GRAPH NEURAL NETWORKS

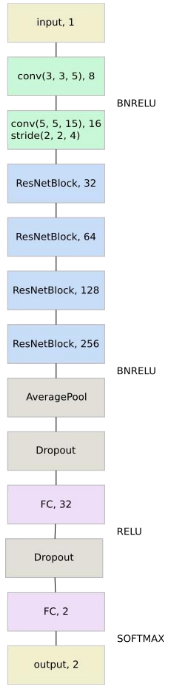
COREY ADAMS
Computational Scientist
Physicist

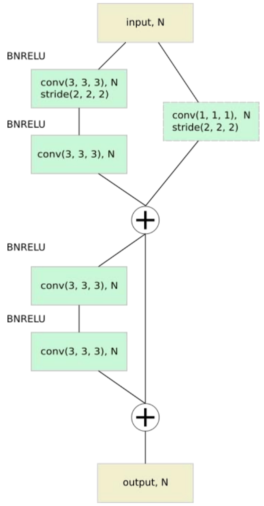
12/10/23
San Juan, PR

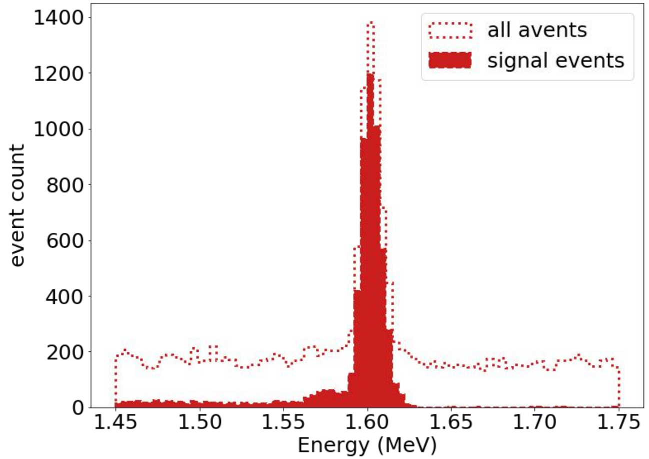
RECAP

- Last time, we covered:
  - History of CNNs
  - CNN Ingredients
  - CNN use cases, and examples of these cases in physics
  - High level thoughts about how to apply a CNN to your dataset.
- This time:
  - Briefly, how to train a CNN on simulation that doesn't quite match data.
  - Then, all about GNNs.

USING AI TO SORT CLASSIFY SIGNAL/BACKGROUND





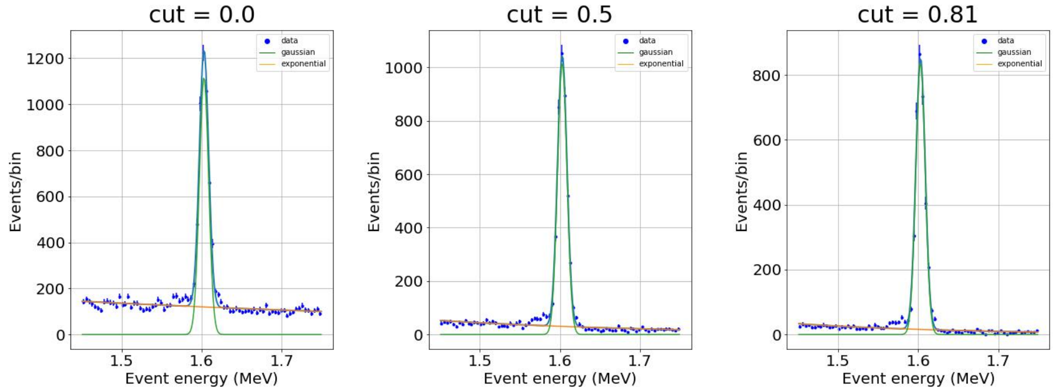


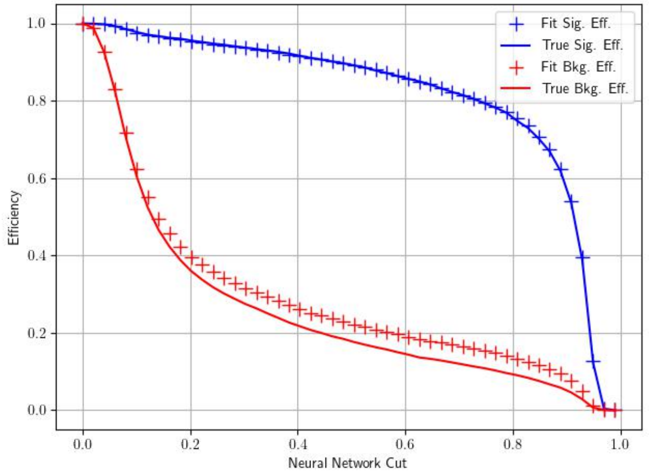
- Input size 40X40x110 (voxel size 10x10x5 mm^3)
- Energy of every event normalized to 1 (so the network does not have information about total event energy)
- ~500000 fiducial events, 35% signal
- Network is **sparse 3D** convolutions

https://link.springer.com/article/10.1007/JHEP01(2021)189

Source Code

DATA DRIVEN EVALUATION





1. Fit the histogram to gaussian (signal) and exponential (background)

2. Integrate to calculate total number of signal and background

3. Apply ith cut on DNN prediction and calculate metrics

$$\epsilon^i_{sig} = \frac{N^i_{sig}}{N^0_{sig}}$$

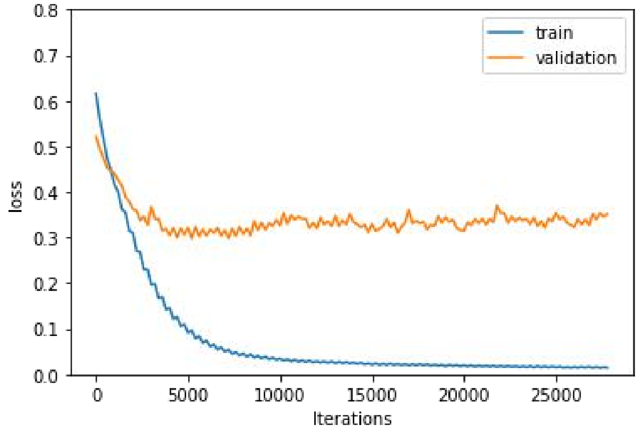
$$\epsilon^i_{bck} = \frac{N^i_{bck}}{N^0_{bck}}$$

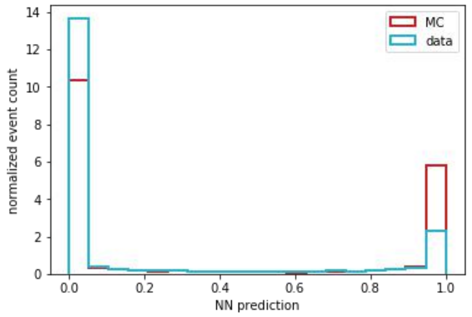
$$\mathrm{f.o.m} = \frac{\epsilon^i_{sig}}{\sqrt{\epsilon^i_{bck}}}$$

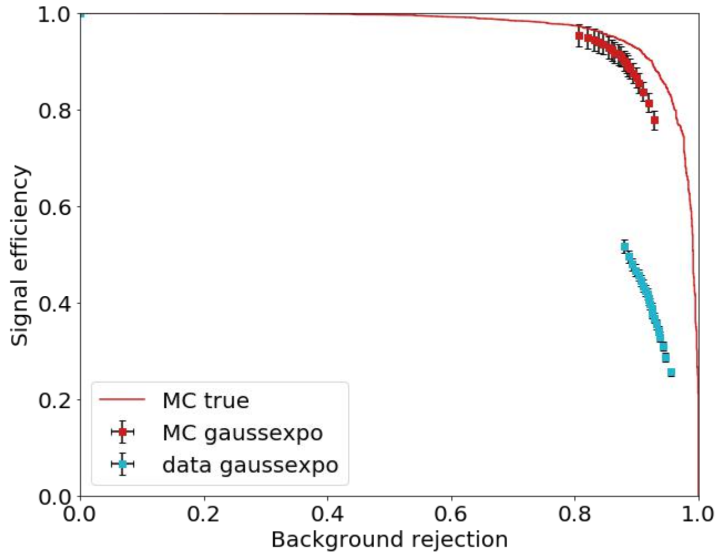
https://link.springer.com/article/10.1007/JHEP01(2021)189

Source Code

DATA VS MC







Prediction on data is biased towards lower values

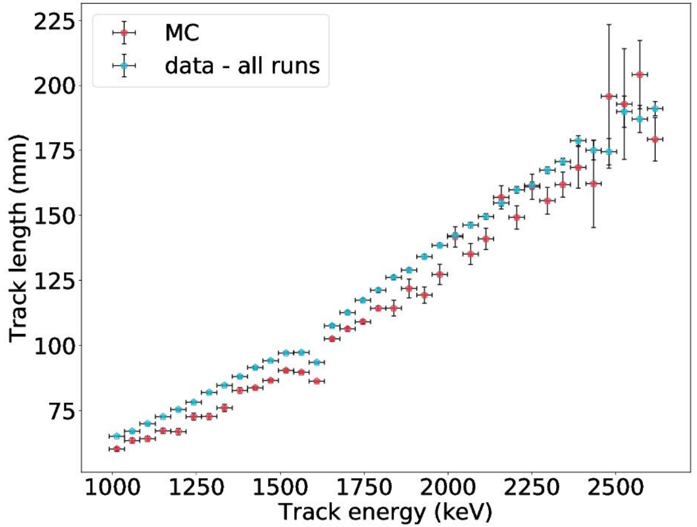
Significant disagreement and performance hit on data

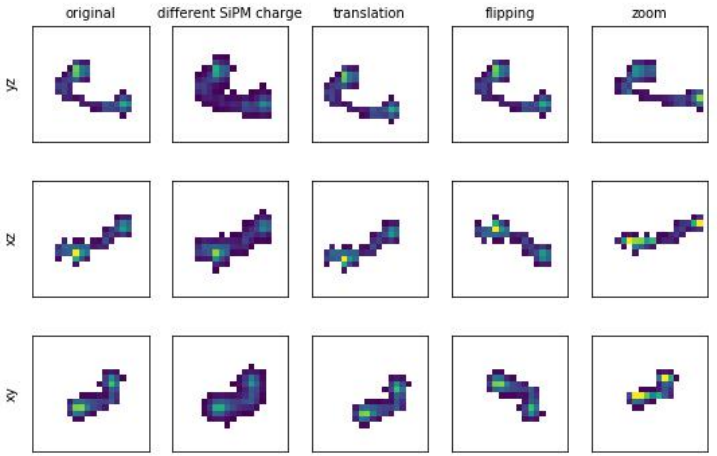
https://link.springer.com/article/10.1007/JHEP01(2021)189
Source Code

DATA AUGMENTATION TO PREVENT OVERFITTING

From classical analysis we know there are some MC/data differences, e.g. track length, blob energy...

We apply on-the-fly augmentation to the training data to prevent the network from using these features during training.



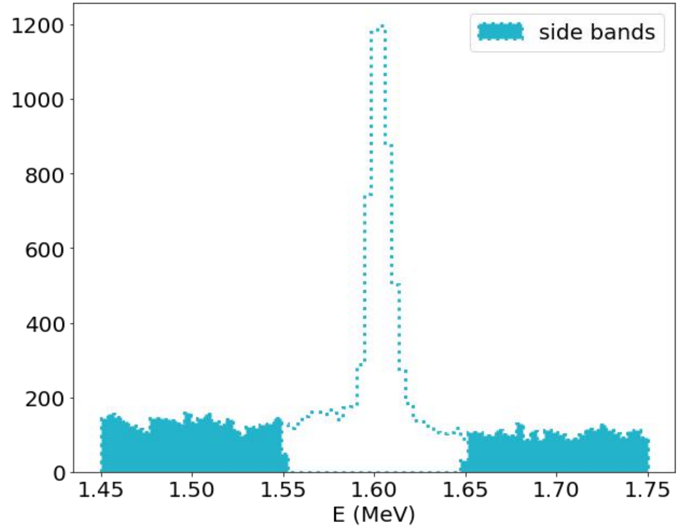


https://link.springer.com/article/10.1007/JHEP01(2021)189

Source Code

HOW TO QUANTIFY THE OVERFITTING?

- Can we know if it will work **before** we look at the interesting data? Can we know if augmentation is improving things?
- Yes – we can use energy distance to compare the flattened layers in the sidebands around the interesting data!
- More about energy distance



$$E_{n,m}(X,Y) := 2A - B - C$$

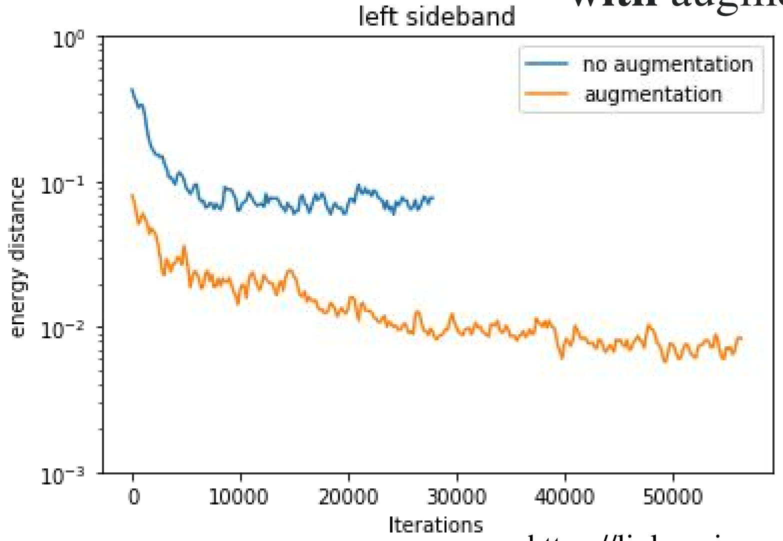
$$A := \frac{1}{nm}\sum_{i=1}^{n}\sum_{j=1}^{m}\|x_i - y_j\|, B := \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n}\|x_i - x_j\|, C := \frac{1}{m^2}\sum_{i=1}^{m}\sum_{j=1}^{m}\|y_i - y_j\|$$

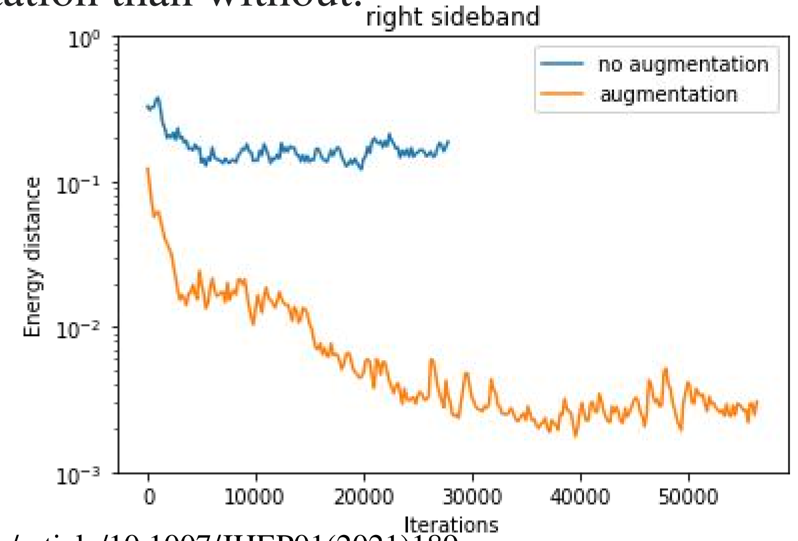
https://link.springer.com/article/10.1007/JHEP01(2021)189

Source Code

TRACK ENERGY DISTANCE DURING TRAINING

The energy distance is significantly lower with augmentation than without – this implies that the features learned by the network are much more closely related between data/MC **with** augmentation than without.

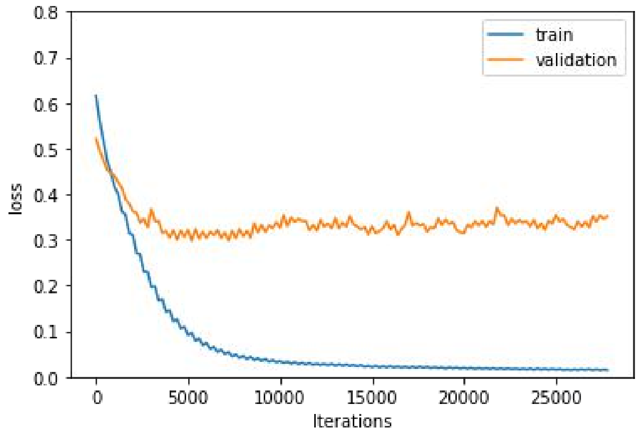


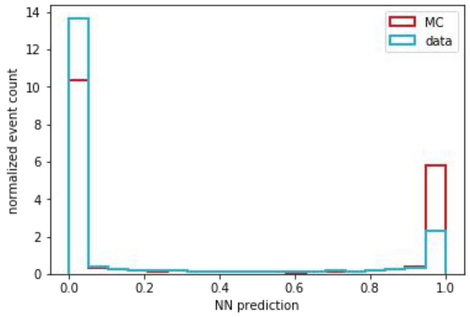


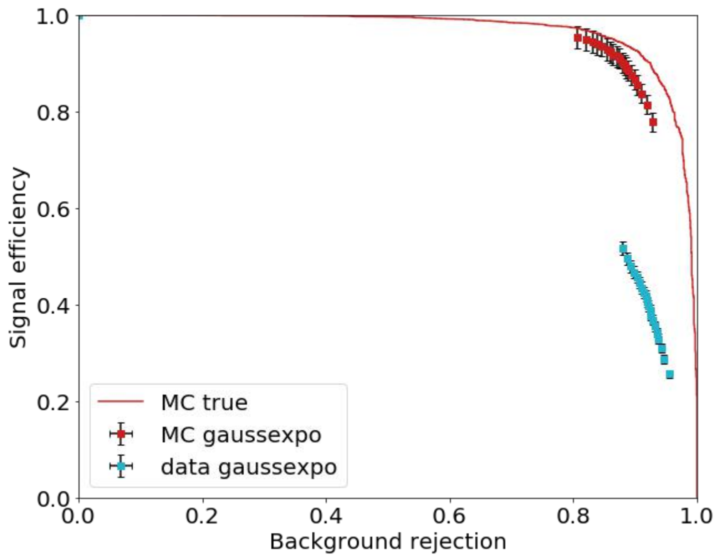
https://link.springer.com/article/10.1007/JHEP01(2021)189

Source Code

DATA VS MC (REPEAT)







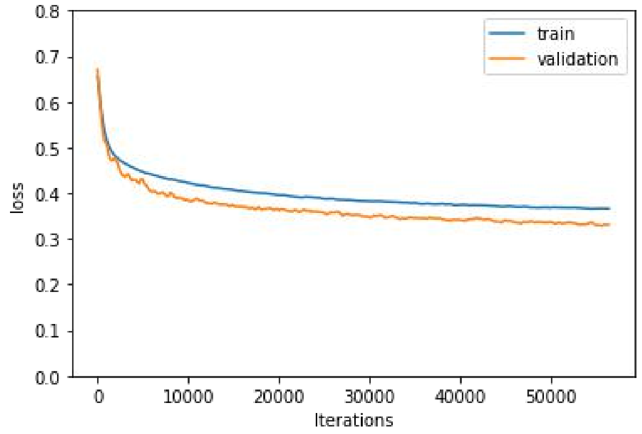
Prediction on data is biased towards lower values

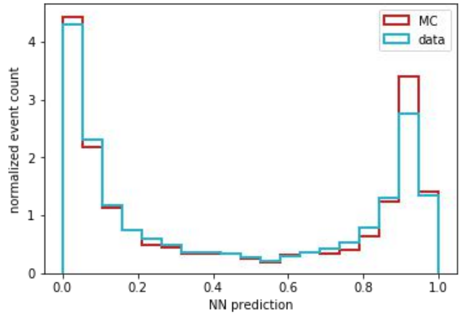
Significant disagreement and performance hit on data

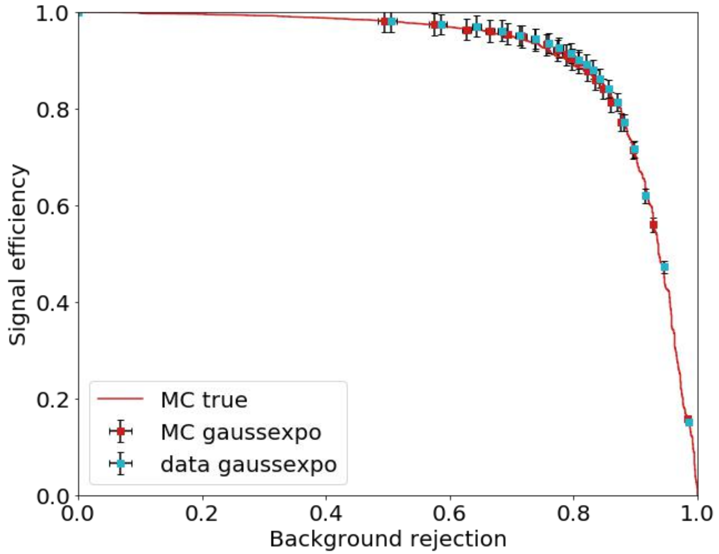
https://link.springer.com/article/10.1007/JHEP01(2021)189

Source Code

WITH AUGMENTATION







Benefit: The data/MC discrepancy is significantly reduced

Cost: the overall performance of the network is reduced.

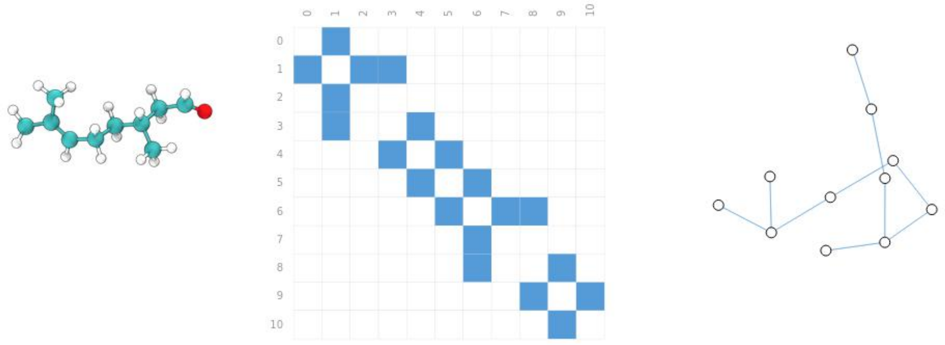
TAKEAWAYS

- When your training data does not perfectly match the target data:
  - Understand the discrepancies - dead pixels? Different sensor responses?
  - Understand the symmetries: is there a "direction" to events? Does dropping pixels from the training data change the labels?
  - If you can remake the training data to match the target data - great!
  - If you can't, use augmentation on the fly.
    - Already a well known technique in Comp Sci.
    - Focus on **label preserving augmentation techniques.**
  - Use sidebands and statistical measurements (energy distance not unique) to quantify discrepancies and know if you are improving things!

GRAPH NEURAL NETWORKS

WHAT MAKES GRAPHS DIFFERENT?

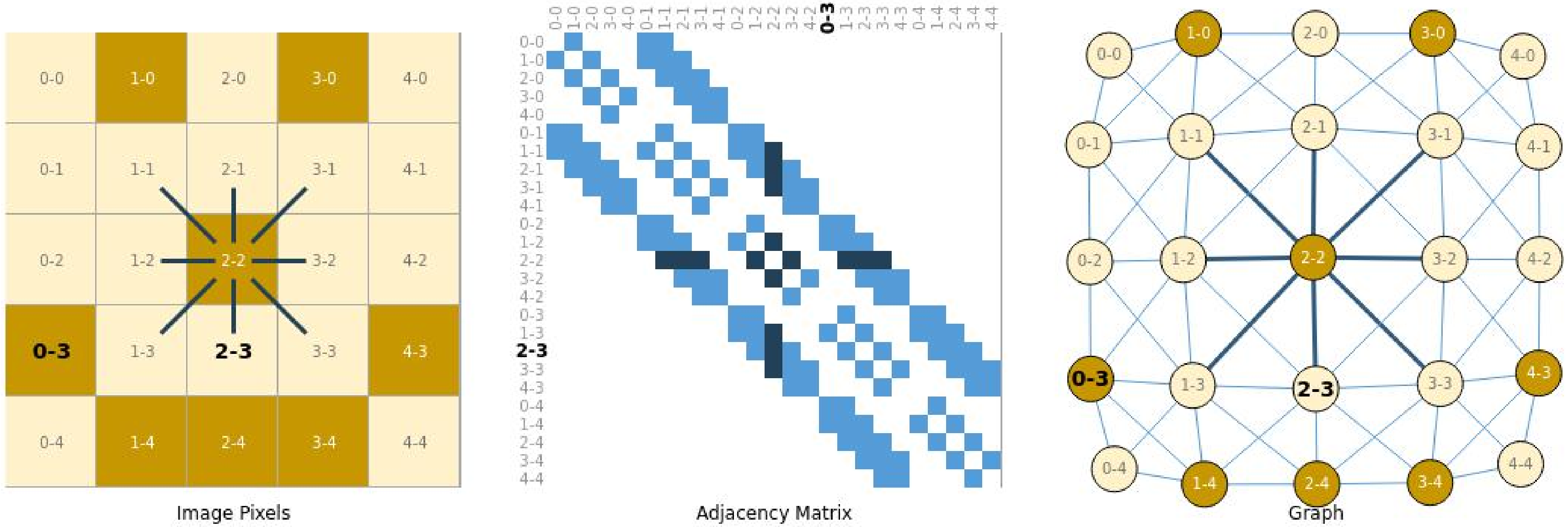
- Graphs are a collection of:
  - **Nodes -** Individual locations that represent some piece of local information
  - **Edges -** Connections between two nodes
  - **Globals** - information that applies to the entire graph.



(Left) 3d representation of the Citronellal molecule (Center) Adjacency matrix of the bonds in the molecule (Right) Graph representation of the molecule.

https://distill.pub/2021/gnn-intro/

IMAGES ARE GRAPHS!

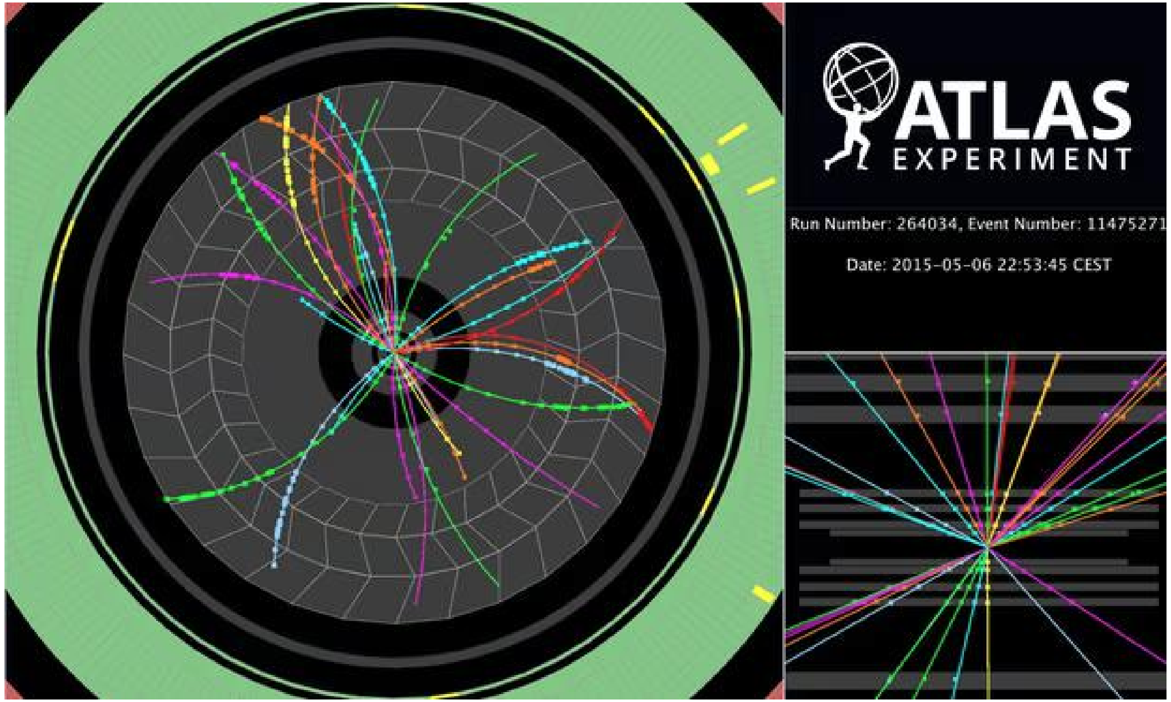


https://distill.pub/2021/gnn-intro/

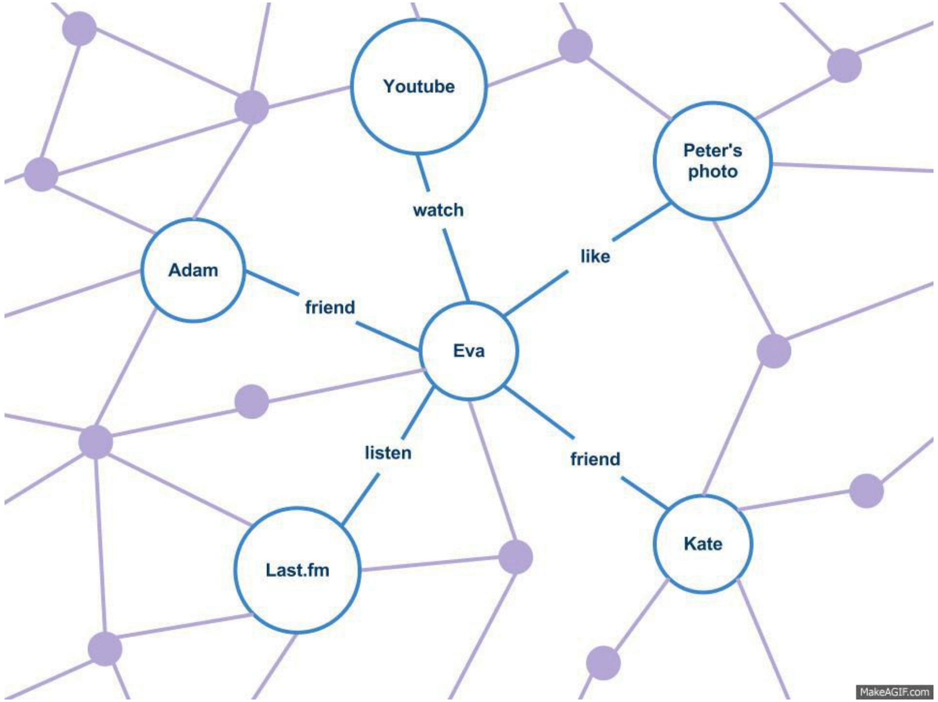
GRAPHS ARE MORE GENERAL THAN IMAGES

Generalize images by relaxing requirements:

- non-uniform pixel spacing
- non-uniform connections
- can be **directed**
- computationally, must be permutation invariant


ATLAS
EXPERIMENT
Run Number: 264034, Event Number: 11475271
Date: 2015-05-06 22:53:45 CEST

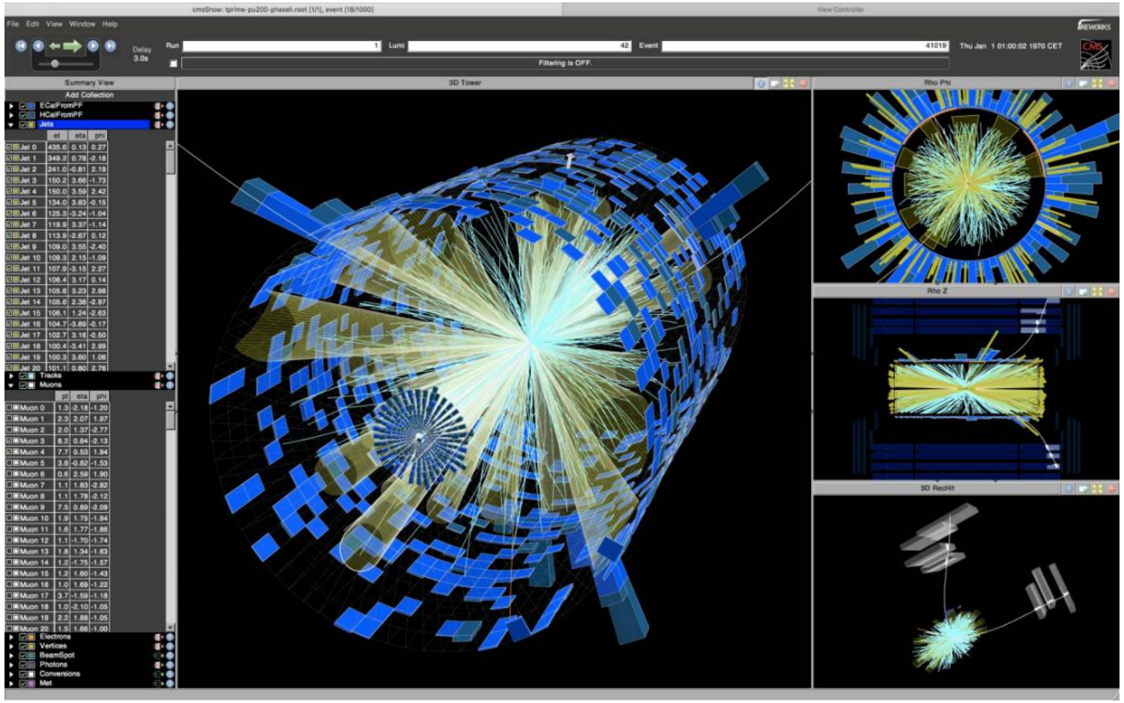
GRAPHS ARE MORE VARIED



Connections in a social network are graph data.

The "type" of connections (friend / watch / like) is an example of an edge feature.

PHYSICS GRAPHS ARE MORE IMAGE-Y



	et	eta	phi
Jet 0	435.6	0.13	0.27
Jet 1	349.2	0.78	-2.18
Jet 2	241.0	-0.81	2.19
Jet 3	150.2	3.66	-1.73
Jet 4	150.0	3.59	2.42
Jet 5	134.0	3.83	-0.15
Jet 6	125.3	-3.24	-1.04
Jet 7	119.9	3.37	-1.14
Jet 8	113.9	-2.67	0.12
Jet 9	109.0	3.55	-2.40
Jet 10	109.3	2.15	-1.09
Jet 11	107.9	-3.15	2.27
Jet 12	106.4	3.17	0.14
Jet 13	105.8	3.23	2.98
Jet 14	105.6	2.38	-2.97
Jet 15	106.1	1.24	-2.63
Jet 16	104.7	-3.89	-0.17
Jet 17	102.7	3.16	-0.50
Jet 18	100.4	-3.41	2.99
Jet 19	100.3	3.60	1.06
Jet 20	101.1	0.80	2.76

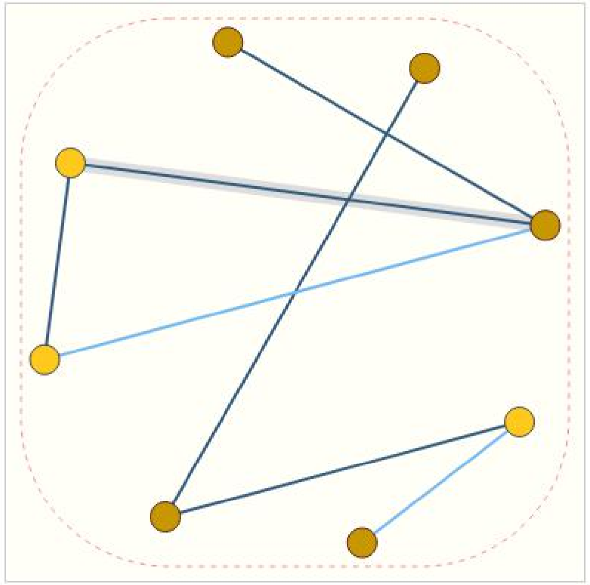
	pt	eta	phi
Muon 0	1.3	-2.18	-1.20
Muon 1	2.3	2.07	1.97
Muon 2	2.0	1.37	-2.77
Muon 3	8.2	0.84	-2.13
Muon 4	7.7	0.53	1.94
Muon 5	3.8	-0.82	-1.53
Muon 6	0.6	2.59	1.90
Muon 7	1.1	1.83	-2.82
Muon 8	1.1	1.78	-2.12
Muon 9	7.5	0.89	-2.09
Muon 10	1.9	1.75	-1.94
Muon 11	1.6	1.77	-1.86
Muon 12	1.1	-1.70	-1.74
Muon 13	1.8	1.34	-1.63
Muon 14	1.2	-1.75	-1.57
Muon 15	1.2	1.60	-1.43
Muon 16	1.0	1.69	-1.22
Muon 17	3.7	-1.59	-1.18
Muon 18	1.0	-2.10	-1.05
Muon 19	2.2	1.88	-1.05
Muon 20	1.5	1.66	-1.00

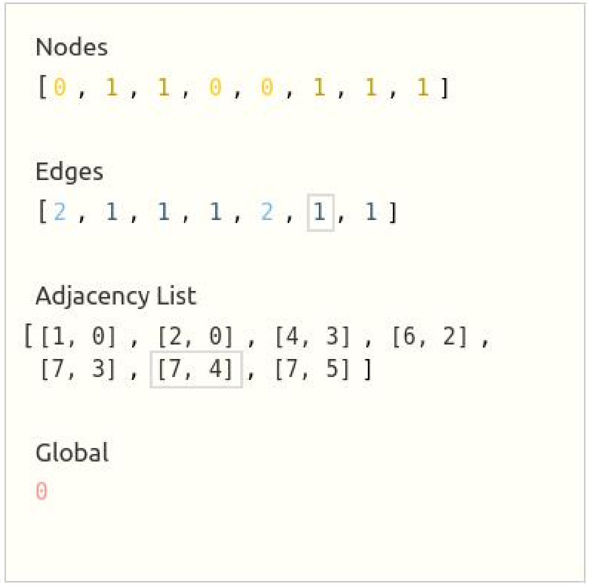
We often use graphs when images don’t make sense:

- How to embed circular/cylindrical data into an image?
- What if pixels are a differnt size?
- What about GEANT output, directed particle interactions at specified locations?

GRAPH MATH

- Graphs can be large, and often **sparse.**
  - **Nodes** can be stored in arrays of [n_nodes, n_node_features]
  - **Edges** can be stored in arrays of [n_edges, n_edge_features]
  - The **adjacency matrix** is a **sparse** connectivity between nodes





Nodes

[0 , 1 , 1 , 0 , 0 , 1 , 1 , 1]

Edges

[2 , 1 , 1 , 1 , 2 , 1 , 1]

Adjacency List

[[1, 0] , [2, 0] , [4, 3] , [6, 2] ,
[7, 3] , [7, 4] , [7, 5]]

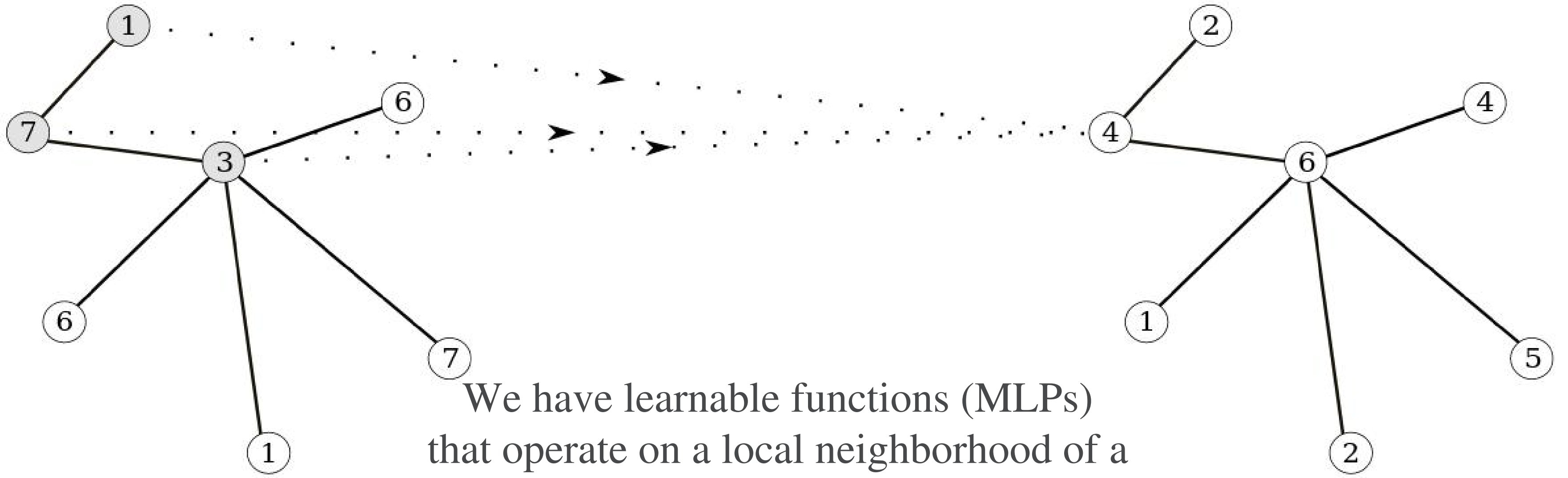
Global

0

https://distill.pub/2021/gnn-intro/

GRAPH OPERATIONS

If Graphs are generalizations of images, what’s the generalization of convolutions?



We have learnable functions (MLPs) that operate on a local neighborhood of a node to update the node.

https://distill.pub/2021/gnn-intro/

GRAPH OPERATIONS

Formally, if $h_k^{(i-1)}$ represents the node k at layer i-1:

$$a_k^i = \text{AGGREGATE}(h_u^{(i-1)}, \text{u connected to k})$$

$$h_k^i = \text{COMBINE}(h_k^{(i-1)}, a_k^i)$$

Most graph convolutions can be written like this

(assuming all edges have no features, like images)

How Powerful are Graph Neural Networks?

GRAPH OPERATIONS

After multiple layers of graph AGGREGATE and COMBINE, you can READOUT your graph information for the prediction of the network.

Possible graph-level readout: POOL over the nodes

(aka, sum(nodes, axis=0) where nodes has shape [n_nodes, n_features])

AGGREGATE, COMBINE, READOUT all often have learnable parameters, typically as MLPs. READOUT is almost certainly required to be **permutation-invariant.**

How Powerful are Graph Neural Networks?

WHAT ABOUT THE EDGES?

For nodes h and edges e:

$$m_v^{t+1} = \sum_{w \in N(v)} M_t(h_v^t, h_w^t, e_{vw})$$

$$h_v^{t+1} = U_t(h_v^t, m_v^{t+1})$$

Where M, and U are learnable “message” and “update” functions.

(Upon close inspection, the previous operations are variants of Message Passing!)

Neural Message Passing for Quantum Chemistry

BUILDING YOUR GNN

- Most parameters of a GNN come from the learnable parameters of aggregation, message creation, etc.
- Normalization of your data is still valuable!
- Pooling of information **between nodes**, with or without messages, is critical for the network.
- Pooling of the **graph**, however, is not a well defined operation in many cases!

GRAPH POOLING

Graph Pooling is an ill-defined, ambiguous task.

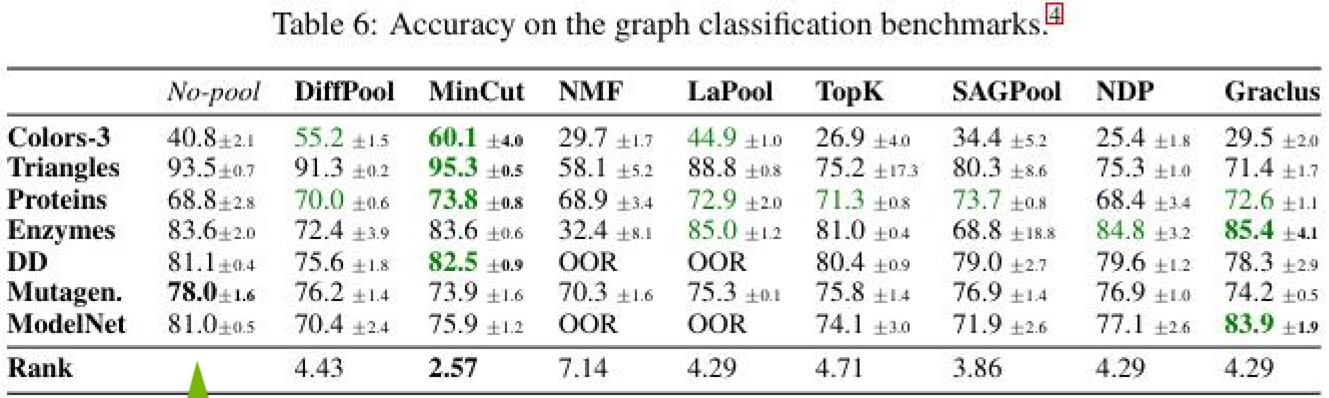


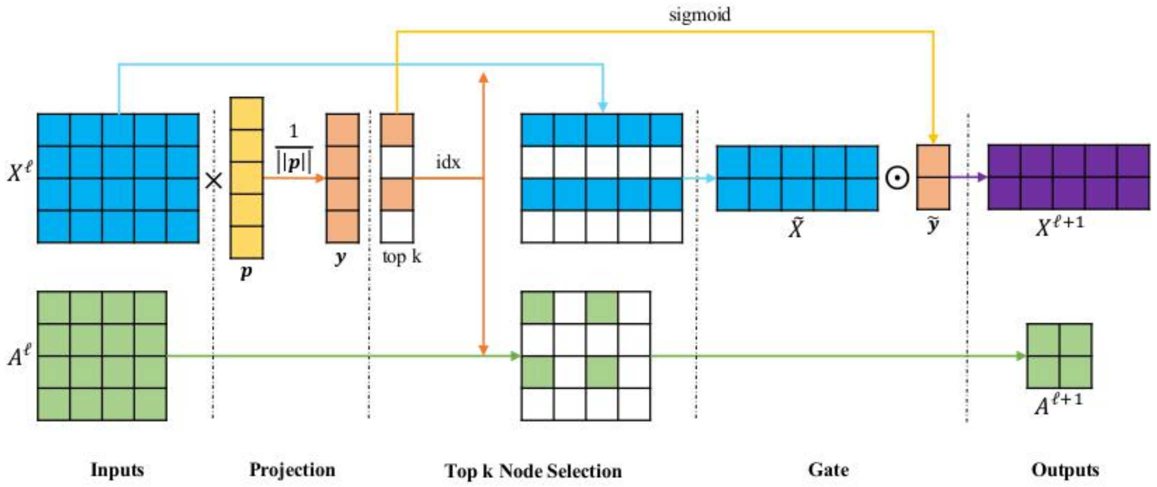
Table 6: Accuracy on the graph classification benchmarks.[4]

	No-pool	**DiffPool**	**MinCut**	**NMF**	**LaPool**	**TopK**	**SAGPool**	**NDP**	**Graclus**
Colors-3	40.8 ±2.1	55.2 ±1.5	**60.1 ±4.0**	29.7 ±1.7	44.9 ±1.0	26.9 ±4.0	34.4 ±5.2	25.4 ±1.8	29.5 ±2.0
Triangles	93.5 ±0.7	91.3 ±0.2	**95.3 ±0.5**	58.1 ±5.2	88.8 ±0.8	75.2 ±17.3	80.3 ±8.6	75.3 ±1.0	71.4 ±1.7
Proteins	68.8 ±2.8	70.0 ±0.6	**73.8 ±0.8**	68.9 ±3.4	72.9 ±2.0	71.3 ±0.8	73.7 ±0.8	68.4 ±3.4	72.6 ±1.1
Enzymes	83.6 ±2.0	72.4 ±3.9	83.6 ±0.6	32.4 ±8.1	85.0 ±1.2	81.0 ±0.4	68.8 ±18.8	84.8 ±3.2	**85.4 ±4.1**
DD	81.1 ±0.4	75.6 ±1.8	**82.5 ±0.9**	OOR	OOR	80.4 ±0.9	79.0 ±2.7	79.6 ±1.2	78.3 ±2.9
Mutagen.	**78.0 ±1.6**	76.2 ±1.4	73.9 ±1.6	70.3 ±1.6	75.3 ±0.1	75.8 ±1.4	76.9 ±1.4	76.9 ±1.0	74.2 ±0.5
ModelNet	81.0 ±0.5	70.4 ±2.4	75.9 ±1.2	OOR	OOR	74.1 ±3.0	71.9 ±2.6	77.1 ±2.6	**83.9 ±1.9**
Rank		4.43	**2.57**	7.14	4.29	4.71	3.86	4.29	4.29

Recent results call into question if it is worth pooling at all...

https://arxiv.org/pdf/2110.05292.pdf

GRAPH U NETS



- Downsampling learns a trainable projection layer, per downsample, to select nodes.
- Also used in a sigmoid to gate information flow (and make it differentiable!)

https://arxiv.org/abs/1905.05178

GRAPH U NETS

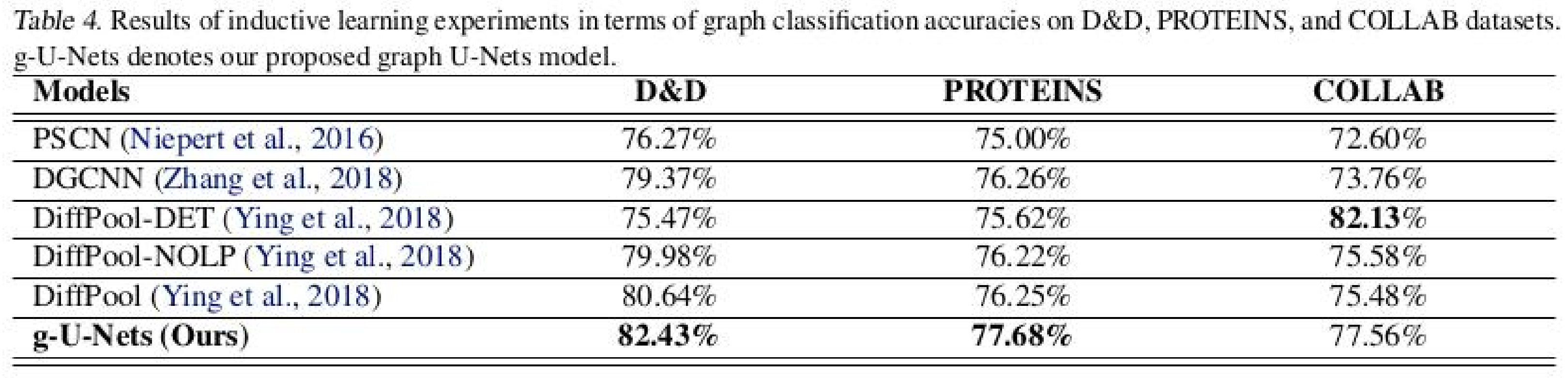


Table 4. Results of inductive learning experiments in terms of graph classification accuracies on D&D, PROTEINS, and COLLAB datasets. g-U-Nets denotes our proposed graph U-Nets model.

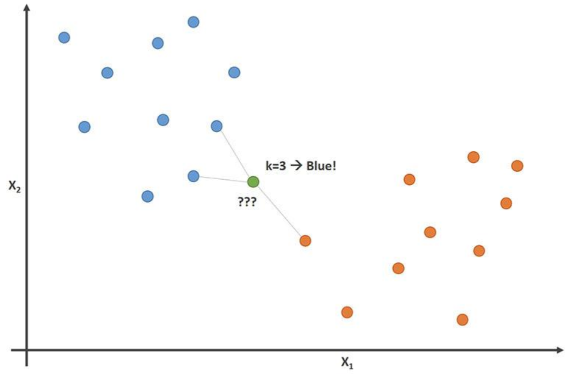
Models	**D&D**	**PROTEINS**	**COLLAB**
PSCN (Niepert et al., 2016)	76.27%	75.00%	72.60%
DGCNN (Zhang et al., 2018)	79.37%	76.26%	73.76%
DiffPool-DET (Ying et al., 2018)	75.47%	75.62%	**82.13%**
DiffPool-NOLP (Ying et al., 2018)	79.98%	76.22%	75.58%
DiffPool (Ying et al., 2018)	80.64%	76.25%	75.48%
g-U-Nets (Ours)	**82.43%**	**77.68%**	77.56%

Results improve upon contemporary models but not in all cases....

https://arxiv.org/abs/1905.05178

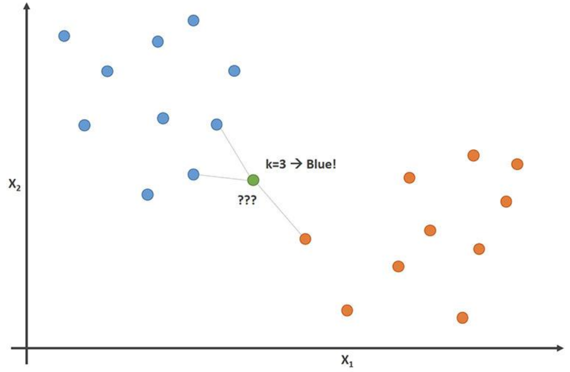
CONNECTING YOUR GNN

- Usually, defining nodes is easy in graphs, especially physics graphs.
  - Typically have some “position” (x/y/z or similar) and some “feature”.
  - You can often just concatenated these together and have node features.
  - Much like MLPs and dense neural networks, feature normalization can be useful.
- Connecting nodes and defining edges:
  - Sometimes easy (particle flow data)
  - Sometimes ambiguous.
  - When in doubt, try k-NN algorithm
  - Usually, don’t connect all-to-all!
- Defining Edge features?
  - If nothing else, displacement vector of features

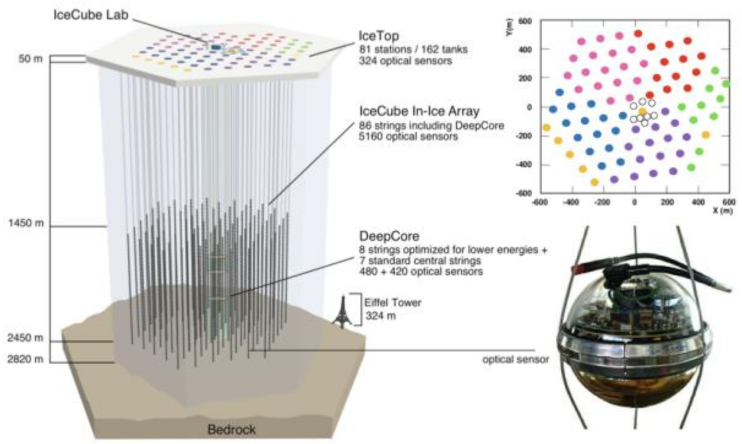


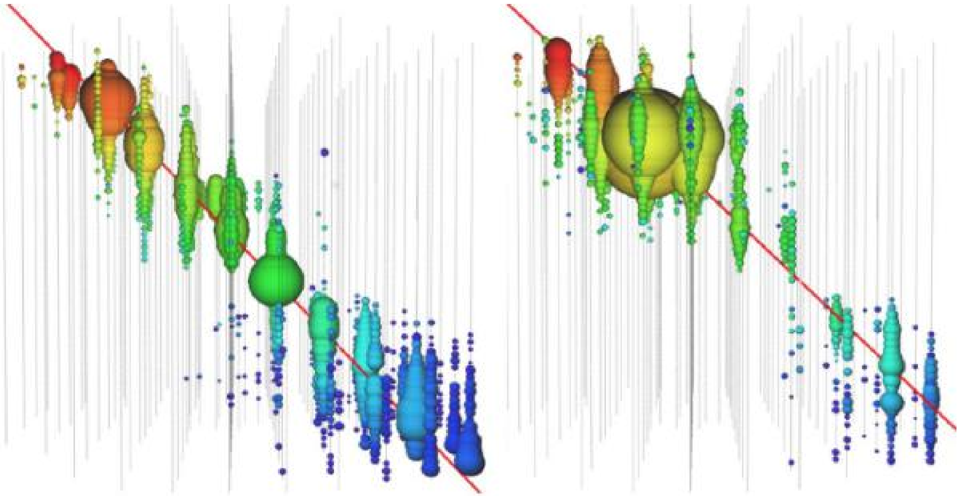
CONNECTING YOUR GNN

- Usually, defining nodes is easy in graphs, especially physics graphs.
  - Typically have some “position” (x/y/z or similar) and some “feature”.
  - You can often just concatenated these together and have node features.
  - Much like MLPs and dense neural networks, feature normalization can be useful.
- Connecting nodes and defining edges:
  - Sometimes easy (particle flow data)
  - Sometimes ambiguous.
  - When in doubt, try k-NN algorithm
  - Usually, don’t connect all-to-all!
- Defining Edge features?
  - If nothing else, displacement vector of features



EXAMPLE: CLASSIFICATION WITH GNNS





Icecube, at the south pole, has regular but not rectangular sensor layouts - CNNs are not a great fit. GNNs instead are an improvement.

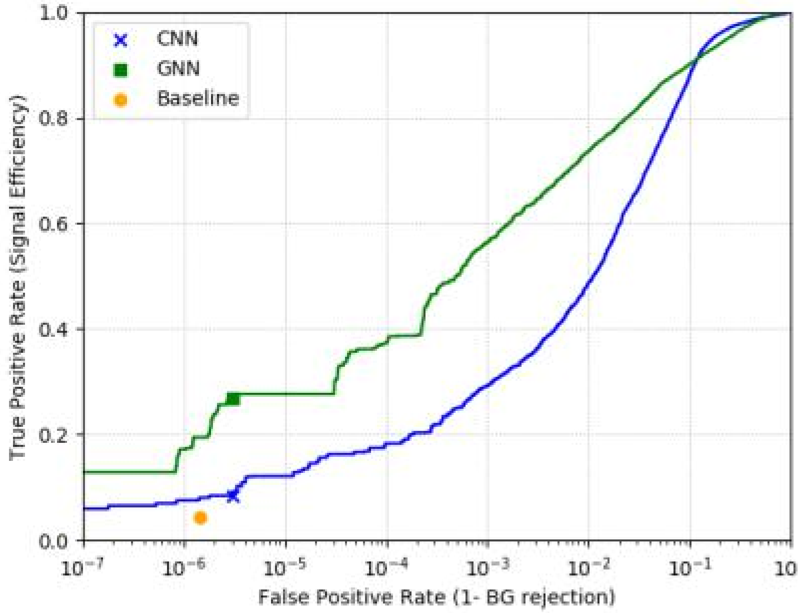
https://arxiv.org/pdf/1809.06166.pdf

EXAMPLE: CLASSIFICATION WITH GNNS

$$\mathrm{GConv}(\mathbf{X}^{(t)}) = [\mathbf{A}\mathbf{X}^{(t)},\ \mathbf{X}^{(t)}](\mathbf{a}^{(t)})^\top + b^{(t)}\mathbf{1},$$

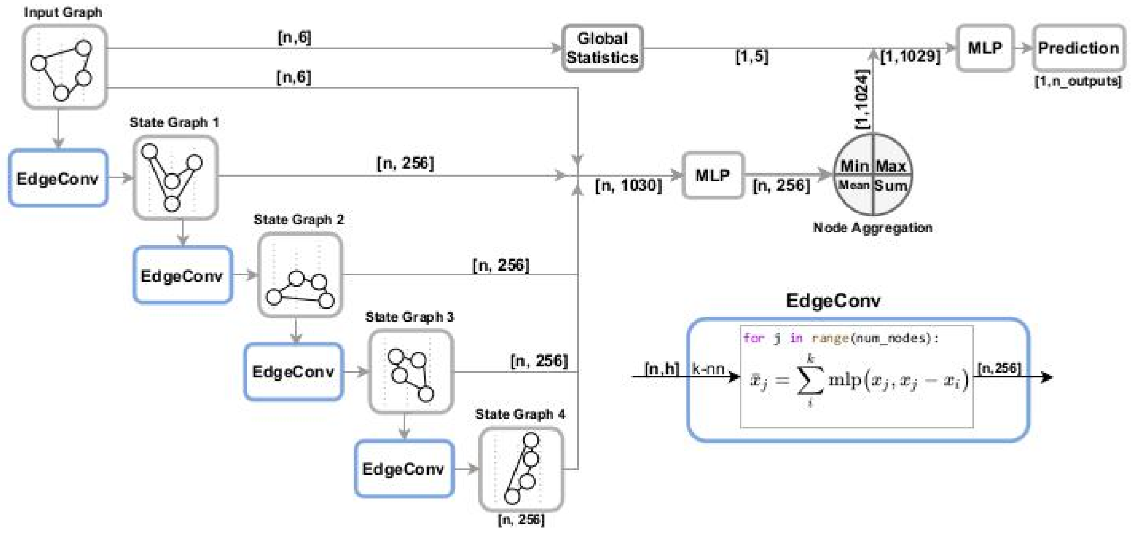
Both GNNs and CNNs (on transformed data) are tested, using a Graph Convolution operator. GNNs significantly outperform CNNs.

	# events per year		
Method	Signal	Background	Signal:Noise
Physics Baseline	0.922	0.934	0.987
3D CNN	1.815	1.937	0.937
GNN	**5.772**	1.937	**2.980**



https://arxiv.org/pdf/1809.06166.pdf

EXAMPLE: CLASSIFICATION WITH GNNS

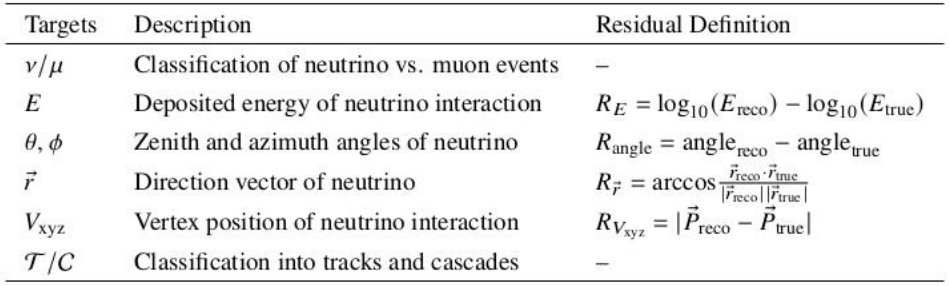


$$\tilde{x}_j = \sum_{i=1}^{N_{\text{neighbors}}} \text{MLP}(x_j, x_j - x_i),$$

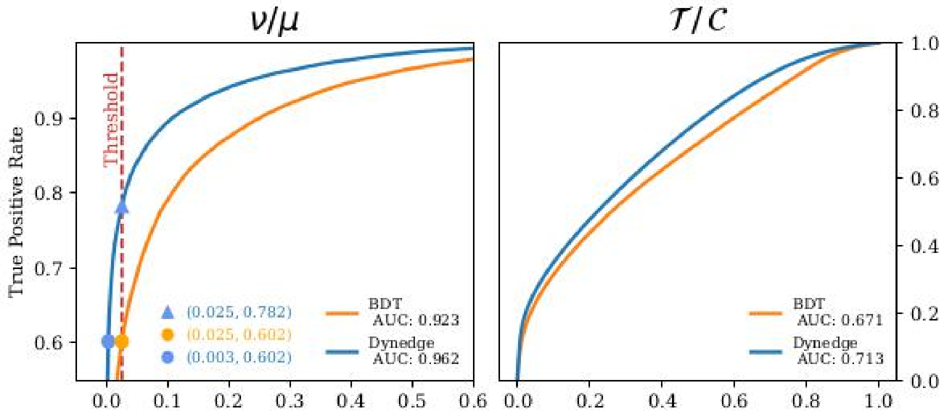
2022 Update from Icecube extends GNN usecase to lower energy events.

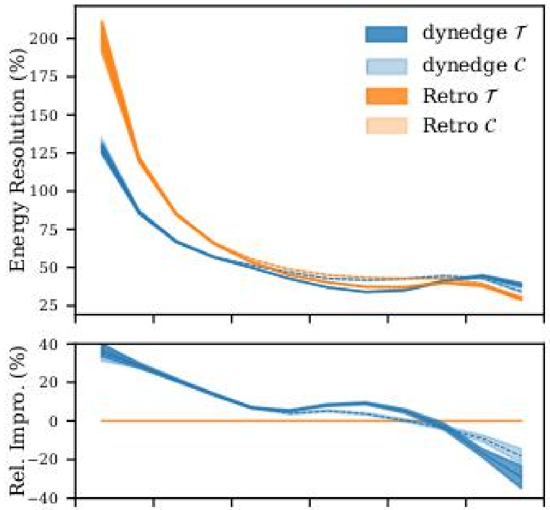
https://arxiv.org/abs/2209.03042

EXAMPLE: CLASSIFICATION WITH GNNS



Targets	Description	Residual Definition
ν/μ	Classification of neutrino vs. muon events	–
E	Deposited energy of neutrino interaction	$R_E = \log_{10}(E_{\text{reco}}) - \log_{10}(E_{\text{true}})$
θ, ϕ	Zenith and azimuth angles of neutrino	$R_{\text{angle}} = \text{angle}_{\text{reco}} - \text{angle}_{\text{true}}$
$\vec{r}$	Direction vector of neutrino	$R_{\vec{r}} = \arccos \frac{\vec{r}_{\text{reco}} \cdot \vec{r}_{\text{true}}}{\lvert\vec{r}_{\text{reco}}\rvert \lvert\vec{r}_{\text{true}}\rvert}$
V_{xyz}	Vertex position of neutrino interaction	$R_{V_{\text{xyz}}} = \lvert\vec{P}_{\text{reco}} - \vec{P}_{\text{true}}\rvert$
$\mathcal{T}/\mathcal{C}$	Classification into tracks and cascades	–

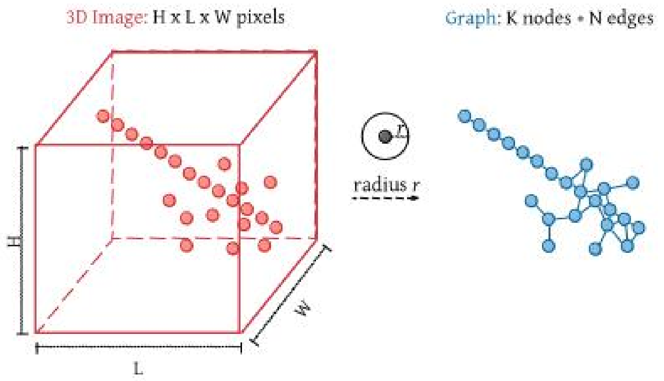


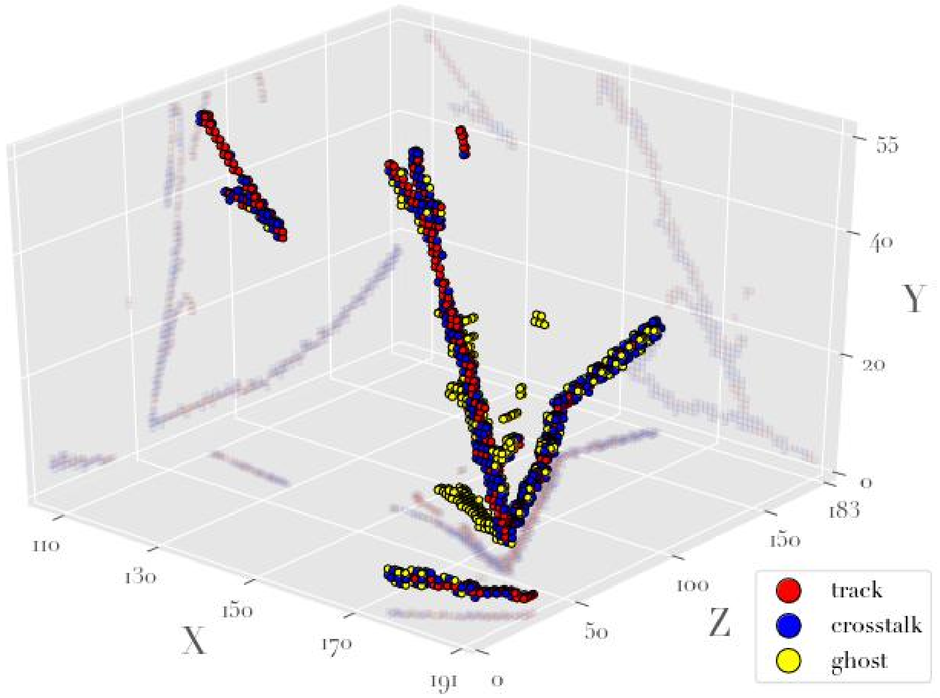


Improvement in classification metrics as well as in regression variables (energy, angles) compared to traditional reconstruction.

https://arxiv.org/abs/1902.08570

EXAMPLE: SEGMENTATION WITH GNNS

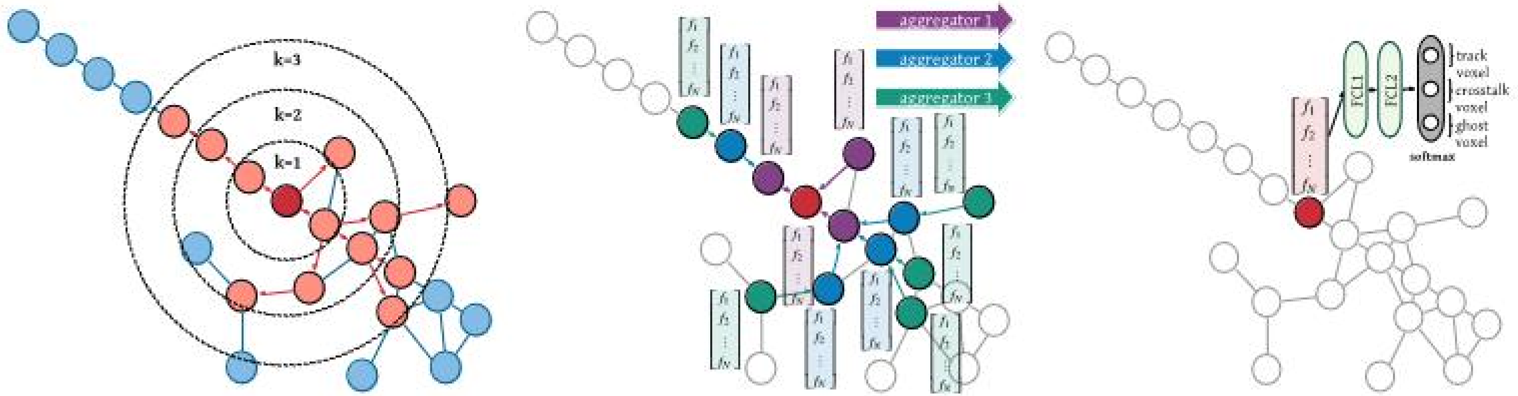




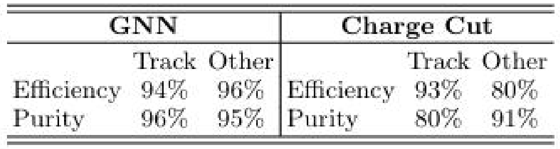
Challenge: 2D scintillator data projected to 3D images can produce fake 3D hits from crosstalk or coincidence hits (ghost) - can a GNN label everything correctly looking at all 3D hits?

Graph neural network for 3D classification of ambiguities and optical crosstalk in scintillator-based neutrino detectors.

EXAMPLE: SEGMENTATION WITH GNNS



3 layers of GraphSAGE + aggregation + fully connected + per-node segmentation

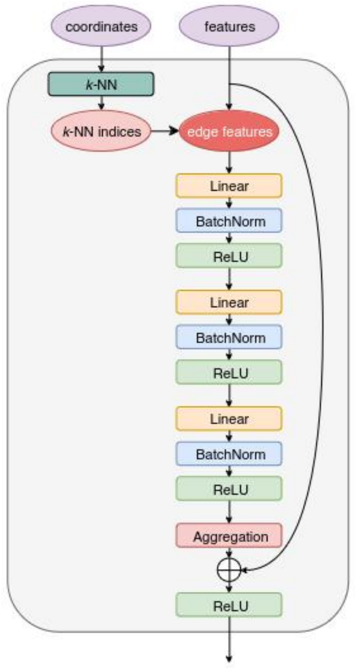


GNN			Charge Cut		
	Track	Other		Track	Other
Efficiency	94%	96%	Efficiency	93%	80%
Purity	96%	95%	Purity	80%	91%

TABLE IV: Mean efficiencies and purities of voxel classification for the GNN and a simple charge cut.

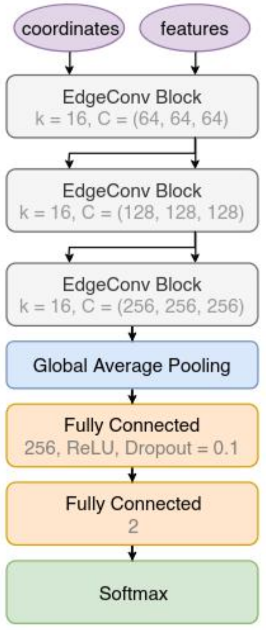
Graph neural network for 3D classification of ambiguities and optical crosstalk in scintillator-based neutrino detectors.

EXAMPLE: CLASSIFICATION WITH GNNS

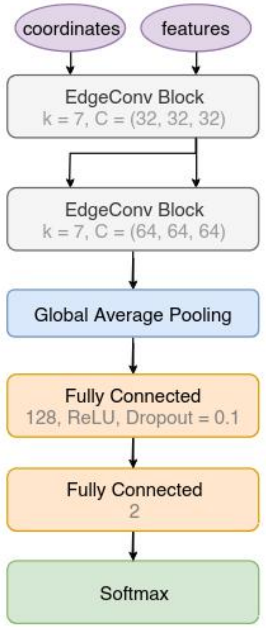


Challenge: Jet images are sparse and CNNs are highly inefficient - can GNNs improve over CNNs?

Input features include significant derived physics information.



(a) ParticleNet

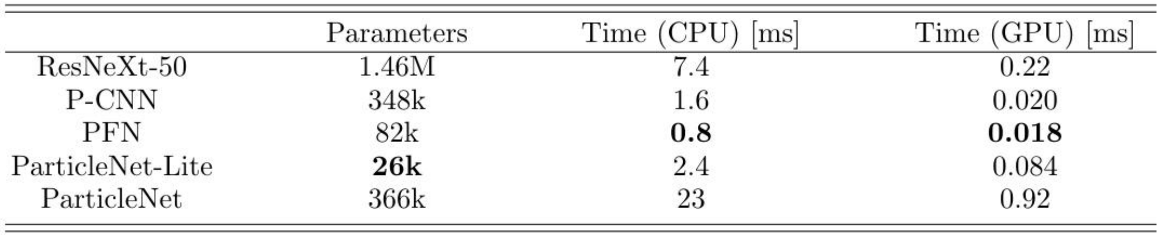


(b) ParticleNet-Lite

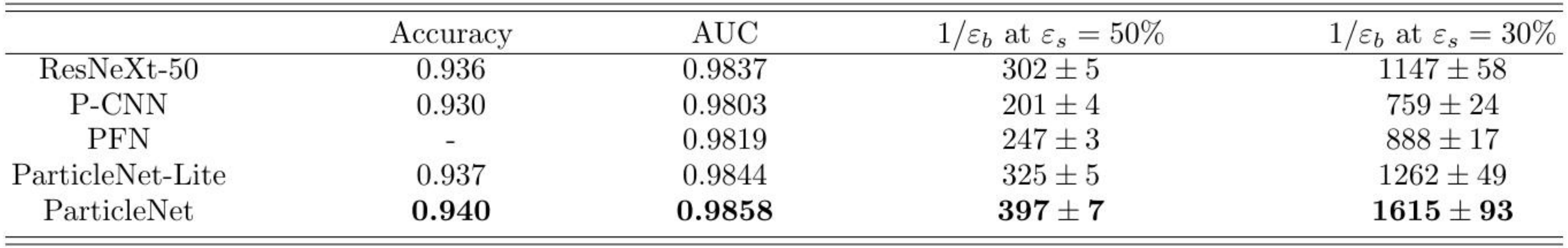
Jet Tagging via Particle Clouds

EXAMPLE: CLASSIFICATION WITH GNNS

In tests, ParticleNet outperforms other methods in accuracy metrics.



	Parameters	Time (CPU) [ms]	Time (GPU) [ms]
ResNeXt-50	1.46M	7.4	0.22
P-CNN	348k	1.6	0.020
PFN	82k	**0.8**	**0.018**
ParticleNet-Lite	**26k**	2.4	0.084
ParticleNet	366k	23	0.92

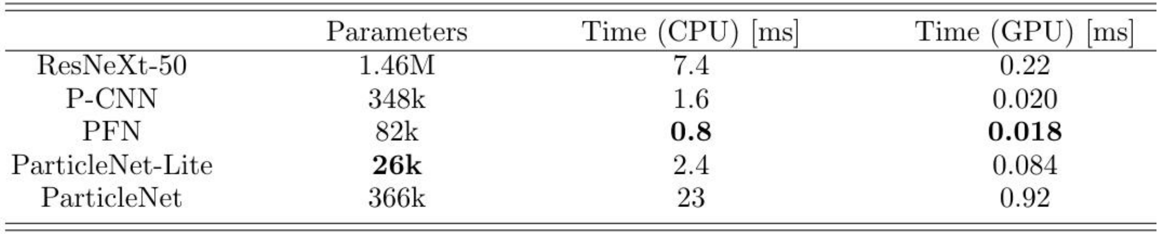


	Accuracy	AUC	$1/\varepsilon_b$ at $\varepsilon_s = 50\%$	$1/\varepsilon_b$ at $\varepsilon_s = 30\%$
ResNeXt-50	0.936	0.9837	302 ± 5	1147 ± 58
P-CNN	0.930	0.9803	201 ± 4	759 ± 24
PFN	-	0.9819	247 ± 3	888 ± 17
ParticleNet-Lite	0.937	0.9844	325 ± 5	1262 ± 49
ParticleNet	**0.940**	**0.9858**	$\mathbf{397 \pm 7}$	$\mathbf{1615 \pm 93}$

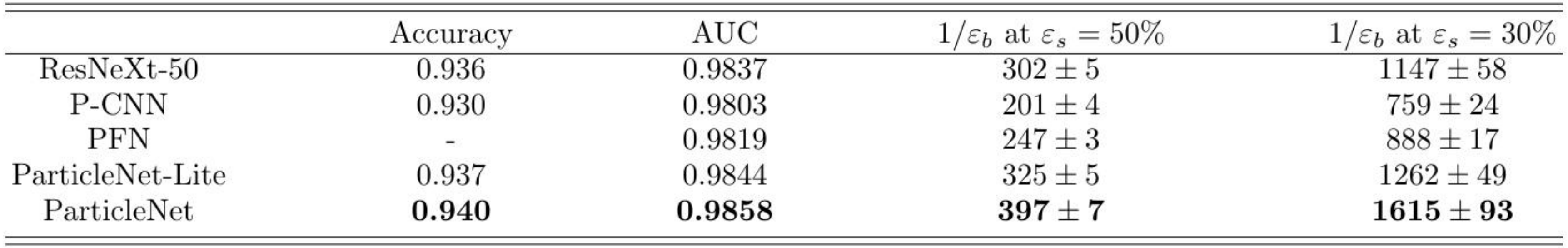
Jet Tagging via Particle Clouds

EXAMPLE: CLASSIFICATION WITH GNNS

But, it is hard to compete against vendor optimized convolutional kernels for performance



	Parameters	Time (CPU) [ms]	Time (GPU) [ms]
ResNeXt-50	1.46M	7.4	0.22
P-CNN	348k	1.6	0.020
PFN	82k	**0.8**	**0.018**
ParticleNet-Lite	**26k**	2.4	0.084
ParticleNet	366k	23	0.92



	Accuracy	AUC	$1/\varepsilon_b$ at $\varepsilon_s = 50\%$	$1/\varepsilon_b$ at $\varepsilon_s = 30\%$
ResNeXt-50	0.936	0.9837	302 ± 5	1147 ± 58
P-CNN	0.930	0.9803	201 ± 4	759 ± 24
PFN	-	0.9819	247 ± 3	888 ± 17
ParticleNet-Lite	0.937	0.9844	325 ± 5	1262 ± 49
ParticleNet	**0.940**	**0.9858**	$\mathbf{397 \pm 7}$	$\mathbf{1615 \pm 93}$

Jet Tagging via Particle Clouds

EXAMPLE: THEORY WITH GNNS

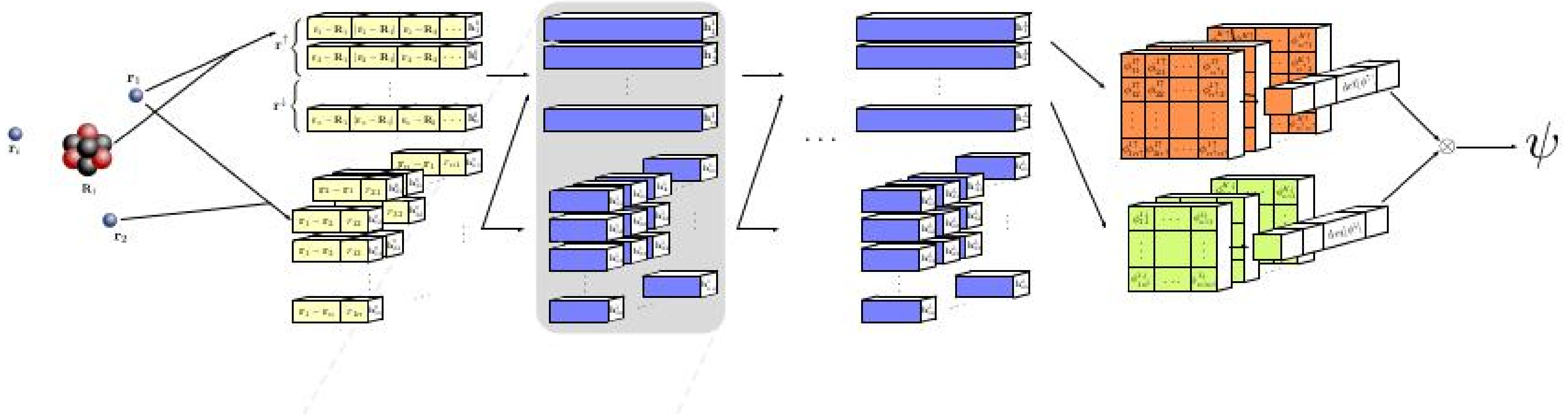
- Variational Monte Carlo is a numerical technique for solutions to the Schrodinger Equation:
  - Requires an “ansatz” aka a trial wavefunction that can be optimized.
  - The wavefunction must obey physial principles (twice differentiable, continous, antisymmetric under exchange of Fermions)
  - The wavefunction must be sufficiently general to capture all the physics of the system.

$$\frac{\langle \Psi_V | H | \Psi_V \rangle}{\langle \Psi_V | \Psi_V \rangle} = E_V \geq E_0$$

Graph Neural Networks are an ideal candidate for building an ansatz.

https://github.com/google-deepmind/ferminet

EXAMPLE: FERMINET

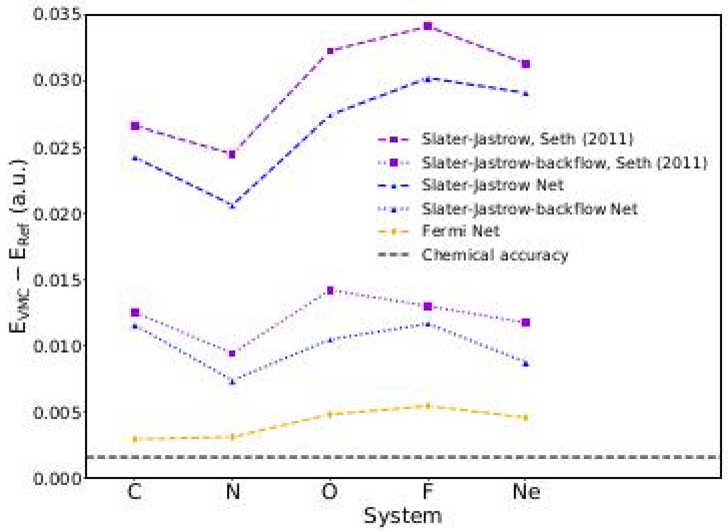


Ferminet solves molecular physics by encoding electron locations in a dynamic graph.

https://github.com/google-deepmind/ferminet

EXAMPLE: FERMINET

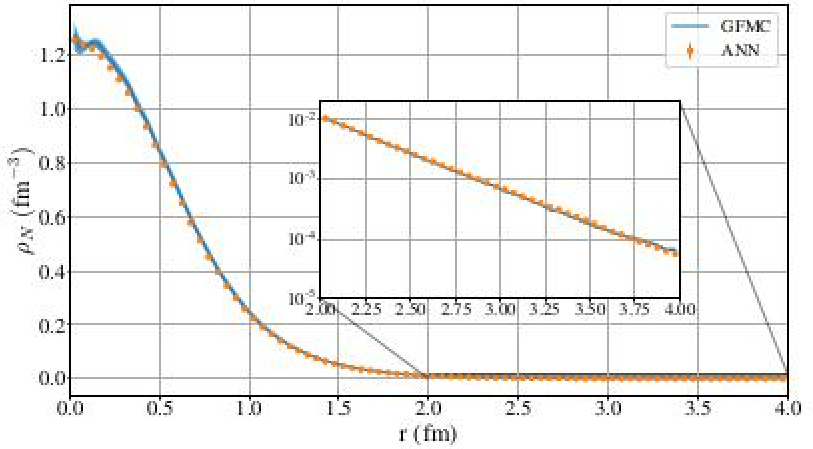
- Nuclei locations are “constants” while electron positions are inputs to the network.
- Two-body correlations directly learned by the network.
- Ferminet beats traditional methods of encoding wavefunctions by applying permutation invariant methods to the particle positions.
- Antisymmetry is enforced through the use of Slater Determinants - the permutation invariance of GNNs is critical to constructing a physical wavefunction.



https://github.com/google-deepmind/ferminet

EXAMPLE: VARIATIONAL MONTE CARLO

$$H_{LO} = -\sum_i \frac{\vec{\nabla}_i^2}{2m_N} + \sum_{i<j} (C_1 + C_2\, \vec{\sigma_i} \cdot \vec{\sigma_j})\, e^{-r_{ij}^2 \Lambda^2/4}$$
$$+ D_0 \sum_{i<j<k} \sum_{\text{cyc}} e^{-(r_{ik}^2 + r_{ij}^2)\Lambda^2/4}, \qquad ($$



Simpler than FermiNet: dump a bunch of protons and neutrons together in an all-to-all GNN and solve for the minimum energy.

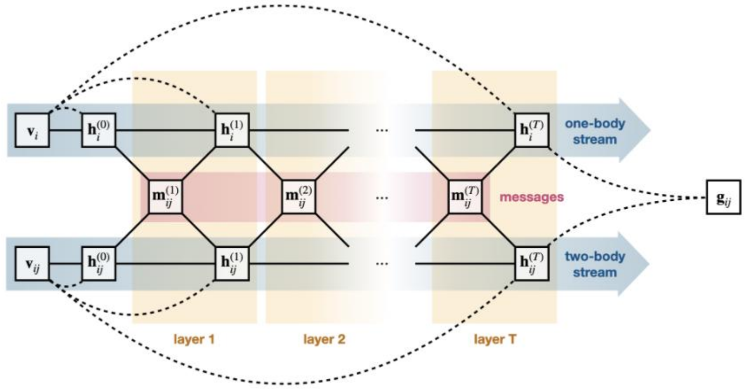
$$\mathcal{F}(\mathbf{x}_1, \ldots, \mathbf{x}_A) = \rho_{\mathcal{F}}\left(\sum_{\mathbf{x}_i} \phi_{\mathcal{F}}(\mathbf{x}_i)\right) \qquad \mathcal{F} = \mathcal{U}, \mathcal{V}\,.$$

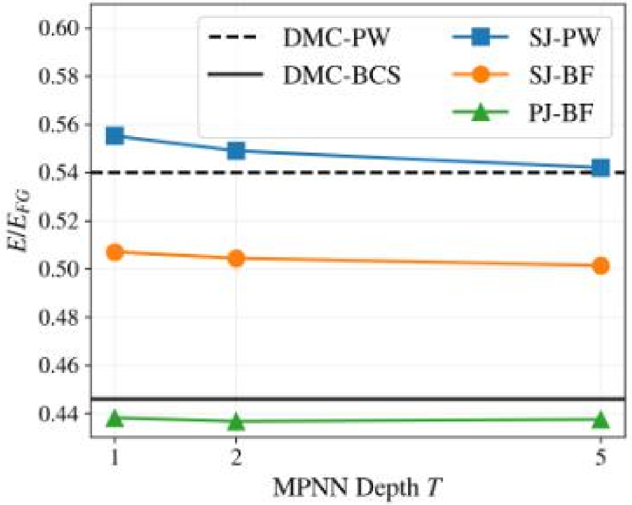
	Λ	VMC-ANN	VMC-JS	GFMC	GFMC$_c$
^{2}H	4 fm^{-1}	−2.224(1)	−2.223(1)	−2.224(1)	-
	6 fm^{-1}	−2.224(4)	−2.220(1)	−2.225(1)	-
^{3}H	4 fm^{-1}	−8.26(1)	−7.80(1)	−8.38(2)	−7.82(1)
	6 fm^{-1}	−8.27(1)	−7.74(1)	−8.38(2)	−7.81(1)
^{4}He	4 fm^{-1}	−23.30(2)	−22.54(1)	−23.62(3)	−22.77(2)
	6 fm^{-1}	−24.47(3)	−23.44(2)	−25.06(3)	−24.10(2)

https://arxiv.org/abs/2007.14282

EXAMPLE: VARIATIONAL MONTE CARLO

"Ultra cold Fermi Gas" - apply a message passing GNN to transform electron positions and features (spin) before applying an antisymmetric function ("pfaffian").





https://arxiv.org/pdf/2305.08831.pdf

WRITING A GNN FOR YOURSELF

- Use the libraries out there!
  - Pytorch Geometric
    - Dynamic dispatch (just like torch), fast and easy to use
    - Probably leaves performance on the table
  - Jraph (JAX)
    - Disclaimer: I haven’t used it!
    - Probably very fast, due to JIT compilation
    - Pads data to enable static runtime shapes (required in JAX).
  - TF GNN
    - It also exists?

WRAPPING UP GNNS

- If you want to apply machine learning and your dataset *doesn't* have rectangular structure, Graph Neural Networks can be a powerful tool.
- Compared to CNNs, there is much less consensus about what makes a good GNN.
  - Part of this is because the variety in graphs is much much larger than in image data!
- What I've covered is really the tip of the iceberg:
  - Graph Transformers
  - Graph Residual Networks
  - Going Deeper with GNNs
- All of the challenges with CNNs may still apply! (data/mc in particular)

THANK YOU FOR LISTENING, AND I HOPE IT WAS USEFUL!

QUESTIONS?