A short overview on “Reducing model bias in a deep learning classifier using domain adversarial neural networks in the MINERA experiment”

Anushree Ghosh, UTFSM, Chile

Fermilab

Date: 2018-11-07

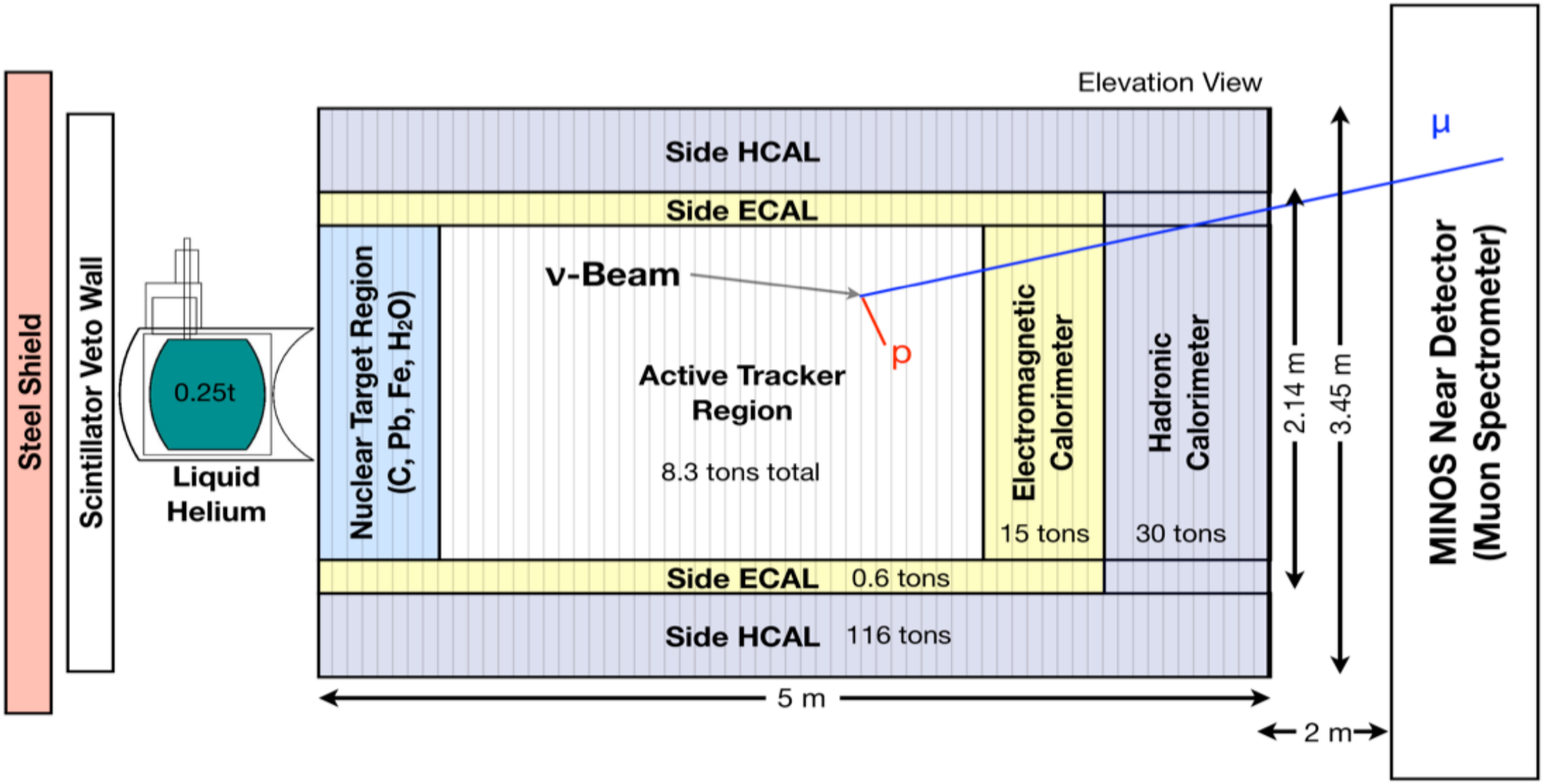
Outline

- MINERvA detector and the problem with the vertex reconstruction in DIS events
- Deep convolutional neural network
- Results from ML based vertex reconstruction
- Implication of domain adversarial neural network to remove/limit the model bias

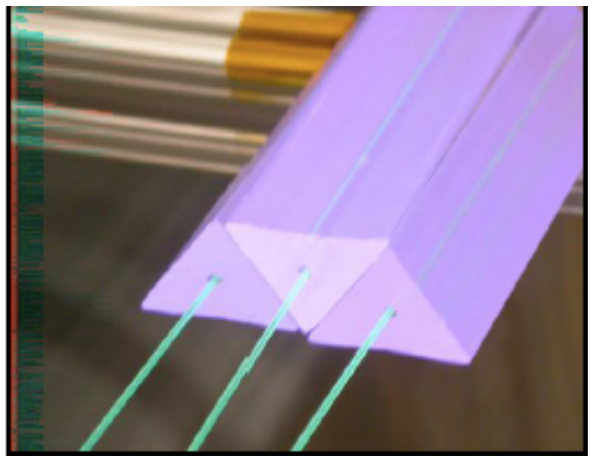
What is model bias?

-We train the ML model using simulated events and test the model on real data.
-Our models are not perfect ->domain discrepancies arises
- Find ways to reduce any biases in the algorithm that may come from training our models in one domain and applying them in another

MINERvA Detector

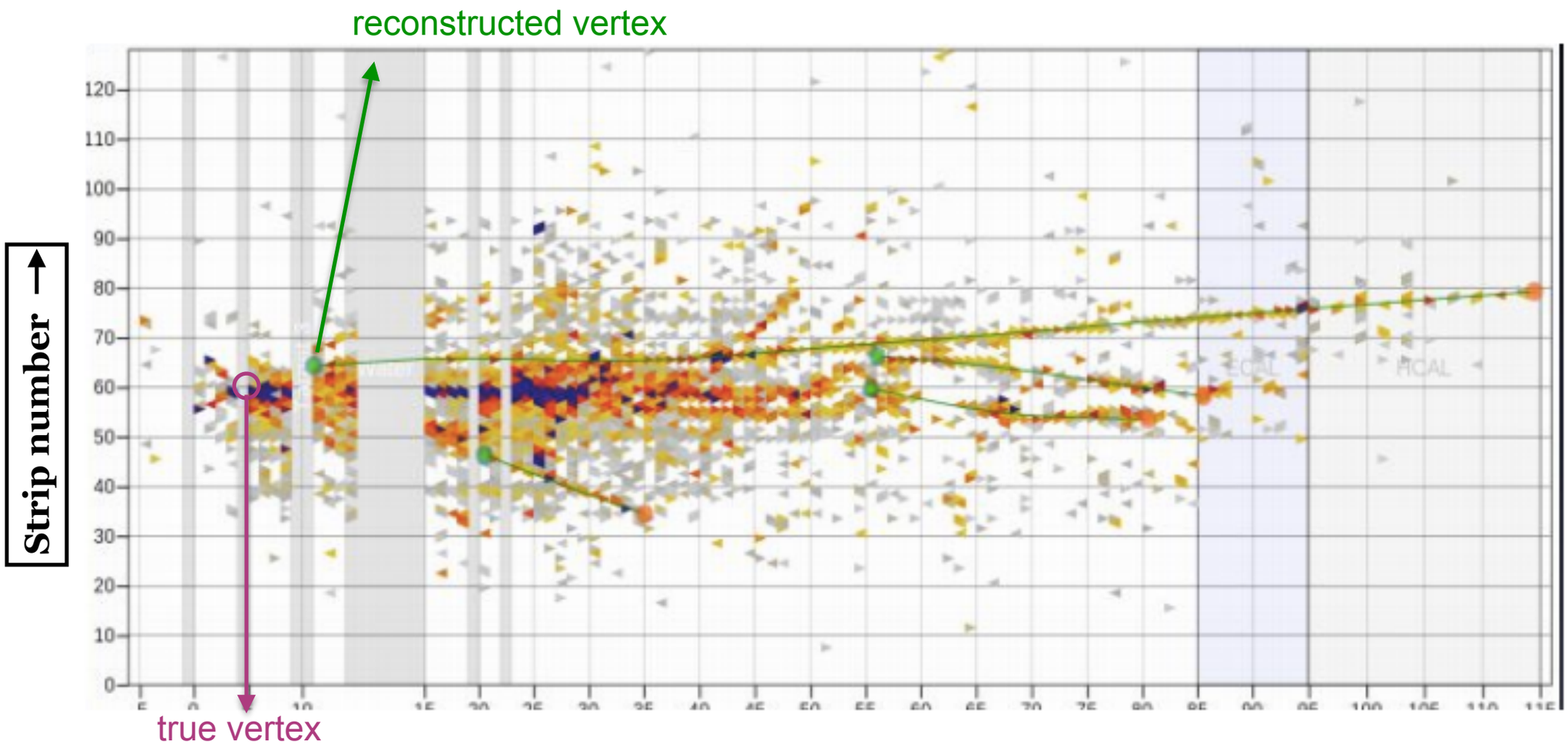


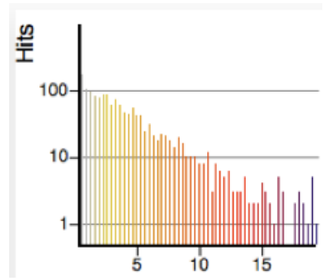
- Consists of a core of scintillator strips surrounded by ECAL and HCAL
- MINOS Near Detector for muon charge and momentum



Problem with vertex finding: motivation behind ML technique

- With the increase of our beam energy, there is an increase in the hadronic showers near the event of interactions.
- Cause more difficulty in vertexing with increase rates of failure in getting the correct vertex position

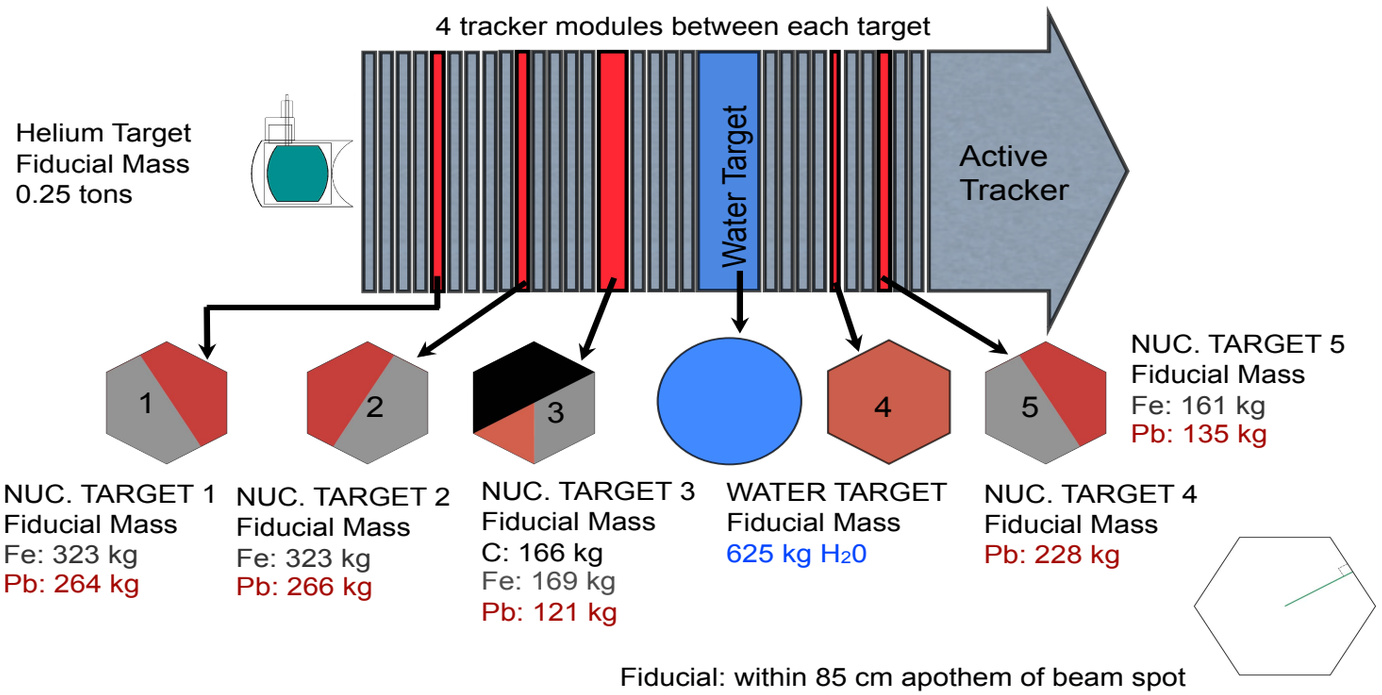


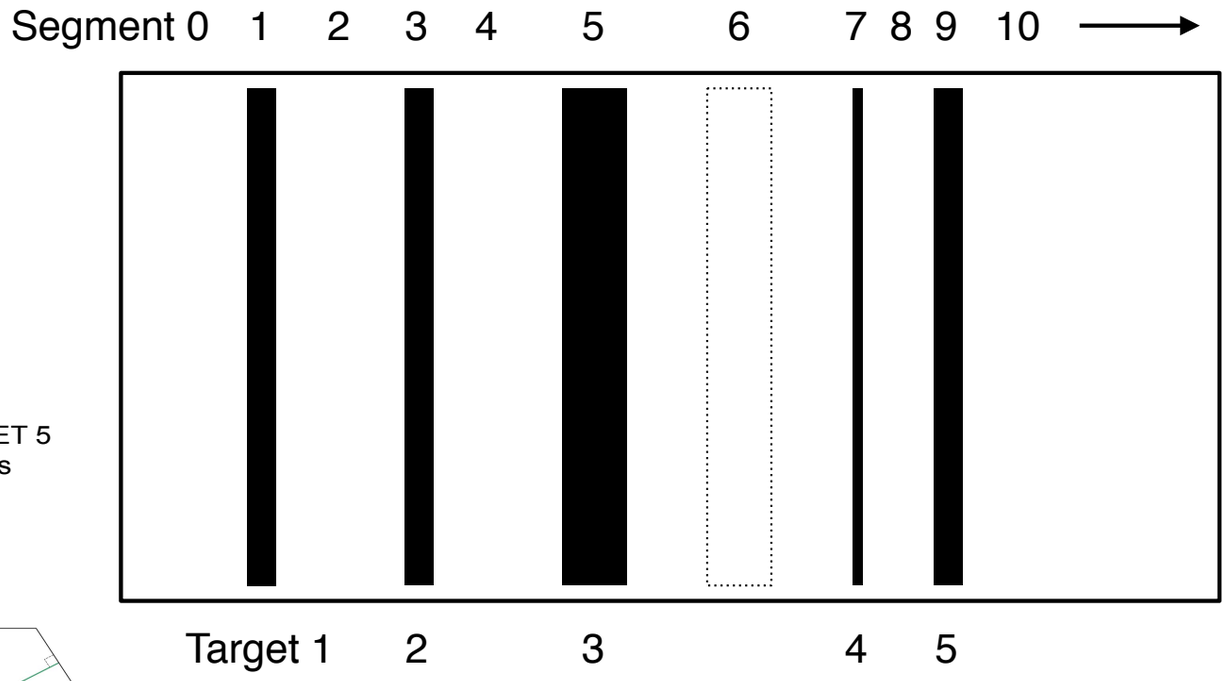


ML Approach To Determine Event Vertex

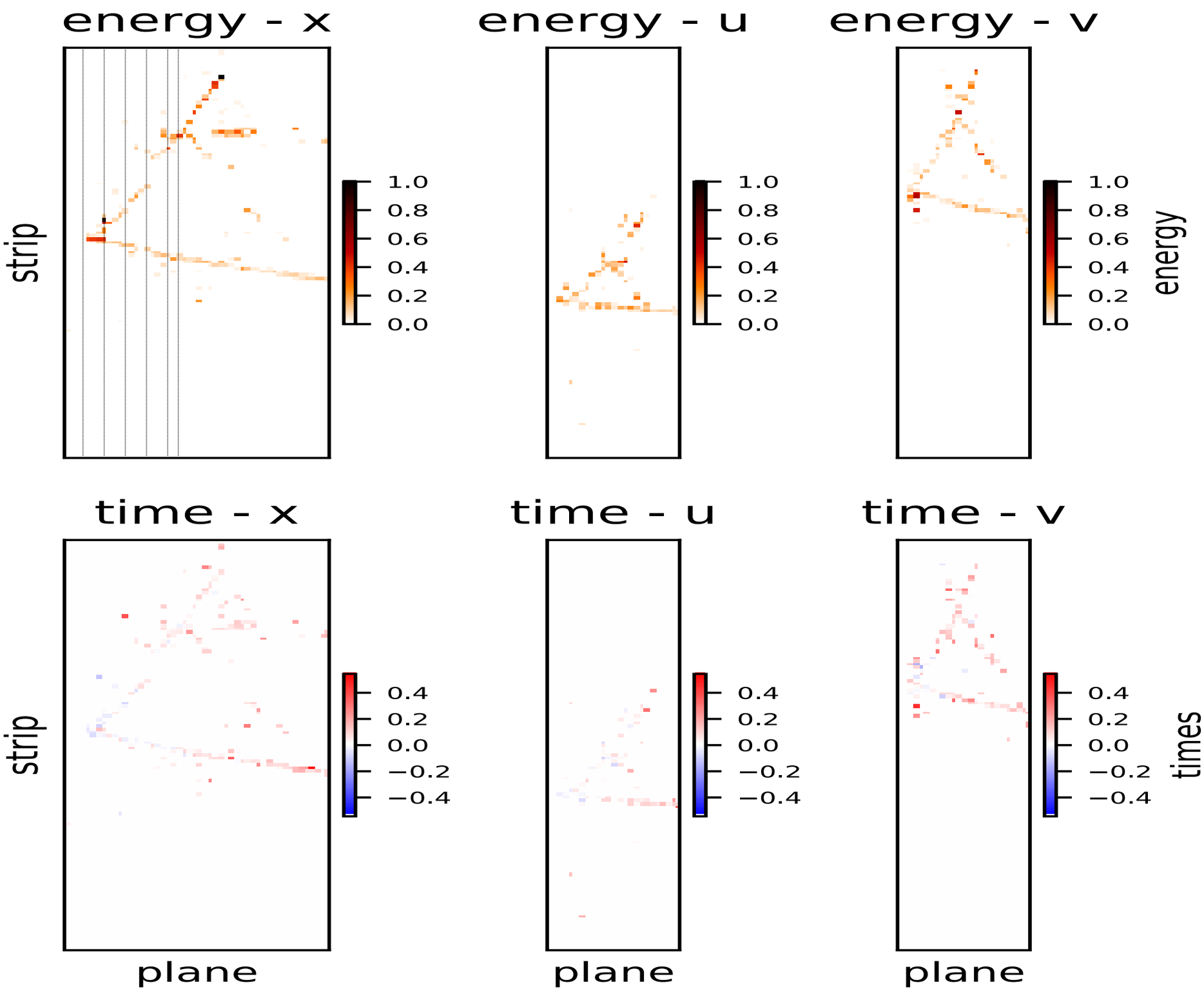
- Goal: Find the location of the event vertex

  -Treat the localization as a classification problem



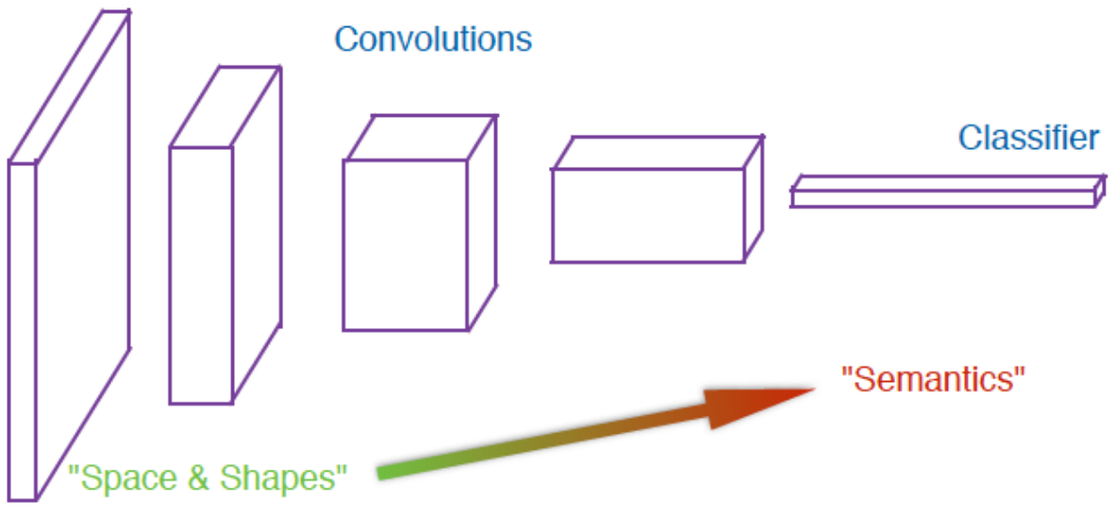


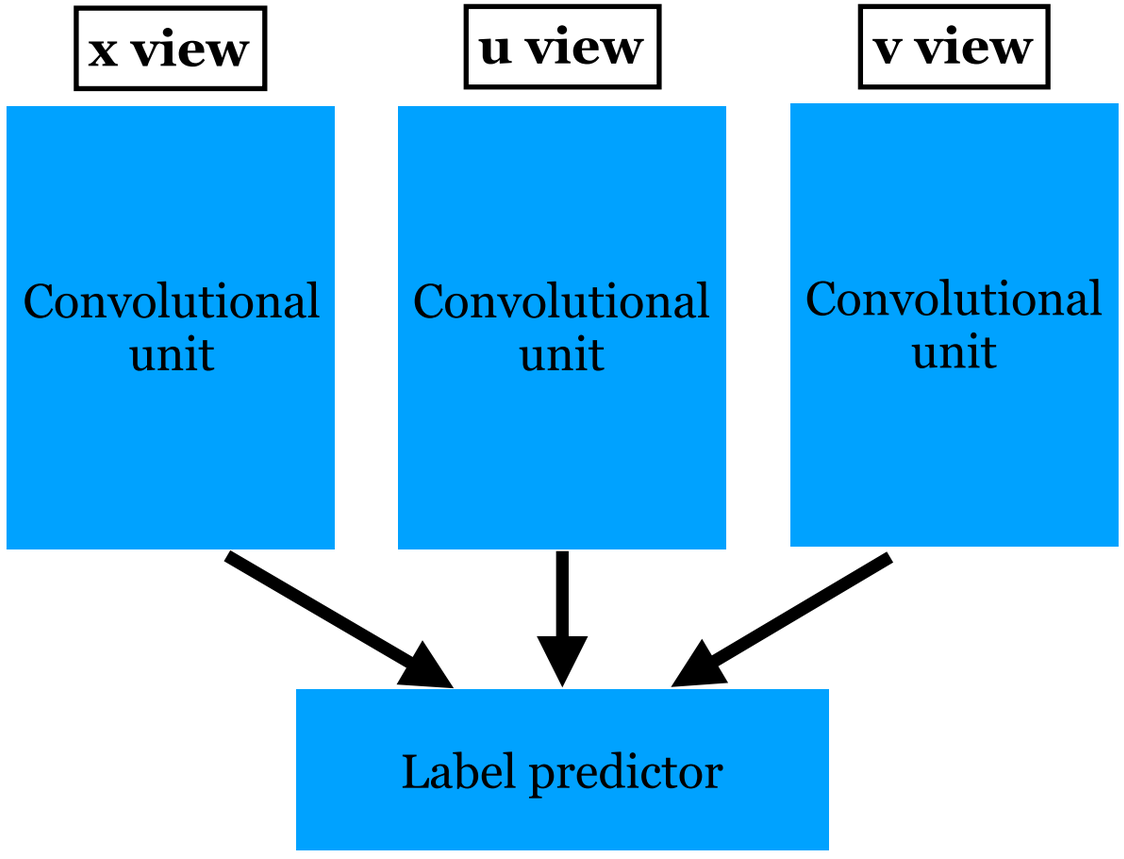
Make images for three different views → DNN(Convolutional neural network) → Prediction at which segment an interaction occurs


energy - x
energy - u
energy - v
strip
1.0
0.8
0.6
0.4
0.2
0.0
energy
time - x
time - u
time - v
0.4
0.2
0.0
−0.2
−0.4
times
plane

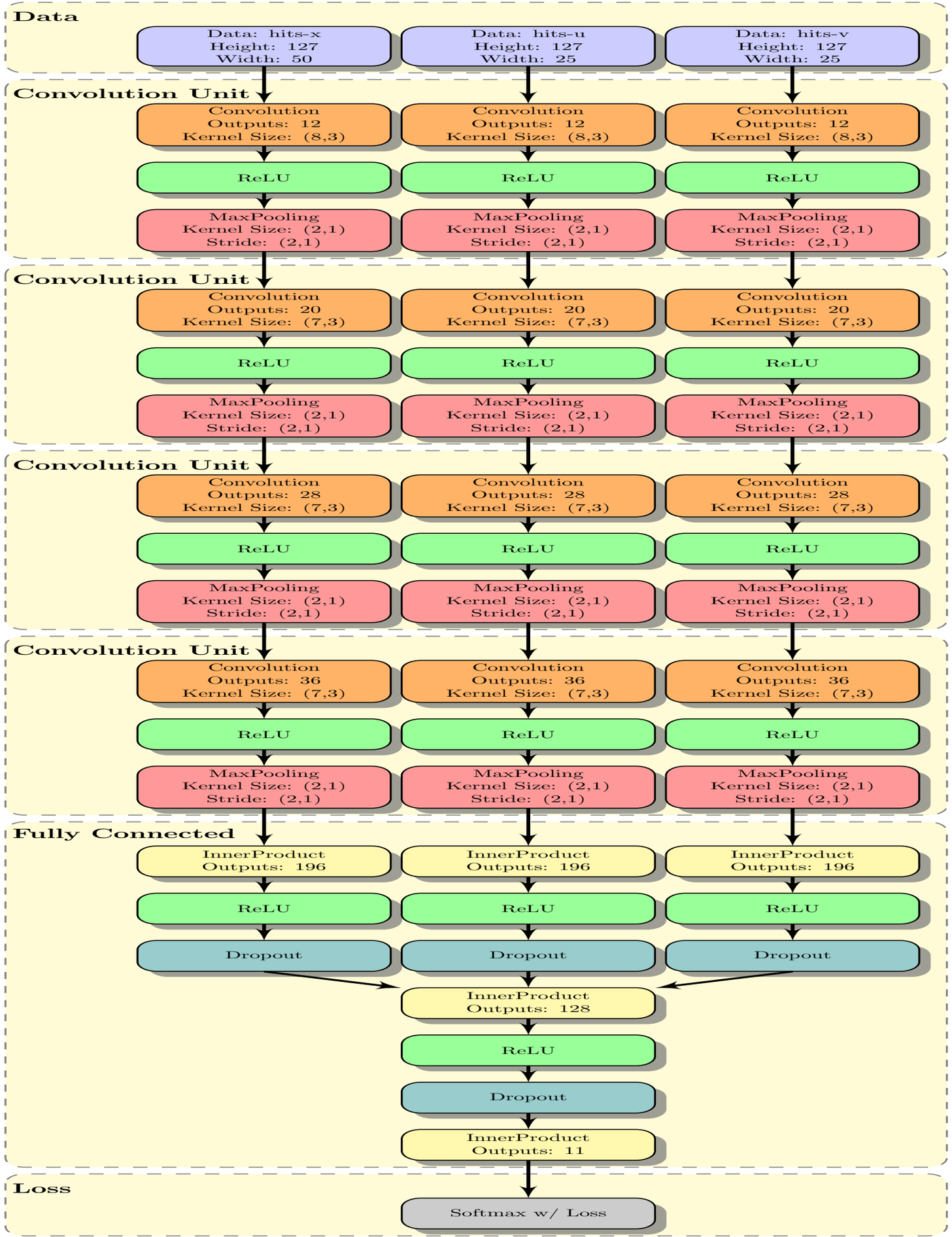
Convolutional neural network (CNN)

Stacking layers of convolutions leads from geometric / spatial representation to semantic representation:





We have three separate convolutional towers that look at each of the X, U, and V images.

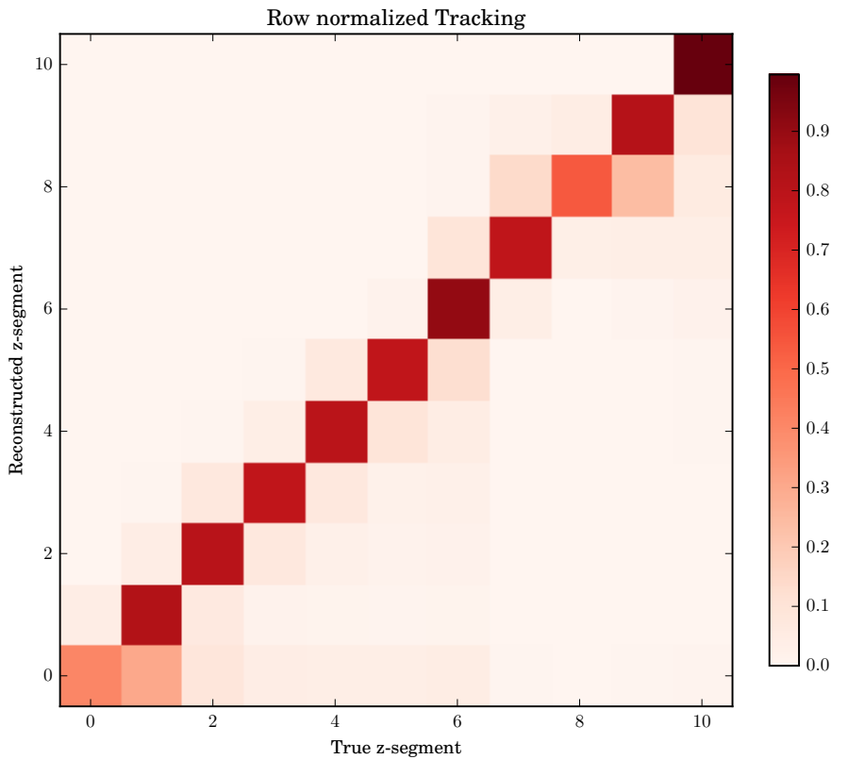

Data
Data: hits-x
Height: 127
Width: 50
Data: hits-u
Height: 127
Width: 25
Data: hits-v
Height: 127
Width: 25
Convolution Unit
Convolution
Outputs: 12
Kernel Size: (8,3)
ReLU
MaxPooling
Kernel Size: (2,1)
Stride: (2,1)
Convolution Unit
Convolution
Outputs: 20
Kernel Size: (7,3)
ReLU
MaxPooling
Kernel Size: (2,1)
Stride: (2,1)
Convolution Unit
Convolution
Outputs: 28
Kernel Size: (7,3)
ReLU
MaxPooling
Kernel Size: (2,1)
Stride: (2,1)
Convolution Unit
Convolution
Outputs: 36
Kernel Size: (7,3)
ReLU
MaxPooling
Kernel Size: (2,1)
Stride: (2,1)
Fully Connected
InnerProduct
Outputs: 196
ReLU
Dropout
InnerProduct
Outputs: 128
ReLU
Dropout
InnerProduct
Outputs: 11
Loss
Softmax w/ Loss

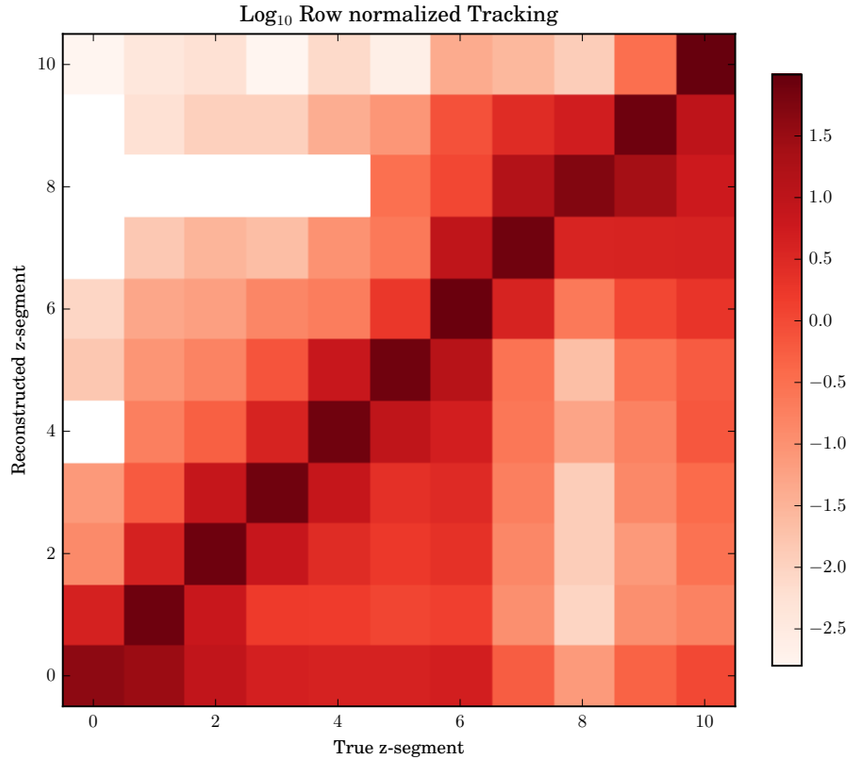
Network structure

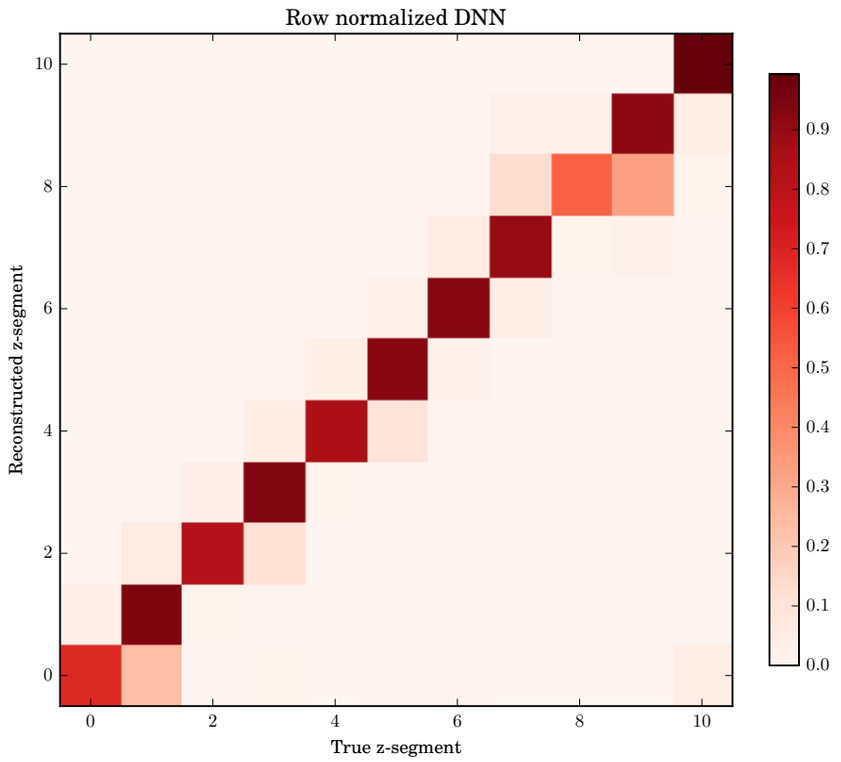
- We have three separate convolutional towers that look at each of the X
- Each tower consists of four iterations of convolution and max pooling layers with ReLUs acting as the non-linear activations and after that there is a fully connected layer
- The out of three views are concatenated and fed into another fully connected layer .This is the input to the final fully connected layer with output -> input to the softmax layer.

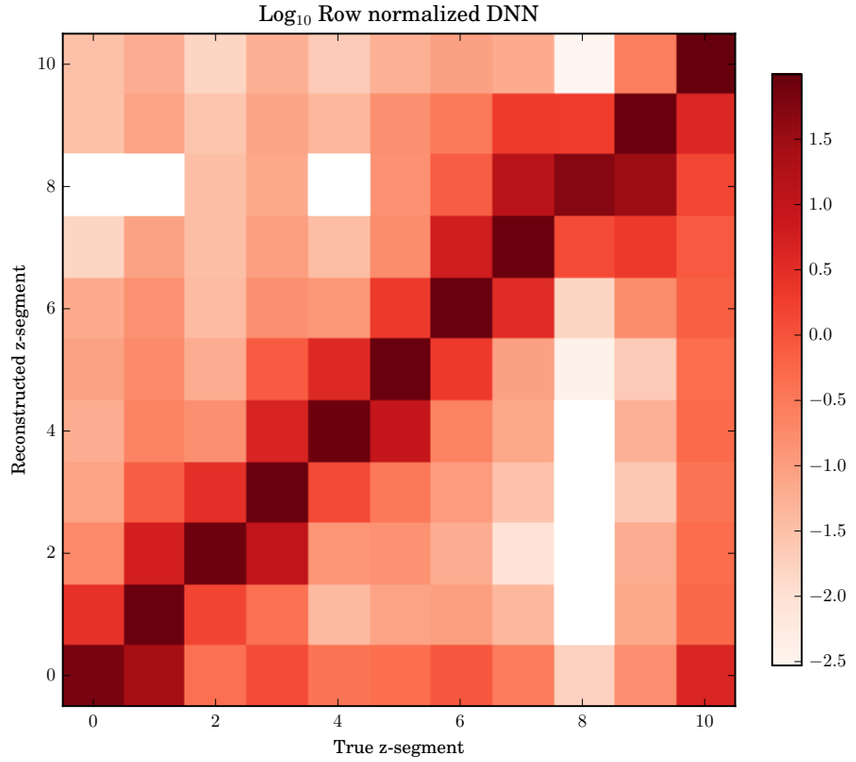
- We use non-square kernels, they are much larger along the transverse direction than along the z direction-> localization information contained directly in the energy distribution along Z. So, we allow the images to shrink along the transverse dimension but largely preserved the image size along the Z axis. Also, we pooled the tensor elements together only along the transverse axis, not along the z axis.

Confusion matrix

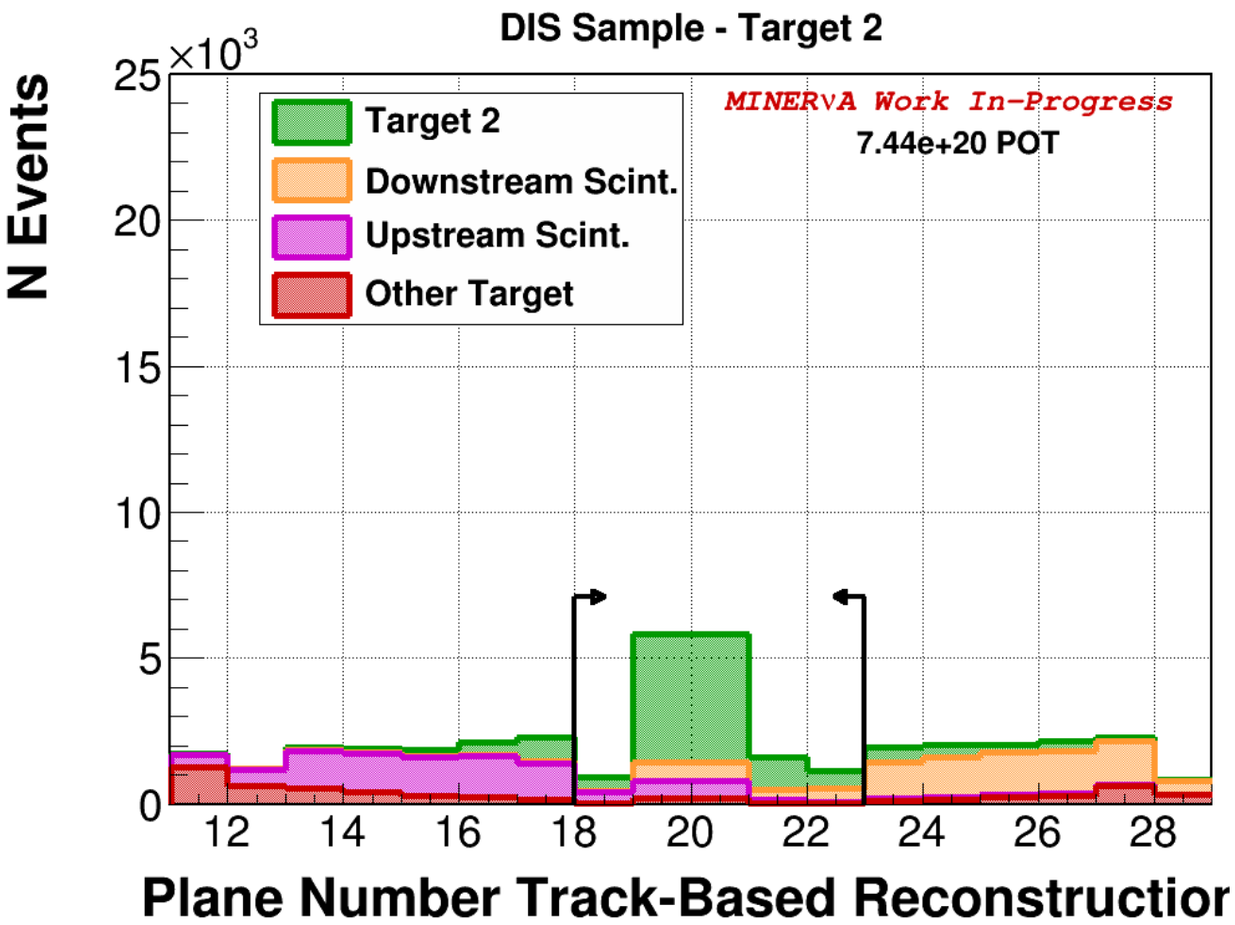


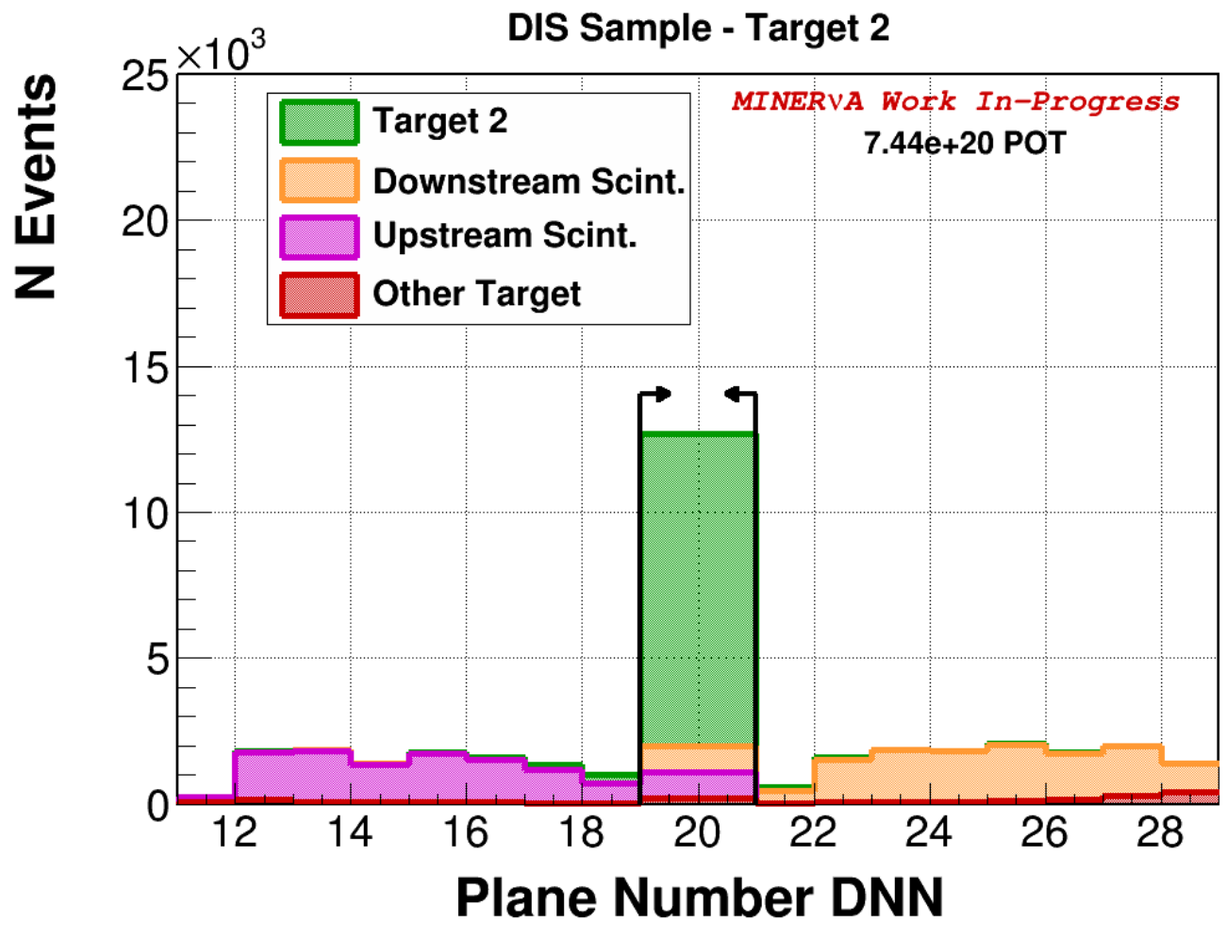






Track-based approach vs ML approach





Signal purity has been improved by the factor of 2-3 using ML technique compared to track based approach

Domain Adversarial Neural Network (DANN)

http://adsabs.harvard.edu/cgi-bin/bib_query?arXiv:1505.07818

CNN:

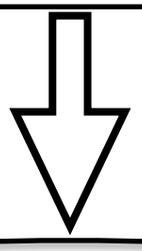
- Train with labeled data: in our case it is Monte Carlo
- Test with unlabeled data: in our case it is real data

Limitation:

Labeled simulated data for training >> unlabeled real data for testing

Our models are not perfect ->domain discrepancies arises

Need strategy to reduce any biases in the algorithm that may come from training our models in one domain and applying them in another



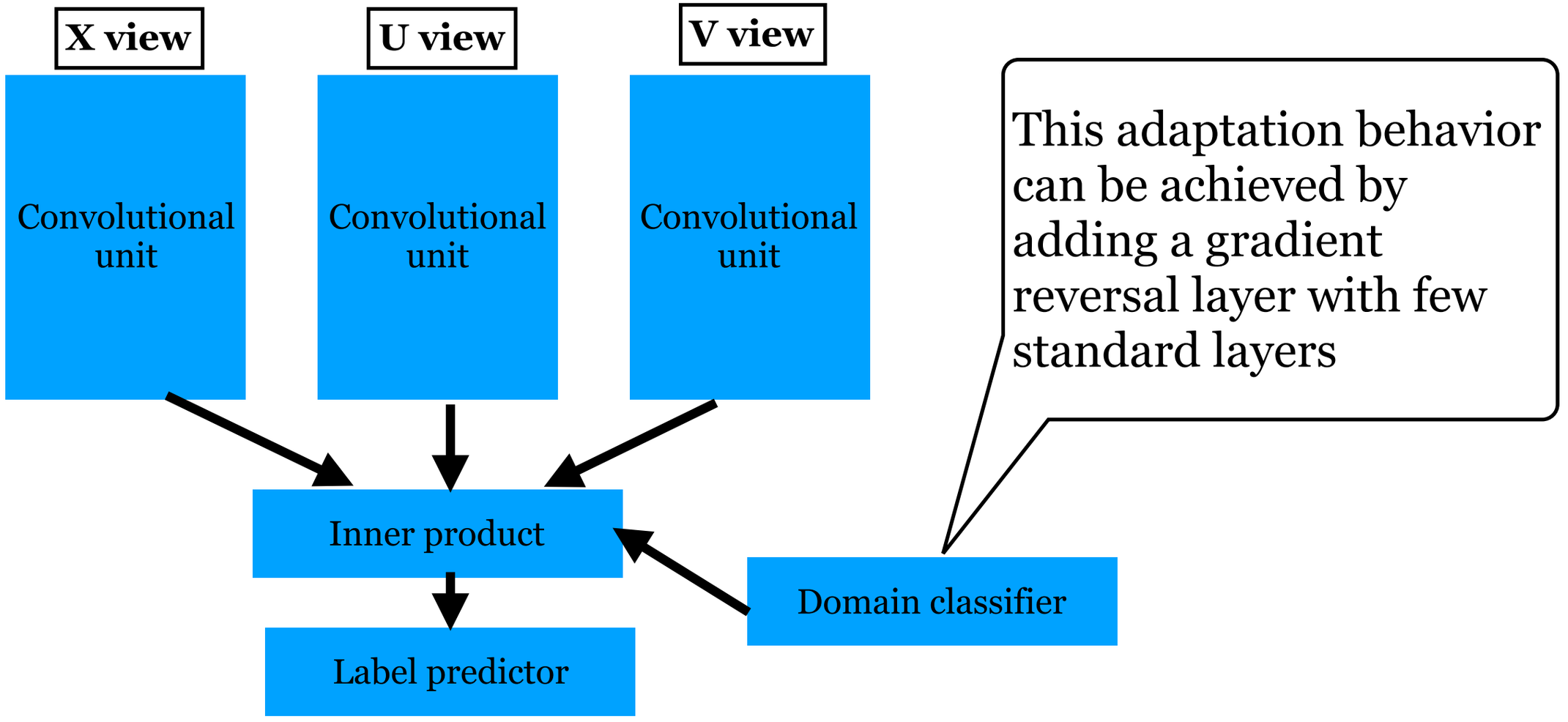
Here DANN comes into the picture

DANN

Train from the ***labeled source domain (MC)*** and ***unlabeled target domain (real data)***

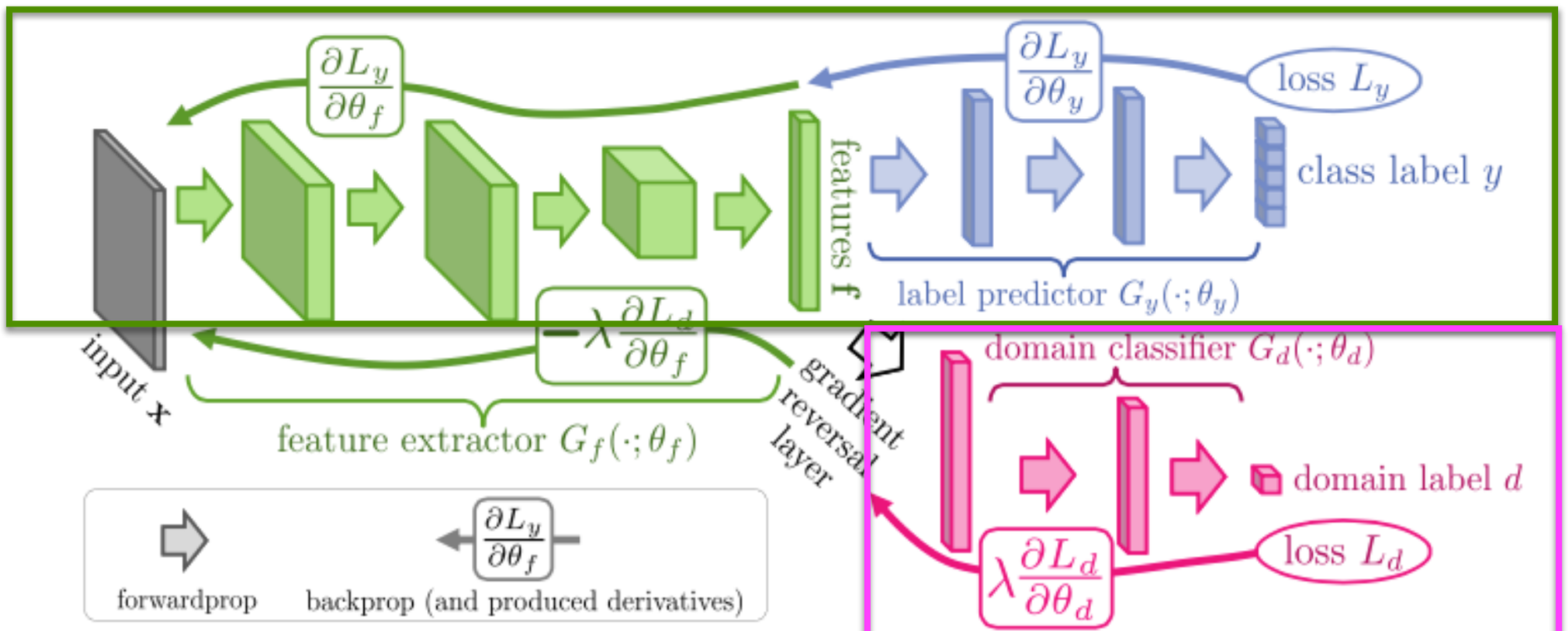
Goal to achieve the features:

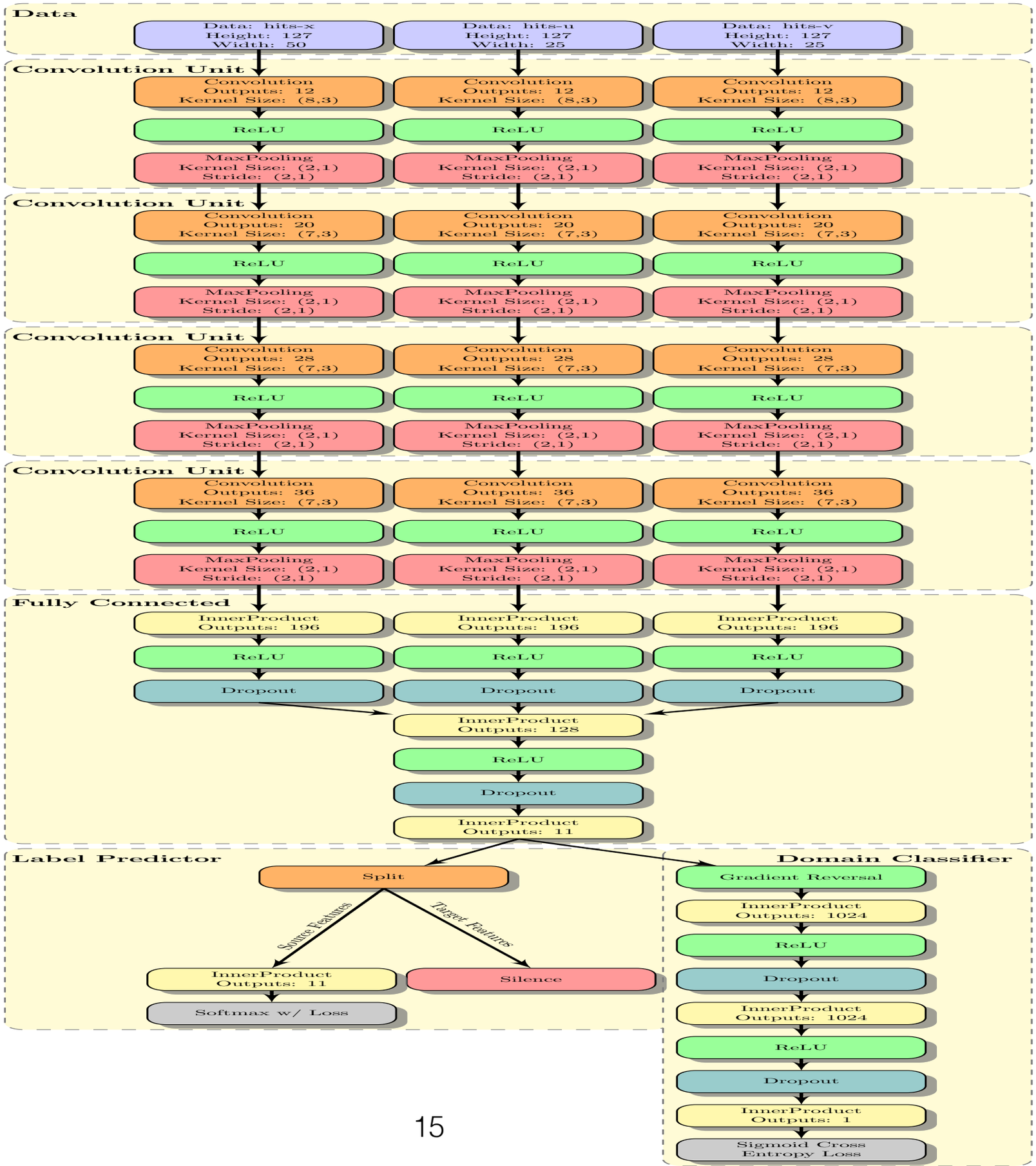
1) ***discriminative*** for the main learning task on the source domain
2) ***indiscriminate*** with respect to the ***shift between domains***



DANN

- Two classifiers into the network:
  - Label predictor: output
  - Domain classifier: works internally
- Minimize the loss of the label classifier so that network can predicts the input level
- Maximize the loss of the domain classifier so that network can not distinguish between source and target domain.
- The network develops an insensitivity to features that are present in one domain but not the other, and train only on features that are common to both domains.




Data
Data: hits-x
Height: 127
Width: 50
Data: hits-u
Height: 127
Width: 25
Data: hits-v
Height: 127
Width: 25
Convolution Unit
Convolution
Outputs: 12
Kernel Size: (8,3)
ReLU
MaxPooling
Kernel Size: (2,1)
Stride: (2,1)
Convolution Unit
Convolution
Outputs: 20
Kernel Size: (7,3)
ReLU
MaxPooling
Kernel Size: (2,1)
Stride: (2,1)
Convolution Unit
Convolution
Outputs: 28
Kernel Size: (7,3)
ReLU
MaxPooling
Kernel Size: (2,1)
Stride: (2,1)
Convolution Unit
Convolution
Outputs: 36
Kernel Size: (7,3)
ReLU
MaxPooling
Kernel Size: (2,1)
Stride: (2,1)
Fully Connected
InnerProduct
Outputs: 196
ReLU
Dropout
InnerProduct
Outputs: 128
ReLU
Dropout
InnerProduct
Outputs: 11
Label Predictor
Split
Source Features
Target Features
InnerProduct
Outputs: 11
Softmax w/ Loss
Silence
Domain Classifier
Gradient Reversal
InnerProduct
Outputs: 1024
ReLU
Dropout
InnerProduct
Outputs: 1024
ReLU
Dropout
InnerProduct
Outputs: 1
Sigmoid Cross
Entropy Loss

How to test DANN ?

- Find source and target with distinct features.
  - our source and target domains may be too similar for the domain classifier to be able to distinguish between them.

We train with Monte Carlo (MC) events and use different MC as target

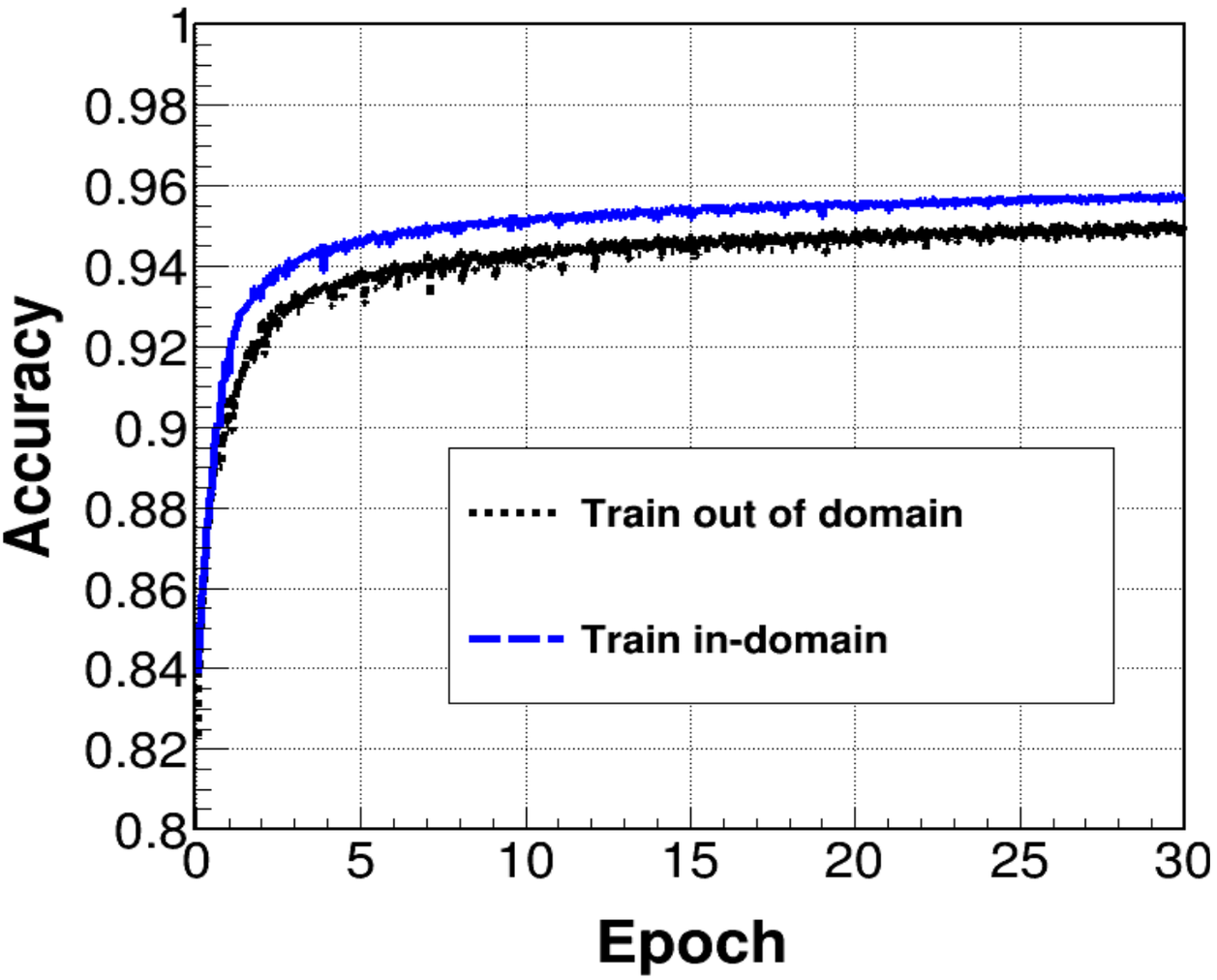
- We tried by few ways to get the target sample having different features than source: changing the flux, physics model, kinematic division etc.

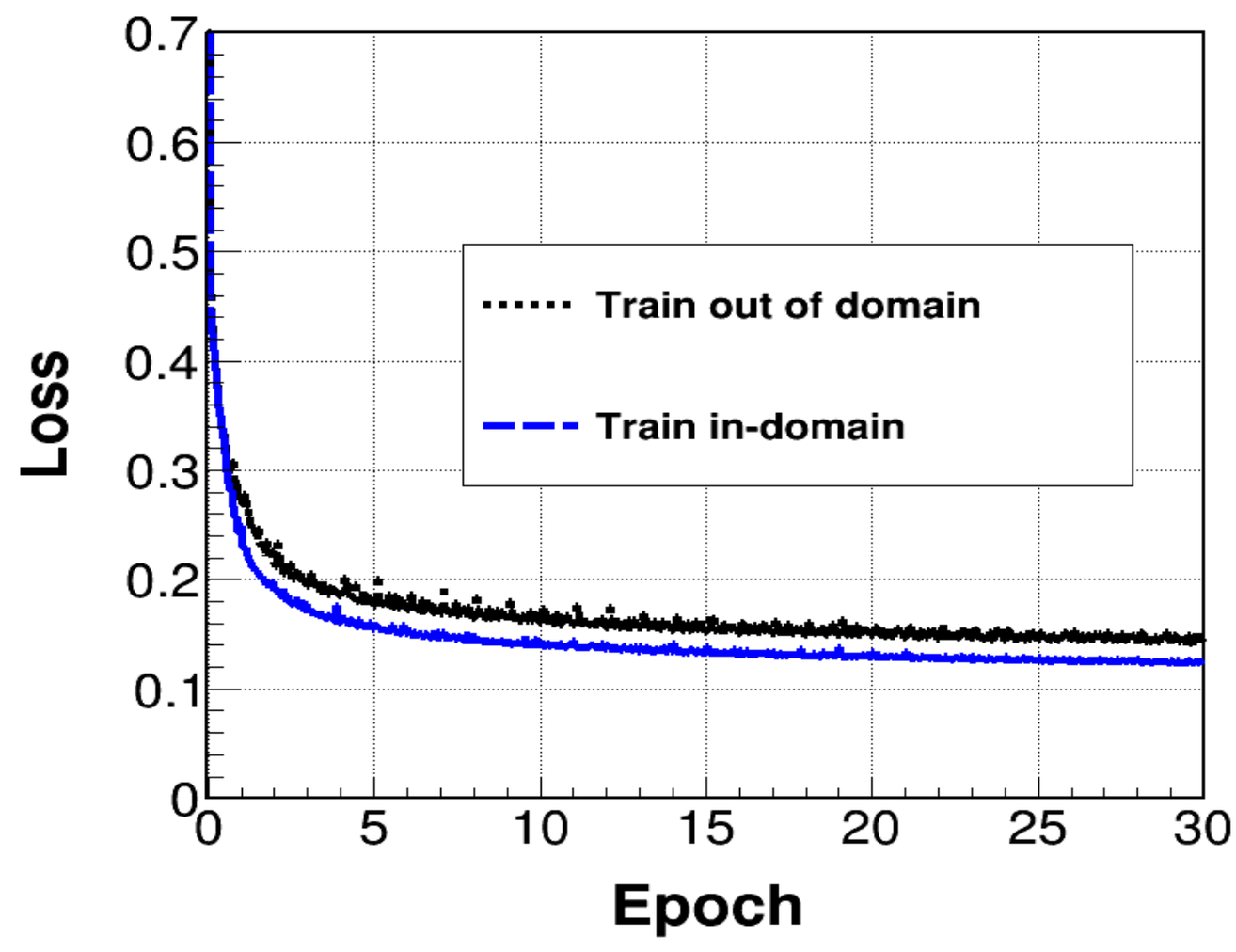
Final state interaction(FSI) On/Off

- We assume that "***FSI is on***" in real world and so we ***turned on FSI*** in our ***testing sample***

Training sample (Source domain)	DANN partner (target domain)	Testing sample	Model
FSI off (1.2M)	**N/A**	**FSI on**	**out of domain**
FSI on(1.2M)	**N/A**	**FSI on**	**In domain**

The expectation: ***CNN in domain*** will perform *better* than ***CNN out of domain***.





Blue vs black: model trained in the same domain (FSI active,Blue curve) is ***better*** than a model trained with an out-of domain physics model (FSI inactive, black curve)

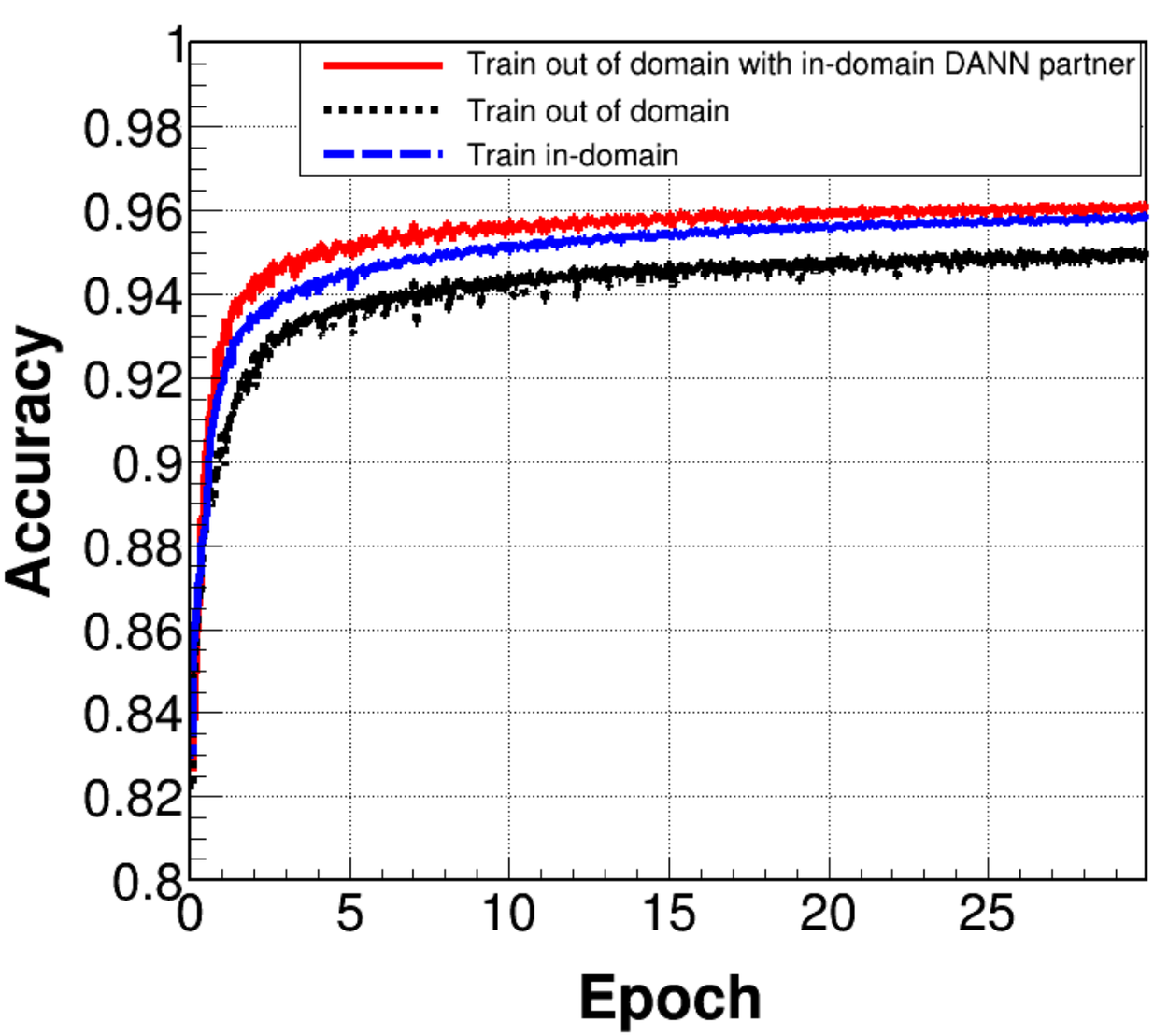
Final state interaction(FSI) On/Off

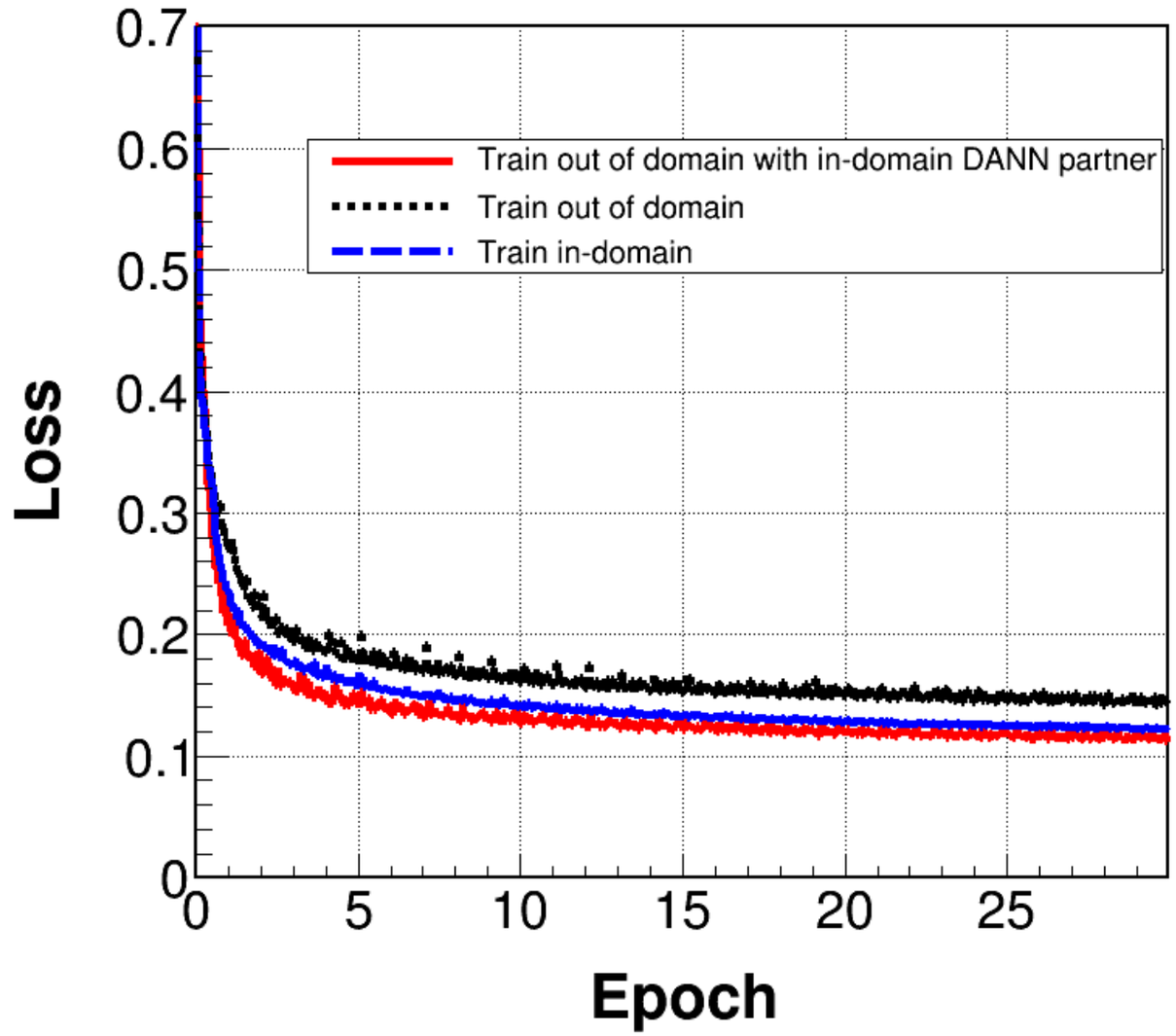
- We assume that "***FSI is on***" in real world and so we ***turned on FSI*** in our ***testing sample***

Training sample (Source domain)	DANN partner (target domain)	Testing sample	Model
FSI off (1.2M)	**N/A**	**FSI on**	**Out of domain**
FSI on(1.2M)	**N/A**	**FSI on**	**In domain**
FSI off(1.2M)	**FSI off(1.2M)**	**FSI on**	**Out of domain with in domain DANN partner**

The expectation: ***CNN in domain*** will perform *better* than ***CNN out of domain***.

The expectation: though model is trained "out of domain", it would show the ***similar performance*** as ***"CNN in domain"*** since we consider ***"in domain" DANN partner.***





Red curve: Adding a DANN partner to the model trained in the out-of domain we are able to ***recover the performance*** of the model natively trained in the correct domain

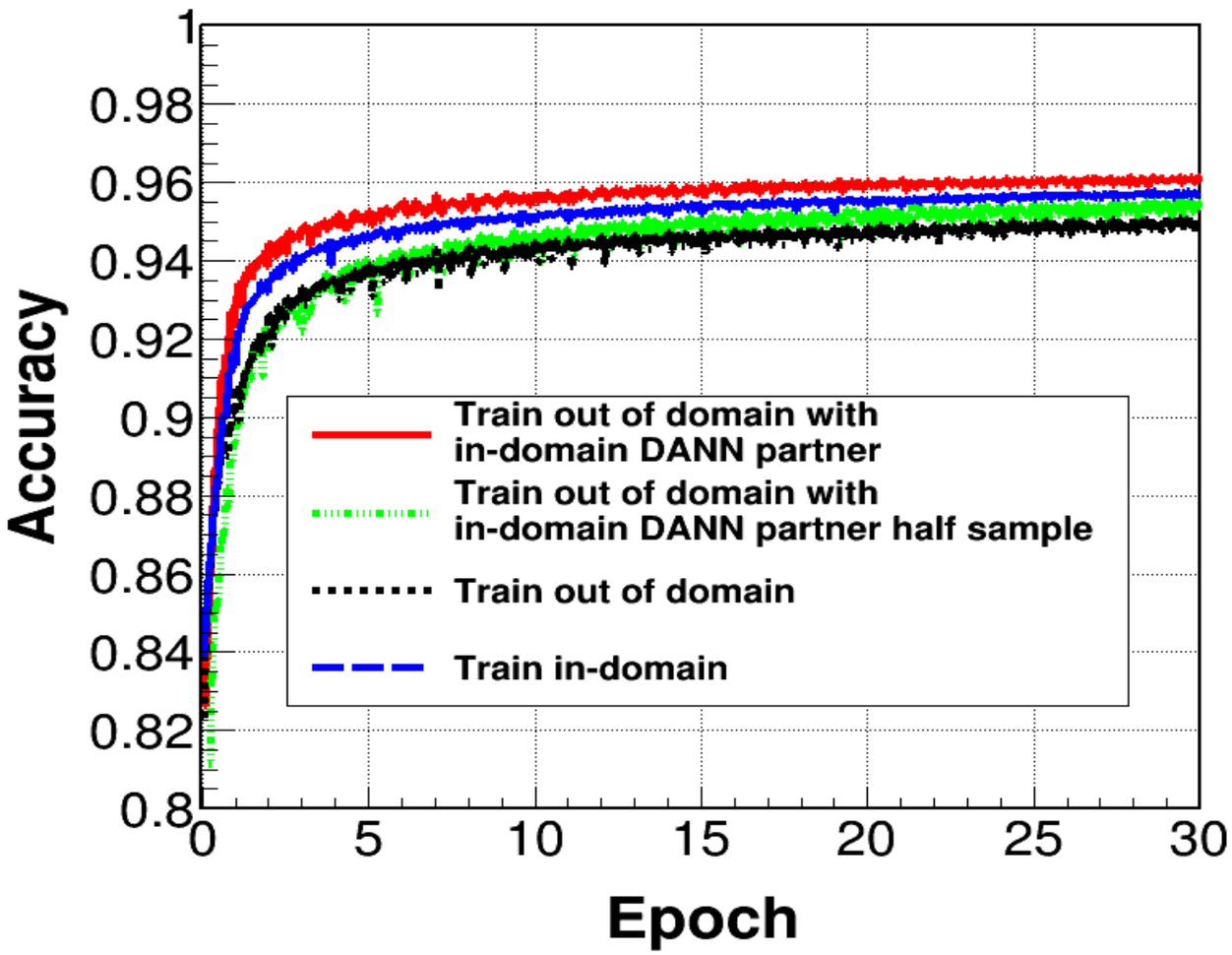
Final state interaction(FSI) On/Off

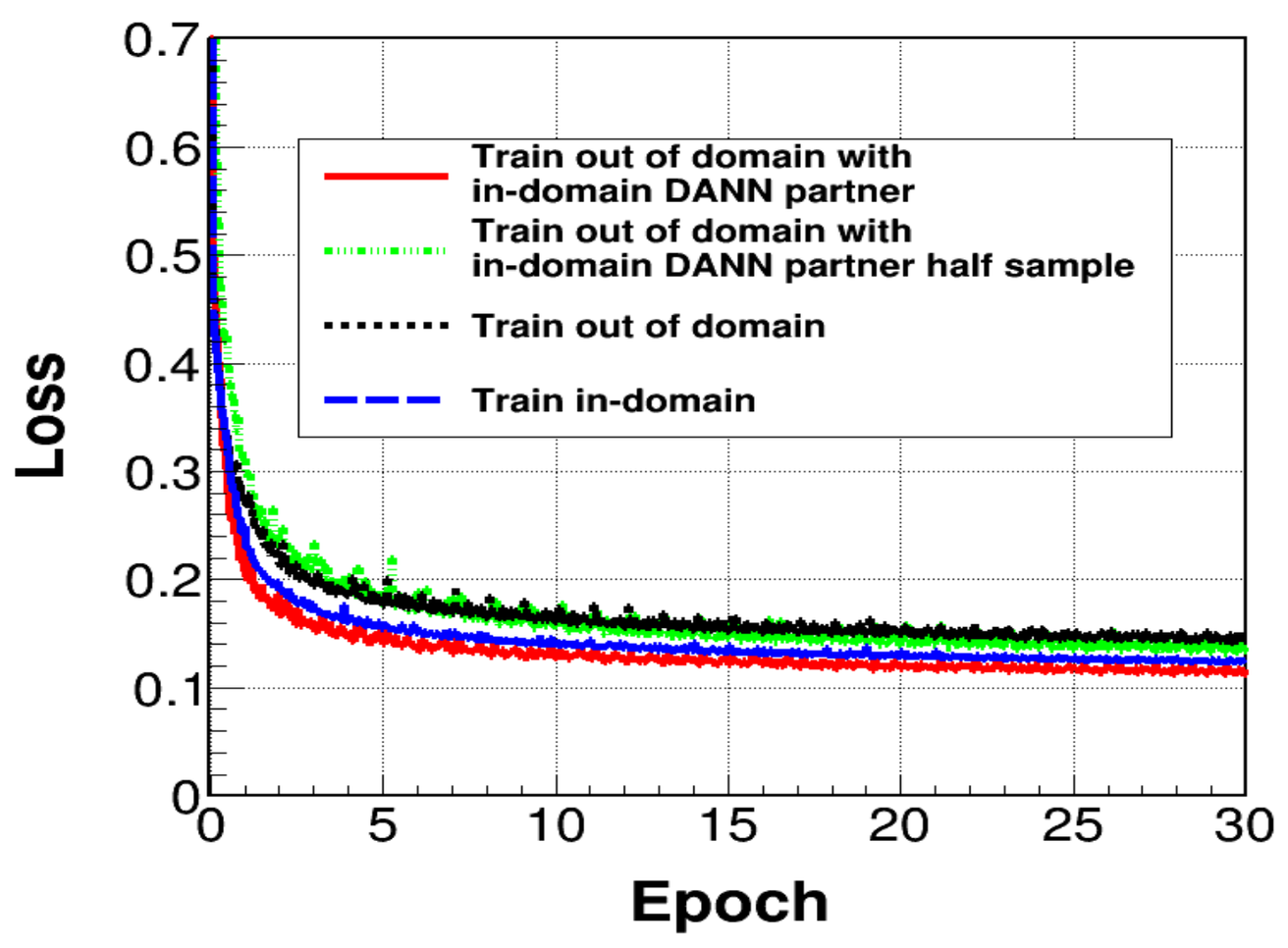
- We assume that "***FSI is on***" in real world and so we ***turned on FSI*** in our ***testing sample***

Training sample (Source domain)	DANN partner (target domain)	Testing sample	Model
FSI off (1.2M)	**N/A**	**FSI on**	**Out of domain**
FSI on(1.2M)	**N/A**	**FSI on**	**In domain**
FSI off(1.2M)	**FSI off(1.2M)**	**FSI on**	**Out of domain with in domain DANN partner**
FSI off(0.6M)	**FSI off(0.6M)**	**FSI on**	**Out of domain with in domain DANN partner(half sample)**

The expectation: ***CNN in domain*** will perform *better* than ***CNN out of domain***.

The expectation: though model is trained "out of domain", it would show the ***similar performance*** as ***"CNN in domain"*** since we consider ***"in domain" DANN***





Green curve:

- ***Perform worse*** than red curve as the sample size is reduced by half
- ***Perform better*** than than black curve as it has information ***from the correct domain***

DANN helps to recover the domain information

Summary

- We see ***improvement factor of ~2-3*** with DNN based reconstruction over track-based reconstruction

- We simulated with different FSI behavior and we saw the cross-domain performance degradation. However, ***by using DANN*** to restrict the feature extraction only to features in both domains ***we can train a domain-invariant classifier***

- MINERvA is expanding ML infrastructure in other studies like hadron multiplicity, particle identification and we will use ***DANN to reduce the bias coming from the physics model.***

Backup slides

classifying events in 11 segments

Target	Track-Based Row Normalized Event Counts +stat error (%)	DNN Row Normalized Event Counts+stat error (%)	Improvement+ stat error (%)
Upstream of Target 1	41.11±0.95	68.1±0.6	27±1.14
1	82.6±0.26	94.4±0.13	11.7±0.3
Between target 1 and 2	80.8±0.46	82.1±0.37	1.3±0.6
2	77.9±0.27	94.0±0.13	16.1±0.3
Between target 2 and 3	80.1±0.46	84.8±0.34	4.7±0.6
3	78±0.3	92.4±0.16	14.4±0.34
Between target 3 and 4	90.5±0.2	93.0±0.14	2.5±0.25
4	78.3±0.35	89.6±0.22	11.3+0.42
Between target 4 and 5	54.3±1.12	51.6±0.95	-2.7±0.15
5	81.6±0.3	91.2±0.18	9.5±0.34
Downstream of target 5	99.6±0.01	99.3±0.13	-0.3±0.02