ChemGNS: Train, Test, Predict!

Fabiana Ferracina

May 3, 2024

Pacific Northwest NATIONAL LABORATORY

WASHINGTON STATE UNIVERSITY

Our Contribution to Current GNS

- Inspired by Kumar and Vantassel 2022's Pytorch GNS: `https://www.geoelements.org/gns`
- Multi-dimensional time-changing features
- Multi-dimensional node properties
- Alternative activation functions
- Prediction pipeline
- Data transformation pipeline
- Output analysis pipeline

Be Prepared 

Open your terminal and go to the directory where folders `chem_data` and `gns` were placed. Make sure you are in the environment containing the necessary packages.

```
# Prepare the raw data - assumed to be in specific txt format
python -m chem_data.chemgns --action='prepare'
    --raw_data_path='<raw-data-path>'
        --preped_data_path='<output path for prepared data>'
    --universe=<integer> --material_properties='material
        property list'
    --gases='gas chemistry list' --particle_chem='particle
        chemistry list'
    --share_path='<path for sharing files between processes>'
```

```
# Train for the first time
python -m gns.train --data_path='<prepared data path>'
    --model_path='<model storage path>'
    --output_path='<rollout storage path>'
        -ntraining_steps=<integer total steps>
# Train some more
python -m gns.train --data_path='<prepared data path>'
    --model_path='<model storage path>'
    --output_path='<rollout storage path>'
        --model_file='model-<last timestep>.pt'
    --train_state_file='train_state-<last timestep>.pt'
        -ntraining_steps=<integer total steps>
```


```
# Create a rollout using the test dataset
python -m gns.train --mode='rollout'
    --data_path='<prepared data path>'
    --model_path='<model storage path>'
    --output_path='<rollout storage path>'
    --model_file='model-<last timestep>.pt'
    --train_state_file='train_state-<last timestep>.pt'
```

Analyze

The rollout dataset needs to be prepared for analysis. This is were we bring the data back to a state that makes sense to scientists.

```
# Prepare the raw data - assumed to be in specific txt format
python -m chem_data.chemgns --action='analyze'
    --rollout_data_path='<rollout storage path>'
    --material_properties='material property list'
    --gases='gas chemistry list' --particle_chem='particle
        chemistry list'
    --proc_data_path='<path for rollout dictionaries>'
    --share_path='<path for sharing files between processes>'
```

Or just run.py

run.py was written to prepare, train, test, and ready the data for analysis with one simple command: `python run.py`, and some variable settings:

```
    #### Set these as appropriate:
PartMC-MOSAIC Data:
raw_data_path = "./chem_data/processed_output_some/"
rollout_dicts = "./chem_data/proc_data/"
npz_path = "./gns/data/"
model_path = "./gns/model/"
rollouts_path = "./gns/output/"
material_properties = ['aero_number', 'BC', 'OC']
particle_chem = ['H2O', 'SO4']
gases = ['H2SO4']
train_steps = 300
scenarios = [0, 1, 3, 8]
total_reps = 0 # repeat one scenario n times
```

Predictions

Have a folder with the initial values for each chemistry in txt format.

```
# Prepare the raw data for prediction
python -m chem_data.chemgns --action='predict'
    --raw_data_path='<raw-data-path>'
        --preped_data_path='<output path for prepared data>'
    --universe=<integer> --material_properties='material
        property list' --gases='gas chemistry list'
        --particle_chem='particle chemistry list'
        --share_path='<path for sharing files between
        processes>'
# Predict!
python -m gns.train --mode='predict' --data_path='<prepared
    data path>' --model_path='<model storage path>'
    --output_path='<rollout storage path>'
    --model_file='model-<last timestep>.pt'
    --train_state_file='train_state-<last timestep>.pt'
```

Package requirements

```
glob, pathlib, os, re, absl, pickle,
matplotlib, numpy, random, json, collections,
sys, time, tqdm, typing, torch, pyg
```

PyG

For Pytorch-geometric (pyg) find out your CUDA from python IDE:

```
[In] print(f"PyTorch has version {torch.__version__}
    with cuda {torch.version.cuda}")
[Out] PyTorch has version 2.1.0+cu121 with cuda 12.1
```

In terminal:

```
# Install torch geometric
pip install torch-cluster -f
    https://data.pyg.org/whl/torch-2.1.0+cu121.html
pip install torch-scatter -f
    https://data.pyg.org/whl/torch-2.1.0+cu121.html
pip install torch-sparse -f
    https://data.pyg.org/whl/torch-2.1.0+cu121.html
pip install torch-geometric
```

CUDA Troubleshooting

```
RuntimeError: ... something something CUDA ...
```

Or you get a warning:

```
UserWarning: CUDA initialization: ...
rank = None, cuda = False
Training step: 0/1000. Loss: 6.990038871765137.
Training step: 1/1000. Loss: 6.946468830108643.
# slower
```

Simply run the following:

```
sudo rmmod nvidia_uvm
sudo modprobe nvidia_uvm
```

Results from GNS Simulations: Water

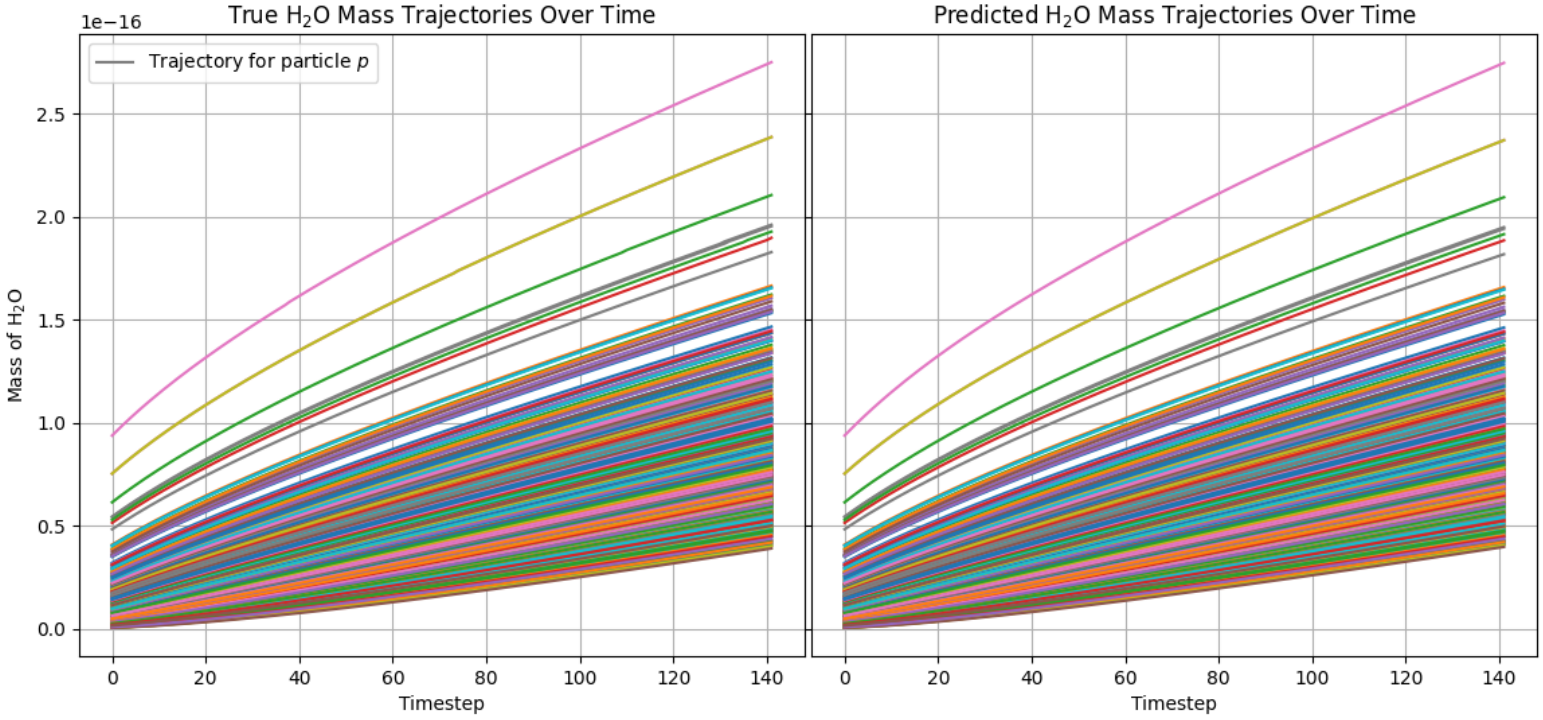


Figure: Results from Chem GNS on a simple particle-gas system. 142 timesteps, 808 particles, 3 time changing features, 3 time fixed particle properties, 2 chemical types. NMAE ≈ 0.004. Time to predict ≈ 0.4 seconds

Results from GNS Simulations: Sulfate

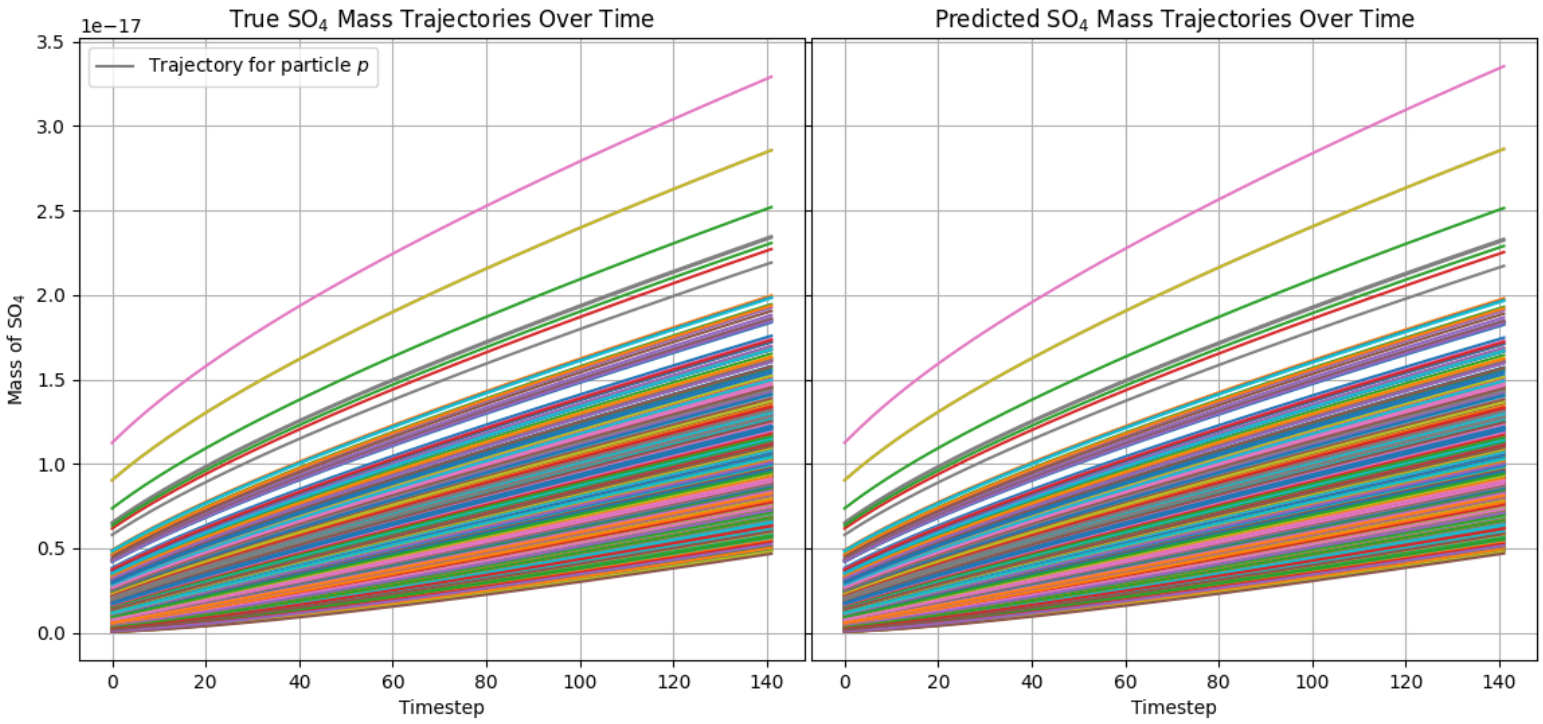


Figure: Results from Chem GNS on a simple particle-gas system. 142 timesteps, 808 particles, 3 time changing features, 3 time fixed particle properties, 2 chemical types. NMAE ≈ 0.007. Time to predict ≈ 0.4 seconds

Results from GNS Simulations: Sulfuric Acid

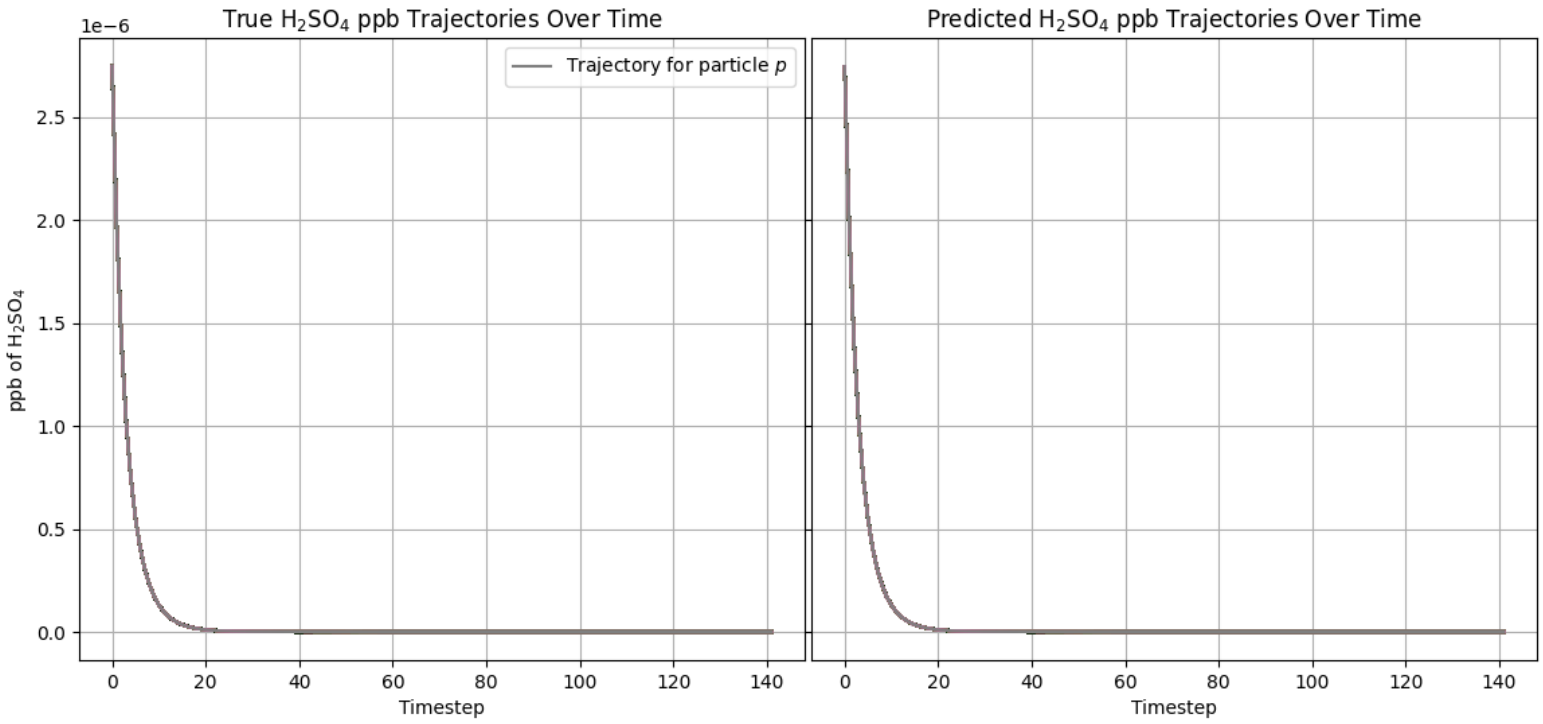


Figure: Results from Chem GNS on a simple particle-gas system. 142 timesteps, 808 particles, 3 time changing features, 3 time fixed particle properties, 2 chemical types. NMAE ≈ 0.011. Time to predict ≈ 0.4 seconds

Results from GNS Simulations: Dry diameter

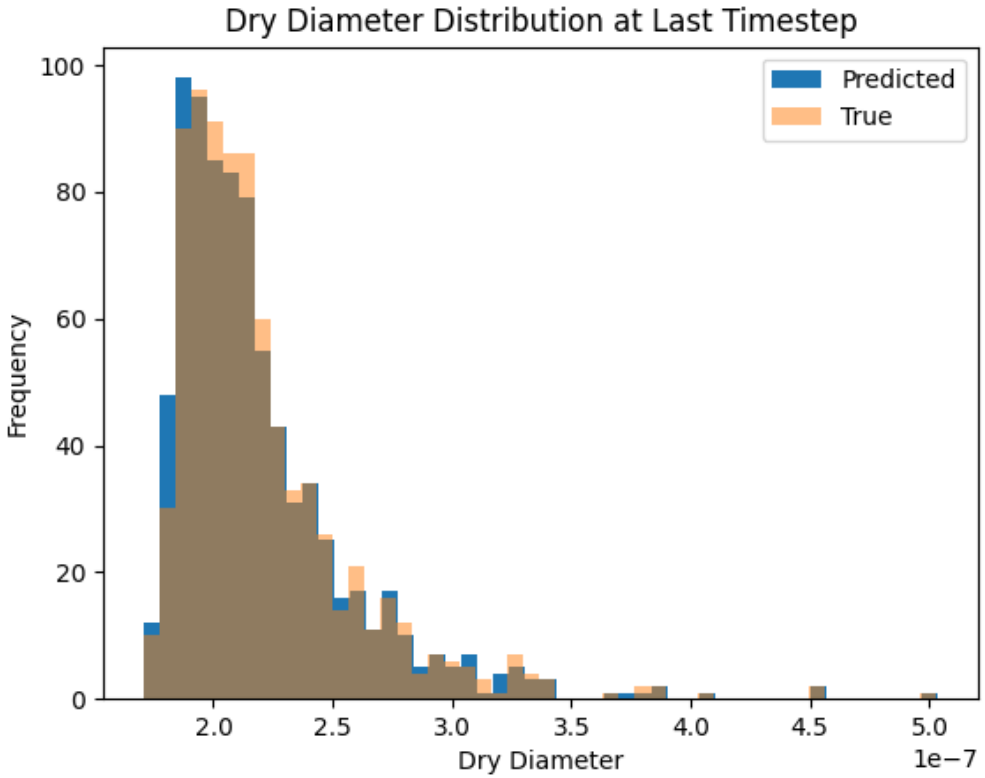


Figure: Results from Chem GNS on a simple particle-gas system. 142 timesteps, 808 particles, 3 time changing features, 3 time fixed particle properties, 2 chemical types. MSE $\approx 8.126 \times 10^{-7}$ in rate of change rate. Time to predict ≈ 0.4 seconds

Graph Neural Networks (Scarselli et al. 2008)

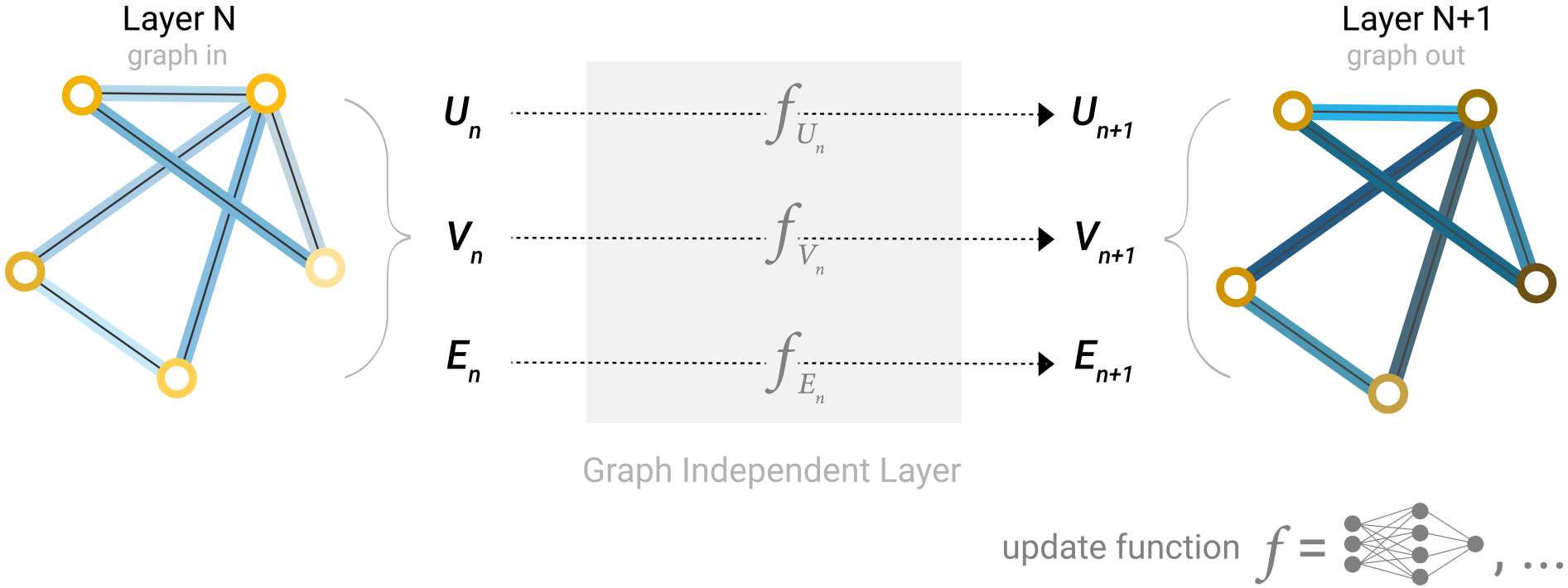
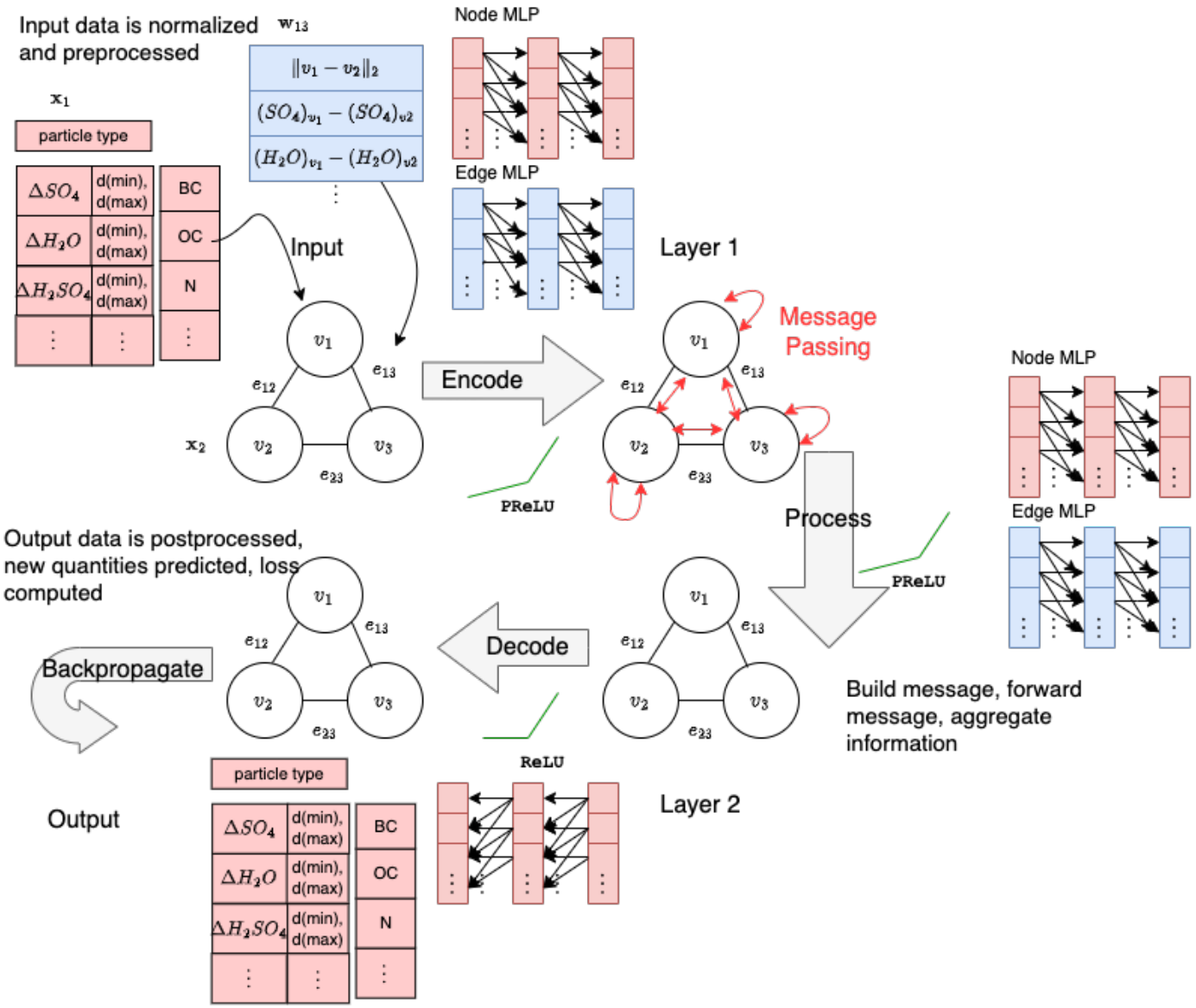


Figure: A single layer of a simple GNN. A graph is the input, and each component (V,E,U) gets updated by a MLP to produce a new graph. Each function subscript indicates a separate function for a different graph attribute at the n-th layer of a GNN model. *source*: `https://distill.pub/2021/gnn-intro/`

GNN Schematics



Chem GNS

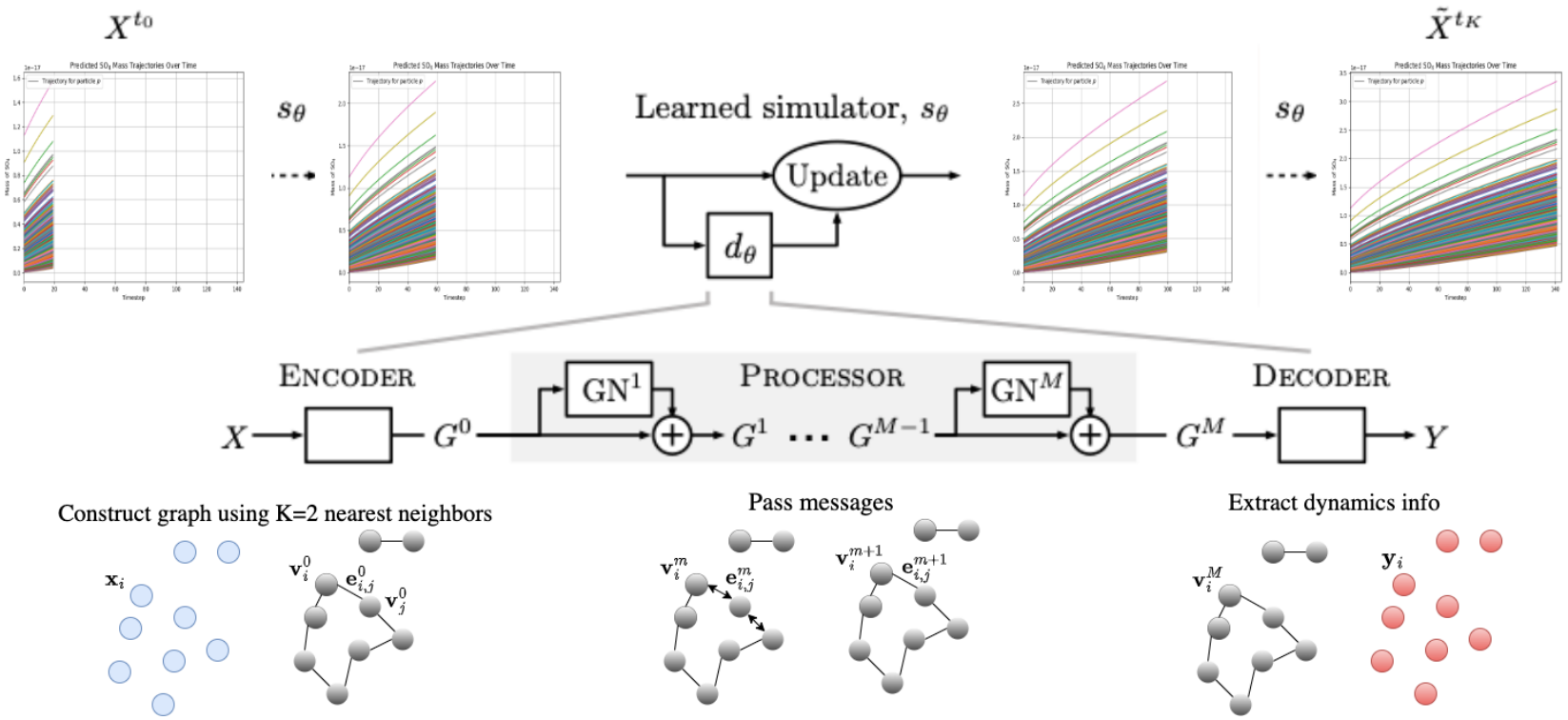


Figure: Uses an "encode-process-decode" scheme, which computes dynamics information, Y, from input state, X

Future Directions

- Improve accuracy in high dimension non-linear space - may require more careful selection of functions
- Particle-particle interaction
- Global nodes with environmental information

Conclusion

- Originally proposed for physics applications, GNS can work in chemical domain
- Making it bigger and better will require thinking outside Euclidean space
- Inclusion in climate models could be significant, if fast speeds can be maintained

References

- Kumar, Krishna and Joseph Vantassel (2022). "GNS: A generalizable Graph Neural Network-based simulator for particulate and fluid modeling". In: *arXiv preprint arXiv:2211.10228*.
- Scarselli, Franco et al. (2008). "The graph neural network model". In: *IEEE transactions on neural networks* 20.1, pp. 61–80.