MicroBooNE DNN inputs

- Doesn’t seem like there are higher level reconstructed inputs?

The key part of our regressional segmentation network is a Sparse U-Net [46] which extracts a feature vector for each pixel or voxel. Figure 11 shows details of a level 2 Sparse U-Net. One can add more levels by iteratively inserting more “Concat-Join” blocks. We use level 5 in this paper. More implementation details can be found in [47]. The SparseConvNet works on sparsified 2D or 3D images, which consist of a list of pixels (2D) or voxels (3D). The Wire-Cell reconstructed 3D points are placed into voxels, which is a cube with 0.5 cm in each of three dimensions. If multiple 3D points fall into the same voxel, the values are averaged. The SparseConvNet takes two tensors as input. One is a coordinate tensor (integer type) with position information for each voxel. The other one is a feature tensor (float type) with the charge information. The output of the SparseConvNet is also a list of features for each voxel, for now we extract only one feature which we call *confidence value* related to the distance between a point and the truth vertex. The truth label for this *confidence value* is calculated before voxelization, and the equation used is:

$$Conf_{\text{truth}} = \exp\left(-\frac{\|\vec{x} - \vec{v}_{\text{truth}}\|^2}{2\sigma^2}\right), \tag{2.2}$$

where $Conf_{\text{truth}}$ is the truth label of the *confidence value*; $\vec{x}$ and $\vec{v}_{\text{truth}}$ are the reconstructed 3D charge and truth vertex positions; σ is a regularization parameter (1 cm is used). *Confidence value*s from multiple voxels (pixels) form a “Confidence Map” [48]. Figure 12 shows an example of the input (charge) and label before voxelization.

Prepared for submission to JINST

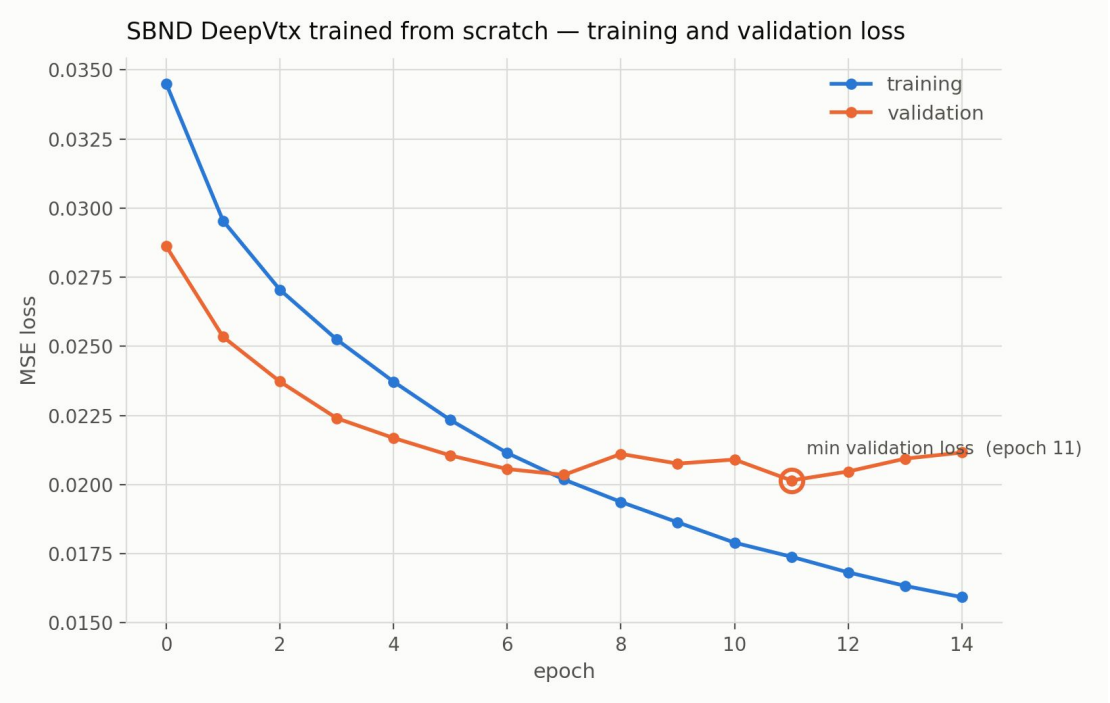
Wire-Cell 3D Pattern Recognition Techniques for Neutrino Event Reconstruction in Large LArTPCs: Algorithm Description and Quantitative Evaluation with MicroBooNE Simulation

https://arxiv.org/pdf/2110.13961

Training settings and parameters

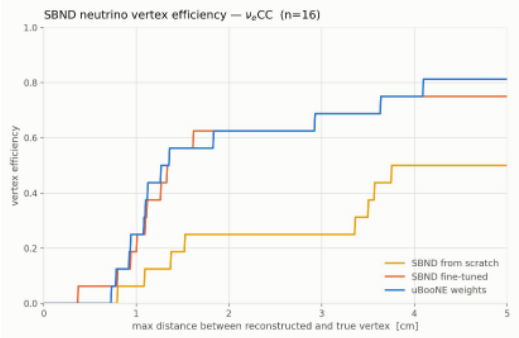
- Initial MicroBooNE weights t48k-m16-l5-lr5d-res0.5-CP24.pth, found in wire-cell-data (trained on 48k events)
- SBND sample: 15k BND nu + cosmic sample, reconstructed with wirecell and converted to npz. 12,122/3,030 training/validation split
- Optimizer: Adam with initial learning rate of 10^{-5}
- Scheduler: exponential decay every epoch, so learning rate is $10^{-5} * e^{-0.05 * \text{epoch}}$
- Batch size of 1
- Loss is MSE of confidence value against Gaussian target with σ = 1cm
- Input is summed charge in every 0.5cm voxel

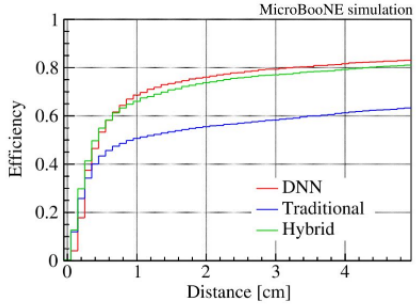
Training on just SBND data



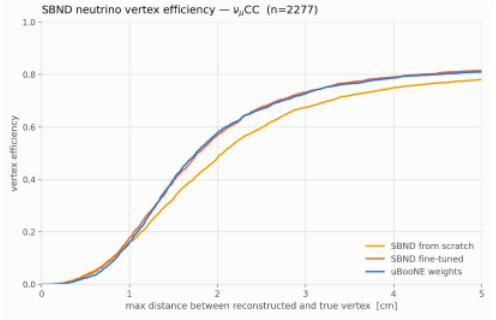
Training on just SBND data

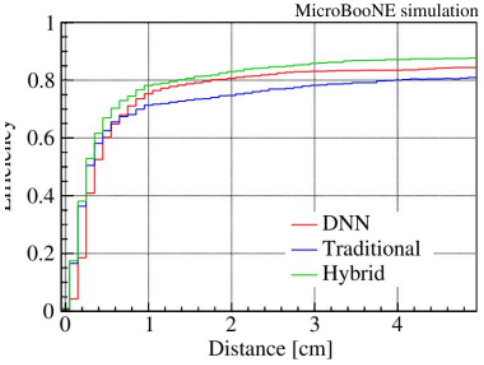
nu_eCC





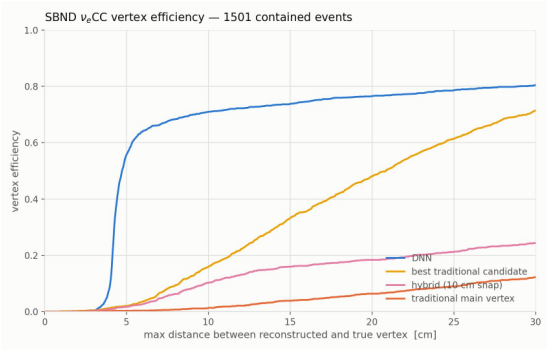
nu_muCC





Preliminary traditional vs hybrid vs DNN comparison

- Traditional vertexing algorithm found in wire-cell-toolkit in /clus/src/NeutrinoVertexFinder.cxx
- Run on a slice of mc_MCP2025B_prodgenie_corsika_proton_rockbox_ccnue_sbnd_reco1_sbnd
- Next: try to replicate microboone results with open dataset



Traditional input	**Source**
Wire signals, optical flashes, dead/bad channel masks	Art ROOT file
3D/2D Charge, positions	Wirecell reconstruction
Detector geometry	sbnd_v02_05.gdml
SCE map	sbnd_data/SCEoffsets.root
Optical model	sbnd/photodet/semi-analytical-sbnd.json
particle dE/dx tables + other info	clus/src/ParticleDataSet.cxx

Boundary fitting update

- Working on re-running fitting endpoint distribution to Gaussian instead of space-point distribution to turn-on curve
- Need much larger MC and data sample for equivalent granularity, so running wirecell reconstruction on those now