Introduction to Graph Neural Networks

朱 彤

04/19/2023

Outline

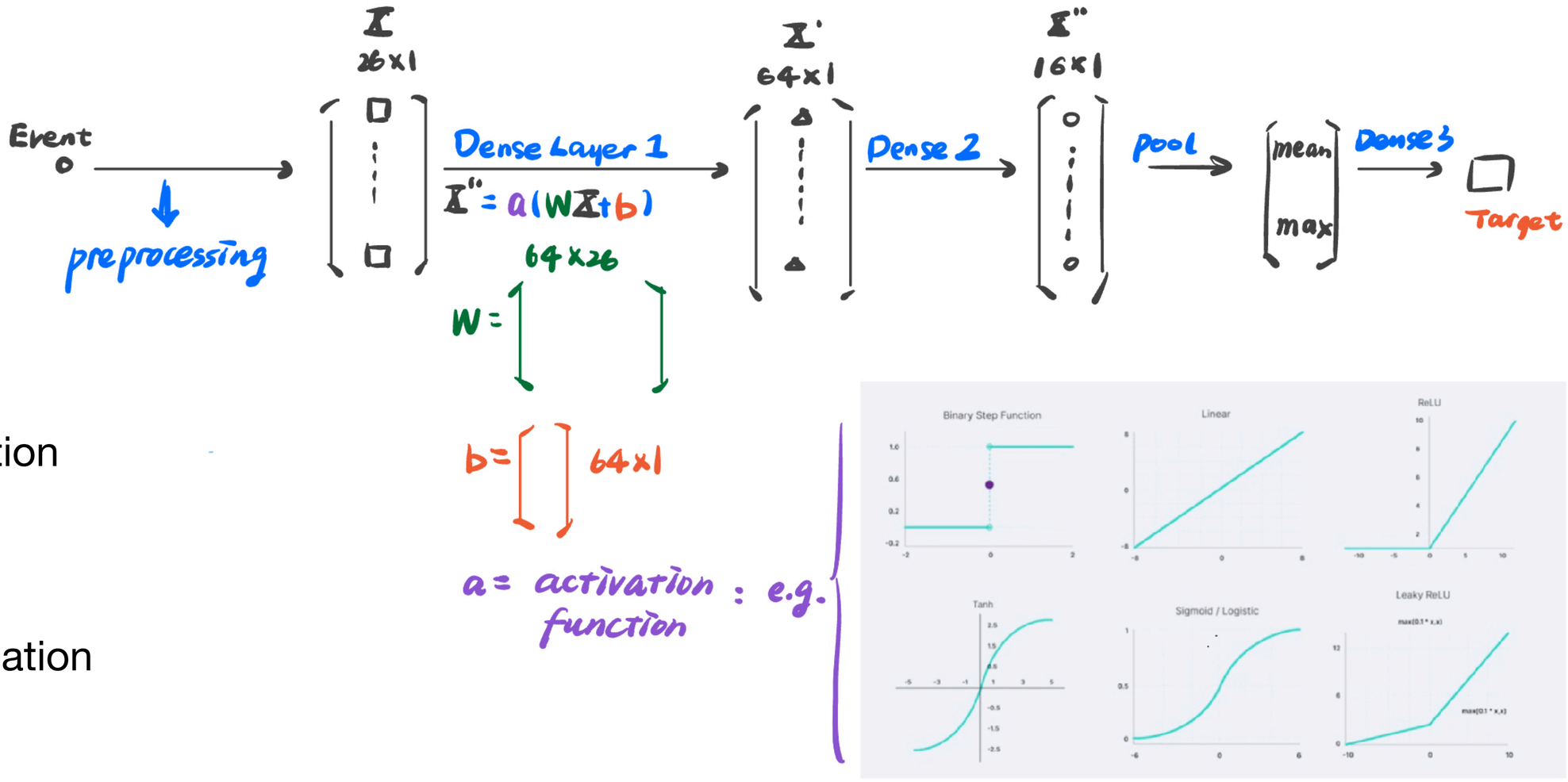
- **Warm-up:** Glossary of Machine Learning
- **Convolutional Neural Networks (CNNs):** A classic way to deal with pictures.
- **Graph Fundamentals:** Graphs and Operations on Graphs.
- **Graph Neural Networks (GNNs):** Categorization, Applications, Pros, and Cons.
- **Example 1:** EP-Separation@DAMPE (Binary Classification)
- **Example 2:** Energy and Angular Reconstruction@ IceCube (Regression)
- **Hands-on:** Work with TFGNN

Glossary of Machine Learning

- Model
- Dense Layers
- Pool Layers
- Activation Function
- Metrics
- Forward Propagation

Forward Propagation

Task: Regression(e.g. Energy Reconstruction)

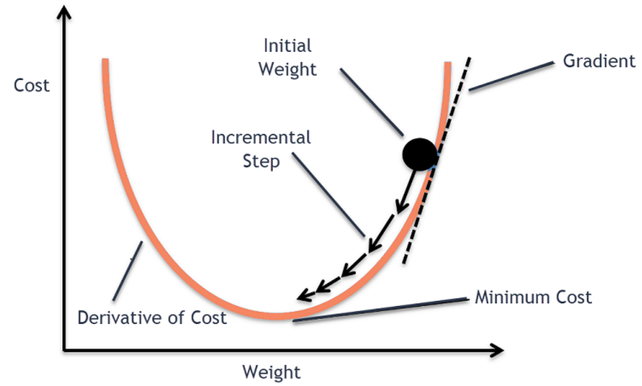


e.g. A Toy NN Model
Number of Parameter = 64*26+64+16*64+16+1*2+1 = 2771

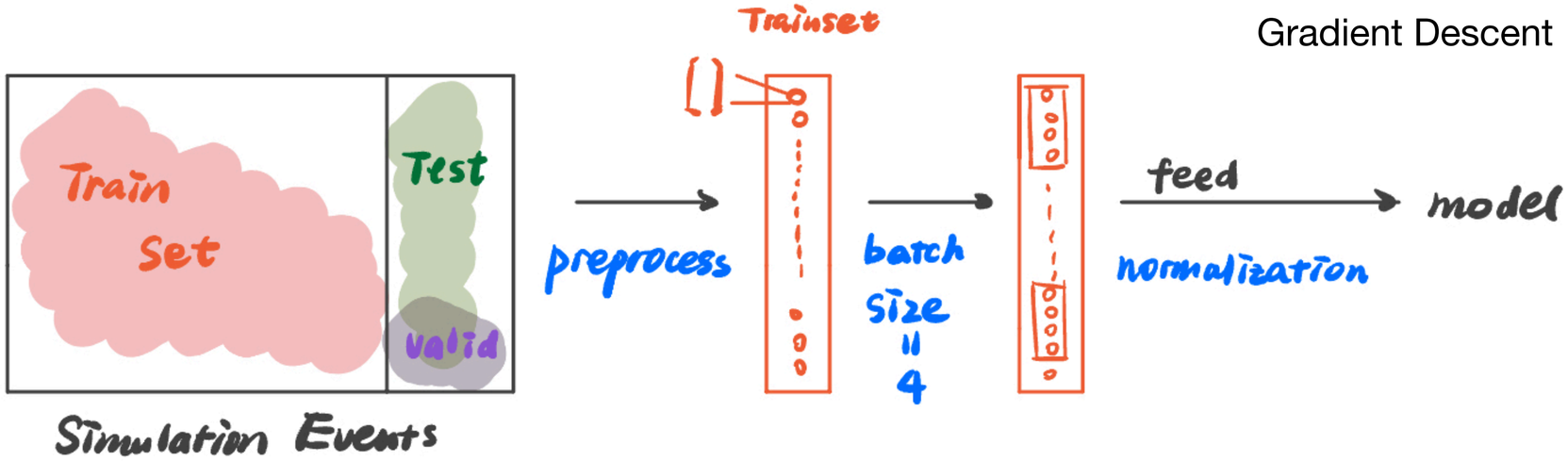
Glossary of Machine Learning

Data Pipeline & Training

- Train/Test/Valid Set
- Batch
- Normalization
- Loss Function
- Optimizer
- Epoch
- Overfit



Gradient Descent

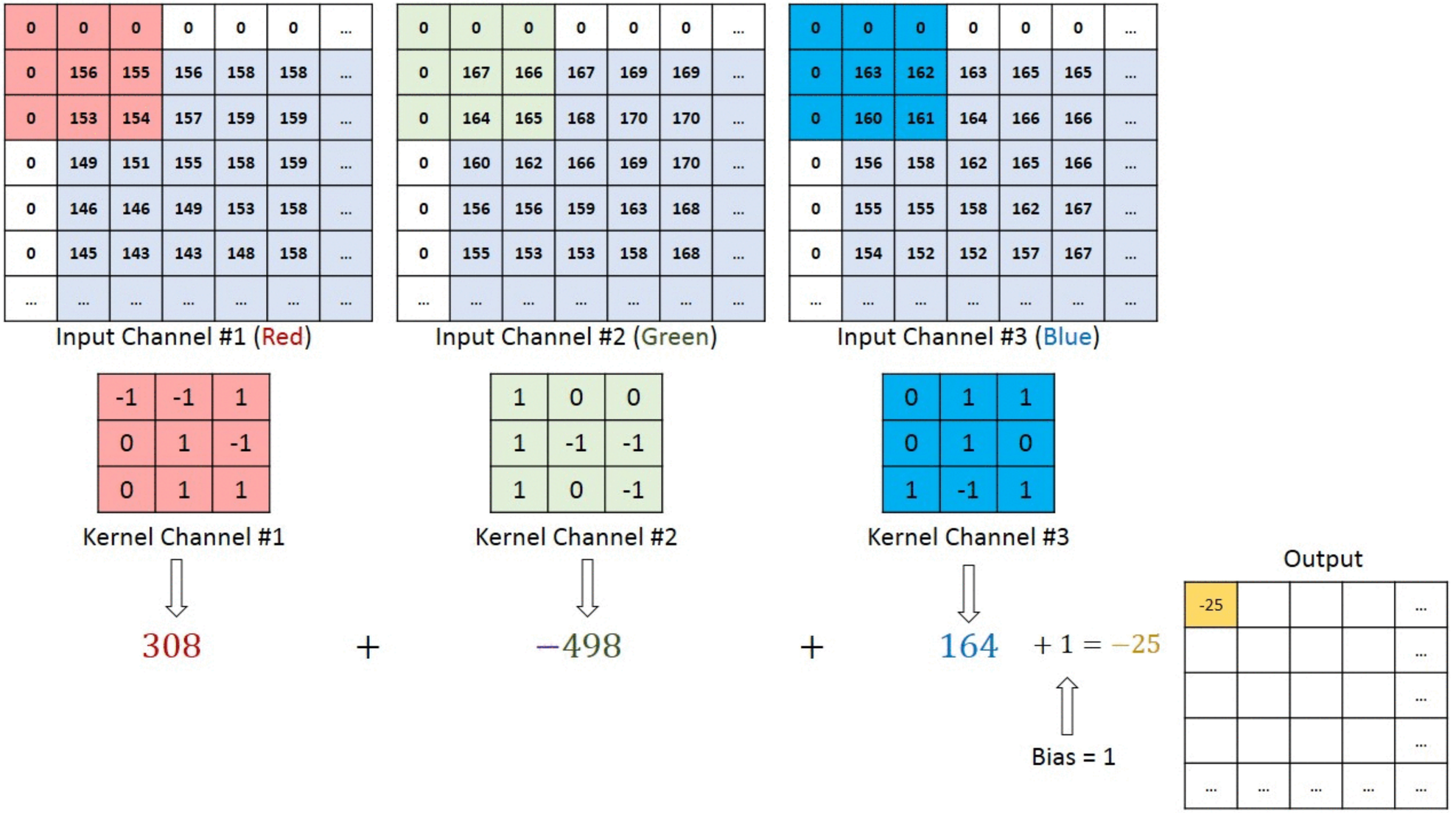


Categorization

Supervised/ Unsupervised/ Semi-supervised Learning

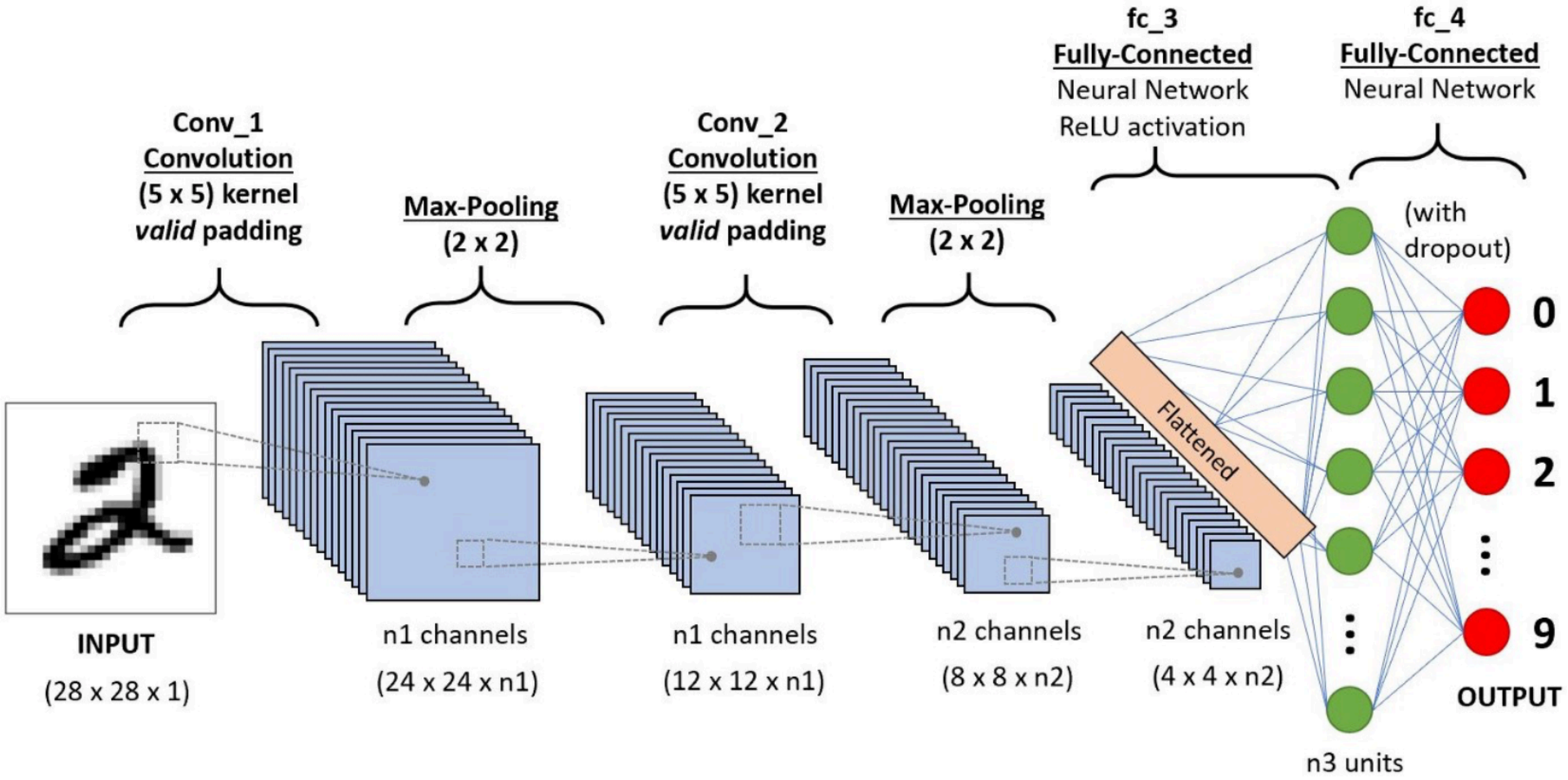
Convolutional Neural Networks

Convolution

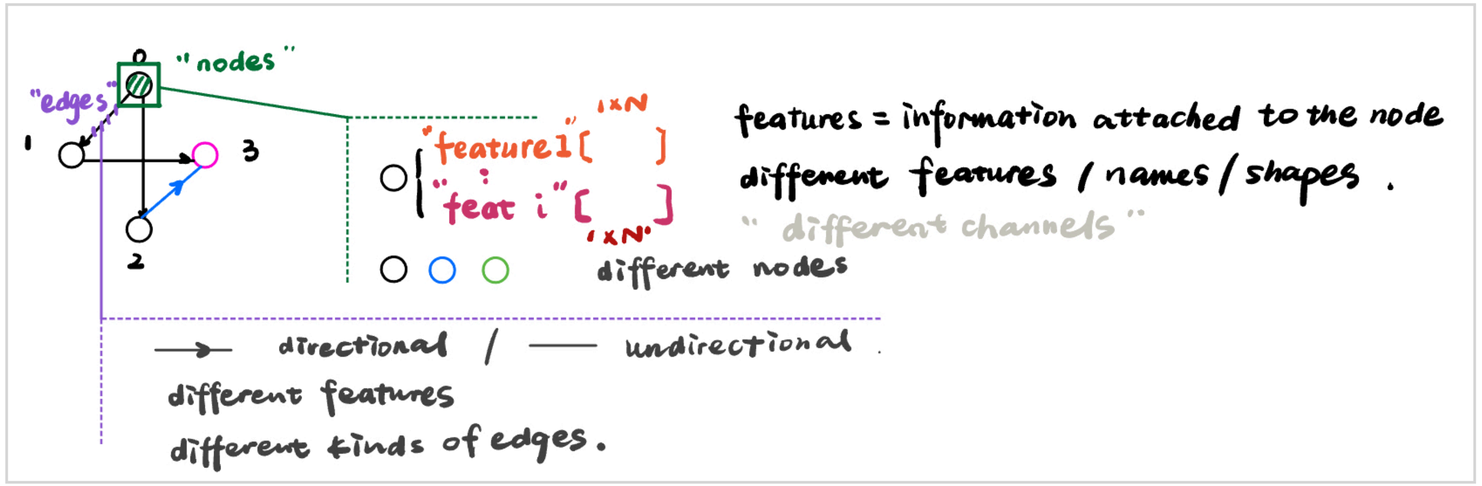


Convolutional Neural Networks

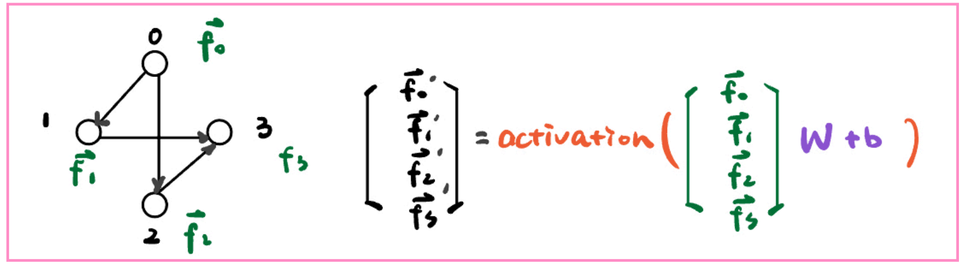
Model



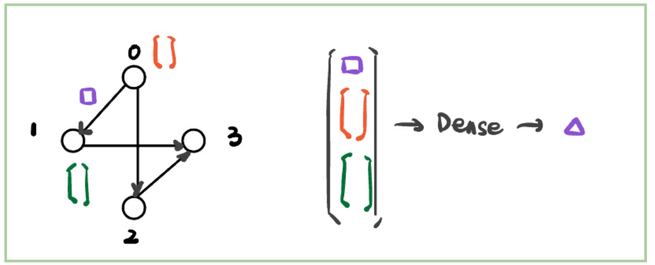
Graph Fundamentals



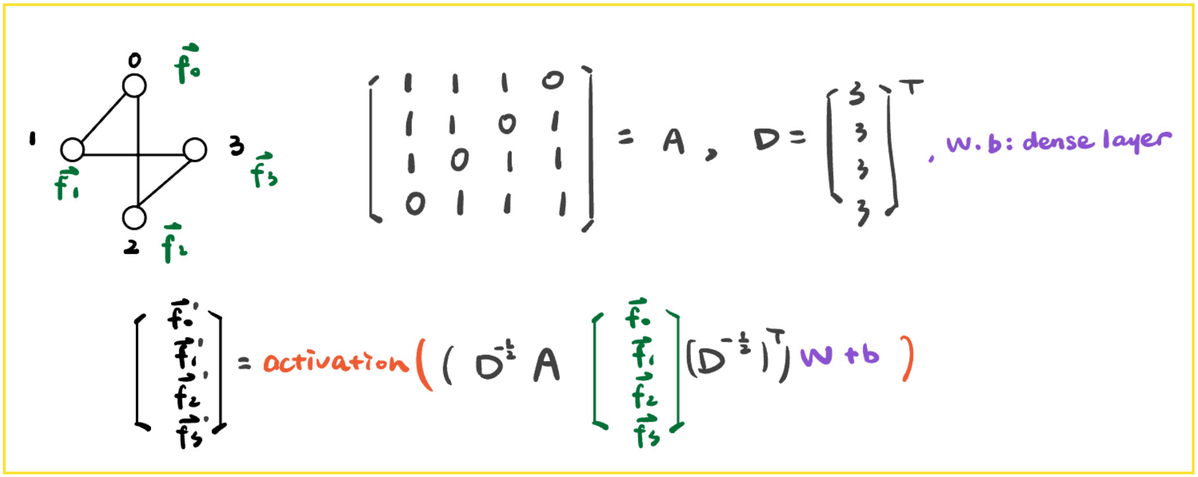
Graph



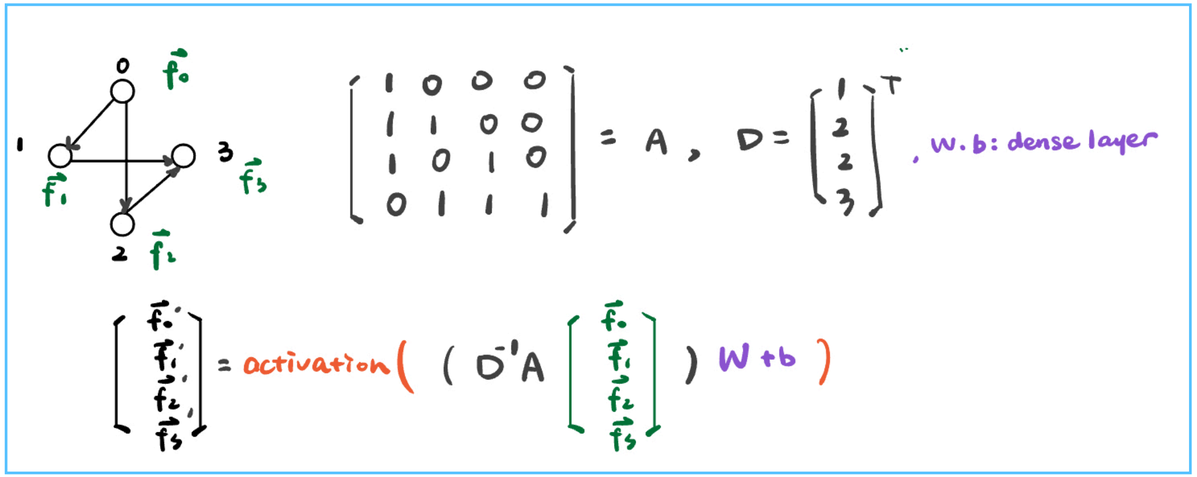
Dense



Attention Calculation
GAT: 1710.10903

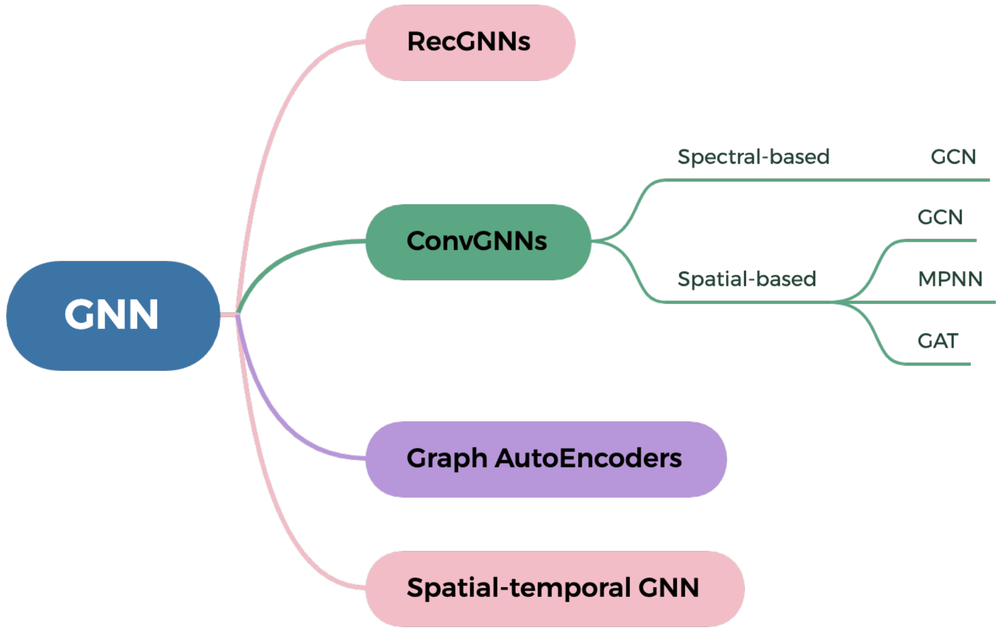


Graph Convolution
GCN:1609.02907



Message Passing
MPNN: 1704.01212

Graph Neural Networks: Overview



Challenges:

1. Scalability trade-off: GNNs tend to have high computational costs, which makes them challenging to apply to large graphs.

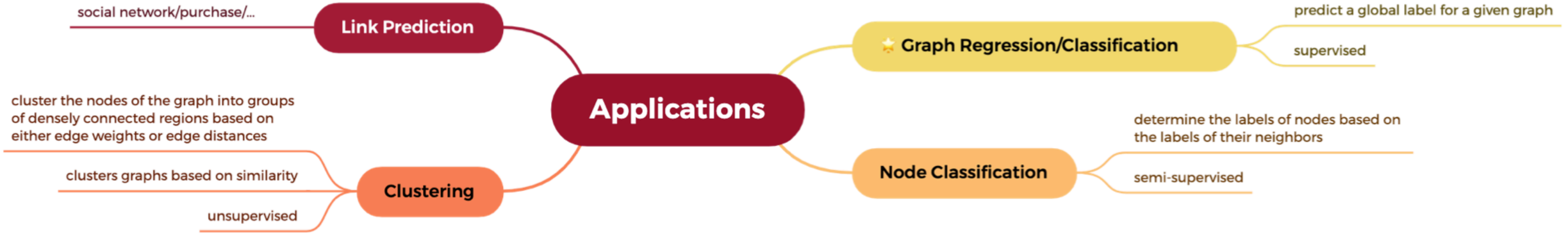
2. Model depth: Deep architectures may not work for ConvGNNs owing to over-smoothing. (1801.07606)

3. Interpretability: GNNs can be difficult to interpret, which can make it challenging to understand how they are making decisions.

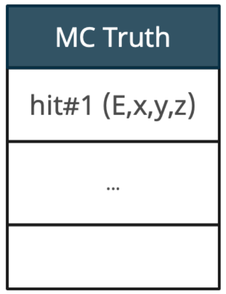
4. Heterogeneity: The majority of current GNNs assume homogeneous graphs. It is difficult to directly apply current GNNs to heterogeneous graphs.

Future Directions:

Graph interpreter, GNN Architecture Design..



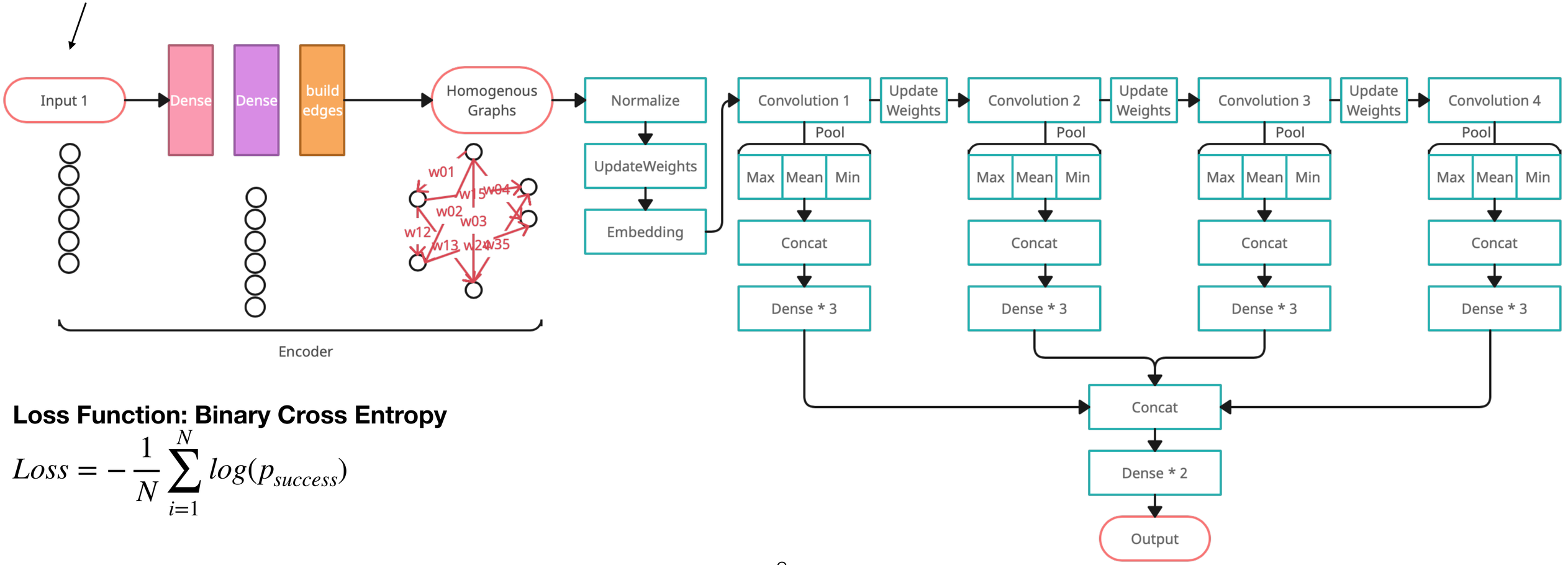
Example 1: EP Separation



Task: Binary Classification

Trainset~18k
Testset~30k
p/e ~ O(1)
A100, 80GB, ~9min, 20 epochs

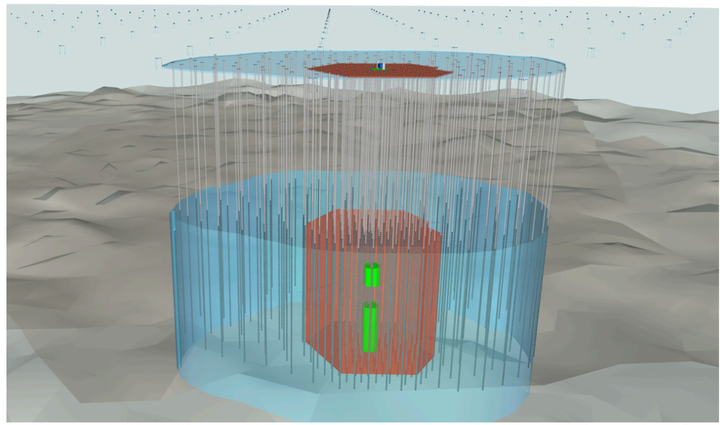
Signals from detectors can be well represented by graphs but in many cases, we don't have natural edges.
When the graph sizes are not too large~O(100), we can use this network to dynamically generate homogenous graphs for later training.

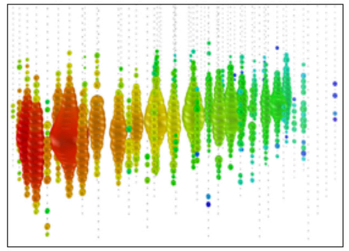


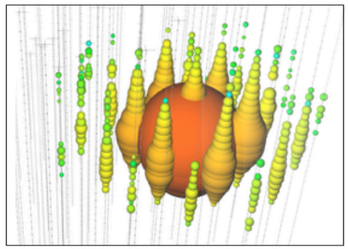
Loss Function: Binary Cross Entropy

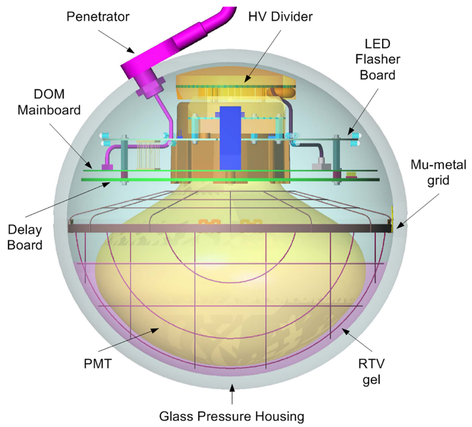
$$Loss = -\frac{1}{N}\sum_{i=1}^{N} log(p_{success})$$

Example 2: Energy/Angular Reconstruction @ IceCube









Task: Reconstruction (1D/3D Regression)

Why GNN?

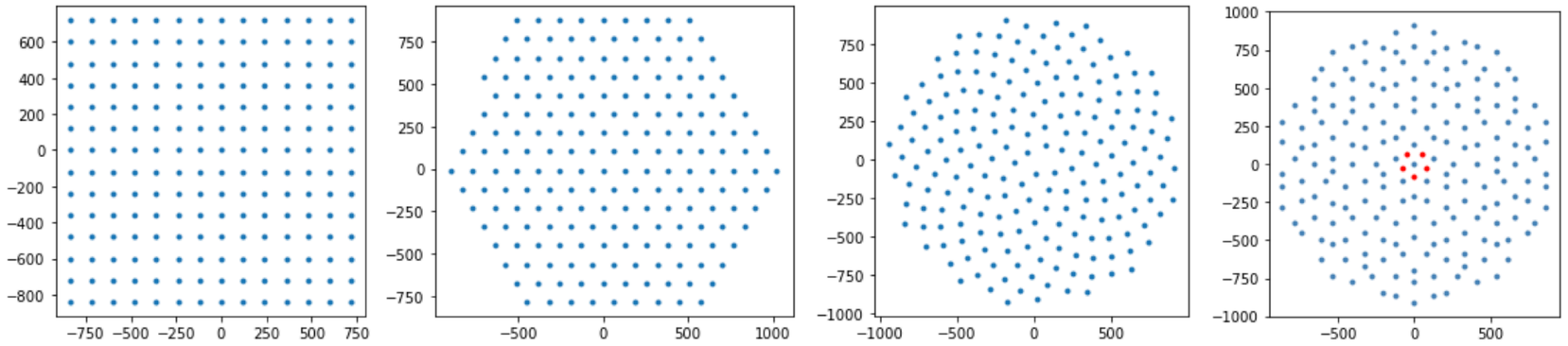
- Irregular input can be well represented by graphs.
- GNN is portable in different geometries.
- Confronted directly with raw photons, we expect that GNN can extract hidden information.
- Less parameters for potential FPGA application.

Candidate Geometries

16 candidates in 4 different shapes and 4 different geometric areas.
They have the same number of Digital Optical Modules (196 strings* 80).

IC Geometric Areas(km^2) ~ 1

4 Geometric Areas(km^2): 2.4, 5.5, 9.7, 15.2

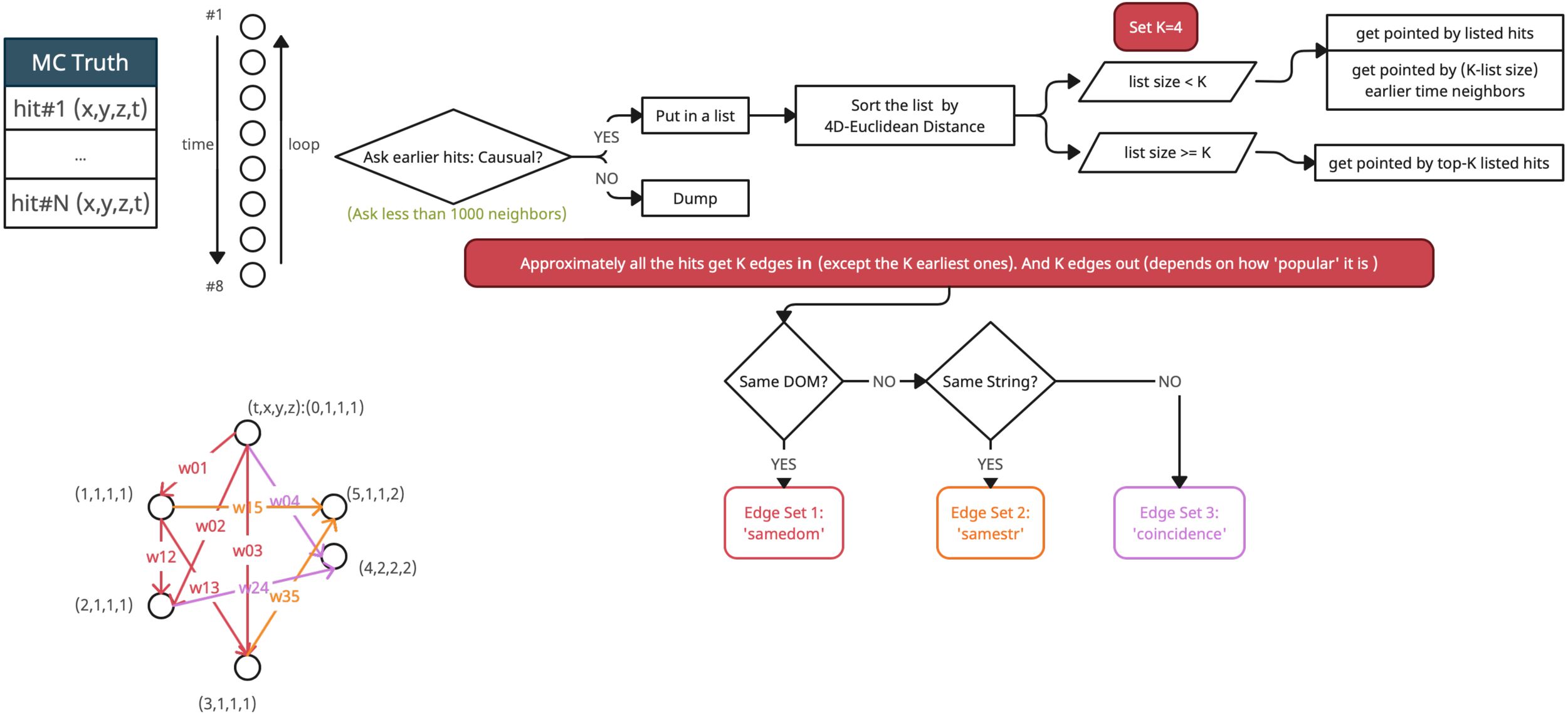


Orthogonal Hexagonal Sunflower Penrose

These are XY projections of candidate arrays. Each dot here indicates a string with 80 Digital Optical Modules (DOMs).

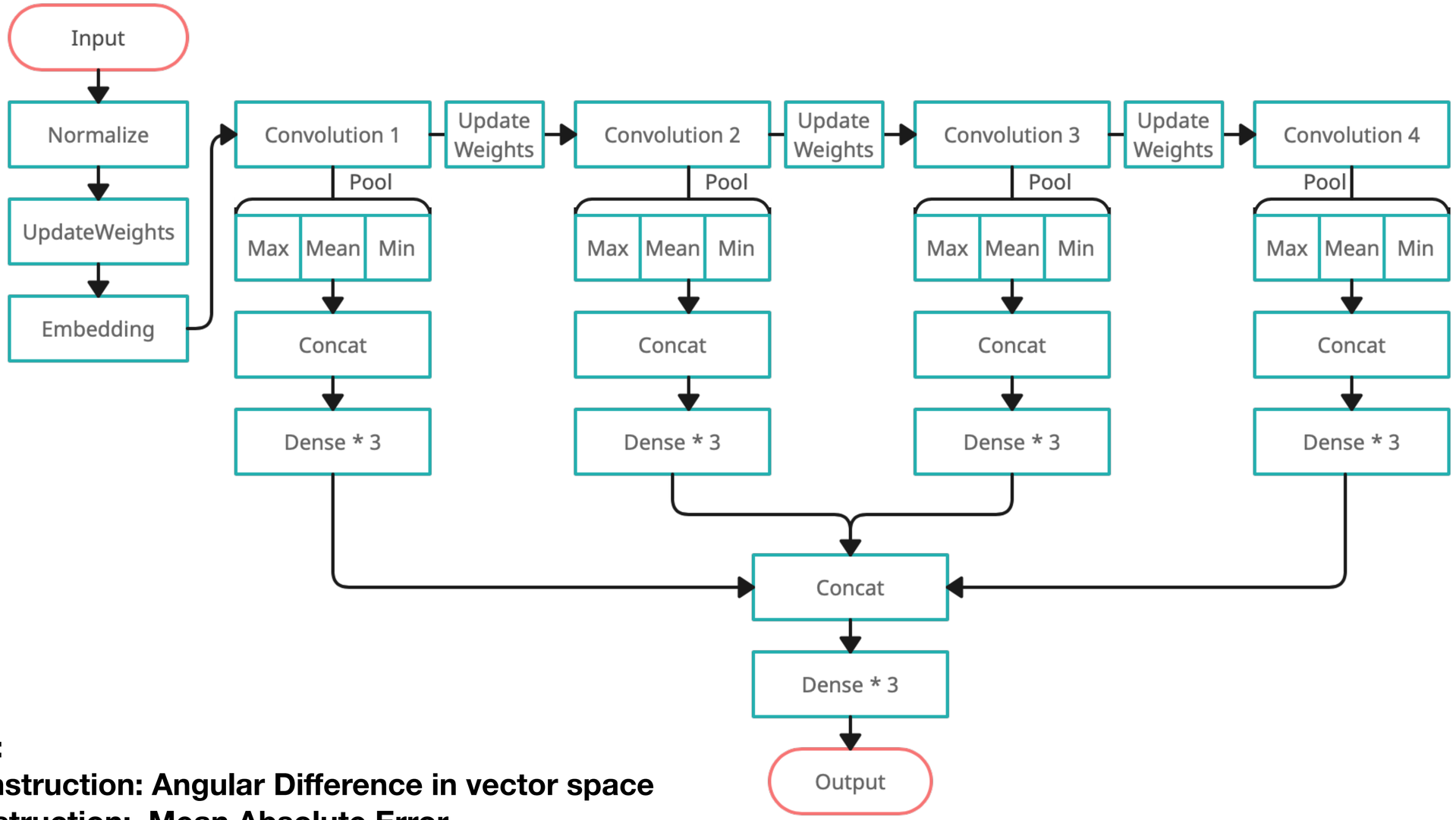
Example 2: Energy/Angular Reconstruction @ IceCube

Data Preprocessing



Example 2: Energy/Angular Reconstruction @ IceCube

Model

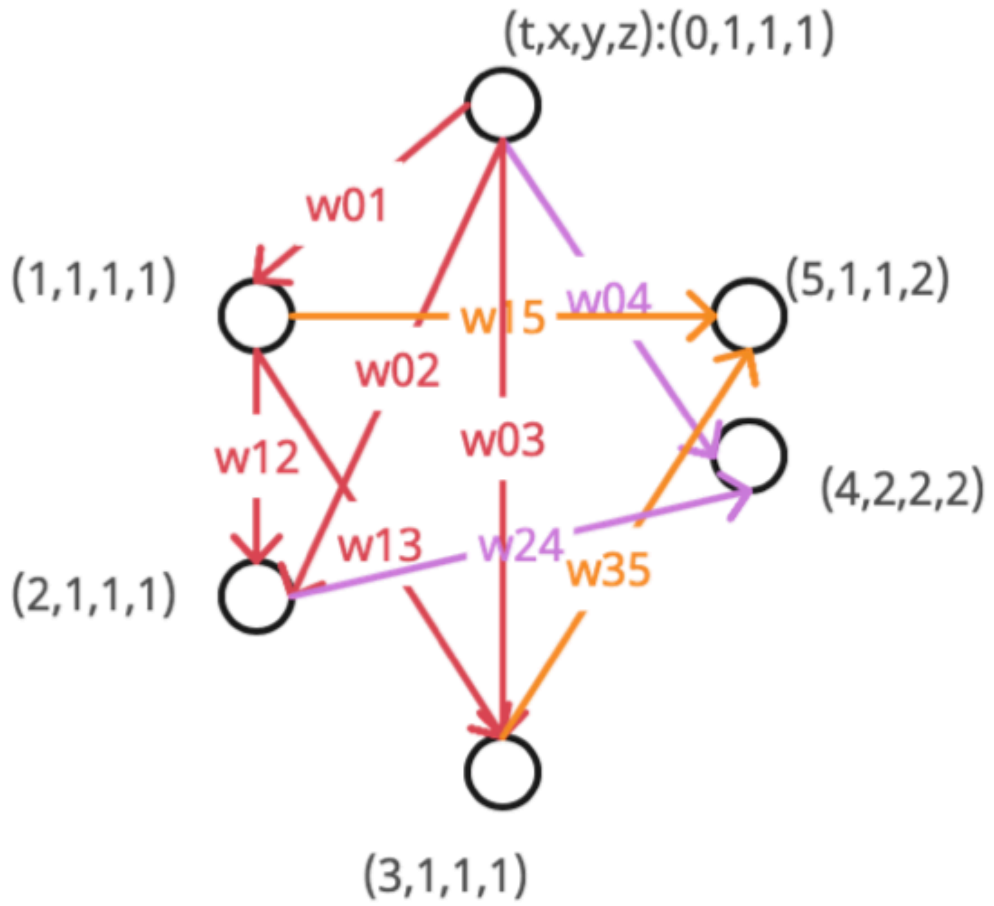


Loss Function:
Angular Reconstruction: Angular Difference in vector space
Energy Reconstruction: Mean Absolute Error

Example 2: Energy/Angular Reconstruction @ IceCube

Convolution on Heterogenous Graphs

$$\begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 2 & 1 & 1 & 1 \\ 3 & 1 & 1 & 1 \\ 4 & 2 & 2 & 2 \\ 5 & 1 & 1 & 2 \end{pmatrix}$$



$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ w_{01} & 1 & 0 & 0 & 0 & 0 \\ w_{02} & w_{12} & 1 & 0 & 0 & 0 \\ w_{03} & w_{13} & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 2 & 1 & 1 & 1 \\ 3 & 1 & 1 & 1 \\ 4 & 2 & 2 & 2 \\ 5 & 1 & 1 & 2 \end{pmatrix} \rightarrow \begin{pmatrix} 0 & 1 & 1 & 1 \\ \frac{1}{w_{01}+1} & 1 & 1 & 1 \\ \frac{w_{12}+2}{w_{02}+w_{12}+1} & 1 & 1 & 1 \\ \frac{w_{13}+3}{w_{03}+w_{13}+1} & 1 & 1 & 1 \\ 4 & 2 & 2 & 2 \\ 5 & 1 & 1 & 2 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & w_{15} & 0 & w_{35} & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 2 & 1 & 1 & 1 \\ 3 & 1 & 1 & 1 \\ 4 & 2 & 2 & 2 \\ 5 & 1 & 1 & 2 \end{pmatrix} \rightarrow \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 2 & 1 & 1 & 1 \\ 3 & 1 & 1 & 1 \\ 4 & 2 & 2 & 2 \\ \frac{w_{15}+3w_{35}+5}{w_{15}+w_{35}+1} & 1 & 1 & \frac{w_{15}+w_{35}+2}{w_{15}+w_{35}+1} \end{pmatrix}$$

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ w_{04} & 0 & w_{24} & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 2 & 1 & 1 & 1 \\ 3 & 1 & 1 & 1 \\ 4 & 2 & 2 & 2 \\ 5 & 1 & 1 & 2 \end{pmatrix} \rightarrow \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 2 & 1 & 1 & 1 \\ 3 & 1 & 1 & 1 \\ \frac{2w_{24}+4}{w_{04}+w_{24}+1} & \frac{w_{04}+w_{24}+2}{w_{04}+w_{24}+1} & \frac{w_{04}+w_{24}+2}{w_{04}+w_{24}+1} & \frac{w_{04}+w_{24}+2}{w_{04}+w_{24}+1} \\ 5 & 1 & 1 & 2 \end{pmatrix}$$

$$\begin{pmatrix} 0 & 1 & 1 & 1 \\ \frac{1}{w_{01}+1} & 1 & 1 & 1 \\ \frac{w_{12}+2}{w_{02}+w_{12}+1} & 1 & 1 & 1 \\ \frac{w_{13}+3}{w_{03}+w_{13}+1} & 1 & 1 & 1 \\ 4 & 2 & 2 & 2 \\ 5 & 1 & 1 & 2 \end{pmatrix} \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 2 & 1 & 1 & 1 \\ 3 & 1 & 1 & 1 \\ 4 & 2 & 2 & 2 \\ \frac{w_{15}+3w_{35}+5}{w_{15}+w_{35}+1} & 1 & 1 & \frac{w_{15}+w_{35}+2}{w_{15}+w_{35}+1} \end{pmatrix} \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 2 & 1 & 1 & 1 \\ 3 & 1 & 1 & 1 \\ \frac{2w_{24}+4}{w_{04}+w_{24}+1} & \frac{w_{04}+w_{24}+2}{w_{04}+w_{24}+1} & \frac{w_{04}+w_{24}+2}{w_{04}+w_{24}+1} & \frac{w_{04}+w_{24}+2}{w_{04}+w_{24}+1} \\ 5 & 1 & 1 & 2 \end{pmatrix}$$

↓

DENSE

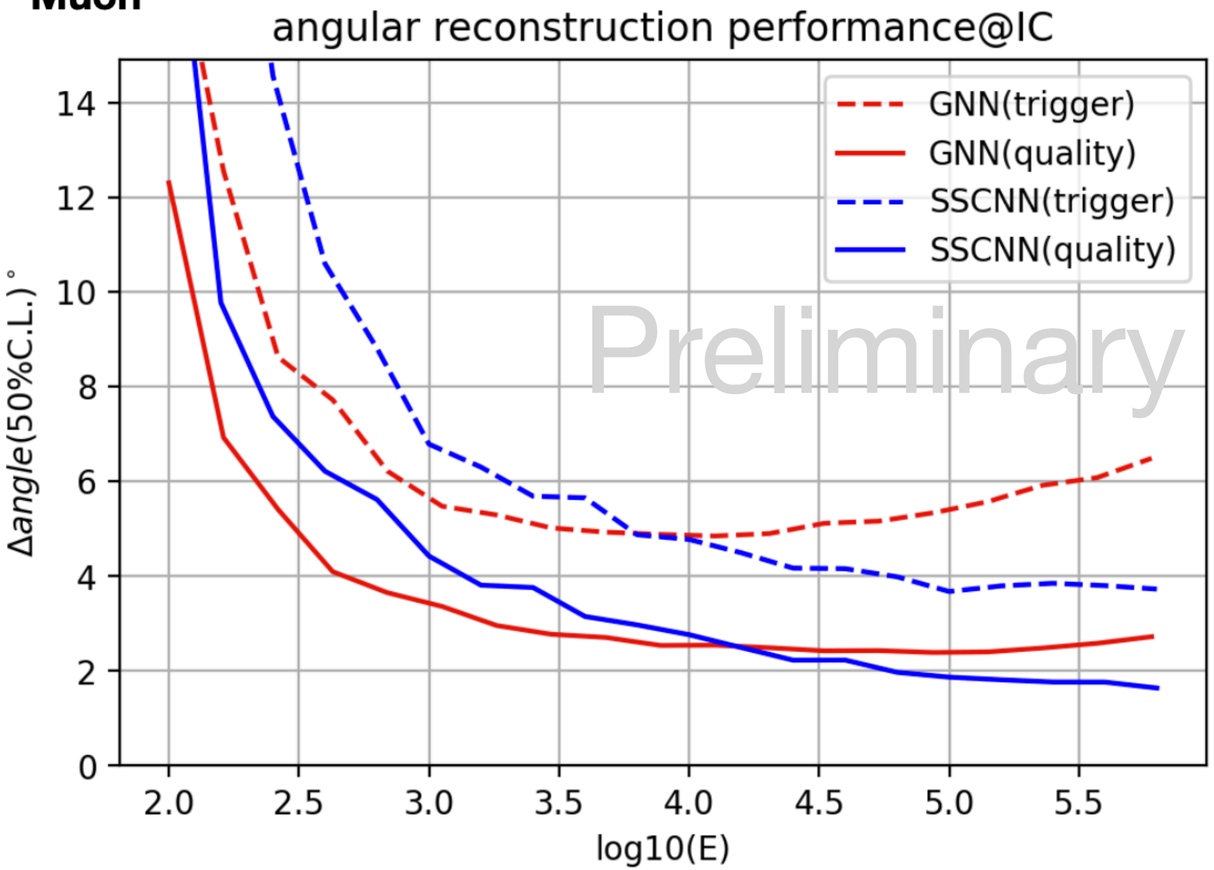
↓

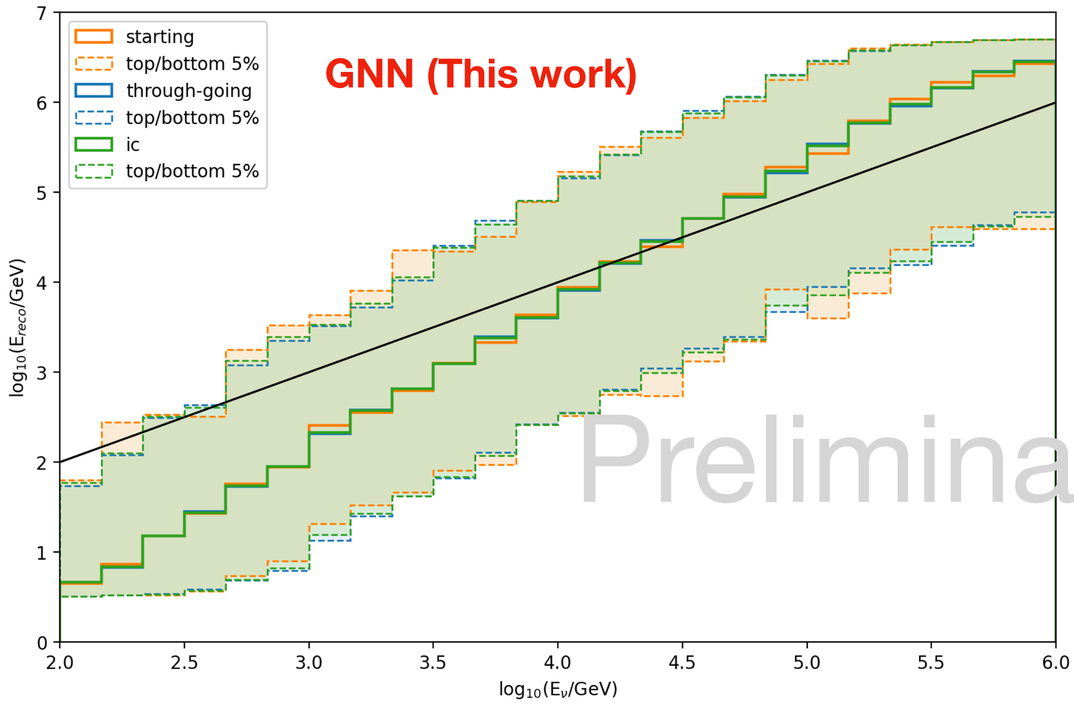
ACTIVATION

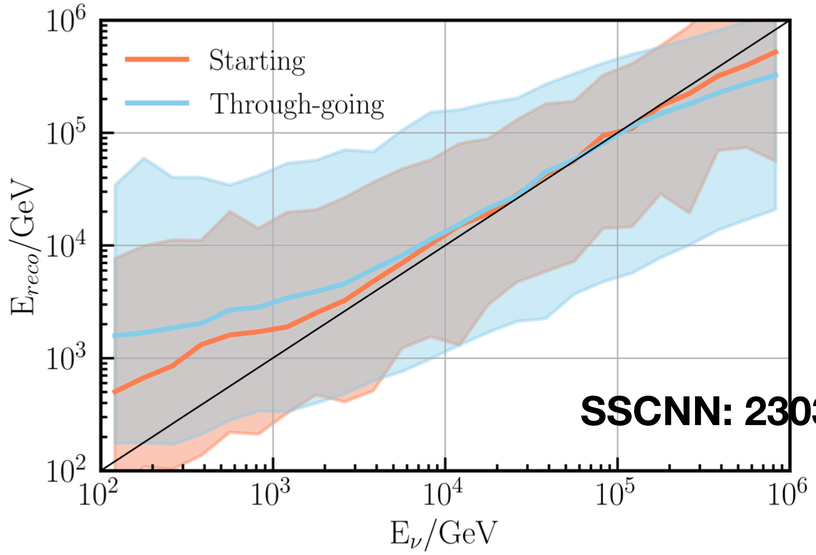
Example 2: Energy/Angular Reconstruction @ IceCube

Performance: Muon, IceCube + Deepcore

Muon



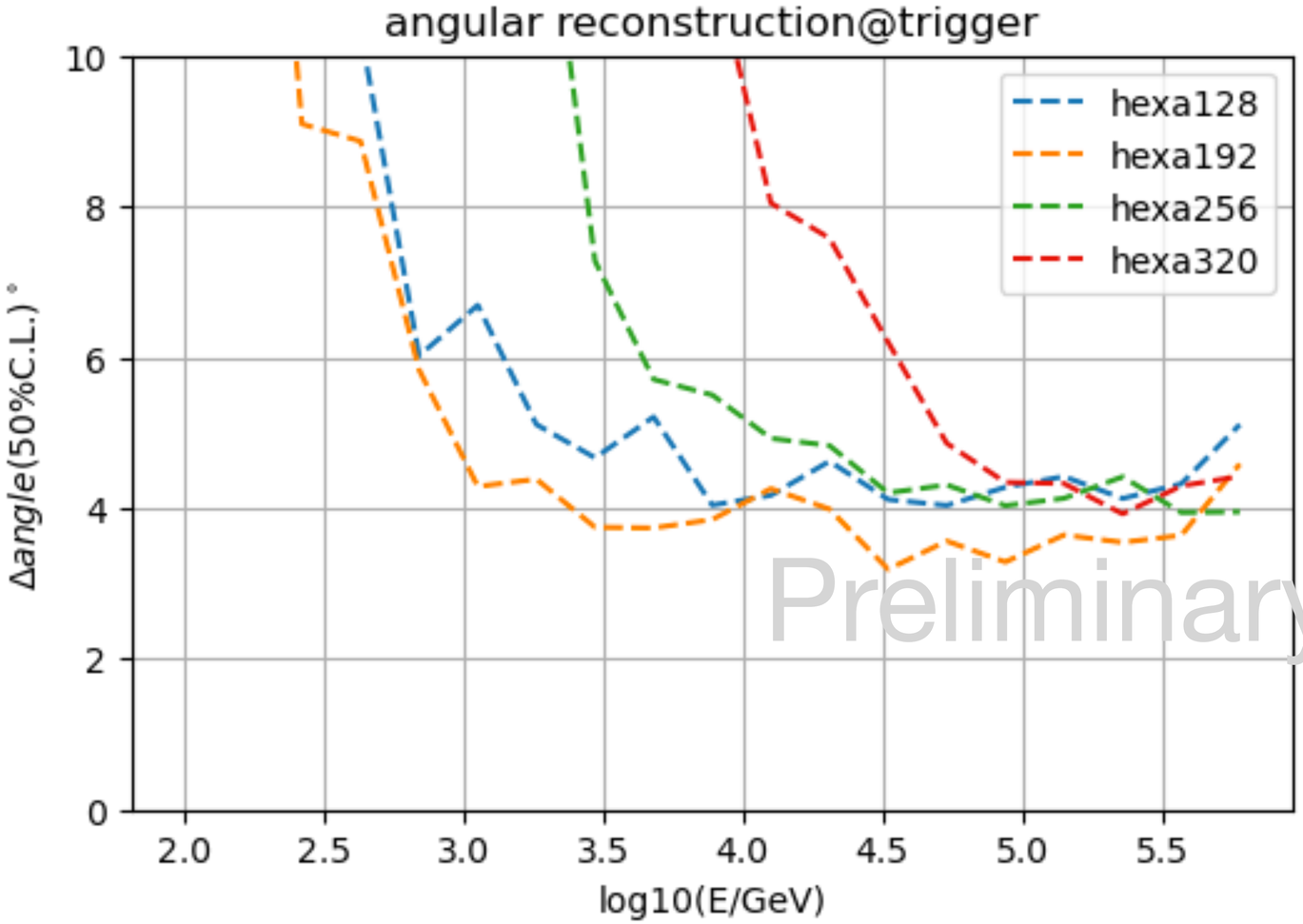


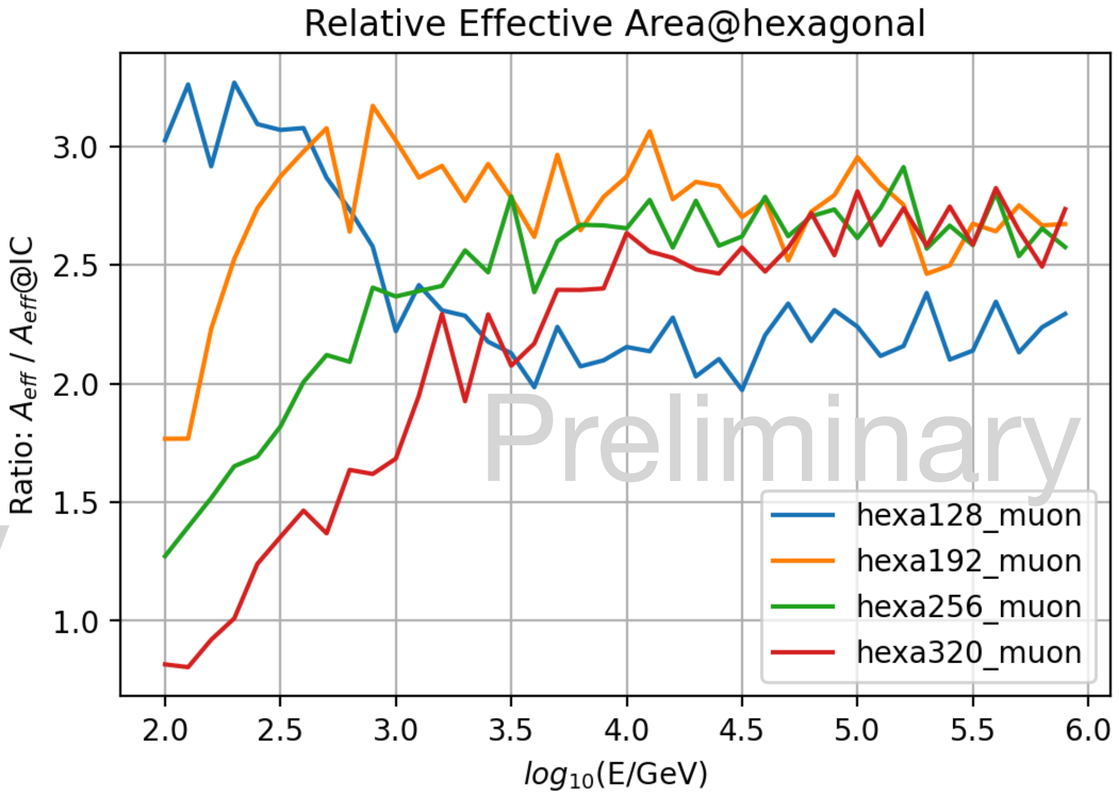


SSCNN: 2303.08812

Example 2: Energy/Angular Reconstruction @ IceCube

Performance: Muon, Hexagonal Shape, Angular Reconstruction





Hands-on

GitHub Repo: https://github.com/tongzhu194/DMP_epgnn/blob/main/env_setup.md
This is a private repo for playing with GNN on DAMPE simulation events.
If interested please send me your GitHub account for access:)
Once you've received an invitation email and confirmed, you will be able to visit the website and get the codes by:

```
git clone https://github.com/tongzhu194/DMP_epgnn.git
```

Following Steps guide users to set up a working environment with a PC.

1. **Create a new conda environment**

- Create a new environment and give it an interesting name: `conda create --name sam python=3.9.12`
  Here I set the python version to 3.9.12, which is one of the compatible versions. If it doesn't work later due to an update, check tensorflow_gnn for information or simply try to get the latest Python.
- Activate the environment: `conda activate sam`

2. **Install the tools**
   This part is based on the instruction provided by tensorflow_gnn. There might be an update for this repo so please check the TensorFlow/Bazel version required by the latest release if the following settings fail to work.

- **Clone tensorflow_gnn**
  `git clone https://github.com/tensorflow/gnn.git tensorflow_gnn`
- **Install Tensorflow**
  `conda install tensorflow=2.10.0`
- **Install Bazel**
  `conda install bazel=5.2.0`
- **Install tfgnn**
  `cd tensorflow_gnn && python setup.py install`

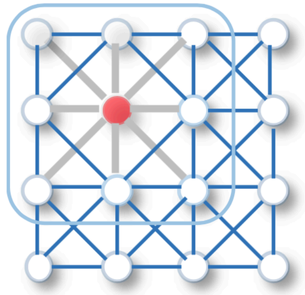
We should be all set!

Resources

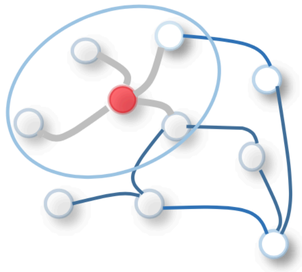
For a quick start with TFGNN: https://www.kaggle.com/code/fidels/introduction-to-tf-gnn
Here is an easy example of binary classification using MPNN.
For better knowledge of TFGNN: https://github.com/tensorflow/gnn/tree/main/tensorflow_gnn/docs/guide
One can find instructions for GNN and TFGNN in this folder.
For an overview&guidepost to concrete GNN architecture: https://arxiv.org/pdf/1901.00596.pdf and its references

Thank you!

Comments & Questions are welcome:)



(a) 2D Convolution. Analogous to a graph, each pixel in an image is taken as a node where neighbors are determined by the filter size. The 2D convolution takes the weighted average of pixel values of the red node along with its neighbors. The neighbors of a node are ordered and have a fixed size.



(b) Graph Convolution. To get a hidden representation of the red node, one simple solution of the graph convolutional operation is to take the average value of the node features of the red node along with its neighbors. Different from image data, the neighbors of a node are unordered and variable in size.

Fig. 1: 2D Convolution vs. Graph Convolution.