Southern Methodist University Physics Department

Convolutional Neural Networks Fast Inference Deployment on FPGAs

Andrew Reis (Undergraduate), Dr. Allison Deiana, Dr. Rohin Narayan, Dr. Stephen Sekula

Presentation Outline

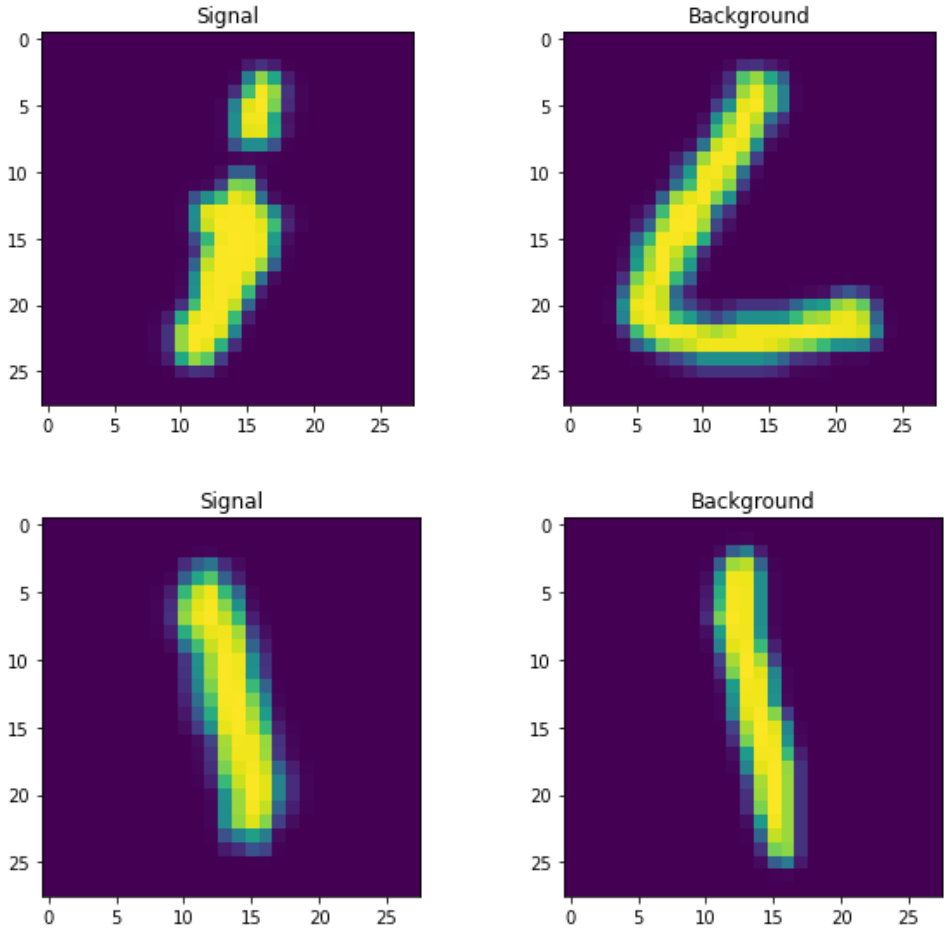
- Example CNN model use case and project goals.
- Baseline model performance and synthesis estimates.
- Quantization aware training (QAT) using qkeras for deployment optimization.
- Results after quantization and updating the configuration IOType.
- IO_Stream Conv2D implementation latency scaling with respect to layer input size.
- IO_Stream depthwise separable convolution vs regular 2D convolution.

Example Model and Project Goals

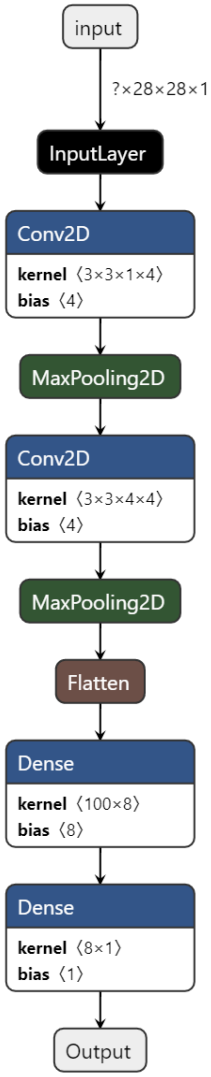
Binary Classification Task

Identify the letter `I` (signal) vs the letter `L` (background)

Data sourced from the EMNIST Dataset (https://arxiv.org/abs/1702.05373v1)



Model Architecture



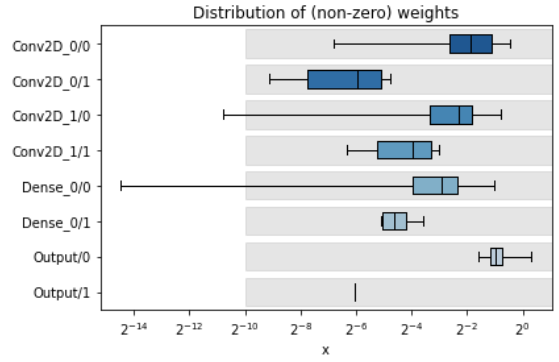
Project Goal

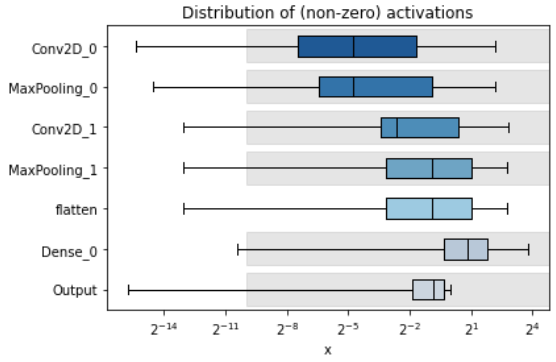
Deploy this CNN architecture to an FPGA using HLS4ML with a total latency of O(1 µs)

Baseline Model Performance

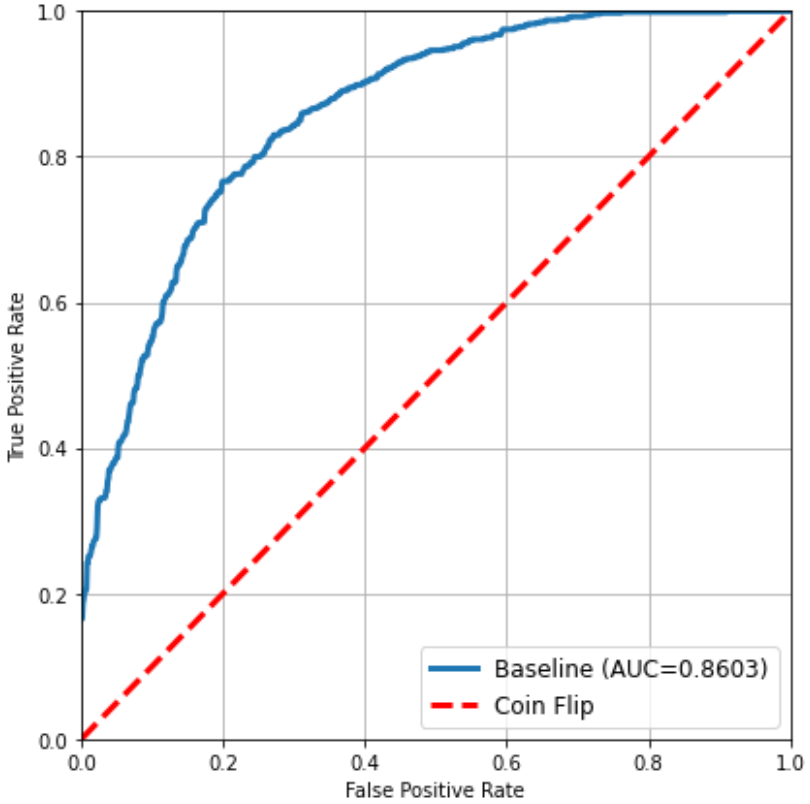
Synthesis Details

- Chip: Xilinx Kintex Ultrascale (xcku115-flvb2104-2-i)
- Clock Period: 5 ns
- Vivado HLS Version: 2020.1
- IOType: io_parallel
- Resource strategy with the lowest reuse factor for all layers
- Model datatype: ap_fixed<16,6>





Baseline Model ROC Curve

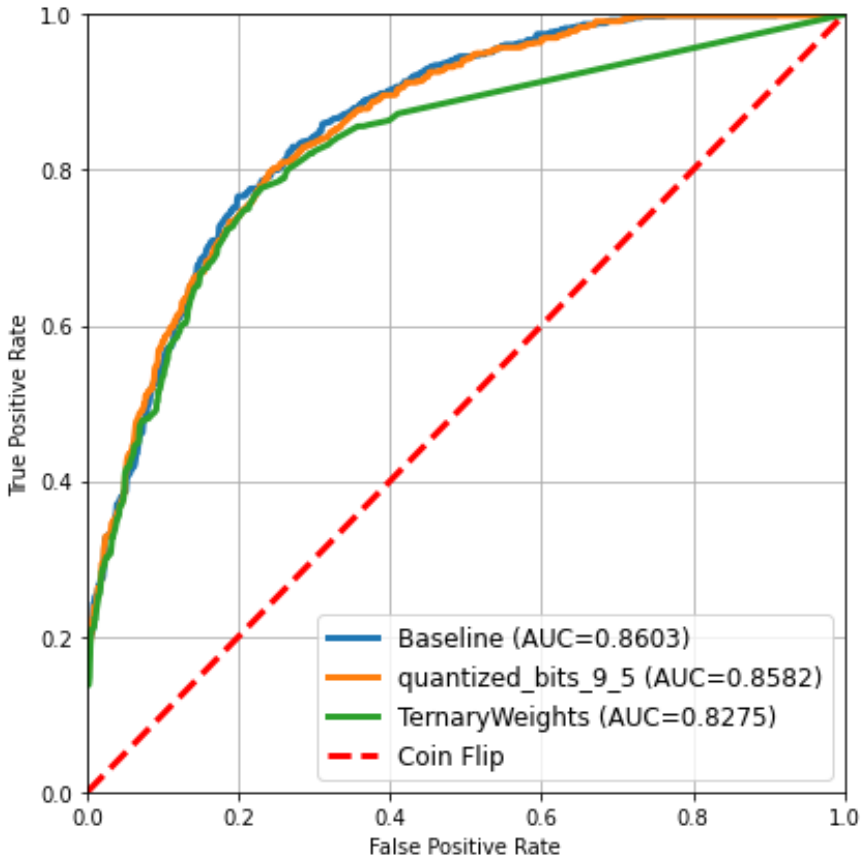


Synthesis Average Latency Estimate: 125 µs

Two orders of magnitude greater than where we want to be.

Quantization Aware Training (QAT)

ROC Curves for Quantized Models



- Two different quantization levels were used in model training.
  - quantized_bits(8,4)/ap_fixed<9,5>
  - Ternary weights (-1,0,1)
- While both performed similarly to the baseline, the ternary weight model experiences slight performance degradation due to the aggressive quantization.
- As such, the 9-bit fixed point model was chosen for deployment.

IO_Stream Synthesis Results

While quantization of the model helps, the majority of the heavy lifting in getting us down to our latency goal is done through the new streaming implementations [PR #220] of convolutional layers.

CSIM Metric Evaluation	
Binary Accuracy	AUC
78%	86%

CO-SIM Metric Evaluation	
Binary Accuracy	AUC
78%	86%

Co-Sim Latency Results

Latency (Clock Cycles)			Latency (Time)		
Min	Avg	Max	Min	Avg	Max
1580	1580	1580	7.9 µs	7.9 µs	7.9 µs

Conclusion: We are able to meet the project goal by synthesizing the CNN architecture with a latency of O(1 µs)

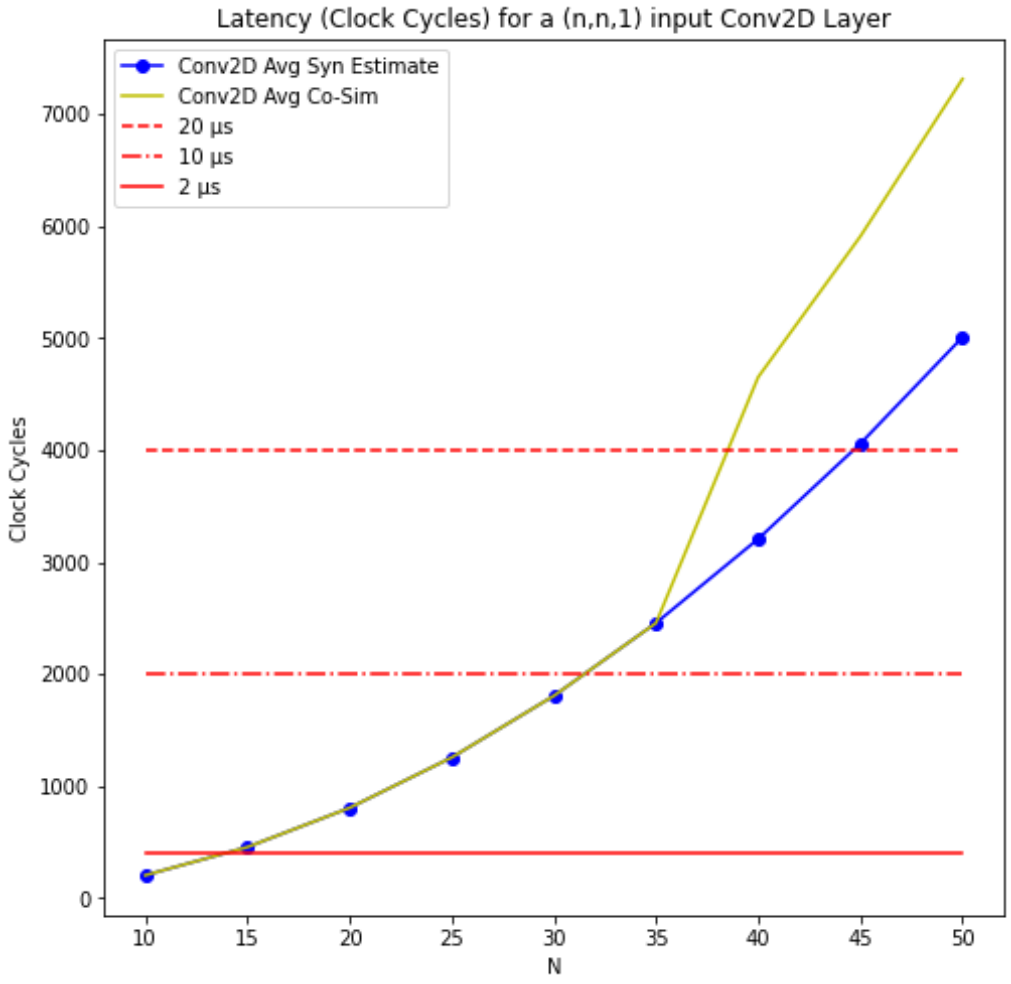
Synthesis Latency Estimates

Latency (cycles)		Latency (absolute)		Interval (cycles)		
min	max	min	max	min	max	Type
1596	1596	7.980 us	7.980 us	1576	1576	dataflow

Synthesis Area Estimates

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	-	0	32	-
FIFO	20	-	1093	1639	-
Instance	10	422	24018	46019	-
Memory	-	-	-	-	-
Multiplexer	-	-	-	36	-
Register	-	-	6	-	-
Total	30	422	25117	47726	0
Available	4320	5520	1326720	663360	0
Available SLR	2160	2760	663360	331680	0
Utilization (%)	~0	7	1	7	0
Utilization SLR (%)	1	15	3	14	100

IO_Stream Conv2D Latency Scaling



While we have shown that it is feasible to deploy one particular model with a O(1 µs) latency, it might be useful to others to see how convolutional layer latency scales with changing input/output sizes.

Synthesis Details

- Chip: Xilinx Kintex Ultrascale (xcku115-flvb2104-2-i)
- Clock Period: 5 ns
- Vivado HLS Version: 2020.1
- IOType: io_stream
- Layer datatype: ap_fixed<9,1>

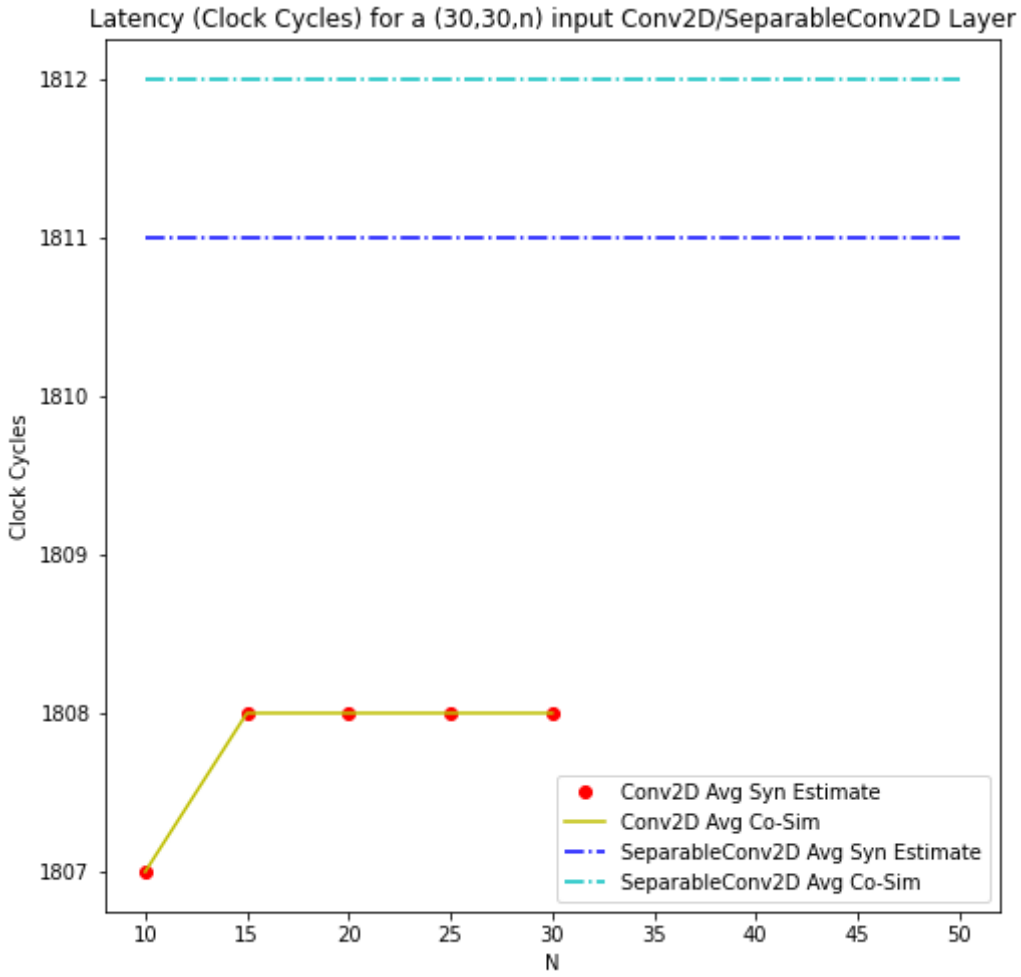
Layer Details

- Input: (n,n,1)
- Filters: 8

IO_Stream Depthwise Separable Convolution

The io_stream updates [PR #220] also come with support for depthwise separable convolution, which for 3x3 kernels can reduce the number of required multiplications by a factor of 8 or 9 (https://arxiv.org/abs/1610.02357).

However, this does not manifest in a latency reduction with its streaming implementation when compared to a standard convolutional layer. Instead, it does allow for higher input channel counts, which may prove to be extremely useful.



Synthesis Details

- Chip: Xilinx Kintex Ultrascale (xcku115-flvb2104-2-i)
- Clock Period: 5 ns
- Vivado HLS Version: 2020.1
- IOType: io_stream
- Layer datatype: ap_fixed<9,1>

Layer Details

- Input: (30,30,n)
- Filters: 8

Questions?

GitHub Code Repository

Acknowledgements

Vladimir Loncar – HLS4ML streaming convolutional layer implementations.