Looking into DNNROI Training

Jake Calcutt
Local ProtoDUNE Meeting
July 15, 2026

Introduction

One bottleneck in SPNG is forming MP2/3

- Much slower on CPU, not super fast on GPU

I’ve been trying to understand: **can we work without MP2/3?**

Introduction - Cont.

I’ve been investigating some peculiar results from running wire-cell-python dnn training

- Training with filtered/decon’d input only seems to give better results than what is quoted in the DNNROI paper

Looking at metrics as a function of epoch and also seeing effect of training set size

Was able to make some improvements to wcpy dnn along the way

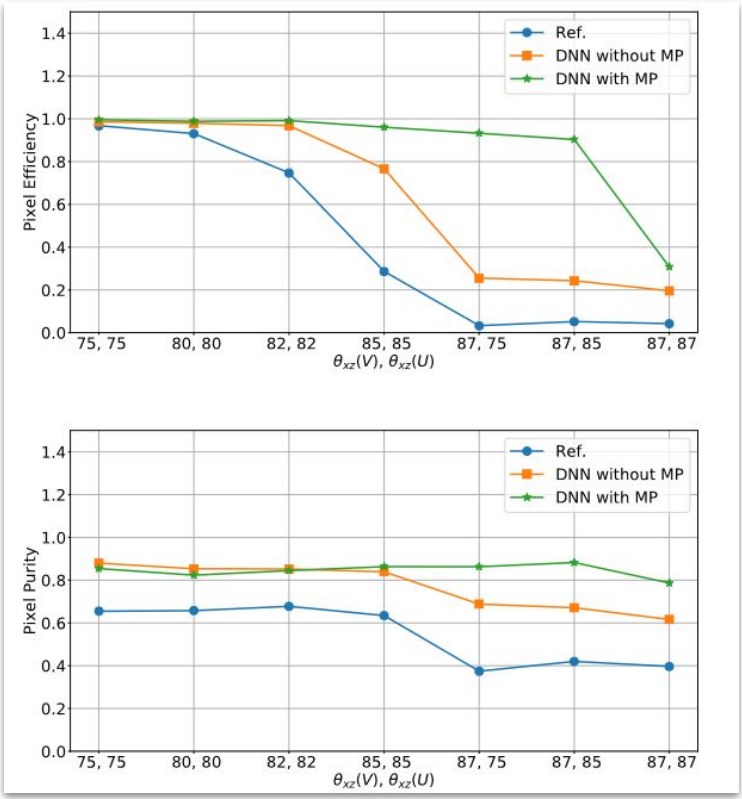


Fig. 8 DNNROI Paper

My training setup

Using dunesgpu0004 SDCC machine (8x nvidia L40S gpus)

- Submitting from dunesub01

Made ~20k input events

- PDHD APA1 with ideal response (no HV issue) cosmics + radiologicals
- Signal Processing done with SPNG implementation

Submitting training with condor using scripts in this repo

wcpy branch has some updates including

- model.eval() when running eval (doesn't affect training loss, just eval loss)
- Automatic Mixed Precision (faster to train + larger batch size)
- cuDNN.benchmark
- Fixed batch scaling on eval loss

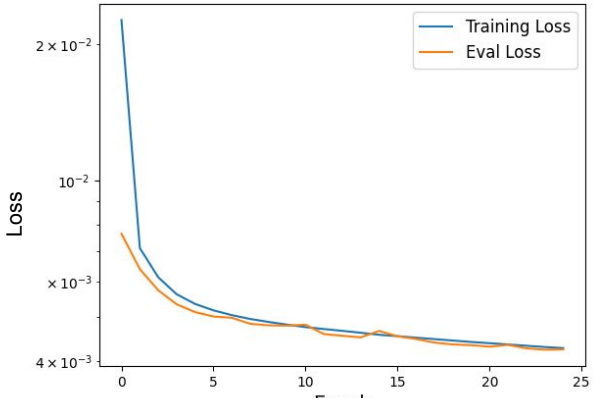
Full Set Results

16 hrs to train / 25 epochs / batch-size 8 / SGD lr=0.005

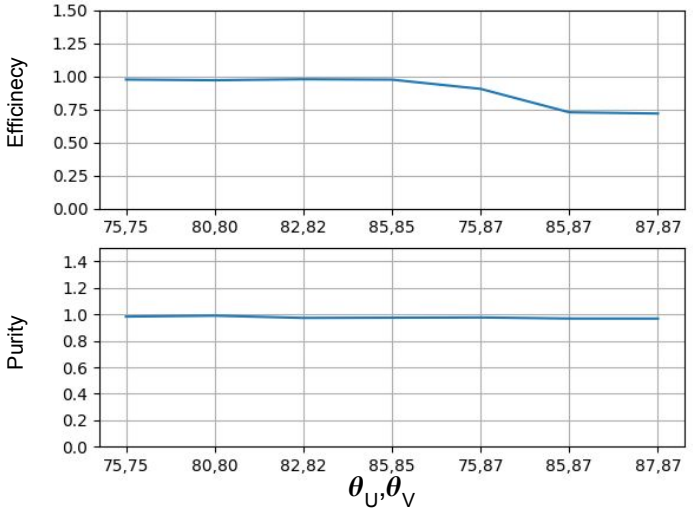
- Loss not diverging, could probably keep training
- Note: Ref lr=0.1

Purity mostly flat, ~97% at most extreme angle

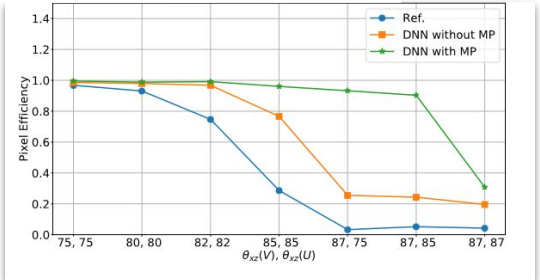
Efficiency follows a similar trend but much higher at high end (almost same at 2nd most-extreme angle)



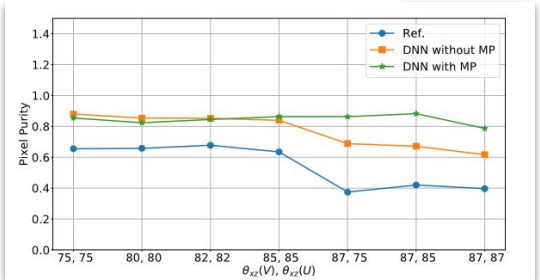
My results



DNNROI Paper Efficiency



DNNROI Paper Purity



Understanding differences

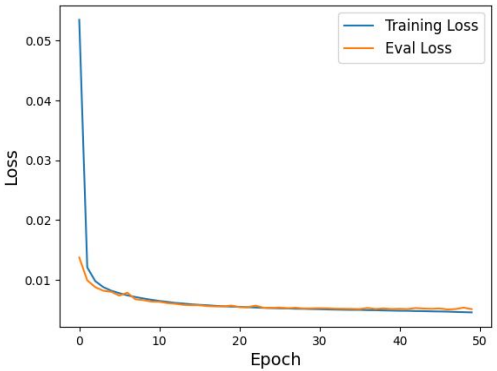
Want to understand why my metrics appear so much better

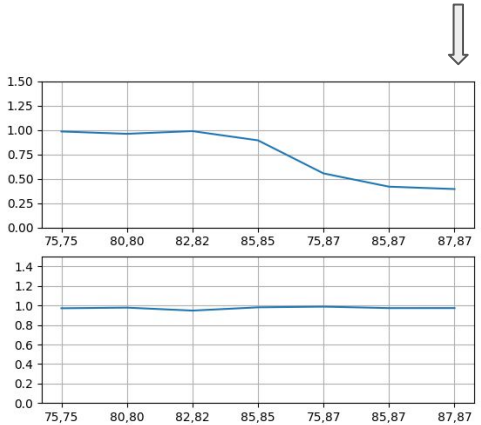
Tried to replicate the DNNROI plots. Training with 500 events only

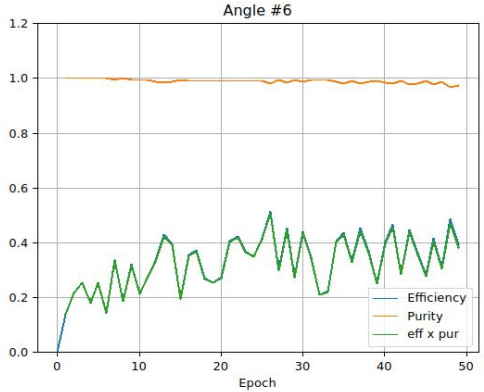
- Same events & seed (same shuffling / epoch)
- Various learning rates (.005, .01, .1), and 2 batch-sizes (8 & 1)
- DNNROI w/ & w/out MP
    - Again: SPNG implementation (simple threshold for initial ROIs)

Also looked at things epoch-to-epoch to see “stability” of training

Dense, LR=.005, Batch=1

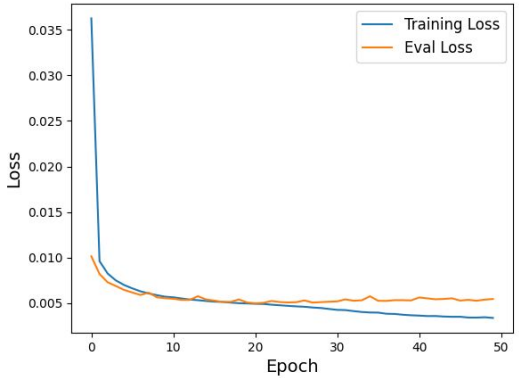


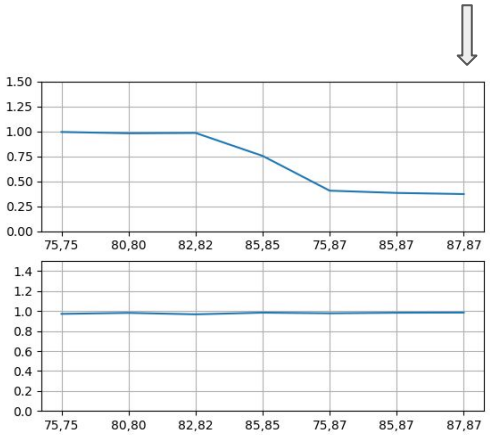


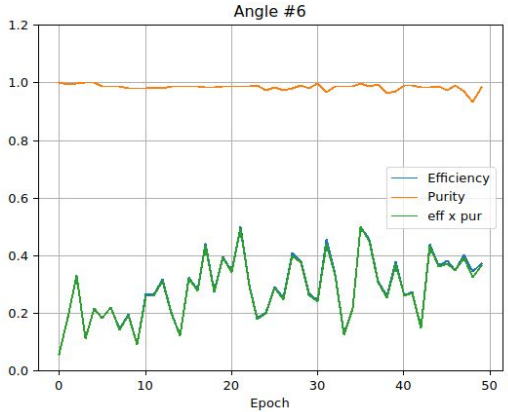


Efficiency highly variable epoch-to-epoch

Dense, LR=.01, Batch=1



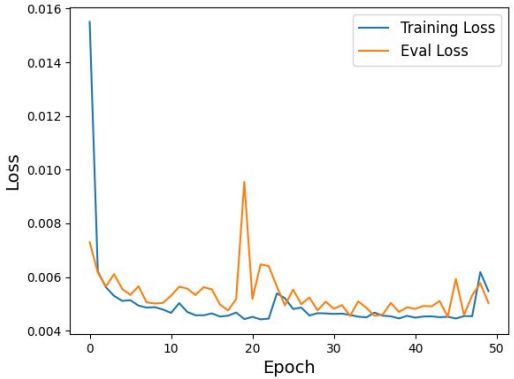




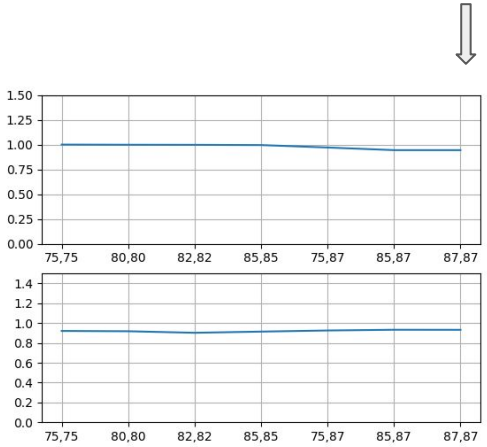
Eval Loss diverges ~25

Efficiency highly variable epoch-to-epoch

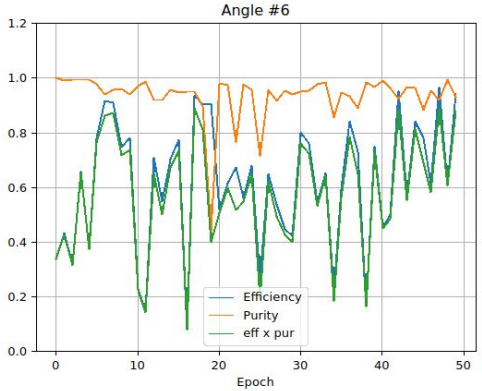
Dense, LR=.1, Batch=1



Loss curves look really bad

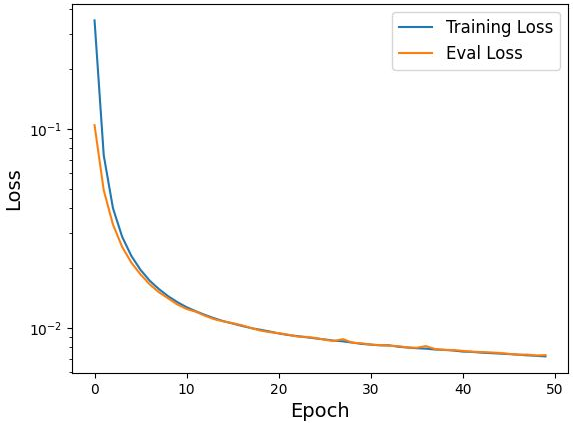


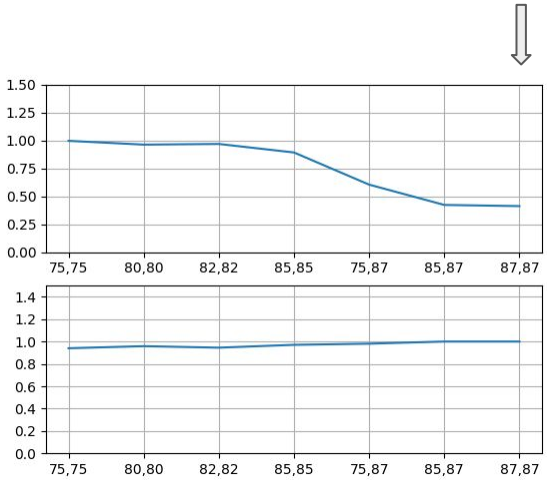
Metrics look ~perfect!

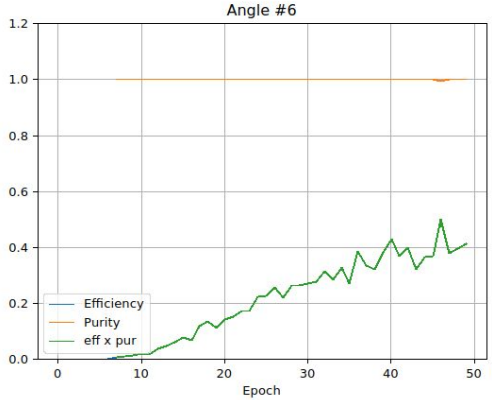


Efficiency highly variable epoch-to-epoch

Dense, LR=.005, Batch=8



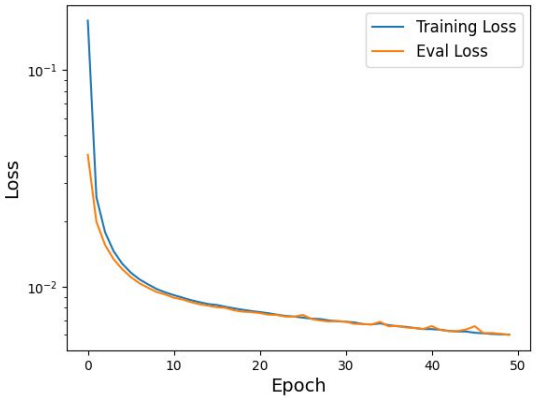


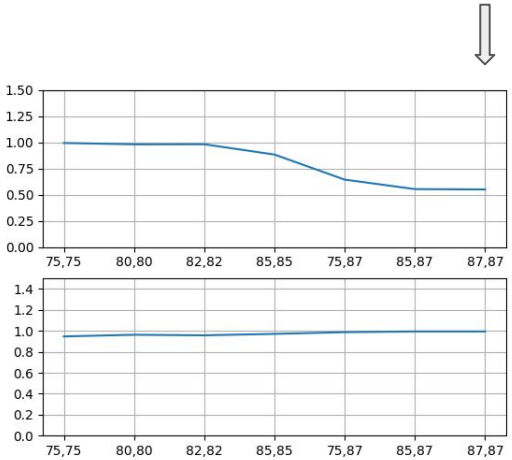


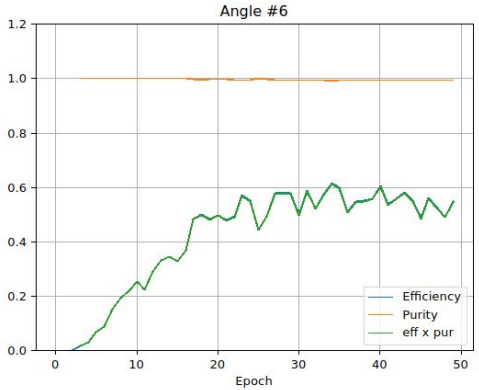
Loss not diverging → Could keep training?

Some effic. variability, continuing to increase?

Dense, LR=.01, Batch=8



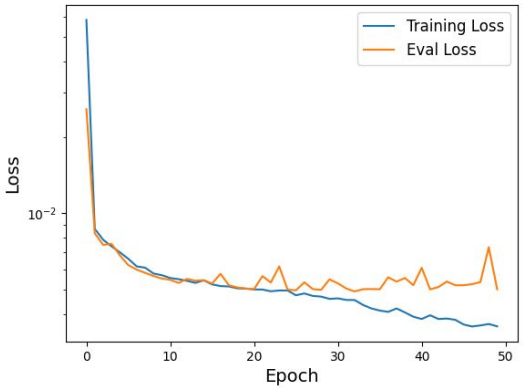


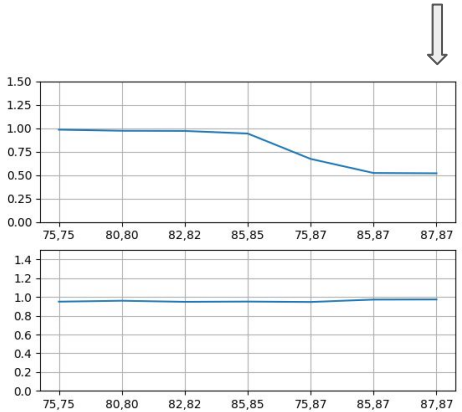


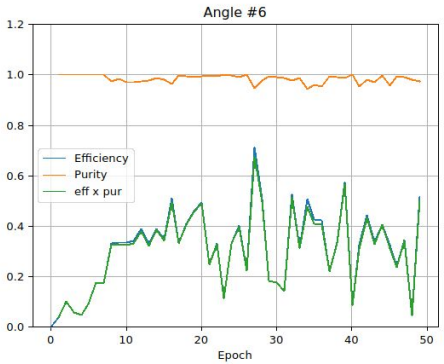
Loss not diverging → Could keep training?

Some effic. variability, but flattening out

Dense, LR=.1, Batch=8



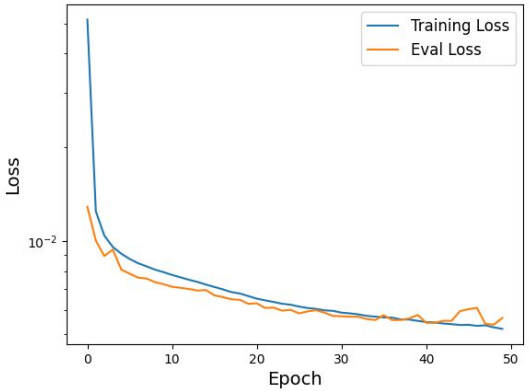


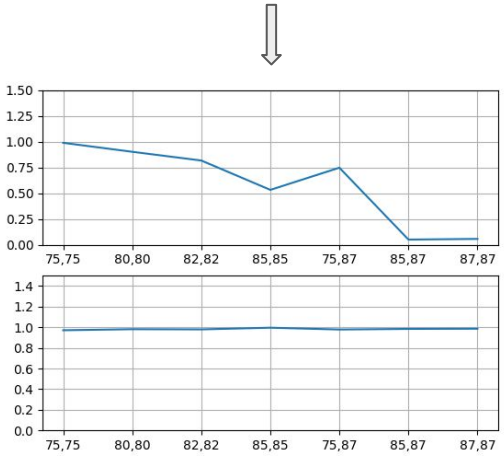


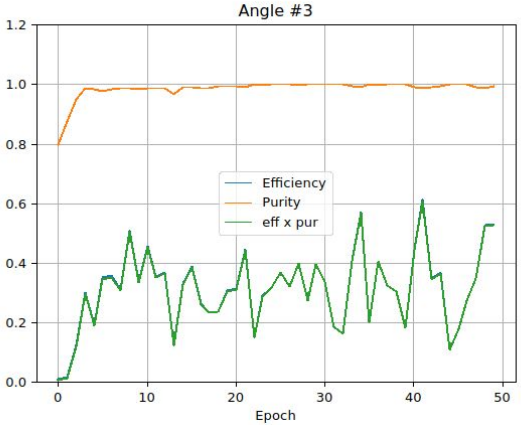
Eval Loss diverges ~15

Efficiency highly variable epoch-to-epoch

MP, LR=.005, Batch=1

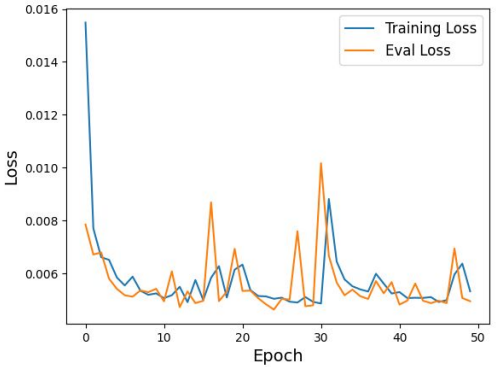


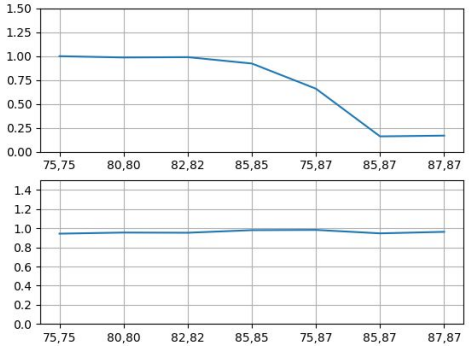


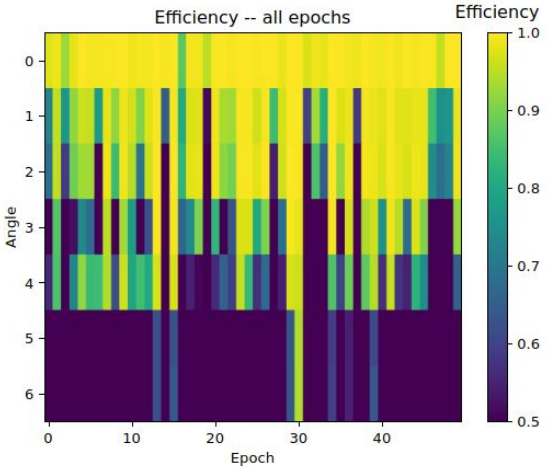


Efficiency highly variable epoch-to-epoch (angle 85,85)

MP, LR=.01, Batch=1



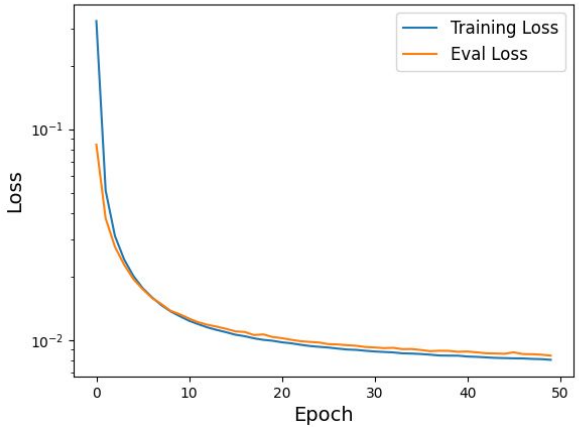


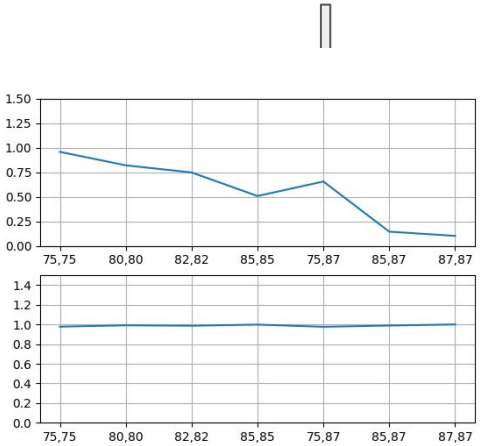


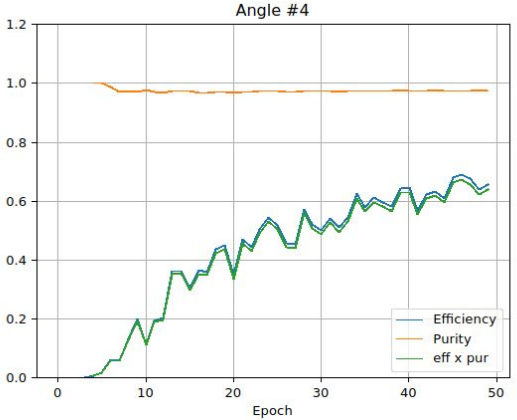
Loss curves terrible

Efficiency highly variable epoch-to-epoch (most angles)

MP, LR=.005, Batch=8



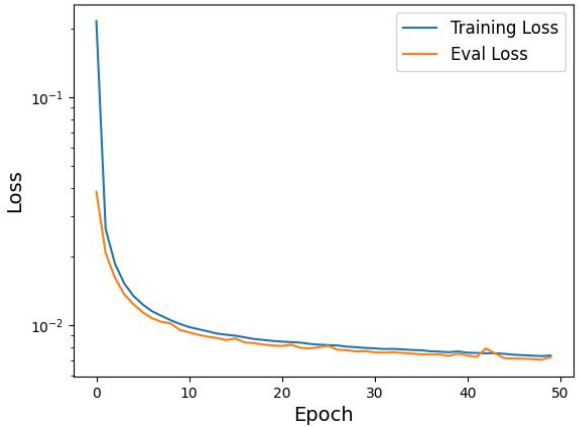


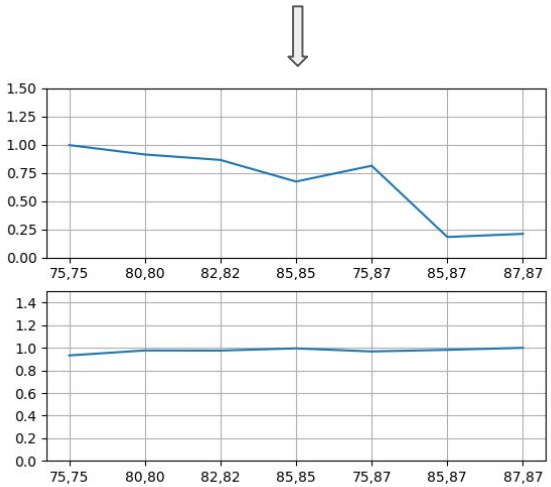


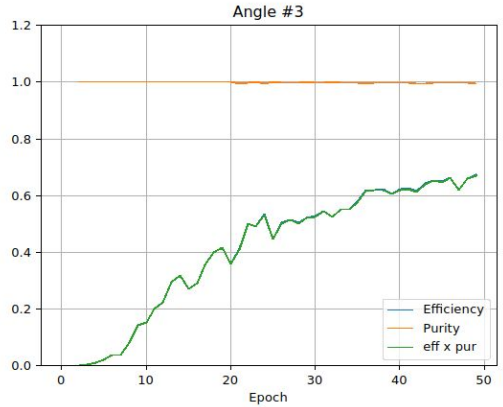
Loss not diverging – could keep training?

Efficiency growing for this angle (75,87)

MP, LR=.01, Batch=8



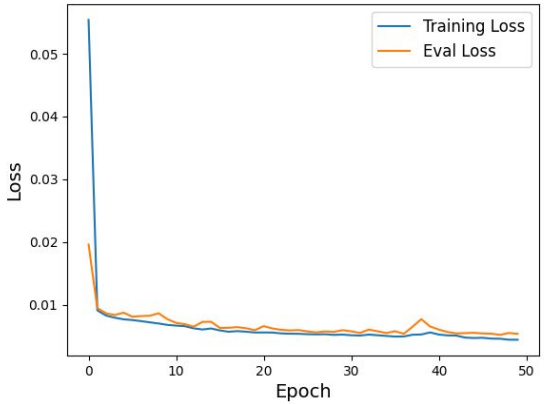


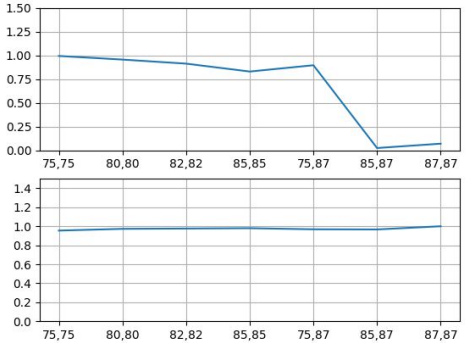


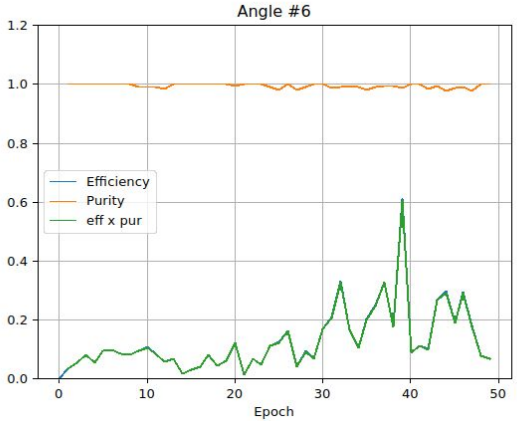
Loss not diverging – could keep training?

Efficiency growing for this angle (85,85)

MP, LR=.1, Batch=8

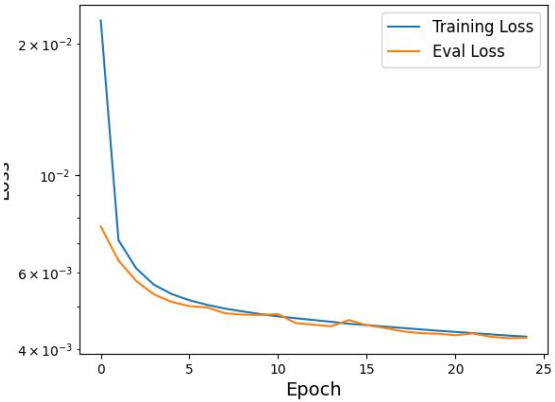


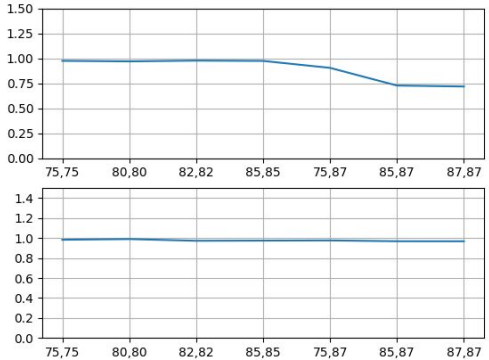


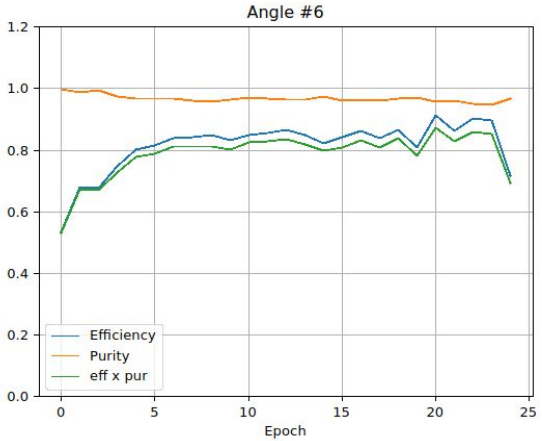


Efficiency highly variable

20k Events, Dense, LR=0.005, Batch=8







Efficiency starting to degrade towards higher epochs

Conclusion

Training dense-only input to DNNROI with (much) larger training sample seems to achieve better results than what's published in the DNNROI paper

- Lower training sample size → Artificially bad performance?

Leaves me with some open questions about how recent training is currently being done

Next steps:

- Make even more training data to see how high we can push dense-only training
- Merge wcpy dnn improvements (AMP, cuDNN backending, fixed batching)
- Implement Distributed Data Parallel in wcpy dnn to further speed up training with larger training size