SPNG Tensor Data Model and Classes

Brett Viren

December 12, 2025

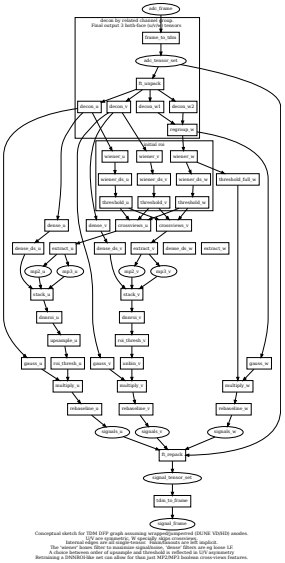
Topics

- Status of the “TDM” refactoring.
- Performance
- Issues found along the way.

How it started

Updated sketch: U/V vs W assymetry.

- U+V sees DNNROI
  - CrossViews with MP2/MP3 extraction.
  - Reflect 2 choices in ROI processing (more later)
- W has direct ROI on Wiener

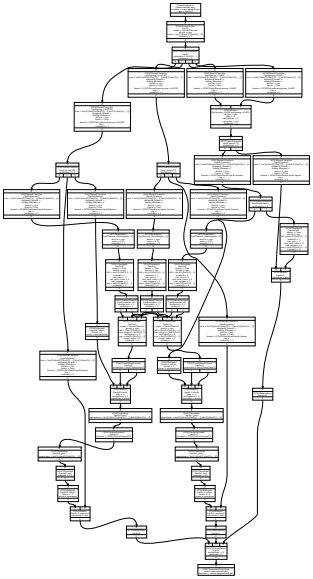


How it's going

CLI parameterized ADC->signal SPNG graph.

- "Any" detector (as long as it is PDHD)
- CPU (`"cpu"`) or GPU (`"gpu"`, `"gpu1"`)
- Pgrapher (ST) or TbbFlow (MT)
- Semaphore count (MT + GPU).

`test-spng-tpc.jsonnet` and `spng/cfg/spng/*.jsonnet`



Example job

Generate ADC files[1]

```
wire-cell spng/test-detsim.jsonnet \
  -A detector=pdhd \
  -A input=muon-depos-moved.npz \
  -A output=muon-anode%d.npz
```

Run SPNG

```
wire-cell spng/test-spng-tpc.jsonnet \
  -A detector=pdhd \
  -A device=gpu \
  -A tpcid=0 \
  -A input=muon-anode0.npz \
  -A output=muon-crossviews-anode0.pkl
```

[1] Caveat: as of writing, I see I have broken the test-detsim.jsonnet. Will fix.

Performance

	wall	core	time	sem
pgrapher cpu	3.6	54.3	6.2	
tbbflow cpu	6.3	190.5	5.4	
pgrapher gpu	3.5	5.4	6.5	
tbbflow gpu	8.4	19.1	6.0	1
tbbflow gpu	7.2	20.4	5.8	2
tbbflow gpu	5.3	16.0	5.7	4

All times are in seconds.

- **core** is `std::clock`, counts core-seconds.
- **wall** is `<chrono>` counts real time per node.
- **time** is fish shell's `time` program, overall job wall time.

Semaphore count only matters with TbbFlow + GPU.
Reminder: Pgrapher is single threaded, TbbFlow is MT.

Comments on performance

- Single thread **Pgrapher** does well, especially in core-time.
  - But: the job has low parallelism at WCT graph level.
  - And: Torch CPU parallelism was not constrained.
    - Average 15 cores for “pgrapher cpu”.
- GPU gives about $10\times$ acceleration compared to single-core CPU.
  - **Meets my hope/goal**
  - But, I think we can do better!
    - Low GPU RAM usage: forget chunking, let’s batch!

Caveat, tests run without compiler optimization of WCT!

Top time takers With Pgrapher, CPU:

```
0.787 wall-sec, 0.782 core-sec: (WireCell::Sio::FrameFileSource) "muon-anode0.npz"
0.699 wall-sec, 21.434 core-sec: (WireCell::SPNG::TensorForward) "tpc0v1"              (<--DNNROI inference)
0.668 wall-sec, 21.085 core-sec: (WireCell::SPNG::TensorForward) "tpc0v0"
0.232 wall-sec, 1.112 core-sec: (WireCell::SPNG::CrossViews) "tpc0v0"                  (<--MP2/MP3 prep)
0.228 wall-sec, 1.172 core-sec: (WireCell::SPNG::CrossViews) "tpc0v1"
0.212 wall-sec, 0.395 core-sec: (WireCell::SPNG::TensorSetPickleSink) "muon-crossviews-anode0.pkl"
0.210 wall-sec, 0.489 core-sec: (WireCell::SPNG::TensorSetPickleSink) "tensors-decon-0.pkl"
0.209 wall-sec, 0.384 core-sec: (WireCell::SPNG::TensorSetPickleSink) "tensors-unpack-0.pkl"
0.041 wall-sec, 0.203 core-sec: (WireCell::SPNG::Rebaseliner) "tpc0v2"                 (<--apply ROI)
0.034 wall-sec, 0.766 core-sec: (WireCell::SPNG::FrameToTdm) "tpc0"
0.023 wall-sec, 0.162 core-sec: (WireCell::SPNG::Rebaseliner) "tpc0v1"
0.023 wall-sec, 0.162 core-sec: (WireCell::SPNG::Rebaseliner) "tpc0v0"
0.023 wall-sec, 0.739 core-sec: (WireCell::SPNG::KernelConvolve) "tpc0group_v1c2f1f0" (<--2D decon)
0.022 wall-sec, 0.643 core-sec: (WireCell::SPNG::KernelConvolve) "tpc0group_v0c2f1f0"
0.017 wall-sec, 0.484 core-sec: (WireCell::SPNG::KernelConvolve) "tpc0group_v2c0f1"
0.014 wall-sec, 0.352 core-sec: (WireCell::SPNG::KernelConvolve) "tpc0v0wiener"
0.014 wall-sec, 0.202 core-sec: (WireCell::SPNG::TensorSetPickleSink) "tensors-threshold-0.pkl"
0.013 wall-sec, 0.312 core-sec: (WireCell::SPNG::KernelConvolve) "tpc0v2wiener"
0.012 wall-sec, 0.304 core-sec: (WireCell::SPNG::KernelConvolve) "tpc0v1dnnroi"
0.012 wall-sec, 0.432 core-sec: (WireCell::SPNG::KernelConvolve) "tpc0v1wiener"
0.010 wall-sec, 0.232 core-sec: (WireCell::SPNG::KernelConvolve) "tpc0v0dnnroi"
0.010 wall-sec, 0.373 core-sec: (WireCell::SPNG::KernelConvolve) "tpc0group_v2c0f0"
0.006 wall-sec, 0.255 core-sec: (WireCell::SPNG::KernelConvolve) "tpc0v2gauss"
0.006 wall-sec, 0.255 core-sec: (WireCell::SPNG::KernelConvolve) "tpc0v1gauss"
```

Top time takers With Pgrapher, GPU:

```
0.791 wall-sec, 0.791 core-sec: (WireCell::Sio::FrameFileSource) "muon-anode0.npz"
0.650 wall-sec, 0.825 core-sec: (WireCell::SPNG::CrossViews) "tpc0v1"                  (<--MP2/MP3 prep)
0.622 wall-sec, 0.805 core-sec: (WireCell::SPNG::CrossViews) "tpc0v0"
0.212 wall-sec, 0.293 core-sec: (WireCell::SPNG::TensorSetPickleSink) "tensors-decon-0.pkl"
0.211 wall-sec, 0.212 core-sec: (WireCell::SPNG::TensorSetPickleSink) "tensors-unpack-0.pkl"
0.209 wall-sec, 0.210 core-sec: (WireCell::SPNG::TensorSetPickleSink) "muon-crossviews-anode0.pk
0.191 wall-sec, 0.191 core-sec: (WireCell::SPNG::TensorForward) "tpc0v1"               (<--DNNROI inference)
0.140 wall-sec, 0.141 core-sec: (WireCell::SPNG::Rebaseliner) "tpc0v2"                 (<--apply ROI)
0.078 wall-sec, 0.079 core-sec: (WireCell::SPNG::Rebaseliner) "tpc0v0"
0.067 wall-sec, 0.068 core-sec: (WireCell::SPNG::Rebaseliner) "tpc0v1"
0.040 wall-sec, 0.041 core-sec: (WireCell::SPNG::TensorForward) "tpc0v0"
0.037 wall-sec, 0.822 core-sec: (WireCell::SPNG::FrameToTdm) "tpc0"
0.013 wall-sec, 0.014 core-sec: (WireCell::SPNG::TensorSetPickleSink) "tensors-threshold-0.pkl"
0.009 wall-sec, 0.161 core-sec: (WireCell::SPNG::KernelConvolve) "tpc0group_v2c0f1"    (<--2D decon)
0.008 wall-sec, 0.132 core-sec: (WireCell::SPNG::Threshold) "tpc0wfull"
0.004 wall-sec, 0.005 core-sec: (WireCell::SPNG::KernelConvolve) "tpc0v2wiener"
0.004 wall-sec, 0.005 core-sec: (WireCell::SPNG::KernelConvolve) "tpc0group_v0c2f1f0"
0.004 wall-sec, 0.005 core-sec: (WireCell::SPNG::KernelConvolve) "tpc0v0wiener"
0.004 wall-sec, 0.005 core-sec: (WireCell::SPNG::KernelConvolve) "tpc0v1wiener"
0.004 wall-sec, 0.014 core-sec: (WireCell::SPNG::KernelConvolve) "tpc0v1dnnroi"
0.004 wall-sec, 0.012 core-sec: (WireCell::SPNG::KernelConvolve) "tpc0v0dnnroi"
0.004 wall-sec, 0.023 core-sec: (WireCell::SPNG::KernelConvolve) "tpc0group_v1c2f1f0"
0.001 wall-sec, 0.002 core-sec: (WireCell::SPNG::KernelConvolve) "tpc0v2gauss"
```

Issues along the way

- DNNROI output processing order.
- ~~Possible bug in how DNNROI inference has been done so far.~~
  - This was a misunderstanding on my part.

DNNROI output processing order

DNNROI produces a **floating point** image roughly bounded in [0.0, 1.0] requires **3 operations**

- T Must **threshold** FP values to obtain **binary** {0,1} mask.
- U Must **upsample/unrebin**[2] by $4\times$ to undo upstream rebin.
- A Must **apply the ROIs** to the Gauss-filtered decon.

May choose at least any of these orderings:

T+A+U *or* ***U+T+A*** *or* ***T+U+A***

[2] I say (un)rebin to imply interval domain operation, {up,down}sample means FFT resample method.

DNNROI output processing order implications

T+A+U
- cons: Must down-sample Gauss, ROI sizes multiples of 4.
- pros: no possible ringing

U+T+A
- cons: FFT upsample of DNNROI **may** (or not) cause ringing, any ROI size
- pros: Gauss is not downsampled, ROI sizes multiples of 4.

T+U+A
- cons: ROI sizes multiples of 4.
- pros: Gauss is not downsampled, no possible ringing.

Details / tracking at:
`https://github.com/WireCell/spng/issues/32`
Start with **T+U+A** as this is what OSP does.

Misunderstanding ~~possible bug~~ in past DNNROI inference.

I misunderstood the DNNROI training so redact this slides.
The (non) issue is now:

- DNNROI **training and inference** scale all three features images by 1/4000.
- I misunderstood that training only scale “loose LF” while inference scaled both.

It is still perhaps odd (to me) to also scale MP2/MP3 by 4000, but the BatchNorm2D will find a suitable self normalization across the feature dimension.
For reference, I keep the BatchNorm2D slide in backups.

Next up

- ☐ Validate post-CrossView outputs.
- ☐ Configuration:
  - Output `IFrame` from SPNG graph
  - Make sim and SPNG test files to be “modular”.
  - A “depo flux splat” subgraph (“true signals”)
  - An OSP subgraph
  - Individual and combined main config files.
- ☐ More rigorous profiling, performance eval.
- ☐ Revisit run-time story (eg containers, CVMFS).
- ☐ Incorporate depo files from Jake and “SPDIR” depos.
- ☐ Continue DNNROI++ ideas (MPn $\rightarrow$ CrossViews and Jake’s Multi-UNet)
- ☐ … What else?

$\mathcal{FIN}$

BatchNorm2D

DNNROI starts with Conv2D and then has a BatchNorm2D:

$$y = \gamma \frac{x - E[x]}{\sqrt{Var[x] + \epsilon}} + \beta$$

$E[x], Var[x]$ mean/variance pixel value across feature dimension.

γ, β per-feature learned parameters.

By default, these parameters are all frozen after training.