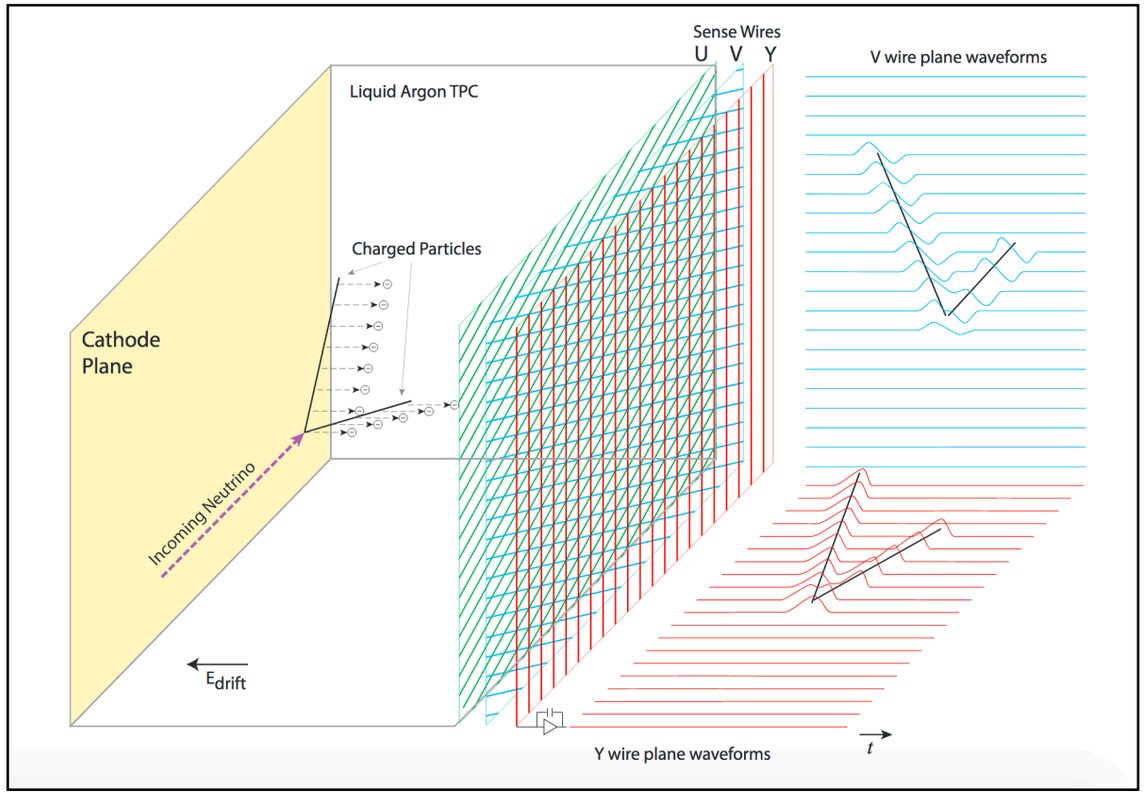
Graph convolutional networks for LArTPCs

Jeremy Hewes
Exa.TrkX All Hands Meeting
7th April 2020

Introduction

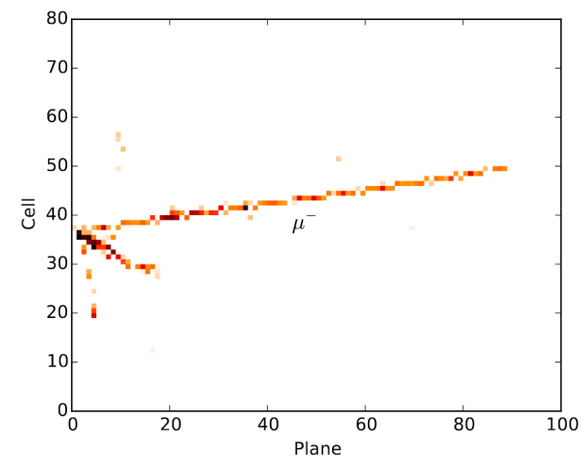
- Liquid Argon Time Projection Chambers (LArTPCs) are currently a very important detector technology for neutrino physics.
  - At FNAL: MicroBooNE, Icarus, SBND.
  - Future: DUNE (70kT LArTPC deep underground, plus near detector).

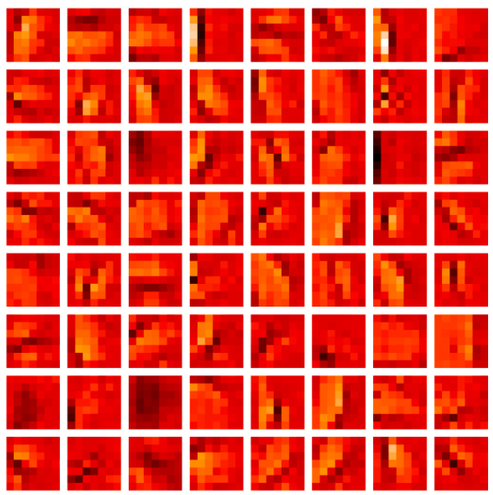
- Charged particles ionize liquid argon as they travel.
- Ionisation electrons drift due to HV electrode field, and are collected by anode wires.
- Wire spacing ~3mm – produce **high-resolution images**.



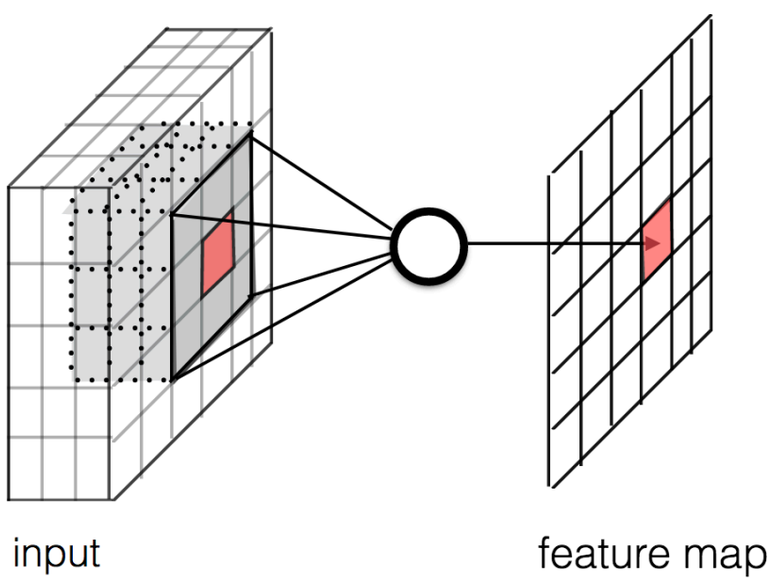
CNNs in neutrino physics

- Convolutional neural networks show great promise in image classification over the past decade.
- Most neutrino detector technologies naturally provide pixel maps which can be classified using CNNs.
- Examples: NOvA, MicroBooNE, DUNE.





arXiv:1604.01444



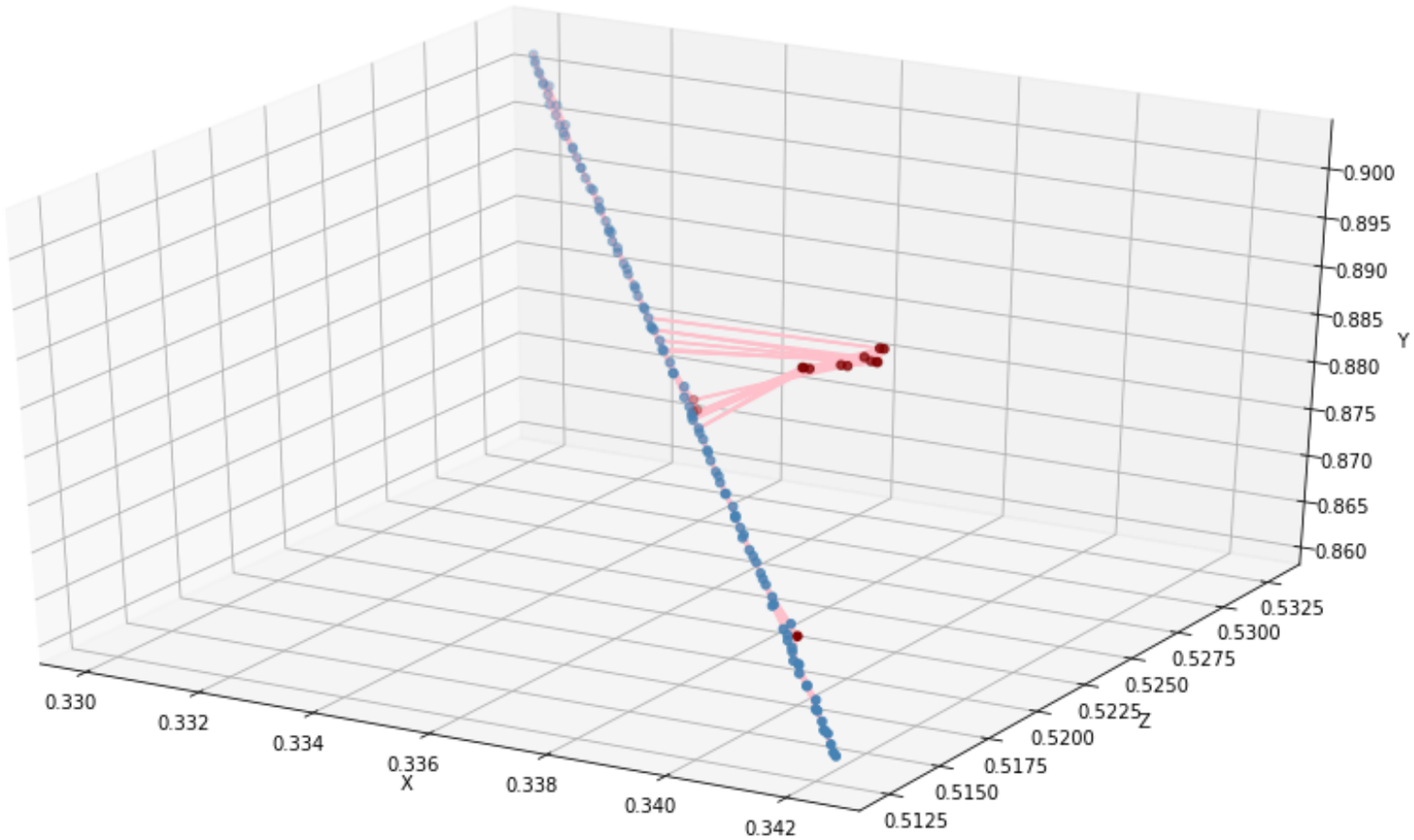
- Issues with this approach:
  - **Dense** representation of **sparse** data.
  - Operate over mostly empty space!
  - Need to transform 3D representation into voxels.
- GNNs can work with reconstructed spacepoints natively.

Why graph networks?

- Convolutional neural networks are a heavily used computational technique.
  - But they assume your data is assembled in a grid structure!
    - For a 2D TPC readout, this is a good assumption (wire vs time).
    - For 3D spacepoints reconstructed from hits, this is less true.
    - For other data structures, this may not be true at all.

- Investigate techniques that are able to meet our data on its own terms.
- The fact that graph networks can be applied to heterogeneous data structures offers the potential to run at multiple information scales.
  - Operate on spacepoints, then on clusters, then on the event as a whole.

Applications in DUNE



- Use **3D WireCell spacepoints** as input to network.
- Train network to **group 3D spacepoints into clusters** for downstream processing.
- Toy study: use Pandora 3D spacepoints from ProtoDUNE MC as input.

Graph construction

- Detector geometry in IceCube and CMS provide initial constraints:
  - **Quantised DOM structure** in IceCube limits number of graph nodes.
  - **Layer structure** in CMS provides natural constraint on edges (limited to adjacent detector layers).
- By contrast: ProtoDUNE MC readout window has ~50k Pandora spacepoints.
  - Some creativity required to construct a sensibly defined graph structure.
- First pass: construct **cluster-wise** graphs from Pandora PFParticles.
  - Construct graph from reconstructed spacepoints in local area, and design a graph network to classify spacepoints.
  - Filter out oversize events (>10k spacepoints), limit number of edges per node.
  - Start with publicly available HEP.TrkX network and iteratively adapt to achieve optimal performance in a TPC.

Message-passing architecture

arxiv:1810.06111

- **Edge classifier:**
  - Input for each node is the features of incoming and outgoing nodes.
  - Two multi-layer perceptrons, using Tanh and sigmoid activations.
  - Outputs sigmoid score on each edge.
- **Node classifier:**
  - Uses edge score to aggregate each node's features with incoming & outgoing edges as input.
  - Two multi-layer perceptrons with Tanh activation.
  - Produces new features for each node.

Training parameters

Objective: Binary cross-entropy
Optimiser: Adam
Learning rate: 0.001
7 model iterations
Batch size: 3

Model parameters: 26433

Memory usage: ~16GB

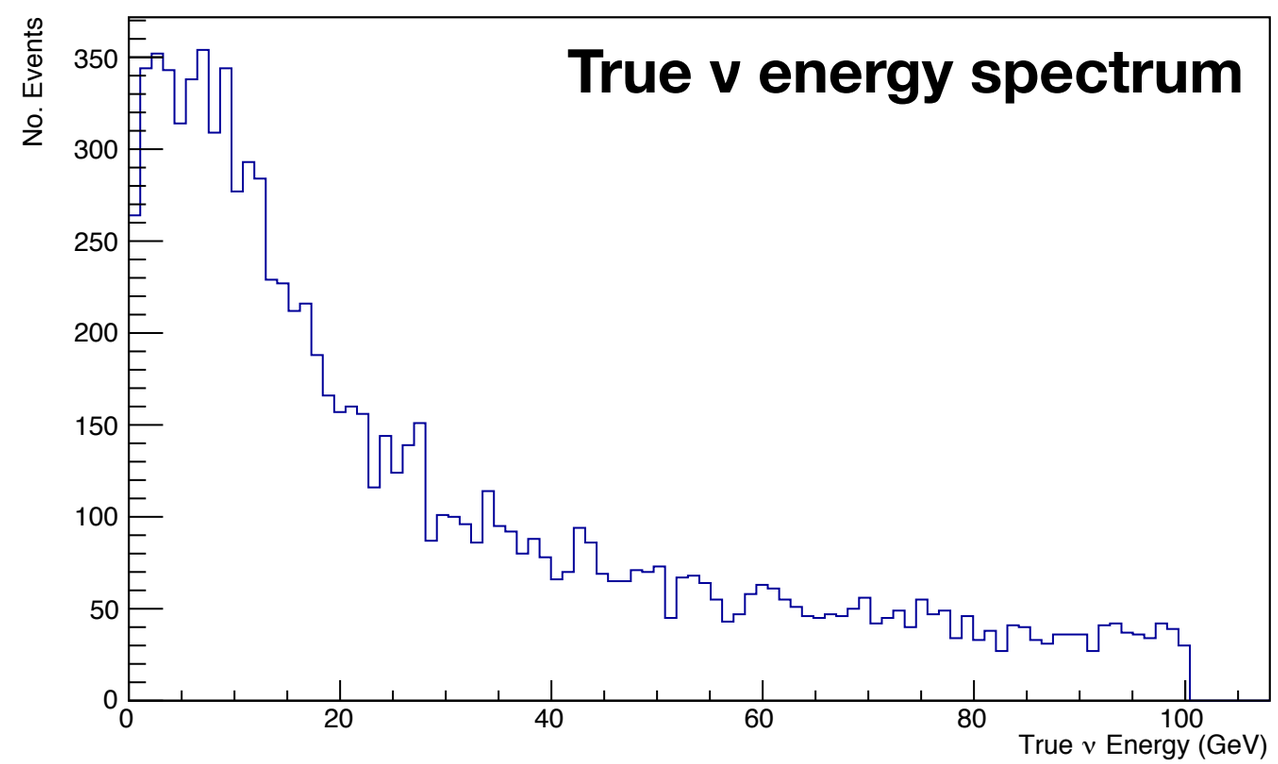
Network performance

- Some small evidence of learning over the first few epochs, but clearly **much room for improvement**!
- Succeeded in initial goal: **constructing workflow** to produce graphs and train networks in TPCs

- **Next steps**:
  - Try with a simpler problem (atmospheric interactions in DUNE 10kt FD).
    - Cluster entire readout simultaneously.
  - Train on WireCell spacepoints.
    - Less dense point clouds may prevent the need to train on cluster-wise graphs.
  - Explore entirely different graph constructions.
    - Cluster-wise graphs for particle ID?

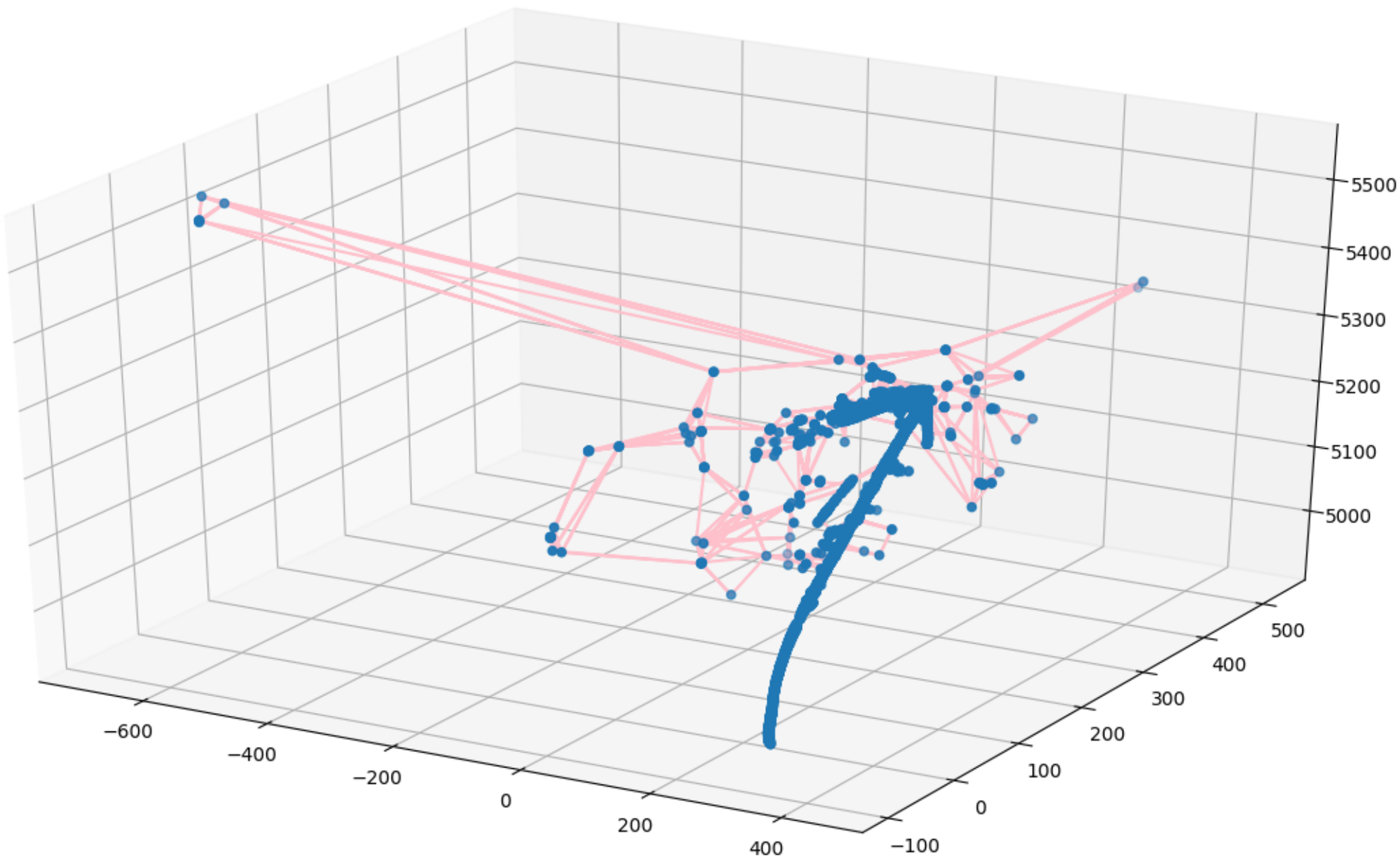
Simulation

- Simulation is **atmospheric neutrino** interactions in the **full 10kt geometry**.
- Produced three 600k MC training samples.
  - $\nu_{e,\mu,\tau}$ (each flux-swapped from initial ν_e & ν_μ flux).
  - Flatten Honda flux below few-GeV to avoid MC being completely overwhelmed by very low-energy interactions.
- Standard simulation chain, run reconstruction up to 3D spacepoint finding (**SpacePointSolver**).
- Since the principal objective of this study was clustering, focused on ν_μ interactions.



Graph inputs

- Spacepoint graph looks something like this:

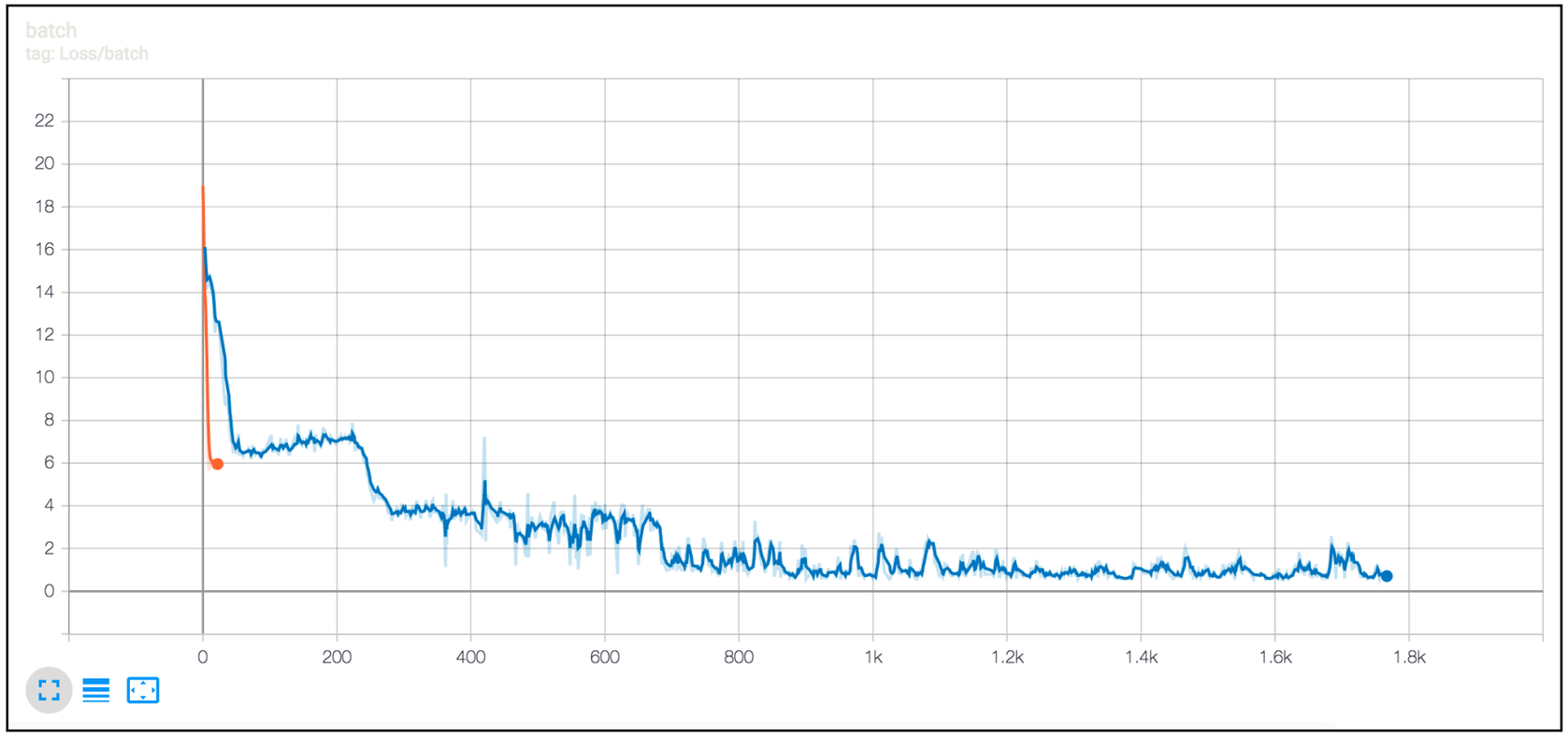


Input production

- Graphs are produced from **reconstructed spacepoints**.
  - The initial goal of this study was to cluster spacepoints from WireCell.
  - Produced graphs using both Pandora and SpacePointSolver.
    - These studies will show the latter, since they are the most direct analogue to WireCell spacepoints.
  - Construct a graph node for each spacepoint.
  - Node features include **charge of associated hits**, **xyz position**, **number of nearest neighbours**, and the **G4 track ID of the MC particle that contributed the most charge**.
    - G4 ID is obviously not used during training, but is used to construct the ground truth during preprocessing.
      - Edge label is 1 if nodes share the same G4 ID, 0 if not.
      - A more complex ground truth definition may be necessary (see later).

Training

- Training objective is **binary cross-entropy** loss on edge labels.
- When training over four GPUs simultaneously, training one epoch of ~200k graphs takes around **seven minutes**.
- **Batch size of 100** distributed across those four GPUs.
- Network trains very quickly, and then plateaus at a loss of around 0.5.

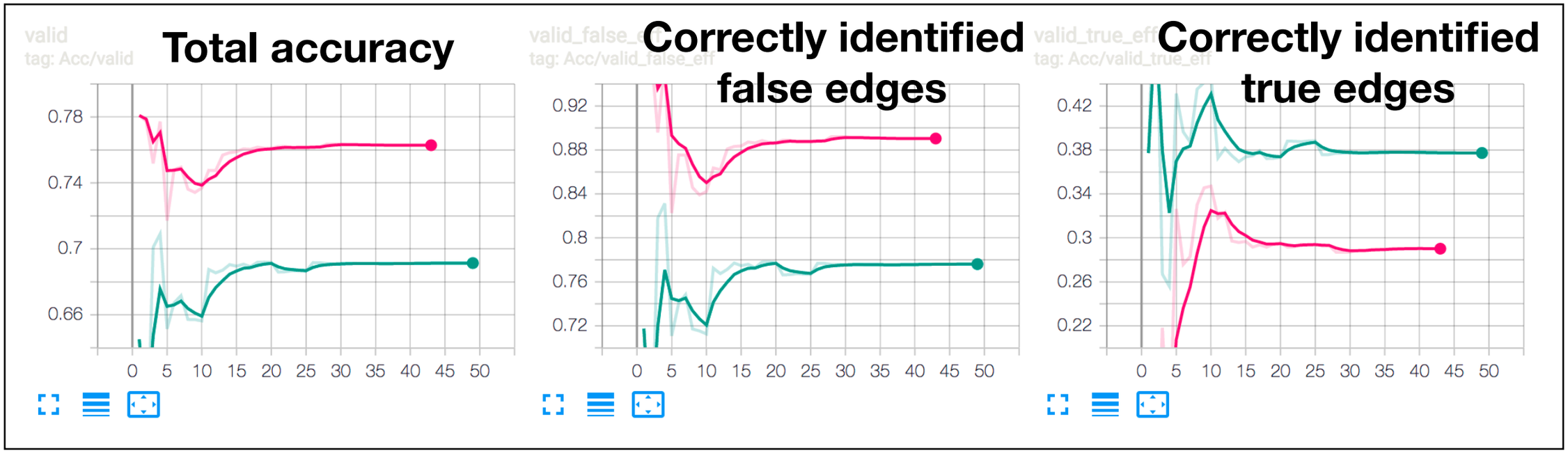


Training parameters	
Learning rate	0.0005
Optimiser	Adam
Message passing iterations	6

Performance

- Initial loss reduction was encouraging, but in this case, misleading.
  - The majority of edges in the graph are false, and so the network can achieve high accuracy by simply classifying all edges as false.
- We can prevent it from doing this by weighting the loss function during training.
  - Weight true edges up, and false edges down, to encourage the network to pay more attention to true edges.
  - This is a balancing act! Weight too heavily, and the network will learn the opposite lesson, ie. classify all edges as true.
  - Finding the right weights to get the network to sit in a middle ground.

Further tests



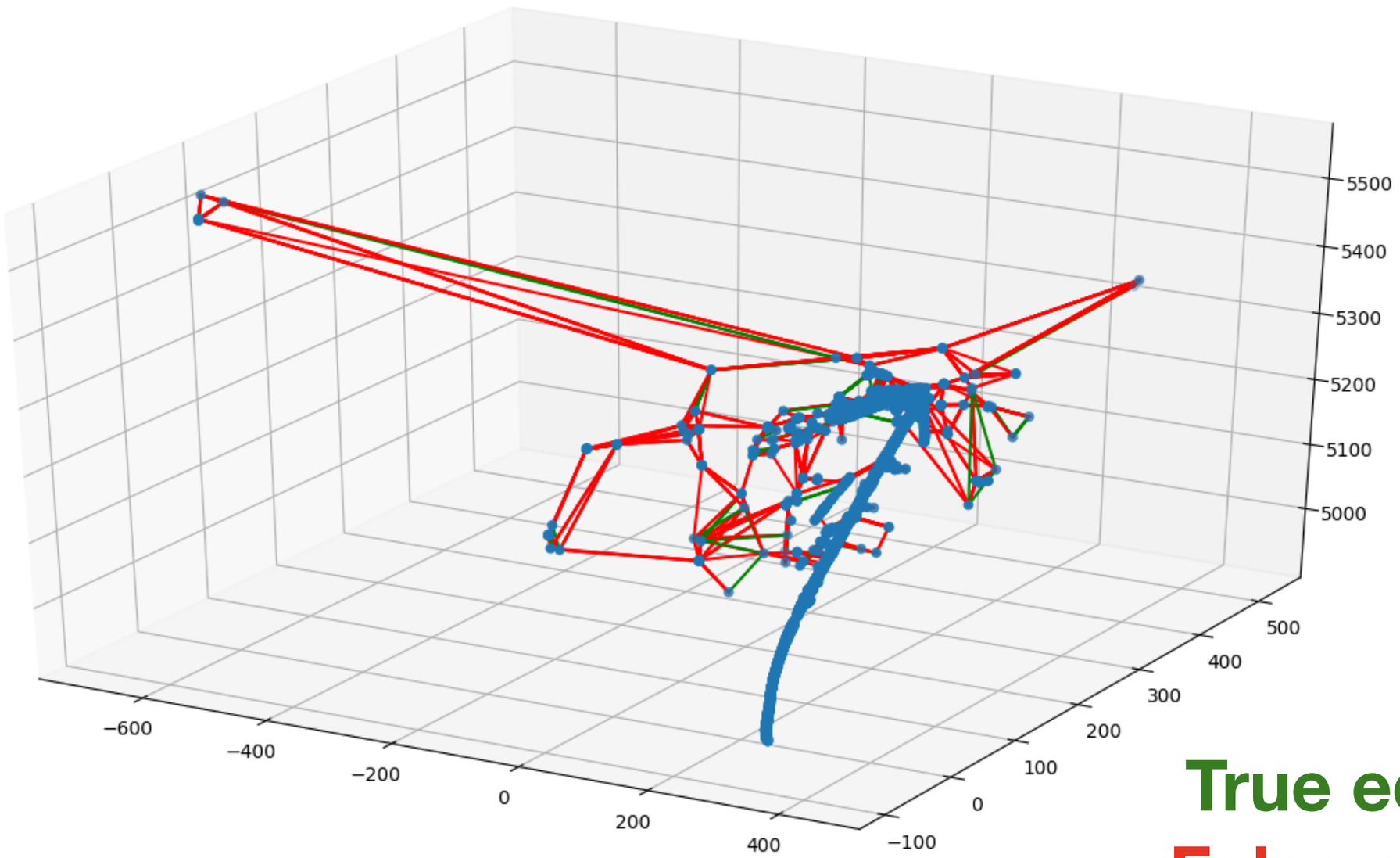
- Weighted true edges up and false edges down.
- Monitored efficiency for true and false edges independently.
- Network started to learn, but plenty of tradeoff in efficiency labelling true and false edges.

True weight: 1.5
False weight: 0.7

True weight: 1.5
False weight: 0.6

Graph analysis

- Look closer at graph definition:



True edges

False edges

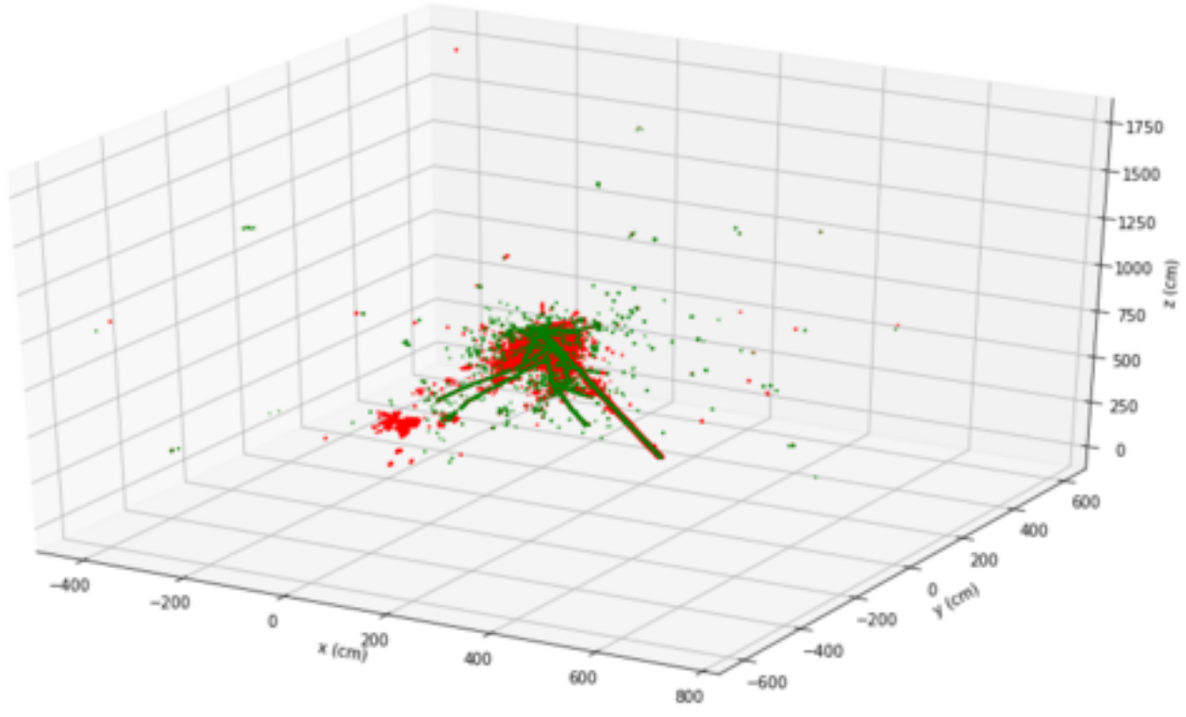
Graph analysis

- Not clear that a naïve definition of ground truth (matching true G4 ID between spacepoints) was good enough.
- From closer analysis, 20-40% of nodes in a typical graph have no true edge attached.
- Suspected G4 IDs from small particles are confusing things here.
- Two important lessons from this first pass:
  - Don’t try to do too much at once.
  - Thinking carefully about how to define ground truth is critical.
- Using Exa.TrkX networks as a starting point was also very instructive in understanding the ways in which the needs of a LArTPC differ.

Spacepoint deghosting

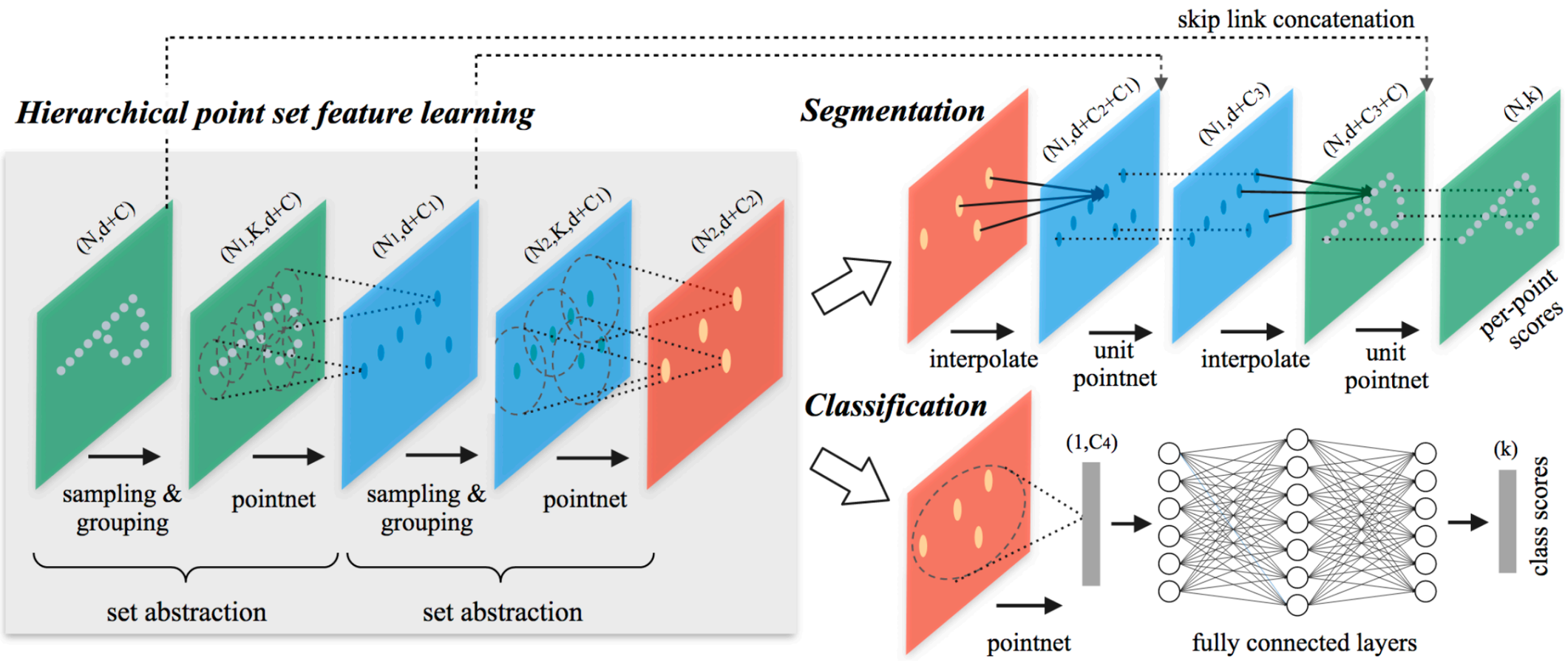
- Took a different approach to developing graph networks: **deghosting 3D spacepoints**.
- Take a high-efficiency but low-purity spacepoint reconstruction, and **improve the purity by rejecting "false" spacepoints**.

- Definition of ground truth:
  - Require that all associated 2D hits derive the majority of their energy from the **same true particle**.
    - For "true" spacepoints, this also provides a true particle association.
  - Calculate the **point of closest approach** between the reconstructed spacepoint and the true particle trajectory.
  - Sort spacepoints in ascending distance order, and mark spacepoints as true provided **hits are not reused**.



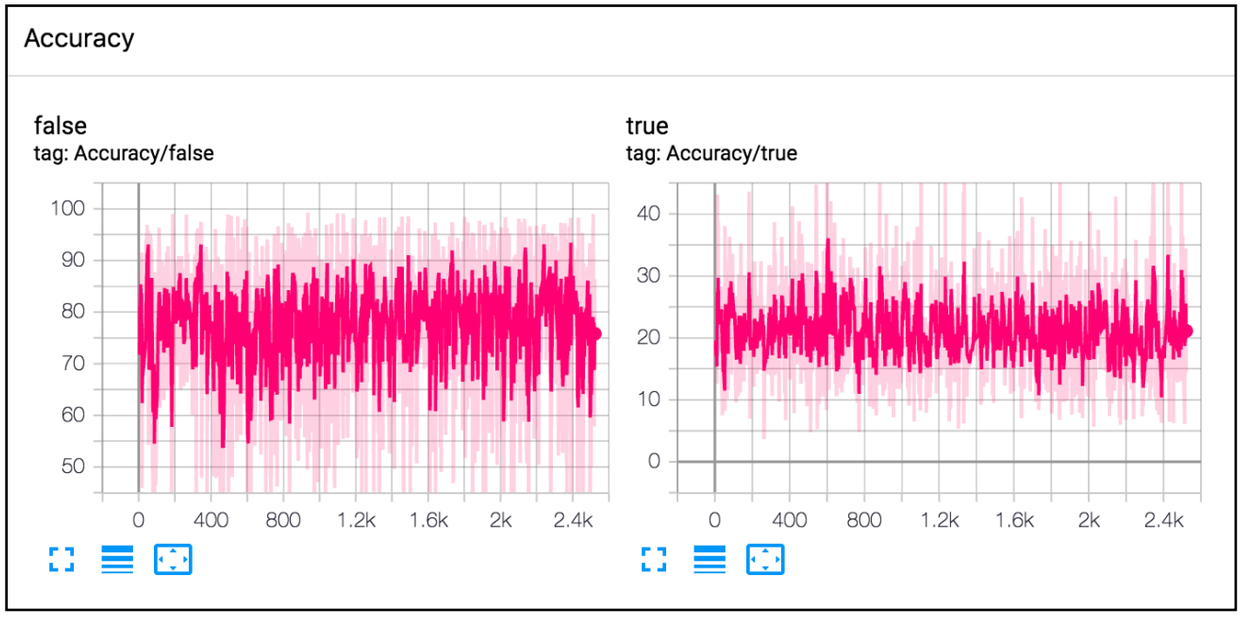
PointNet++ network

- One graph network that may be effective when applied to this problem is called **PointNet++** (arxiv:1706.02413).
  - This network is specifically designed to operate on point clouds.
  - Utilises **set abstraction** to aggregate local features, similar to a U-net for CNNs.



Training results

- PointNet++ is not able to run effectively in pytorch-geometric yet.
- Convolutions are fast, but the utilities for performing upsampling and downsampling are not optimised for large graphs.
  - This makes PointNet++ prohibitively slow to run (for now).
- In the interim, attempted training with a message-passing network similar to ones I've used in the past, but with a node classification output.



- Node classification network doesn't perform well out of the box.
- I was somewhat expecting this, since the problem of drawing edges in a dense point cloud means the graph is not fully connected
- I also suspect the ground truth definition is again also to blame.

Current and future plans

New training samples

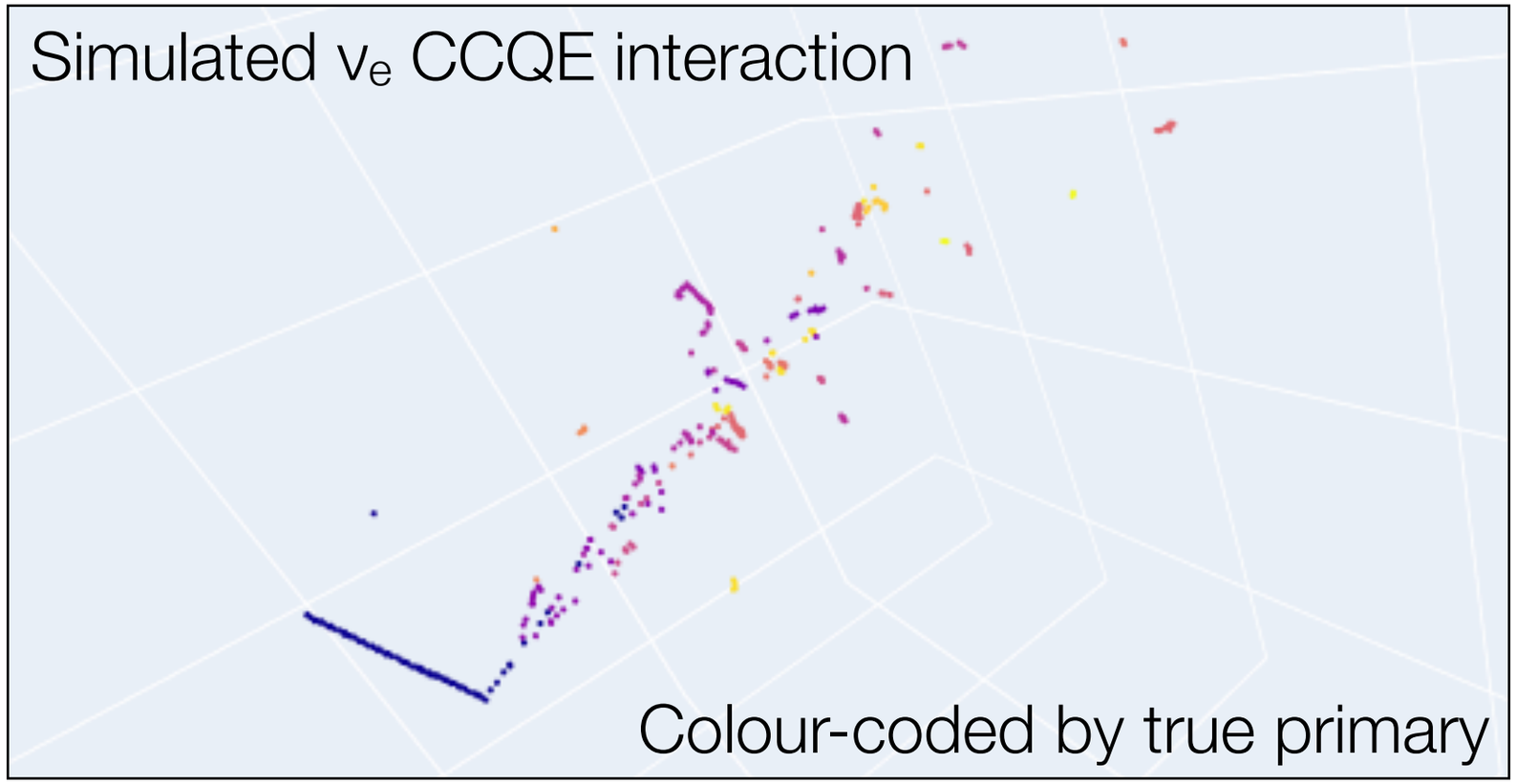
- Previously working with a training sample of **high-energy neutrino interactions in the DUNE far detector**.
  - Made this decision because these types of interaction are the ones we're most interested in from a physics perspective.
  - As event displays on previous slides have shown, these events tend to be large, messy interactions with many overlapping particles.
- Default GEANT4 behaviour in LAr simulation is to discard any simulated particles associated with EM processes.
- Causes many issues with defining a ground truth when information is missing.

SHOWER DAUGHTERS

Any particle produced by:
Pair production
Compton scattering
Bremsstrahlung
Photoelectric effect
Annihilation

New training samples

- Produced samples of **CCQE neutrino interactions** in the DUNE far detector.
- Sample of clean, low-occupancy events to use for testing different techniques.
- Keep all “shower daughters” in order to have access to all information when defining ground truth.

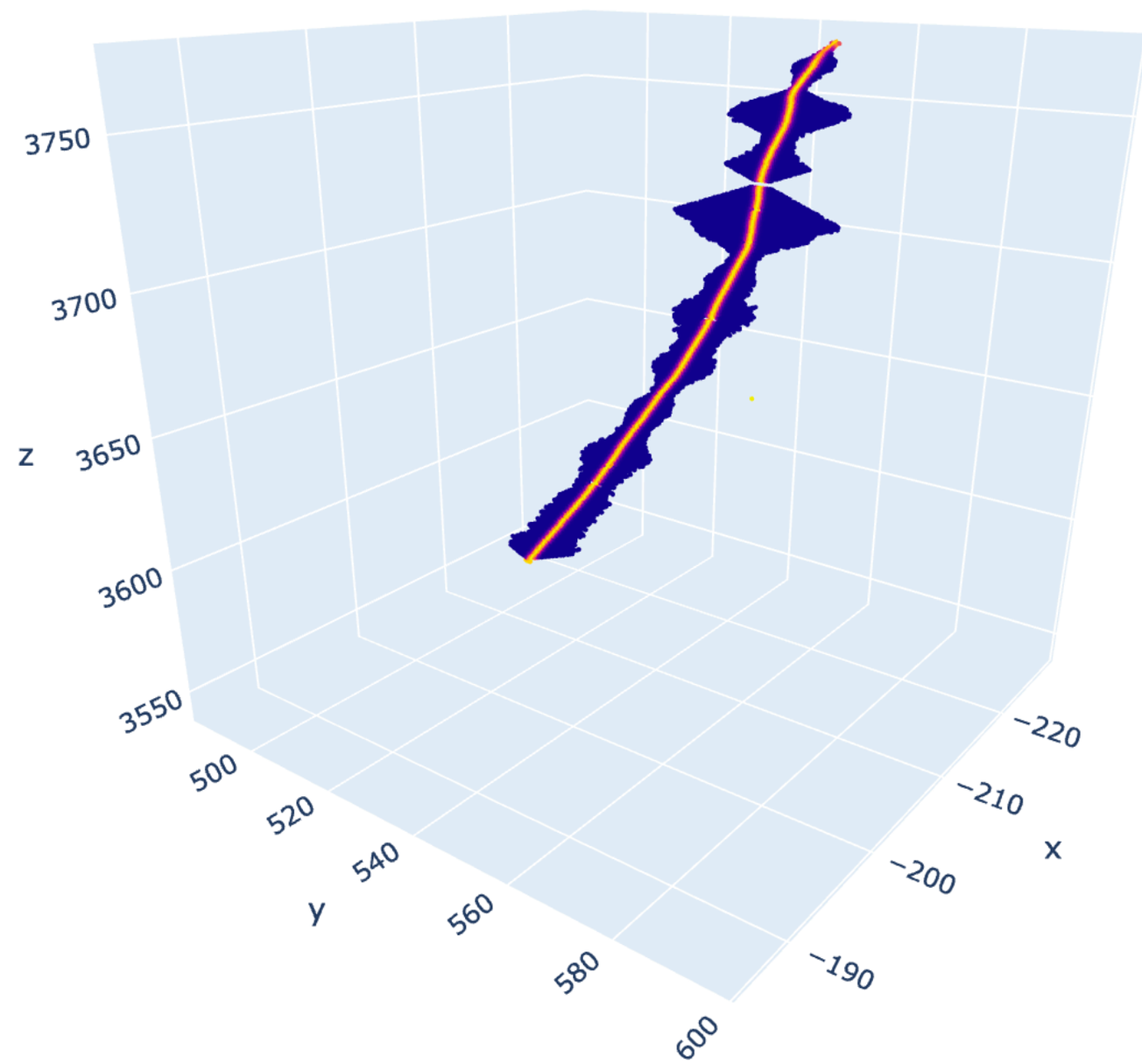


Fuzzy ground truth

- I suspect a lot of the problems with training thus far have been a result of using **binary classification** for the ground truth.
- We use multiple factors to decide whether an edge should be considered true or false, and then give the network the difficult task of trying to reverse-engineer that logic.
- Solution: stop using binary cross-entropy loss, which expects a binary ground truth (0 or 1), and instead use a **categorical cross-entropy** that allows the truth to be a probability between 0 and 1.
- We can directly **give the network the parameter we care about** (in this case, point of closest approach between spacepoint and true particle trajectory) and **perform decision-making on the output**.
  - Rather than arbitrarily cutting on distance to define a binary ground truth, train a network with a smoothly varying truth and then cut on the model's output.

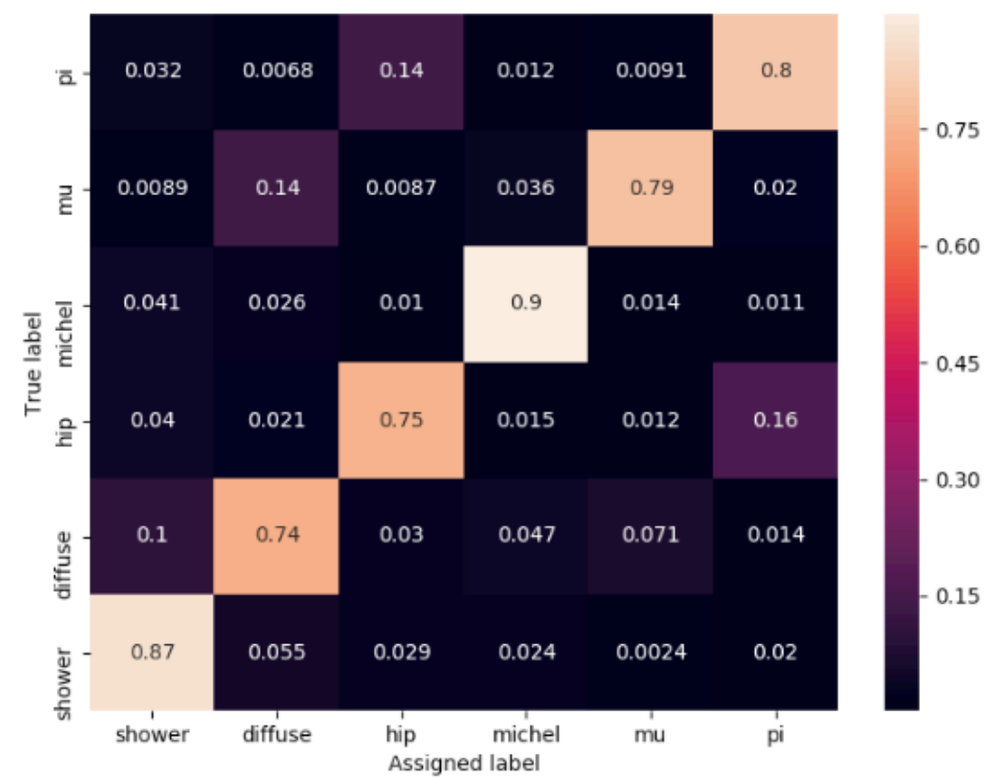
Spacepoint deghosting

- Raw deghosting truth for spacepoints is distance d between reconstructed point and true trajectory.
- Apply exponential decay to produce ground truth $y = e^{-\frac{d}{l}}$, where l is a configurable characteristic length (1cm in this example).
- Use categorical cross-entropy loss with floating point truth.



Spacepoint segmentation

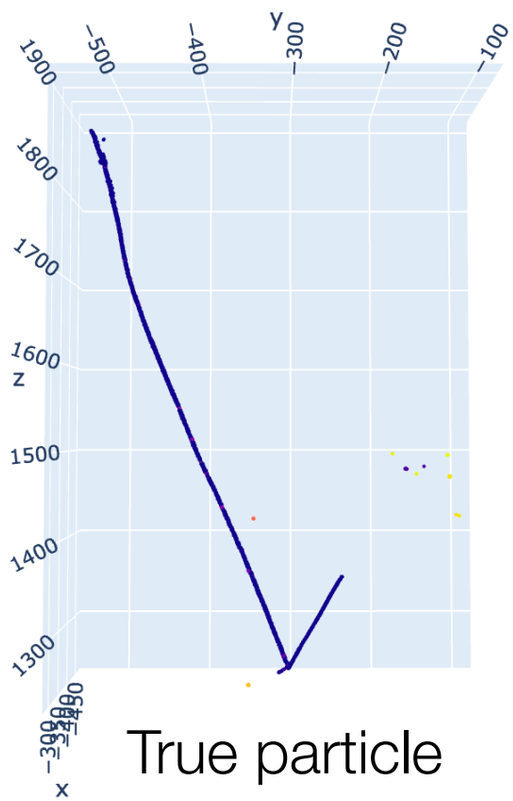
- Promising early results using sparse segmentation (CNN-based approach) for particle ID in ProtoDUNE.
  - Based on Kazu et al's paper (arxiv:1903.05663).
- **Segment** voxels by **particle type**:
  - μ, π, κ, HIP, shower, Michel e, etc.
- Take existing work on ground truth, developed in context of ProtoDUNE, and apply to DUNE far detector.
- Instead of voxelising spacepoints and utilising a CNN, define ground truth per individual spacepoints.
- Test GNN ability to segment different particle types.



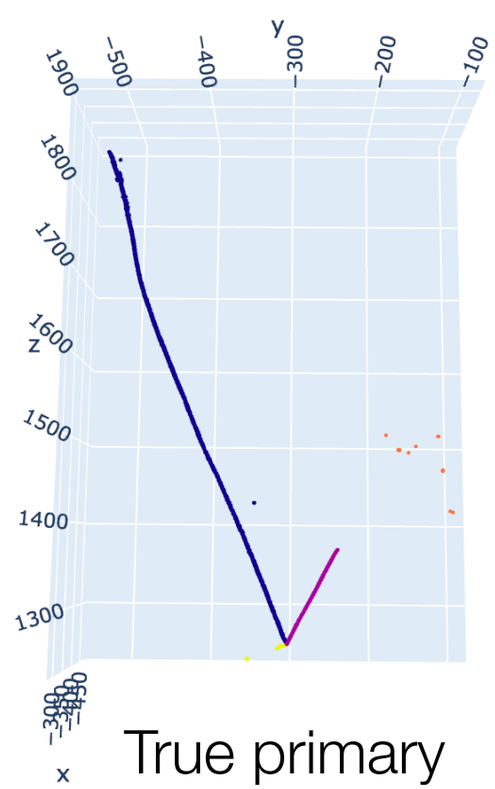
C. Sarasty – ProtoDUNE indico

Hierarchical particle clustering

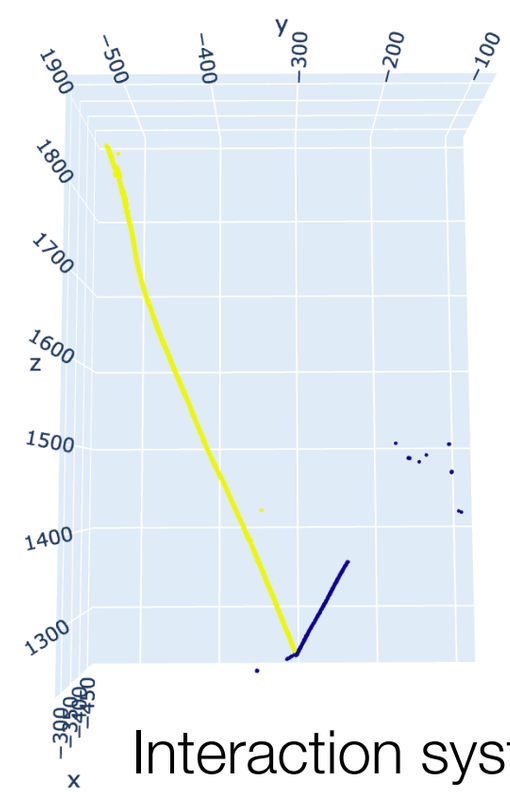
- Longer-term goal is to build towards a hierarchical cluster-based technique.
- Instead of edge classification between individual spacepoints, perform graph edge classification between spacepoints and higher-level objects (true particle, true primary, interaction system).



True particle



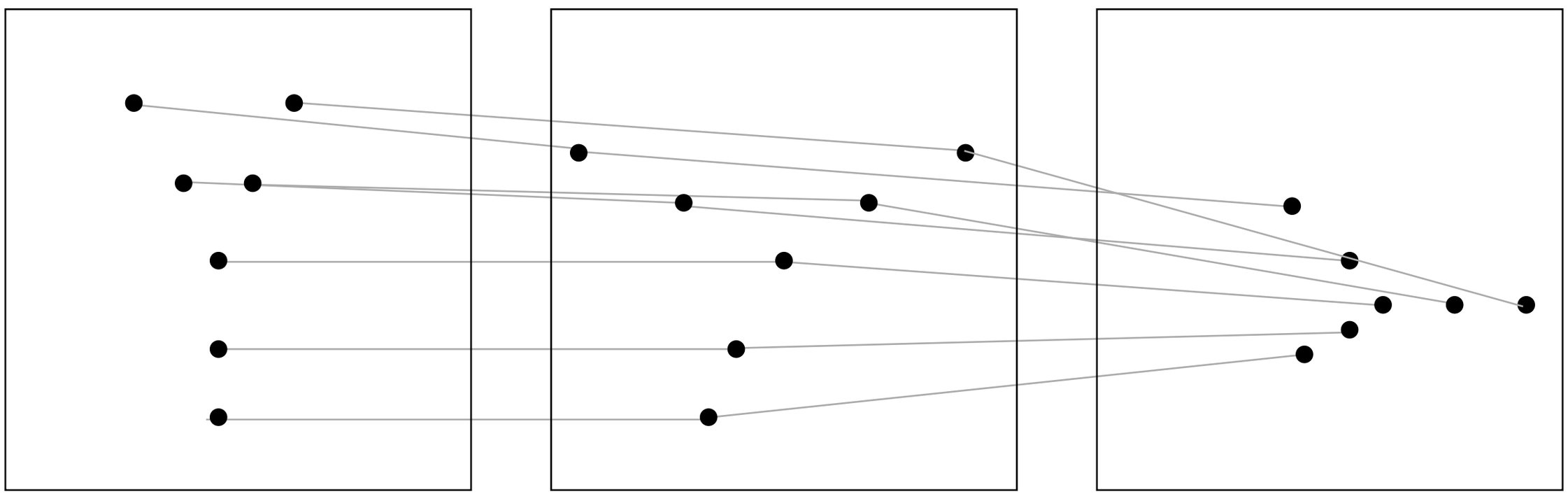
True primary



Interaction system

2D hit matching

- Working natively in 3D relies upon a reliable method for reconstructing 2D hits into 3D.
- Instead of relying on 3D spacepoint reconstruction, consider starting with 2D hits on three wire planes, and form connections between those.
- With a graph structure, is it possible to form objects without reconstructing individual 3D spacepoints?
- Unlike point cloud, hit time and wire plane naturally reduce number of potential edges.



Summary

- Investigating development of bottom-up techniques for graph network reconstruction in LArTPCs.
- Initial efforts inspired by HEP.TrkX did not bear much fruit.
- Conclusions: simpler topologies, more point-cloud-based techniques.
- Current & future steps:
  - Spacepoint deghosting.
  - Node segmentation.
  - Hierarchical graph reconstruction.
  - Hit-matching in 2D representations.