Anomaly detection with decision tree autoencoder on FPGA for L1 trigger system at LHC

Stephen Roche (SLU)*
On behalf of my co-authors (Pitt, Westmont)
of 2304.03836 [hep-ex]

SAINT LOUIS UNIVERSITY
SCHOOL OF MEDICINE

University of Pittsburgh

WESTMONT

Pheno 2023
May 9, 2023
https://indico.cern.ch/event/1218225/

Outline

Introduction

- Autoencoders for anomaly detection
- Machine learning at L1

Decision tree autoencoder

- Novel training method

Firmware design

- Novel latent-spaceless design for FPGA

Physics & FPGA results

- Exotic decay of Higgs to pseudoscalars to 2e 2μ
- “LHC anomaly detection” dataset

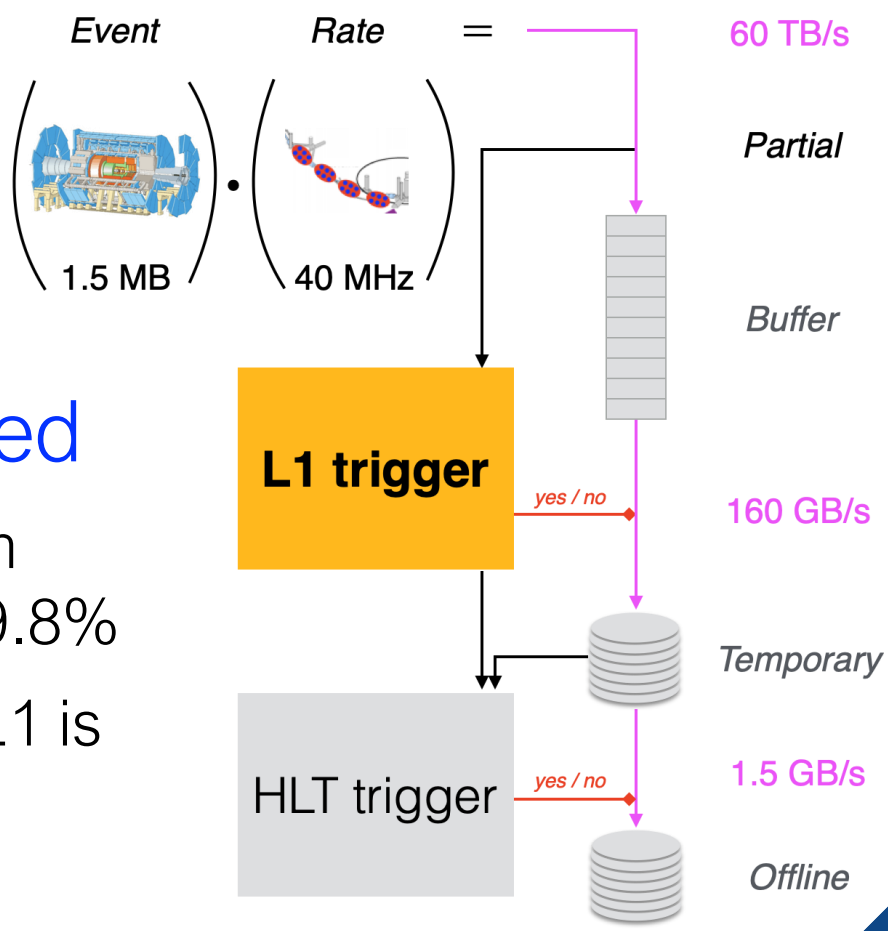
How to save BSM at L1 without models

Anomaly detection in HEP

Stephen Roche

Model-agnostic detection of BSM signals

- Many anomaly detection methods have been devised and tested on a variety of different HEP problems [https://iml-wg.github.io/HEPML-LivingReview]
- Anomaly detection in ATLAS analysis [ATLAS-CONF-2022-045]



Can't analyze data that's not saved

- L1 triggers at ATLAS & CMS use custom electronics such as FPGAs to discard 99.8%
- Implementing anomaly detection at the L1 is challenging and possible (this talk)

Source: http://cern.ch/twiki/pub/Atlas/TDAQSpeakersCommitteeCommonReferences/tdaqFullNew2017.pdf

Autoencoder intro

Stephen Roche

What is an autoencoder (AE)

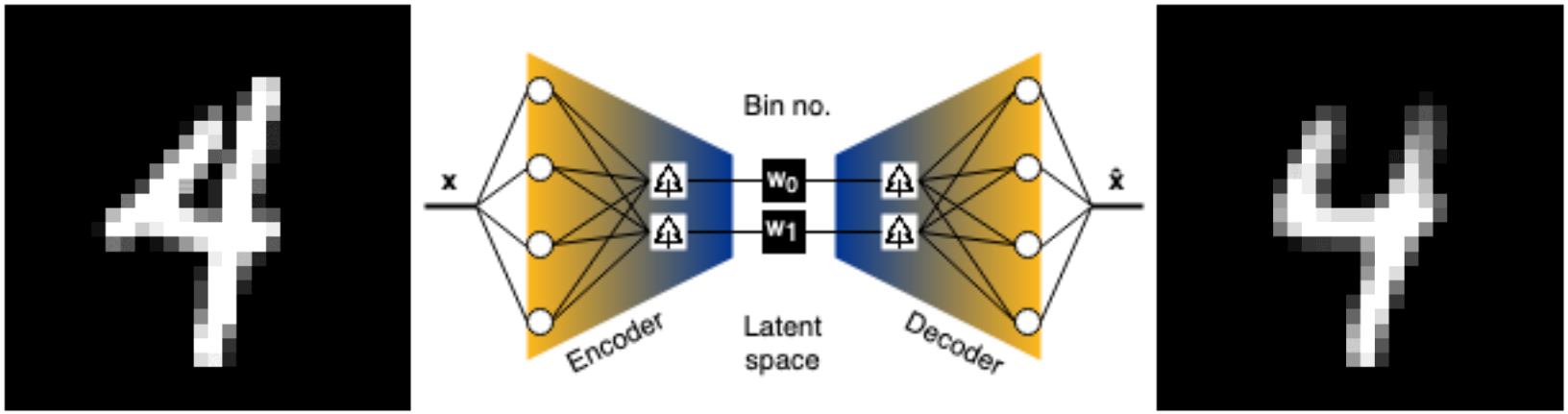
- Autoencoders can be used for data compression-decompression
  - Typical methods use neural networks to encode-then-decode
- We use decision trees (see below)

Real example

784 8-bit no.

One 20-bit no.

784 8-bit no.



Input

Output looks good!

Details

- MNIST 28^2 8-bit input/output, 1 tree depth 20, trained on 0,1,2,3,4
- Input-output distance is relatively small = good compression

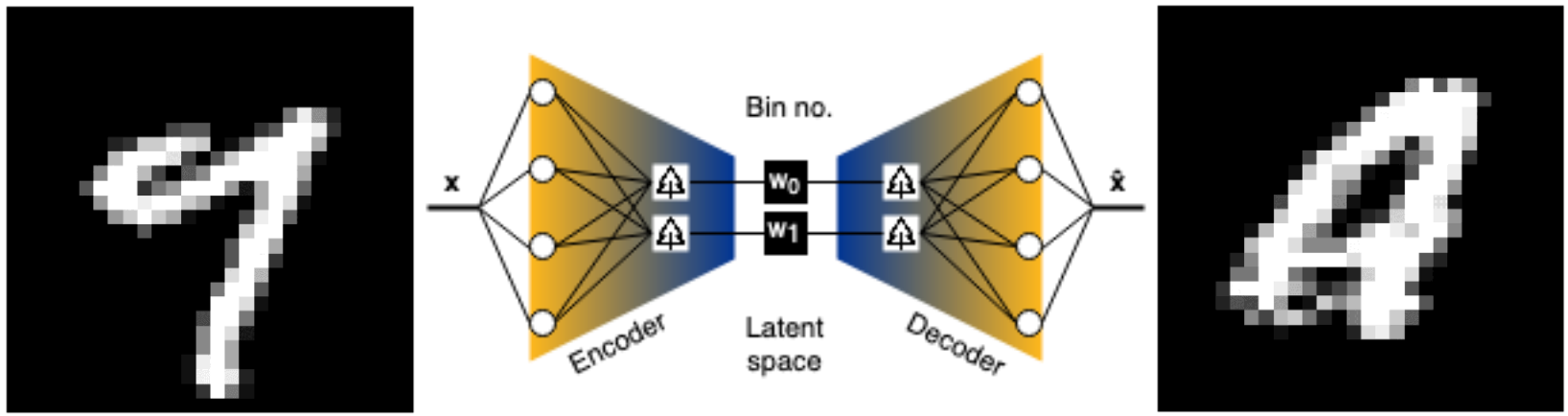
AE for anomaly detection

AE can be used for anomaly detection

- Train the autoencoder with a sample S
  - If it encounters input similar to S, then output is good (prev. slide)
  - If it encounters input different than S, then output garbled (below)

Real example

784 8-bit no. | One 20-bit no. | 784 8-bit no.



Input

Output looks bad!

Details

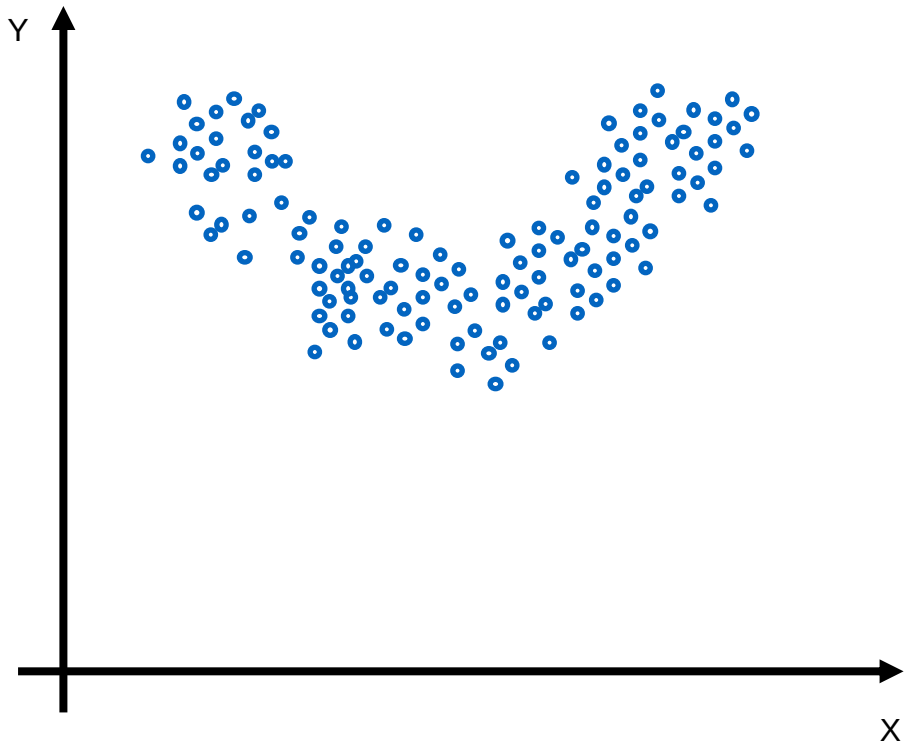
- MNIST 28^2 8-bit input/output, 1 tree depth 20, trained on 0,1,2,3,4
- Input-output distance is relatively large = anomaly

Decision tree autoencoders

Stephen Roche

Training philosophy (novel method described in paper)

- Place small “bins” around locations of high event density
- Example
  - 2d toy dataset, say x = p_T and y = eta for some SM sample

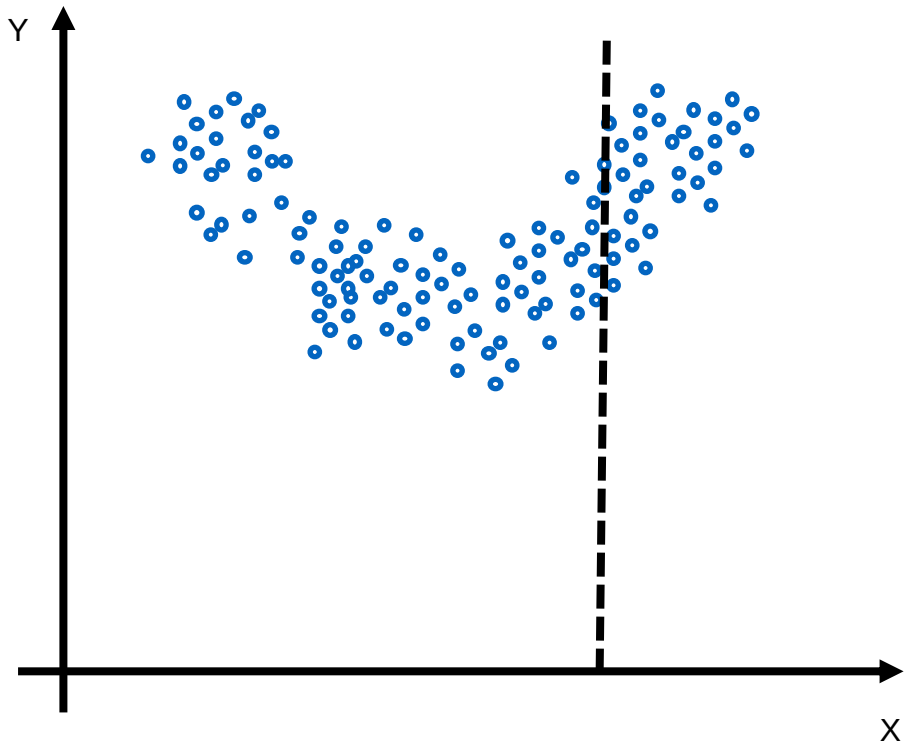


Decision tree autoencoders

Stephen Roche

Training philosophy (novel method described in paper)

- Place small “bins” around locations of high event density
- Example
  - 2d toy dataset, say x = p_T and y = eta for some SM sample

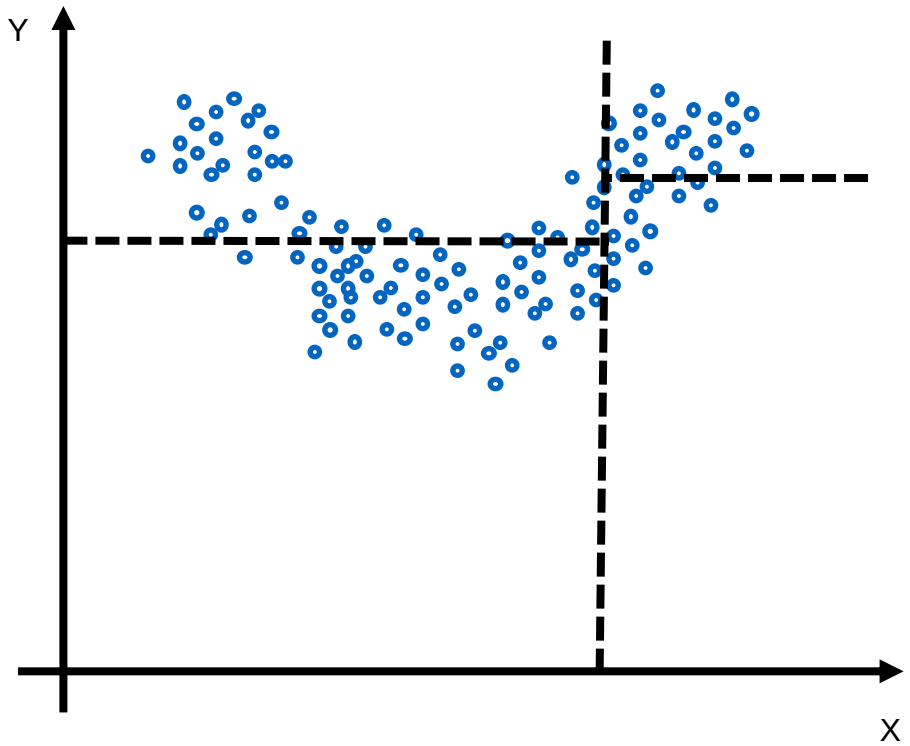


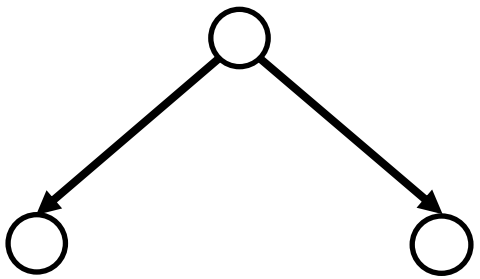
Decision tree autoencoders

Stephen Roche

Training philosophy (novel method described in paper)

- Place small “bins” around locations of high event density
- Example
  - 2d toy dataset, say x = p_T and y = eta for some SM sample



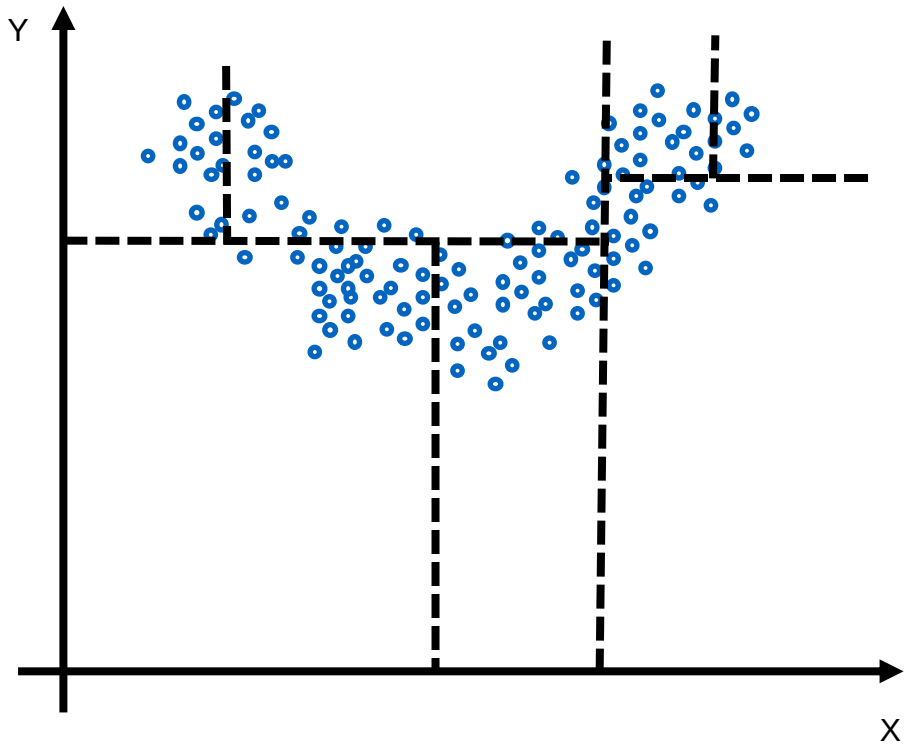


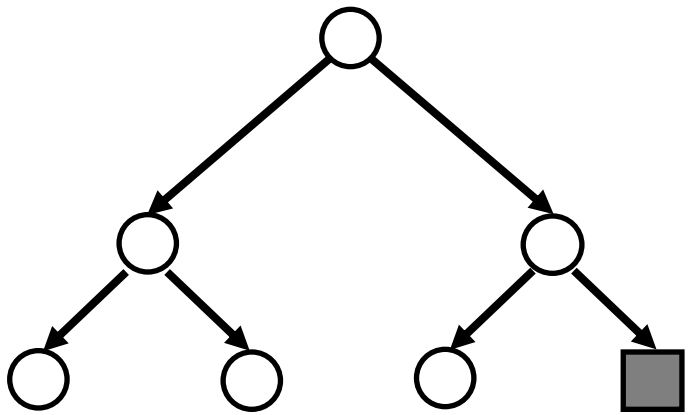
Decision tree autoencoders

Stephen Roche

Training philosophy (novel method described in paper)

- Place small “bins” around locations of high event density
- Example
  - 2d toy dataset, say x = p_T and y = eta for some SM sample



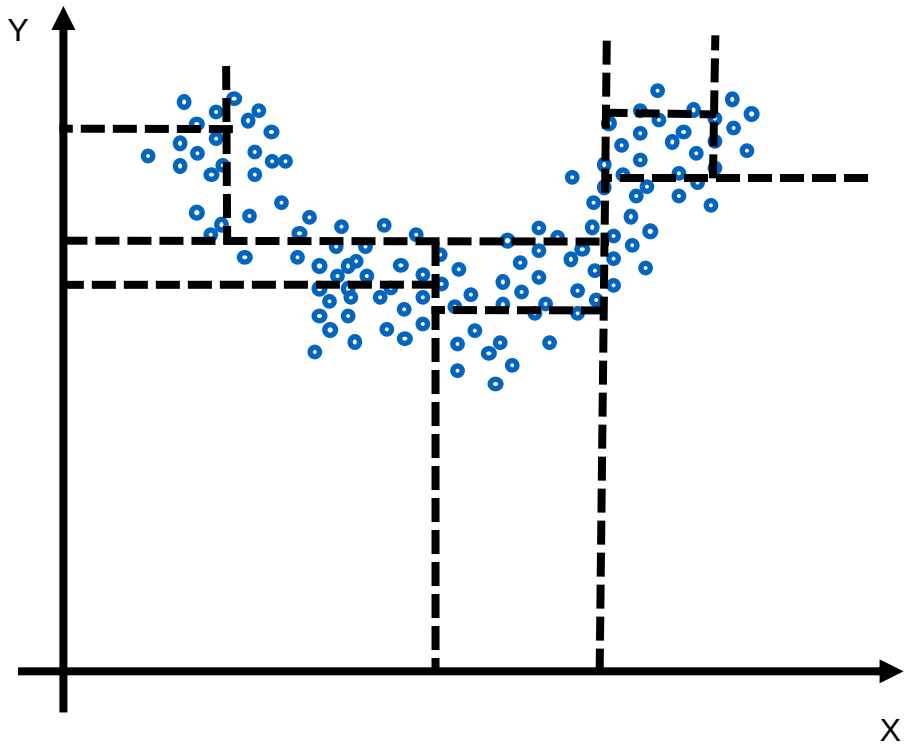


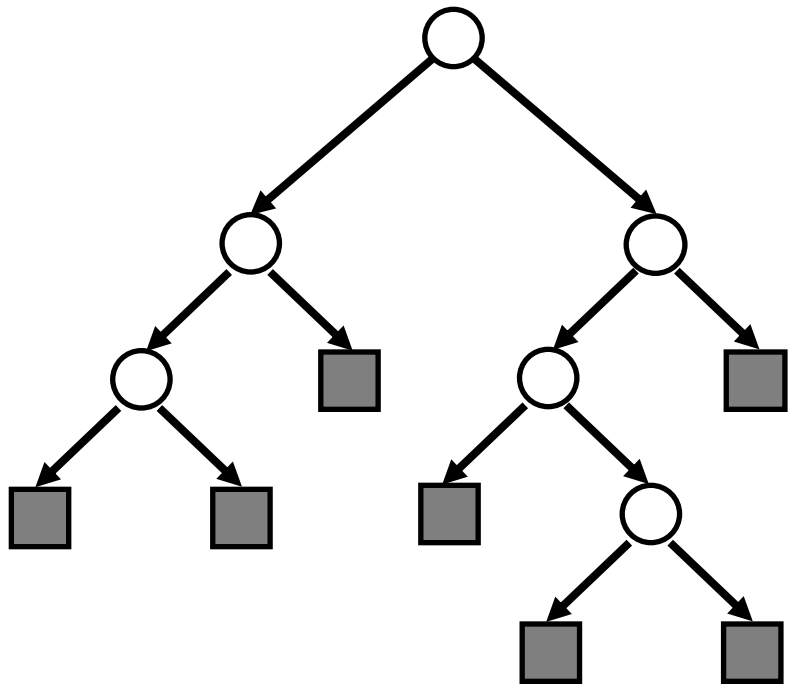
Decision tree autoencoders

Stephen Roche

Training philosophy (novel method described in paper)

- Place small “bins” around locations of high event density
- Example
  - 2d toy dataset, say x = p_T and y = eta for some SM sample





Decision tree autoencoders

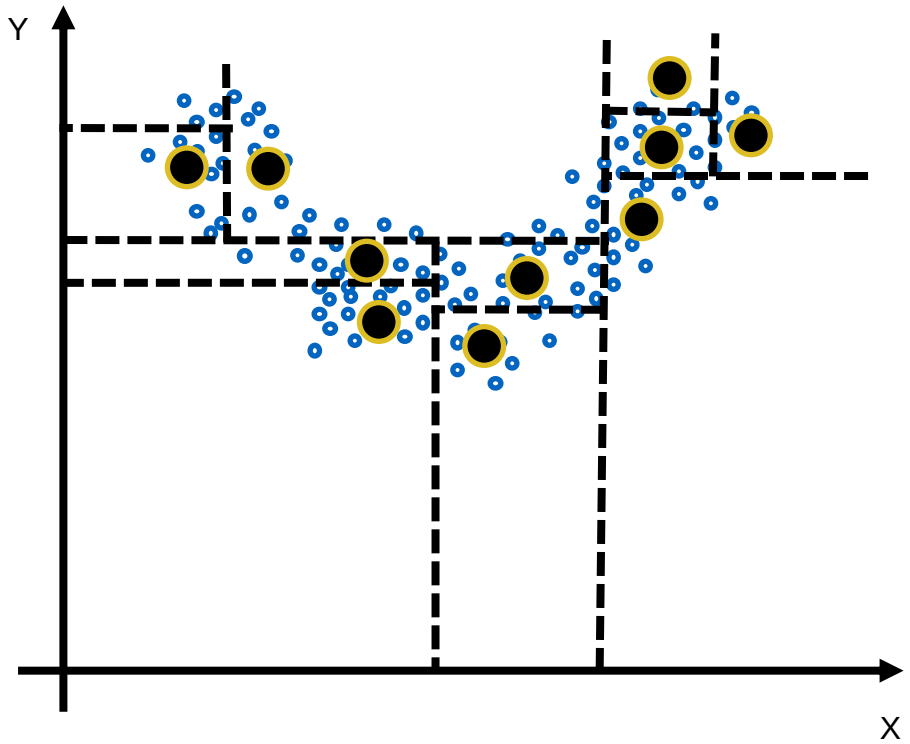
Stephen Roche

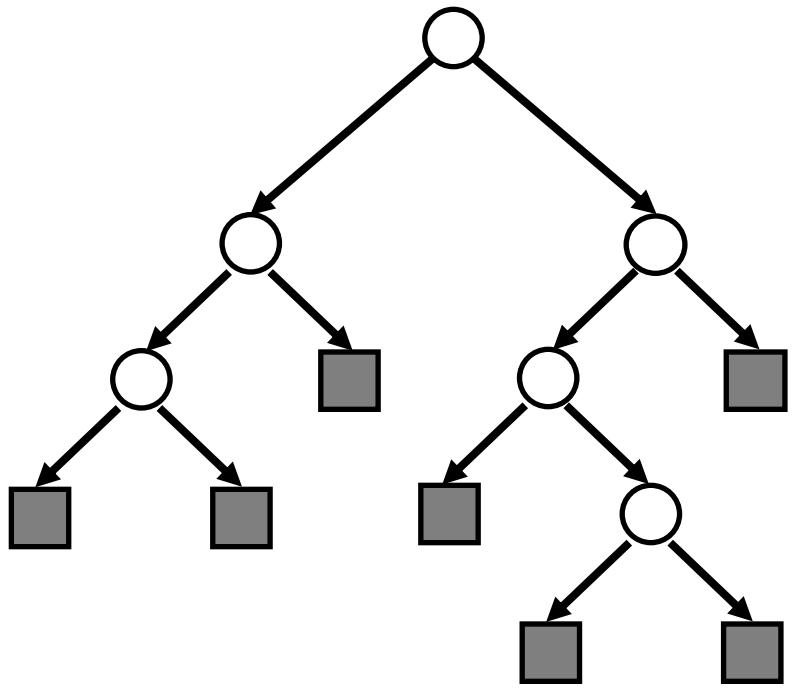
Latent space is bin number

- Encoding: Event → which bin it's in

Decode by returning a "reconstruction point"

- Decoding: Bin → median of the training data in bin



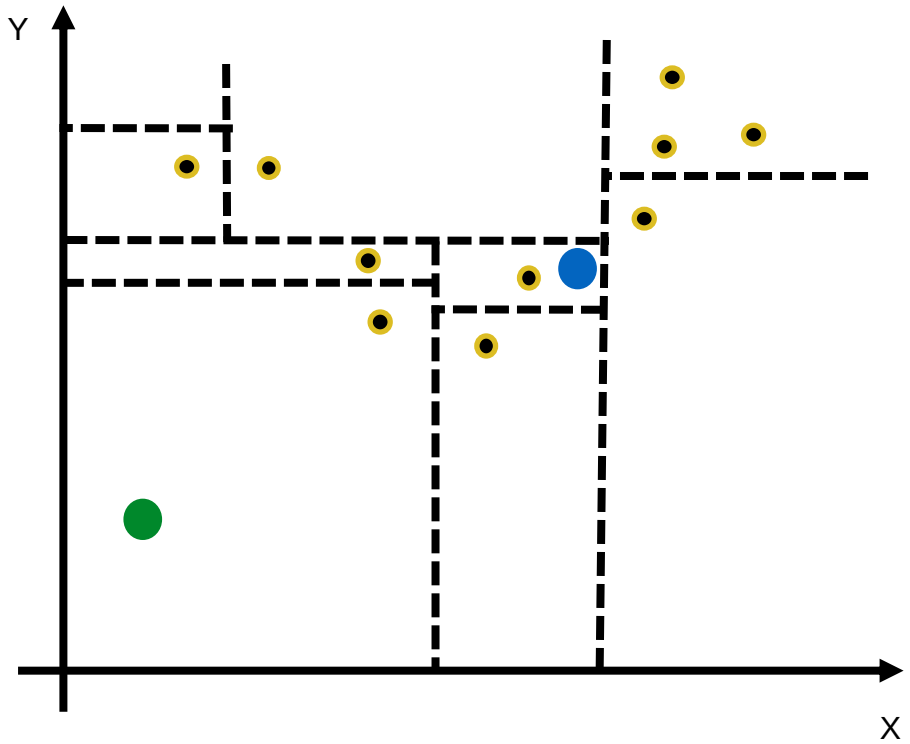


Decision tree autoencoders

Stephen Roche

How does this detect anomalies?

- Define: Distance between input – output = anomaly score

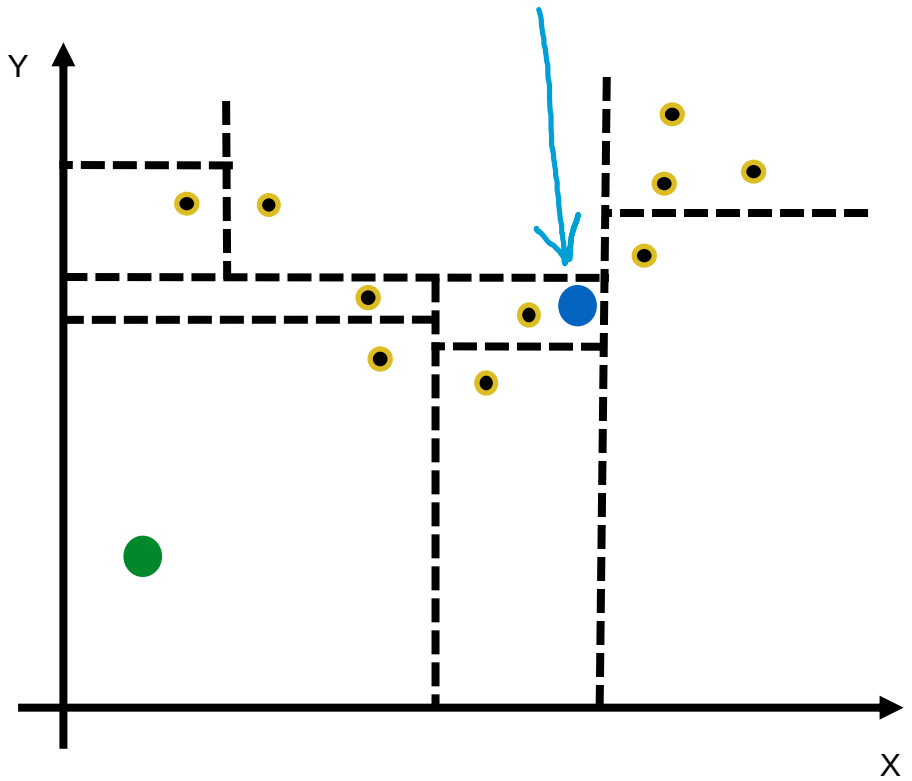


Decision tree autoencoders

Stephen Roche

How does this detect anomalies?

- Define: Distance between input – output = anomaly score
- Non-anomaly
  - Input is similar to training data
  - Will likely land in a **small bin** → close to reconstruction point

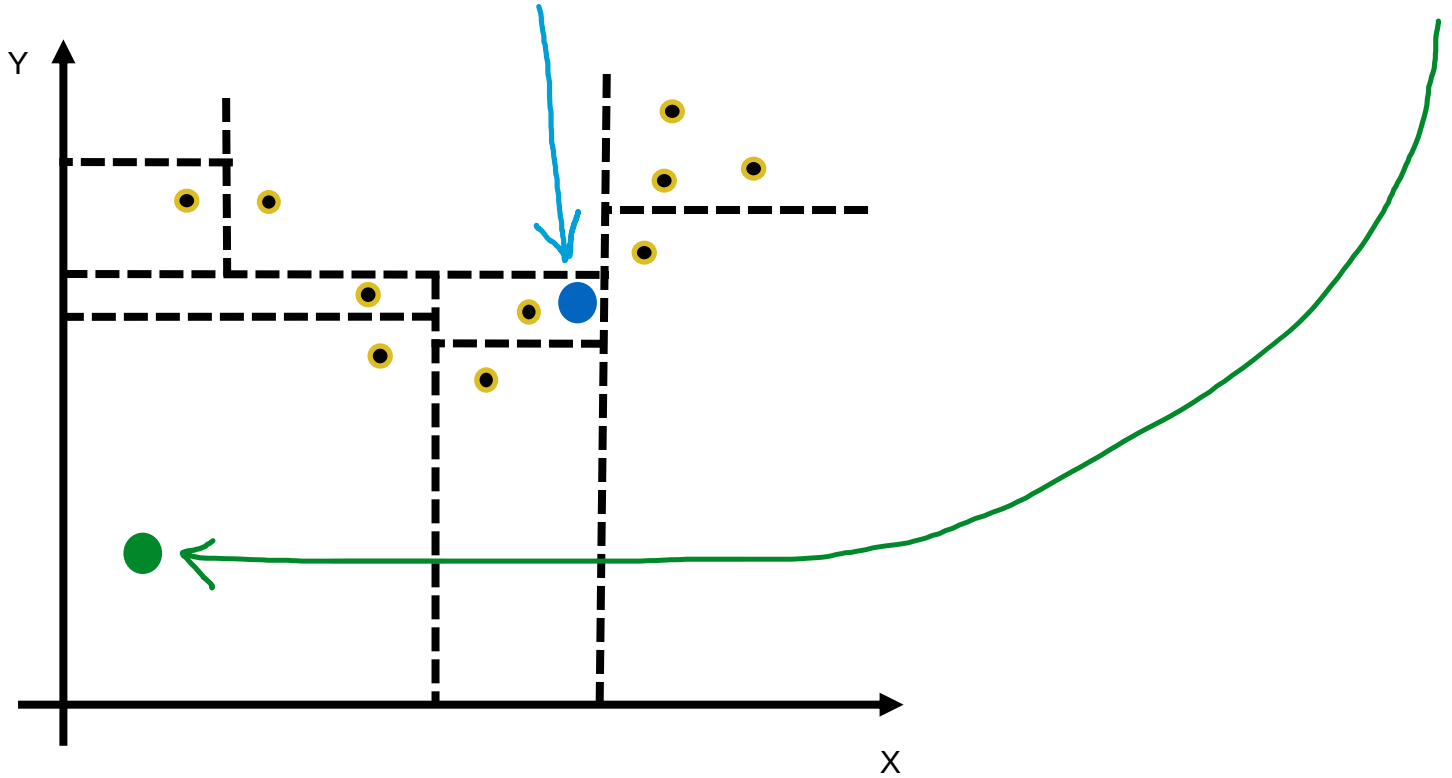


Decision tree autoencoders

Stephen Roche

How does this detect anomalies?

- Define: Distance between input – output = anomaly score
- Non-anomaly
  - Input is similar to training data
  - Will likely land in a **small bin** → close to the reconstruction point
- Anomaly
  - Input is not similar to training data
  - Will likely land in a **large bin** → far from the reconstruction point



Outline

Introduction

- Autoencoders for anomaly detection
- Machine learning at L1

Decision tree autoencoder

- Novel training method

Firmware design

- Novel latent-spaceless design for FPGA

Physics & FPGA results

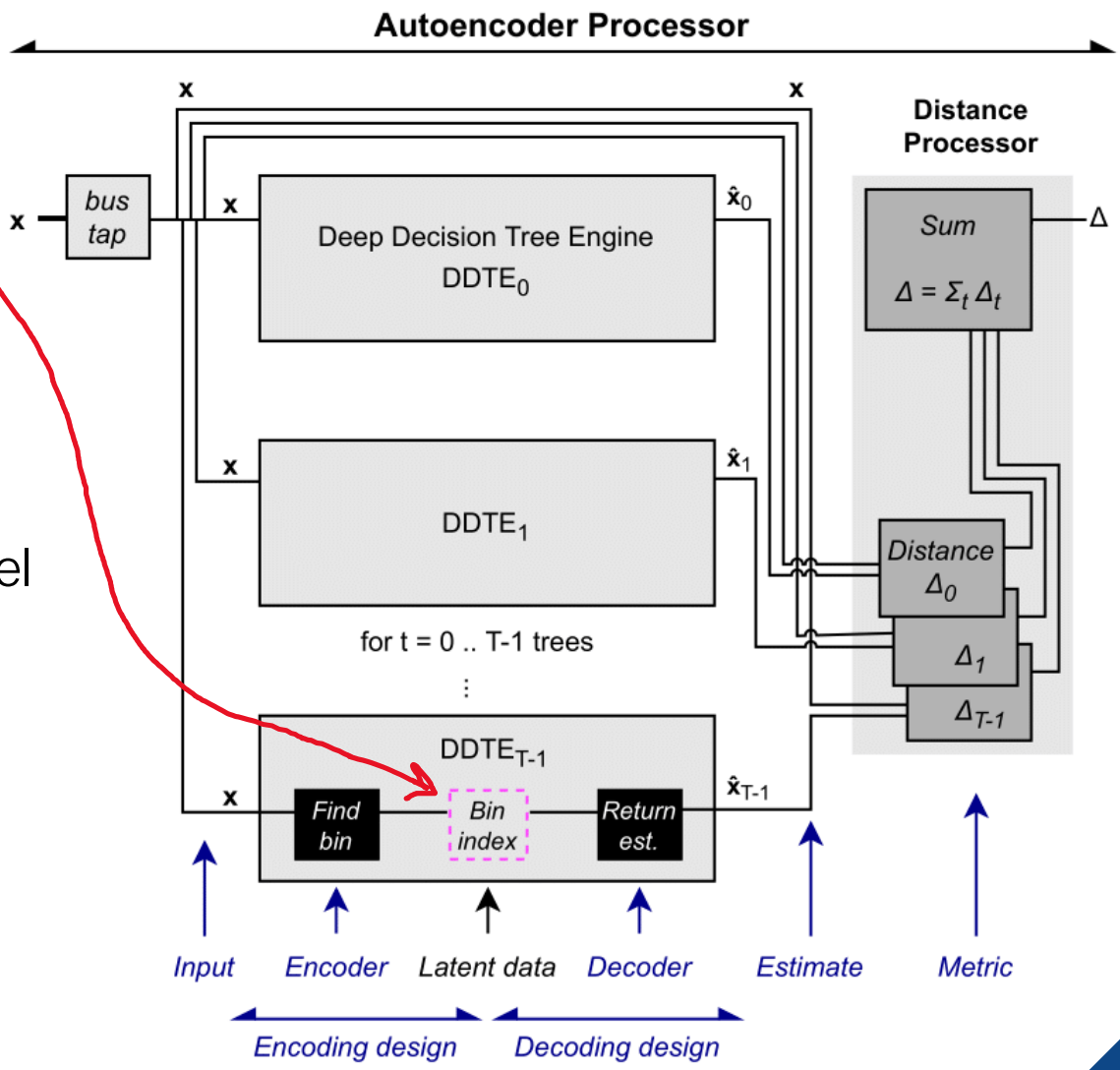
- Exotic decay of Higgs to pseudoscalars to 2e 2μ
- “LHC anomaly detection” dataset

Logic flow

- Block diagram shows left-to-right logic flow (right)
- Encoding is decoding
  - We bypass the latent space!

Details

- Parallel computing
  - TREE ENGINES evaluated in parallel
  - All combinatoric logic, so no clocking between steps = fast
  - Mostly comparisons = fast
  - No multiplication = fast
- Technical info in backup & see [2304.03836]

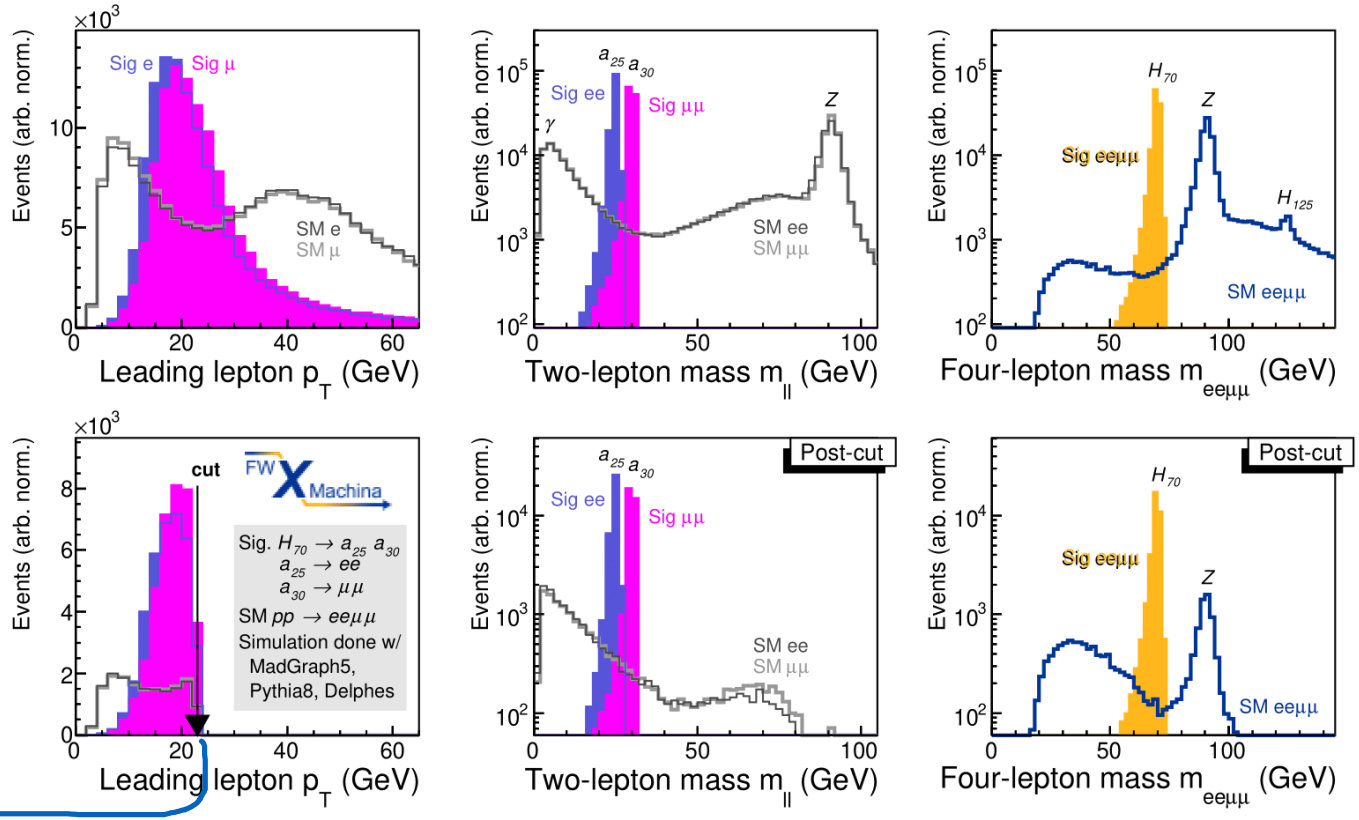


SM 2e 2μ vs. ?

Stephen Roche

Proof of concept problem

- Background: we generate all SM with 2e 2μ (predominantly ZZ*)
- Signal: ggF H→ a_1 a_2 → e^+ e^- μ^+ μ^- (different m_H & m_a)



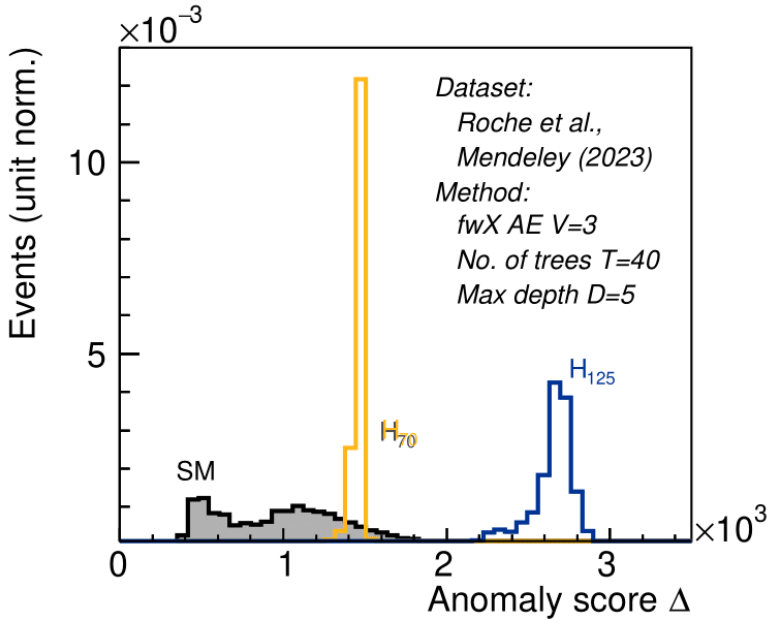
Veto events with lepton p_T > 23 GeV

- Consider only events that won't be already captured by L1 trigger

SM 2e 2μ vs. ?

Proof of concept problem

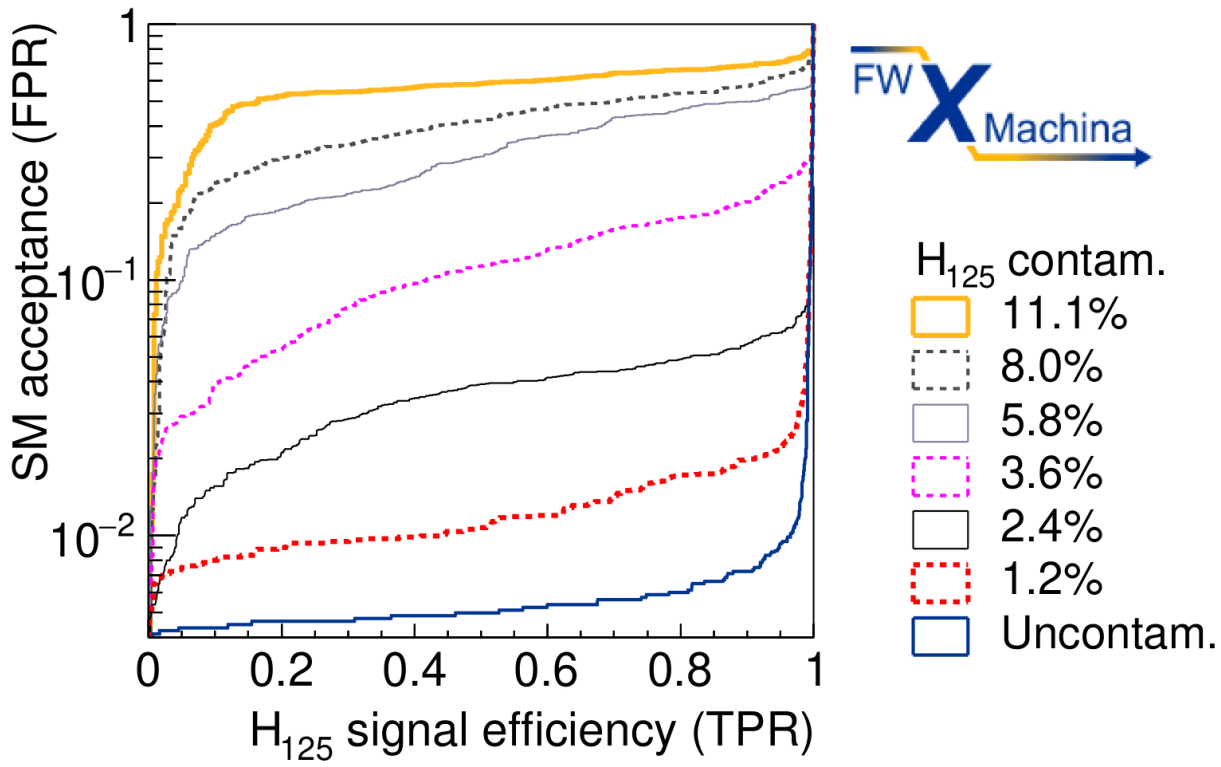
- Design
  - 40 decision trees with maximum depth of 5
  - 3 variables: m_{ee}, $m_{\mu\mu}$, m_{4l}
- Physics results (see figure)
  - Great separation for H_{125}
  - May need a “window selection” for H_{70}
- FPGA results (see table)
  - Latency within 25 ns = 1 BC
  - Percent-level (or smaller) resource usage
  - No multiplications!



Parameter	Value
Clock speed	320 MHz
Latency	8 ticks (25 ns)
Interval	1 tick (3.125 ns)
FF	10k (0.4 %)
LUT	31k (2.6%)
DSP	3 (0.04%)
BRAM	0

Can we train the AE with real data?

- But real data would contain the signal you're looking for…
  - Study the results for different levels of signal contamination in training



Can we train the AE with real data?

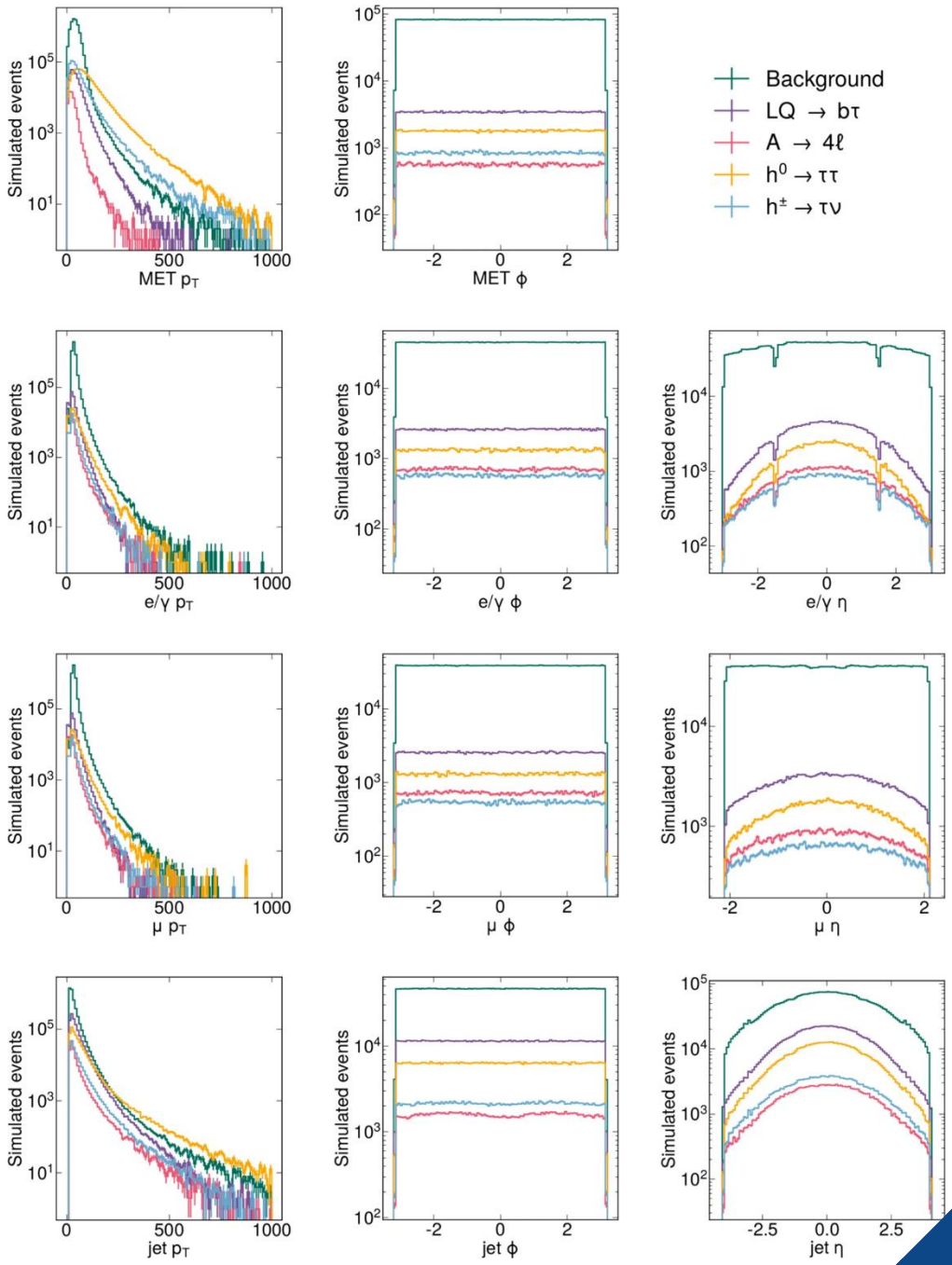
- Looks reasonable for percent-level contamination

Cross-check with public data

Stephen Roche

We try the "LHC anomaly detection dataset" [Sci Data 9, 118]

- Background
  - W → l ν, Z → l l, multijet, ttbar
- Signal
  - 4 BSM scenarios
- Input variables
  - 54 variables
  - p_T, η, φ of the 4 leading μ, 4 leading e, 10 leading jets, MET
  - See distributions on the right
- Sample selection
  - Require ≥1 lepton w/ p_T > 23 GeV

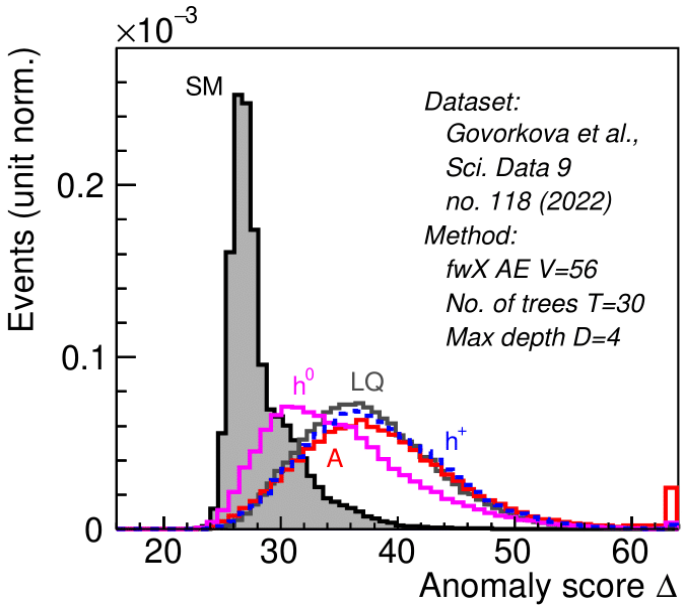


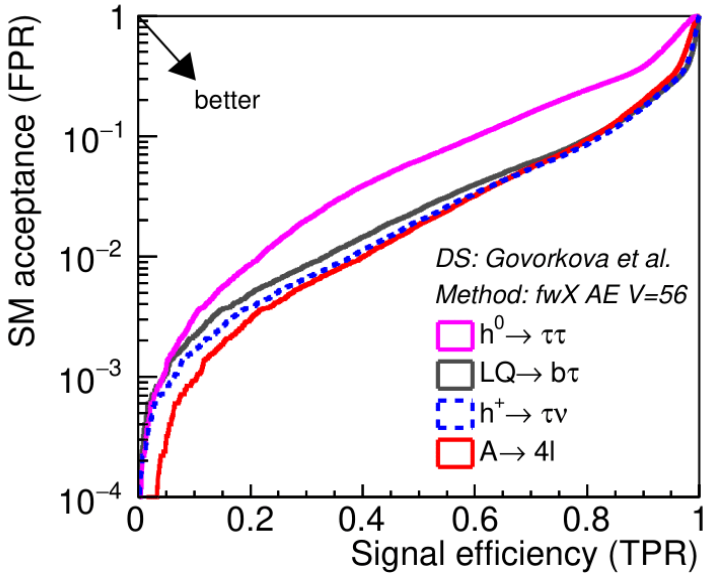
Cross-check with public data

Stephen Roche

It works

- Physics (plots)
- FPGA (table)





Comparison to prev. results

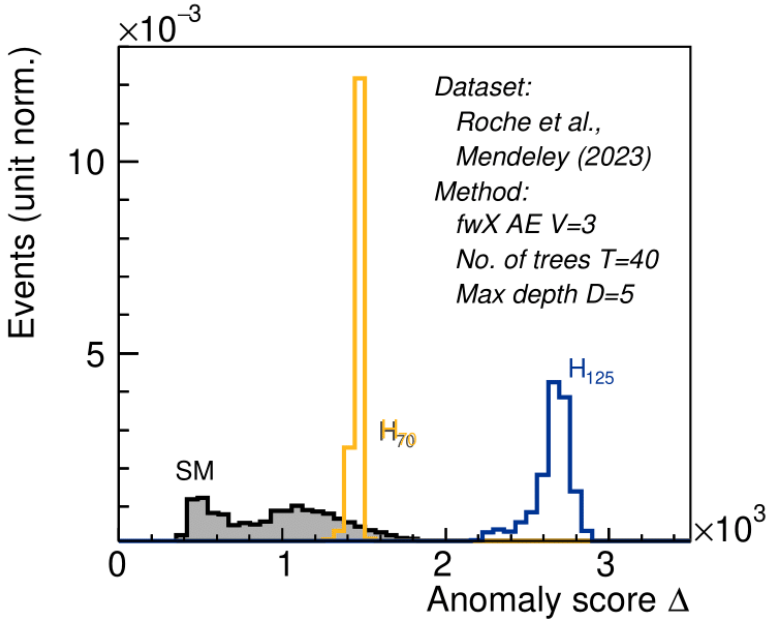
- Hls4ml-based neural network AE
  [Nature Mach. Intell. 4 (2022) 154–161]
- Physics: similar AUC values
- FPGA (range since 4 designs)
  - Latency: 80 – 1480 ns
  - FF: 0.5 – 5%
  - LUT: 3 – 47%
  - DSP: 1 – 8%
  - BRAM: 0.6 – 6%

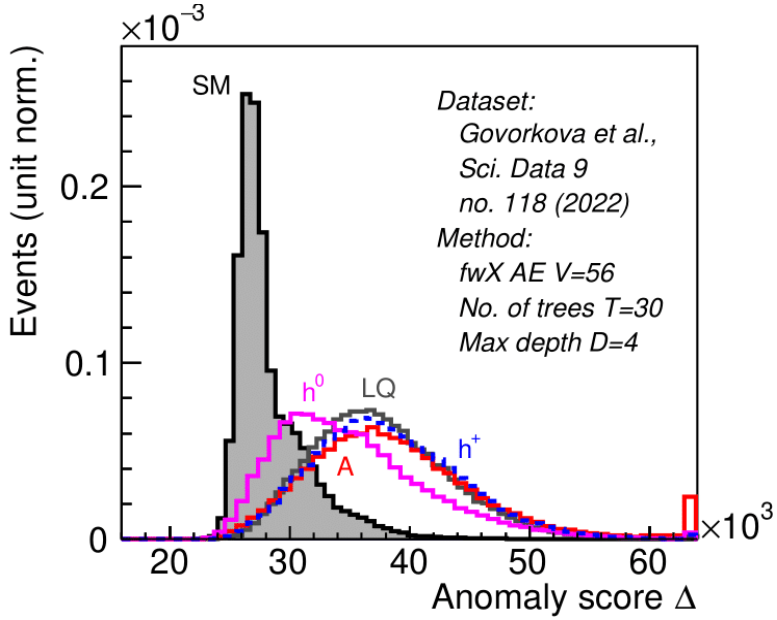
Parameter	Value
Clock speed	200 MHz
Latency	6 ticks (30 ns)
Interval	1 tick (5 ns)
FF	15k (0.6%)
LUT	109k (9.2%)
DSP	56 (0.8%)
BRAM	0

Conclusions

Fast autoencoder for FPGA

- Decision trees as anomaly detector
  - Novel FPGA design. Without latent space!
  - Novel training method. It's interpretable!
- Performance
  - Good physics separation, FPGA results
  - Resistant to percent-level signal contamination





Questions?

- Preprint [2304.03836]
- More info at http://fwx.pitt.edu/
  - Emails stephen.roche@health.slu.edu, bcarlson@westmont.edu, tmhong@pitt.edu
  - Testbench at http://d-scholarship.pitt.edu/44431/

Backup slides

Decision tree autoencoders

Stephen Roche

Goal: use decision trees as autoencoders in order to use the existing fwX framework to evaluate them on FPGAs

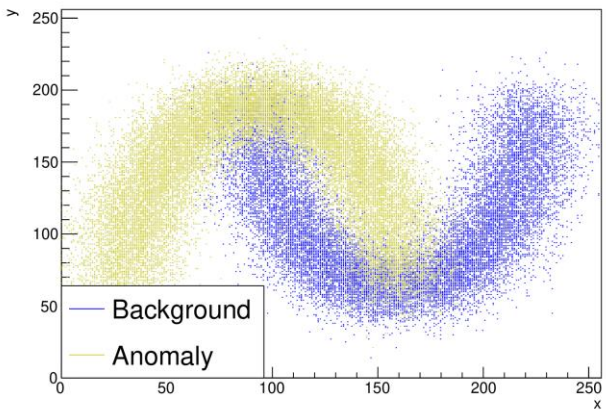
- Few decision-tree based anomaly detectors exist [1709.09018], [2301.00880] and are not designed for particle physics applications or FPGA optimization

Solution: develop our own training algorithm for training decision tree autoencoders

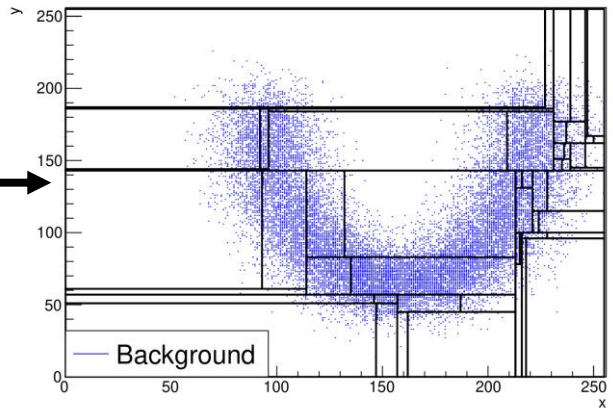
- Training done entirely on background

Decision tree autoencoders

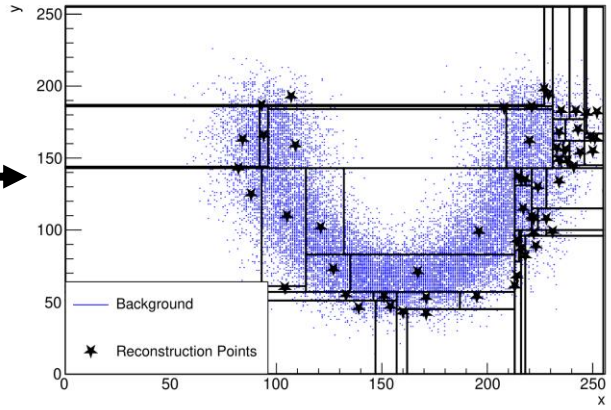
Stephen Roche



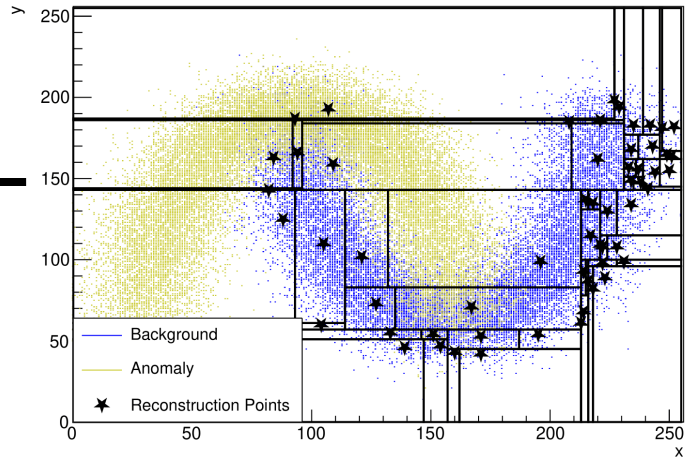
Consider toy dataset



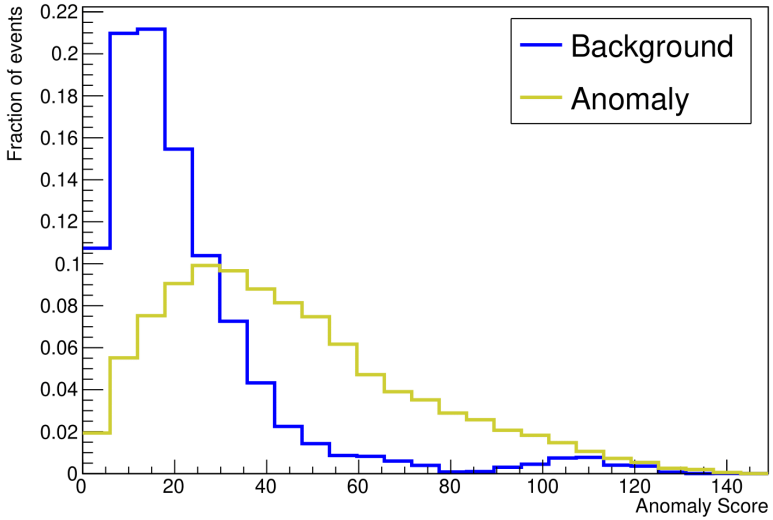
Train on background



Find reconstruction points



Evaluate on background & anomaly



Examine anomaly scores

Decision tree autoencoders

Stephen Roche

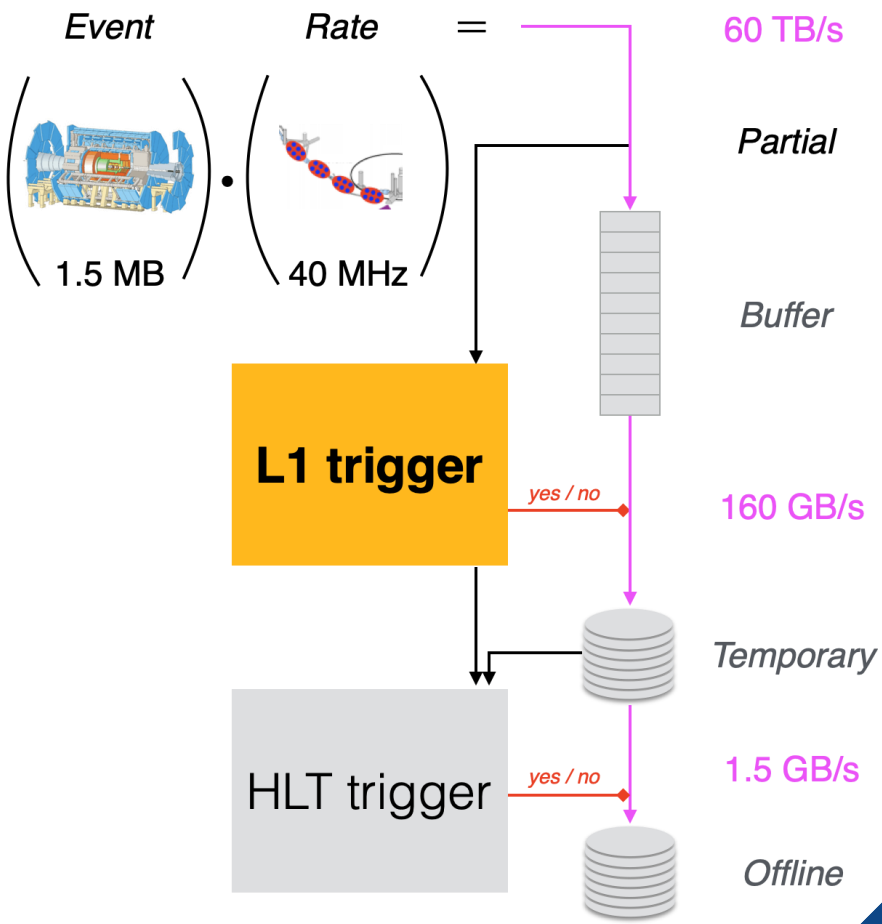
There are some advantages to using decision trees for anomaly detection

- Interpretability: it's easier to understand why a given event was classified as anomalous
- Physics performance: works on MNIST, toy datasets, and the 2 physics datasets in the coming slides
- Can be readily implemented on FPGA with the fwX framework: next slides

Stephen Roche

Deploying AE at the L1 trigger is challenging

- Latency requirements: O(0.1 – 1)μs since 25ns bunch crossing rate
- Use custom electronics such as field programmable gate arrays (FPGA) with optimized algorithms for parallelization



FWXMACHINA is a Python package

- Developed by us
- Takes decision trees trained in software to create FPGA design
  - For Classification, JINST 16 (2021) P08016
  - For Regression, JINST 17 (2022) P09039
  - Today's AE in [2304.03836] (2023)
- Our approach
  - Optimizations made to allow for parallelization, conversion of floating-point values to n-bit integers, and pruning
  - Automatic test-point validation & testbench generation

Literature

- HLS4ML has AE built on
  - neural network, Nature Mach. Intell. 4 (2022) 154–161

Firmware performance

Stephen Roche

Trained models implemented on vu9p FPGA

- Very low latency and interval within constraints of L1 trigger systems
- Relatively little resource usage

	H→2e2mu	LHC anomaly detection
Number of input variables	3	56
Number of trees	40	30
Max depth	5	4
Clock speed	320 MHz	200 MHz
Latency	8 ticks (25 ns)	6 ticks (30 ns)
Interval	1 tick (3.125 ns)	1 tick (5 ns)
FF	10k (0.4 %)	15k (0.6%)
LUT	31k (2.6%)	109k (9.2%)
DSP	3 (0.04%)	56 (0.8%)
BRAM	0	0