

# FWXMACHINA

Nanosecond machine learning with boosted decision trees  
for high energy physics



University of  
Pittsburgh



Tae Min Hong

- *Paper* [JINST 16 P08016 \(2021\)](#)
- *Info* <http://fwx.pitt.edu>
- *Code* <http://gitlab.com/PittHongGroup/fwX>

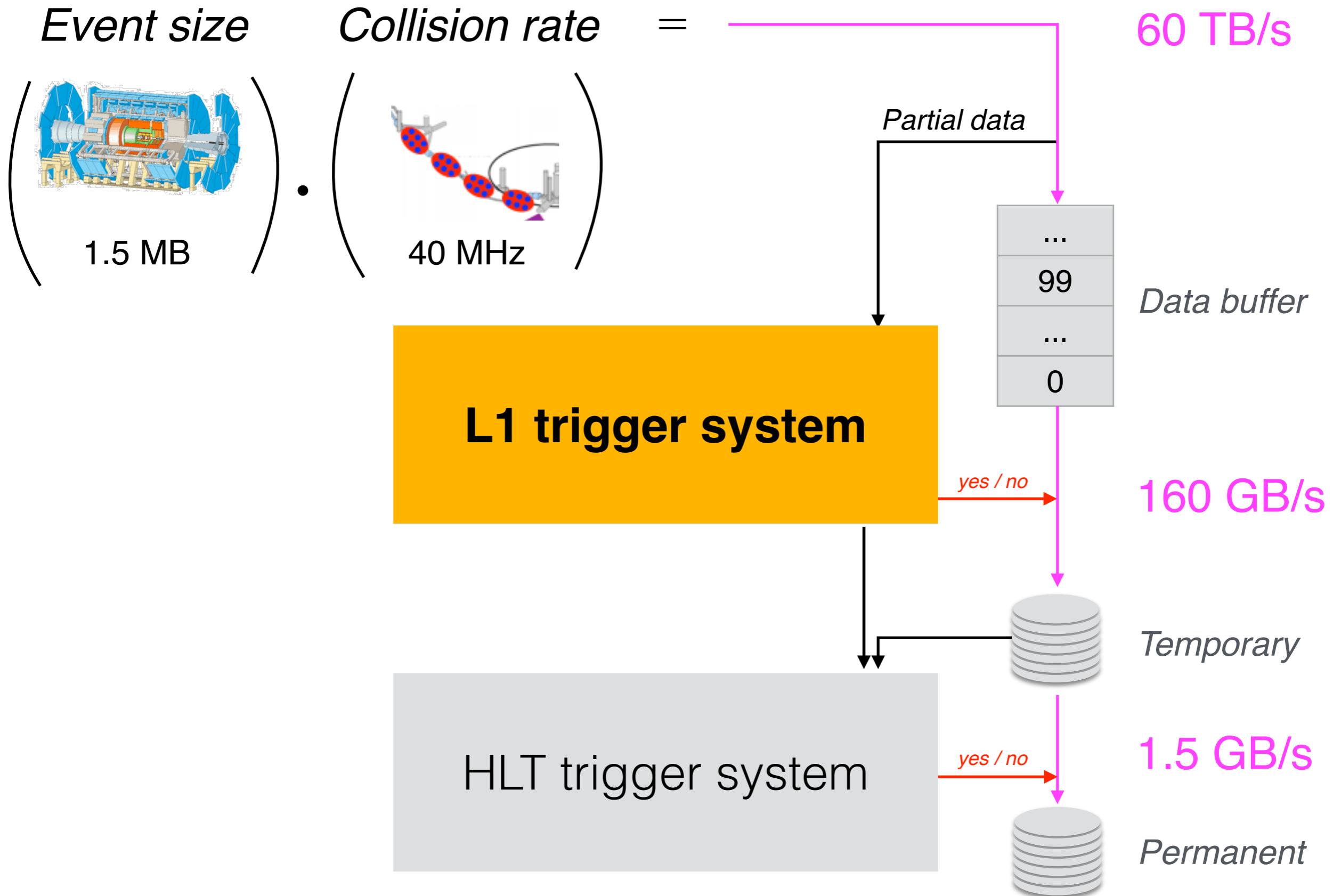
PIKIMO 2021

December 4, 2021

<https://indico.cern.ch/event/1091676/>

# Machine learning at L1 trigger

TM Hong





## BDT design

- Algorithm structure
- Firmware design

## Results

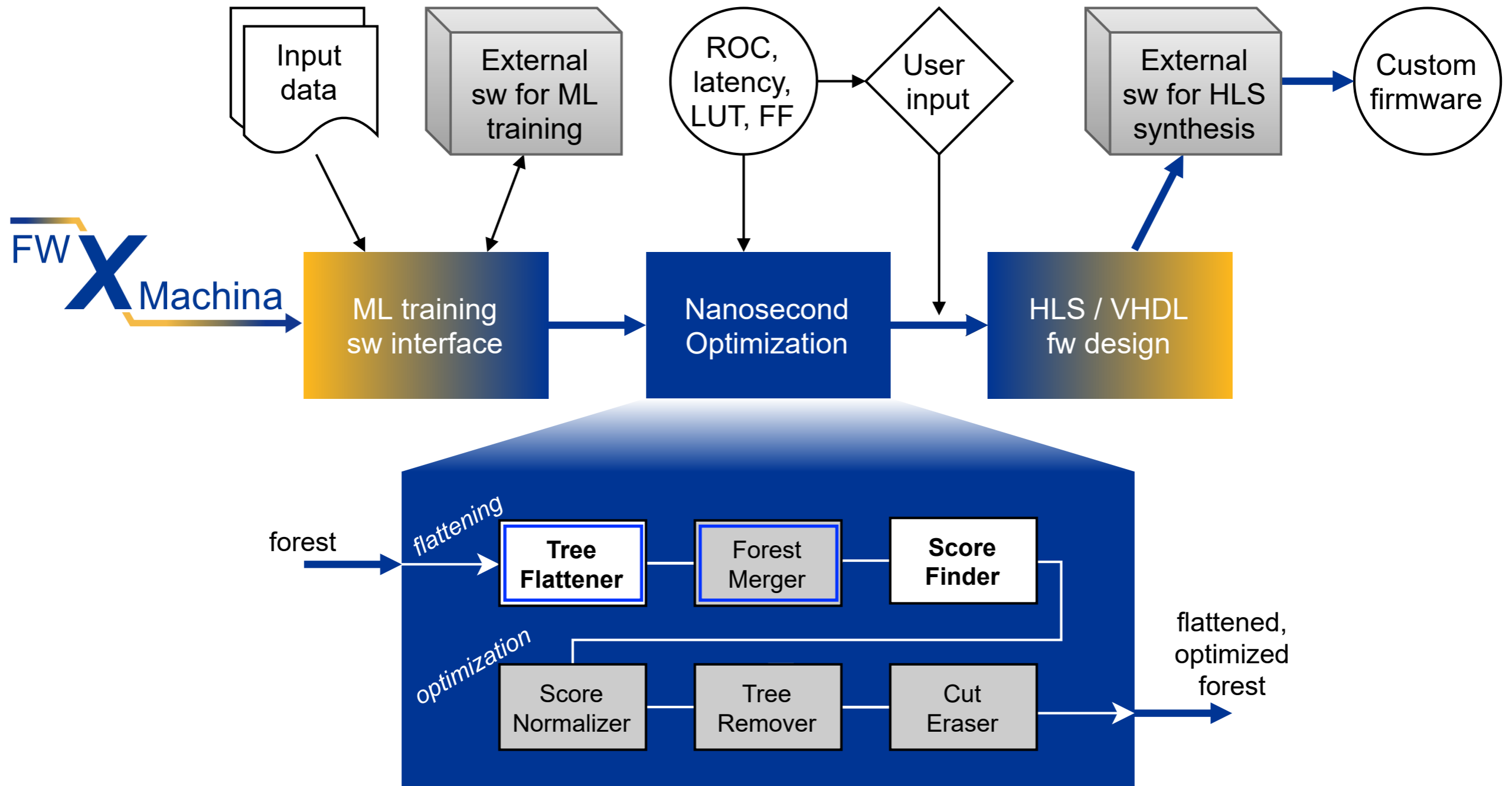
- VBF Higgs vs. multijet
- (Electrons vs. photons)

## Comparisons

- vs. hls4ml's BDT
- vs. hls4ml's neural network

[JINST 16 P08016 \(2021\)](#)

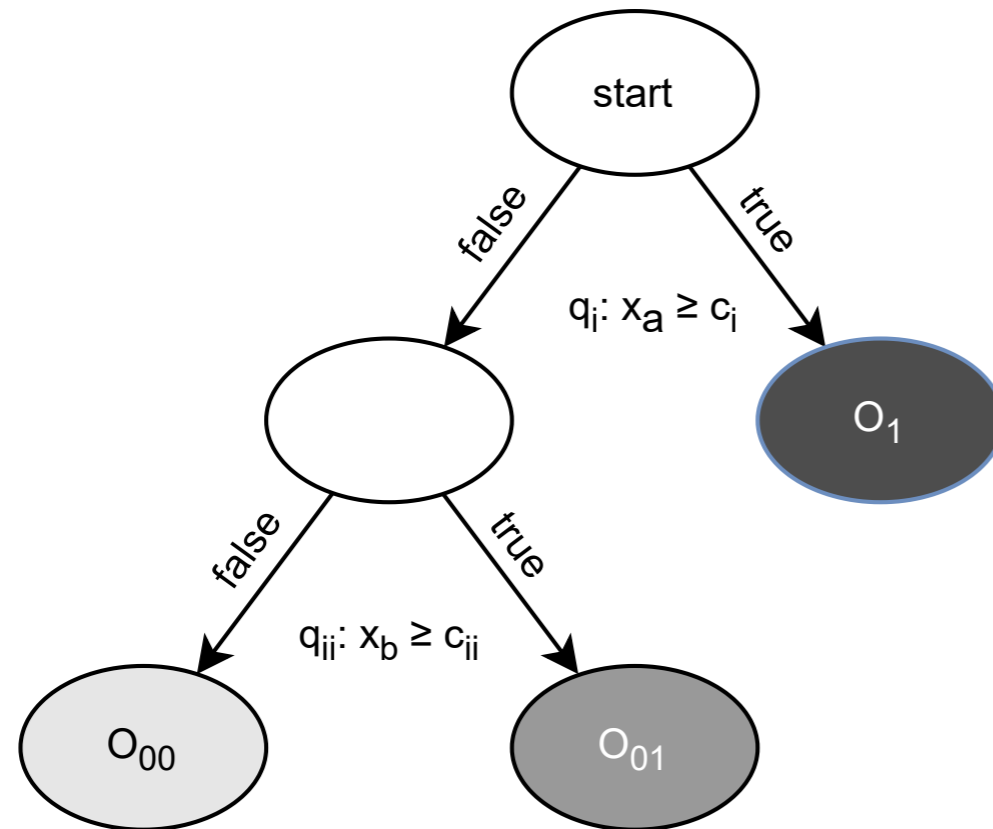
Not in paper



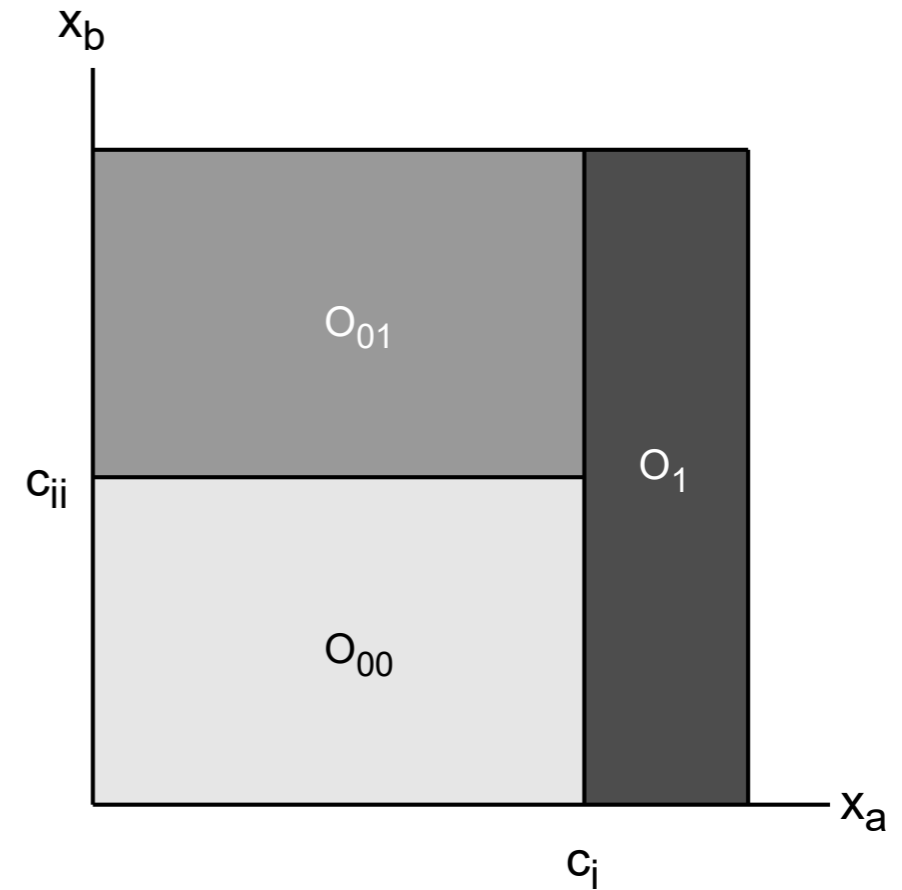
- I will focus on the first two steps



Conventional tree structure



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

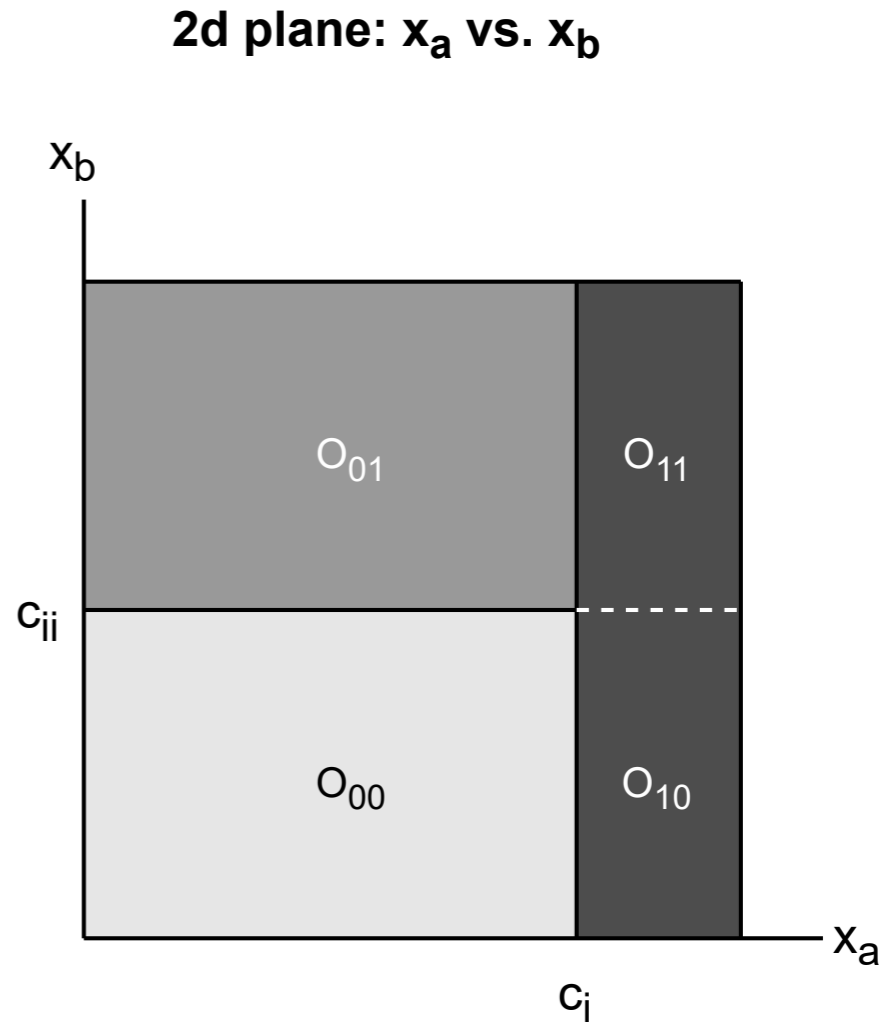
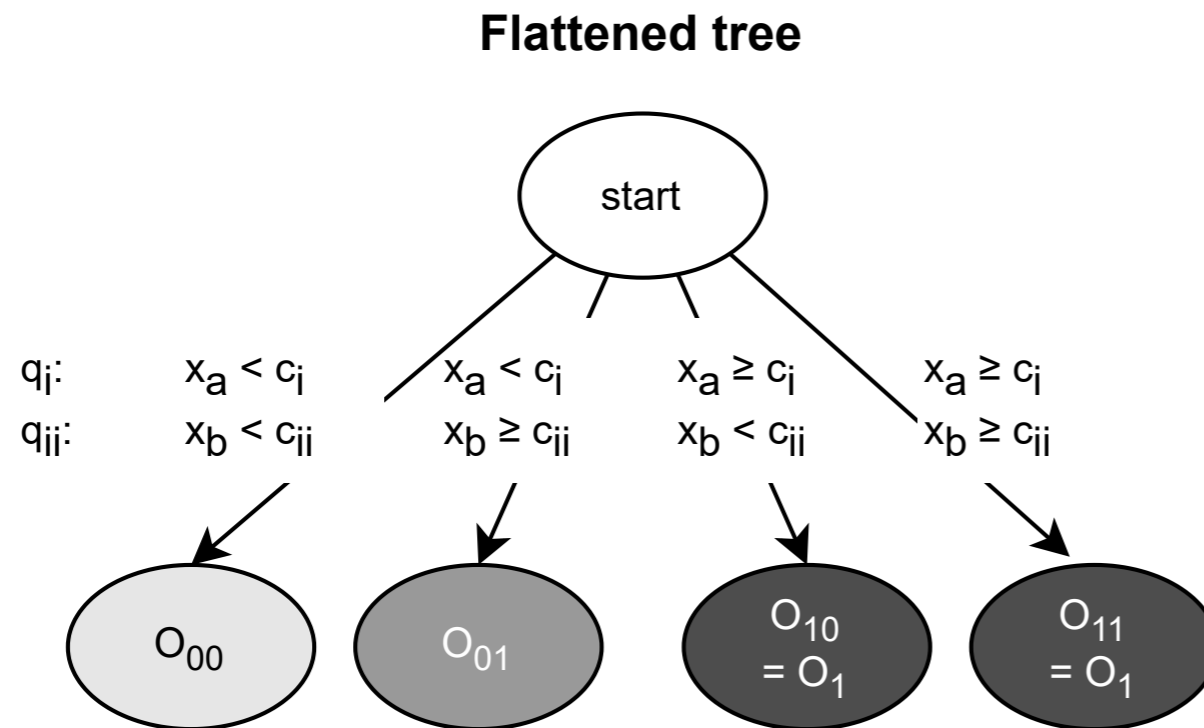


## Normal decision tree

- Recursive in the number of depths

## Firmware

- **Not** our design

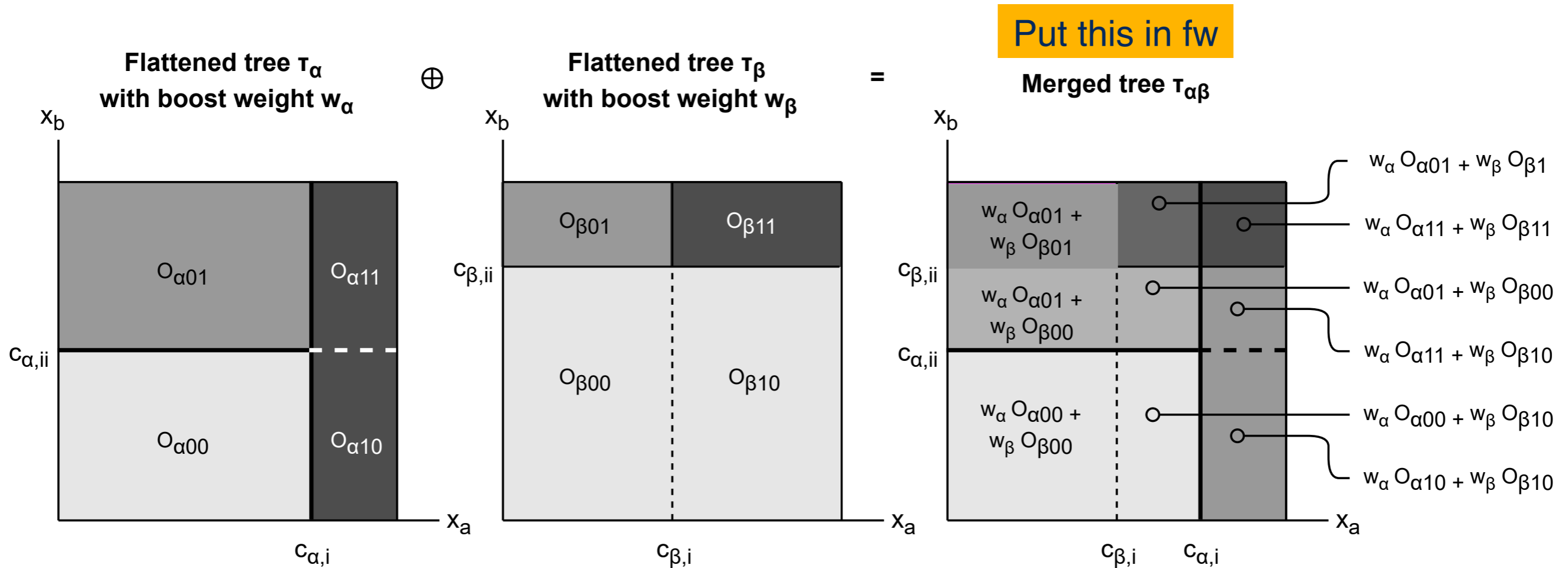


## Flattened decision tree

- Axes are independent  $\rightarrow$  Bin search problem on a grid

## Firmware

- Our design



## Pre-merging trees

- Pre-processed in software before implementation in firmware
- No impact on physics performance

## Firmware

- Our look-up table design



## Same production, two decays

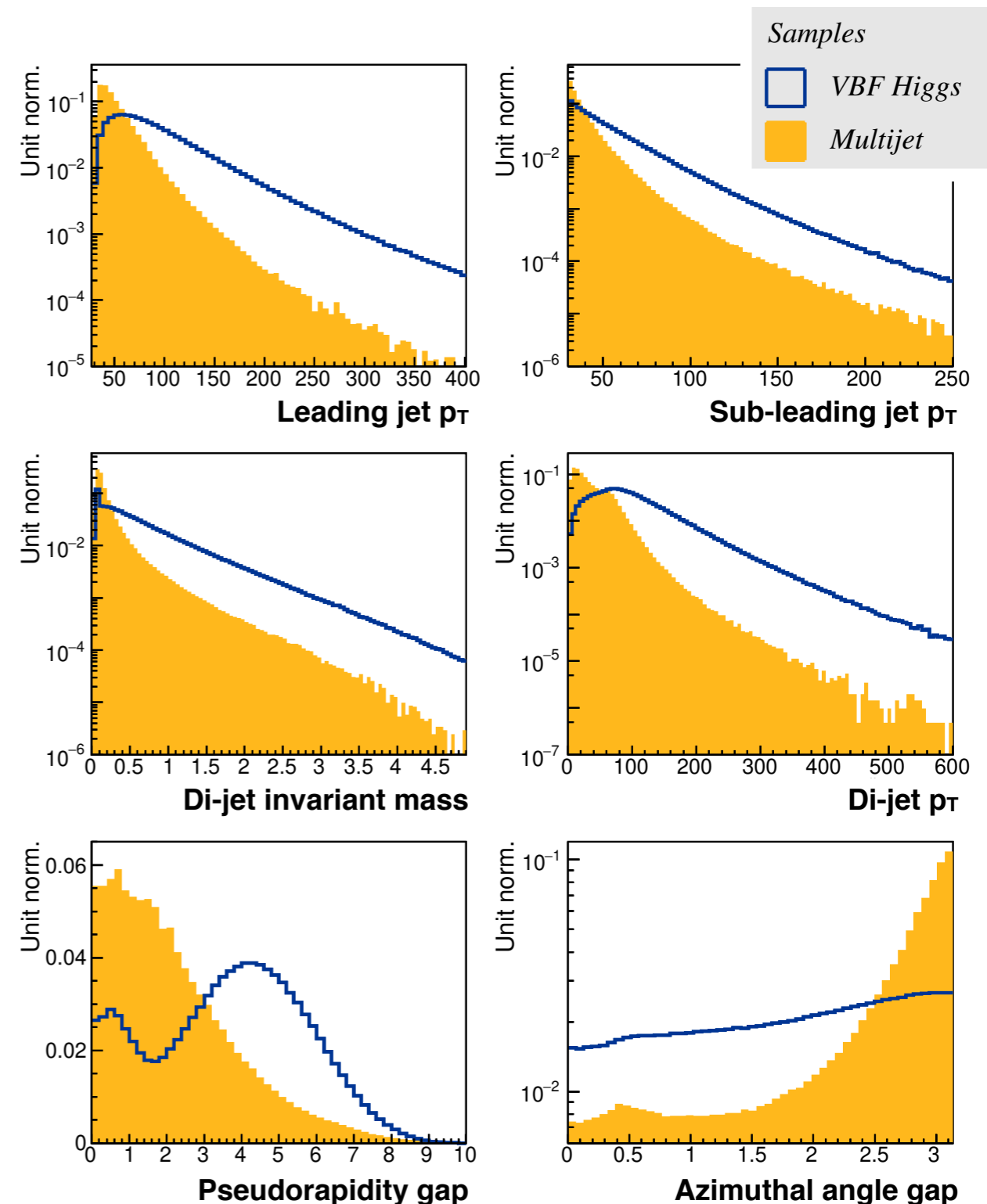
- $H \rightarrow \nu\bar{\nu}\nu\bar{\nu}$ , "invisible"
- $H \rightarrow b\bar{b}b\bar{b}$ , thru pseudoscalars

## Strategy

- Train on **Multijet** vs. **VBF H**  $\rightarrow \nu\bar{\nu}\nu\bar{\nu}$
- Apply to **Multijet** vs. **VBF H**  $\rightarrow b\bar{b}b\bar{b}$

## Why

- Can trigger on VBF Higgs  $\rightarrow$  anything
- Does it work? Yes, next slide





# Results

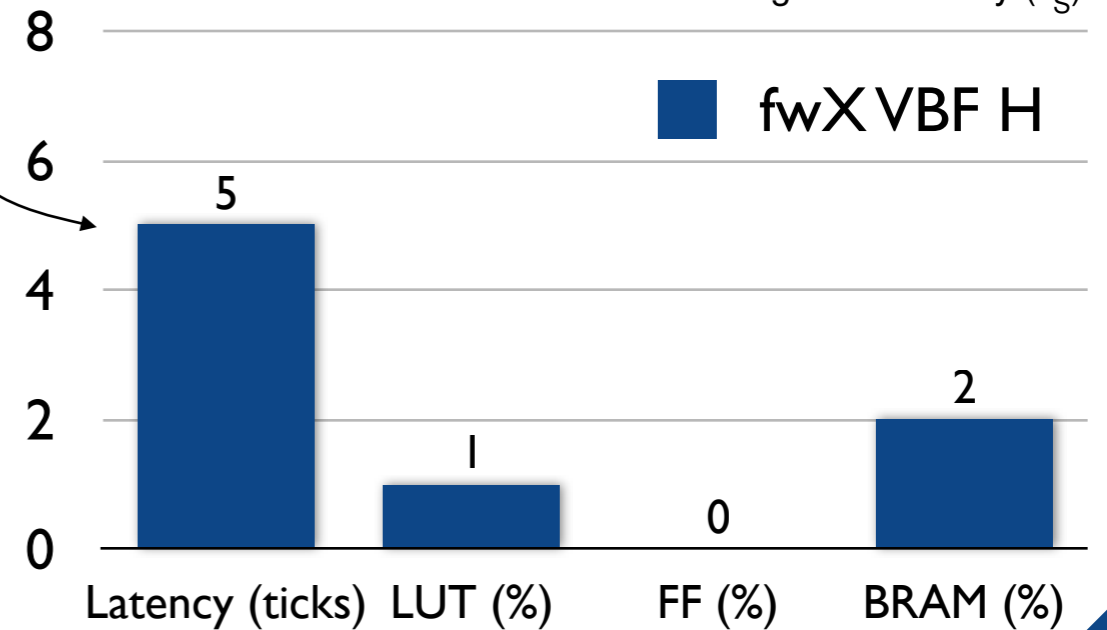
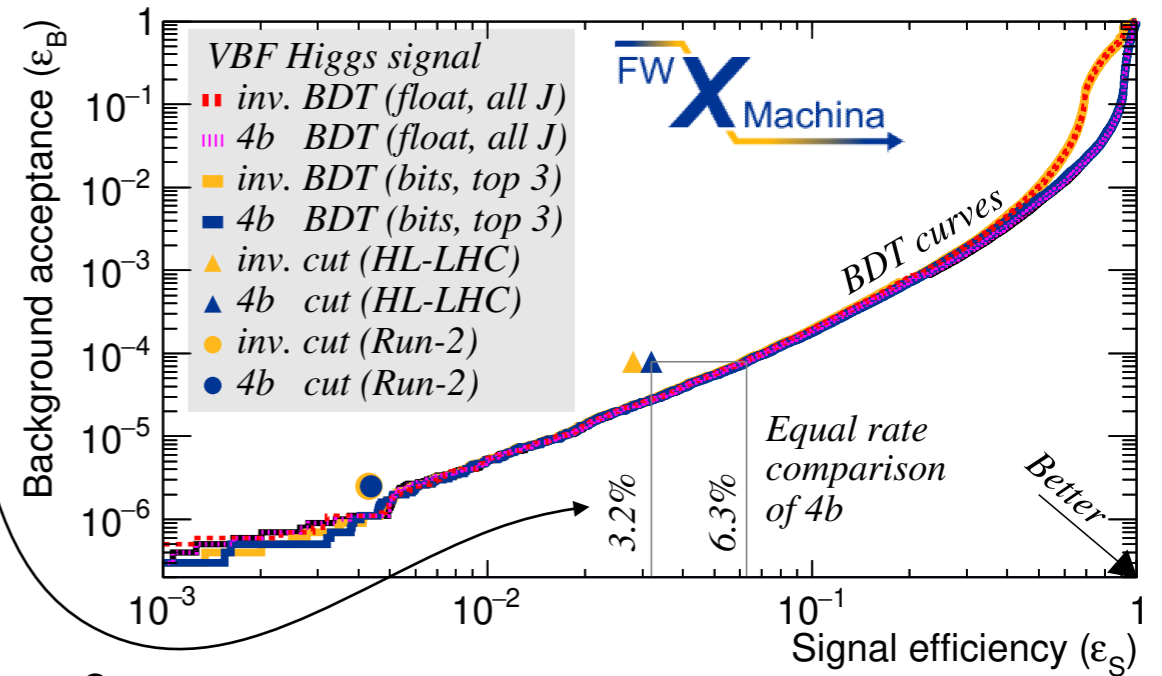
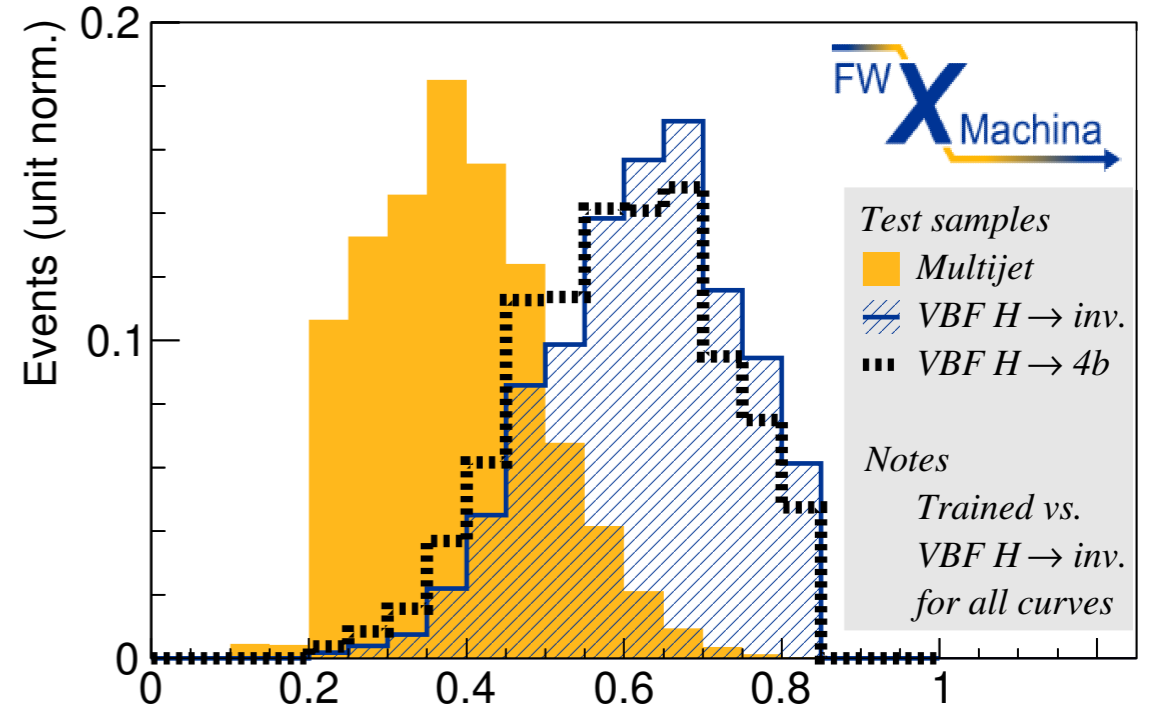
It works!

## Performance

- **Efficiency** : 2x better vs. HL-LHC ATLAS
- **Latency** : 16 ns = 5 clock ticks

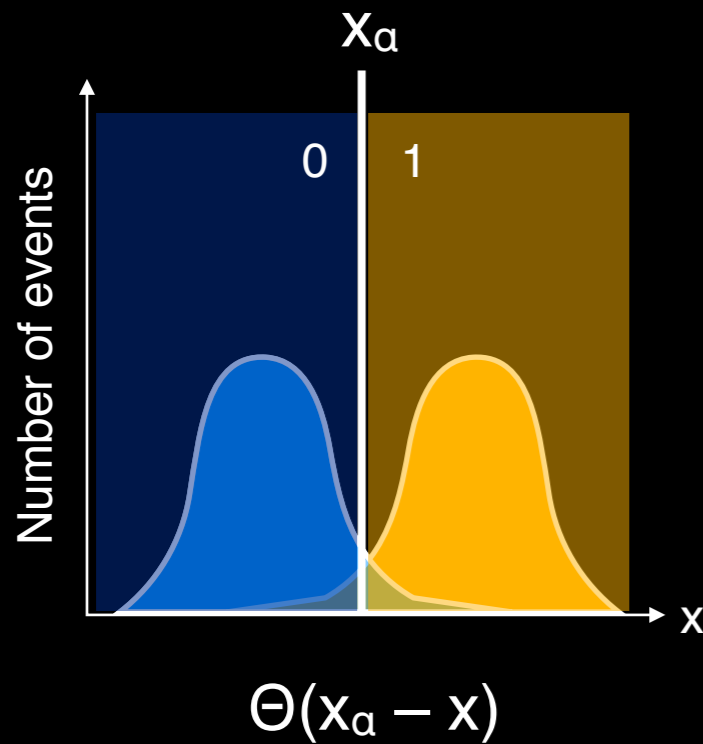
## Details

- **Validation** : Eff. matches ATLAS Run-2 paper

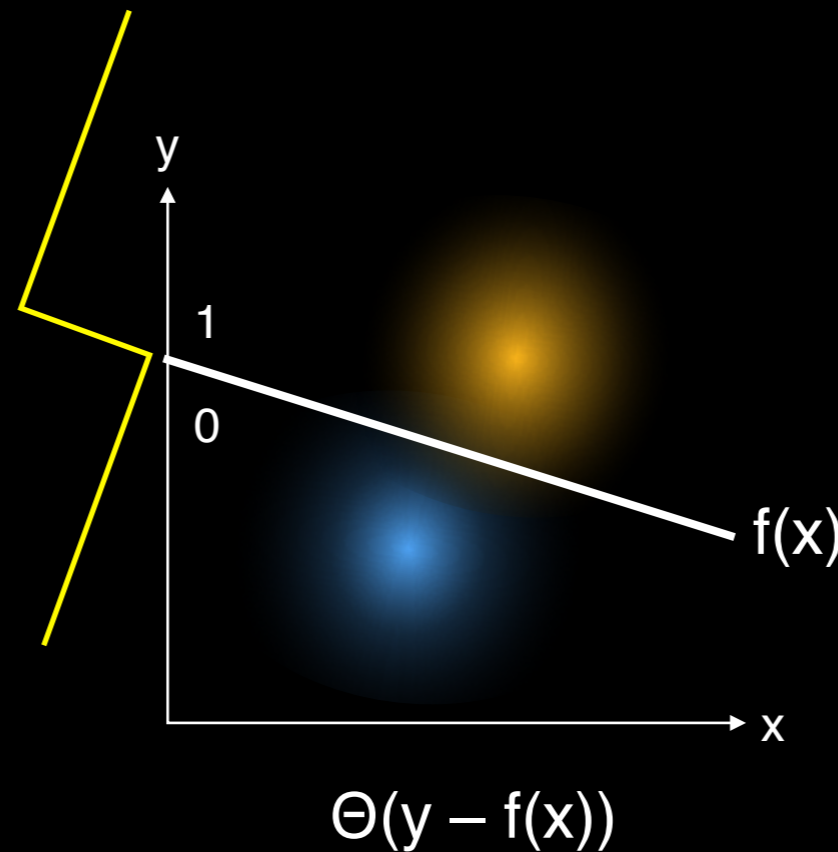


# BDT vs. neural networks

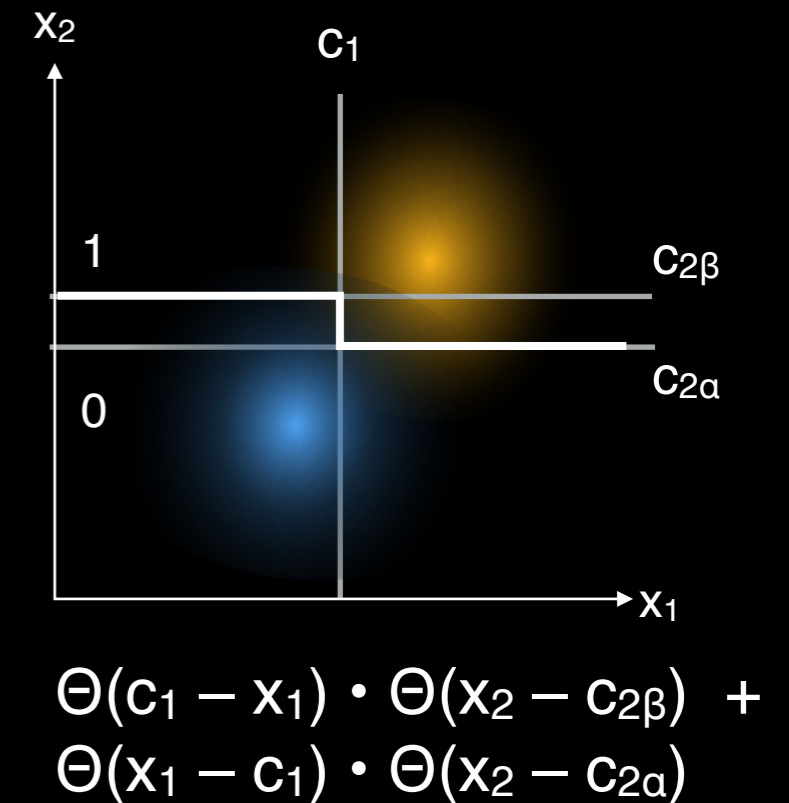
Step function for 1d



Neural network 2d



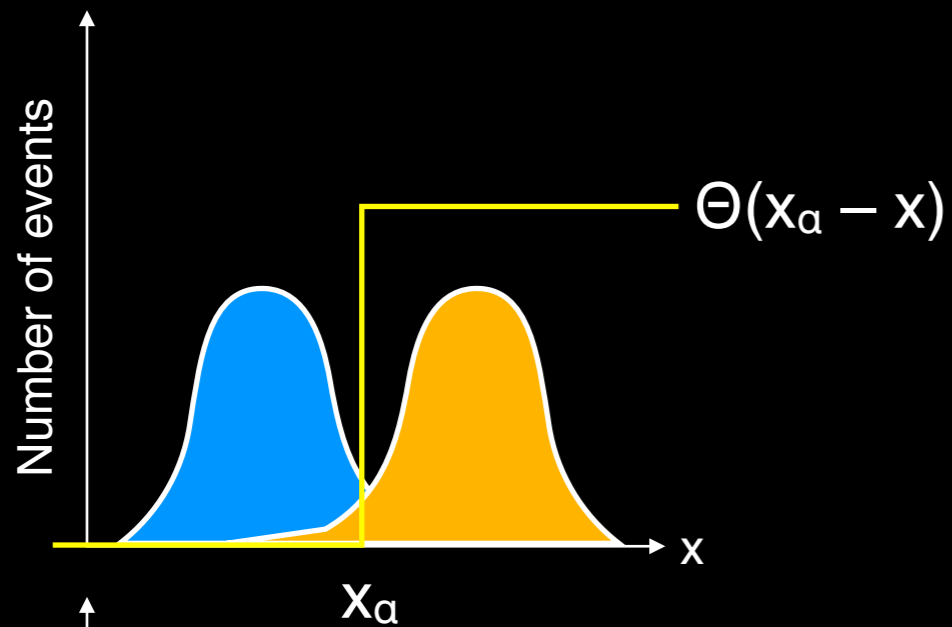
Decision tree 2d



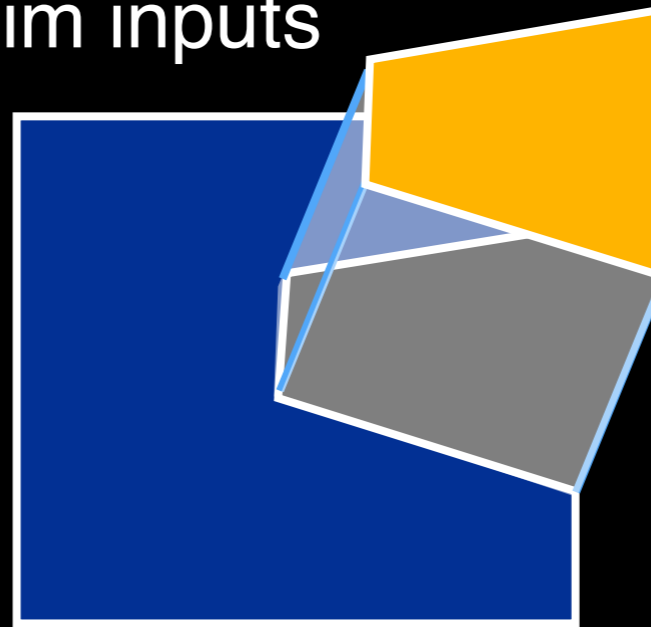
# Activation function for NN

Fuzzy boundary using a turn-on function

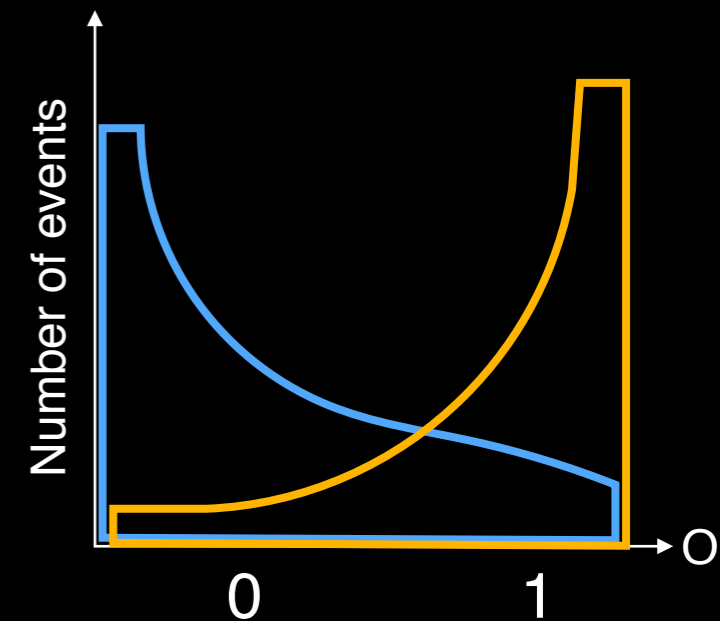
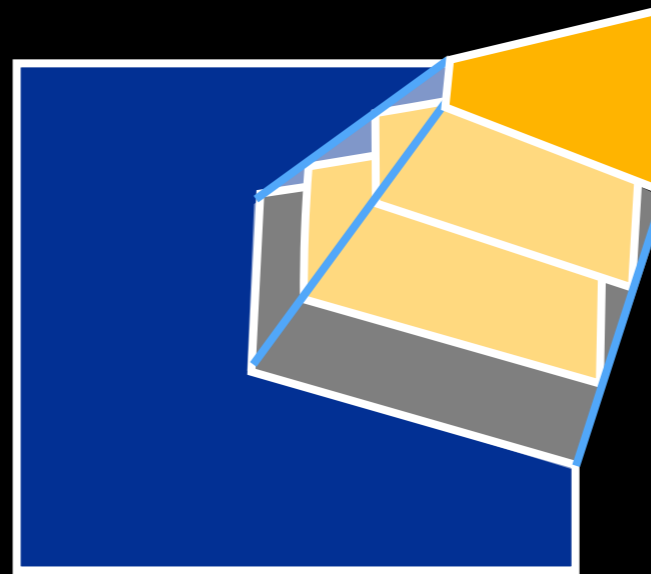
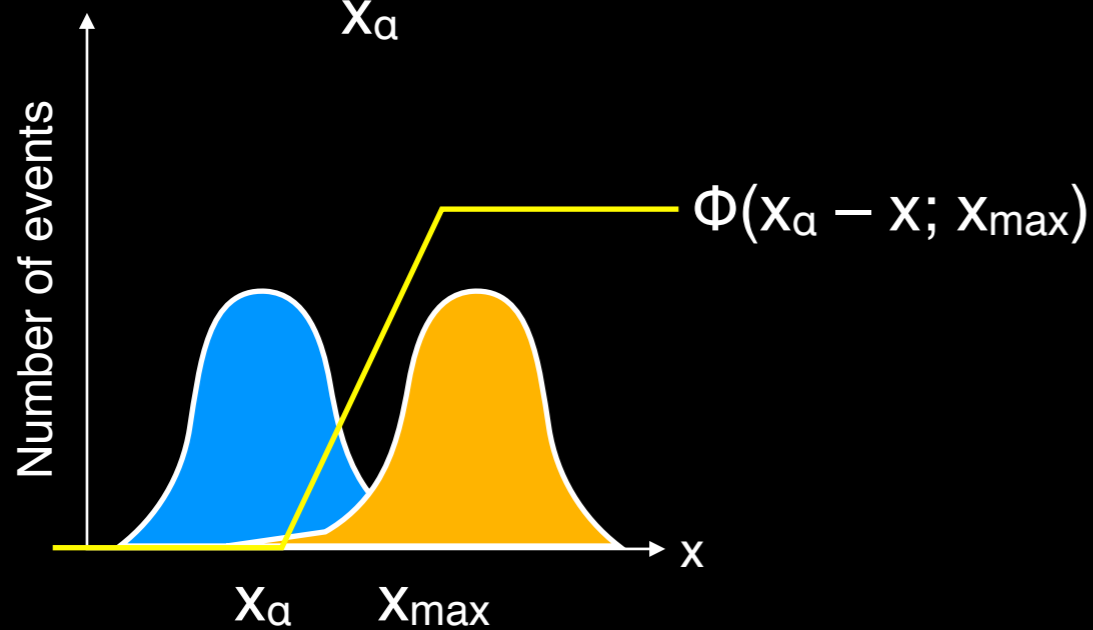
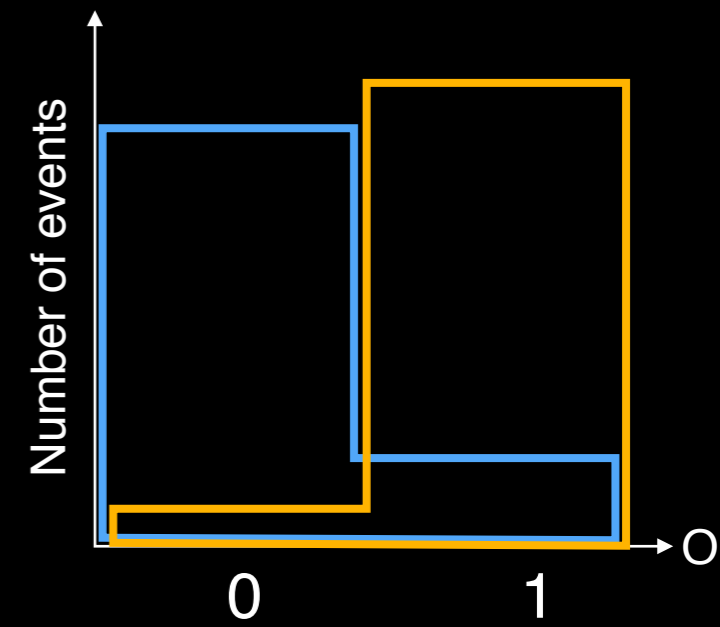
1-dim input



2-dim inputs

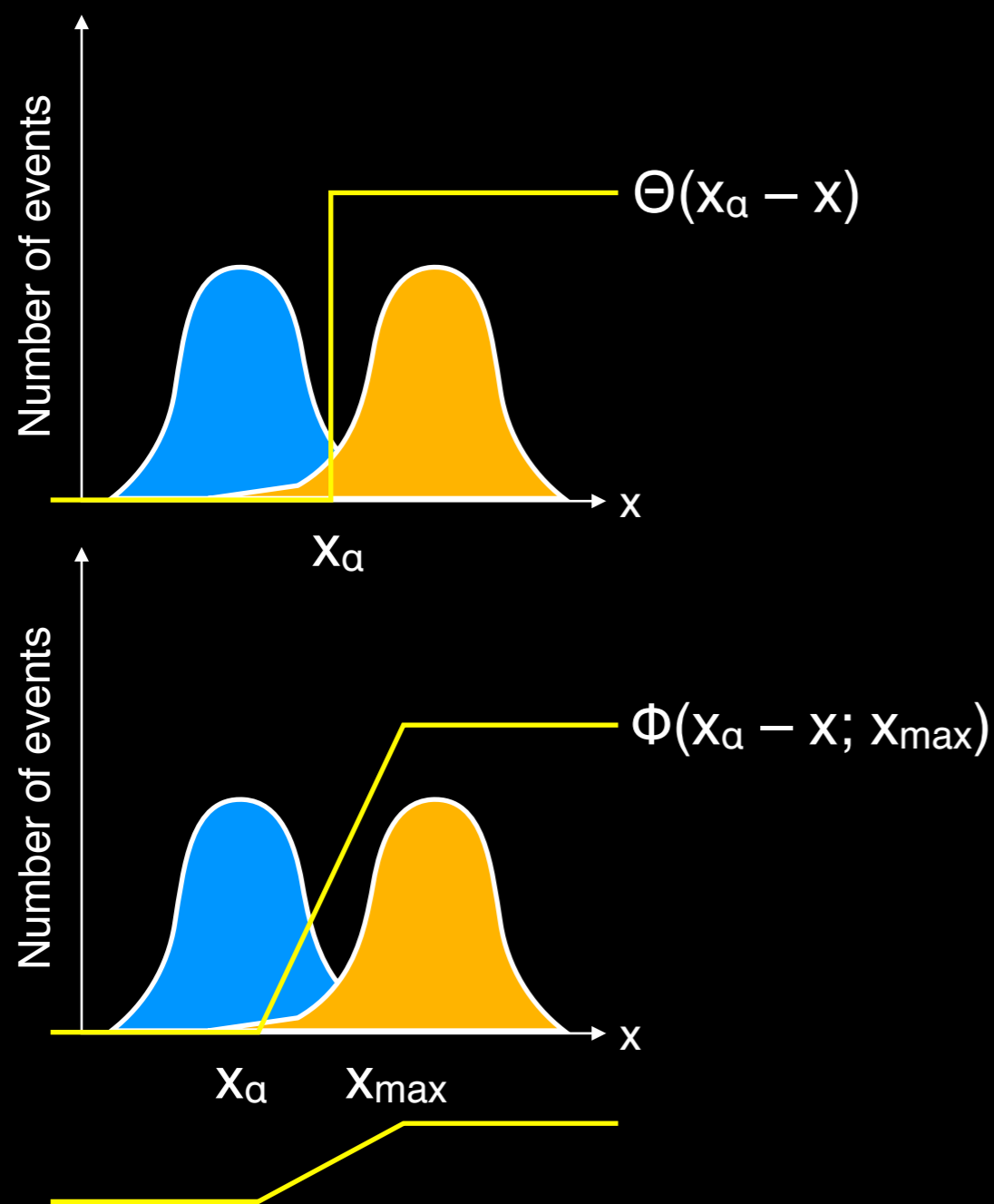


Output score

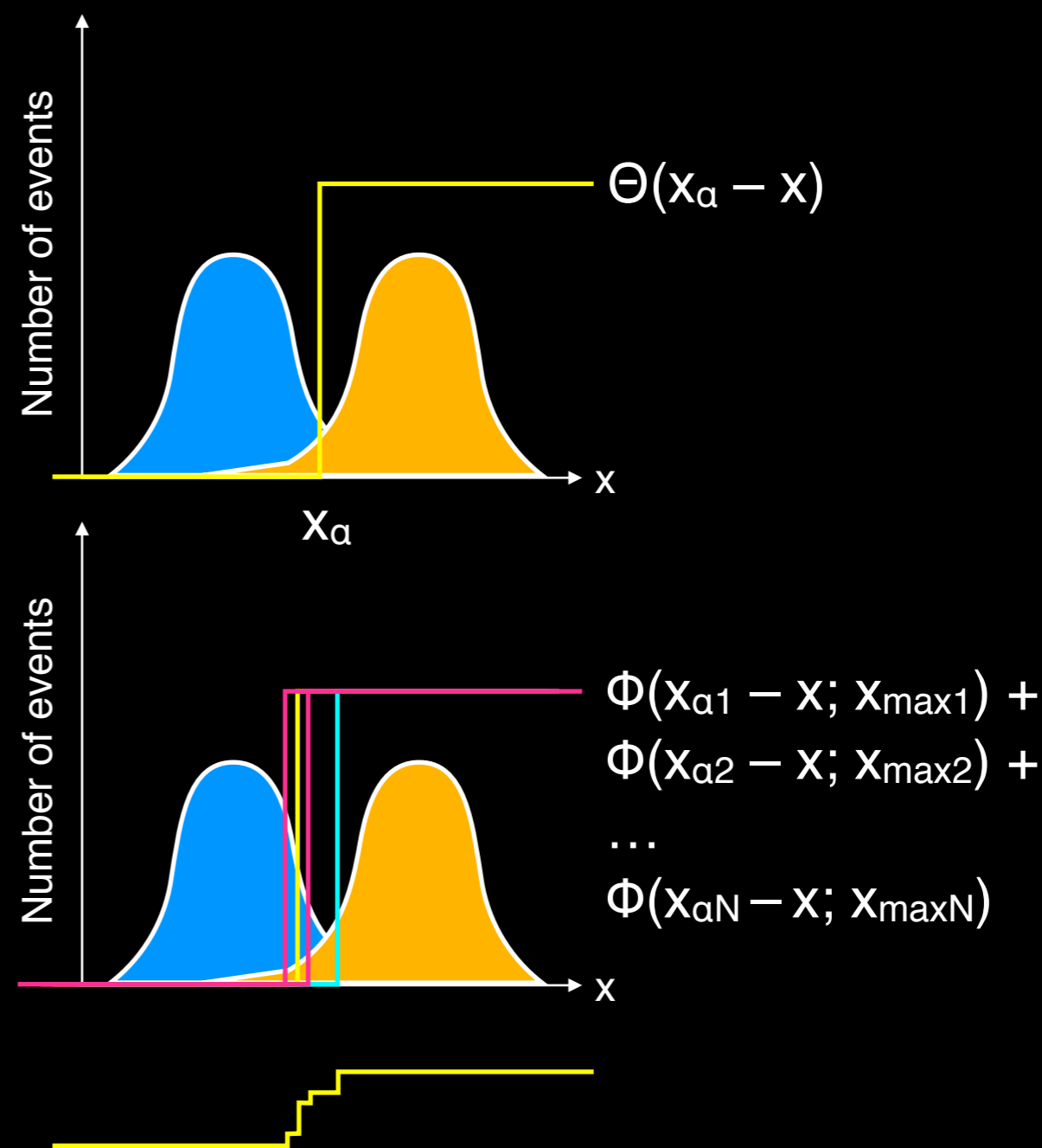


# Forest of decision trees

## Neural network 1d



## Boosted decision tree 1d



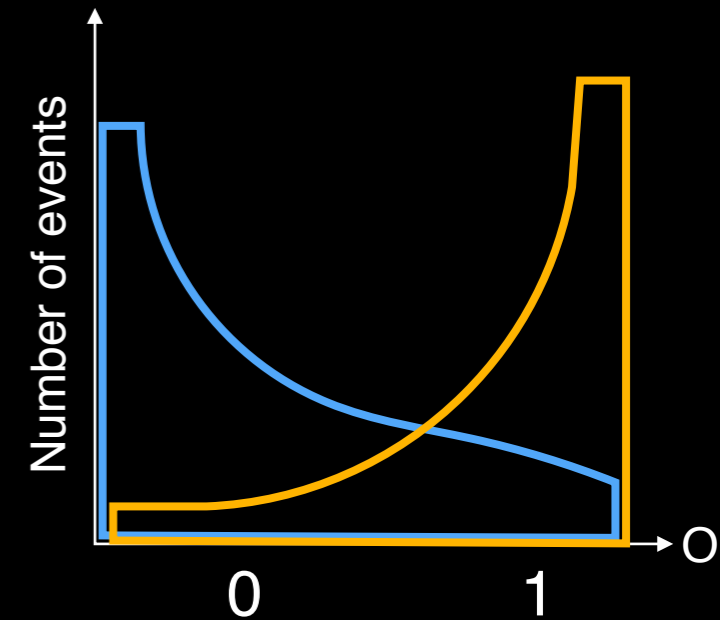
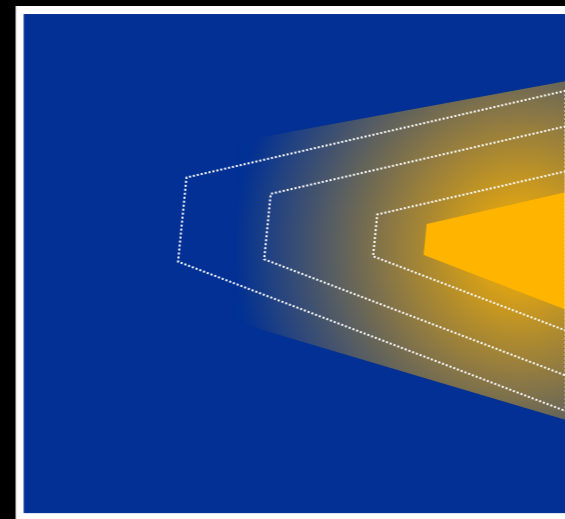
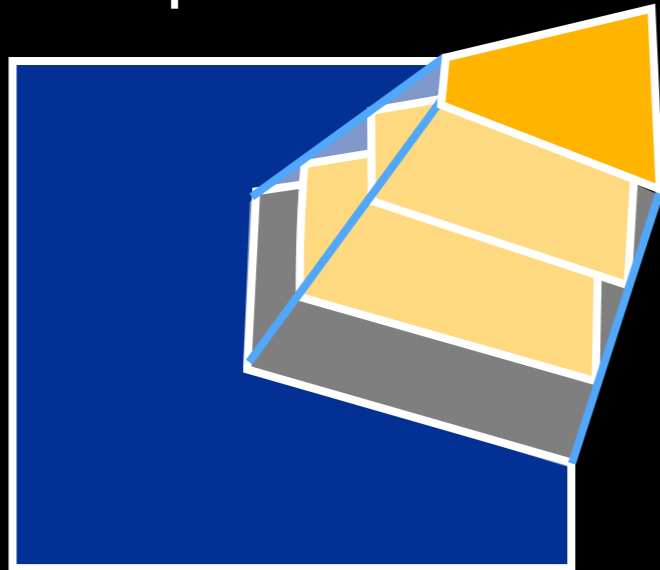
# Fuzzy boundaries

2-dim inputs

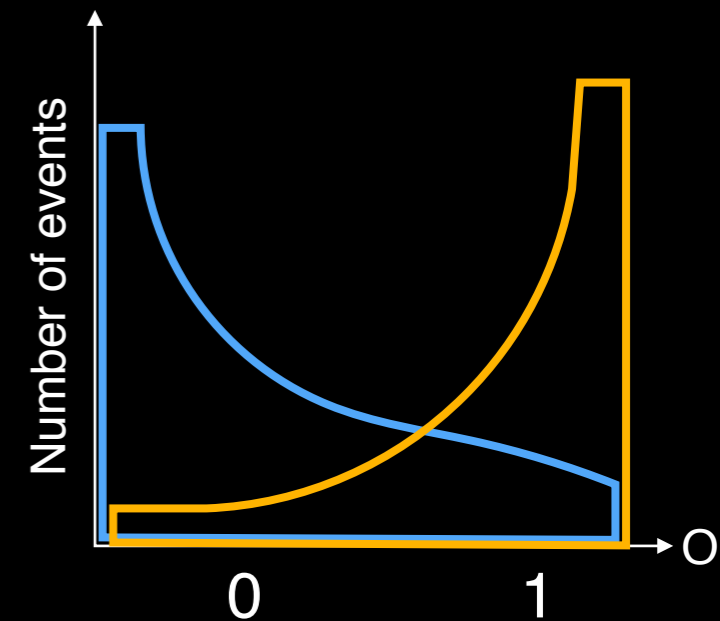
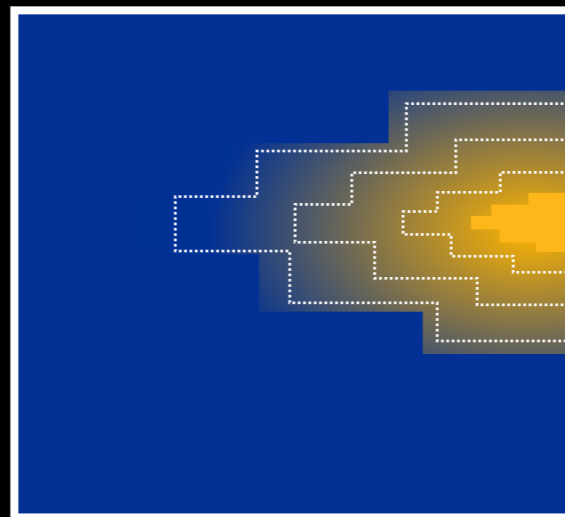
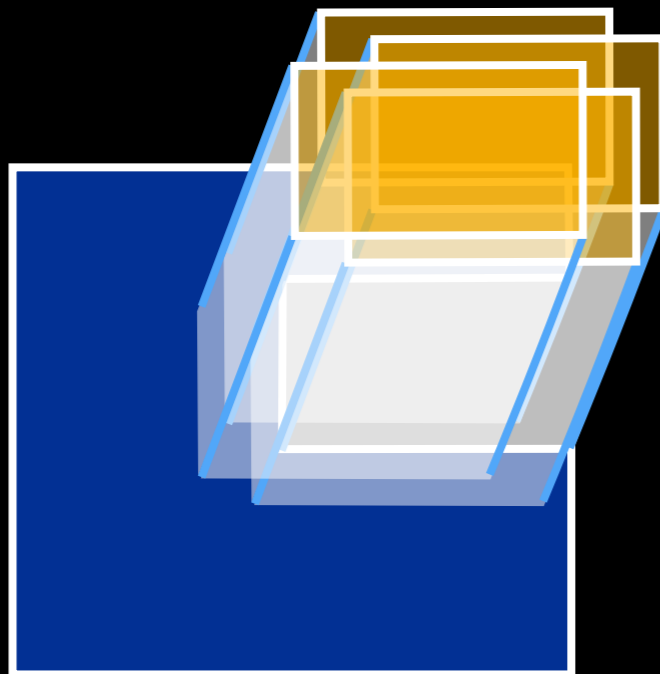
Projection

Output score

NN



BDT



# Data format

- We encode in N-bit integers

E.g., ap\_int<3> means 0 - 7 range

000  
001  
010  
011  
100  
101  
110  
111

- Advantages

Represent variety of precision, e.g.,  
 $p_T$  from GeV-TeV vs.  $\phi$  from  $0-2\pi$

- Subtleties

Transformation adds 1-bit ambiguity

$$c_{\text{int}} = f(c_{\text{float}}) = \left\lfloor \frac{c_{\text{float}} - c_{\text{min}}}{c_{\text{max}} - c_{\text{min}}} \cdot (2^N - 1) \right\rfloor$$



*Equal up to one bit because of floor*

↓

$$f(x_1 + x_2) = f(x_1) + f(x_2)$$



*Firmware adds the pre-evaluated f(x)*

- hls4ml encodes in fixed pt

E.g., ap\_fixed<5,2> means

00.000  
00.001  
00.002  
00.003  
00.004  
00.005  
00.006  
...

- Advantages

More familiar to physicists who use  
floating / fixed values

- Subtleties

Need to use "quantized aware training"  
to reduce the number of bits



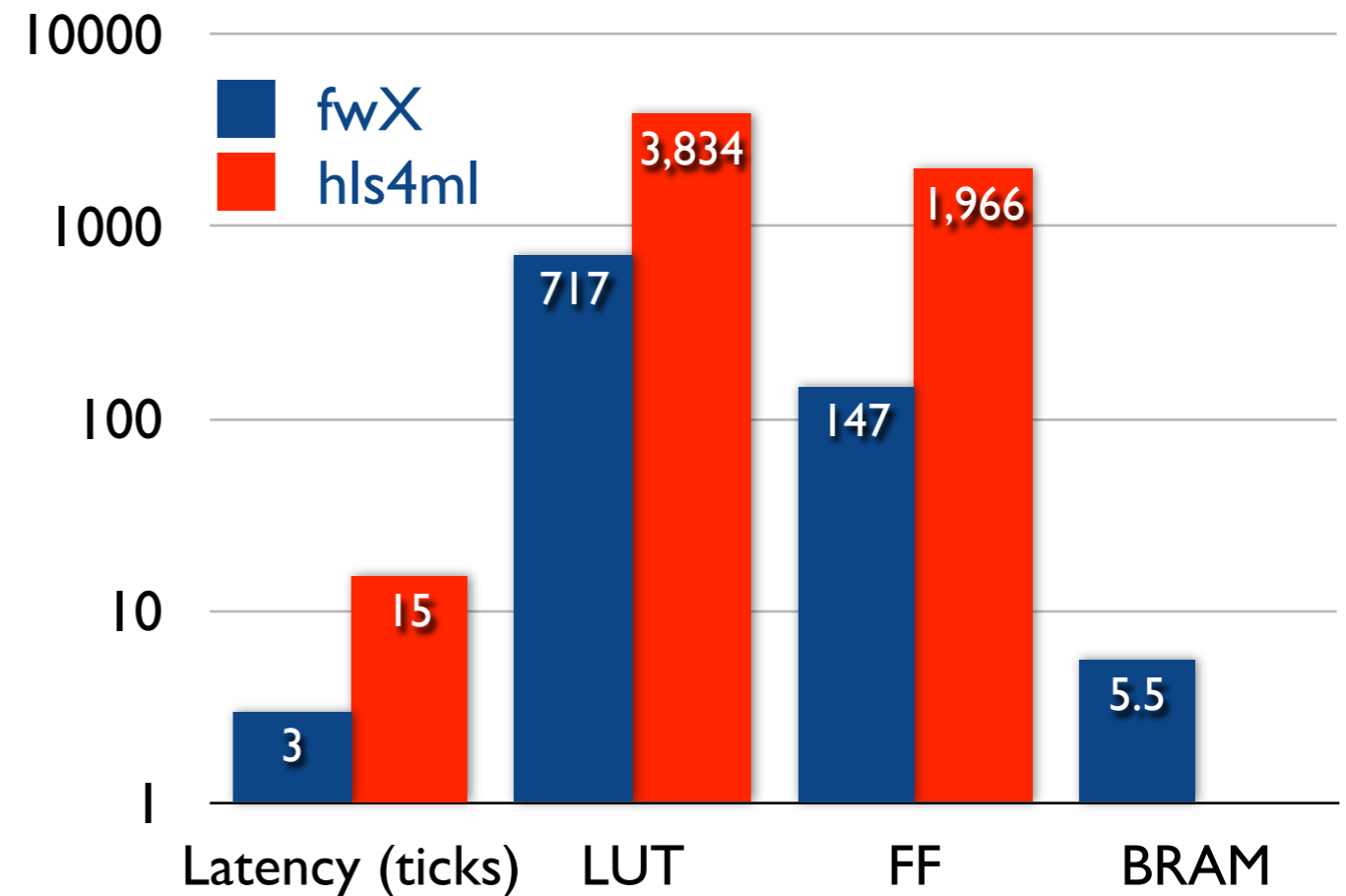
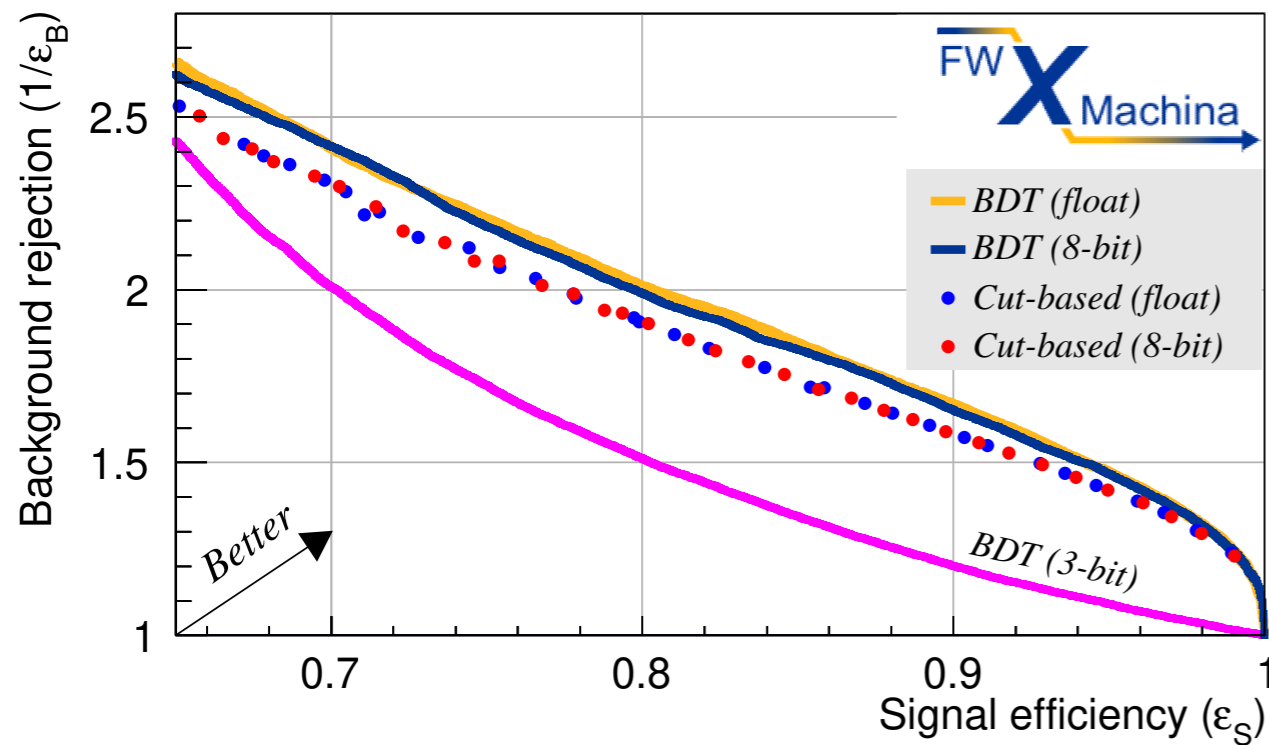
# fwX BDT vs. hls4ml BDT

## Setup

- Details in paper, same as possible
- Public datasets of  $e$  vs.  $\gamma$
- ROC is same bec. use same BDT

## Comparison

- Lower latency
- Lower LUT, FF
- Higher BRAM (but 0.1% of avail.)



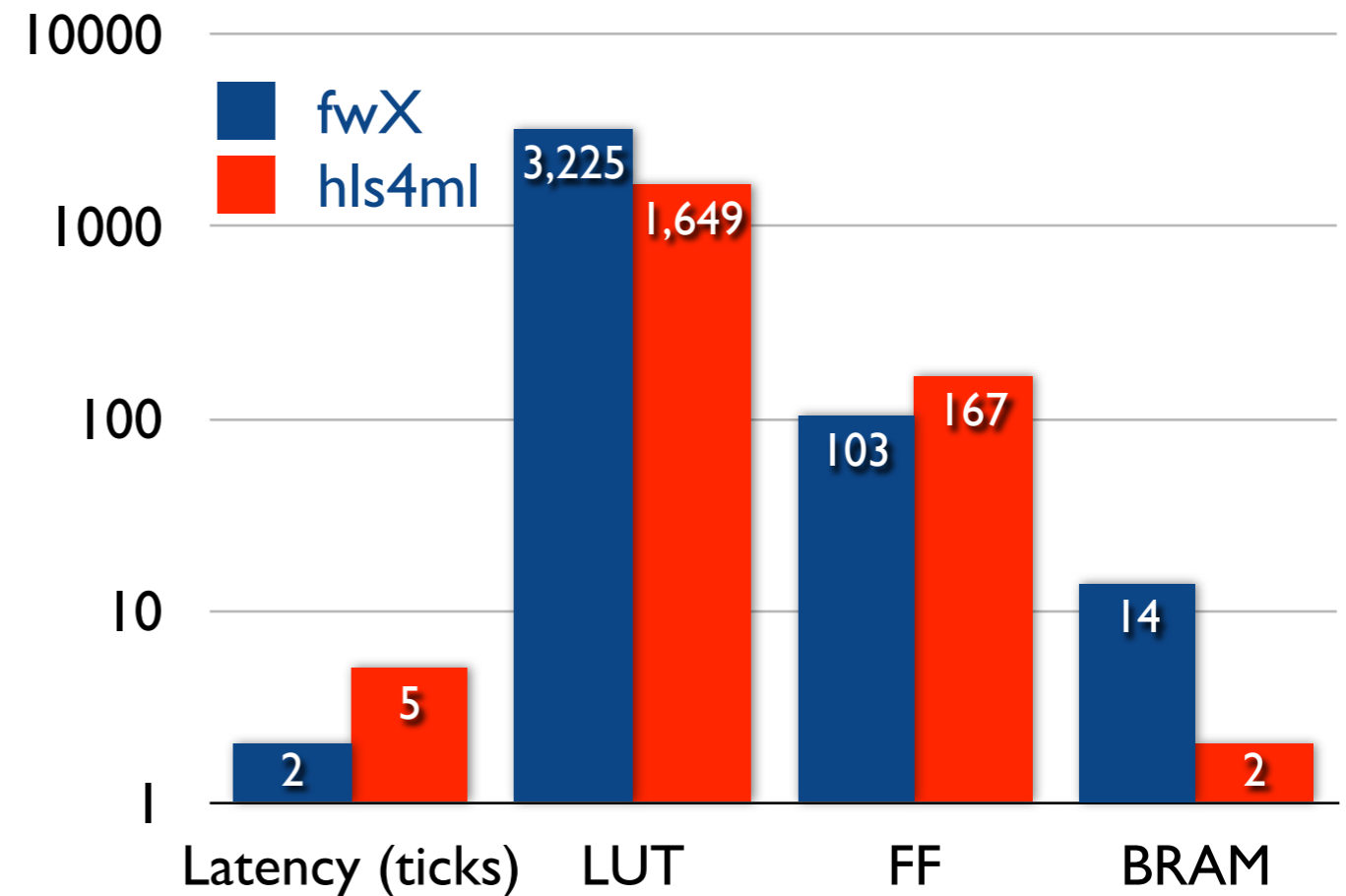
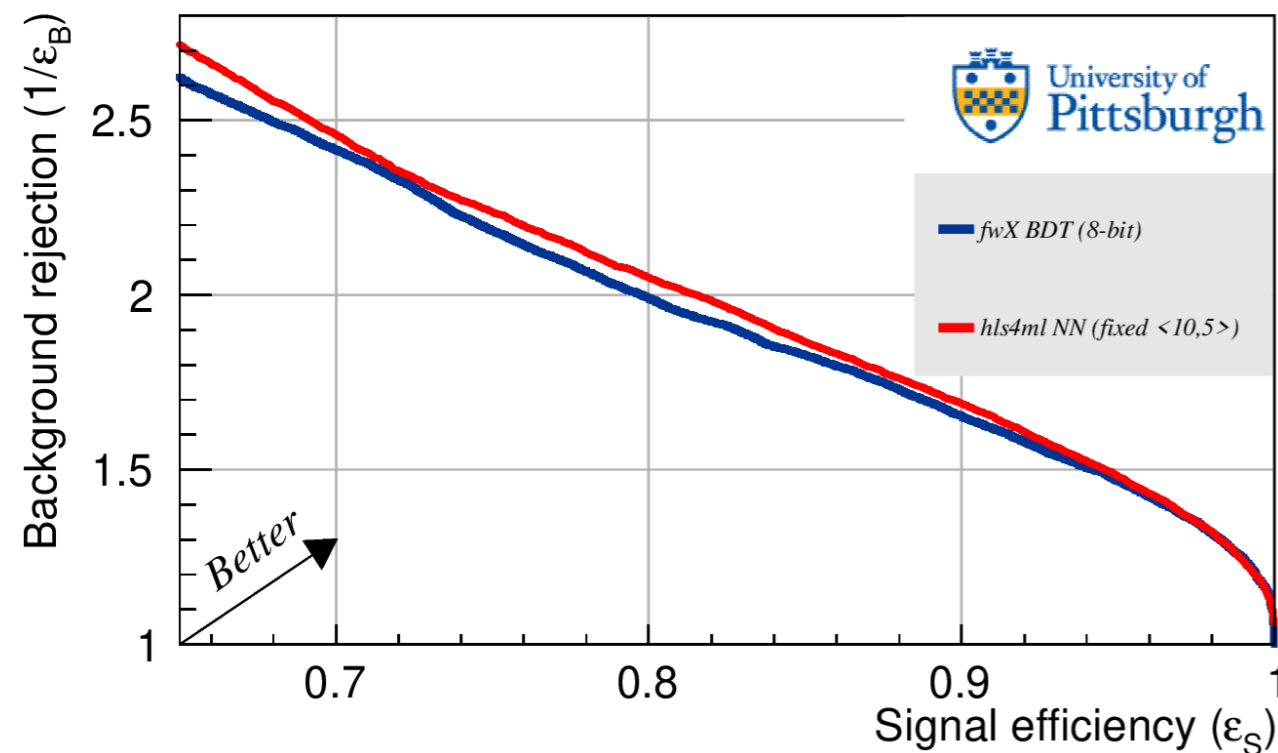
# fwX BDT vs. hls4ml neural network

## Setup

- Details **not** in paper
- Using 200 MHz clock here
- Chose NN architecture to match ROC performance of BDT

## Comparison

- Lower latency
- Comparable LUT, FF
- Higher BRAM (but 0.3% of avail.)







## Paper authors

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

*Undergraduate researchers*

† Now a Pitt MS student in EE

‡ Now a Pitt PhD student in EE

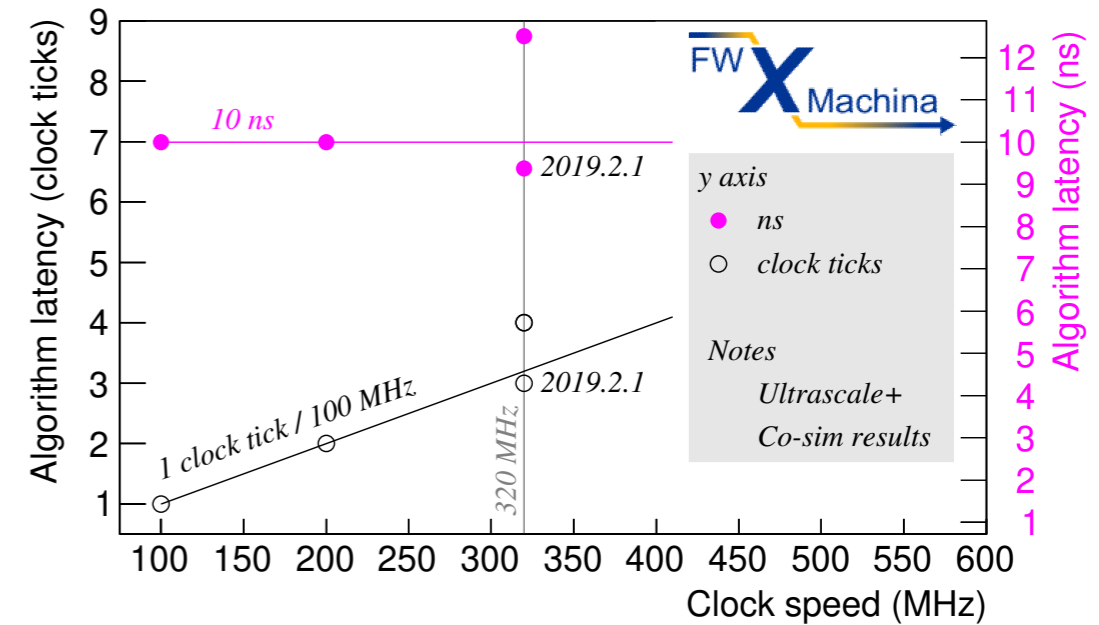
## New member

A. Rodic (did the comparison with hls4ml's neural network)

**Back up**



Parameter	Value	Comments
<b>FPGA setup</b>		
Chip family	Xilinx Virtex Ultrascale+	
Chip model	xcvu9p-flga2104-2L-e	
Vivado version	2019.2.1	
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</b>		
ML training method	Boosted decision tree	Binary classification
Boost method	Adaptive	AdaBoost with yes/no leaf
No. of event types to classify	2	Signal vs. background
No. of input variables	4	
No. of trees used for training	100	
Maximum tree depth	4	
<b>Nanosecond Optimization configuration</b>		
BIN ENGINE type	BIT SHIFT BIN ENGINE (BSBE)	
No. of bits for input variables	8 bits for each	
No. of bits for cut thresholds	8 bits for each	
No. of bits for BDT output score	8 bits	
No. of trees after merging	10	TREE MERGER via ordered list
No. of final trees	10, none removed	TREE REMOVER by truncation
No. of bins	26132	CUT ERASER not used
<b>FPGA cost</b>		
Latency	3 clock ticks	9.375 ns
Interval	1 clock tick	3.125 ns
Look up tables	1903 out of 1182240	< 0.2% of available
Flip flops	138 out of 2364480	< 0.01% of available
Block RAM	8 out of 4320	< 0.2% of available
Ultra RAM	0 out of 960	-
Digital signal processors	0 out of 6840	-

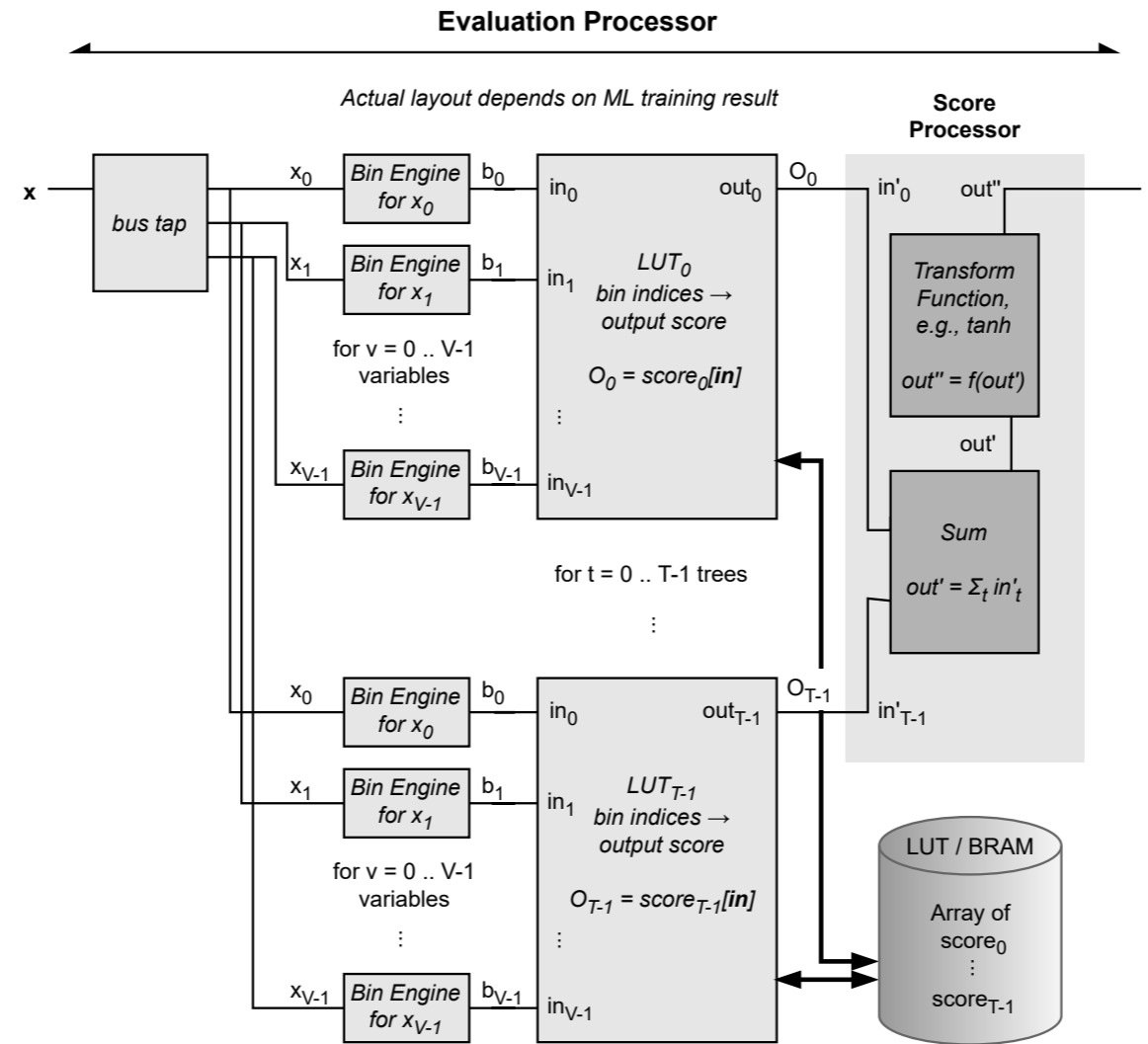


- 10 ns is independent of clock from 100-320 MHz

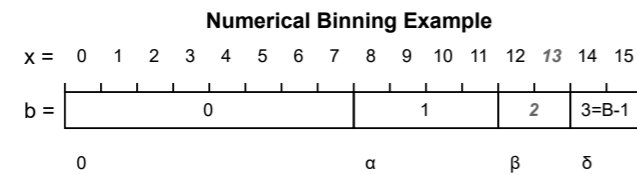
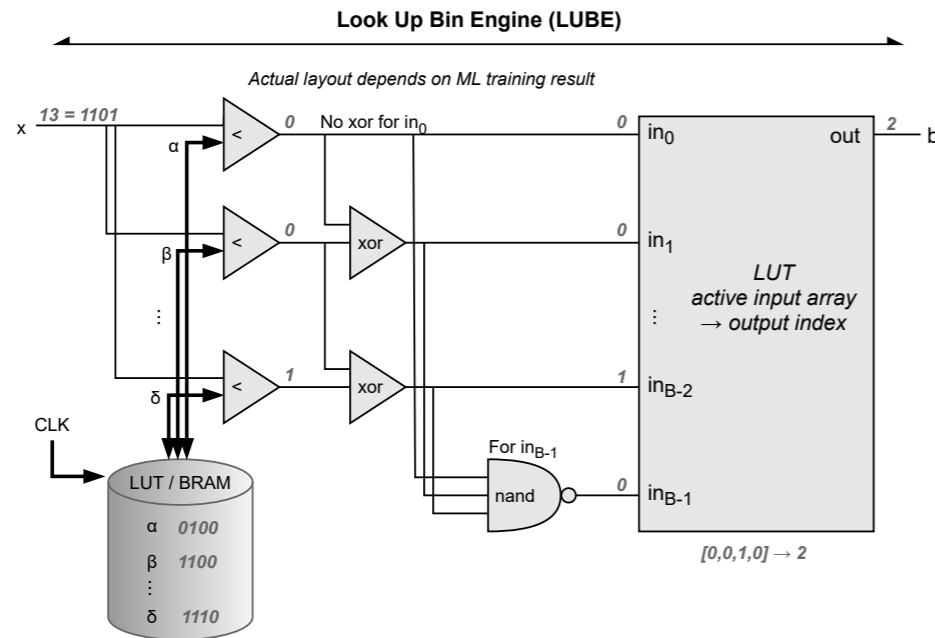


Table 9: List of input variables for the classification of the VBF Higgs boson vs. multijet process. Also given are the ATLAS-inspired cut-based offline thresholds for Run 2 [64] and HL-LHC [65]. For Run-2, differences arise with respect to the document when the  $m_{jj}$  threshold is quoted as 1100 GeV for L1 MJJ-500-NFF; we use the  $> 99\%$  offline efficiency point, which is achieved around  $m_{jj} > 1300$  GeV. For others the offline thresholds are used. For HL-LHC, the single-level scheme values are quoted. The performance of the cut-based approach using these values is compared the performance to the BDT result in figure 16. The non-optimized (non-opt) configuration includes the five variables from the optimized configuration.

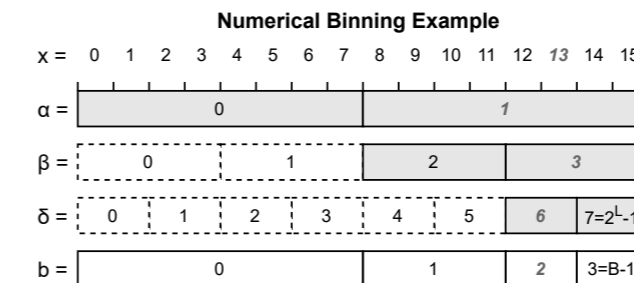
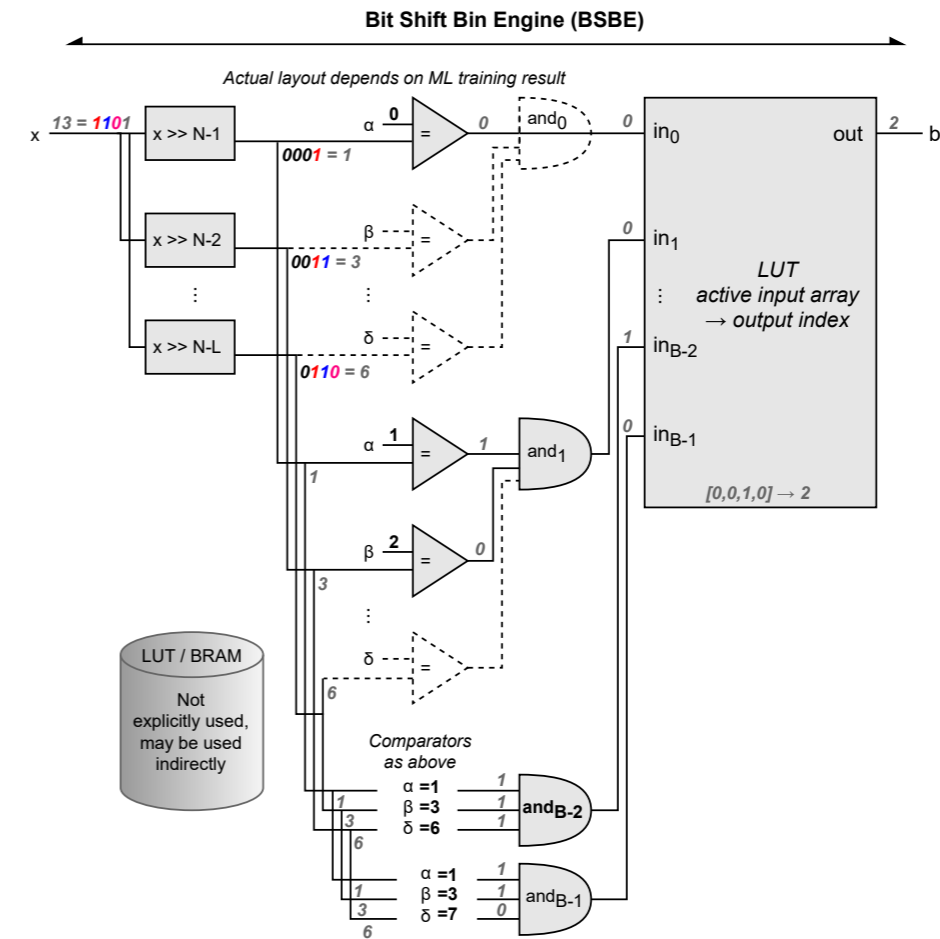
Input variable	Description	ATLAS Run-2 offline cut [64], see caption	ATLAS HL-LHC offline cut [65], see caption	Used in BDT
$p_{T1}$	Leading jet $p_T$	$> 90$ GeV	$> 75$ GeV	-
$p_{T2}$	Subleading jet $p_T$	$> 80$ GeV	$> 75$ GeV	Optimized
$p_{T12}$	Sum $p_{T1} + p_{T2}$	-	-	Optimized
$ \eta_1 $	Leading jet $\eta$	$< 3.2$	-	-
$ \eta_2 $	Subleading jet $\eta$	$< 4.9$	-	-
$\prod_\eta$	Product $\eta_1 \cdot \eta_2$	-	-	Optimized
$ \Delta\eta $	Separation in $ \eta_2 - \eta_1 $	$> 4.0$	$> 2.5$	-
$ \Delta\phi $	Separation in $ \phi_2 - \phi_1 $	$< 2.0$	$< 2.5$	non-opt
$ \Delta R $	$\sqrt{(\Delta\eta)^2 + (\Delta\phi)^2}$	-	-	non-opt
$m_{jj}$	Dijet invariant mass	$> 1300$ GeV	-	Optimized
$p_T^{jj}$	Dijet $p_T$	-	-	Optimized



- Each variable is processed independently of each other

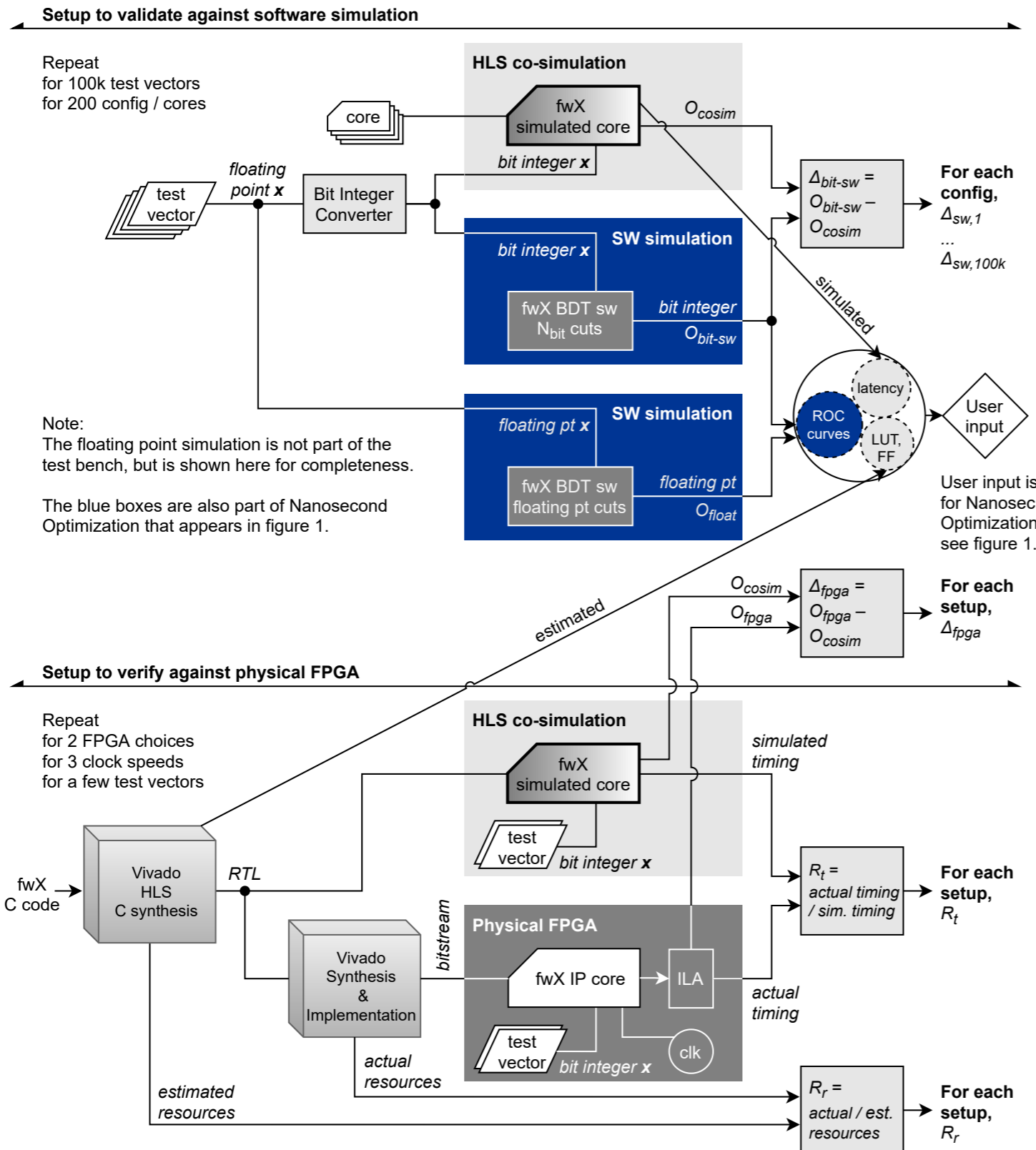


**Parameter Max Value**  
 N = 4, input bits  
 B = 4, max bin  
 Comparator input values



**Parameter Max Value**  
 N = 4, input bits  
 l = 1, layer no.  
 l = 2, layer no.  
 L = l = 3, max layer  
 B = 4, max bin

- Look up thresholds in memory, compare
- Bit shift to localize data
  - This is *fast*
- Use combinatoric logic as much as possible without multiplication. No explicit clocked operations.



- No difference seen wrt software implementation



All info at  
<https://fwx.pitt.edu>



**FW X Machina**

### Welcome!

Information regarding the **fwX** project will be available on this page. This project is developed by members of the [Hong Group](#) in the [Department of Physics and Astronomy](#).

### Where to find information

- The pre-print is available on the arXiv [\[2104.03408\]](#)
- Data samples used for the VBF Higgs and multijet study
  - Stephen Roche, Benjamin Carlson, Tae Min Hong (2021), *fwXmachina example: VBF Higgs vs multijet*, Mendeley Data, V1, doi: 10.17632/kp3myh3v89.1
- Download code
  - v1.0.0: <https://gitlab.com/PittHongGroup/fwX/-/tree/v1.0.0> and [Doxygen documentation](#)
- COMING SOON
  - User Guide
  - Tutorials

### Communicate with us

- Sign-up for the announcements at [Mailman](#) to receive emails from [fwx@list.pitt.edu](mailto:fwx@list.pitt.edu).
- Send inquiries to [fwx-developers@list.pitt.edu](mailto:fwx-developers@list.pitt.edu).

data set



git repo



doxygen







The screenshot shows the GitLab interface for a repository named 'fwX'. The left sidebar contains navigation options: Project overview, Details, Activity, Releases, Repository, Issues (0), Merge requests (0), Requirements, CI/CD, Security & Compliance, Operations, Packages & Registries, Analytics, Wiki, Snippets, Members, and Settings. The main content area displays project statistics (5 Commits, 1 Branch, 1 Tag, 4 MB Files, 4 MB Storage, 1 Release) and a file list table.

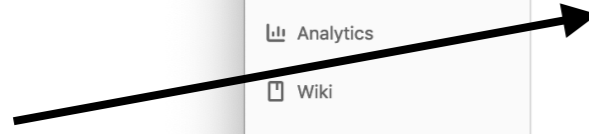
Name	Last commit	Last update
doc	first commit	20 minutes ago
examples	update stuff	55 minutes ago
fwXmachina	update stuff	55 minutes ago
images	update stuff	55 minutes ago
.gitignore	first commit	21 minutes ago
CHANGELOG	update stuff	55 minutes ago
EULA.md	first commit	25 minutes ago
README.md	update stuff	55 minutes ago
fwX.py	update stuff	55 minutes ago
setup.py	update stuff	55 minutes ago

Below the table is the 'README.md' content, featuring the 'FWX Machina' logo and a link to the project page: <https://PittHongGroup.gitlab.io/fwXmachina/>.

examples



code



driver file





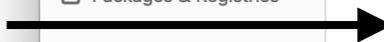
install

The screenshot shows a GitLab repository page for 'PittHongGroup / fwX'. The main content is the README.md file, which features the 'FWX Machina' logo and a flowchart illustrating the process: 'Perform ML' (receiving 'Input data' and 'TMVA scikit-learn ...') leads to 'Ultrafast Optimization' (outputting 'ROC, Latency, LUT, ...'), which then leads to 'Convert to HLS' (receiving 'User input' and using 'Xilinx HLS Vivado Suite' to produce 'Custom Firmware').

Below the flowchart, the README includes a section for '#Dependencies' and a detailed 'Vivado HLS Download and Installation' section with the following steps:

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

Other sections include 'Other' (listing dependencies like ROOT and Python 3), 'Installation', 'Local Installation', 'Dependencies' (mentioning CERN's ROOT framework), and 'Steps' (starting with a git clone command).





The screenshot shows a web browser displaying a GitLab repository page for 'PittHongGroup/fwX'. The browser address bar shows the URL 'https://gitlab.com/PittHongGroup/fwX/-/tree/v1.0.0'. The page features a sidebar with navigation options like 'Project overview', 'Repository', 'Files', 'Commits', 'Branches', 'Tags', 'Contributors', 'Graph', 'Compare', 'Locked Files', 'Issues', 'Merge requests', 'Requirements', 'CI/CD', 'Security & Compliance', 'Operations', 'Packages & Registries', 'Analytics', 'Wiki', 'Snippets', 'Members', and 'Settings'. The main content area displays a file explorer for the 'bd\_t\_hls\_firmware' directory, with 'solution1' selected. Below the explorer, a code editor shows the contents of 'bd\_t.cpp', which is an auto-generated implementation file for Vivado HLS. The code includes a function definition for 'fwXbdt' with various pragmas and a loop that iterates over a tree structure to calculate a score sum.

these files are not located in the HLS project itself, but in the folder named `src`, which is located next to the HLS project folder. In `bd_t.cpp`, we can see two functions. The first (`fwXbdt()`) is what is known as the top-level function, or the function that gives the physical interface shape for the synthesizer.

```

Synthesis(solution1)(fwXbdt_csynth.rpt)  bdt.cpp  bdt.hpp
1  /* auto generated bdt implementation file for Vivado HLS */
2
3  #include "bdt.hpp"
4
5
6  out_t fwXbdt(in_t event[NVARS]) {
7  #pragma HLS ARRAY PARTITION variable=event complete dim=1
8  #pragma HLS INTERFACE ap_ctrl_none port=return
9  #pragma HLS INTERFACE ap_none port=event
10
11 #pragma HLS PIPELINE II=1
12
13
14
15     out_t scoreSum = 0;
16
17     for (int idxTree = 0; idxTree<NTREES; ++idxTree) {
18         in_t binMap[NVARS];
19         bin(event, binMap, idxTree);
20         scoreSum += getScore(binMap, idxTree);
21     }
22     return scoreSum;
23 }
24
// Collapse sidebar
https://gitlab.com/PittHongGroup/fwX/-/raw/v1.0.0/images/vhls_main_func.png ent[NVARS], in_t idxBin[NVARS], int idxTree) {
X  Q tree  Highlight All  Match Case  Match Diacritics  Whole Words  1 of 18 matches
    
```

fwXmachina: Class List
fwXmachina: tree.Tree Class Reference

https://www.fwx.pitt.edu/doc/v1.0.0/html/class\_tree\_1\_1Tree.html
Most Visited | Reload via ULS | Kick Ass

fwXmachina
Search

Main Page | Related Pages | Packages | **Classes** | Files

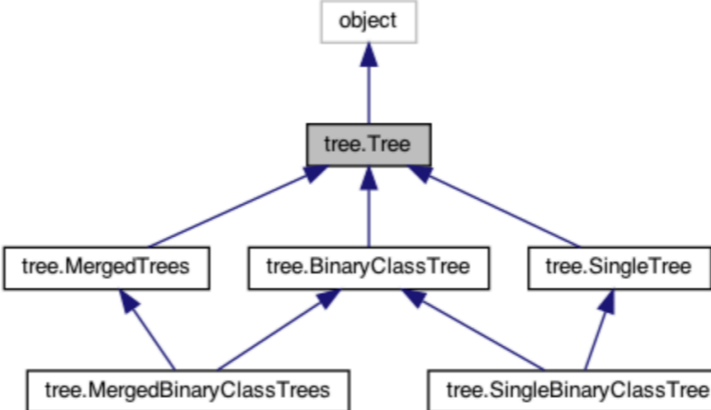
- fwXmachina
  - fwXbdt Documentation
  - Todo List
  - Packages
  - Classes
    - Class List
      - bd
      - bin
      - cuts
      - forest
      - generalClasses
      - include
      - node
      - set
      - testpoints
      - tree
        - Tree**
          - SingleTree
          - MergedTrees
          - BinaryClassTree
          - SingleBinaryClassTree
          - MergedBinaryClassTrees
        - variable
      - Class Index
      - Class Hierarchy
      - Class Members
      - Files

## tree.Tree Class Reference

Public Member Functions | Public Attributes | Private Member Functions | List of all members

Class from which [SingleTree](#) and [MergedTrees](#) inherit. [More...](#)

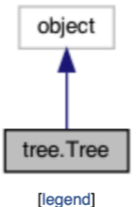
Inheritance diagram for tree.Tree:



```

graph BT
    object --> tree_Tree[tree.Tree]
    tree_Tree --> tree_MergedTrees[tree.MergedTrees]
    tree_Tree --> tree_BinaryClassTree[tree.BinaryClassTree]
    tree_Tree --> tree_SingleTree[tree.SingleTree]
    tree_MergedTrees --> tree_MergedBinaryClassTrees[tree.MergedBinaryClassTrees]
    tree_BinaryClassTree --> tree_SingleBinaryClassTree[tree.SingleBinaryClassTree]
  
```

Collaboration diagram for tree.Tree:



```

graph BT
    object --> tree_Tree[tree.Tree]
  
```

### Public Member Functions

```
def __init__(self, str comes_from=None, BDT_object=None)
```

Initializes a tree. [More...](#)

```
str get_title(self)
```

Return tree's title. [More...](#)

```
'Tree' __deepcopy__(self, memo)
```

Alters copy.deepcopy a little for this class's purpose. [More...](#)

```
np.ndarray get_cuts(self)
```

Get all the cuts and turn them into one big 2d Numpy array for each data type. [More...](#)

```
np.ndarray get_intermediates(self)
```

Gets the intermediates for each variable. [More...](#)

tree > Tree
Generated on Mon May 10 2021 22:03:00 for fwXmachina by [doxygen](#) 1.9.1

# vs. hls4ml's BDT

Parameter	FWXMACHINA	hls4ml/Conifer
<b>ML training setup</b>		
Training software	TMVA	TMVA
Physics problem	electron vs. photon	electron vs. photon
Training samples	from ref. [56]	from ref. [56]
No. of event classes	2	2
No. of training trees	100	100
Max. depth	4	4
No. of input variables	4	4
Other TMVA parameters	TMVA defaults	TMVA defaults
Nanosec. Optimization	Flattened & merged to 10 final trees, without TREE REMOVER or CUT ERASER	N/A
<b>FPGA and firmware setup</b>		
Chip family	Xilinx Virtex Ultrascale+	Xilinx Virtex Ultrascale+
Chip model	xcvu9p-flga2104-2L-e	xcvu9p-flga2104-2L-e
Vivado HLS version	2019.2	2019.2
Clock speed, period	320 MHz, 3.125 ns	320 MHz, 3.125 ns
Precision	ap_int<8>	ap_ufixed<10,5>
BIN ENGINE	BSBE	N/A
<b>FPGA cost</b>		
Actual timing values and resource usage by RTL synthesis and implementation		
Latency	3 clock ticks, 9.375 ns	15 clock ticks, 46.875 ns
Interval	1 clock ticks, 3.125 ns	1 clock tick, 3.125 ns
LUT	717, 0.06% of total	3834, 0.3% of total
FF	147, < 0.01% of total	1966, < 0.1% of total
BRAM 18k	5.5, 0.1% of total	0
URAM	0	0
DSP	2, 0.03% of total	0

