Imperial College London

Post-training ReLU Sparsification for Faster CNN Inference on FPGA Streaming Accelerators

Krish Agrawal, Zhewen Yu, Alexander Montgomerie-Corcoran, and Christos-Savvas Bouganis

Intelligent Digital Systems Lab

Dept. of Electrical and Electronic Engineering

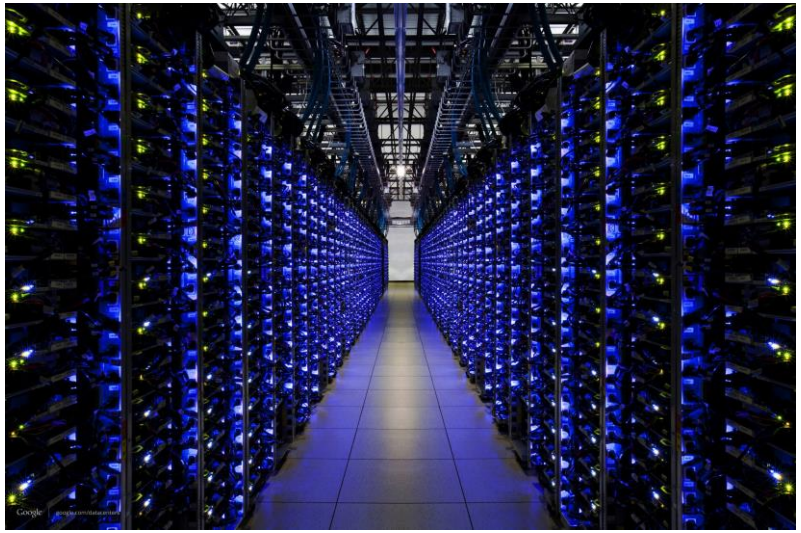
www.imperial.ac.uk/idsl

Where are DNN Models deployed?

Edge: Low Latency

- Compute closer to sensor
- Avoid large latency of sending to servers
- Examples: *Drones, Self-driving cars*

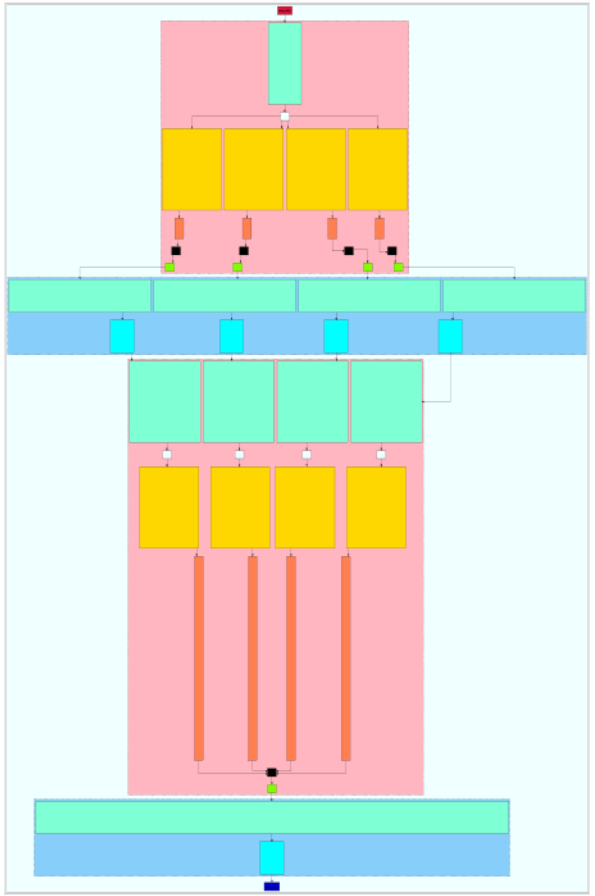
Where are DNN Models deployed?



Datacenter: High Throughput

- Offloading computation to server
- Examples: *Google Translate, ChatGPT*

CNN Accelerator Streaming Architecture



- Hardware is **customized** for a specific CNN Model
- All layers of the CNN Model are **pipelined** together

*(FINN, HLS4ML, **fpgaConvNet**)*

github.com/AlexMontgomerie/fpgaconvnet-hls

iDSL Research

DSE Optimization
(FPL 2022)

Model Compression
(FPT 2021, FPL 2023)

Early Exit Networks
(FCCM 2023)

3D CNNs
(ASAP 2023,
FPL 2023, FCCM 2023)

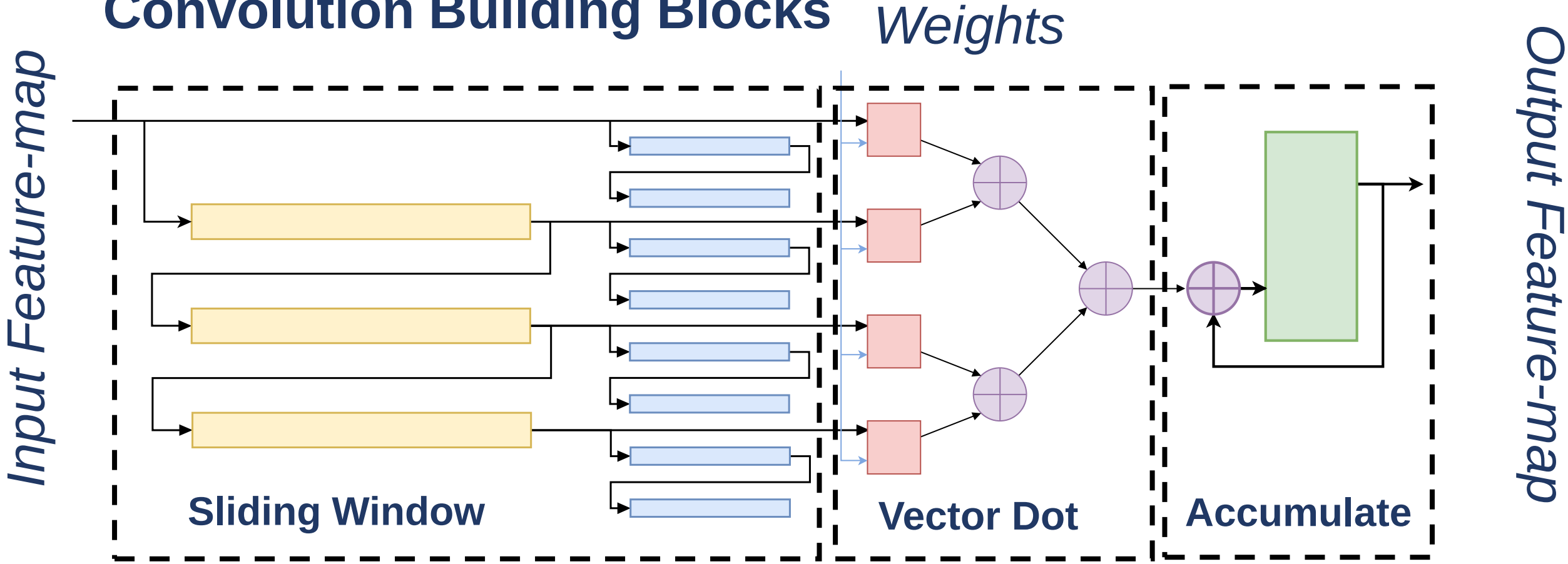
fpgaConvNet

Power Consumption
(FPT 2019, ASP-DAC 2021)

TinyML
(MLPerf Tiny 1.1)

Model Sparsity
(FPL 2023)

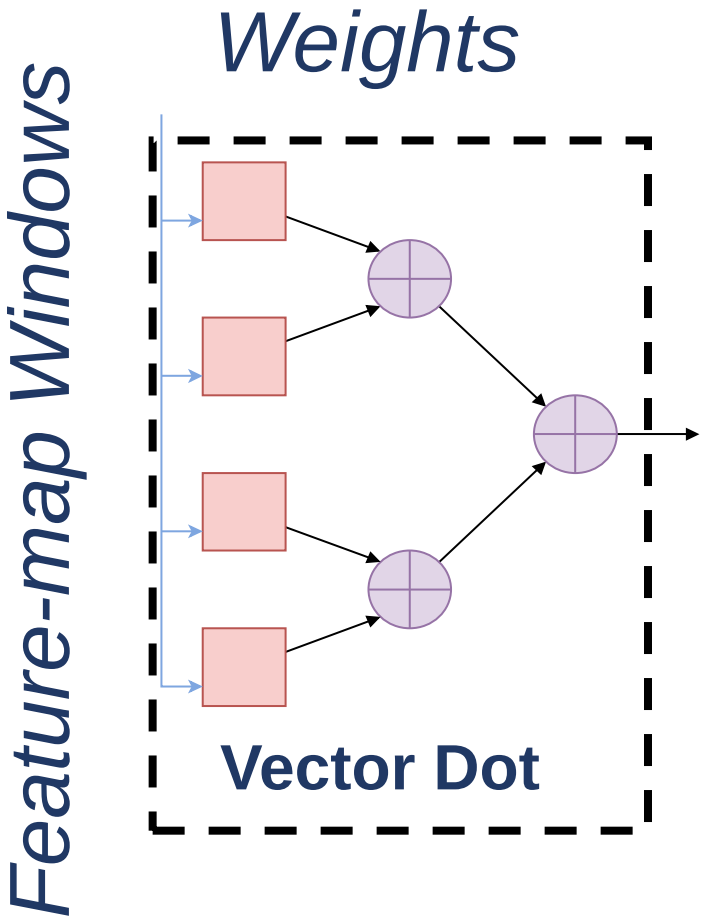
Convolution Building Blocks



Convolution Building Blocks

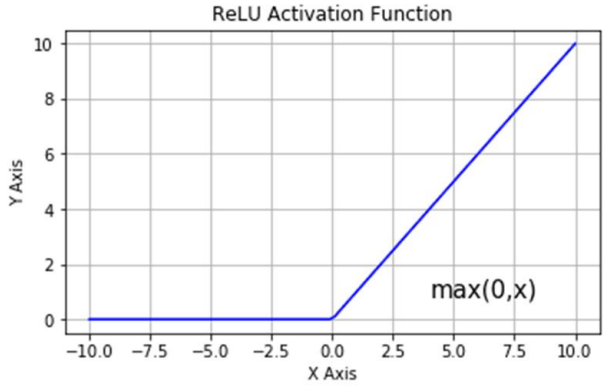
- Dot-product between feature-map and weight windows

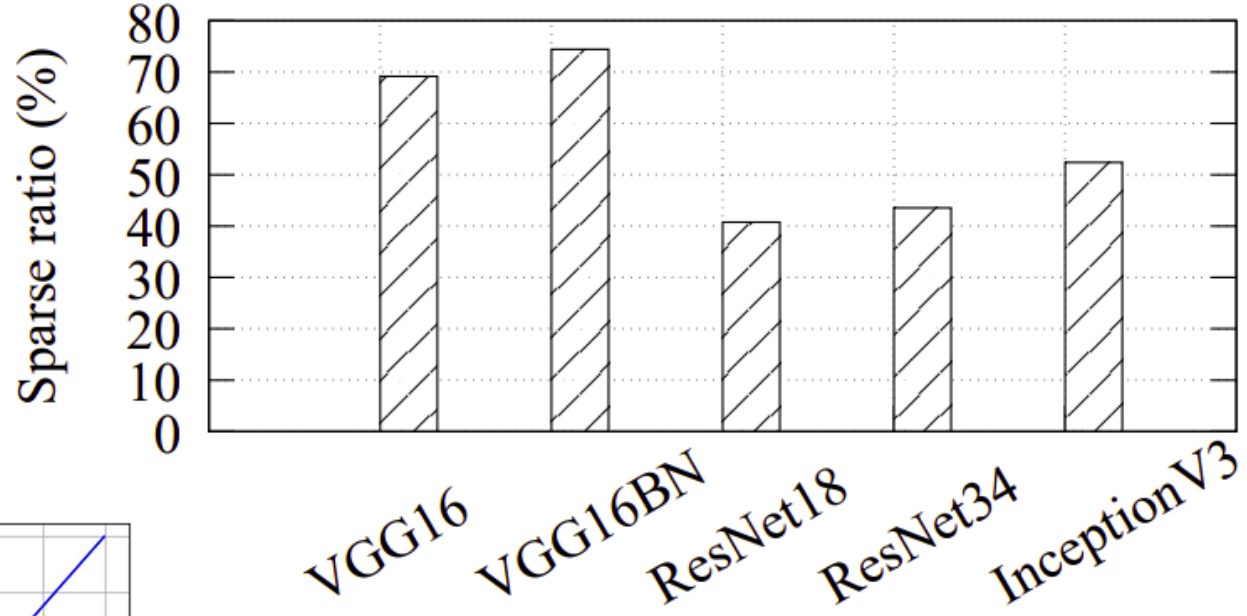
$$\mathbf{O}(f, x, y) = \sum_{c=0}^{C} \sum_{k_x=0}^{K} \sum_{k_y=0}^{K} I\left(c, x + k_x, y + k_y\right) \cdot W(c, k_x, k_y, f)$$



Activation Sparsity in CNN Models

- Up to **80%** of values in the network are zero on average
- Can reach **95%** for feature-maps within the network
- A lot of multiplying by zero
- Result of **ReLU** operations in the network



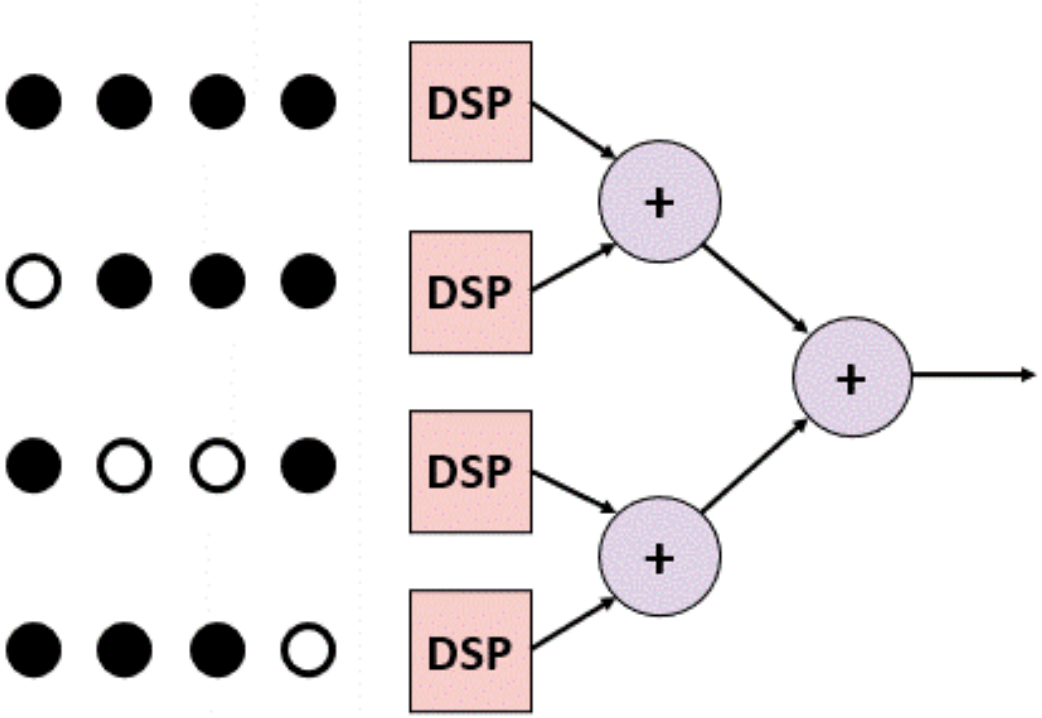


S. Cao et al., "SeerNet: Predicting Convolutional Neural Network Feature-Map Sparsity Through Low-Bit Quantization," 2019

How do we exploit this?

- Most operations are **zero** times **something,** which is just **zero**
- Arithmetic circuit is only **multiply** and **addition**
- The **cost** of **multipliers** is much **greater** than **adders**
- Let's use Multipliers more efficiently!

Vector Dot Product Engine

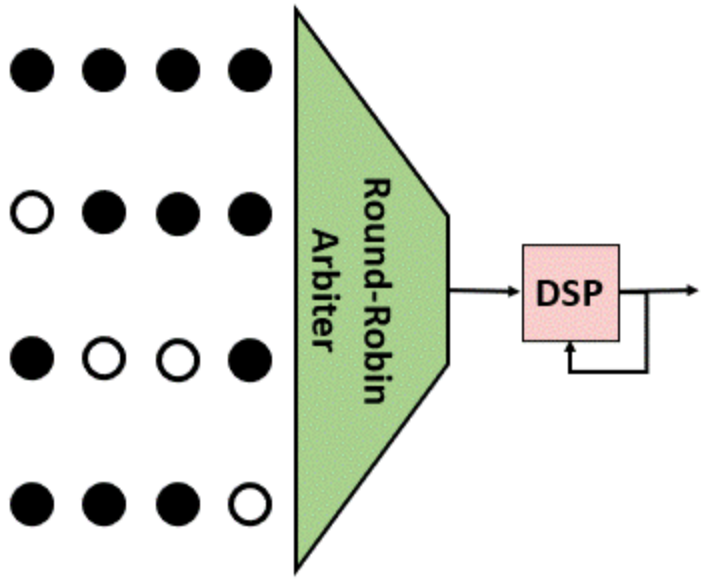


Non-Zero Value

Zero Value

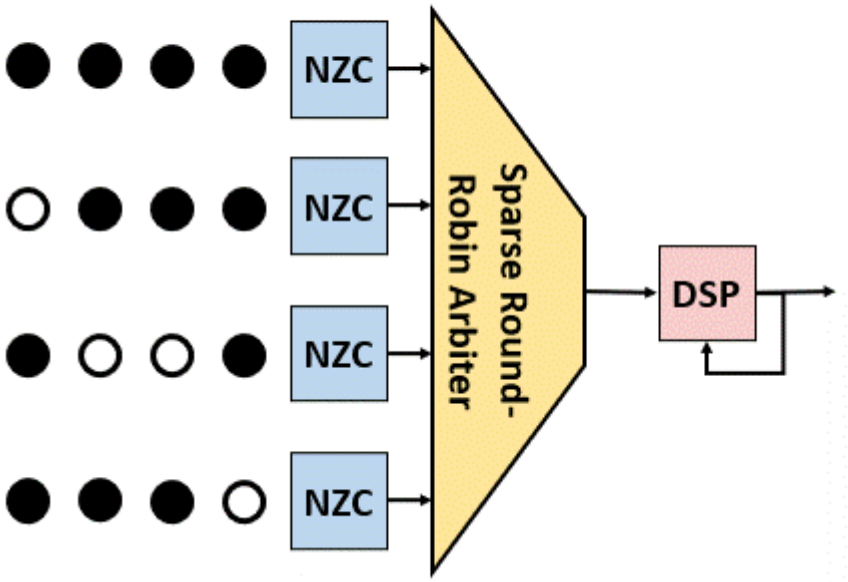
- Initiation interval of **1**
- **4** multipliers
- Only **1** Non-Zero multiplication each cycle
- **3** DSPs doing nothing!

Vector Dot Product Engine - Alternative



- Initiation interval of **4**
- **1** multiplier
- Only **1** Non-zero multiplication every **4** cycles
- Still inefficient use of the DSP

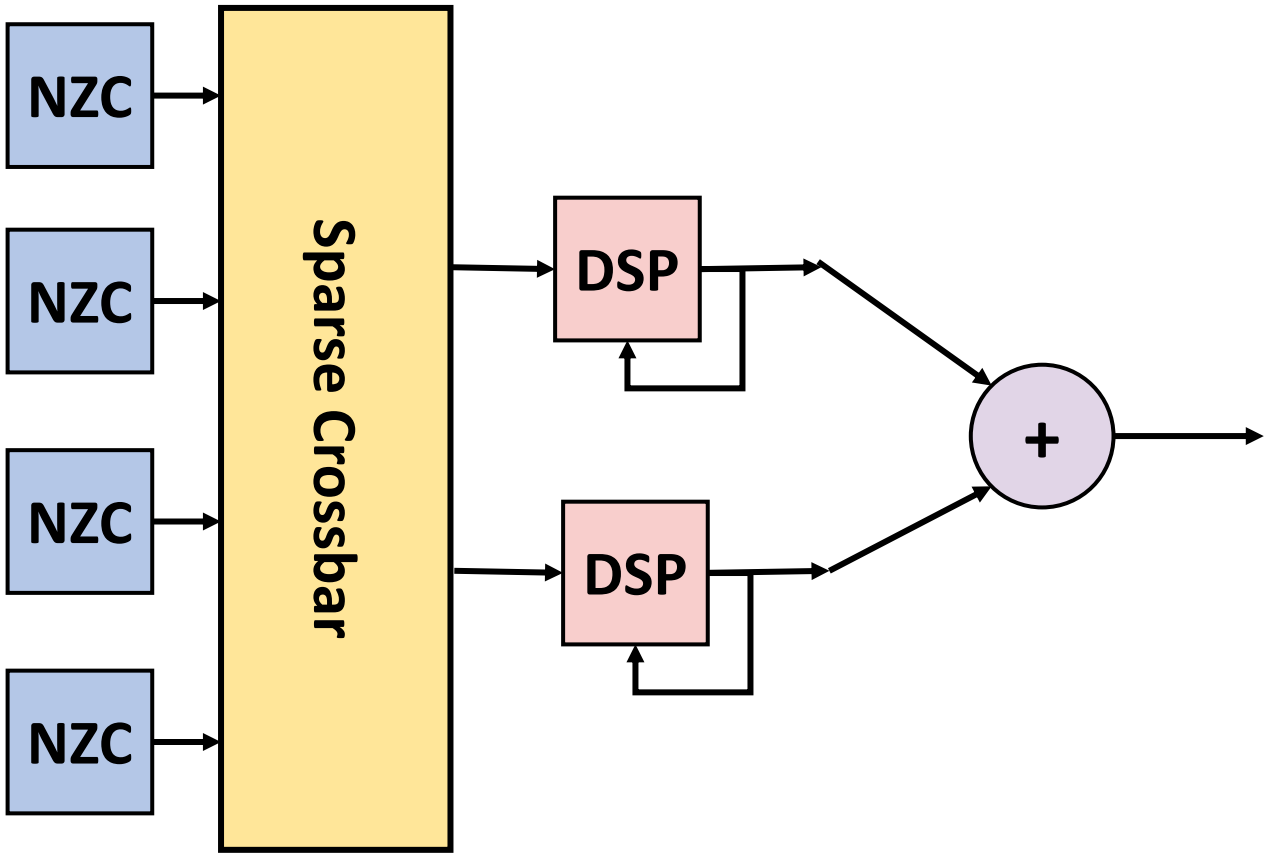
Sparse RR Arbiter Vector Dot Product Engine



NZC **Non-Zero Check**

- Initiation interval of **1**
- **1** multiplier
- Full DSP occupancy!
- (extra overheads for Non-Zero Checking)

Sparse Crossbar Vector Dot Product Engine

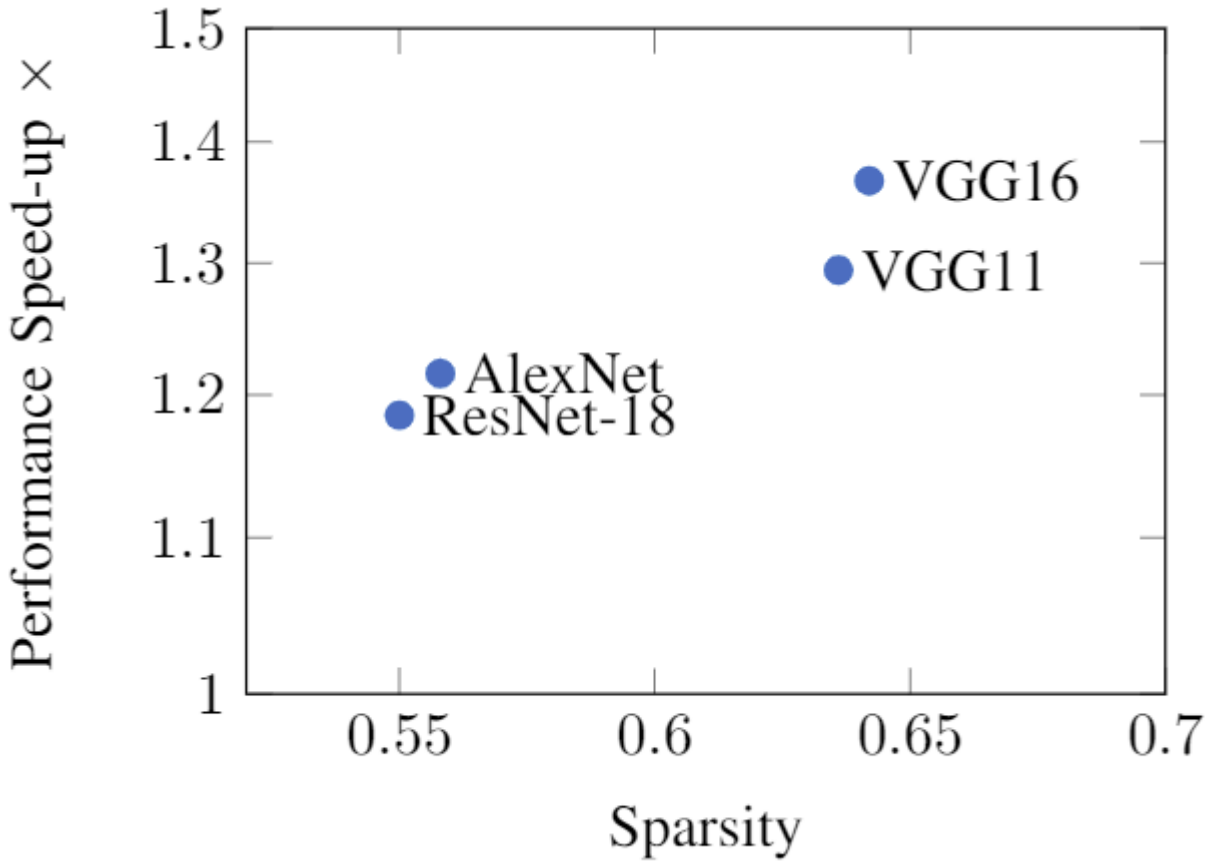


We can generalize this for any **M** inputs to **N** multipliers using a **sparse crossbar** instead.

Speed-up

(compared to baseline fpgaConvNet)

- Performance increases with sparsity



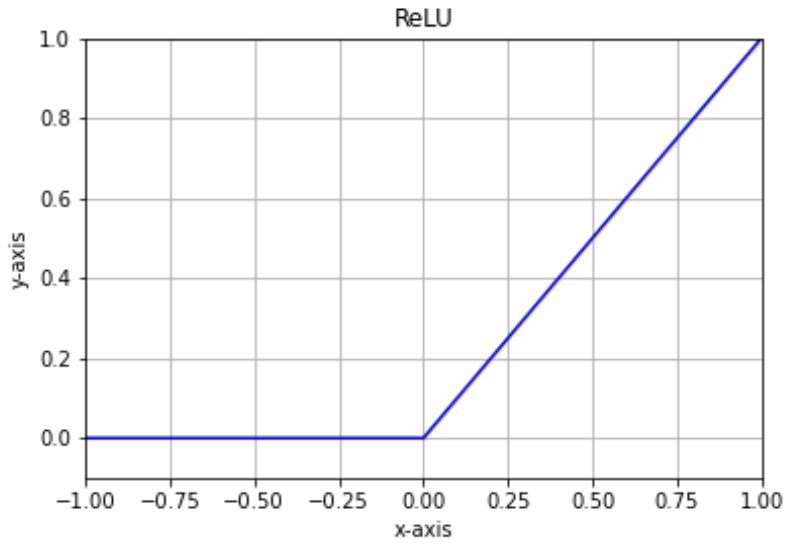
How we make it even faster (without much effort)?

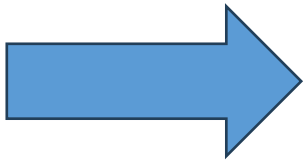
Post-training ReLU-based sparsification!

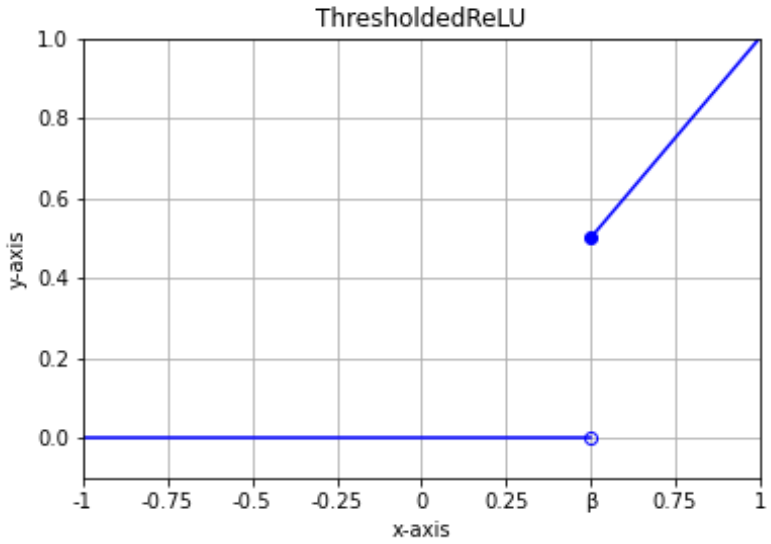
ReLU Sparsification in CNN Models

$$y = \max(0, x)$$

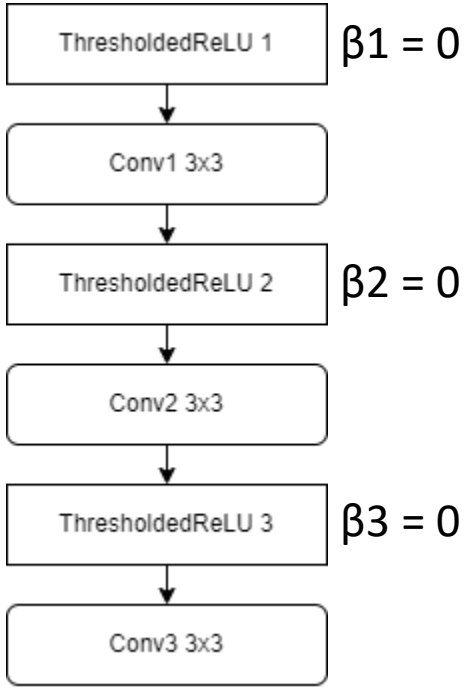
$$y = \begin{cases} x, x \geq \beta \\ 0, x < \beta \end{cases}$$



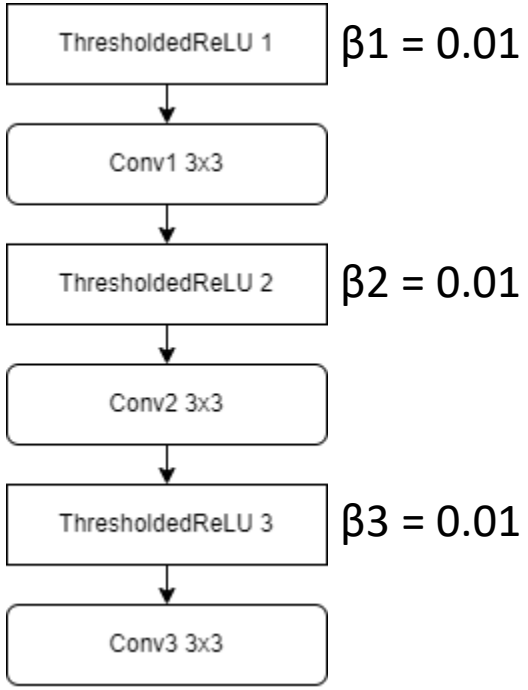




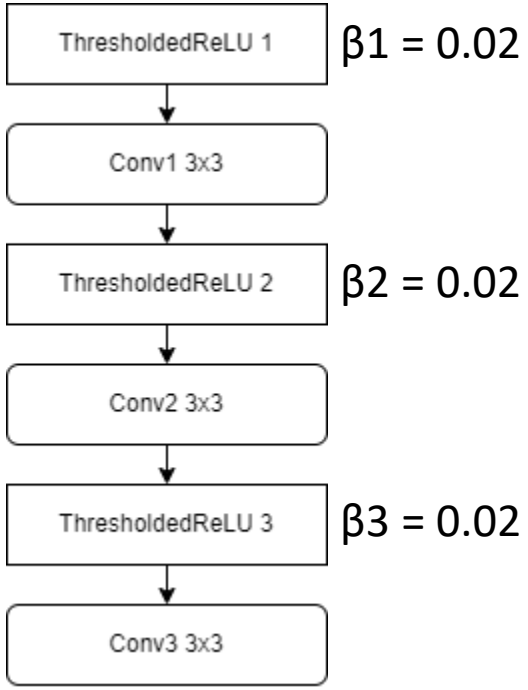
Uniformly increasing threshold for all layers



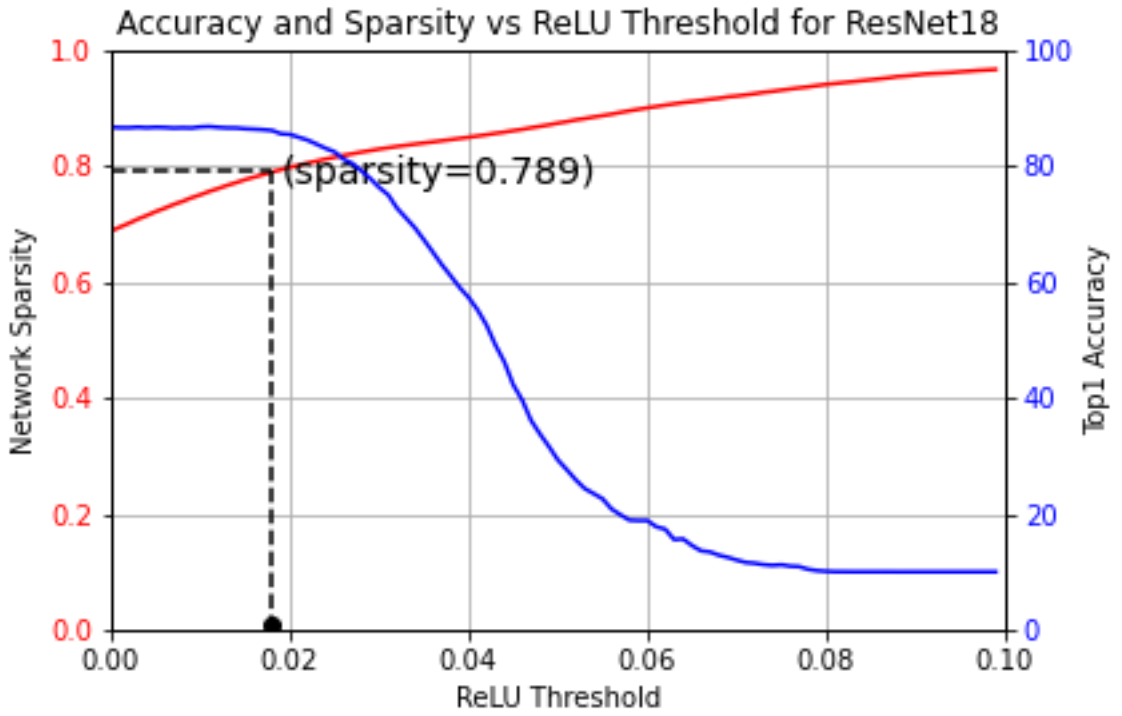
Uniformly increasing threshold for all layers

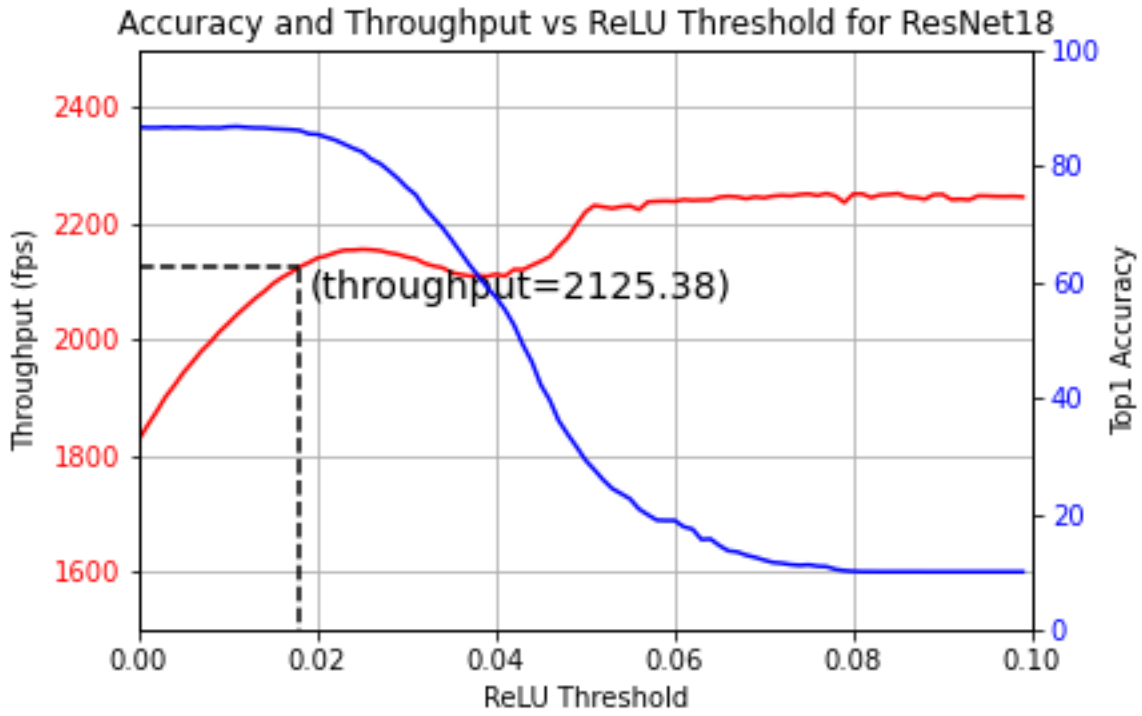


Uniformly increasing threshold for all layers



Uniformly increasing threshold for all layers

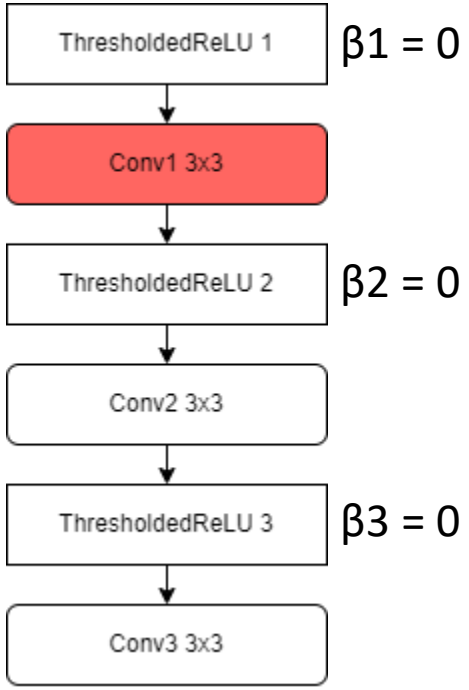




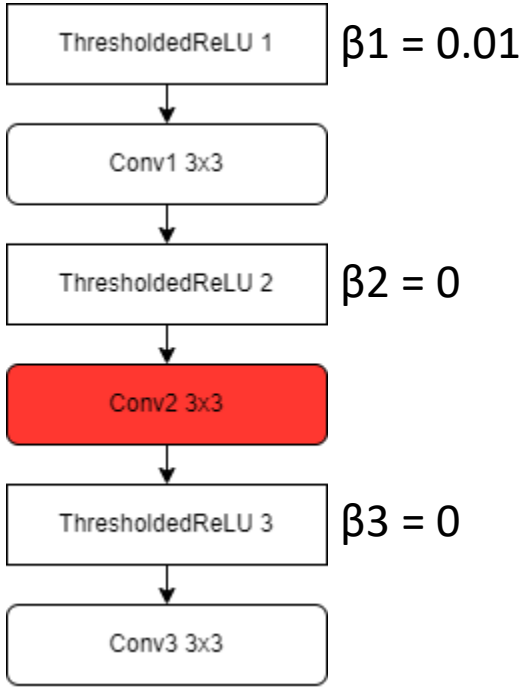
How can we mitigate the accuracy loss?

Throughput depends on the latency of the slowest node

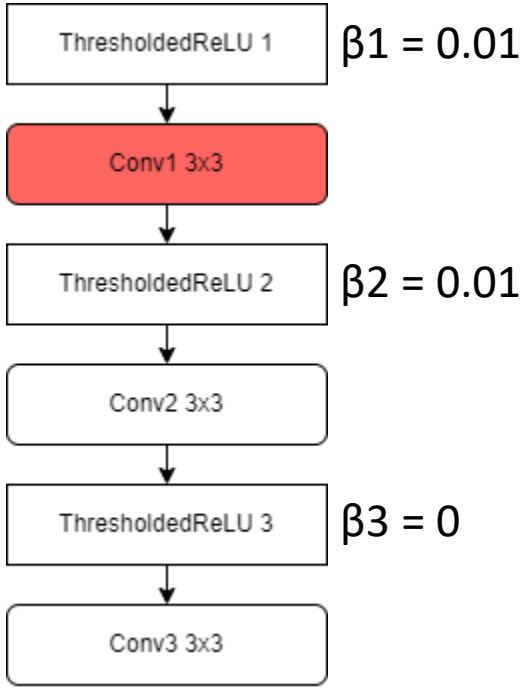
Hardware-Aware Thresholding



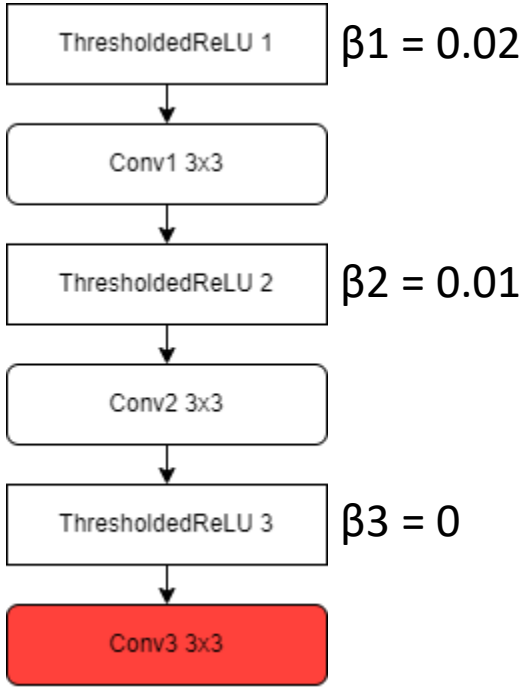
Hardware-Aware Thresholding



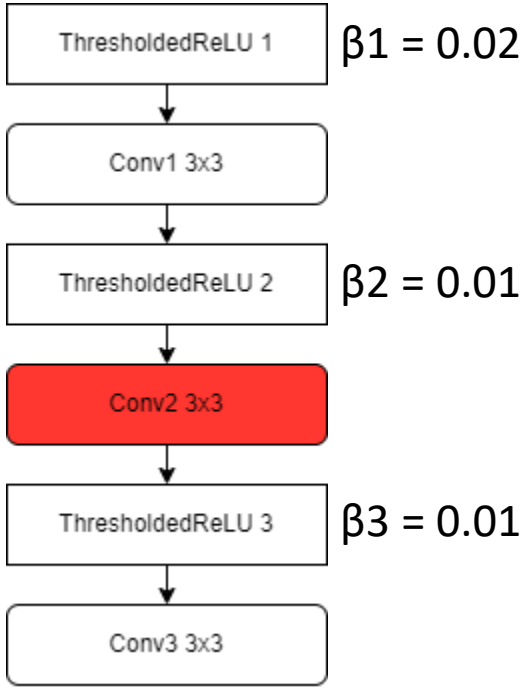
Hardware-Aware Thresholding



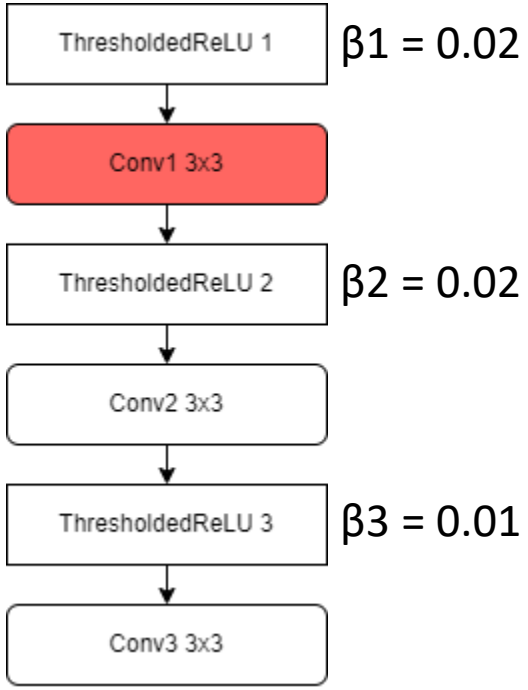
Hardware-Aware Thresholding



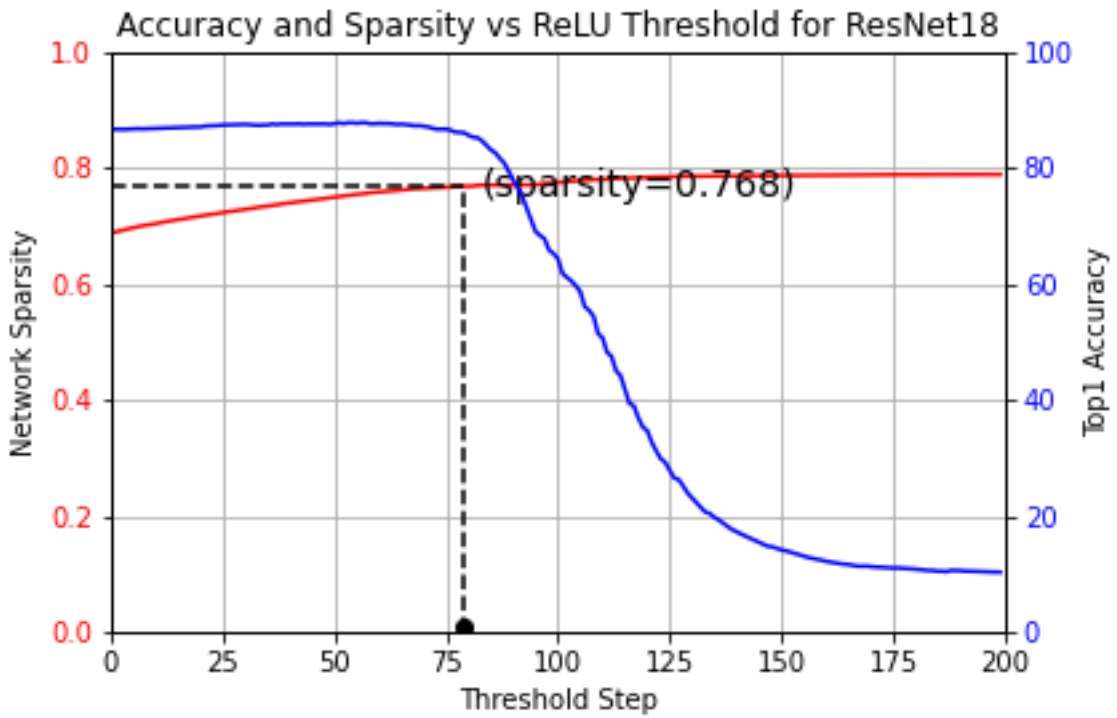
Hardware-Aware Thresholding

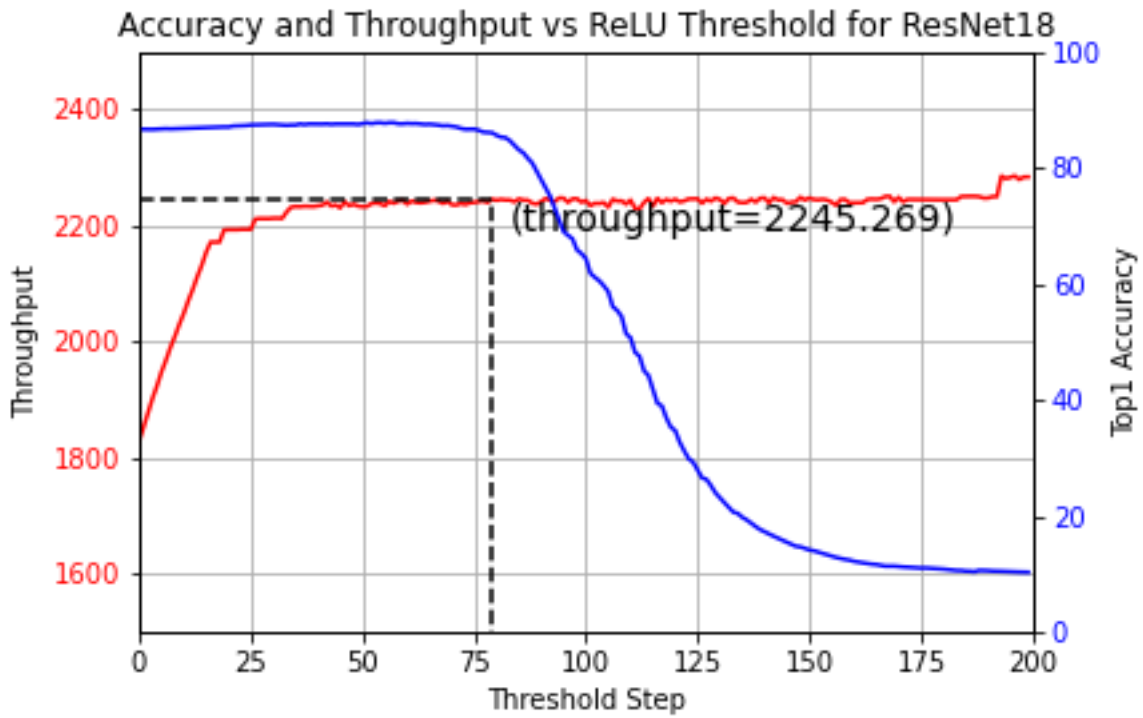


Hardware-Aware Thresholding



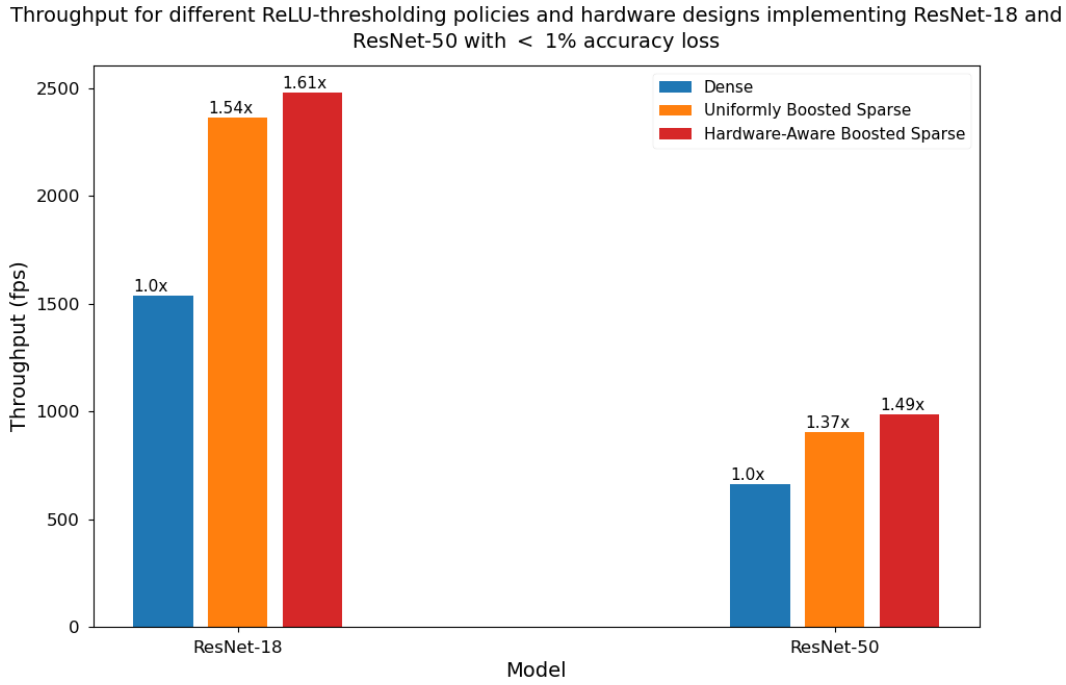
Hardware-Aware ReLU Sparsification in CNN Models





Evaluation

- Iteratively pushing the slowest node's sparsity provides more speedup for the same accuracy loss



Conclusion

Thank you for listening!

fpgaConvNet Website

ka720@ic.ac.uk