Machine Learning

Saúl Alonso-Monsalve

CERN

T2K Cross-Section Workshop

18/10/19

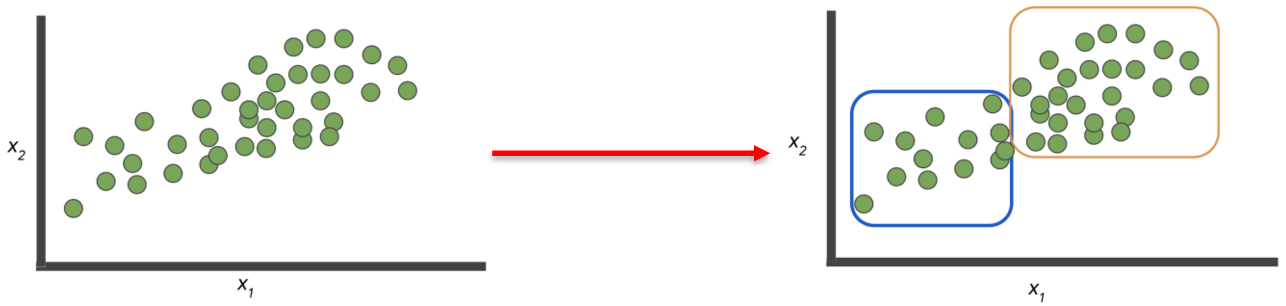
Overview

- Introduction to Machine Learning.
- Practical example.
  - Binary classification problem (logistic regression).
- Neural networks and deep learning.
- Deep Learning in neutrino experiments.
  - Deep Underground Neutrino Experiment (DUNE).
  - Tokai to Kamioka (T2K) – SuperFGD.

Machine Learning

- **Supervised learning**: we are given a dataset and already know what the correct output should look like.
  - **Regression problems**: we are trying to predict results within a continuous output.
    - Example: predicting house prices based on house size.
  - **Classification problems**: We are trying to predict results in a discrete output.
    - Example: tagging photos as '*cat*' or '*dog*.'

- **Unsupervised learning**: we try to approach problems with little or no idea what results should look like.
    - Example*: identifying meaningful patterns in 2D data.



*https://developers.google.com/machine-learning/problem-framing/cases

Example: binary classification (I)

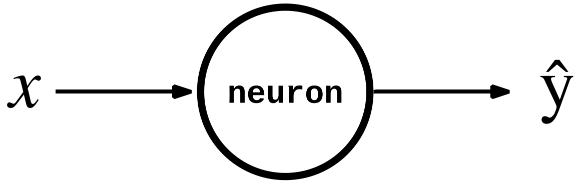
- **Dataset:**

Model	x (weight)	y (0=car, 1=motorcycle)
Renault Megane	1.175 tonnes	0
Yamaha YZF-R1	0.199 tonnes	1
MINI Cooper	1.360 tonnes	0
Ford C-MAX	1.550 tonnes	0
Kawasaki Ninja H2	0.240 tonnes	1
…	…	…

- **Goal:** tag each vehicle as '*car*' or '*motorcycle*' based on the vehicle weight.
    - Given x, we want to predict $\hat{y} = P(y = 1 \mid x)$, where $0 \leq \hat{y} \leq 1$.

Example: binary classification (II)

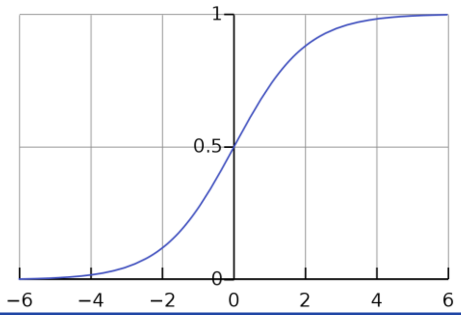
- **Goal:** tag each vehicle as '*car*' or '*motorcycle*' based on the vehicle weight.
  - Given x, we want to predict $\hat{y} = P(y = 1 \mid x)$, where $0 \leq \hat{y} \leq 1$.
  - Output: $\hat{y} = \sigma(wx + c)$



$$\begin{tikzcd} x \arrow[r] & \boxed{\text{neuron}} \arrow[r] & \hat{y} \end{tikzcd}$$

  - Parameters: $w, c \in \mathbb{R}$.
  - Activation function: sigmoid (σ).
    - $\sigma(z) = \frac{1}{1+e^{-z}}$

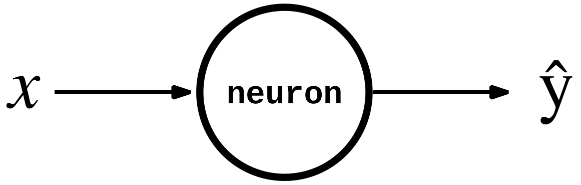
Model	x	y
Renault Megane	1.175 tonnes	0
Yamaha YZF-R1	0.199 tonnes	1
MINI Cooper	1.360 tonnes	0
Ford C-MAX	1.550 tonnes	0
Kawasaki Ninja H2	0.240 tonnes	1
…	…	…



- If z is a large positive number,
  - $\sigma(z) = \frac{1}{1+e^{-\infty}} \approx \frac{1}{1+0} \approx 1$
- If z is a large negative number,
  - $\sigma(z) = \frac{1}{1+e^{\infty}} \approx \frac{1}{1+\infty} \approx 0$

Example: binary classification (III)

- **Methodology:**
  1. Randomly initialise the parameters w, c.
  2. **Forward propagation**:
     - Select a training example.
     - Calculate output $\hat{y}$.
     - Calculate error (loss).
  3. **Backward propagation**:
     - Compute partial derivatives of the loss.
       - With respect to w and c.
     - Update w anc c.
     - **Go back to 2**.



$$\hat{y} = \sigma(wx + c)$$

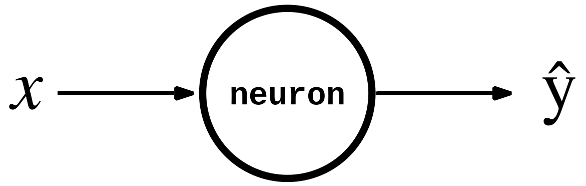
Update the model parameters (w, c) based on the error.

Example: binary classification (IV)

- **Steps:**
  - Randomly initialise the parameters w, c.
    - i.e., $w = 0.5$, $c = 0.5$.
  - **Forward propagation**:
    - Calculate output $\hat{y}$ using the first training example (x= 1.175, Renault Megane):
      - $\hat{y} = \sigma(wx + c) = \frac{1}{1+e^{-(wx+c)}} = \frac{1}{1+e^{-(0.5\times1.175+0.5)}} = 0.747911$
    - Calculate error (how good the prediction is?):
      - True output: $y = 0$
      - Predicted output: $\hat{y} = 0.747911$
      - Error (loss function):

$$\mathcal{L}(\hat{y}, y) = -(y \log \hat{y} + (1-y)\log(1-\hat{y}))$$
$$= -(\cancel{0 \log 0.747911} + (1-0)\log(1-0.747911)) = 1.37797$$

Model	x	y
Renault Megane	1.175 tonnes	0
Yamaha YZF-R1	0.199 tonnes	1
…	…	…



$\hat{y} = \sigma(wx + c)$

Example: binary classification (V)

- **Steps:**
  - **Backward propagation**:
    - Compute partial derivatives of the loss (with respect to w and c).

$$\frac{\partial \mathcal{L}(\mathrm{w,c})}{\partial w} = \frac{\partial}{\partial w}\left(-\log\left(1-\frac{1}{1+e^{-(wx+c)}}\right)\right) = \frac{xe^{c+wx}}{e^{c+wx}+1} = \frac{1.175\ \times\ e^{(0.5+0.5\times 1.175)}}{e^{(0.5+0.5\times 1.175)}+1} = 0.878795$$

$$\frac{\partial \mathcal{L}(\mathrm{w,c})}{\partial c} = \frac{\partial}{\partial c}\left(-\log\left(1-\frac{1}{1+e^{-(wx+c)}}\right)\right) = \frac{e^{c+wx}}{e^{c+wx}+1} = \frac{e^{(0.5+0.5\times 1.175)}}{e^{(0.5+0.5\times 1.175)}+1} = 0.747911$$

- Update w anc c (using a learning rate α of 0.1) ← α is always positive

$$w \rightarrow w\ - \alpha\frac{\partial \mathcal{L}(\mathrm{w,c})}{\partial w} = 0.5\ - 0.1 \times 0.878795 = 0.4121205$$

$$c \rightarrow c\ - \alpha\frac{\partial \mathcal{L}(\mathrm{w,c})}{\partial c} = 0.5\ - 0.1 \times 0.747911 = 0.4252089$$

- Calculate ŷ again (using updated parameters):

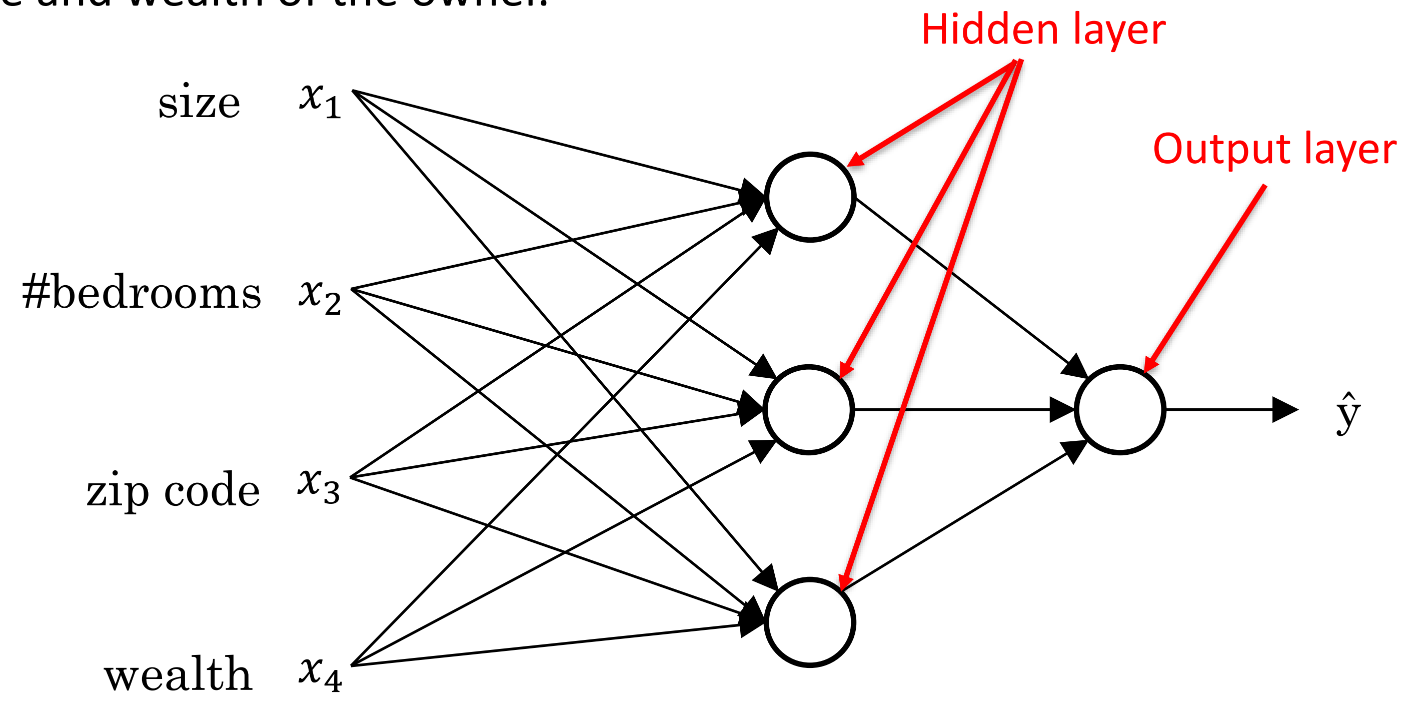
$$\hat{y} = \ \sigma(wx + c) = \frac{1}{1+e^{-(wx+c)}}$$

$$= \frac{1}{1+e^{-(0.4121205\times 1.175+0.4252089)}} = 0.7128877$$

Closer than before!
0.747911 > **0.7128877** > 0
Still not predicting that it is a car, but we should repeat the same procedure for the entire dataset.

Neural networks

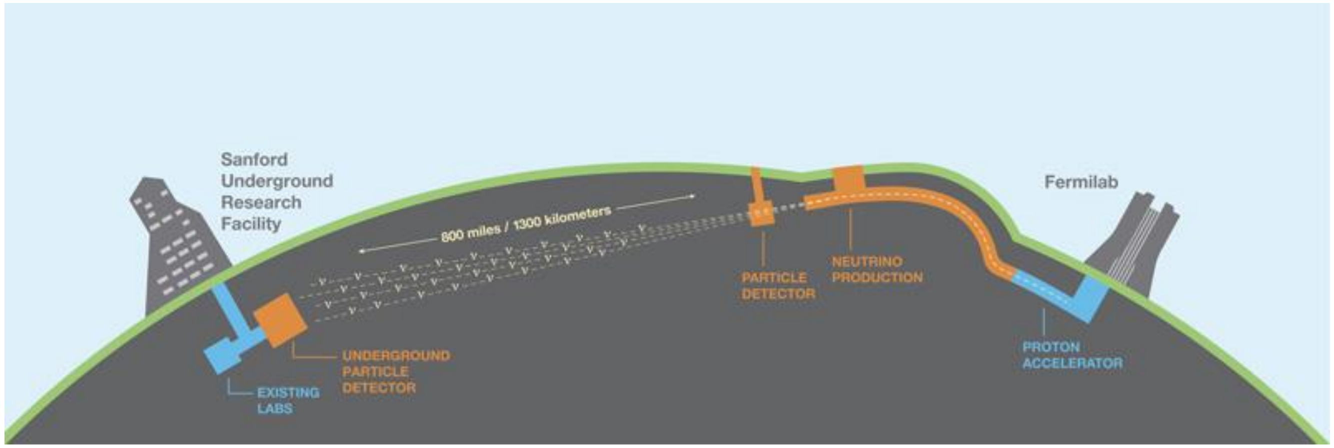
- Same idea, but multiple neurons (forming layers).
  - Example: house price prediction based on size, number of bedrooms, zip code and wealth of the owner.



- **Deep learning (DL)**: normally refers to methods based on neural networks with a large number of layers (deep neural networks).

Deep Learning in DUNE

- DUNE is a next-generation neutrino oscillation experiment.


Sanford Underground Research Facility
800 miles / 1300 kilometers
Fermilab
PARTICLE DETECTOR
NEUTRINO PRODUCTION
UNDERGROUND PARTICLE DETECTOR
EXISTING LABS
PROTON ACCELERATOR

- Far Detectors (FD) are 800 miles from the neutrino beam source.
  - Four modules, each with 10,000 ton of liquid argon.
- High power muon neutrino beam produced at Fermilab.
  - Can switch polarity to produce a muon antineutrino beam.
- Look for the appearance of electron (anti)neutrinos at the FD.
  - Measure *CP*-violation.

NB: I will only write neutrino from now on, but the same is applicable for antineutrinos

Ingredients for the CP-violation analysis

- We need to consider two signal channels and their backgrounds.
  - Charged current $\overset{(-)}{\nu}_\mu$ disappearance – main background is NC $1\pi^{\pm}$.
  - Charged current $\overset{(-)}{\nu}_e$ appearance – main background is NC $1\pi^{0}$.

- Primary goal:
  - Classify the neutrino flavour as ν_e ,ν_μ , ν_τ or NC.

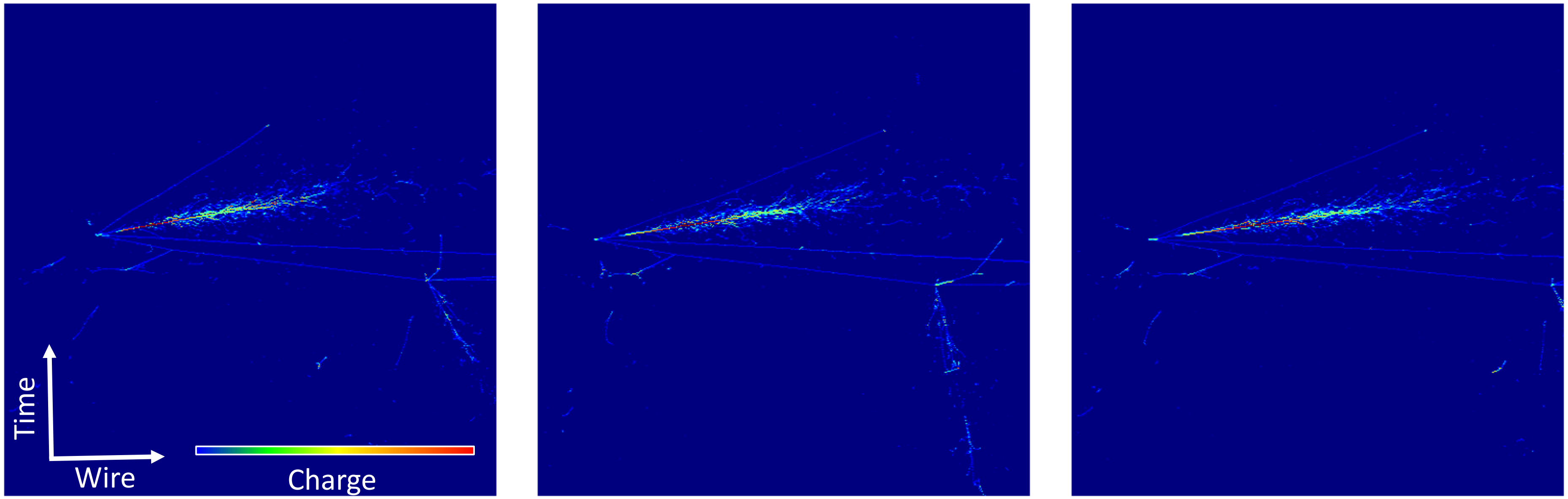
- We need a DL approach to perform the classification task.
  - We use a convolutional neural network (CNN).

- See my talk at the Reconstruction and Machine learning in Neutrino Experiments workshop:
  - https://indico.desy.de/indico/event/21853/session/2/contribution/35

Far Detector Data

- The Far Detectors contain three wire readout planes.
  - This provides three “images” of each neutrino interaction.

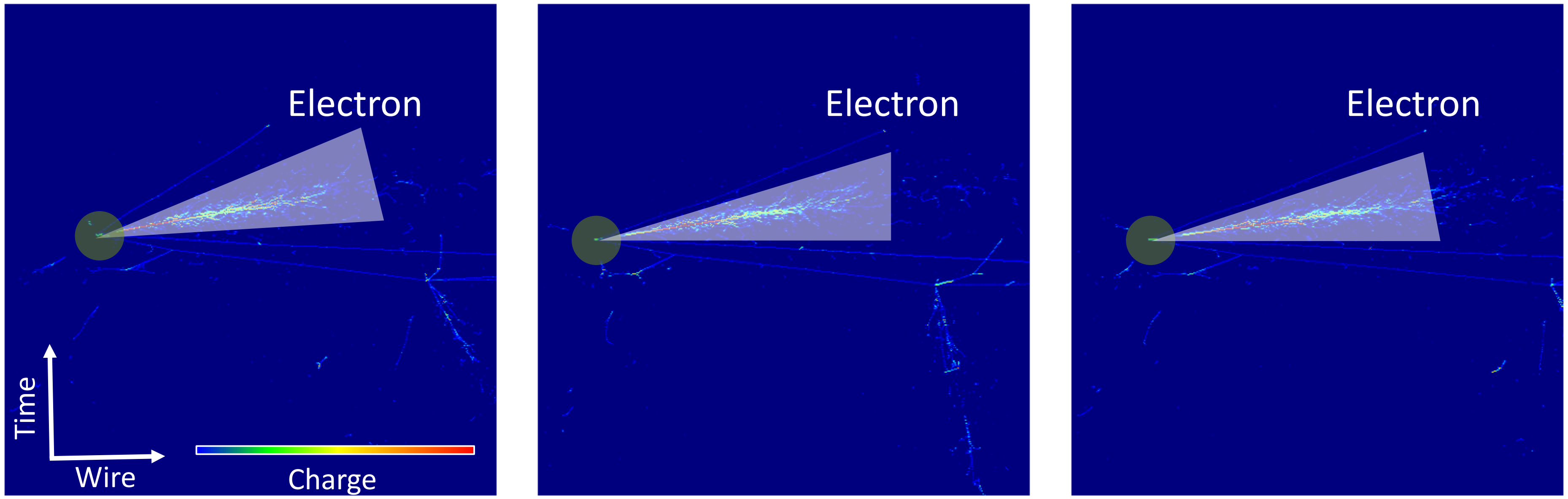
- Official simulated electron neutrino interaction (signal).



Far Detector Data

- The Far Detectors contain three wire readout planes.
  - This provides three “images” of each neutrino interaction.

- Official simulated electron neutrino interaction (signal).

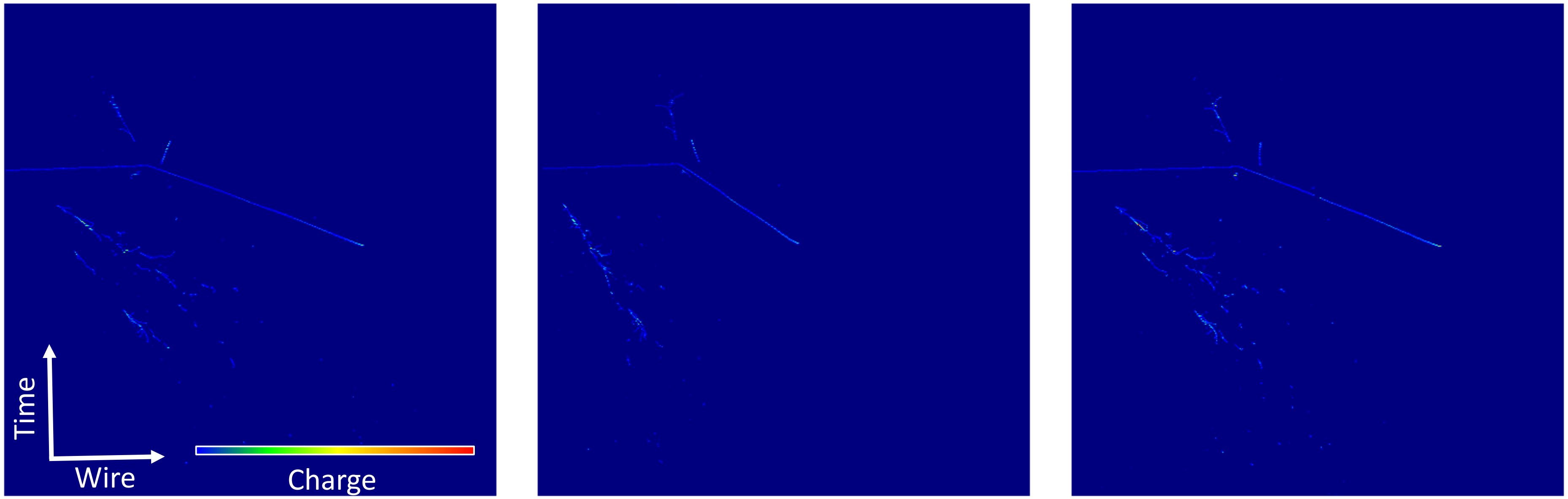


- Electron produces the highlighted shower, beginning at the vertex.

Far Detector Data

- The Far Detectors contain three wire readout planes.
  - This provides three "images" of each neutrino interaction.

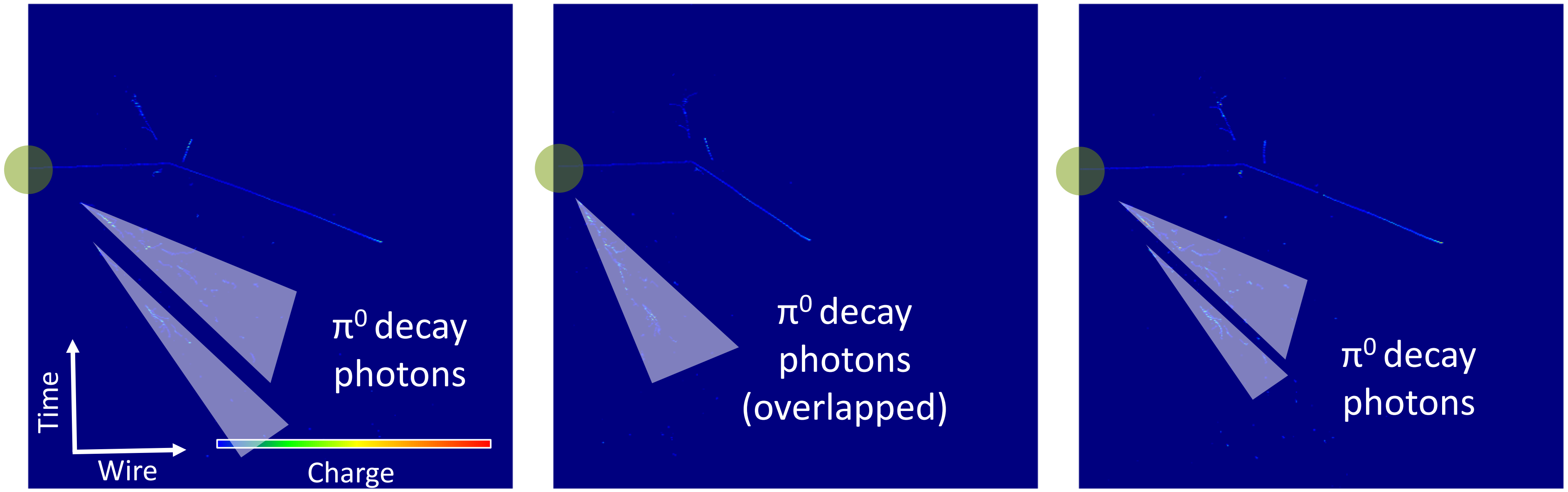
- Official simulated neutral current π^0 interaction (background).



Far Detector Data

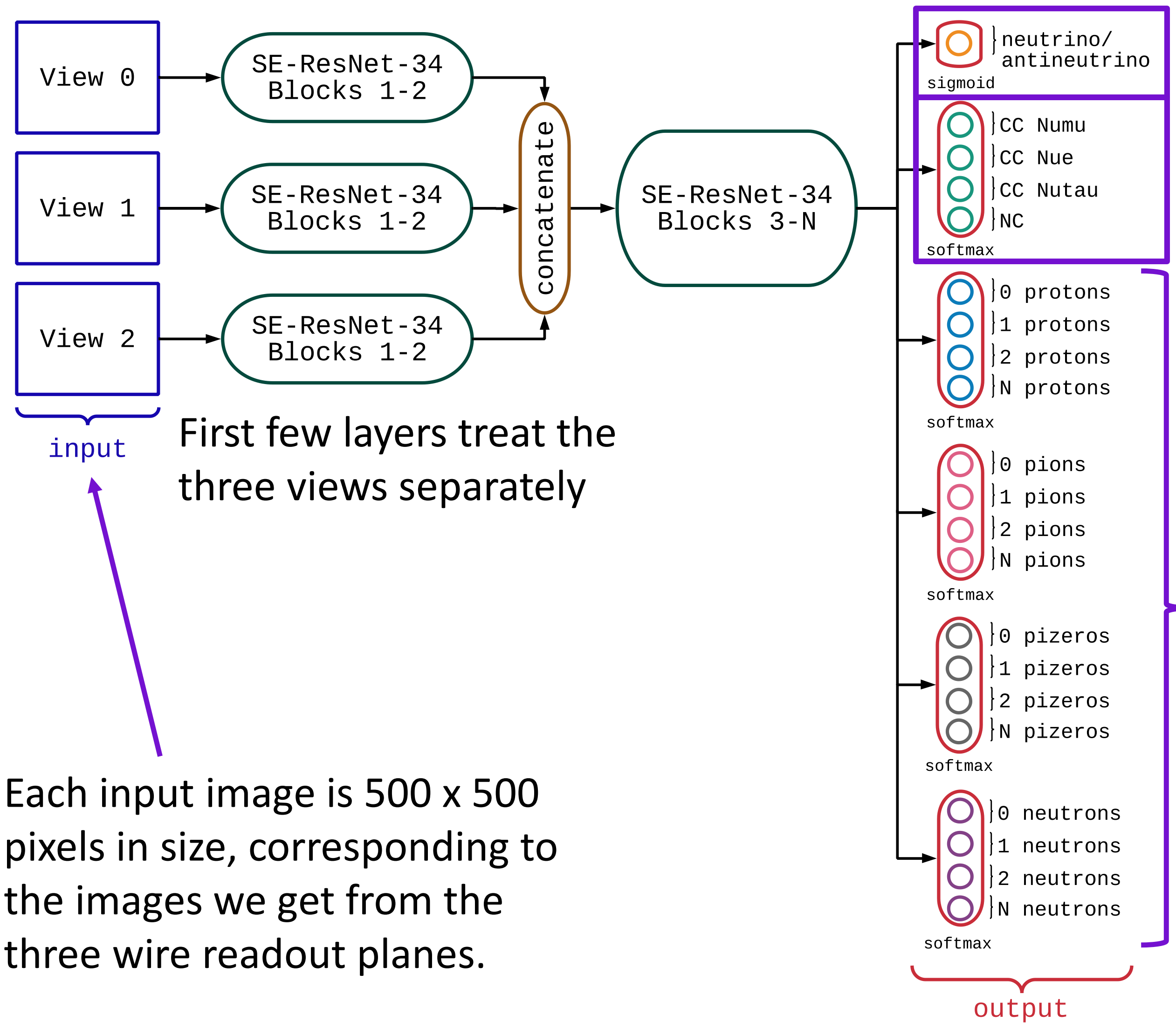
- The Far Detectors contain three wire readout planes.
  - This provides three “images” of each neutrino interaction.

- Official simulated neutral current π^0 interaction (background).



- π^0 decay photon showers are displaced from vertex.

CVN Architecture Overview



Neutrino / antineutrino

Primary output: Flavour identification

Secondary outputs: Particle counting for exclusive final states

First few layers treat the three views separately

Each input image is 500 x 500 pixels in size, corresponding to the images we get from the three wire readout planes.

Training and using the CNN

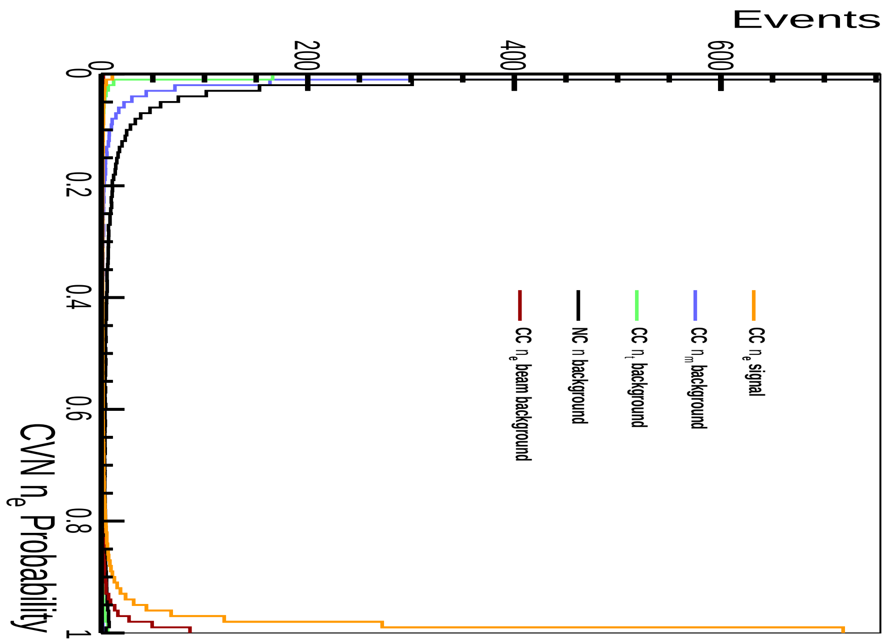
- Use millions of images (~10M images) of simulated neutrino interactions with the true neutrino flavour known.
  - Allows the CNN to learn the features of each type of neutrino interaction.
  - The CNN filters are not predefined – it needs to learn which filters to use to extract the information required to classify events.
  - Tested on a fully independent sample.

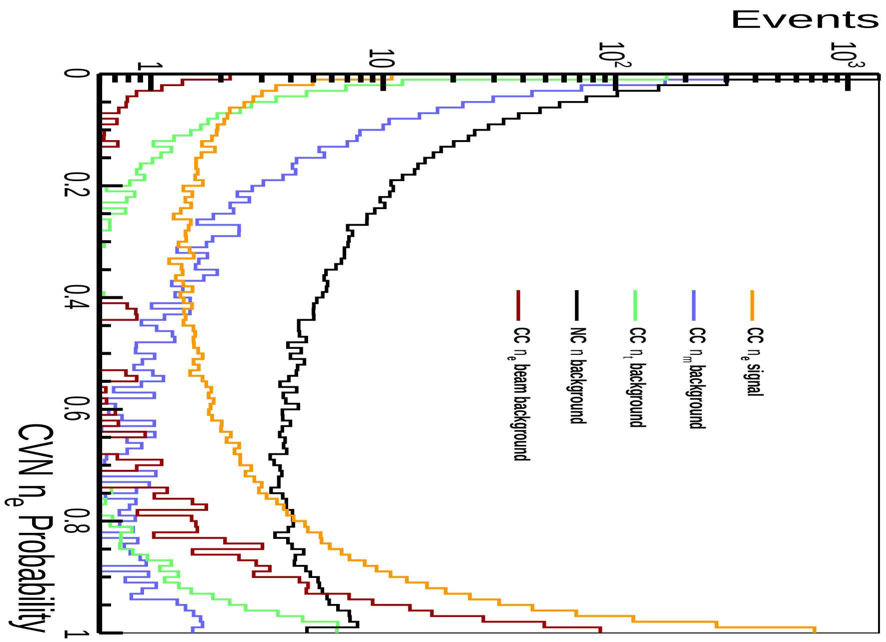
- Once the CNN is trained it is applied to images with no truth information attached – eventually the experimental data.

- The CNN gives probabilities for each event to be the following:
  - Charged-current ν_e, ν_μ, ν_τ and neutral-current (all flavours).
  - Outputs sum to one.
  - Use these probabilities for the event selection.

Selecting Electron Neutrinos

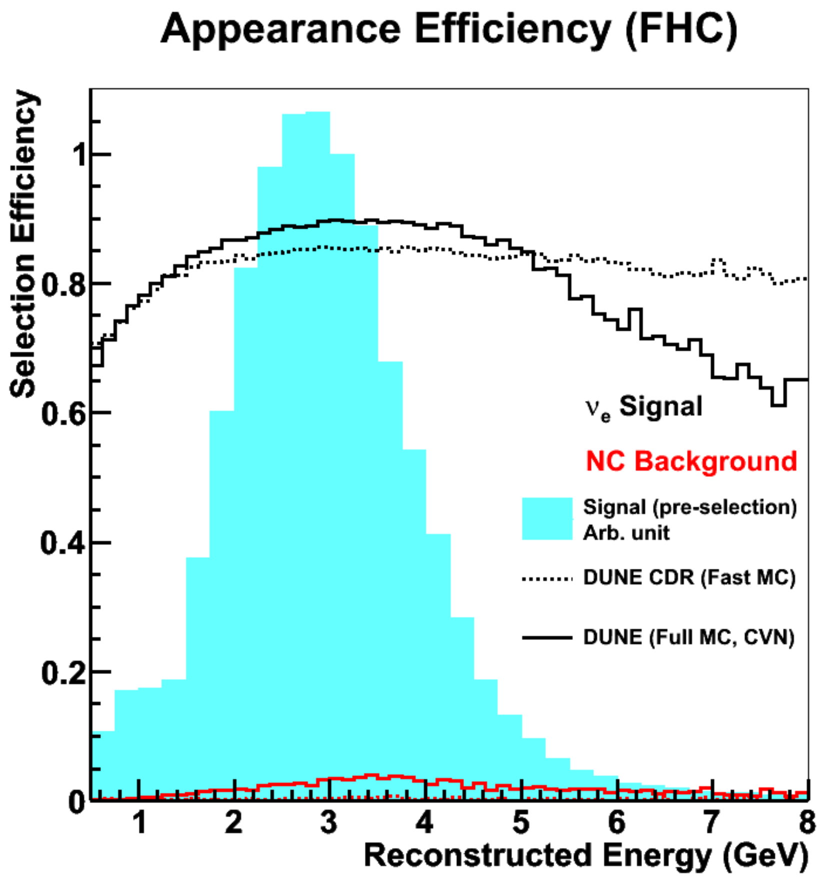
- Electron neutrino probability spectra from the DUNE CVN.
  - Curves combine neutrinos and antineutrinos.





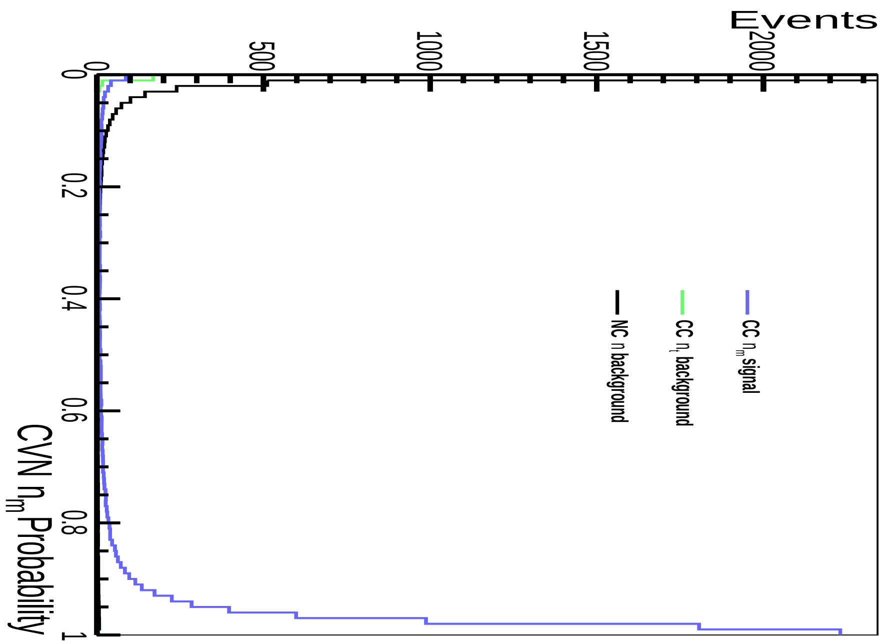
Electron Neutrino Efficiency

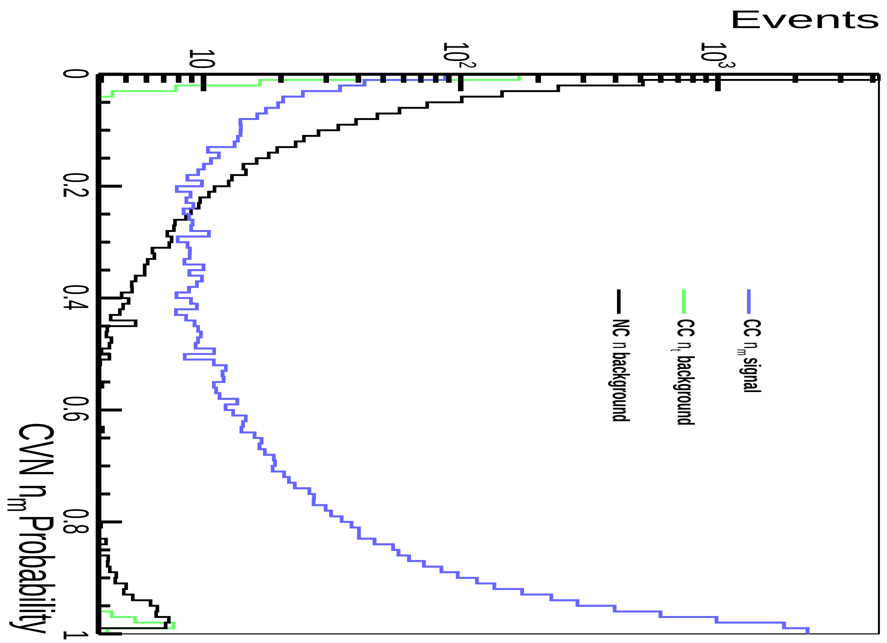
- Select all events that are more than 85% likely to be electron neutrinos.
- Over 90% selection efficiency in the flux peak.
- Efficiency better for antineutrinos due to typically cleaner final state (neutron instead of proton).



Selecting Muon Neutrinos

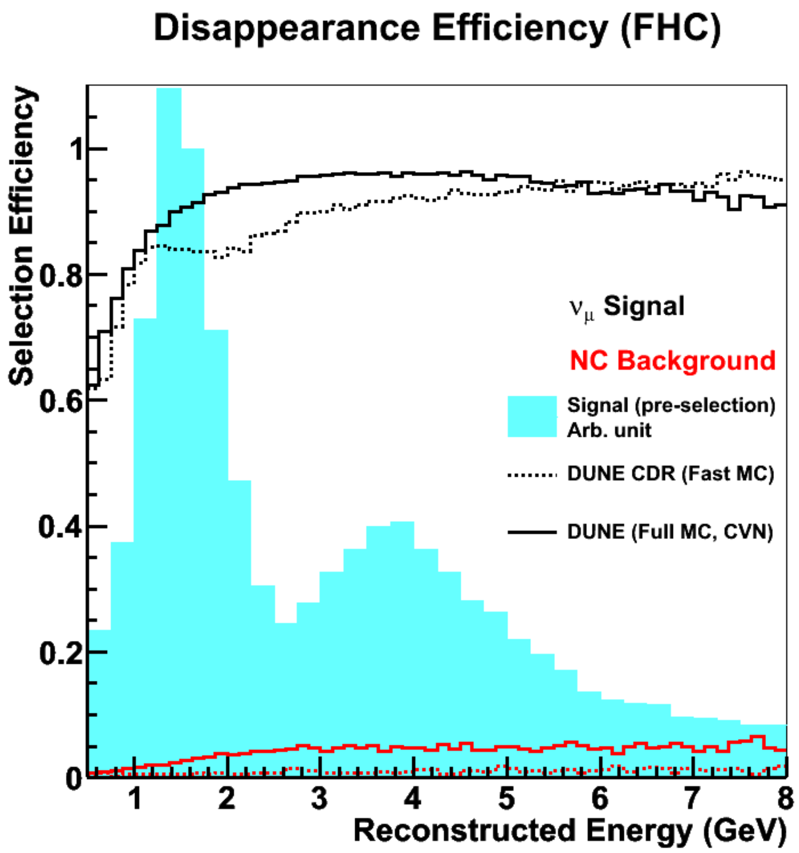
- Muon neutrino probability spectra from the DUNE CVN.
  - Curves combine neutrinos and antineutrinos.





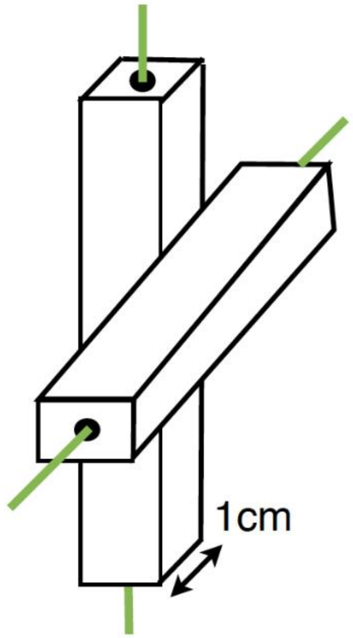
Muon Neutrino Efficiency

- Select all events that are more than 50% likely to be muon neutrinos.
- Over 90% selection efficiency in the flux peak.
- Efficiency better for antineutrinos due to typically cleaner final state (neutron instead of proton).



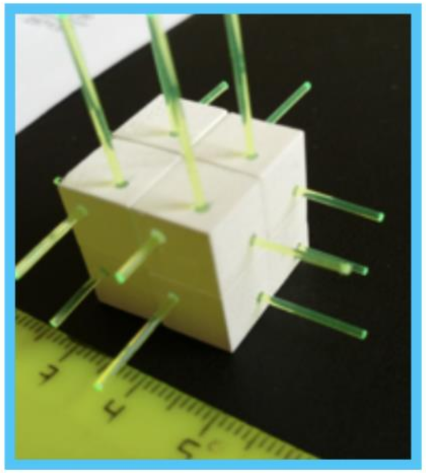
Deep Learning in T2K

- Full active FGD with three views: SuperFGD.
  - Optically independent cubes: spatial localization of scintillation light.
  - Lower momentum threshold: 1 single hit gives immediately XYZ.
  - Plastic scintillator provides very good time resolution.



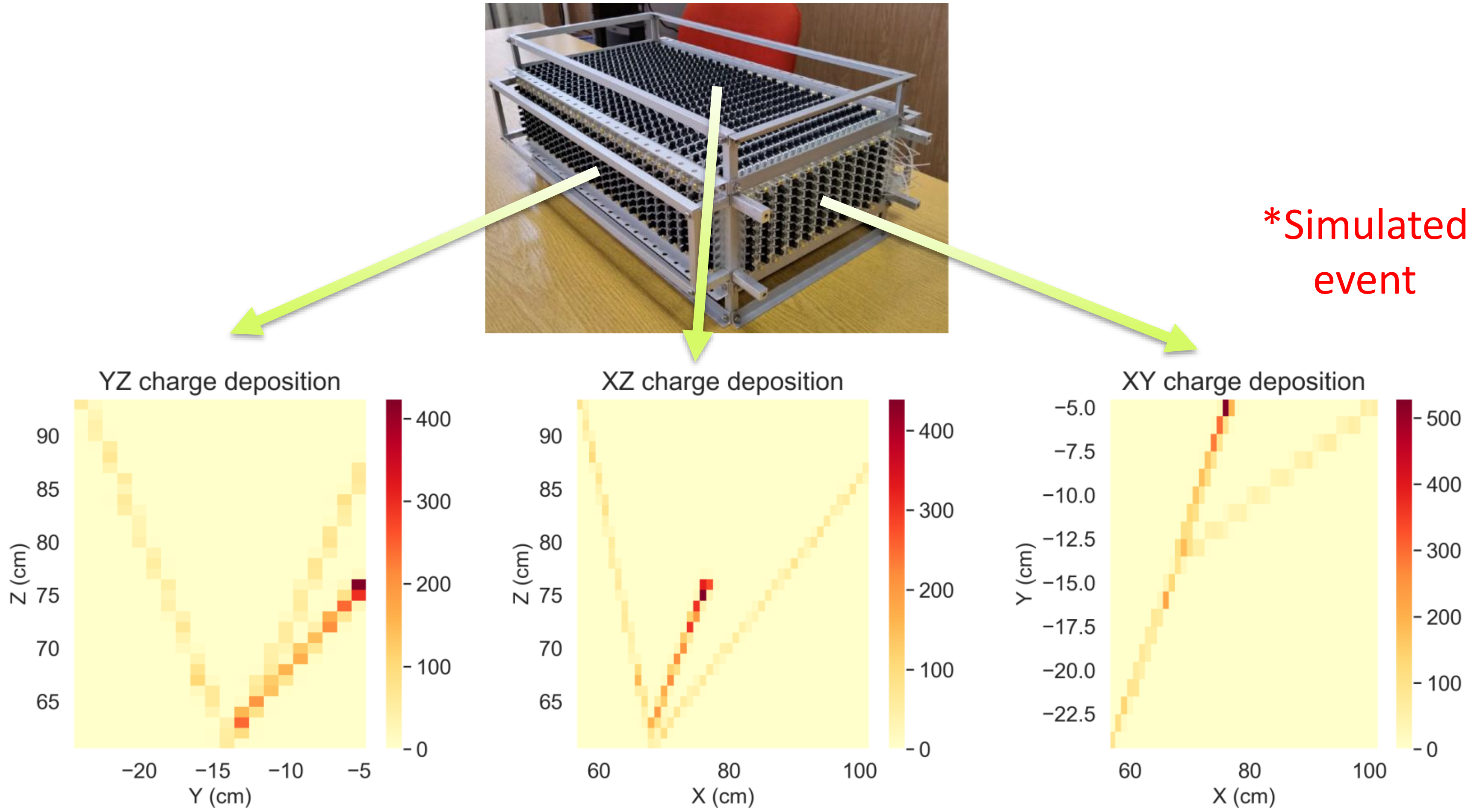
2018 JINST 13 P02006

Polystirene-based Plastic scintillator
1.5% paraterphenyl and 0.01% POPOP
1x1x1 cm^3 cubes
Chemical etching as reflector
WLS fibers (Kuraray Y11, 2-clad, 1mm)



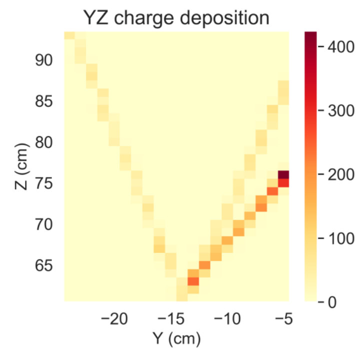
SFGD 2D Events

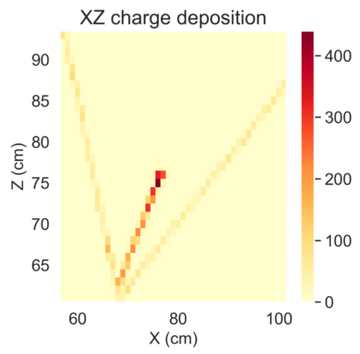
- Three 2D charge deposition views (XY, XZ, YZ):

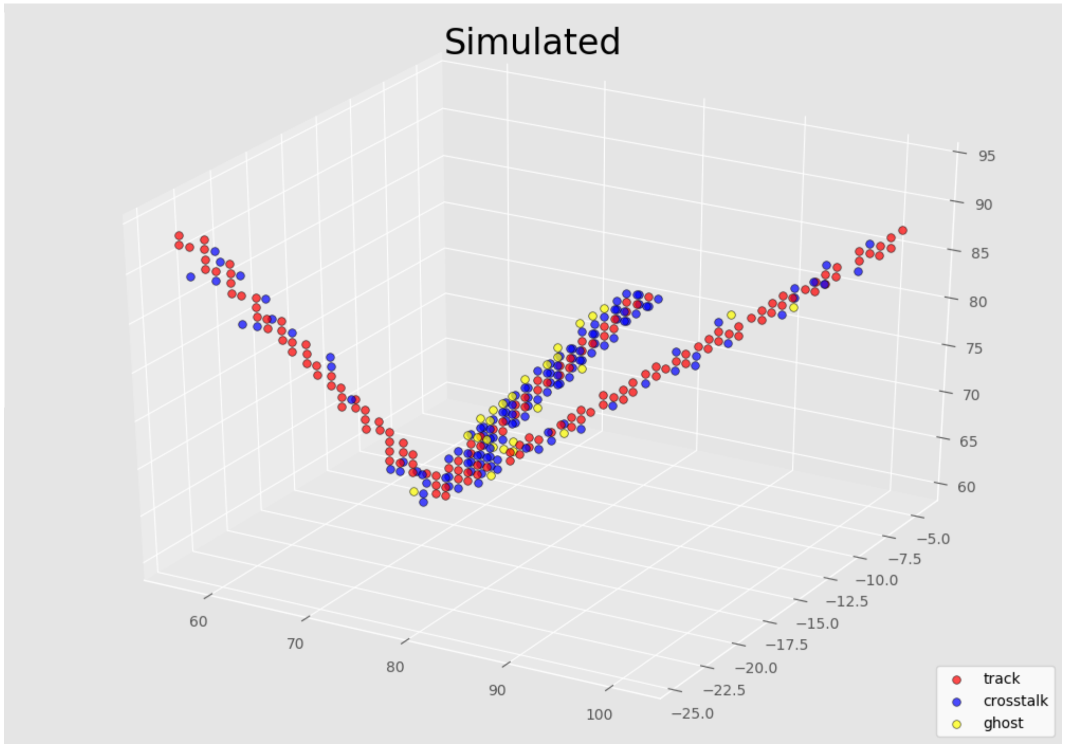


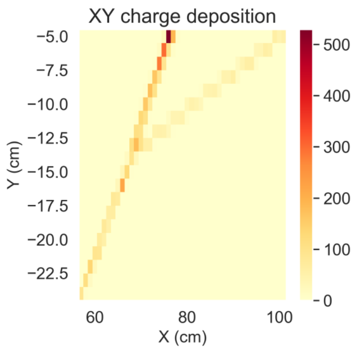
SFGD 2D to 3D

- Matching the common axis 2-to-2 in the three views XY, XZ, YZ we obtain the 3D information.









- Drawback: non-physical voxels appear due to lack of information during the 2D to 3D reconstruction algorithm, called ghost voxels.

Problem description

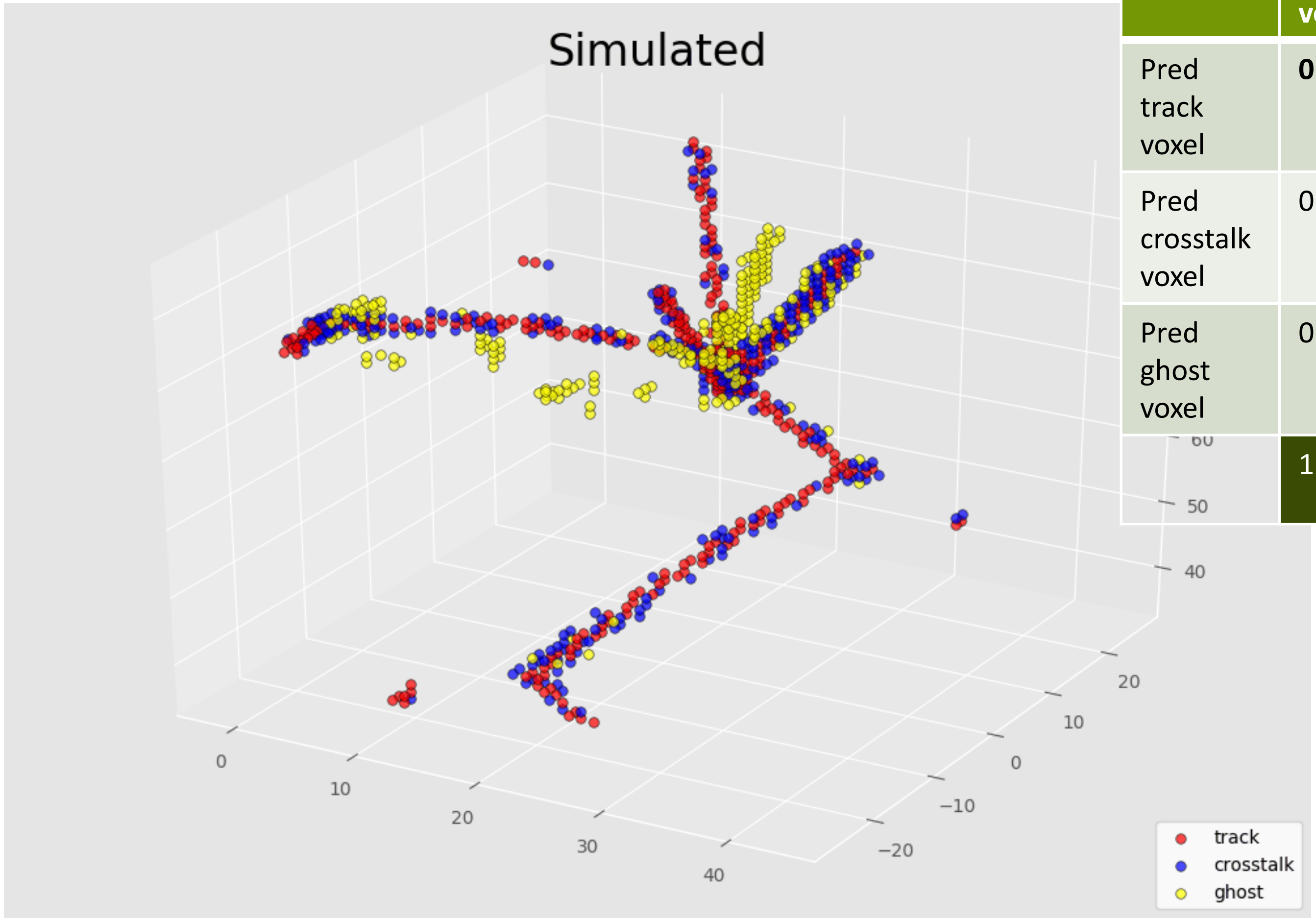
- After 3D-matching, we classify each individual 3D voxel into one of the following:
  - **Track voxel**: a cube where the track has passed by.
  - **Crosstalk voxel**: a cube with a real deposition but where any track has passed through it (all light comes from cube-to-cube optical crosstalk).
  - **Ghost voxel**: a cube that does not have any real deposition and is formed from the 2D ambiguity when reconstructing the 3D event.

- Approach: use a supervised deep learning algorithm (GraphSAGE*) to perform the classification task.
  - The approach based around a graph neural network (GNN) handles each individual voxel as a list of variables (physics information) associated to it.
  - See S. Pina-Otey's talk at the ND 280 Upgrade Meeting: https://indico.cern.ch/event/842568/contributions/3578802/

**W. L. Hamilton, R. Ying, J. Leskovec, Inductive Representation Learning on Large Graphs, arXiv:1706.02216.*

Results (GraphSAGE)

Event 1: simulated vs pred. (GIF image*):

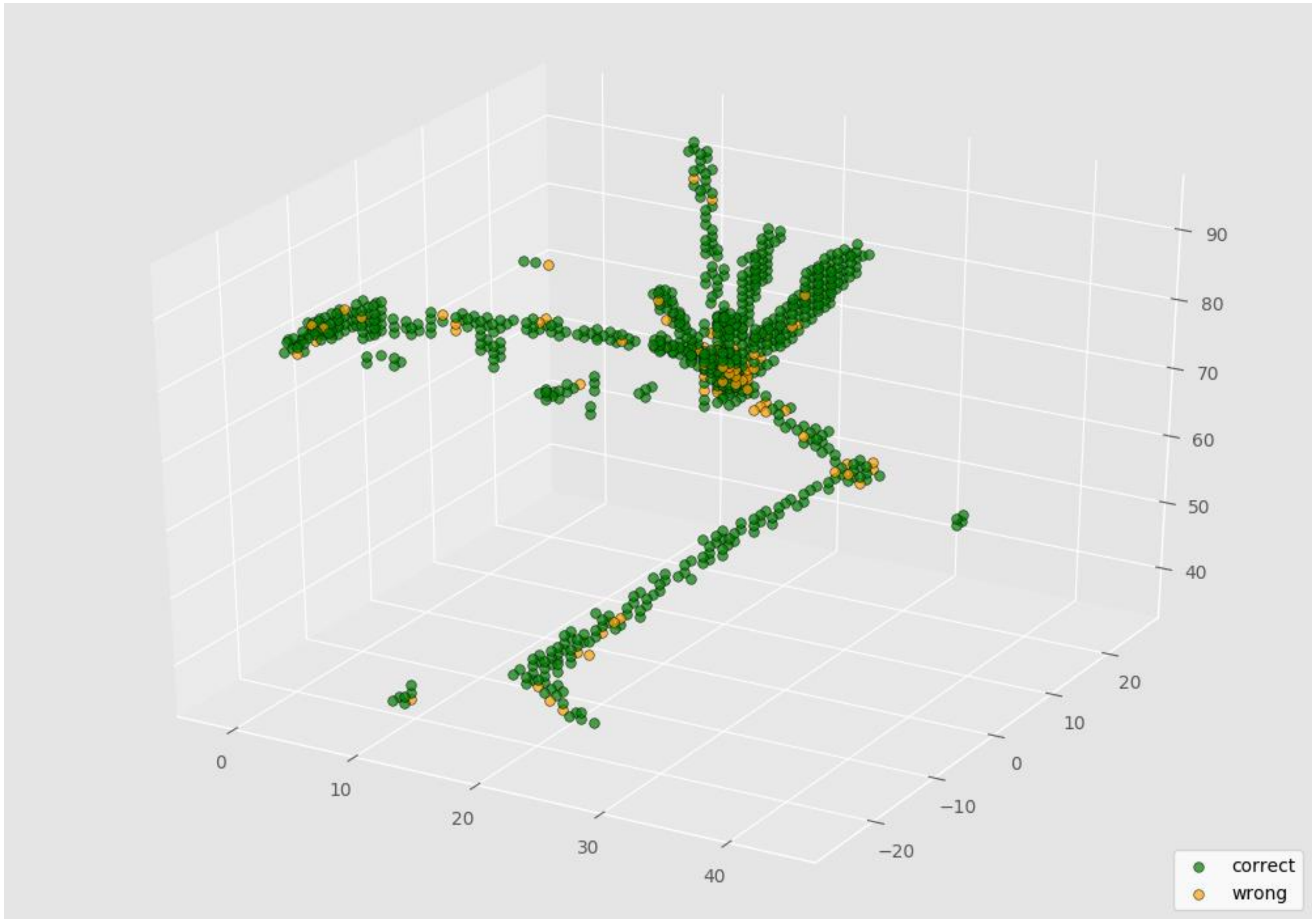


	True track voxel	True crosstalk voxel	True ghost voxel
Pred track voxel	**0.898**	0.070	0.029
Pred crosstalk voxel	0.099	**0.890**	0.041
Pred ghost voxel	0.003	0.040	**0.930**
	1.000	1.000	1.000

**slide show* to see the animated GIF

Results (GraphSAGE)

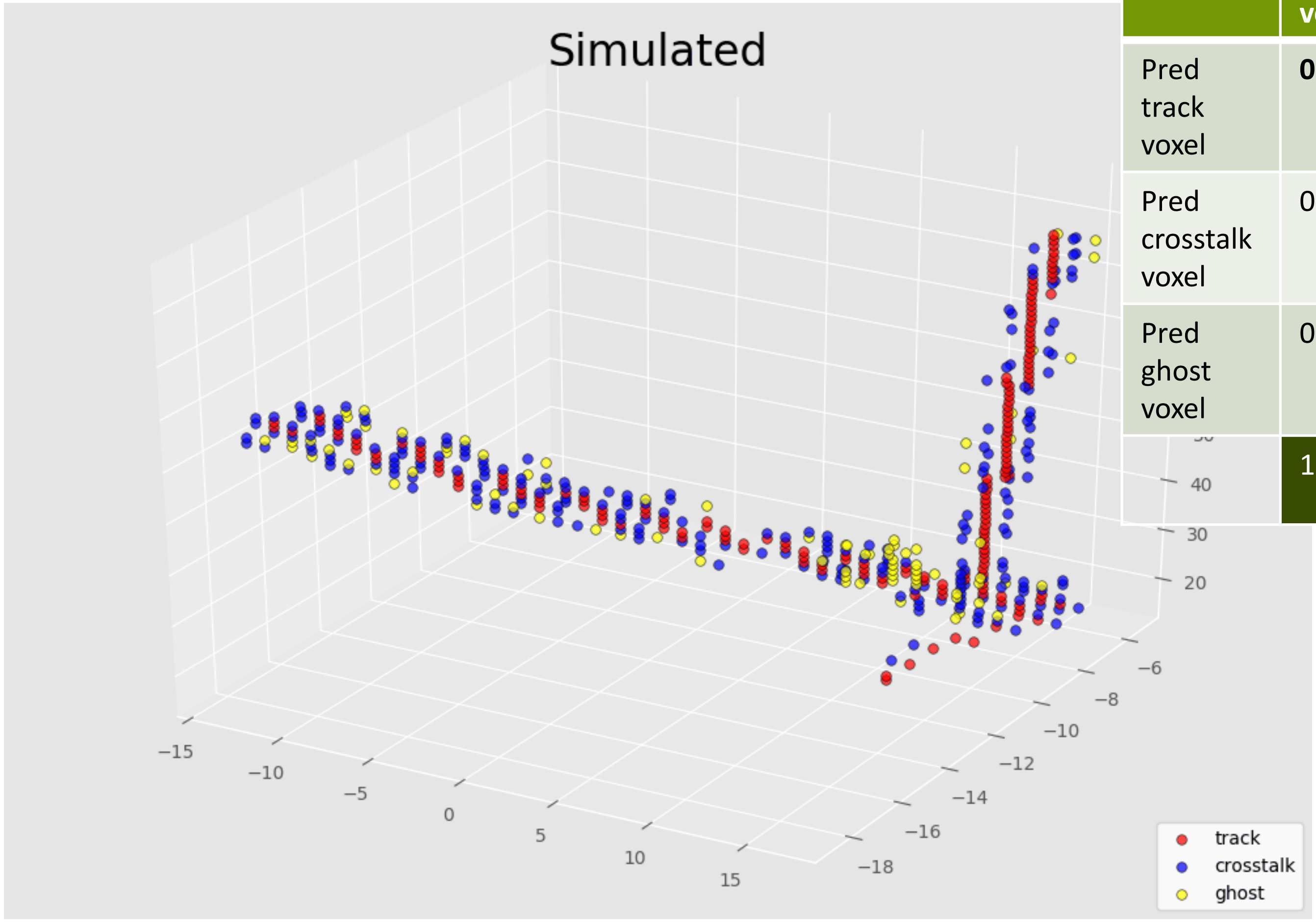
Event 1: correct classified voxels.



The ghost track would probably be removed by standard techniques by matching the light detected in the three 2D views. This approach has been already tested by S. Martinenko (see his talk at the T2K collaboration meeting).

Results (GraphSAGE)

Event 2: simulated vs pred. (GIF image*):

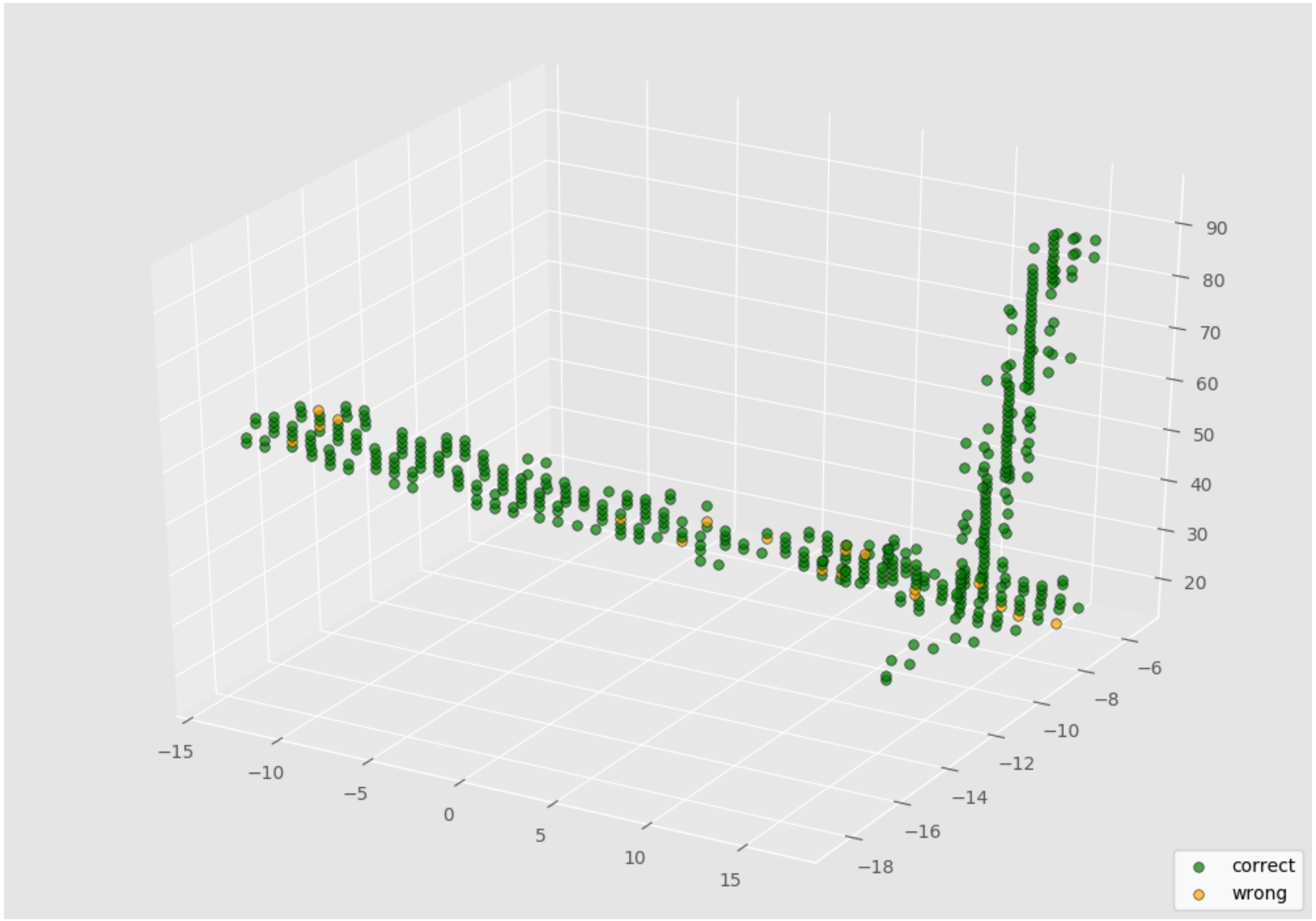


	True track voxel	True crosstalk voxel	True ghost voxel
Pred track voxel	**0.970**	0.022	0.000
Pred crosstalk voxel	0.024	**0.955**	0.050
Pred ghost voxel	0.006	0.023	**0.950**
	1.000	1.000	1.000

**slide show* to see the animated GIF

Results (GraphSAGE)

Event 2: correct classified voxels.



Summary

- Machine Learning, and Deep Learning in particular, provide many powerful mechanisms for classifying input data from many different fields, including high-energy physics.

- In DUNE, we use a convolutional neural network for a powerful neutrino interaction flavour classification.
  - Already working on demonstrating good performance of exclusive final-states in the coming months.

- In T2K, we use a graph neural network (GNN) for voxel classification in SuperFGD.
  - First results look promising both in purity and in efficiency.

Backup Slides

Results (GraphSAGE)

- **Training on 6k events.**
- **Confusion matrix (from 60k events):**

	True track voxels	**True crosstalk voxels**	**True ghost voxels**	
Pred track voxels	**5,140,704**	167,236	13,089	5,321,029
Pred crosstalk voxels	286,890	**5,001,600**	124,886	5,413,376
Pred ghost voxels	16,561	140,140	**1,401,551**	1,558,252
	5,444,155	5,308,976	1,539,526	**12,292,657**

Observation: a 0.96% of voxels cannot be calssified by GraphSAGE due to not having any edge in the graph.

Results (GraphSAGE)

- **Purity (left) vs Efficiency (right)**

	True track voxels	**True crosstalk voxels**	**True ghost voxels**	
Pred track voxels	**0.9661**	0.0314	0.0025	1.0000
Pred crosstalk voxels	0.0530	**0.9239**	0.0231	1.0000
Pred ghost voxels	0.0106	0.0899	**0.8995**	1.0000

	True track voxels	**True crosstalk voxels**	**True ghost voxels**
Pred track hits	**0.9443**	0.0315	0.0085
Pred crosstalk hits	0.0527	**0.9421**	0.0811
Pred ghost hits	0.0030	0.0264	**0.9104**
	1.0000	1.0000	1.0000