Generative Modeling with Graph Neural Networks for the CMS HGCal

Sam Bein, Soham Bhattacharya, Engin Eren,
Frank Gaede, Gregor Kasieczka, William Korcari,
Dirk Krücker, Peter McKeown, **Moritz Scham**,
Moritz Wolf

HamGen

Round Table on Deep Learning, 12.10.2022

The DeGeSim Project

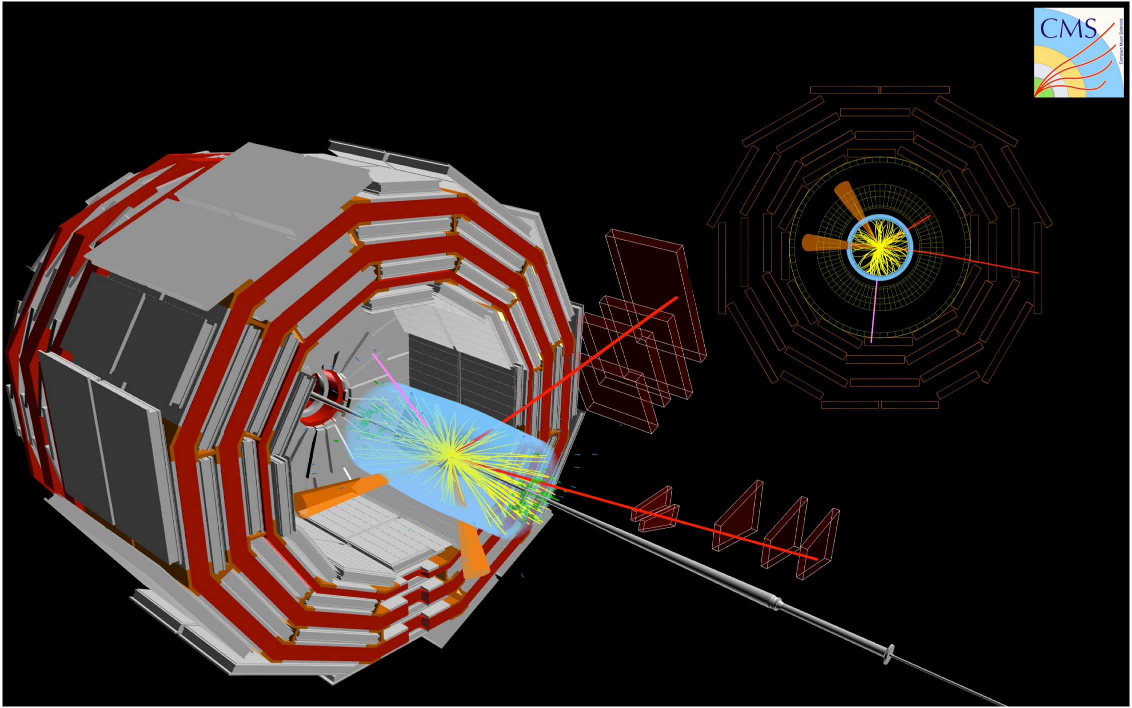
HELMHOLTZAI
ARTIFICIAL INTELLIGENCE
COOPERATION UNIT

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

RWTH AACHEN UNIVERSITY

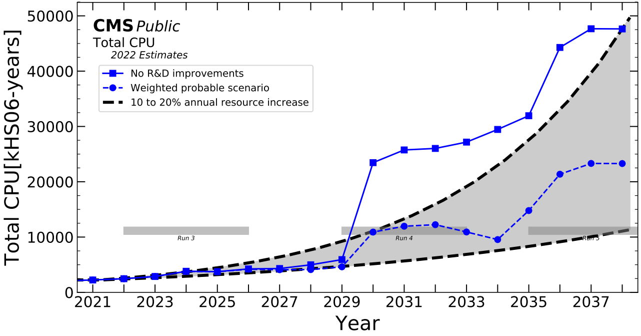
JÜLICH Forschungszentrum
JÜLICH SUPERCOMPUTING CENTRE

DESY.



Computing Challenge

- **>** High Luminosity phase
  - More particles to simulate
- **>** HGCAL – More cells and channels
  - Complex and time-consuming simulations



CPU time requirements [Link]

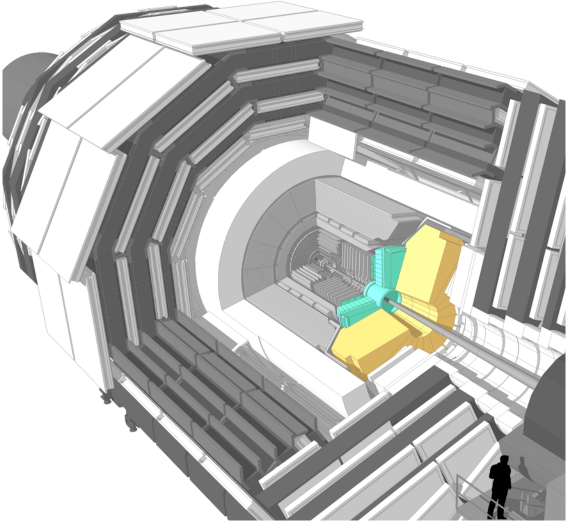
$\Rightarrow$ Increase in computing time beyond the expected increase in resources.

Simulating Particle Showers in the CMS HGCAL

- More cells & data
  - ⇒ More CPU hours needed for simulation
  - ⇒ Speedup with generative model?
- Challenges:
  - **Sparsity**
  - **Irregular** geometry
  - **Number** of channels

⇒ No ML model yet powerful enough

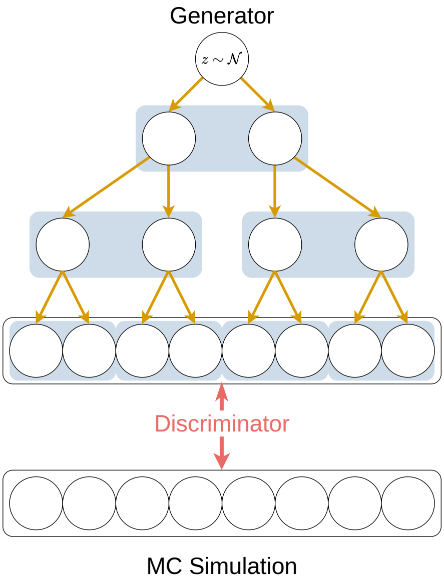
⇒ Data structures: point clouds and graphs



Tree Based Approach

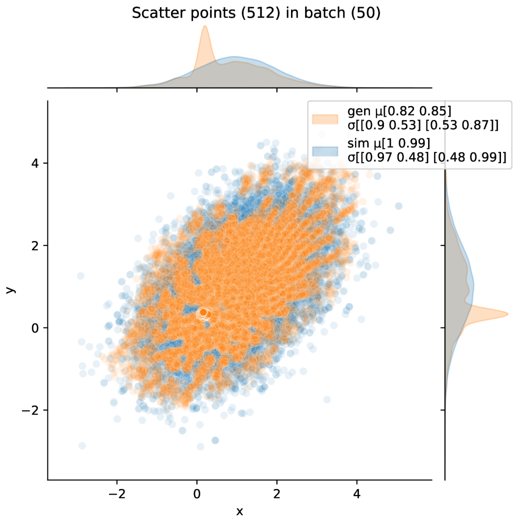
Previous work: TreeGAN

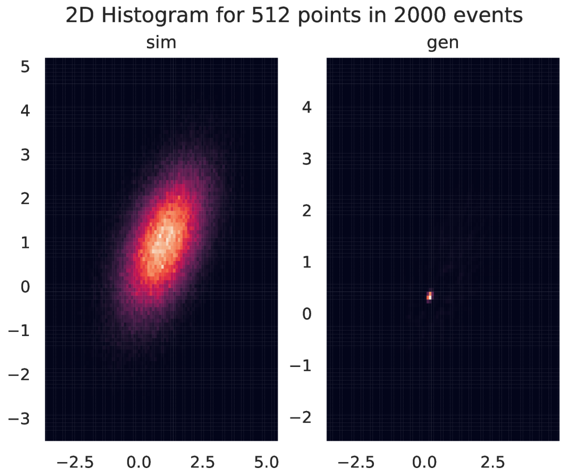
Shu et. arXiv:1905.06292



Generating Gaussian Distribution with TreeGAN

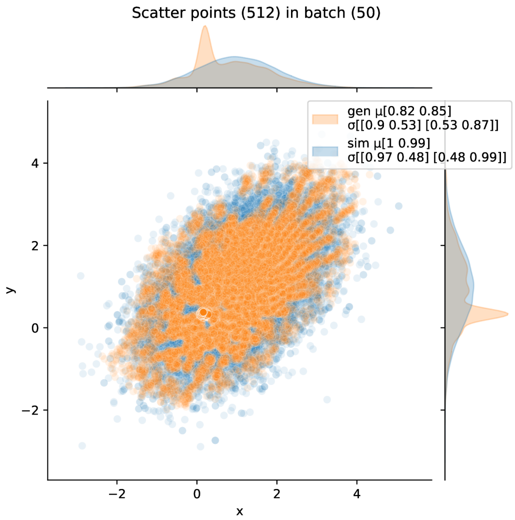
Results

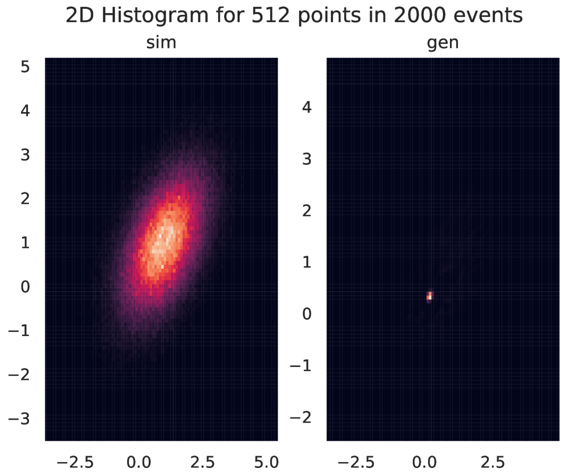




Generating Gaussian Distribution with TreeGAN

Results





⇒ TreeGan can model the shape of the distribution, but not the density

DeepTreeGAN

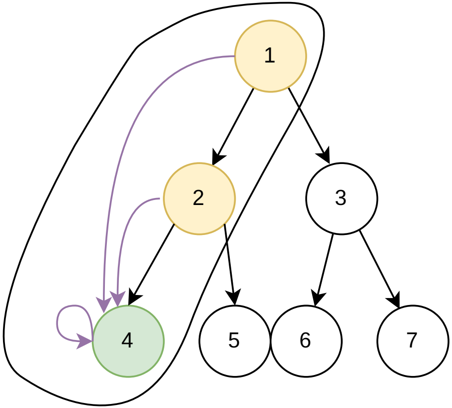
Own Work

- > TreeGAN needs significant improvements
- > Idea: Use FFNs instead of matrices, formulate as Message Passing network

Message Passing

For DeepTree

Update Node 4
(w/ Neighbors, Messages)



Used here: GINConv `arXiv`:1810.00826

1 Message:

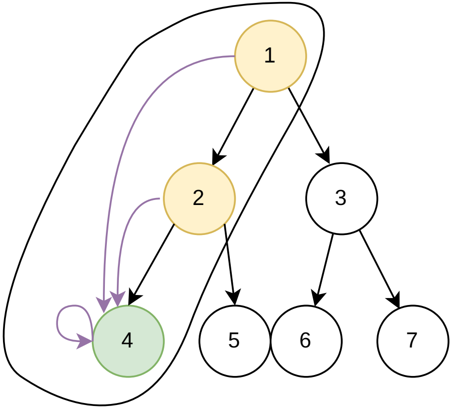
$$\mathbf{Msg}_{j \to i} = \mathbf{x}_j$$

$$\mathbf{x}_i' = \mathrm{NN}\left((1+\epsilon) \cdot \mathbf{x}_i + \sum_{j \in \mathcal{N}(i)} \mathbf{x}_j\right)$$

Message Passing

For DeepTree

Update Node 4
(w/ Neighbors, Messages)



Used here: GINConv `arXiv:1810.00826`

1 Message:

$$\mathbf{Msg}_{j \to i} = \mathbf{x}_j$$

2 Aggregate:

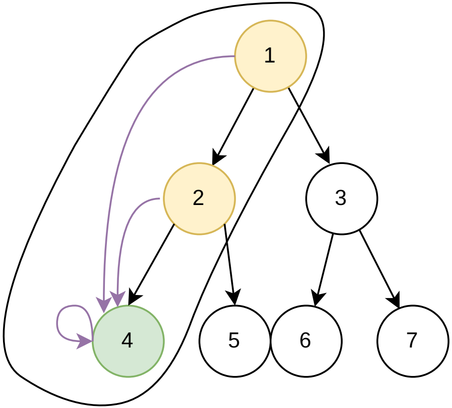
$$\mathbf{Aggr}_i = \sum_{j \in \mathcal{N}(i)} \mathbf{Msg}_{j \to i}$$

$$\mathbf{x}'_i = \mathrm{NN}\left((1+\epsilon) \cdot \mathbf{x}_i + \sum_{j \in \mathcal{N}(i)} \mathbf{x}_j\right)$$

Message Passing

For DeepTree

Update Node 4
(w/ Neighbors, Messages)



Used here: GINConv arXiv:1810.00826

1 Message:

$$\mathbf{Msg}_{j\to i} = \mathbf{x}_j$$

2 Aggregate:

$$\mathbf{Aggr}_i = \sum_{j\in\mathcal{N}(i)} \mathbf{Msg}_{j\to i}$$

3 Update:

$$\mathbf{x}_i \leftarrow \mathrm{NN}\left((1+\epsilon)\mathbf{x}_i + \mathbf{Aggr}_i\right)$$

$$\mathbf{x}_i' = \mathrm{NN}\left((1+\epsilon)\cdot\mathbf{x}_i + \textstyle\sum_{j\in\mathcal{N}(i)} \mathbf{x}_j\right)$$

Smearing

- Generator: continuous spectrum *vs.*
- Dataset: discrete values for $x, y, \texttt{layer}$

⇒ Discriminator can distinguish the samples very easily

⇒ Generator can't learn

Smearing

- Generator: continuous spectrum *vs.*
- Dataset: discrete values for $x, y, \texttt{layer}$

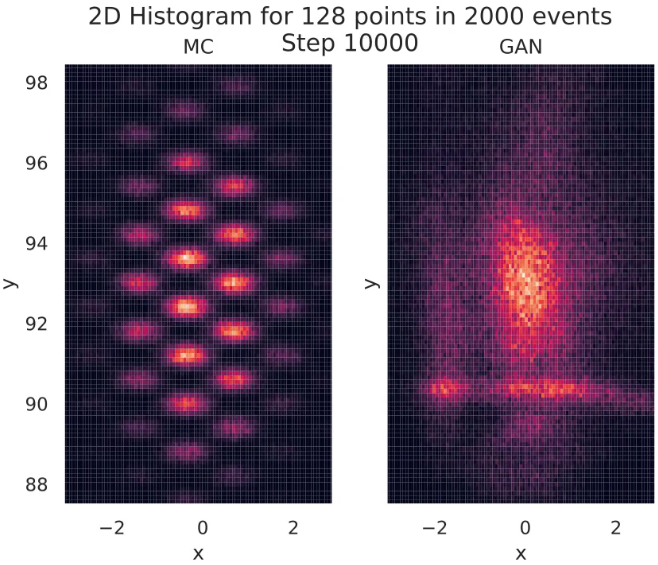
⇒ Discriminator can distinguish the samples very easily

⇒ Generator can't learn

- Idea:
  - Smear out the distributions ⇒ easier task for the generator
  - Gradually turned off the smearing

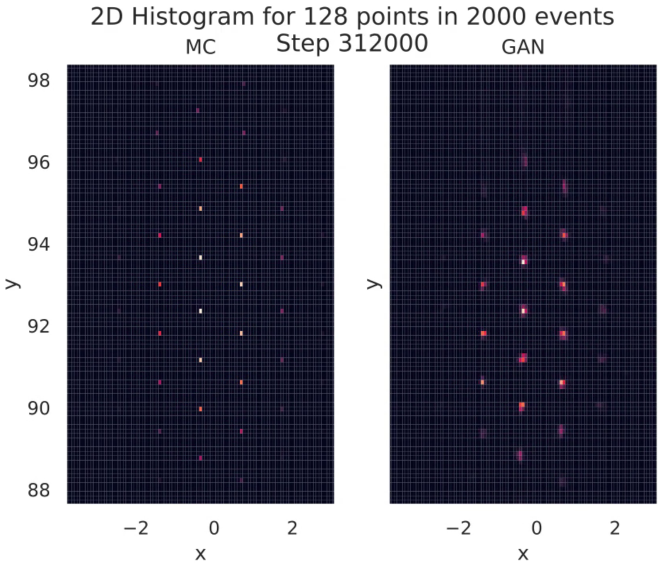
Development during the Training

X vs Y



Development during the Training

X vs Y



Experience on Maxwell

- Work mostly done on Maxwell
- Good user experience
  - Easy access (eg. No AFS tokens)
  - Workers with internet connection (can use web services e.g. CometML)
  - Stable and reliable
- Hyperparameter tuning with ray [Link]

Experience on Maxwell

- Work mostly done on Maxwell
- Good user experience
  - Easy access (eg. No AFS tokens)
  - Workers with internet connection (can use web services e.g. CometML)
  - Stable and reliable
- Hyperparameter tuning with ray [Link]
  $\Rightarrow$ Please don't use it, I need the slots

Experience on Maxwell

- Work mostly done on Maxwell
- Good user experience
  - Easy access (eg. No AFS tokens)
  - Workers with internet connection (can use web services e.g. CometML)
  - Stable and reliable
- Hyperparameter tuning with ray [Link]
  $\Rightarrow$ Please don't use it, I need the slots
- Jobs need to be able to catch being preempted:

```
class SigTermHandel:
    def __init__(self, ...):
        signal.signal(signal.SIGTERM, self.handle)
    def handle(self, _signo, _stack_frame):
        model.save_checkpoint()
        exit()
```

Graph Growing Approach

Graph Growing

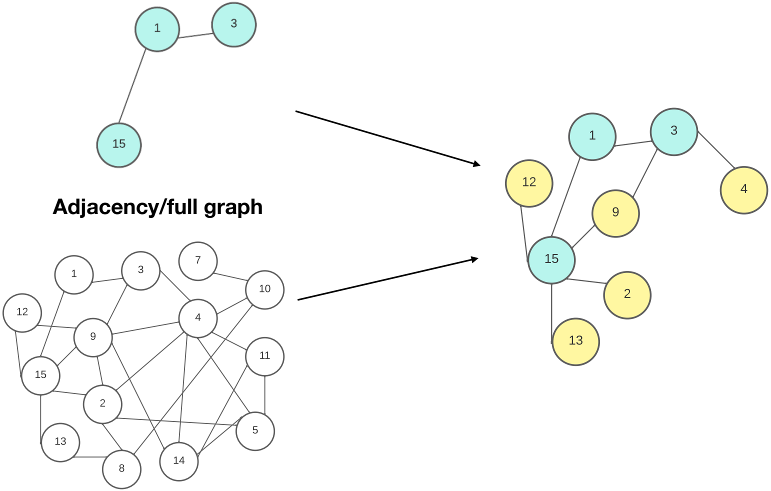
Work by William Korcari

> Idea:
  - Use existing information about the geometry
  - Grow the graph iteratively

Generating Sequence

Step 1

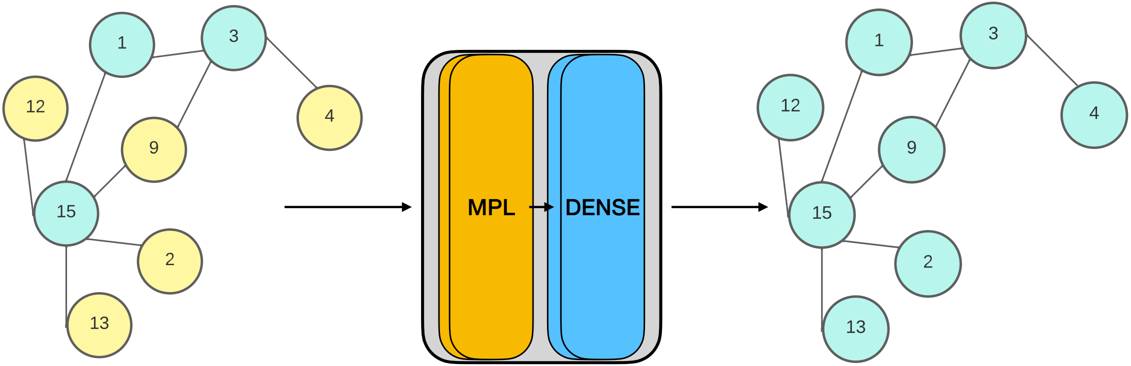
> Add nodes adjacent to the ones currently in the graph



Generating Sequence

Step 2

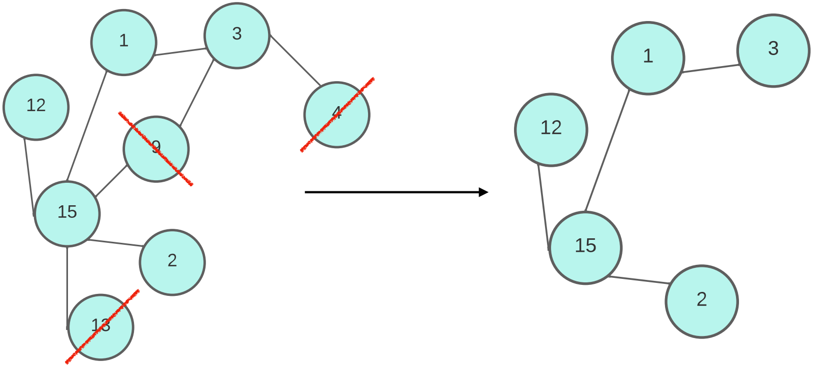
> Update the nodes with a small GNN



Generating Sequence

Step 3

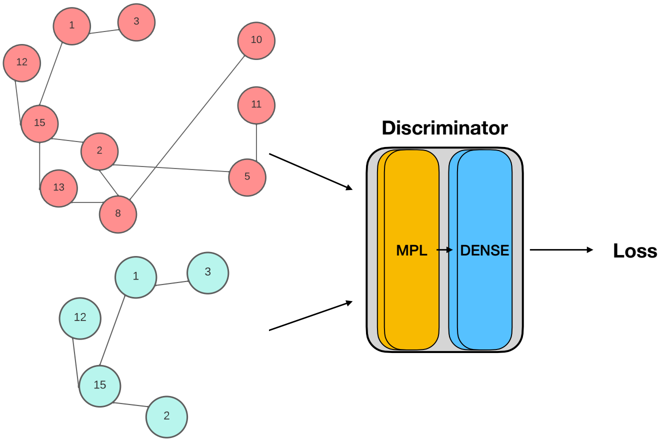
> Remove nodes with feature under threshold



⇒ Repeat the steps for n times

Discriminator

> Feed generated graph and real graph into the discriminator.



Thank you!

DESY. Deutsches
Elektronen-Synchrotron
www.desy.de

Sam Bein, Soham Bhattacharya, Engin Eren,
Frank Gaede, Gregor Kasieczka, William Korcari,
Dirk Krücker, Peter McKeown, **Moritz Scham**,
Moritz Wolf
moritz.scham@desy.de

Backup

Smearing w/ Cos Turnoff

Smearing

```
def smooth_features(x, step):
    return x + np.random.multivariate_normal(
        [0.0, 0.0, 0.0, 0.0],
        np.diag(smoothing_vars) * turnoff(step, rate),
    )
```

Turnoff

```
def turnoff(step, rate):
    step *= rate
    if step > np.pi:
        return 0
    else:
        return (np.cos(step) + 1) / 2
```

- Standard deviations of the Gaussian noise
  - E: 0
  - x: 0.315 cm
  - y: 0.18 cm (needs to be smaller than for x, because more values in the marginal)
  - layer: .3
- Turnoff rate: 1e-6

Density-aware Chamfer Distance

as a Comprehensive Metric for Point Cloud Completion

arXiv:2111.12702v1

$$d_{DCD}(S_1, S_2) = \frac{1}{2}\left(\frac{1}{|S_1|}\sum_{x \in S_1}\left(1 - \frac{1}{n_{\hat{y}}}e^{-\alpha||x-\hat{y}||_2}\right) + \frac{1}{|S_2|}\sum_{y \in S_2}\left(1 - \frac{1}{n_{\hat{x}}}e^{-\alpha||y-\hat{x}||_2}\right)\right)$$

MPGAN Disc I

Message Passing layers for MPGAN

```
A = torch.cat((
    x.repeat(1, 1, num_nodes).view(
        batch_size, num_nodes * num_nodes,
         ↪  node_size
        ),
    x.repeat(1, num_nodes, 1),
    2).view(batch_size * num_nodes * num_nodes,
    ↪  out_size)
# upscaling
A = self.fe(A).view(batch_size, num_nodes,
 ↪  num_nodes, self.fe_layers[-1])
A = torch.sum(A, 2) if self.sum else
 ↪  torch.mean(A, 2)
x = torch.cat((A, x), 2).view(batch_size *
 ↪  num_nodes, -1)
# downscaling
x = self.fn(x).view(batch_size, num_nodes,
 ↪  self.output_node_size)
```

$$\mathtt{x} = \begin{pmatrix} \vec{a} \\ \vec{b} \end{pmatrix} \qquad A = \begin{bmatrix} \vec{a} & \vec{a} \\ \vec{a} & \vec{b} \\ \vec{b} & \vec{a} \\ \vec{b} & \vec{b} \end{bmatrix}$$

$$\mathtt{ffn(A).view(...)} = \begin{bmatrix} \begin{bmatrix} ffn(\vec{a},\vec{a}) \\ ffn(\vec{a},\vec{b}) \end{bmatrix} \\ \begin{bmatrix} ffn(\vec{b},\vec{a}) \\ ffn(\vec{b},\vec{b}) \end{bmatrix} \end{bmatrix}$$

$$\mathtt{A.sum(2)} = \begin{bmatrix} ffn(\vec{a},\vec{a}) + ffn(\vec{a},\vec{b}) \\ ffn(\vec{b},\vec{a}) + ffn(\vec{b},\vec{b}) \end{bmatrix}$$

MPGAN Disc II

- 2 MP layers
- Masked sum particles
- Feed-Forward-Network
- Final activation

Hyperparameter Tuning on Maxwell

> Tuning with ray works nicely