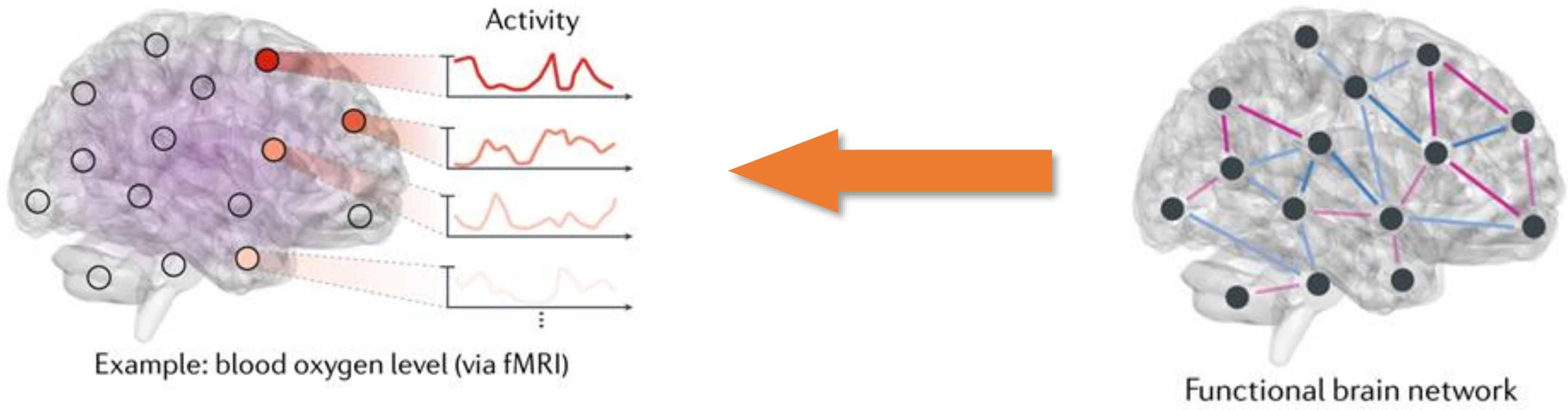
Neural **R**elational **I**nference for interacting systems

by

Kipf, Thomas, et al., ICML 2018

NRI – Backgrounds

- Various kinds of **time series data** are governed by an underlying **network**.



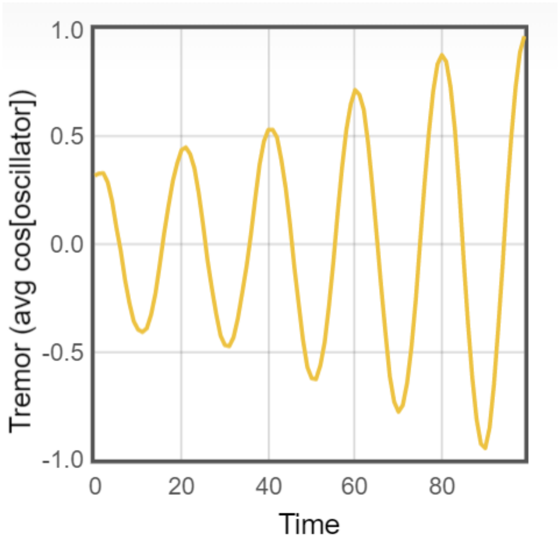
Lynn, Christopher W., and Danielle S. Bassett. "The physics of **brain network** structure, function and control." *Nature Reviews Physics*. 2019

Time series data ***Network***

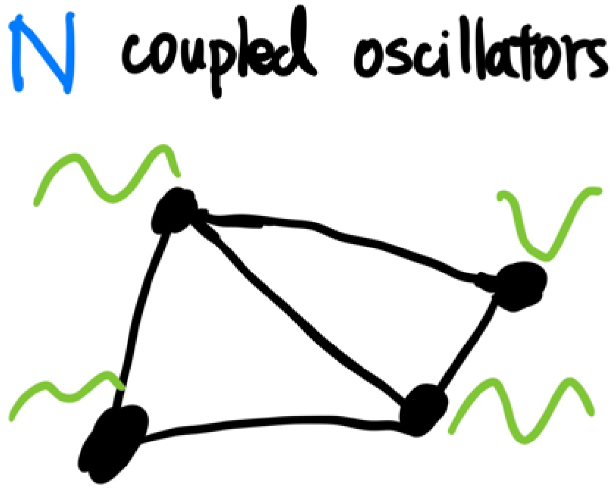
NRI – Backgrounds

- Various kinds of **time series data** are governed by an underlying **network**.

Coupled oscillators $$\dot{\boldsymbol{\theta}}_{\boldsymbol{n}}(\boldsymbol{t}) = \boldsymbol{\omega}_{\boldsymbol{n}} + \sum_{\boldsymbol{m}=\mathbf{1}}^{\boldsymbol{N}} \boldsymbol{K}_{\boldsymbol{mn}} \sin\left(\boldsymbol{\theta}_{\boldsymbol{m}}(\boldsymbol{t}) - \boldsymbol{\theta}_{\boldsymbol{n}}(\boldsymbol{t})\right)$$



Time series data



Network

NRI – Backgrounds

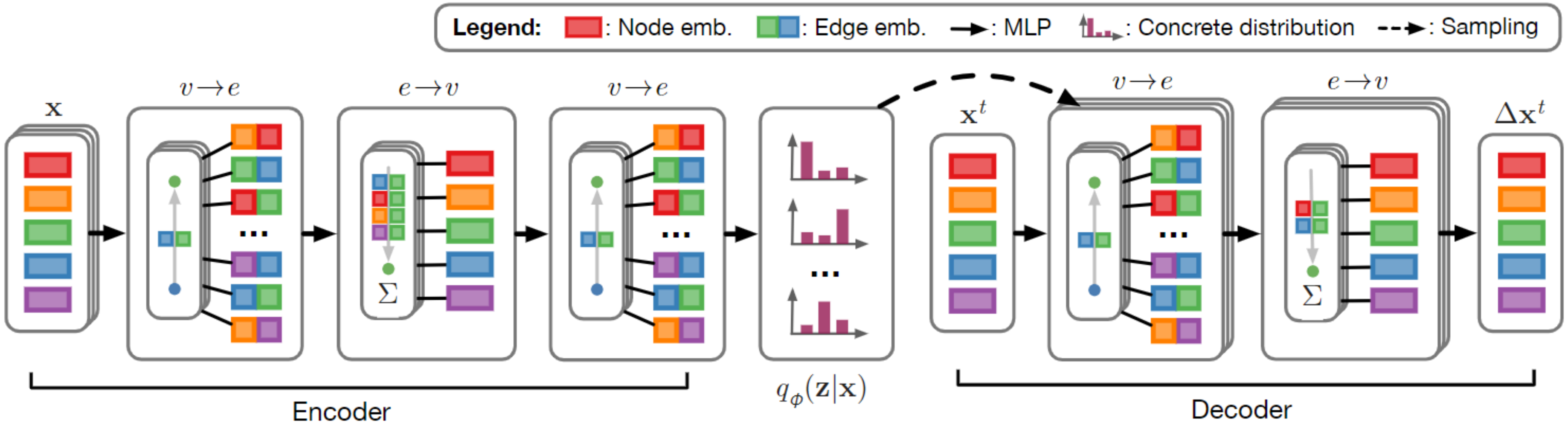
- Besides examples in nature, human behaviors can also be governed by network interactions.
  - Motion of players in a basketball game
  - Traffic, social activities…

Time series data → **Network**

Can we reversely inference the network interactions?

- Statistical methods exist, such as correlation and granger causality. Here we present and reimplement a neural network method, **N**eural **R**elational **I**nference.

NRI – Methods



- NRI consists an **encoder** and a **decoder** network.
- In the implementation, NRI takes the advantage of **G**raph **N**eural **N**etwork (GNN) and **V**ariational **A**uto **E**ncoder (VAE).
- We will introduce NRI step by step!

NRI – Methods

1. GNN refers to a set of neural networks that takes **graph structured data as inputs**.
2. GNN are neural models that **captures the dependence of graphs** via message passing between the nodes of graphs (Zhou et al., 2020).

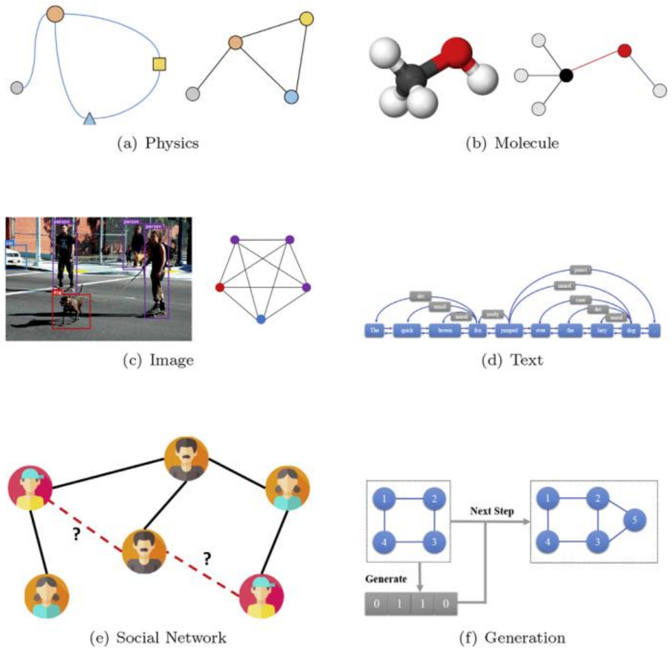


Fig. 6. Application scenarios. (Icons made by Freepik from Flaticon)

Zhou et al., 2020

How GNN captures the dependence of graphs?

NRI – Methods

- In neural network model, embedding/representation might be the key for successful learning.
- In GNN, we design **node embedding** and **edge embedding** for training.

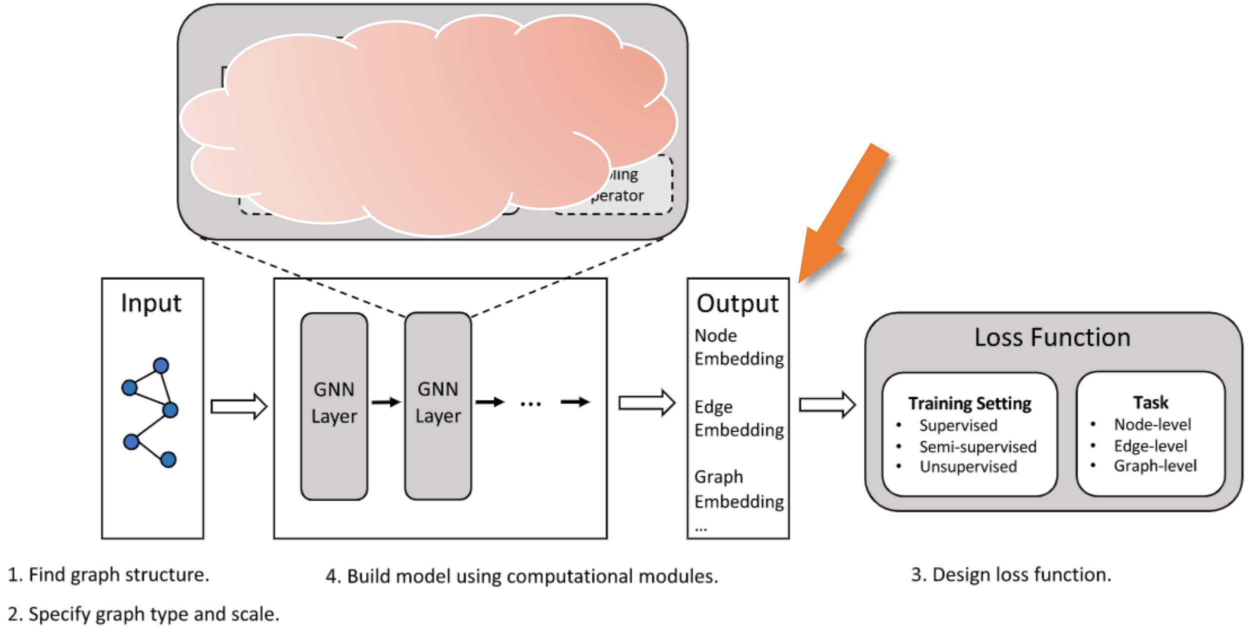


Fig. 2. The general design pipeline for a GNN model.

Zhou et al., 2020

NRI – Methods

- In neural network model, embedding/representation might be the most important key for successful learning.
- In GNN, we design **node embedding** and **edge embedding** for training.

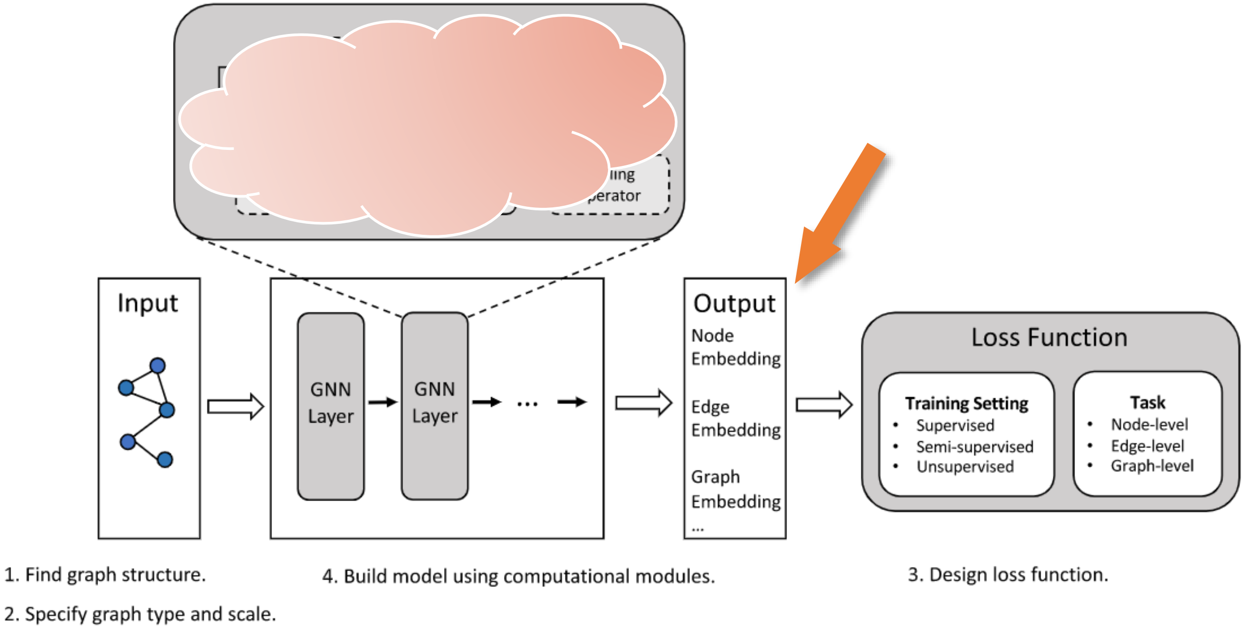


Fig. 2. The general design pipeline for a GNN model.

$$\mathbf{h}_j^1 = f_{\mathrm{emb}}(\mathbf{x}_j)$$

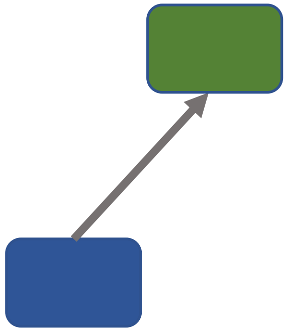
$$v \to e: \quad \mathbf{h}_{(i,j)}^1 = f_e^1([\mathbf{h}_i^1, \mathbf{h}_j^1])$$

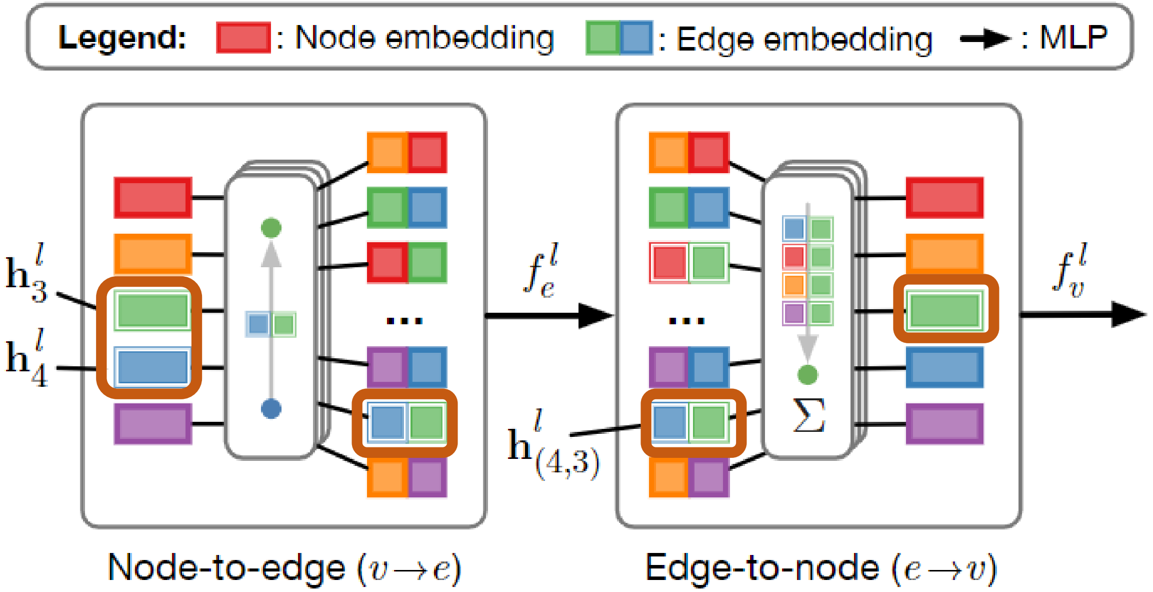
$$e \to v: \quad \mathbf{h}_j^2 = f_v^1\left(\sum_{i \neq j} \mathbf{h}_{(i,j)}^1\right)$$

The GNN Layer can be MLP, RNN or even CNN.

Zhou et al., 2020

NRI – Methods





Kipf et al., 2018

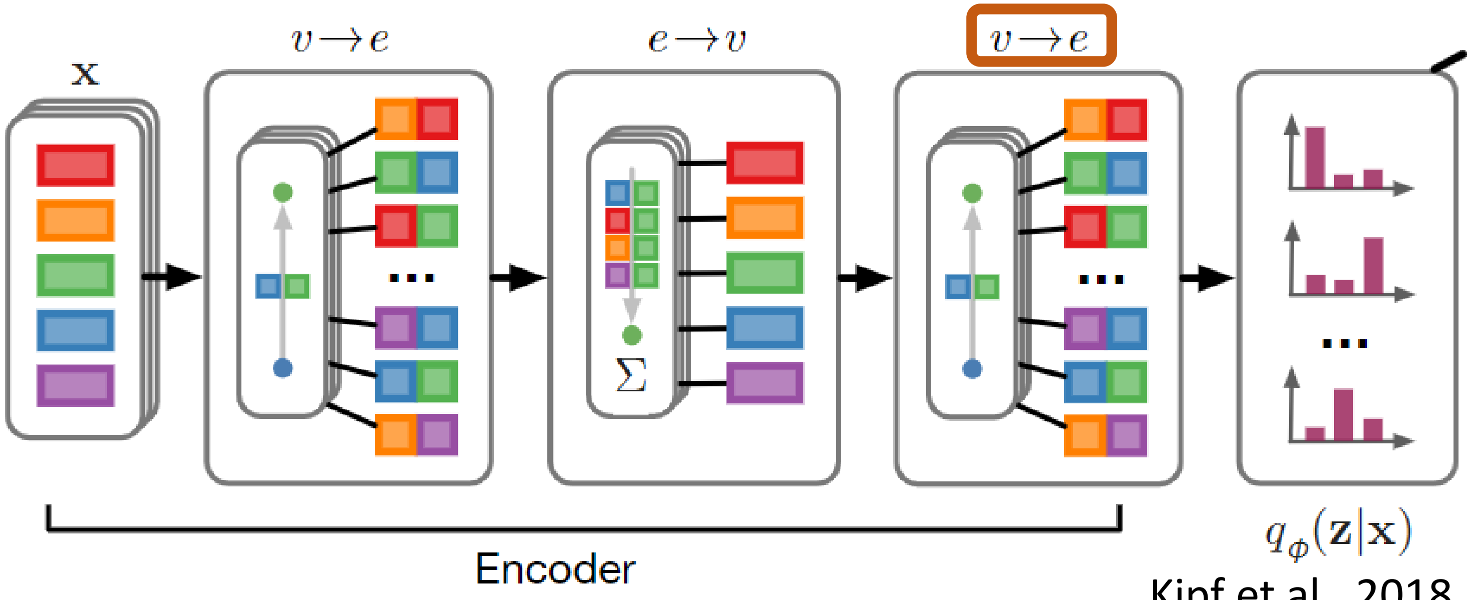
h_3^l, h_4^l are node embeddings. $h_{(4,3)}^l$ is edge embedding.

f_e^l is a neural network that takes edge embeddings as inputs.

f_v^l is a neural network that takes node embeddings as inputs.

NRI – Methods

- NRI consists of an encoder and a decoder. In the last stage, the edge embeddings are used to perform **edge classification**.
- Although we use encoder as the example, ideas of node/edge embedding are also in the design of decoder.

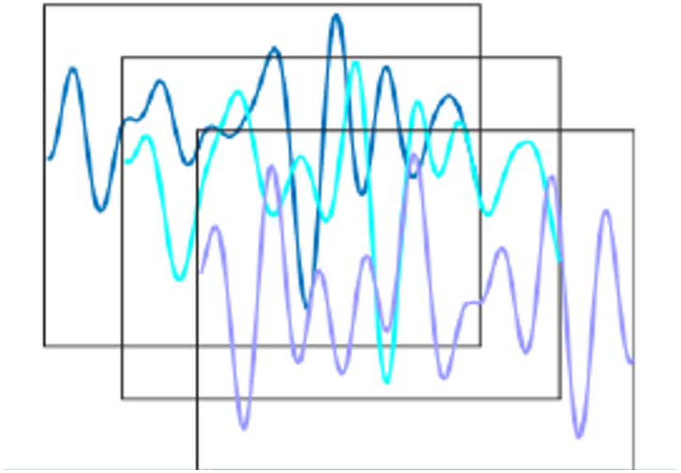


Kipf et al., 2018

$\boldsymbol{z}$ represents edge types.
$\boldsymbol{q_\phi}(\mathbf{z}|\mathbf{x})$ is the probability of edge types for edge $\boldsymbol{z_{ij}}$.

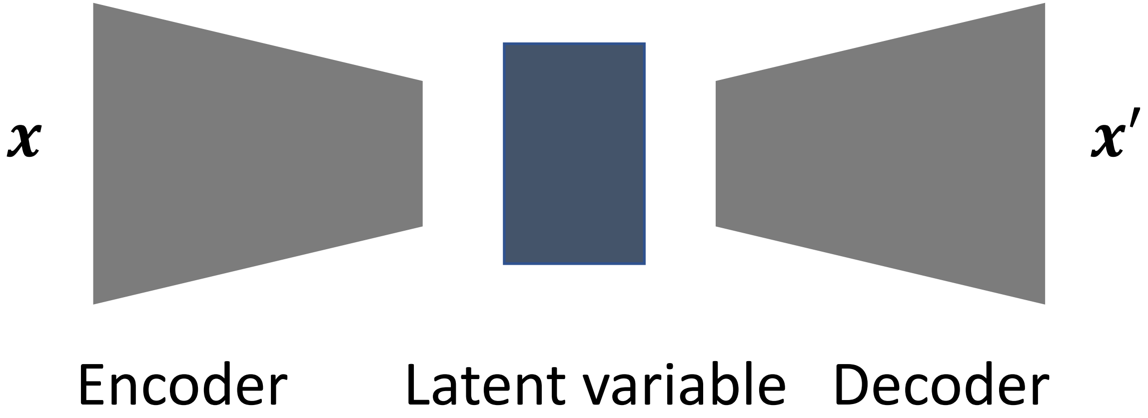
NRI – Methods

- If our training dataset contains ground truth edge types, then **Encoder alone** can inference the edge type for testing dataset.
  This is true for **simulation data**.
- However, in some cases (e.g. online edge inference), what we **ONLY** have is the time series.
  That is, our training dataset does not contain ground truth edge types.



NRI – Methods

- To overcome the barrier, NRI takes the advantage of VAE.
- VAE is suitable for **data without labelling** (unsupervised learning).

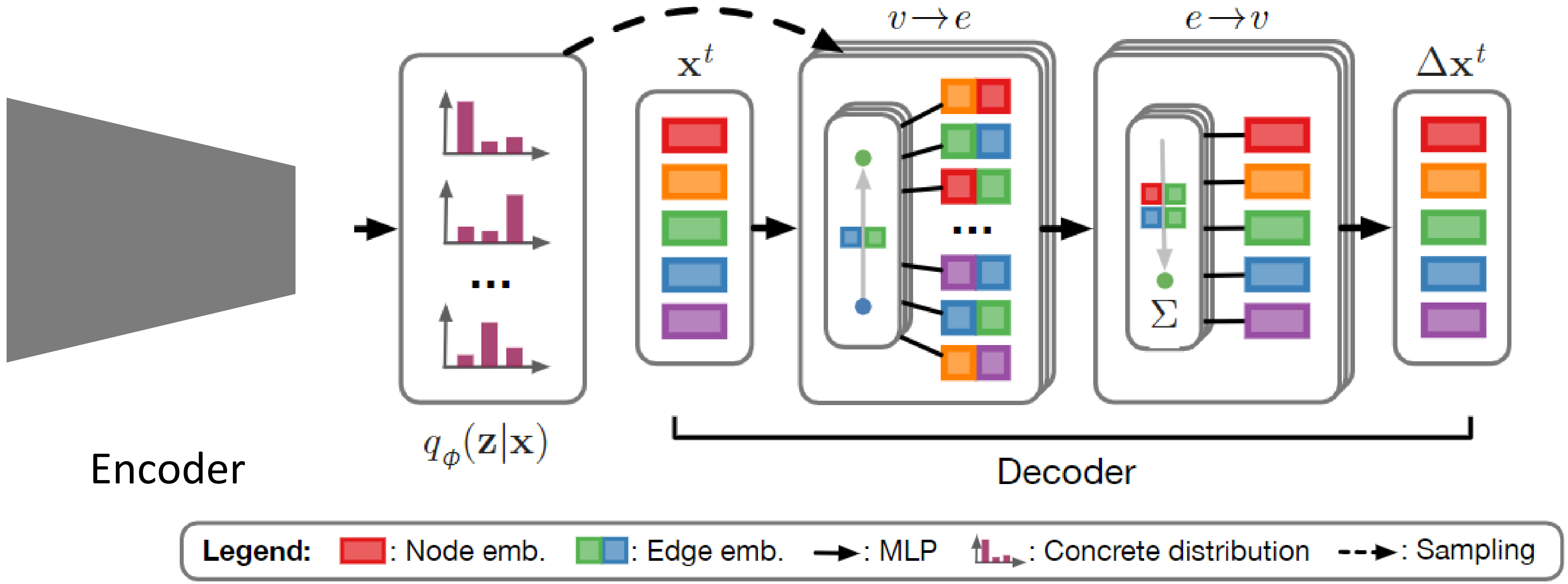


The encoder compresses input $\boldsymbol{x}$ to the latent space.

The decoder produces $\boldsymbol{x}'$ (from latent space) as similar as $\boldsymbol{x}$.

NRI – Methods

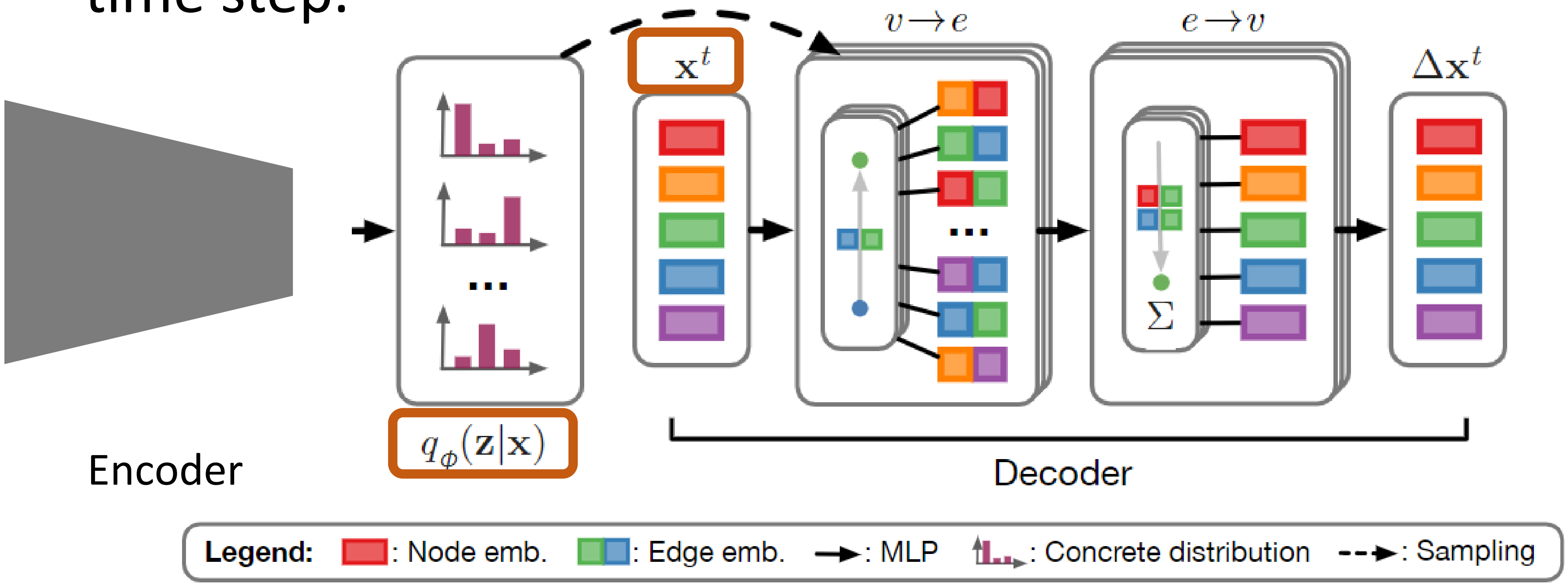
- The decoder has a similar structure as the encoder.
  - MLP layer
  - $V \rightarrow E$ and $E \rightarrow V$ operations

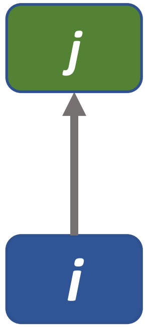


Kipf et al., 2018

NRI – Methods

- The decoder takes x_t and edge type probabilities q_ϕ as input. Two types of inputs are combined.
- The decoder output Δx^t is the predicted increasement for the next time step.



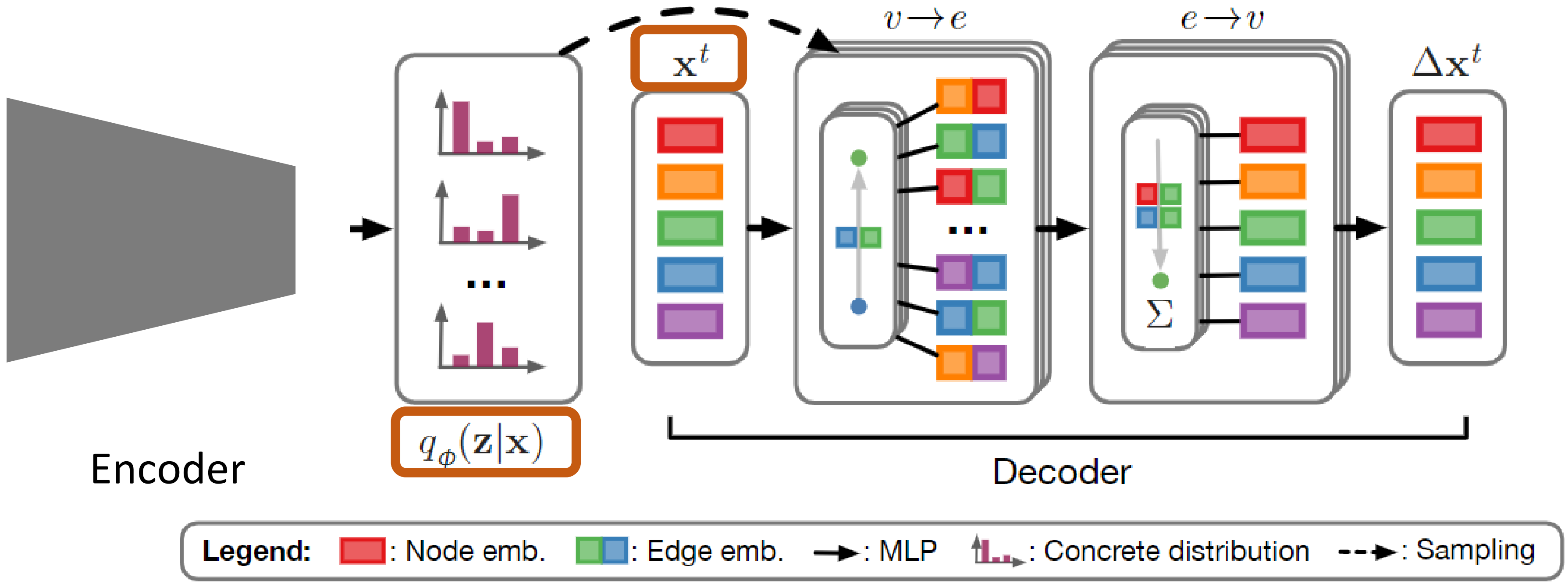


$* \ q_\phi(z_{ij}|x)$

Kipf et al., 2018

NRI – Methods

- To sum up, in case we **ONLY** have time series, we have to infer the edge (interaction) types through VAE. If the latent variable $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ represents edge types correctly, the decoder should be able to produce matched increasement.



Kipf et al., 2018

NRI – Implementations

- First, let's take a look at the encoder.

```
class MLPBlock(nn.Module):
    pass
class MLPEncoder(nn.Module):
    pass
```

src/nri_encoder.py

```
def parse_args():
    pass
def train(args):
    """Performs both training and validation
    """
    pass
def test(args, best_model_path):
    """Test the best performance model
    """
    pass
if __name__ == "__main__":
    pass
```

scripts/train_enc.py

NRI – Implementations

- let’s take a look at the decoder.

```python
class MLPDecoder(nn.Module):
    pass
```

src/nri_decoder.py

```python
def parse_args():
    pass
def reconstruction_error(pred, target):
    pass
def train(args):
    """Performs both training and validation
    """
    pass
def test(args, best_model_path):
    """Test the best performance model
    """
    pass
if __name__ == "__main__":
    pass
```

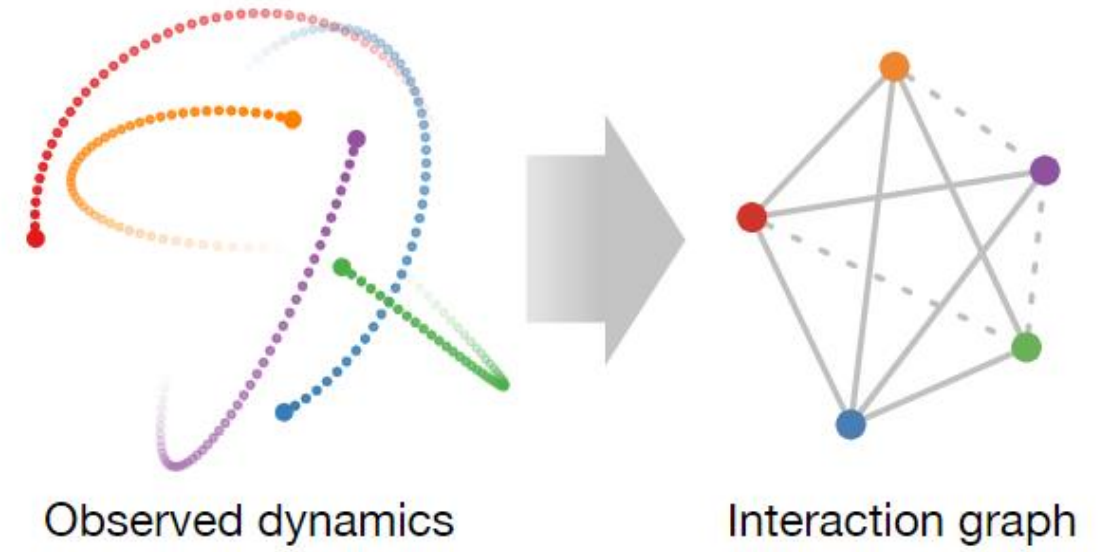
scripts/train_dec.py

NRI – Implementations

- Use `scripts/generate_dataset.py` to generate simulation data. This file is provided by the paper author.
- You can use `run_encoder.py` and `run_decoder` to play with the trained model. Use `traj_plot.ipynb` to generate prediction trajectory.
- Let’s go to Github for details.

NRI – Results, Encoder

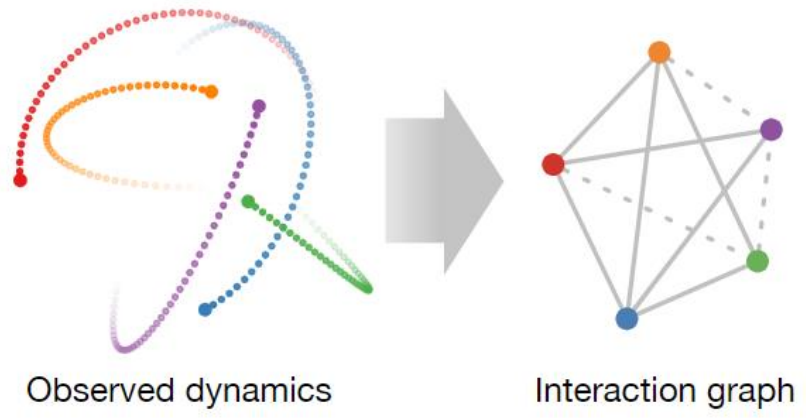
- In the experiments, we use simulated 2D springs-coupled data in training and testing.



Kipf et al., 2018

NRI – Results, Encoder

- We train the encoder with 40 epochs, batch_size=5, each trial with 49 times steps. Use `nn.CrossEntropyLoss` as the loss function. The hidden layer dimension of MLP layer is 256.



Test loss: 0.20062
Test acc: **0.91605**

NRI – Results, Encoder

- We train the encoder with 40 epochs, batch_size=5, each trial with 49 times steps. The hidden layer dimension of MLP layer is 256.

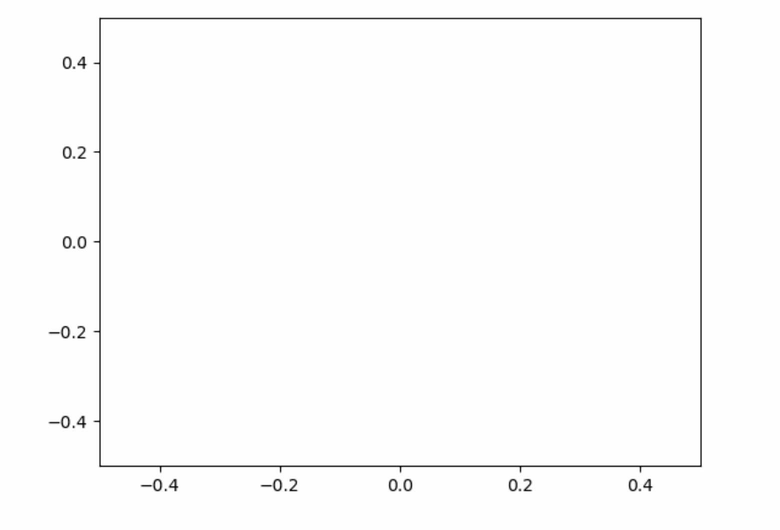
```
array([[ 2.58110523e-01,  2.72232652e-01],
       [-1.16690552e+00,  2.87807488e+00],
       [ 2.96102691e+00, -3.51592392e-01],
       [-1.77556610e+00,  5.04293703e-02],
       [ 5.95626593e-01,  2.90788710e-01],
       [-2.01289678e+00,  2.76325107e+00],
       [ 1.98042321e+00, -1.14150584e-01],
       [-3.09398627e+00,  1.89460647e+00],
       [-3.32151628e+00,  6.86369181e-01],
       [-3.44780421e+00,  1.92884123e+00],
       [ 6.40936494e-01, -1.27347276e-01],
       [-1.68725765e+00, -8.71044755e-01],
       [ 1.71941495e+00, -1.61469197e+00],
       [ 8.53864312e-01, -7.06104159e-01],
       [ 1.14158404e+00, -8.64673257e-02],
```

```
array([0., 1., 0., 1., 0., 1., 0., 1., 1., 1., 0., 1., 0., 0., 0., 1., 1.,
       1., 1., 1., 1., 0., 0., 0., 1., 0., 0., 0., 0., 0., 1., 0., 0., 0.,
       1., 1., 0., 0., 0., 1., 1., 0., 0., 0., 1., 0., 1., 1., 0., 0., 0.,
       0., 0., 1., 0., 0., 0., 1., 0., 0., 1., 0., 0., 1., 1., 1., 1., 0.,
       0., 1., 1., 0., 0., 1., 1., 0., 1., 0., 0., 0., 1., 1., 1., 0., 1.,
       0., 0., 1., 1., 0., 0., 1., 1., 0., 0., 0., 0., 1., 1., 0.])
```

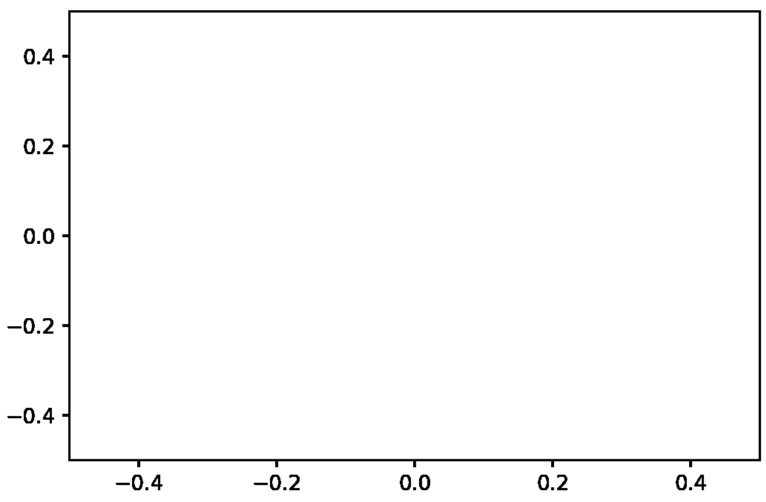
Any idea to better present the result?

NRI – Results, Decoder-1

Use decoder alone, provided with edge types,
1 Step ahead prediction, then teacher force



Target trajectory



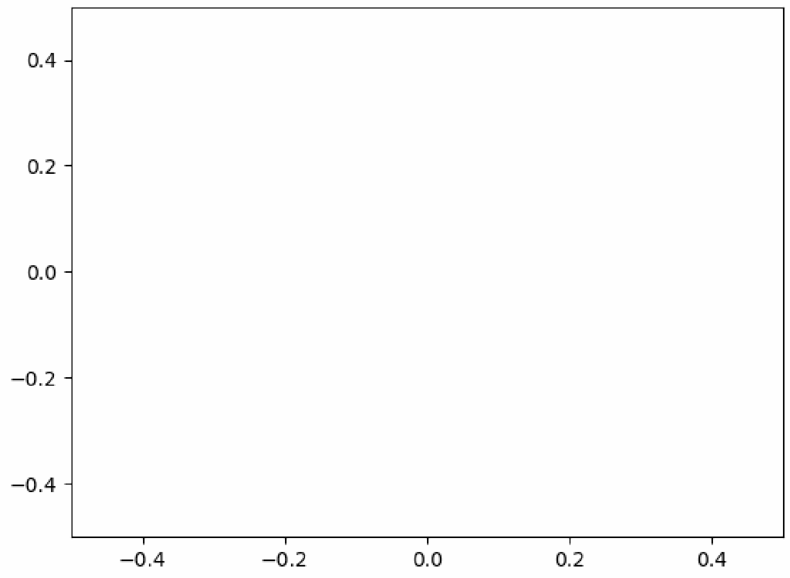
Prediction trajectory

Test error: **0.009824**

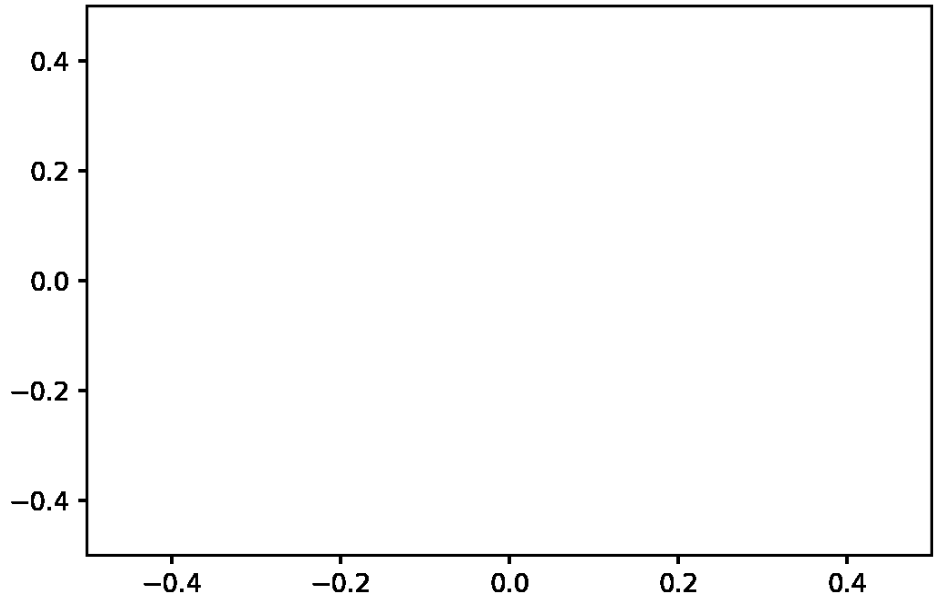
Test baseline error: 0.030993

NRI – Results, Decoder-2

<u>10</u> Steps prediction, then teacher force



Target trajectory



Prediction trajectory

NRI – Paper Results

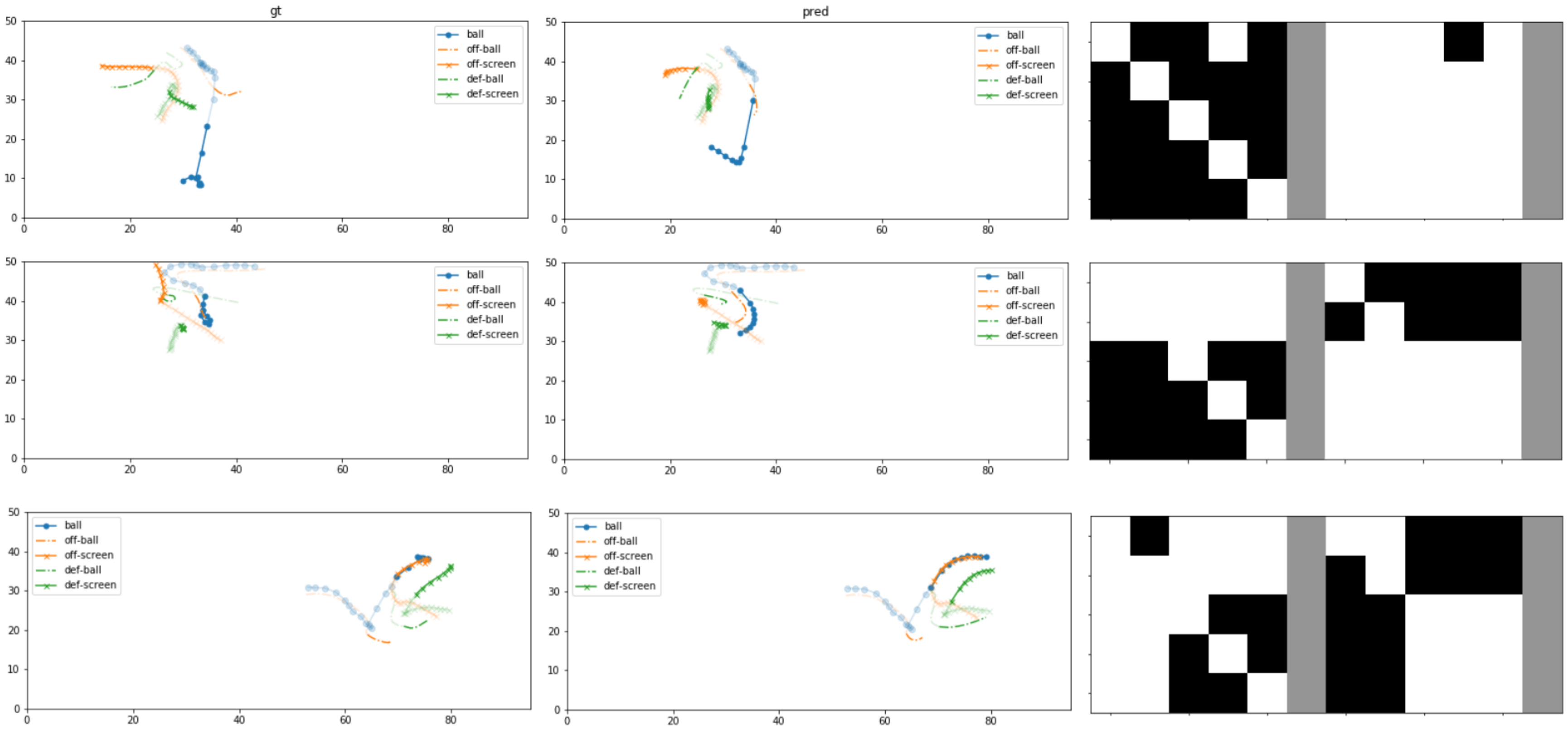


Figure 11. Visualization of NBA trajectories. *Left*: ground truth; *middle*: model prediction; *right*: sampled edges.

NRI – Issues

- Appreciate any helps for the followings issues.

- 1. Test and add support for GPU platform.
- 2. MLP building block has been implemented for the encoder and decoder. Consider other building blocks (e.g. CNN, RNN).
- 3. For data without ground truth edge information, we need to combine the encoder and the decoder together (**optional**).
- 4. Visualization for encoder output result.

- Any minor adjustments are also welcomed! ☺

Reference

Kipf, Thomas, et al. "Neural relational inference for interacting systems." *International Conference on Machine Learning*. PMLR, 2018.

Zhou, Jie, et al. "Graph neural networks: A review of methods and applications." *AI Open* 1 (2020): 57-81.