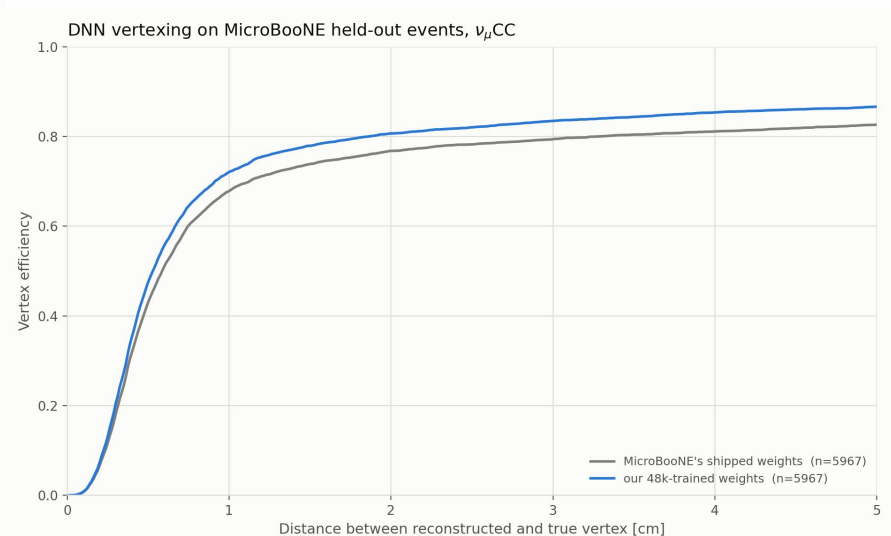
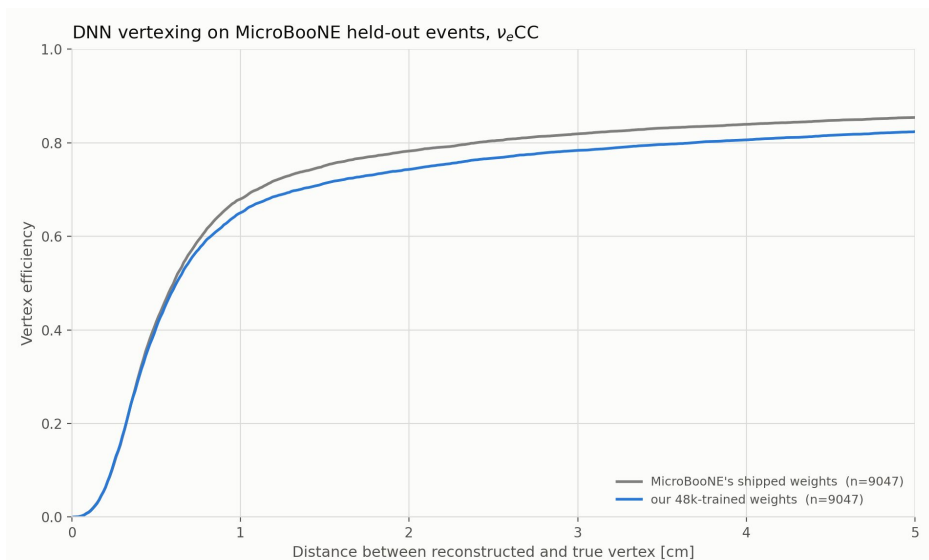
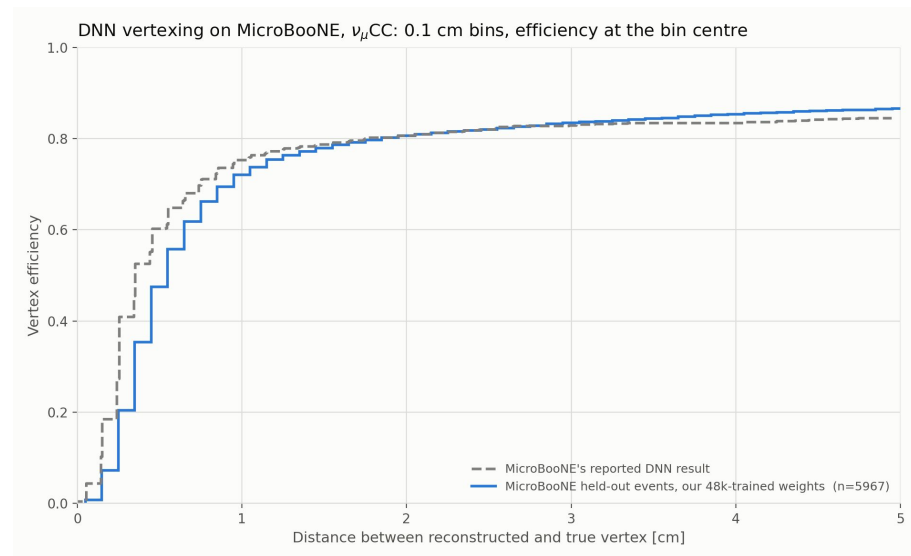
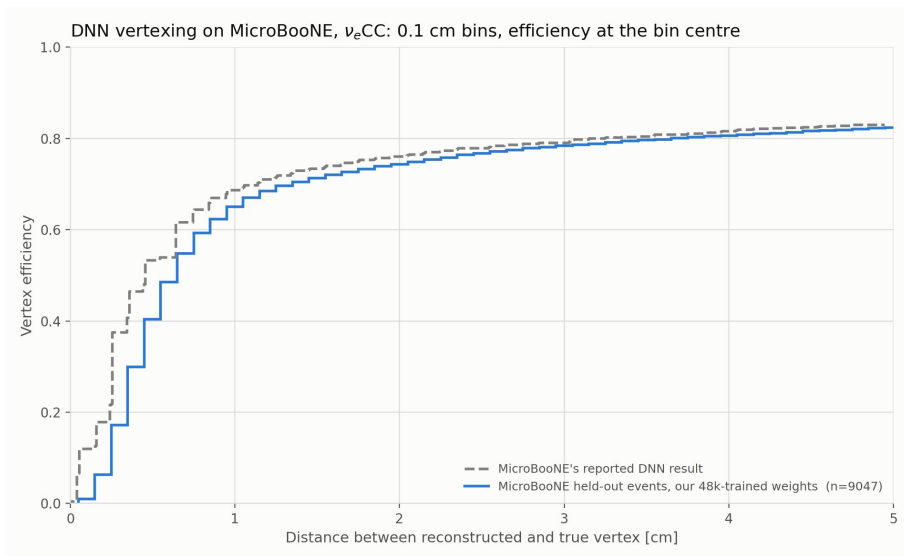


Re-trained on MicroBooNE sample provided by Haiwang

We do better on $\nu_\mu\text{CC}$ and worse on $\nu_e\text{CC}$ because the paper only trained on $\nu_e\text{CC}$ events. This MicroBooNE sample had <48k $\nu_e\text{CC}$ events so it's probably independent from the paper's sample.



Reproduction is slightly worse. Why?

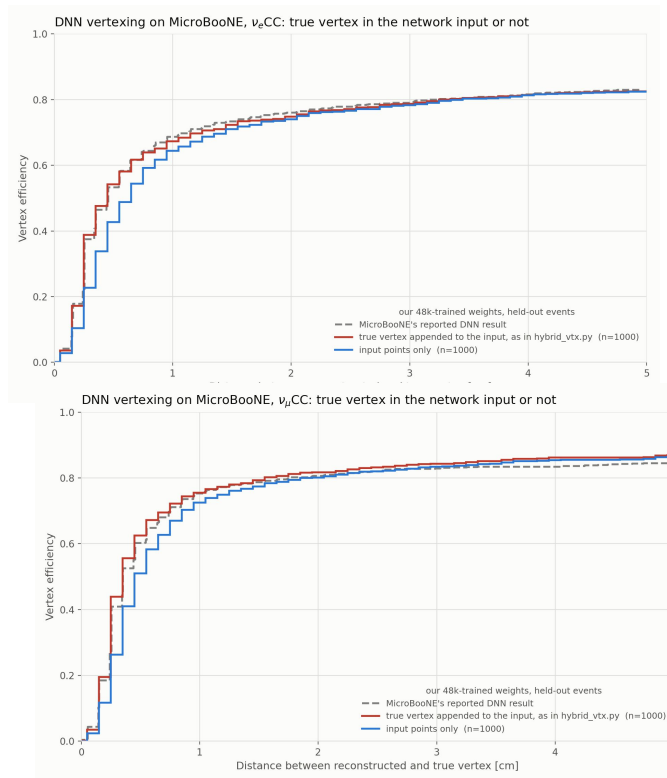


Is it possible there's some truth leakage in the paper?

There's an option to include the truth vertex as a zero-charge occupied voxel in some of the code which is on in `hybrid_vtx.py`, which performs traditional, DNN and hybrid vertexing.

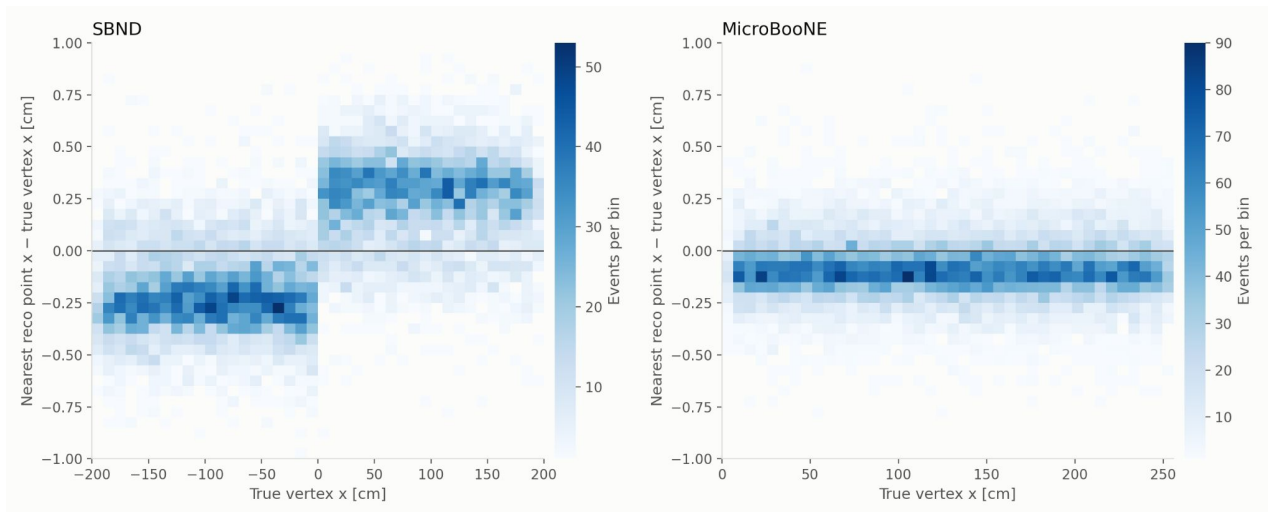
That script

- Has the same binning structure as paper's plots
- Saves CSV columns `[d_dnn, d_trad, d_hybrid,...]` which would be input into a ROOT plot
- No other evaluation script runs all three
- This evaluation script has truth leakage in every commit (the mode `= "vox"` was never turned off, from the commit history)



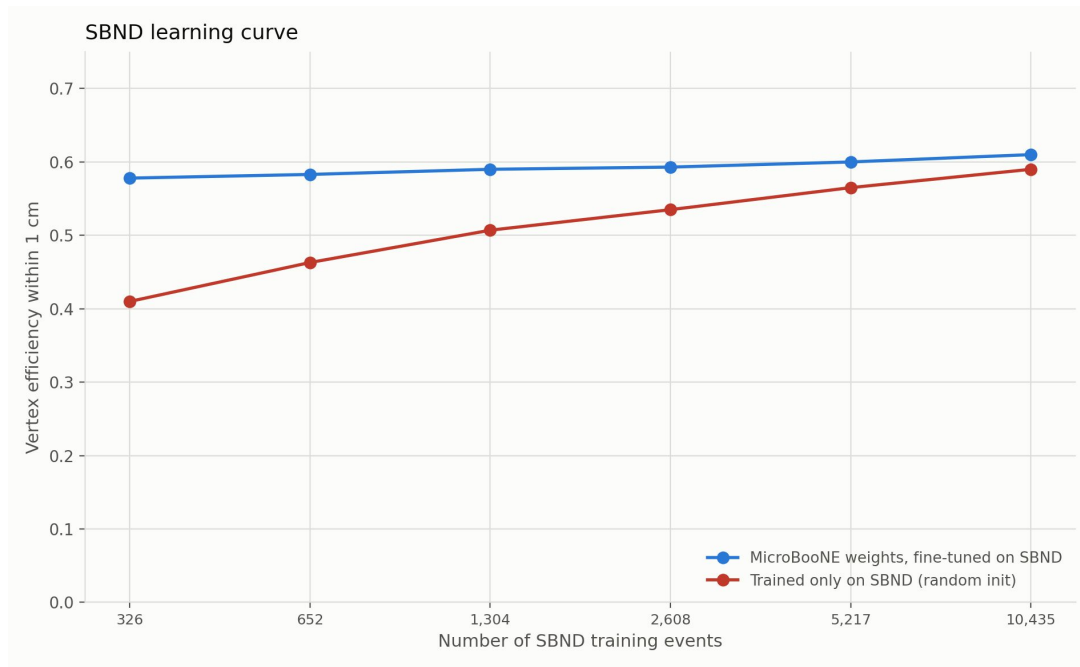
Apparent drift-speed glitch

```
[snehadri@sophia-login-02 snehadri]$ F=/lus/eagle/projects/neutrinoGPU/snehadri/wct/spack/opt/spack/linux-x86_64_v3/wire-cell-data-0.2.0-jbu22fdchzcf5ii5nvepy6fcgspf4cnu/share/wirecell/garfield-sbnd-v1.json.bz2
bzip2 $F | grep -o '"speed": *[0-9.e+-]*'
"speed": 0.00159
[snehadri@sophia-login-02 snehadri]$ grep -n "drift_speed :" /lus/eagle/projects/neutrinoGPU/snehadri/sbnd/wct_cfg_src/pgrapher/experiment/sbnd/simparams.jsonnet
16:     drift_speed : 1.563*wc.mm/wc.us,
```



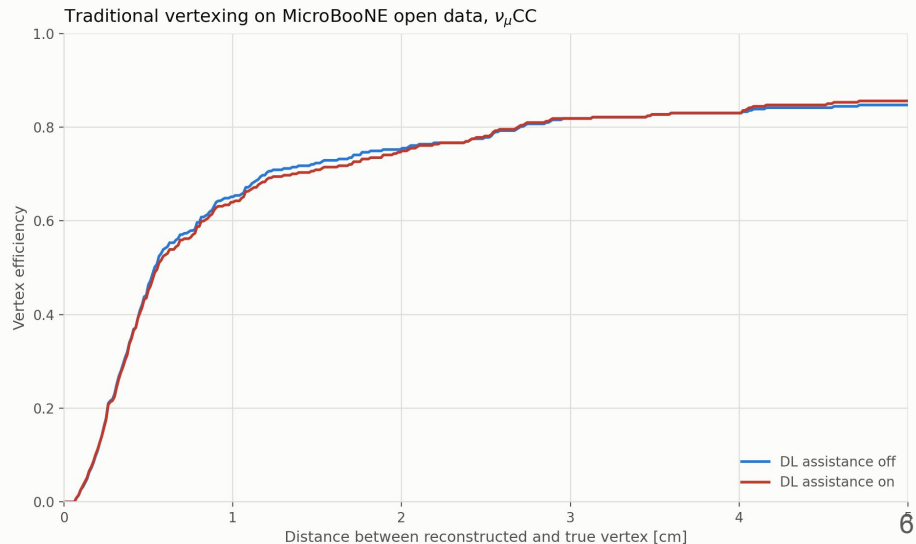
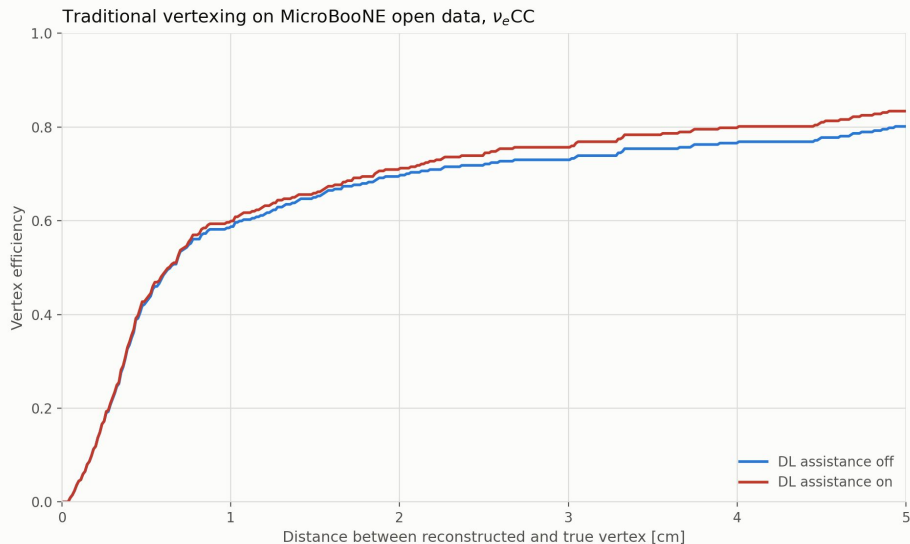
Fine-tuning efficiency is maybe data-limited

Evaluated on 233 ν_e CC
and 167 ν_μ CC held-out
SBND events

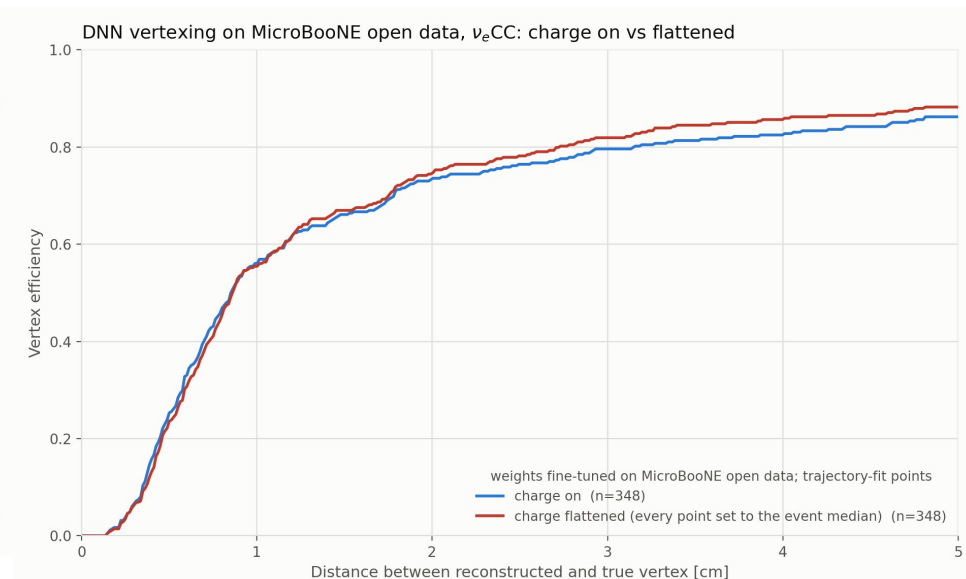
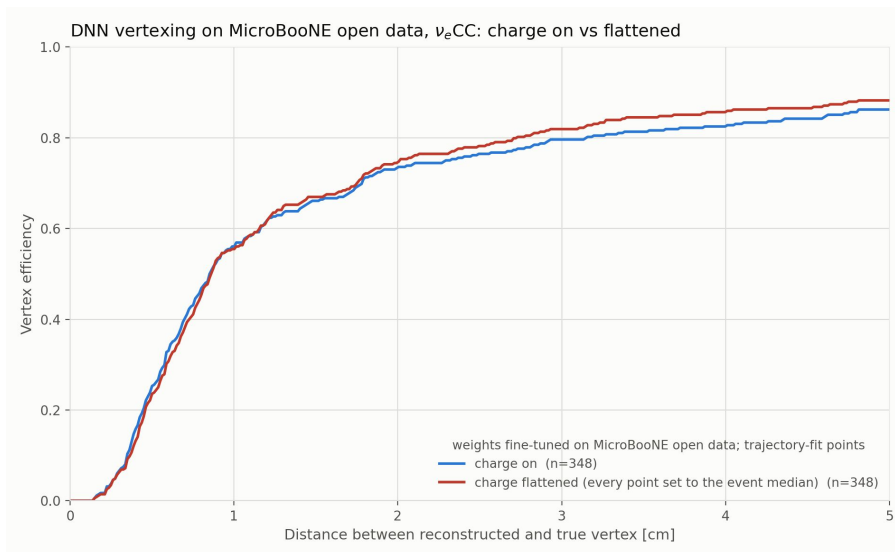


DL-assistance in traditional vertexing

DL-assistance overrides the traditional choice if the DNN confidence is large enough. It doesn't make a huge difference so I'll just keep it off from now on.



Charge on vs off

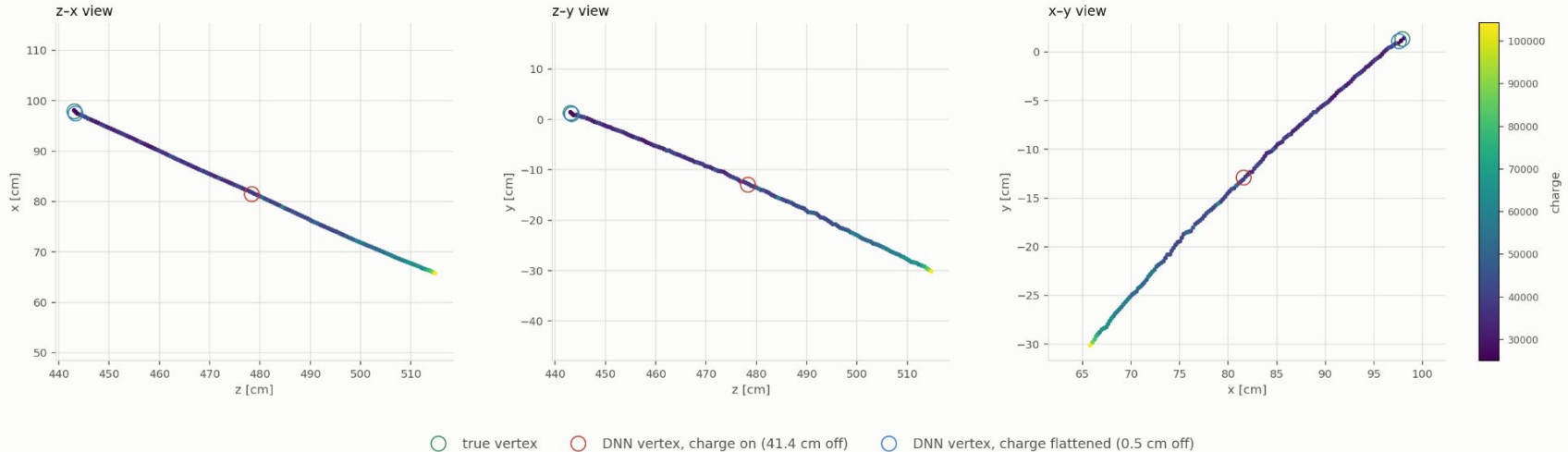


Turning charge off saves some failures

Events are uploaded at
<https://www.phy.bnl.gov/twister/bee/set/ed30d2bd-b68b-4dee-a972-b126bee0cccf/>

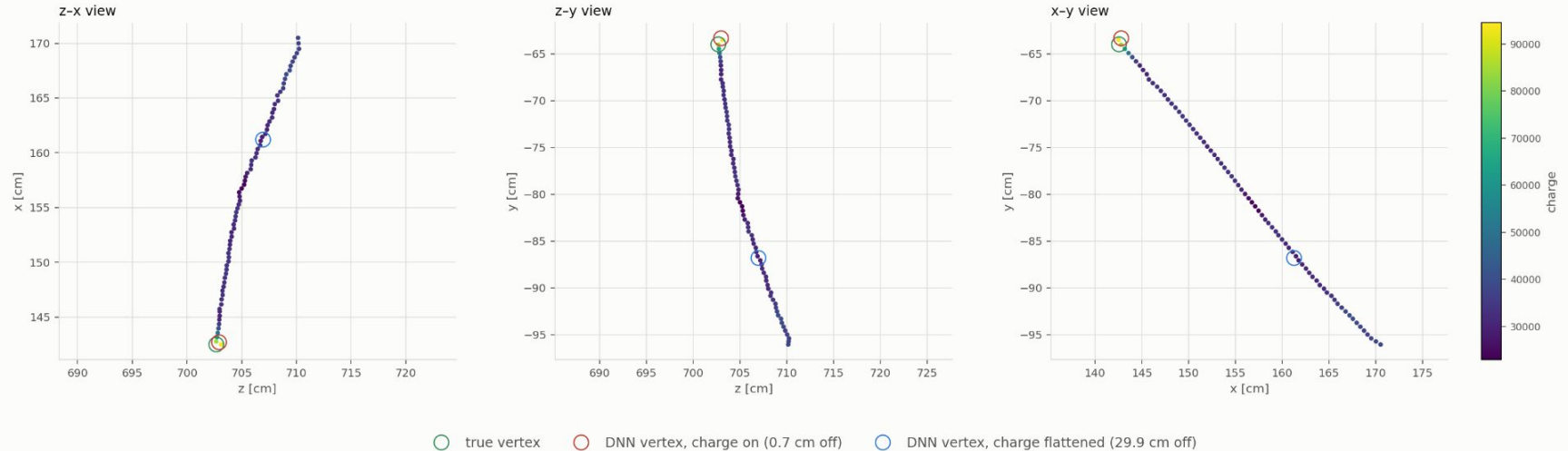
In this event, electron shower was never reconstructed and only proton track was seen. Turning charge off saves this as it relies on the geometry more instead of seeing one faint track attached to a bright track.

MicroBooNE open data ν_e CC, run 7023 / subrun 273 / event 13680 DNN confidence at the true vertex: charge on 0.03, flattened 0.76



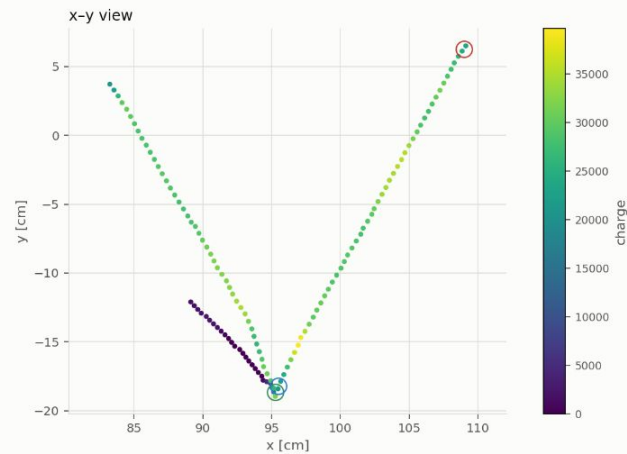
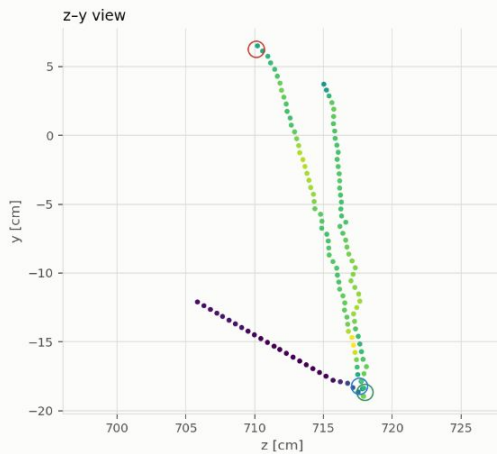
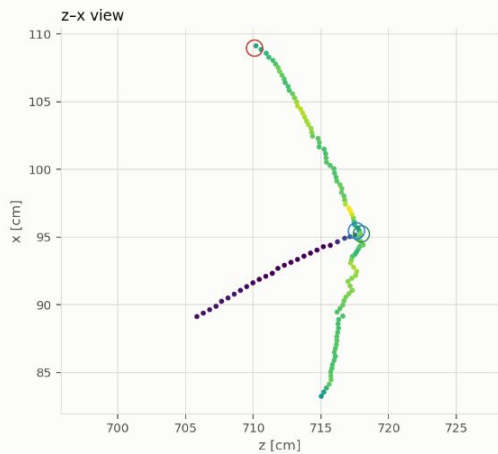
But similar cases seem to have the exact problem in the reverse direction?

MicroBooNE open data ν_μ CC, run 7017 / subrun 715 / event 35757 DNN confidence at the true vertex: charge on 0.72, flattened 0.02



Still some wacky choices

MicroBooNE open data ν_μ CC, run 7014 / subrun 1059 / event 52981



- true vertex — DNN confidence here: charge on 0.11, flattened 0.24
- DNN vertex, charge on (29.5 cm off) — confidence here: charge on 0.22, flattened 0.07
- DNN vertex, charge flattened (0.6 cm off) — confidence here: charge on 0.16, flattened 0.31

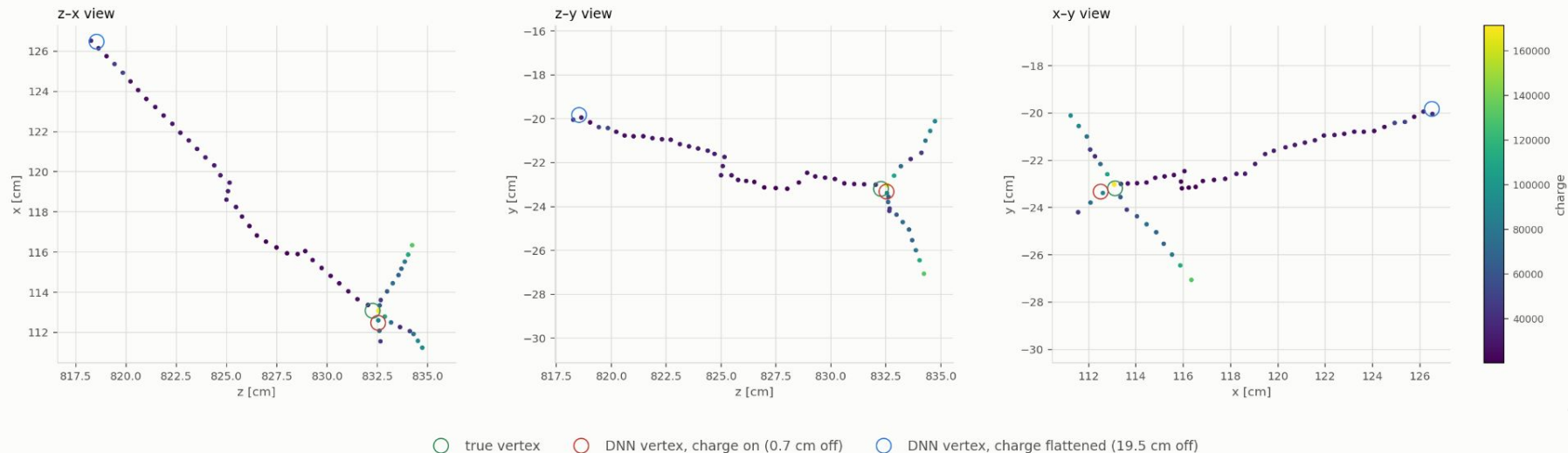
But charge is still useful in cases like this

In a sample of 696 uB open data events, where right = <1cm of truth

Q helps for ν_μ CC, but makes no net difference for ν_e CC

	Q off right	Q off wrong
Q on right	375	41
Q on wrong	28	293

MicroBooNE open data ν_μ CC, run 5557 / subrun 101 / event 5074 DNN confidence at the true vertex: charge on 0.81, flattened 0.03



Next step

- Do more thorough handscans and investigate failure modes
- Start writing SCE boundary fitting tech note

Commands for use of vertexers

Traditional:

- `wire-cell -l stderr -L info \`
 - `--tla-str input=/lus/eagle/projects/neutrinoGPU/snehadri/dlvtx/ubopen/pr_2v_filt/nue_WithWire_00-c000/ev0.npz \`
 - `--tla-str beezip=/lus/eagle/projects/neutrinoGPU/snehadri/dlvtx/ubopen/pr_sbndop_out/nue_WithWire_00-c000_ev0.zip \`
 - `--tla-code run_pr=true \`
 - `--tla-code sbnd_op=true \`
 - `--tla-str`
`dead_input=/lus/eagle/projects/neutrinoGPU/snehadri/dlvtx/ubopen/pr_2v_filt/nue_WithWire_00-c000/ev0-dead.npz \`
 - `--tla-str pr_sampler=charge_stepped \`
 - `--tla-code pr_tbins=1 \`
 - `--tla-code time_offset_us=-430.0 \`
 - `--tla-str dl_weights= \`
 - `-c wct-mabc-uboone.jsonnet`
- Chosen vertex comes out in the BEE zip

Commands for use of vertexers

- DNN:
 - Adapted from hybrid_vtx, Haiwang's code
 - `m = DeepVtx(dimension=3, nln=1, device="cpu"); m.train();`
`m.load_state_dict(torch.load(weights))` <- loading
 - `coords, feat = voxelize_fast(points, charge_features, 0.5)` <- voxelization
 - `score = m([torch.LongTensor(coords), torch.FloatTensor(feat[:, :1])])` <- inference
 - `dnn_vertex = coords[argmax(score[:,1] - score[:,0])] * 0.5 + points.min(0) + 0.25` <- vertex selection
- Hybrid is my custom code that just implements the logic described in the paper