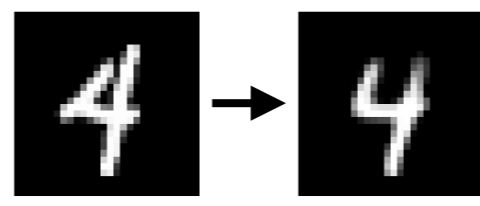
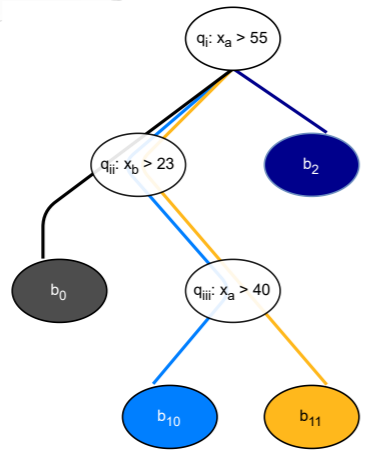


Tae Min Hong



University of Pittsburgh



autoencoder



on FPGA

Decision tree



Streaming Readout Workshop SRO-XII

December 3, 2024

<https://indico.bnl.gov/event/24286/contributions/99100/>

# Outline

## Introduction

- Who we are

## FPGA design

- Autoencoder
- Parallelizing decision trees
- HLS trees → VHDL trees

## Thoughts on SRO

- Data compression
- Anomaly detection

## More info

- Code structure & git
- Slides & video tutorials



**Nanosecond machine learning event classification with boosted decision trees in FPGA for high energy physics**  
2021 JINST 16 P0016

**Nanosecond machine learning regression with deep boosted decision trees in FPGA for high energy physics**  
2022 JINST 17 P0039

**Nanosecond anomaly detection with decision trees and real-time application to exotic Higgs decays**  
nature communications

**fwX Machine Project**

**What is fwX**

- Its full name is "firmware ex machina," a play of the phrase in Latin / Greek *deus ex machina* / *ἄθεός ἐκ μηχανῆς*. Since it's a mouthful to say, we refer to it as fwX.
- It is a software package to design nanosecond implementation of machine learning / artificial intelligence algorithms on FPGA for use in high energy physics.

**Some figures**

Figure	Caption
S1	Illustrative example of $\ast$ coder as two visual representations of the same decision tree. Deep decision tree (left) rendered as the decision tree grid (center) and implemented by the parallel decision paths (right). Two-deep deep decision tree (DDT) is the encoder (step 1) shown as a conventional binary split diagram; the latent space is the bin number (step 2); the latent space data is decoded using the decision tree grid (DTG) (step 3); and the simultaneous encoding and decoding with $\ast$ coder (star-coder) architecture (right) represented by parallel decision paths (PDP) of Ref. [19]. The DTG is the visualization as a grid of partitions in $n$ -dimensional space. In this example, the input $x = (55, 70)$ yields the output $x' = (27, 25)$ without needing to explicitly produce the latent layer.
S2	Demonstration of decision tree-based autoencoder and a demonstration of data transmission / anomaly detection using the MNIST dataset, which is a set of images of handwritten numbers converted to $28 \times 28$ pixels, or 784-length input vector $y = 784$ , with $n = 8$ bits per pixel. The ML training is done on 15k images of handwritten 0 to 4, but not 5 to 9, on one tree $T = 1$ at a maximum depth of $D = 20$ . The output is a 784-length vector with 8 bits per pixel. The data compression factor, the ratio of input-output bits to the latent space dimensions, $\ast$ -MCF, $\ast$ -MCF = $784 / 20 \approx 39.2$ , is about 200. The figure shows two input-output pairs as examples. The output of 4 resembles 4 while the output of 6 is garbled. The former yields a smaller input-output distance relative to the latter case. The input data shown here are not part of the training sample.



University of Pittsburgh

Undergraduate students

Steve Roche	Dan Stumpp	Steve Racz	Aleksa Rodic	Quincy Bayer	Will Ouligian	Pavel Serhiayenka	Santiago Cane	Kyle Mo
lead for all sw	lead for v1.0 fw	lead bit shifter	studies vs. hls4ml	lead for v2.0 fw	lead AE trainer	lead for HDL fw	lead ATLAS integration	testing

Non students

Brandon Eubanks	Emre Ercikti	Yuvaraj Elangovan	Isabelle Taylor	Joerg Stelzer	Rajat Gupta	Ben Carlson	Tae Min Hong
testbench support	testbench support	testbench support	testbench support	algorithm support	physics support	physics lead	project lead

[tmhong@pitt.edu](mailto:tmhong@pitt.edu)

Skip this slide

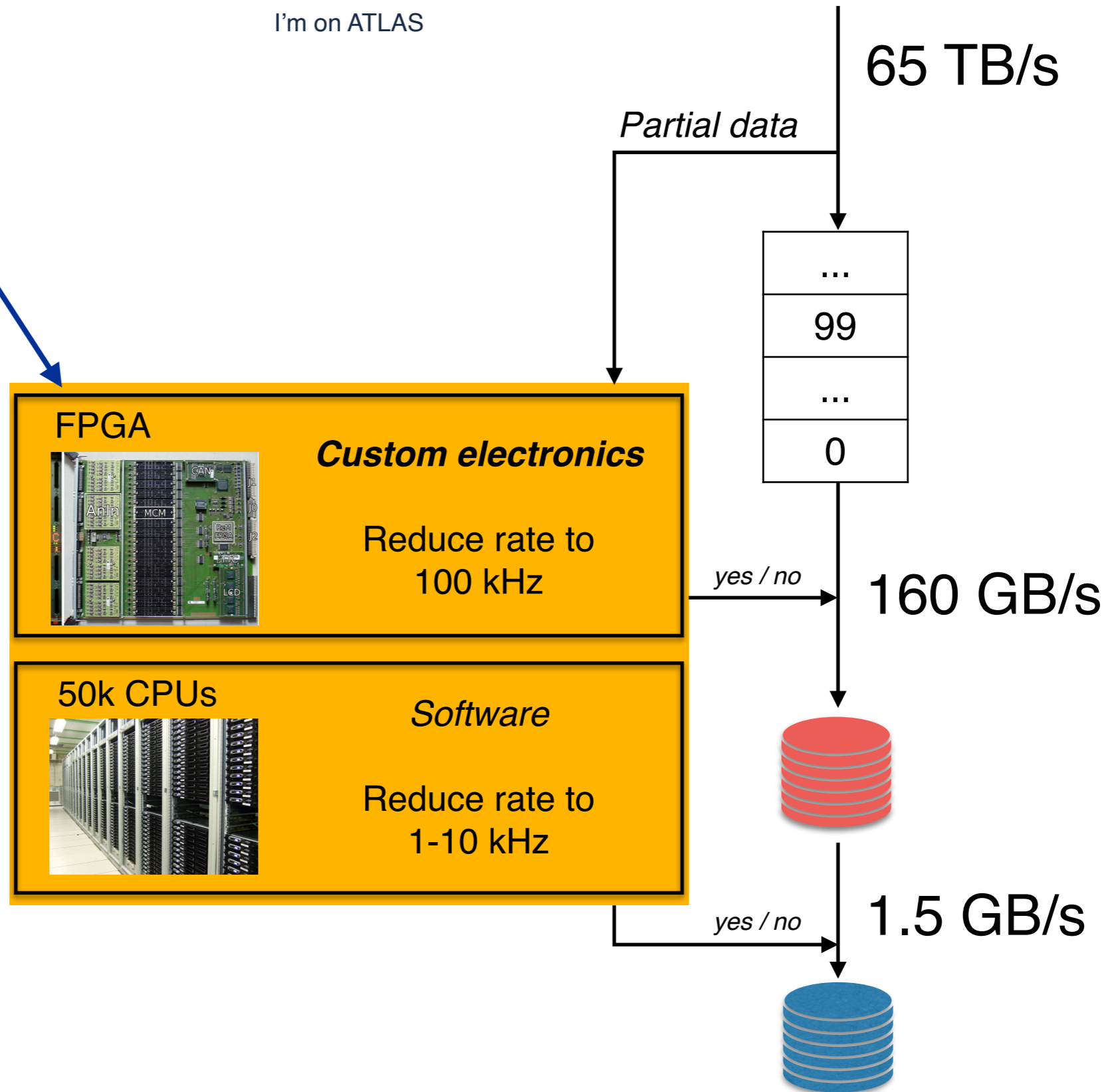
# targets LHC: BUT TOOL IS GENERAL

I'm on ATLAS

*Level Latency*

**L1**    **25 ns - 1 μs**

**HLT**    **O(1) s**



## 1. Classification

parallel cuts using HLS

## 2. Regression

parallel paths using HLS

## 3. Autoencoder

in-house training  
bypass latent space

## 4. Hardware trees

faster & more efficient  
no more HLS

*Jinst* PUBLISHED BY IOP PUBLISHING FOR SISSA MEDIALAB

RECEIVED: April 9, 2021  
ACCEPTED: June 29, 2021  
PUBLISHED: August 4, 2021

### Nanosecond machine learning event classification with boosted decision trees in FPGA for high energy physics

T.M. Hong,<sup>a</sup> B.T. Carlson, B.R. Eubanks, S.T. Racz, S.T. Roche, J. Stelzer and D.C. Stumpp

<sup>a</sup>Department of Physics and Astronomy, University of Pittsburgh, 100 Allen Hall, 3941 O'Hara St., Pittsburgh, PA 15260, U.S.A.  
E-mail: tmhong@pitt.edu

ABSTRACT: We present a novel implementation of classification using the machine learning/artificial intelligence method called boosted decision trees (BDT) on field programmable gate arrays (FPGA). The firmware implementation of binary classification requiring 100 training trees with a maximum depth of 4 using four input variables gives a latency value of about 10 ns, independent of the clock speed from 100 to 320 MHz in our setup. The low timing values are achieved by restructuring the BDT layout and reconfiguring its parameters. The FPGA resource utilization is also kept low at a range from 0.01% to 0.2% in our setup. A software package called **FWXACHINA** achieves this implementation. Our intended user is an expert in custom electronics-based trigger systems in high energy physics experiments or anyone that needs decisions at the lowest latency values for real-time event classification. Two problems from high energy physics are considered, in the separation of electrons vs. photons and in the selection of vector boson fusion-produced Higgs bosons vs. the rejection of the multijet processes.

KEYWORDS: Digital electronic circuits; Trigger algorithms; Trigger concepts and systems (hardware and software); Data reduction methods

ARXIV EPRINT: 2104.03408

\*Corresponding author.

© 2021 IOP Publishing Ltd and Sissa Medialab <https://doi.org/10.1088/1748-0221/16/08/P08016>

2021 JINST 16 P08016

*Jinst* PUBLISHED BY IOP PUBLISHING FOR SISSA MEDIALAB

RECEIVED: July 13, 2022  
ACCEPTED: August 23, 2022  
PUBLISHED: September 27, 2022

### Nanosecond machine learning regression with deep boosted decision trees in FPGA for high energy physics

B.T. Carlson,<sup>a,b</sup> Q. Bayer,<sup>b</sup> T.M. Hong<sup>b,\*</sup> and S.T. Roche<sup>b</sup>

<sup>a</sup>Department of Physics and Engineering, Westmont College, 955 La Paz Road, Santa Barbara, CA 93108, U.S.A.  
<sup>b</sup>Department of Physics and Astronomy, University of Pittsburgh, 100 Allen Hall, 3941 O'Hara St., Pittsburgh, PA 15260, U.S.A.  
E-mail: tmhong@pitt.edu

ABSTRACT: We present a novel application of the machine learning / artificial intelligence method called boosted decision trees to estimate physical quantities on field programmable gate arrays (FPGA). The software package **FWXACHINA** features a new architecture called parallel decision paths that allows for deep decision trees with arbitrary number of input variables. It also features a new optimization scheme to use different numbers of bits for each input variable, which produces optimal physics results and ultraefficient FPGA resource utilization. Problems in high energy physics of proton collisions at the Large Hadron Collider (LHC) are considered. Estimation of missing transverse momentum ( $E_T^{\text{miss}}$ ) at the first level trigger system at the High Luminosity LHC (HL-LHC) experiments, with a simplified detector modeled by Delphes, is used to benchmark and characterize the firmware performance. The firmware implementation with a maximum depth of up to 10 using eight input variables of 16-bit precision gives a latency value of  $O(10)$  ns, independent of the clock speed, and  $O(0.1\%)$  of the available FPGA resources without using digital signal processors.

KEYWORDS: Data reduction methods; Digital electronic circuits; Trigger algorithms; Trigger concepts and systems (hardware and software)

ARXIV EPRINT: 2207.05602

\*Corresponding author.

© 2022 IOP Publishing Ltd and Sissa Medialab <https://doi.org/10.1088/1748-0221/17/09/P09039>

2022 JINST 17 P09039

nature communications

Article <https://doi.org/10.1038/s41467-024-47704-8>

### Nanosecond anomaly detection with decision trees and real-time application to exotic Higgs decays

S. T. Roche<sup>1,2</sup>, Q. Bayer<sup>2</sup>, B. T. Carlson<sup>2,3</sup>, W. C. Ouljian<sup>2</sup>, P. Serhiyenko<sup>2</sup>, J. Stelzer<sup>2</sup> & T. M. Hong<sup>2</sup>\*

Received: 23 May 2023  
Accepted: 9 April 2024  
Published online: 25 April 2024

Check for updates

We present an interpretable implementation of the autoencoding algorithm, used as an anomaly detector, built with a forest of deep decision trees on FPGA, field programmable gate arrays. Scenarios at the Large Hadron Collider at CERN are considered, for which the autoencoder is trained using known physical processes of the Standard Model. The design is then deployed in real-time trigger systems for anomaly detection of unknown physical processes, such as the detection of rare exotic decays of the Higgs boson. The inference is made with a latency value of 30 ns at percent-level resource usage using the Xilinx Virtex UltraScale+ VU9P FPGA. Our method offers anomaly detection at low latency values for edge AI users with resource constraints.

Unsupervised artificial intelligence (AI) algorithms enable signal-agnostic searches beyond the Standard Model (SM) physics at the Large Hadron Collider (LHC) at CERN. The LHC is the highest energy proton and heavy ion collider that is designed to discover the Higgs boson<sup>1</sup> and study its properties<sup>2</sup> as well as to probe the unknown and undiscovered BSM physics (see, e.g.,<sup>3</sup>). Due to the lack of signs of BSM in the collected data despite the plethora of searches conducted at the LHC, dedicated studies look for rare BSM events that are even more difficult to parse among the mountain of ordinary Standard Model processes<sup>4</sup>. An active area of AI research in high energy physics is in using autoencoders for anomaly detection, much of which provides methods to find rare and unanticipated BSM physics. Much of the existing literature, mostly using neural network-based approaches, focuses on identifying BSM physics in already collected data<sup>5–7</sup>. Such ideas have started to produce experimental results on the analysis of data collected at the LHC<sup>8–10</sup>. A related but separate endeavor, which is the subject of this paper, is enabling the identification of rare and anomalous data on the real-time trigger path for more detailed investigation offline.

The LHC offers an environment with an abundance of data at a 40 MHz collision rate, corresponding to the 25 ns time period between successive collisions. The real-time trigger path of the ATLAS and CMS experiments<sup>11,12</sup>, e.g., processes data using custom electronics using field programmable gate arrays (FPGA) followed by software trigger algorithms executed on a computing farm. The first-level FPGA portion of the trigger system accepts between 100 kHz to 1 MHz of collisions, discarding the remaining ~99% of the collisions. Therefore, it is essential to discover that the FPGA-based trigger system is capable of triggering potential BSM events. A previous study aimed at LHC data has shown that an anomaly detector based on neural networks can be implemented on FPGA with latency values between 80 to 1480 ns, depending on the design<sup>13</sup>.

In this paper, we present an interpretable implementation of an autoencoder using deep decision trees that make inferences in 30 ns. As discussed previously<sup>14</sup>, decision tree designs depend only on threshold comparisons resulting in fast and efficient FPGA implementation with minimal reliance on digital signal processors. We train the autoencoder on known Standard Model (SM) processes to help trigger the rare events that may include BSM.

In scenarios for which a specific BSM model is targeted and its dynamics are known, dedicated supervised training against the SM sample, i.e., BSM-vs-SM classification, would likely outperform an unsupervised approach of SM-only training. The physics scenarios considered in this paper are examples to demonstrate that our autoencoder is able to trigger on BSM scenarios as anomalies without this prior knowledge of the BSM specifics. Nevertheless, we consider a benchmark where our autoencoder outperforms the existing conventional cut-based algorithms.

<sup>1</sup>School of Medicine, Saint Louis University, Saint Louis, MO, USA. <sup>2</sup>Department of Physics and Astronomy, University of Pittsburgh, Pittsburgh, PA, USA. <sup>3</sup>Department of Physics and Engineering, Westmont College, Santa Barbara, CA, USA. \*e-mail: tmhong@pitt.edu

Nature Communications | (2024)15:3527

PITT-PACC-2409-v1

### Nanosecond hardware regression trees in FPGA at the LHC

P. Serhiyenko<sup>a</sup>, S. T. Roche<sup>a,b</sup>, B. T. Carlson<sup>a,c</sup>, and T. M. Hong<sup>a,\*</sup>

<sup>a</sup>Department of Physics and Astronomy, University of Pittsburgh  
<sup>b</sup>School of Medicine, Saint Louis University  
<sup>c</sup>Department of Physics and Engineering, Westmont College

September 20, 2024

Abstract

We present a generic parallel implementation of the decision tree-based machine learning (ML) method in hardware description language (HDL) on field programmable gate arrays (FPGA). A regression problem in high energy physics at the Large Hadron Collider is considered: the estimation of the magnitude of missing transverse momentum using boosted decision trees (BDT). A forest of twenty decision trees each with a maximum depth of 10 using eight input variables of 16-bit precision is executed with a latency of about 10 ns using  $O(0.1\%)$  resources on Xilinx UltraScale+ VU9P—approximately ten times faster and five times smaller compared to similar designs using high level synthesis (HLS)—without the use of digital signal processors (DSP) while eliminating the use of block RAM (BRAM). We also demonstrate a potential application in the estimation of muon momentum for ATLAS RPC at HL-LHC.

Keywords: Data processing methods, Data reduction methods, Digital electronic circuits, Trigger algorithms, and Trigger concepts and systems (hardware and software).

\*Corresponding author, tmhong@pitt.edu

1

Hong et al.  
JINST 16, P08016 (2021)  
<http://doi.org/10.1088/1748-0221/16/08/P08016>

Carlson et al.  
JINST 17, P09039 (2022)  
<http://doi.org/10.1088/1748-0221/17/09/P09039>

Roche et al.  
Nat. Comm. 15 (2024) 3527  
<https://arxiv.org/abs/2304.03836>

Serhiyenko et al.  
Submitted to NIM-A  
[2409.20506]

# Outline

## Introduction

- Who we are

## FPGA design

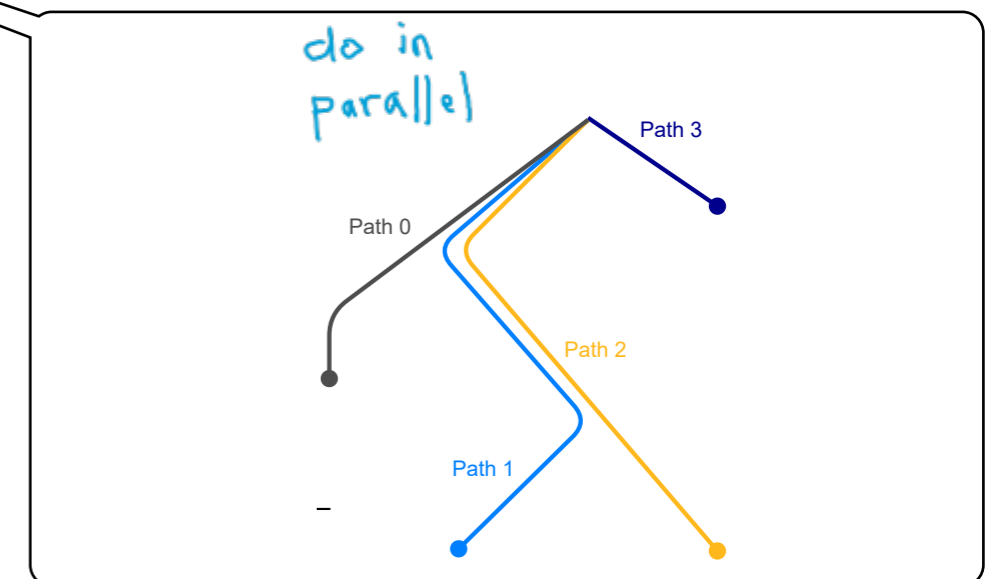
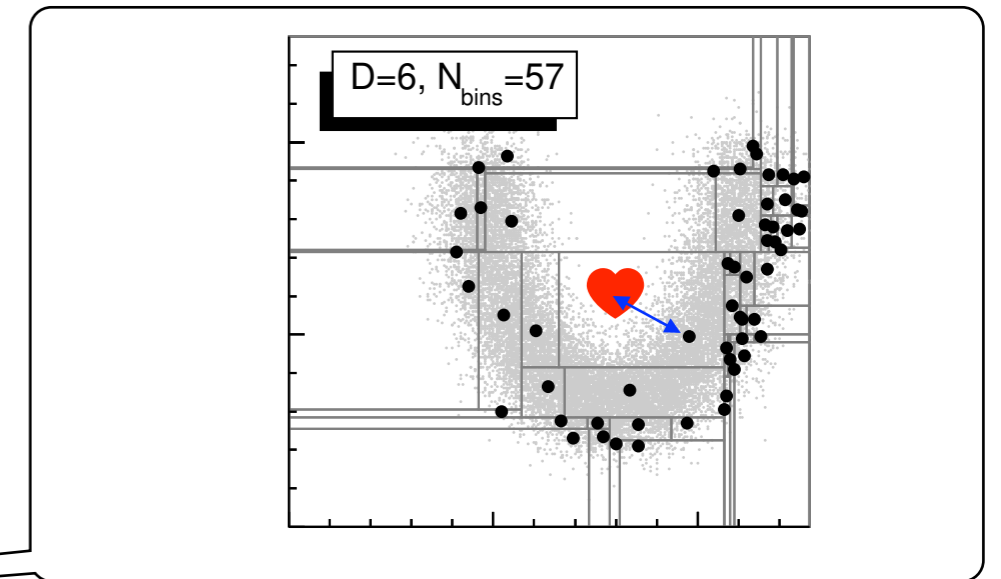
- Autoencoder
- Parallelizing decision trees
- HLS trees → VHDL trees

## Thoughts on SRO

- Data compression
- Anomaly detection

## More info

- Relevant papers from us
- Where to find code, tutorials

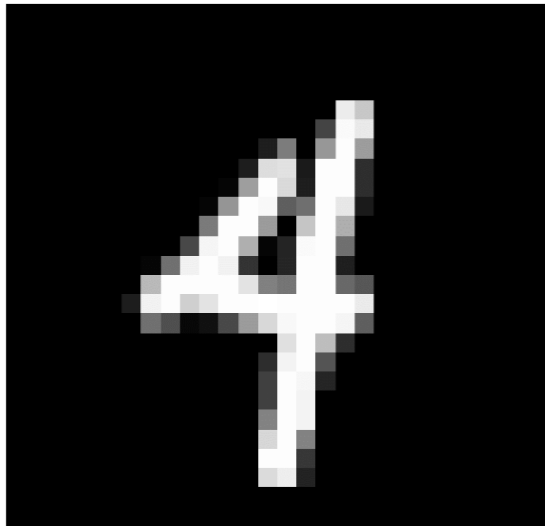




## Example: handwritten numbers

- Teach it 0, 1, 2, 3, 4 with a sample (doesn't know about 9!)

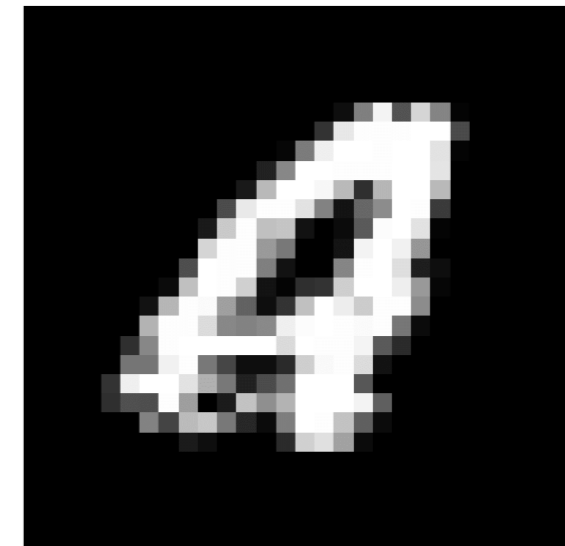
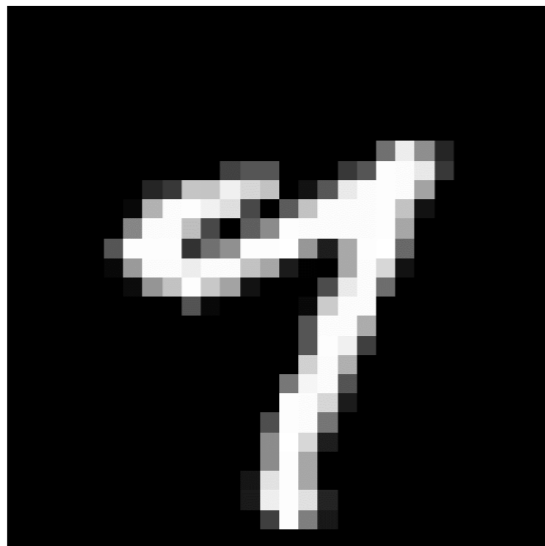
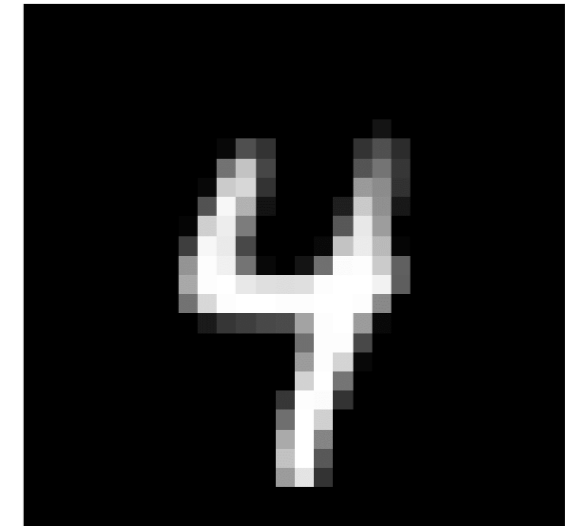
784 variables (8-bit)



1 variable (20 bit)

300x compression

784 variables (8-bit)



## Details

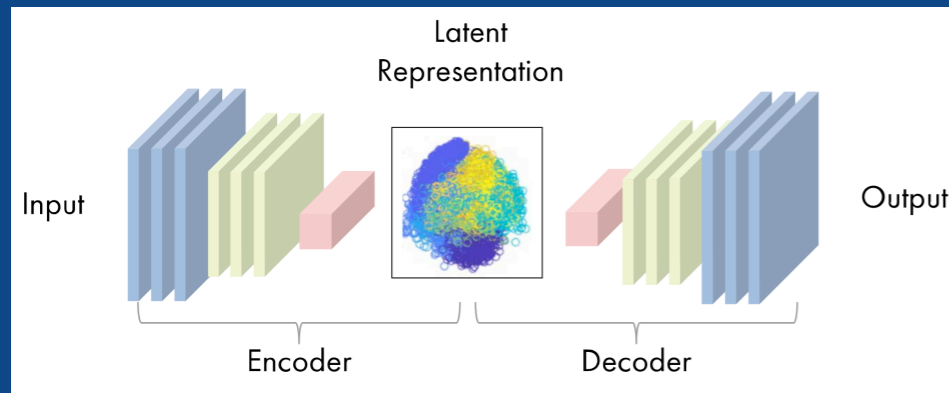
- Input-output distance is relatively **small** = good compression
- Input-output distance is relatively **large** = bad compression

# Tree autoencoder, What?!



## NN AE

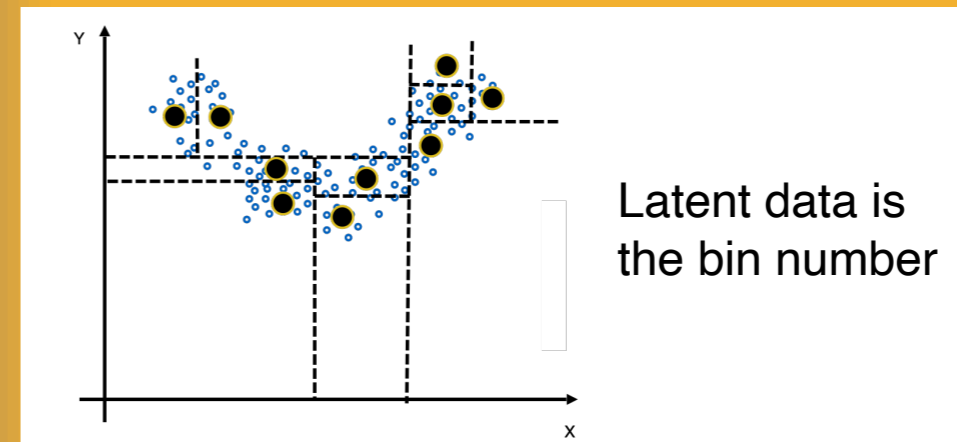
- Training is a black box, done offline
- Latent space is complex



From CMS Machine Learning Group  
<https://cms-ml.github.io/documentation/training/autoencoders.html>

## Tree AE

- Training is sampling of 1d pdfs
- Latent space is simple / interpretable



- FPGA version simplified for anomaly at CMS
- FPGA version can optionally skip latent sp.

From CMS Public Note, DP-2023/079  
[https://cds.cern.ch/record/2876546/files/DP2023\\_079.pdf](https://cds.cern.ch/record/2876546/files/DP2023_079.pdf)

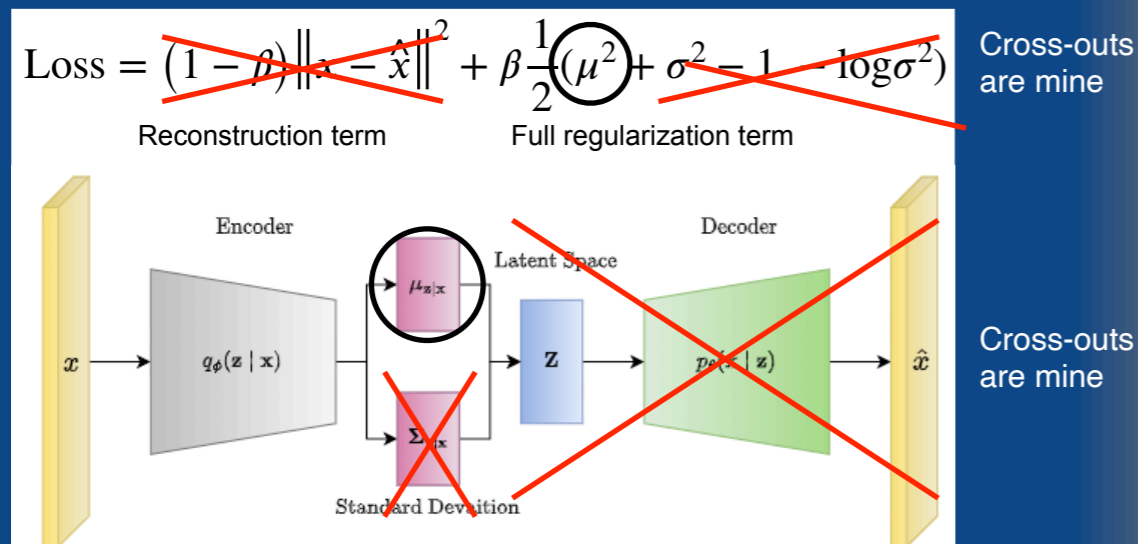
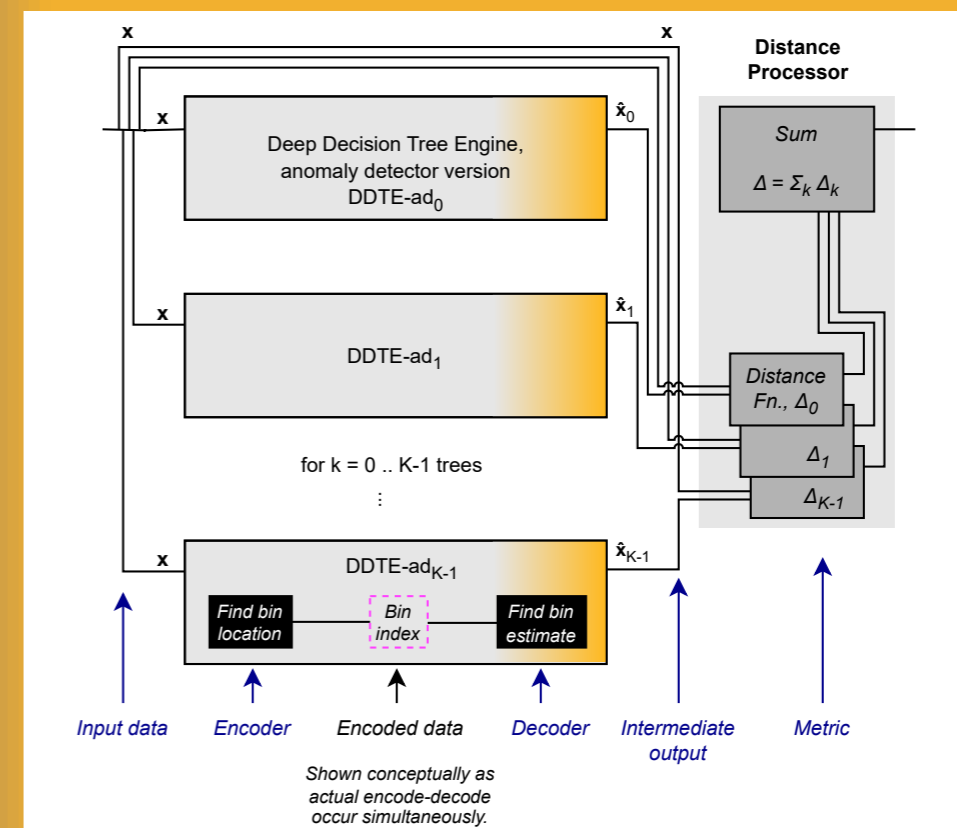


Image from  
<https://medium.com/@rushikesh.shende/autoencoders-variational-autoencoders-vae-and-beta-vae-ceba9998773d>





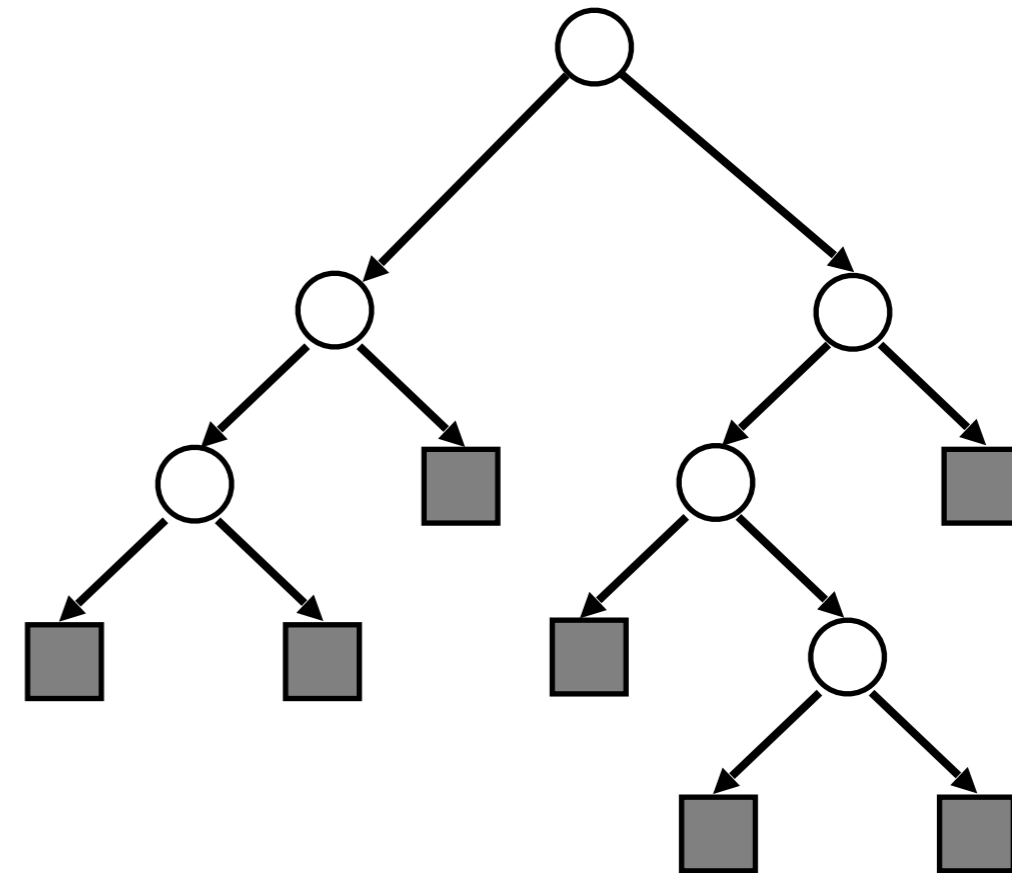
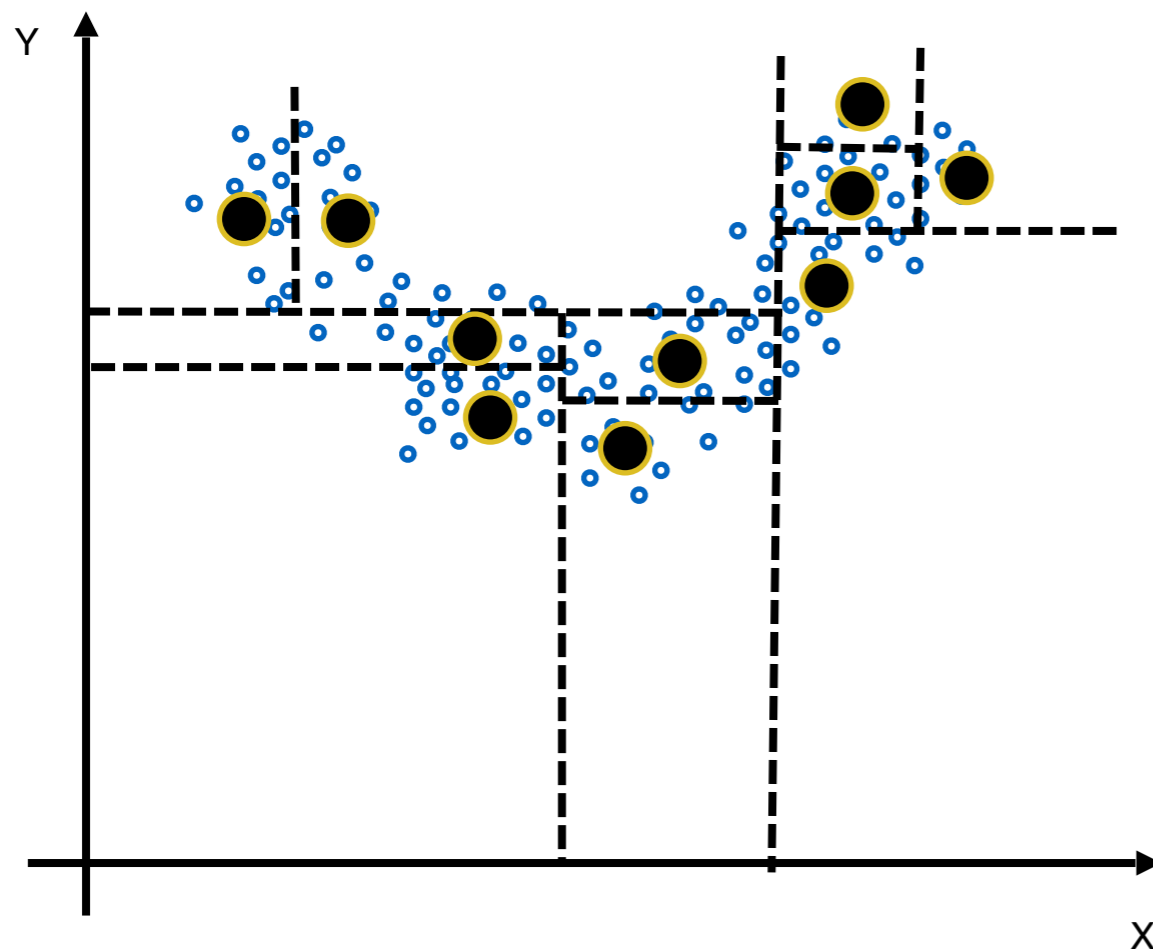


## Train by sampling 1d projections

- Encoding: Event  $\rightarrow$  which bin it's in

## Decoding returns “reconstruction point”

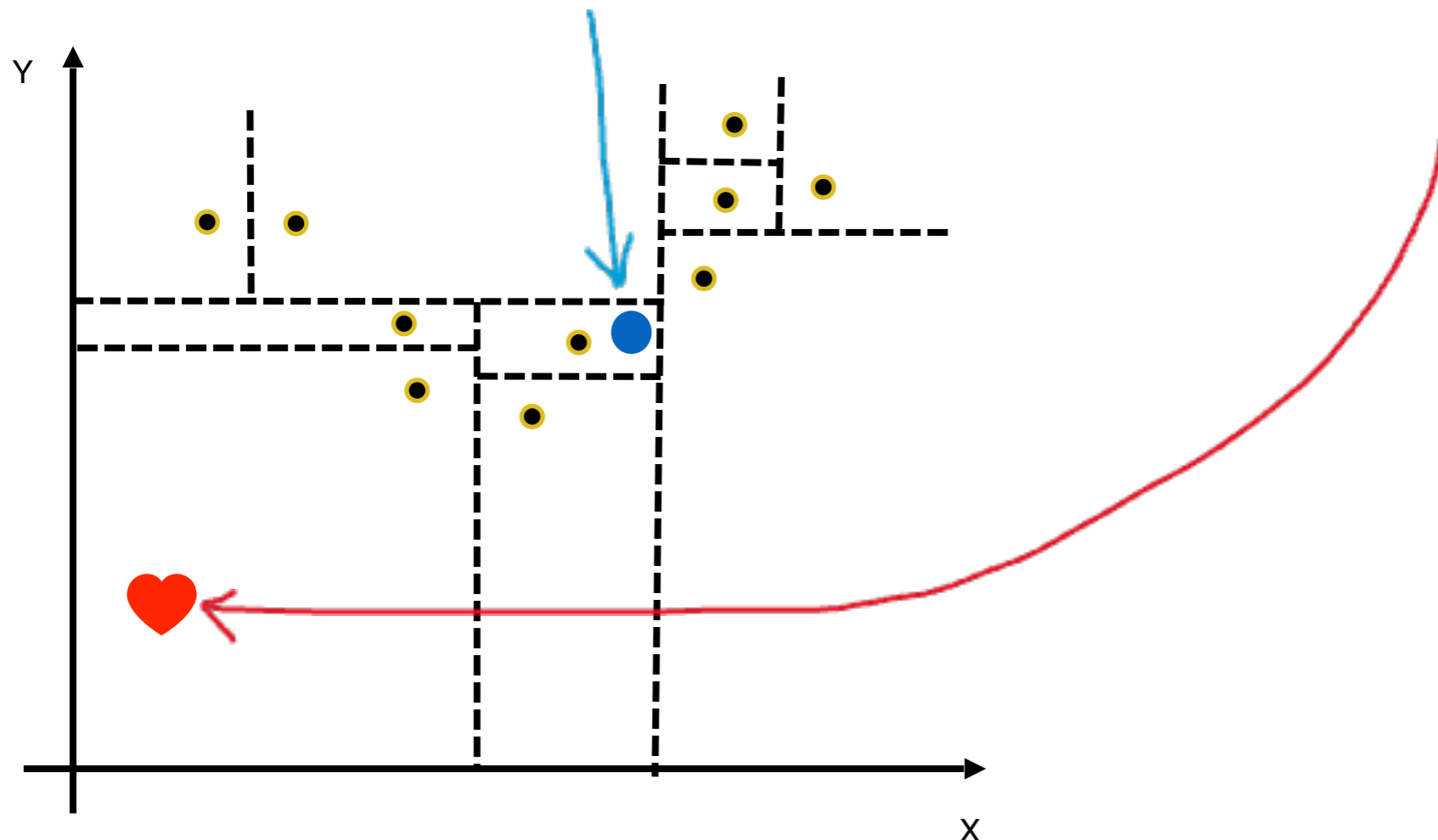
- Decoding: Bin  $\rightarrow$  median of the training data in bin

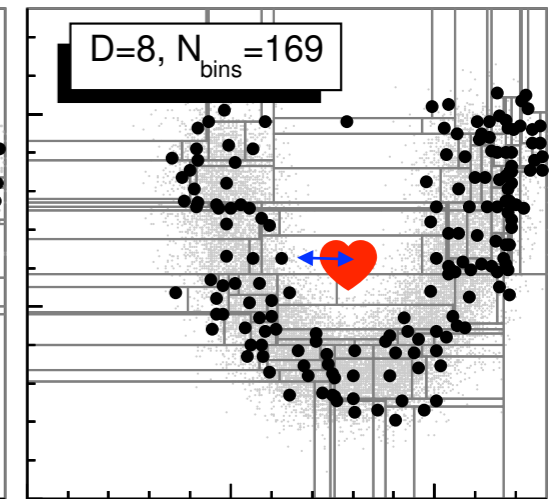
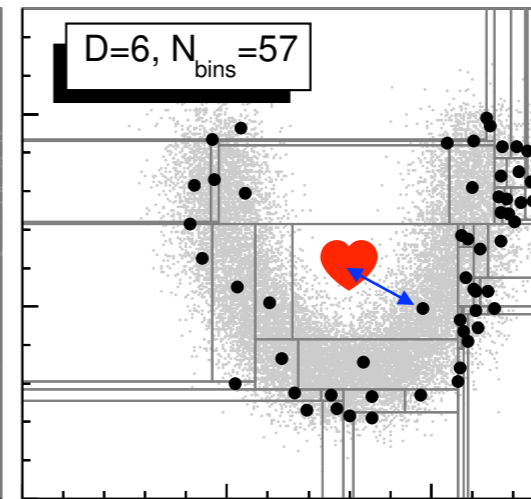
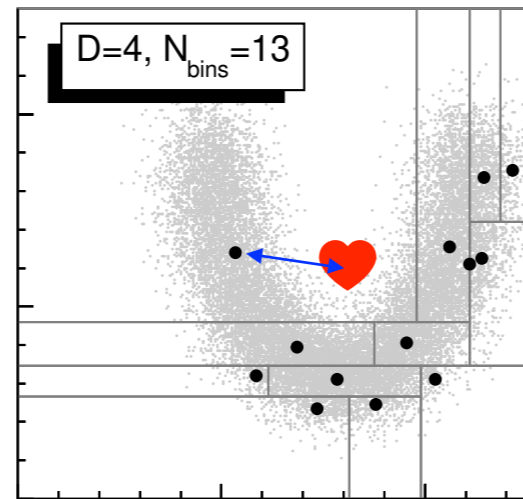
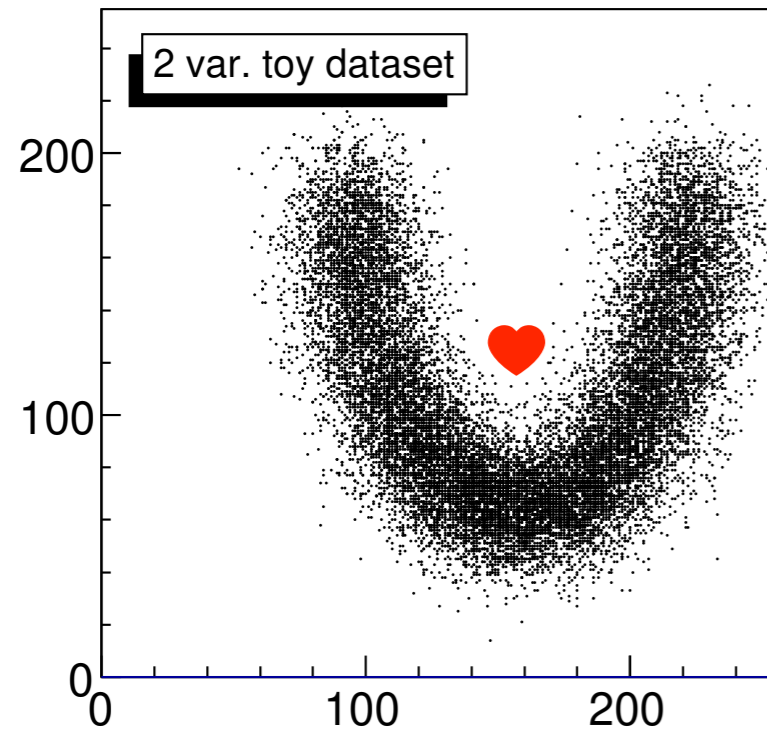




## How does this detect anomalies?

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

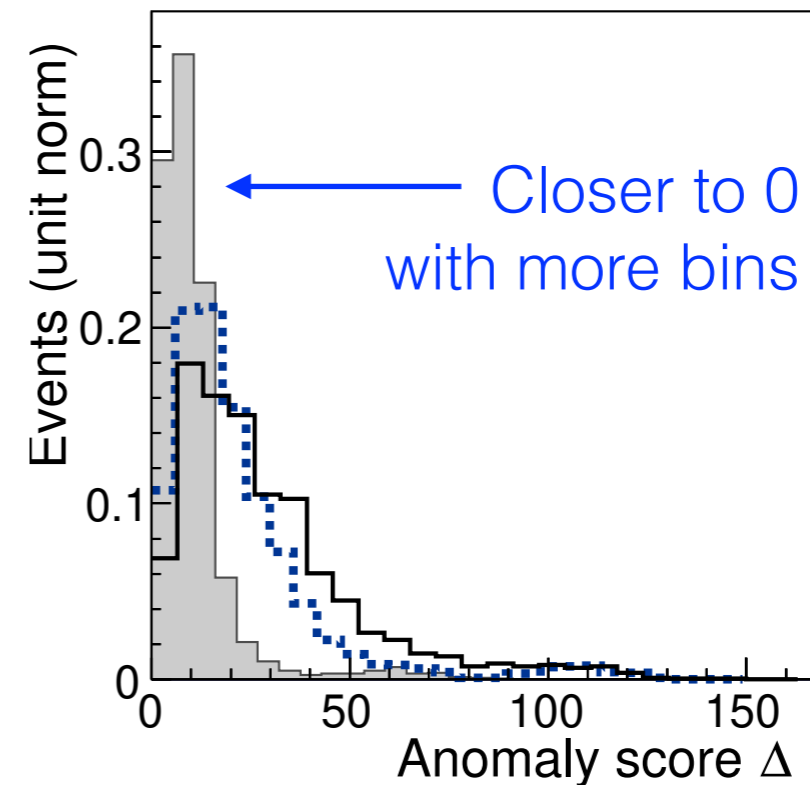




more bins

## Anomaly score

- Feed back in the training sample
  - Should be near 0, like  $E_T^{\text{miss}}$  resolution



Max. depth

— D=4

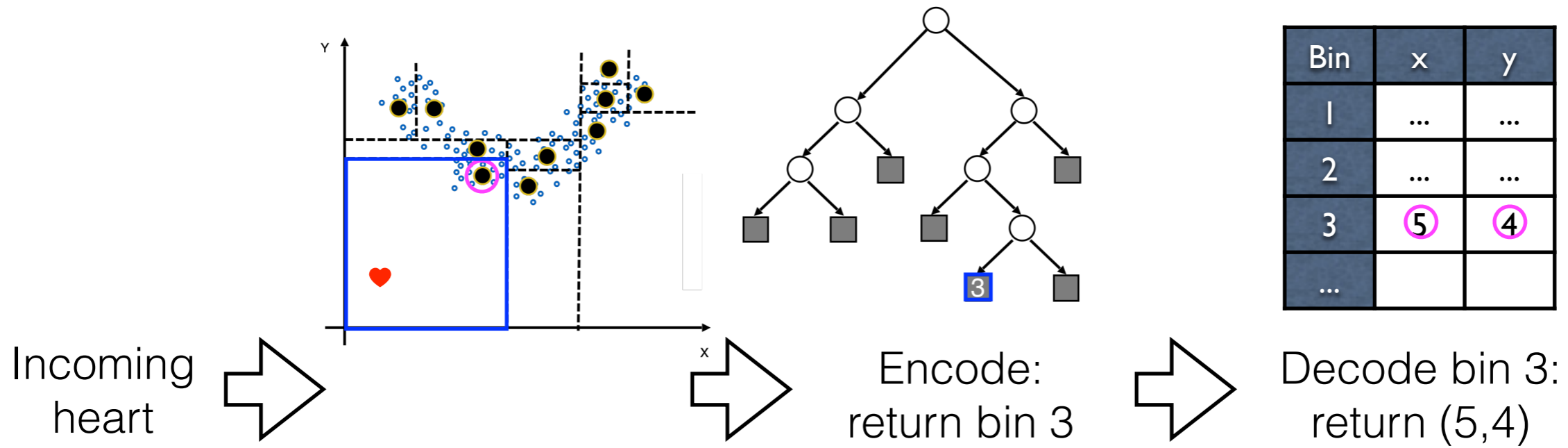
⋯ D=6

■ D=8

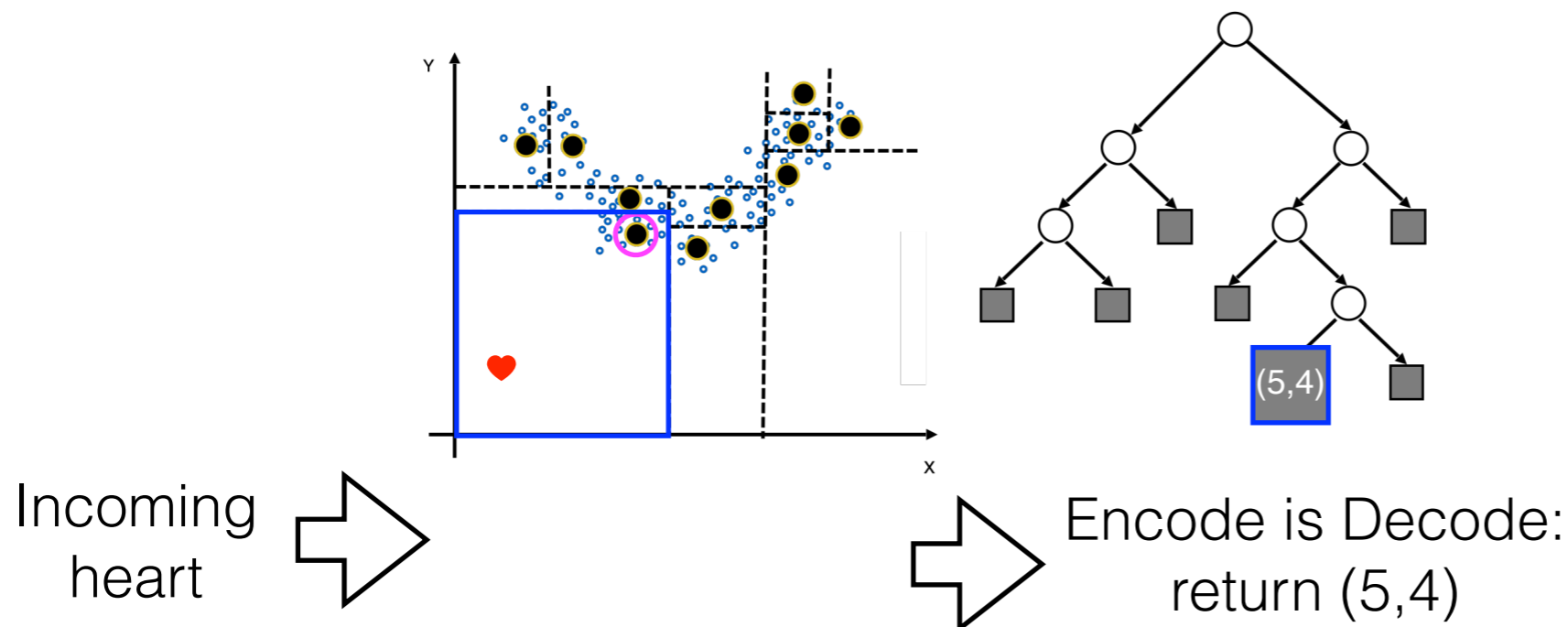


## Don't need latent space in firmware

- Closer look at what it means to encode



- Skip the encoding & decoding





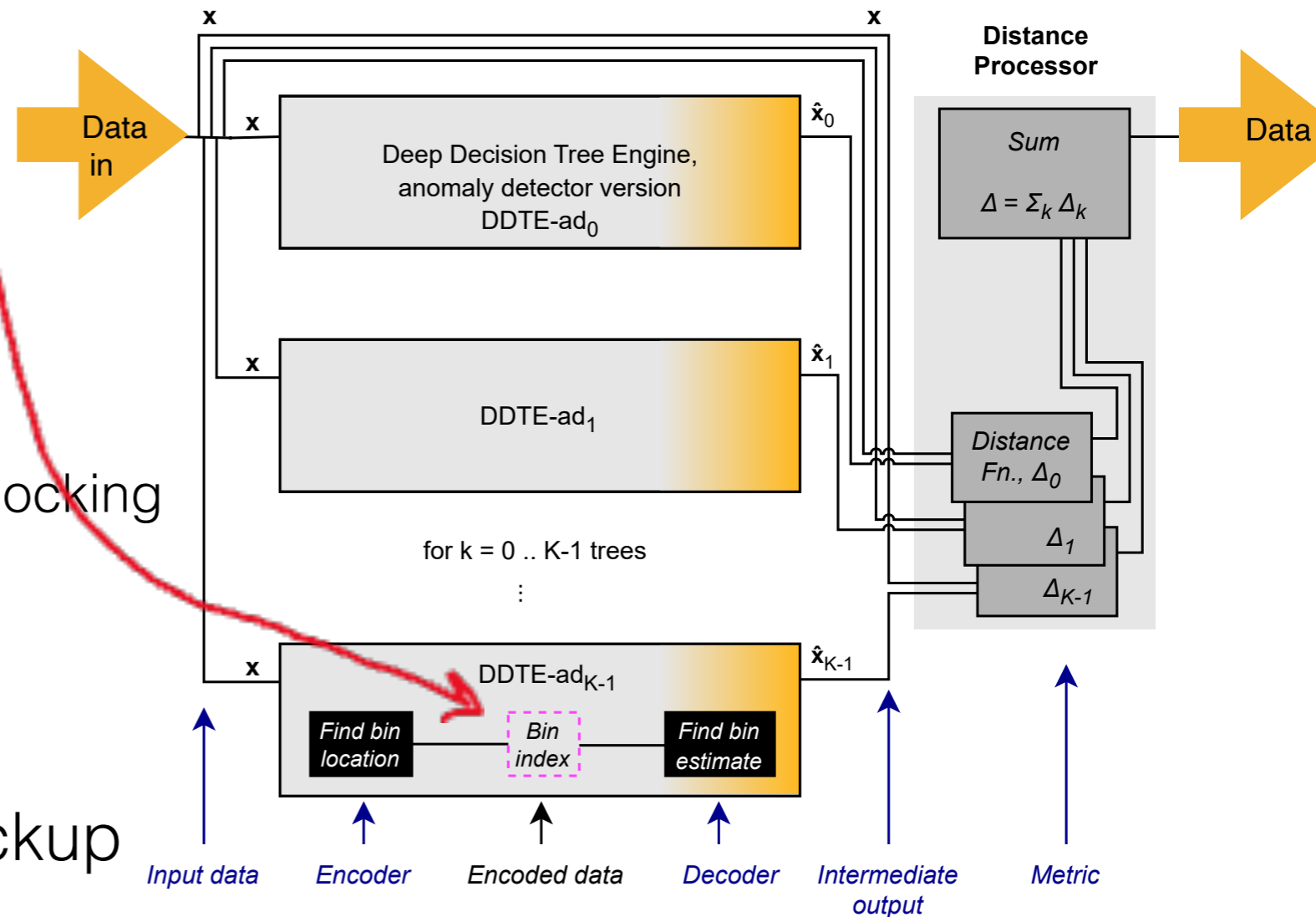
## Logic flow

- Left-to-right data flow (see right)
- Realized that we can bypass the latent space!
  - Encoding = Decoding

## Details

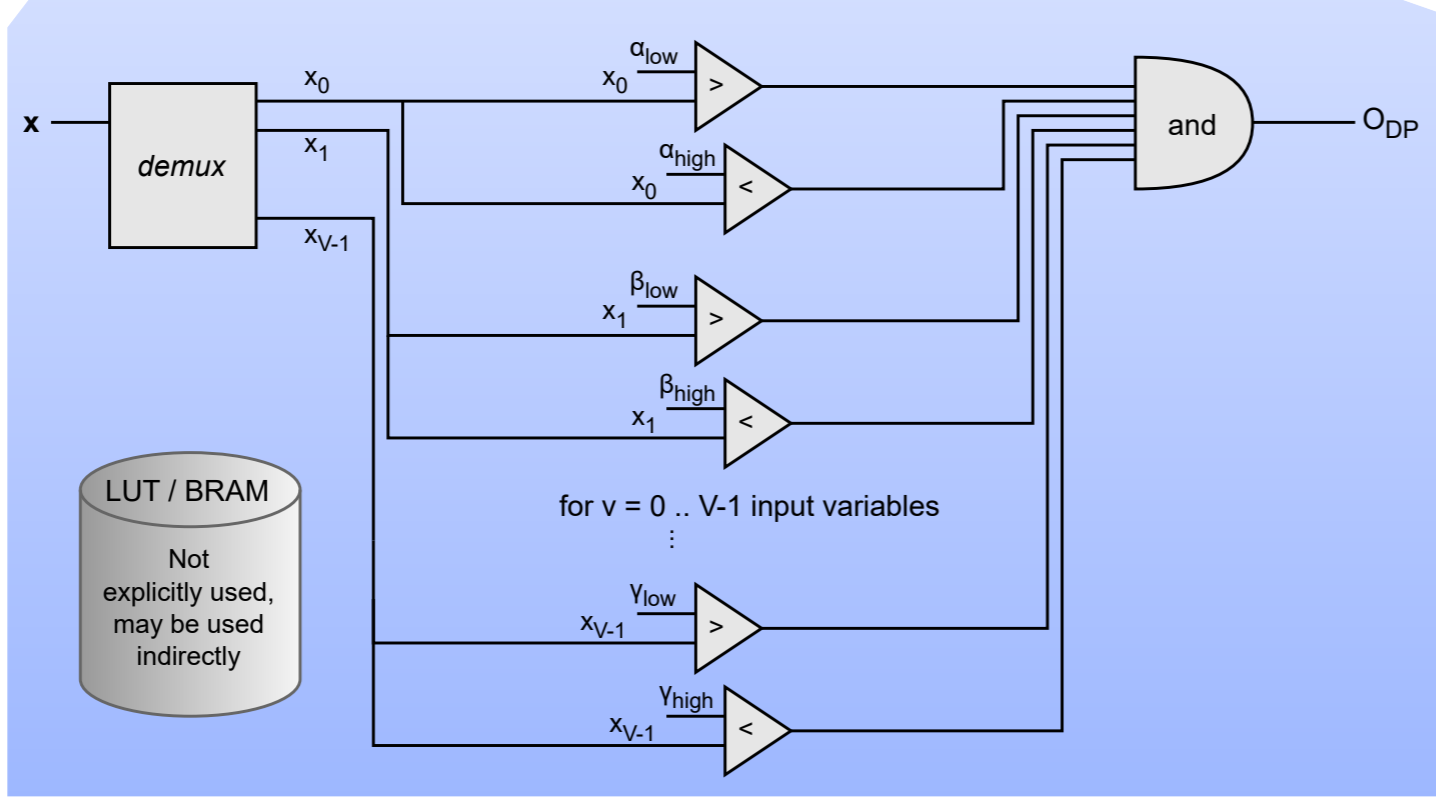
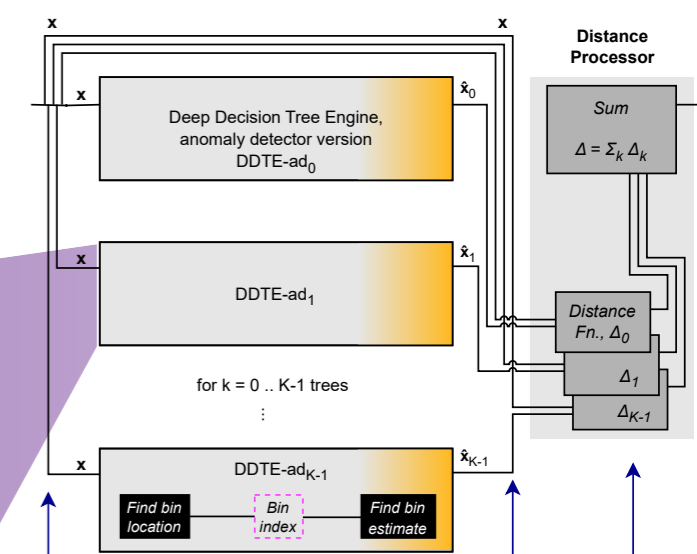
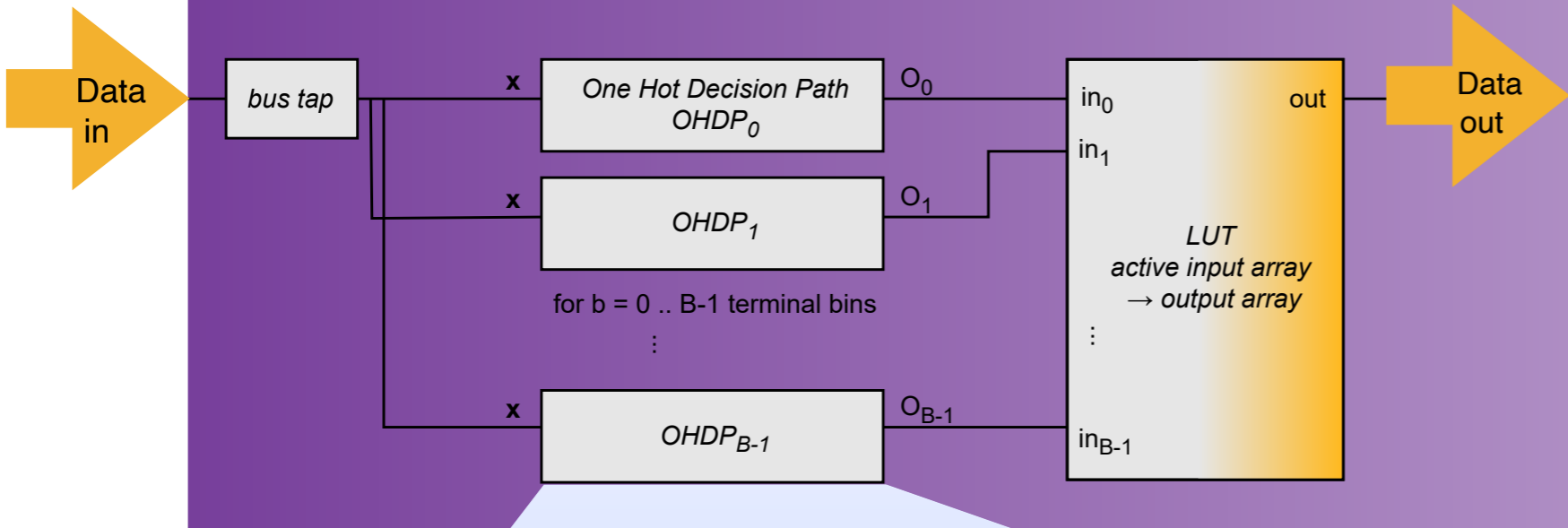
- Parallel computing
  - TREE ENGINES eval. in parallel
  - All combinatoric logic, so no clocking between steps = fast
  - Mostly comparisons = fast
  - No multiplication = fast

- Technical info in backup & [\[2304.03836\]](#)



Shown conceptually as actual encode-decode occur simultaneously.

Skip this slide



Skip this slide

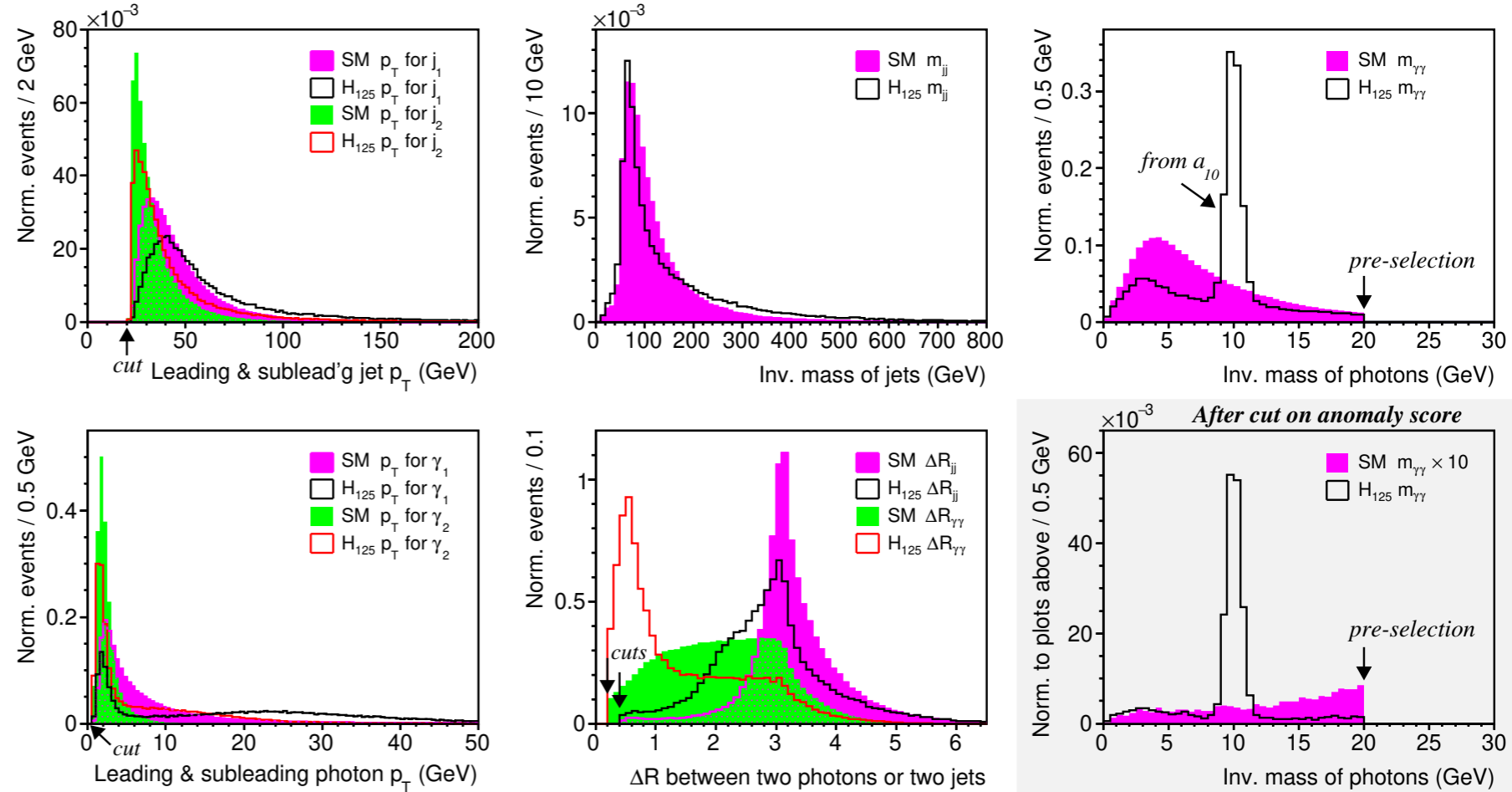
## Inputs

- Sample

- MadGraph5\_aMC 2.9.5
- Hadron'n+Shower: Pythia8
- Detector: Delphes 3.5.0, CMS

- Variables

- 8 inputs: jets, photons,  $\Delta R$



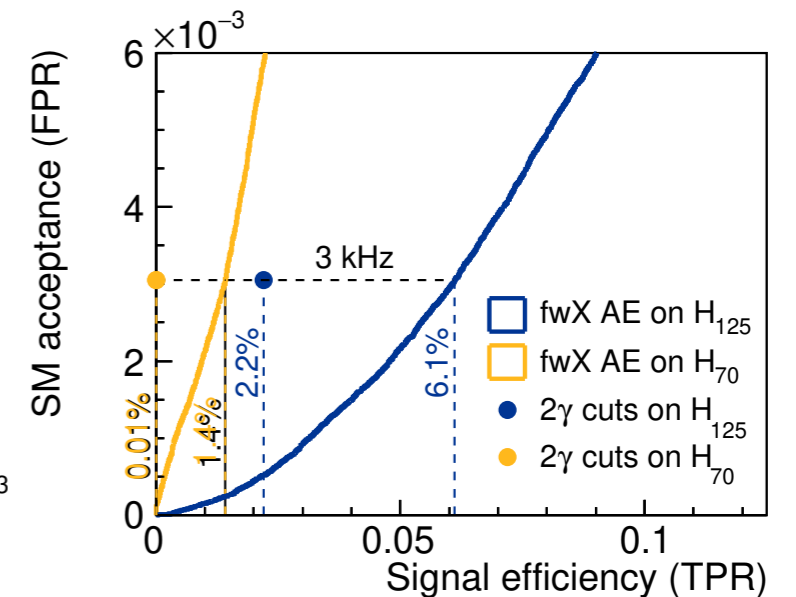
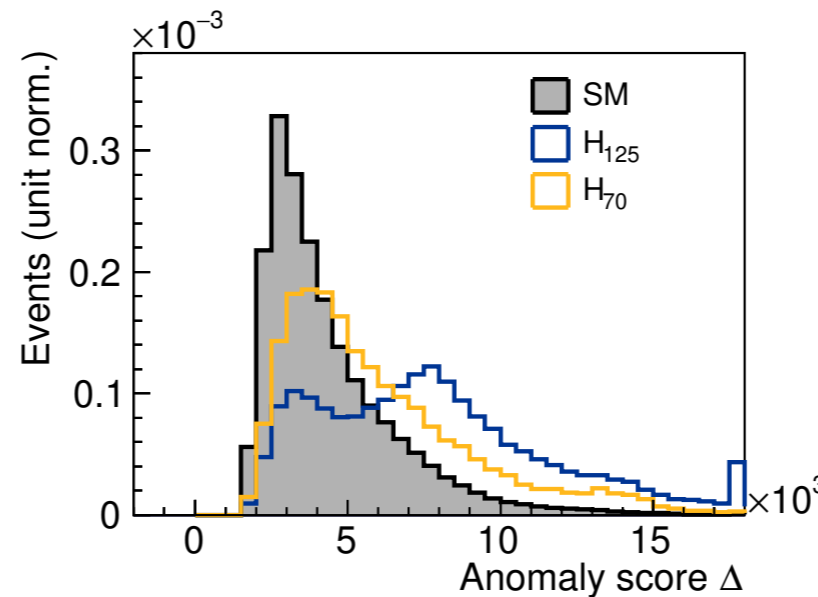
## Results

- Compare

- vs. 3 kHz Run-2 ATLAS rate

- Better

- 3x gain in signal

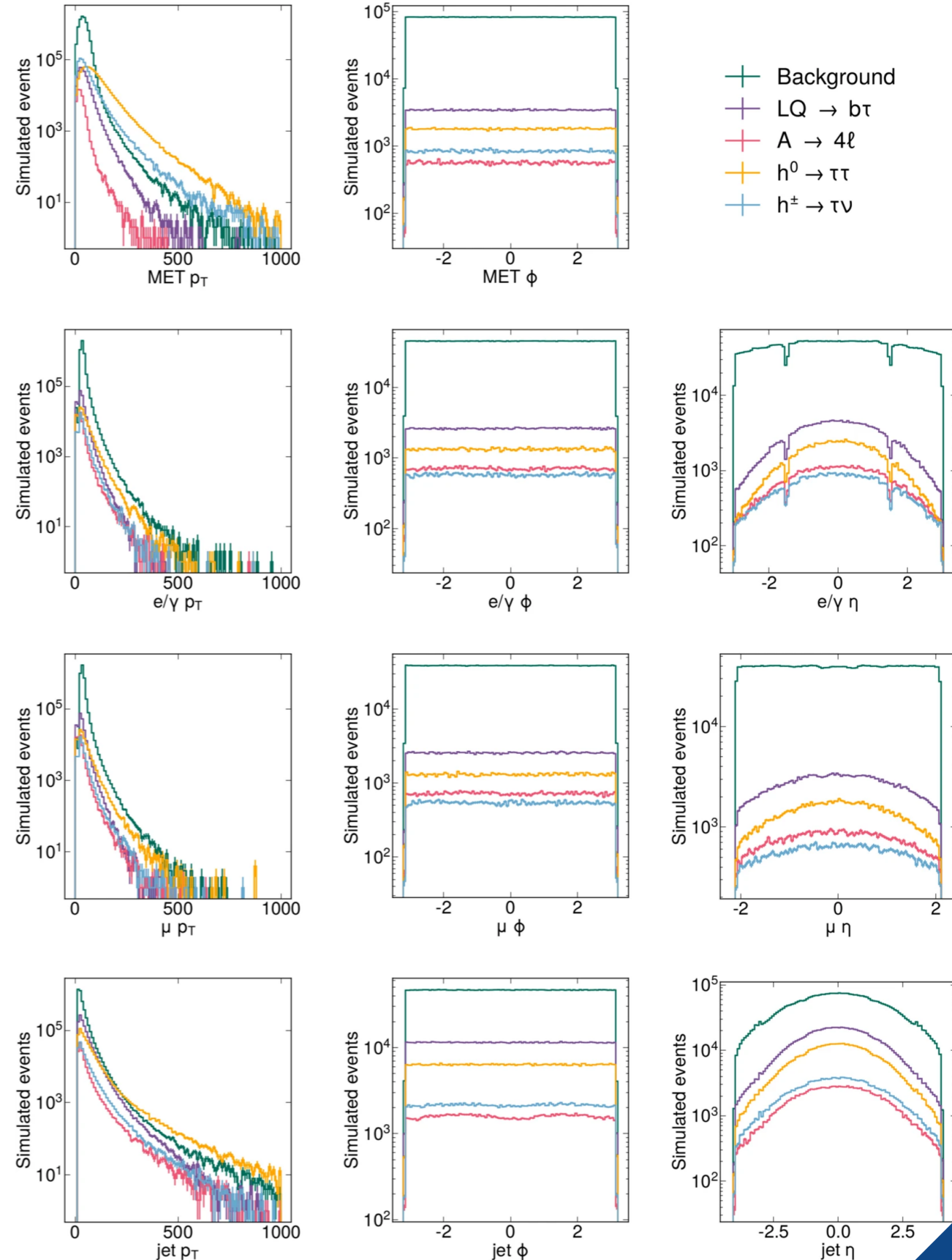


Skip this slide



## LHC anomaly detection ds [Sci Data 9, 118]

- Background
  - $W \rightarrow l\nu, Z \rightarrow ll, \text{multijet}, ttbar$
- Signal
  - 4 BSM scenarios
- Input variables
  - 54 variables
  - $p_T, \eta, \phi$  of the 4 leading  $\mu$ , 4 leading  $e$ , 10 leading jets, MET
  - See distributions on the right
- Sample selection
  - Require  $\geq 1$  lepton w/  $p_T > 23$  GeV
  - (L1 will already save these...)







## Works well

- Physics (plots)
- FPGA (table)

## Comparison

- Hls4ml NN-AE  
[[Nature Mach. Intell. 4 \(2022\) 154–161](#)]

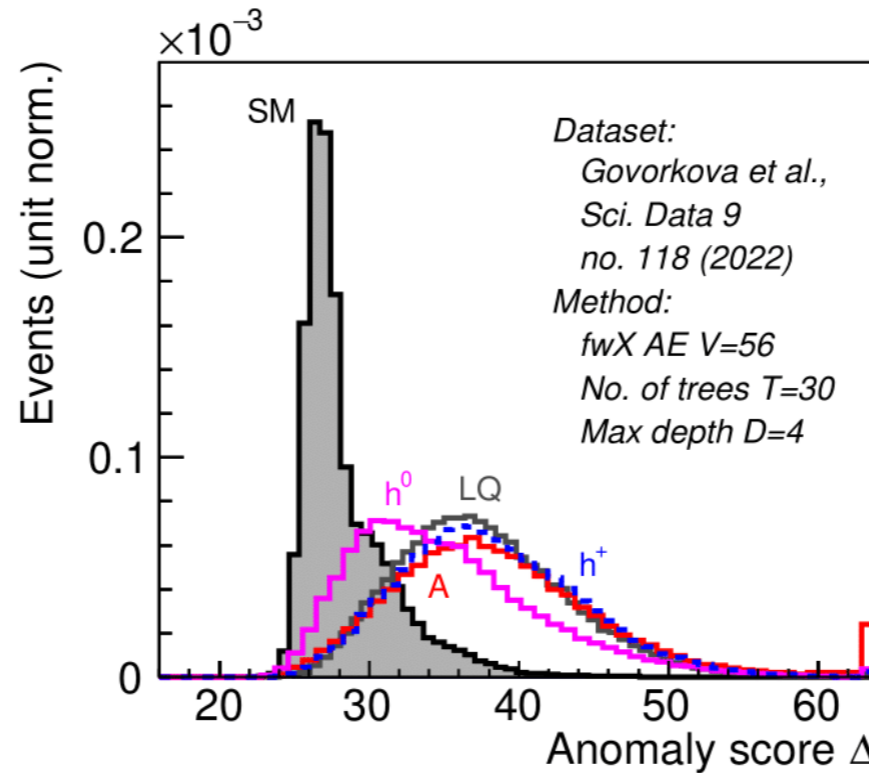
- Physics: comparable AUC

- FPGA results

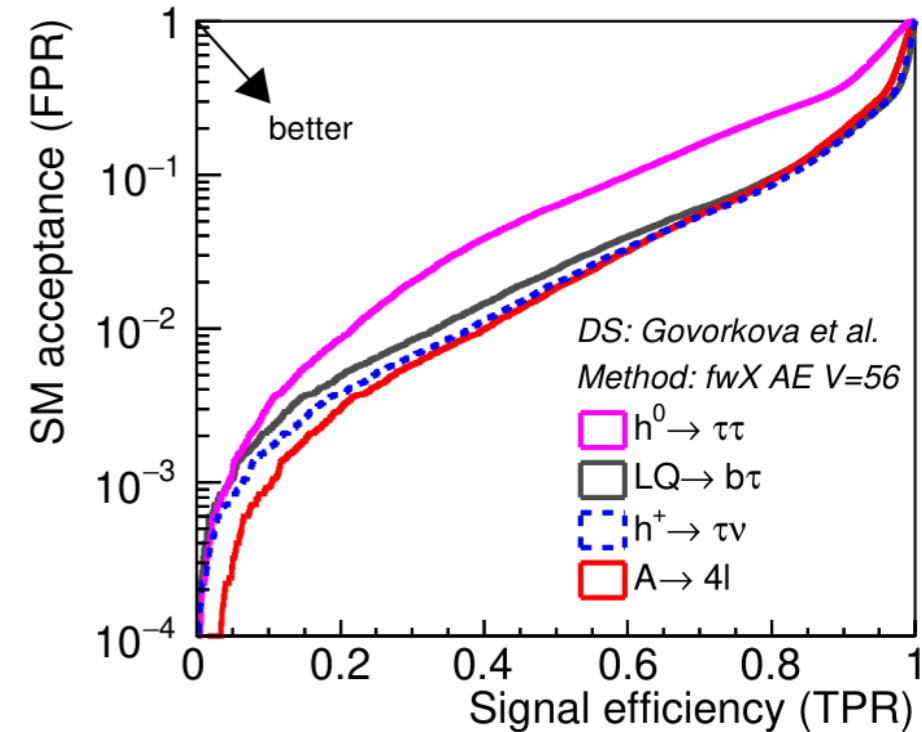
Key take-away:

This result uses HLS trees. Using VHDL trees projected to be faster and smaller (next slides).

## Distribution

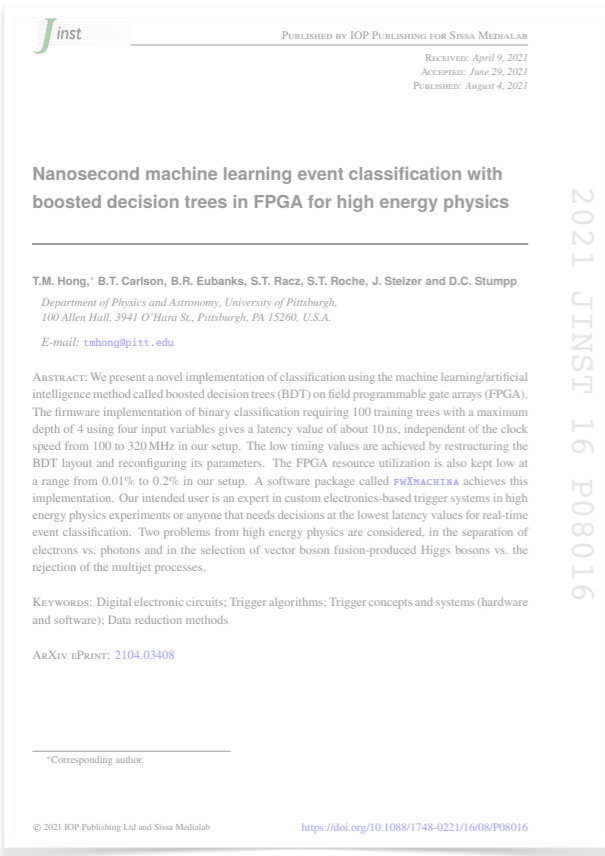


## ROC curve



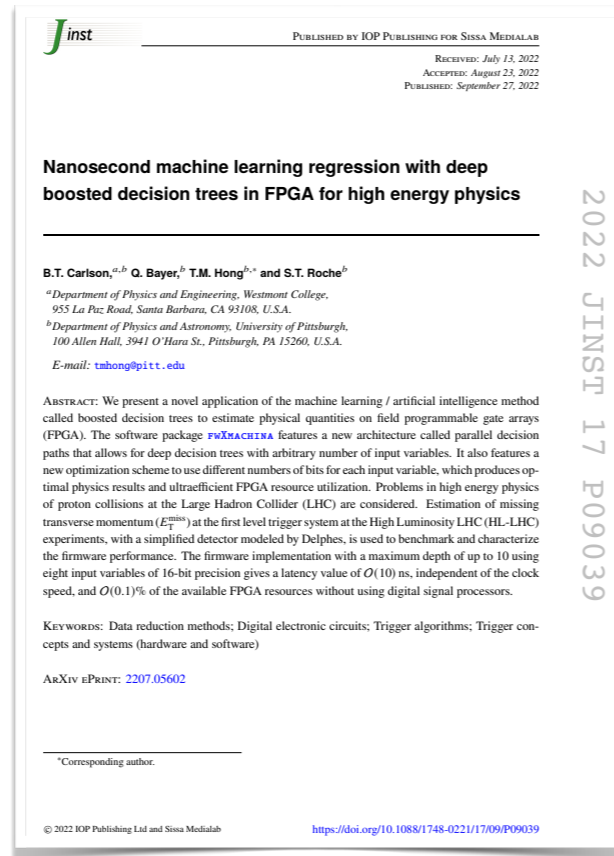
	hls4ml	fwX (this)
Clock speed	200 MHz	200 MHz
Latency	80 ns	30 ns
Interval	5 ns	5 ns
FF	0.5%	0.6%
LUT	3%	9%
DSP	1%	0.8%
BRAM	0.3%	0

## 1. Classification parallel cuts using HLS



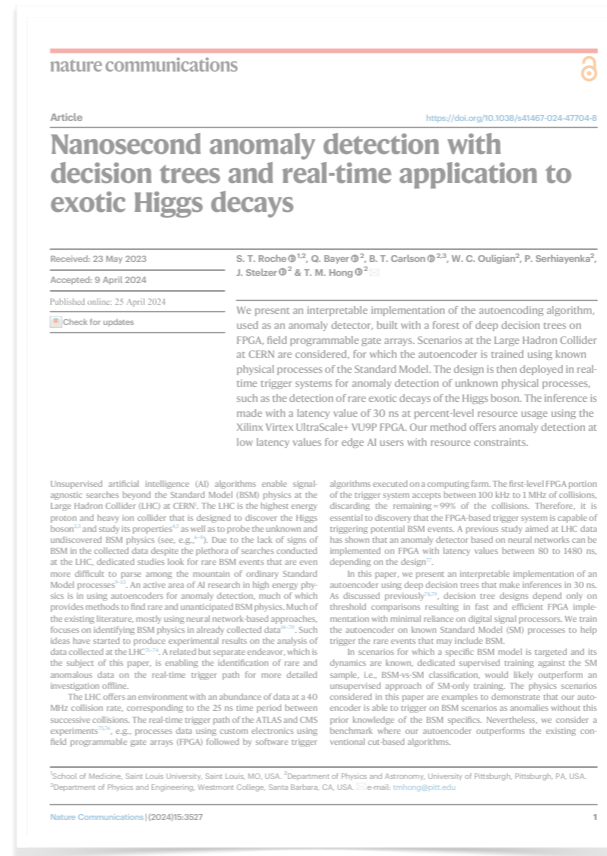
Hong et al.  
JINST 16, P08016 (2021)  
<http://doi.org/10.1088/1748-0221/16/08/P08016>

## 2. Regression parallel paths using HLS



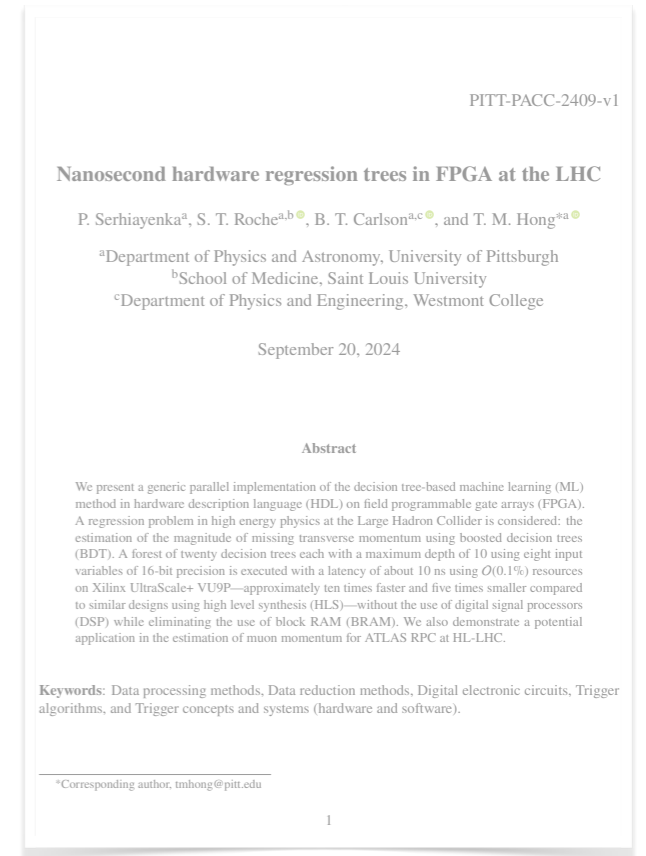
Carlson et al.  
JINST 17, P09039 (2022)  
<http://doi.org/10.1088/1748-0221/17/09/P09039>

## 3. Autoencoder in-house training bypassing latent space



Roche et al.  
Nat. Comm. 15 (2024) 3527  
<https://arxiv.org/abs/2304.03836>

## 4. Hardware trees faster & more efficient no more HLS



Serhiayenka et al.  
Submitted to NIM-A  
[2409.20506]

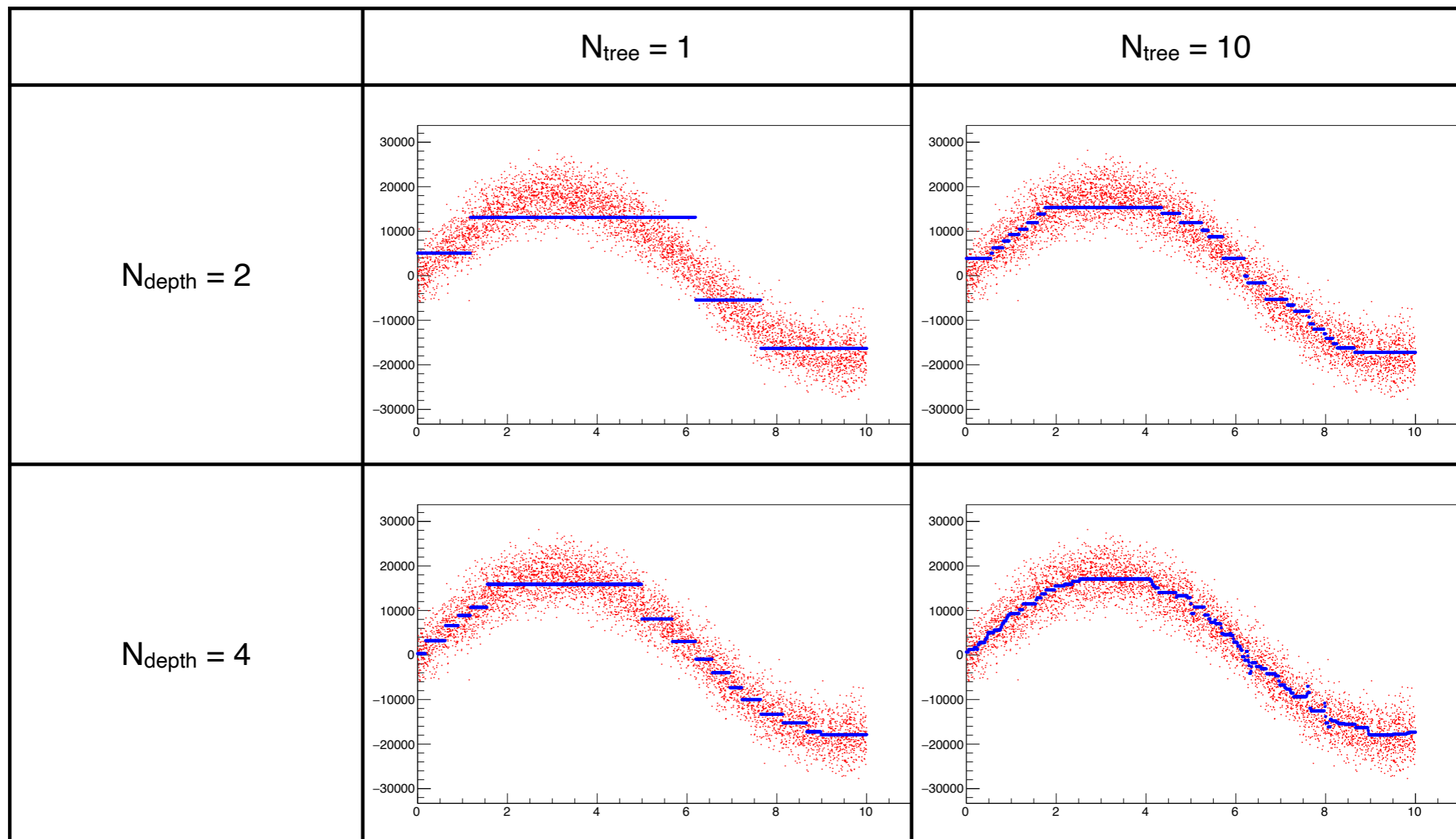
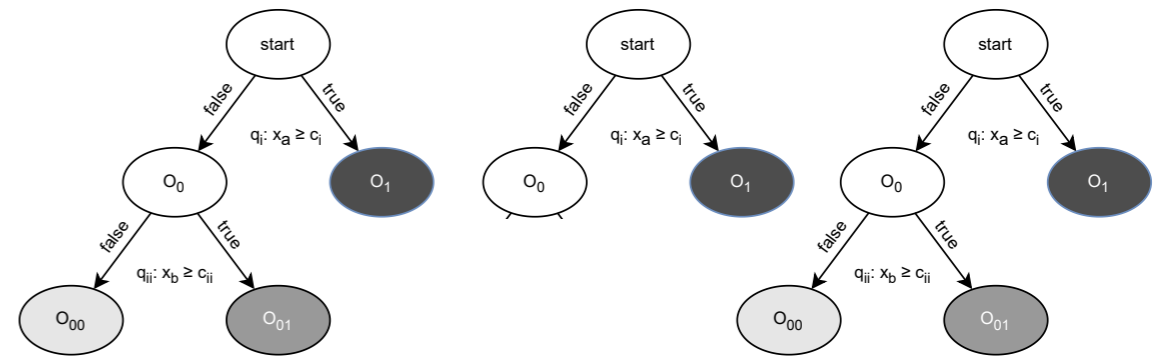


Focus today



## Regression (using BDT)

- Toy problem in 1-d
- Train / test on  $f(x) = \sin(x) + \text{Gaussian}(x)$
- For sample of  $x$ :  $y = f(x)$  in 16 bits



$(x,y) =$   
 $(N_{\text{tree}}, N_{\text{depth}})$

Ntree = 1

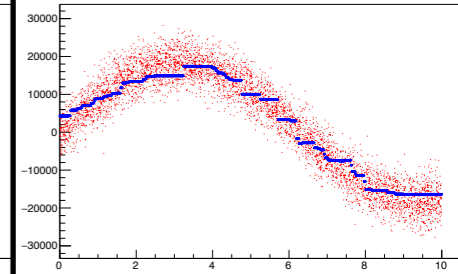
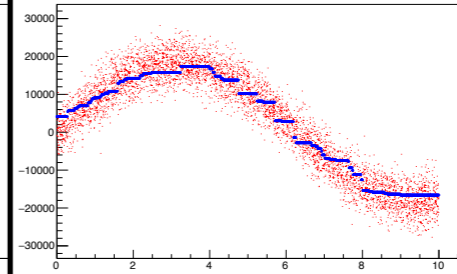
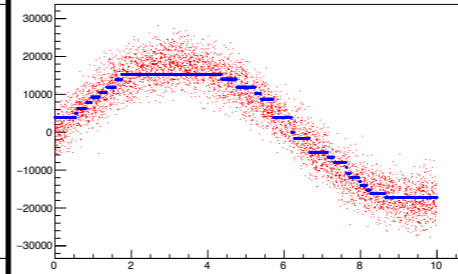
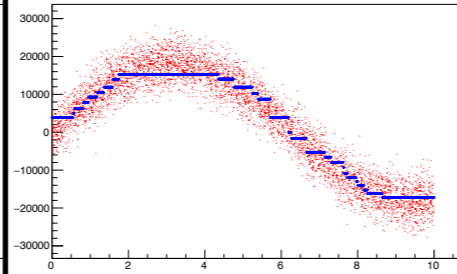
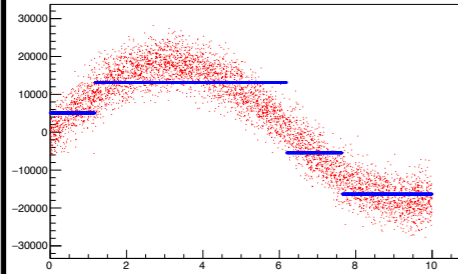
10

25

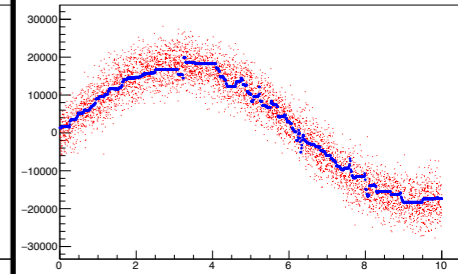
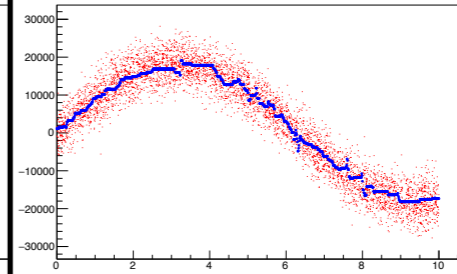
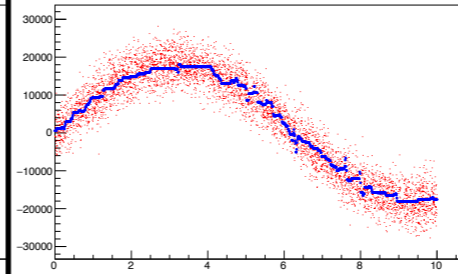
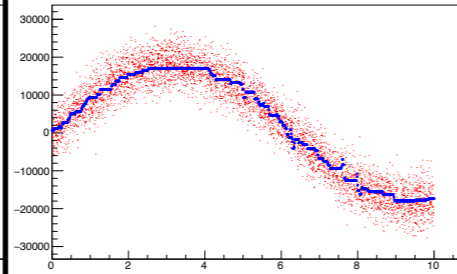
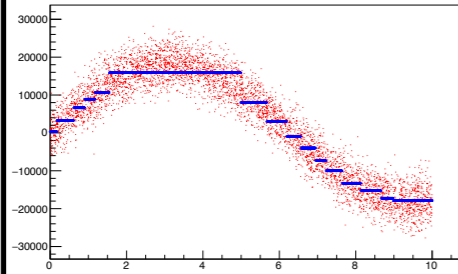
100

400

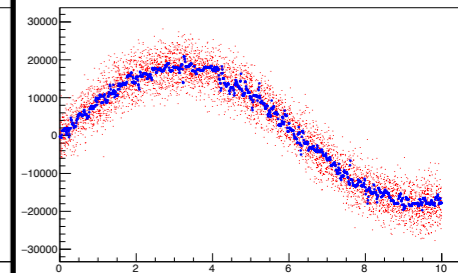
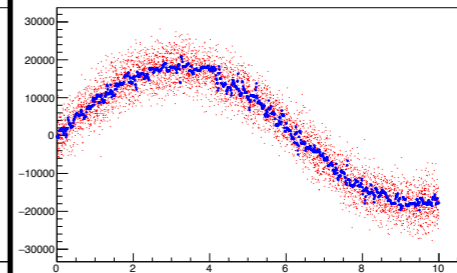
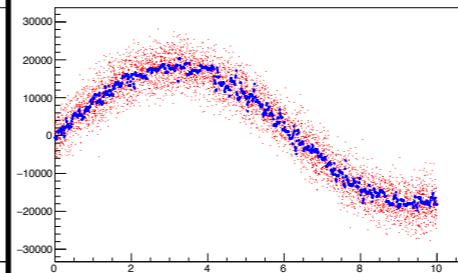
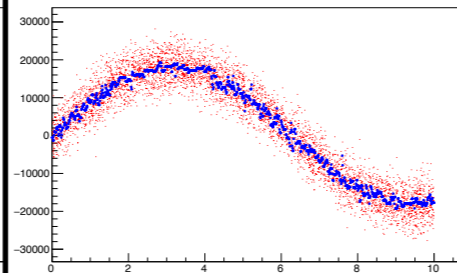
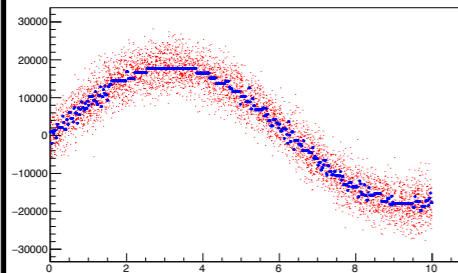
2



4



8

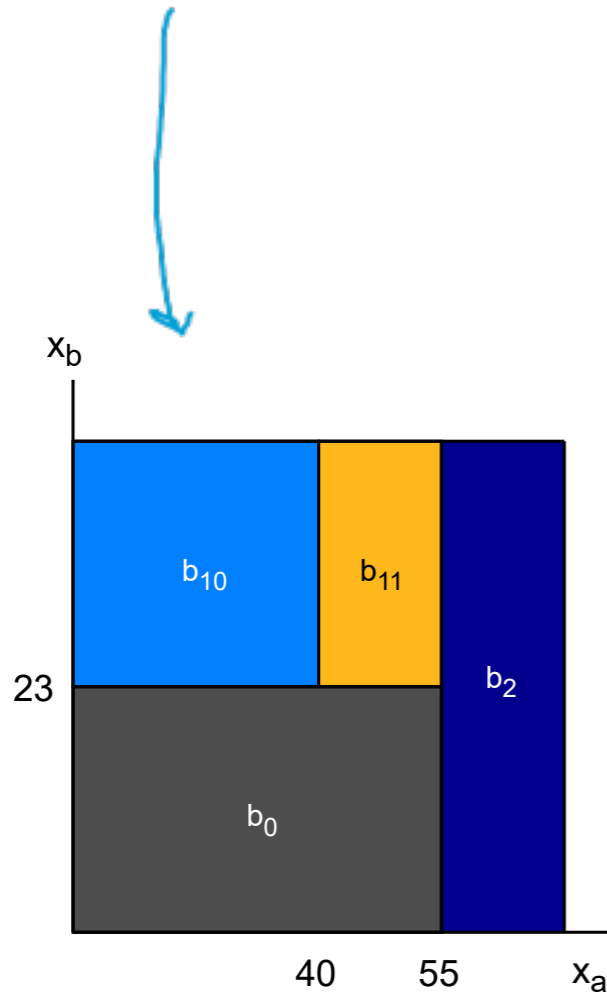


Skip this  
slide

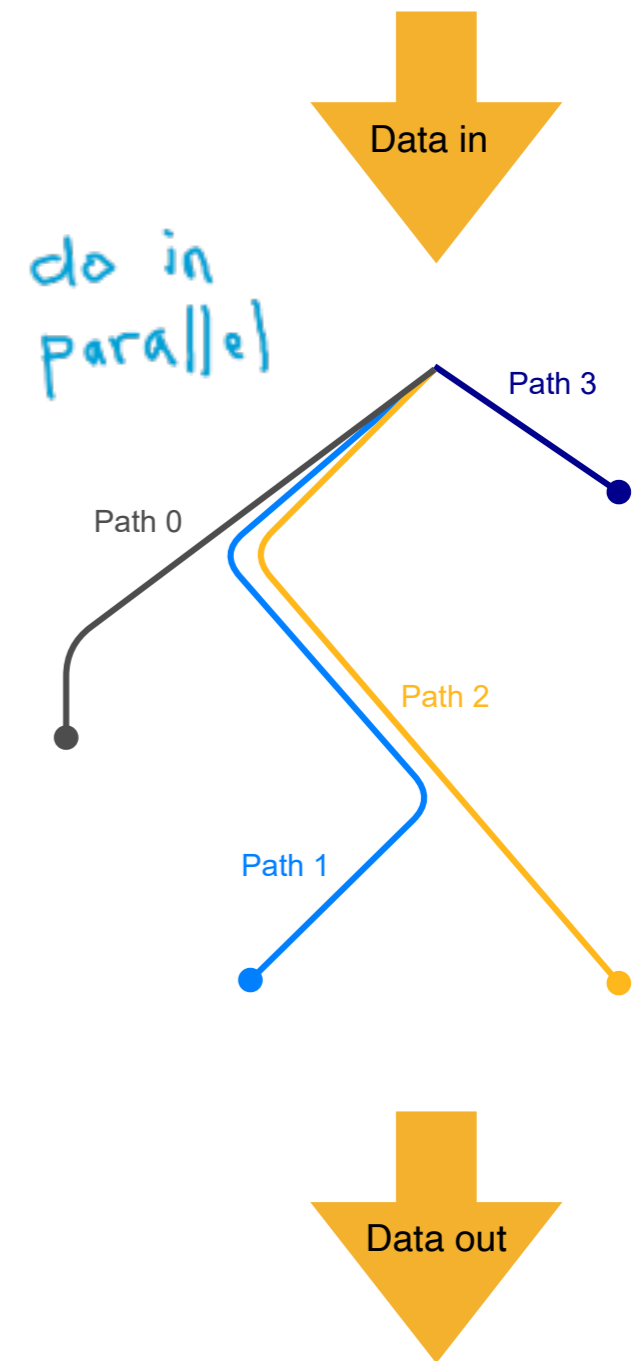
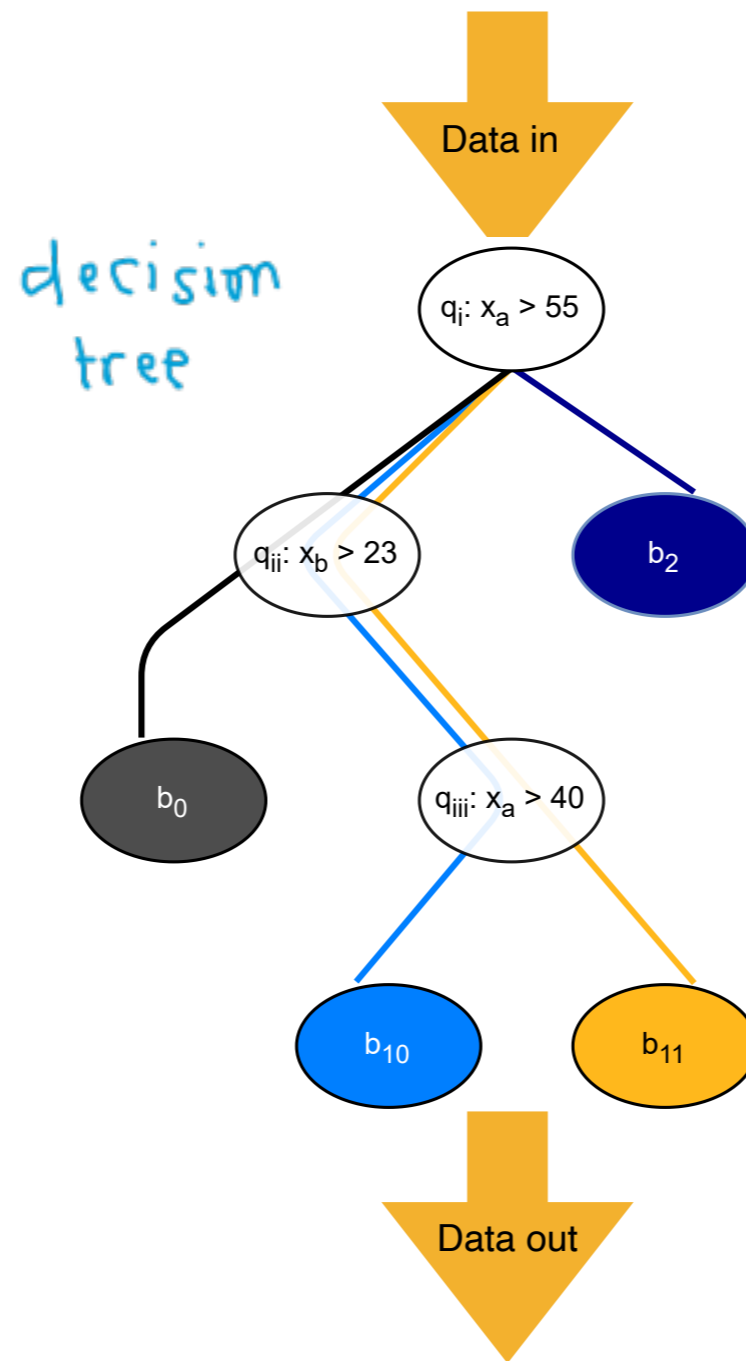


- Example

- 2d toy dataset, say  $x = p_T$  and  $y = \eta$  for some SM sample



Destination bin	Decision path
$b_0$	$\text{not}(q_i)$ and $\text{not}(q_{ij})$
$b_2$	$q_i$
$b_{10}$	$\text{not}(q_i)$ and $q_{ij}$ and $\text{not}(q_{iii})$
$b_{11}$	$\text{not}(q_i)$ and $q_{ij}$ and $q_{iii}$



Key idea: Paths can be evaluated in parallel



Key idea:

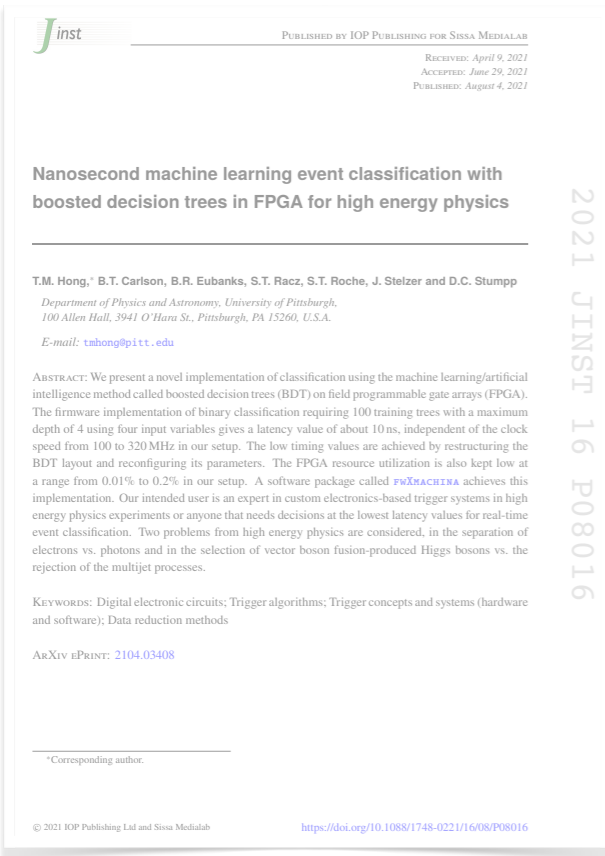
Can implement deep trees

Table 3: Benchmark configuration and the FPGA cost. Three groups of information are given. The top-most group defines the FPGA setup. The second group defines the ML training used for the MET problem and the Nanosecond Optimization. The third group gives the actual results measured on the FPGA for four tree-depth combinations of 40-5, 40-6, 20-7, and 10-8.

Parameter	Value	Comments
<b>FPGA setup</b>		
Chip family	Xilinx Virtex Ultrascale+	
Chip model	xcvu9p-flga2104-2L-e	
Vivado version	2019.2	
Synthesis type	C synthesis	
HLS or RTL	HLS	HLS interface pragma: None
Clock speed	320 MHz	Clock period is 3.125 ns
<b>ML training configuration &amp; Nanosecond Optimization configuration</b>		
ML training method	Boosted decision tree	Regression, Adaptive boosting
No. of input variables	8	
BIN ENGINE type	DEEP DECISION TREE ENGINE (DDTE)	
No. of bits for all variables	16 bits for each	binary integers
<b>FPGA cost for 40 trees, 5 depth</b>		
Latency	6 clock ticks	18.75 ns
Look up tables	1675 out of 1 182 240	0.1% of available
Flip flops	1460 out of 2 364 480	< 0.1% of available
<b>FPGA cost for 40 trees, 6 depth</b>		
Latency	9 clock ticks	28.125 ns
Look up tables	4566 out of 1 182 240	0.4% of available
Flip flops	2516 out of 2 364 480	0.1% of available
<b>FPGA cost for 20 trees, 7 depth</b>		
Latency	15 clock ticks	46.875 ns
Look up tables	4568 out of 1 182 240	0.4% of available
Flip flops	2697 out of 2 364 480	0.1% of available
Block RAM	4.5 out of 4320	0.1% of available
<b>FPGA cost for 10 trees, 8 depth</b>		
Latency	21 clock ticks	65.625 ns
Look up tables	2556 out of 1 182 240	0.2% of available
Flip flops	2299 out of 2 364 480	0.1% of available
Block RAM	5 out of 4320	0.1% of available
<b>Common values for the above configurations</b>		
Interval	1 clock tick	3.125 ns
Block RAM	0 out of 4320	If not listed above
Ultra RAM	0 out of 960	Same for all trees and all depth
Digital signal processors	0 out of 6840	Same for all trees and all depth

Skip this slide

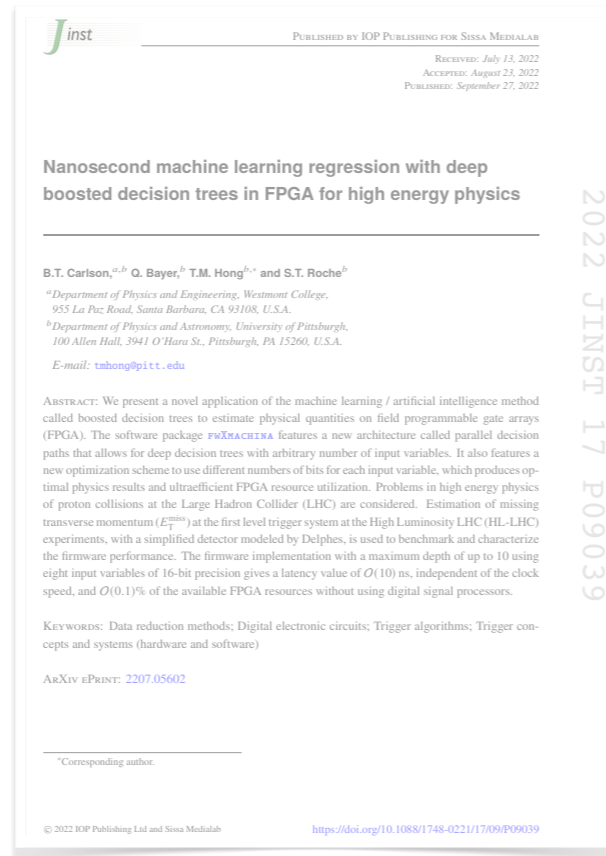
## 1. Classification parallel cuts using HLS



2021 JINST 16 P08016

Hong et al.  
JINST 16, P08016 (2021)  
<http://doi.org/10.1088/1748-0221/16/08/P08016>

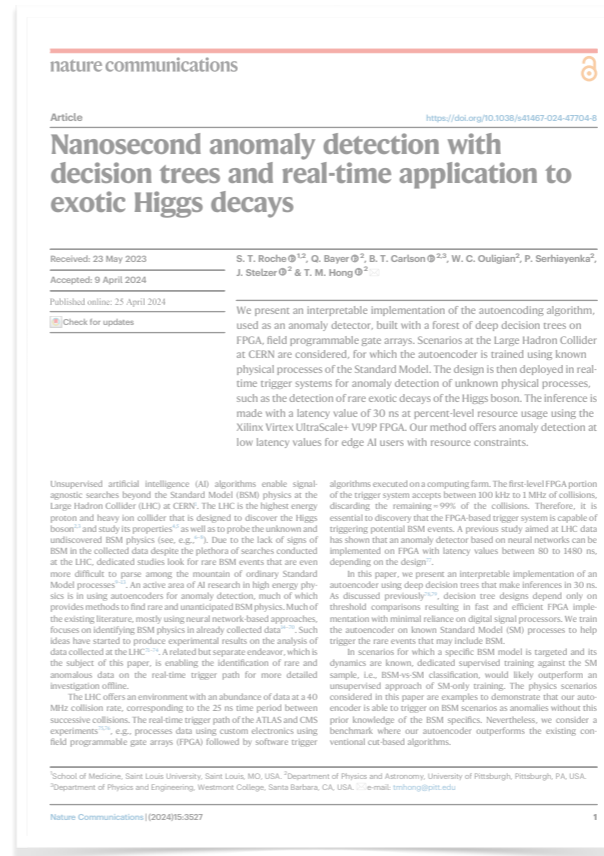
## 2. Regression parallel paths using HLS



2022 JINST 17 P09039

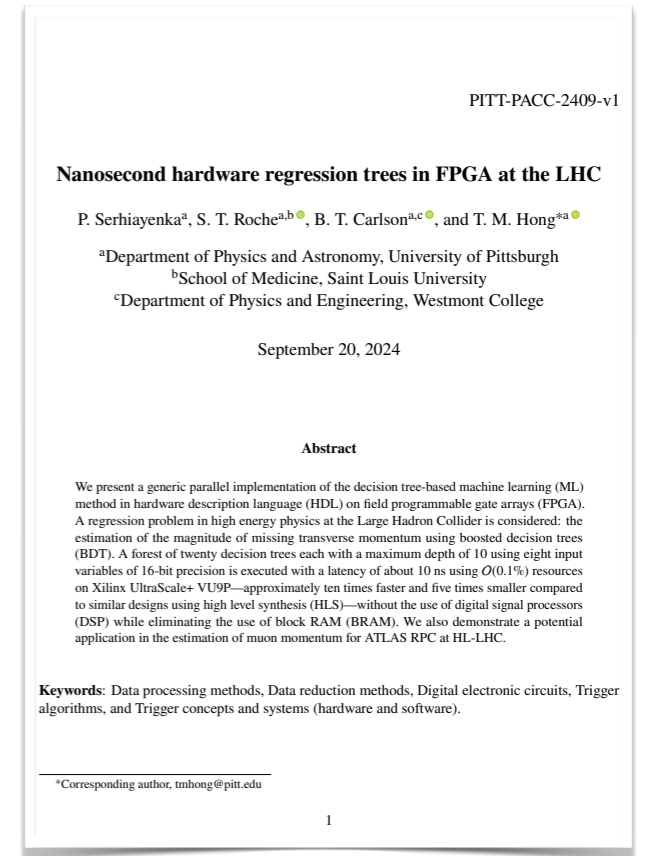
Carlson et al.  
JINST 17, P09039 (2022)  
<http://doi.org/10.1088/1748-0221/17/09/P09039>

## 3. Autoencoder in-house training bypassing latent space



Roche et al.  
Nat. Comm. 15 (2024) 3527  
<https://arxiv.org/abs/2304.03836>

## 4. Hardware trees faster & more efficient no more HLS



Serhiayenka et al.  
Submitted to NIM-A  
[2409.20506]



Focus today



## Summary

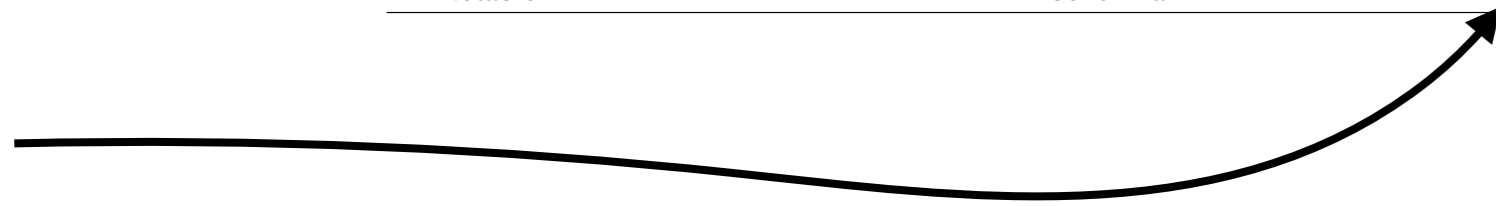
- Python to write VHDL

Table 1: FPGA results and comparison with Refs. [7, 8, 11]. All results in the table uses the same FPGA model Xilinx Ultrascale+ VU9P (vu9p-flgb2104-2L-e) with the following available resources 1.2 M LUT, 2.4 M FF, 6.8 k DSP, and 4.3 k BRAM. Effective depth  $d$  is defined as so that  $2^d = N_{bin}/N_{tree}$ .

Goal	5 classif'n	2 classif'n	$E_T^{miss}$	regression	$E_T^{miss}$	regression		
Reference	[11]	[7]	[8]	.....	This paper	.....		
<b>Setup</b>								
Design	VHDL	HLS	HLS	HLS	VHDL	VHDL	VIDL	VHDL
Sum strategy	-	-	-	-	pipeline	combin.	combin.	pipeline
Parallelize	-	cutwise	pathwise	pathwise	pathwise	pathwise	pathwise	pathwise
Clock (MHz)	250	320	320	320	320	320	200	320
Bit precision	fixed <sub>18</sub>	int <sub>8</sub>	int <sub>16</sub>	int <sub>16</sub>	int <sub>16</sub>	int <sub>16</sub>	int <sub>16</sub>	int <sub>16</sub>
$N_{var}$	16	4	8	8	8	8	8	8
$N_{tree}$	100	100	40	10	40	10	20	100
Max. depth $D$	4	4	6	8	6	8	10	12
$N_{bin}$	-	-	1.7k	1.4k	1.7k	1.4k	2.9k	15.7k
Effective depth $d$	-	-	5.4	7.2	5.4	7.2	7.2	7.3
<b>Notable</b>								
				identical	identical	slower clock		larger forest
<b>Results</b>								
LUT	96k	1k	6.4k	75k	5.1k	10k	15.5k	38k
FF	43k	0.1k	35k	24k	1.6k	4.7k	6.6k	19.4k
DSP	0	2	0	0	0	0	0	0
BRAM	0	5.5	0	10	0	0	0	0
URAM	-	0	0	0	0	0	0	0
Latency (ns)	52 ns	9.375 ns	38 ns	119 ns	25 ns	19 ns	10 ns	28 ns
" (tick)	13	3	12	38	8	6	2	9
Interval (tick)	1	1	1	1	1	1	1	1
<b>Notable</b>								
				benchmark				in abstract

## Results

- 5x smaller
- 10x faster



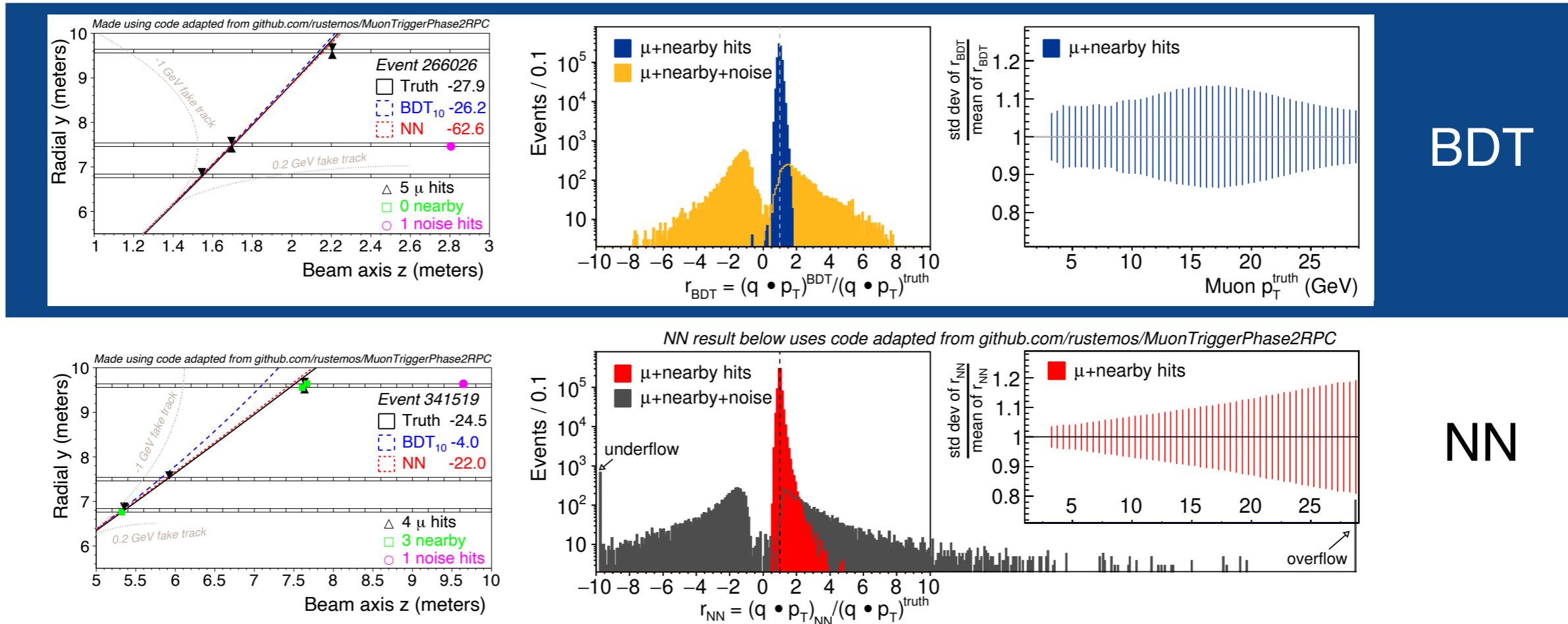




## Test case

- Mock-up ATLAS RPC for Phase-2

R. Ospanov, C. Feng, W. Dong, W. Feng, and S. Yang, *Development of FPGA-based neural network regression models for the ATLAS Phase-II barrel muon trigger upgrade*, Eur. Phys. J. Web of Conf. **251**, 04031 (2021).



## Results

- NN core is sharper
- BDT tail is shorter

Skip this slide

# Outline

## Introduction

- Who we are

## FPGA design

- ➔ • Autoencoder
- Parallelizing decision trees
- HLS trees → VHDL trees

## Thoughts on SRO

- Data compression
- Anomaly detection

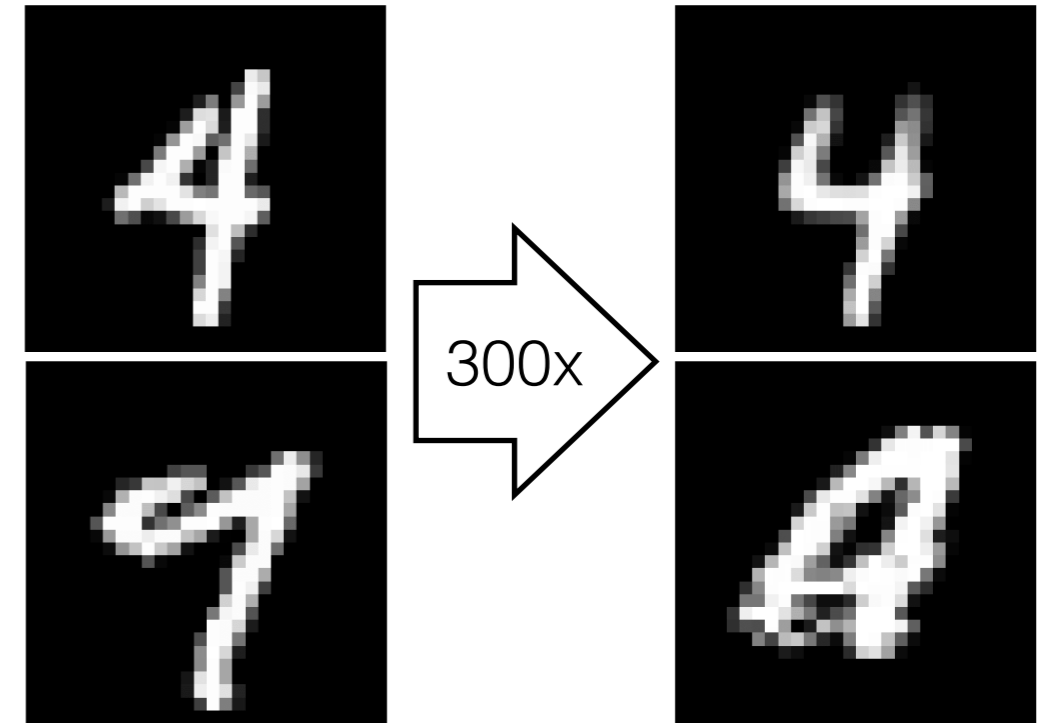
## More info

- Relevant papers from us
- Where to find code, tutorials



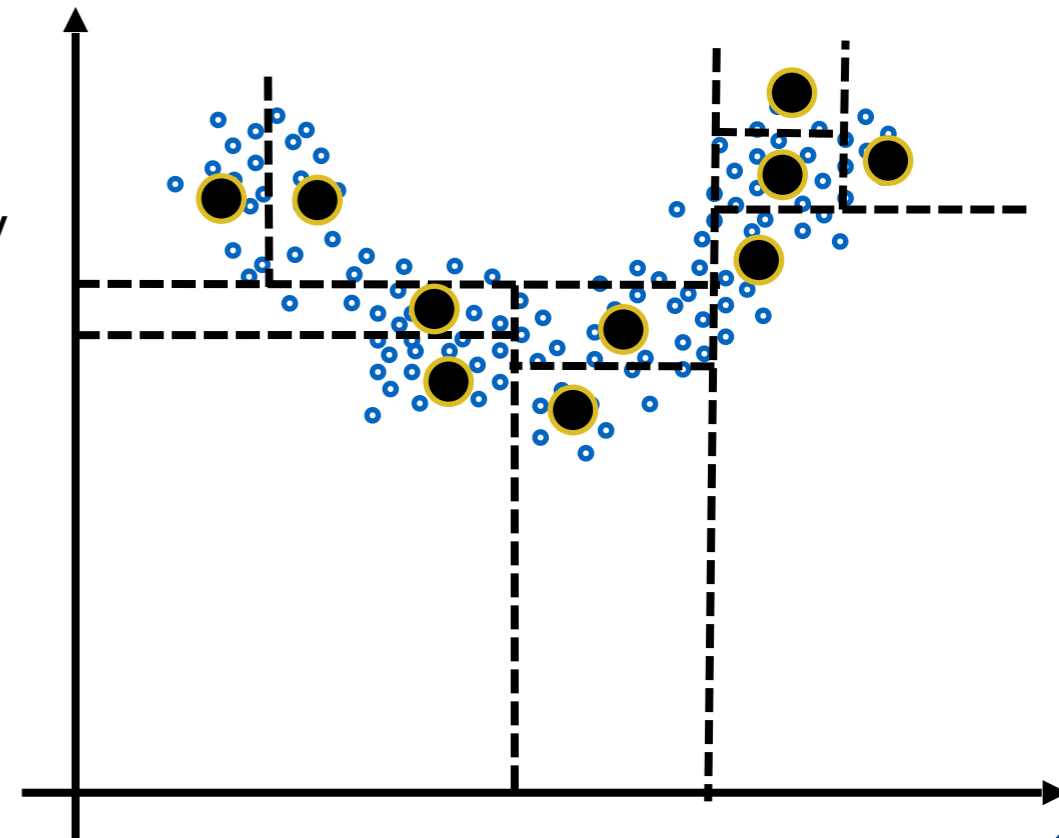
## MNIST example shows capability

- Input space  
784 variables of 8-bits = 6272 bits
- Latent space  
1 variable of 20-bit = 20 bits
- Compression = 314x
- Physics compression  
Looking for collaborators



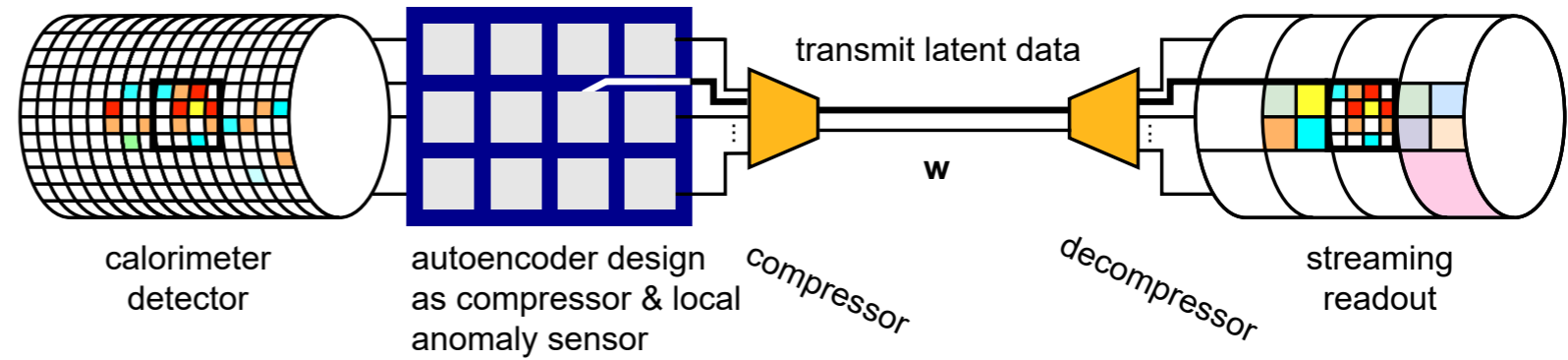
## Interpretability

- Learning is based on transparent density estimation of the input variable space  
Representative coordinates of a bin is the median value of the training sample in the bin  
Latent space data is the bin number
- Train on the fly? Sample 1d histograms  
Looking for collaborators

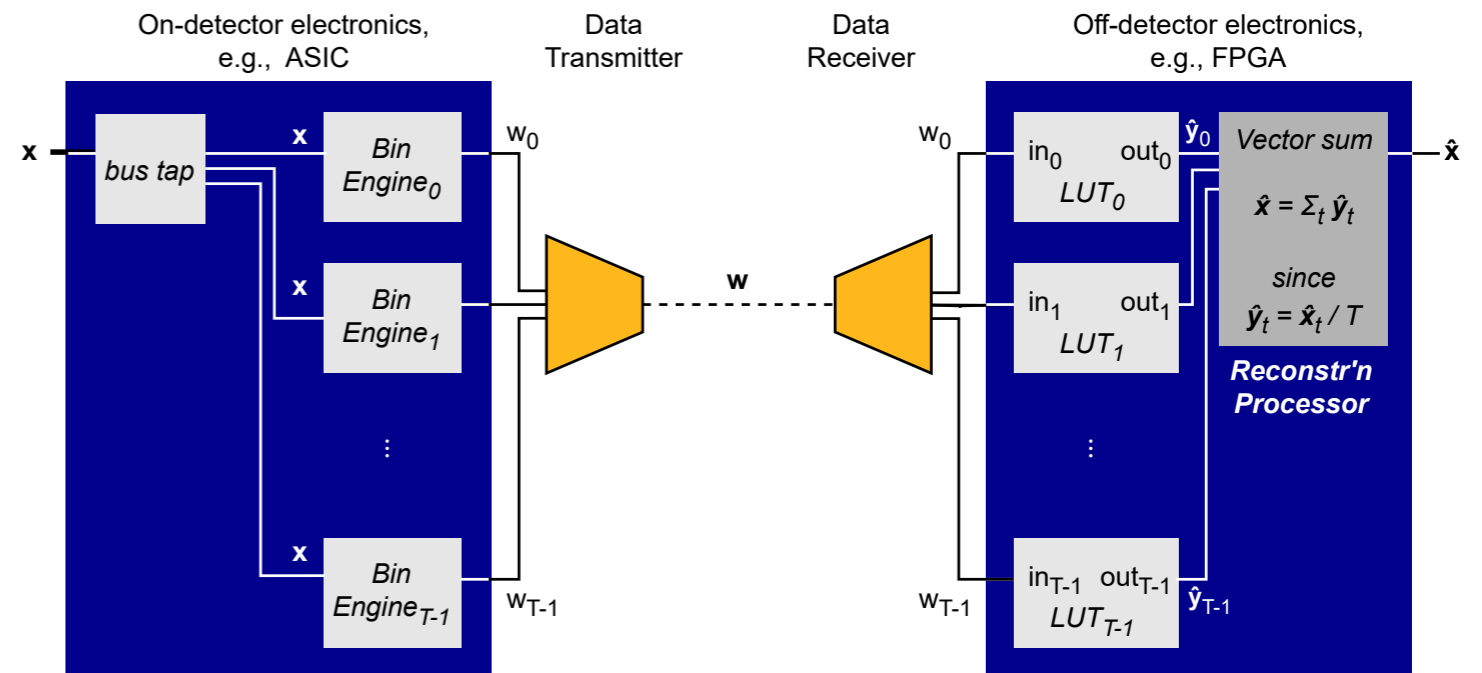




## Regional compression

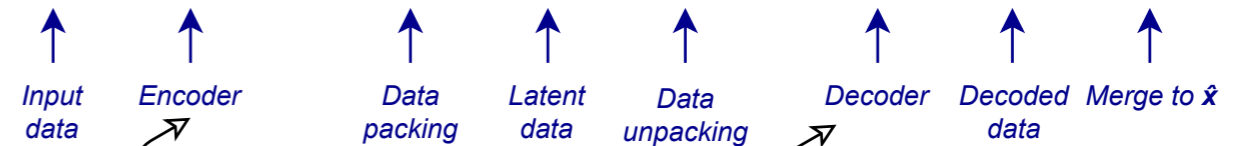


## Block diagram



Key question:

How to achieve dynamic compression



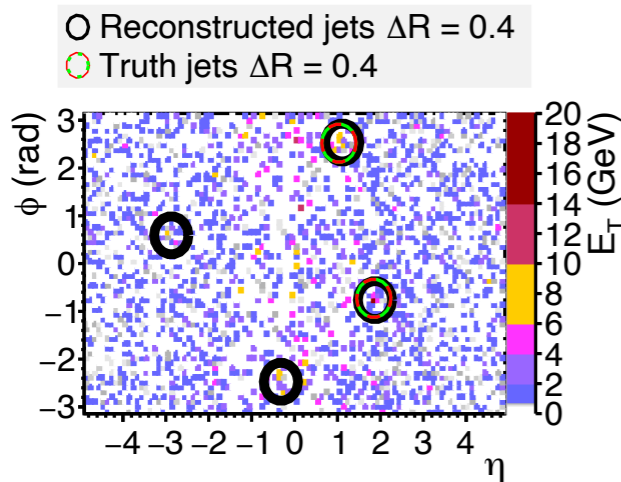
Modified Deep Decision Tree Engine (DDTE) is split up into two parts



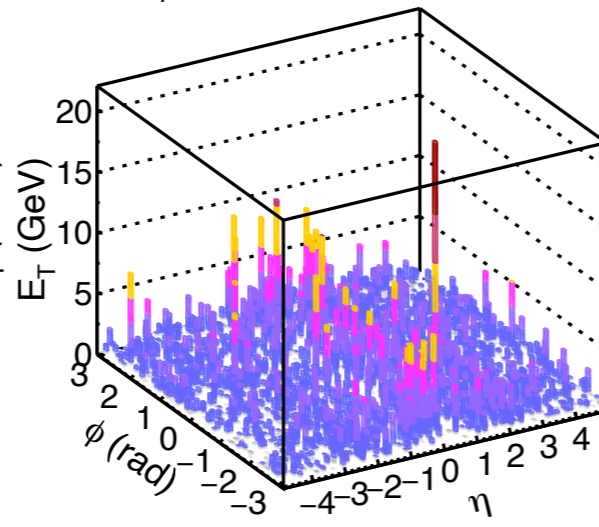
## Prototype study with Prof. B. Carlson Westmont College

- Look at jets at LHC pileup=200, sum energy in 4 rings around seed

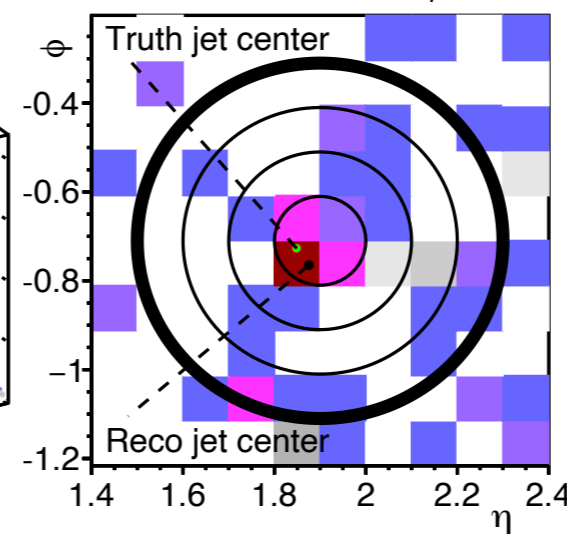
One simulated event #8,  $\langle \mu \rangle = 50$   
Delphes ATLAS  $0.1 \times 0.1$  towers



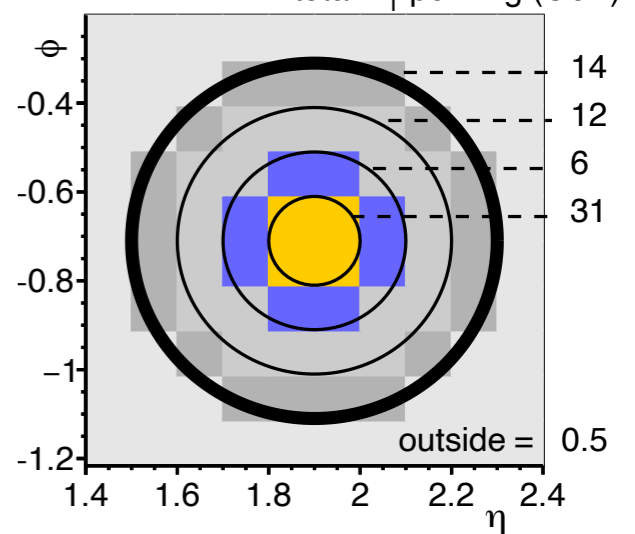
Tower  $E_T$  representation



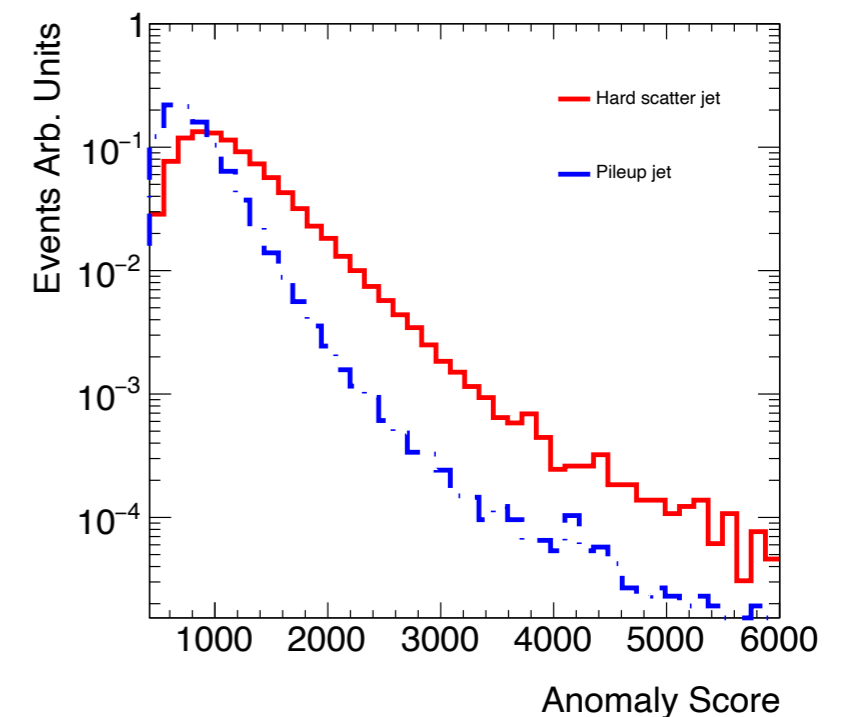
Zoom-in highest- $\phi$  jet,  $E_T = 63$  GeV



Ring definition total  $E_T$  per ring (GeV)



- Train DT autoencoder on pileup jets
- Hard scatter jets are anomalous wrt pileup
- Compression would depend on anomaly



# Outline

## Introduction

- Who we are

## FPGA design

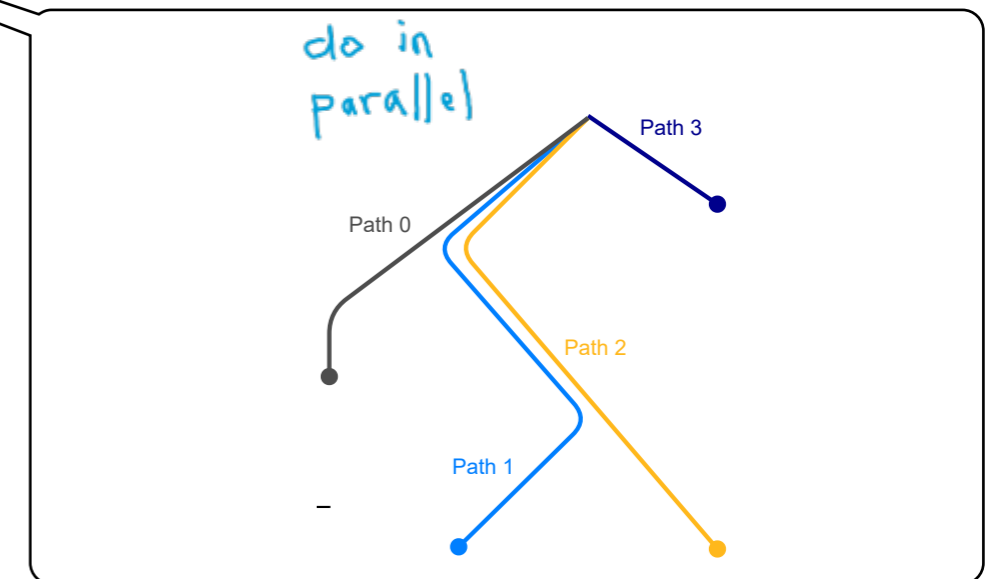
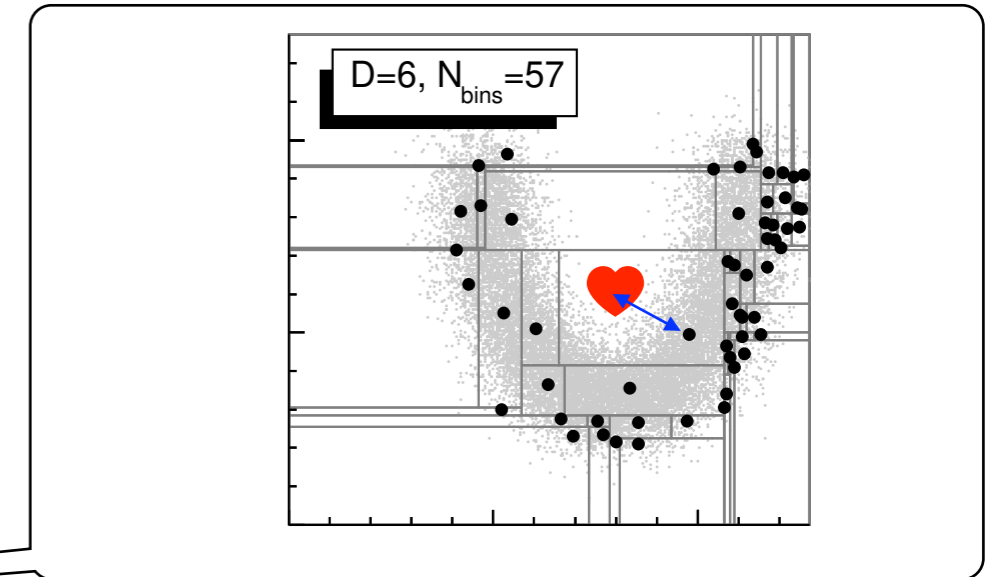
- Autoencoder
- Parallelizing decision trees
- HLS trees → VHDL trees

## Thoughts on SRO

- Data compression
- Anomaly detection

## More info

- Relevant papers from us
- Where to find code, tutorials



# Python-based code

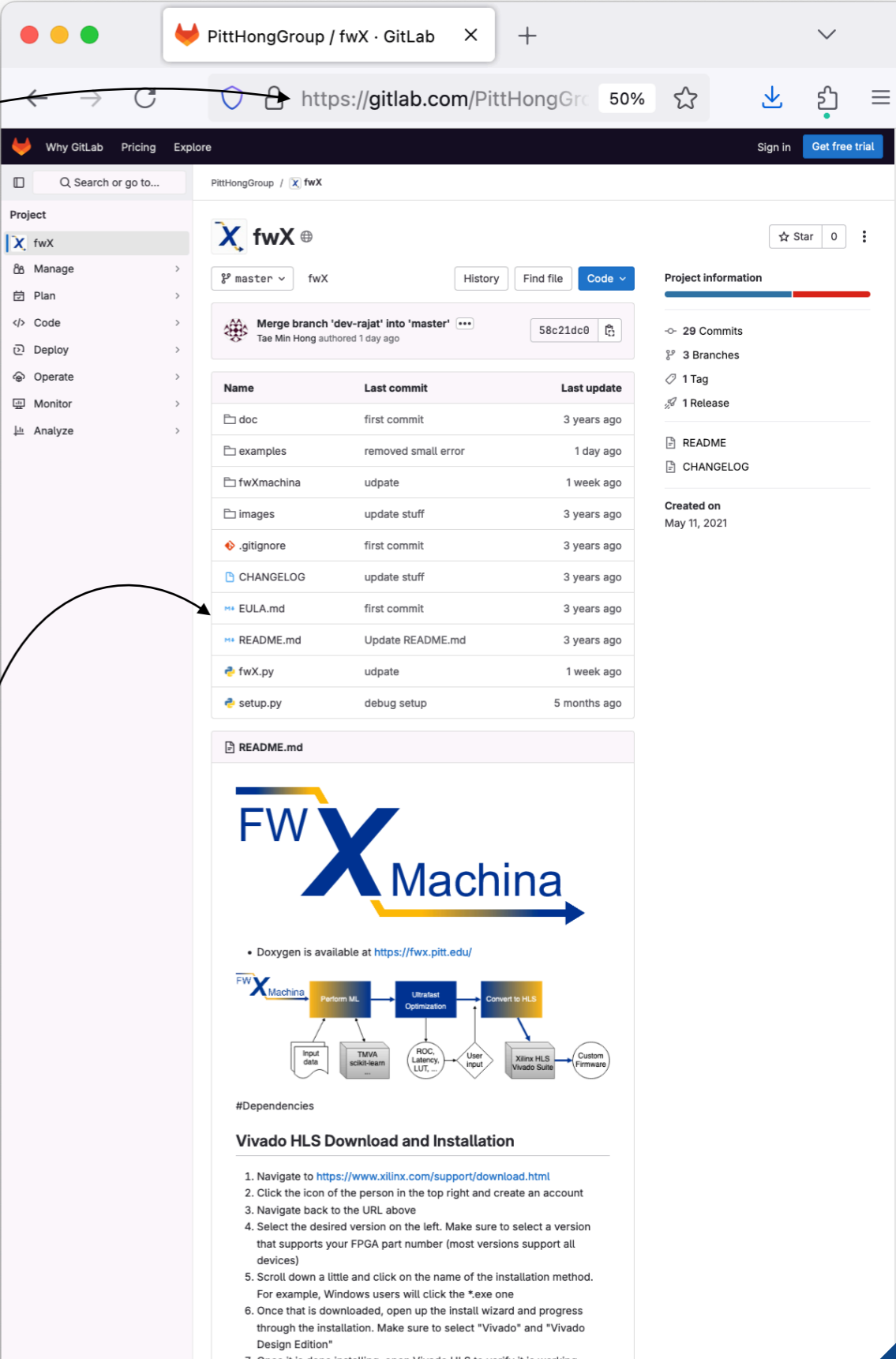
## Availability

- [gitlab.com/PittHongGroup/fwX](https://gitlab.com/PittHongGroup/fwX)  
parallel cuts (paper 1)
- Shared by email request  
parallel paths (paper 2)  
autoencoder (paper 3)  
hardware tree (paper 4)

## Licensing

- Will share for “Non-Commercial, Educational and Research Purposes”
- For commercial use, contact Univ. of Pittsburgh Innovation Institute
- See EULA for details

Skip this  
slide



The screenshot shows the GitLab interface for the repository `PittHongGroup / fwX`. The main content area displays a table of files and folders:

Name	Last commit	Last update
doc	first commit	3 years ago
examples	removed small error	1 day ago
fwXmachina	update	1 week ago
images	update stuff	3 years ago
.gitignore	first commit	3 years ago
CHANGELOG	update stuff	3 years ago
EULA.md	first commit	3 years ago
README.md	Update README.md	3 years ago
fwX.py	update	1 week ago
setup.py	debug setup	5 months ago

The README.md content includes the **FWX Machina** logo and a flowchart illustrating the process: `Input data` → `TMVA scikit-learn` → `Perform ML` → `Ultrafast Optimization` → `Convert to HLS` → `Xilinx HLS Vivado Suite` → `Custom Firmware`. The flowchart also shows `ROC, Latency, LUT, ...` as an output of the optimization step.

The README also provides instructions for downloading and installing Vivado HLS:

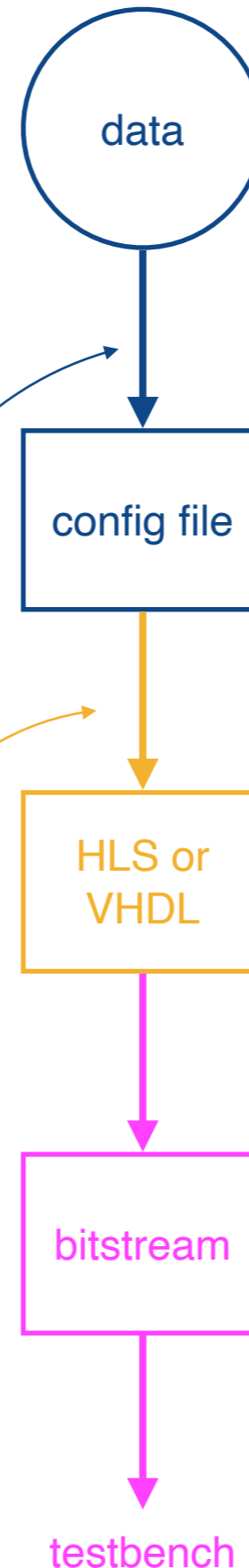
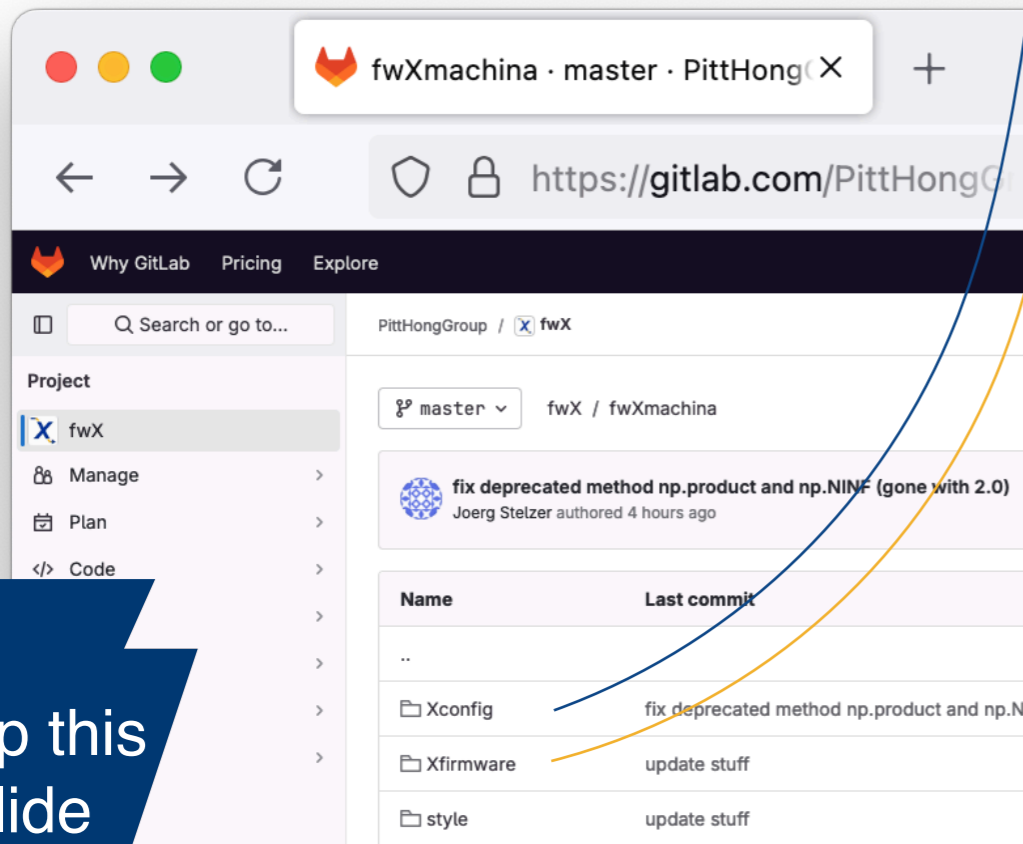
### Vivado HLS Download and Installation

1. Navigate to <https://www.xilinx.com/support/download.html>
2. Click the icon of the person in the top right and create an account
3. Navigate back to the URL above
4. Select the desired version on the left. Make sure to select a version that supports your FPGA part number (most versions support all devices)
5. Scroll down a little and click on the name of the installation method. For example, Windows users will click the \*.exe one
6. Once that is downloaded, open up the install wizard and progress through the installation. Make sure to select "Vivado" and "Vivado Design Edition"
7. Once it is done installing, open Vivado HLS to verify it is working

# Git structure

Same structure for all methods

- [gitlab.com/PittHongGroup/fwX](https://gitlab.com/PittHongGroup/fwX)  
parallel cuts (paper 1) - tutorial today
- Available by request
  - parallel paths (paper 2)
  - autoencoder (paper 3)
  - hardware tree (paper 4)



- Xconfig  
creates model configuration  
**tutorial - part 1**
- Xfirmware  
writes HLS or VHDL  
**tutorial - part 2**
- Vivado  
synthesize & testbench  
**tutorial - part 3**

Skip this  
slide



# More info

## Start page

- [fwx.pitt.edu](https://fwx.pitt.edu)

## Content

Links to **papers**

Links to **talks**

Links to **datasets**

Links to **testbenches**

3	Anomaly detection with end-to-end decision tree-based autoencoder in HLS	<ul style="list-style-type: none"> <li>• fwXmachina example: Anomaly detection, Mendeley Data, doi: <a href="https://doi.org/10.17632/y69855kscs.1">10.17632/y69855kscs.1</a> (2023-04-11). This sample is used in v1 of the paper draft [arXiv:2304.03836v1]</li> <li>• fwXmachina example: Anomaly detection for two photons and two jets, Mendeley Data, doi: <a href="https://doi.org/10.17632/444976dyrj.1">10.17632/444976dyrj.1</a> (2024-02-05). This sample is used in the final version of the paper.</li> </ul>	<ul style="list-style-type: none"> <li>• Python: Available upon request</li> <li>• IP testbench: Xilinx inputs for nanosecond anomaly detection with decision trees, <a href="http://d-scholarship.pitt.edu/id/eprint/44431">http://d-scholarship.pitt.edu/id/eprint/44431</a> (2023-04-23). This testbench is used in v1 of the paper draft [arXiv:2304.03836v1]</li> <li>• IP testbench: Xilinx inputs for nanosecond anomaly detection with decision trees for two photons and two jets, <a href="http://d-scholarship.pitt.edu/id/eprint/45784">http://d-scholarship.pitt.edu/id/eprint/45784</a> (2024-02-01). This testbench is used in the final version of the paper.</li> </ul>
4	Application in ATLAS Upgrade	-	-

### Talks / Posters

#	Date	Type: Title	Venue / Link	Speaker
1	2021-05-24	Talk: Comparisons to hls4ml's boosted decision tree results	Phenomenology Symposium, Pheno 2021, <a href="#">indico</a>	T.M. Hong
2	2021-06-06	Poster: Nanosecond machine learning with BDT for high energy physics	Virtual HEP conference on Run4@LHC, Offshell 2021, <a href="#">indico</a>	B.T. Carlson
3	2021-07-13	Talk: Nanosecond machine learning with BDT for high energy physics	Division of Particles and Fields (DPF) in the American Physical Society (APS), <a href="#">indico</a>	B.T. Carlson
4	2021-09-28	Seminar: Invisible Higgs decays & trigger challenges at the LHC	University of Geneva, Switzerland	T.M. Hong
5	2021-10-18	Talk: Presentation of fwX BDT	18th Int'l Conf. on Accelerator and Large Experimental Physics Control Systems, ICALEPCS 2021, <a href="#">indico</a>	S.T. Roche
6	2021-10-22	Seminar: Machine learning in real-time triggers at the LHC: A discussion on Machine learning, Boosted decision trees, Real-time trigger, and ML on FPGA	Department of Physics, University of Tennessee, Knoxville	T.M. Hong
7	2021-10-20	Poster: Presentation of fwX BDT	IEEE Nuclear Science Symposium and Medical Imaging Conference, 2021 IEEE NSS MIC, <a href="#">link</a>	S.T. Racz
8	2021-12-04	Talk: Comparisons of fwX's BDT to hls4ml's neural network results	PIKIMO 11, <a href="#">indico</a>	T.M. Hong
9	2023-05-12	Talk: Decision tree autoencoder anomaly detection on FPGA at L1 triggers	Phenomenology Symposium, Pheno 2023, <a href="#">indico</a>	S.T. Roche
10	2023-09-25	Talk: fwXmachina part 1: Classification with boosted decision trees on FPGA for L1 trigger	Fast Machine Learning for Science Workshop 2023, <a href="#">indico</a>	T.M. Hong

## Tutorial

- **SMARTHEP** Edge ML School 9/24/24

Slides

[indico.cern.ch/event/1405026/contributions/6103378/](https://indico.cern.ch/event/1405026/contributions/6103378/)

Videos on synthesizing & test bench

[indico.cern.ch/event/1405026/contributions/6103386/](https://indico.cern.ch/event/1405026/contributions/6103386/)

# Conclusion

## Introduction

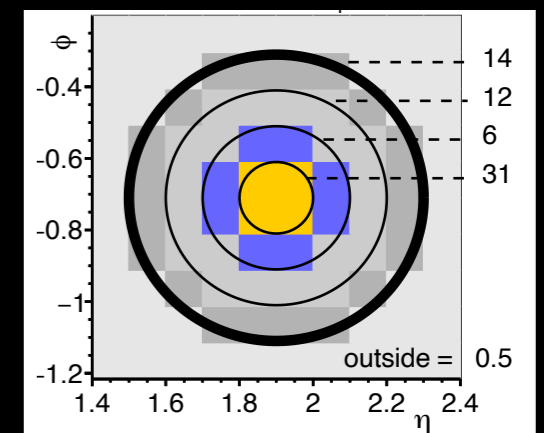
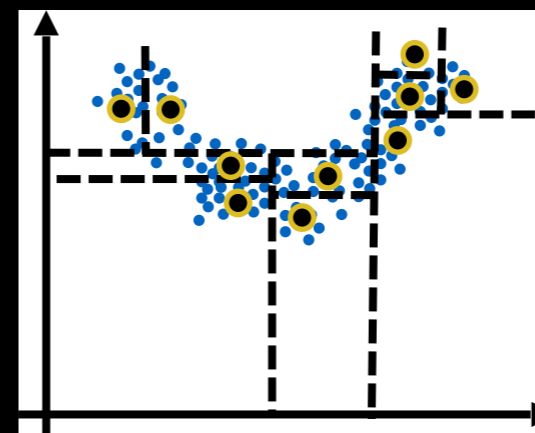
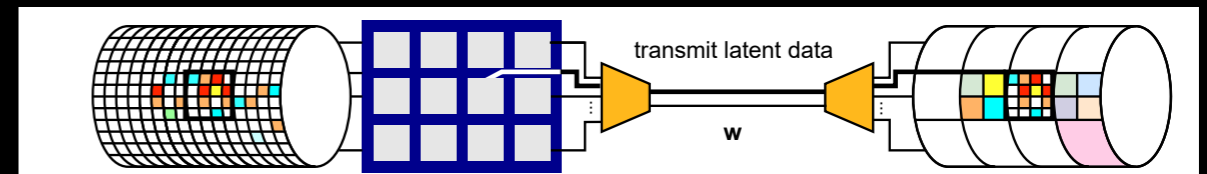
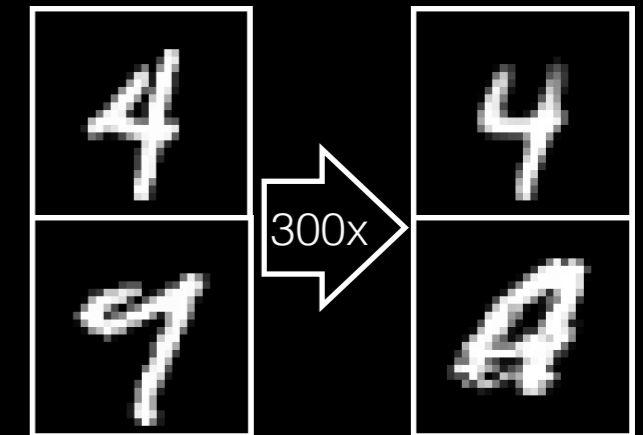
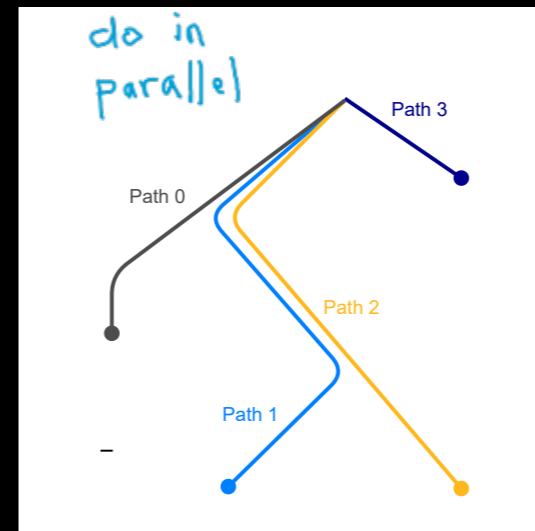
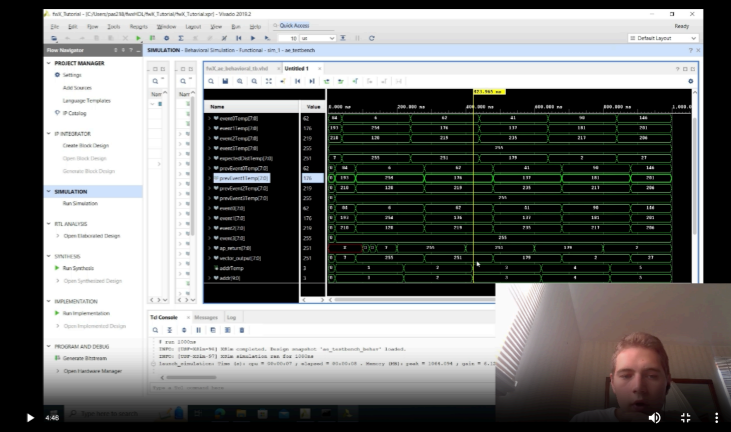
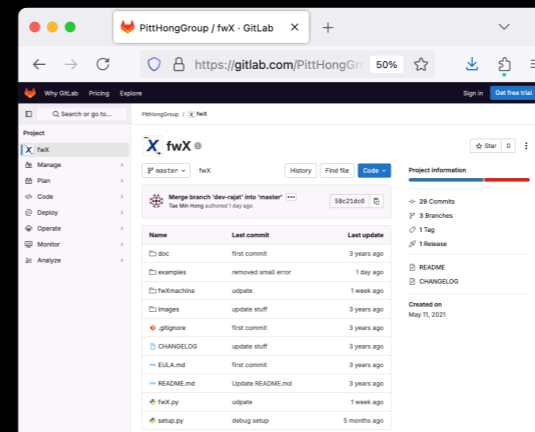
- Papers

## FPGA design

- Decision tree autoencoder
- Parallel decision trees in VHDL

## Thoughts on SRO

- Transparent interpretation
- 30 ns data compression
- 30 ns anomaly detection



*We're excited & open to collaboration*

Backup

## 1. Classification parallel cuts using HLS

**Jinst** PUBLISHED BY IOP PUBLISHING FOR SISSA MEDIALAB

RECEIVED: April 9, 2021  
ACCEPTED: June 29, 2021  
PUBLISHED: August 4, 2021

**Nanosecond machine learning event classification with boosted decision trees in FPGA for high energy physics**

T.M. Hong,<sup>a</sup> B.T. Carlson, B.R. Eubanks, S.T. Roche, J. Stelzer and D.C. Stump

<sup>a</sup>Department of Physics and Astronomy, University of Pittsburgh, 100 Allen Hall, 3941 O'Hara St., Pittsburgh, PA 15260, U.S.A.  
E-mail: tmhong@pitt.edu

ABSTRACT: We present a novel implementation of classification using the machine learning/artificial intelligence method called boosted decision trees (BDT) on field programmable gate arrays (FPGA). The firmware implementation of binary classification requiring 100 training trees with a maximum depth of 4 using four input variables gives a latency value of about 10 ns, independent of the clock speed from 100 to 320 MHz in our setup. The low timing values are achieved by restructuring the BDT layout and reconfiguring its parameters. The FPGA resource utilization is also kept low at a range from 0.01% to 0.2% in our setup. A software package called **FWXACHIMA** achieves this implementation. Our intended user is an expert in custom electronics-based trigger systems in high energy physics experiments or anyone that needs decisions at the lowest latency values for real-time event classification. Two problems from high energy physics are considered, in the separation of electrons vs. photons and in the selection of vector boson fusion-produced Higgs bosons vs. the rejection of the multijet processes.

KEYWORDS: Digital electronic circuits; Trigger algorithms; Trigger concepts and systems (hardware and software); Data reduction methods

ARXIV EPRINT: 2104.03408

\*Corresponding author.

© 2021 IOP Publishing Ltd and Sissa Medialab <https://doi.org/10.1088/1748-0221/16/08/P08016>

2021 JINST 16 P08016

Hong et al.  
JINST 16, P08016 (2021)  
<http://doi.org/10.1088/1748-0221/16/08/P08016>

## 2. Regression parallel paths using HLS

**Jinst** PUBLISHED BY IOP PUBLISHING FOR SISSA MEDIALAB

RECEIVED: July 13, 2022  
ACCEPTED: August 23, 2022  
PUBLISHED: September 27, 2022

**Nanosecond machine learning regression with deep boosted decision trees in FPGA for high energy physics**

B.T. Carlson,<sup>a,b</sup> Q. Bayer,<sup>b</sup> T.M. Hong<sup>b,\*</sup> and S.T. Roche<sup>b</sup>

<sup>a</sup>Department of Physics and Engineering, Westmont College, 955 La Paz Road, Santa Barbara, CA 93108, U.S.A.  
<sup>b</sup>Department of Physics and Astronomy, University of Pittsburgh, 100 Allen Hall, 3941 O'Hara St., Pittsburgh, PA 15260, U.S.A.  
E-mail: tmhong@pitt.edu

ABSTRACT: We present a novel application of the machine learning / artificial intelligence method called boosted decision trees to estimate physical quantities on field programmable gate arrays (FPGA). The software package **FWXACHIMA** features a new architecture called parallel decision paths that allows for deep decision trees with arbitrary number of input variables. It also features a new optimization scheme to use different numbers of bits for each input variable, which produces optimal physics results and ultraefficient FPGA resource utilization. Problems in high energy physics of proton collisions at the Large Hadron Collider (LHC) are considered. Estimation of missing transverse momentum ( $E_T^{\text{miss}}$ ) at the first level trigger system at the High Luminosity LHC (HL-LHC) experiments, with a simplified detector modeled by Delphes, is used to benchmark and characterize the firmware performance. The firmware implementation with a maximum depth of up to 10 using eight input variables of 16-bit precision gives a latency value of  $O(10)$  ns, independent of the clock speed, and  $O(0.1\%)$  of the available FPGA resources without using digital signal processors.

KEYWORDS: Data reduction methods; Digital electronic circuits; Trigger algorithms; Trigger concepts and systems (hardware and software)

ARXIV EPRINT: 2207.05602

\*Corresponding author.

© 2022 IOP Publishing Ltd and Sissa Medialab <https://doi.org/10.1088/1748-0221/17/09/P09039>

2022 JINST 17 P09039

Carlson et al.  
JINST 17, P09039 (2022)  
<http://doi.org/10.1088/1748-0221/17/09/P09039>

## 3. Autoencoder in-house training bypassing latent space

nature communications

Article <https://doi.org/10.1038/s41467-024-47704-8>

**Nanosecond anomaly detection with decision trees and real-time application to exotic Higgs decays**

S. T. Roche<sup>1,2</sup>, Q. Bayer<sup>2</sup>, B. T. Carlson<sup>2,3</sup>, W. C. Ouljian<sup>2</sup>, P. Serhiayenka<sup>2</sup>, J. Stelzer<sup>2</sup> & T. M. Hong<sup>2</sup>\*

Received: 23 May 2024  
Accepted: 9 April 2024  
Published online: 25 April 2024

Check for updates

We present an interpretable implementation of the autoencoding algorithm, used as an anomaly detector, built with a forest of deep decision trees on FPGA, field programmable gate arrays. Scenarios at the Large Hadron Collider at CERN are considered, for which the autoencoder is trained using known physical processes of the Standard Model. The design is then deployed in real-time trigger systems for anomaly detection of unknown physical processes, such as the detection of rare exotic decays of the Higgs boson. The inference is made with a latency value of 30 ns at percent-level resource usage using the Xilinx Virtex UltraScale+ VU9P FPGA. Our method offers anomaly detection at low latency values for edge AI users with resource constraints.

Supervised artificial intelligence (AI) algorithms enable signal agnostic searches beyond the Standard Model (SM) physics at the Large Hadron Collider (LHC) at CERN. The LHC is the highest energy proton and heavy ion collider that is designed to discover the Higgs boson<sup>1</sup> and study its properties<sup>2</sup> as well as to probe the unknown and undiscovered BSM physics (see, e.g.,<sup>3–5</sup>). Due to the lack of signs of BSM in the collected data despite the plethora of searches conducted at the LHC, dedicated studies look for rare BSM events that are even more difficult to parse among the mountain of ordinary Standard Model processes<sup>6–8</sup>. An active area of AI research in high energy physics is in using autoencoders for anomaly detection, much of which provides methods to find rare and unanticipated BSM physics. Much of the existing literature, mostly using neural network-based approaches, focuses on identifying BSM physics in already collected data<sup>9–11</sup>. Such ideas have started to produce experimental results on the analysis of data collected at the LHC<sup>12–14</sup>. A related but separate endeavor, which is the subject of this paper, is enabling the identification of rare and anomalous data on the real-time trigger path for more detailed investigation offline.

The LHC offers an environment with an abundance of data at a 40 MHz collision rate, corresponding to the 25 ns time period between successive collisions. The real-time trigger path of the ATLAS and CMS experiments<sup>15–17</sup>, e.g., processes data using custom electronics using field programmable gate arrays (FPGA) followed by software trigger algorithms executed on a computing farm. The first-level FPGA portion of the trigger system accepts between 100 kHz to 1 MHz of collisions, discarding the remaining ~99% of the collisions. Therefore, it is essential to discover that the FPGA-based trigger system is capable of triggering potential BSM events. A previous study aimed at LHC data has shown that an anomaly detector based on neural networks can be implemented on FPGA with latency values between 80 to 1480 ns, depending on the design<sup>18</sup>.

In this paper, we present an interpretable implementation of an autoencoder using deep decision trees that make inferences in 30 ns. As discussed previously<sup>19</sup>, decision tree designs depend only on threshold comparisons resulting in fast and efficient FPGA implementation with minimal reliance on digital signal processors. We train the autoencoder on known Standard Model (SM) processes to help trigger the rare events that may include BSM.

In scenarios for which a specific BSM model is targeted and its dynamics are known, dedicated supervised training against the SM sample, i.e., BSM-vs-SM classification, would likely outperform an unsupervised approach of SM-only training. The physics scenarios considered in this paper are examples to demonstrate that our autoencoder is able to trigger on BSM scenarios as anomalies without this prior knowledge of the BSM specifics. Nevertheless, we consider a benchmark where our autoencoder outperforms the existing conventional cut-based algorithms.

<sup>1</sup>School of Medicine, Saint Louis University, Saint Louis, MO, USA. <sup>2</sup>Department of Physics and Astronomy, University of Pittsburgh, Pittsburgh, PA, USA. <sup>3</sup>Department of Physics and Engineering, Westmont College, Santa Barbara, CA, USA. ✉e-mail: tmhong@pitt.edu

Nature Communications | (2024)15:3527

Roche et al.  
Nat. Comm. 15 (2024) 3527  
<https://arxiv.org/abs/2304.03836>

## 4. Hardware trees faster & more efficient no more HLS

PITT-PACC-2409-v1

**Nanosecond hardware regression trees in FPGA at the LHC**

P. Serhiayenka<sup>a</sup>, S. T. Roche<sup>a,b</sup>, B. T. Carlson<sup>a,c</sup>, and T. M. Hong<sup>a,\*</sup>

<sup>a</sup>Department of Physics and Astronomy, University of Pittsburgh  
<sup>b</sup>School of Medicine, Saint Louis University  
<sup>c</sup>Department of Physics and Engineering, Westmont College

September 20, 2024

Abstract

We present a generic parallel implementation of the decision tree-based machine learning (ML) method in hardware description language (HDL) on field programmable gate arrays (FPGA). A regression problem in high energy physics at the Large Hadron Collider is considered: the estimation of the magnitude of missing transverse momentum using boosted decision trees (BDT). A forest of twenty decision trees each with a maximum depth of 10 using eight input variables of 16-bit precision is executed with a latency of about 10 ns using  $O(0.1\%)$  resources on Xilinx UltraScale+ VU9P—approximately ten times faster and five times smaller compared to similar designs using high level synthesis (HLS)—without the use of digital signal processors (DSP) while eliminating the use of block RAM (BRAM). We also demonstrate a potential application in the estimation of muon momentum for ATLAS RPC at HL-LHC.

Keywords: Data processing methods, Data reduction methods, Digital electronic circuits, Trigger algorithms, and Trigger concepts and systems (hardware and software).

\*Corresponding author, tmhong@pitt.edu

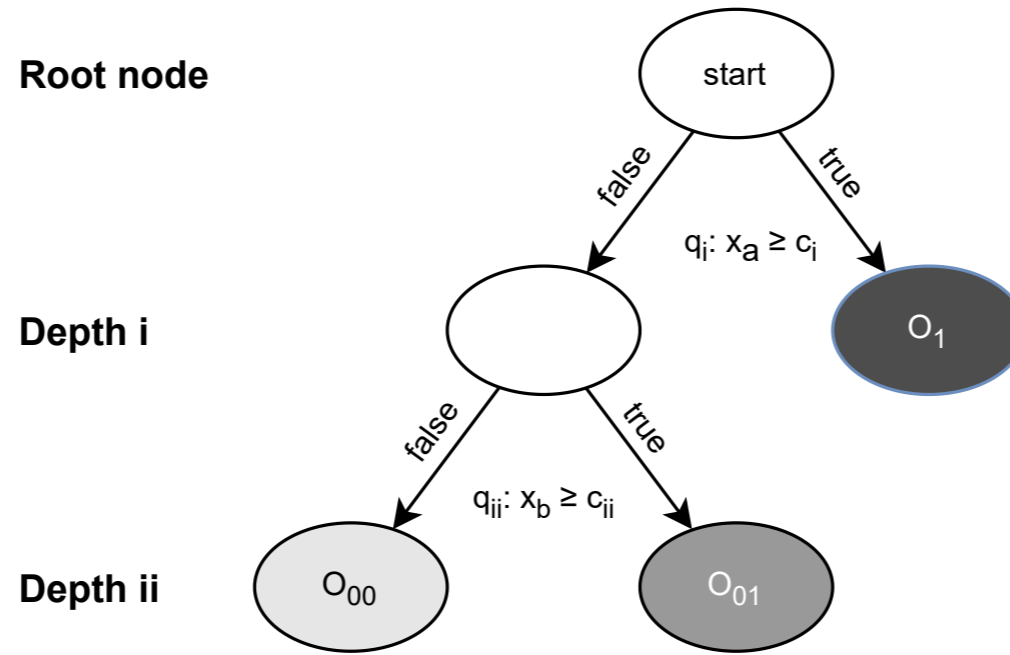
Serhiayenka et al.  
Submitted to NIM-A  
[2409.20506]

Skip this slide

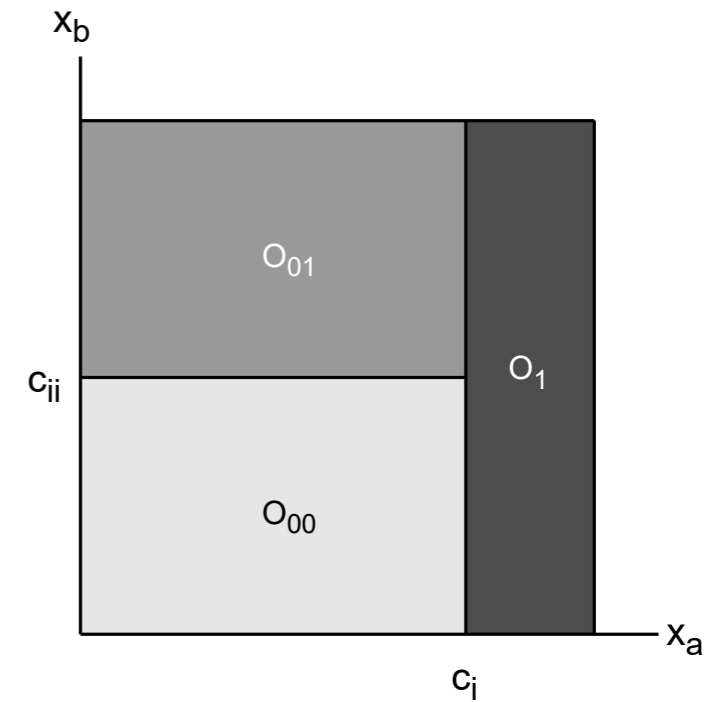
Focus today



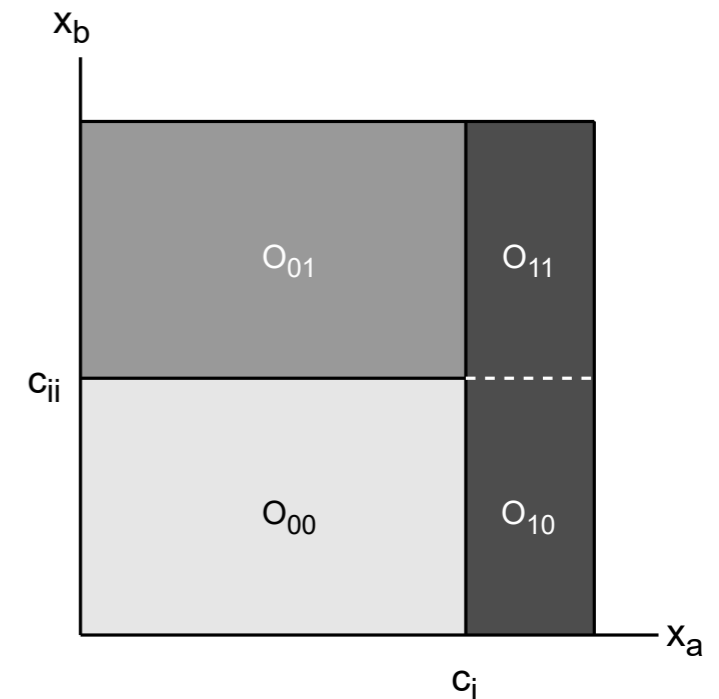
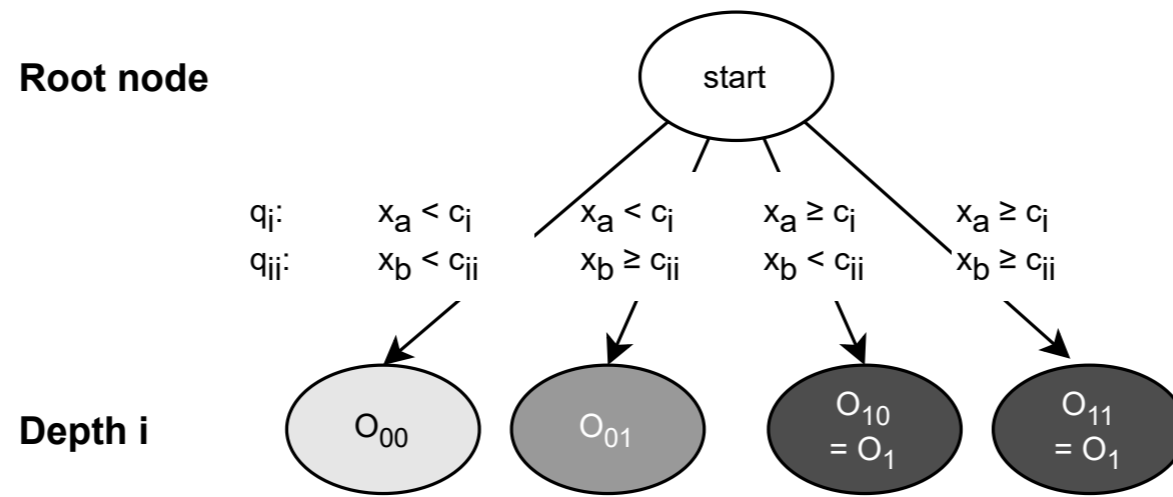
Decision tree



2d plane:  $x_a$  vs.  $x_b$

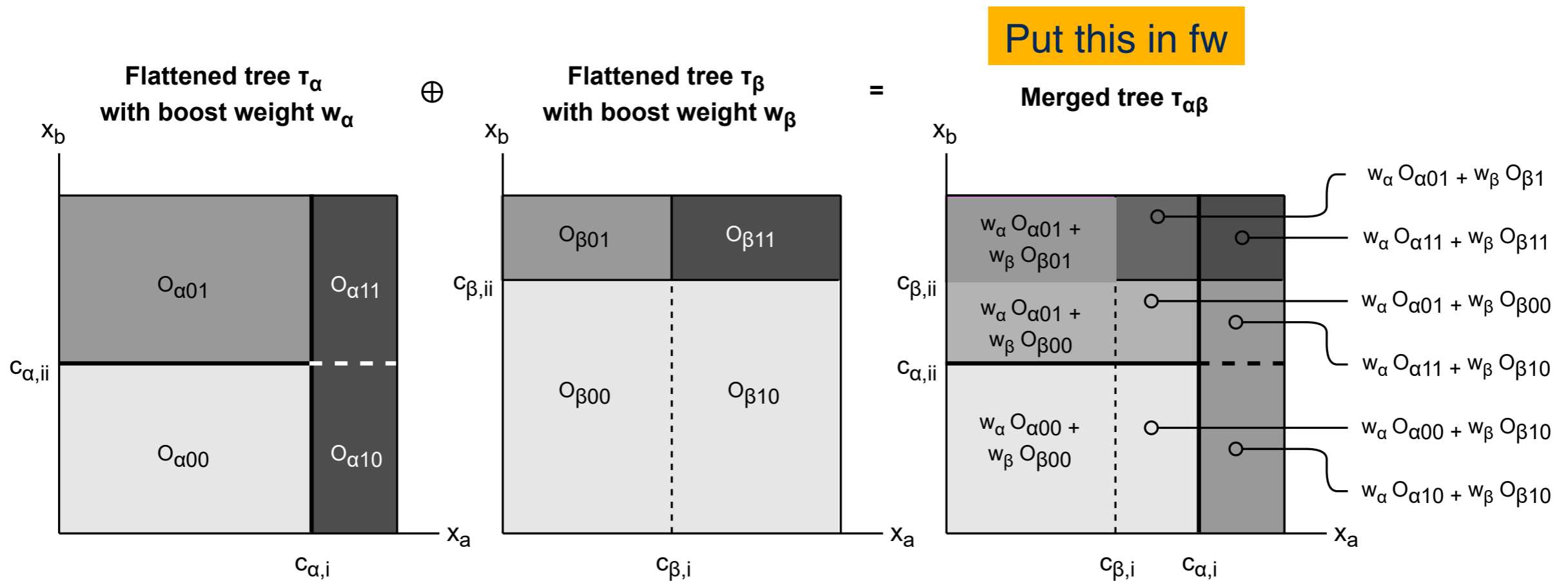


Parallelize cuts



Skip this slide

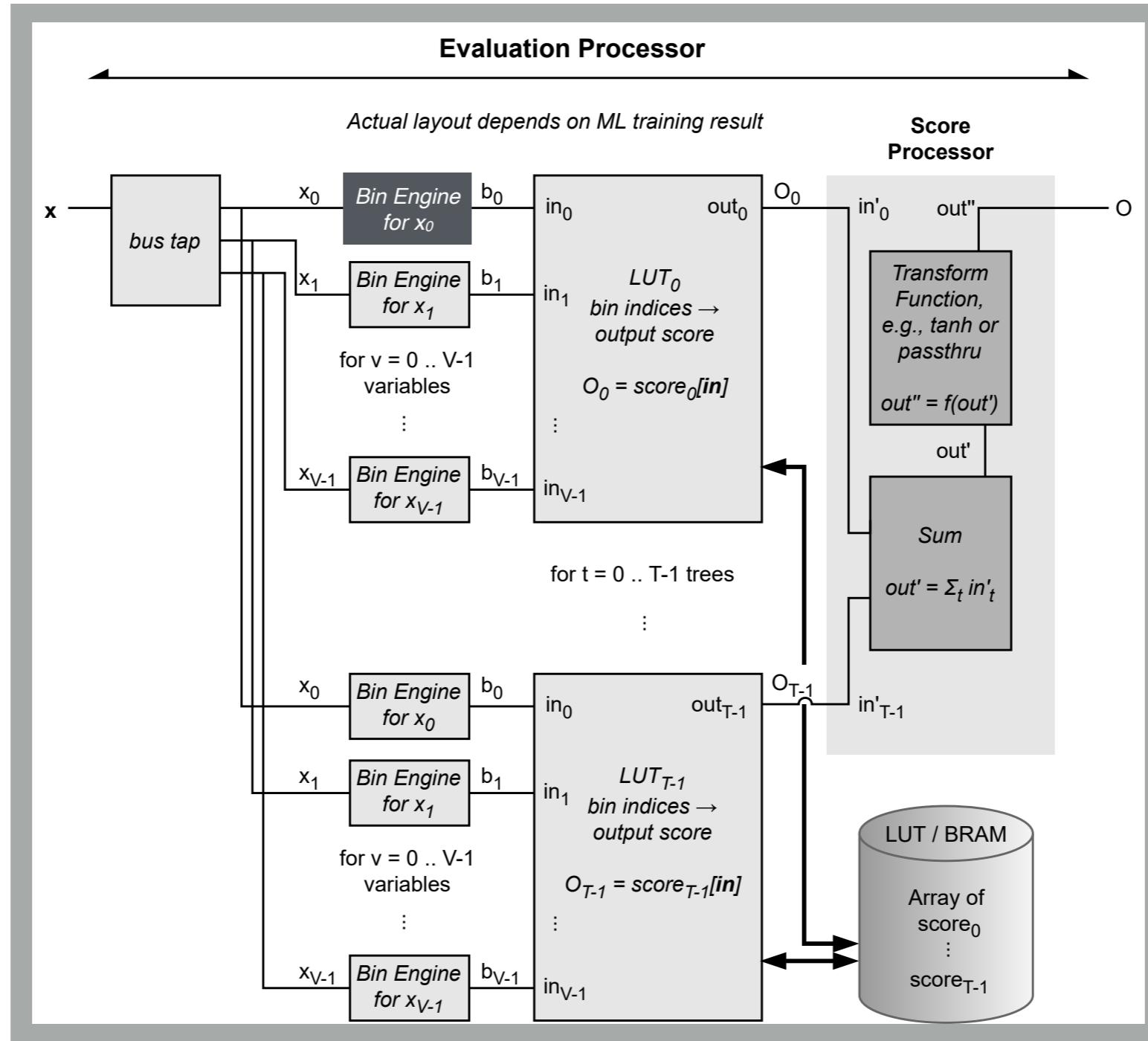
Key idea: Data on each axis can be binned in parallel



Key idea:

Forest can be merged prior to firmware implementation

Skip this slide



Skip this slide

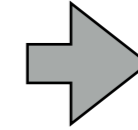
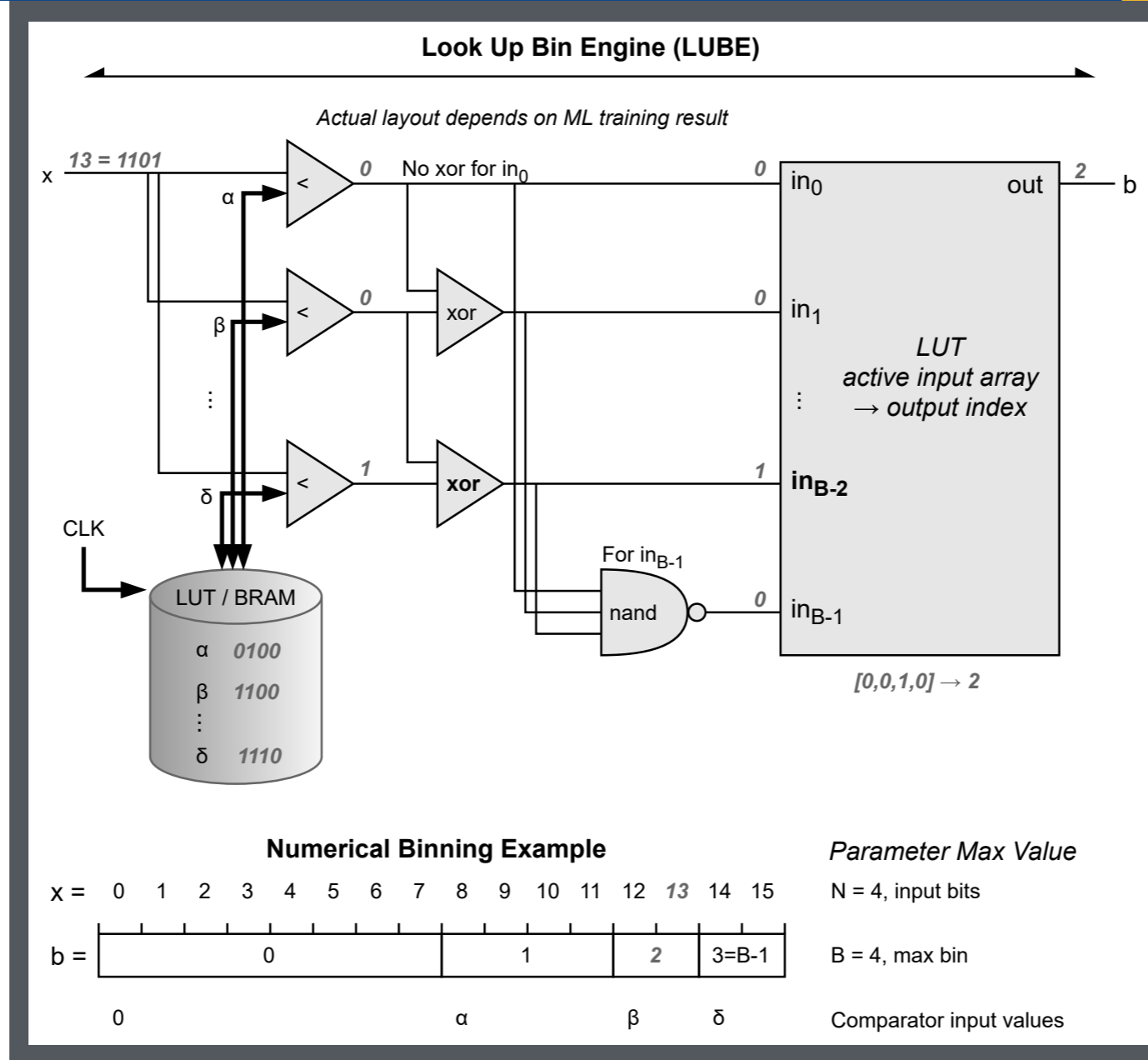
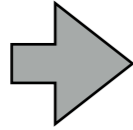
Parallel process per variable (next slide)

Look up the output score given per decision tree

Combine the scores of the trees in the forest



Input variable



Bin index

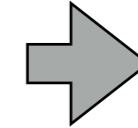
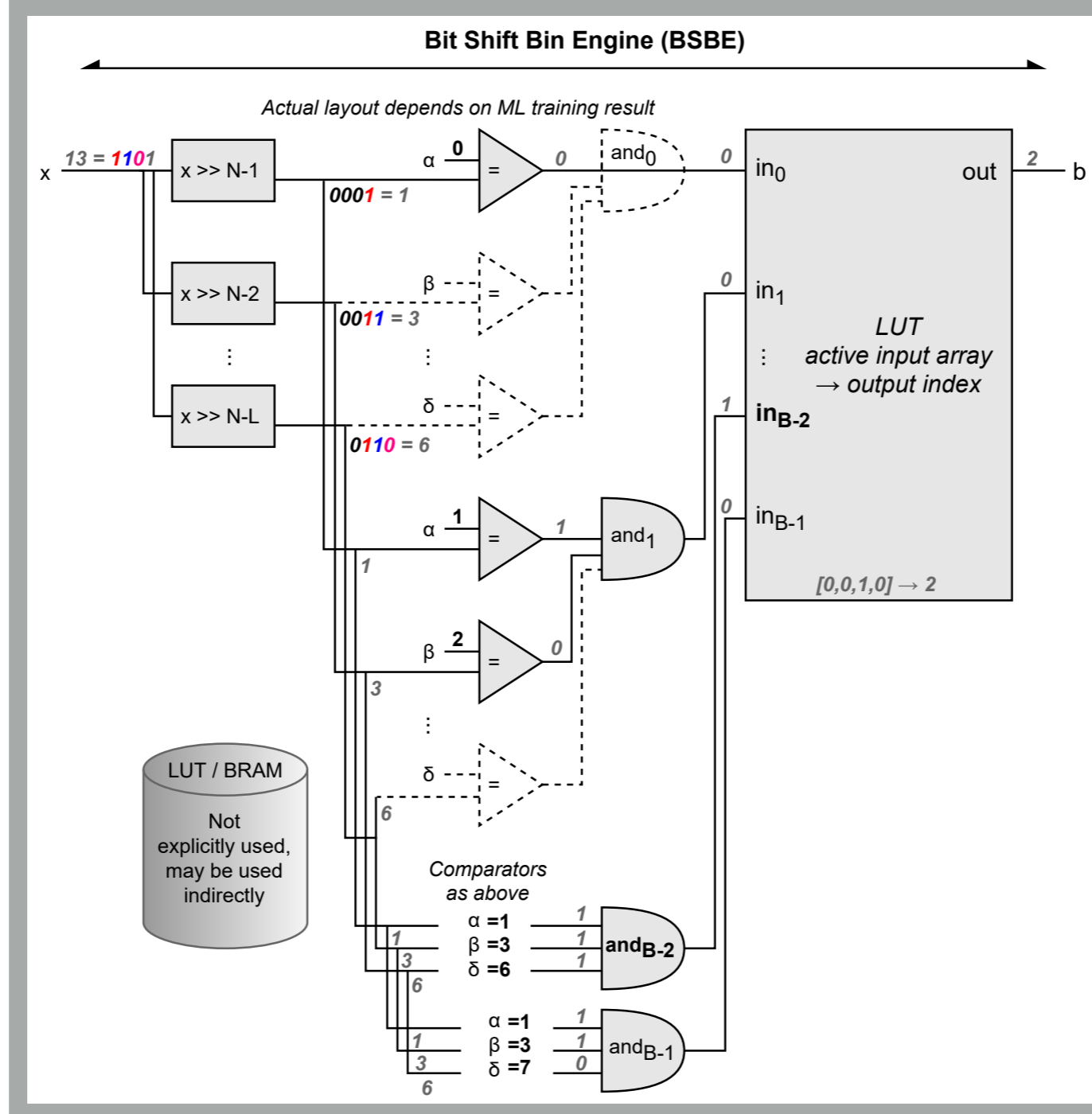
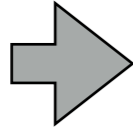
- Search for the bin where the data point lives

Skip this slide





Input variable



Bin index

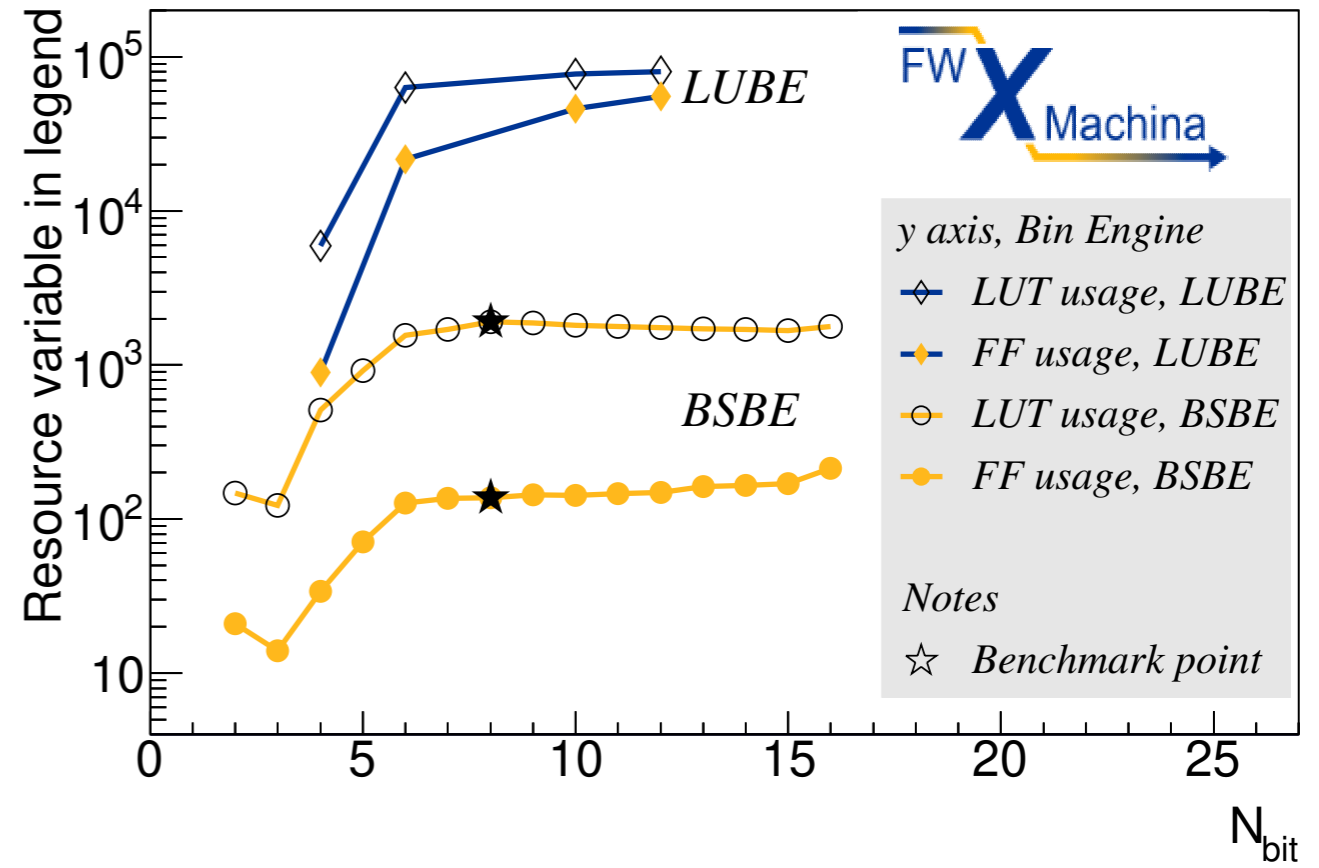
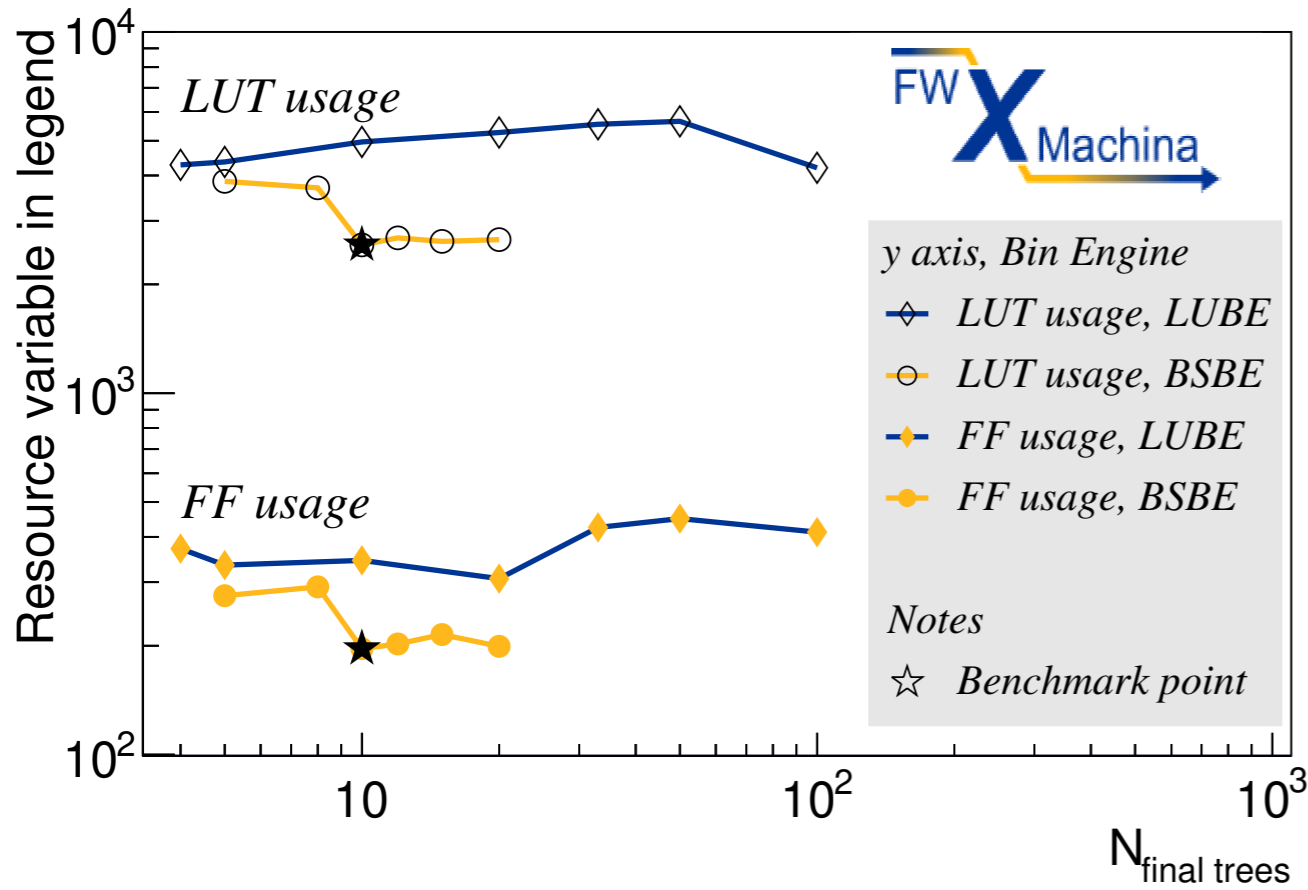
Numerical Binning Example

x =	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$\alpha =$	0							1								
$\beta =$	0		1		2		3									
$\delta =$	0	1	2	3	4	5	6	7	$7=2^L-1$							
b =	0			1			2		3 = B-1							

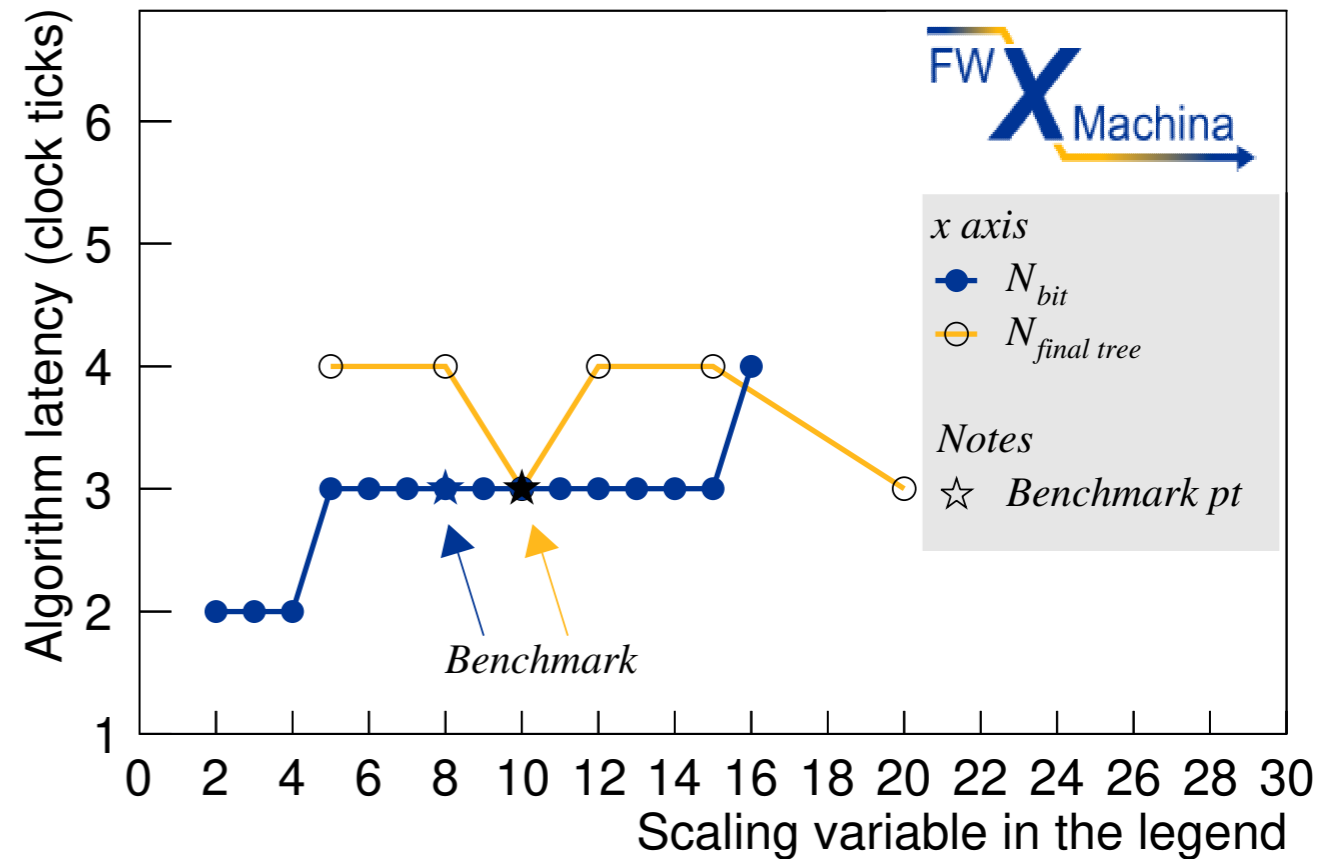
Parameter Max Value

- N = 4, input bits
- $\ell = 0$ , layer no.
- $\ell = 2$ , layer no.
- $\ell = 3 = L - 1$ , max layer
- B = 4, max bin

Skip this slide



- Does not scale with trees or bits



Skip this slide



- Setup

Physics

4 variables for e vs.  $\gamma$ <sup>1</sup>

FWX paper 1

100 trees, 4 deep

hls4ml BDT

” identical config for BDT

hls4ml NN

Out-of-the-box config

- fwX paper 1

vs. hls4ml NN

Comparable<sup>2</sup>

vs. hls4ml BDT

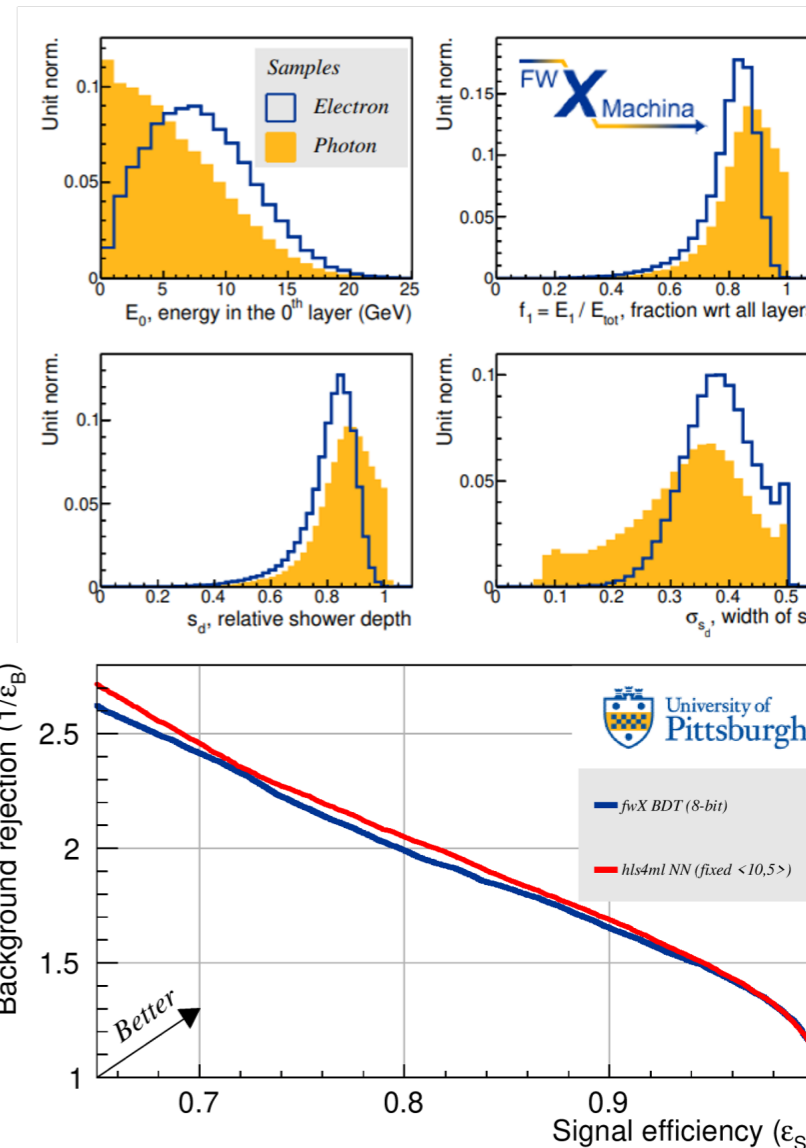
Same (since identical config)

Resource

< 1% for all methods

Latency

FWX's parallel + no clocked operations



	fwX paper 1	hls4ml NN	hls4ml conifer-BDT
# bits	< 8 >	< 10, 5 >	< 10, 5 >
LUT	0.06%	0.1%	0.3%
Flip Flops	0.01%	0.01%	0.1%
BRAM	0.1%	0.2%	0
DSP	0.03%	0.02%	0
Latency	10 ns	25 ns	47 ns
Interval	1 clock tick	1 clock tick	1 clock tick

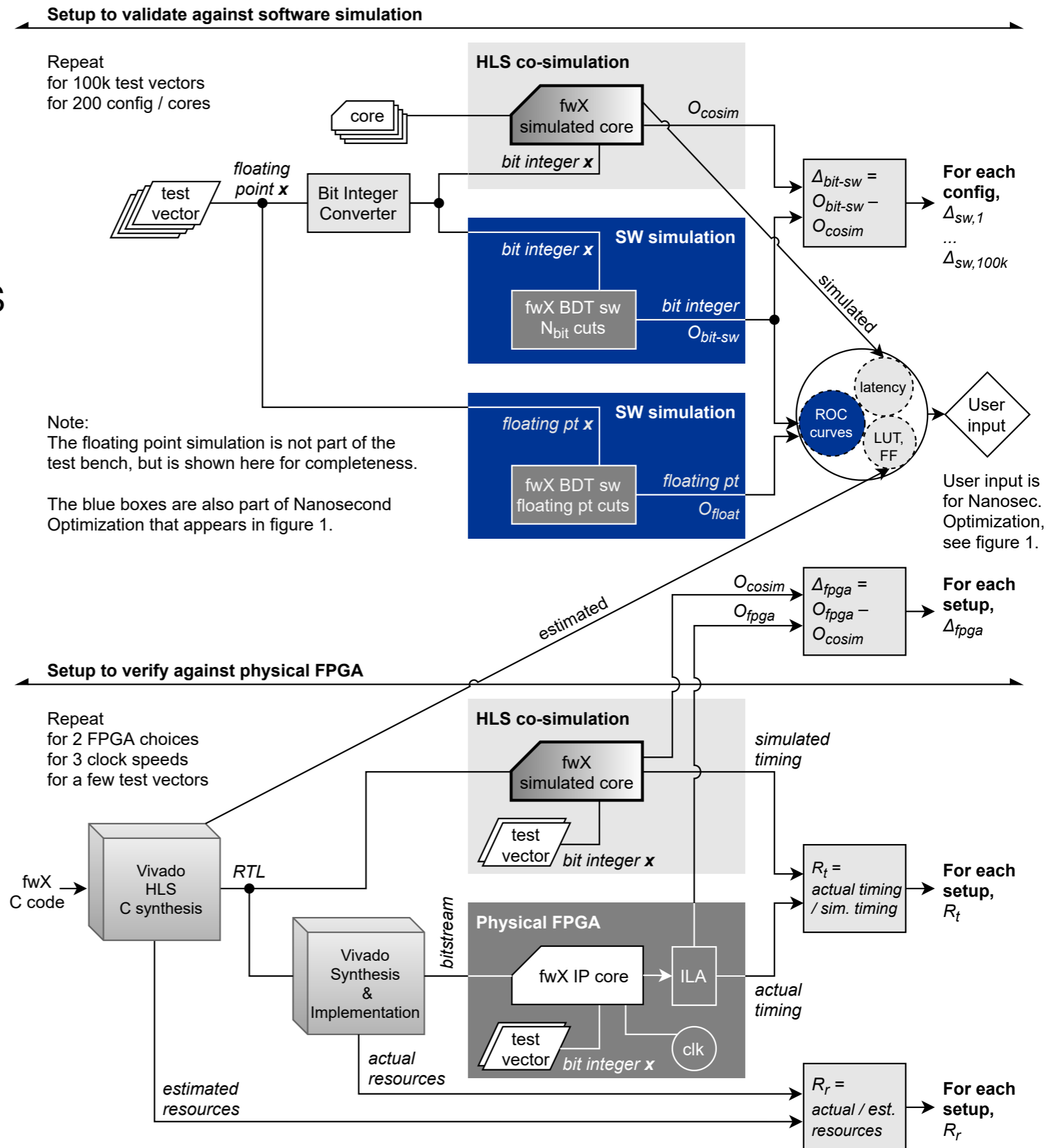
<sup>1</sup> Nachman et al., <https://data.mendeley.com/datasets/kp3myh3v89/1>

<sup>2</sup> Hong, PIKIMO 11, <https://indico.cern.ch/event/1091676/contributions/4639362/>

Skip this slide

## Philosophy

- Every training ships with test vectors
- Every design creates its own testbench
- Performance values from implementation, not estimate





## Compared

- Estimated usage / latency vs. actual usage / latency

Table 12: FPGA cost verification against physical FPGA. Comparison of the FPGA cost using the bitstream on the FPGA (actual), simulated timing using co-simulation and estimated resources using Vivado HLS (estimated). The actual-to-estimated ratios are given as  $R$ . Two FPGA choices and three clock speeds are considered; the 320 MHz group of columns represent the benchmark clock. For all other configurable parameters, see table 1. The timing values are reported in units of clock ticks. The Xilinx Vivado version used for the actual and estimated columns are noted. For the ratios, “1” signifies no difference.

Parameter	Benchmark FPGA						Smaller FPGA		
FPGA setup									
Family	Xilinx Virtex Ultrascale+ .....						Xilinx Artix-7 .....		
Model	xcvu9p-flga2104-2L-e .....						xc7z020-clg400-1 .		
Speed	320 MHz .....		200 MHz .....		100 MHz .....		100 MHz .....		
Period	3.125 ns .....		5 ns .....		10 ns .....		10 ns .....		
Vivado	2019.2	2019.2	2018.2	2018.2	2018.2	2018.2	2019.1	2019.2	
FPGA cost	actual	/ estim.	= $R$	actual	/ estim.	= $R$	actual	/ estim.	= $R$
Latency	3	/ 3	= 1	2	/ 2	= 1	1	/ 1	= 1
Interval	1	/ 1	= 1	1	/ 1	= 1	1	/ 1	= 1
LUT	717	/ 1903	= 0.4	717	/ 4015	= 0.2	717	/ 4007	= 0.2
FF	147	/ 138	= 1.1	147	/ 113	= 1.3	147	/ 2	= 73.
BRAM	5.5	/ 8	= 0.7	5.5	/ 15	= 0.4	5.5	/ 15	= 0.4
URAM	0	/ 0	= 1	0	/ 0	= 1	0	/ 0	= 1
DSP	2	/ 0	= NA	2	/ 2	= 1	2	/ 2	= 1

- Not always 1



<http://d-scholarship.pitt.edu/45784/>

Screenshots in the document

## Autoencoder Firmware Testbench Tutorial

Please download Vivado 2019.2 at the following link, if you do not currently have it:

<https://www.xilinx.com/support/download/index.html/content/xilinx/en/downloadNav/vivado-design-tools/archive.html>

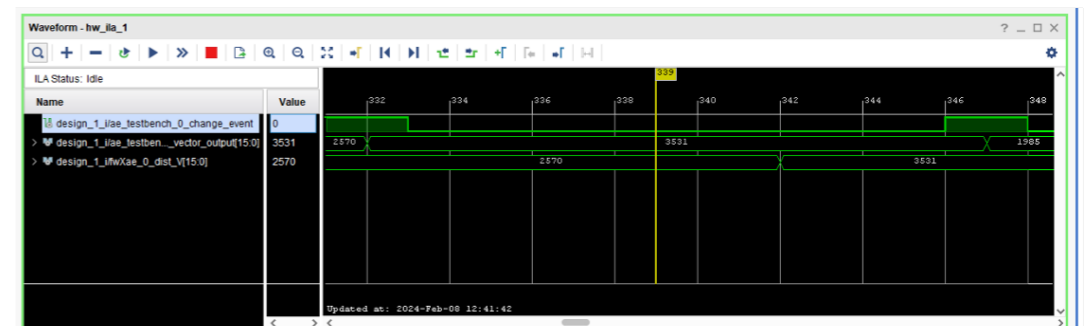
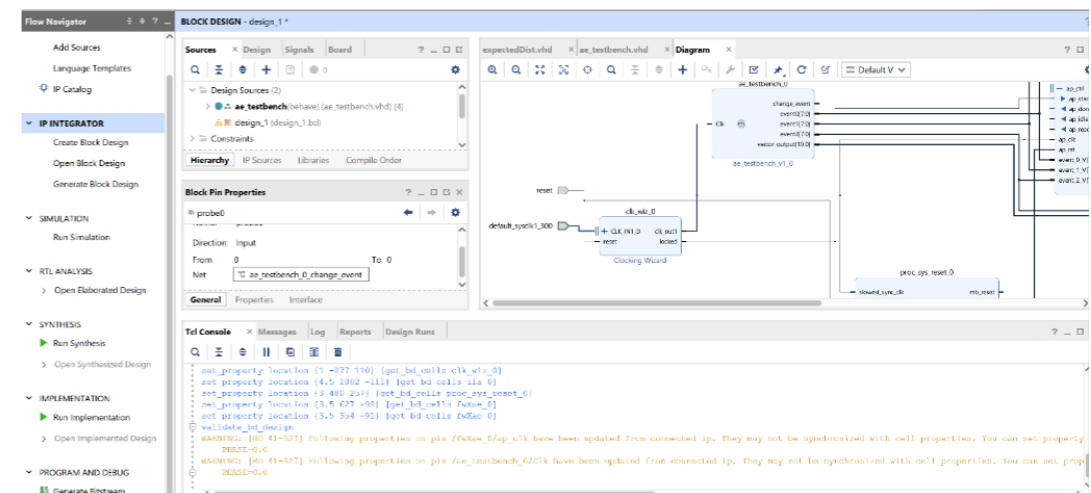
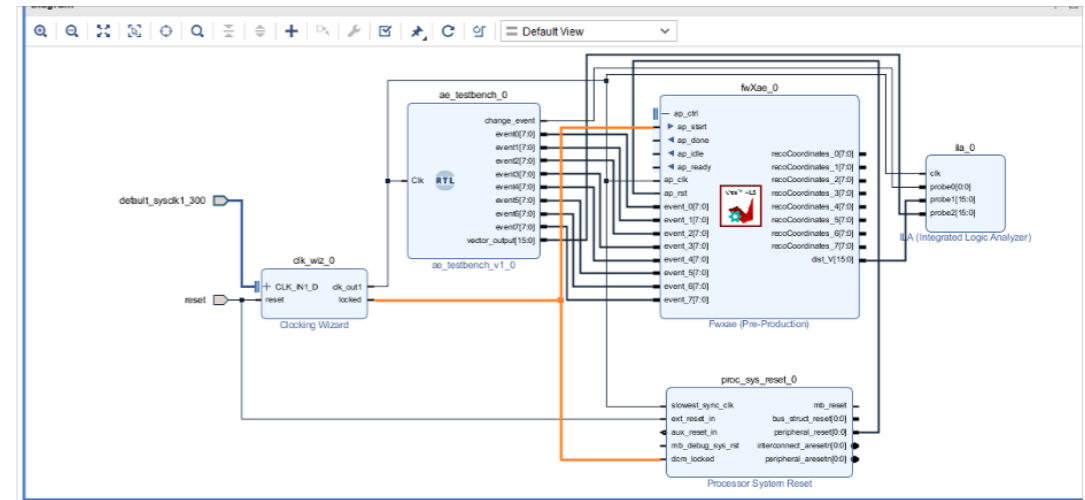
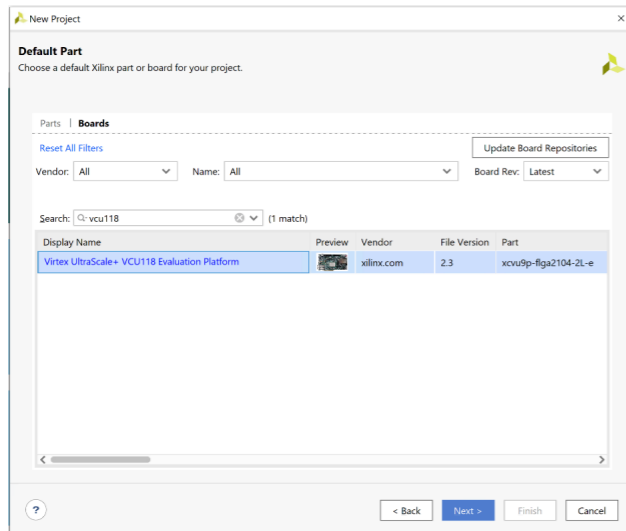
### Before Beginning

Before beginning, please make sure that you have (and know the location of) the autoencoder IP folder, and the VHDL testbench files:

Name	Date Modified	Type	Size
autoencoder8var_ip	2/7/2024 1:30 PM	File folder	
tb_vhd_files	2/8/2024 11:50 AM	File folder	

### Creating New Project in Vivado

Open Vivado 2019.2 and select "create new Project." On the following pop-up, select "next," and you will be prompted to name the project. Name the project as you wish and choose a location to store it. Keep clicking next until you reach a page that prompts you to select the part/ board. For this tutorial, we will be using the Virtex UltraScale+ VCU118 board. After you have selected your part or board, keeping clicking "next" until you have reached the end of the setup page.

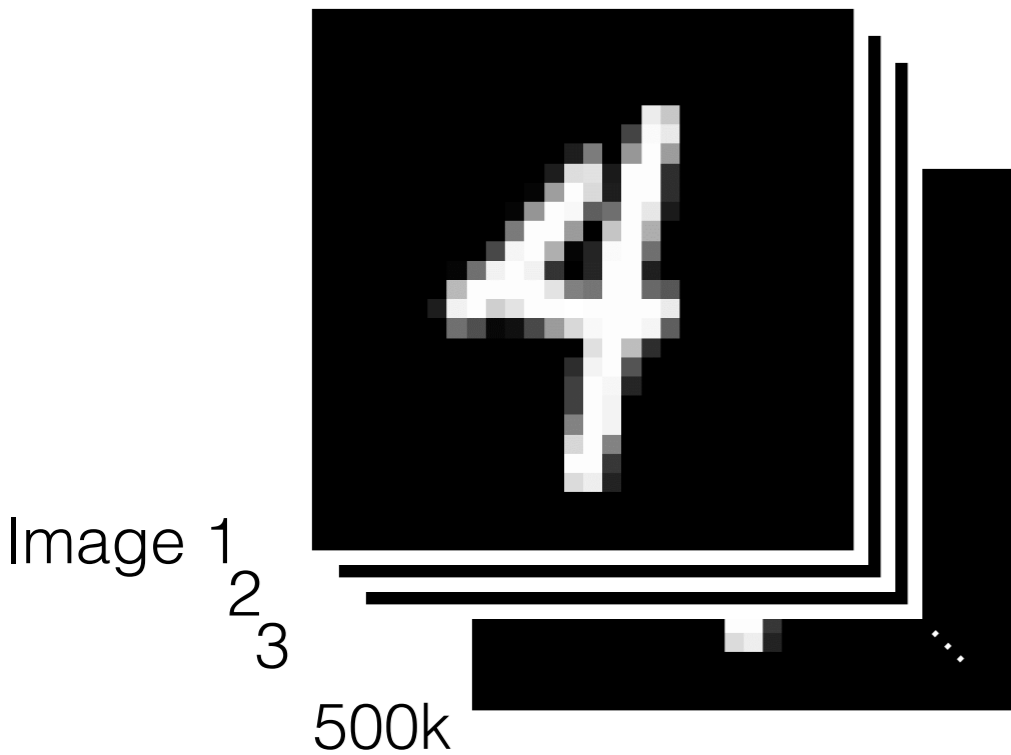




## Example: handwritten numbers

- Teach it about the number 4

784 variables (8-bit)



Corresponding data set

=

Image	Pixel 1	Pixel 2	...	Pixel 300	...	Pixel 783	Pixel 784
1	0	0	...	240	...	0	0
2	0	1	...	255	...	0	0
...	...	...	...	...	...	...	...
500k	0	0	...	231	...	0	0

## Details

- Each pixel in the data set are unrelated to each other

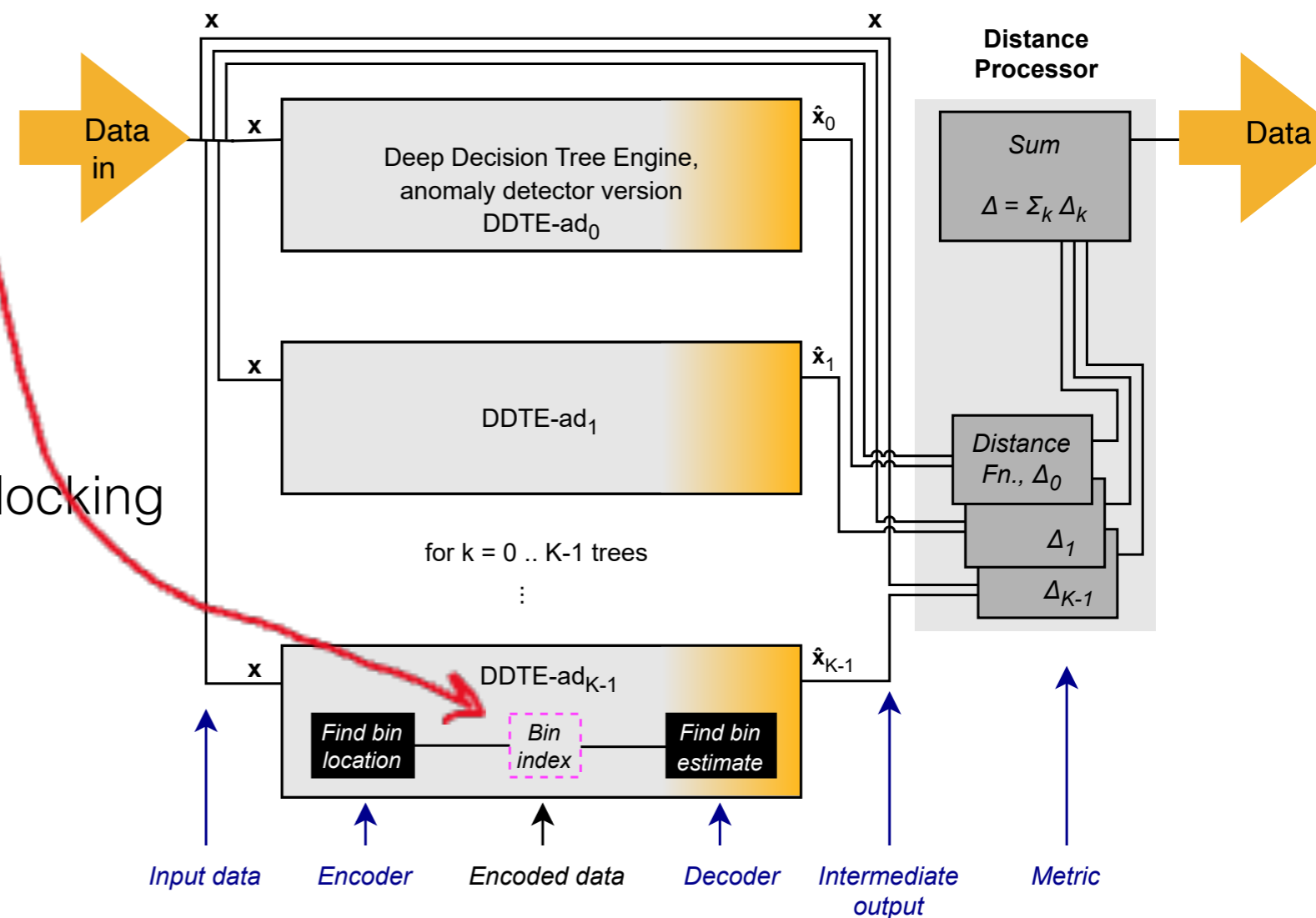


## Logic flow

- Left-to-right data flow (see right)
- Realized that we can bypass the latent space!
  - Encoding = Decoding

## Details

- Parallel computing
  - TREE ENGINES eval. in parallel
  - All combinatoric logic, so no clocking between steps = fast
  - Mostly comparisons = fast
  - No multiplication = fast
- Technical info in backup & [\[2304.03836\]](#)



Shown conceptually as actual encode-decode occur simultaneously.



# Design v2: Parallelize terminal bins

Go deeper from 4  $\rightarrow$  8

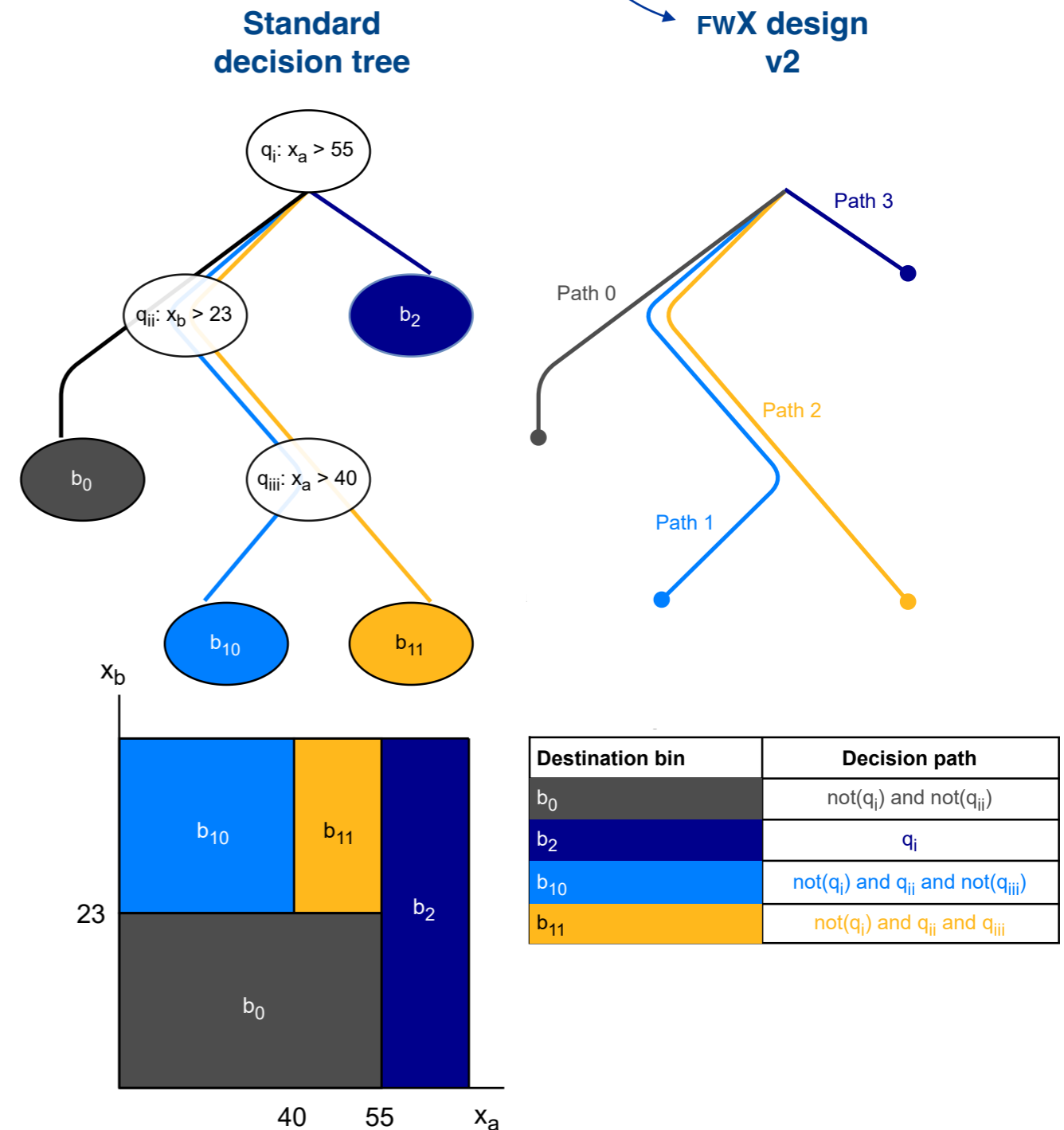
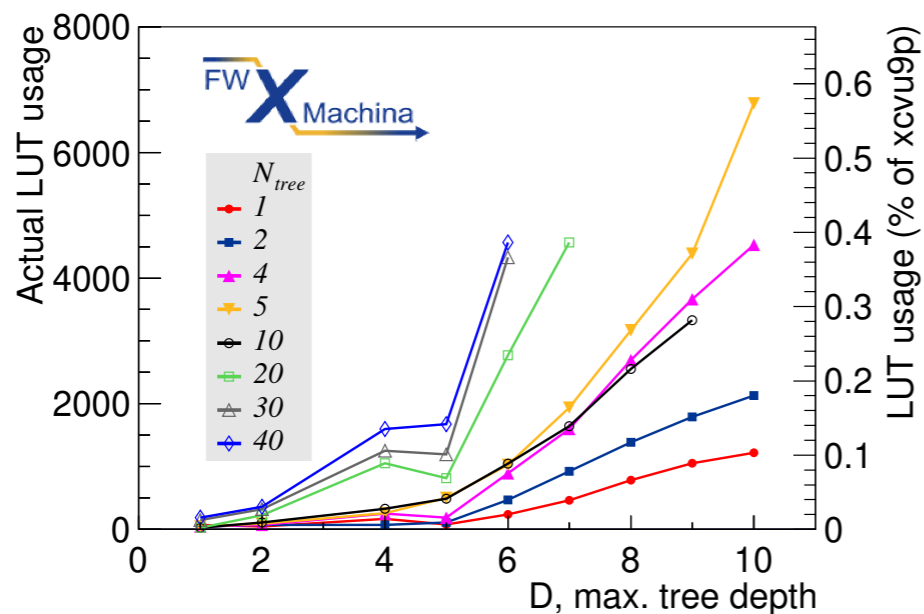
- Improve FWXv1

**Challenge** Does not scale well w/  
tree depth & # variables  
*Cut redundancy  $2^D$*

- FWXv2

**Key design** Evaluate decision paths

**Benefit** Softer scaling vs  $2^D$



# Machine learning

Focus on the most popular use cases in HEP

## Supervised classification

- Neural networks & Boosted decision trees
- Others (SVM, kNN, Matrix element, etc.)

Not covered

## Structural similarities: NN & BDT

- Step function boundary
- Fuzzy boundary

Focus of this section

## Use cases

- Regression
- Classification S vs. B
- Anomaly detection B vs. not-B

Previous slides

If time

Later slides

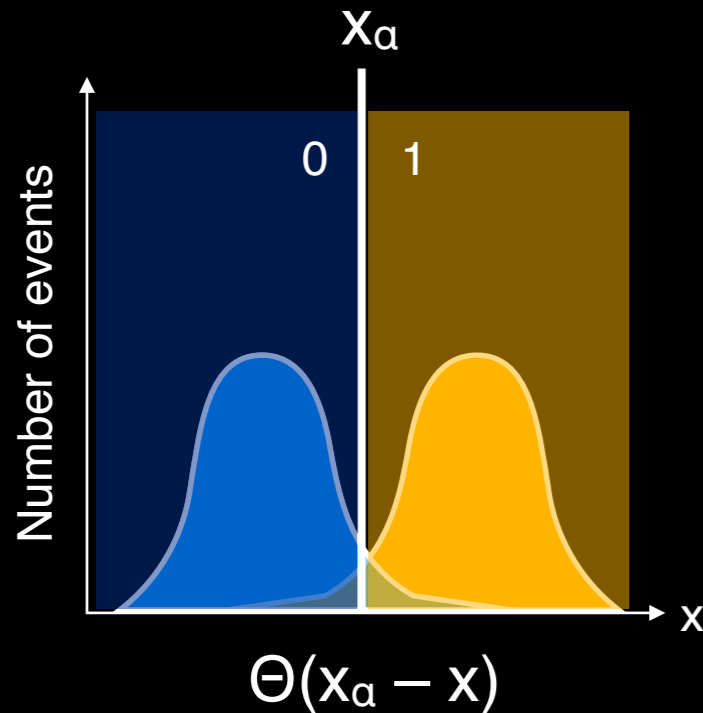


Will discuss other approaches (estimation, unsupervised) after intro

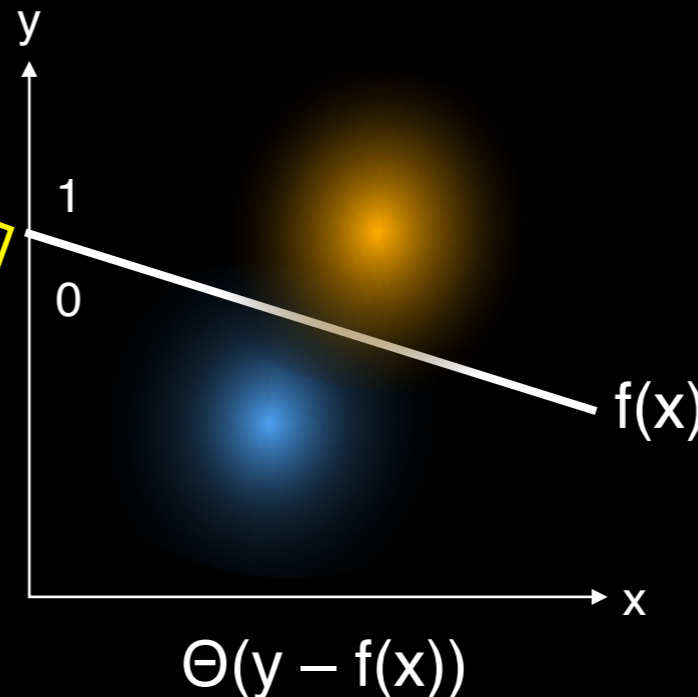
# Neural networks basics

From Bruce Denby, *Tutorial on Neural Network Applications in High Energy Physics: A 1992 Perspective*, FERMILAB-CONF-92 / 121-E

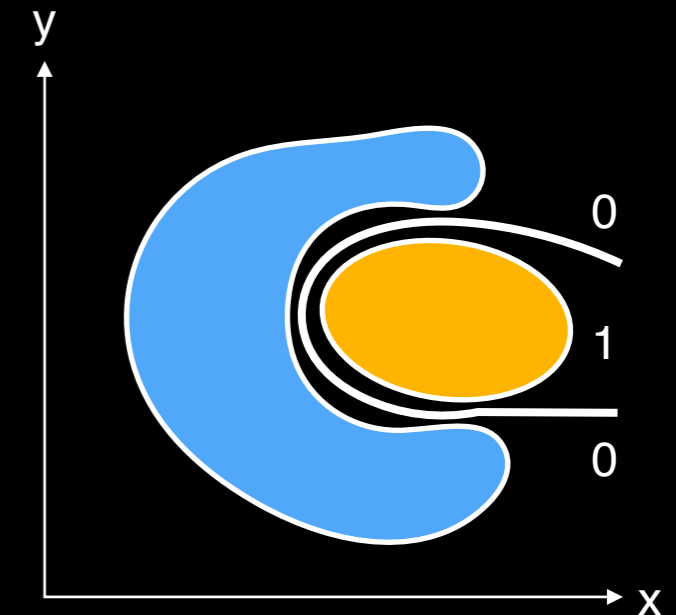
Step function for 1d



Step function for 2d



Curved step fn? for 2d



$$= \Theta(y - (mx + b))$$

$$= \Theta(c_1 y + c_2 x + b)$$

$$= \Theta(c_1 x_1 + c_2 x_2 + b)$$

$$= \Theta(\mathbf{c} \cdot \mathbf{x} + b)$$

substitute

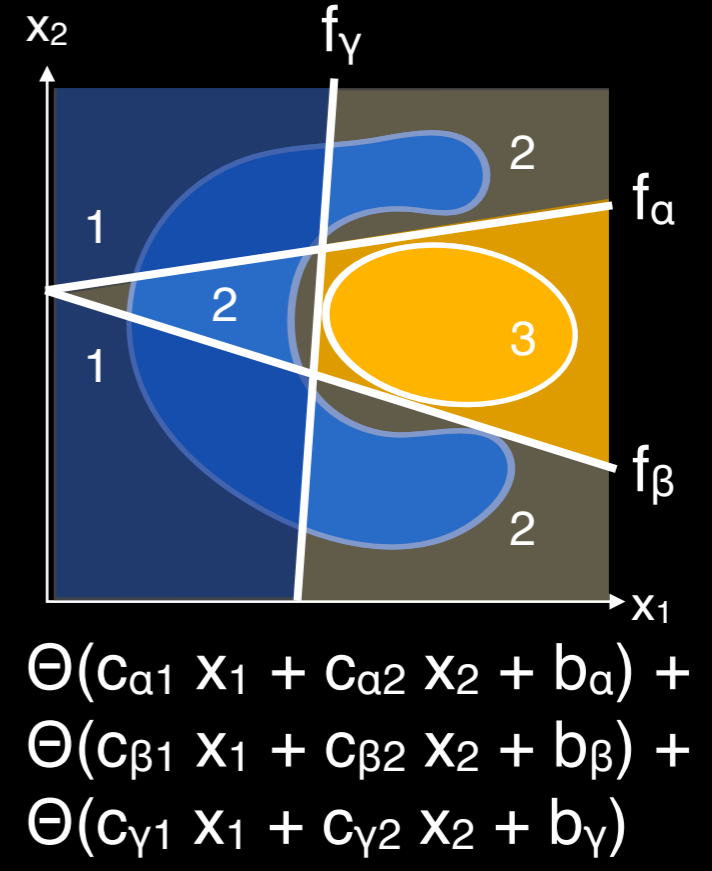
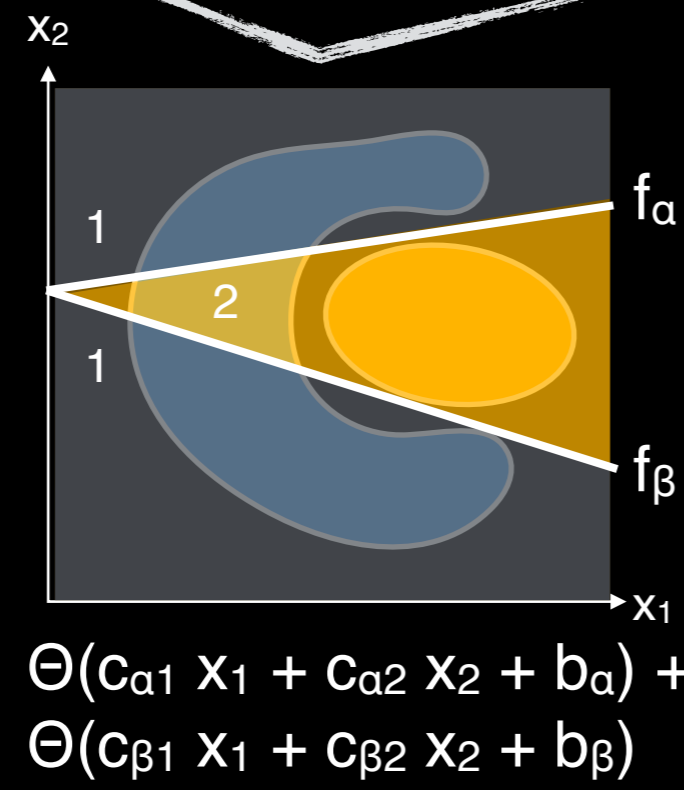
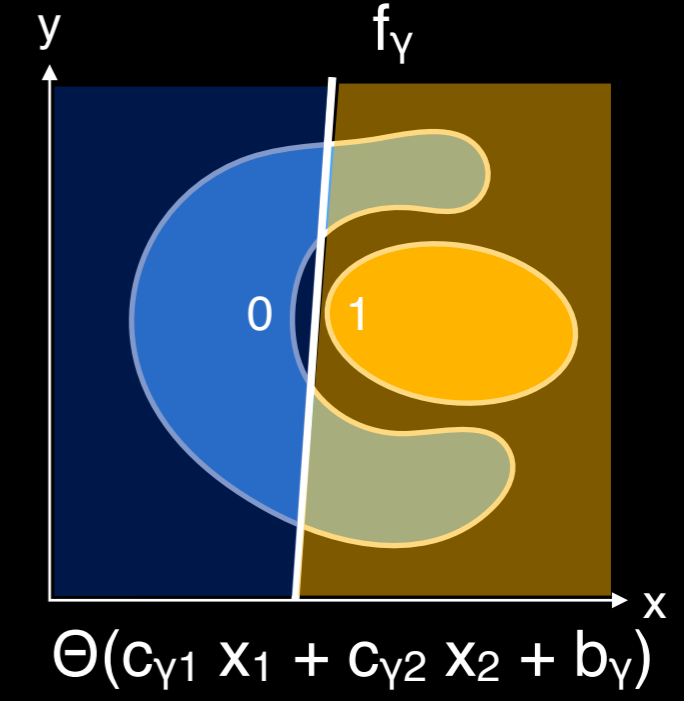
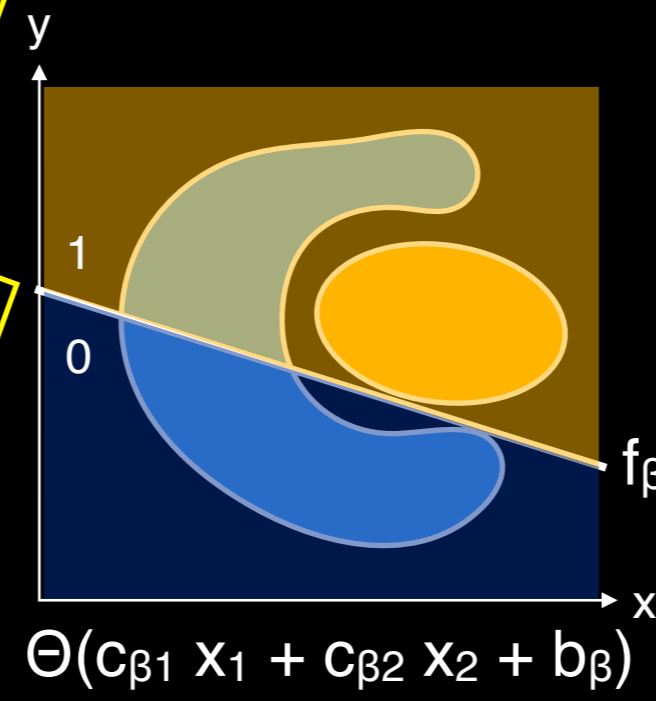
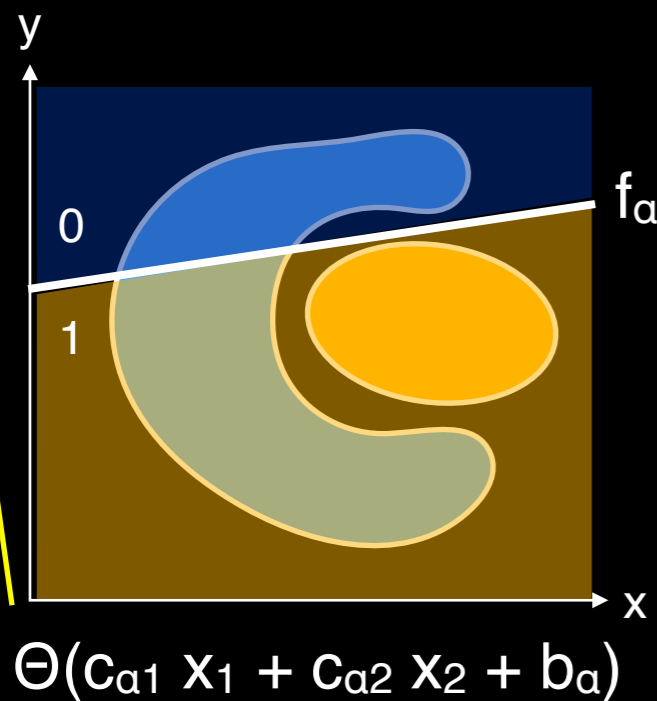
multiply by  $c_1$  & define  $c_2$

generalized notation

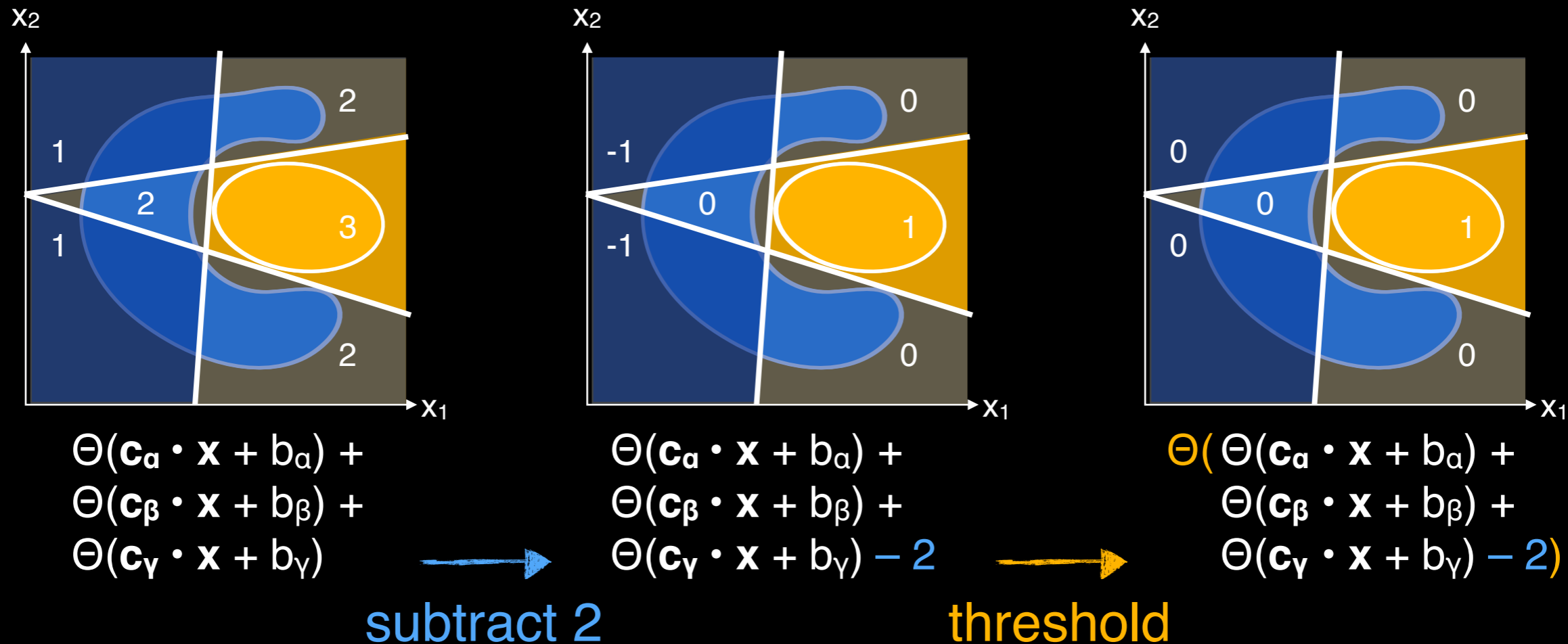
vector notation



Step functions divide samples given a desired true / false positive rates



Sum of step functions can approximate the desired contour



Step function for  
2-dim inputs



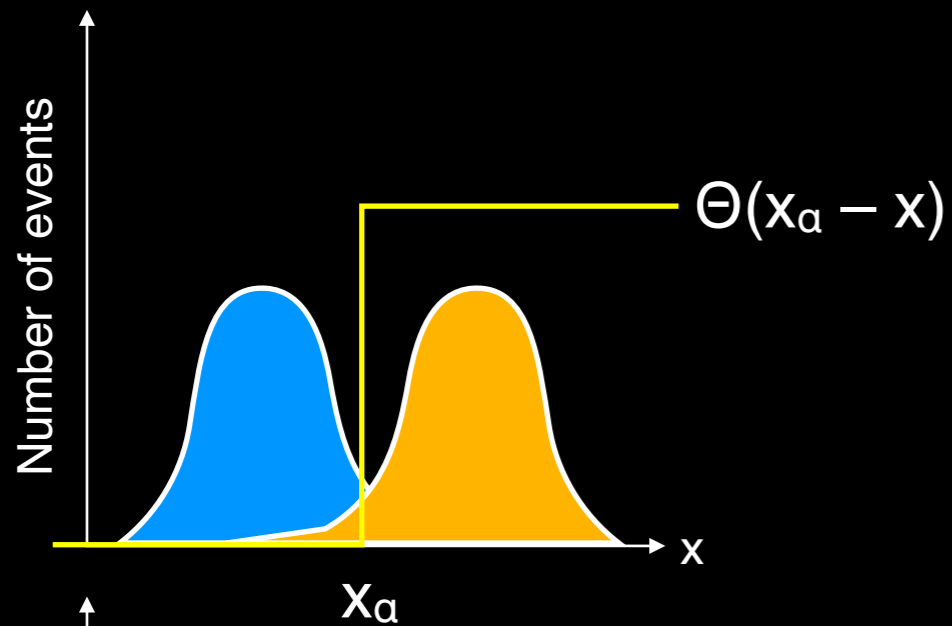
The contour is converted to the final step function



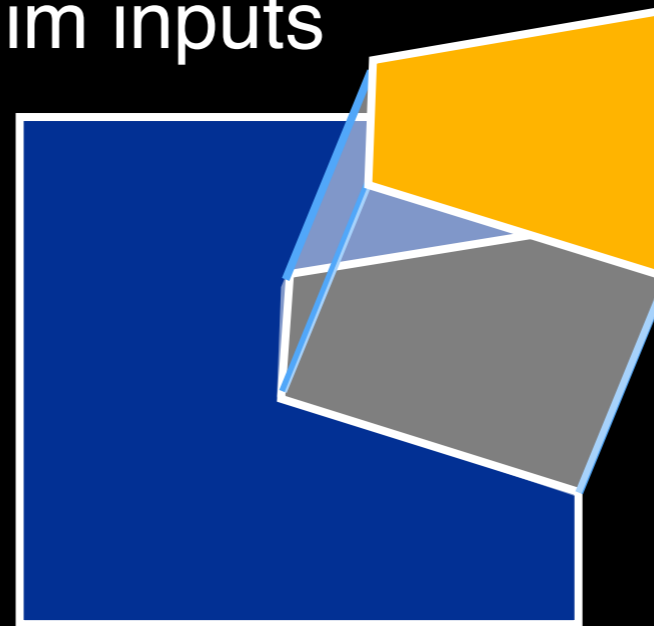
# Activation function

Fuzzy boundary using a function

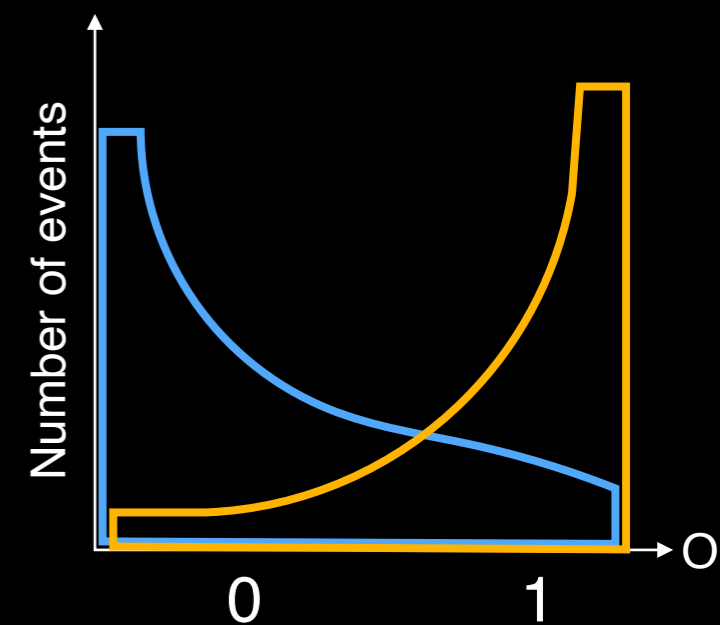
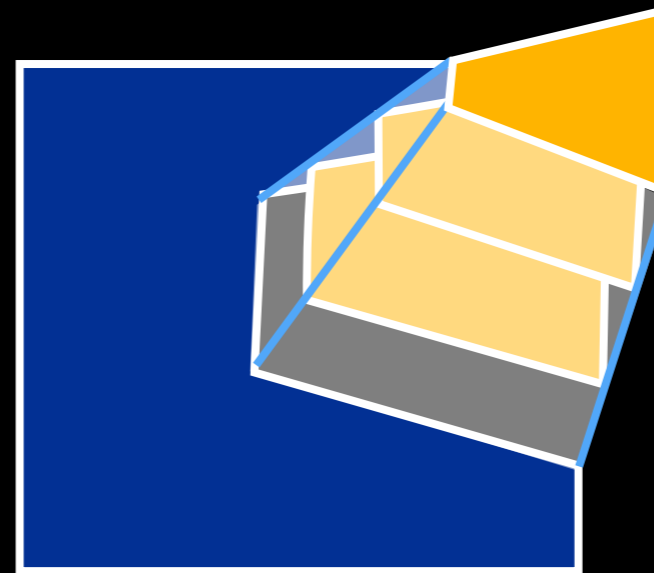
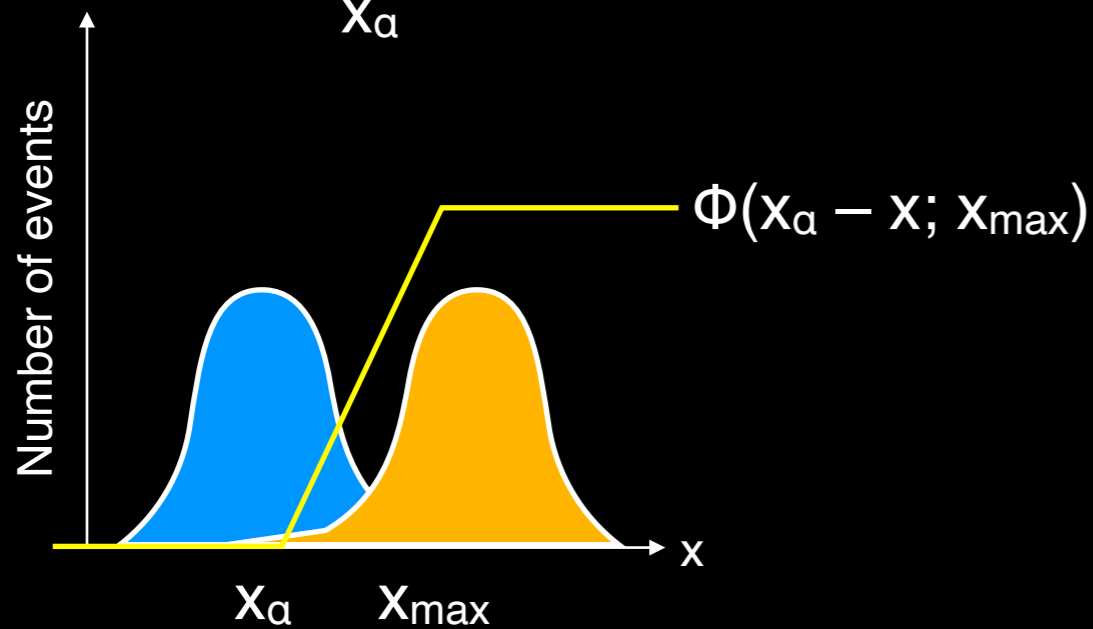
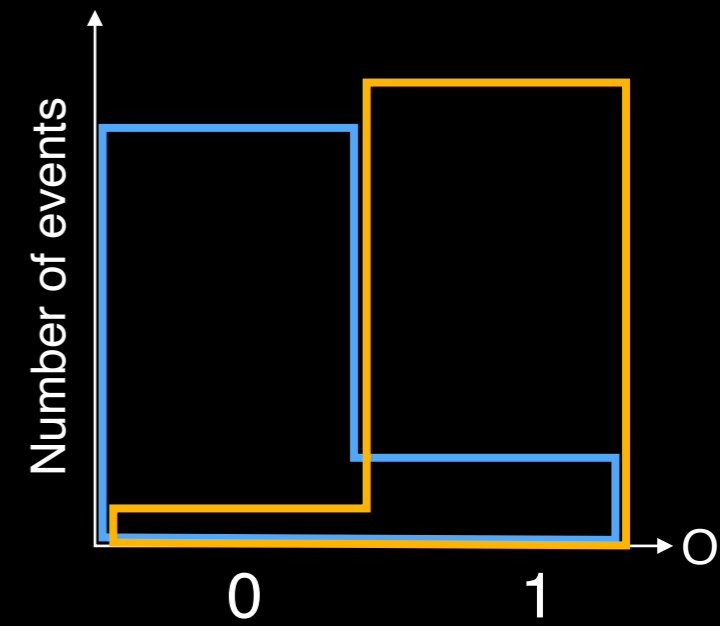
1-dim input



2-dim inputs



Output score

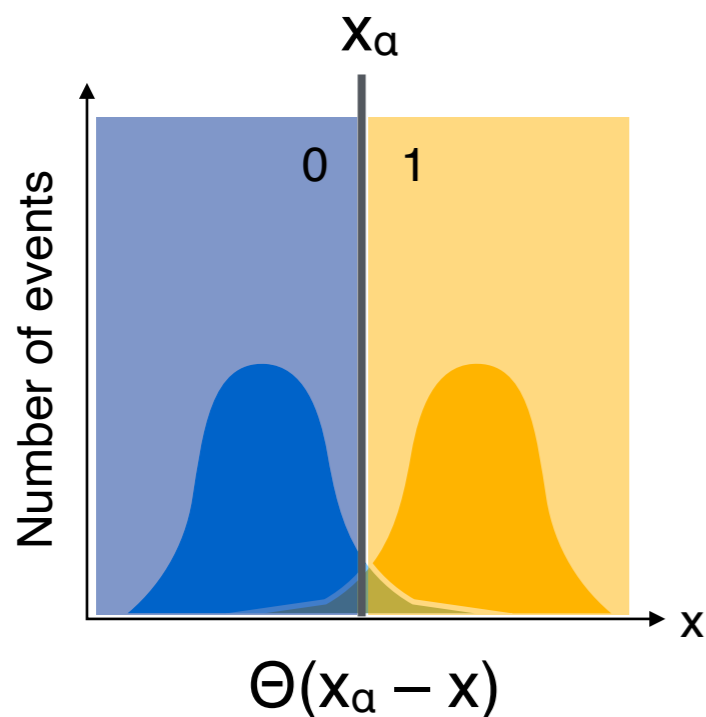


Activation fn gives users a handle to control true / false positive rates

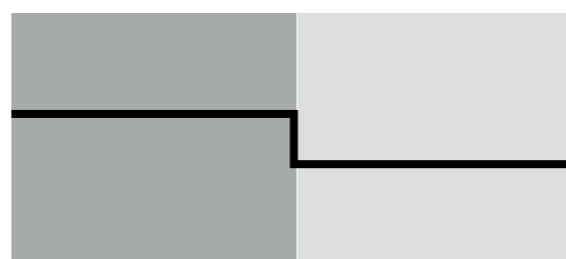
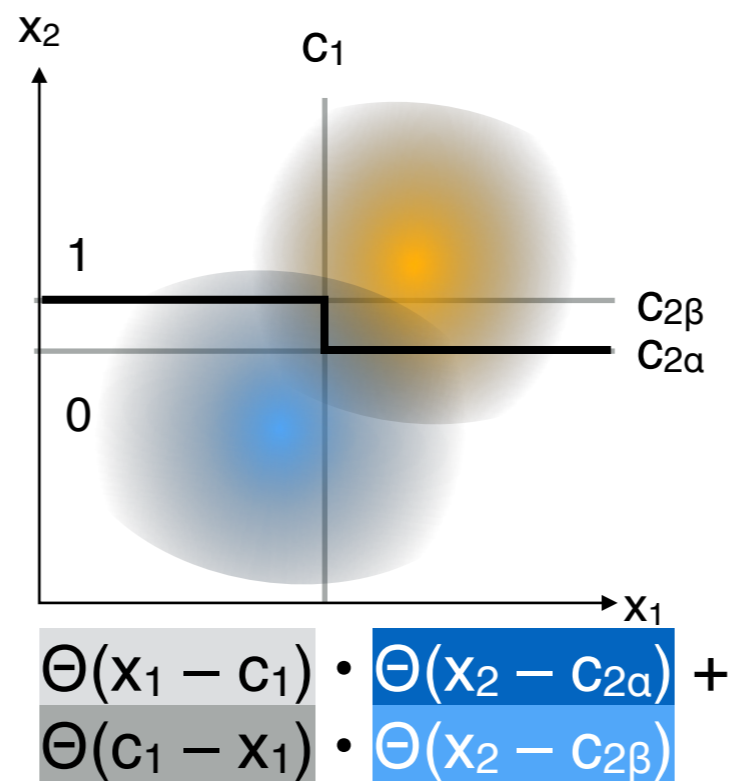
# Decision tree basics

And how it achieves the same result as NN

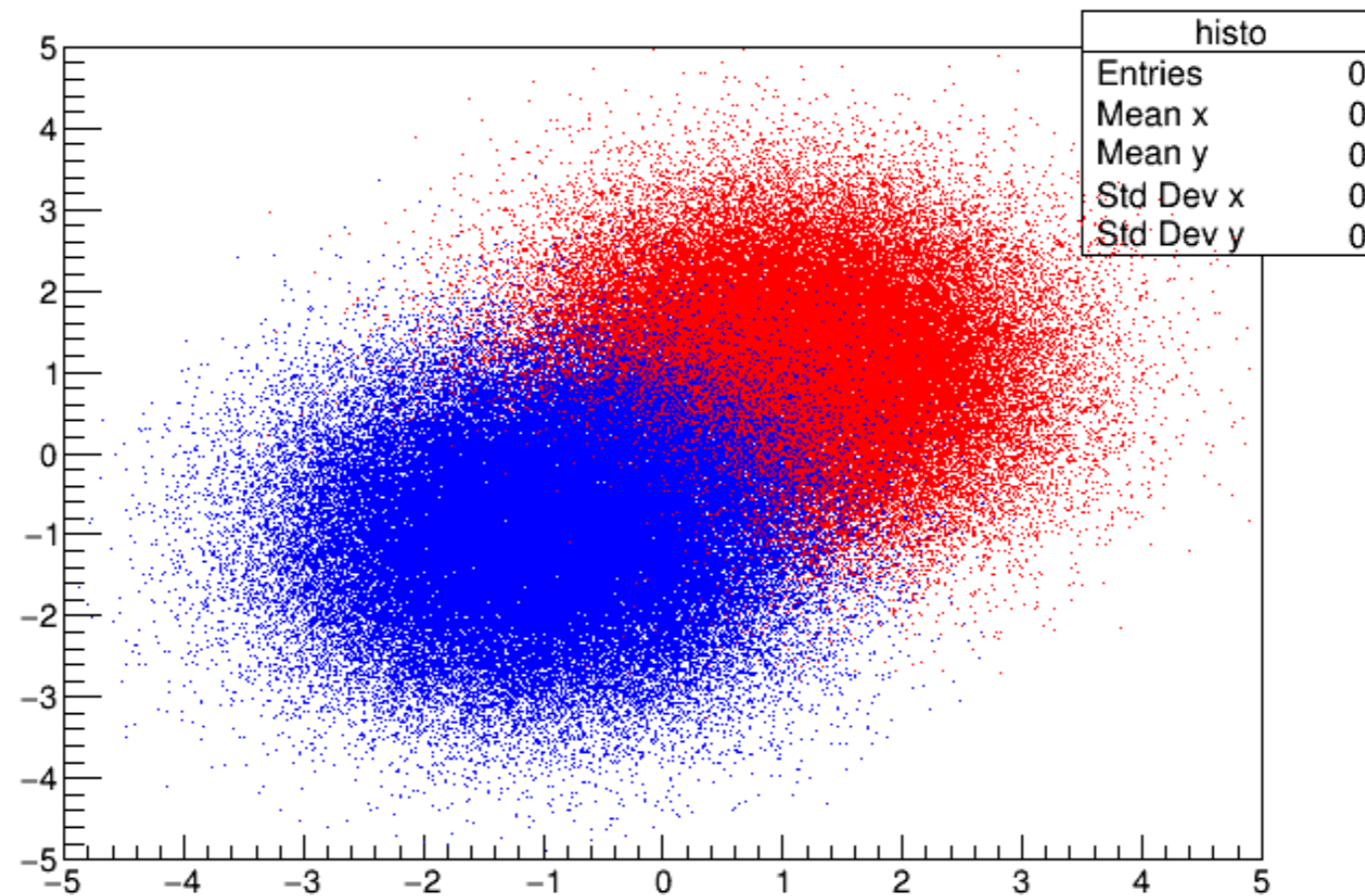
Step function for 1d



Step function for 2d



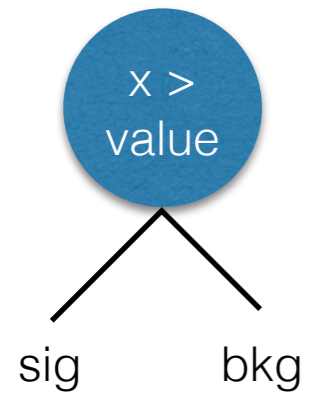
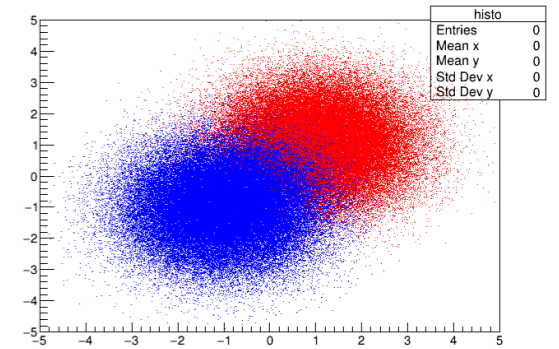
# Flip book



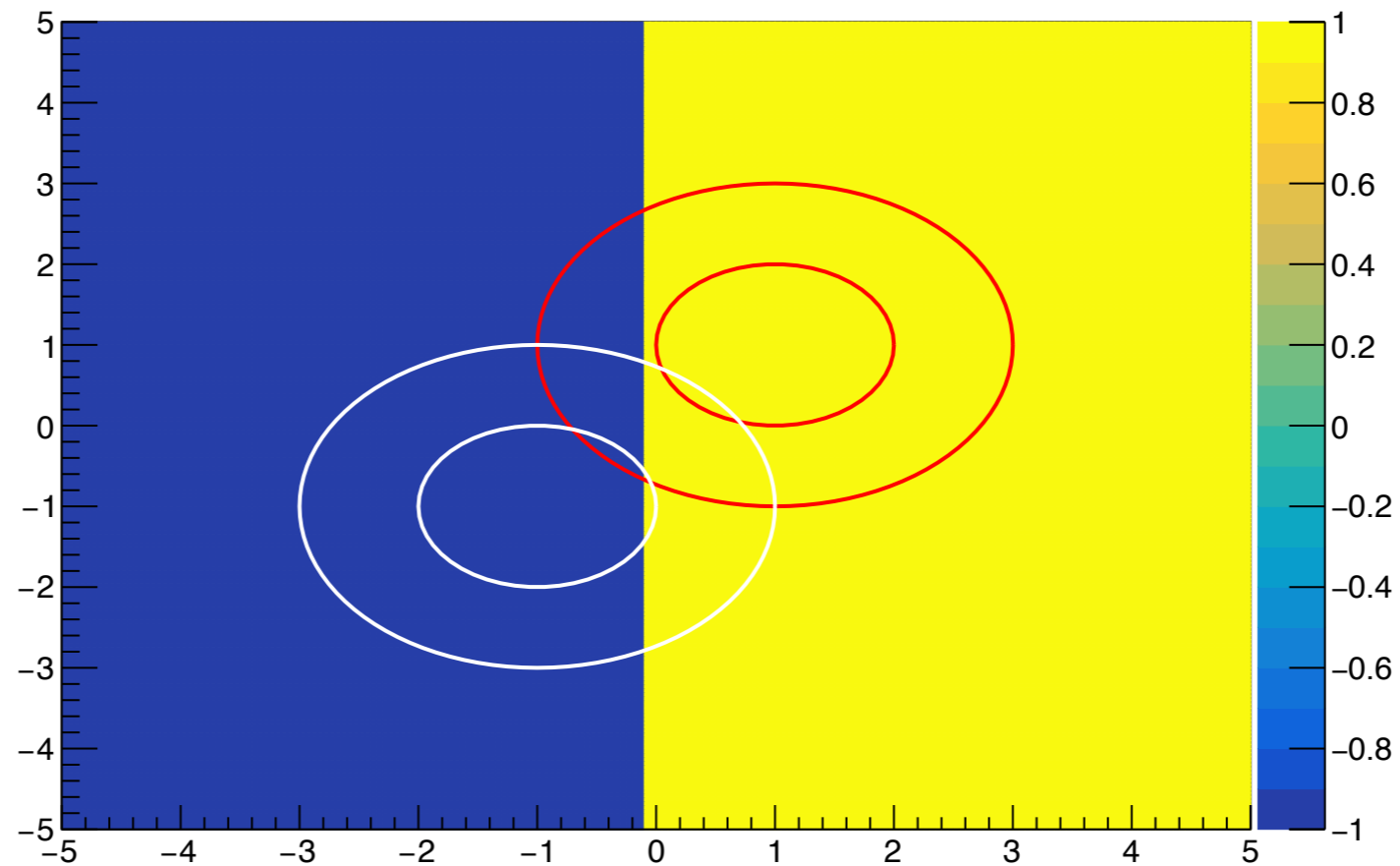
Unit gaussians of two variables



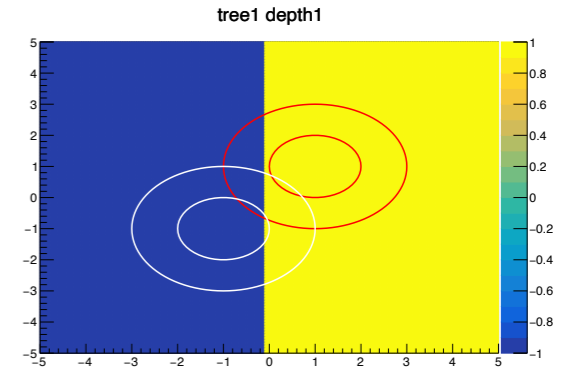
# One decision tree



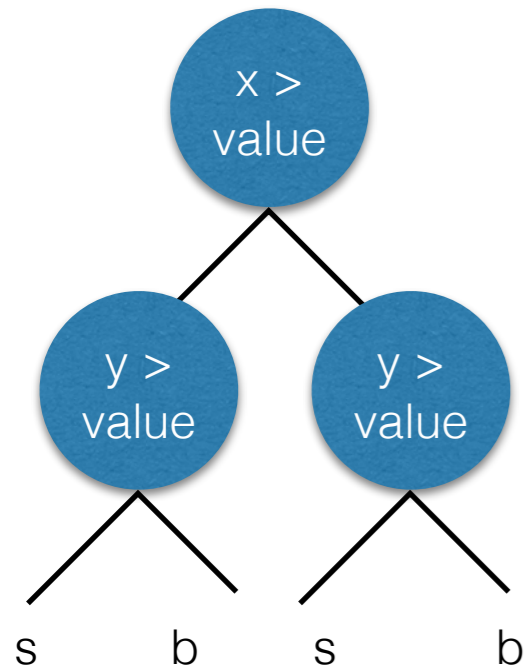
tree1 depth1



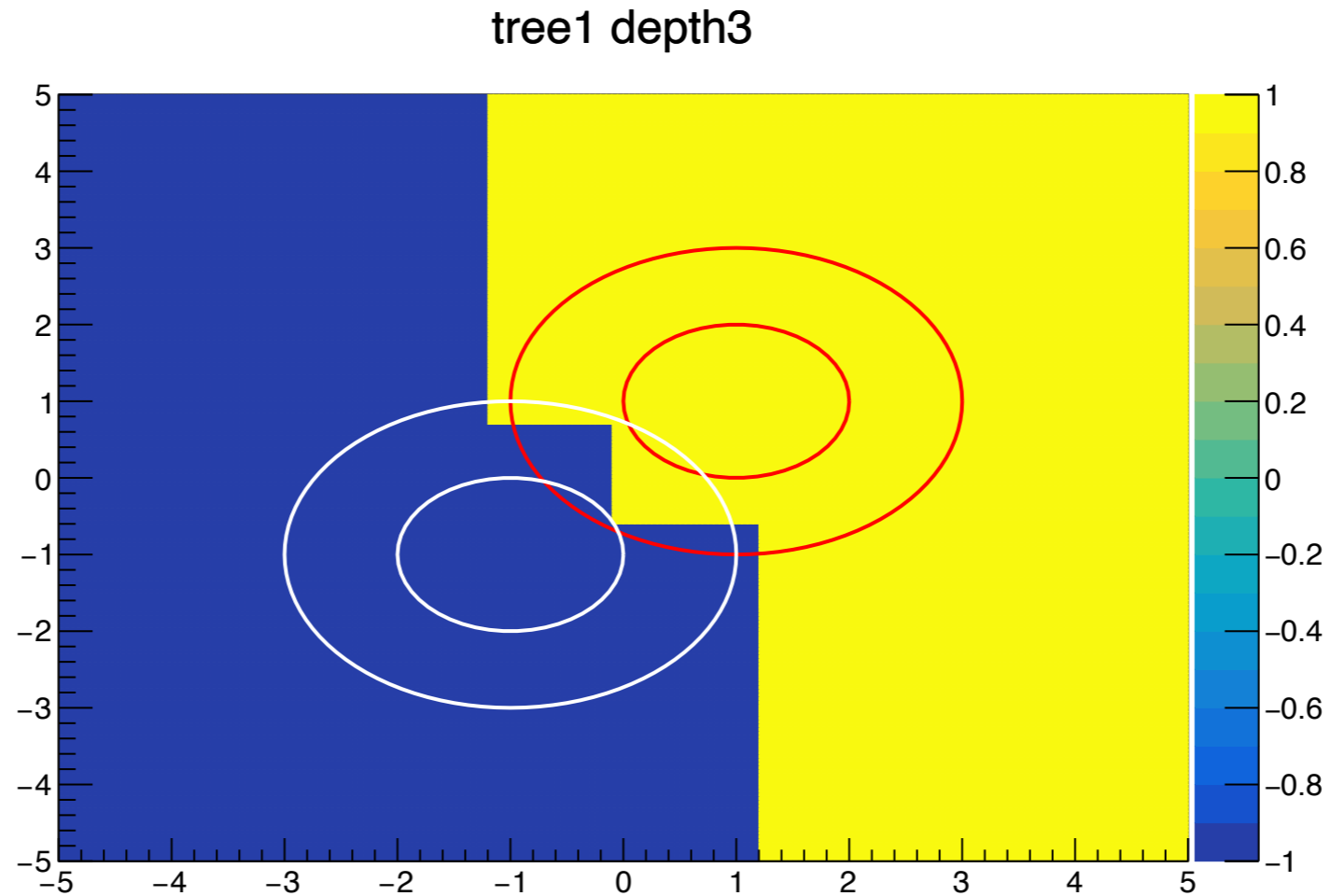
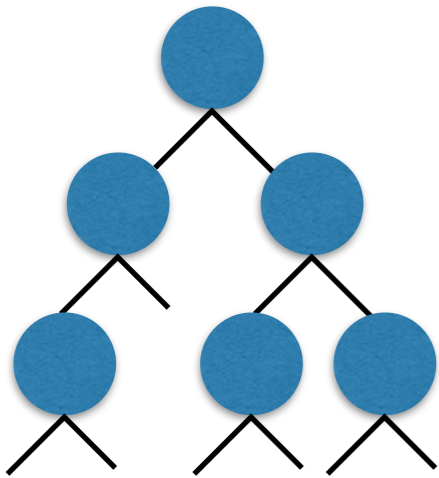
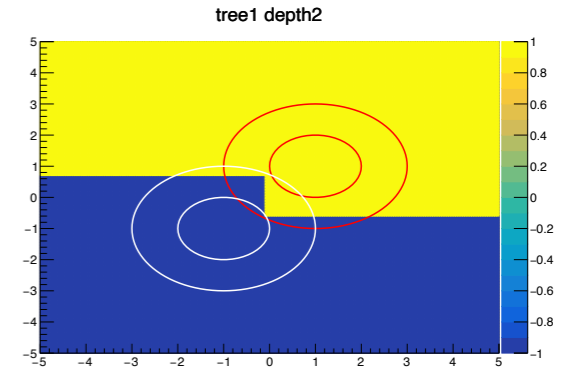
# One decision tree



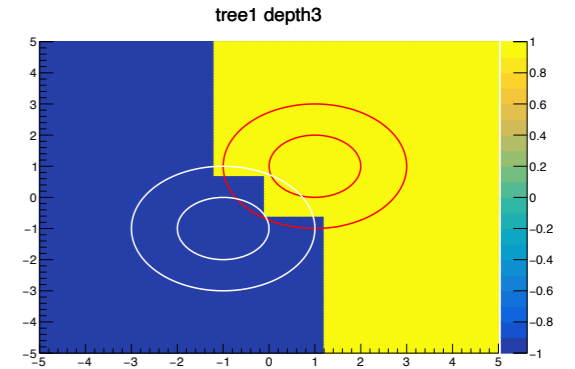
tree1 depth2



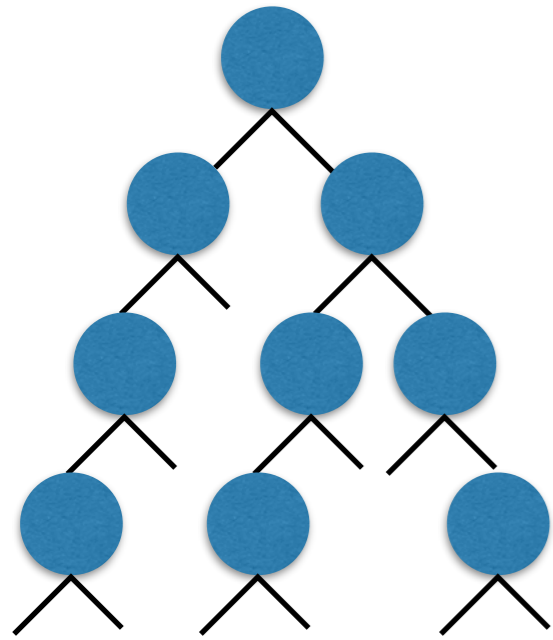
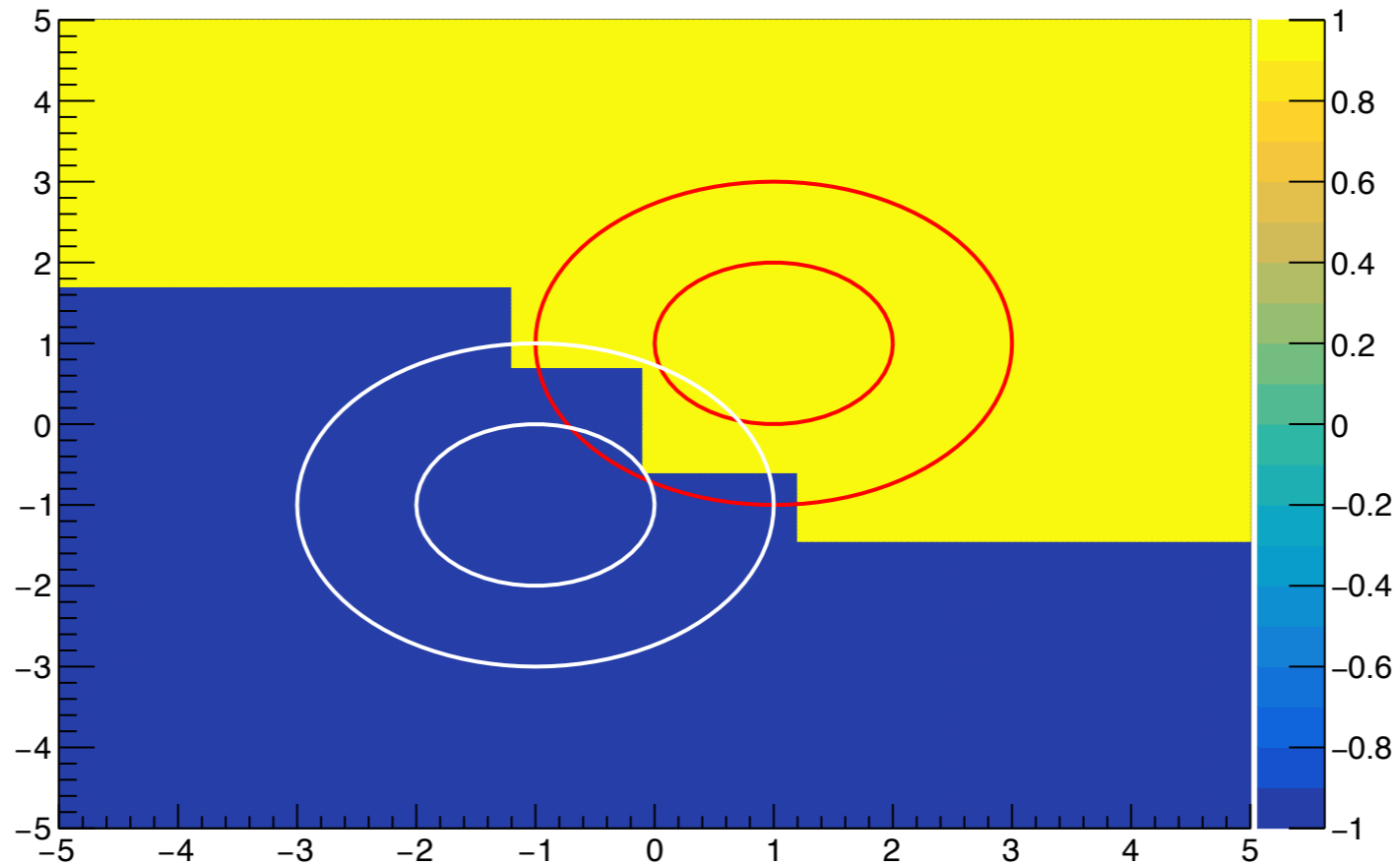
# One decision tree



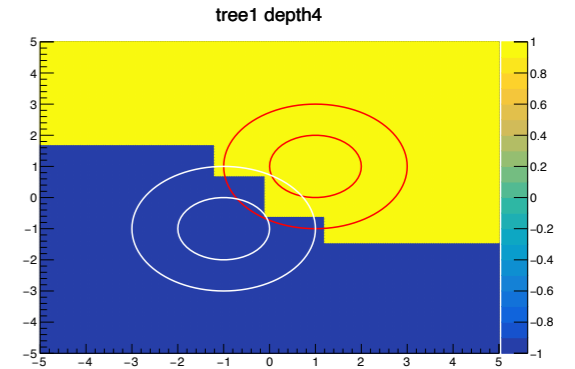
# One decision tree



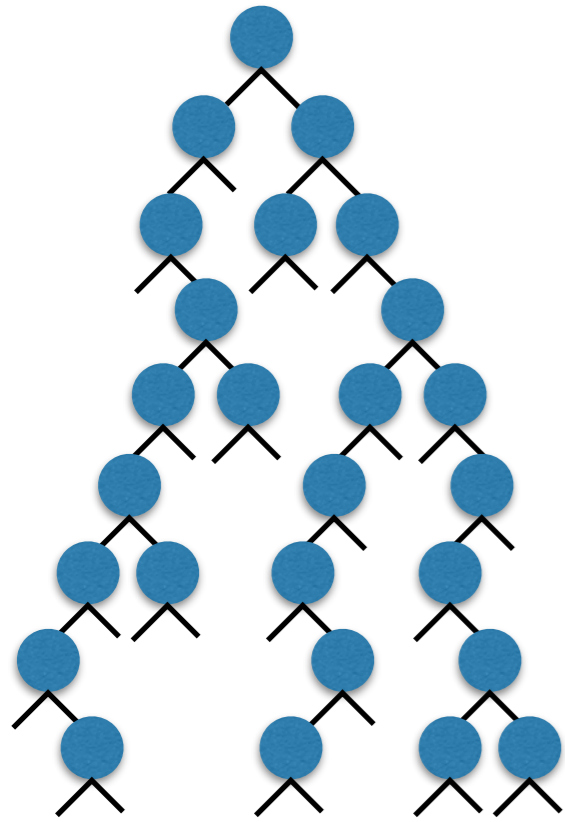
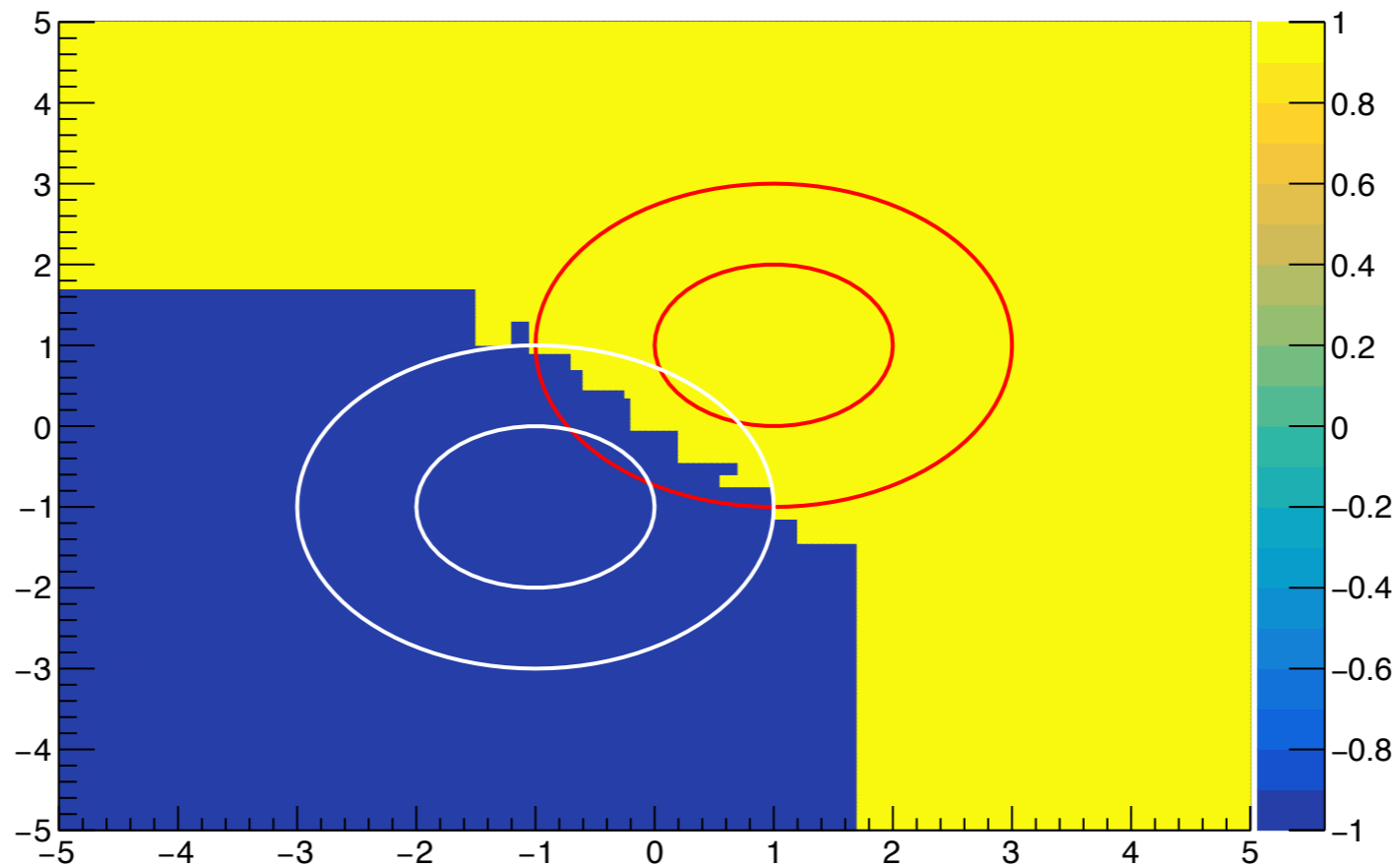
tree1 depth4



# One decision tree



tree1 depth8



Draws diagonal

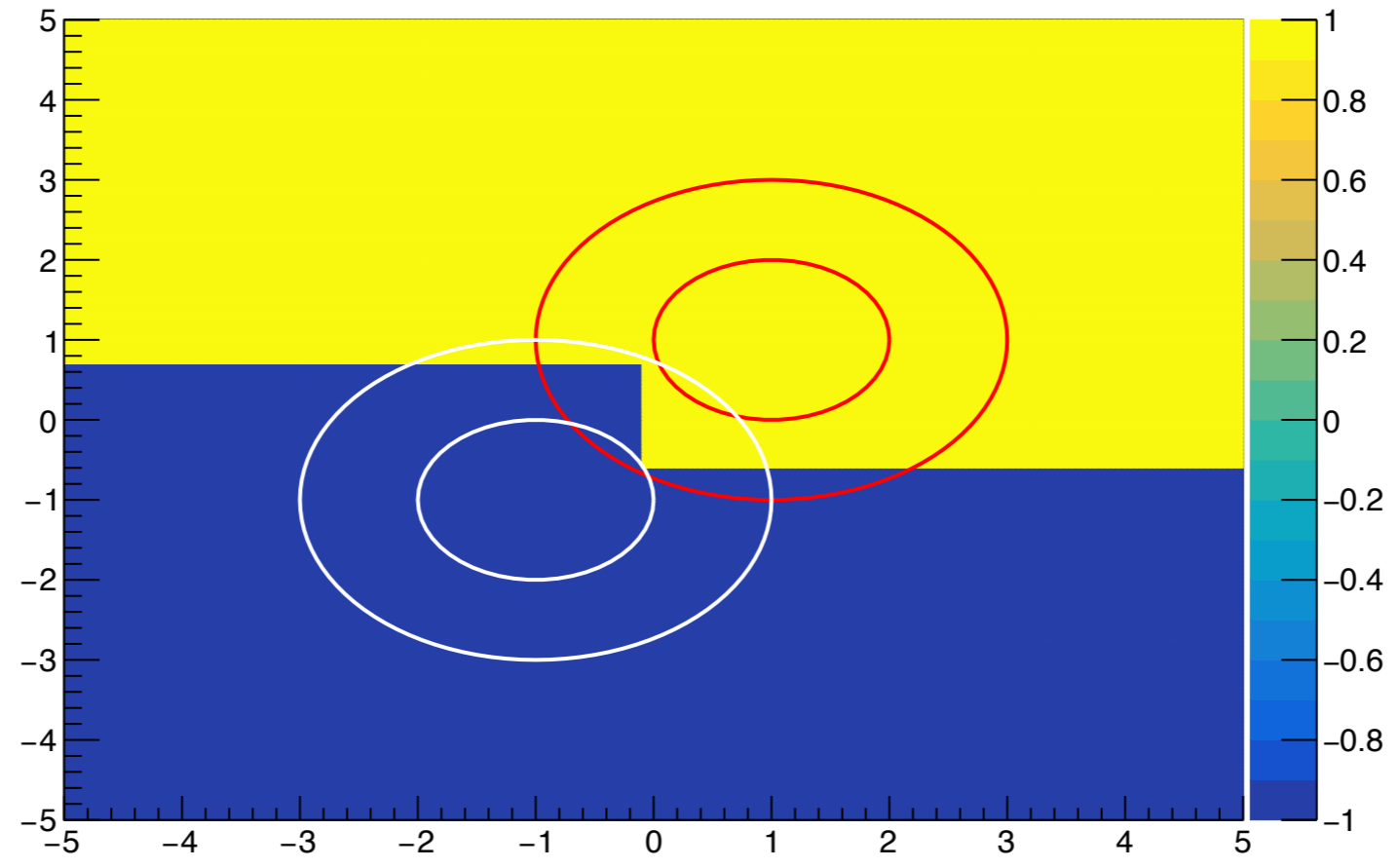
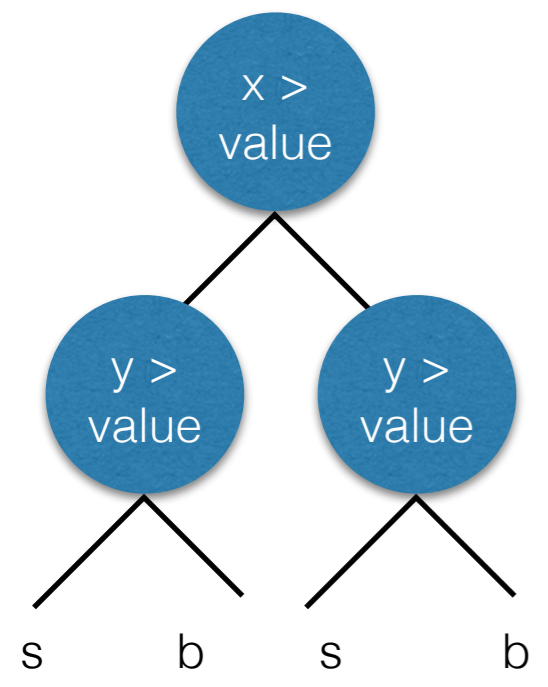
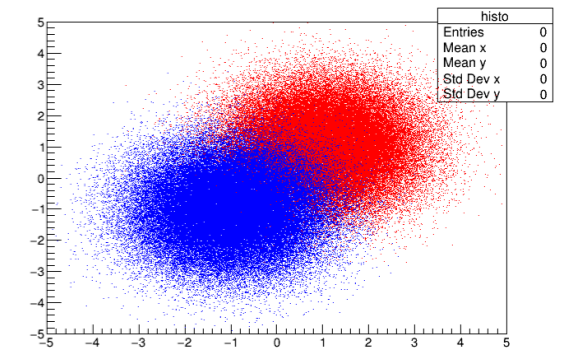
# Depth 2

vary trees



# Depth 2

vary  
↓  
tree1 depth2

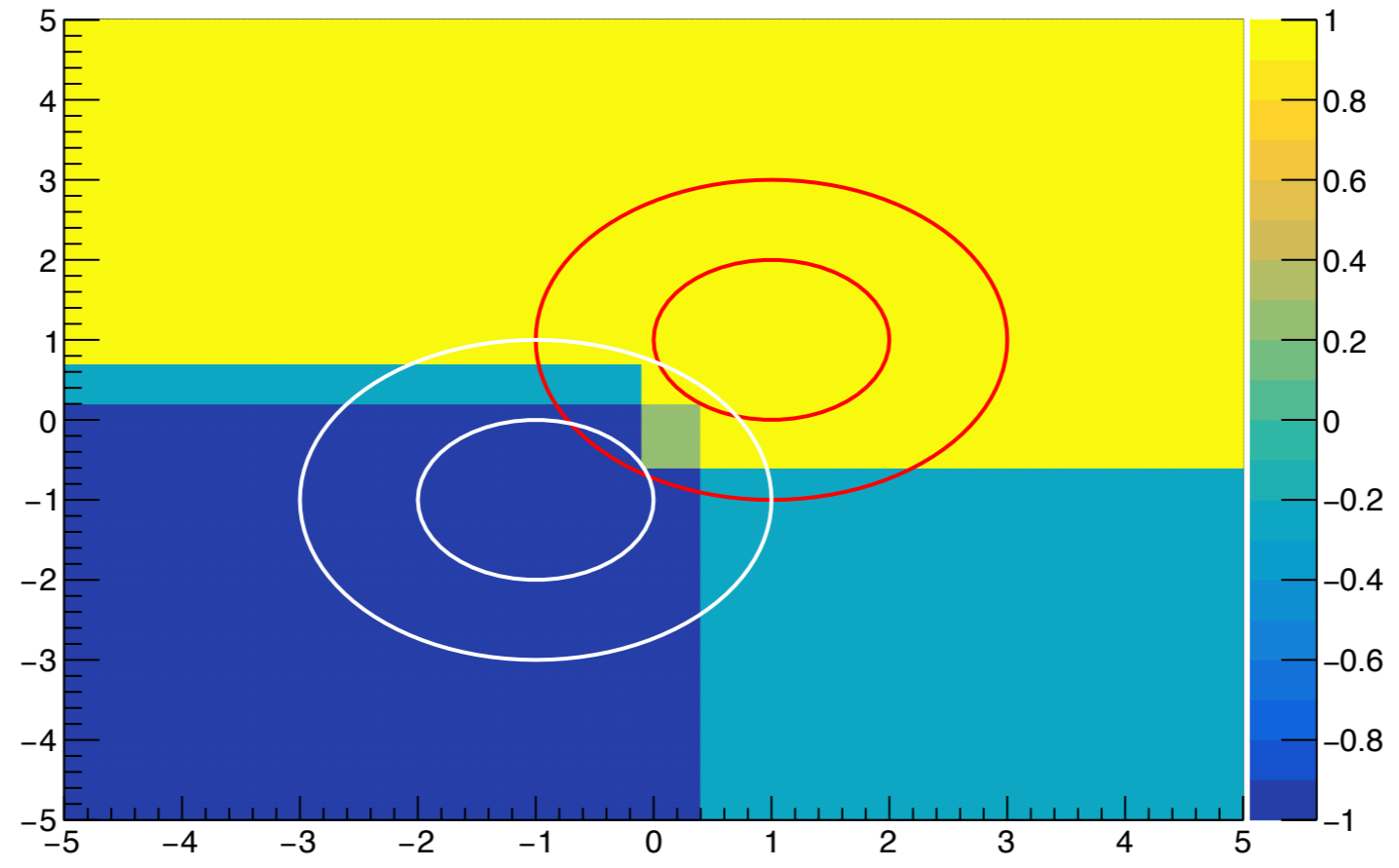
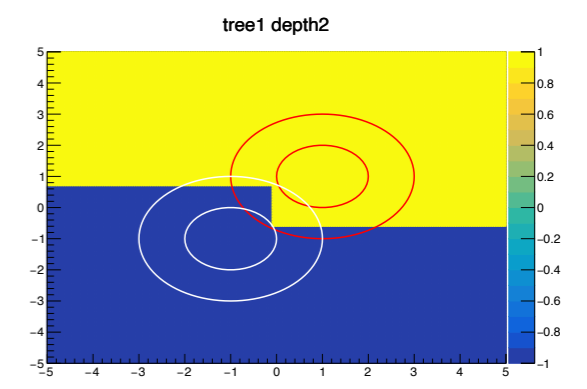
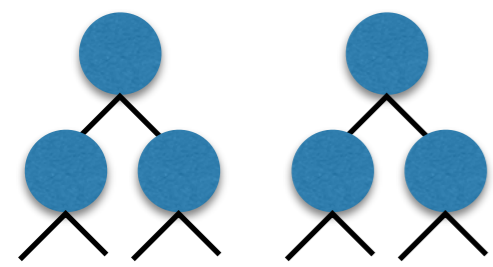


# Depth 2

vary



tree2 depth2



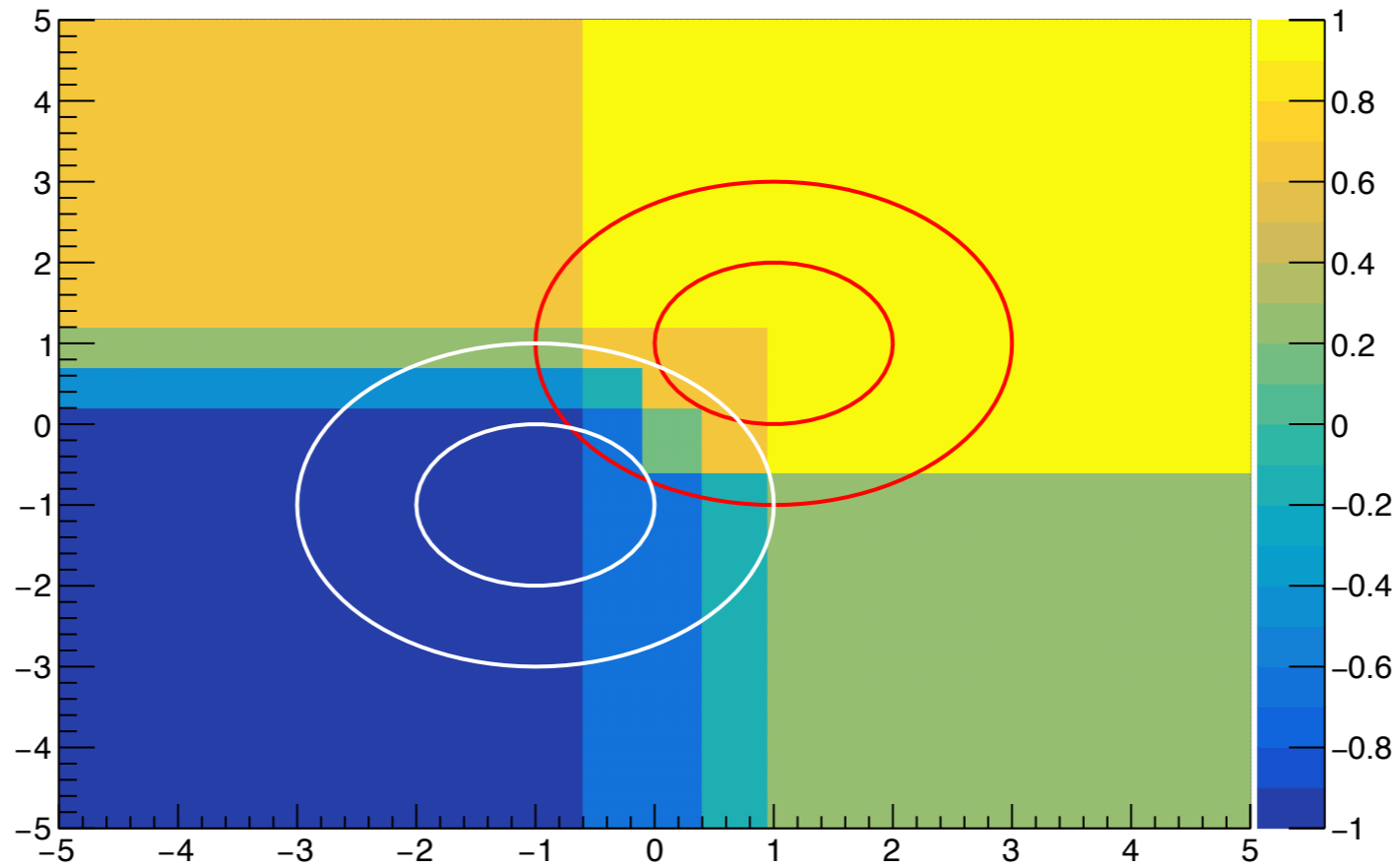
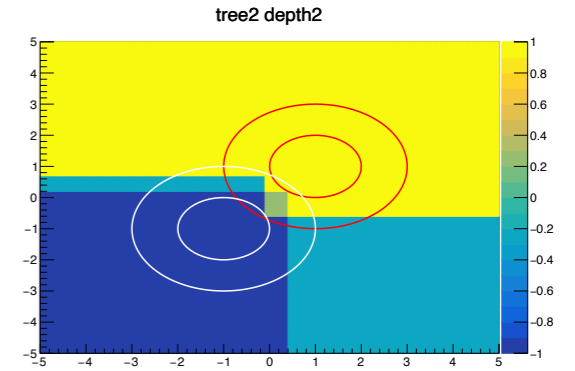
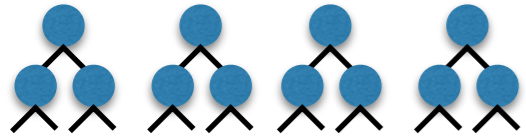


# Depth 2

vary



tree4 depth2

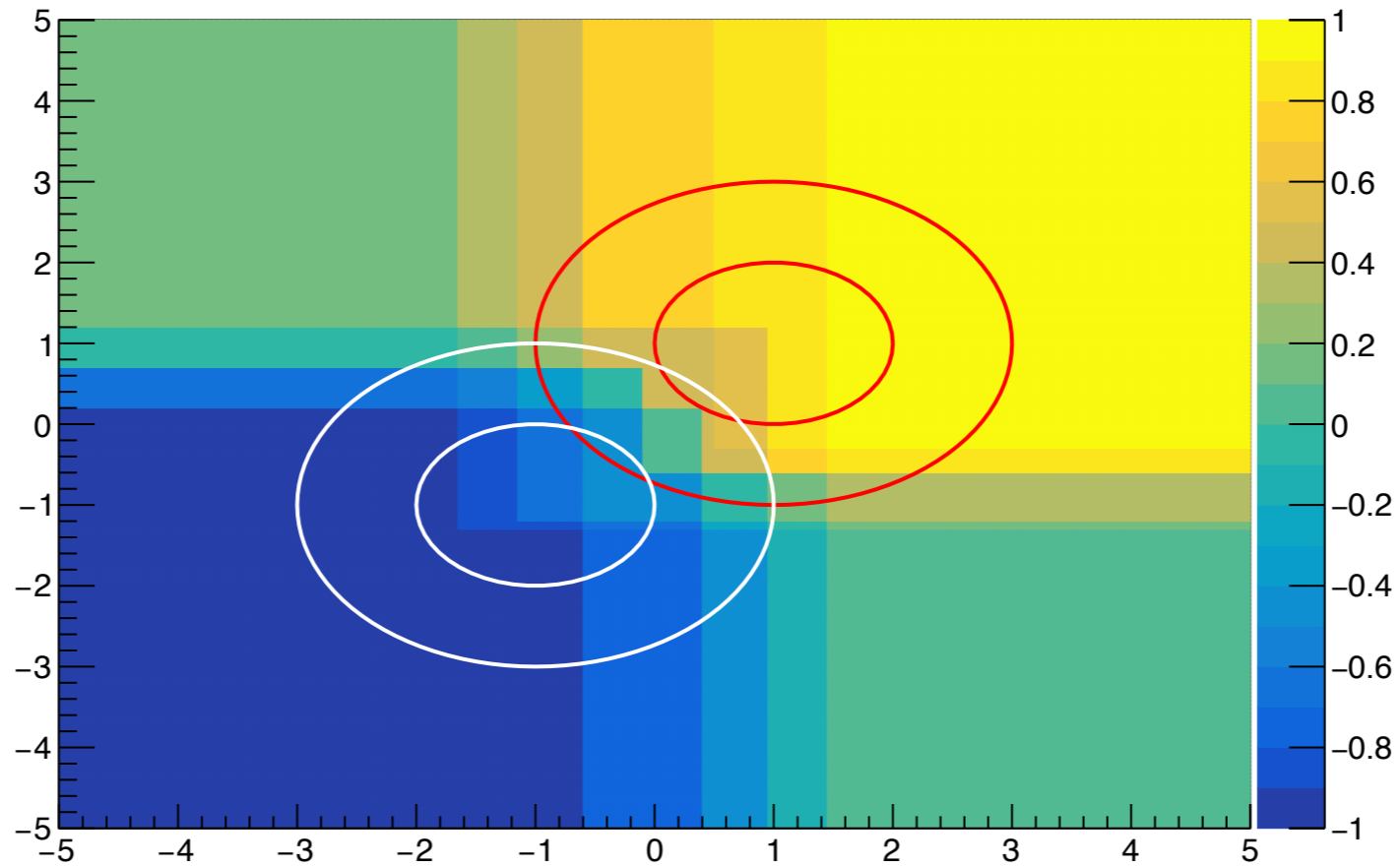
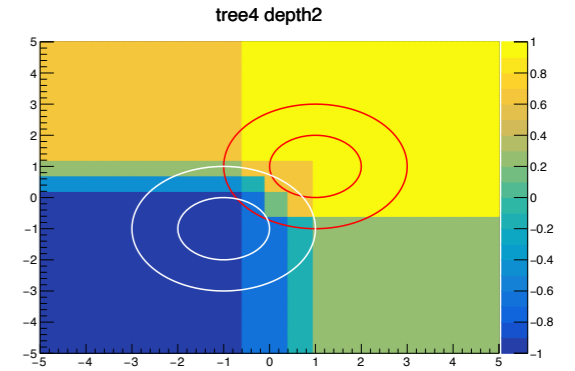
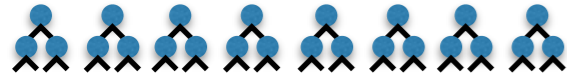


# Depth 2

vary



tree8 depth2

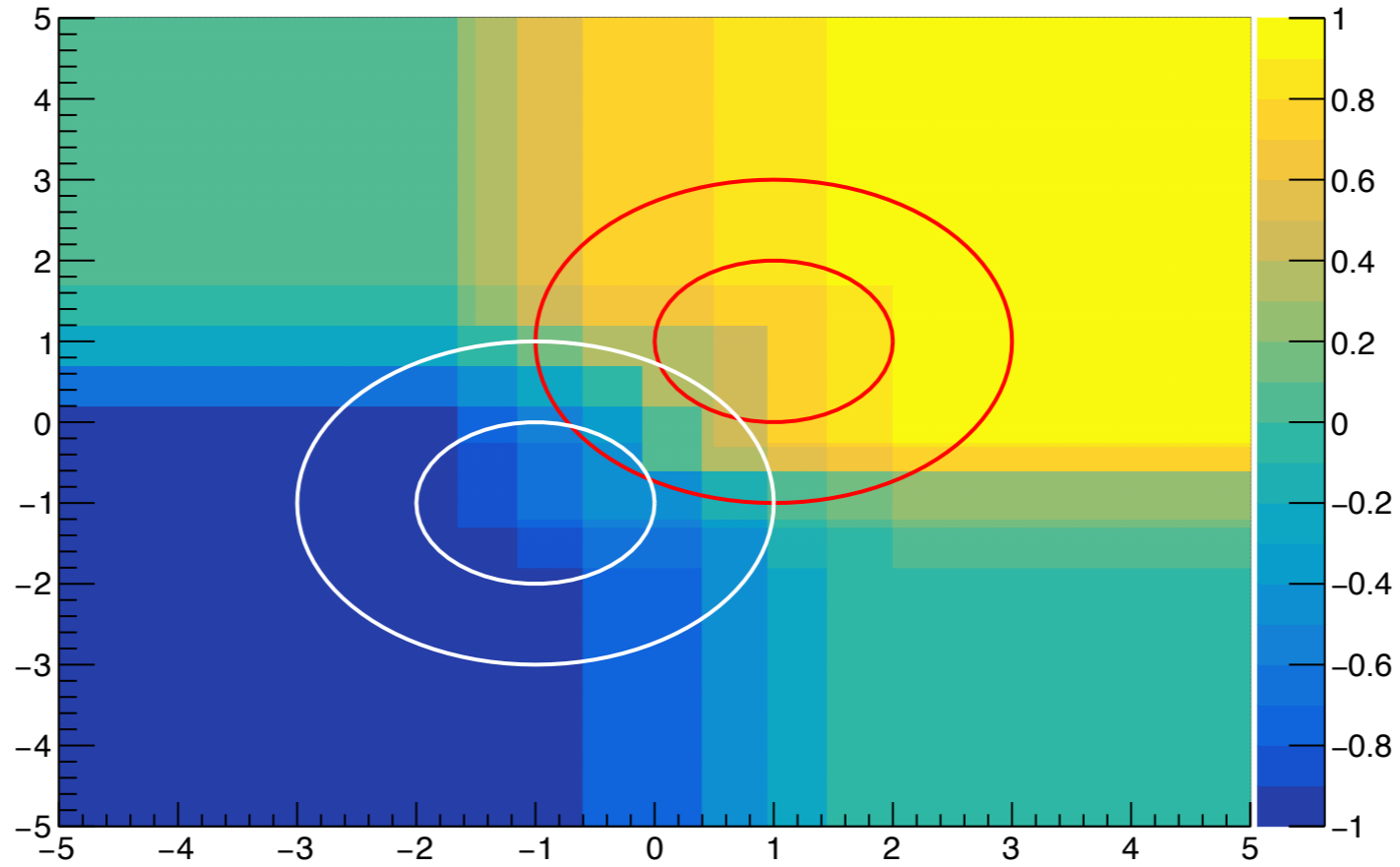
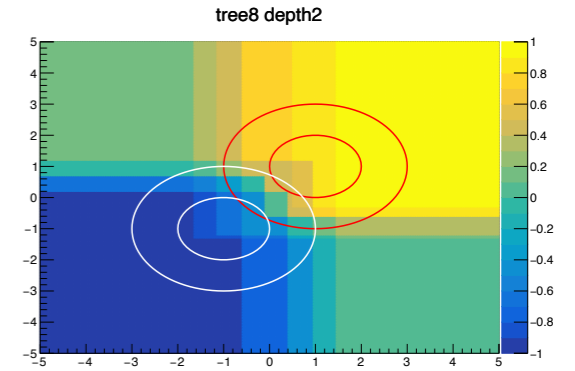


# Depth 2

vary



tree16 depth2

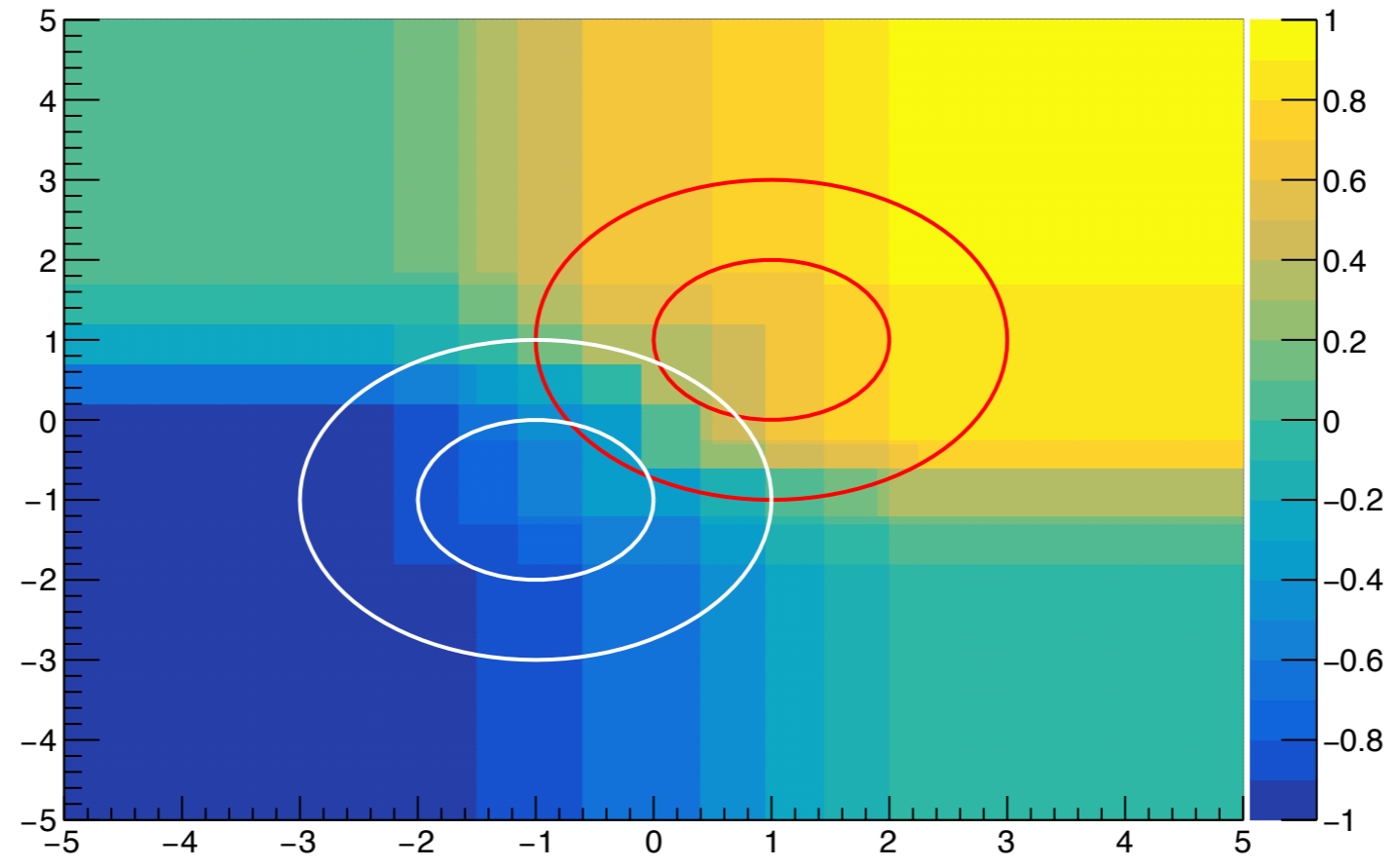
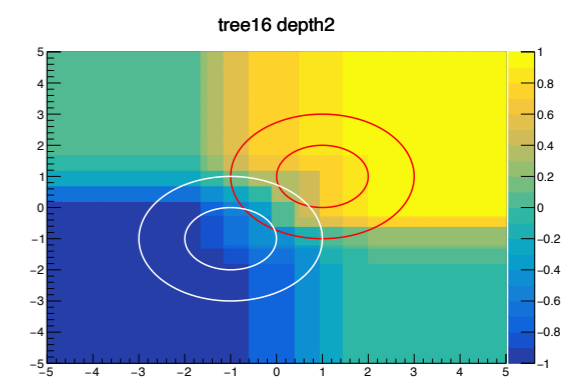


# Depth 2

vary



tree32 depth2

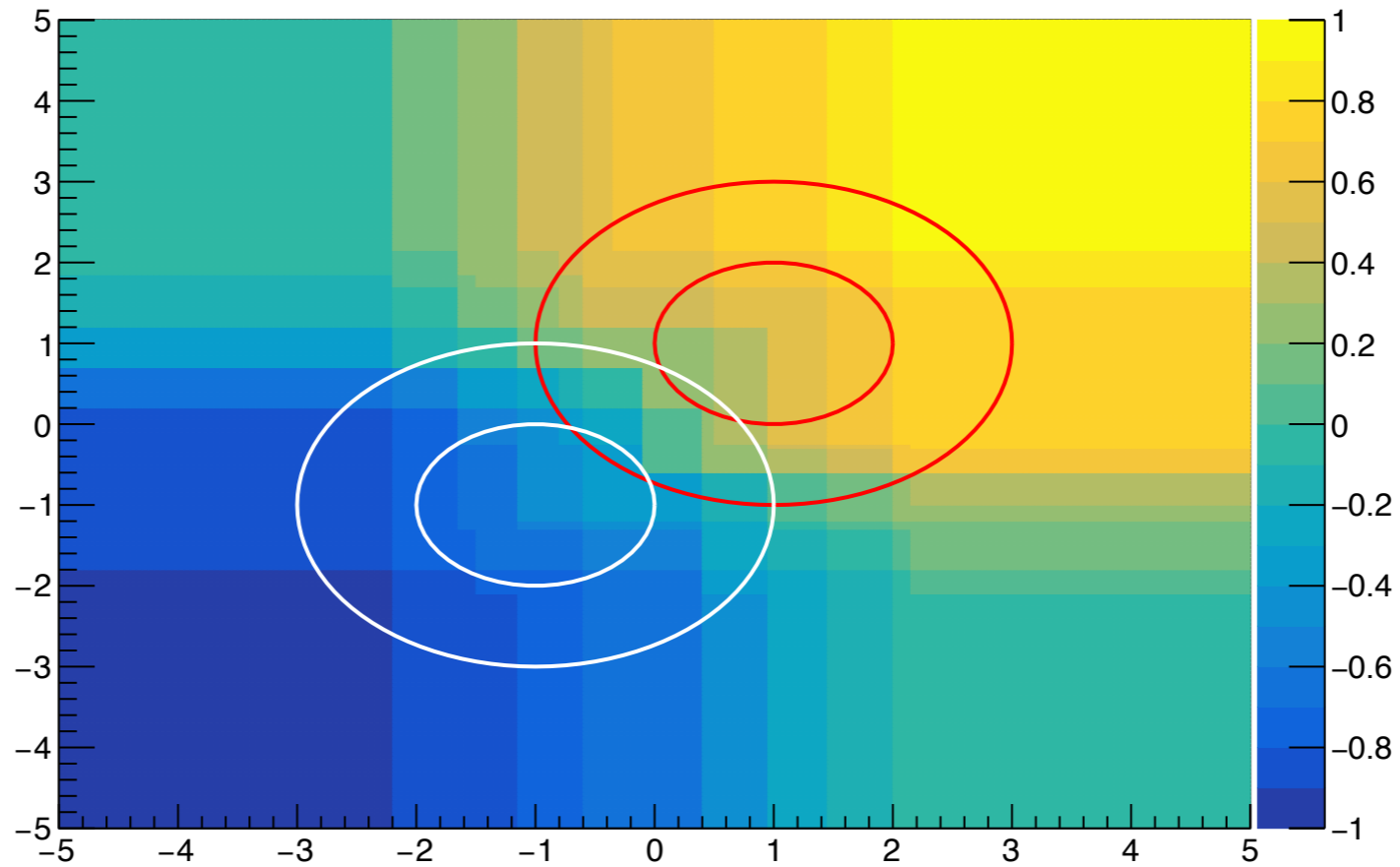
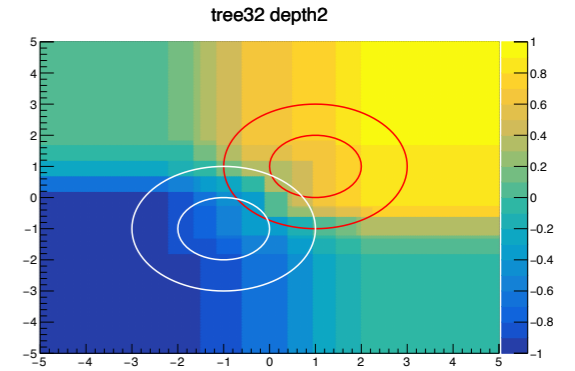


# Depth 2

vary



tree64 depth2

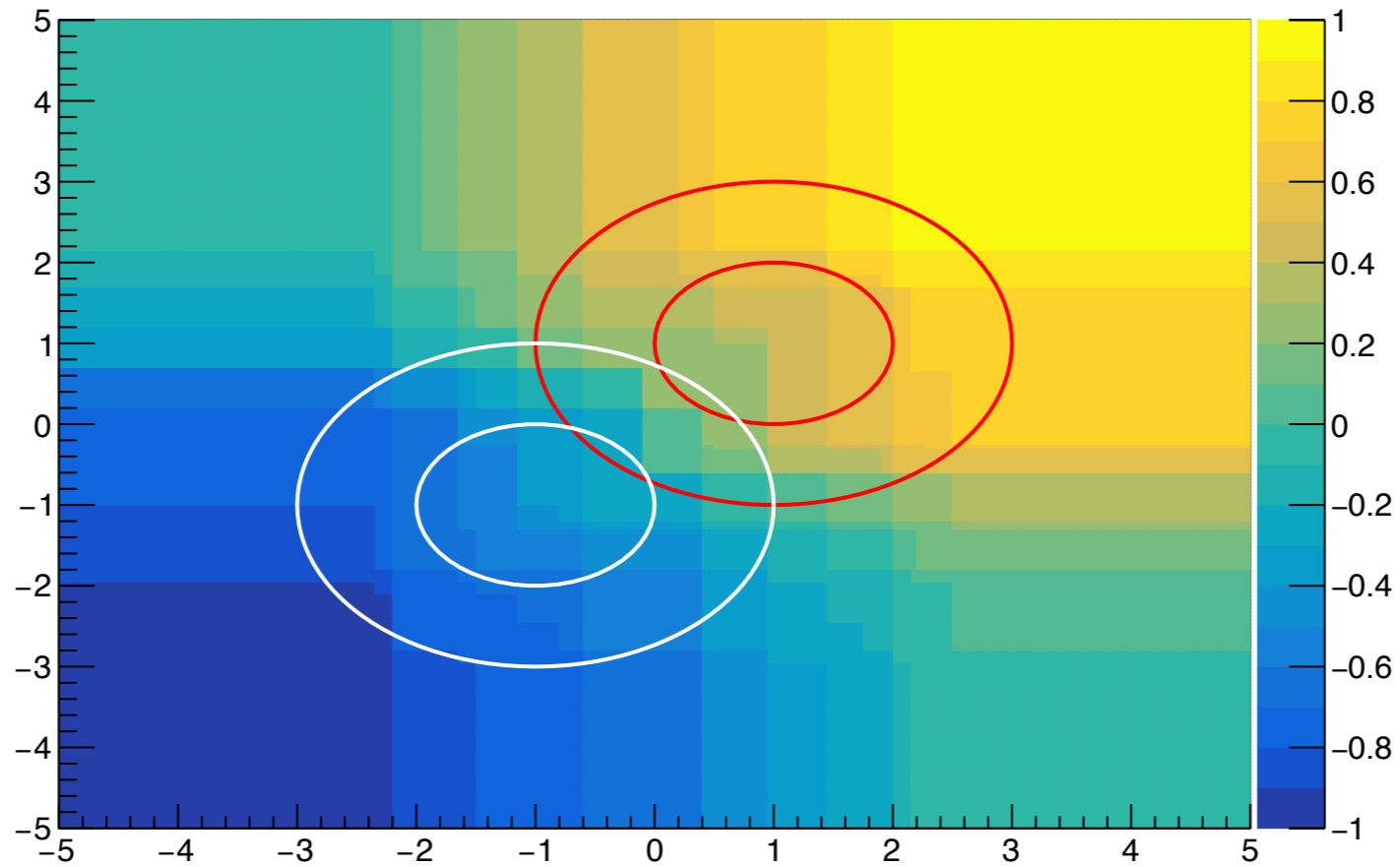
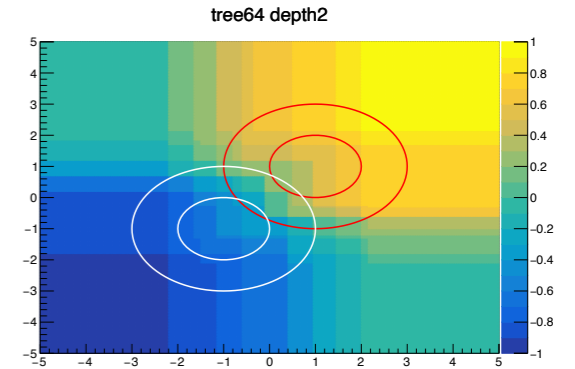


# Depth 2

vary



tree128 depth2



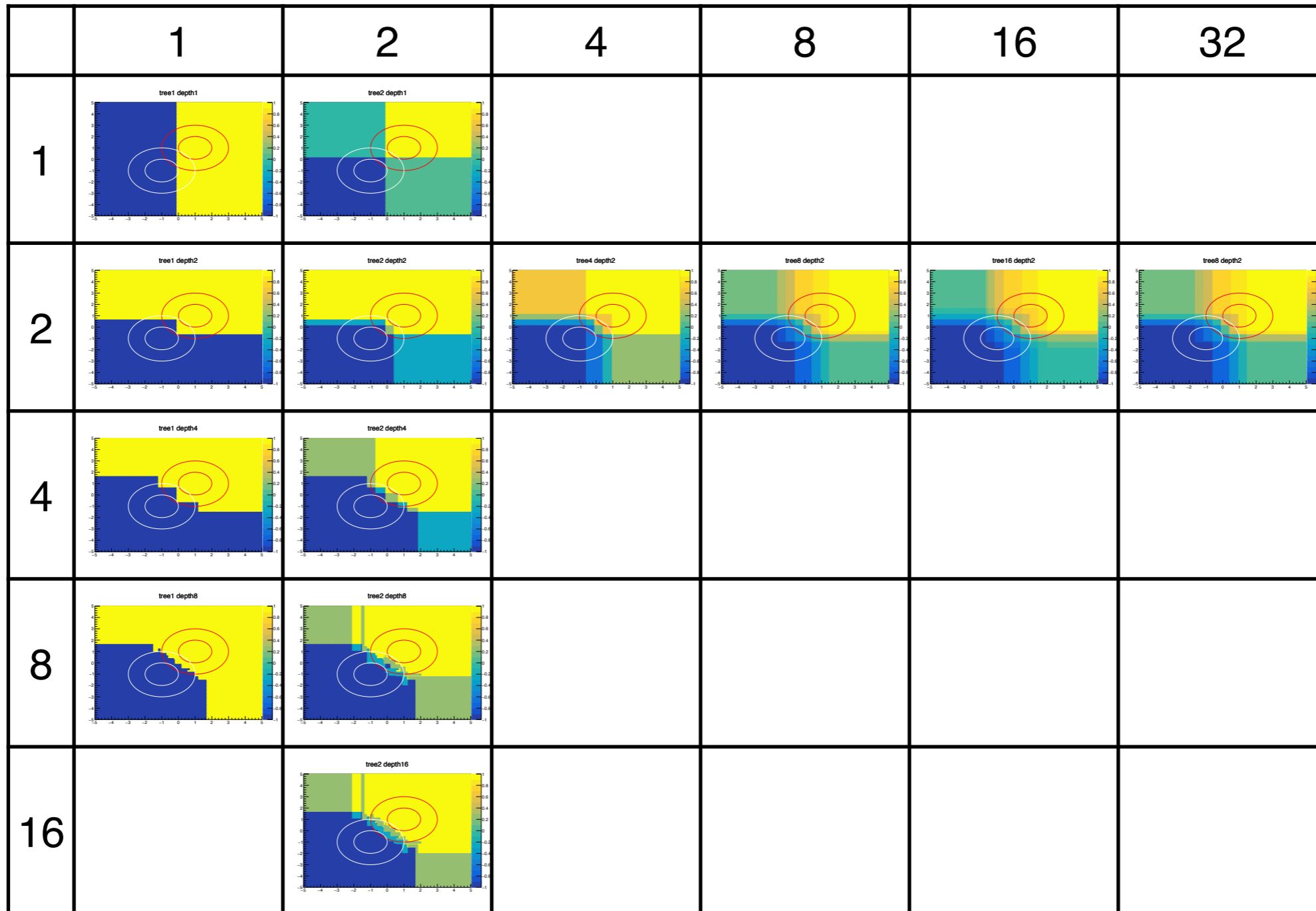
becomes very blurry



# Put it together on one slide

Tree →

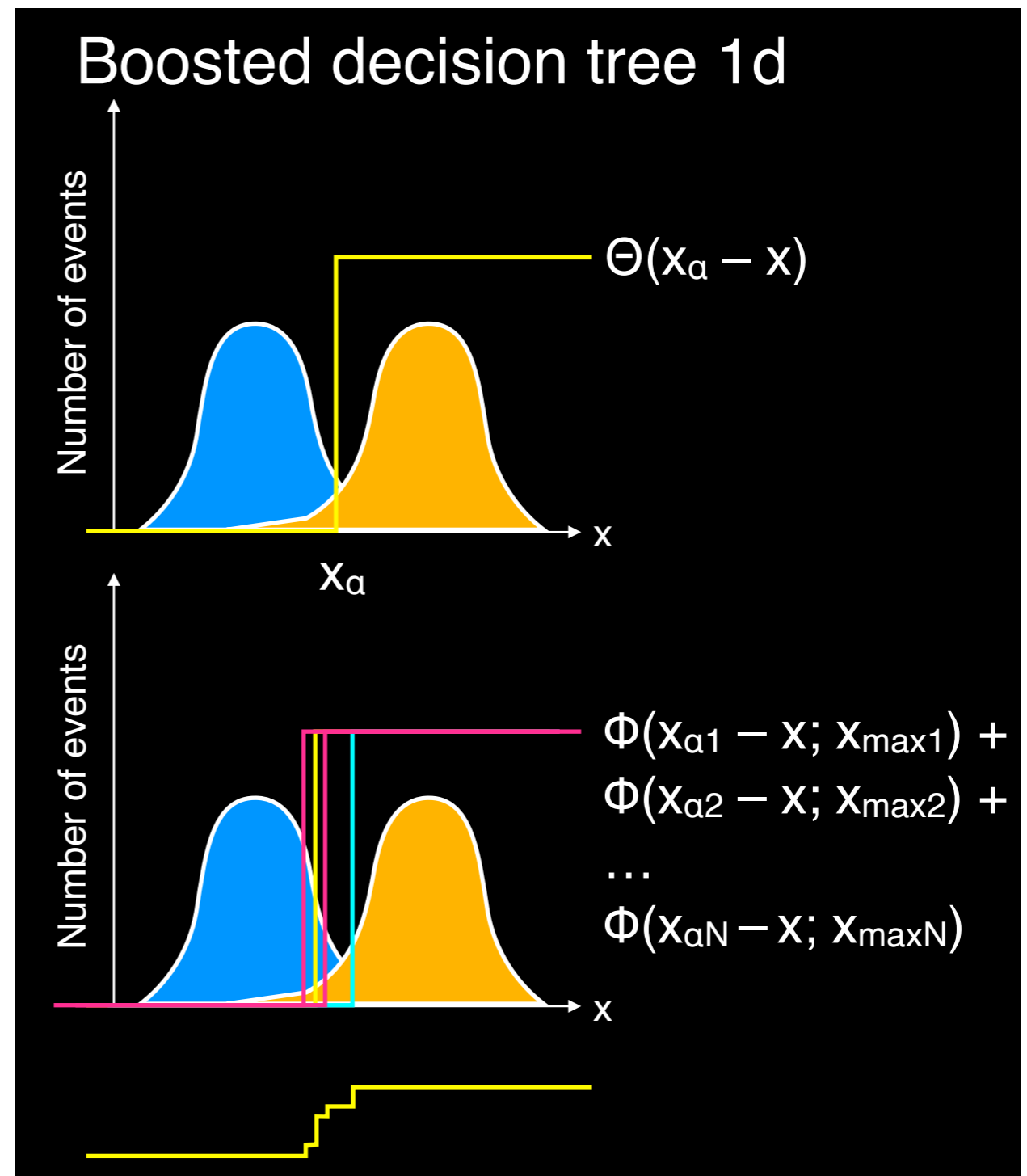
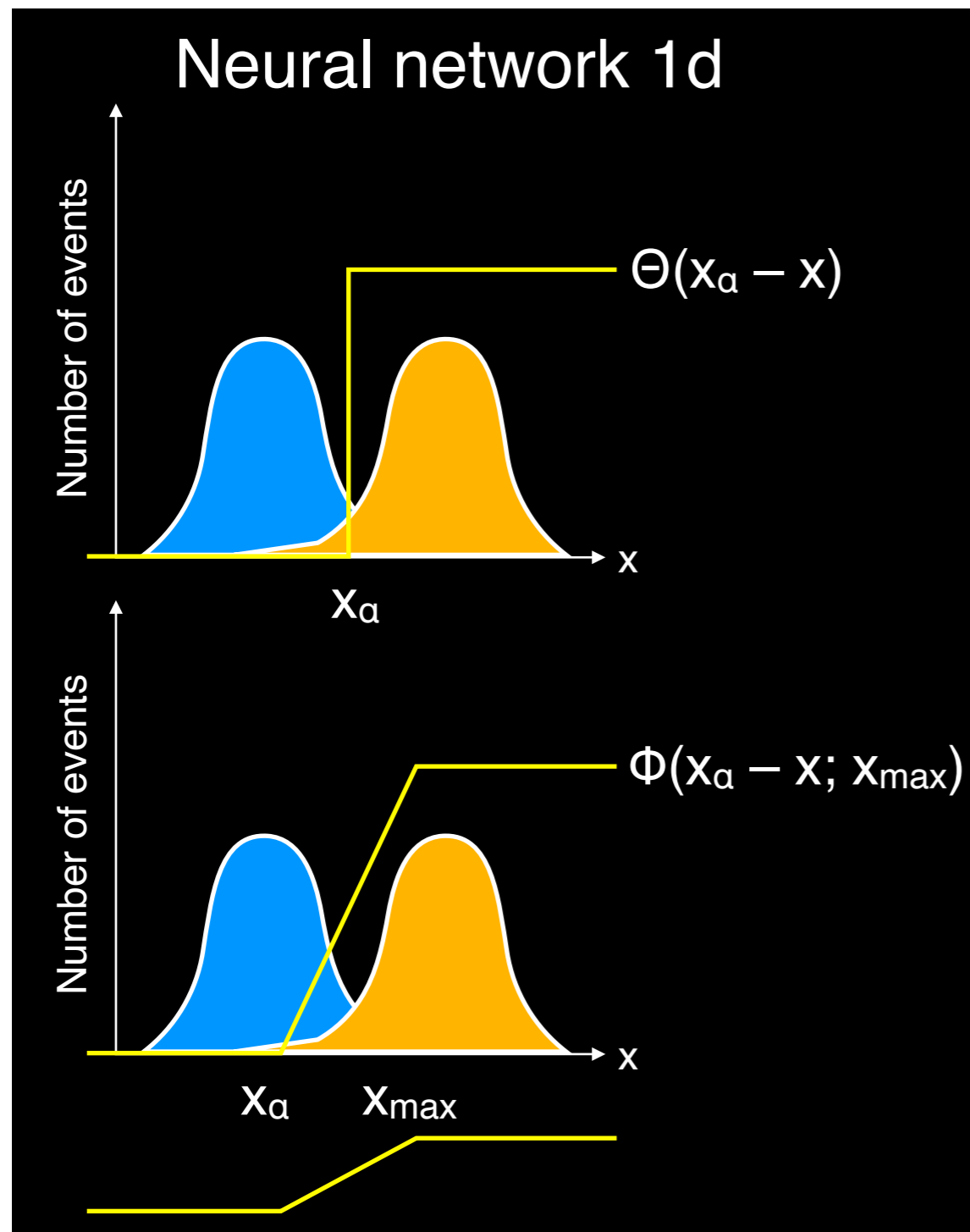
Depth ↓



Sweet spot depends on the physics problem

# Forest of decision trees

Fuzzy boundary by averaging step functions



Forest of decision trees provides the gradient



# Activation function

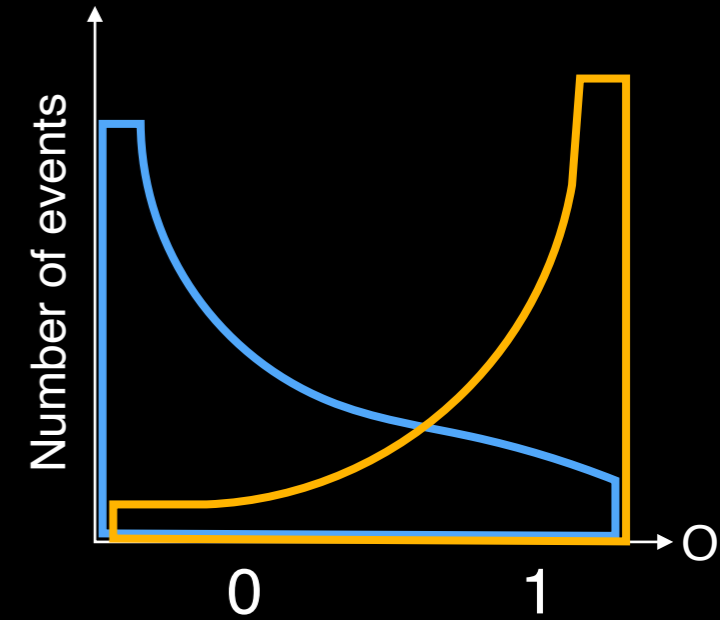
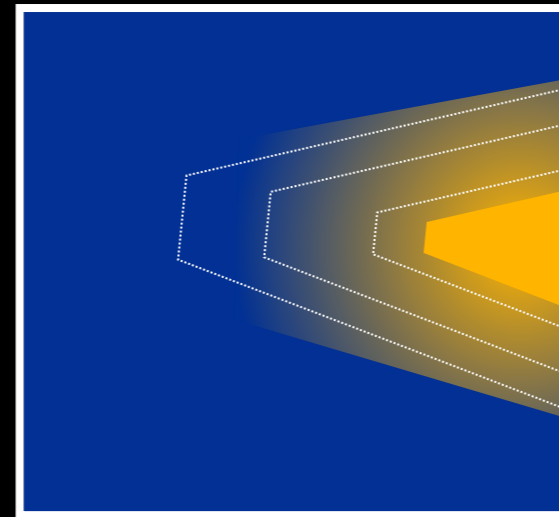
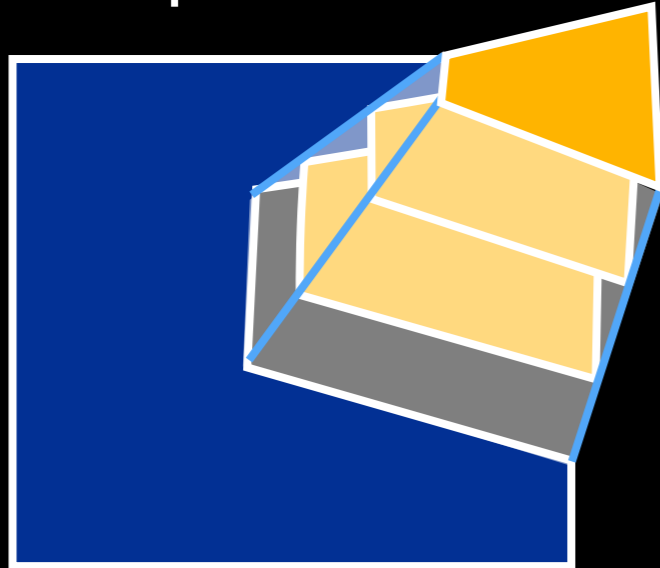
Fuzzy boundary using a function

2-dim inputs

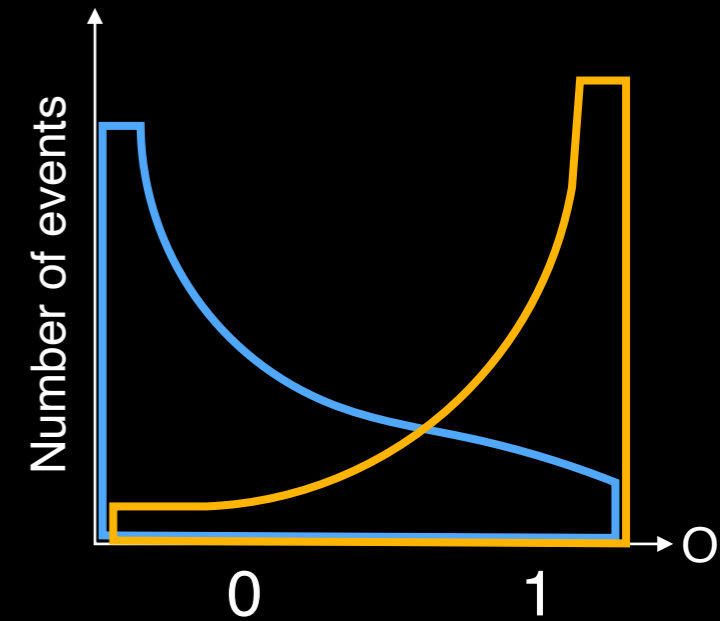
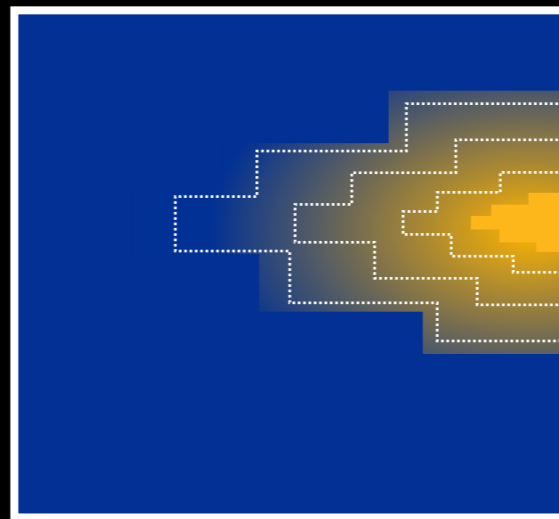
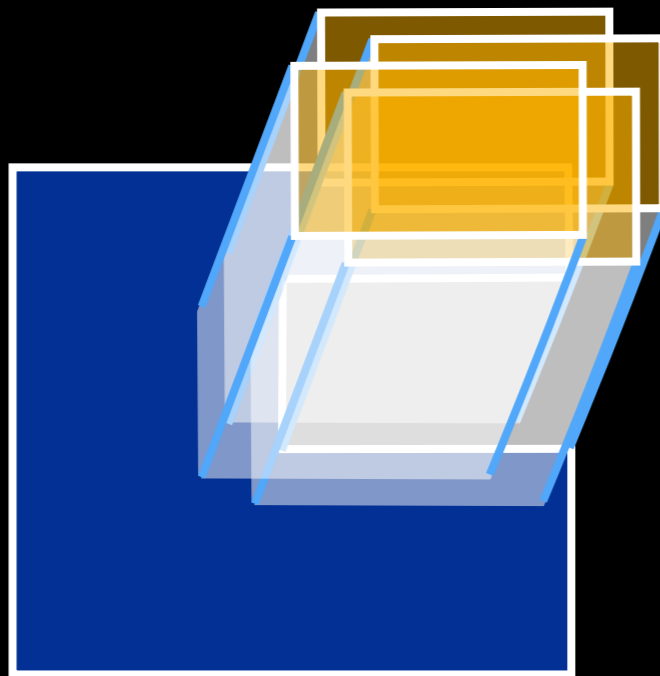
Projection

Output score

NN



BDT



Different approach, but same result