Rapid HEP ML inference with logic gate neural nets

ACAT 2025

09.09.25

This presentation contains several animations and is best viewed as .key or .pptx, or you can view it online here

Lino Gerlach, Elliott Kauffman, Liv Våge

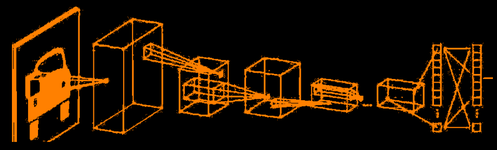
Contents

Differentiable logic gate networks

difflogic

CICADA

Differentiable logic gates for CICADA



FPGA implementation

AMD Vitis™ HLS

Foundational work

difflogic

SOTA inference times on MNIST
- 10-20x faster than others

Original difflogic paper presented at NEURIPS 2022 [1]

Code repo for the original paper

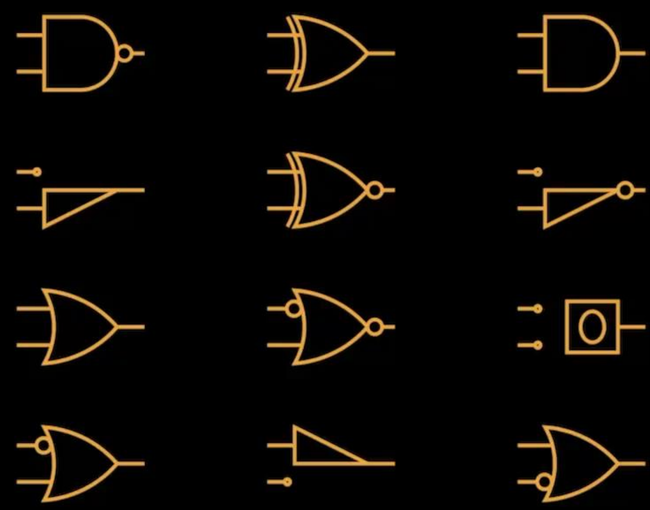
Follow up paper on difflogic CNN from NEURIPS 2024

TLDR

Good physics performance and rapid inference for CICADA student model with LGCN

[1] Felix Petersen et al.

Logic gate neural networks



Construct neural network from logic gates
instead of nodes, we have logic gates

Convert the input to a binary representation
different representations can produce different results

Each node receives two inputs
The connections are randomly initialised

Outputs are aggregated so we can classify or regress

Output dimension Neuron output

$$\hat{y}_i = \sum_{j=i\cdot n/k+1}^{(i+1)\cdot n/k} a_j/\tau + \beta$$

Final output Optional offset

Number of output neurons Normalisation temperature

Video adapted from [2] , made by Felix Petersen et al.

Why it is fast

At inference each 16 gate block is replaced by most probable gate

Binary computations are fast

Compiler can optimise the binary logic

No matrix multiplications!

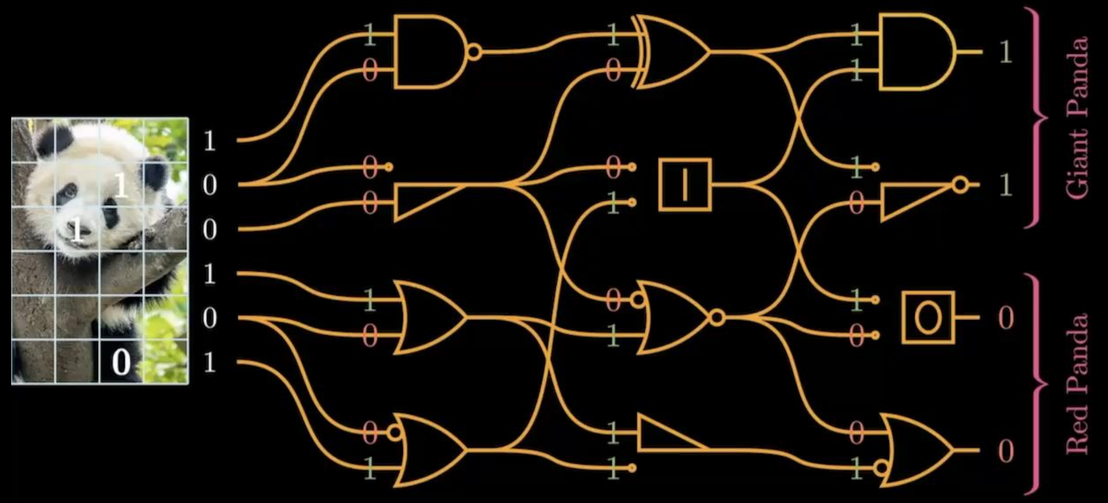
Making it differentiable

Relax gate operations to continuous approximations
binary inputs are usually not differentiable

Softmax of the 16 gate blocks
we need learn which gates to use

During training evaluate 16 gate blocks
slow training, but quick inference as we replace each block of 16 with one gate

ID	Operator	real-valued	00	01	10	11
0	False	0	0	0	0	0
1	$A \wedge B$	$A \cdot B$	0	0	0	1
2	$\neg(A \Rightarrow B)$	$A - AB$	0	0	1	0
3	A	A	0	0	1	1
4	$\neg(A \Leftarrow B)$	$B - AB$	0	1	0	0
5	B	B	0	1	0	1
6	$A \oplus B$	$A + B - 2AB$	0	1	1	0
7	$A \vee B$	$A + B - AB$	0	1	1	1
8	$\neg(A \vee B)$	$1 - (A + B - AB)$	1	0	0	0
9	$\neg(A \oplus B)$	$1 - (A + B - 2AB)$	1	0	0	1
10	$\neg B$	$1 - B$	1	0	1	0
11	$A \Leftarrow B$	$1 - B + AB$	1	0	1	1
12	$\neg A$	$1 - A$	1	1	0	0
13	$A \Rightarrow B$	$1 - A + AB$	1	1	0	1
14	$\neg(A \wedge B)$	$1 - AB$	1	1	1	0
15	True	1	1	1	1	1



Video adapted from [2], made by Felix Petersen et al.

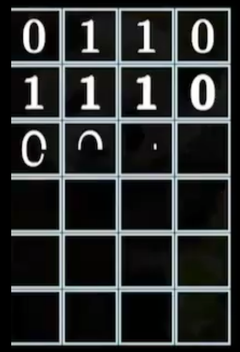
Learnable weights

Logic gate operation

Neuron output

$$a' = \sum_{i=0}^{15} \frac{e^{\mathbf{w_i}}}{\sum_j e^{\mathbf{w_j}}} \cdot f_i(a_1, a_2)$$

Convolutional differentiable logic gate neural networks



Linear layers is not enough
empirically struggled to train over 6 layers

Replace normal kernel by binary tree
aggregates information while keeping expressivity

One channel for each input bit
We learn the significance of each bit, analogous to colour

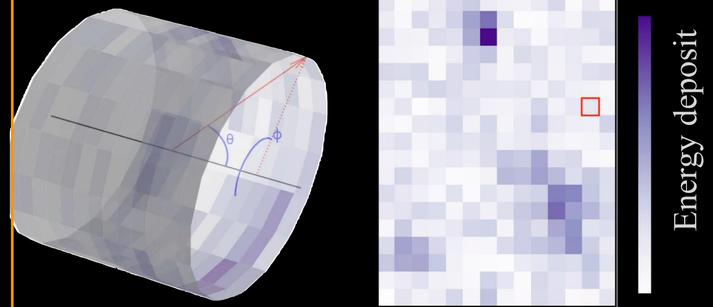
Create special *Or* pooling layers
fast and only need to propagate through the maximum activations

$$\perp_{max}(a, b) = max(a, b)$$

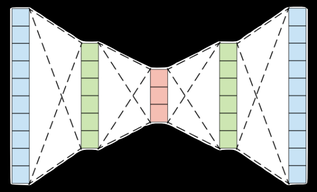
Video adapted from [2] , made by Felix Petersen et al.

CICADA anomaly detection at the CMS Level-1 Trigger

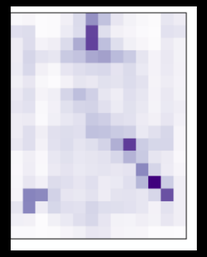
Teacher



18 x 14 calorimeter input

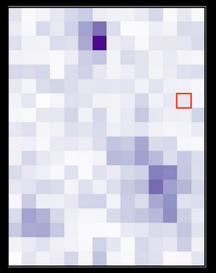


Autoencoder



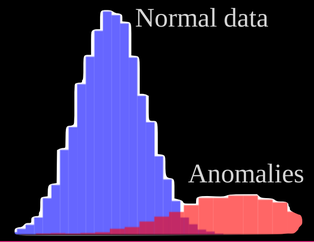
Reconstructed image

Student



18 x 14 calorimeter input

Teacher loss



Student is implemented on FPGA in the L1T which has output rate of 100 kHz

Can we achieve better physics performance or latency?

[3] CICADA collaboration

Results

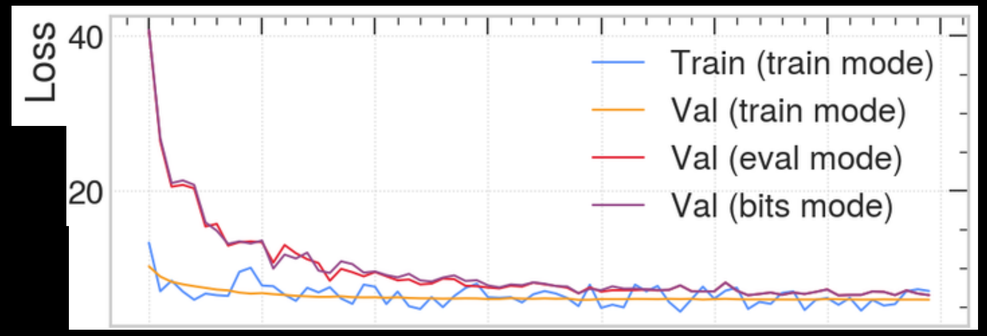
Using 2017 CMS Open data
will be openly available very soon

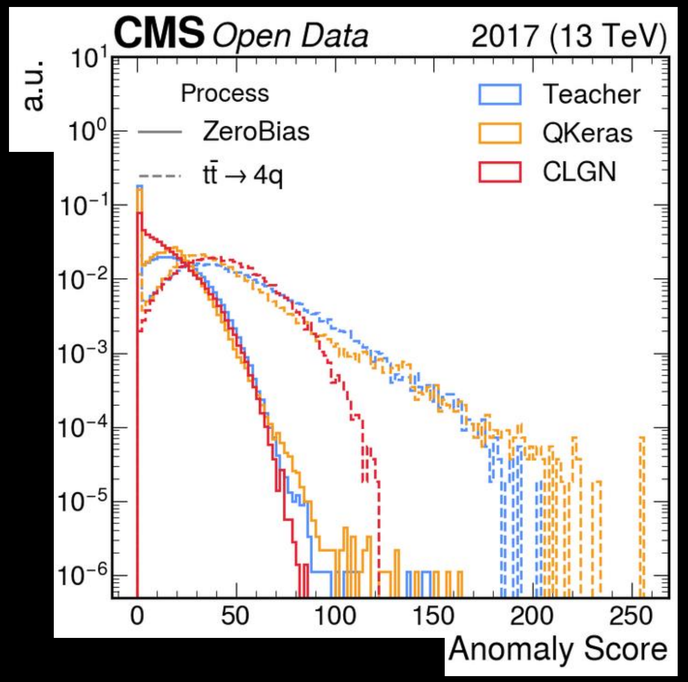
Background is Zero Bias, $\bar{t}t$ is outlier
There are three possible outlier final states, two are reserved for validation and testing

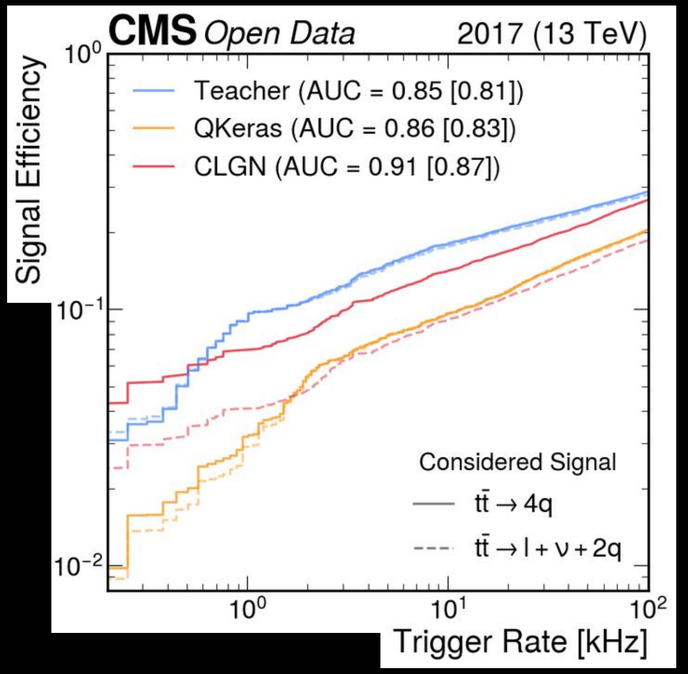
To best convert inputs to binary implement thermometer encoding
learns how to threshold binary numbers during training, as in [4]

Very high anomaly scores lower since binarised inputs
This is okay, we can set the trigger threshold

Convolutional LGN outperforms QKeras implementation
QKeras is especially worse on low trigger rates







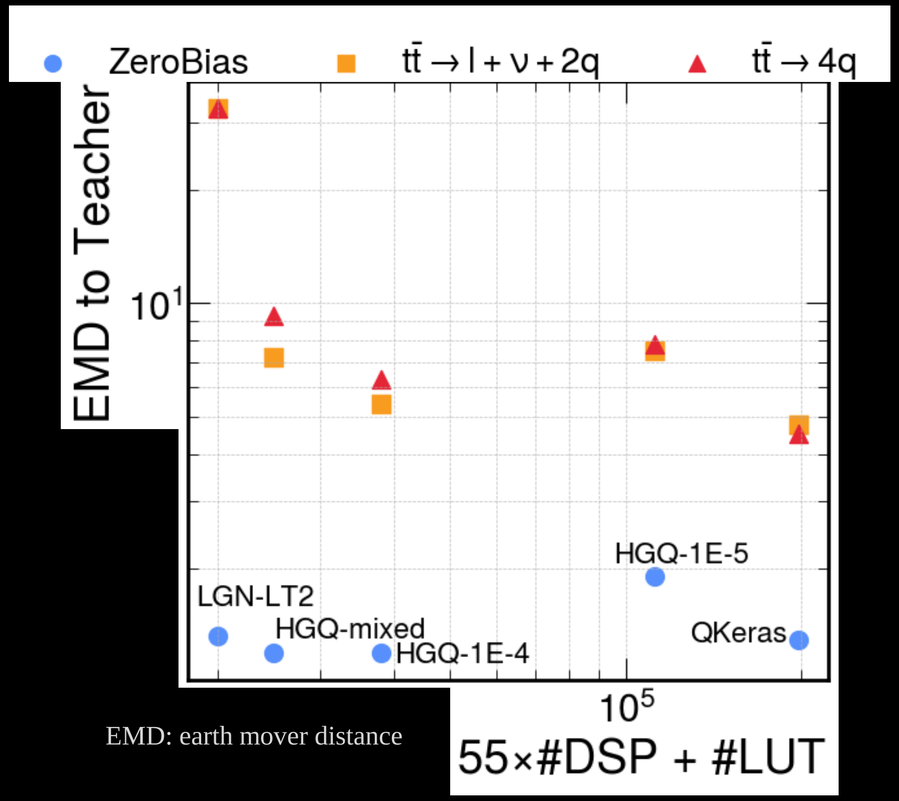
FPGA synthesis

Synthesised for AMD Virtex-7
using AMD Vitis HLS

This had ~half the trainable parameters as compared to previous slide
the FPGA performance is therefore likely an underestimation
results are good on background, not on signal

Still we see very promising FPGA performance
no DSP usage since we don't have matrix multiplications

Latency outperforms QKeras and HGQ implementations
details of those can be found in [5]



EMD: earth mover distance

Model Label	Quantization Library	Latency (in cc)	DSPs	FFs	LUTs	HGQ EBOPs
QKeras	QKeras	16	697	50368	159447	-
HGQ-1E-5	HGQ	17	4	27776	111848	39170
HGQ-1E-4	HGQ	11	1	6229	38111	7570
HGQ-mixed	HGQ	8	0	3019	24947	3301
LGN-LT2	LGN	3	0	856	19977	-

Vitas synthesis estimates

Conclusion

Implemented difflog convolutions with:
- CPU & GPU
- FPGA synthesis

Results:
- Good physics performance for CICADA student
- Promising FPGA performance with 0 DSP use

Further work

Almost done with a custom CUDA implementation

There have been developments in the field to consider

Code implementation

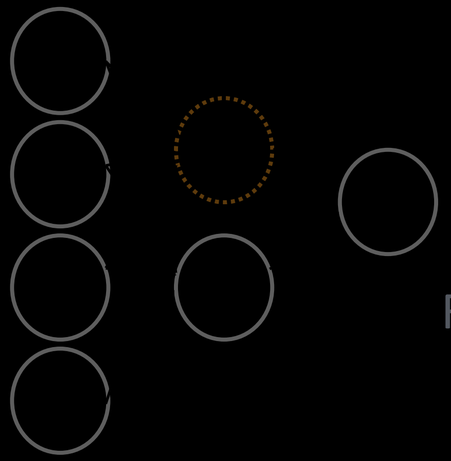
Will be included in proceedings

Backup

MLP vs difflog MLP

ID	Operator	real-valued	00	01	10	11
0	False	0	0	0	0	0
1	$A \wedge B$	$A \cdot B$	0	0	0	1
2	$\neg(A \Rightarrow B)$	$A - AB$	0	0	1	0
3	A	A	0	0	1	1
4	$\neg(A \Leftarrow B)$	$B - AB$	0	1	0	0
5	B	B	0	1	0	1
6	$A \oplus B$	$A + B - 2AB$	0	1	1	0
7	$A \vee B$	$A + B - AB$	0	1	1	1
8	$\neg(A \vee B)$	$1 - (A + B - AB)$	1	0	0	0
9	$\neg(A \oplus B)$	$1 - (A + B - 2AB)$	1	0	0	1
10	$\neg B$	$1 - B$	1	0	1	0
11	$A \Leftarrow B$	$1 - B + AB$	1	0	1	1
12	$\neg A$	$1 - A$	1	1	0	0
13	$A \Rightarrow B$	$1 - A + AB$	1	1	0	1
14	$\neg(A \wedge B)$	$1 - AB$	1	1	1	0
15	True	1	1	1	1	1

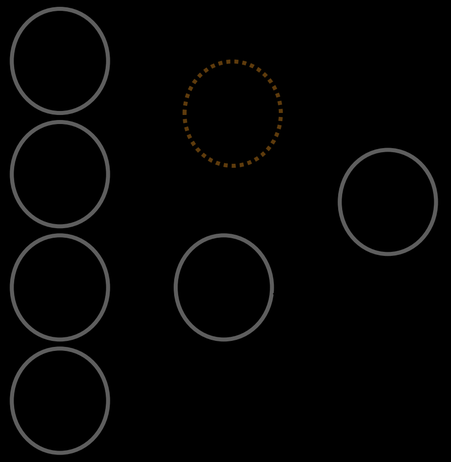
Normal neural net



ReLu

X	W
121	0.1 -0.5 0.9 2
244	
31	
121	

Logic gate neural net



X	f(x1, x2)	w
0.21	0	-0.1
0.63	0.21 * 0.63	0.2
	0.21 - 0.21 * 0.63	2
	....	....

3% 10% 40 % ...
false and or

Fully connected

Matrix multiplication in nodes

Same in training as inference

Randomly connected; each node has two inputs

Nodes evaluate logic gates

At inference, inputs are binary and only use most probable logic gate for each node

No matrix multiplication, binary logic can be simplified by compiler, binary computations faster

Thermometer thresholding

We need to convert continuous numbers to binary

We do this based on thresholds, as implemented in [4]

We learn the bin edges as we train

each bin checks whether the number ≥ threshold
1 if it is, 0 otherwise

Our representation matters

the best edges will depend on the problem

In our case, we let it change, then freeze it

we use a 6 bit representation
one of the bins settles on a high threshold - likely to capture high anomaly scores