New Machine Learning Approaches for Level-1 Trigger

Sydney Jenkins

CMS TRIGGER

Triggering: filter events to reduce data rates for offline processing

Level-1 Trigger → High Level Trigger → Offline

- 40 MHz
- ~10 µs latency
- ASICs/FPGAs

- ~500 kHz
- ~100 ms latency
- CPUs

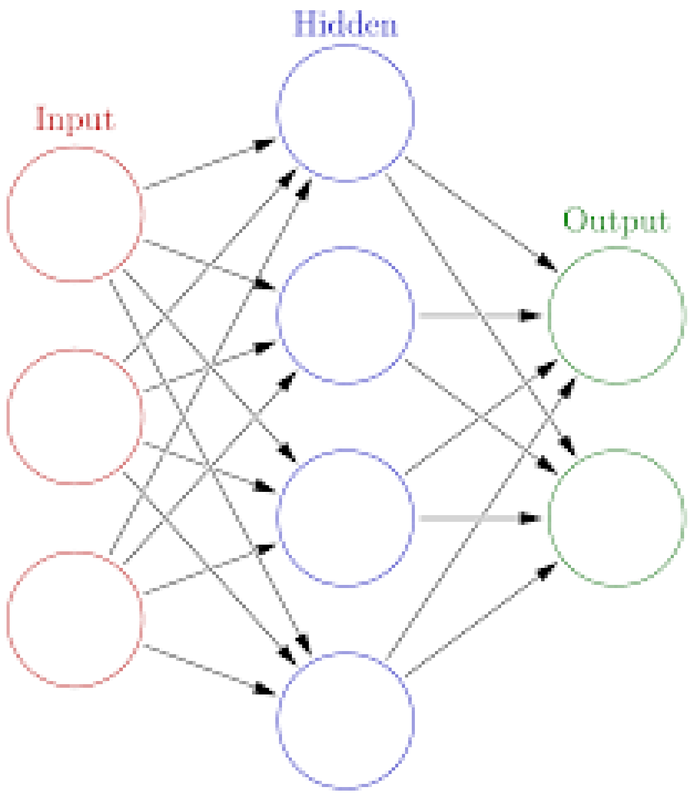
Upgrade challenges:

- Increased beam intensity, more data to evaluate
- Include outer tracker data

Neural Networks

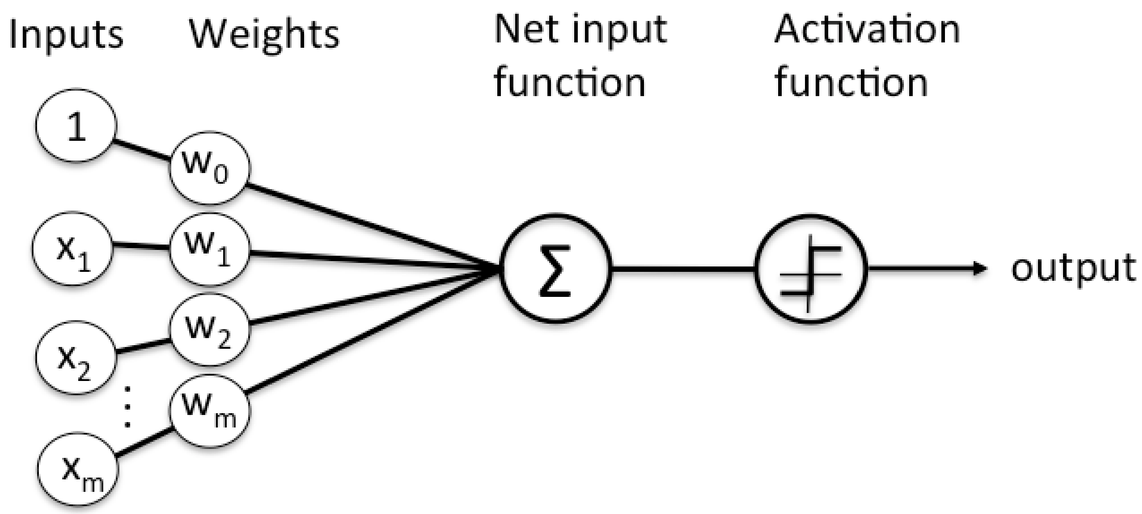
Structure

- Composed of layers, layers contain nodes
- Nodes: Inputs and corresponding weights are passed through activation function



Training

- Forward-propagation
- Calculate loss
- Backward propagation of errors (supervised learning)



Neural Networks with FPGAs

Inference performed on FPGA

Neural Networks

- Parallelizable
- Model non-linear effects

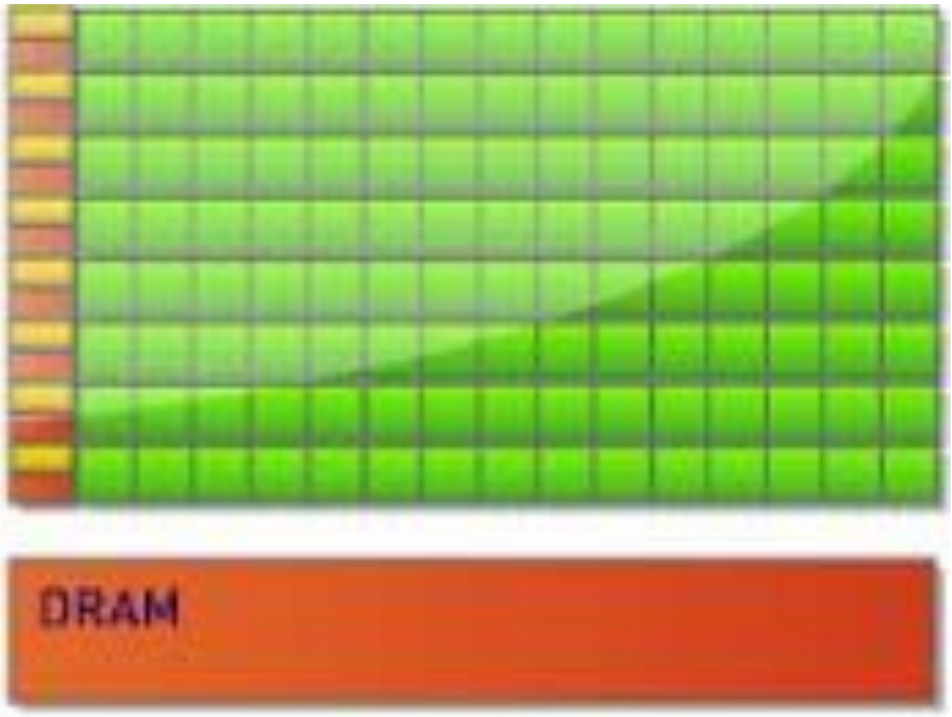
FPGAs

- Parallel architecture
- Reprogrammable
- Low latencies

Neural networks are well-suited to specialized hardware

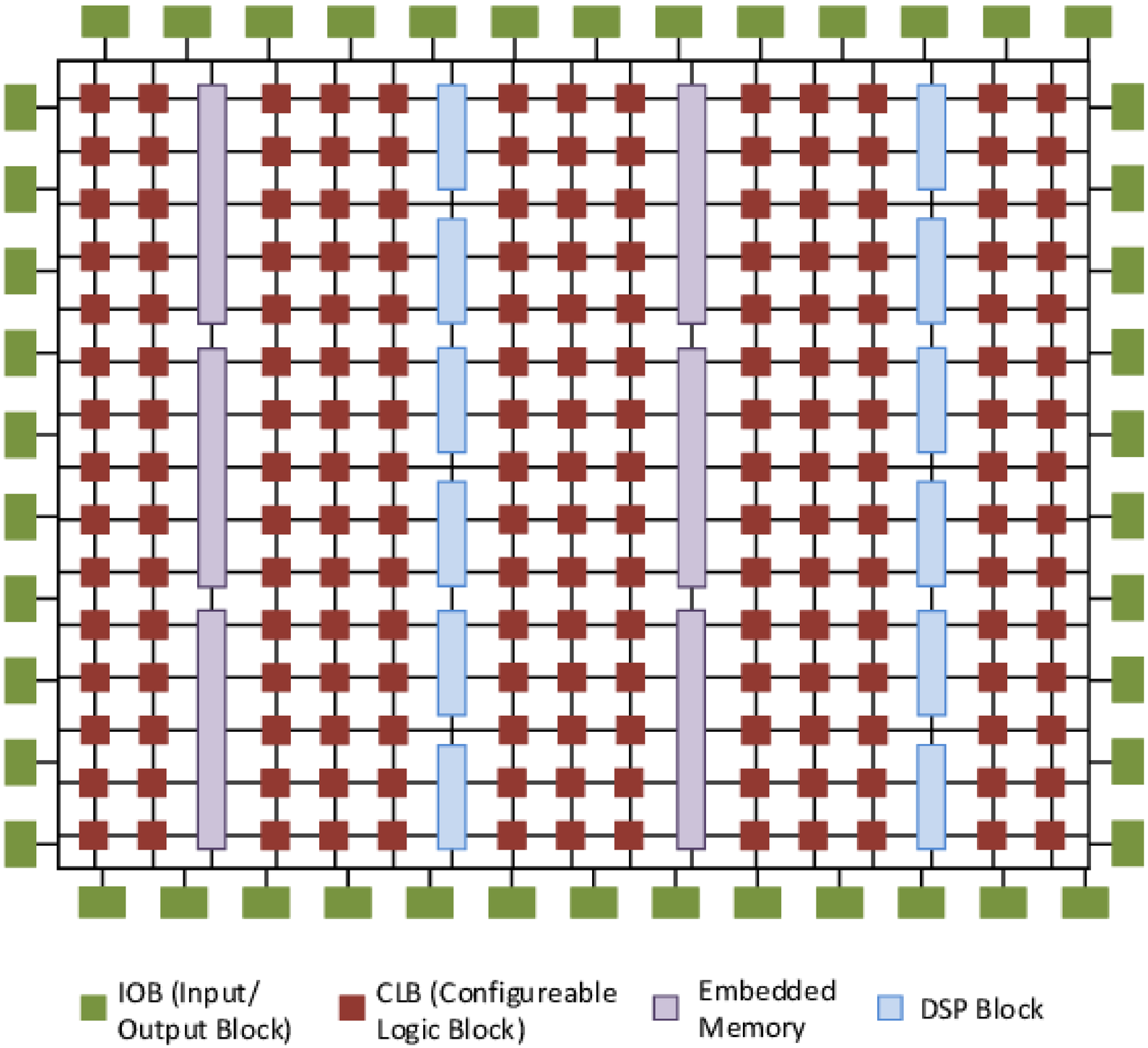
PROCESSORS

CPU

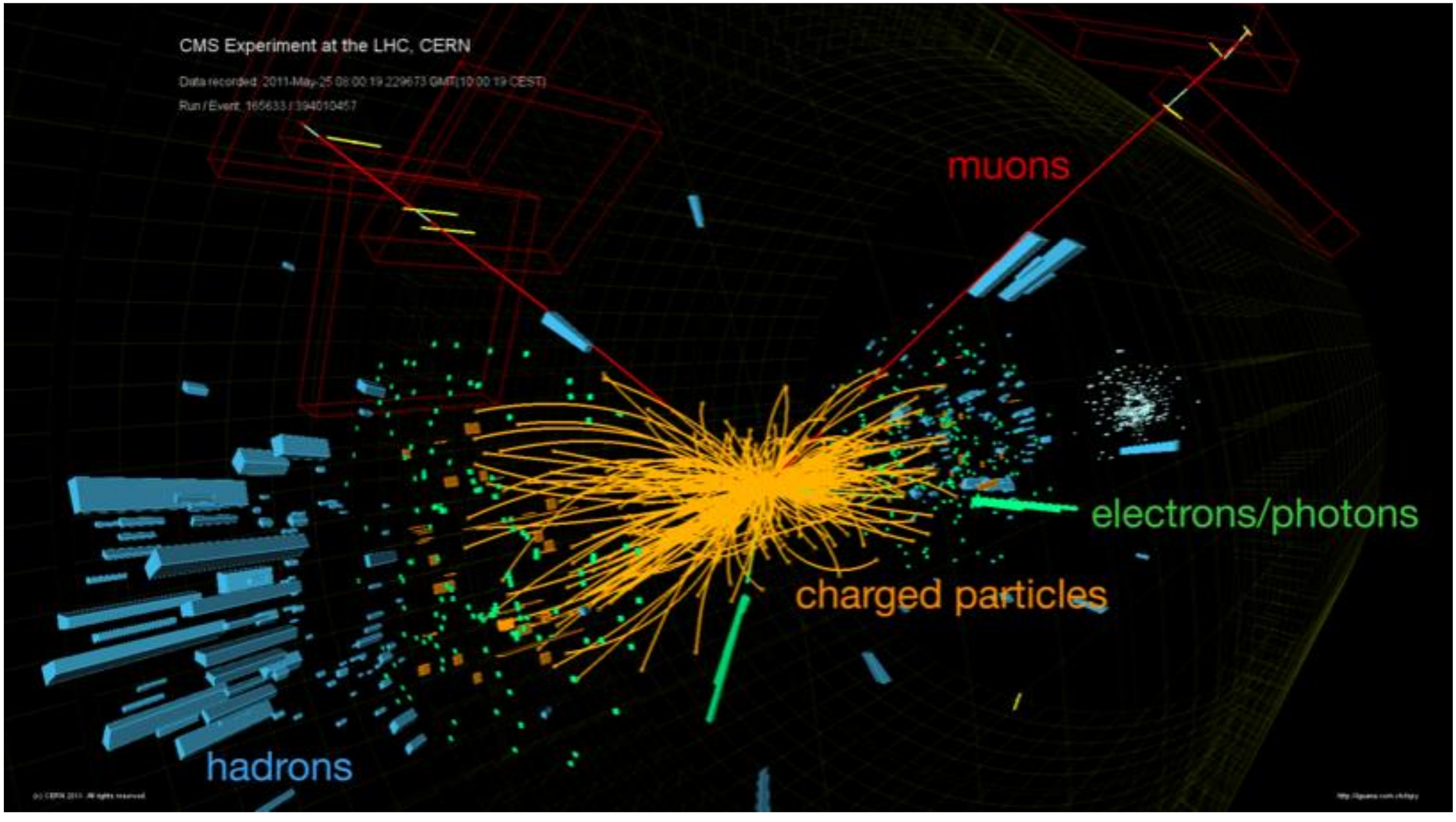


GPU

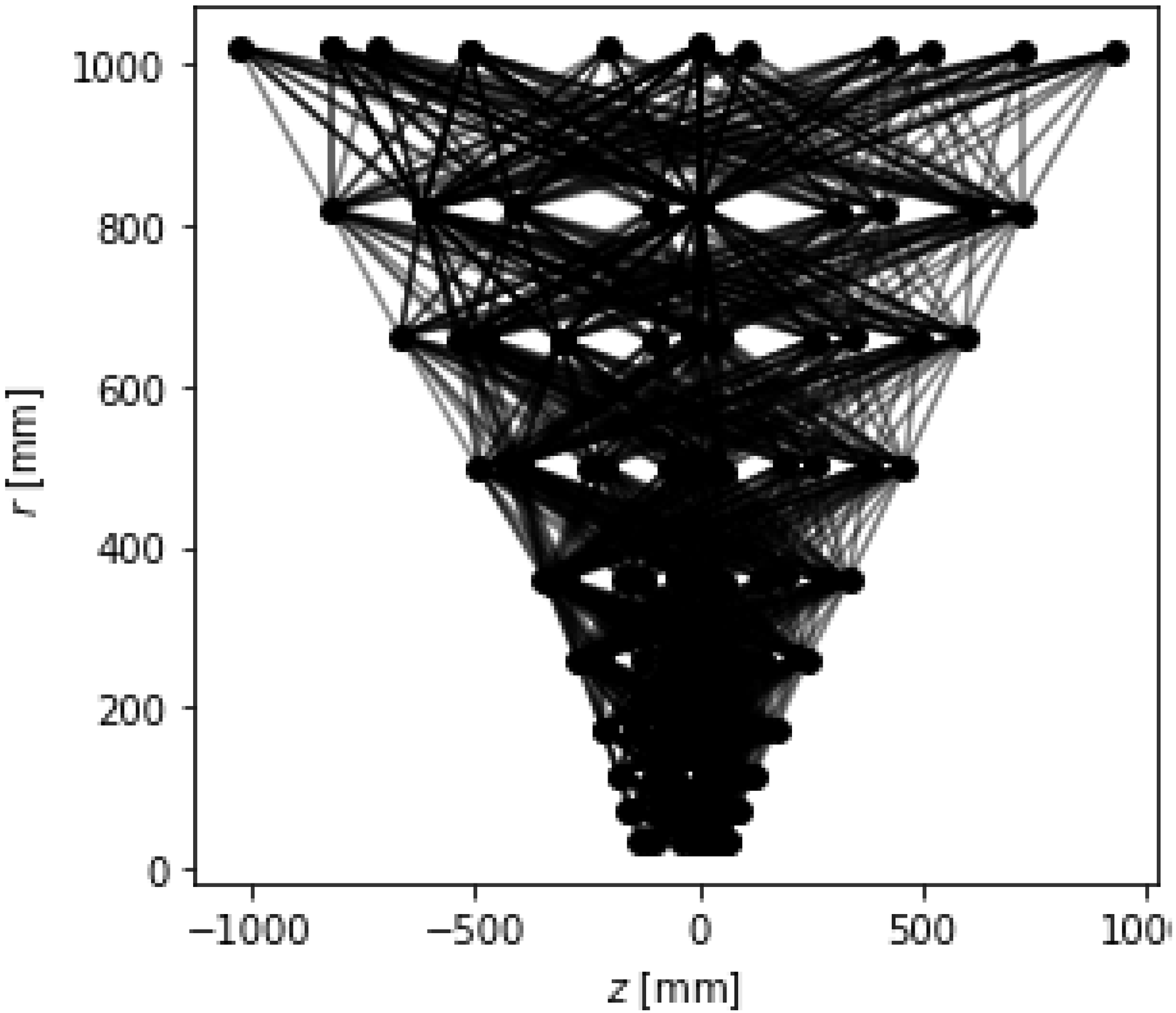
FPGA


IOB (Input/ Output Block)
CLB (Configureable Logic Block)
Embedded Memory
DSP Block

PATTERN RECOGNITION/TRACKING

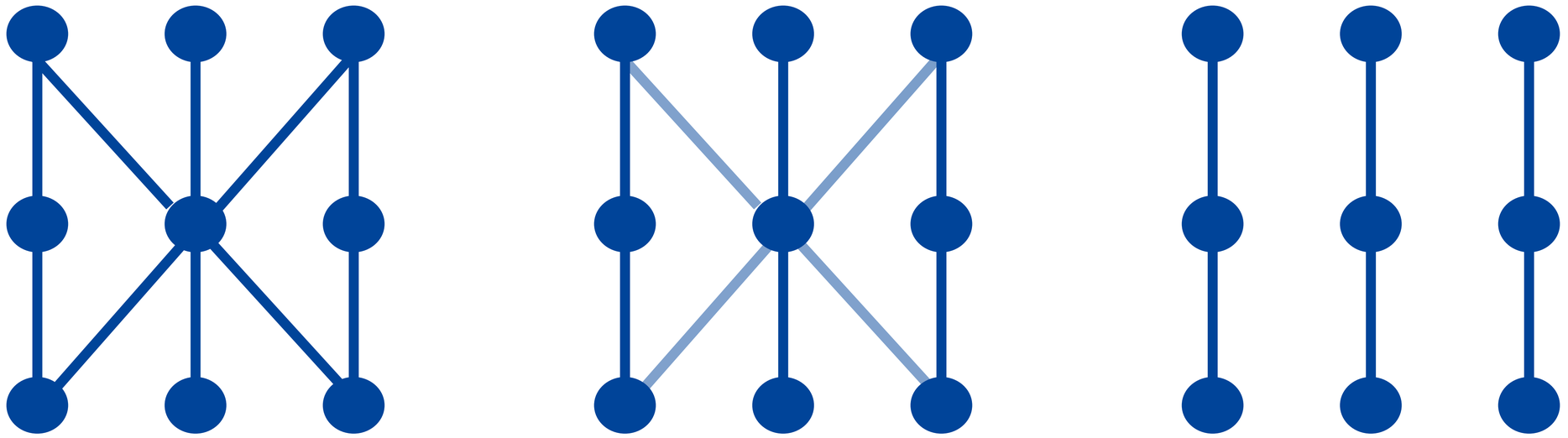


EXAMPLE TRACKING DATA



Graph-Based Neural Networks

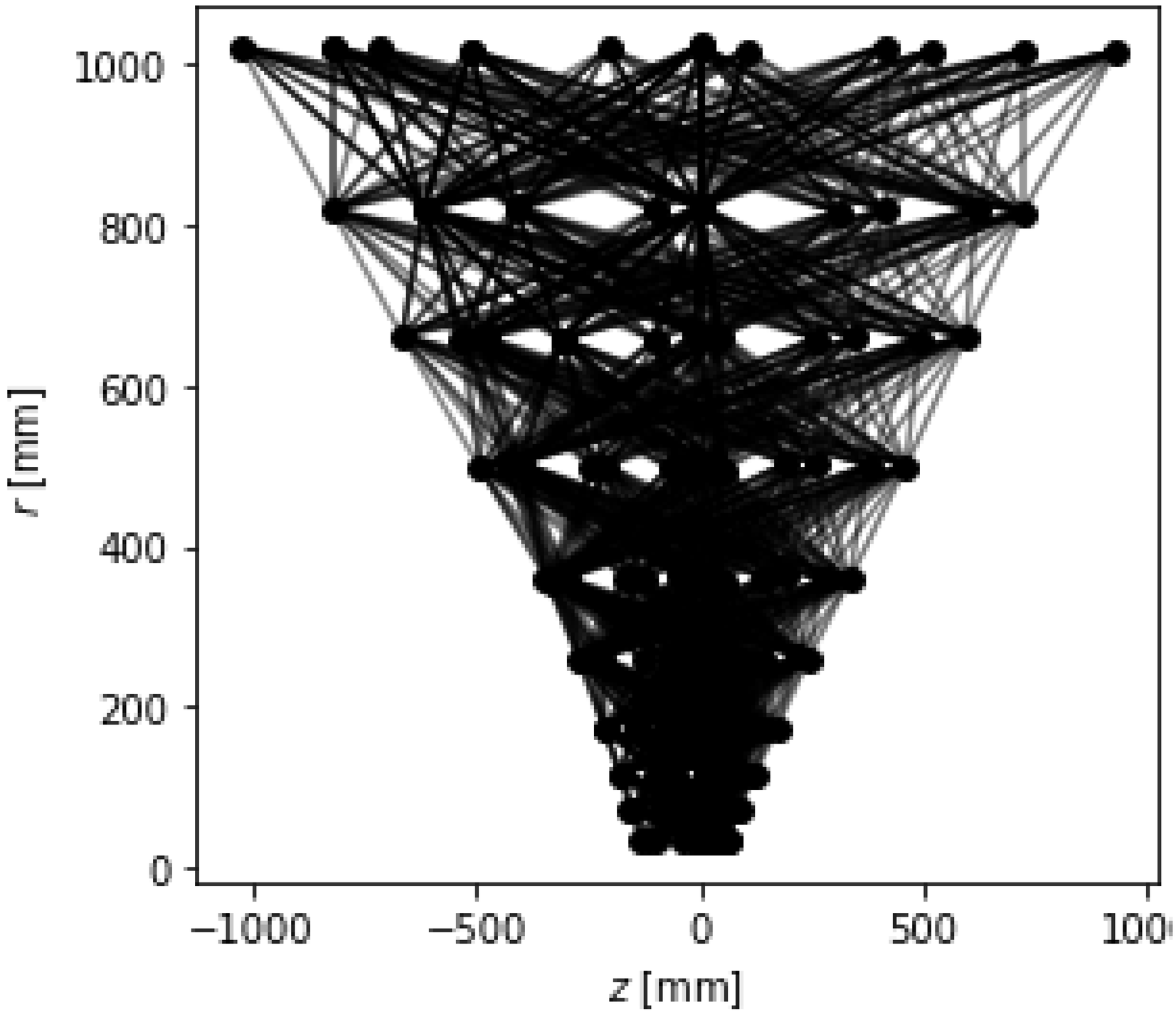
- Data structured as graph of connected hits
- Many possible architectures
- Components of our neural net:
  - Edge network: computes edge weights
  - Node network: updates node features



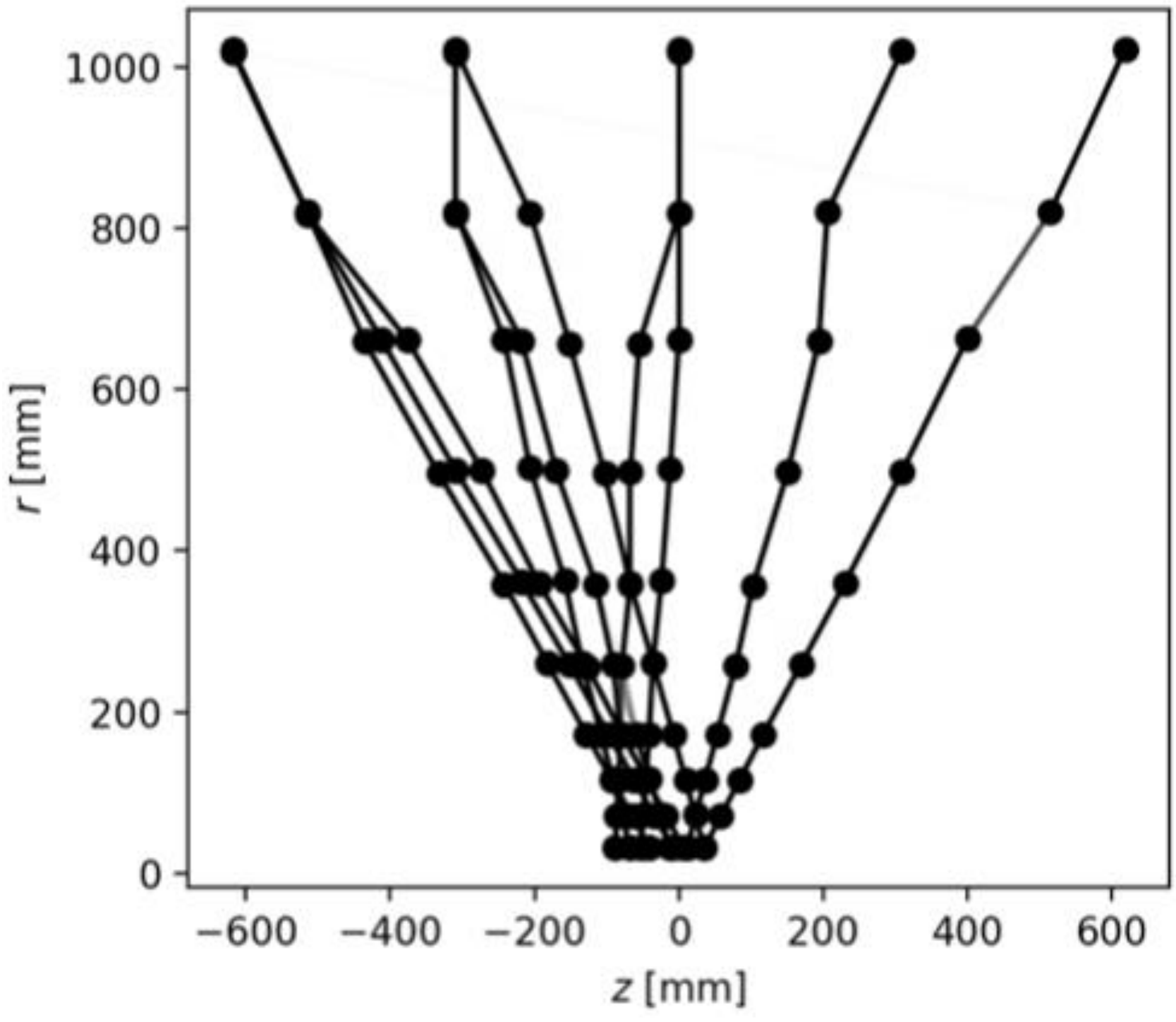
https://heptrkx.github.io/
https://indico.cern.ch/event/658267/contributions/2881175/attachments/1621912/2581064/Farrell_heptrkx_ctd2018.pdf

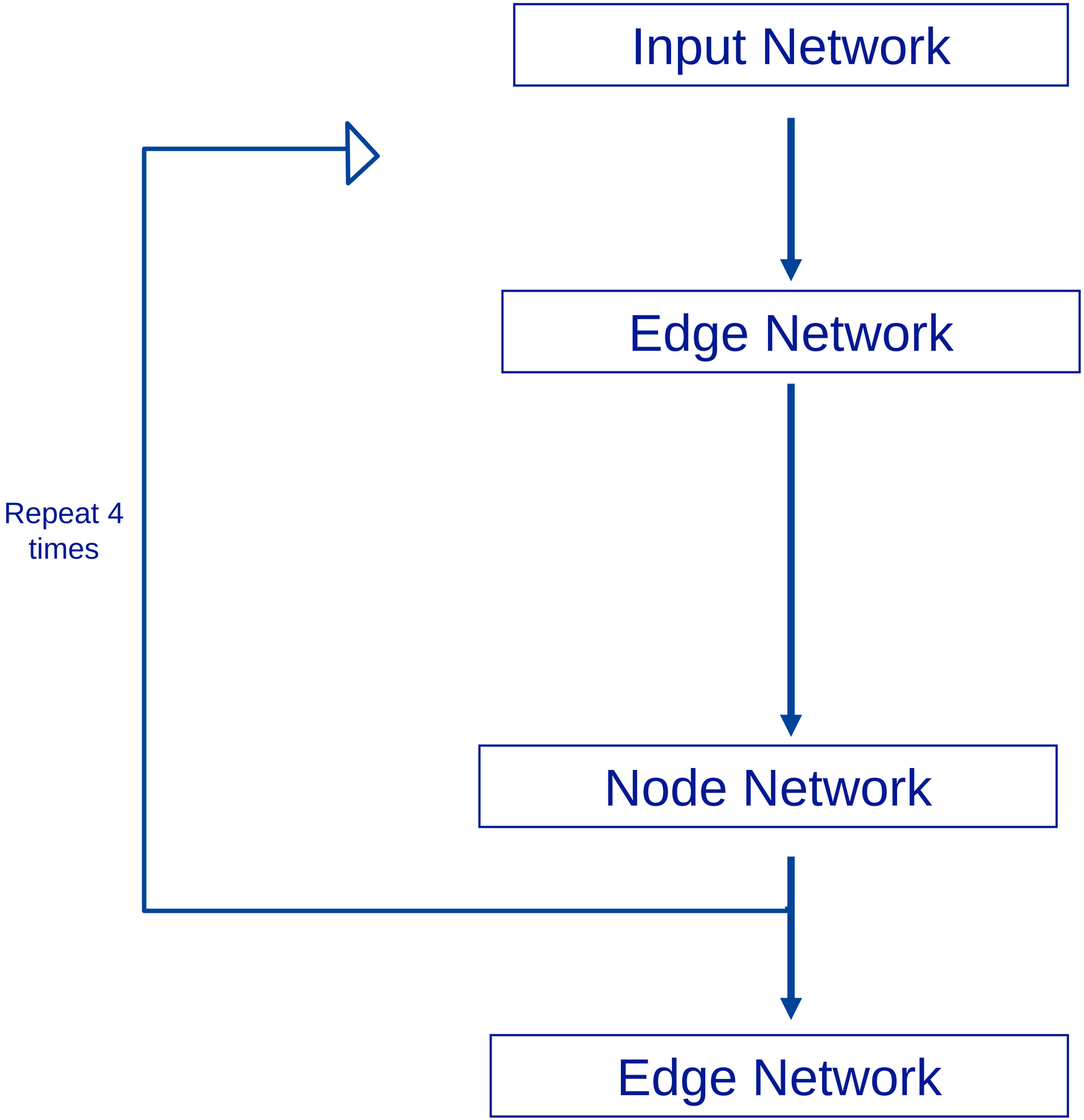
https://arxiv.org/abs/1612.00222
https://arxiv.org/abs/1609.02907
https://arxiv.org/abs/1704.01212

Evolution of Graph Edges



Evolution of Graph Edges




Input Network
Edge Network
Node Network
Edge Network
Repeat 4 times

Input Network

Input variables: H (100x35), R_i (900x100), R_o (900x100)
Inputs: $R_i^T H$ (900x35) + $R_o^T H$ (900x35)

Edge Network

2272 weights

Outputs: edge weight, e (900x1)

Repeat 4 times

Input variables: H (100x35), e (900x1), R_i (900x100), R_o (900x100)
Inputs: $[R_i \odot e]R_o^T H$ (100 x 35) + $[R_o \odot e]R_i^T H$ (100 x 35) + H (100x35)

Node Network

4384 weights
Outputs: O’ (100x32)

Edge Network

Scenario
Tracker layers: 10
Tracks: 10
Hits: 100
Allowed connections: 900

900 x 2272 = **2,044,800** multiplications

31,500 multiplications

100 x 4272 = **438,400** multiplications

12,113,200 total multiplications

Optimizing Efficiency

How can we fit this neural network on an FPGA?

Model

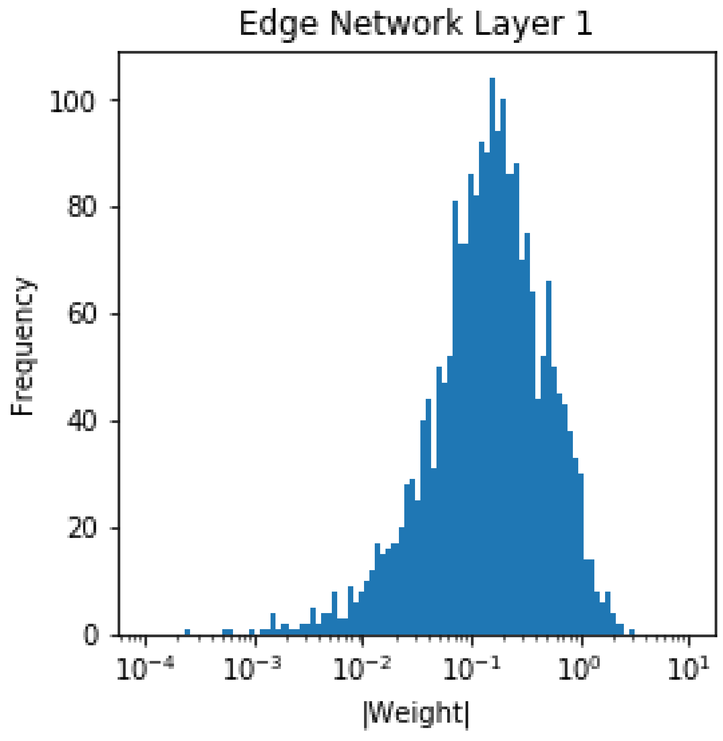
- Hidden features reduction
- Edge reduction
- Network compression

FPGA

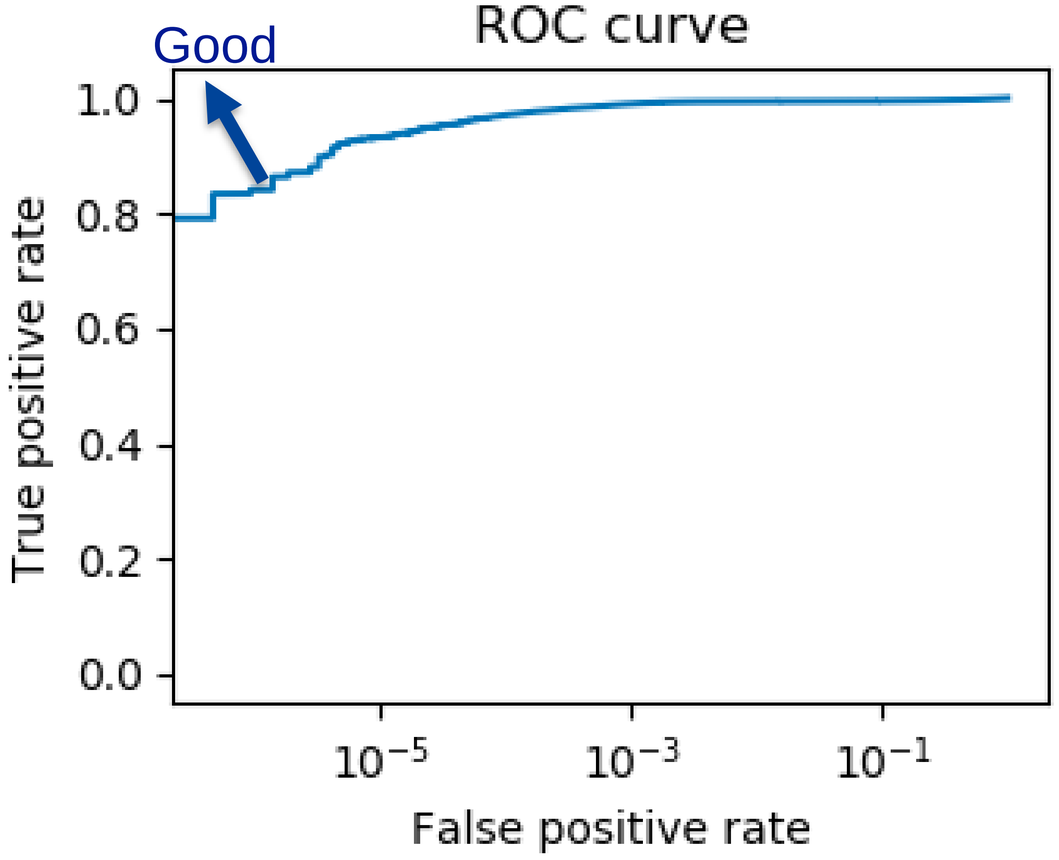
- Reduced precision
- Reuse resources

ORIGINAL TRAINING

Weight Distribution



Accuracy



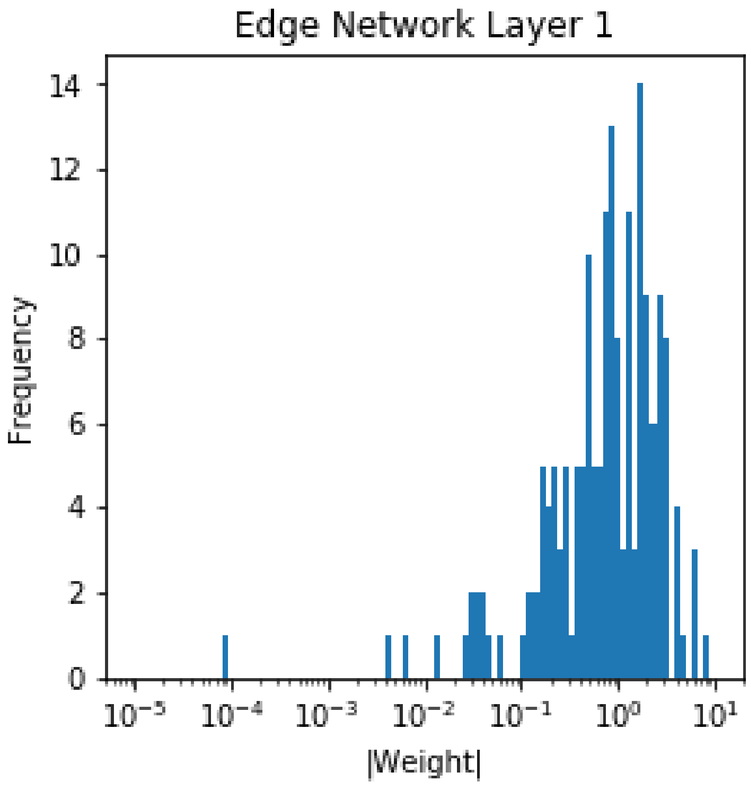
Hidden features: 32

Geometric cuts: None

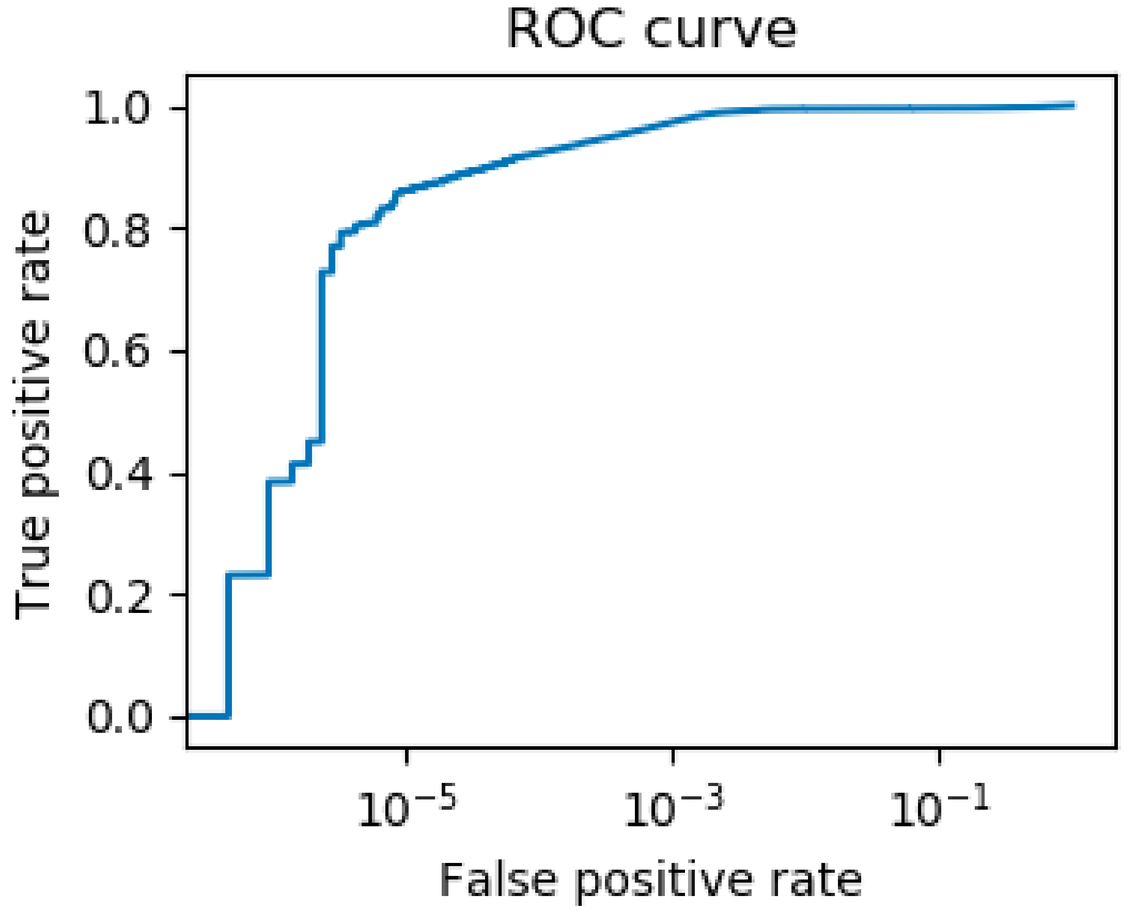
L1 Regularization: 0

Hidden Features Reduction

Weight Distribution



Accuracy

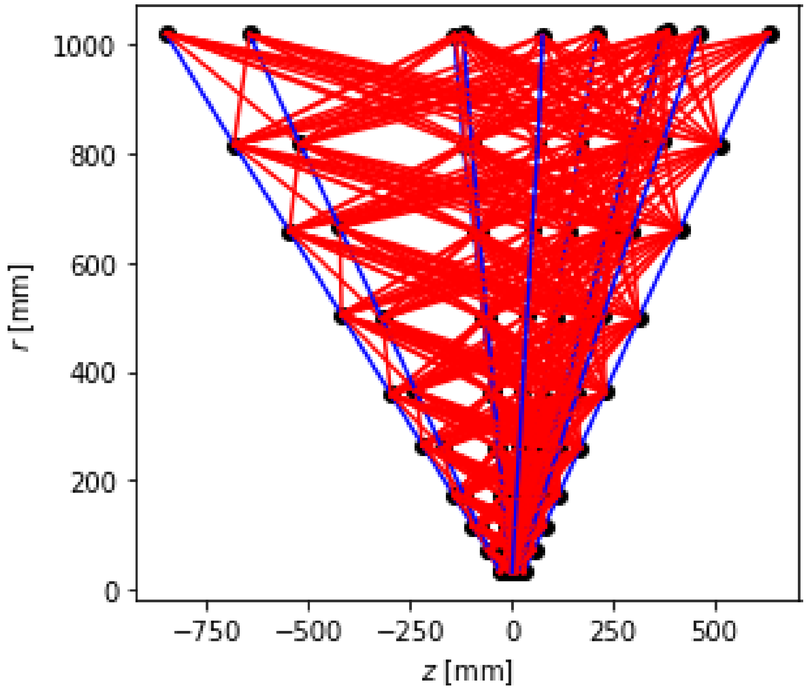


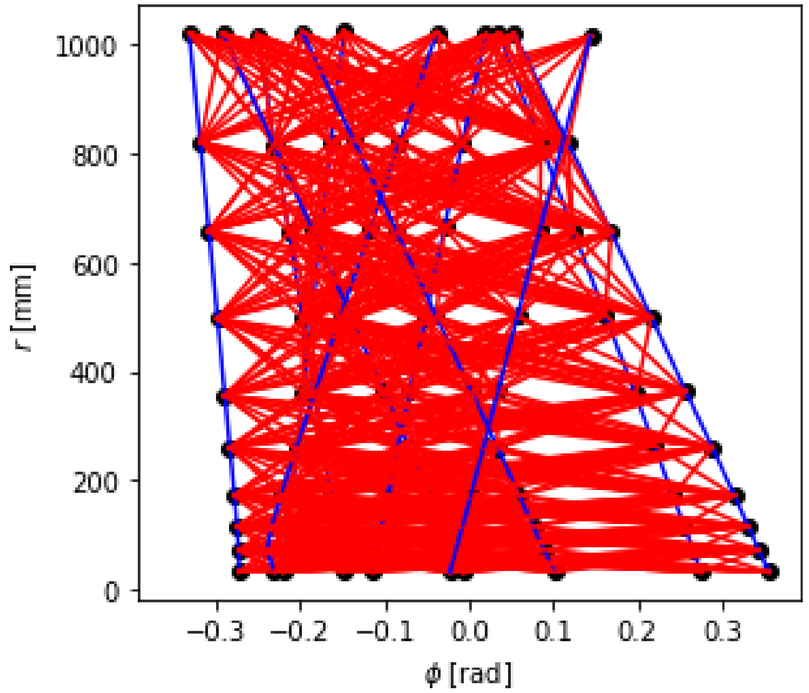
Hidden features: 8

Geometric cuts: None

L1 Regularization: 0

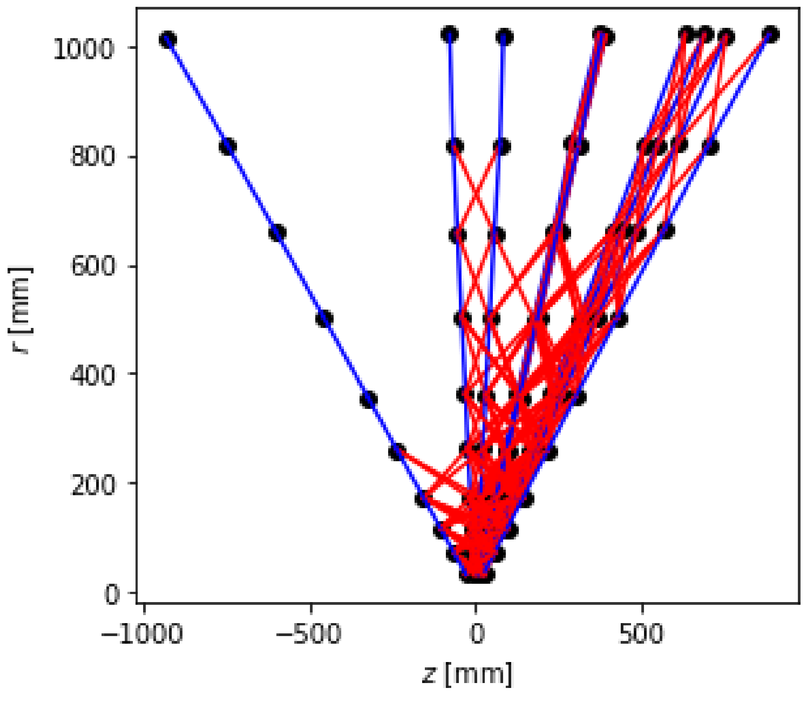
Edge Restriction

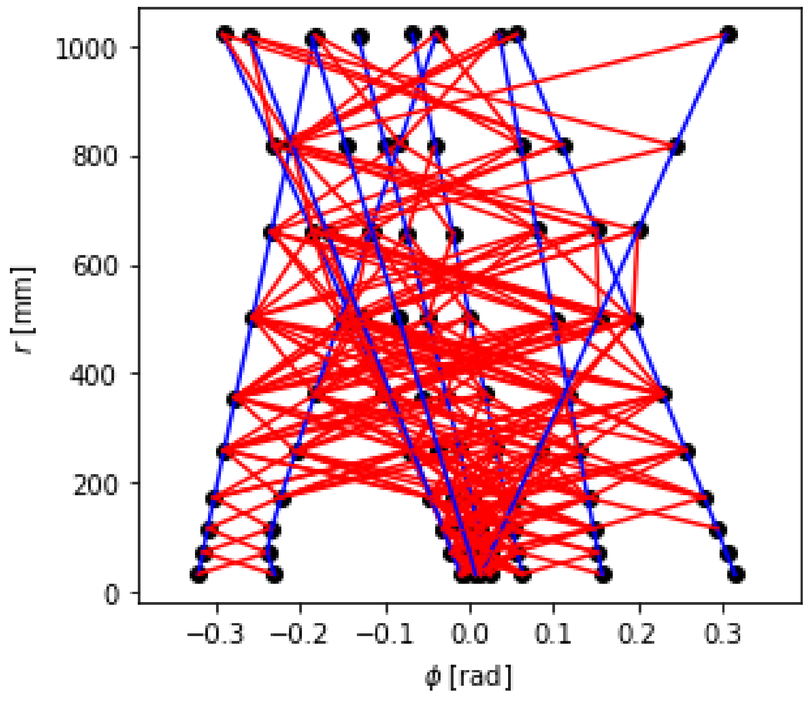




900 edges

No restrictions



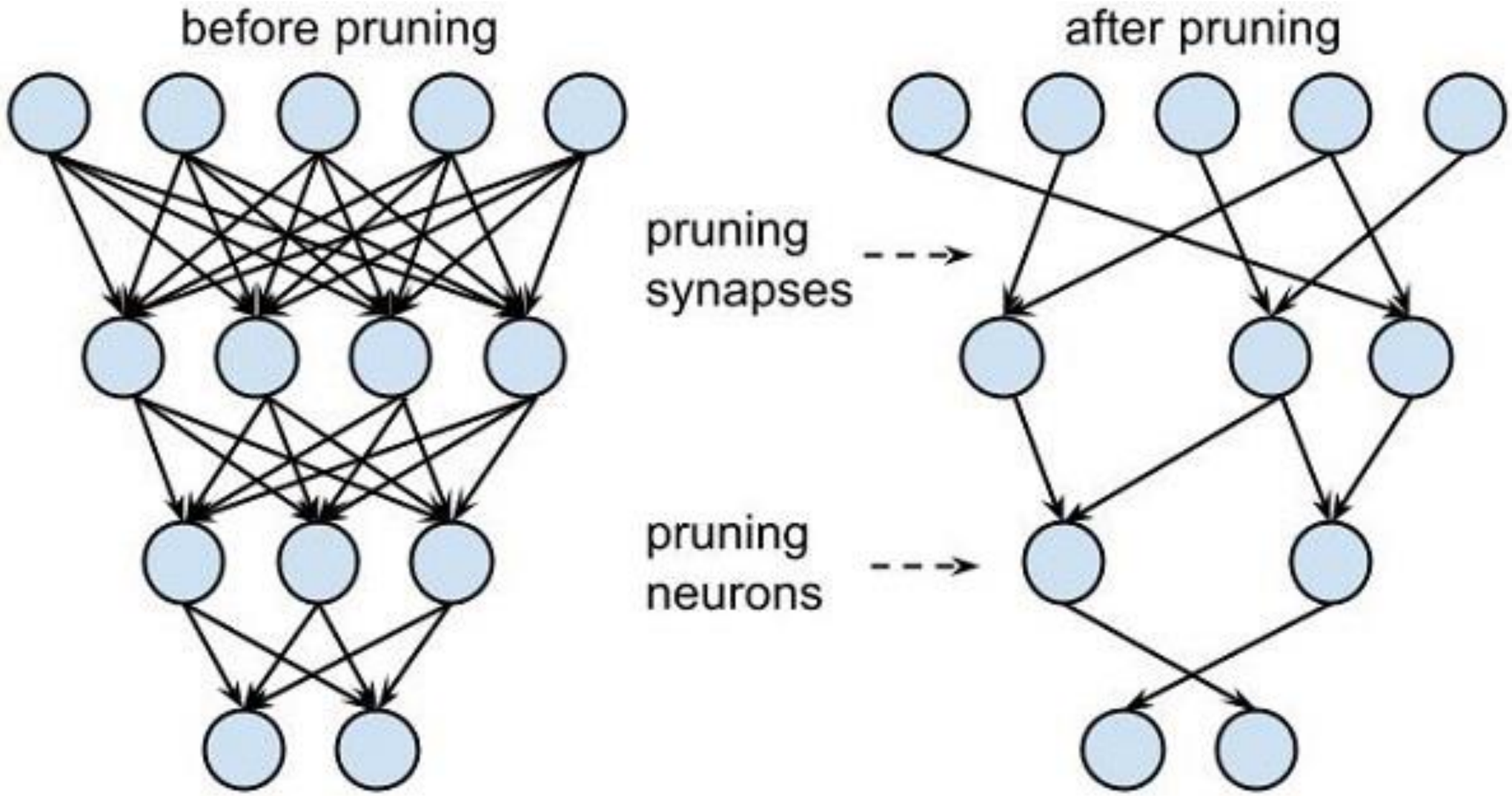


~350 edges

Z restriction: 600
Φ inner restriction: .003
Φ outer restriction: .006

Network Compression

Pruning: mask unimportant weights

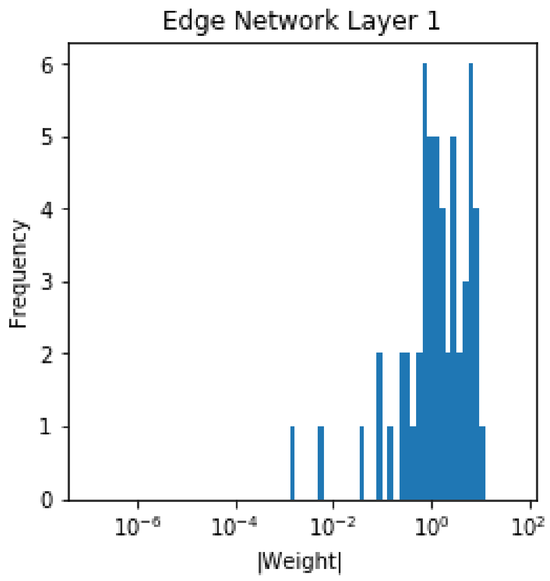


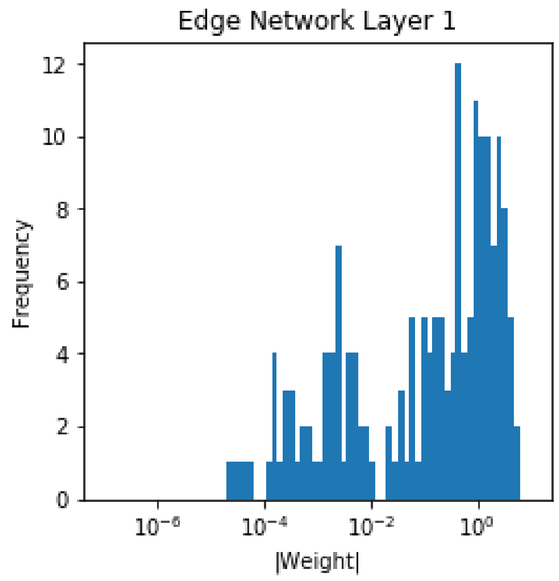
Network Compression

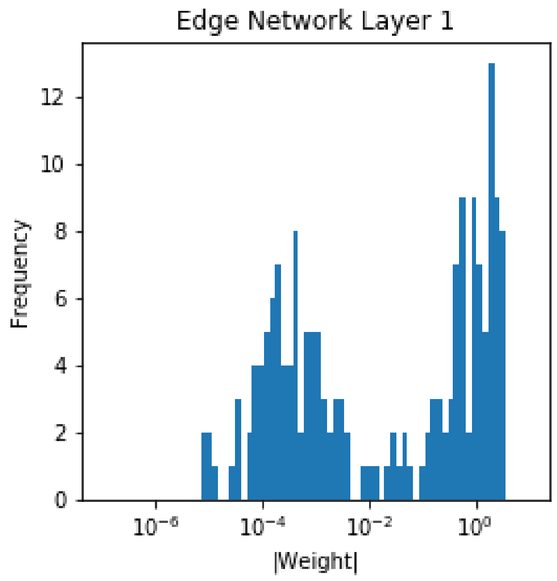
- Use L1 regularization during training to avoid overfitting and down-weight unimportant weights of the model
- Set weights below a certain threshold to zero
- Re-train with those weights fixed to zero

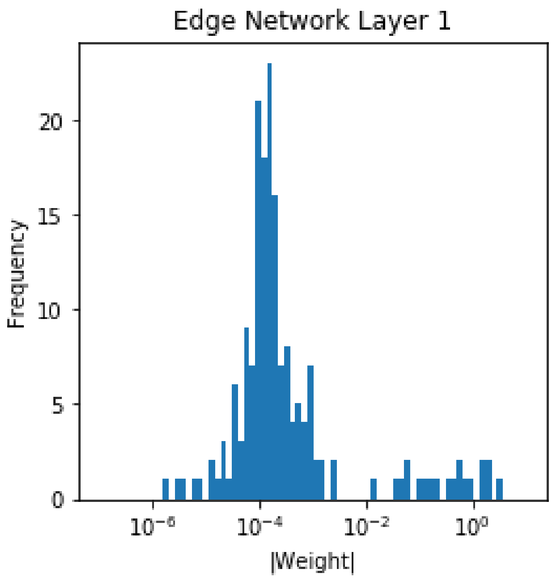
$$\text{Regularization term} = \lambda \sum_{i=0}^{k} |w_i|$$

$\lambda = 0$ $5*10^{-5}$ 10^{-4} 10^{-3}









GRADUAL MASKED TRAINING

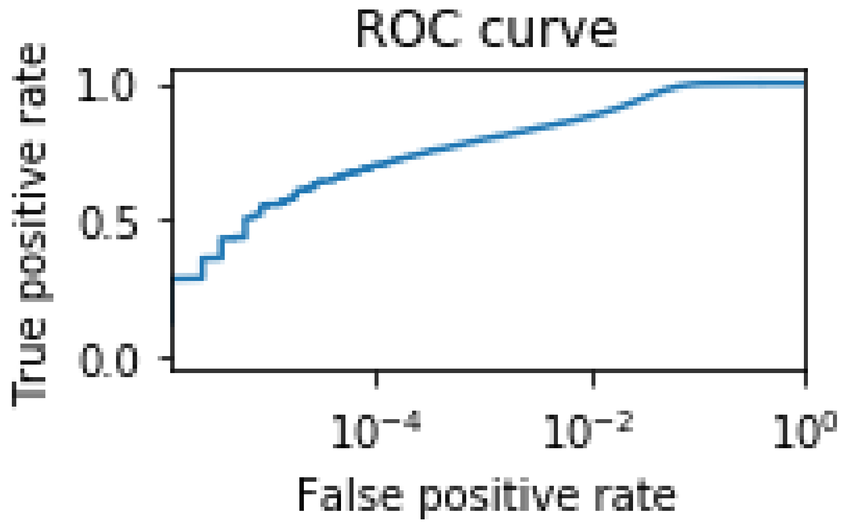
Iterative process

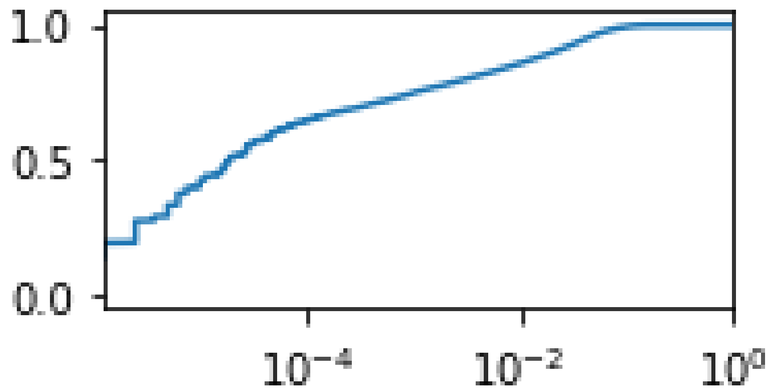
Train

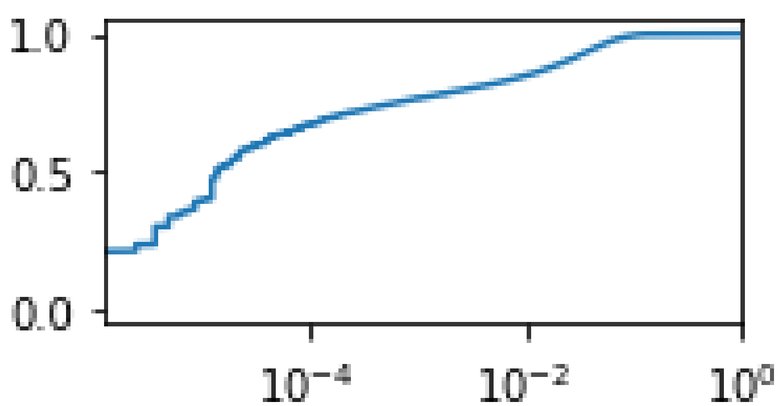
Apply mask

Train

Output







$\lambda = 3*10^{-5}$

100% weights

cut: $5*10^{-3}$

88% weights

cut: 10^{-2}

81% weights

Input Network

Input variables: H (100x35), R_i (900x100), R_o (900x100)
Inputs: $R_i^T H$ (900x35) + $R_o^T H$ (900x35)

Edge Network

2272 weights

Outputs: edge weight, e (900x1)

Repeat 4 times

Input variables: H (100x35), e (900x1), R_i (900x100), R_o (900x100)
Inputs: $[R_i \odot e]R_o^T H$ (100 x 35) + $[R_o \odot e]R_i^T H$ (100 x 35) + H (100x35)

Node Network

4384 weights
Outputs: O’ (100x32)

Edge Network

Scenario
Tracker layers: 10
Tracks: 10
Hits: 100
Allowed connections: 368

54,000 multiplications

4,000 multiplications

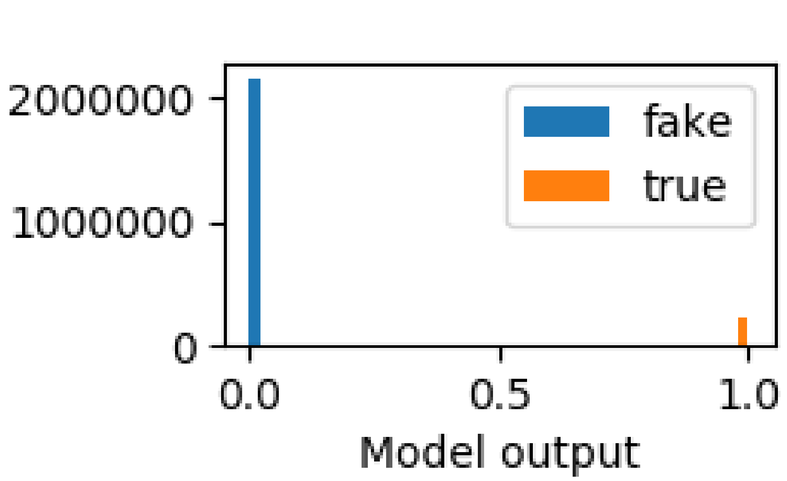
38,700 multiplications

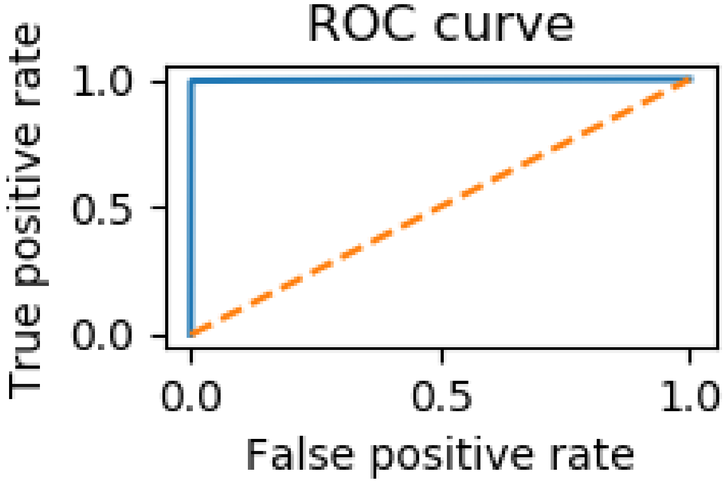
427,200 total multiplications

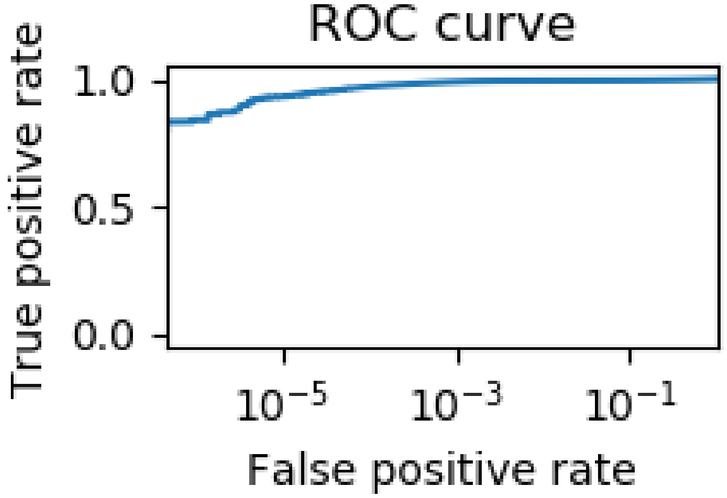
COMPARISON

Optimal Performance

Hidden features: 32 Regularization: 0

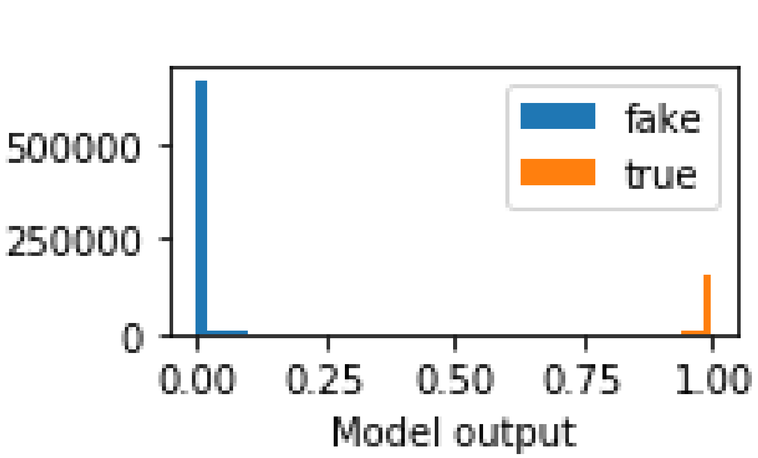


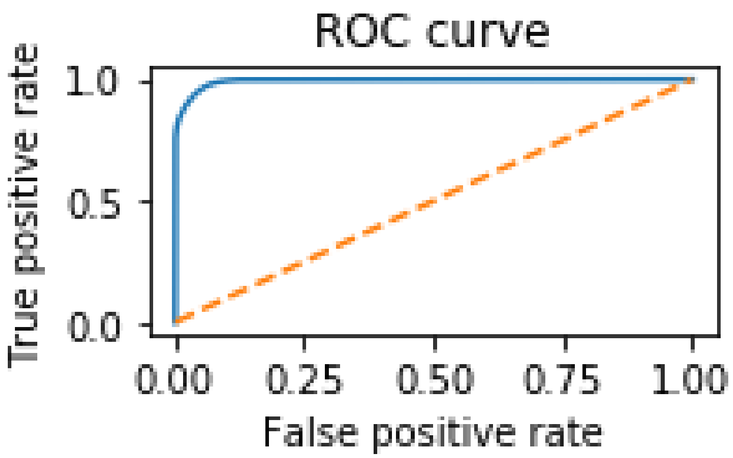


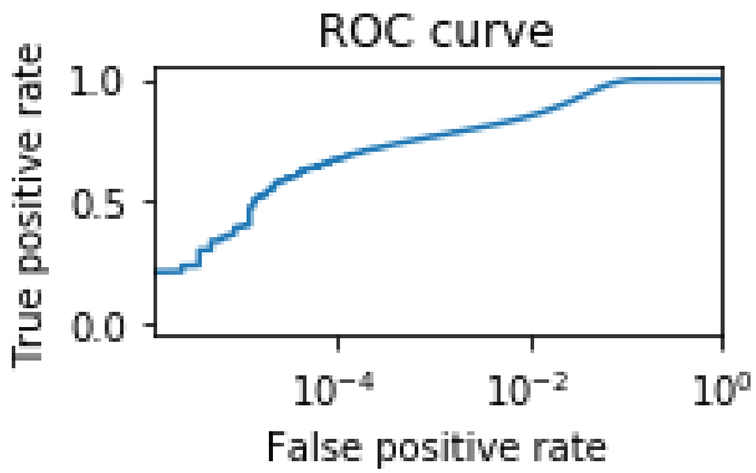


Optimal Efficiency

Hidden features: 8 Regularization: $3*10^{-5}$







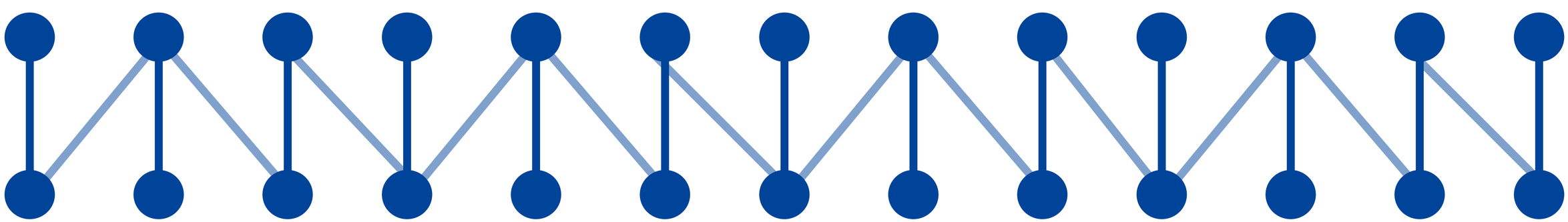
SUMMARY

Project

- Reduced multiplications necessary for inference (~12,000,000 to ~500,000)
- Now feasible to implement neural network on FPGA

Personal

- Learned about accelerator physics
- Experience with PyTorch

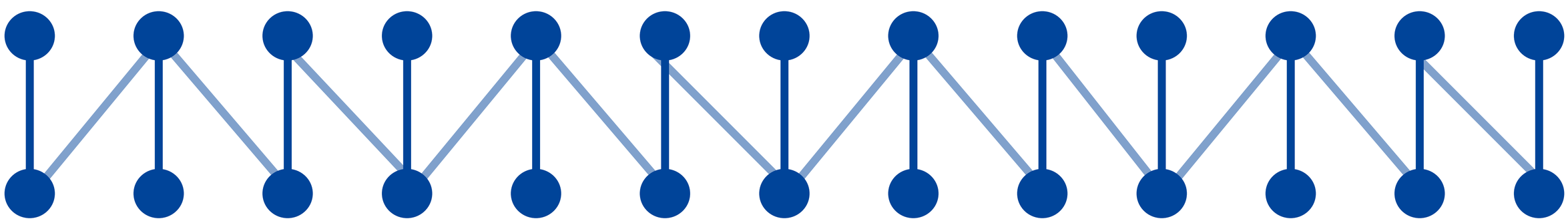


Next Steps

- Implement on FPGA using High-Level Synthesis called hls4ml: https://hls-fpga-machine-learning.github.io/hls4ml/

- Generalize model to account for missing hits, secondary vertices, etc.

- Test model on CMSSW Phase-2 Upgrade Simulation muon trigger inputs



QUESTIONS?