DEEP UNDERGROUND NEUTRINO EXPERIMENT

BROOKHAVEN NATIONAL LABORATORY

Semantic segmentation (pixel-wise classification) in ProtoDUNE.

Arbin Timilsina

ProtoDUNE sim/reco meeting
February 27, 2018

Introduction

ProtoDUNE event contains beam particles and many cosmics

⇒ Classification of an event itself is not useful; however, to be able to answer **"What's in this event and where in the event display are different particles located?"** is very valuable

Overview

⇒ This presentation: Semantic segmentation (pixel-wise classification) to perform cosmic ray and beam particle separation

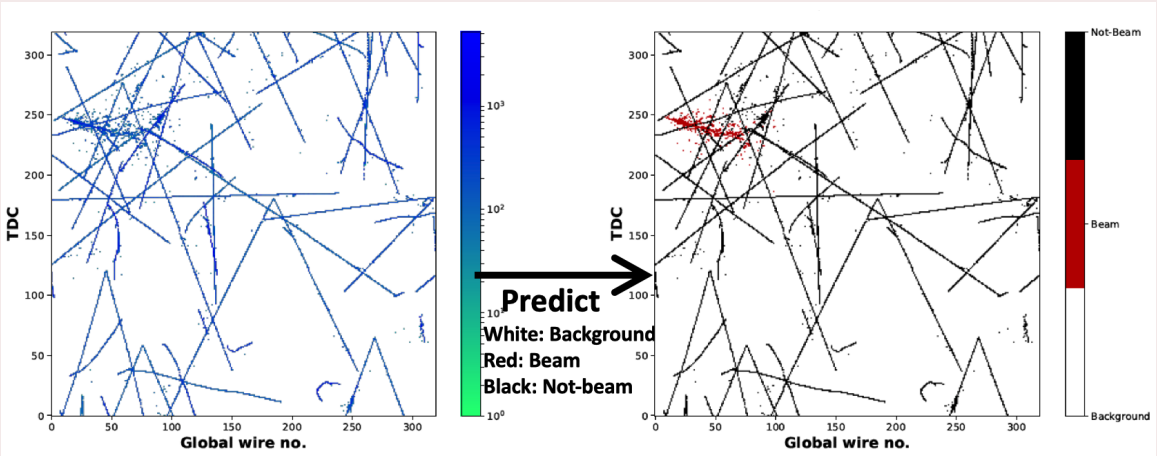
- Three classes: Background (non-hits), beam, and not-beam (includes muon halo)

⇒ Next step: Expand beam class to various particle species

⇒ Ultimate goal: Quantify the performance of the network with data

Semantic segmentation

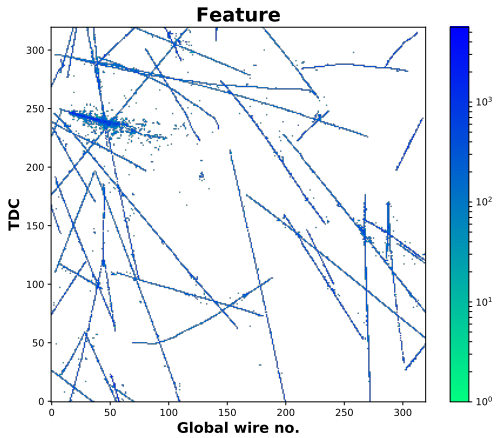
Semantic segmentation associates each pixel of an event display with a class label (such as background, cosmic, or beam) → recognize, understand what's in the image at the pixel level.

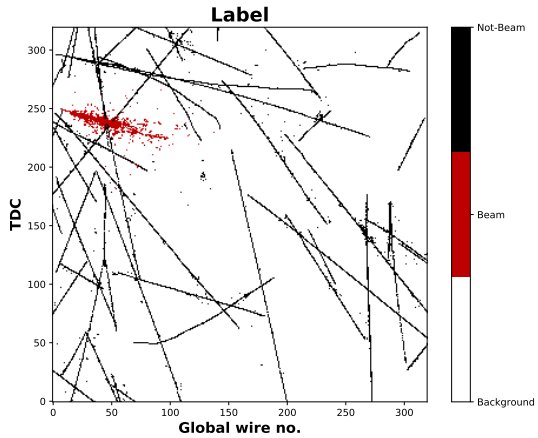


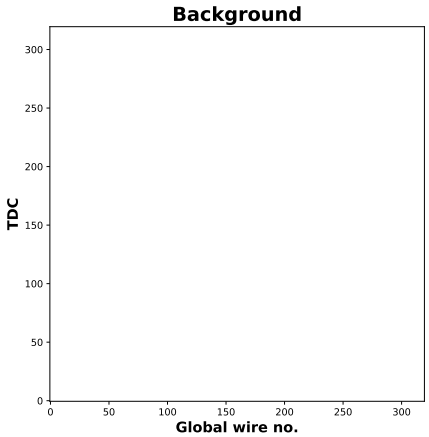
ProtoDUNE event

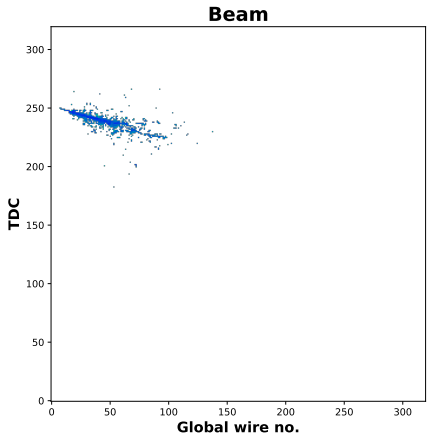
⇒ Running over MCC-11: ProtoDUNE single phase, beam momentum = 7 GeV, space-charged enabled samples

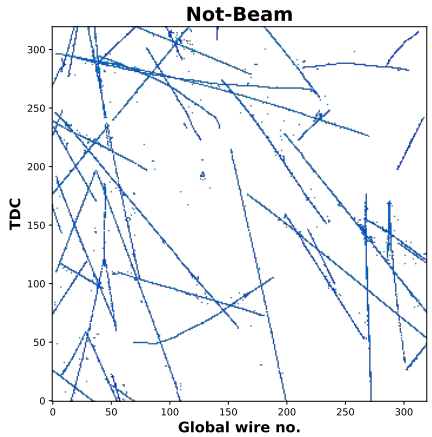
⇒ Images are created as TDC vs. global wire number with ADCs for the pixel value





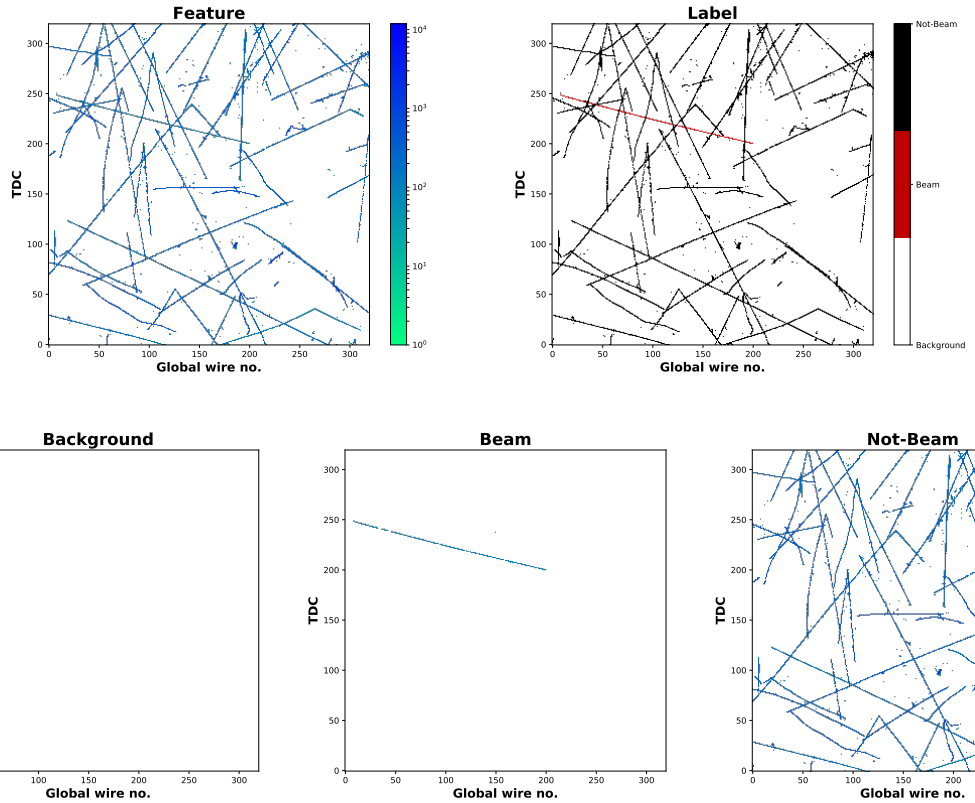






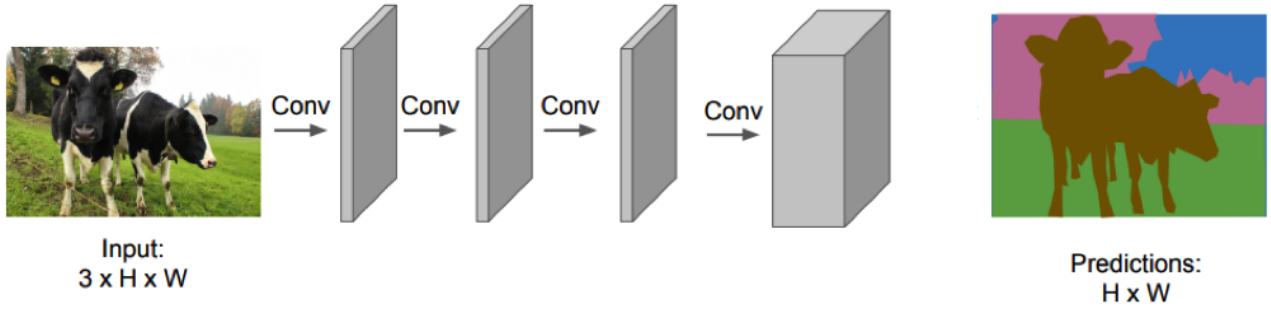
ProtoDUNE event

- ⇒ Input images are generated using just the collection plane
- ⇒ Images are downsampled from 6000 × 1440 to 320 × 320; currently choosing maximum ADC value during downsampling (as well as to label between not-beam and beam); needs optimization
- ⇒ Most of the events don't have many hits from the beam; have required minimum hits of 2000



Encoder/decoder structure

A naive approach towards constructing a semantic segmentation architecture is to simply stack a number of convolutional layers (while preserving dimensions) and output a final segmentation map $\rightarrow$ design a network with convolutional layers to make predictions for pixels all at once. **However, it is very computationally expensive to preserve the full resolution throughout the network.**



One popular approach towards constructing a semantic segmentation architecture is to follow an **encoder/decoder** structure where the spatial resolution of the input is downsampled (developing lower-resolution feature mappings) and the feature representations are upsampled into a full-resolution segmentation map.

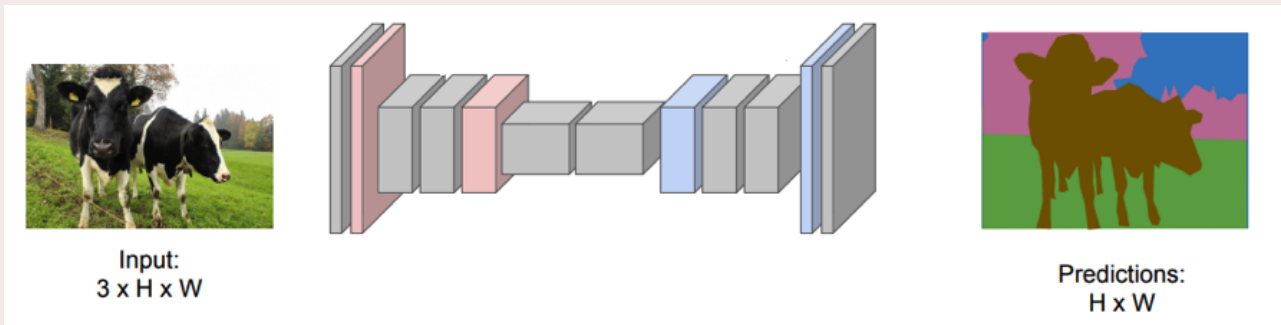


Image credit

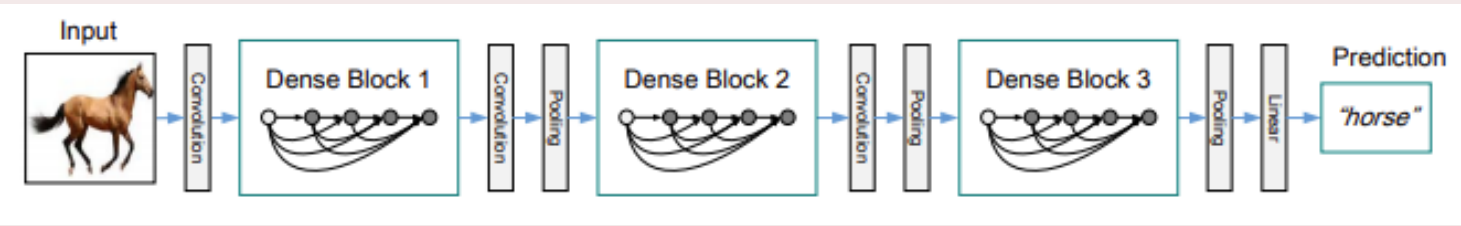
Model architecture

I experimented with many architectures for the task of semantic segmentation. **Tiramisu: Fully Convolutional DenseNets** seems to work the best.

DenseNet

Dense Convolutional Network which connects each layer to every other layer:

- ⇒ Each layer has direct access to the gradients from the loss function and the original input signal → implicit deep supervision
- ⇒ Dense connections have a regularizing effect → reduces over-fitting on tasks with a smaller training set

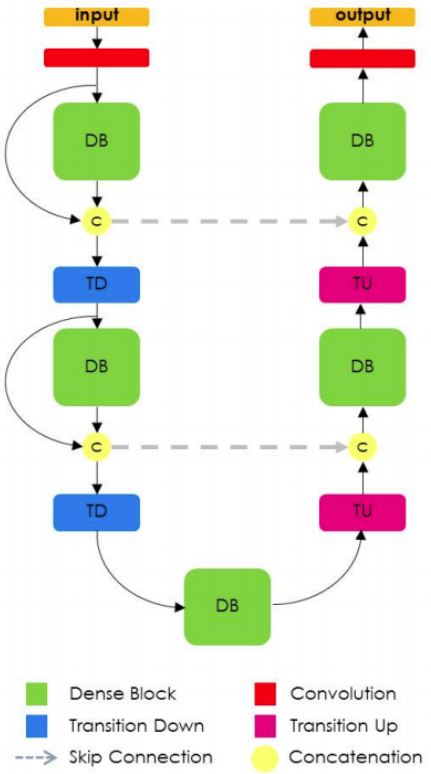


Fully Convolutional Networks

Takes input of arbitrary size and produces correspondingly-sized output with efficient inference and learning:

- ⇒ Skip architecture → combine 'what' information and 'where' information
- ⇒ Deconvolution filter for upsampling is not fixed but learned
- ⇒ Class balancing unnecessary for most cases → no need to balance classes by weighting or sampling the loss

Tiramisu: Fully Convolutional DenseNets



- ⇒ Tiramisu extends DenseNets to work as FCNs by adding an upsampling path to recover the full input resolution
- ⇒ Only upsamples the feature maps created by the preceding dense block → fewer parameters- mitigates feature map explosion
- ⇒ No pre-training or post-processing required to achieve state-of-the-art results

Loss function and optimizer

The loss function takes the predictions of the network and computes a loss score. The optimizer uses this score to adjust the value of the weights a little (in a direction that will lower the loss score).
We have an unbalanced data: for 1 beam label, we have $\approx$ 14 cosmic and $\approx$142 background label.

Loss function

- $\Rightarrow$ Due to the class balancing power of FCN, categorical cross-entropy (log loss) works fine
- $\Rightarrow$ However, once we start to expand beam class to various particle species, data will be even more unbalanced and alternative loss functions may be necessary. See backup for some options that I have experimented with.

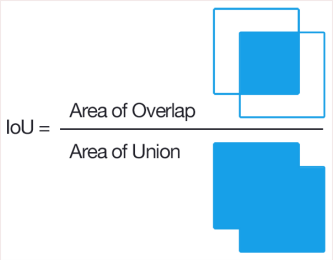
Optimizer: Adam

- $\Rightarrow$ Mini-batch gradient descent applies the same learning rate to all parameter updates
- $\Rightarrow$ For sparse dataset, we want to perform a large update for rarely occurring features
- $\Rightarrow$ Adam computes adaptive learning rates for each parameter, performing larger updates for infrequent and smaller updates for frequent parameters $\rightarrow$ well-suited for dealing with sparse data

Performance metric

Mean Intersection Over Union (mIOU)

IOU measures how good our prediction is when compared with the ground truth; score ranges from 0 (bad) to 1 (excellent).

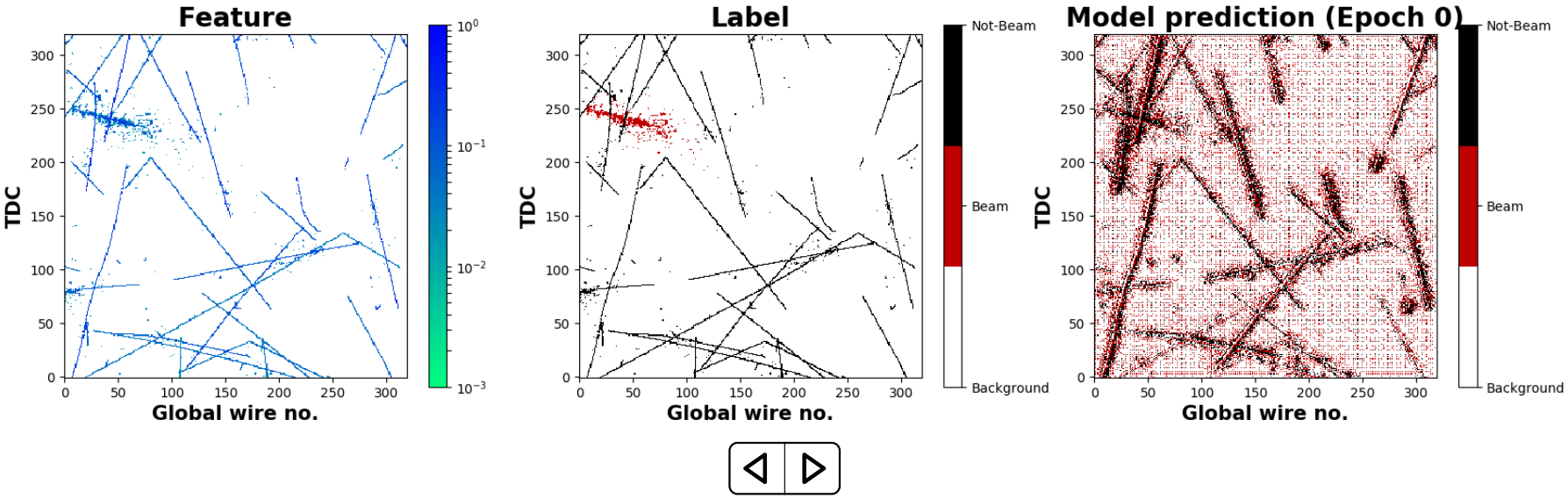


Averaged over three classes $\rightarrow$ mIOU.

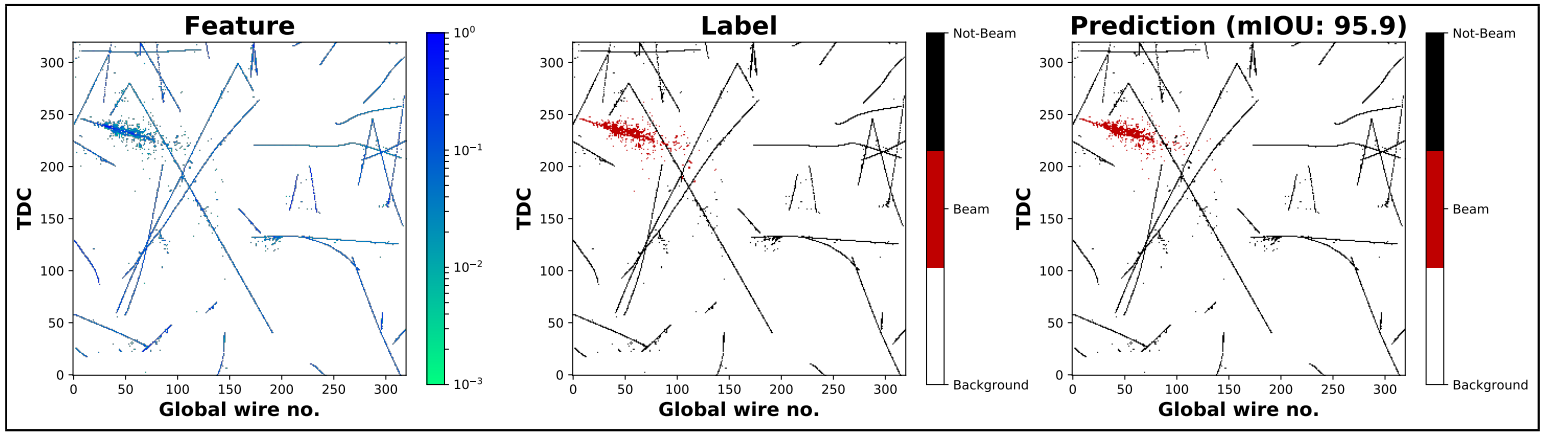
Training details

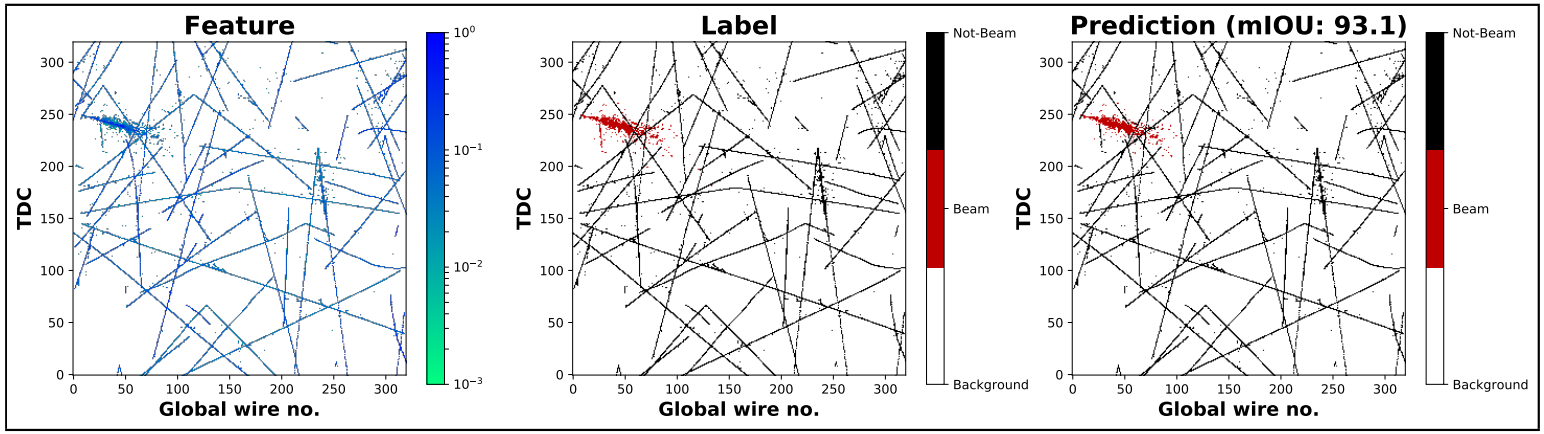
- $\Rightarrow$ 2600 training, 250 validation, and 60 test images
  - Tendency of DenseNets to be less prone to overfitting
- $\Rightarrow$ Trained with batch size of 2; for 390 epoch total (130 epoch at a time)
- $\Rightarrow$ **mIOU achieved on test set: 93.8**
  - IOU for background: 99.98
  - IOU for not-beam: 98.2
  - IOU for beam: 83.1

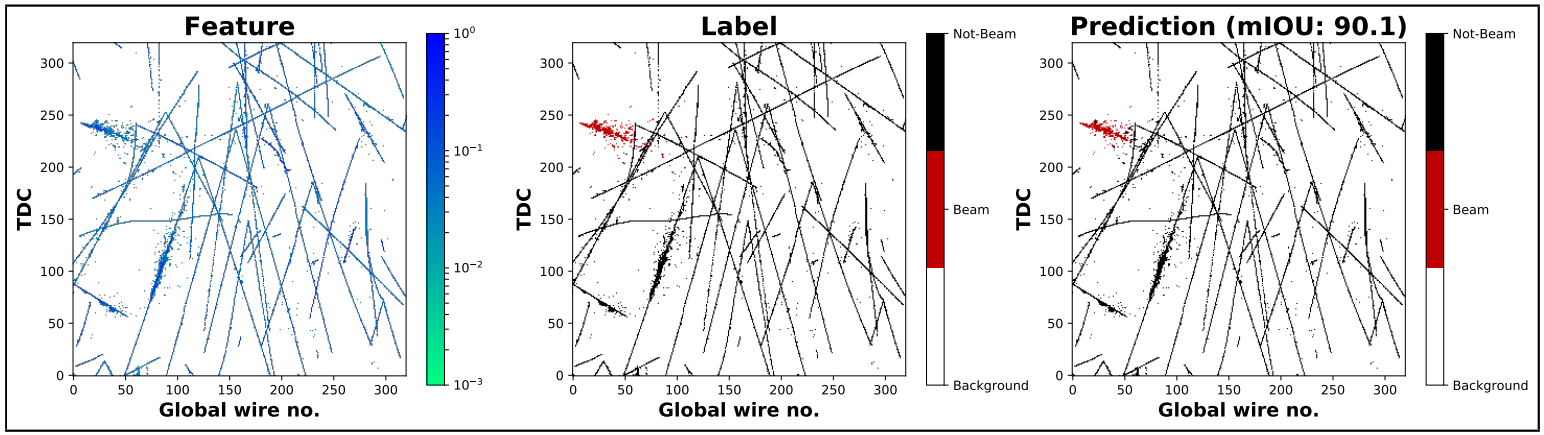
Model at work

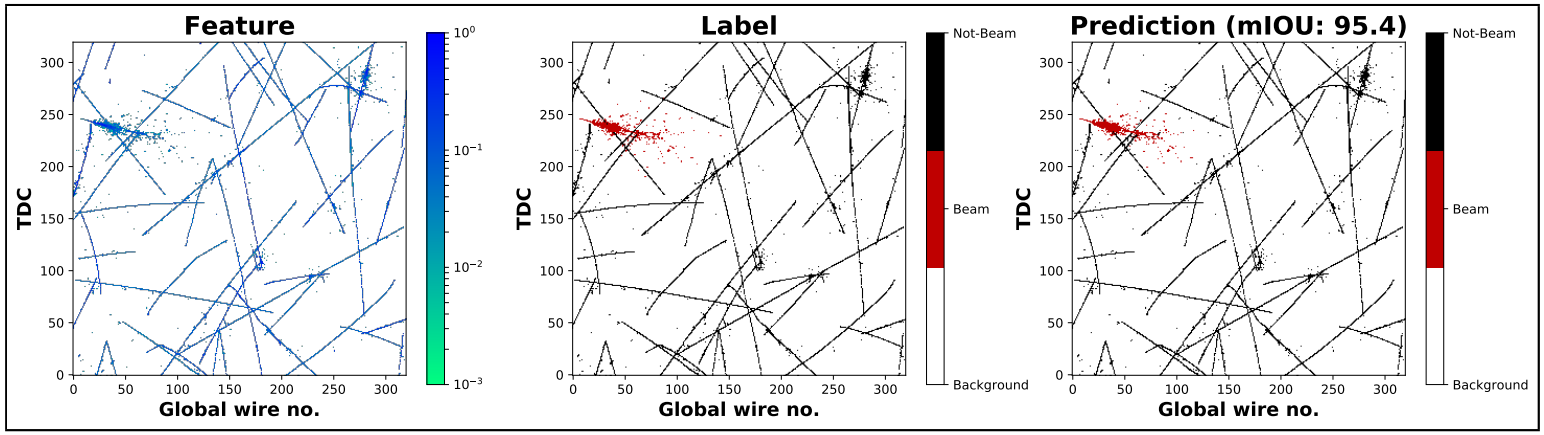


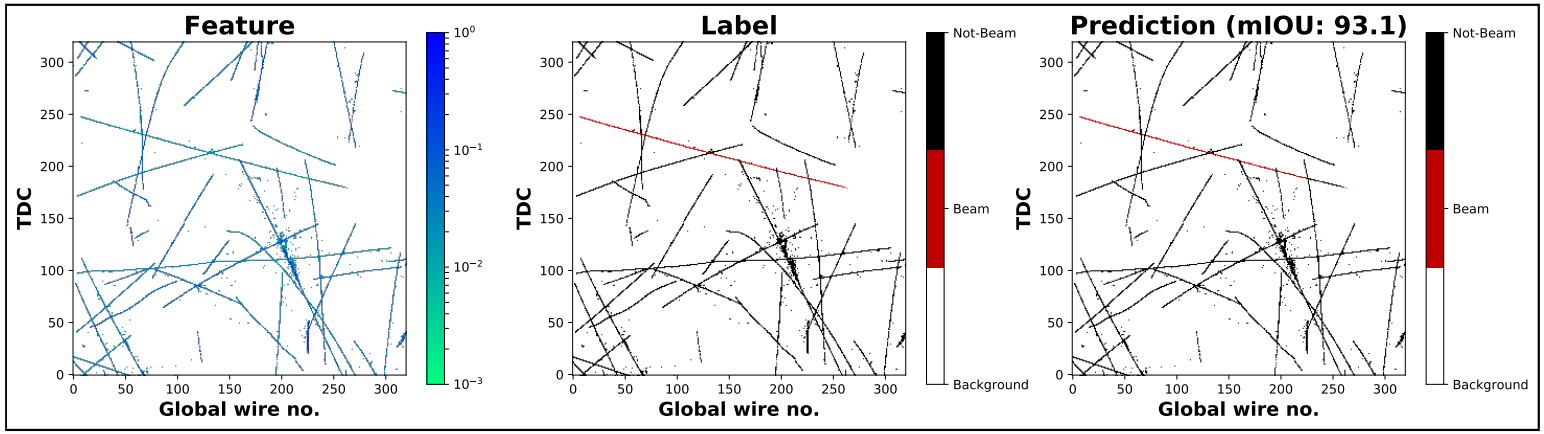
Prediction on test set

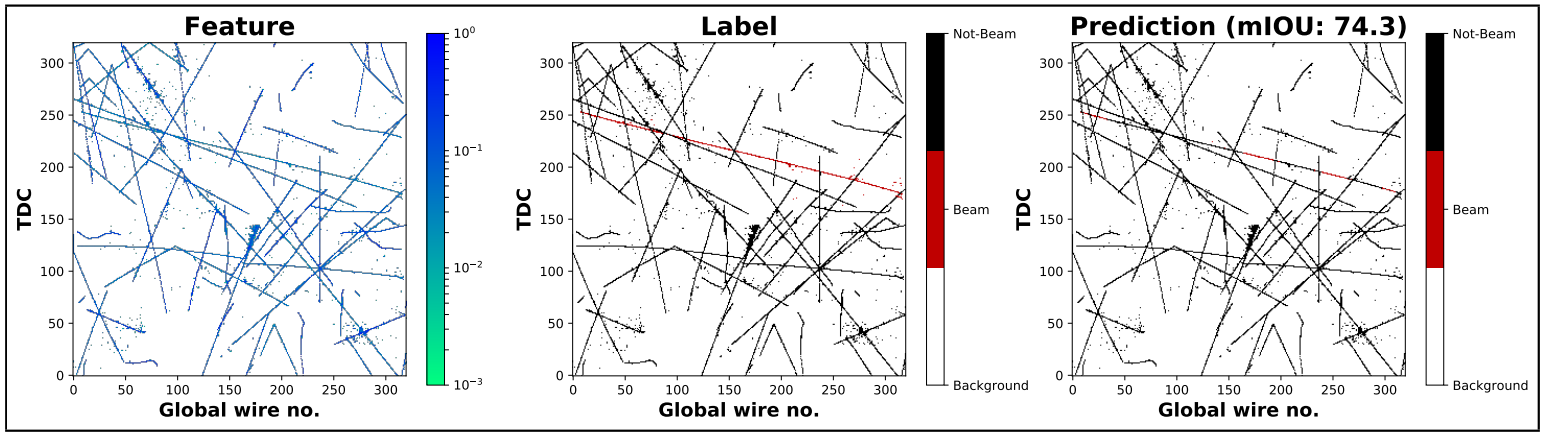

Feature
Label
Prediction (mIOU: 95.9)
TDC
Global wire no.
Not-Beam
Beam
Background


Feature
Label
Prediction (mIOU: 93.1)
TDC
Global wire no.
Not-Beam
Beam
Background

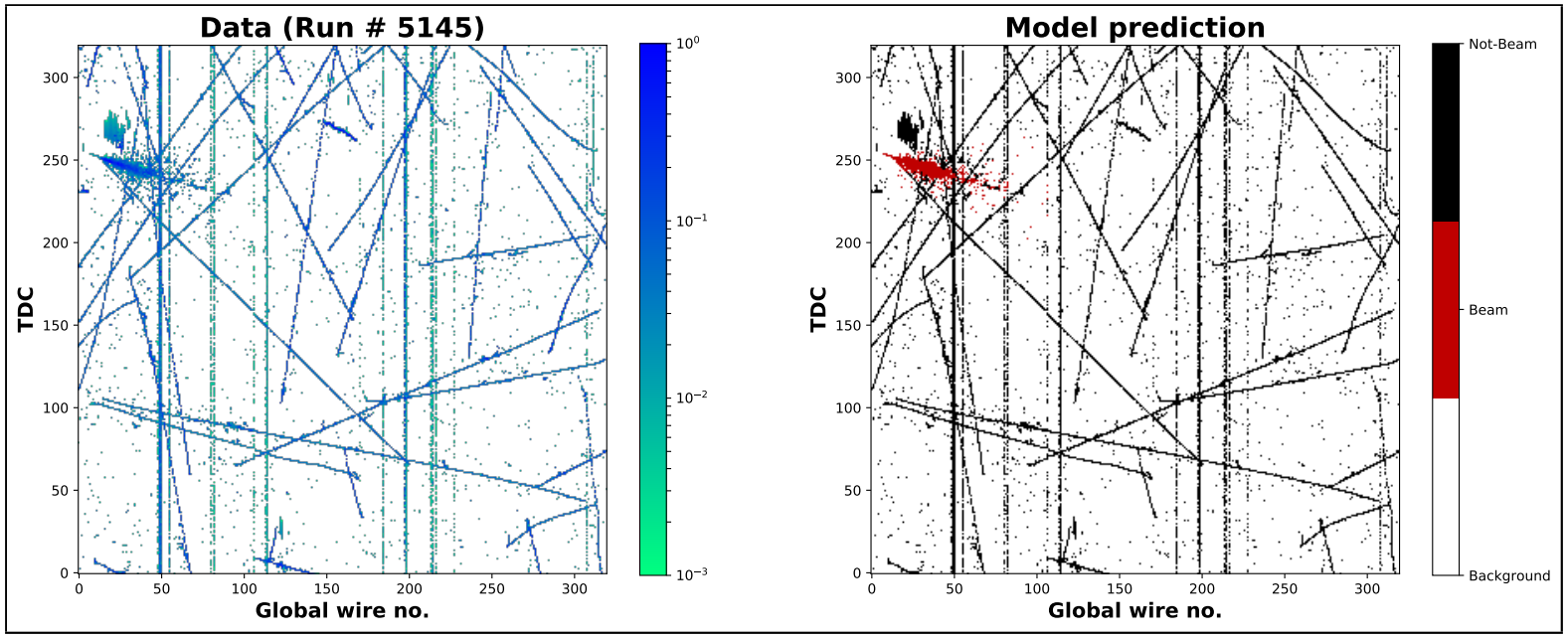

Feature
Label
Prediction (mIOU: 90.1)
TDC
Global wire no.
Not-Beam
Beam
Background

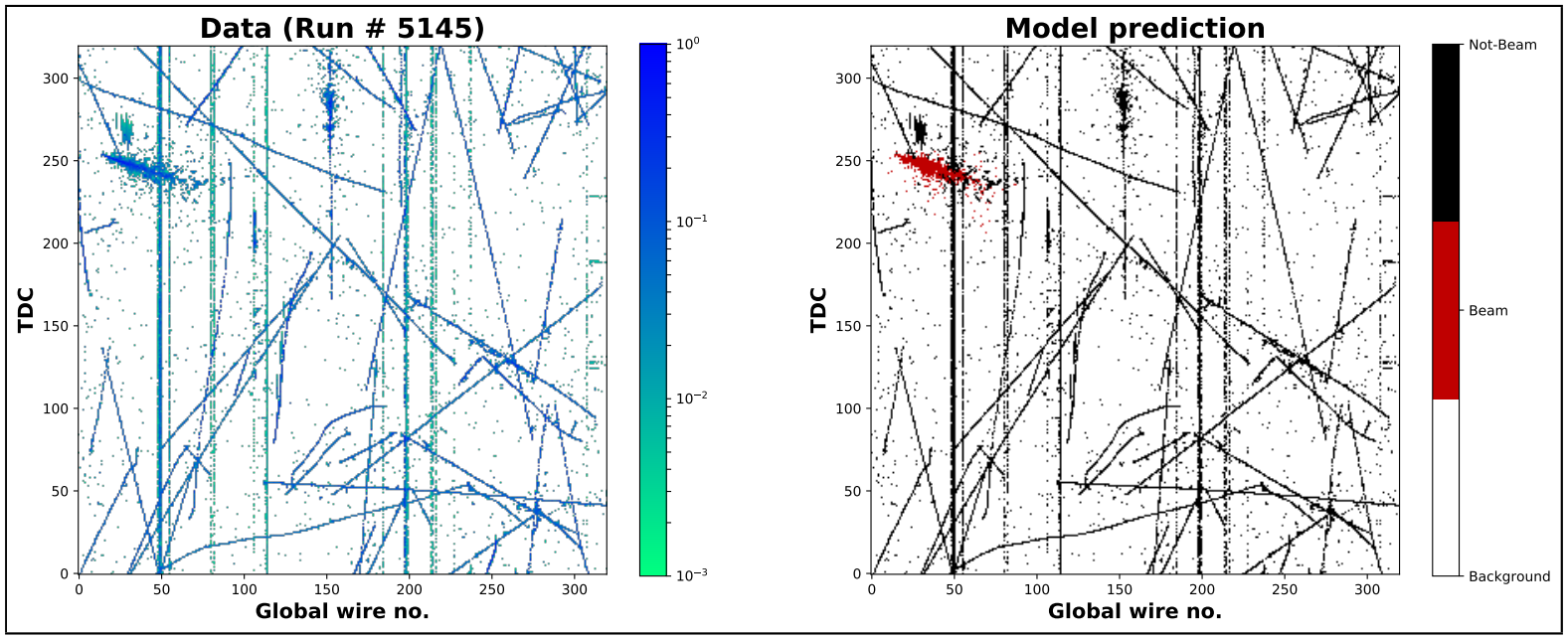

Feature
Label
Prediction (mIOU: 95.4)
TDC
Global wire no.
Not-Beam
Beam
Background

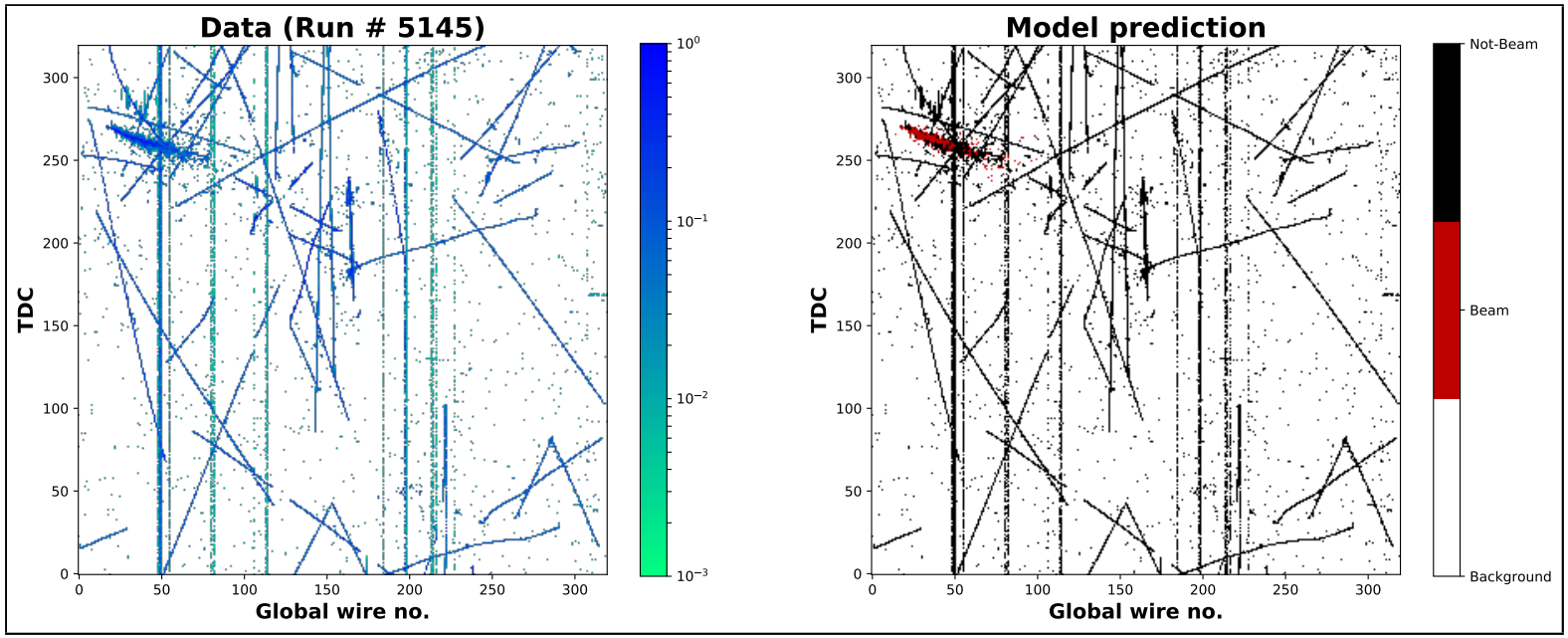

Feature
Label
Prediction (mIOU: 93.1)
TDC
Global wire no.
Not-Beam
Beam
Background

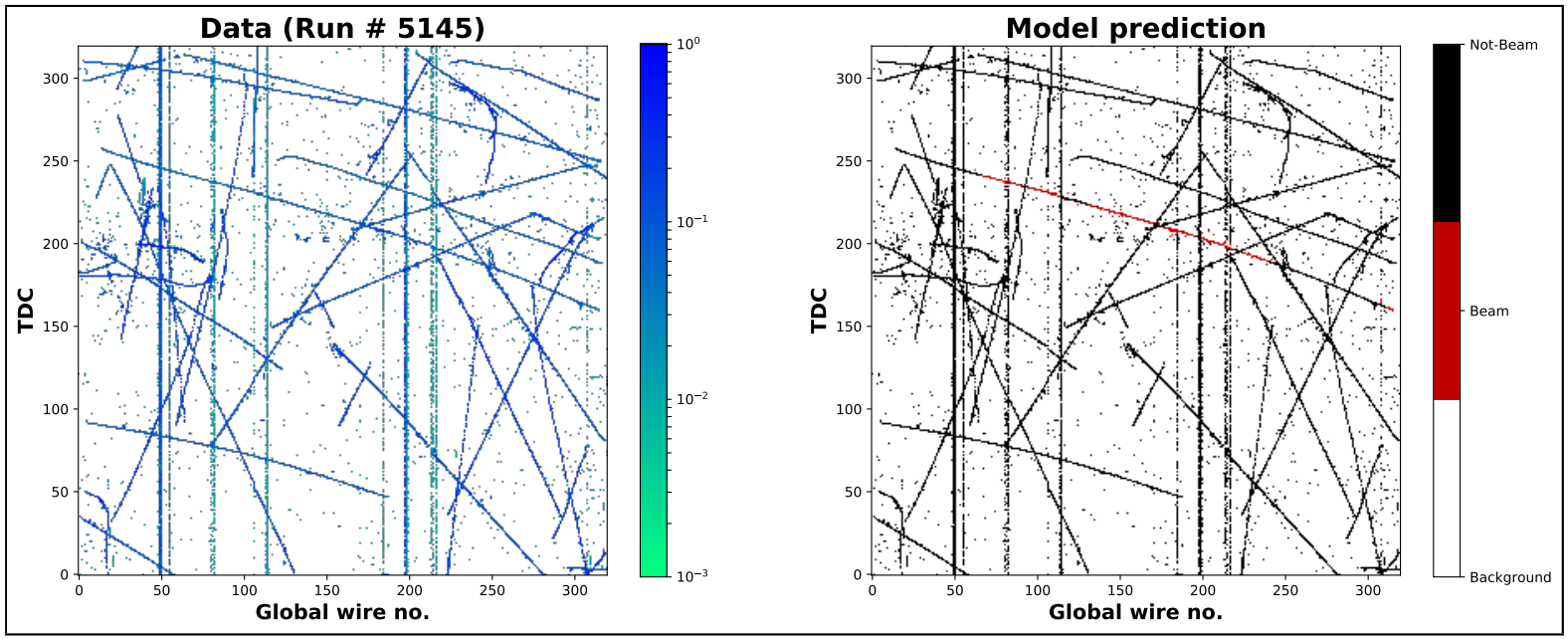

Feature
Label
Prediction (mIOU: 74.3)
TDC
Global wire no.
Not-Beam
Beam
Background

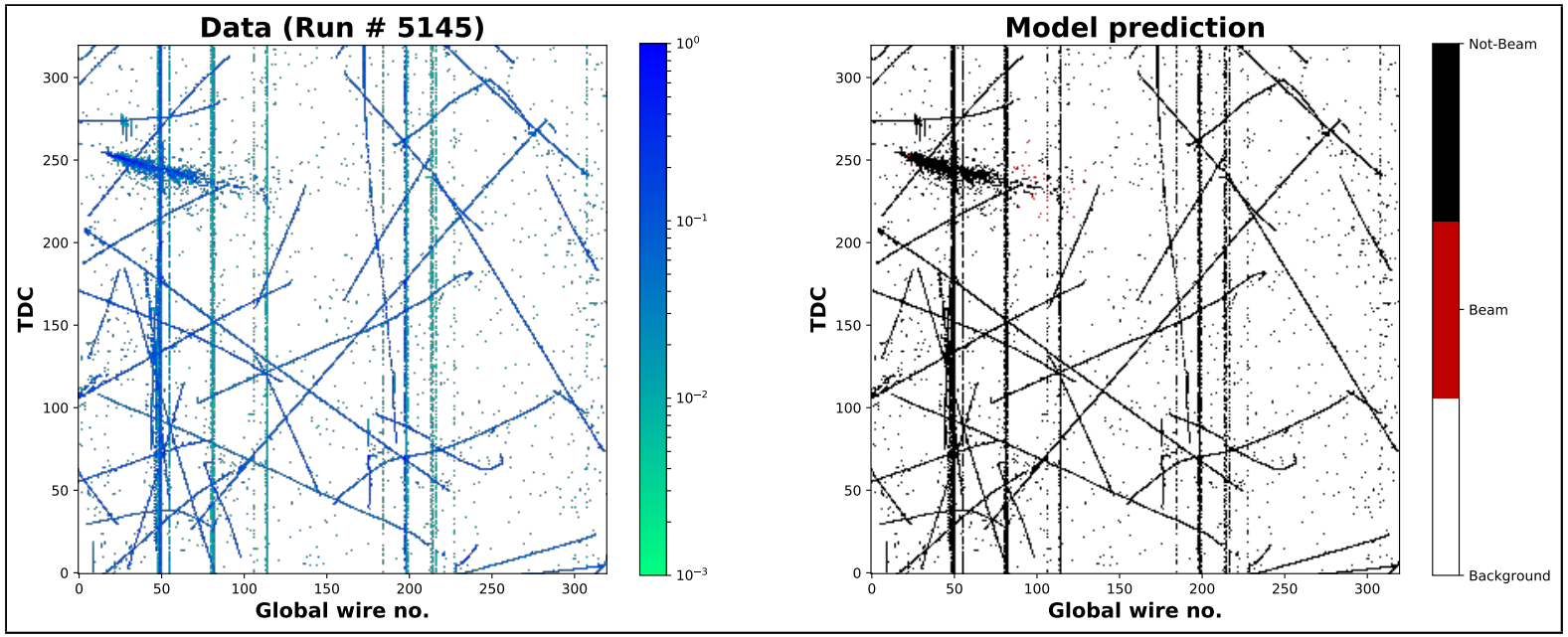
Prediction on beam triggered data

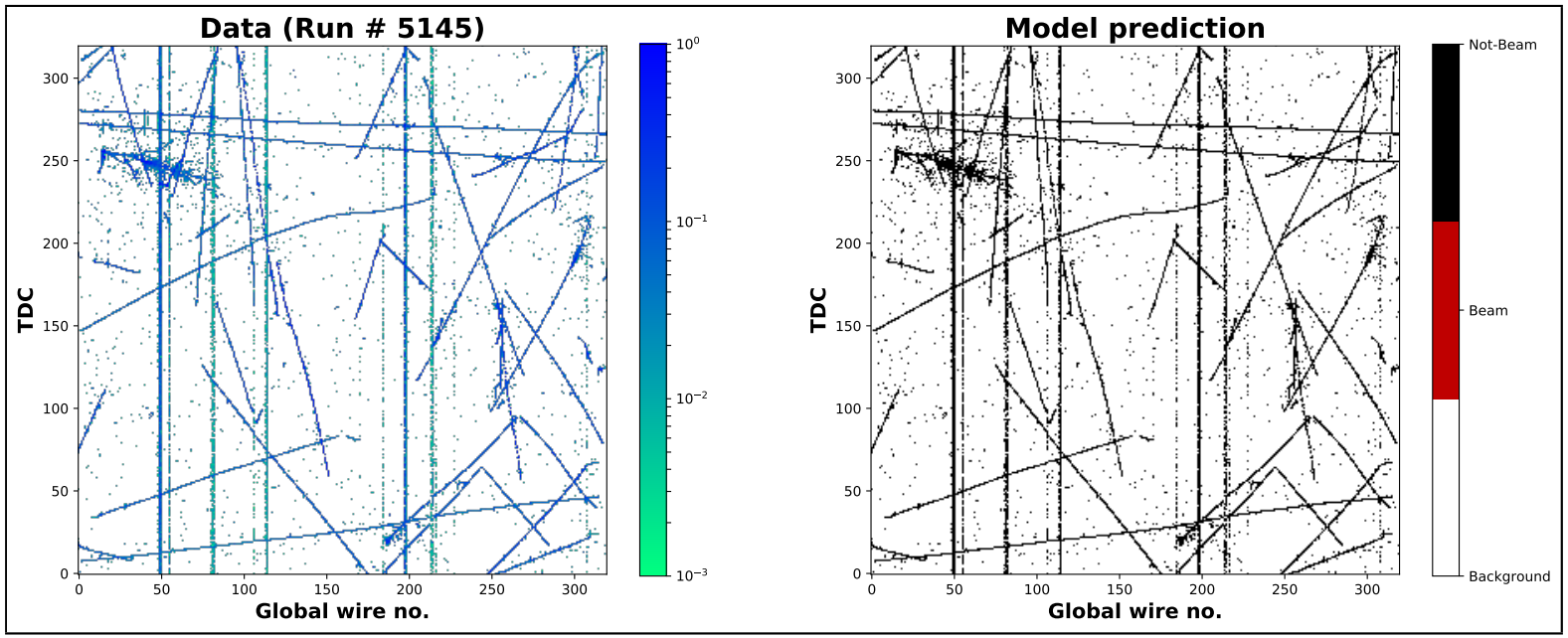

Data (Run # 5145)
Model prediction
TDC
Global wire no.
Not-Beam
Beam
Background


Data (Run # 5145)
Model prediction
TDC
Global wire no.
Not-Beam
Beam
Background


Data (Run # 5145)
Model prediction
TDC
Global wire no.
Not-Beam
Beam
Background


Data (Run # 5145)
Model prediction
TDC
Global wire no.
Not-Beam
Beam
Background


Data (Run # 5145)
Model prediction
TDC
Global wire no.
Not-Beam
Beam
Background


Data (Run # 5145)
Model prediction
TDC
Global wire no.
Not-Beam
Beam
Background

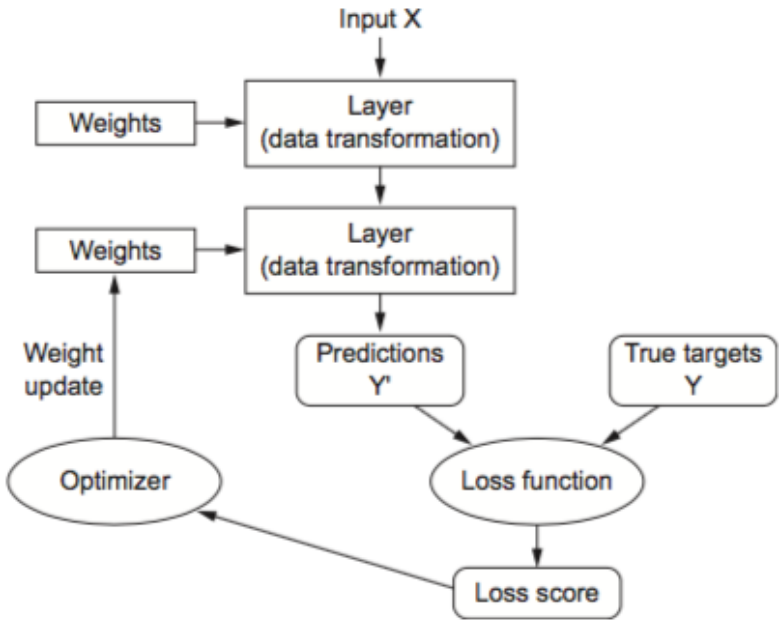
- ⇒ Code to generate training/validation/testing set and data events is in dunetpc feature branch SemanticSegmentationProtoDUNE
- ⇒ Framework for training and everything is in GitHub

Outlook

- ⇒ Optimize method to downsample from 6000×1440 to 320×320
- ⇒ Expand beam class to various particle species; perform optimization on this sample
- ⇒ Using BNL's HPC1 for development work and FNAL's Wilson HPC for training. Proposal to train via PanDA on Oak Ridge's Summit.

Backup

Anatomy Of A Neural Network



Credit: Francois Chollet; Deep learning with Python

- ⇒ At the core, deep learning is about mapping inputs (such as images) to targets (such as the label "background" or "beam")
- ⇒ This mapping is done by a **deep** sequence of simple data transformation, which are **learned** by exposure to examples
- ⇒ To control the output of a neural network, the **loss function** takes the predictions of the network and computes a loss score
- ⇒ The **optimizer** implements Backpropagation algorithm to use this score as a feedback signal to adjust the value of the weights a little (in a direction that will lower the loss score)

Loss function and optimizer

The loss function takes the predictions of the network and computes a loss score. The optimizer uses this score to adjust the value of the weights a little (in a direction that will lower the loss score).
Due to class balancing power of FCN, categorical cross-entropy (or log loss) gets the job done. However, once we start to expand beam class to various particle species, data will be even more unbalanced and sophisticated loss functions maybe necessary.

- ⇒ Modified the categorical cross-entropy loss function (with weights) such that it would penalize more if the network was making mistake on rare classes.
- ⇒ Epoch dependent weighted categorical cross-entropy- weights become larger at a later time.
- ⇒ Focal Loss: authors claim "that the extreme foreground-background class imbalance encountered during training" is the central cause for low accuracy and that "proposed focal loss naturally handles the class imbalance".
- ⇒ Dice loss: Utilizes the Dice score coefficient (DSC)- a harmonic average of the precision and recall.
- ⇒ Modification of F$-\beta$ score