Status report on

DNNROI sigproc & wirecell-dnn

Hokyeong Nam
Chung-Ang University

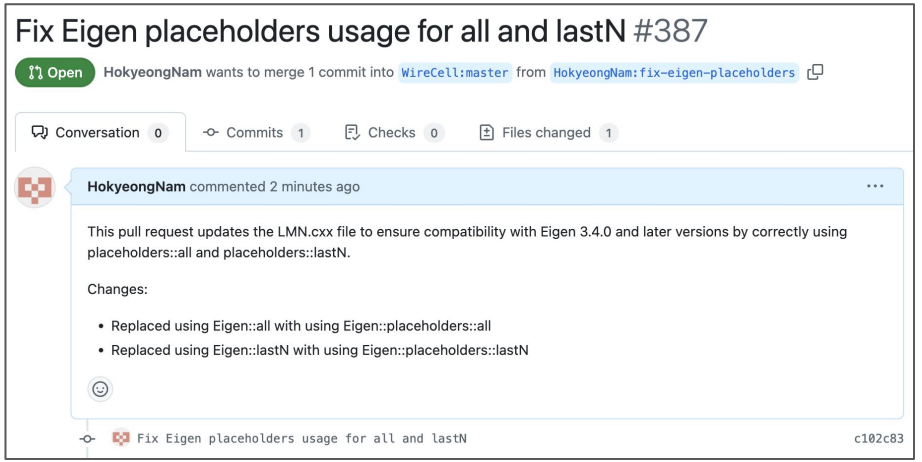
Outline

- Issue in WCT build
- DNNROI Sigproc.
  - WCT standalone simulation
  - PDHD data
- “wirecell-dnn” validation
  - Updated to-ts.py
- Summary & Plan

Issue in WCT build

```
[ 40/349] Compiling util/src/Bits.cxx
[ 41/349] Compiling util/src/LMN.cxx
[ 42/349] Compiling util/src/Array.cxx
[ 43/349] Compiling util/src/PointCloudArray.cxx
../util/src/LMN.cxx:26:16: error: ‘lastN’ has not been declared in ‘Eigen’
   26 |   using Eigen::lastN;
      |                ^~~~~
../util/src/LMN.cxx:27:16: error: ‘all’ has not been declared in ‘Eigen’
   27 |   using Eigen::all;
      |                ^~~
```

```
[822/825] Compiling util/test/test_wireschema_valid.cxx
[823/825] Linking build/util/test_wireschema_valid
[824/825] Linking build/util/test_wireschema_generate_microboone
[825/825] Linking build/util/test_util_stream
[826/826] Compiling wire-cell-toolkit.pc.in
Waf: Leaving directory `/exp/dune/data/users/hnam/wire-cell-hnam/dev/wire-cell-toolkit/build'
'build' finished successfully (23m27.628s)
```



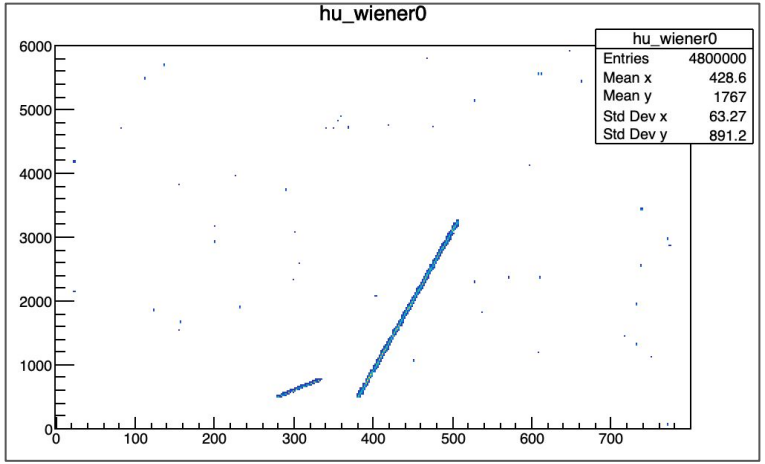
❖ Pull Request #387

- For the Eigen 3.4.0 and later version, Eigen::lastN and Eigen::all being deprecated
- The change has made on wire-cell-toolkit/util/src/LMN.cxx
- PR: https://github.com/WireCell/wire-cell-toolkit/pull/387

DNNROI SP - WCT standalone simulation - issues

```
91      local sp = g.pnode({
92          type: 'DepoFluxSplat',
93          name: sufix,
94          data: {
95              anode: wc.tn(anode),
96              field_response: wc.tn(tools.field), // for speed and origin
97              sparse: true,
98              tick: params.daq.tick,
99              window_start: params.sim.ductor.start_time,
100             window_duration: params.sim.ductor.readout_time,
101             reference_time: 0.0,
102             process_planes: [0,1,2],
103             // Run wirecell-gen morse-* to find these numbers that match the extra
104             // spread the sigproc induces.
105             "smear_long": [
106                 2.691862363980221,
107                 2.6750200122535057,
108                 2.7137567141154055
109             ],
110             "smear_tran": [
111                 0.7377218875719689,
112                 0.7157764520393882,
113                 0.13980698710556544
114             ]
115         },
116     }, nin=1, nout=1, uses=[anode, tools.field]),
```

❖ Edited funcs.jsonnet



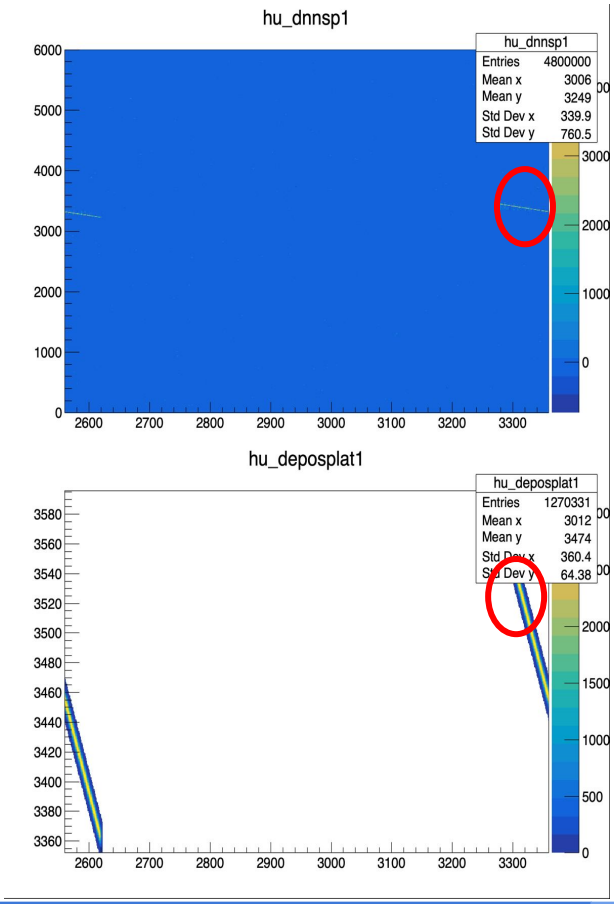
❖ Two tracks on u plane

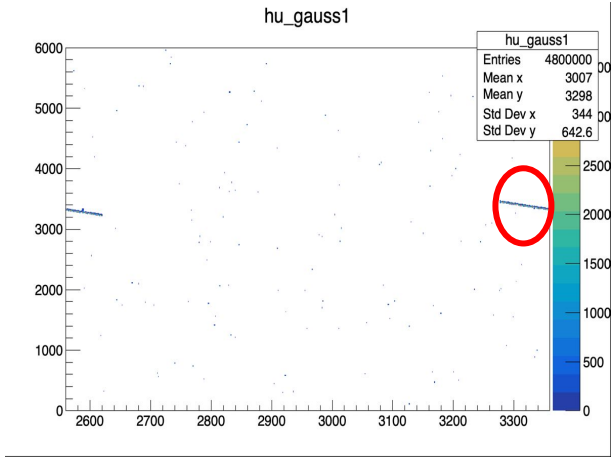
- Running WCT with “wct-sim-drift-deposplat.jsonnet” had solved issues as follows:
  - Out Of Memory (on gpvm) → Solved (unique trace problem)
  - Missing truth information → Partially solved (add sparse, process_planes on funcs.jsonnet)
- Remaining issues
  - Wrong channels and time ticks in truth hdf5 files →
  - Track cfg → One track is set at cfg file, but generates 2 tracks

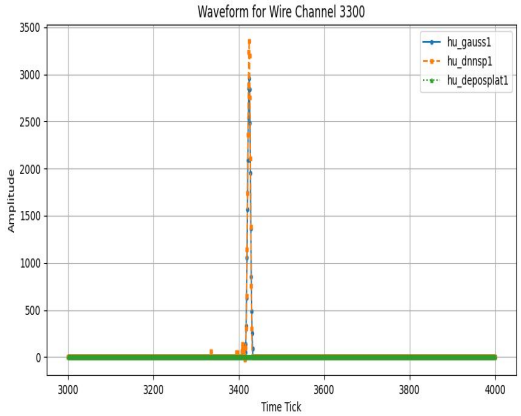
```
❯ h5ls g4-tru-1.h5/0
channels_deposplat1      Dataset {156035}
frame_deposplat1         Dataset {156035, 244}
tickinfo_deposplat1      Dataset {3}
```

❖ Wrong channels & ticks

DNNROI SP - WCT standalone simulation

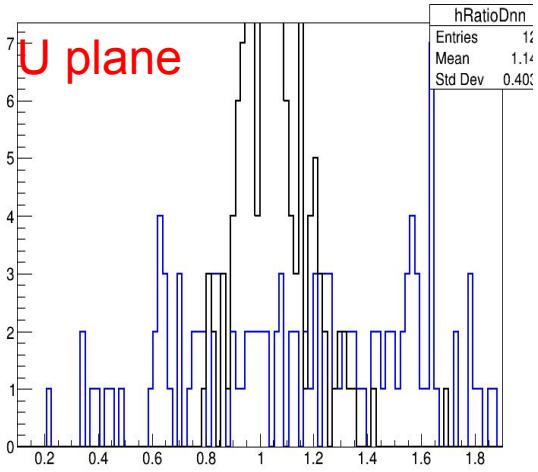


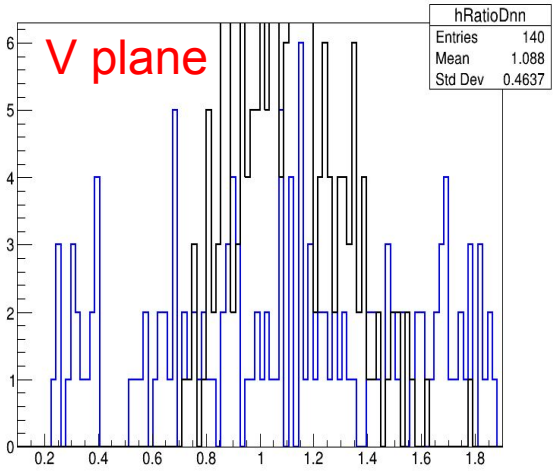




- We can access truth information (deposplat)
- Some errors in 1D waveform plotting code

DNNROI SP - WCT standalone simulation - evaluation






```
=== Bias
Gau bias (%): 5.50951
Dnn bias (%): 14.9479
=== Resolution
Gau RMS (%): 12.4873
Dnn RMS (%): 35.1132
=== Inefficiency
ntru: 143, bad ndnn: 19, bad ngau: 0
```

U plane

```
=== Bias
Gau bias (%): 11.2275
Dnn bias (%): 8.77239
=== Resolution
Gau RMS (%): 17.5455
Dnn RMS (%): 42.6279
=== Inefficiency
ntru: 192, bad ndnn: 52, bad ngau: 0
```

V plane

❖ Charge ratio

$$\text{Bias}_{\text{method}} = 100 \times (\text{Mean}(Charge_{\text{method}}/Charge_{\text{truth}}) - 1.0)$$

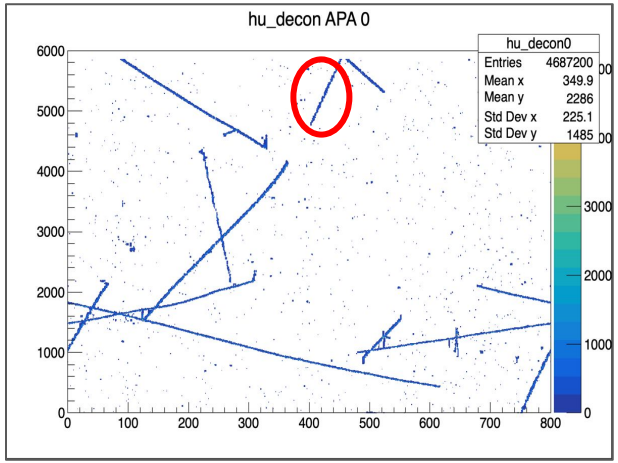
$$\text{Resolution} = 100 \times \frac{\text{RMS}(Charge_{method}/Charge_{truth})}{\text{Mean}(Charge_{method}/Charge_{truth})}$$

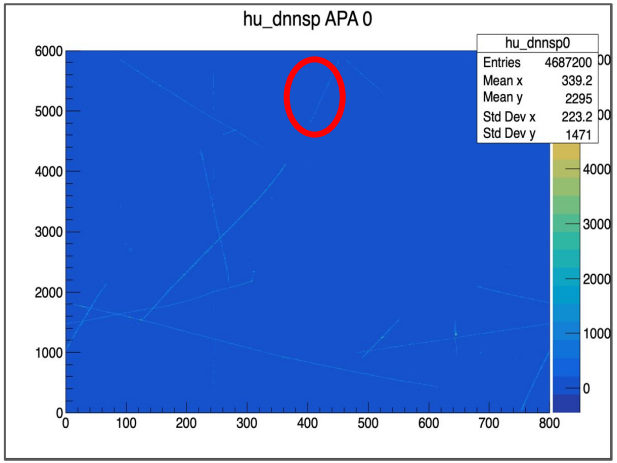
❖ Evaluation

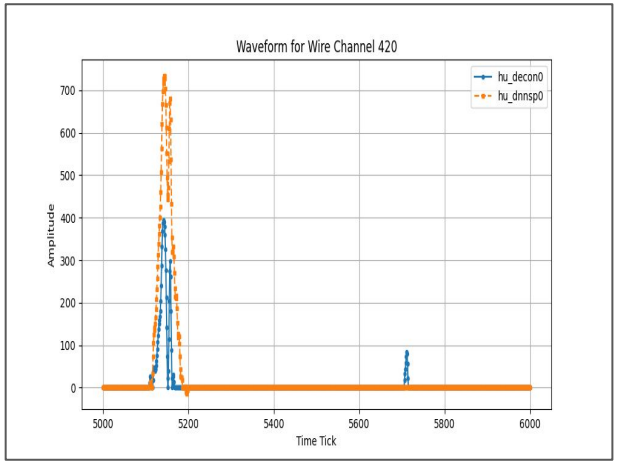
$$\text{hRatioGau} = \frac{\text{Charge}_{\text{Gaussian}}}{\text{Charge}_{\text{Truth}}}$$

$$\text{hRatioDnn} = \frac{\text{Charge}_{\text{DNN}}}{\text{Charge}_{\text{Truth}}}$$

DNNROI SP - PDHD data

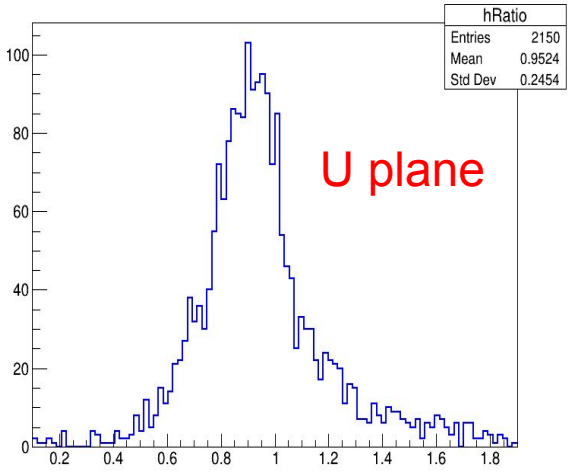


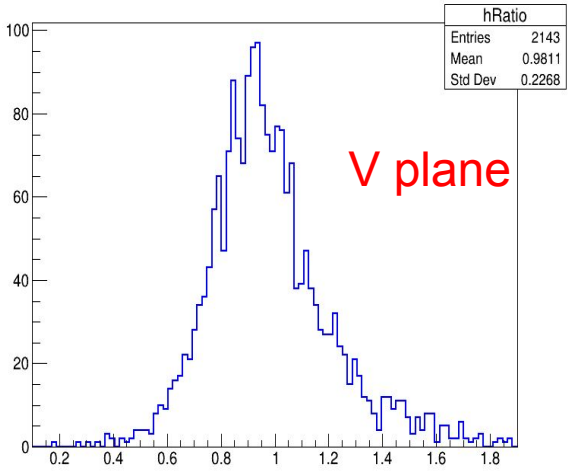




- PDHD data: /exp/dune/app/users/jjo/pdhd_imaging/np04_data/np04hd_raw_run026763_0008_dataflow0_datawriter_0_20240607T071013.hdf5

DNNROI SP - PDHD data - evaluation






```
=== Bias ===
DNN bias (%): -4.75702
=== Resolution ===
DNN RMS (%): 25.7666
=== Inefficiency ===
Total comparisons: 2311, Bad DNN: 161
DNN inefficiency (%): 6.96668 +/- 0.529581
```

U plane

```
=== Bias ===
DNN bias (%): -1.89038
=== Resolution ===
DNN RMS (%): 23.1194
=== Inefficiency ===
Total comparisons: 2353, Bad DNN: 210
DNN inefficiency (%): 8.92478 +/- 0.587744
```

V plane

❖ Charge ratio

$$\text{Bias}_{\text{DNN}} = 100 \times (\text{Mean}(Charge_{\text{DNN}}/Charge_{\text{Gaussian}}) - 1.0)$$

$$\text{Resolution} = 100 \times \frac{\text{RMS}(Charge_{DNN}/Charge_{Gaussian})}{\text{Mean}(Charge_{DNN}/Charge_{Gaussian})}$$

❖ Evaluation

$$\text{Inefficiency} = 100 \times \frac{\text{Bad Count}}{\text{Total Count}}$$

$$\text{hRatio} = \frac{\text{Charge}_{\text{DNN}}}{\text{Charge}_{\text{Gaussian}}}$$

Summary & Plan

- Solved the issue that standalone simulation was not able to save the truth information

- Compared the dnnsp to traditional sp within 2D histogram, 1D wave from, and evaluation
  (Both WCT standalone and PDHD data)

- Updated to-ts.py is now be able to convert .pth file to .ts file

- Next steps for DNNROI SP:
  - Debugging the remaining errors
  - Evaluate the performance of dnnroi with different angles (theta_XZ)
  - Measure the computing resources and time consumption during the dnnroi more specifically

- Next steps for wirecell-dnn validation:
  - Check model structures and loss between Pytorch-UNet & wirecell-dnn

Back Up

DNNROI solved issue: large memory usage

```
channels_break_roi_1st0  Dataset {2560}
channels_break_roi_2nd0  Dataset {2560}
channels_cleanup_roi0    Dataset {2560}
channels_decon_charge0   Dataset {2560}
channels_dnnsp0          Dataset {2560}
channels_extend_roi0     Dataset {2560}
channels_gauss0          Dataset {2560}
channels_loose_lf0       Dataset {2560}
channels_mp2_roi0        Dataset {2560}
channels_mp3_roi0        Dataset {2560}
channels_shrink_roi0     Dataset {2560}
channels_tight_lf0       Dataset {2560}
frame_break_roi_1st0     Dataset {6000, 2560}
frame_break_roi_2nd0     Dataset {6000, 2560}
frame_cleanup_roi0       Dataset {6000, 2560}
frame_decon_charge0      Dataset {6000, 2560}
frame_dnnsp0             Dataset {6000, 2560}
frame_extend_roi0        Dataset {6000, 2560}
frame_gauss0             Dataset {6000, 2560}
frame_loose_lf0          Dataset {6000, 2560}
frame_mp2_roi0           Dataset {6000, 2560}
frame_mp3_roi0           Dataset {6000, 2560}
frame_shrink_roi0        Dataset {6000, 2560}
frame_tight_lf0          Dataset {6000, 2560}
```

❖ Old files for the ML training

```
channels_break_roi_1st0  Dataset {3027}
channels_break_roi_2nd0  Dataset {3022}
channels_cleanup_roi0    Dataset {3023}
channels_decon_charge0   Dataset {582997}
channels_extend_roi0     Dataset {1266}
channels_gauss0          Dataset {2548}
channels_loose_lf0       Dataset {593611}
channels_mp2_roi0        Dataset {3864}
channels_mp3_roi0        Dataset {771}
channels_shrink_roi0     Dataset {3022}
channels_tight_lf0       Dataset {792153}
frame_break_roi_1st0     Dataset {3027, 5978}
frame_break_roi_2nd0     Dataset {3022, 5978}
frame_cleanup_roi0       Dataset {3023, 5978}
frame_decon_charge0      Dataset {582997, 6000}
frame_extend_roi0        Dataset {1266, 5734}
frame_gauss0             Dataset {2548, 5732}
frame_loose_lf0          Dataset {593611, 6000}
frame_mp2_roi0           Dataset {3864, 2785}
frame_mp3_roi0           Dataset {771, 2785}
frame_shrink_roi0        Dataset {3022, 5978}
frame_tight_lf0          Dataset {792153, 6000}
```

❖ Generated bad files

```
channels_break_roi_1st0  Dataset {2560}
channels_break_roi_2nd0  Dataset {2560}
channels_cleanup_roi0    Dataset {2560}
channels_decon_charge0   Dataset {2560}
channels_extend_roi0     Dataset {2560}
channels_gauss0          Dataset {2560}
channels_loose_lf0       Dataset {2560}
channels_mp2_roi0        Dataset {2560}
channels_mp3_roi0        Dataset {2560}
channels_shrink_roi0     Dataset {2560}
channels_tight_lf0       Dataset {2560}
frame_break_roi_1st0     Dataset {2560, 6000}
frame_break_roi_2nd0     Dataset {2560, 6000}
frame_cleanup_roi0       Dataset {2560, 6000}
frame_decon_charge0      Dataset {2560, 6000}
frame_extend_roi0        Dataset {2560, 6000}
frame_gauss0             Dataset {2560, 6000}
frame_loose_lf0          Dataset {2560, 6000}
frame_mp2_roi0           Dataset {2560, 6000}
frame_mp3_roi0           Dataset {2560, 6000}
frame_shrink_roi0        Dataset {2560, 6000}
frame_tight_lf0          Dataset {2560, 6000}
```

❖ Good files after debugging

Different ordering is convention by Haiwang and Sergey

- Sigproc w/ dnnroi (wct-sim-drift-deposplat.jsonnet) raised a memory problem
  - MEM ~40 GB, the job is killed by system
- Succeed to run on the WC cluster, the output file showed wrong wire channel & time tick
- Debugged with Haiwang and fixed the issue in **unique trace** (https://github.com/WireCell/wire-cell-toolkit/commit/83ac1e8070289b80daa68619a1aaa47cbc03bf0e)

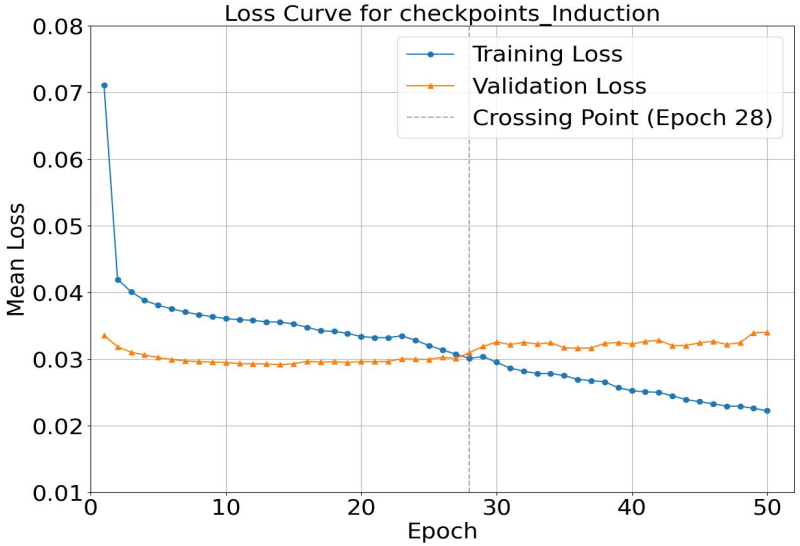
wirecell-dnn setup & training

```
Create dataset in 2.135e+02 s
Total training time: 764.51 seconds
```

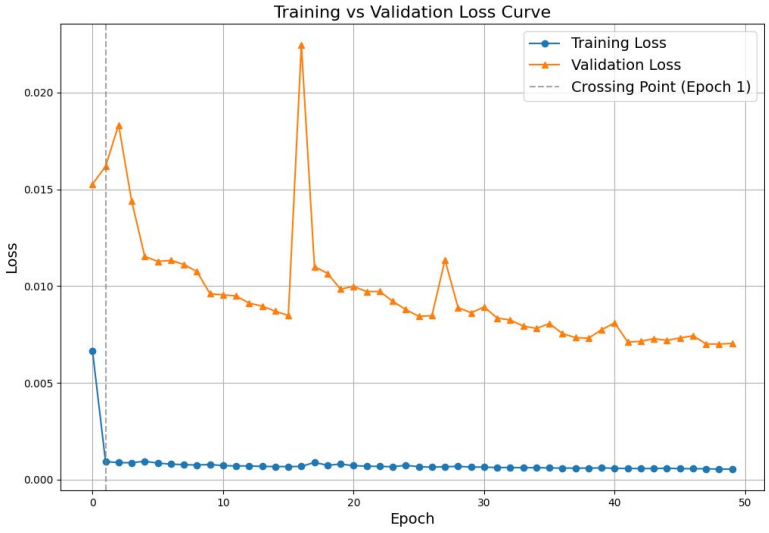
- Training UNet with wirecell-dnn and train3.py on the WC cluster w/ full (118) hdf5 files
  - Total elapsed time (wirecell-dnn): **18m 01s**
  - Total elapsed time (train3.py): **25m 19s**

- Dataset: /nfs/data/1/hnam/train_data_PDHD_fixedbug_separateWC
- Epochs: 50
- Batch Size: 10
- Train Ratio: 0.9, Validation Ratio: 0.1
- Learning Rate: 0.1

Issues: Train vs Val loss curve



❖ modell from train3.py



❖ model from wirecell-dnn train

Issues: .pth file structure diff.

```
>>> print(data.keys())
odict_keys(['inc.conv.conv.0.weight', 'inc.conv.conv.0.bias', 'inc.conv.conv.1.w
eight', 'inc.conv.conv.1.bias', 'inc.conv.conv.1.running_mean', 'inc.conv.conv.1
.running_var', 'inc.conv.conv.1.num_batches_tracked', 'inc.conv.conv.3.weight',
'inc.conv.conv.3.bias', 'inc.conv.conv.4.weight', 'inc.conv.conv.4.bias', 'inc.c
onv.conv.4.running_mean', 'inc.conv.conv.4.running_var', 'inc.conv.conv.4.num_ba
tches_tracked', 'down1.mpconv.1.conv.0.weight', 'down1.mpconv.1.conv.0.bias', 'd
own1.mpconv.1.conv.1.weight', 'down1.mpconv.1.conv.1.bias', 'down1.mpconv.1.conv
.1.running_mean', 'down1.mpconv.1.conv.1.running_var', 'down1.mpconv.1.conv.1.nu
m_batches_tracked', 'down1.mpconv.1.conv.3.weight', 'down1.mpconv.1.conv.3.bias'
, 'down1.mpconv.1.conv.4.weight', 'down1.mpconv.1.conv.4.bias', 'down1.mpconv.1.
conv.4.running_mean', 'down1.mpconv.1.conv.4.running_var', 'down1.mpconv.1.conv.
4.num_batches_tracked', 'down2.mpconv.1.conv.0.weight', 'down2.mpconv.1.conv.0.b
ias', 'down2.mpconv.1.conv.1.weight', 'down2.mpconv.1.conv.1.bias', 'down2.mpcon
v.1.conv.1.running_mean', 'down2.mpconv.1.conv.1.running_var', 'down2.mpconv.1.c
```

```
>>> print(data.keys())
dict_keys(['runs', 'epochs', 'model_state_dict', 'optimizer_state_dict'])
>>> print(data["model_state_dict"].keys())
odict_keys(['unet.down_dconv_0.0.weight', 'unet.down_dconv_0.0.bias', 'unet.down
_dconv_0.1.weight', 'unet.down_dconv_0.1.bias', 'unet.down_dconv_0.1.running_mea
n', 'unet.down_dconv_0.1.running_var', 'unet.down_dconv_0.1.num_batches_tracked'
, 'unet.down_dconv_0.3.weight', 'unet.down_dconv_0.3.bias', 'unet.down_dconv_0.4
.weight', 'unet.down_dconv_0.4.bias', 'unet.down_dconv_0.4.running_mean', 'unet.
down_dconv_0.4.running_var', 'unet.down_dconv_0.4.num_batches_tracked', 'unet.do
wn_dconv_1.0.weight', 'unet.down_dconv_1.0.bias', 'unet.down_dconv_1.1.weight',
'unet.down_dconv_1.1.bias', 'unet.down_dconv_1.1.running_mean', 'unet.down_dconv
_1.1.running_var', 'unet.down_dconv_1.1.num_batches_tracked', 'unet.down_dconv_1
.3.weight', 'unet.down_dconv_1.3.bias', 'unet.down_dconv_1.4.weight', 'unet.down
```

- The model trained with wirecell-dnn has more info.
- Naming convention and structure differences → revising to-ts.py