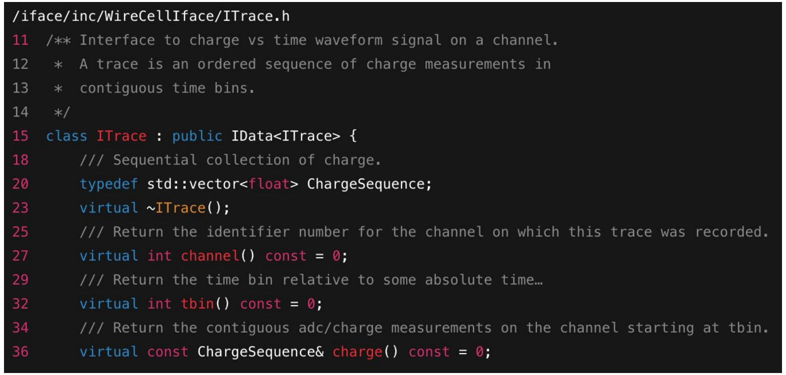
Status report on
DNNROI sigproc

Hokyeong Nam
Chung-Ang University

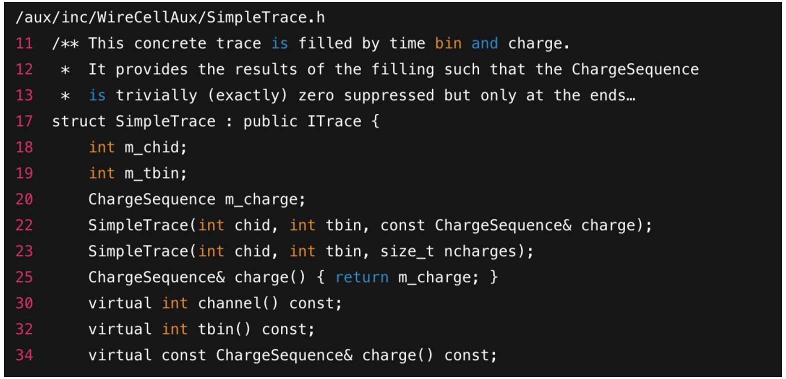
Outline

- WCT Signal Processing
  - Data Product
  - OmnibusSigProc.cxx
- Performance Evaluation w/ Single Track & Shower
- DNN training with rebining

Data Product

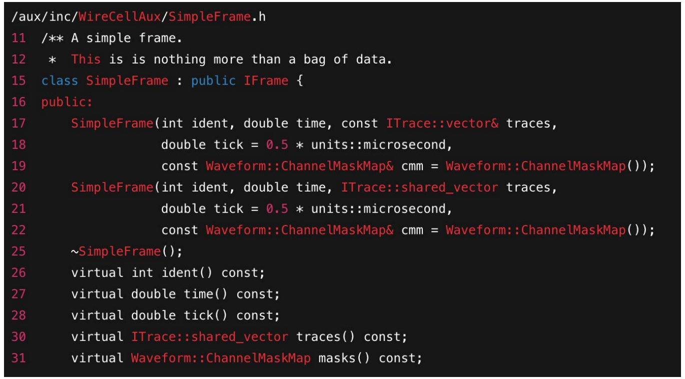


```
/iface/inc/WireCellIface/ITrace.h
/** Interface to charge vs time waveform signal on a channel.
 *  A trace is an ordered sequence of charge measurements in
 *  contiguous time bins.
 */
class ITrace : public IData<ITrace> {
    /// Sequential collection of charge.
    typedef std::vector<float> ChargeSequence;
    virtual ~ITrace();
    /// Return the identifier number for the channel on which this trace was recorded.
    virtual int channel() const = 0;
    /// Return the time bin relative to some absolute time…
    virtual int tbin() const = 0;
    /// Return the contiguous adc/charge measurements on the channel starting at tbin.
    virtual const ChargeSequence& charge() const = 0;
```



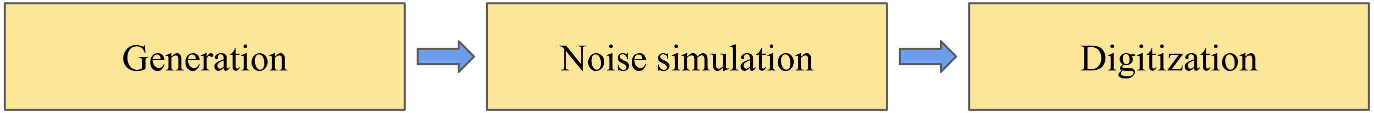
```
/aux/inc/WireCellAux/SimpleTrace.h
/** This concrete trace is filled by time bin and charge.
 *  It provides the results of the filling such that the ChargeSequence
 *  is trivially (exactly) zero suppressed but only at the ends…
struct SimpleTrace : public ITrace {
    int m_chid;
    int m_tbin;
    ChargeSequence m_charge;
    SimpleTrace(int chid, int tbin, const ChargeSequence& charge);
    SimpleTrace(int chid, int tbin, size_t ncharges);
    ChargeSequence& charge() { return m_charge; }
    virtual int channel() const;
    virtual int tbin() const;
    virtual const ChargeSequence& charge() const;
```

- The WCT represents waveform data through the ITrace interface
  - Single channel’s contiguous ADC/charge samples
- Traces are gathered into an IFrame
  - A container of traces
- In process, individual traces are produced by creating SimpleTrace objects and appending them to the frame’s trace list



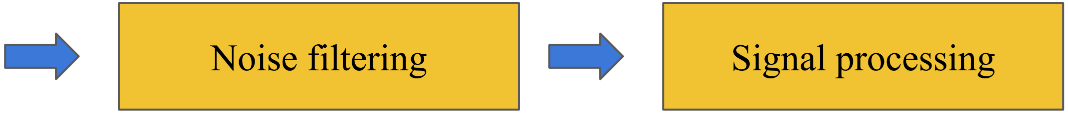
```
/aux/inc/WireCellAux/SimpleFrame.h
/** A simple frame.
 *  This is is nothing more than a bag of data.
class SimpleFrame : public IFrame {
public:
    SimpleFrame(int ident, double time, const ITrace::vector& traces,
                double tick = 0.5 * units::microsecond,
                const Waveform::ChannelMaskMap& cmm = Waveform::ChannelMaskMap());
    SimpleFrame(int ident, double time, ITrace::shared_vector traces,
                double tick = 0.5 * units::microsecond,
                const Waveform::ChannelMaskMap& cmm = Waveform::ChannelMaskMap());
    ~SimpleFrame();
    virtual int ident() const;
    virtual double time() const;
    virtual double tick() const;
    virtual ITrace::shared_vector traces() const;
    virtual Waveform::ChannelMaskMap masks() const;
```

Data Product



Generation → Noise simulation → Digitization

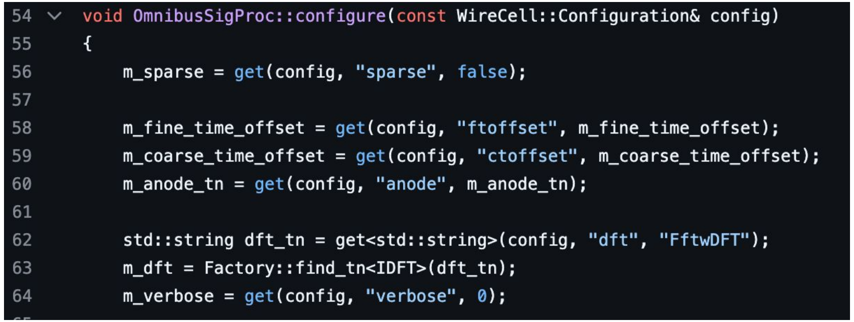
- The relevant files are in /gen/src + processing pipeline is defined in sim.jsonnet
- Energy deposits (depos) are drifted and transformed into analog voltages on each wire plane (e.g. DepoFramer, Ductor, DepoTransform)
- After Noise simulation (AddNoise), the Digitizer converts the analog voltages to ADC counts, creating raw traces in an IFrame



→ Noise filtering → Signal processing

- The relevant files are in /sigproc/src + processing pipeline is defined in sp.jsonnet
- OmnibusNoiseFilter and related filter nodes operate on the ADC traces. Mask information from the noise file and per-channel/group filters is applied, outputting cleaned traces tagged as **"raw"**
- **OmnibusSigProc** performs FFT-based deconvolution using the configured field and electronics responses. It forms and refines ROIs, restores baselines, and saves processed waveforms tagged (e.g. gauss, winner)

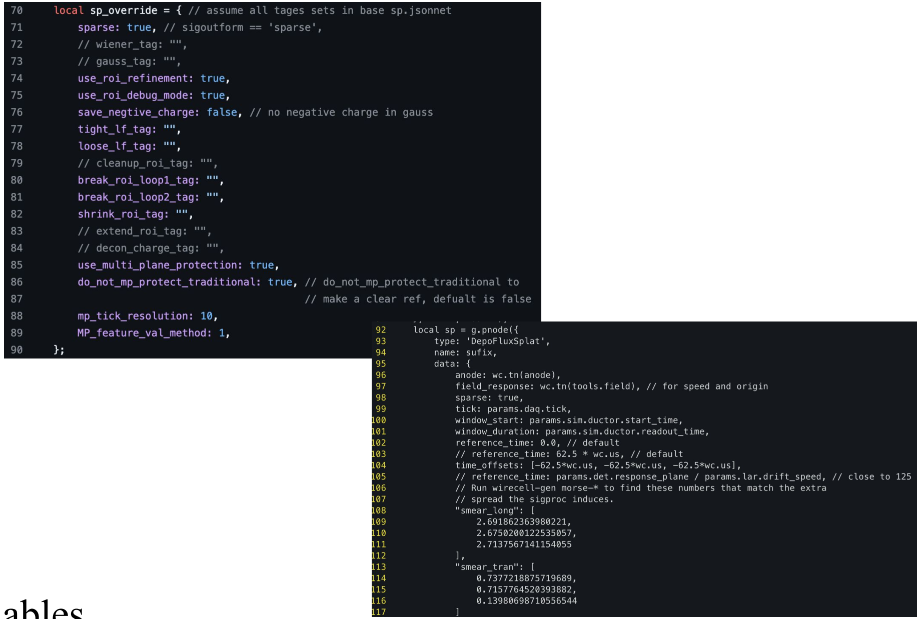
OmnibusSigProc - configure, default_configuration



```
void OmnibusSigProc::configure(const WireCell::Configuration& config)
{
    m_sparse = get(config, "sparse", false);

    m_fine_time_offset = get(config, "ftoffset", m_fine_time_offset);
    m_coarse_time_offset = get(config, "ctoffset", m_coarse_time_offset);
    m_anode_tn = get(config, "anode", m_anode_tn);

    std::string dft_tn = get<std::string>(config, "dft", "FftwDFT");
    m_dft = Factory::find_tn<IDFT>(dft_tn);
    m_verbose = get(config, "verbose", 0);
```

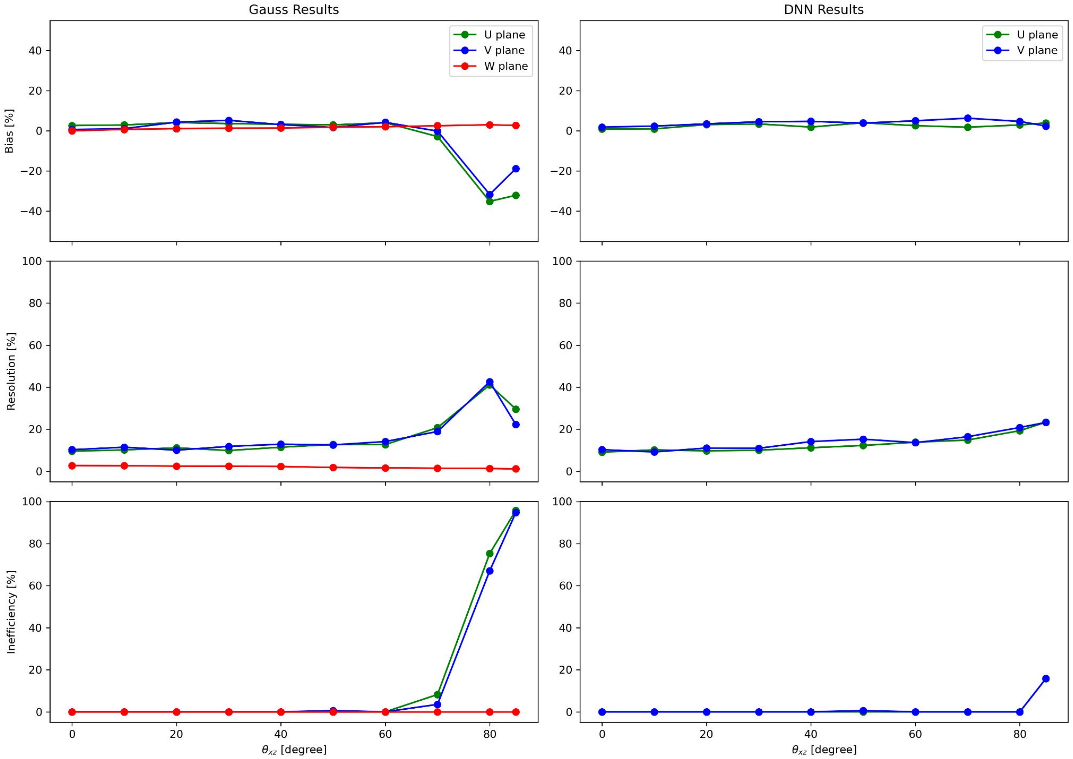


```
local sp_override = { // assume all tages sets in base sp.jsonnet
    sparse: true, // sigoutform == 'sparse',
    // wiener_tag: "",
    // gauss_tag: "",
    use_roi_refinement: true,
    use_roi_debug_mode: true,
    save_negtive_charge: false, // no negative charge in gauss
    tight_lf_tag: "",
    loose_lf_tag: "",
    // cleanup_roi_tag: "",
    break_roi_loop1_tag: "",
    break_roi_loop2_tag: "",
    shrink_roi_tag: "",
    // extend_roi_tag: "",
    // decon_charge_tag: "",
    use_multi_plane_protection: true,
    do_not_mp_protect_traditional: true, // do_not_mp_protect_traditional to
                                         // make a clear ref, defualt is false
    mp_tick_resolution: 10,
    MP_feature_val_method: 1,
};
```

```
local sp = g.pnode({
    type: 'DepoFluxSplat',
    name: sufix,
    data: {
        anode: wc.tn(anode),
        field_response: wc.tn(tools.field), // for speed and origin
        sparse: true,
        tick: params.daq.tick,
        window_start: params.sim.ductor.start_time,
        window_duration: params.sim.ductor.readout_time,
        reference_time: 0.0, // default
        // reference_time: 62.5 * wc.us, // default
        time_offsets: [-62.5*wc.us, -62.5*wc.us, -62.5*wc.us],
        // reference_time: params.det.response_plane / params.lar.drift_speed, // close to 125
        // Run wirecell-gen morse-* to find these numbers that match the extra
        // spread the sigproc induces.
        "smear_long": [
            2.691862363980221,
            2.6750200122535057,
            2.7137567141154055
        ],
        "smear_tran": [
            0.7377218875719689,
            0.7157764520393882,
            0.13980698710556544
        ]
```

- Reads Json/Jsonnet values and and sets internal variables
    - thresholds, filter names, tag values, channel mappings, etc
- They are usually defined in the main configuration file, params.jsonnet, and funcs.jsonnet
- Returns settings as a defaults if there are no settings specified

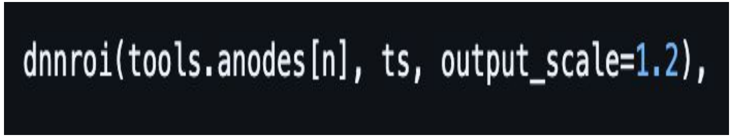
DNN ROI evaluation with track



- The graph shows a downward turn at 85 due to insufficient statistics

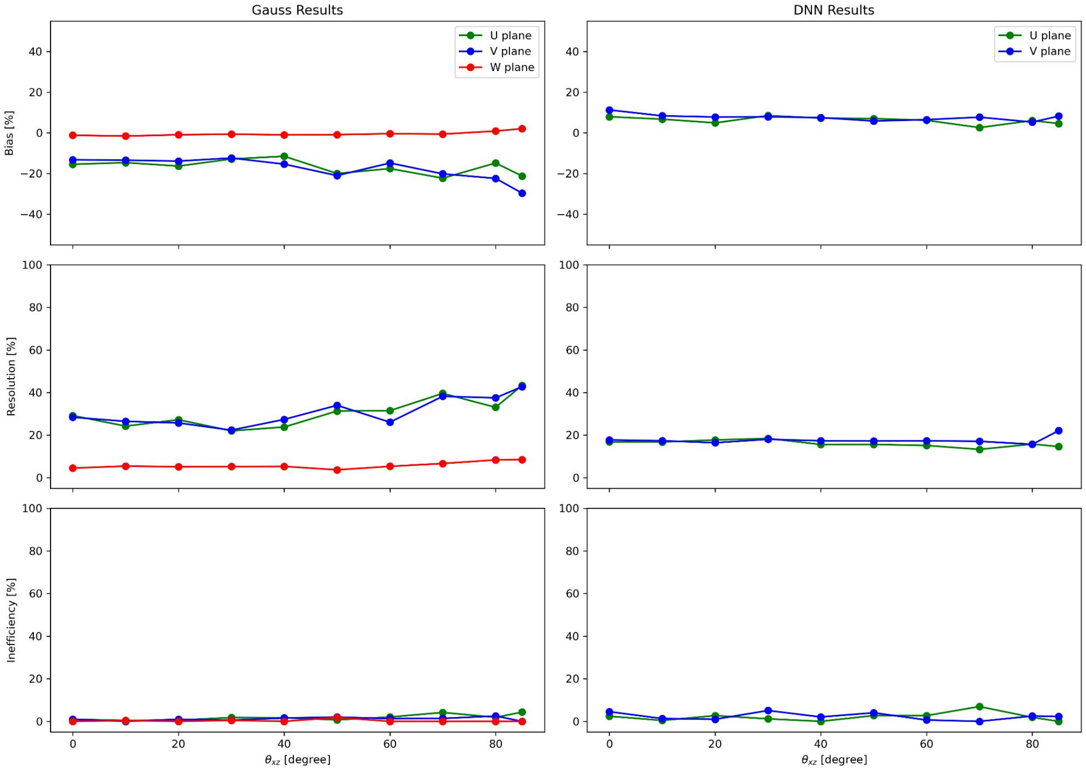
- Modified the total charge calculation function to integrate only within ± 100 time ticks around peak charge

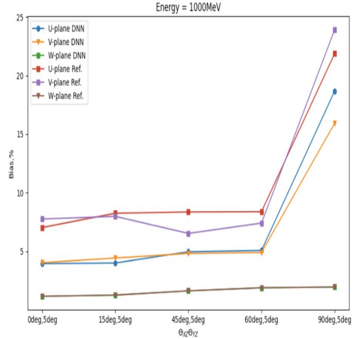
- Set the dnnroi output_scale=1.0

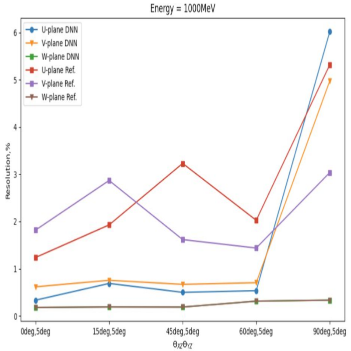


```
dnnroi(tools.anodes[n], ts, output_scale=1.2),
```

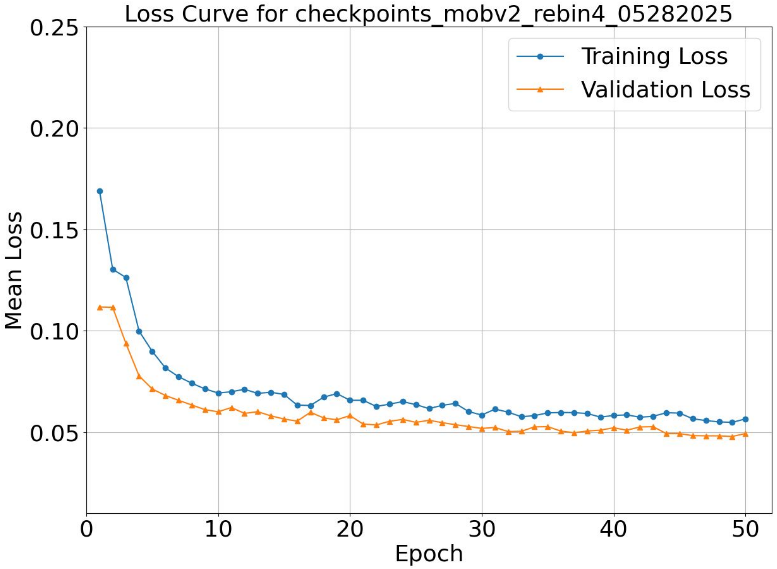
DNN ROI evaluation with shower

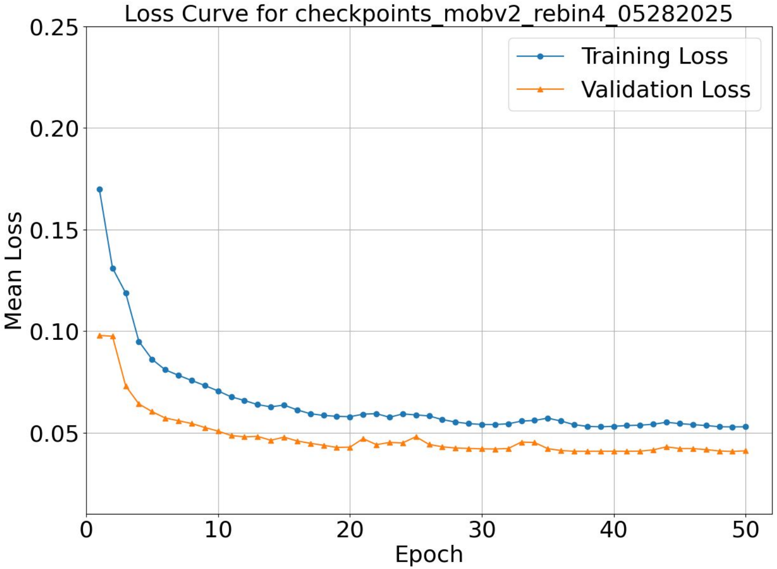






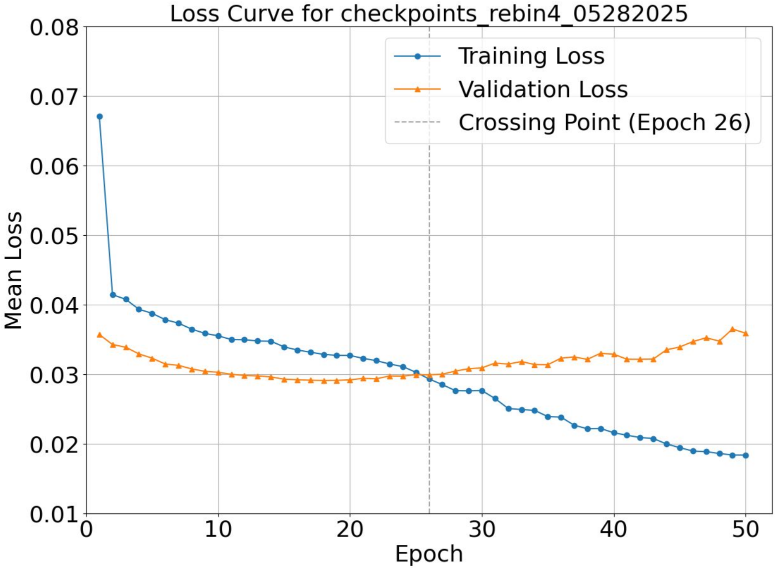
DNN Training with rebining - MobileNetV2

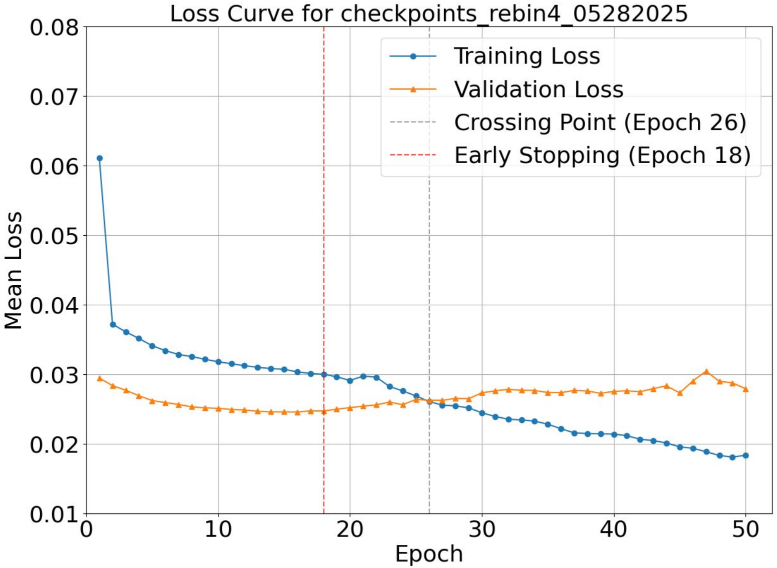




- Rebin in time tick 10 → 4
- Loss: Binary Cross-Entropy, Epoch = 50, Learning Rate = 0.1, Momentum = 0.9, train vs val split = 90:10
- Loss at the last epoch is slightly improved: 0.05 (previously 0.06)

DNN Training with rebining - UNet

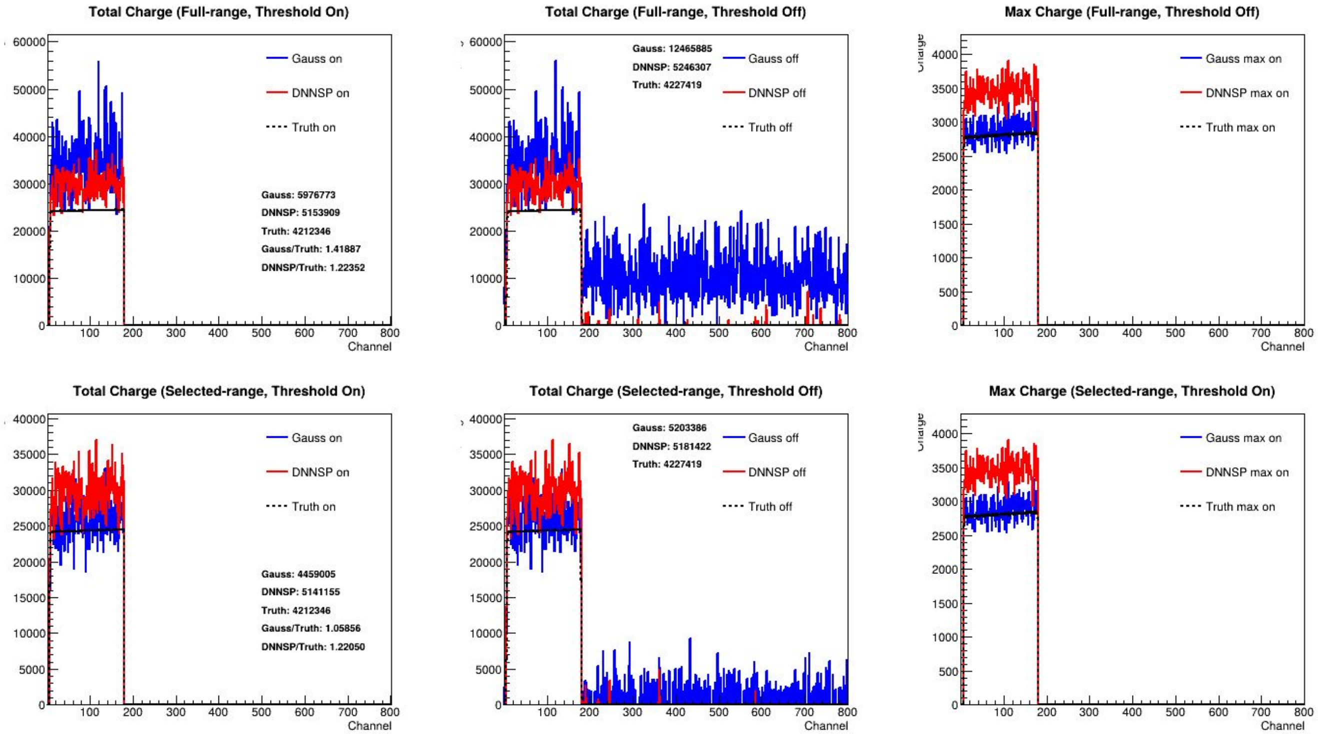




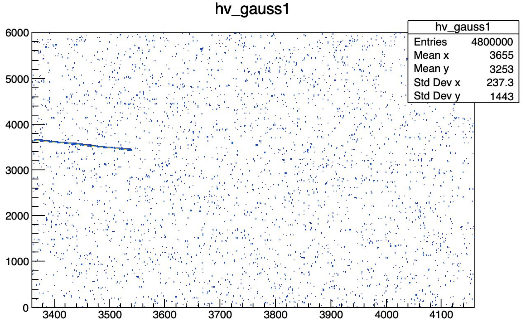
- Rebin in time tick 10 → 4
- Loss: Binary Cross-Entropy, Epoch = 50, Learning Rate = 0.1, Momentum = 0.9, train vs val split = 90:10
- Loss at the crossing point is slightly improved: 0.026 (previously 0.03)

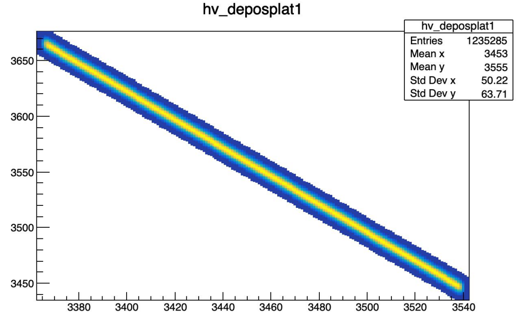
Back Up

DNN ROI evaluation with fixed time offset



- Total charge shows the Bias is due to the noise





Memory and Time consumption on the WC cluster

Server	WCT	Resource	DNN ROI	Mem (MB)	Time (s)	Mem Ratio	Time Ratio
WC cluster	dunesw	None	None	1891.54	40.41	1.00	1.00
WC cluster	dunesw	CPU	UNet	7419.03	91.45	3.92	2.26
WC cluster	dunesw	CPU	MobileNetV2	4593.25	54.51	2.43	1.35
WC cluster	Built	None	None	1890.80	40.97	0.99	1.01
WC cluster	Built	CPU	UNet	5208.58	53.64	2.75	1.33
WC cluster	Built	CPU	MobileNetV2	4853.49	45.41	2.56	1.12
WC cluster	Built	GPU	UNet	5105.16	46.18	2.70	1.14
WC cluster	Built	GPU	MobileNetV2	5110.95	45.33	2.70	1.12

- Model file used: unet-cosmic390-newwc-depofluxsplat-pdhd.ts
- Signal Processing only & All measurements are averaged over 5 PD-HD data files
  - run026763_0008, run027673_0000, run027673_0001, run027673_0002, run028588_0019
- Using the custom-built WCT reduced the CPU inference time from 91.45 s to 53.64 s: ~41% improvement