FWX Machina on FPGA for L1 trigger

Anomaly detection with decision tree autoencoder ^ Take 2

Pheno 2023, Take 1 talk at https://indico.cern.ch/event/1218225/contributions/5380942/

S.T. Roche
St. Louis U.
School of Medicine

B. Carlson
Westmont College

Tae Min Hong
University of Pittsburgh

DPF - Pheno
May 13, 2024
https://indico.cern.ch/event/1358339/contributions/5899270/

Outline

Introduction

- Autoencoders for anomaly detection
- Machine learning at L1

Decision tree autoencoder

- Novel training method
- Novel latent-spaceless design for FPGA

Physics & FPGA results

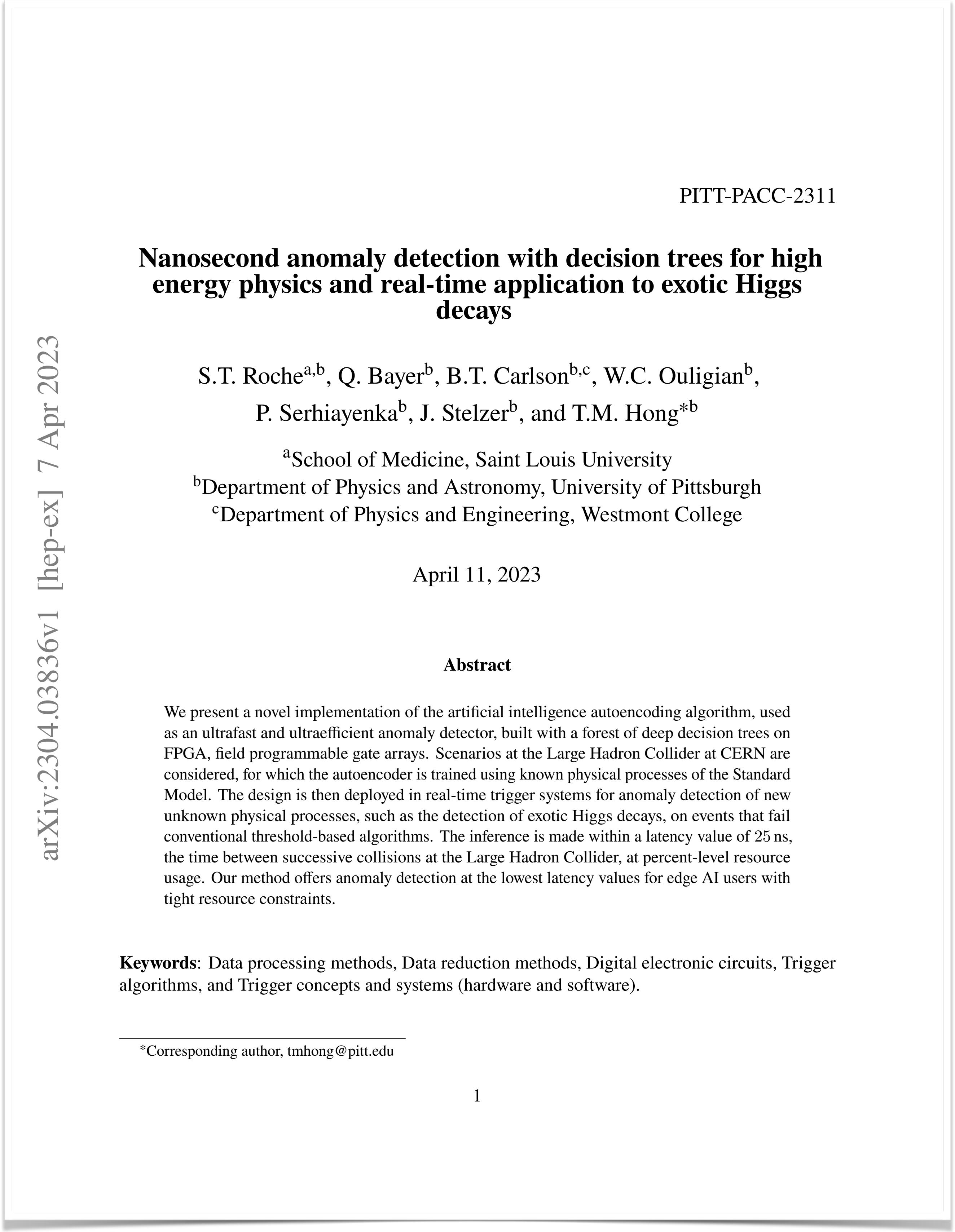
- Exotic decay of Higgs to pseudoscalars to γγ $b\bar{b}$
- “LHC anomaly detection” dataset

Thoughts

save

How to ~~find~~ BSM without models at L1

Take 2


arXiv:2304.03836v1 [hep-ex] 7 Apr 2023

PITT-PACC-2311

Nanosecond anomaly detection with decision trees for high energy physics and real-time application to exotic Higgs decays

S.T. Roche[a,b], Q. Bayer[b], B.T. Carlson[b,c], W.C. Ouligian[b], P. Serhiayenka[b], J. Stelzer[b], and T.M. Hong[*b]

[a]School of Medicine, Saint Louis University
[b]Department of Physics and Astronomy, University of Pittsburgh
[c]Department of Physics and Engineering, Westmont College

April 11, 2023

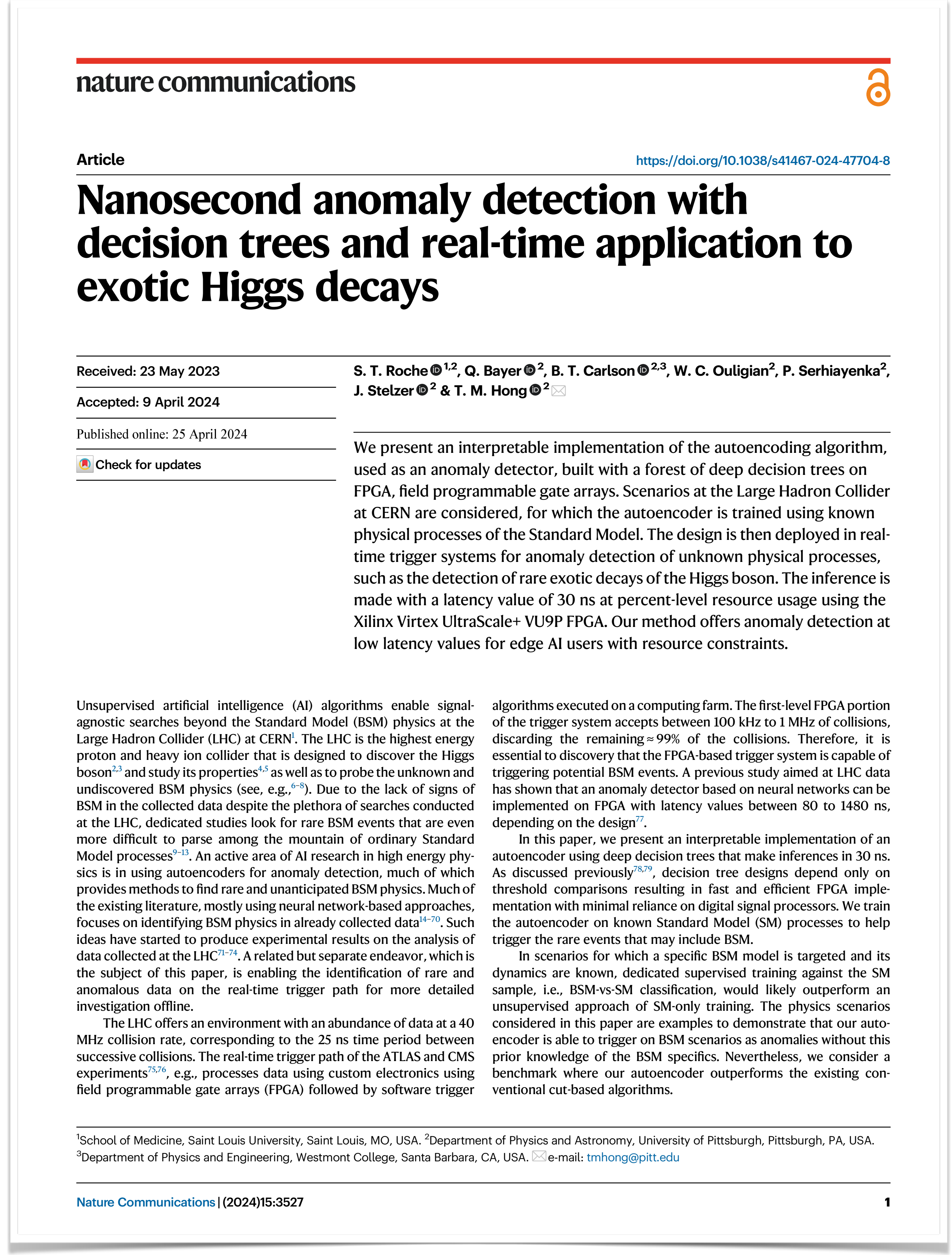
Abstract

We present a novel implementation of the artificial intelligence autoencoding algorithm, used as an ultrafast and ultraefficient 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 new unknown physical processes, such as the detection of exotic Higgs decays, on events that fail conventional threshold-based algorithms. The inference is made within a latency value of 25 ns, the time between successive collisions at the Large Hadron Collider, at percent-level resource usage. Our method offers anomaly detection at the lowest latency values for edge AI users with tight resource constraints.

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


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

Received: 23 May 2023

Accepted: 9 April 2024

Published online: 25 April 2024

Check for updates

S. T. Roche[1,2], Q. Bayer[2], B. T. Carlson[2,3], W. C. Ouligian[2], P. Serhiayenka[2], J. Stelzer[2] & T. M. Hong[2]

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 (BSM) physics at the Large Hadron Collider (LHC) at CERN[1]. The LHC is the highest energy proton and heavy ion collider that is designed to discover the Higgs boson[2,3] and study its properties[4,5] as well as to probe the unknown and undiscovered BSM physics (see, e.g.,[6–8]). 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[9–13]. 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[14–70]. Such ideas have started to produce experimental results on the analysis of data collected at the LHC[71–74]. 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[75,76], 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 discovery 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[77].

In this paper, we present an interpretable implementation of an autoencoder using deep decision trees that make inferences in 30 ns. As discussed previously[78,79], 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.

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

$H_{125} \to a_{10}\, a_{15} \to e^+e^-\, \mu^+\mu^-$

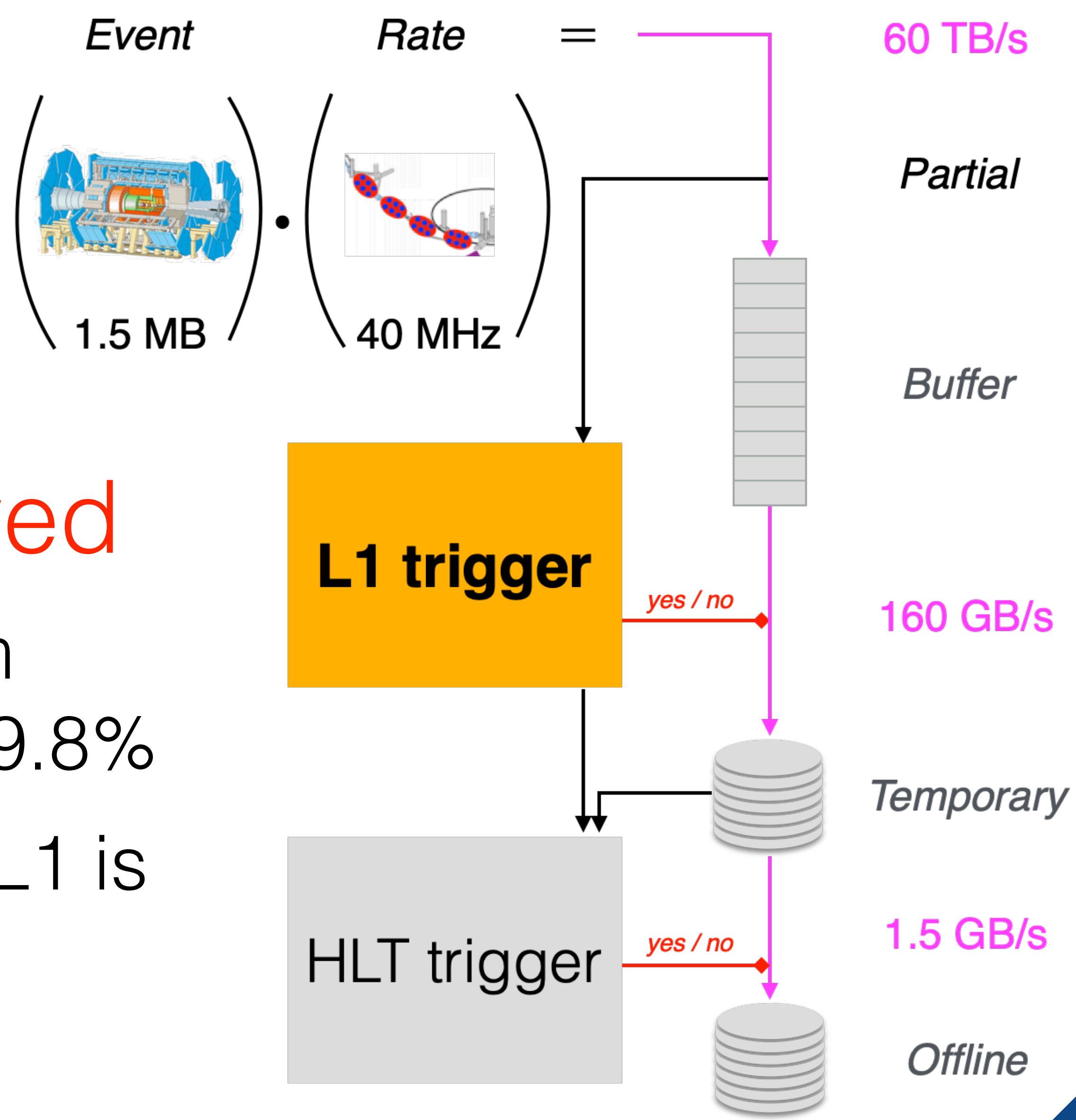
$H_{125} \to a_{10}\, a_{70} \to \gamma\gamma\, b\bar{b}$

+

Thoughts

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

Prior work

TM Hong

Autoencoder

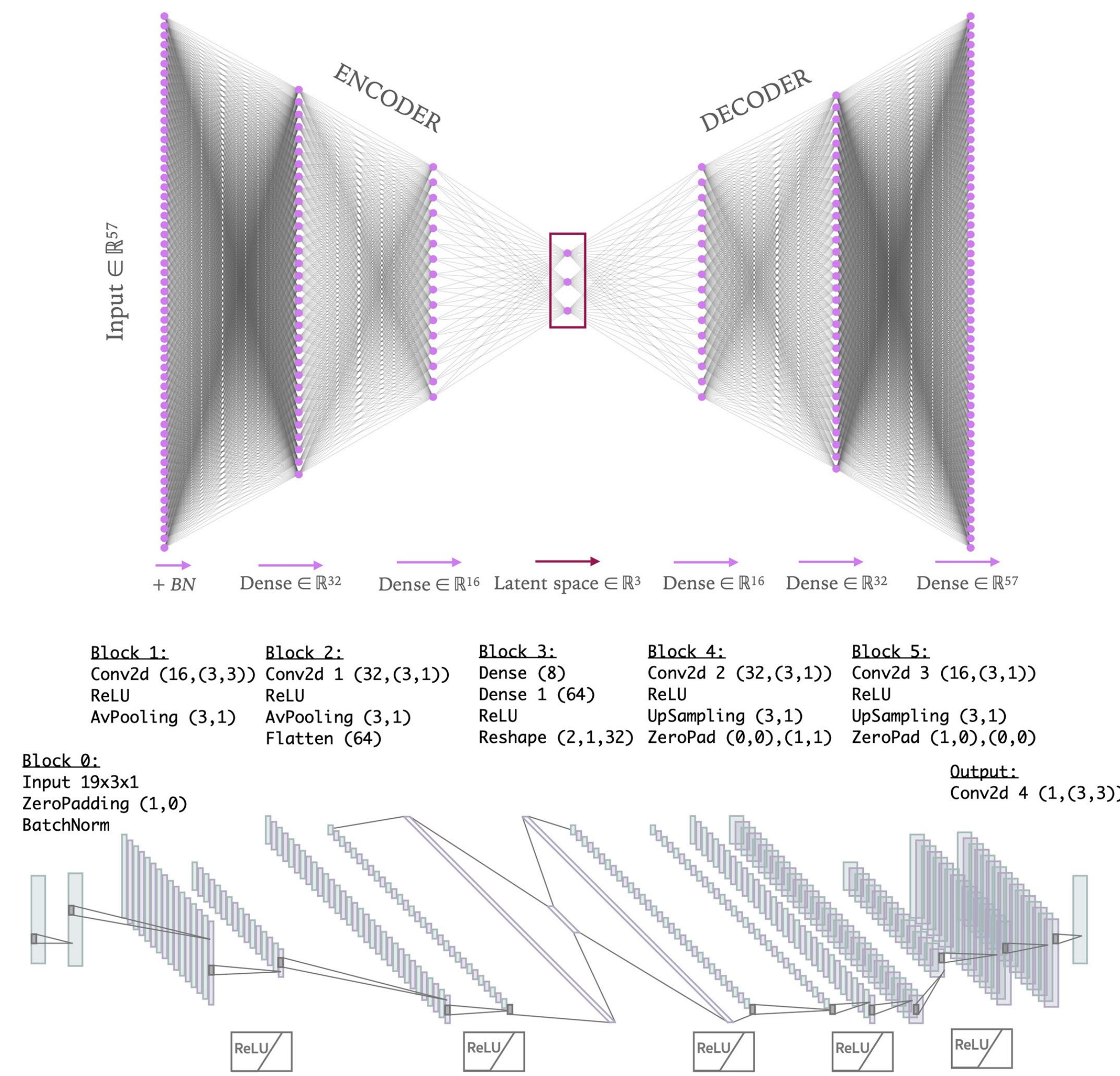
- Typically constructed using neural networks
- Challenge to implement in pure digital logic on FPGA
- NN example shown on right →

Decision tree?

- Used in our work
- Has certain advantages: technical (no multiplication) & philosophical (interpretable)

Govorkova et al., *Autoencoders on field-programmable gate arrays for real-time, unsupervised new physics detection at 40 MHz at the Large Hadron Collider,* Nature Mach. Intell. **4** (2022) 154–161
https://doi.org/10.1038/s42256-022-00441-3

ARTICLES NATURE MACHINE INTELLIGENCE



Extended Data Fig. 1 | Network architectures. Network architecture for the DNN AE (top) and CNN AE (bottom) models. The corresponding VAE models are derived introducing the Gaussian sampling in the latent space, for the same encoder and decoder architectures (see text).

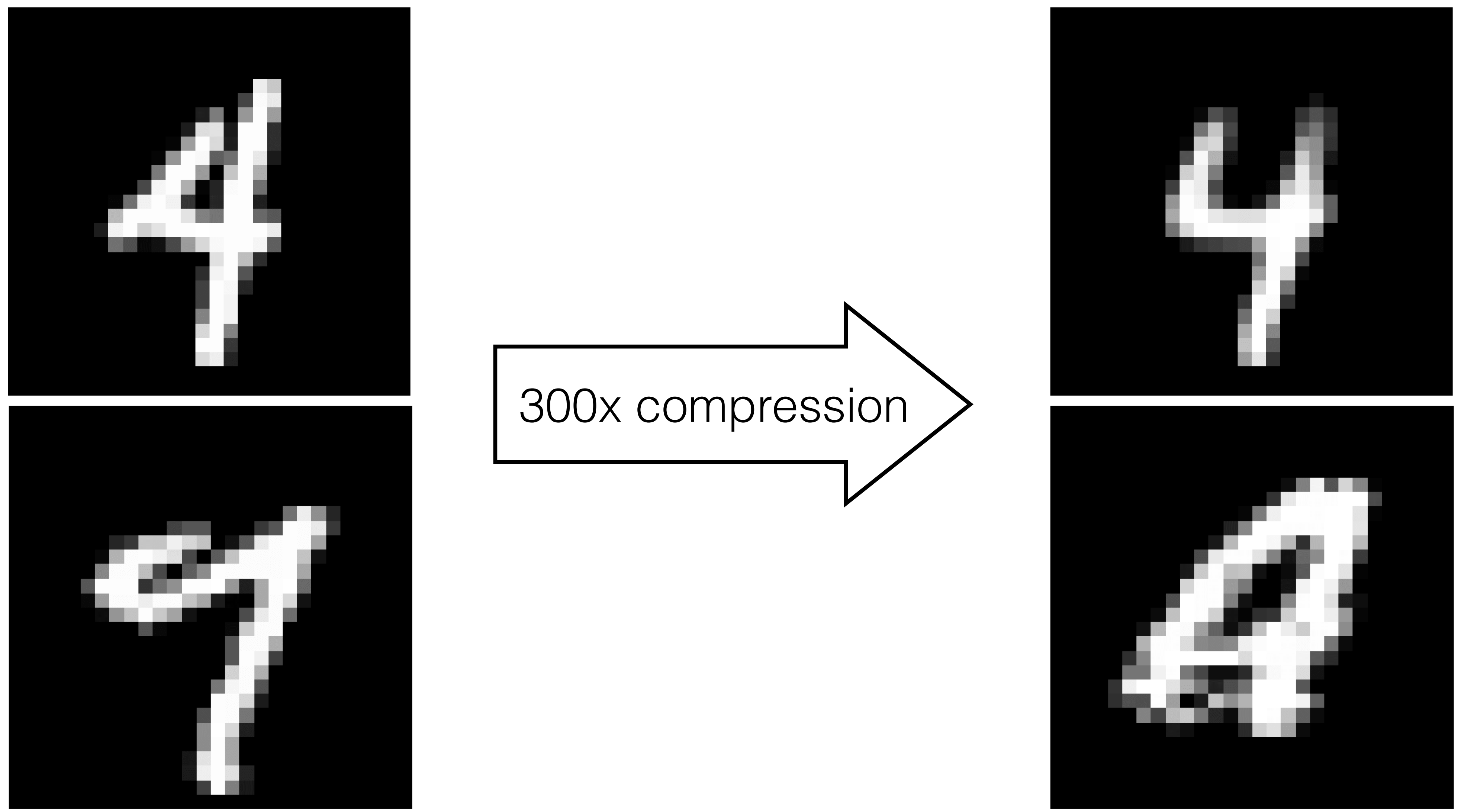
Autoencoder intro

TM Hong

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) | 784 variables (8-bit)



Details

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

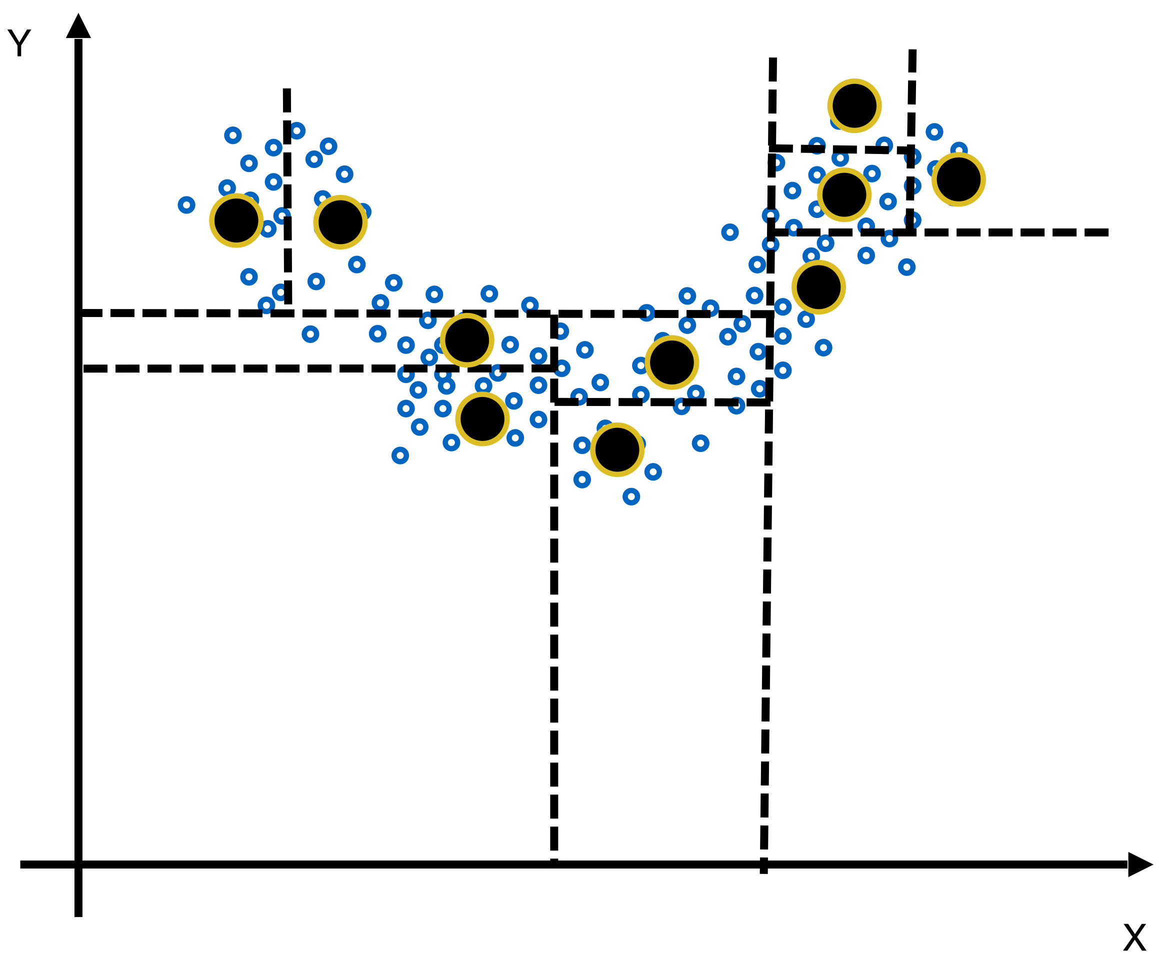
Decision tree autoencoders

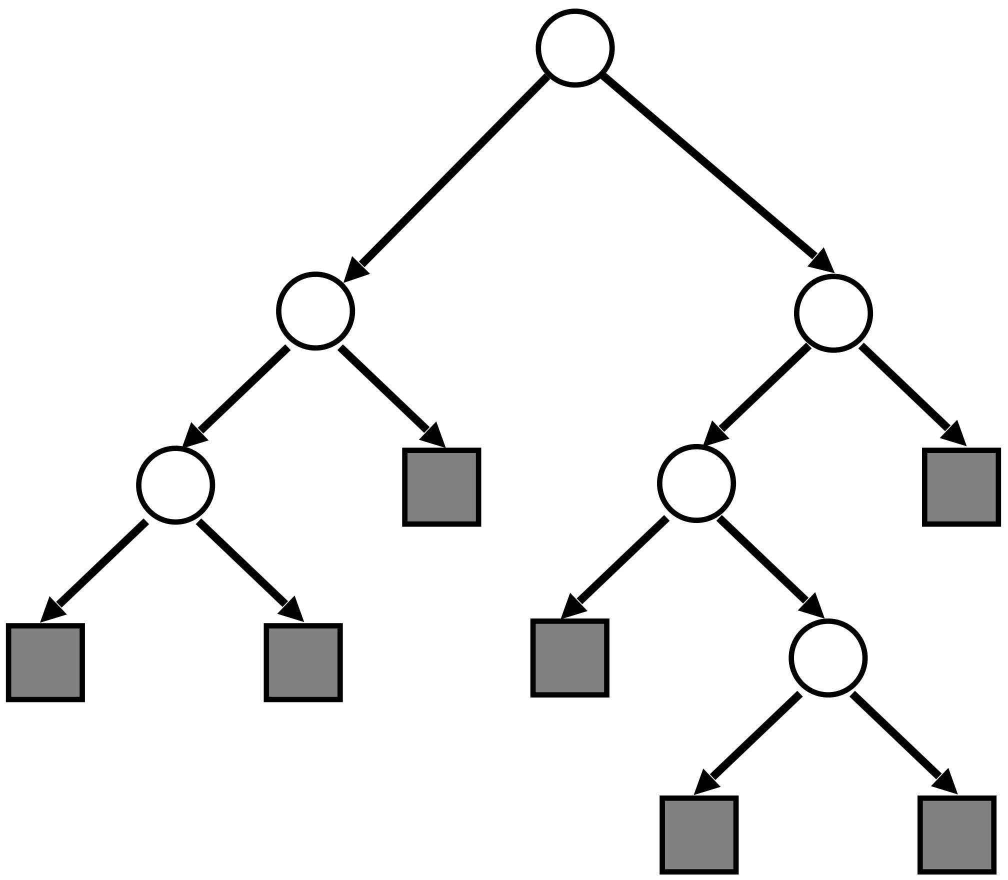
Train by sampling 1d projections

- Encoding: Event → which bin it's in

Decode by returning a "reconstruction point"

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

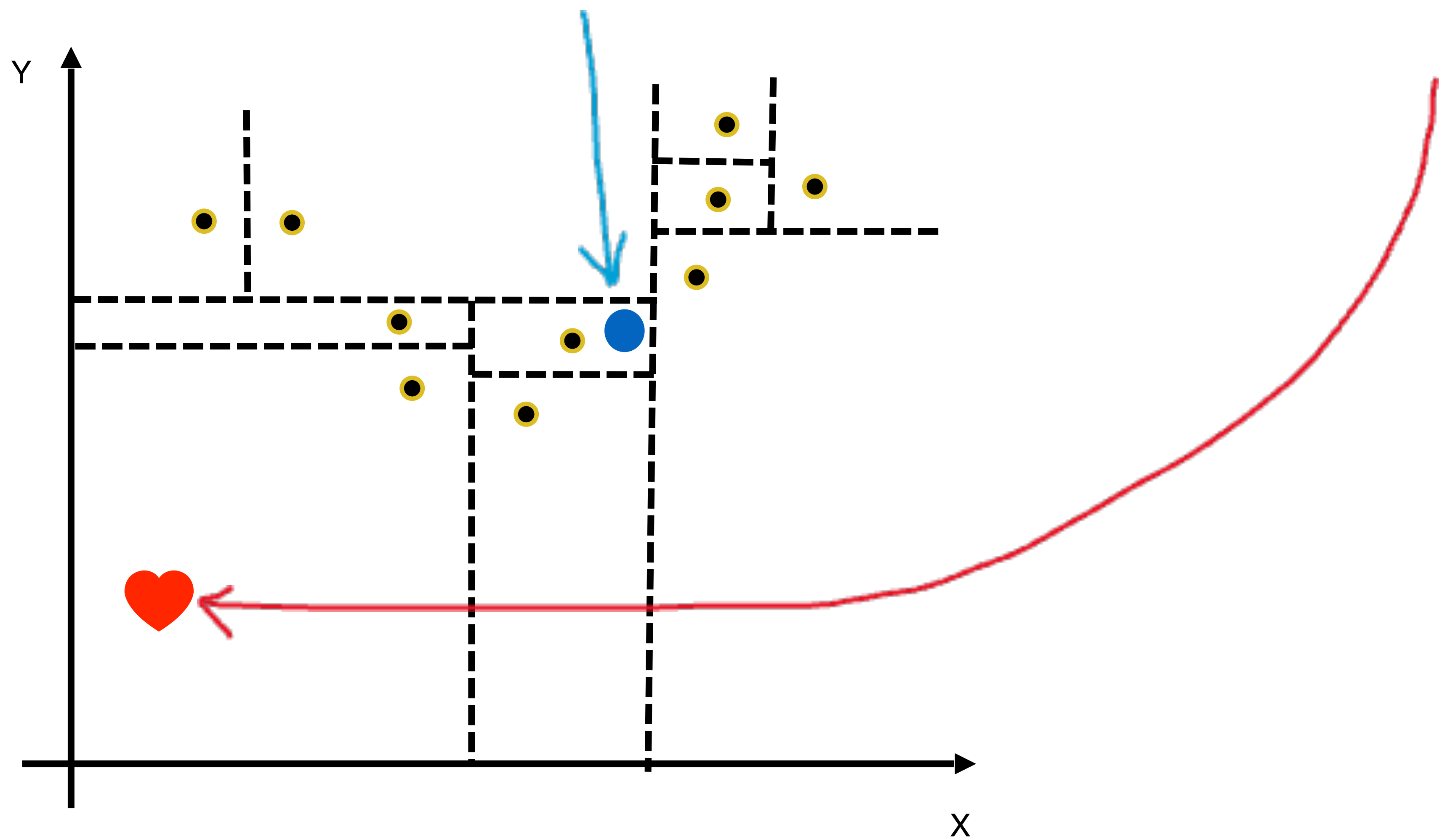




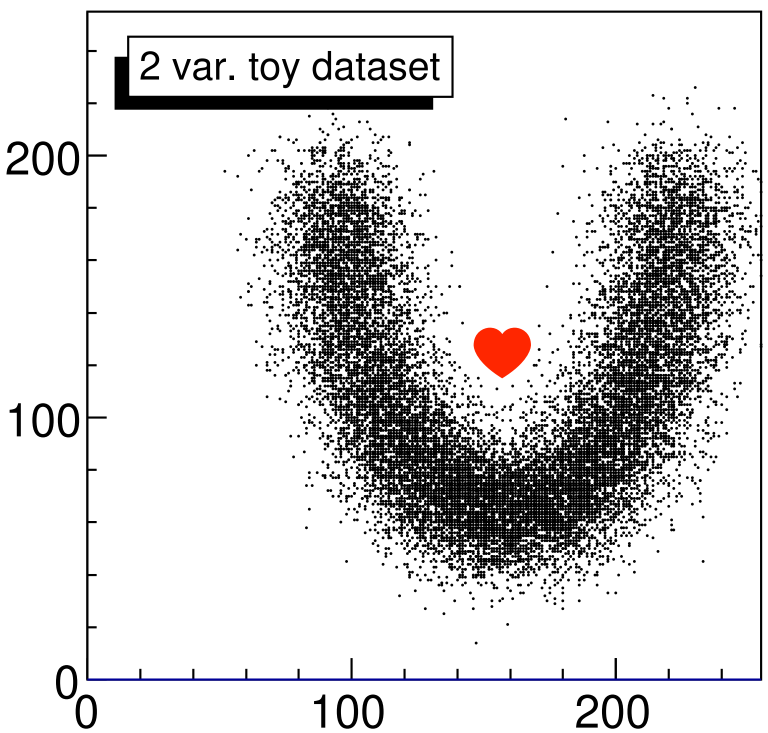
How does this detect anomalies?

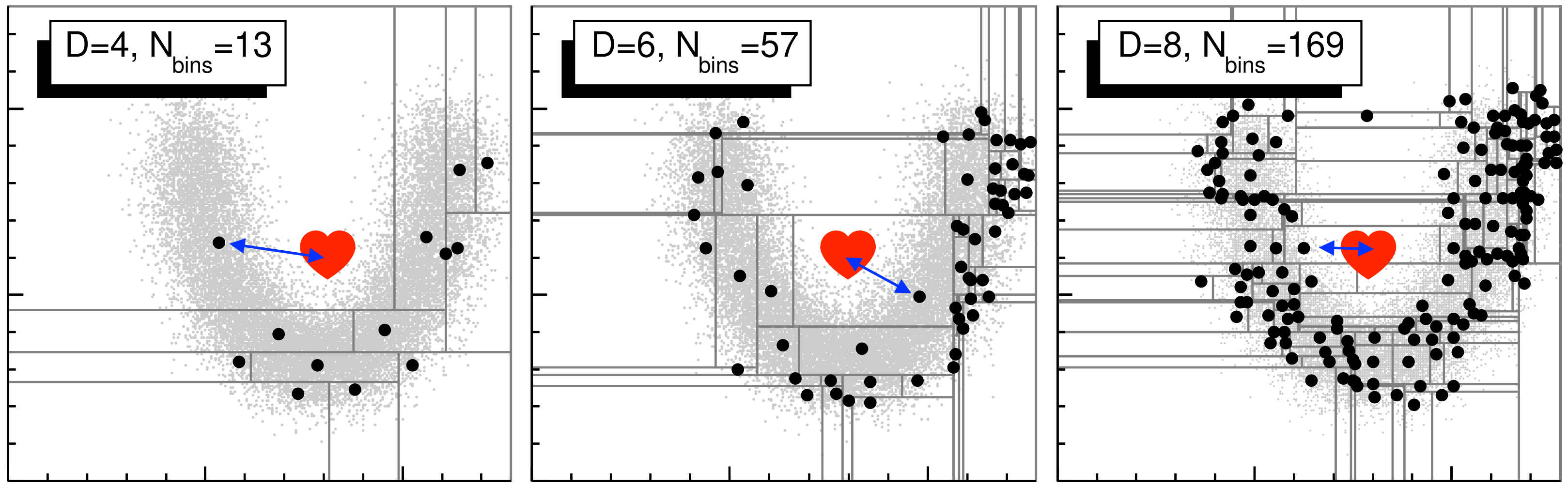
- Define: Distance between input – output = anomaly score

- Non-anomaly
  - Input is similar to training data
  - Will likely land in a small bin → 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



Toy dataset (2 input variables)

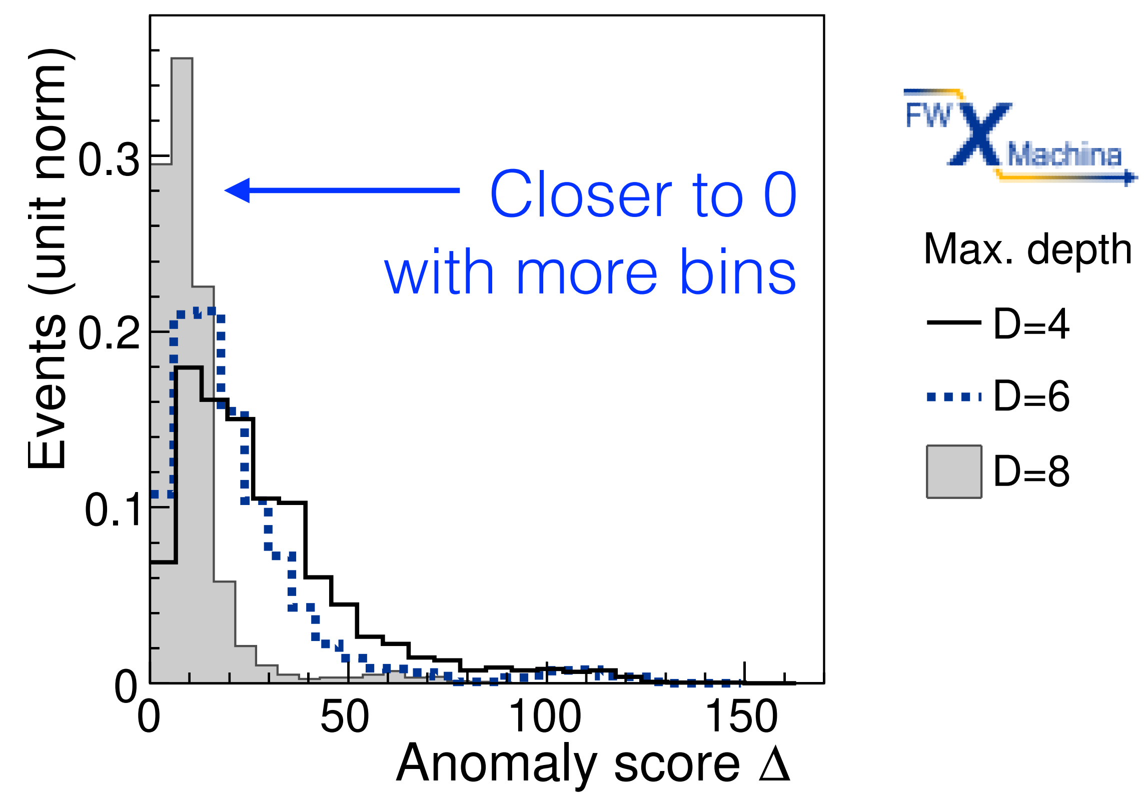




more bins

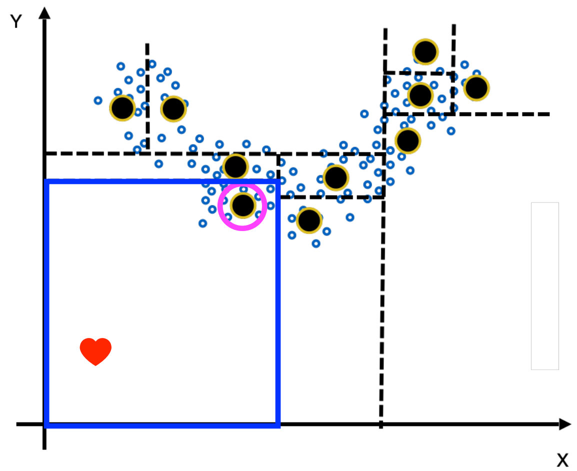
Anomaly score

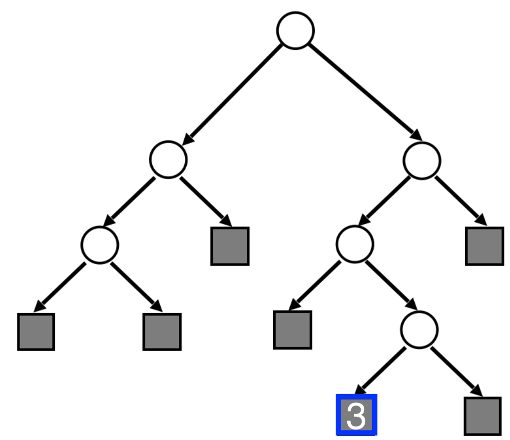
- Feed back in the training sample
  - Should be near 0, like E_T^{miss} resolution



Latent spaceless implementation

- Closer look at what it means to encode





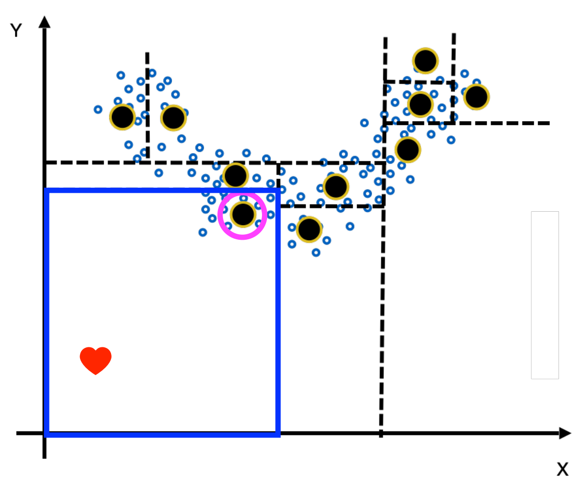
Bin	x	y
1	...	...
2	...	...
3	5	4
...		

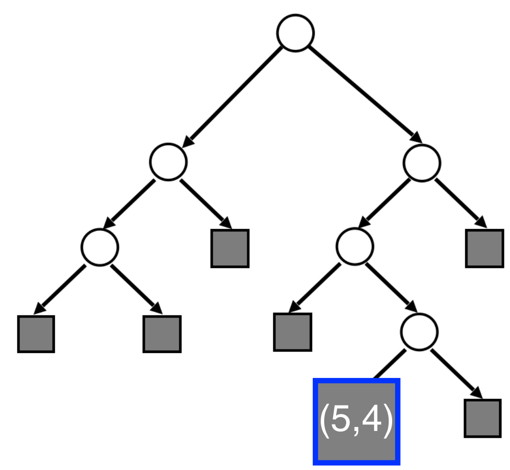
Incoming heart

Encode: return bin 3

Decode bin 3: return (5,4)

- Skip the encoding & decoding





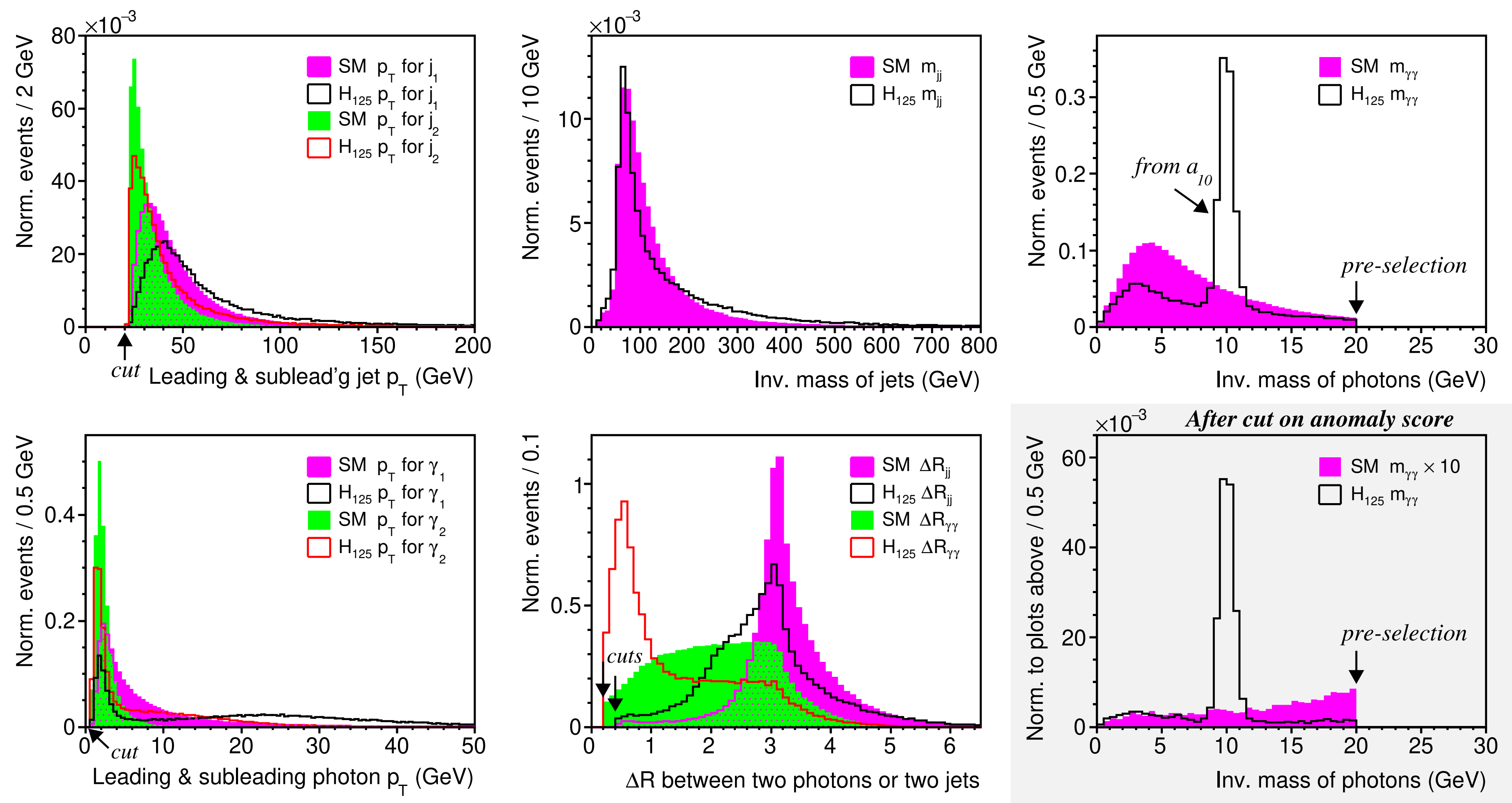
Incoming heart

Encode is Decode: return (5,4)

$H_{125} \rightarrow a_{10}\, a_{70} \rightarrow \gamma\gamma\, b\bar{b}$

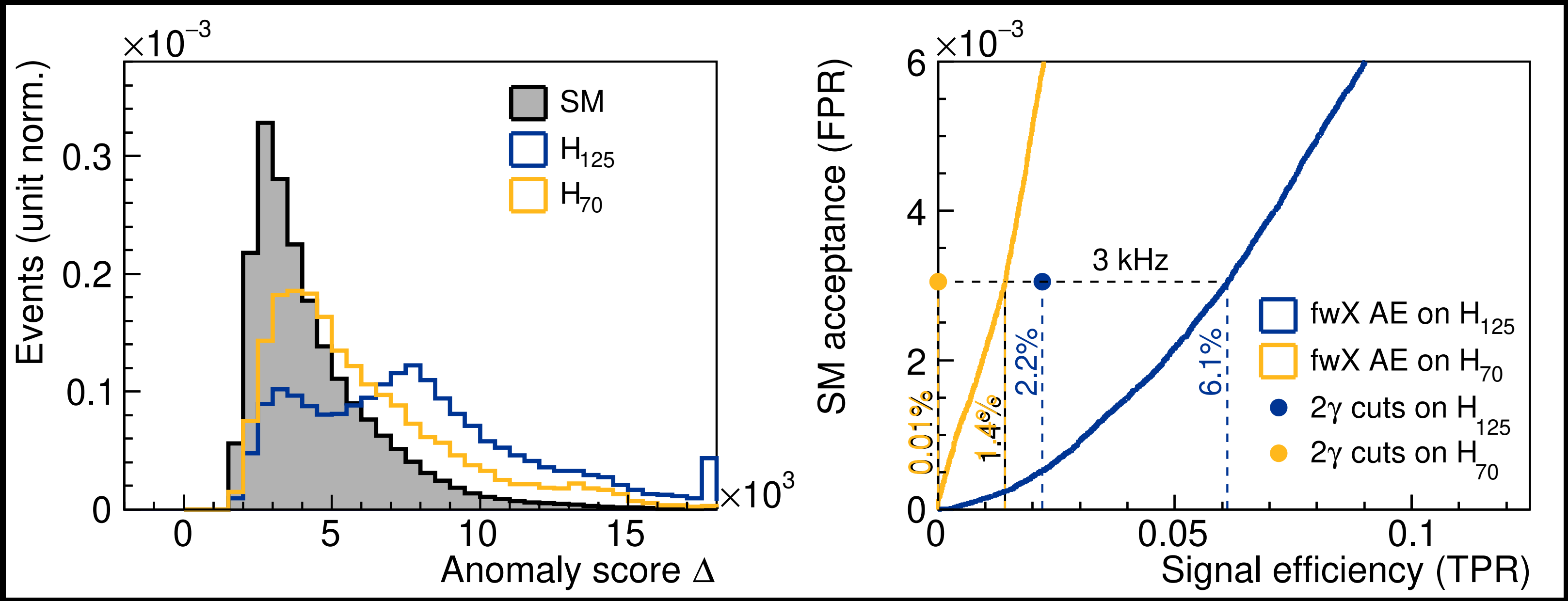
Inputs

- Sample
  - MadGraph5_aMC 2.9.5
  - Hadron'n+Shower: Pythia8
  - Detector: Delphes 3.5.0, CMS
- Variables
  - 8 inputs: jets, photons, ΔR



Results

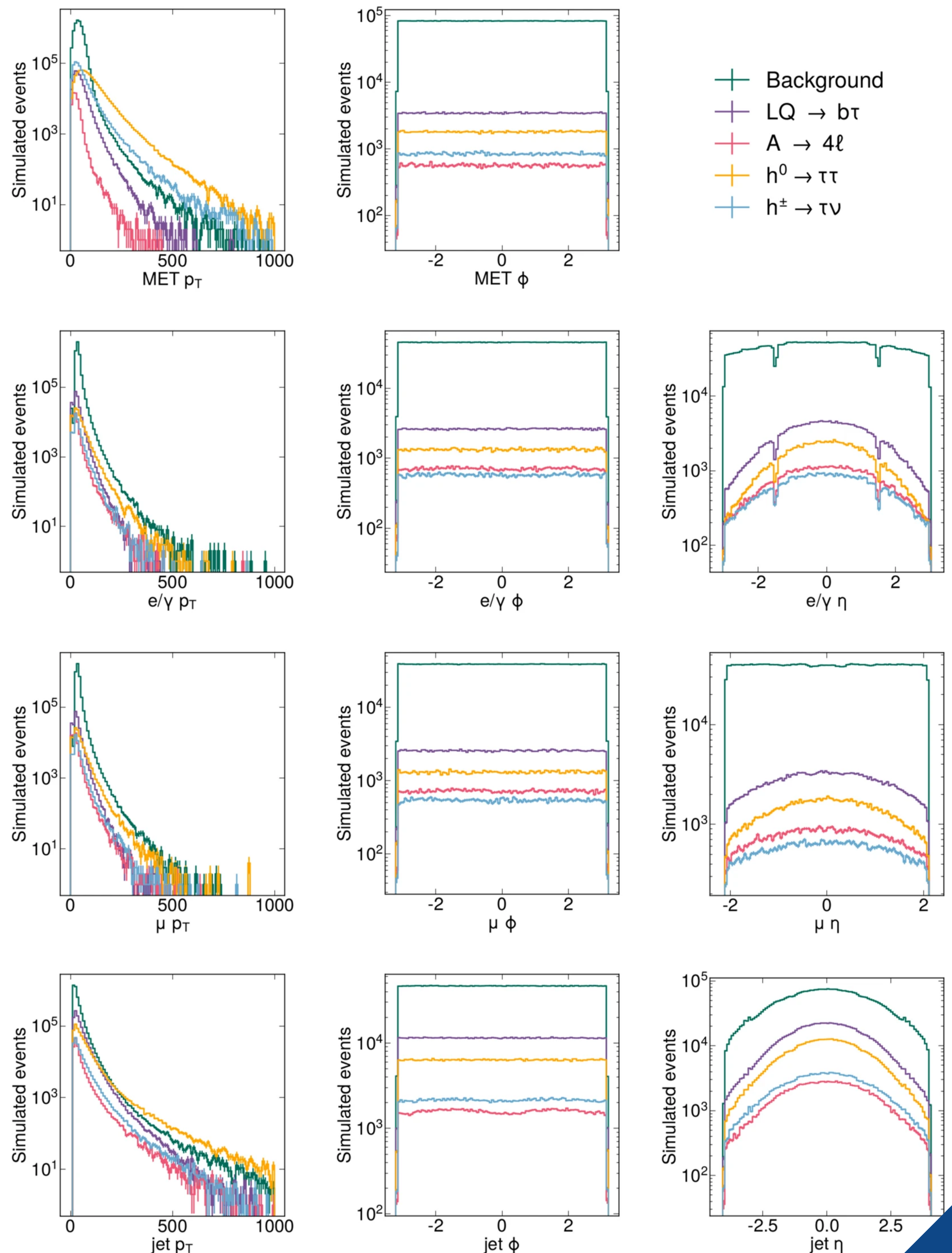
- Compare
  - vs. 3 kHz Run-2 ATLAS rate
- Better
  - 3x gain in signal



Compare with hls4ml

LHC anomaly detection ds [Sci Data 9, 118]

- Background
  - W → lν, Z → ll, 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
  - (L1 will already save these...)

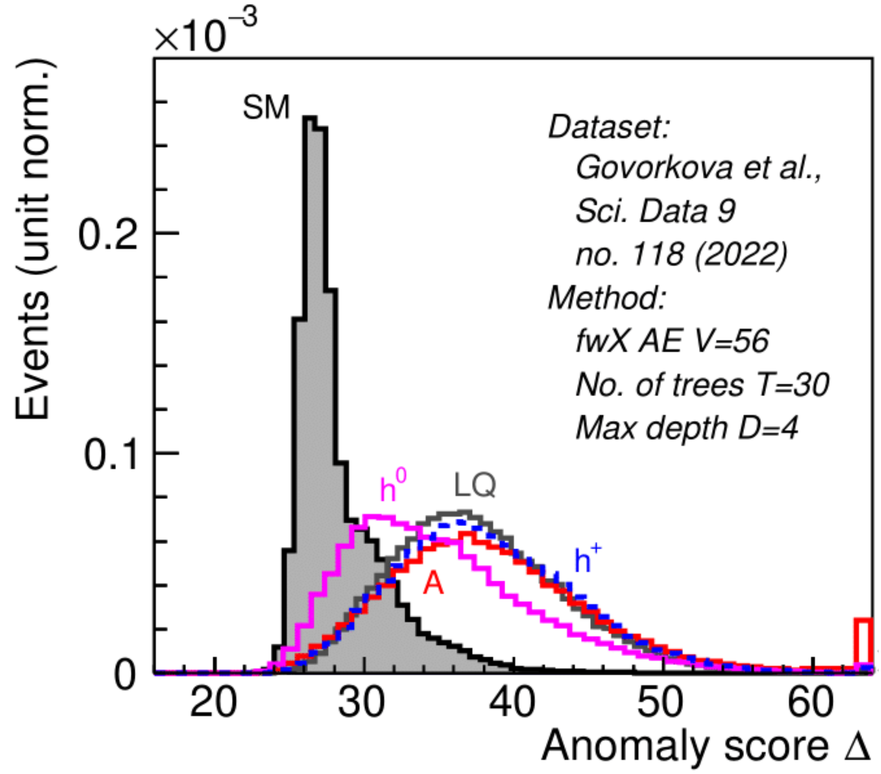


Compare with hls4ml

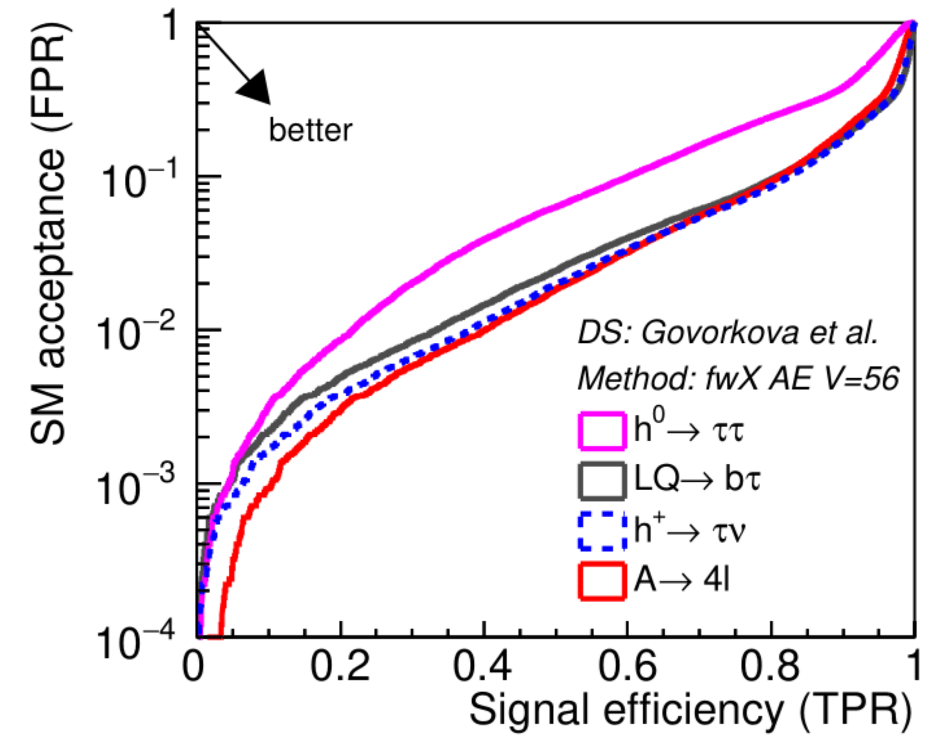
Works well

- Physics (plots)
- FPGA (table)

Distribution



ROC curve



Comparison

- Hls4ml NN-AE
  [Nature Mach. Intell. 4 (2022) 154–161]
- Physics: comparable AUC
- FPGA results

	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

Summary

Decision tree-based autoencoder

- New training method by sampling, it's density estimation
- More transparent (to me) than neural network-based designs
- Can do problems in high energy physics (3 - 50 variables)
- Competitive performance vs. hls4ml

Efficient implementation

- Latent space-less design where encoding = decoding
- Performance on Xilinx Virtex Ultrascale+ VU9P
  - O(1)% level resource usage
  - Fast at 30 ns latency
  - Try it yourself with the provided testbench & IP available online

Thoughts

Then what

- What are we going to do with the events that we save?
  - Everyone is saving rare events that are uncategorized. Who's going to categorize them? CMS recently showed an event display of the most anomalous event. Will we go through one-by-one to try to guess at the physics?
  - There are ideas, but more needed

What about benchmarks?

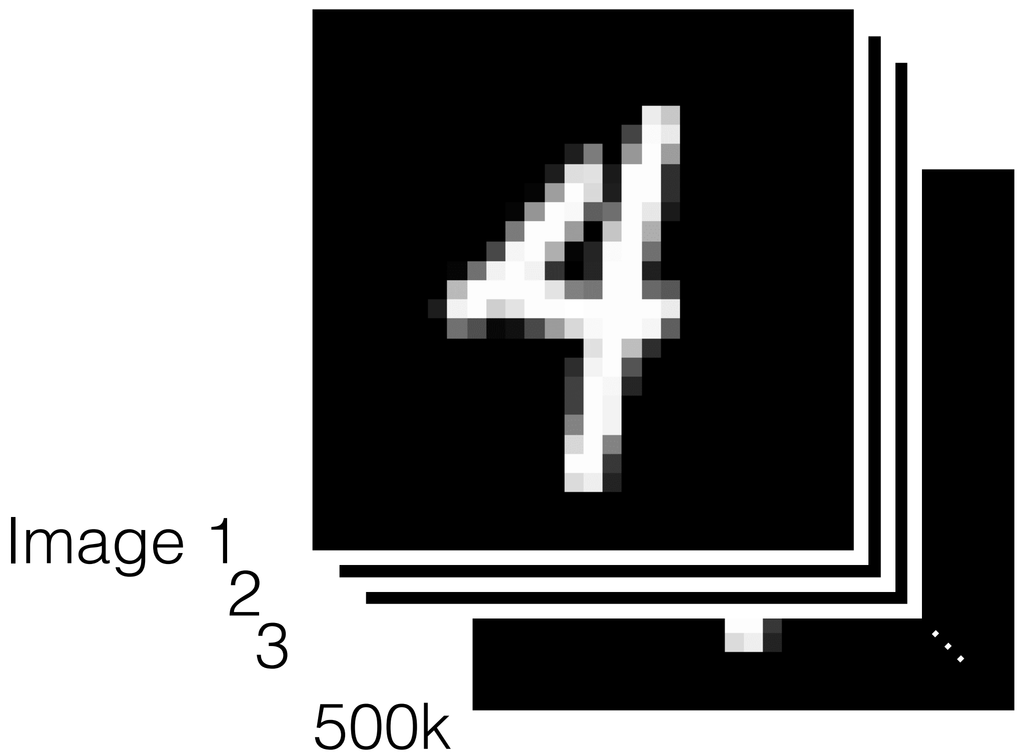
- By construction, it's supposed to pick up events that we don't know about. But to benchmark it, we choose models that we know about. Is this a contradiction? How do we avoid it? Who gets to choose?
- How much trigger bandwidth do we devote to it if we don't know what may be in it?

Backup

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

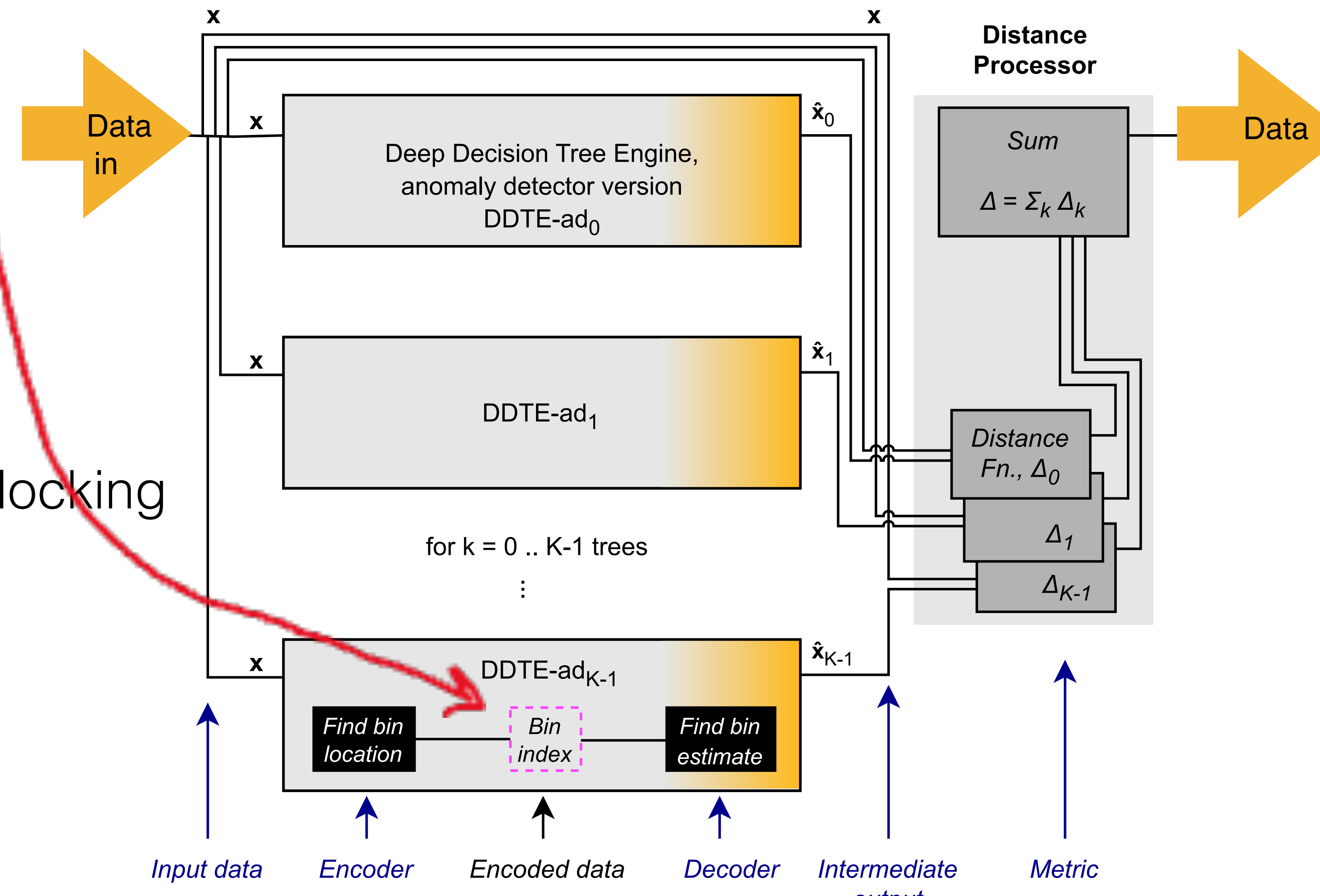
- Each pixel in the data set are unrelated to each other

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 → 8

- **Improve FWXv1**

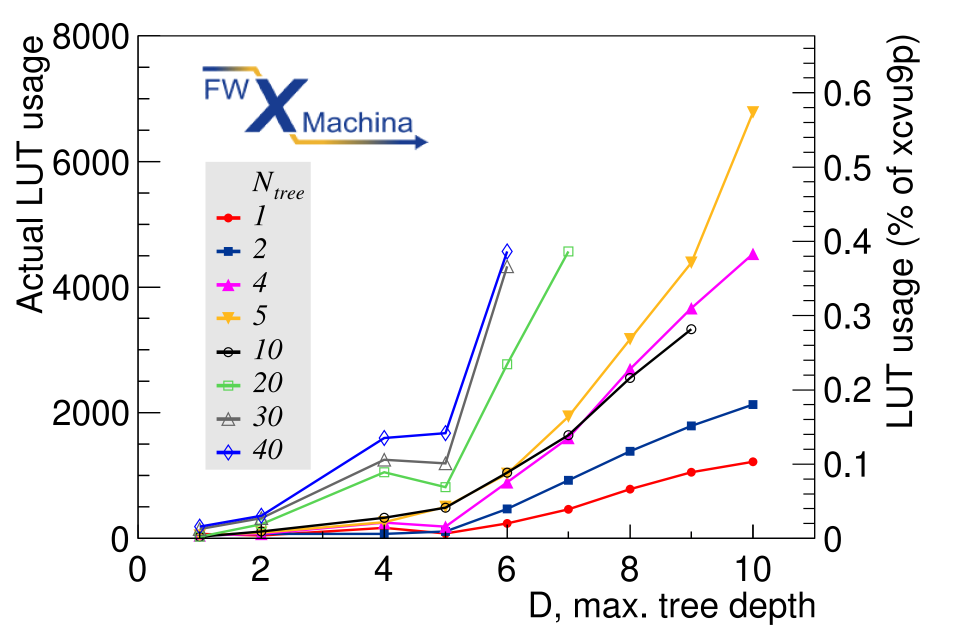
  Challenge: Does not scale well w/ tree depth & # variables

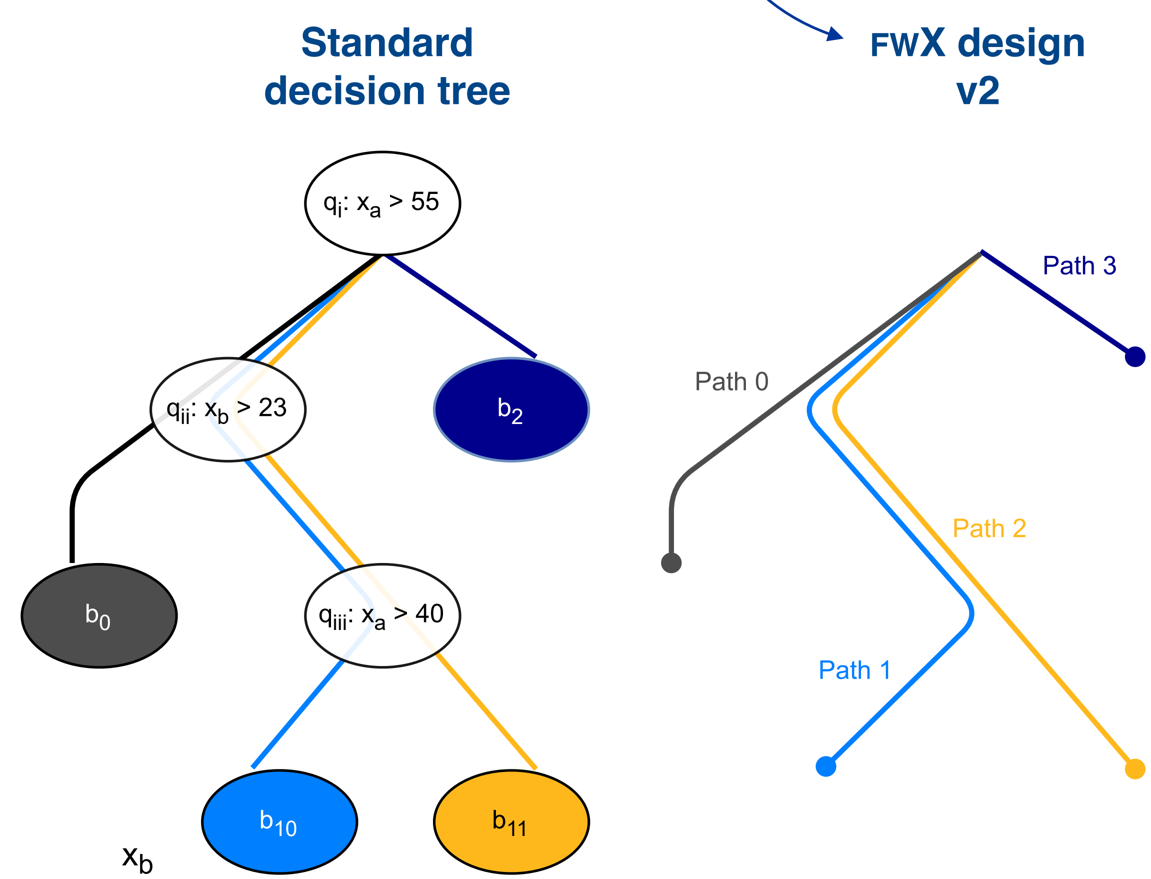
  Cut redundancy 2^D

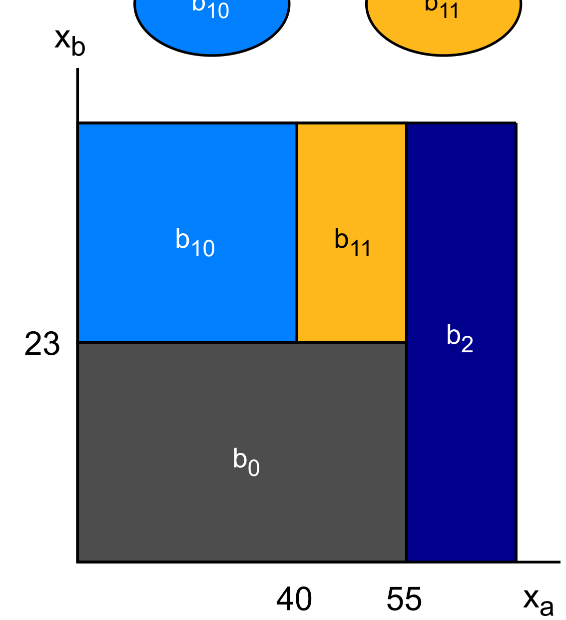
- **FWXv2**

  Key design: Evaluate decision paths

  Benefit: Softer scaling vs 2^D







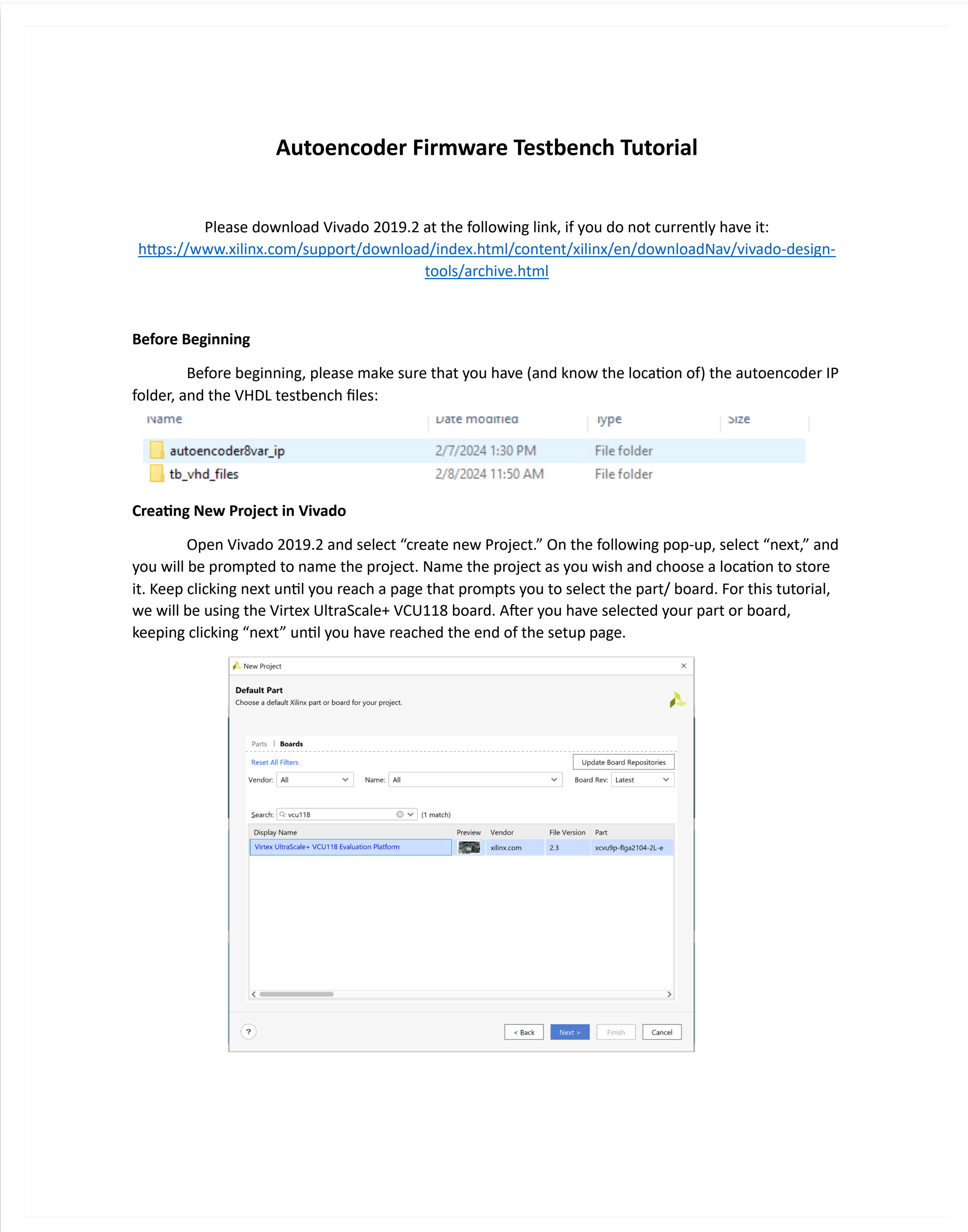
Destination bin	Decision path
b_0	not(q_i) and not(q_{ii})
b_2	q_i
b_{10}	not(q_i) and q_{ii} and not(q_{iii})
b_{11}	not(q_i) and q_{ii} and q_{iii}

Carlson et al., JINST 17, P09039 (2022)

FW testbench w/ IP available

TM Hong

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



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.

Screenshots in the document