Technical requirements to reconstruction software in IceCube

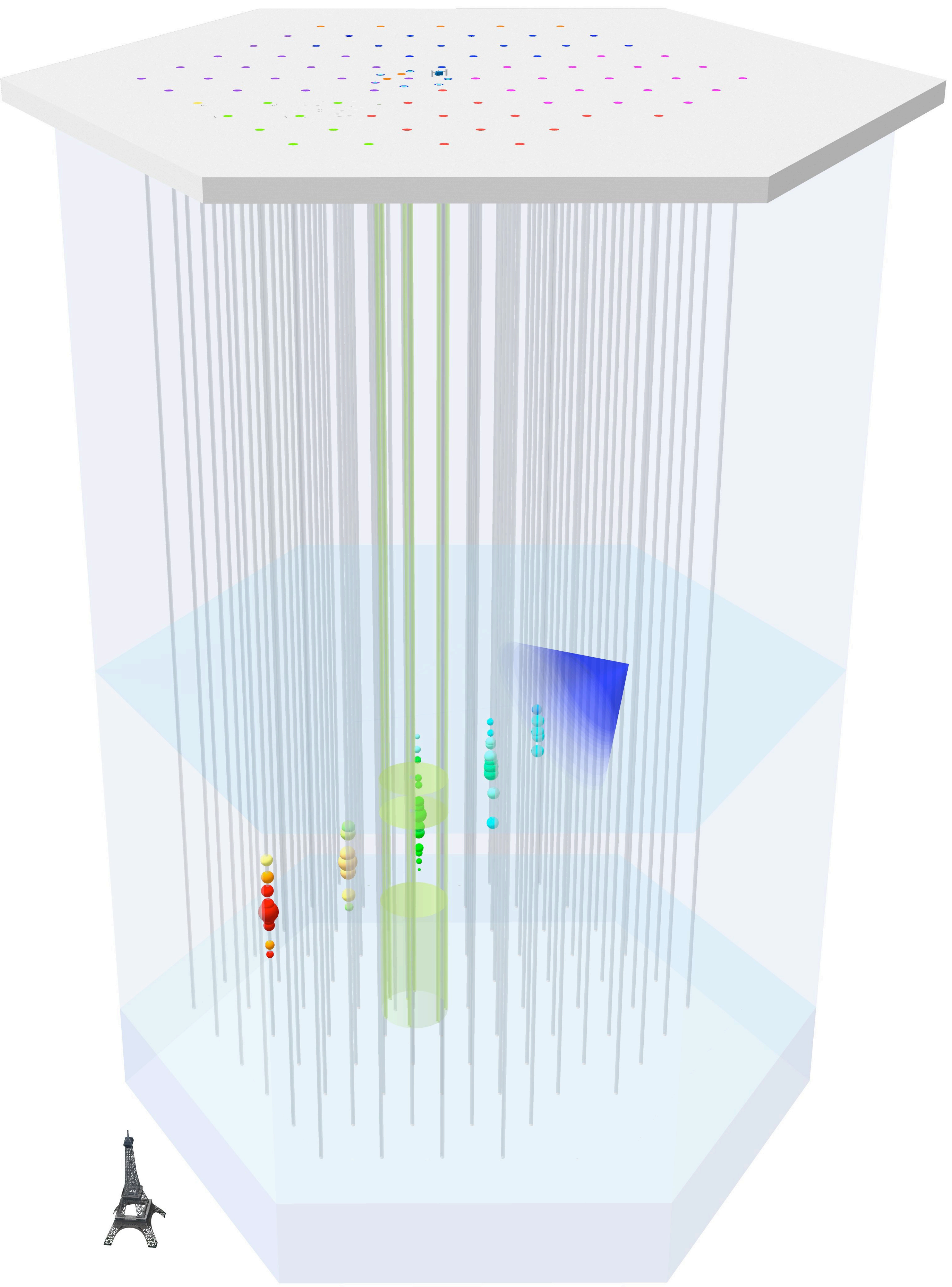
Philipp Soldin

GraphNeT Workshop V

18.08.2025

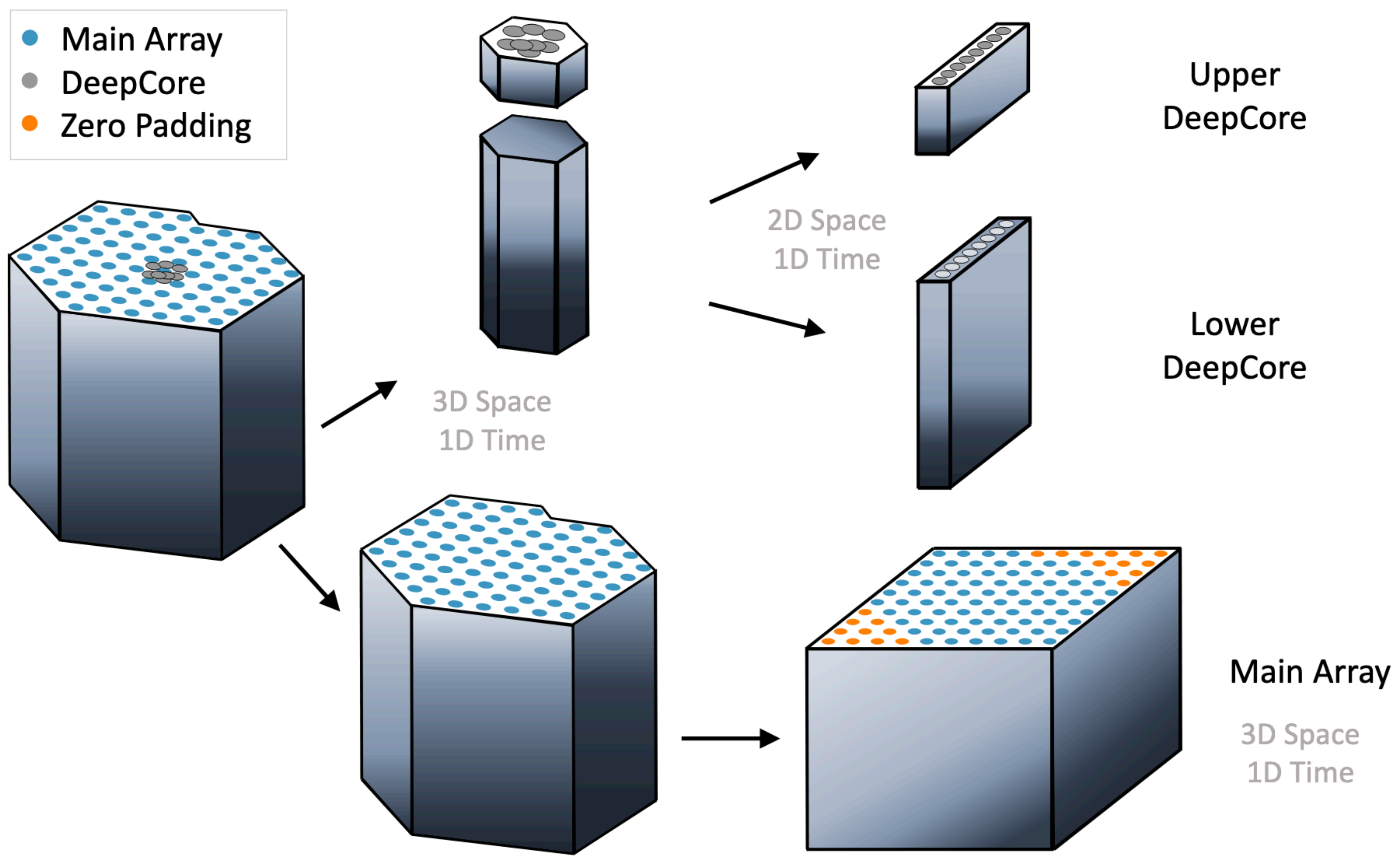
Overview

- Network Architectures in IceCube
  - What I am working on right now & design challenges
- Requirements from IceCube

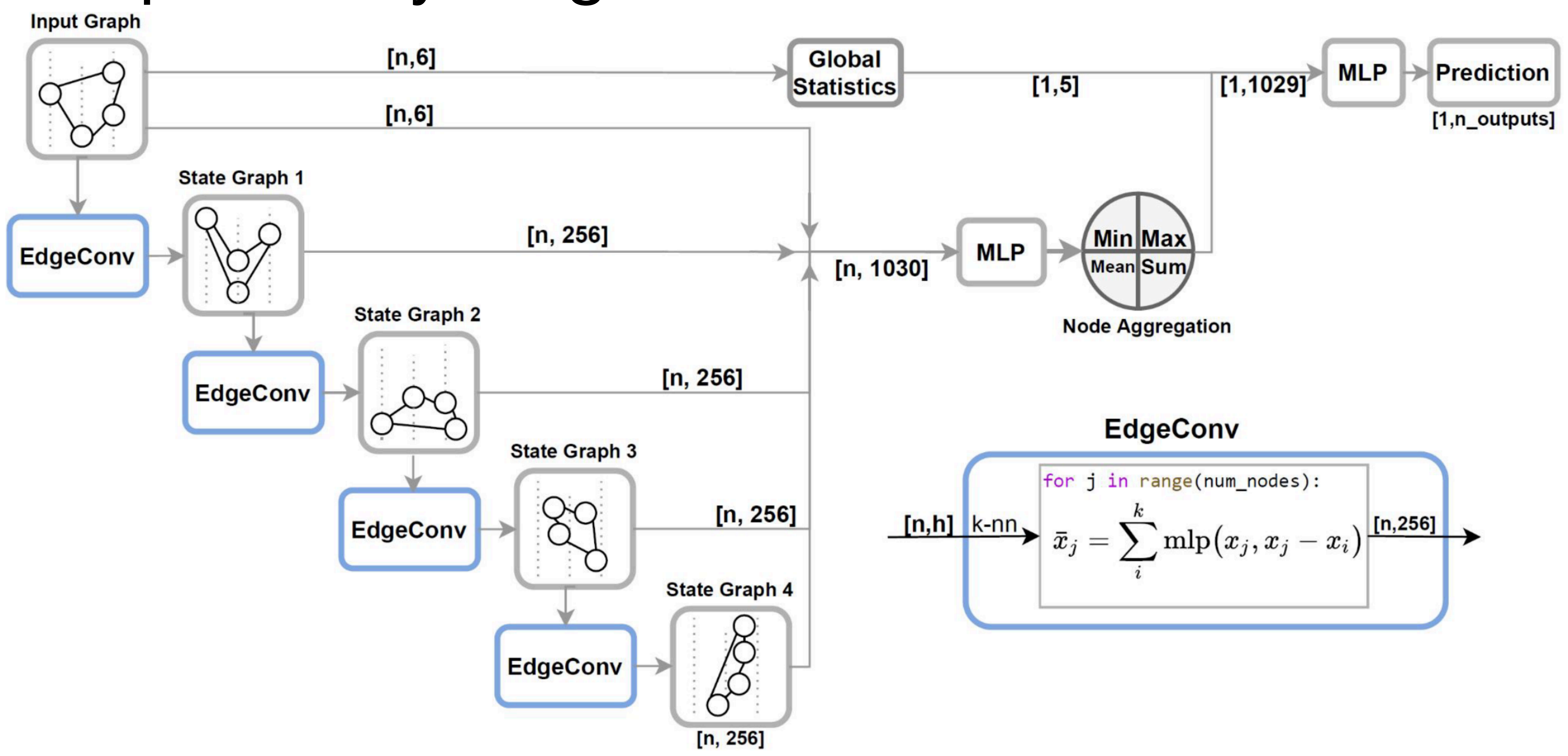


Network Architectures

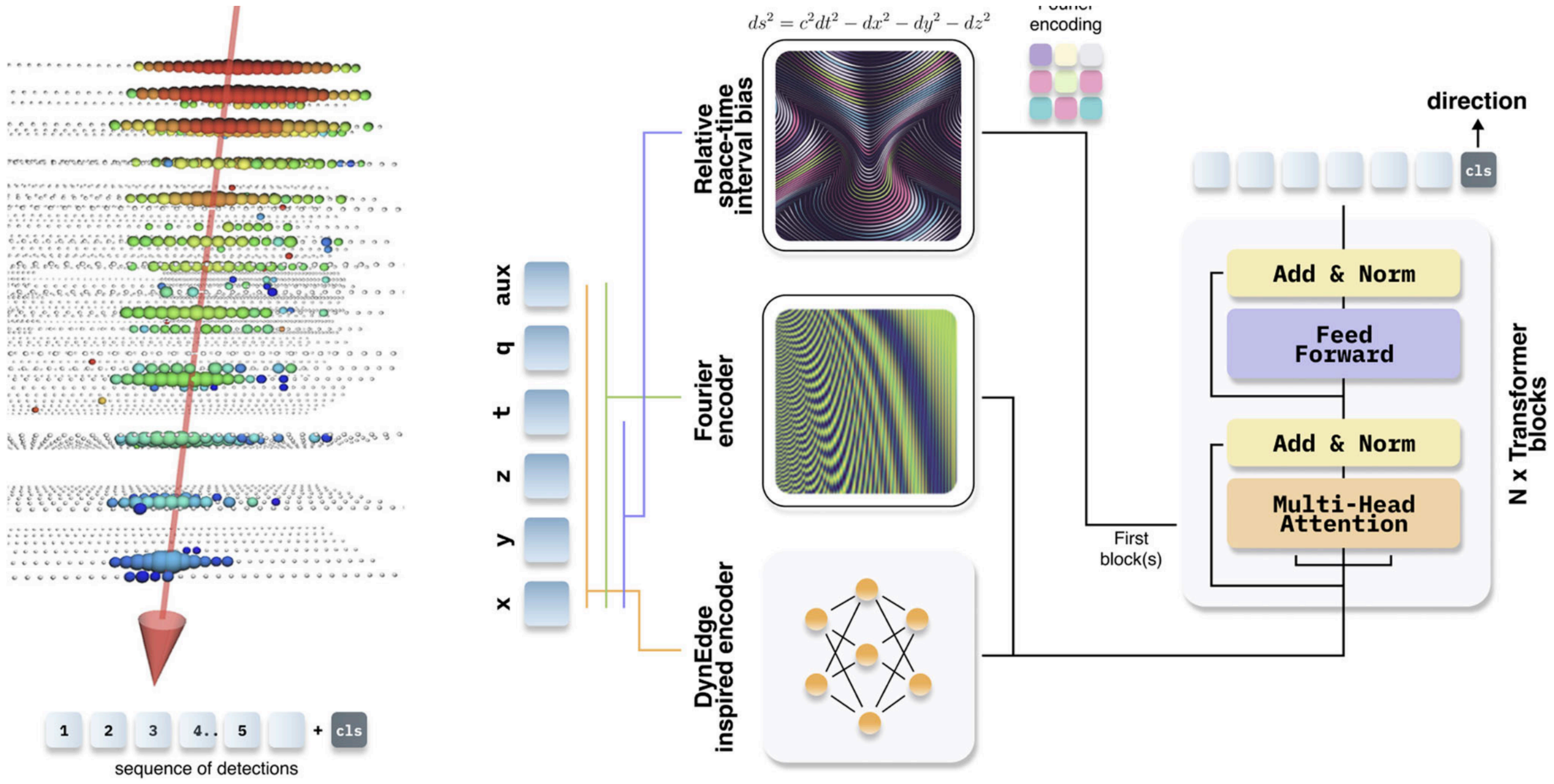
DNNCascade



GraphNeT:dynedge

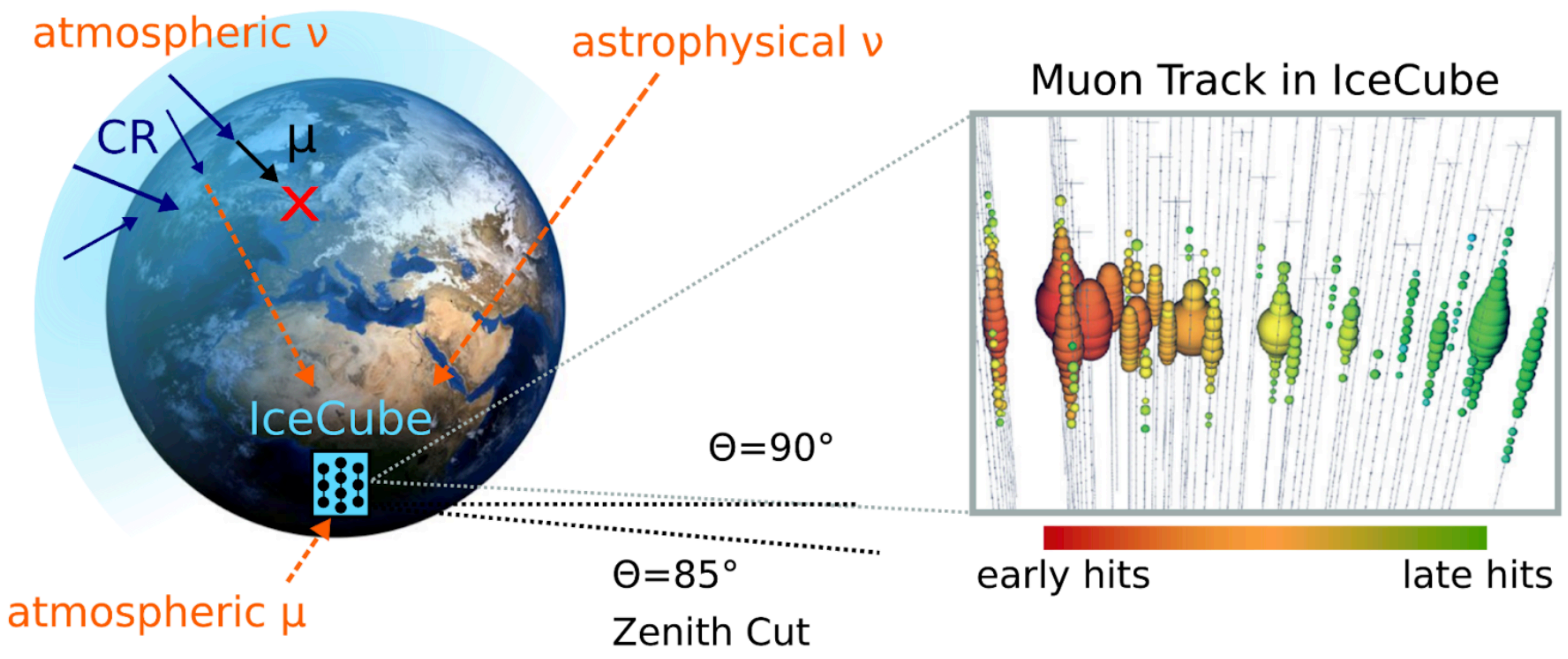


Kaggle Challenge: 2^{nd} solution



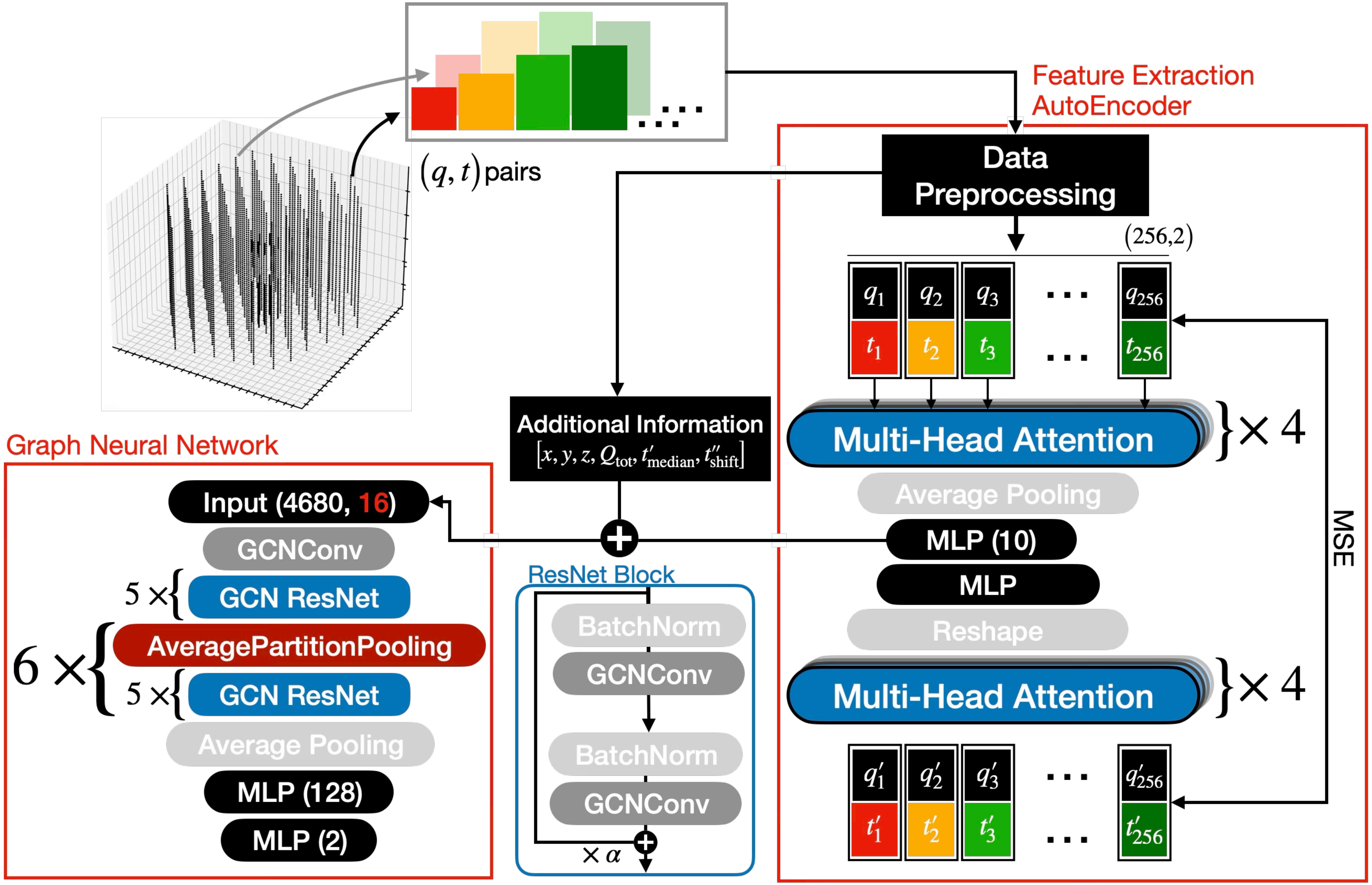
- Different types of network architectures in IceCube (selection)
- All try to work with IceCubes:
  - Irregular detector layout
  - High statistics

What I am working on right now

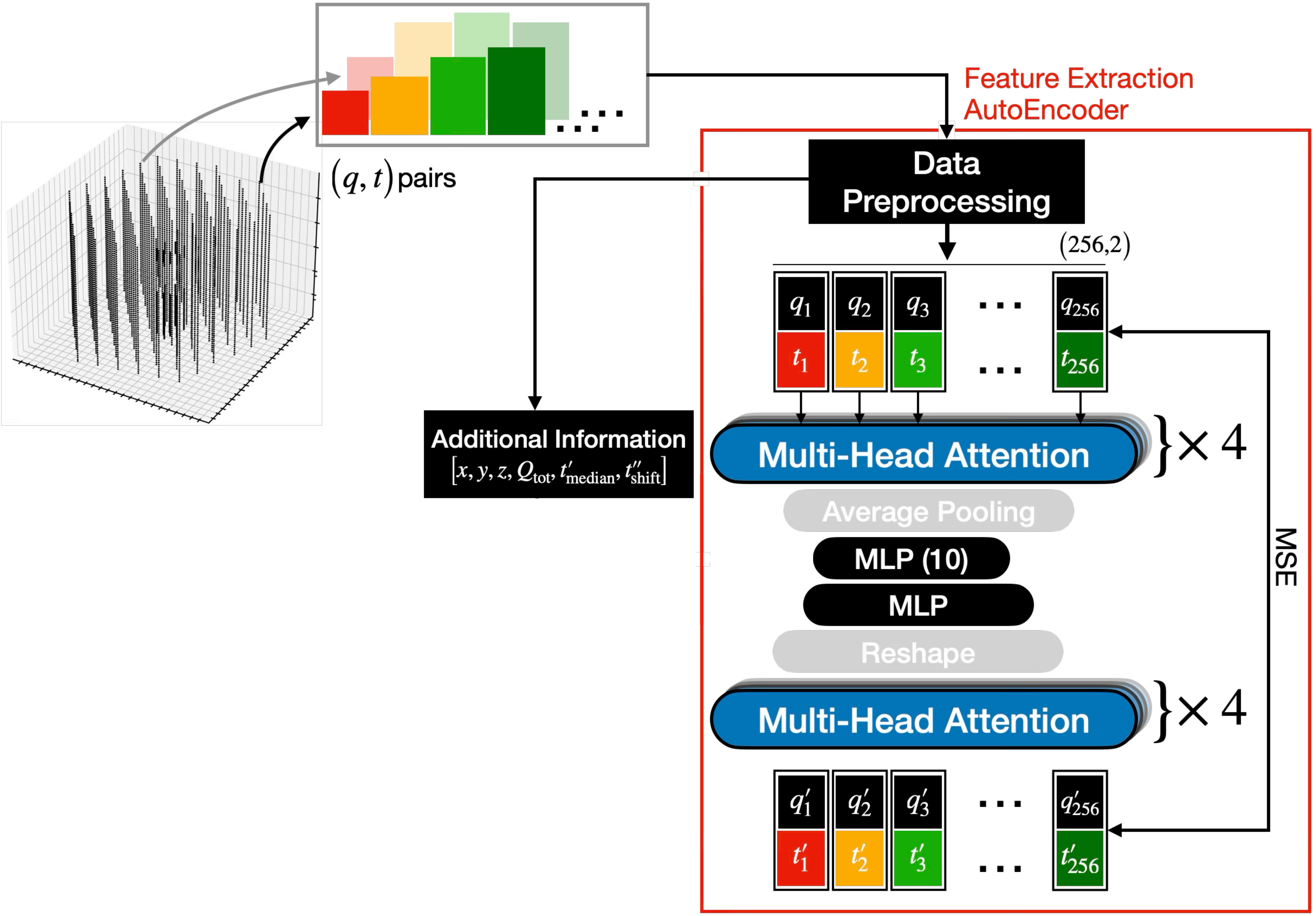


- We are currently working on a new IceCube event selection
- Difference between atm. μ and ν_μ induced μ
- The current implementation works with 11 constructed high level variables (implemented 10 years ago)
- We have developed a graph based reconstruction algorithm for JUNO
- Implement this network for IceCube Selection

Network Architecture

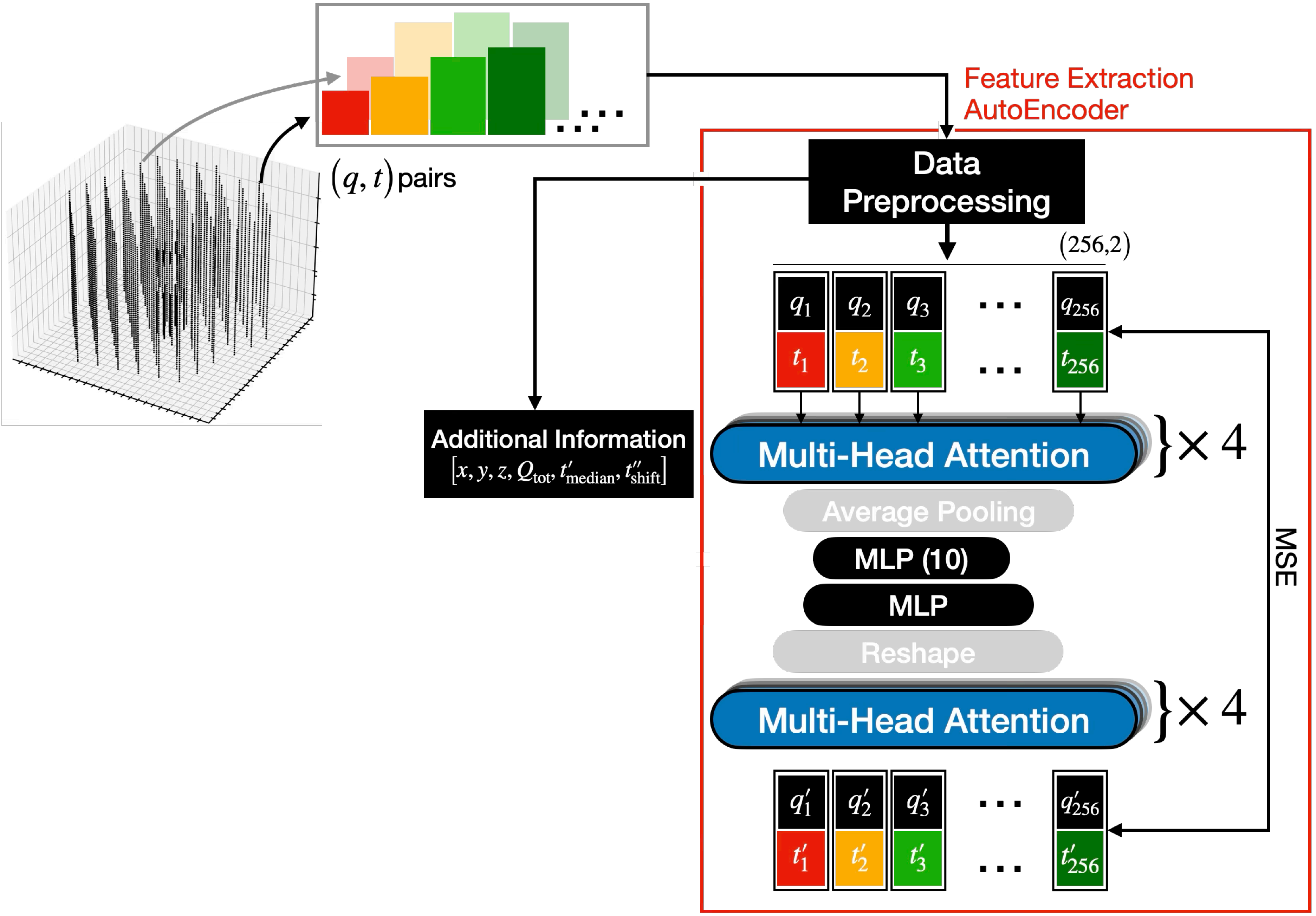


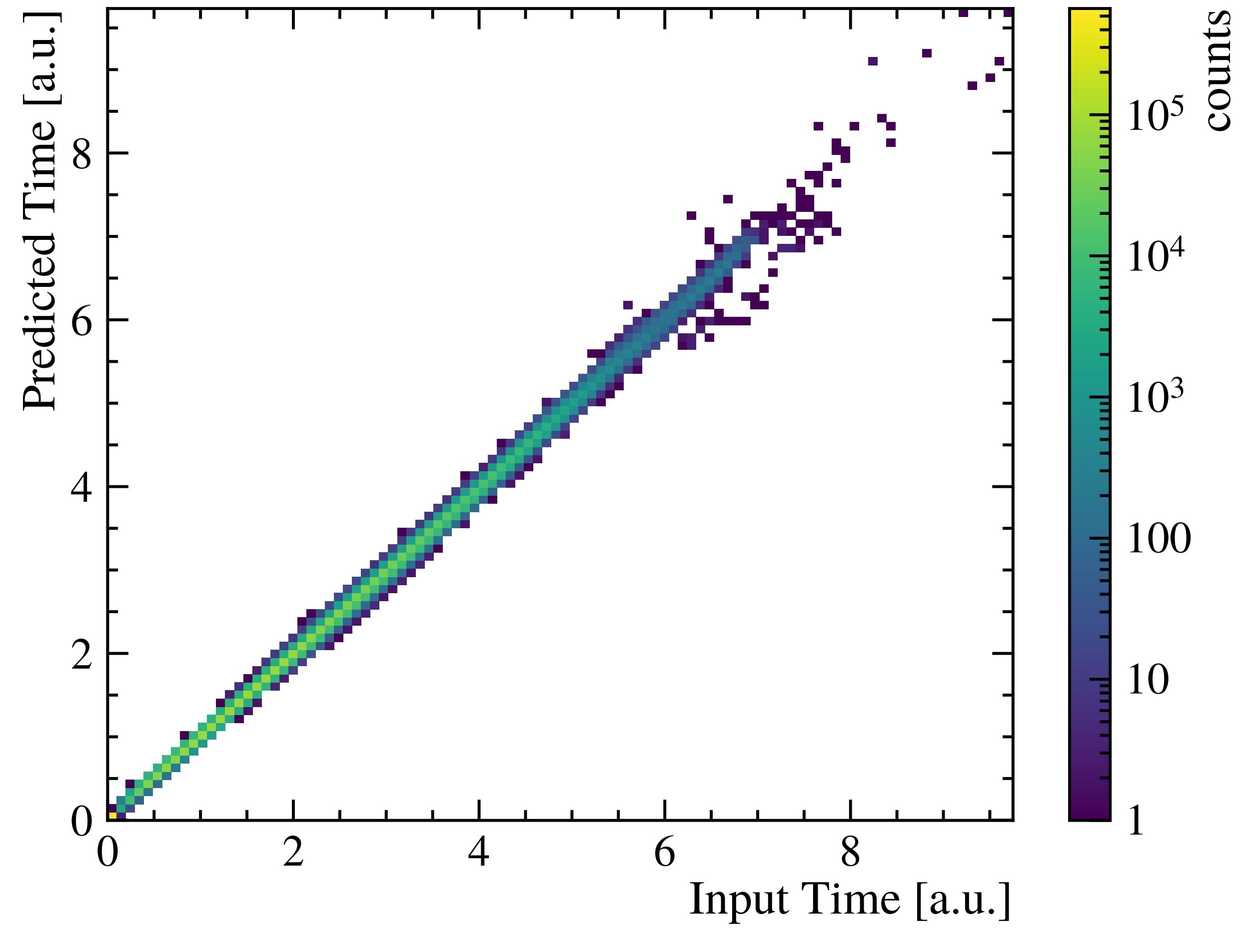
Network Architecture (AutoEncoder)

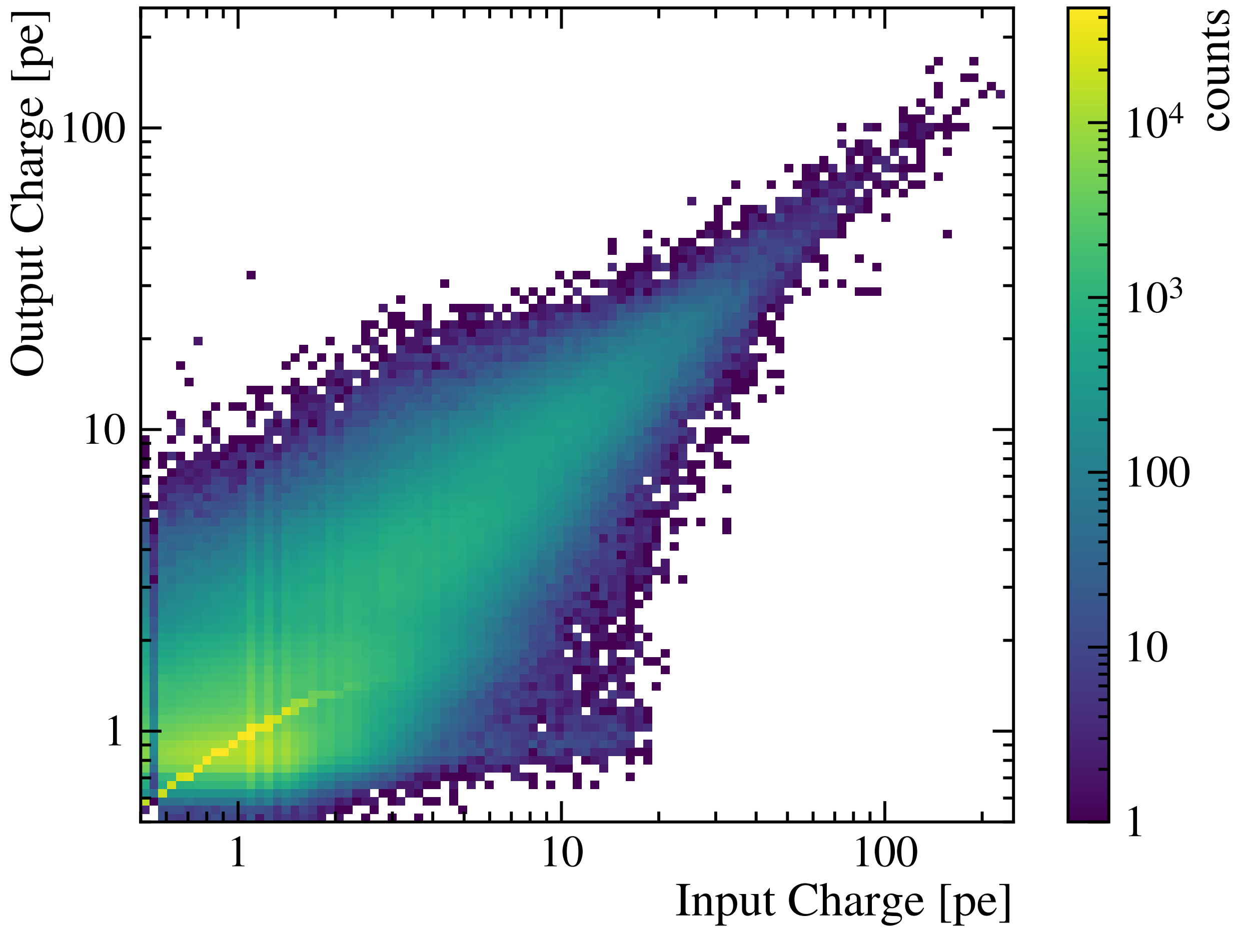


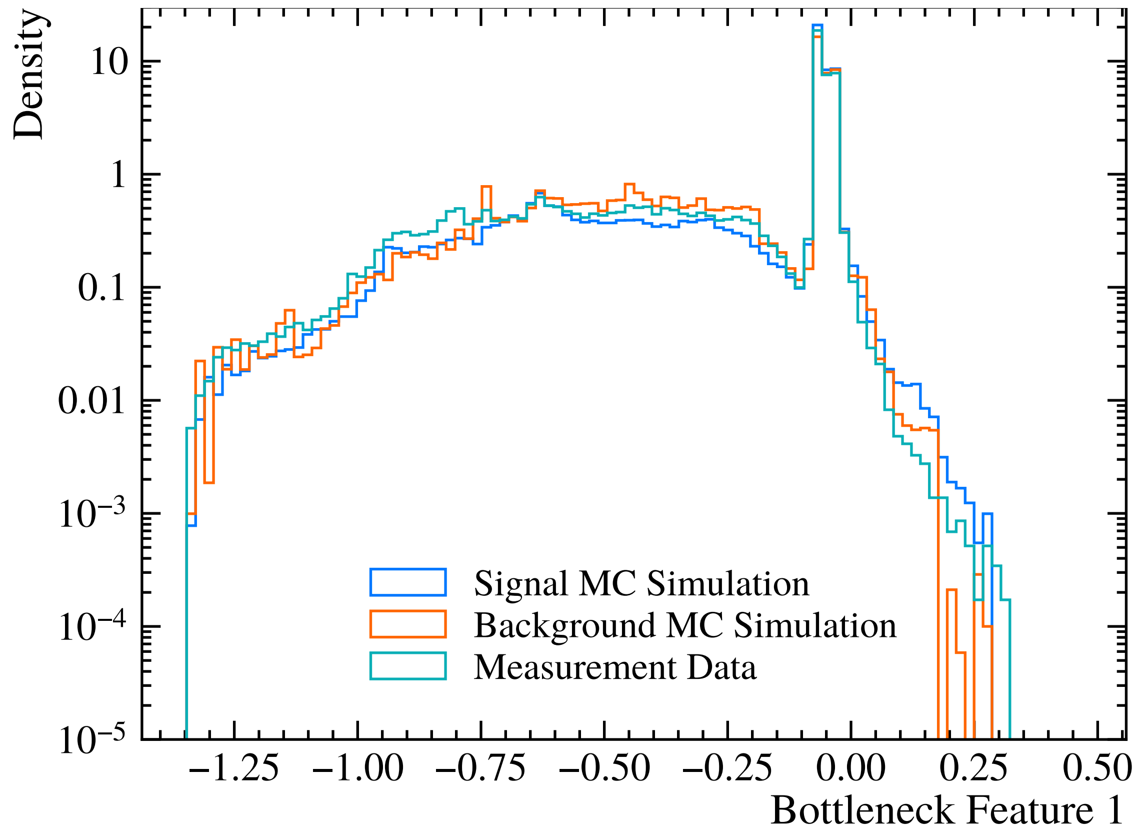
- Reconstructed time and charge pairs (q,t) are preprocessed
  - Dynamic windowing to allocate photon hits in $10\,\mathrm{ns}$ (charged summed, first time hit in window)
  - Divide by standard deviation
  - Time median per DOM is subtracted
  - Signed sqrt of times
  - Correct hits for respective time shift
- Train Transformer based AutoEncoder to learn 10 arbitrary features that describe the per-DOM time series
- Requirement for fast on-the-fly processing
- Possible feedback to write interim results to disk

Network Architecture (AutoEncoder)

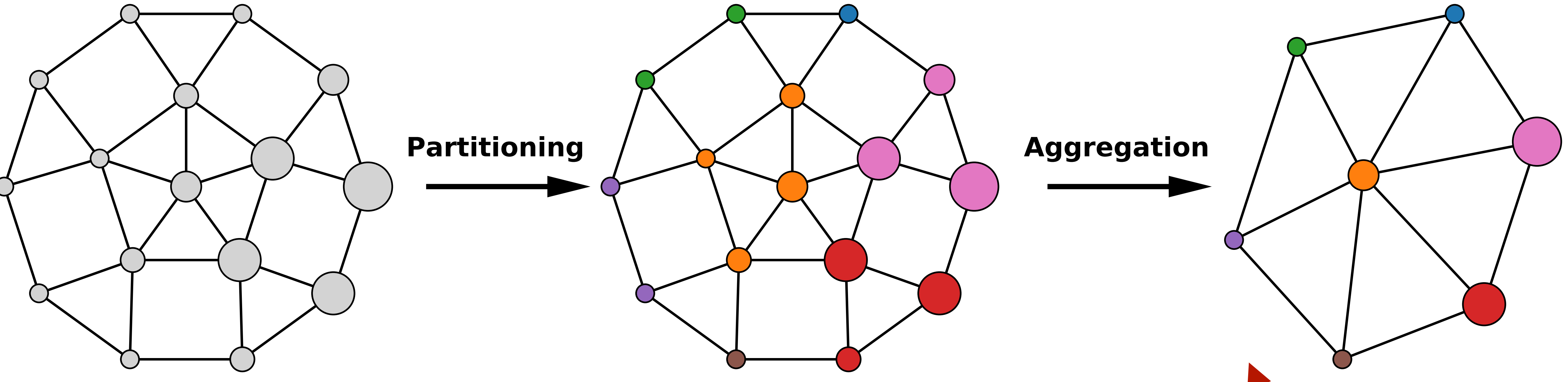


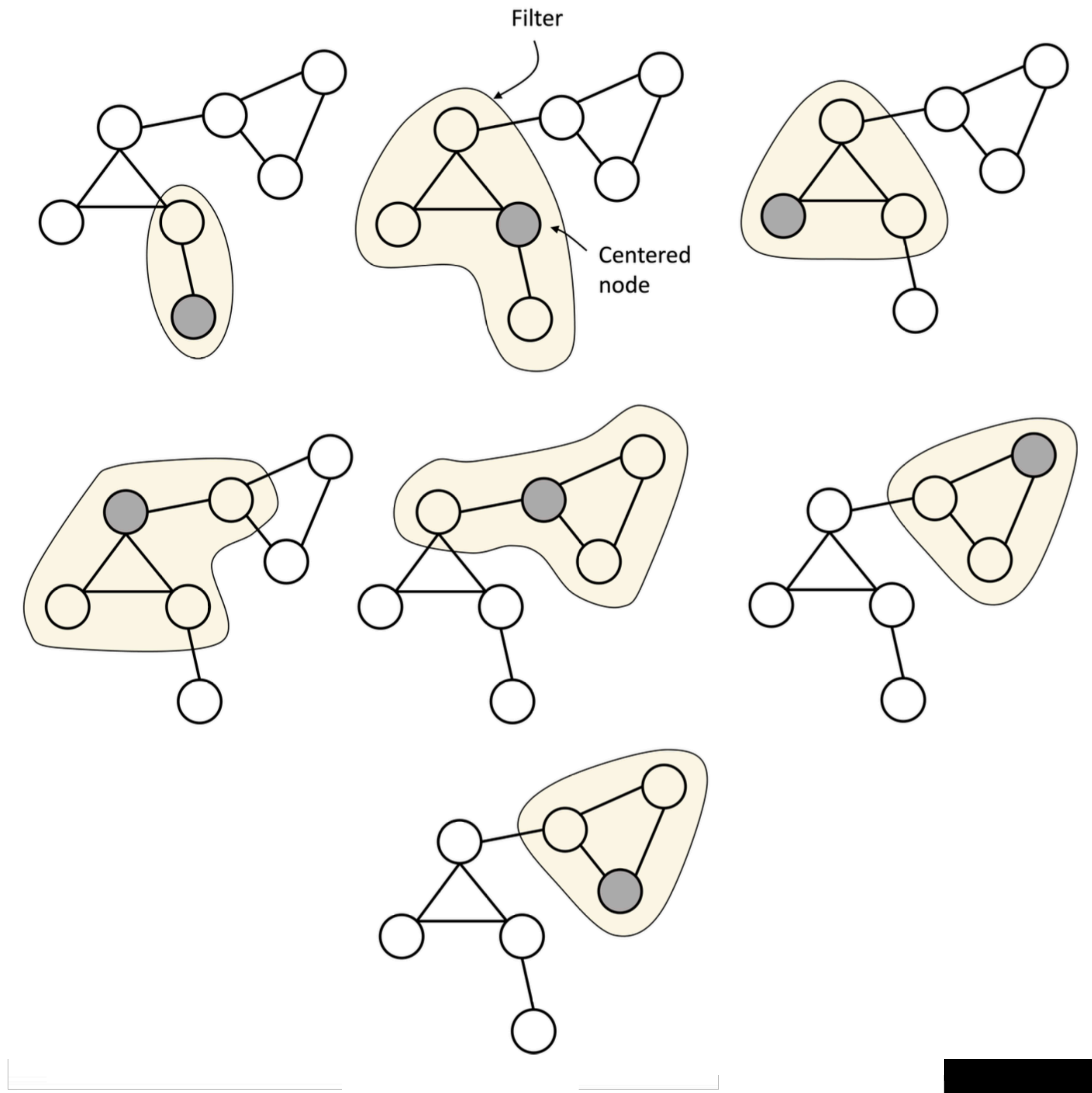




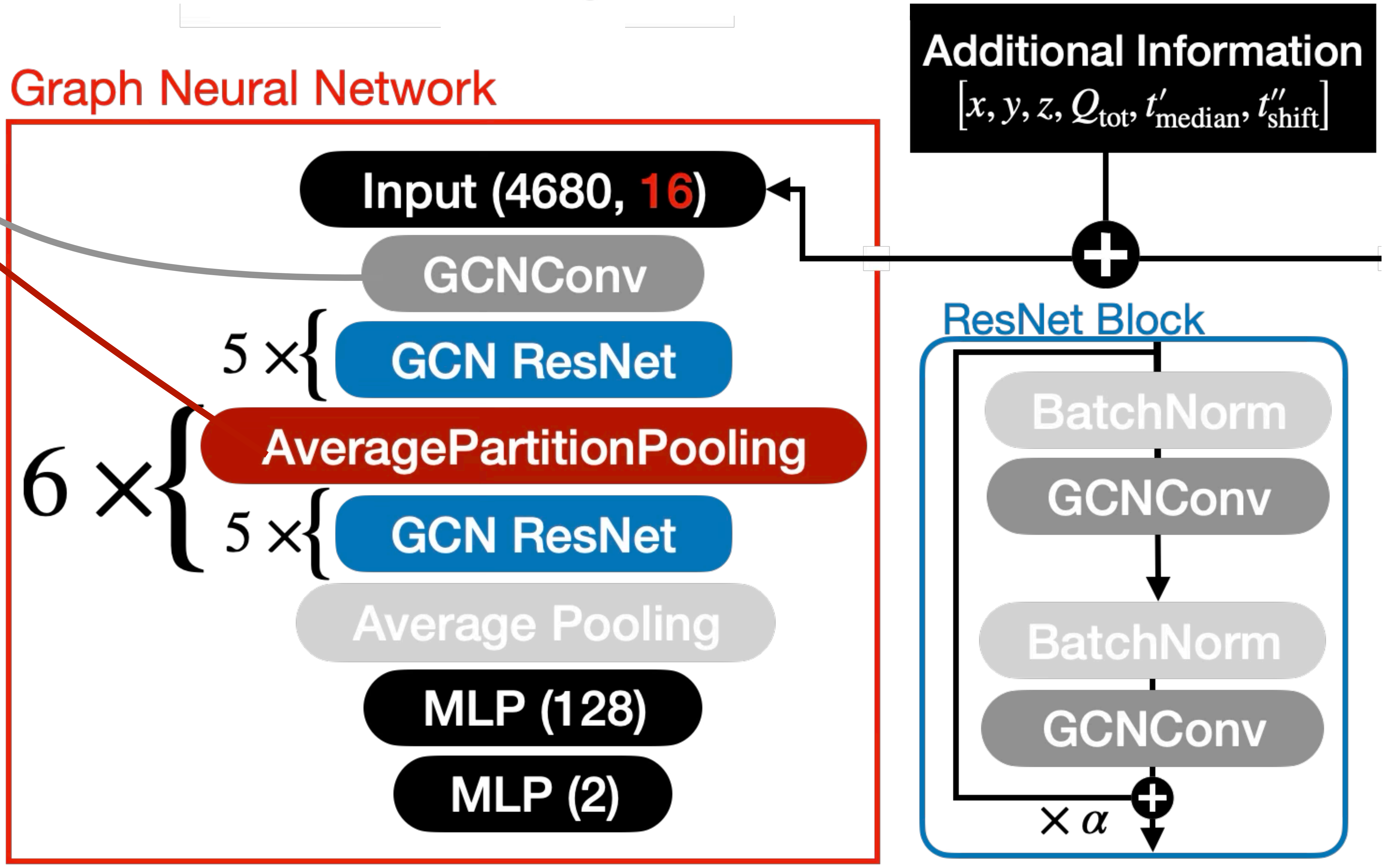


Network Architecture (Graph)

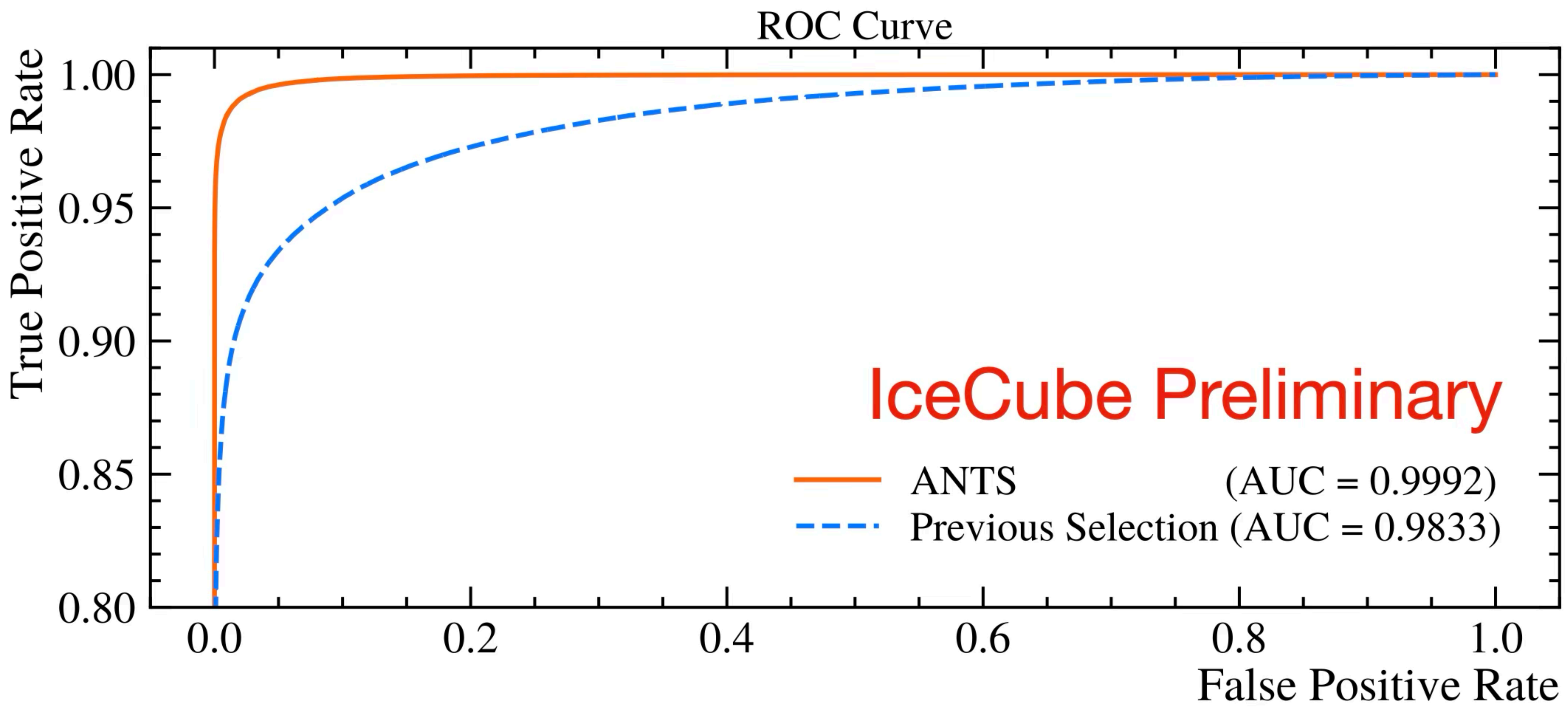


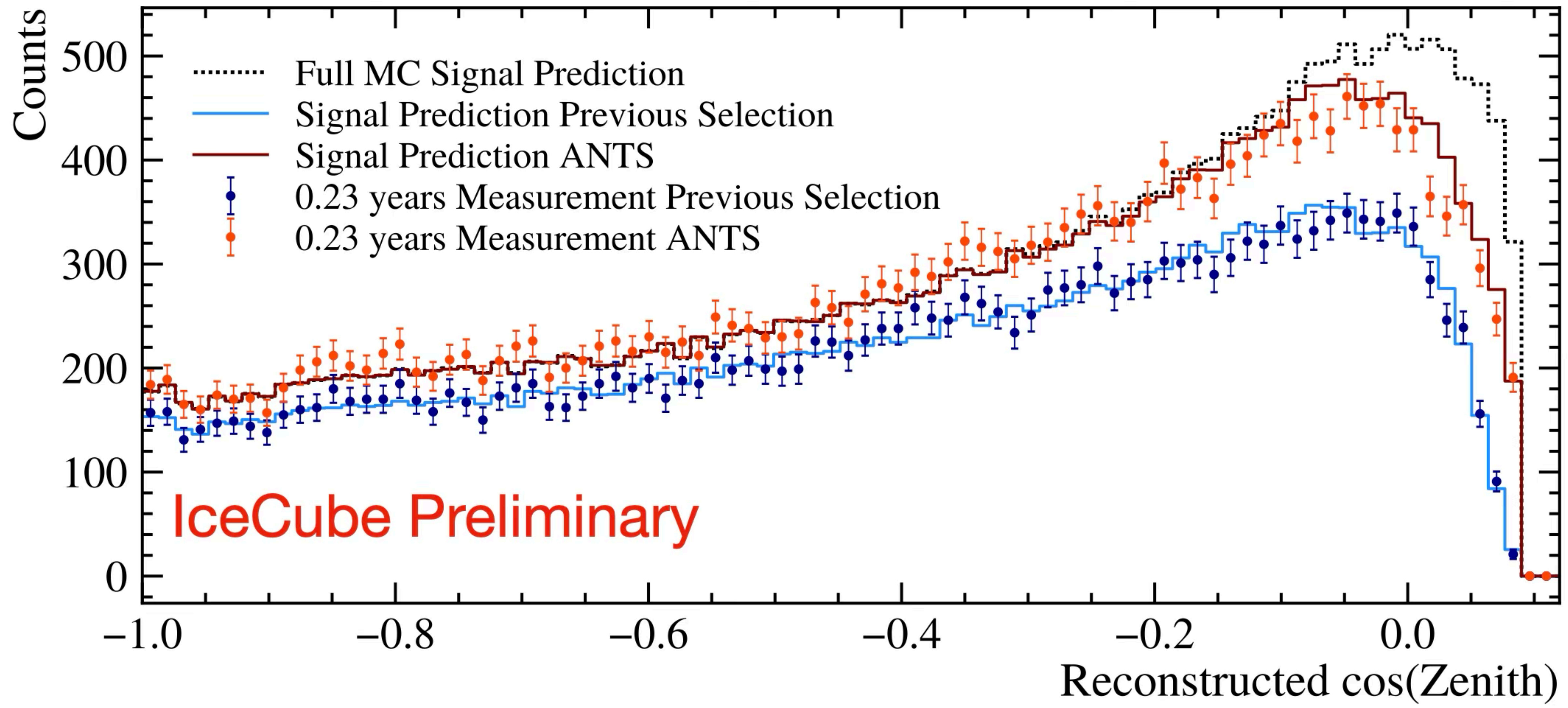


- Different Graph implementation than GraphNet ("spherical harmonics")
- Graph Network inspired by default convolutional neural network
- Own pooling algorithm that aggregates based on node proximity
- ResNet Architecture with ~ 70 GCN layers



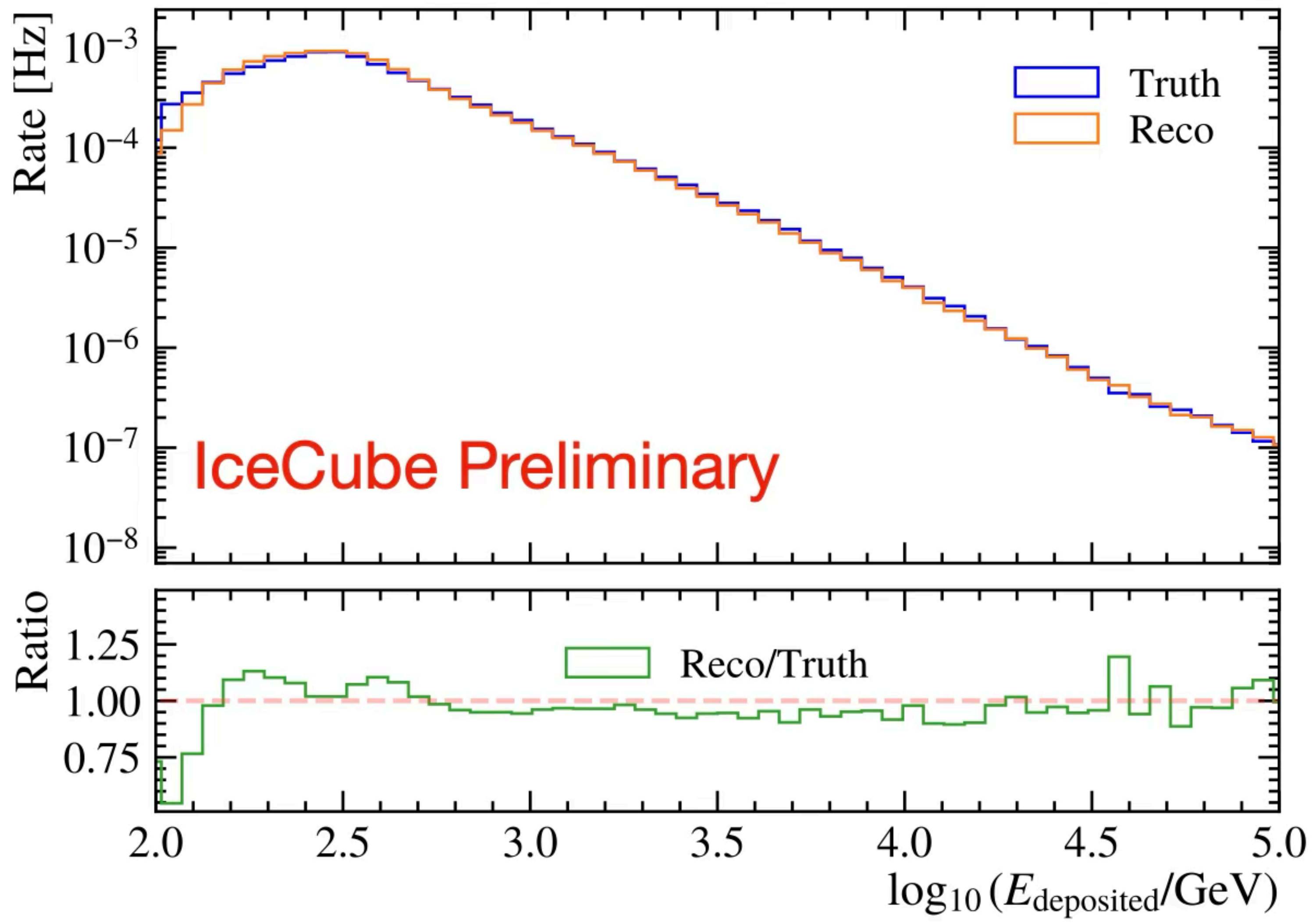
Network Architecture (Output)

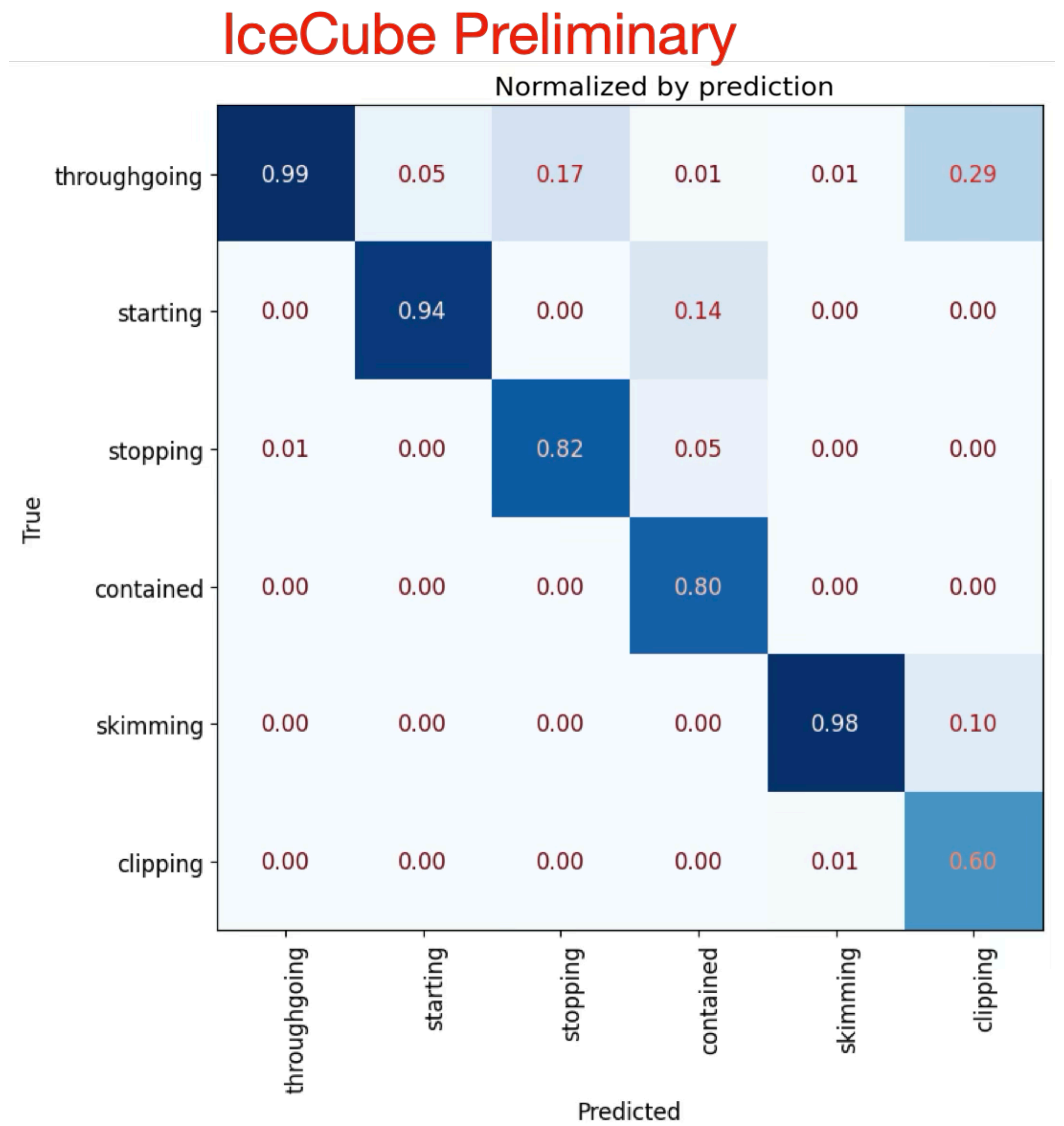




- Strong classifier for neutrino events with excellent Data/MC agreement
- $25\ \%$ statistics improvement
- Other tasks like topology classification, energy- and directional reconstruction work well by easily swapping out th classification head
- Multi-purpose network
- Requirement to swap out the classification head

Network Architecture (Output)

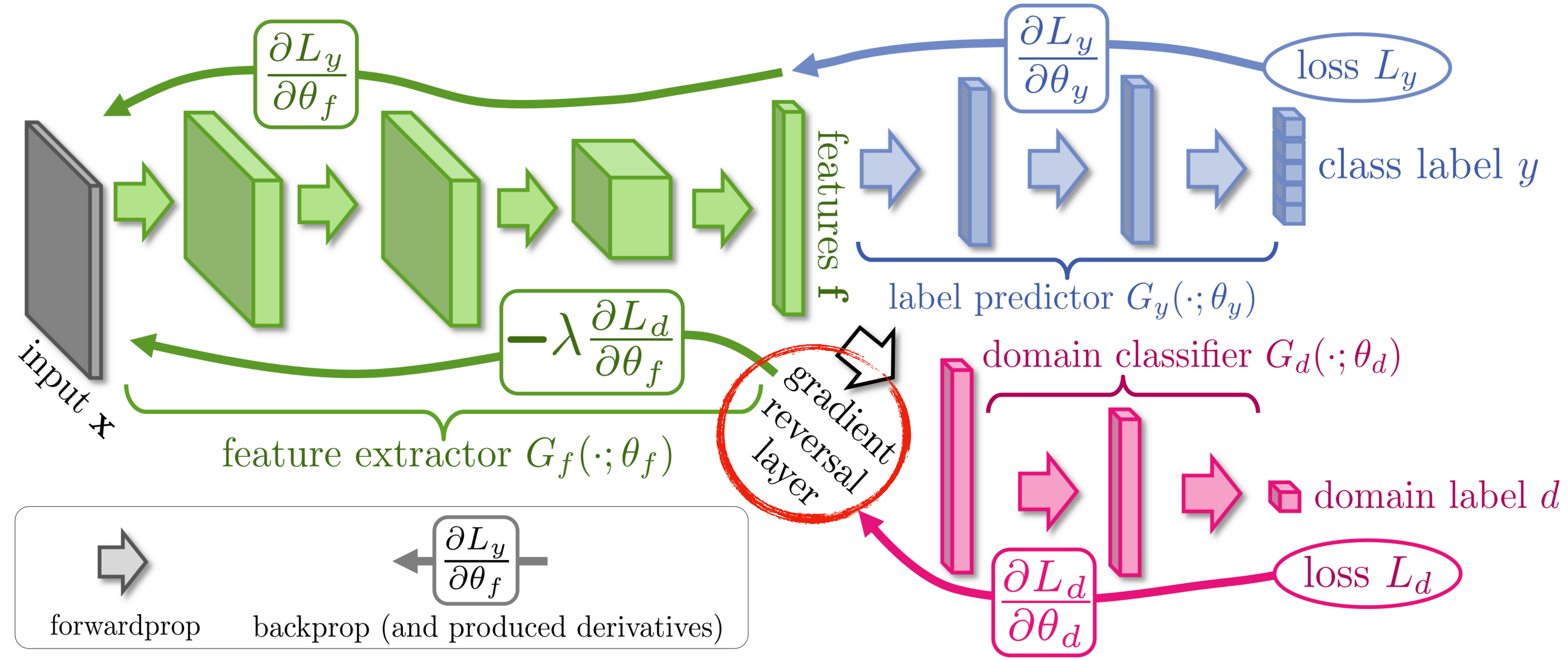


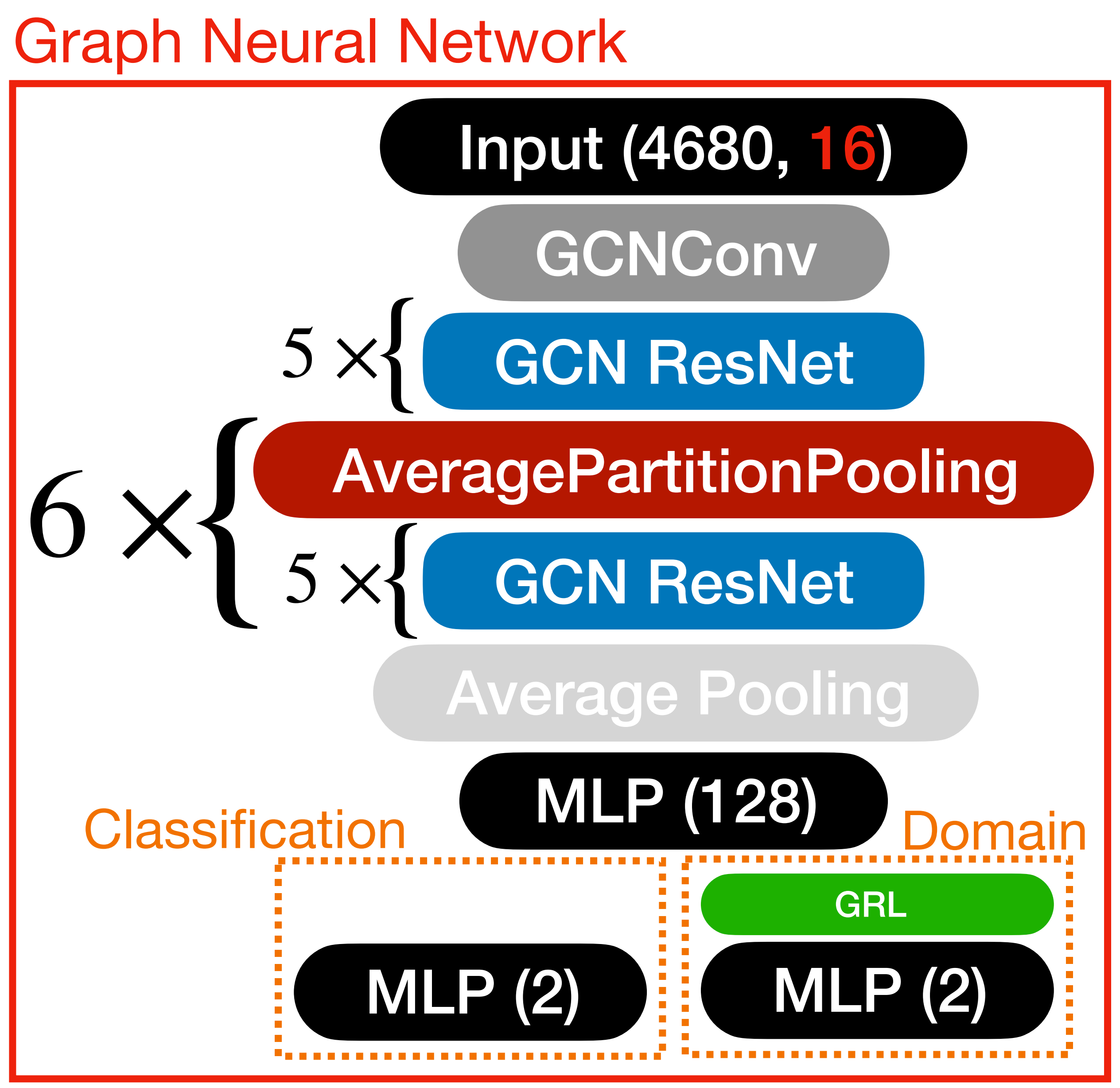


True \ Predicted	throughgoing	starting	stopping	contained	skimming	clipping
throughgoing	0.99	0.05	0.17	0.01	0.01	0.29
starting	0.00	0.94	0.00	0.14	0.00	0.00
stopping	0.01	0.00	0.82	0.05	0.00	0.00
contained	0.00	0.00	0.00	0.80	0.00	0.00
skimming	0.00	0.00	0.00	0.00	0.98	0.10
clipping	0.00	0.00	0.00	0.00	0.01	0.60

- Strong classifier for neutrino events with excellent Data/MC agreement
- $25\ \%$ statistics improvement
- Other tasks like topology classification, energy- and directional reconstruction work well by easily swapping out th classification head
- Multi-purpose network
- Requirement to swap out the classification head

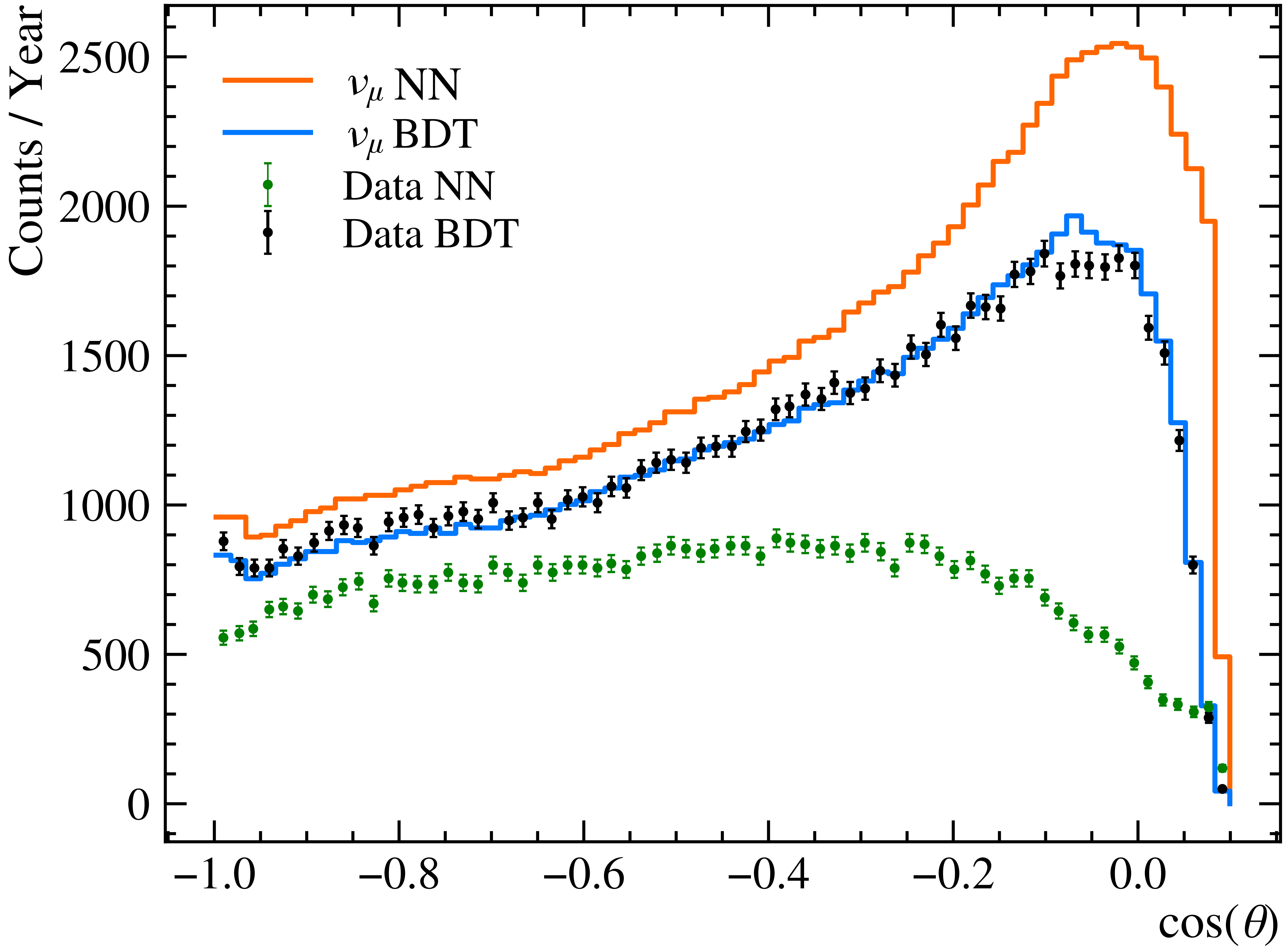
Domain Adaptation

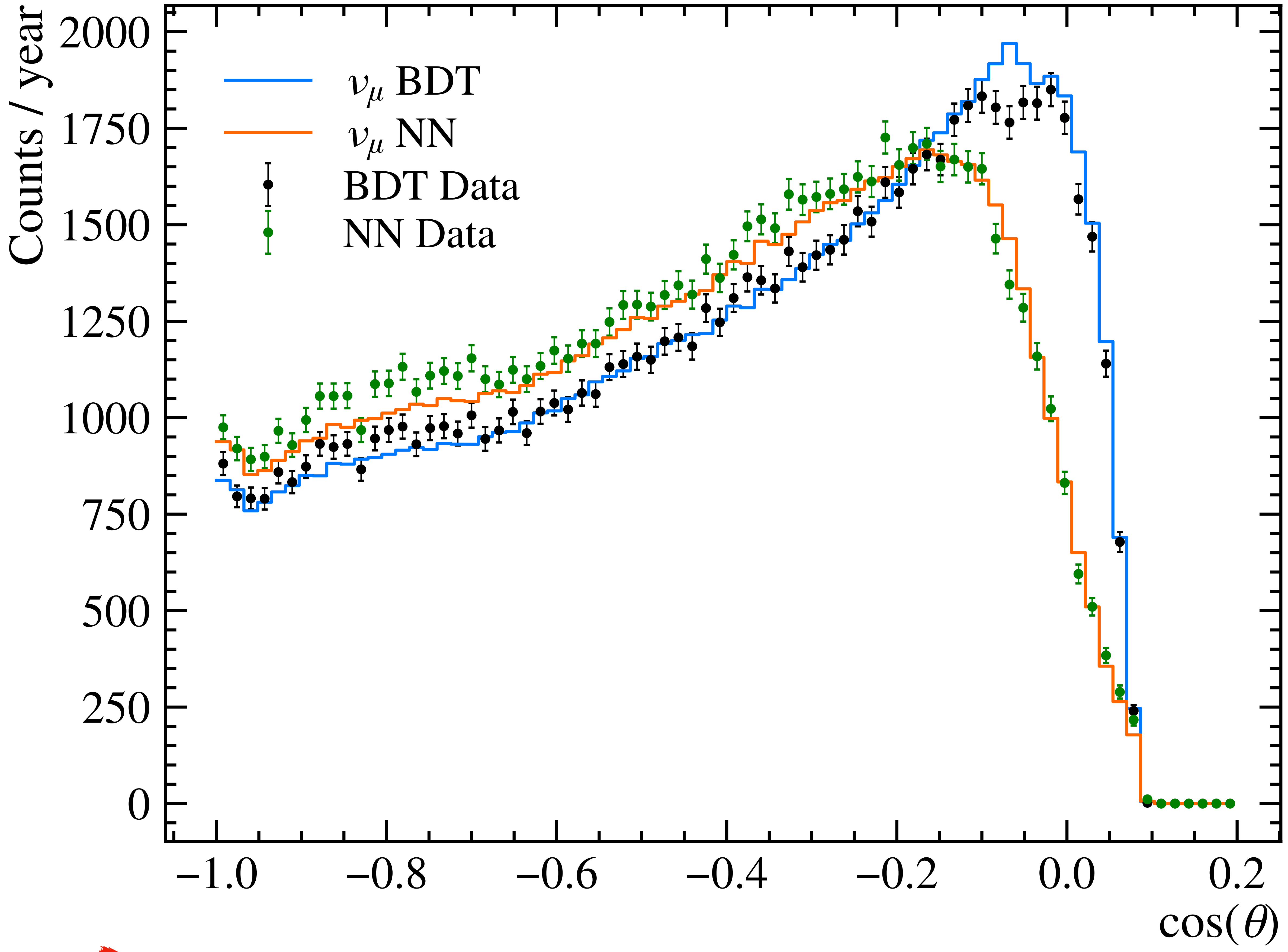




- Initial implementations of the network had terrible DC/MC agreement
- Solution was to use Unsupervised Domain Adaptation by Backpropagation
- Requirement to be able to train with multiple outputs that are trained differently

Domain Adaptation





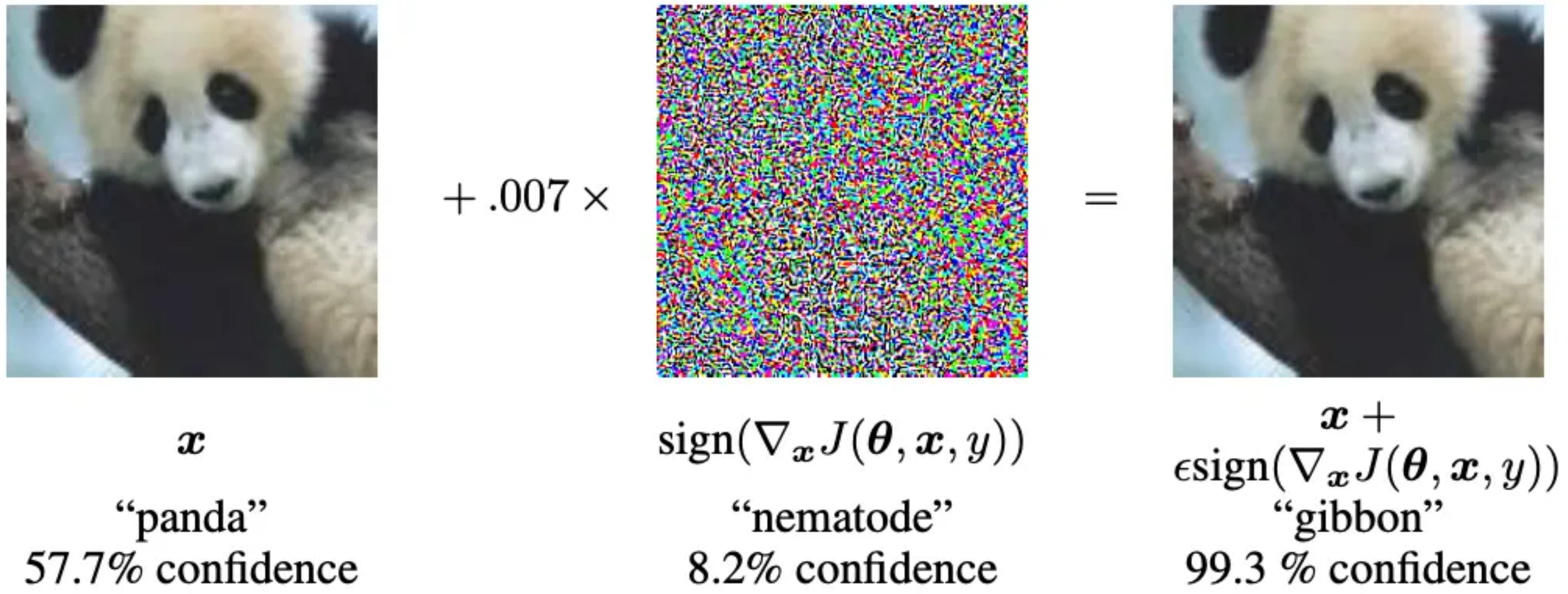
Domain Adaptation

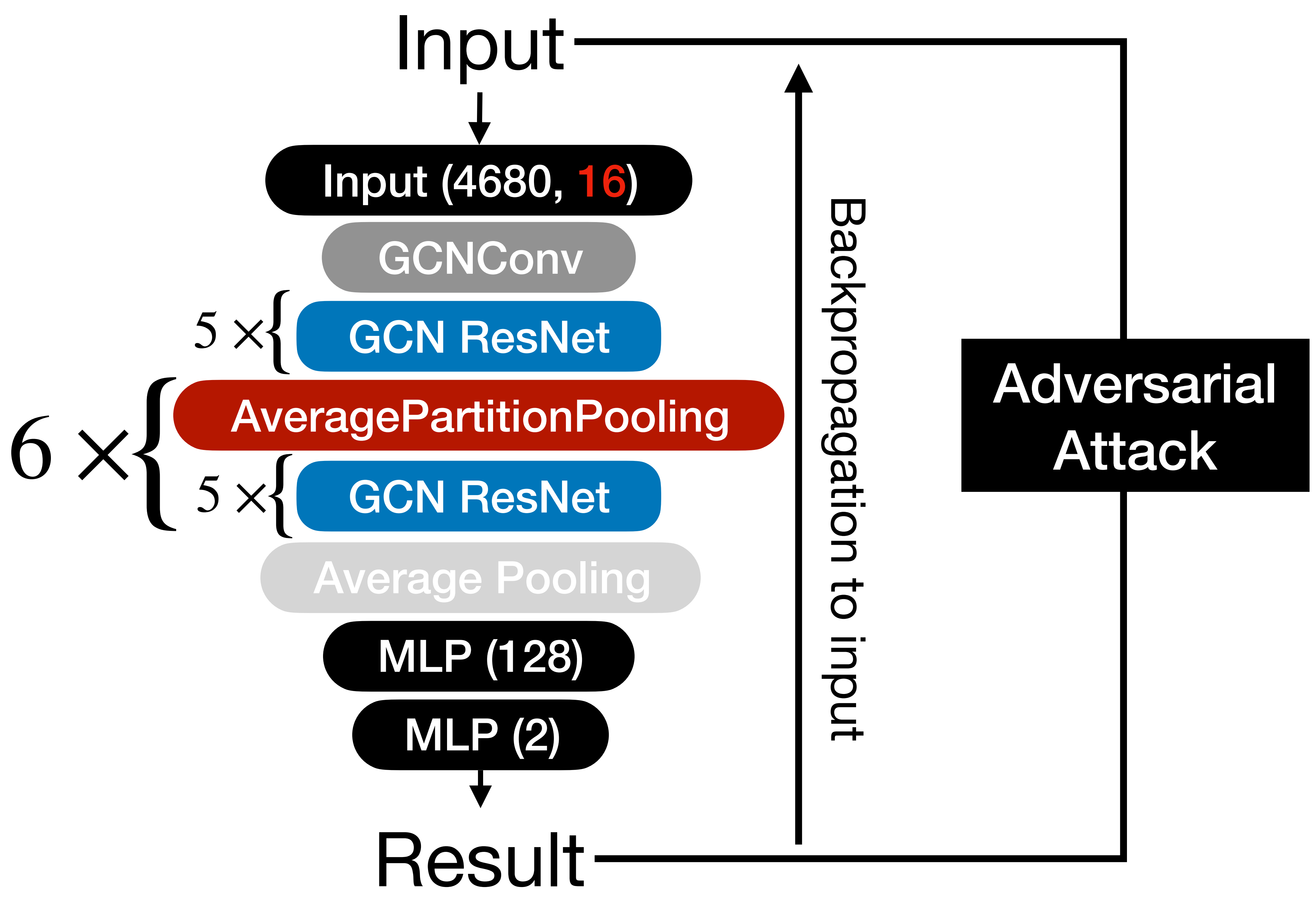
- Better MC / Data agreement
- $\sim 5\,\%$ increase in event statistics, loss at the horizon

Adversarial Training

- Disturb network input to change neural network output
- Networks are susceptible to minuscule changes, that are network specific
- Adversarial attacks can be used for data augmentation during training, **maximally disturbed inputs!**
- This leads to “flatter” loss manifolds, that are more robust against (targeted) noise
- Different methods for attacks!
- Requirement to access model input from output and iteratively apply

Fast Gradient Sign Method



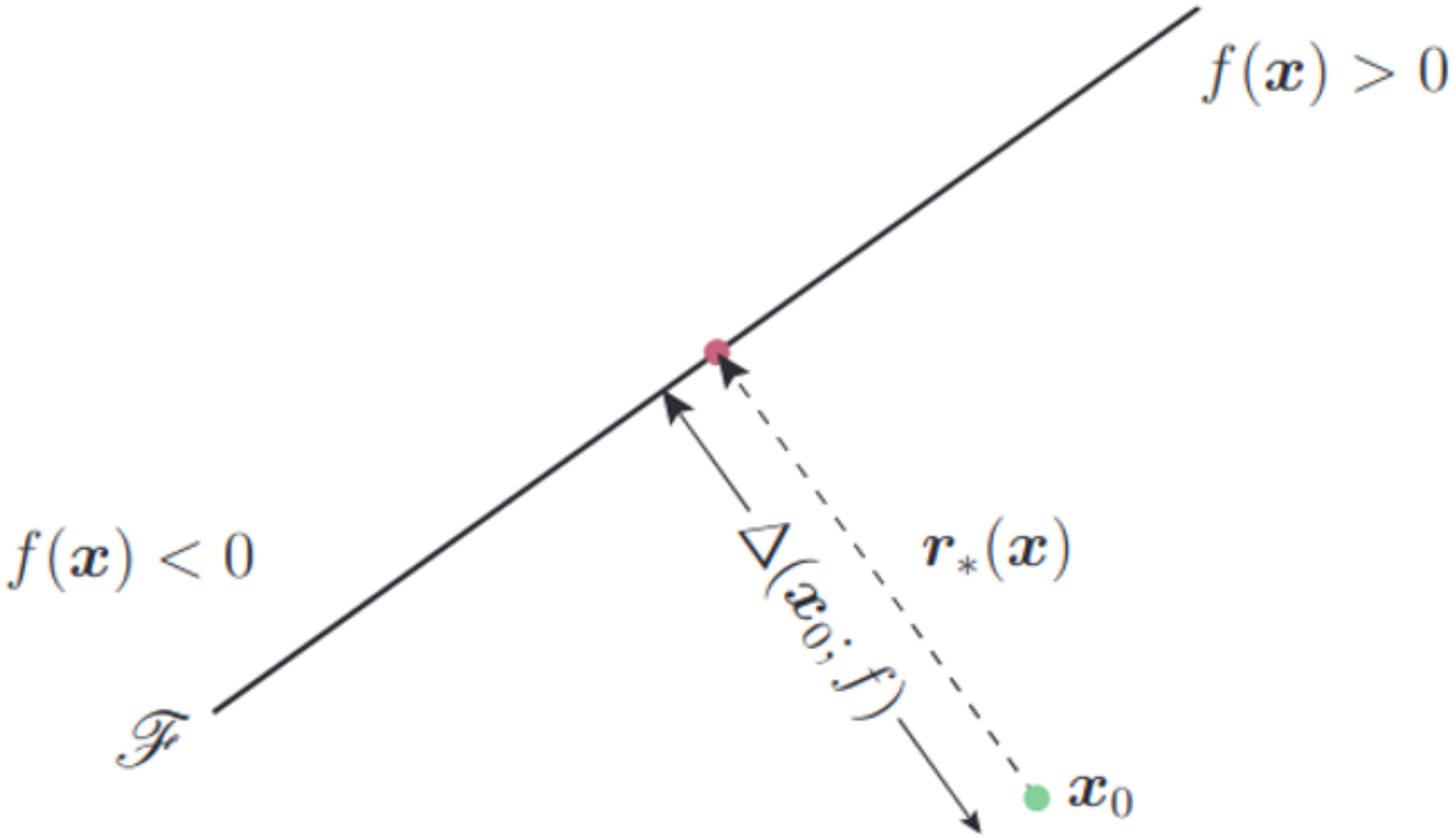


Adversarial Attacks (Deep Fool)

- Projection of the image in the input space to a decision boundary
- Assuming a linear network the solution is:

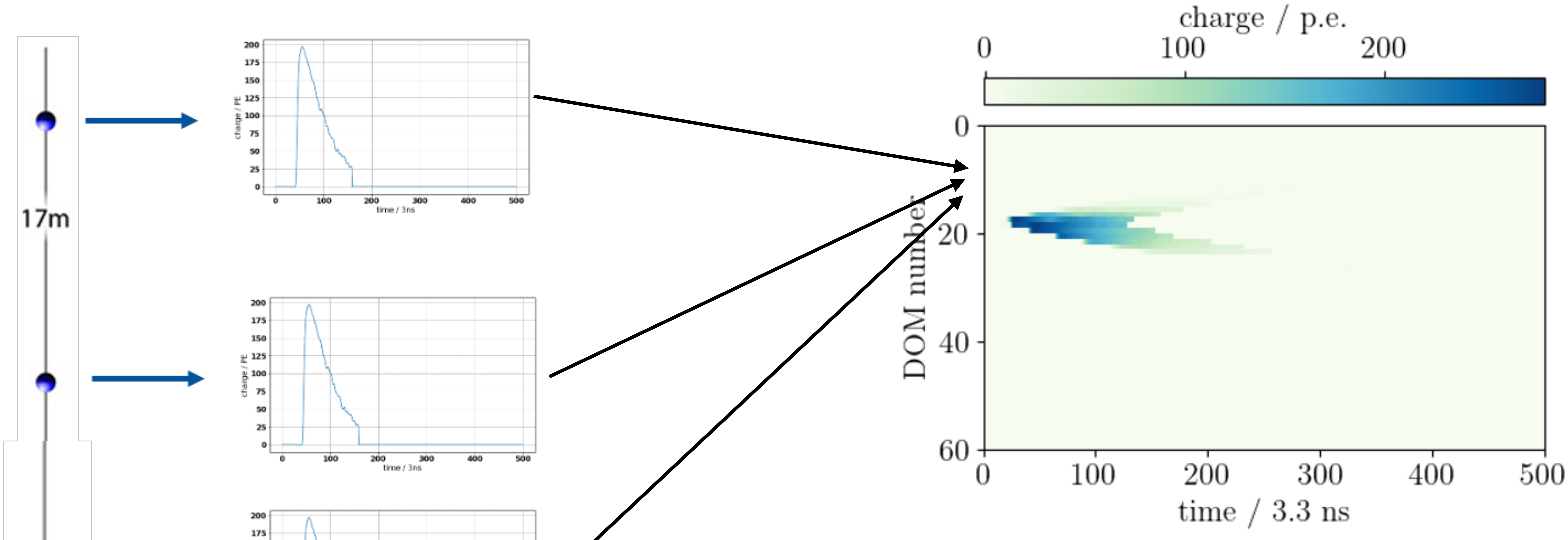
$$\boldsymbol{r}_i \leftarrow -\frac{f(\boldsymbol{x}_i)}{\|\nabla f(\boldsymbol{x}_i)\|_2^2} \nabla f(\boldsymbol{x}_i)$$

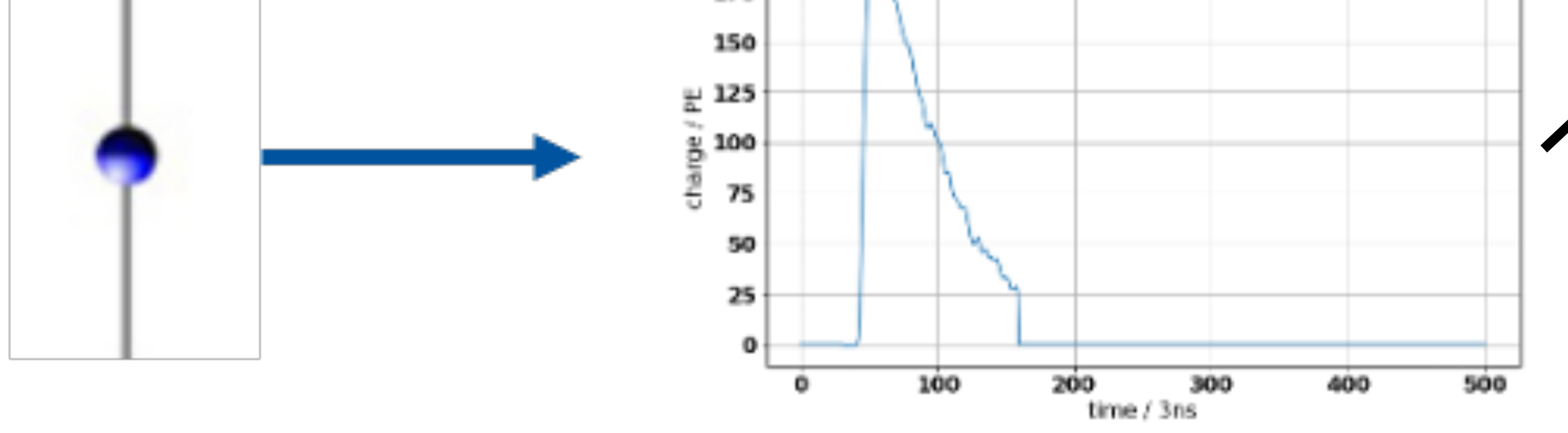
- In real use cases the networks are not linear
- Iterative application of the above step until boundary is reached



arXiv:1511.04599

Adversarial Attacks (Deep Fool)



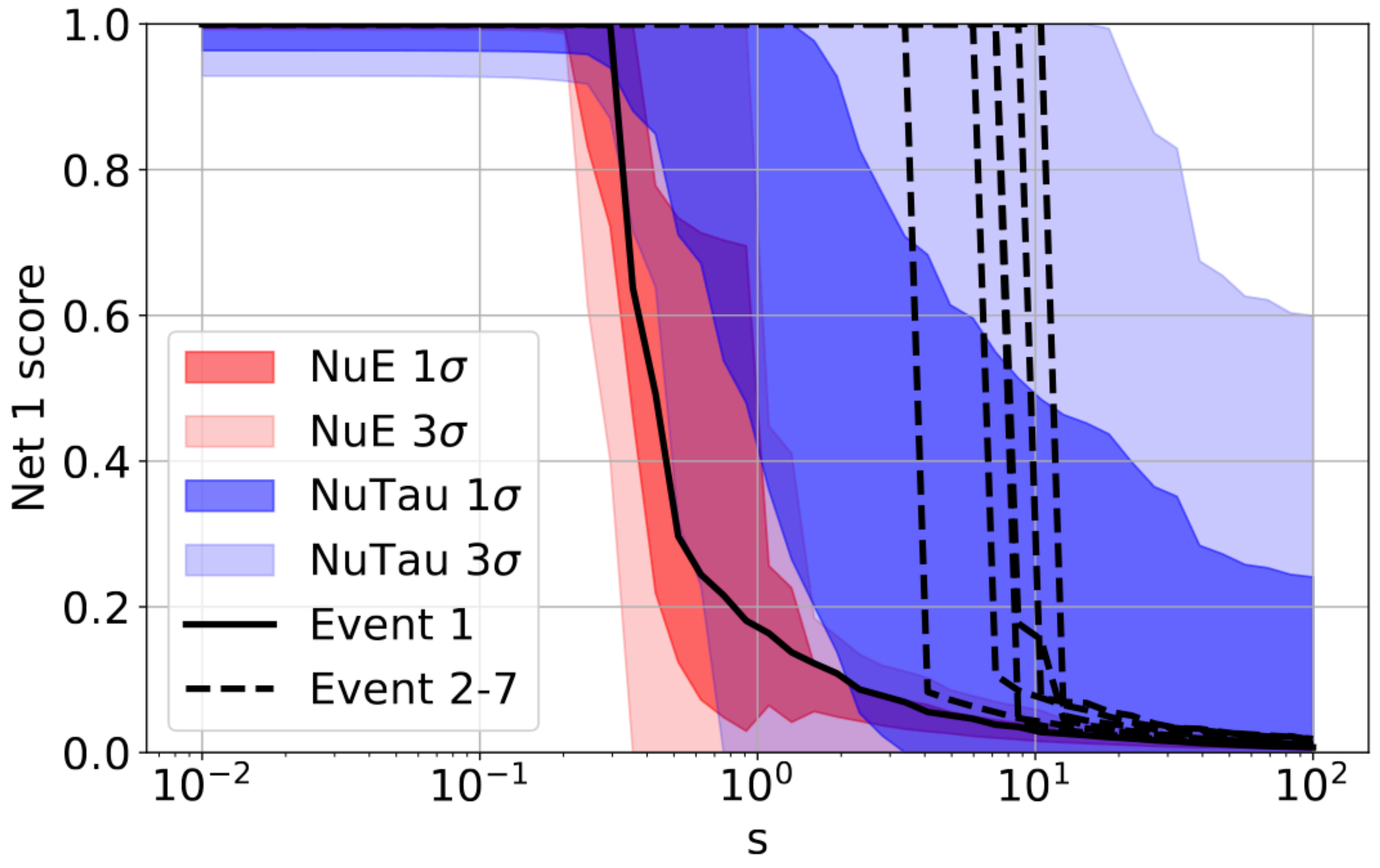


- This has already been successfully applied to the ν_τ analysis in IceCube in the “Double Bang” search

Adversarial Attacks (Deep Fool)

- Apply DeepFool to a simulated ν_e event to make it look like a ν_τ
- This introduces unphysical negative pixel values

Adversarial Attacks (MiniFool)



Average Pixel deviation

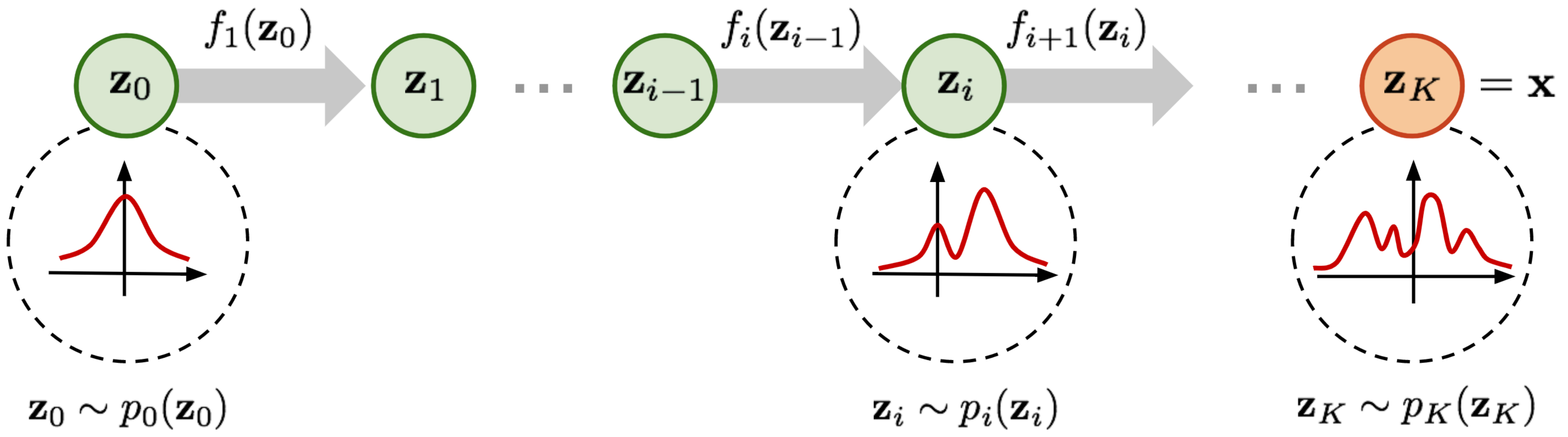
$$f = \frac{1}{N} \sum_{\text{pixel}} \left(\frac{x_i - x^{\text{adv}_i}}{\sigma_i} \right)^2 + (F(x_{\text{adv}} - g)^2$$

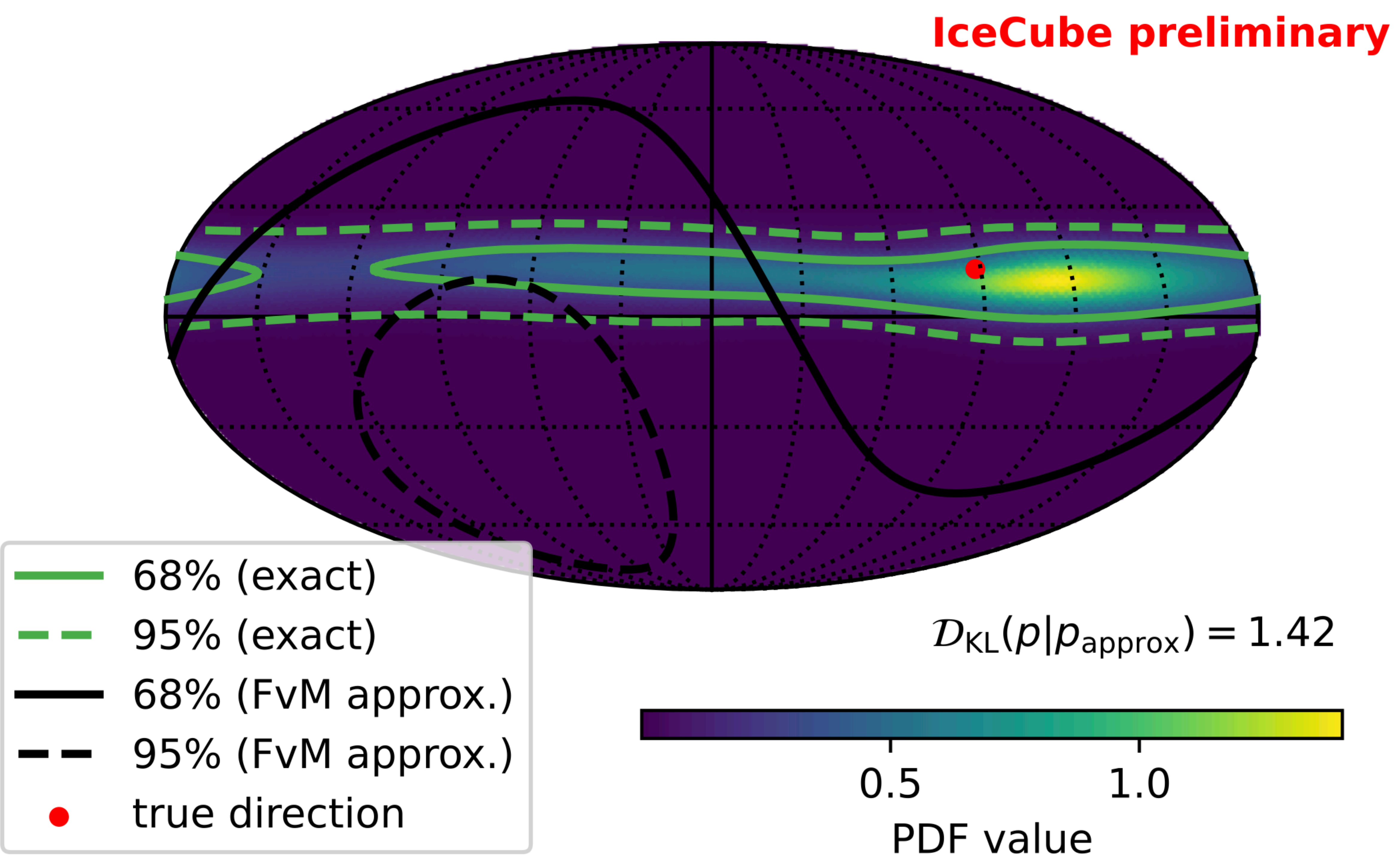
Score of attacked image

Goal Score

- Self developed method (MiniFool) includes physical constraints (also external)
- Wrongly classified events seem to be more susceptible to adversarial attacks
- 7 found ν_τ events, one seems to be more agreeable with background, within expectation
- Publication is on the way!

Normalizing Flows





- Sometimes target distributions do not properly describe the event distribution and uncertainty (MSE ≡ Gaussian)
- Normalizing flows approximate the true Outputs Posterior distribution
- “Normal” Networks can be used to condition flows
- Requirement is to train multiple subsequent models in the same update step

Formal Requirements

- To be truly a universal tool, allow for other neural network implementations to interface (i.e. TensorFlow, JAX)
- Allow for truly dynamic training that incorporates multiple outputs / training schemes
- Reproducibility: Trained networks must be reproducible without the need to retrain the model (must for for the foreseeable future)
- Better interface with IceTray. Especially low level reconstruction that are performed early in the processing chain must be able to write output to i3 frames

Backup

Unsupervised Domain Adaptation by Backpropagation

Figure 2. Examples of domain pairs used in the experiments. See Section 4.1 for details.

METHOD (SOURCE / TARGET)	MNIST / MNIST-M	SYN NUMBERS / SVHN	SVHN / MNIST	SYN SIGNS / GTSRB
SOURCE ONLY	.5749	.8665	.5919	.7400
SA (FERNANDO ET AL., 2013)	.6078 (7.9%)	.8672 (1.3%)	.6157 (5.9%)	.7635 (9.1%)
PROPOSED APPROACH	**.8149** (57.9%)	**.9048** (66.1%)	**.7107** (29.3%)	**.8866** (56.7%)
TRAIN ON TARGET	.9891	.9244	.9951	.9987

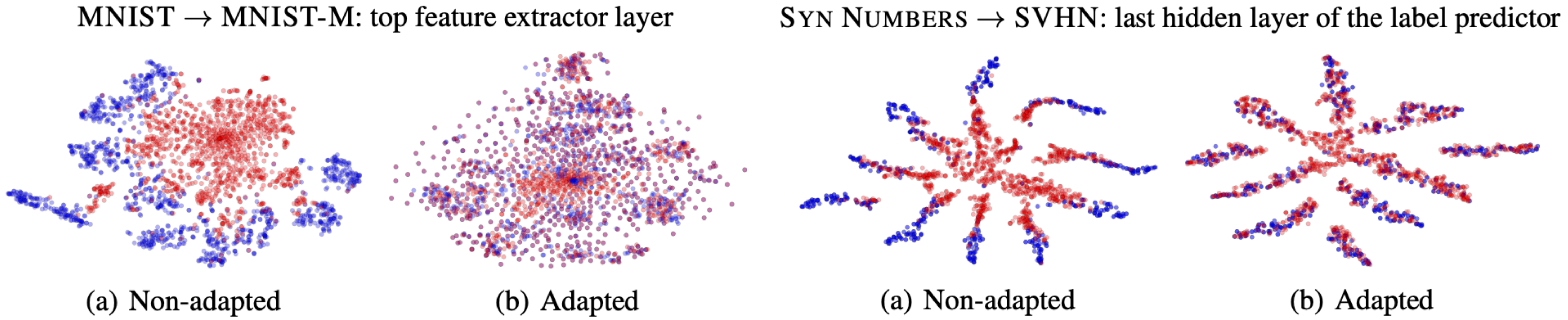
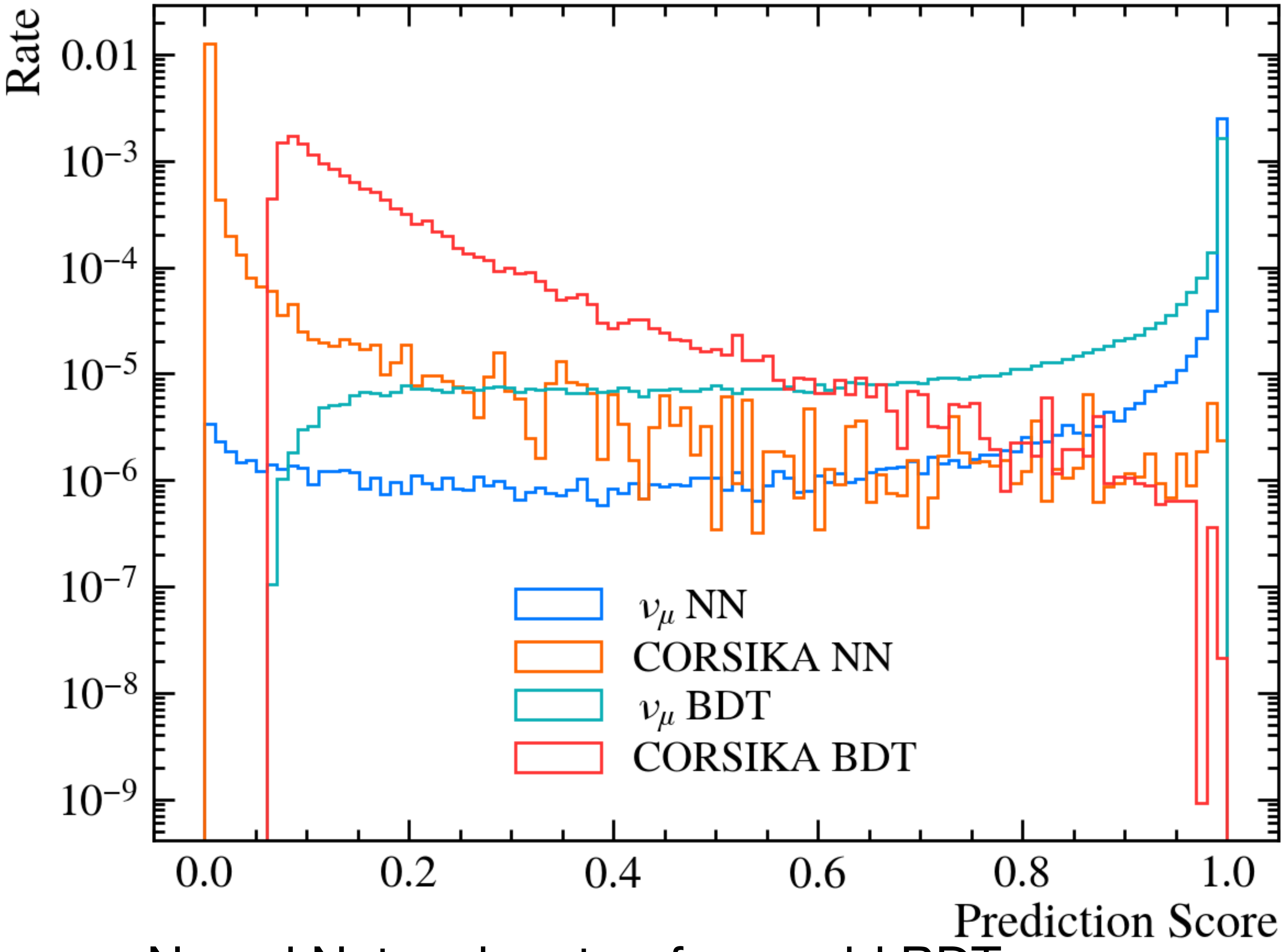
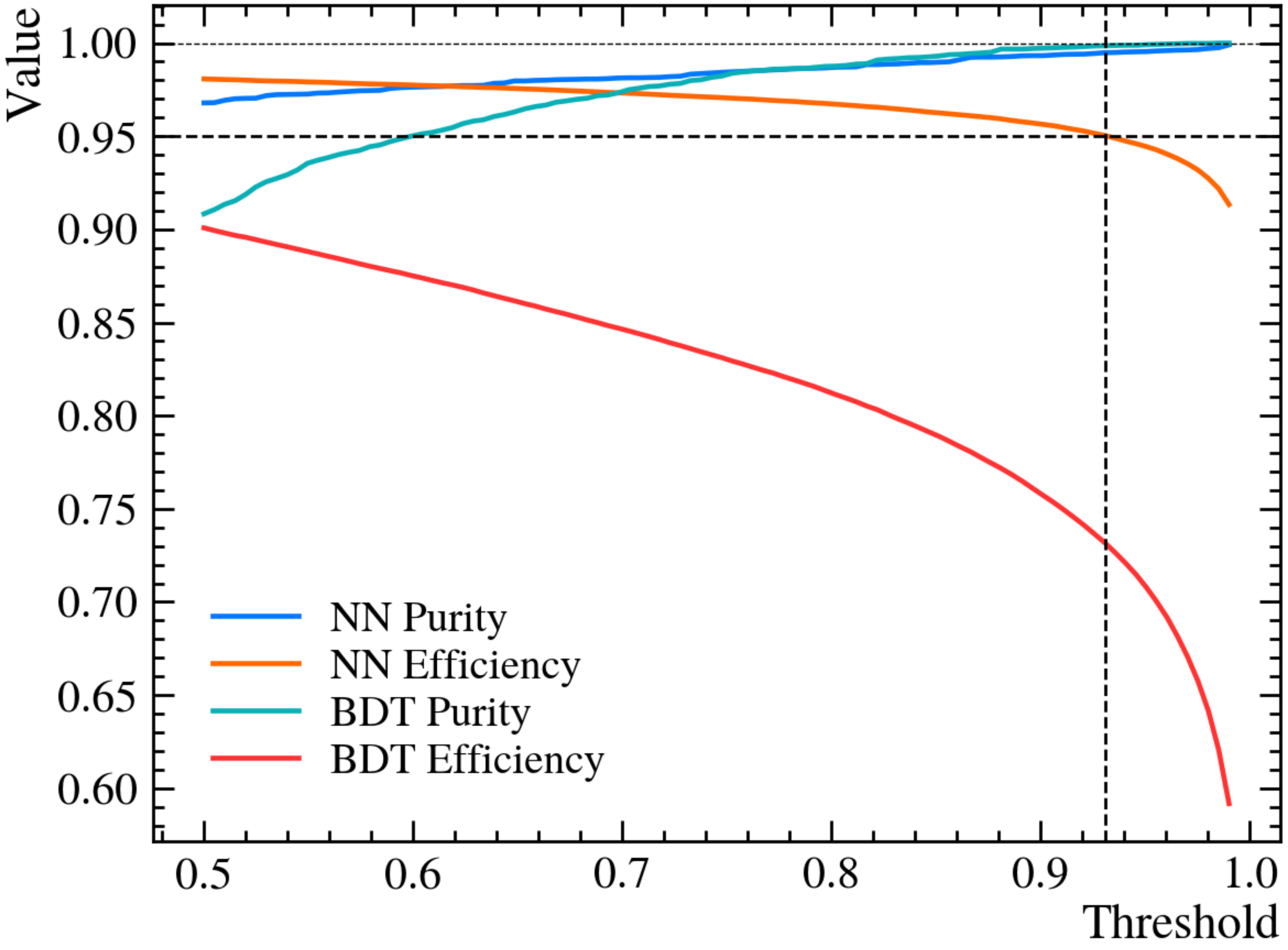


Figure 3. The effect of adaptation on the distribution of the extracted features (best viewed in color). The figure shows t-SNE (van der Maaten, 2013) visualizations of the CNN's activations **(a)** in case when no adaptation was performed and **(b)** in case when our adaptation procedure was incorporated into training. *Blue* points correspond to the source domain examples, while *red* ones correspond to the target domain. In all cases, the adaptation in our method makes the two distributions of features much closer.

Initial Network Performance





- Neural Network outperforms old BDT
- Unfortunately very poor Data / MC agreement!