



Low-Power Neural Network Accelerators: Advancements in Custom Floating-Point Techniques

Yarib Nevarez

Universität Bremen

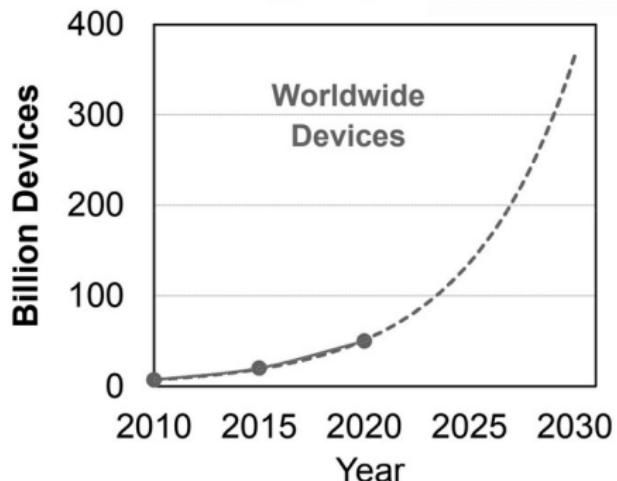
May 22, 2024

Introduction

Expansion of IoT and TinyML: Requirements, Accelerators, and Challenges

Internet-of-Things (IoT) in Smart Cities and Industry 4.0

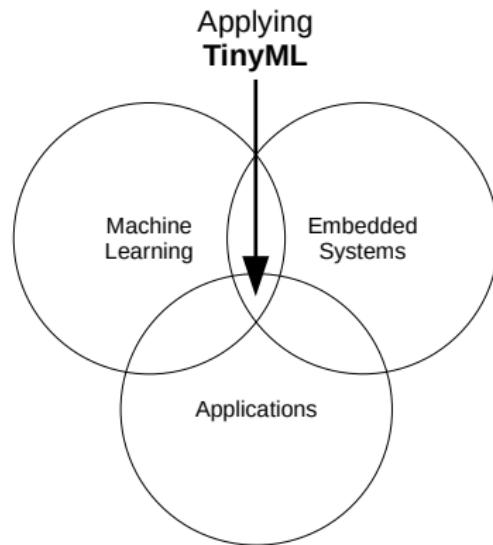
Any device Anybody Anywhere Any business Any network Anytime



Loh, Kou-Hung Lawrence. "1.2 Fertilizing AIoT from roots to leaves." In 2020 IEEE International Solid-State Circuits Conference-(ISSCC), pp. 15-21. IEEE, 2020.

Expansion of IoT and TinyML: Requirements, Accelerators, and Challenges

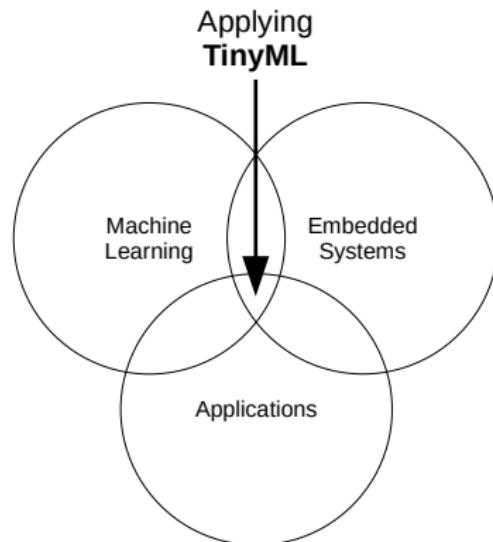
Internet-of-Things (IoT) in Smart Cities and Industry 4.0



Expansion of IoT and TinyML: Requirements, Accelerators, and Challenges

Internet-of-Things (IoT) in Smart Cities and Industry 4.0

Neural network accelerators



Expansion of IoT and TinyML: Requirements, Accelerators, and Challenges

Internet-of-Things (IoT) in Smart Cities and Industry 4.0

Neural network accelerators

Aspects for long-term sustainability:

Energy and resource efficiency

Quality preservation

Application versatility

Platform compatibility

On-device training

Expansion of IoT and TinyML: Requirements, Accelerators, and Challenges

Internet-of-Things (IoT) in Smart Cities and Industry 4.0

Neural network accelerators

Aspects for long-term sustainability:

Energy and resource efficiency

Quality preservation

Application versatility

Platform compatibility

On-device training

Expansion of IoT and TinyML: Requirements, Accelerators, and Challenges

Internet-of-Things (IoT) in Smart Cities and Industry 4.0

Neural network accelerators

Aspects for long-term sustainability:

Energy and resource efficiency

Quality preservation

Application versatility

Platform compatibility

On-device training

Expansion of IoT and TinyML: Requirements, Accelerators, and Challenges

Internet-of-Things (IoT) in Smart Cities and Industry 4.0

Neural network accelerators

Aspects for long-term sustainability:

Energy and resource efficiency

Quality preservation

Application versatility

Platform compatibility

On-device training

Expansion of IoT and TinyML: Requirements, Accelerators, and Challenges

Internet-of-Things (IoT) in Smart Cities and Industry 4.0

Neural network accelerators

Aspects for long-term sustainability:

Energy and resource efficiency

Quality preservation

Application versatility

Platform compatibility

On-device training

Expansion of IoT and TinyML: Requirements, Accelerators, and Challenges

Internet-of-Things (IoT) in Smart Cities and Industry 4.0

Neural network accelerators

Aspects for long-term sustainability:

Energy and resource efficiency

Quality preservation

Application versatility

Platform compatibility

On-device training

Expansion of IoT and TinyML: Requirements, Accelerators, and Challenges

Internet-of-Things (IoT) in Smart Cities and Industry 4.0

Neural network accelerators

Aspects for long-term sustainability:

Energy and resource efficiency

Quality preservation

Application versatility

Platform compatibility

On-device training

Expansion of IoT and TinyML: Requirements, Accelerators, and Challenges

Internet-of-Things (IoT) in Smart Cities and Industry 4.0

Neural network accelerators

Aspects for long-term sustainability:

Current state-of-the-art methods:

Energy and resource efficiency

Quality preservation

Application versatility

Platform compatibility

On-device training

Expansion of IoT and TinyML: Requirements, Accelerators, and Challenges

Internet-of-Things (IoT) in Smart Cities and Industry 4.0

Neural network accelerators

Aspects for long-term sustainability:

Energy and resource efficiency

Quality preservation

Application versatility

Platform compatibility

On-device training

Current state-of-the-art methods:

Extreme quantization

- Complex problems
- Mission-critical applications

Expansion of IoT and TinyML: Requirements, Accelerators, and Challenges

Internet-of-Things (IoT) in Smart Cities and Industry 4.0

Neural network accelerators

Aspects for long-term sustainability:

Energy and resource efficiency

Quality preservation

Application versatility

Platform compatibility

On-device training

Current state-of-the-art methods:

Extreme quantization

-  Complex problems
-  Mission-critical applications

Expansion of IoT and TinyML: Requirements, Accelerators, and Challenges

Internet-of-Things (IoT) in Smart Cities and Industry 4.0

Neural network accelerators

Aspects for long-term sustainability:

Energy and resource efficiency

Quality preservation

Application versatility

Platform compatibility

On-device training

Current state-of-the-art methods:

Extreme quantization

- ✗ Complex problems
- ✗ Mission-critical applications

Fixed precision

- On-device training

Expansion of IoT and TinyML: Requirements, Accelerators, and Challenges

Internet-of-Things (IoT) in Smart Cities and Industry 4.0

Neural network accelerators

Aspects for long-term sustainability:

Energy and resource efficiency

Quality preservation

Application versatility

Platform compatibility

On-device training

Current state-of-the-art methods:

Extreme quantization

-  Complex problems
-  Mission-critical applications

Fixed precision

-  On-device training

Goal and Objectives

- **Goal:** Acceleration techniques with efficient and high-quality inference and iterative optimization

Goal and Objectives

- **Goal:** Acceleration techniques with efficient and high-quality inference and iterative optimization
- **Objectives:**

Goal and Objectives

- **Goal:** Acceleration techniques with efficient and high-quality inference and iterative optimization
- **Objectives:**
 - Investigate design techniques

Goal and Objectives

- **Goal:** Acceleration techniques with efficient and high-quality inference and iterative optimization
- **Objectives:**
 - Investigate design techniques
 - Ensure quality preservation

Goal and Objectives

- **Goal:** Acceleration techniques with efficient and high-quality inference and iterative optimization
- **Objectives:**
 - Investigate design techniques
 - Ensure quality preservation
 - Explore on-device optimization

Goal and Objectives

- **Goal:** Acceleration techniques with efficient and high-quality inference and iterative optimization
- **Objectives:**
 - Investigate design techniques
 - Ensure quality preservation
 - Explore on-device optimization
 - Evaluate deployment

Outline

1 Methodology

2 Floating-Point Multiply-Accumulate Unit

3 Case Studies

4 Conclusions

1 Methodology

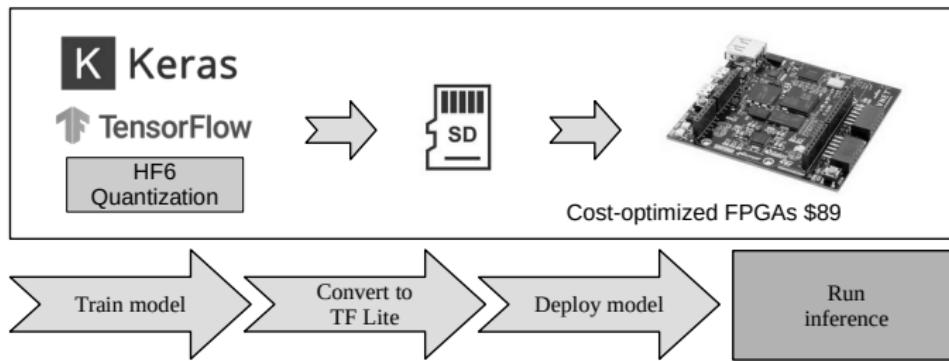
2 Floating-Point Multiply-Accumulate Unit

3 Case Studies

4 Conclusions

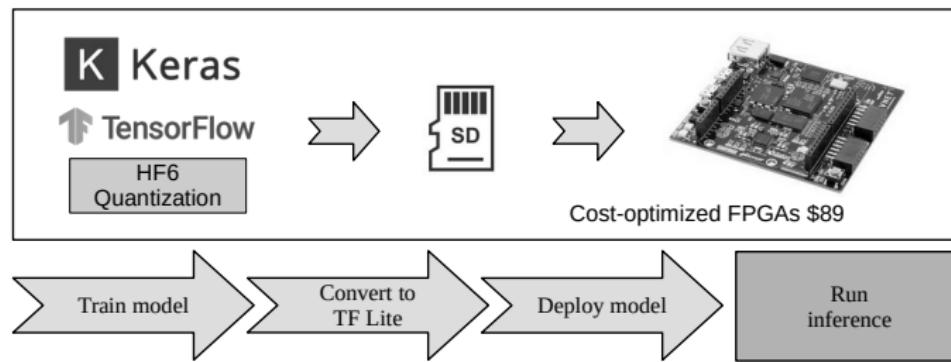
Methodology

Trans-Precision Neural Network Deployment for Low-Power Embedded Systems



Methodology

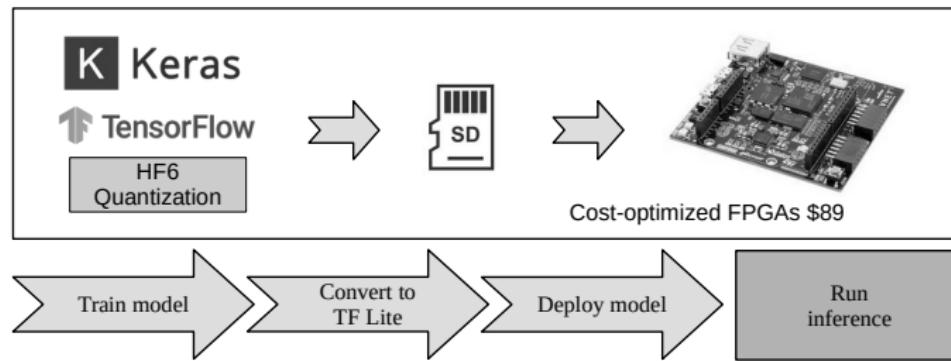
Trans-Precision Neural Network Deployment for Low-Power Embedded Systems



- TinyML

Methodology

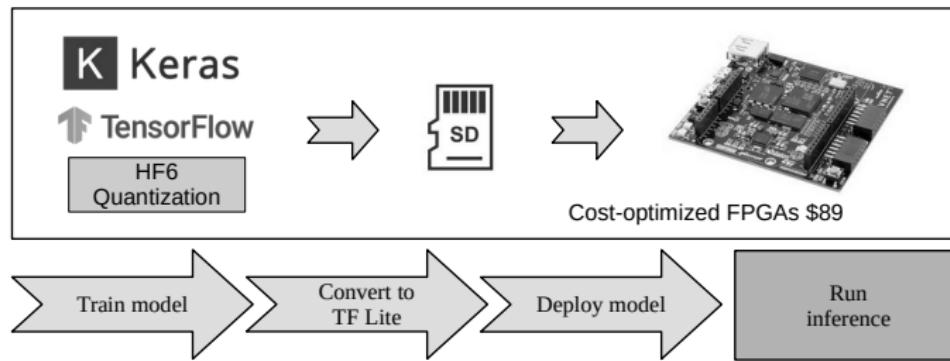
Trans-Precision Neural Network Deployment for Low-Power Embedded Systems



- TinyML
- Floating-point acceleration

Methodology

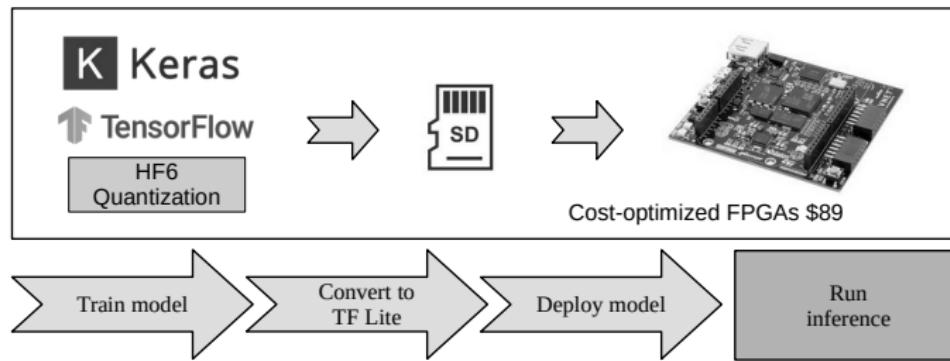
Trans-Precision Neural Network Deployment for Low-Power Embedded Systems



- TinyML
- Floating-point acceleration
- Resource-constrained FPGAs

Methodology

Trans-Precision Neural Network Deployment for Low-Power Embedded Systems



- TinyML
- Floating-point acceleration
- Resource-constrained FPGAs
- Research and general purpose

Trans-Precision Neural Network Deployment for Low-Power Embedded Systems

Abstraction levels:

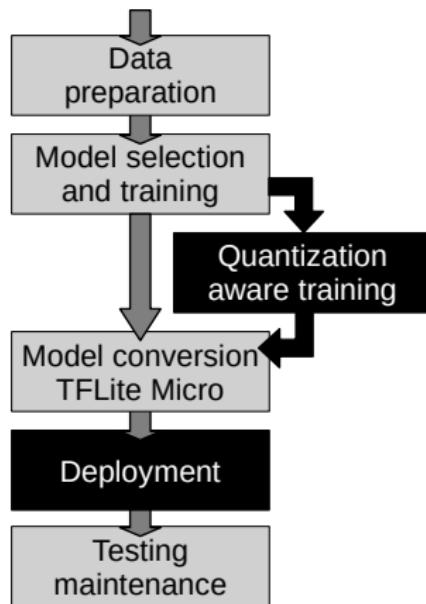
1. Model deployment
2. System infrastructure
3. Streamlined acceleration
4. Optimized processing

Methodology

Trans-Precision Neural Network Deployment for Low-Power Embedded Systems

Abstraction levels:

1. Model deployment
2. System infrastructure
3. Streamlined acceleration
4. Optimized processing

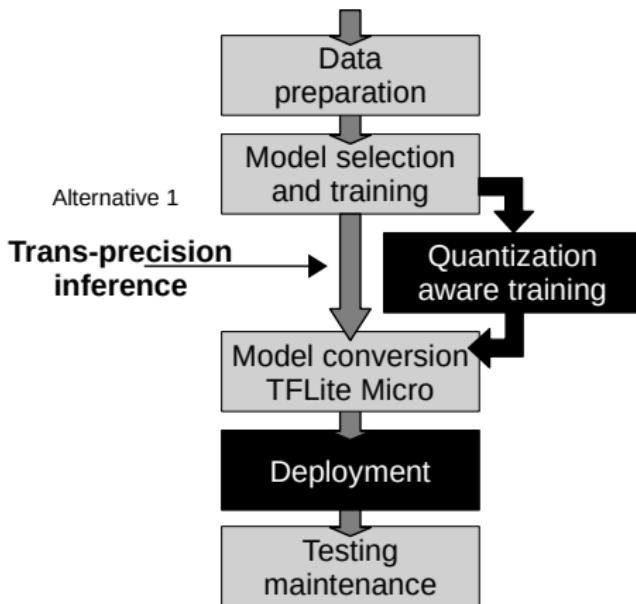


Methodology

Trans-Precision Neural Network Deployment for Low-Power Embedded Systems

Abstraction levels:

1. Model deployment
2. System infrastructure
3. Streamlined acceleration
4. Optimized processing

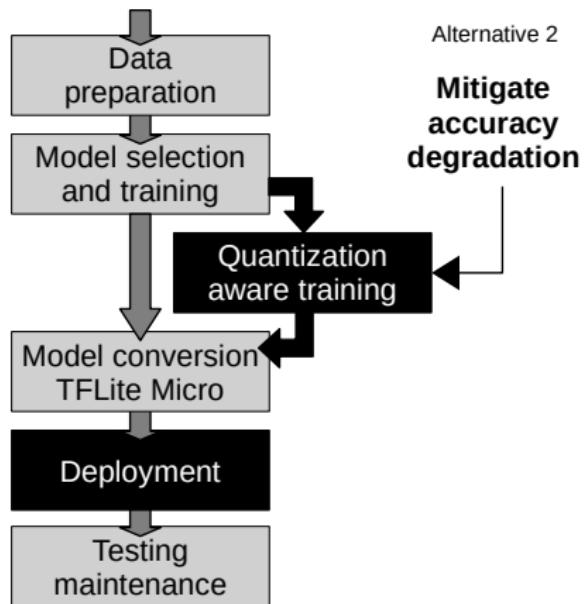


Methodology

Trans-Precision Neural Network Deployment for Low-Power Embedded Systems

Abstraction levels:

1. Model deployment
2. System infrastructure
3. Streamlined acceleration
4. Optimized processing

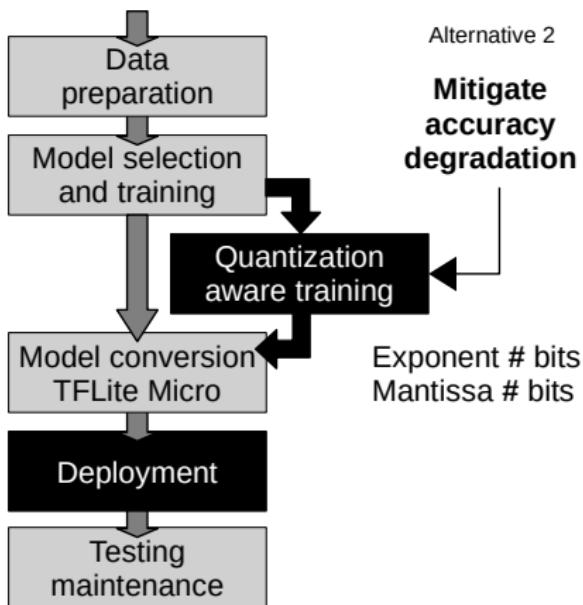


Methodology

Trans-Precision Neural Network Deployment for Low-Power Embedded Systems

Abstraction levels:

1. Model deployment
2. System infrastructure
3. Streamlined acceleration
4. Optimized processing



Trans-Precision Neural Network Deployment for Low-Power Embedded Systems

Abstraction levels:

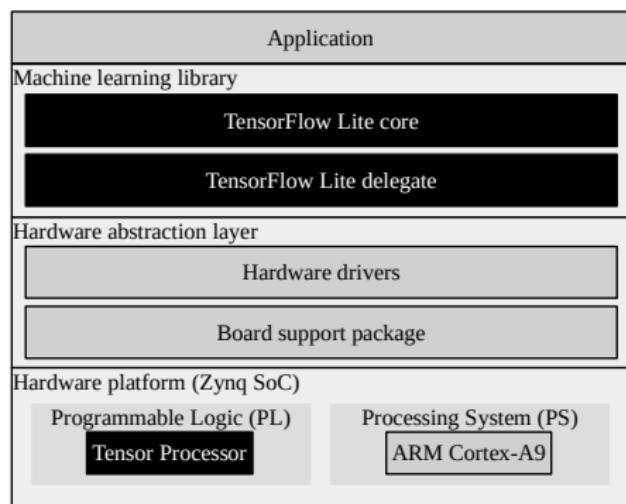
1. Model deployment

2. System infrastructure

3. Streamlined acceleration

4. Optimized processing

HW/SW co-design framework



Methodology

Trans-Precision Neural Network Deployment for Low-Power Embedded Systems

Abstraction levels:

1. Model deployment

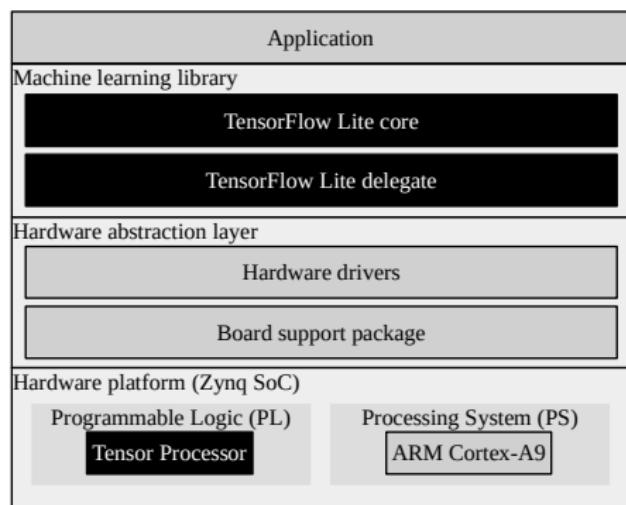
2. System infrastructure

3. Streamlined acceleration

4. Optimized processing

Industry standard framework

HW/SW co-design framework



Methodology

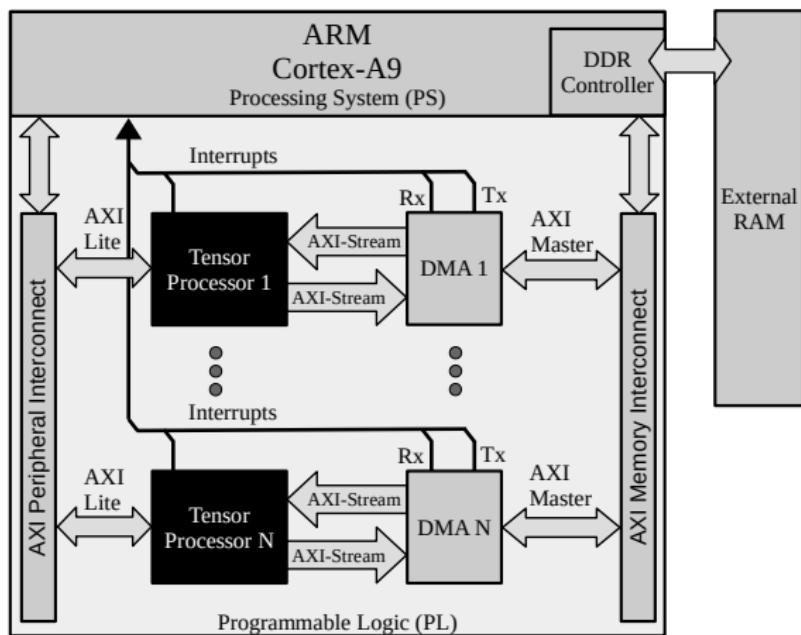
Trans-Precision Neural Network Deployment for Low-Power Embedded Systems

Abstraction levels:

1. Model deployment
2. System infrastructure

3. Streamlined acceleration

4. Optimized processing

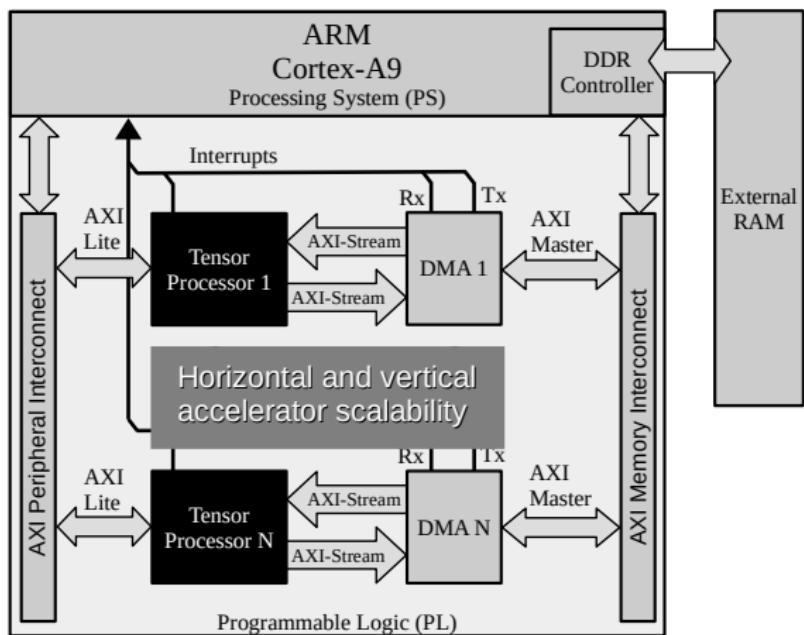


Methodology

Trans-Precision Neural Network Deployment for Low-Power Embedded Systems

Abstraction levels:

1. Model deployment
2. System infrastructure
- 3. Streamlined acceleration**
4. Optimized processing



Methodology

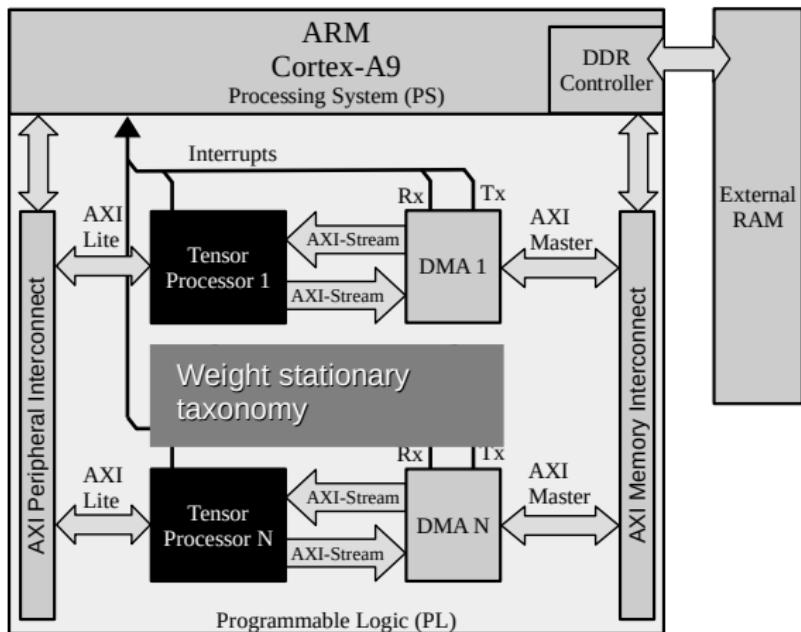
Trans-Precision Neural Network Deployment for Low-Power Embedded Systems

Abstraction levels:

1. Model deployment
2. System infrastructure

3. Streamlined acceleration

4. Optimized processing

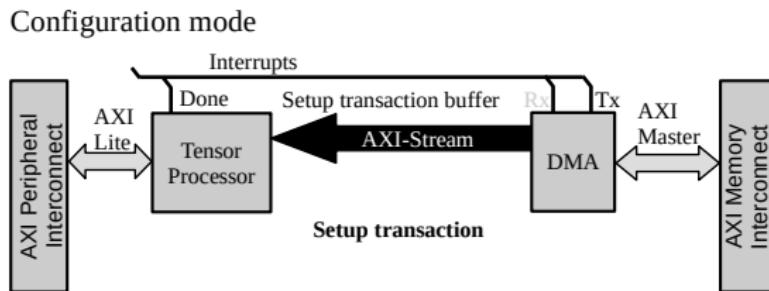


Methodology

Trans-Precision Neural Network Deployment for Low-Power Embedded Systems

Abstraction levels:

1. Model deployment
2. System infrastructure
- 3. Streamlined acceleration**
4. Optimized processing

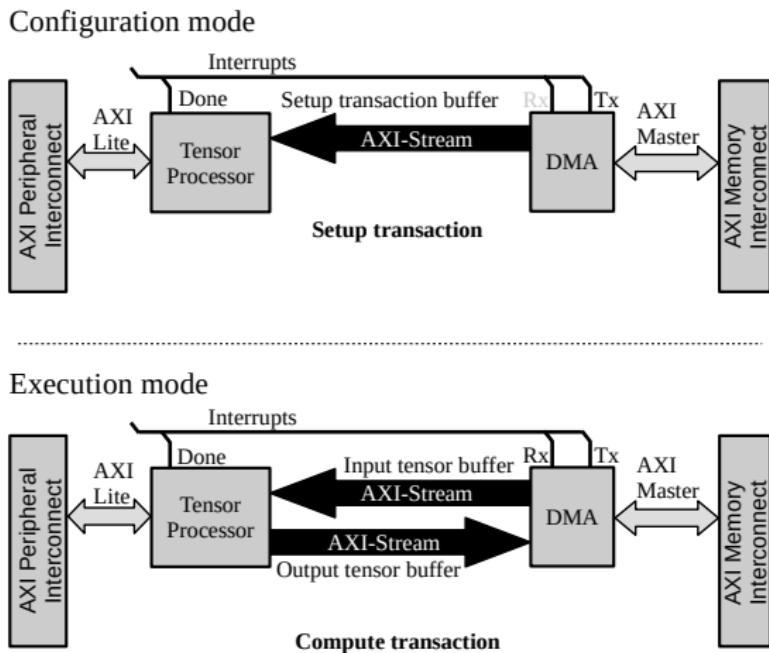


Methodology

Trans-Precision Neural Network Deployment for Low-Power Embedded Systems

Abstraction levels:

1. Model deployment
2. System infrastructure
- 3. Streamlined acceleration**
4. Optimized processing



Methodology

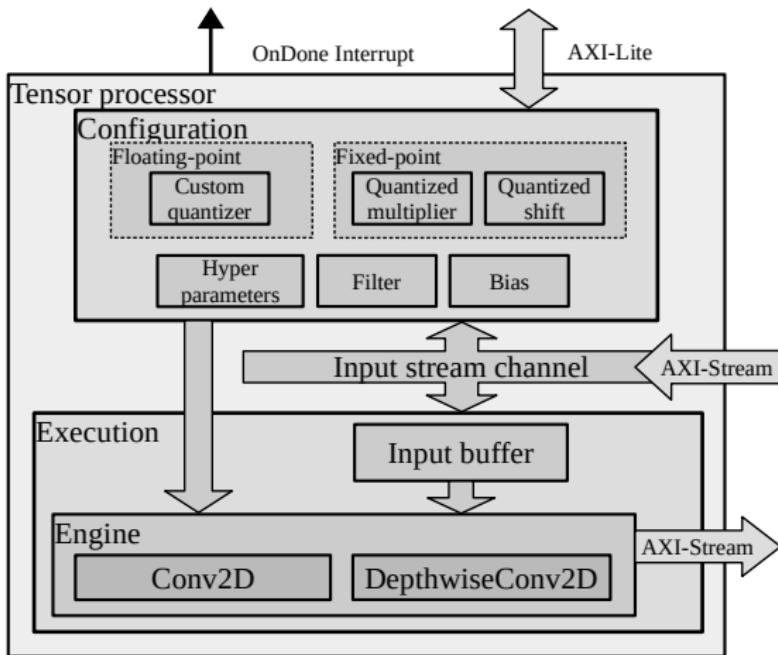
Trans-Precision Neural Network Deployment for Low-Power Embedded Systems

Abstraction levels:

1. Model deployment
2. System infrastructure

3. Streamlined acceleration

4. Optimized processing



Methodology

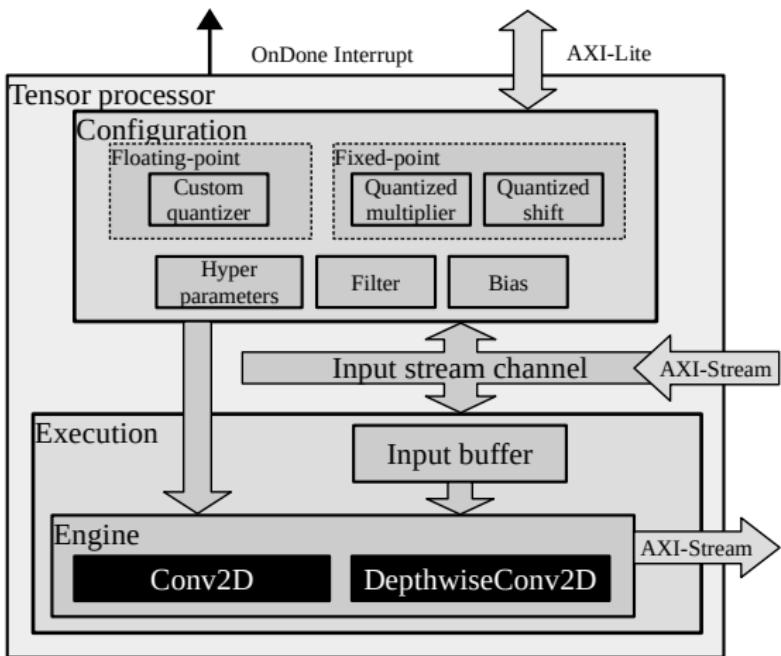
Trans-Precision Neural Network Deployment for Low-Power Embedded Systems

Abstraction levels:

1. Model deployment
2. System infrastructure

3. Streamlined acceleration

4. Optimized processing



Methodology

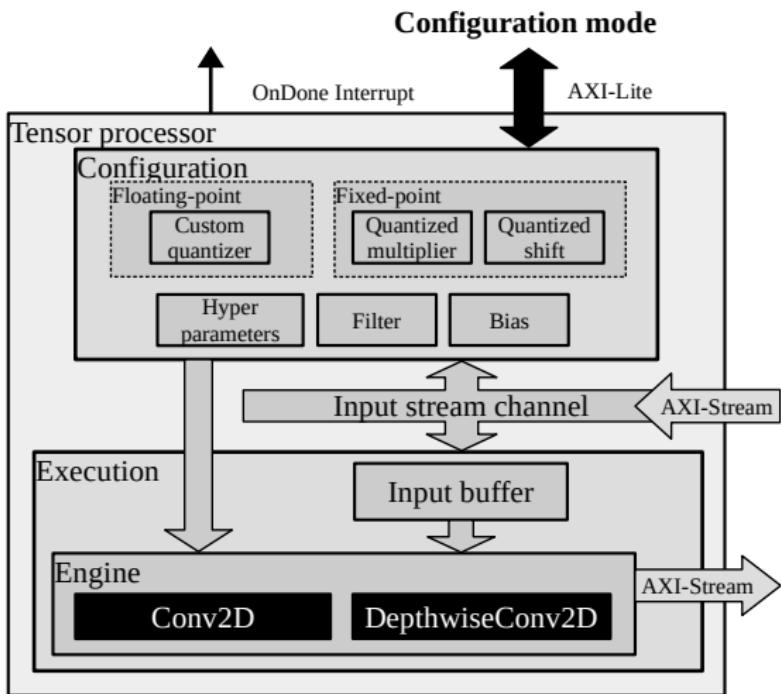
Trans-Precision Neural Network Deployment for Low-Power Embedded Systems

Abstraction levels:

1. Model deployment
2. System infrastructure

3. Streamlined acceleration

4. Optimized processing



Methodology

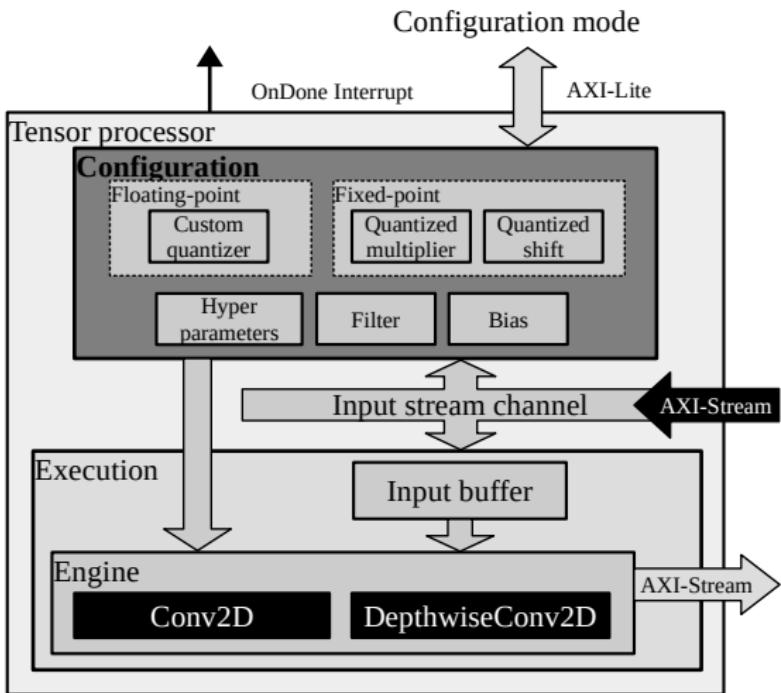
Trans-Precision Neural Network Deployment for Low-Power Embedded Systems

Abstraction levels:

1. Model deployment
2. System infrastructure

3. Streamlined acceleration

4. Optimized processing



Methodology

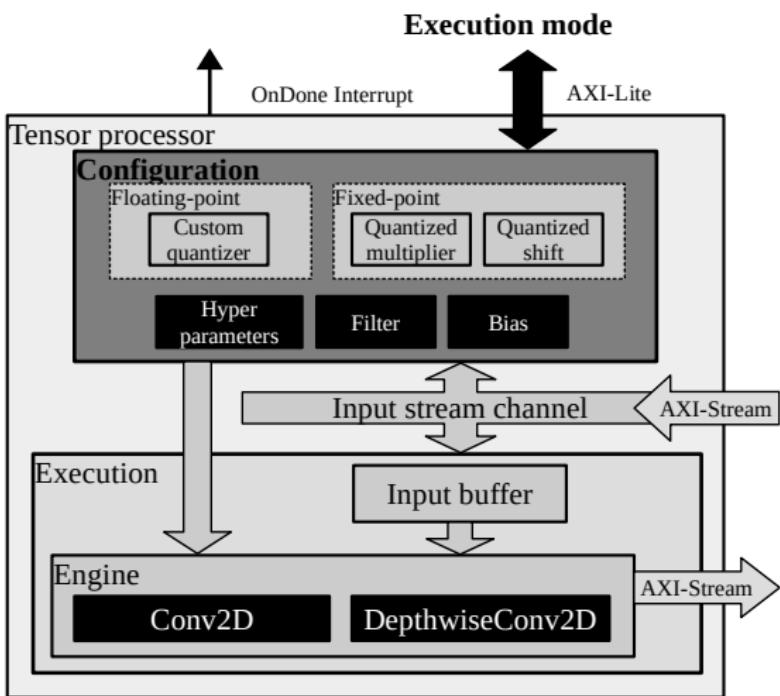
Trans-Precision Neural Network Deployment for Low-Power Embedded Systems

Abstraction levels:

1. Model deployment
2. System infrastructure

3. Streamlined acceleration

4. Optimized processing



Methodology

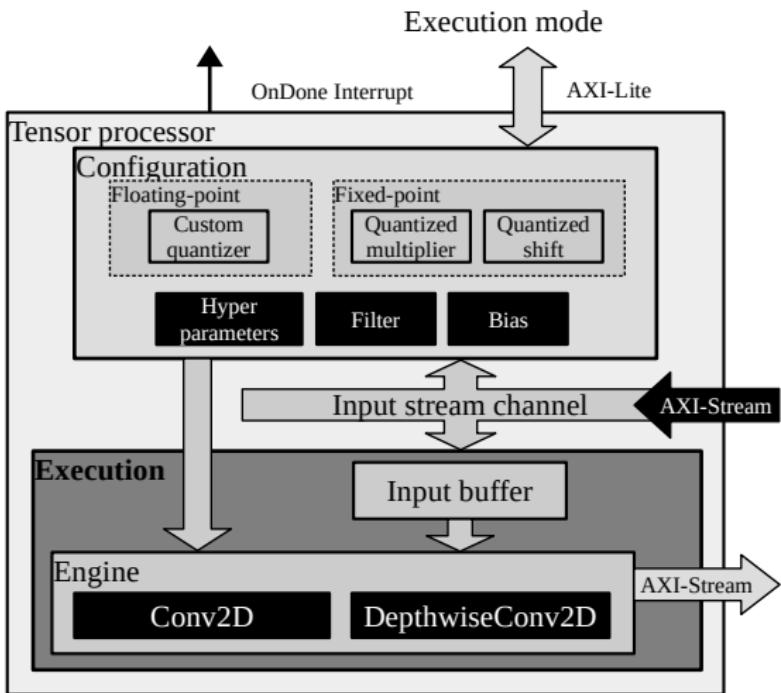
Trans-Precision Neural Network Deployment for Low-Power Embedded Systems

Abstraction levels:

1. Model deployment
2. System infrastructure

3. Streamlined acceleration

4. Optimized processing



Methodology

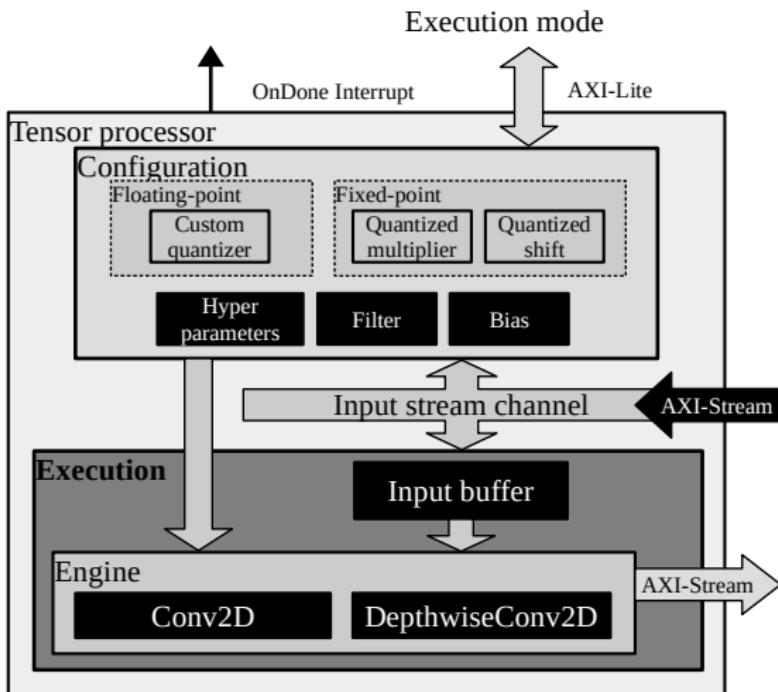
Trans-Precision Neural Network Deployment for Low-Power Embedded Systems

Abstraction levels:

1. Model deployment
2. System infrastructure

3. Streamlined acceleration

4. Optimized processing



Methodology

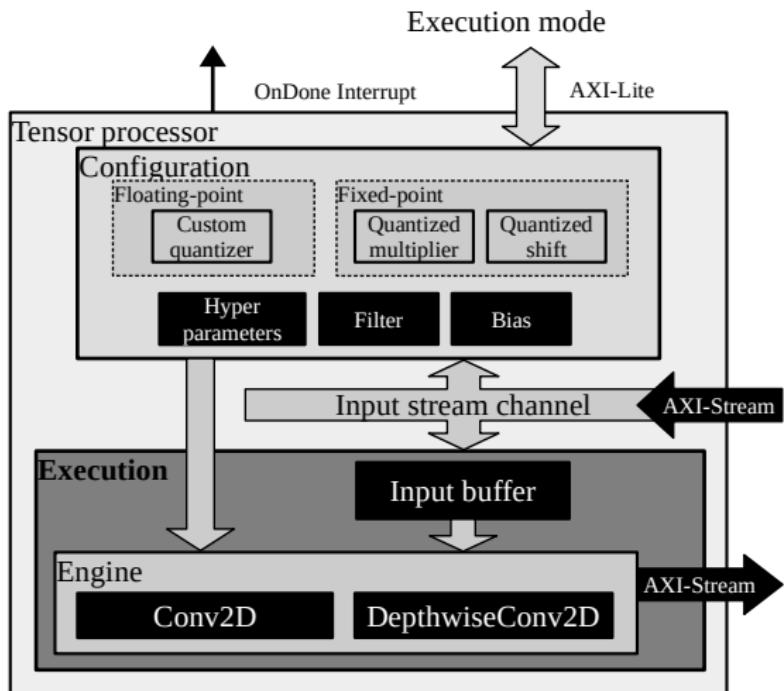
Trans-Precision Neural Network Deployment for Low-Power Embedded Systems

Abstraction levels:

1. Model deployment
2. System infrastructure

3. Streamlined acceleration

4. Optimized processing



Methodology

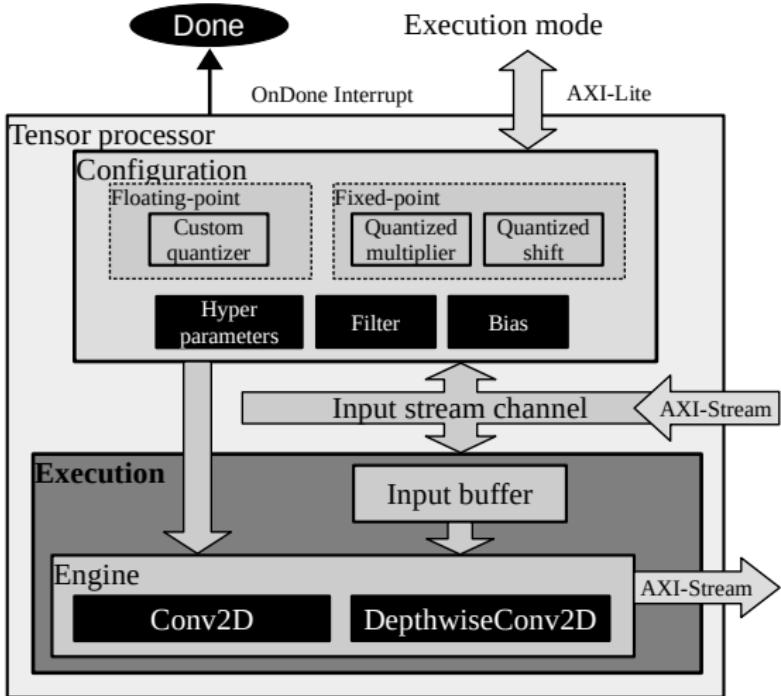
Trans-Precision Neural Network Deployment for Low-Power Embedded Systems

Abstraction levels:

1. Model deployment
2. System infrastructure

3. Streamlined acceleration

4. Optimized processing

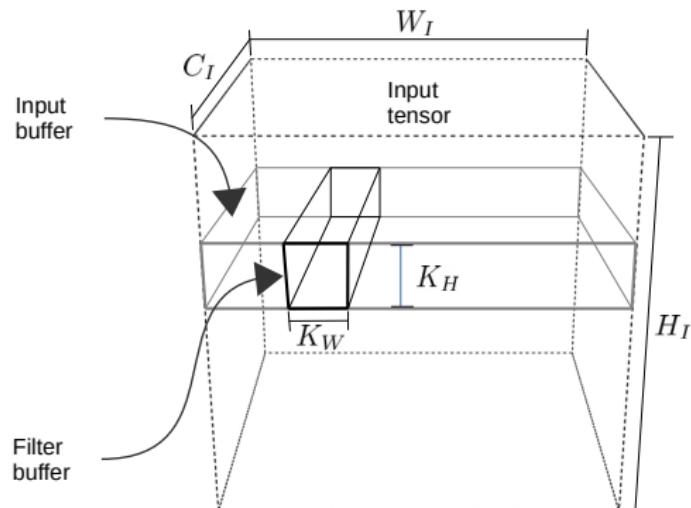


Methodology

Trans-Precision Neural Network Deployment for Low-Power Embedded Systems

Abstraction levels:

1. Model deployment
2. System infrastructure
3. Streamlined acceleration
4. Optimized processing



$$Conv2D(W, b, h)_{i,j,o} = \sum_{k,l,m} h_{(i+k, j+l, m)} W_{(o,k,l,m)} + b_o$$

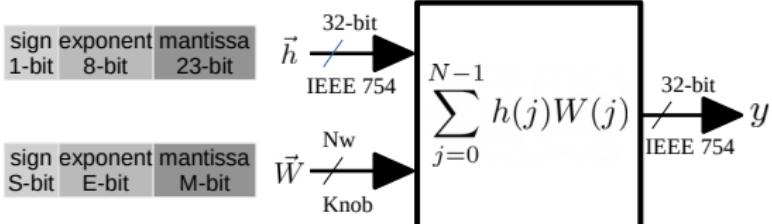
Methodology

Trans-Precision Neural Network Deployment for Low-Power Embedded Systems

Abstraction levels:

Multiply-accumulate unit
Hybrid custom floating-point computation

1. Model deployment



2. System infrastructure

3. Streamlined acceleration

4. Optimized processing

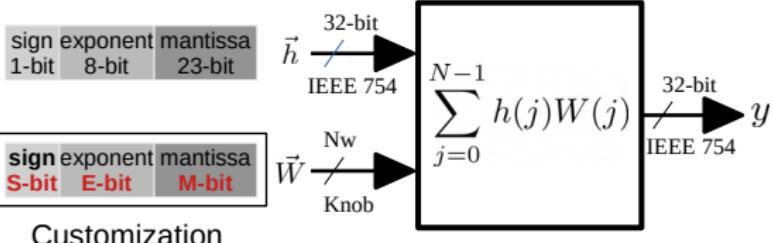
Methodology

Trans-Precision Neural Network Deployment for Low-Power Embedded Systems

Abstraction levels:

Multiply-accumulate unit
Hybrid custom floating-point computation

1. Model deployment



2. System infrastructure
3. Streamlined acceleration

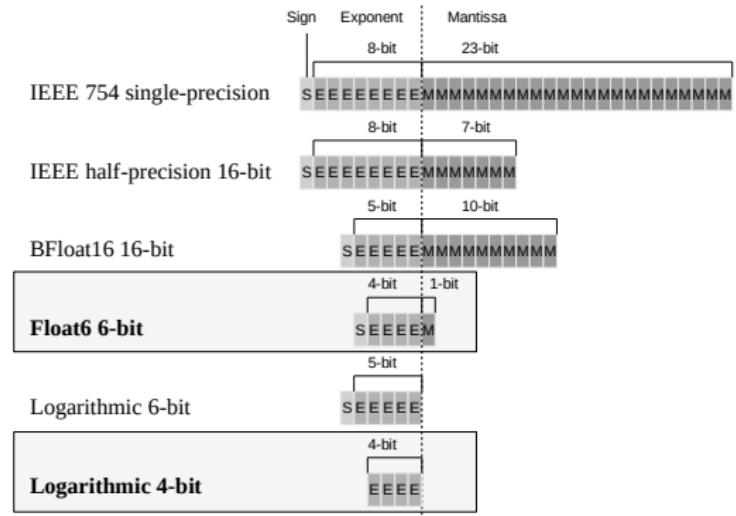
4. Optimized processing

Methodology

Trans-Precision Neural Network Deployment for Low-Power Embedded Systems

Abstraction levels:

1. Model deployment
2. System infrastructure
3. Streamlined acceleration
4. Optimized processing



1 Methodology

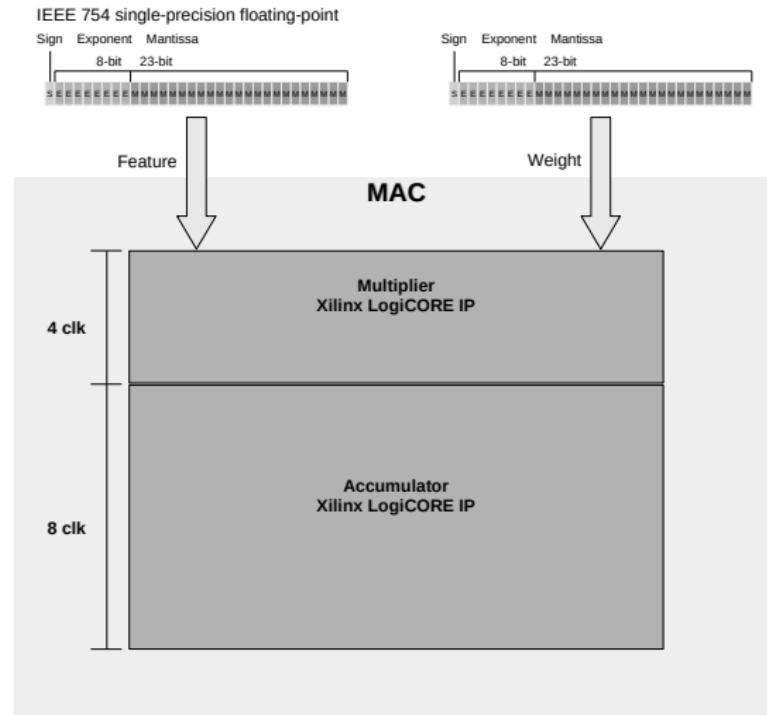
2 Floating-Point Multiply-Accumulate Unit

3 Case Studies

4 Conclusions

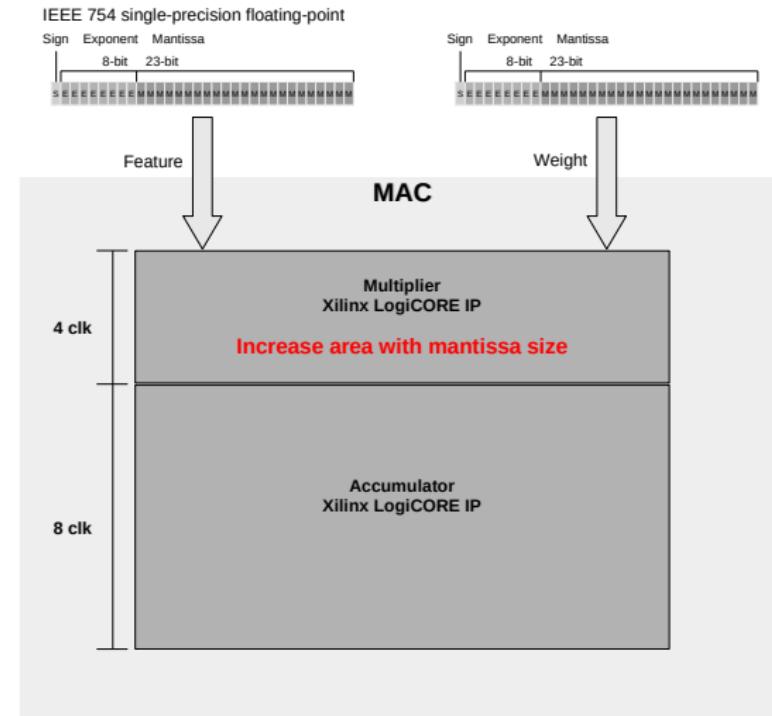
Standard Floating-Point MAC Design

Floating-Point Multiply-Accumulate Unit: Assessing Non-Optimized Operations



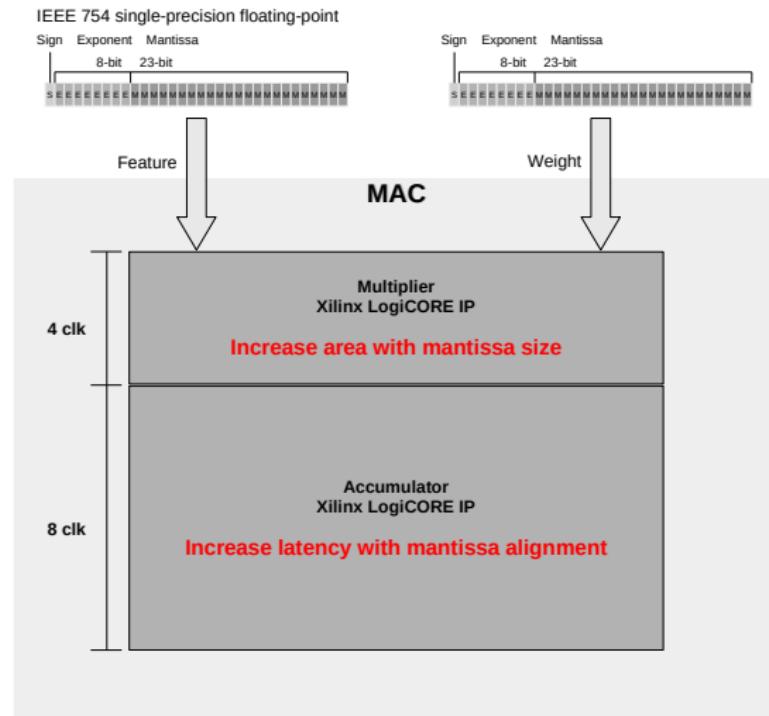
Standard Floating-Point MAC Design

Floating-Point Multiply-Accumulate Unit: Assessing Non-Optimized Operations



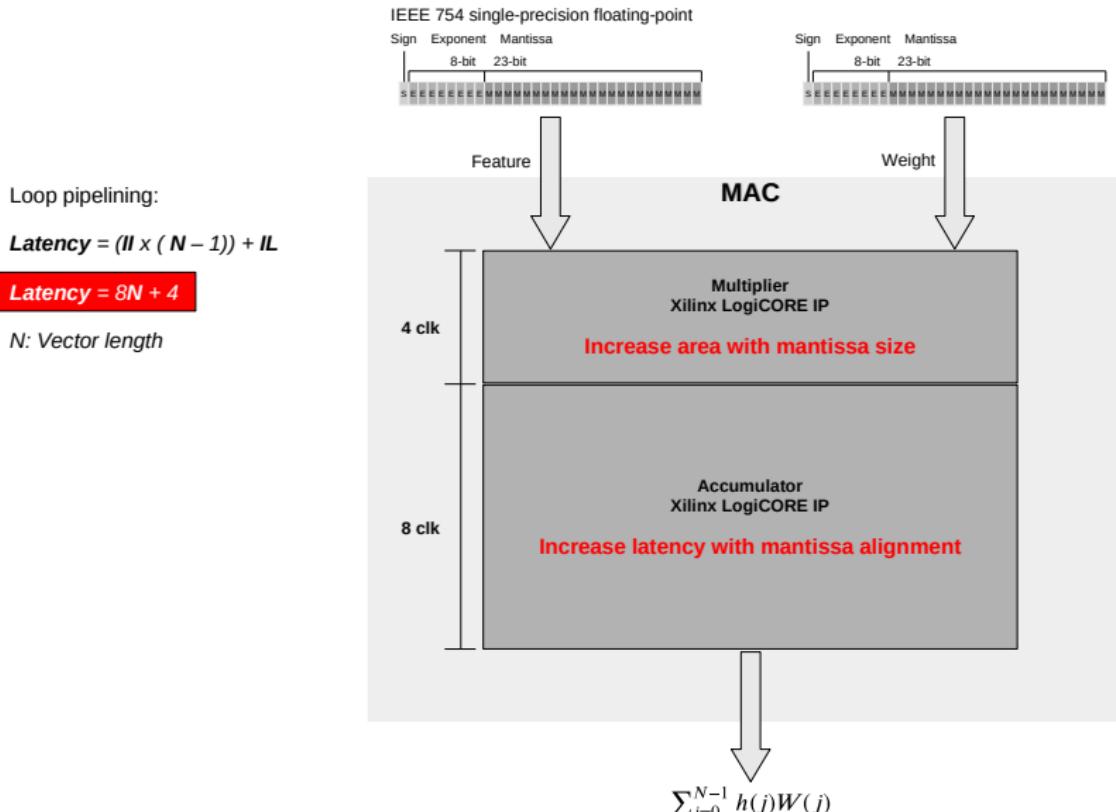
Standard Floating-Point MAC Design

Floating-Point Multiply-Accumulate Unit: Assessing Non-Optimized Operations



Standard Floating-Point MAC Design

Floating-Point Multiply-Accumulate Unit: Assessing Non-Optimized Operations



Standard Floating-Point MAC Design

Floating-Point Multiply-Accumulate Unit: Assessing Non-Optimized Operations

IEEE 754 single-precision floating-point
Sign Exponent Mantissa
8-bit 23-bit

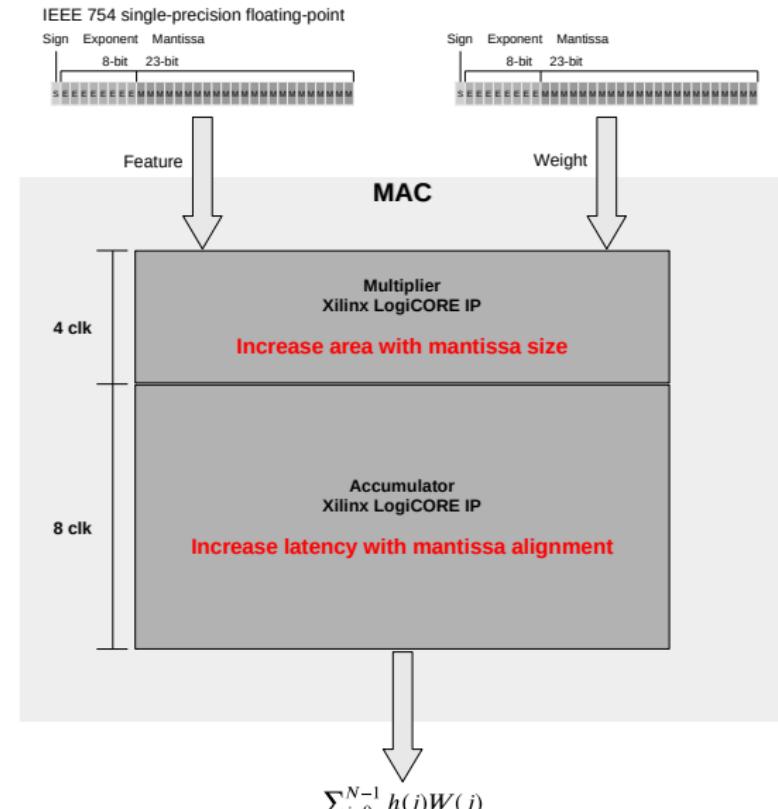
Loop pipelining:

$$\text{Latency} = (lL \times (N - 1)) + IL$$

$$\boxed{\text{Latency} = 8N + 4}$$

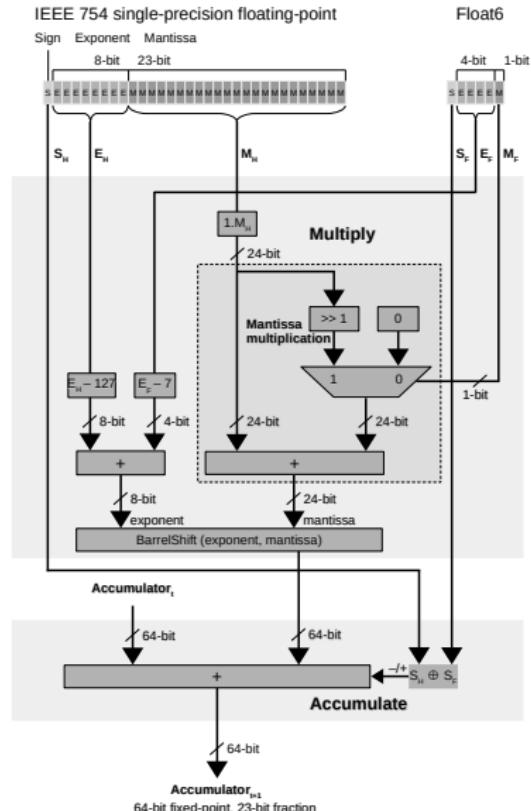
N : Vector length

Infinity and NaN: Not expected



Custom Floating-Point MAC Designs

Hybrid Custom Floating-Point Multiply-Accumulate Unit



Loop pipelining:

$$\text{Latency} = (Il \times (N - 1)) + IL$$

$$\boxed{\text{Latency} = N + 7}$$

Infinity and NaN: Not expected

Subnormal numbers are approximated as normal to simplify hardware design.

When $E_F = 0$ and $M_F = 1$
Approximation error = 3.9e-3

Leverage intrinsic error tolerance

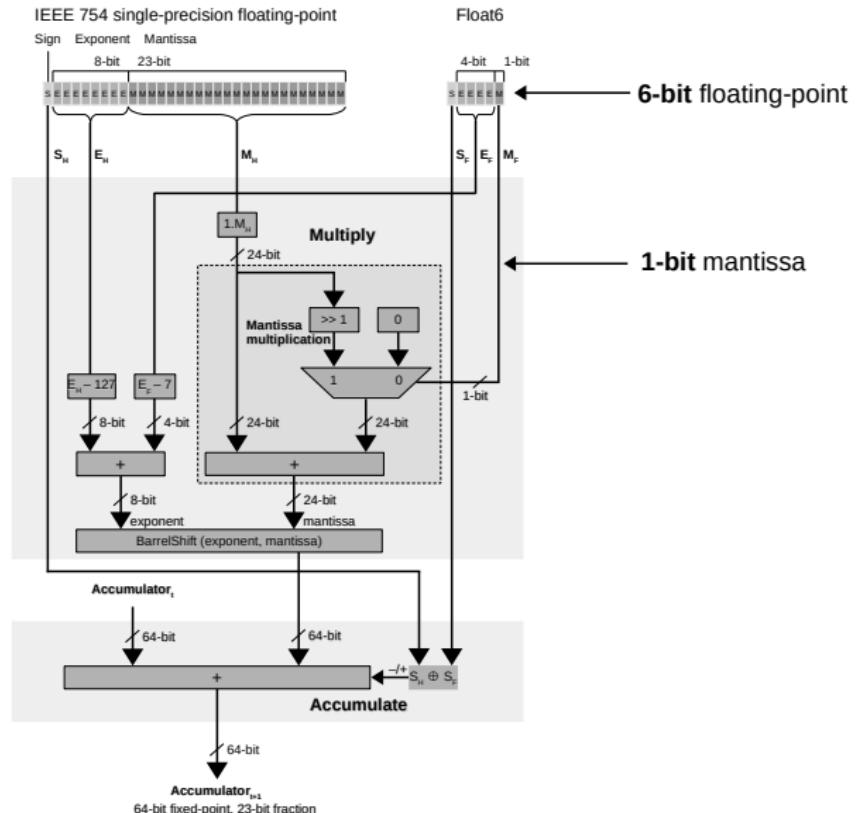
Custom Floating-Point MAC Designs

Hybrid Custom Floating-Point Multiply-Accumulate Unit

Vector dot-product:

Multiplication

Accumulation



Custom Floating-Point MAC Designs

Hybrid Custom Floating-Point Multiply-Accumulate Unit

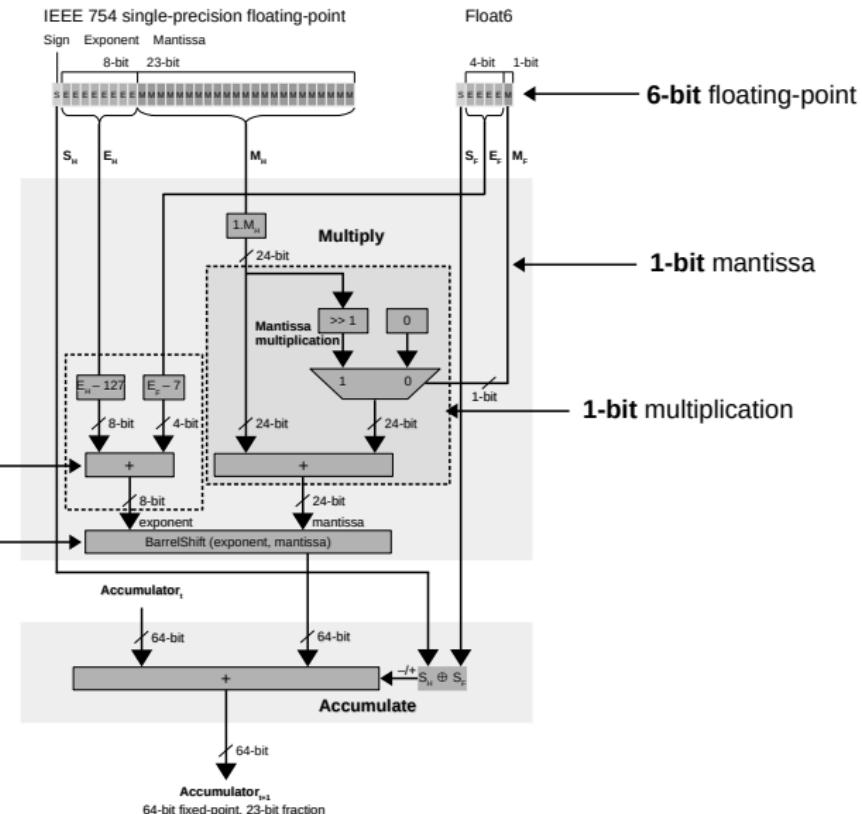
Vector dot-product:

Multiplication

Accumulation

Exponent addition

Denormalization



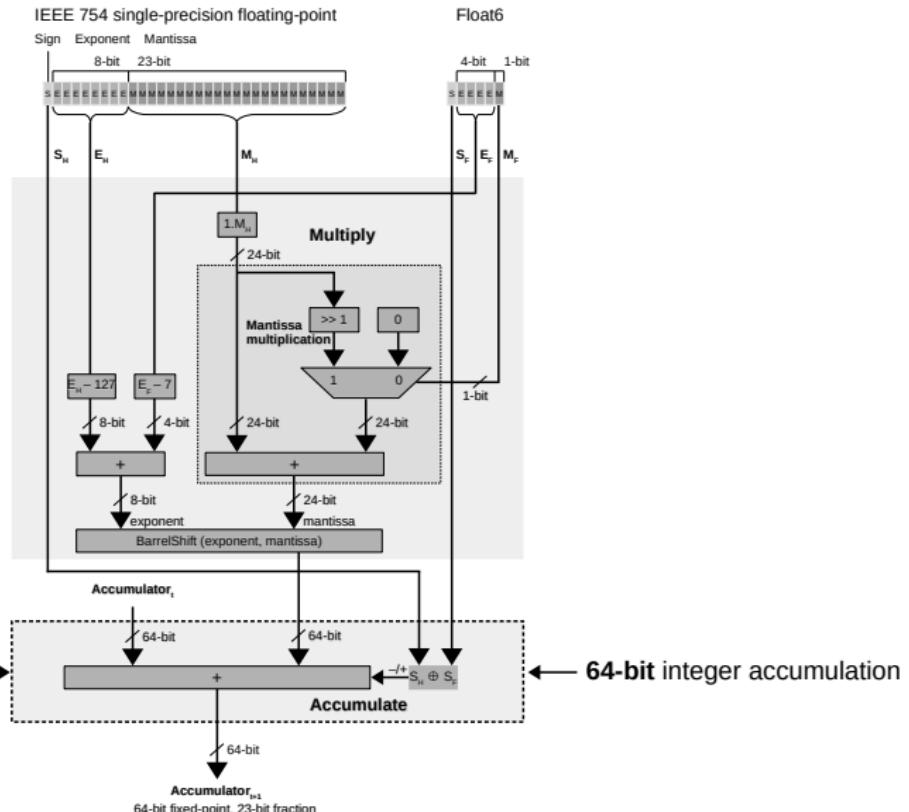
Custom Floating-Point MAC Designs

Hybrid Custom Floating-Point Multiply-Accumulate Unit

Vector dot-product:

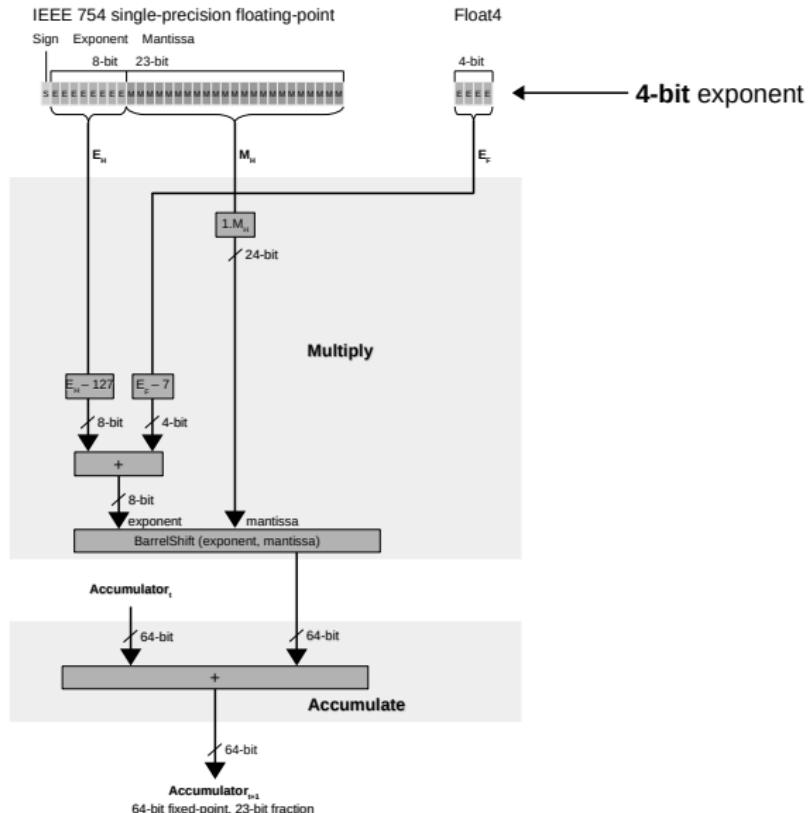
Multiplication

Accumulation



Custom Floating-Point MAC Designs

Hybrid Custom Floating-Point Multiply-Accumulate Unit



Custom Floating-Point MAC Designs

Hybrid Custom Floating-Point Multiply-Accumulate Unit

For noise-robust applications with
non-negativity constraints.

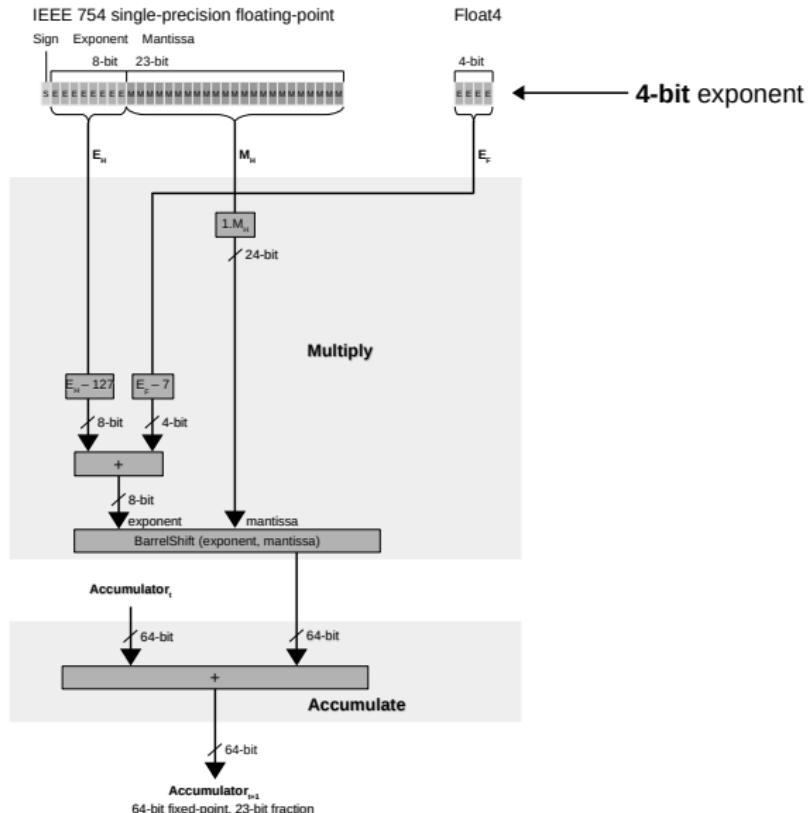
Non-negative Tensor Factorization

Non-negative Sparse Coding

Non-negative Matrix Factorization

Spike-by-Spike Neural Networks

Etc.



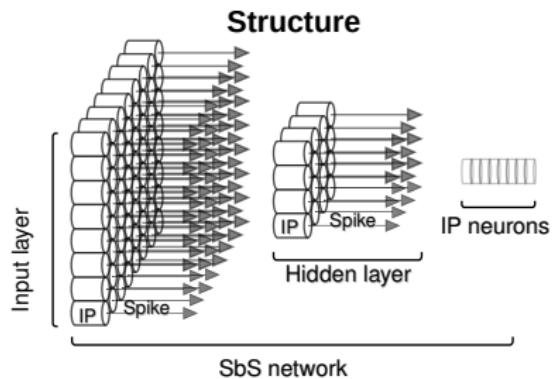
1 Methodology

2 Floating-Point Multiply-Accumulate Unit

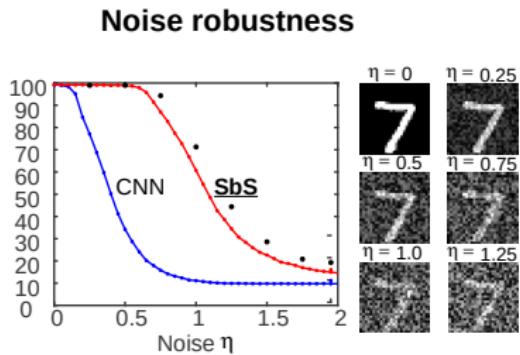
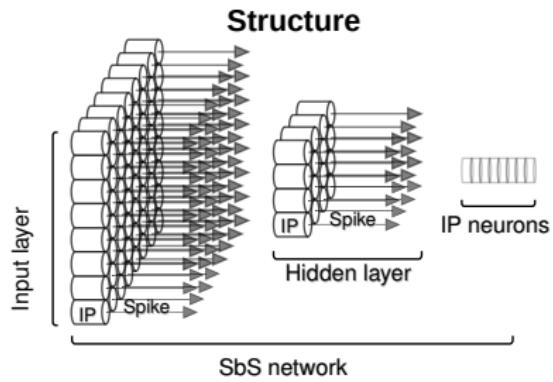
3 Case Studies

4 Conclusions

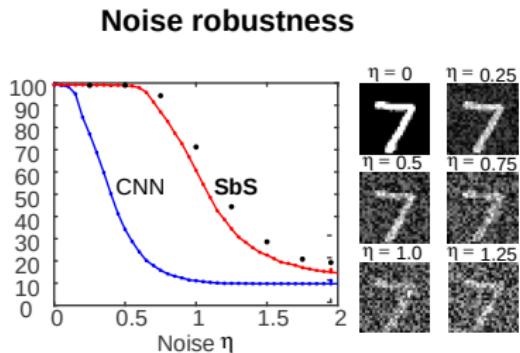
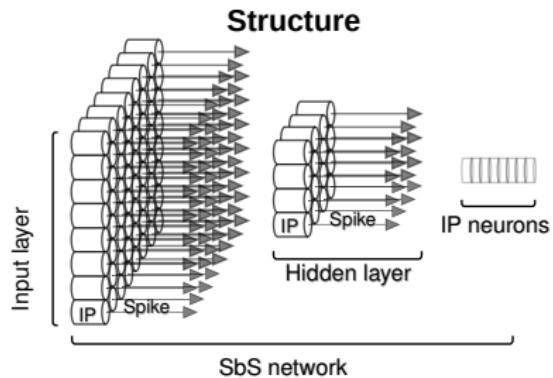
Spike-by-Spike Neural Network



Spike-by-Spike Neural Network



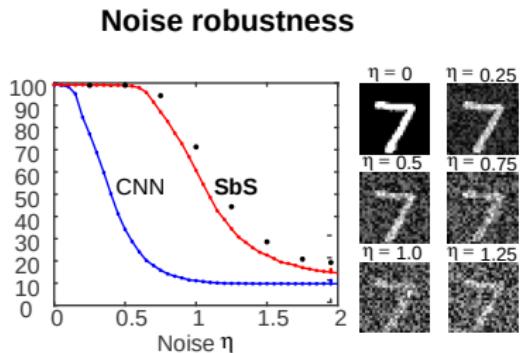
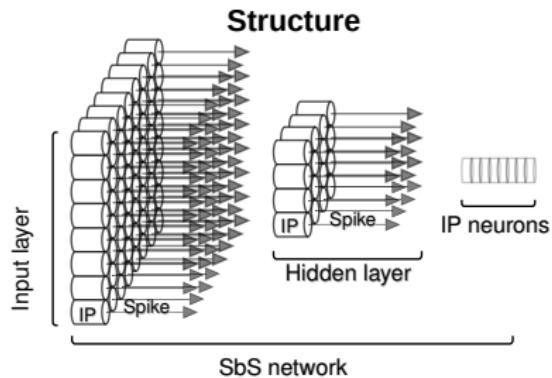
Spike-by-Spike Neural Network



Neuron update

$$h_{\mu}^{new}(i) = \frac{1}{1 + \epsilon} \left(h_{\mu}(i) + \epsilon \frac{h_{\mu}(i)W(s_t|i)}{\sum_j h_{\mu}(j)W(s_t|j)} \right)$$

Spike-by-Spike Neural Network

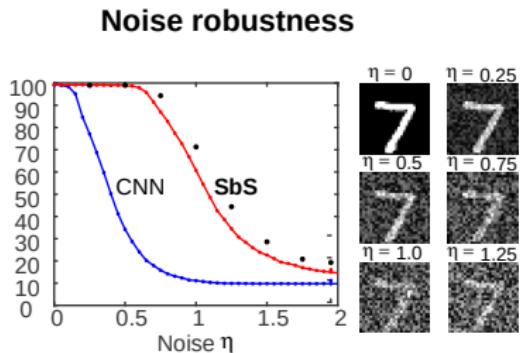
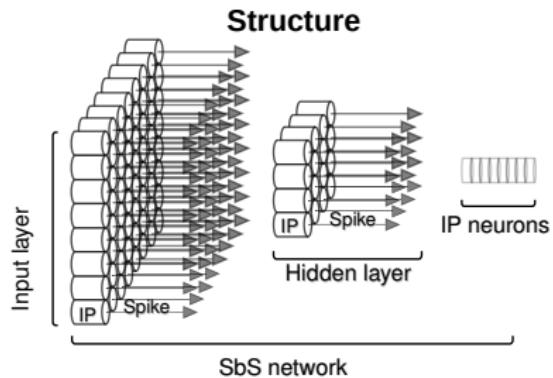


Neuron update

$$h_{\mu}^{new}(i) = \frac{1}{1 + \epsilon} \left(h_{\mu}(i) + \epsilon \frac{h_{\mu}(i)W(s_t|i)}{\sum_j h_{\mu}(j)W(s_t|j)} \right)$$

Non-negativity $0 \leq W(s_t|j) \leq 1$

Spike-by-Spike Neural Network



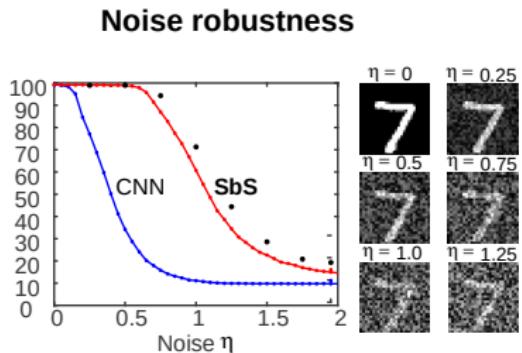
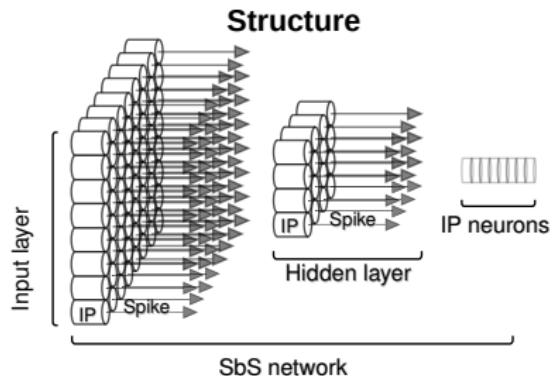
Neuron update

$$h_{\mu}^{new}(i) = \frac{1}{1 + \epsilon} \left(h_{\mu}(i) + \epsilon \frac{h_{\mu}(i)W(s_t|i)}{\sum_j h_{\mu}(j)W(s_t|j)} \right)$$

Non-negativity $0 \leq W(s_t|j) \leq 1$

Normalized $\sum_{s_t=0}^{M-1} W(s_t|j) = 1$

Spike-by-Spike Neural Network



Properties:

- Noise robustness
- Iterative optimization
- No sign bit required
- Requires division
- Compute and memory intensive

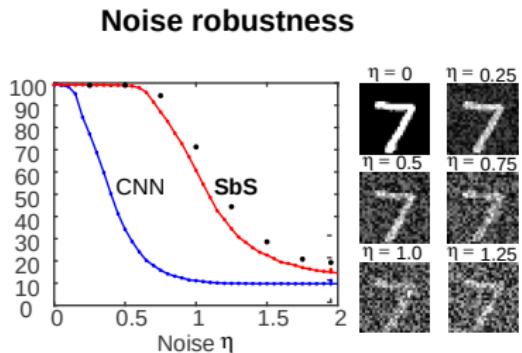
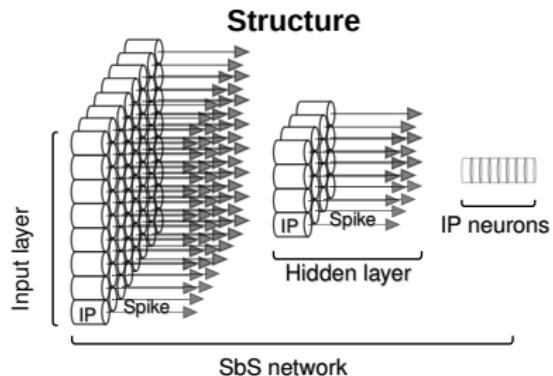
Neuron update

$$h_{\mu}^{new}(i) = \frac{1}{1 + \epsilon} \left(h_{\mu}(i) + \epsilon \frac{h_{\mu}(i)W(s_t|i)}{\sum_j h_{\mu}(j)W(s_t|j)} \right)$$

Non-negativity $0 \leq W(s_t|j) \leq 1$

Normalized $\sum_{s_t=0}^{M-1} W(s_t|j) = 1$

Spike-by-Spike Neural Network



Properties:

Noise robustness

Iterative optimization

No sign bit required

Requires division

Compute and memory intensive

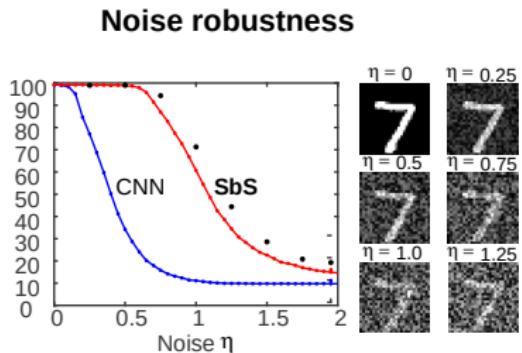
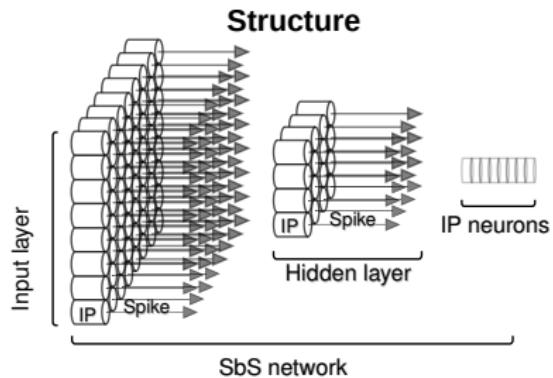
Neuron update

$$h_{\mu}^{new}(i) = \frac{1}{1 + \epsilon} \left(h_{\mu}(i) + \epsilon \frac{h_{\mu}(i)W(s_t|i)}{\sum_j h_{\mu}(j)W(s_t|j)} \right)$$

Non-negativity $0 \leq W(s_t|j) \leq 1$

Normalized $\sum_{s_t=0}^{M-1} W(s_t|j) = 1$

Spike-by-Spike Neural Network



Properties:

Noise robustness

Iterative optimization

No sign bit required

Requires division

Compute and memory intensive

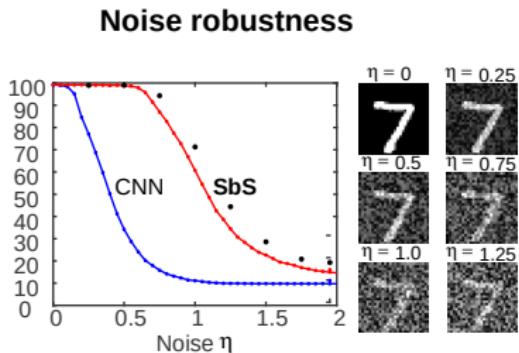
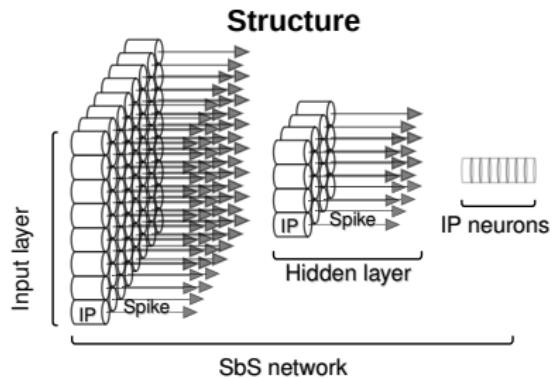
Neuron update

$$h_{\mu}^{new}(i) = \frac{1}{1 + \epsilon} \left(h_{\mu}(i) + \epsilon \frac{h_{\mu}(i)W(s_t|i)}{\sum_j h_{\mu}(j)W(s_t|j)} \right)$$

Non-negativity $0 \leq W(s_t|j) \leq 1$

Normalized $\sum_{s_t=0}^{M-1} W(s_t|j) = 1$

Spike-by-Spike Neural Network



Properties:

Noise robustness

Iterative optimization

No sign bit required

Requires division

Compute and memory intensive

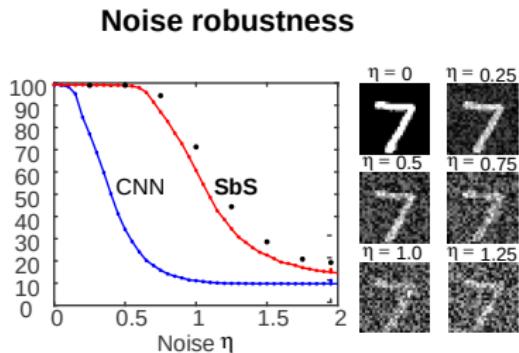
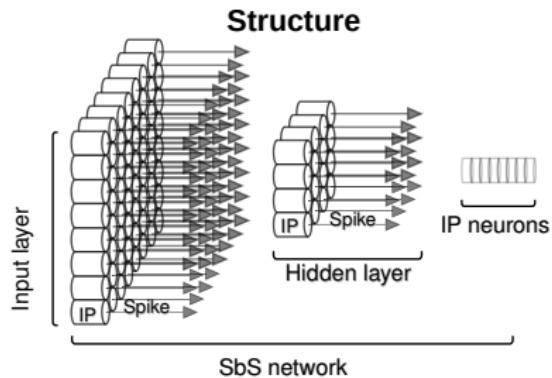
Neuron update

$$h_\mu^{new}(i) = \frac{1}{1+\epsilon} \left(h_\mu(i) + \epsilon \frac{h_\mu(i)W(s_t|i)}{\sum_j h_\mu(j)W(s_t|j)} \right)$$

Non-negativity $0 \leq W(s_t|j) \leq 1$

$$\text{Normalized} \quad \sum_{s_t=0}^{M-1} W(s_t|j) = 1$$

Spike-by-Spike Neural Network



Properties:

Noise robustness

Iterative optimization

No sign bit required

Requires division

Compute and memory intensive

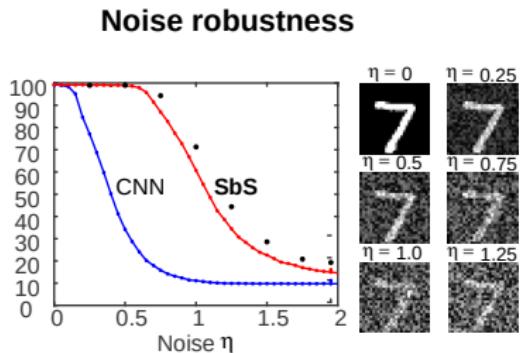
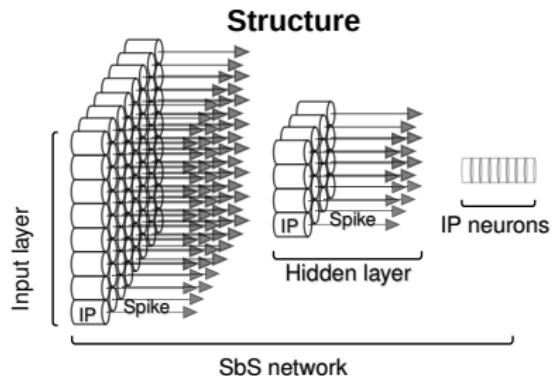
Neuron update

$$h_\mu^{new}(i) = \frac{1}{1+\epsilon} \left(h_\mu(i) + \epsilon \frac{h_\mu(i)W(s_t|i)}{\sum_j h_\mu(j)W(s_t|j)} \right)$$

Non-negativity $0 \leq W(s_t|j) \leq 1$

$$\text{Normalized} \quad \sum_{s_t=0}^{M-1} W(s_t|j) = 1$$

Spike-by-Spike Neural Network



Properties:

Noise robustness

Iterative optimization

No sign bit required

Requires division

Compute and memory intensive

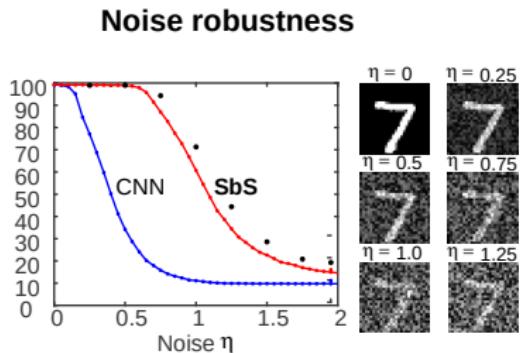
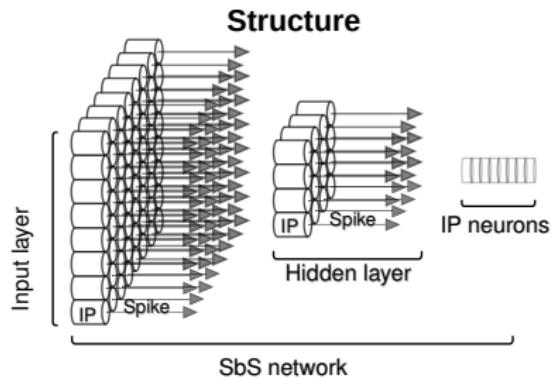
Neuron update

$$h_{\mu}^{new}(i) = \frac{1}{1 + \epsilon} \left(h_{\mu}(i) + \epsilon \frac{h_{\mu}(i)W(s_t|i)}{\sum_j h_{\mu}(j)W(s_t|j)} \right)$$

Non-negativity $0 \leq W(s_t|j) \leq 1$

Normalized $\sum_{s_t=0}^{M-1} W(s_t|j) = 1$

Spike-by-Spike Neural Network



Properties:

- Noise robustness
- Iterative optimization
- No sign bit required
- Requires division
- Compute and memory intensive

Neuron update

$$h_{\mu}^{new}(i) = \frac{1}{1 + \epsilon} \left(h_{\mu}(i) + \epsilon \frac{h_{\mu}(i)W(s_t|i)}{\sum_j h_{\mu}(j)W(s_t|j)} \right)$$

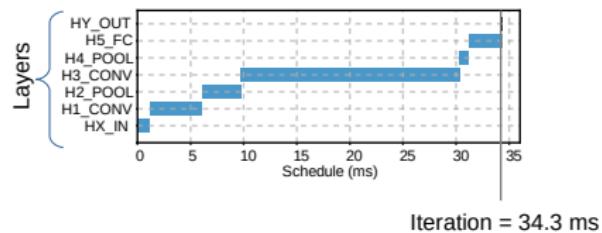
Non-negativity $0 \leq W(s_t|j) \leq 1$

Normalized $\sum_{s_t=0}^{M-1} W(s_t|j) = 1$

Deployment

Floating-point 32-bit

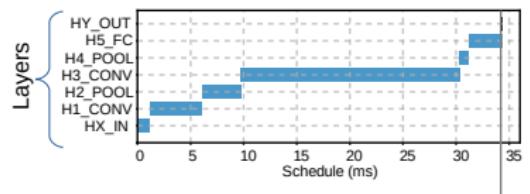
ARM Cortex A9 @ 666 MHz



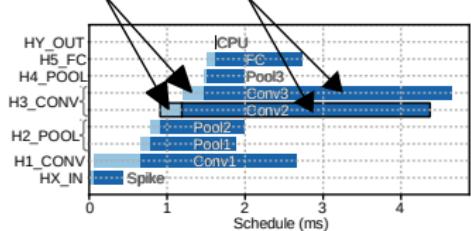
Deployment

Floating-point 32-bit

ARM Cortex A9 @ 666 MHz



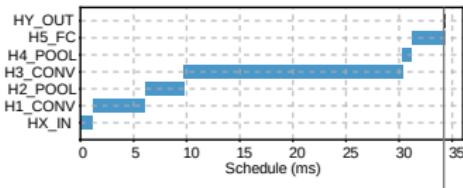
CPU Accelerator Iteration = 34.3 ms



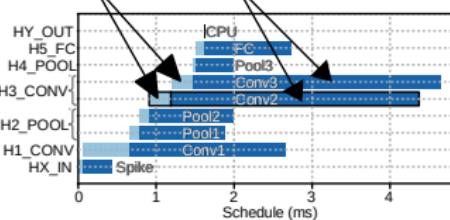
Deployment

Floating-point 32-bit

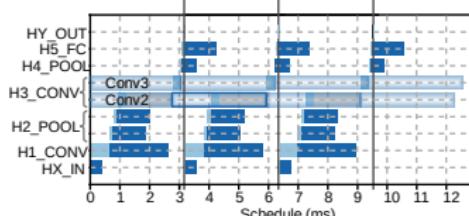
ARM Cortex A9 @ 666 MHz



CPU Accelerator Iteration = 34.3 ms



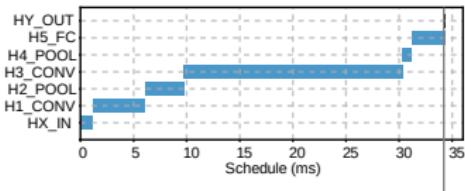
Iteration = 3.18 ms



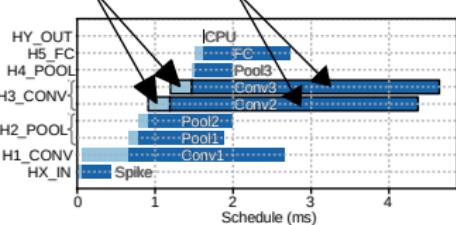
Deployment

Floating-point 32-bit

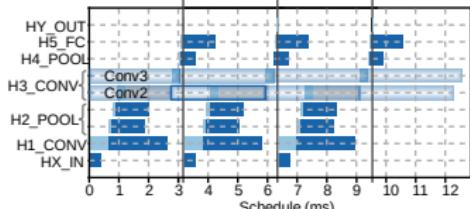
ARM Cortex A9 @ 666 MHz



CPU Accelerator **Iteration = 34.3 ms**



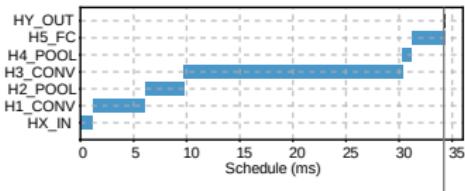
Iteration = 3.18 ms



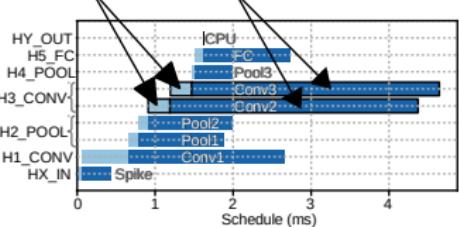
Deployment

Floating-point 32-bit

ARM Cortex A9 @ 666 MHz

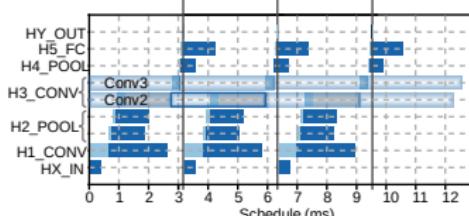


CPU Accelerator Iteration = 34.3 ms



Acceleration = 10.7

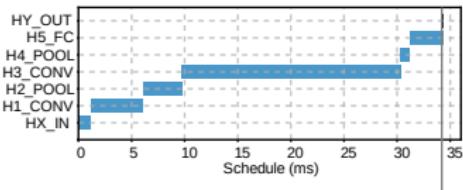
Iteration = 3.18 ms



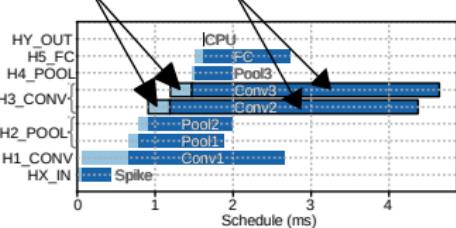
Deployment

Floating-point 32-bit

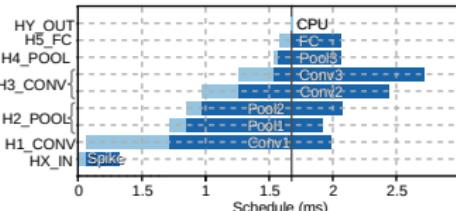
ARM Cortex A9 @ 666 MHz



CPU Accelerator Iteration = 34.3 ms

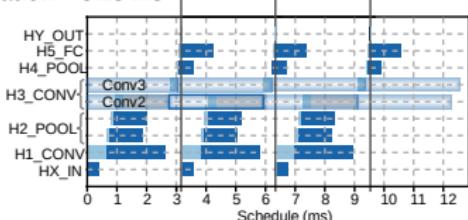


Hybrid logarithmic 4-bit

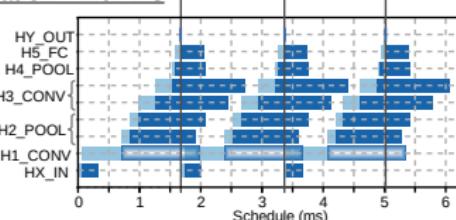


Acceleration = 10.7

Iteration = 3.18 ms



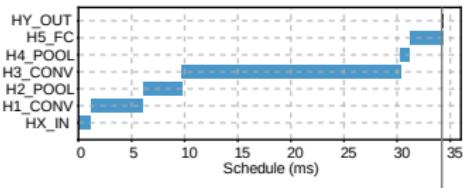
Iteration = 1.67 ms



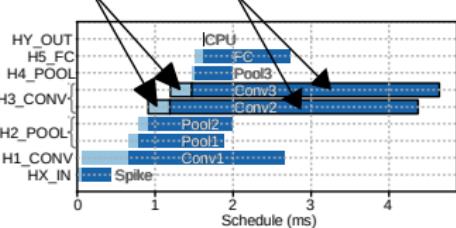
Deployment

Floating-point 32-bit

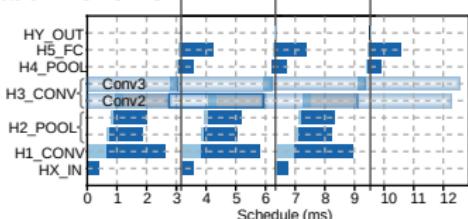
ARM Cortex A9 @ 666 MHz



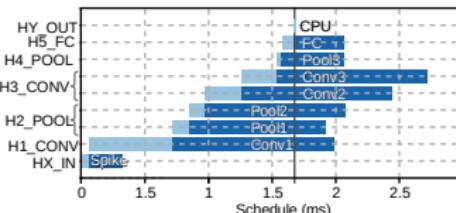
CPU Accelerator Iteration = 34.3 ms



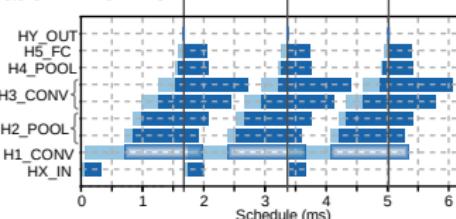
Iteration = 3.18 ms Acceleration = 10.7



Hybrid logarithmic 4-bit

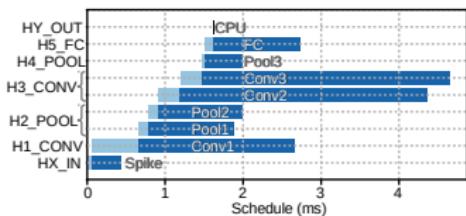


Iteration = 1.67 ms Acceleration = 20.5

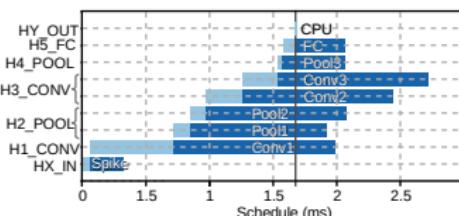


Deployment

Standard floating-point

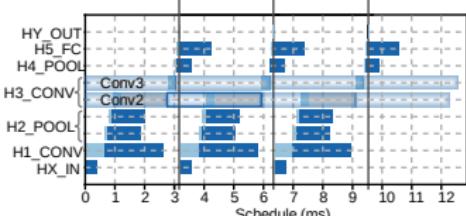


Hybrid logarithmic 4-bit



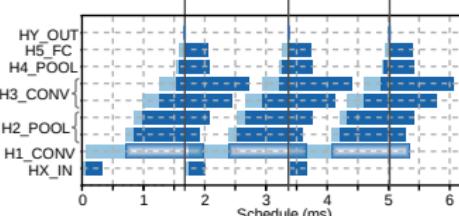
Iteration = 3.18 ms

Acceleration = 10.7



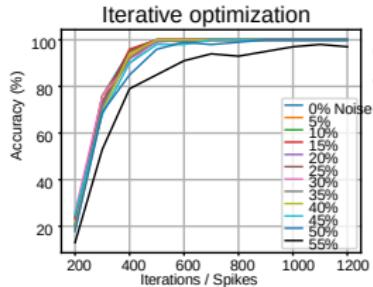
Iteration = 1.67 ms

Acceleration = 20.5

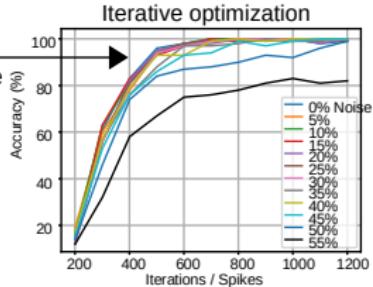


Accelerating Spike-by-Spike Neural Networks with Hybrid Logarithmic 4-bit

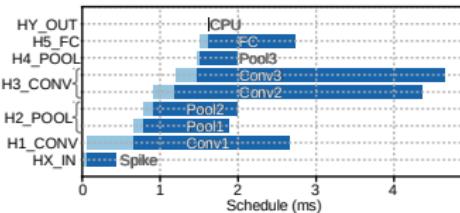
Deployment



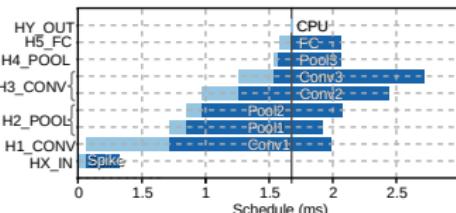
Good convergence applying levels of noise



Standard floating-point

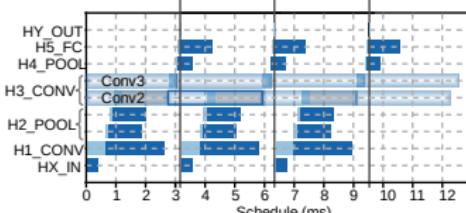


Hybrid logarithmic 4-bit



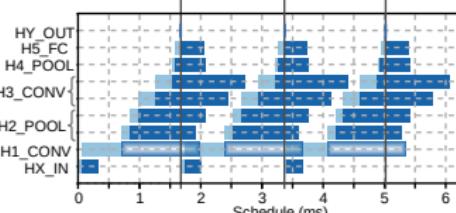
Acceleration = 10.7

Iteration = 3.18 ms



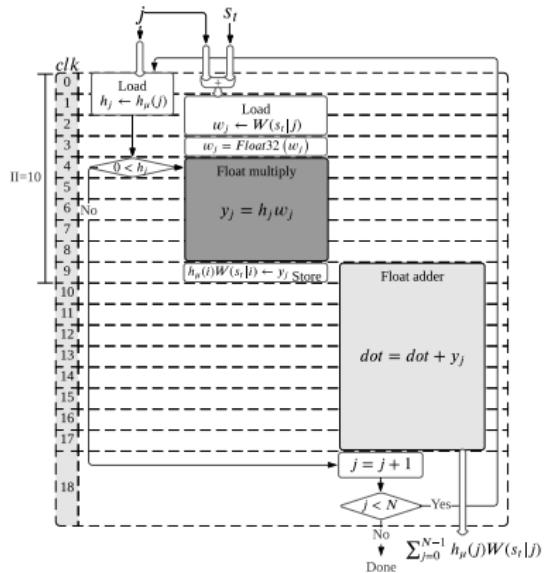
Acceleration = 20.5

Iteration = 1.67 ms

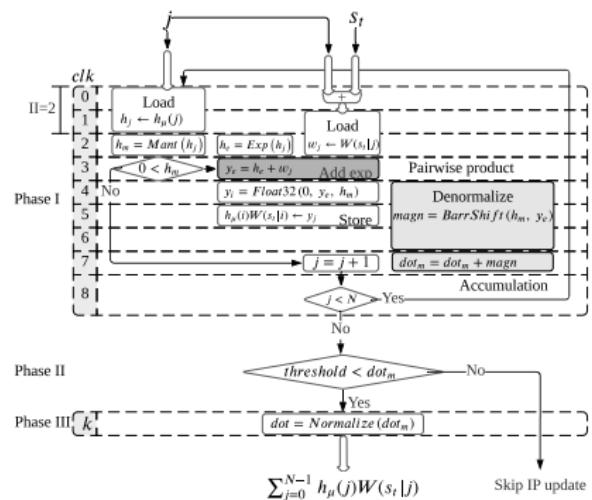


MAC Design: Standard Floating-Point vs. Hybrid Logarithmic 4-bit

Standard floating-point

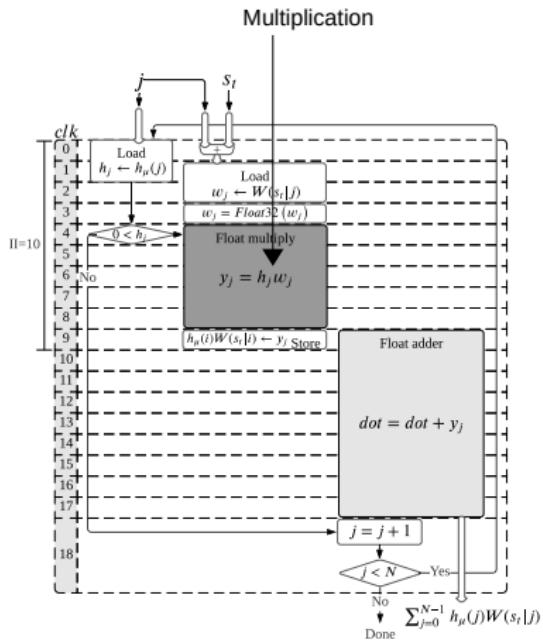


Hybrid logarithmic 4-bit

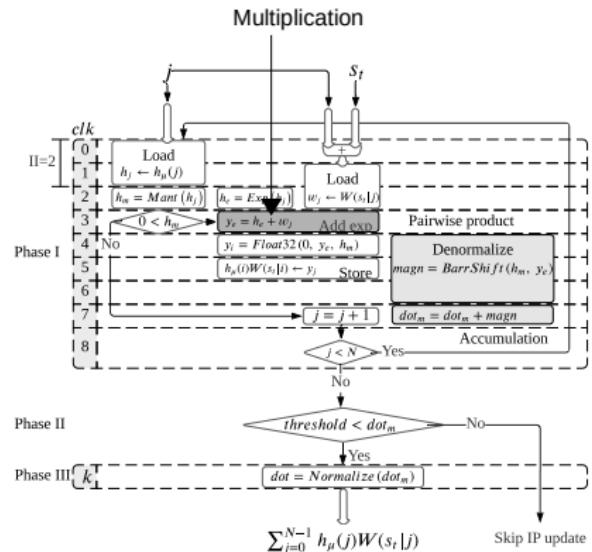


MAC Design: Standard Floating-Point vs. Hybrid Logarithmic 4-bit

Standard floating-point

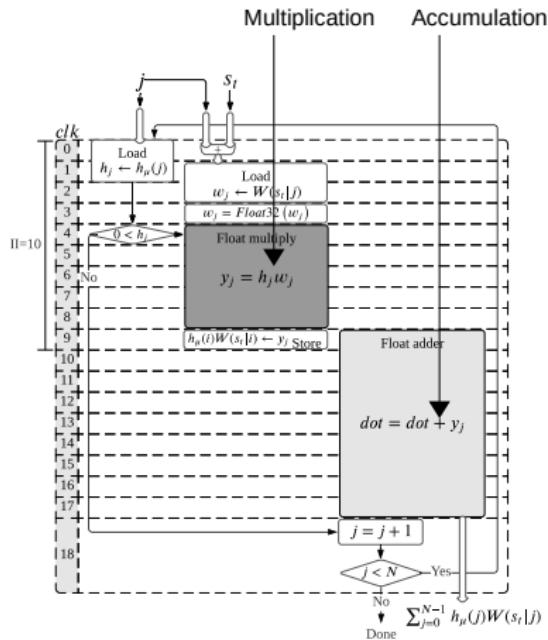


Hybrid logarithmic 4-bit

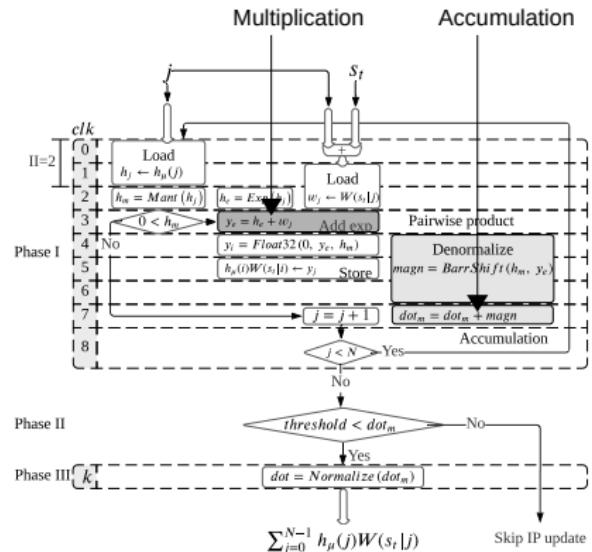


MAC Design: Standard Floating-Point vs. Hybrid Logarithmic 4-bit

Standard floating-point

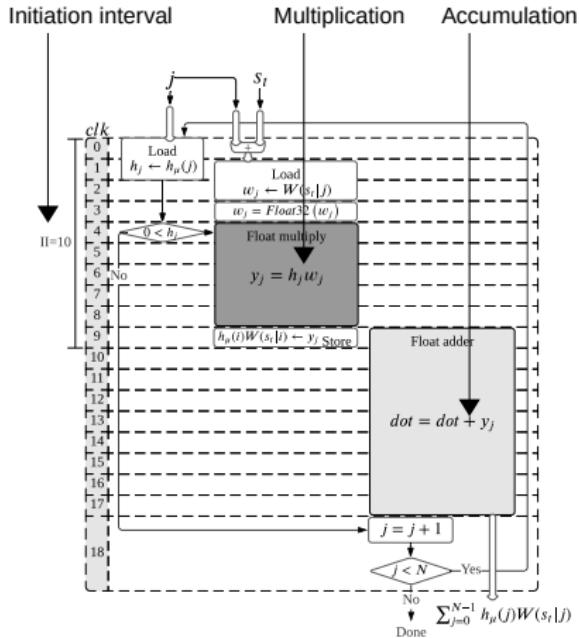


Hybrid logarithmic 4-bit

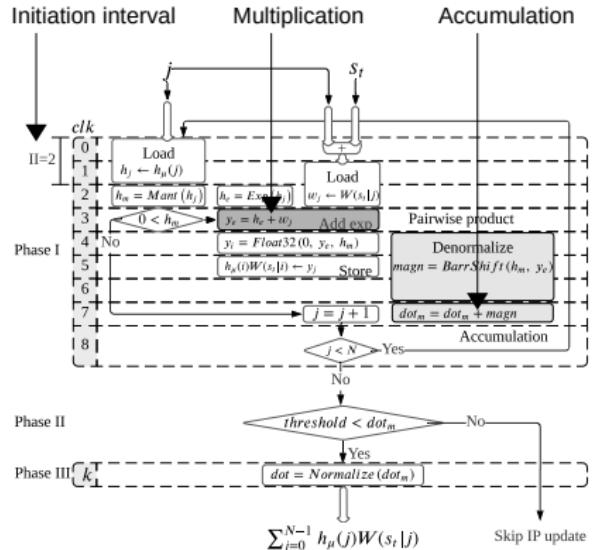


MAC Design: Standard Floating-Point vs. Hybrid Logarithmic 4-bit

Standard floating-point

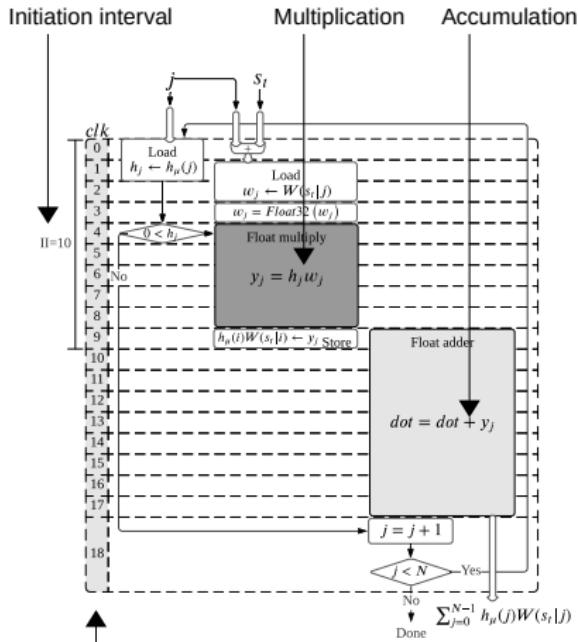


Hybrid logarithmic 4-bit

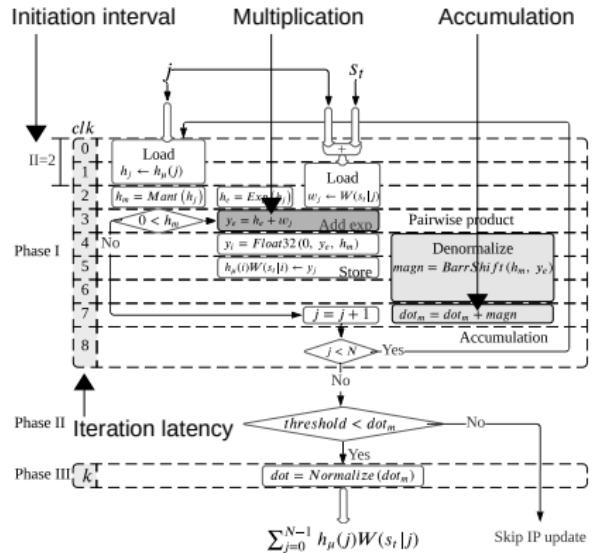


MAC Design: Standard Floating-Point vs. Hybrid Logarithmic 4-bit

Standard floating-point



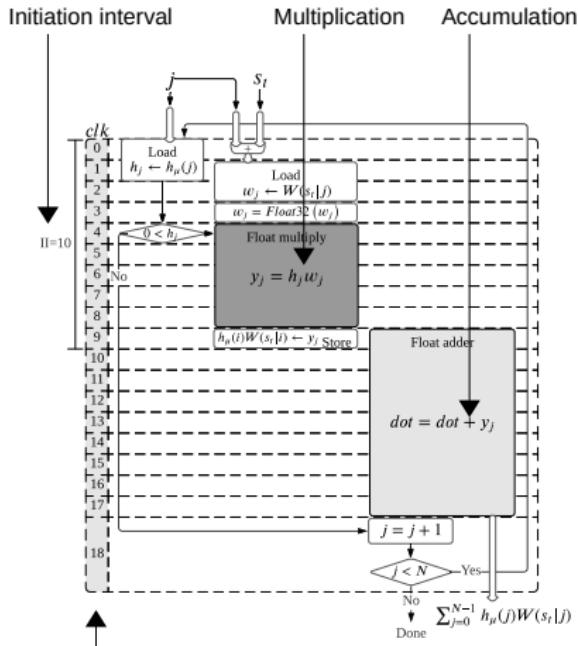
Hybrid logarithmic 4-bit



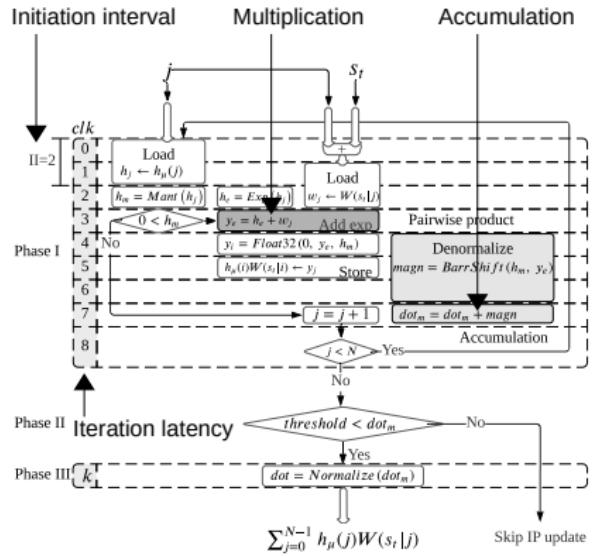
Iteration latency

MAC Design: Standard Floating-Point vs. Hybrid Logarithmic 4-bit

Standard floating-point



Hybrid logarithmic 4-bit



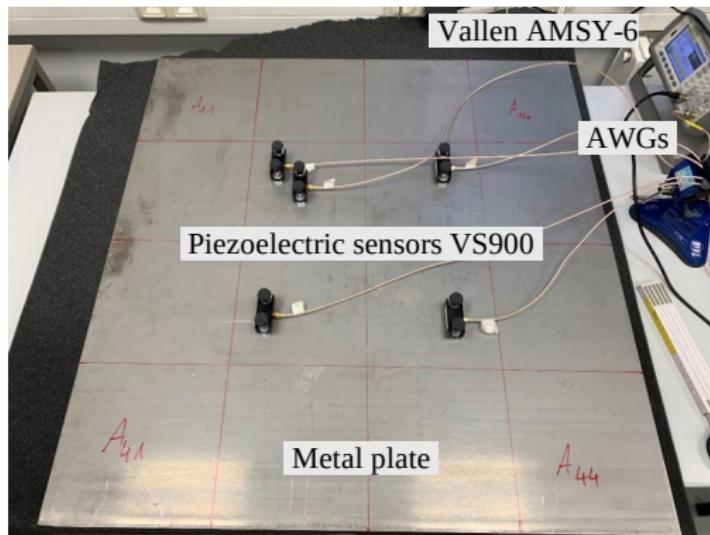
Iteration latency

$$L_{log} = 2N+7$$

Accelerating 2D Convolution with Hybrid Floating-Point 6-bit

TinyML Application: CNN Sensor Analytics for Structural Health Monitoring

Anomaly localization with regression



Experimental setup:

Sensor and noise positions

Training dataset

Testing dataset

Model architecture

Accelerating 2D Convolution with Hybrid Floating-Point 6-bit

TinyML Application: CNN Sensor Analytics for Structural Health Monitoring

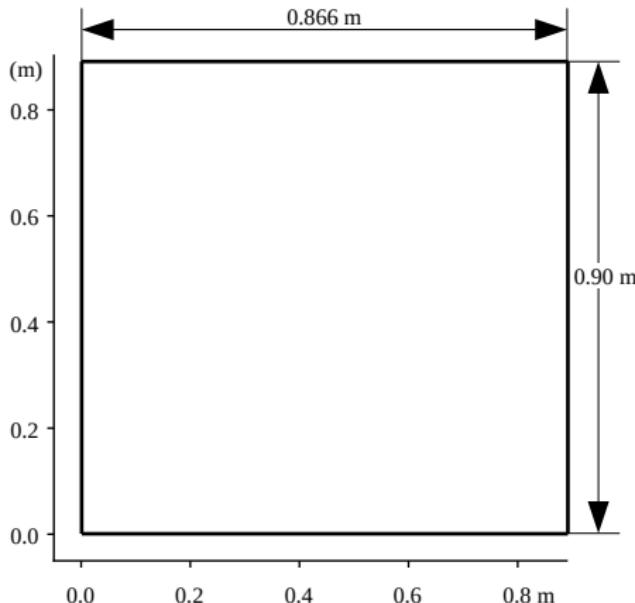
Experimental setup:

Sensor and noise positions

Training dataset

Testing dataset

Model architecture



Accelerating 2D Convolution with Hybrid Floating-Point 6-bit

TinyML Application: CNN Sensor Analytics for Structural Health Monitoring

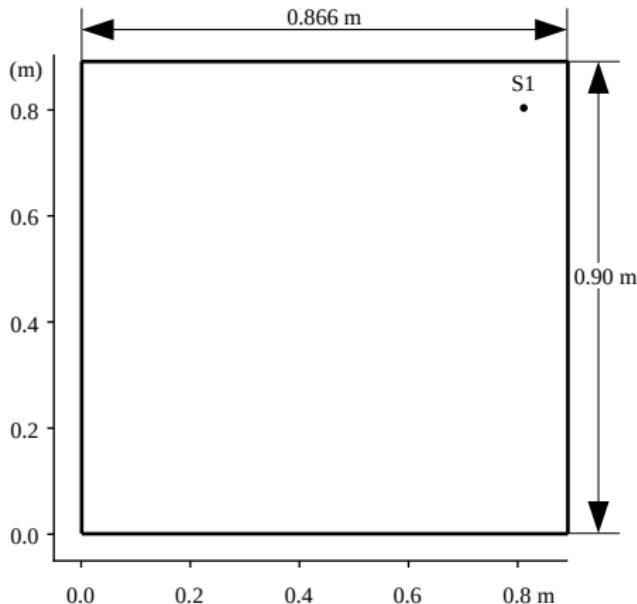
Experimental setup:

Sensor and noise positions

Training dataset

Testing dataset

Model architecture



Accelerating 2D Convolution with Hybrid Floating-Point 6-bit

TinyML Application: CNN Sensor Analytics for Structural Health Monitoring

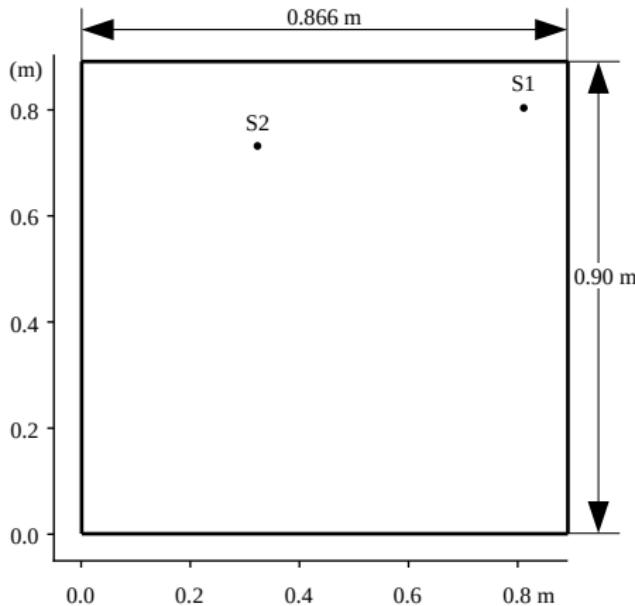
Experimental setup:

Sensor and noise positions

Training dataset

Testing dataset

Model architecture



Accelerating 2D Convolution with Hybrid Floating-Point 6-bit

TinyML Application: CNN Sensor Analytics for Structural Health Monitoring

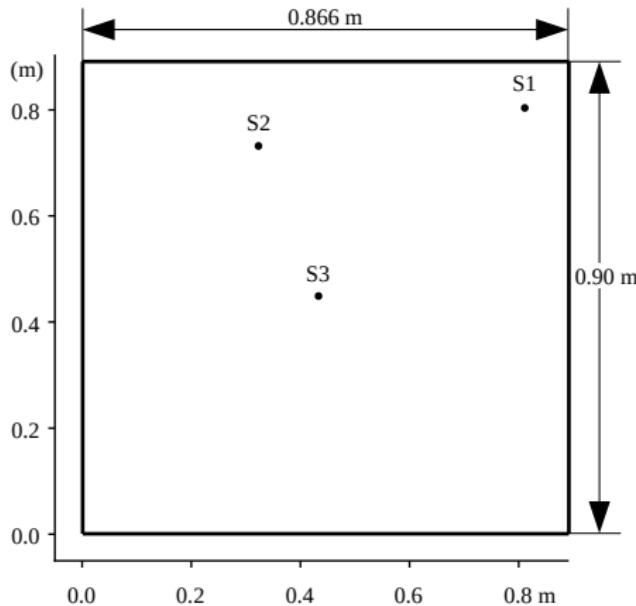
Experimental setup:

Sensor and noise positions

Training dataset

Testing dataset

Model architecture



Accelerating 2D Convolution with Hybrid Floating-Point 6-bit

TinyML Application: CNN Sensor Analytics for Structural Health Monitoring

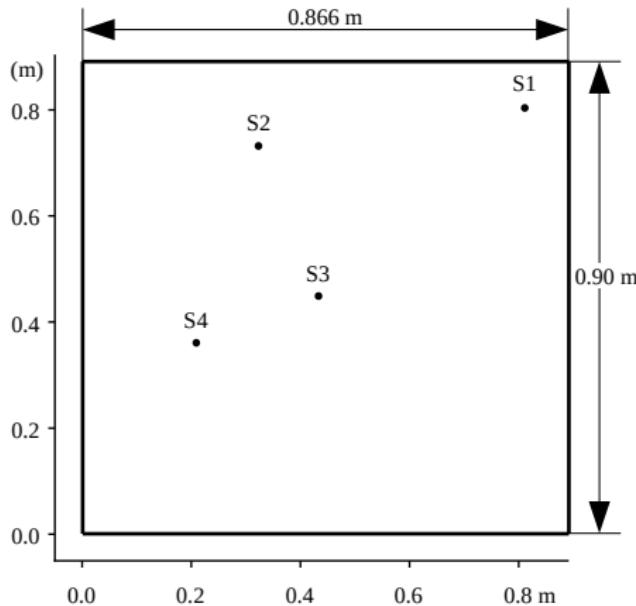
Experimental setup:

Sensor and noise positions

Training dataset

Testing dataset

Model architecture



Accelerating 2D Convolution with Hybrid Floating-Point 6-bit

TinyML Application: CNN Sensor Analytics for Structural Health Monitoring

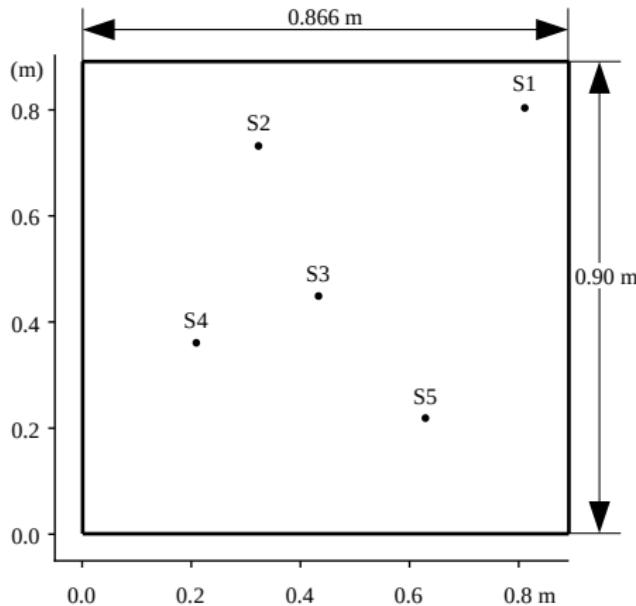
Experimental setup:

Sensor and noise positions

Training dataset

Testing dataset

Model architecture



Accelerating 2D Convolution with Hybrid Floating-Point 6-bit

TinyML Application: CNN Sensor Analytics for Structural Health Monitoring

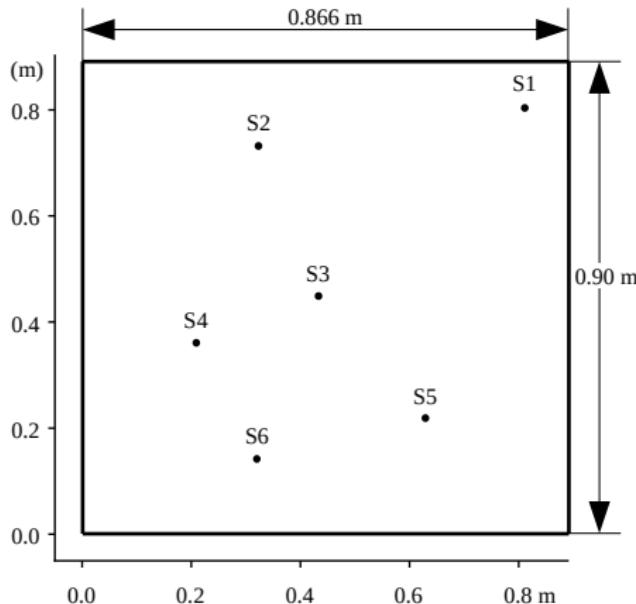
Experimental setup:

Sensor and noise positions

Training dataset

Testing dataset

Model architecture



Accelerating 2D Convolution with Hybrid Floating-Point 6-bit

TinyML Application: CNN Sensor Analytics for Structural Health Monitoring

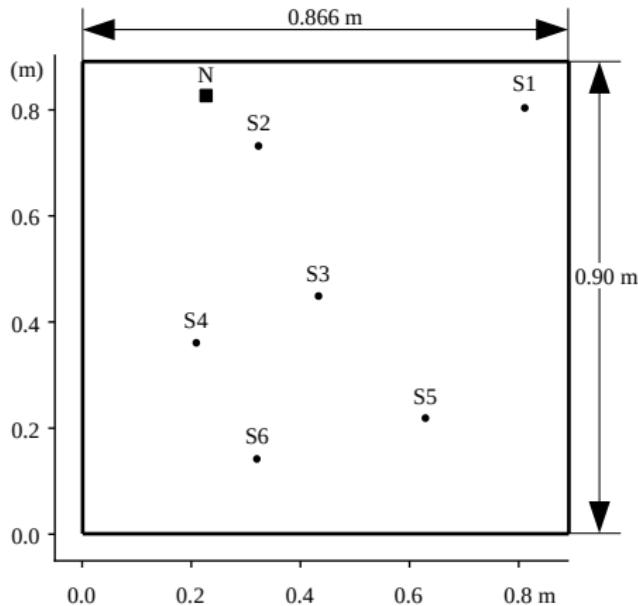
Experimental setup:

Sensor and noise positions

Training dataset

Testing dataset

Model architecture



Accelerating 2D Convolution with Hybrid Floating-Point 6-bit

TinyML Application: CNN Sensor Analytics for Structural Health Monitoring

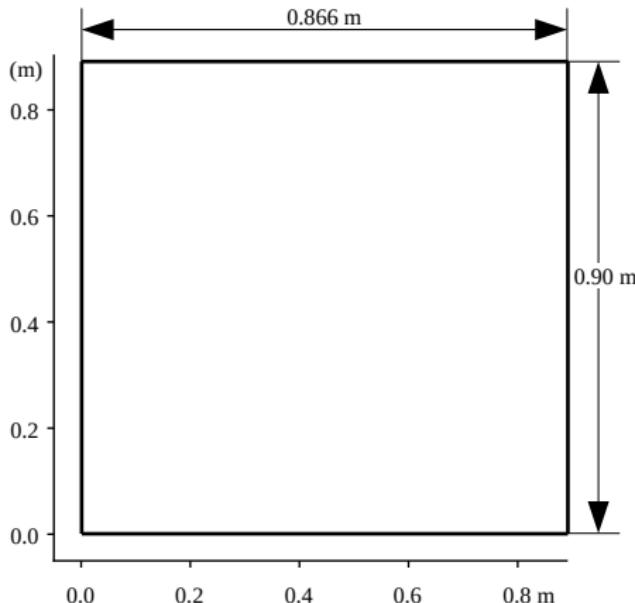
Experimental setup:

Sensor and noise positions

Training dataset

Testing dataset

Model architecture



Accelerating 2D Convolution with Hybrid Floating-Point 6-bit

TinyML Application: CNN Sensor Analytics for Structural Health Monitoring

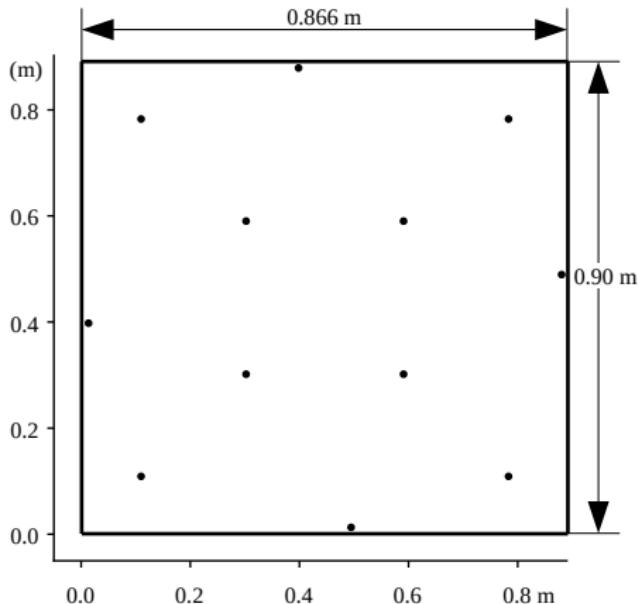
Experimental setup:

Sensor and noise positions

Training dataset

Testing dataset

Model architecture



Training dataset = 12 (excitation points) x 30,000 (samples)

Accelerating 2D Convolution with Hybrid Floating-Point 6-bit

TinyML Application: CNN Sensor Analytics for Structural Health Monitoring

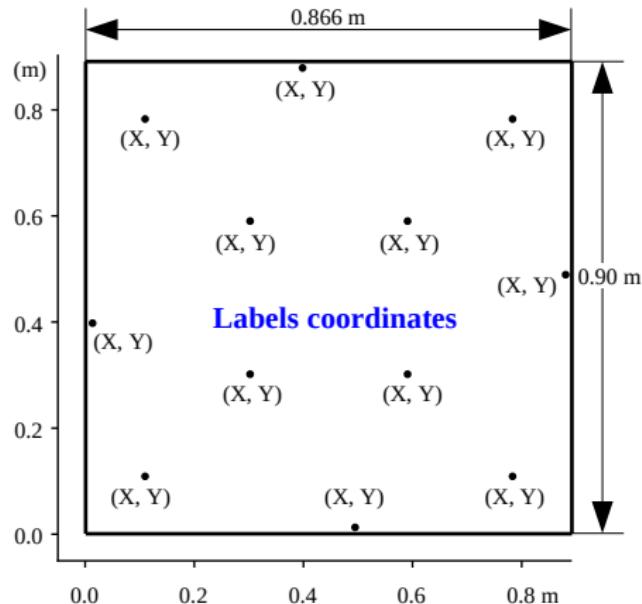
Experimental setup:

Sensor and noise positions

Training dataset

Testing dataset

Model architecture



Training dataset = 12 (excitation points) x 30,000 (samples)

Accelerating 2D Convolution with Hybrid Floating-Point 6-bit

TinyML Application: CNN Sensor Analytics for Structural Health Monitoring

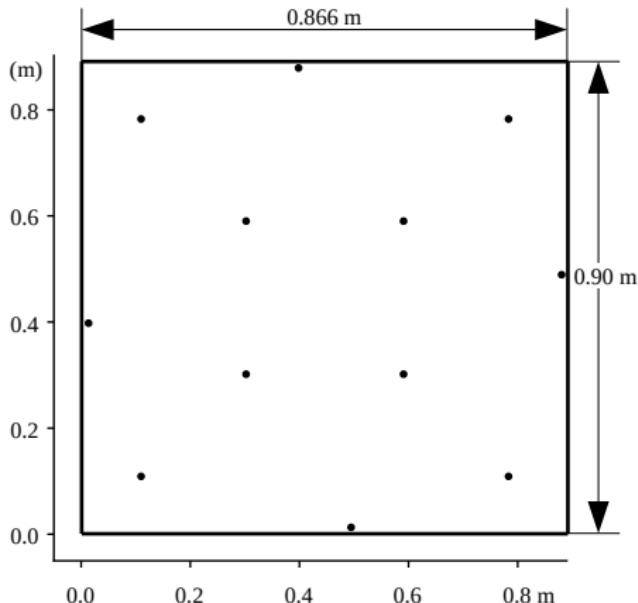
Experimental setup:

Sensor and noise positions

Training dataset

Testing dataset

Model architecture



Training dataset = 12 (excitation points) x 30,000 (samples)

Accelerating 2D Convolution with Hybrid Floating-Point 6-bit

TinyML Application: CNN Sensor Analytics for Structural Health Monitoring

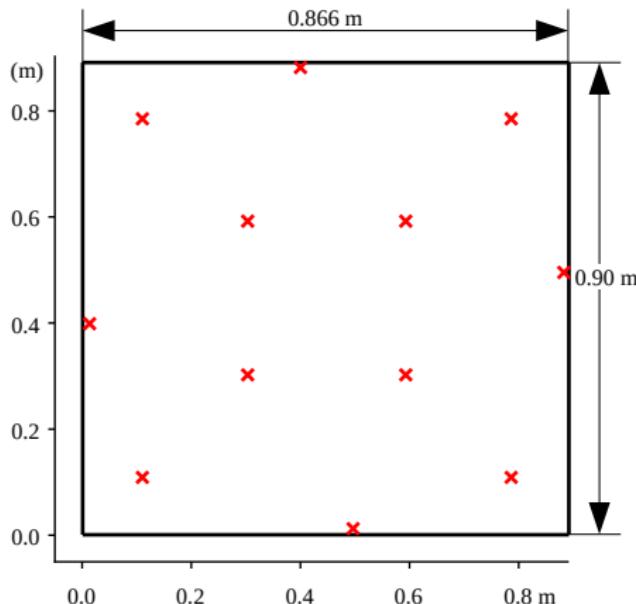
Experimental setup:

Sensor and noise positions

Training dataset

Testing dataset

Model architecture



Accelerating 2D Convolution with Hybrid Floating-Point 6-bit

TinyML Application: CNN Sensor Analytics for Structural Health Monitoring

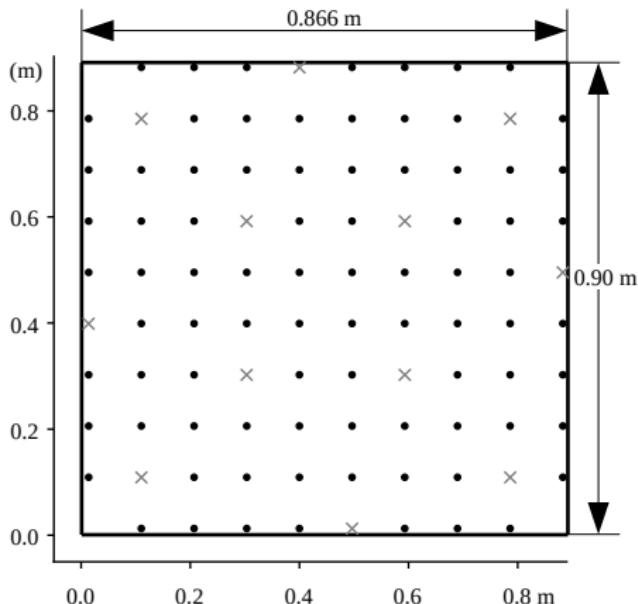
Experimental setup:

Sensor and noise positions

Training dataset

Testing dataset

Model architecture



Training dataset = 84 (excitation points) x 30,000 (samples)

Accelerating 2D Convolution with Hybrid Floating-Point 6-bit

TinyML Application: CNN Sensor Analytics for Structural Health Monitoring

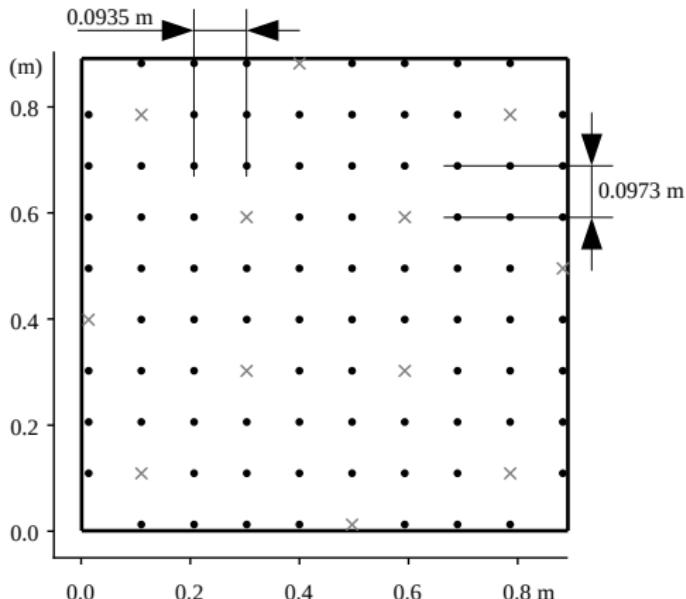
Experimental setup:

Sensor and noise positions

Training dataset

Testing dataset

Model architecture



Training dataset = 84 (excitation points) x 30,000 (samples)

Accelerating 2D Convolution with Hybrid Floating-Point 6-bit

TinyML Application: CNN Sensor Analytics for Structural Health Monitoring

Experimental setup:

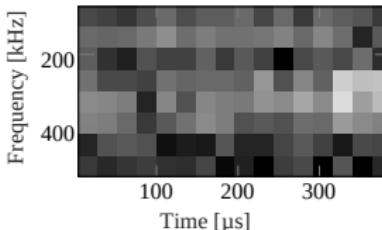
Sensor and noise positions

Training dataset

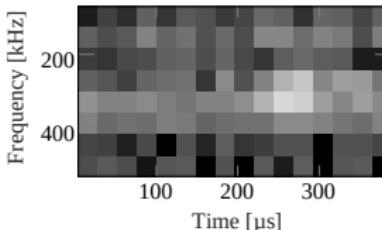
Testing dataset

Model architecture

S1



S2



Sampling frequency = 10 MHz
FFT length = 32 samples
Windowing = Blackman
Overlap = 8 samples
Frequency range = 100kHz - 500kHz

Accelerating 2D Convolution with Hybrid Floating-Point 6-bit

TinyML Application: CNN Sensor Analytics for Structural Health Monitoring

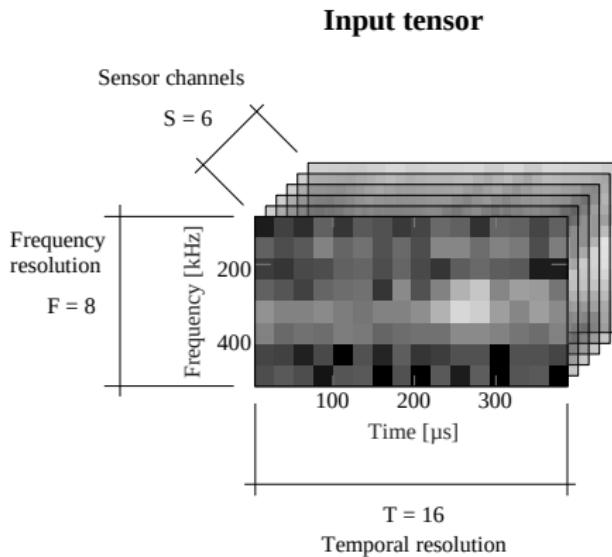
Experimental setup:

Sensor and noise positions

Training dataset

Testing dataset

Model architecture



Accelerating 2D Convolution with Hybrid Floating-Point 6-bit

TinyML Application: CNN Sensor Analytics for Structural Health Monitoring

Experimental setup:

Sensor and noise positions

Training dataset

Testing dataset

Model architecture

CNN-regression model

Output tensor (x, y)

FC (2), Linear
FC (E = 64), ReLu
FC (D = 196), ReLu
Flatten
2 x 2 MaxPool, stride 2
BatchNormalization
3 x 3 Conv (C = 60), ReLu
2 x 2 MaxPool, stride 2
BatchNormalization
3 x 3 Conv (B = 55), ReLu
2 x 2 MaxPool, stride 2
BatchNormalization
3 x 3 Conv (A = 50), ReLu
Input tensor (F x T x S)

TensorFlow Lite

Accelerating 2D Convolution with Hybrid Floating-Point 6-bit

TinyML Application: CNN Sensor Analytics for Structural Health Monitoring

Experimental setup:

Sensor and noise positions

Training dataset

Testing dataset

Model architecture

CNN-regression model

Output tensor (x, y)

FC (2), Linear	
FC (E = 64), ReLu	
FC (D = 196), ReLu	
Flatten	
2 x 2 MaxPool, stride 2	
BatchNormalization	
3 x 3 Conv (C = 60), ReLu	Conv_C
2 x 2 MaxPool, stride 2	
BatchNormalization	
3 x 3 Conv (B = 55), ReLu	Conv_B
2 x 2 MaxPool, stride 2	
BatchNormalization	
3 x 3 Conv (A = 50), ReLu	Conv_A
Input tensor (F x T x S)	

TensorFlow Lite

Accelerating 2D Convolution with Hybrid Floating-Point 6-bit

TinyML Application: CNN Sensor Analytics for Structural Health Monitoring

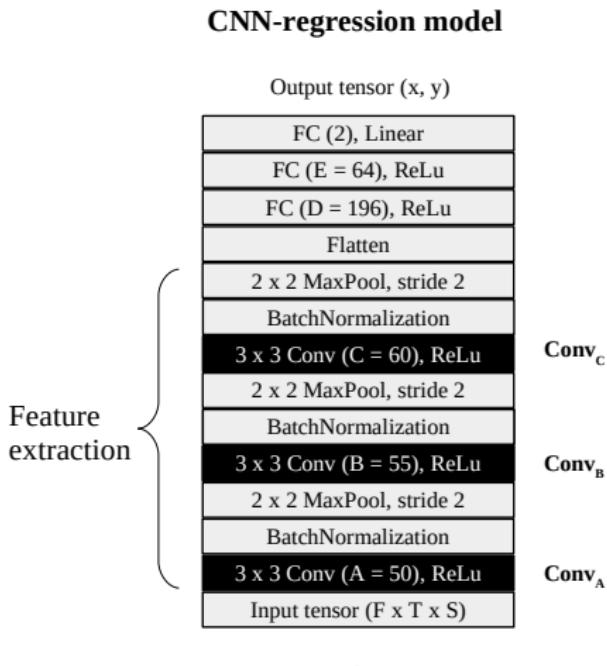
Experimental setup:

Sensor and noise positions

Training dataset

Testing dataset

Model architecture



Accelerating 2D Convolution with Hybrid Floating-Point 6-bit

TinyML Application: CNN Sensor Analytics for Structural Health Monitoring

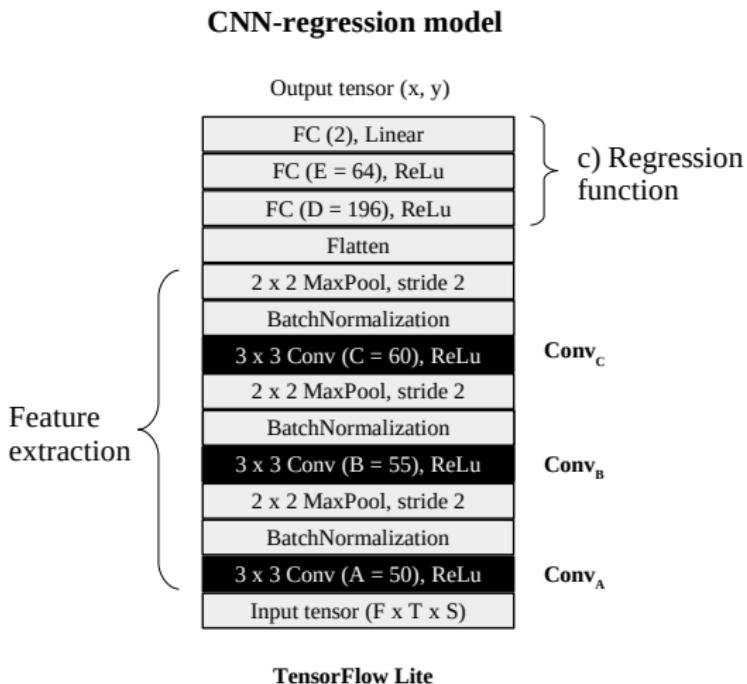
Experimental setup:

Sensor and noise positions

Training dataset

Testing dataset

Model architecture



Accelerating 2D Convolution with Hybrid Floating-Point 6-bit

TinyML Application: CNN Sensor Analytics for Structural Health Monitoring

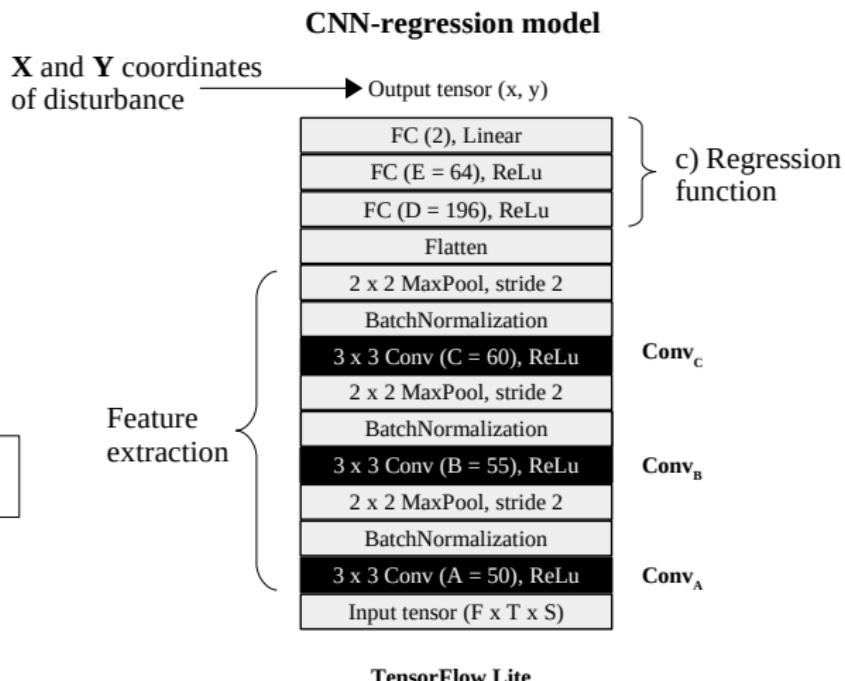
Experimental setup:

Sensor and noise positions

Training dataset

Testing dataset

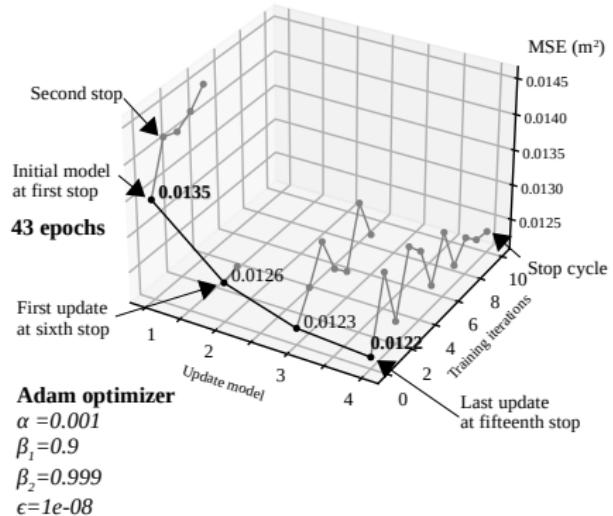
Model architecture



Accelerating 2D Convolution with Hybrid Floating-Point 6-bit

Multi-Phase Model Optimization

Training with iterative early stop
with Adam resets



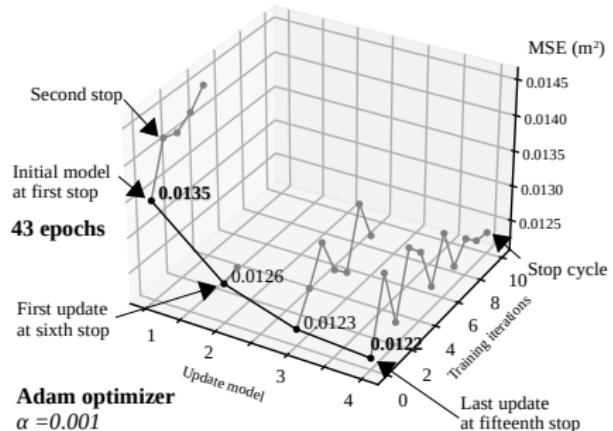
Loss function = MSE

Early stop = 10 epochs
Mini batch = 512 samples

Accelerating 2D Convolution with Hybrid Floating-Point 6-bit

Multi-Phase Model Optimization

Training with iterative early stop
with Adam resets



Adam optimizer

$$\alpha = 0.001$$

$$\beta_1 = 0.9$$

$$\beta_2 = 0.999$$

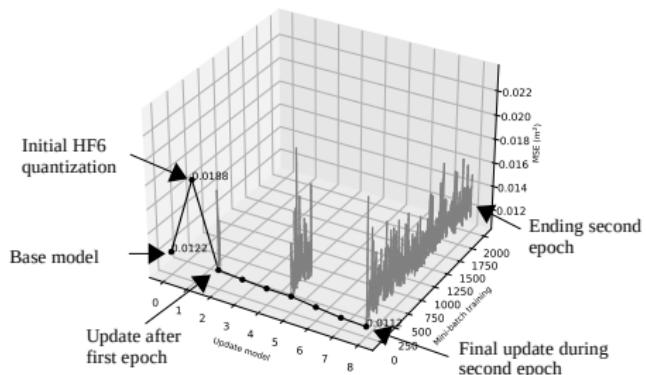
$$\epsilon = 1e-08$$

Loss function = MSE

Early stop = 10 epochs

Mini batch = 512 samples

Quantization-aware training for
hybrid floating-point 6-bit



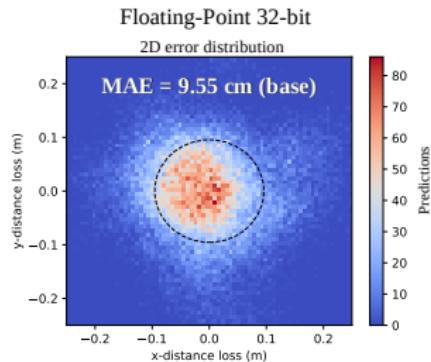
Total time = 53 minutes

CPU = AMD Ryzen 5600H

GPU = NVIDIA GeForce RTX 3050

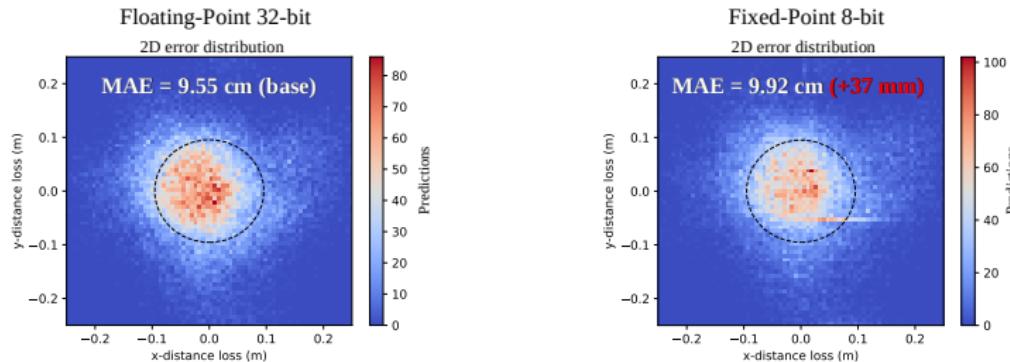
Accelerating 2D Convolution with Hybrid Floating-Point 6-bit

Assessing X and Y Coordinate Prediction Accuracy Across Multiple Quantization Strategies



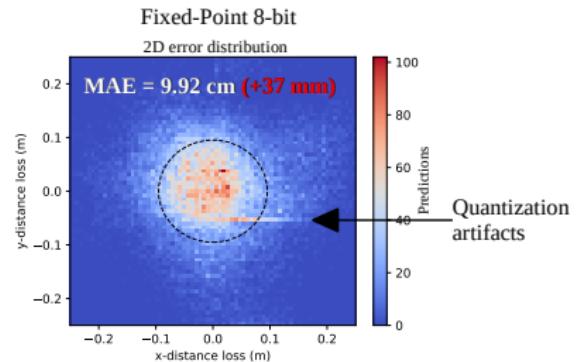
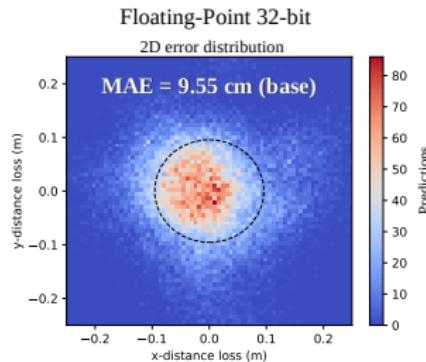
Accelerating 2D Convolution with Hybrid Floating-Point 6-bit

Assessing X and Y Coordinate Prediction Accuracy Across Multiple Quantization Strategies



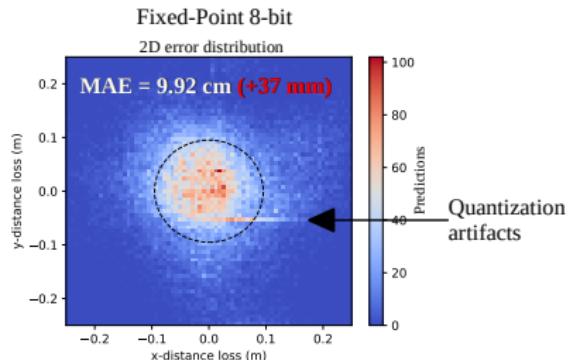
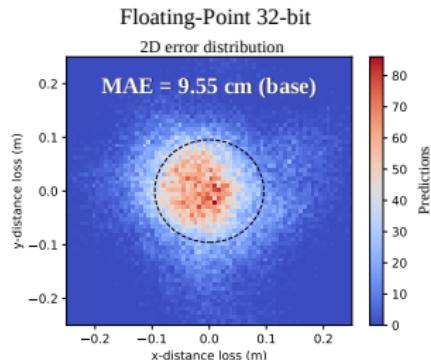
Accelerating 2D Convolution with Hybrid Floating-Point 6-bit

Assessing X and Y Coordinate Prediction Accuracy Across Multiple Quantization Strategies

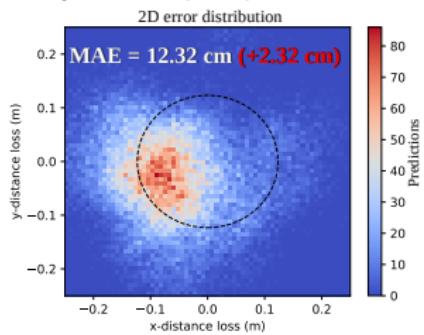


Accelerating 2D Convolution with Hybrid Floating-Point 6-bit

Assessing X and Y Coordinate Prediction Accuracy Across Multiple Quantization Strategies

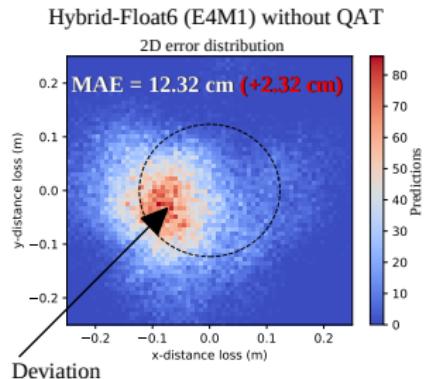
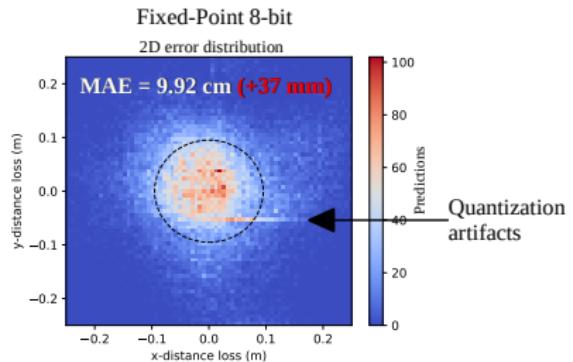
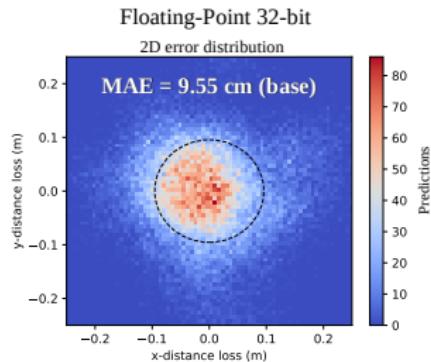


Hybrid-Float6 (E4M1) without QAT



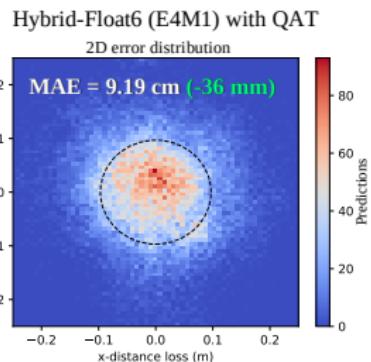
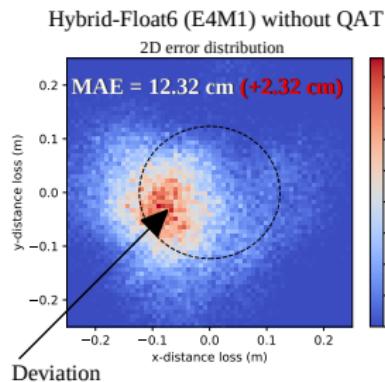
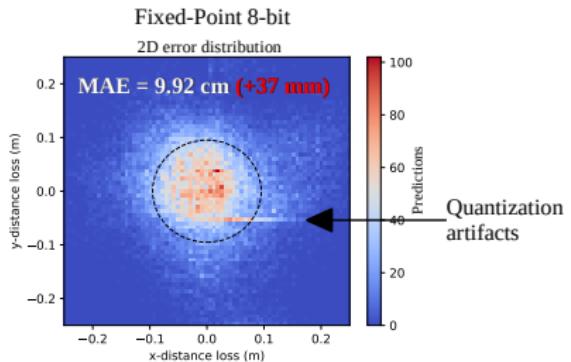
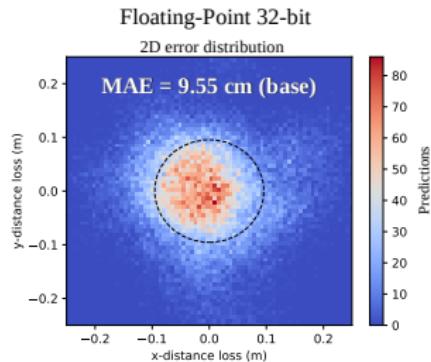
Accelerating 2D Convolution with Hybrid Floating-Point 6-bit

Assessing X and Y Coordinate Prediction Accuracy Across Multiple Quantization Strategies



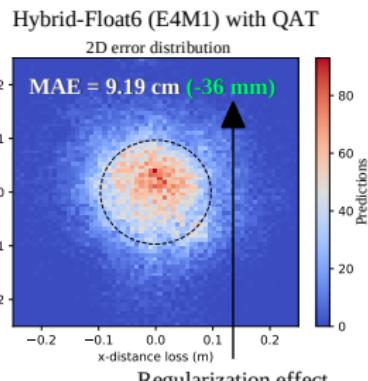
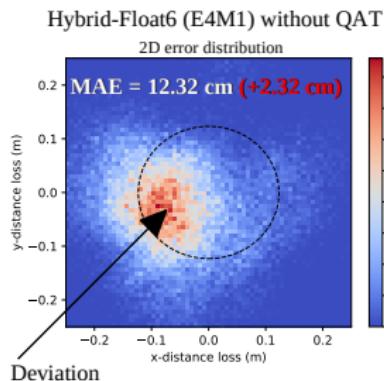
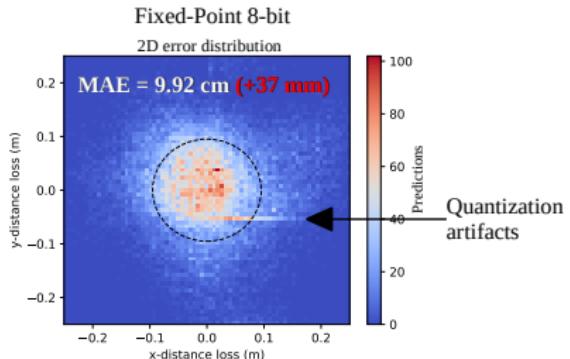
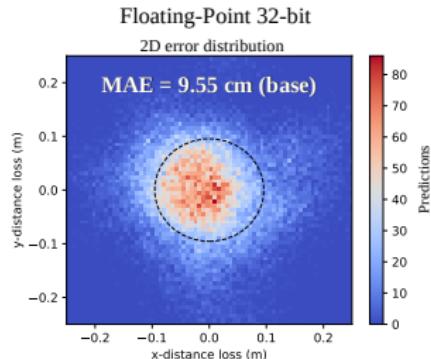
Accelerating 2D Convolution with Hybrid Floating-Point 6-bit

Assessing X and Y Coordinate Prediction Accuracy Across Multiple Quantization Strategies



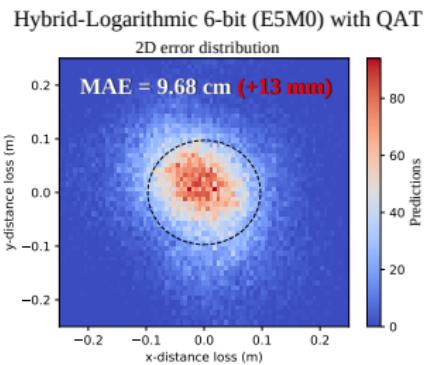
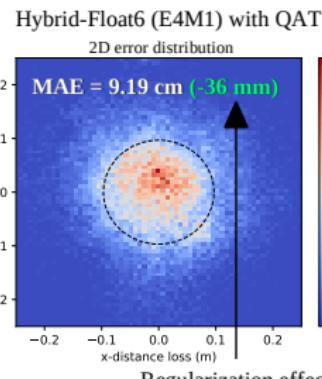
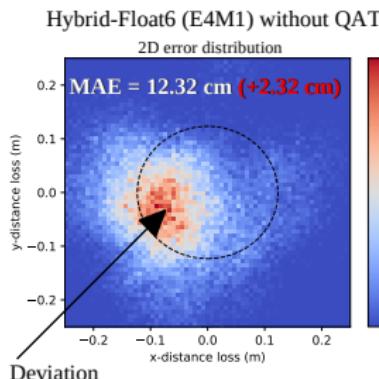
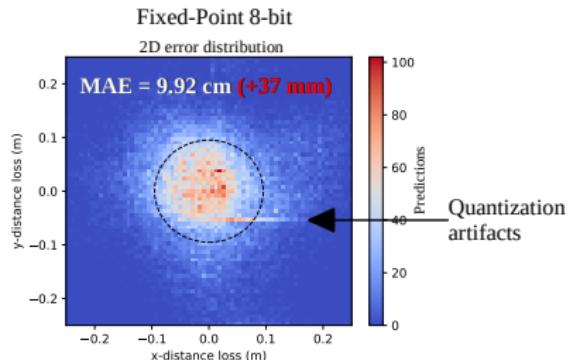
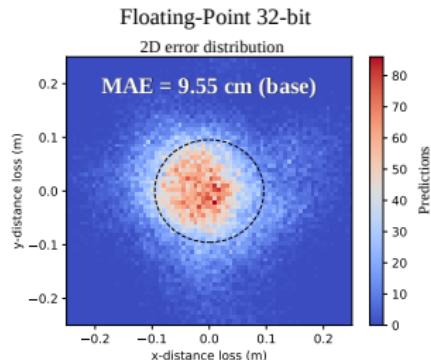
Accelerating 2D Convolution with Hybrid Floating-Point 6-bit

Assessing X and Y Coordinate Prediction Accuracy Across Multiple Quantization Strategies



Accelerating 2D Convolution with Hybrid Floating-Point 6-bit

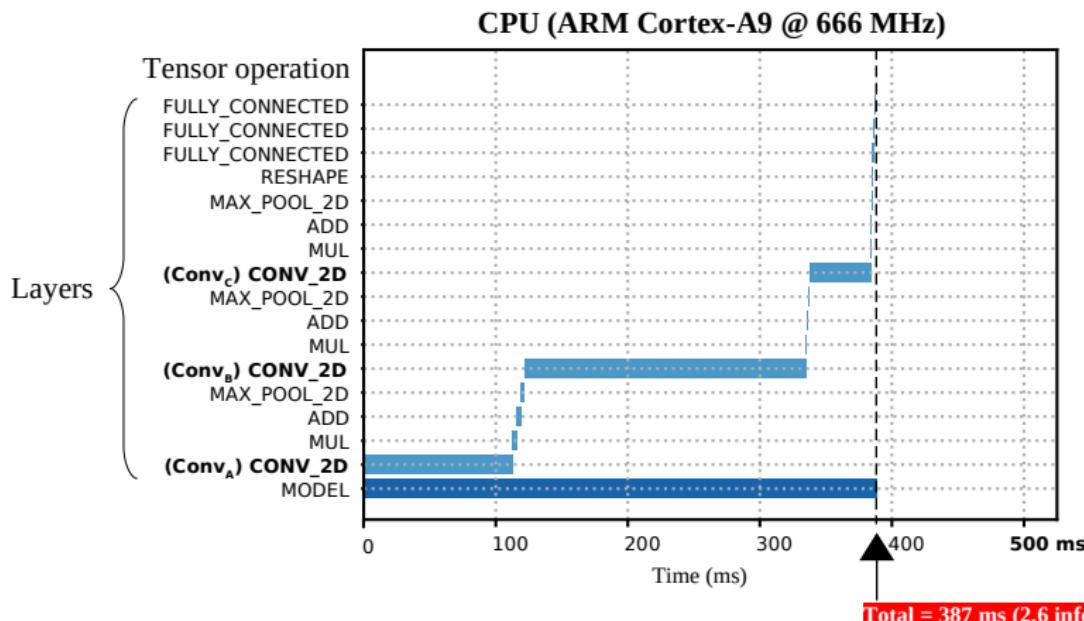
Assessing X and Y Coordinate Prediction Accuracy Across Multiple Quantization Strategies



Accelerating 2D Convolution with Hybrid Floating-Point 6-bit

Runtime Profiling: Latencies and Scheduling on Floating-Point and Hybrid Float 6 Accelerators

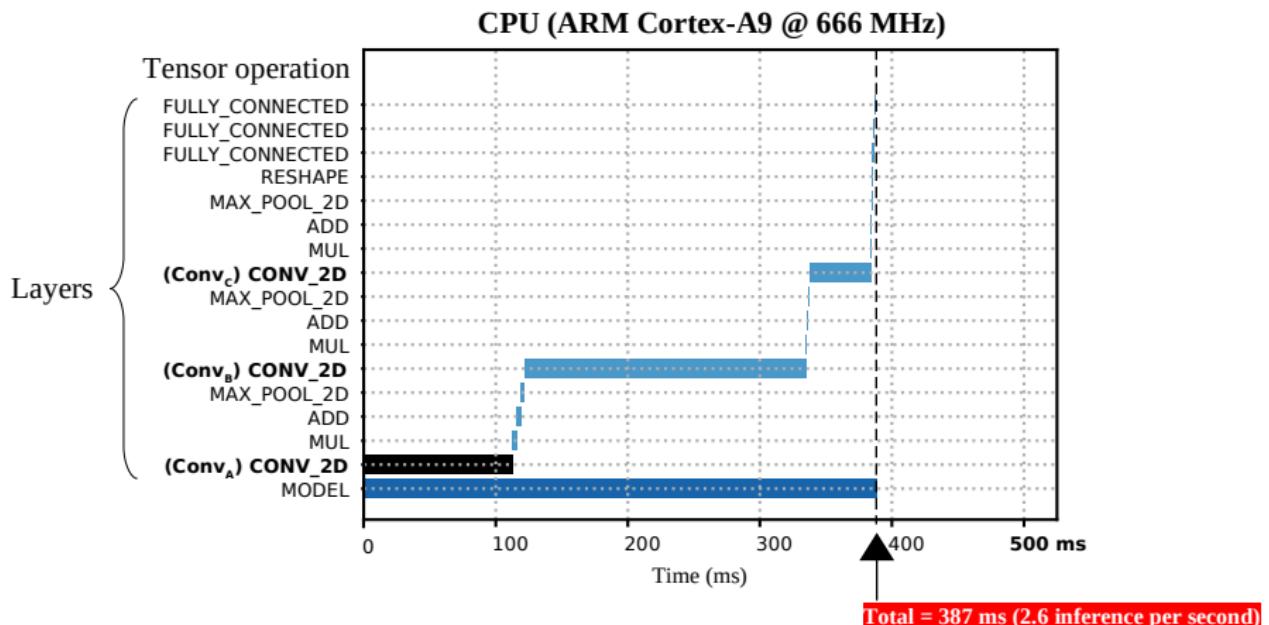
Tensor Op	CPU (ms)	TP FP32 (ms)	Gain	TP HF6 (ms)	Gain
Conv _A					
Conv _B					
Conv _C					



Accelerating 2D Convolution with Hybrid Floating-Point 6-bit

Runtime Profiling: Latencies and Scheduling on Floating-Point and Hybrid Float 6 Accelerators

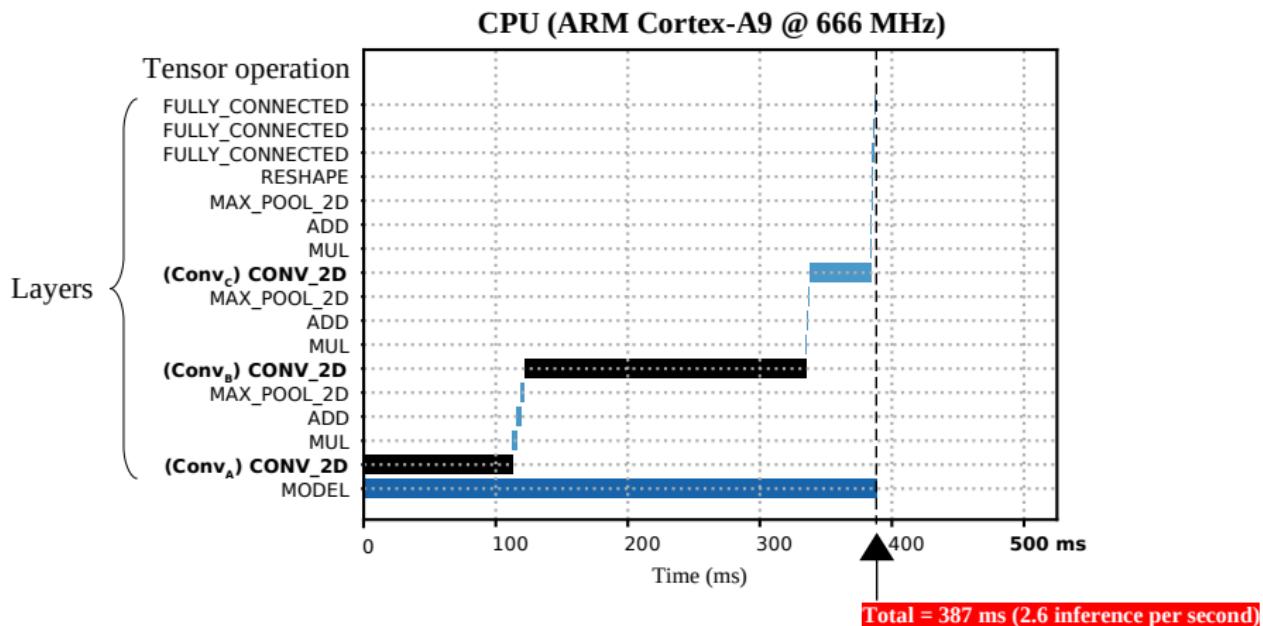
Tensor Op	CPU (ms)	TP FP32 (ms)	Gain	TP HF6 (ms)	Gain
Conv _A	112.24				
Conv _B					
Conv _C					



Accelerating 2D Convolution with Hybrid Floating-Point 6-bit

Runtime Profiling: Latencies and Scheduling on Floating-Point and Hybrid Float 6 Accelerators

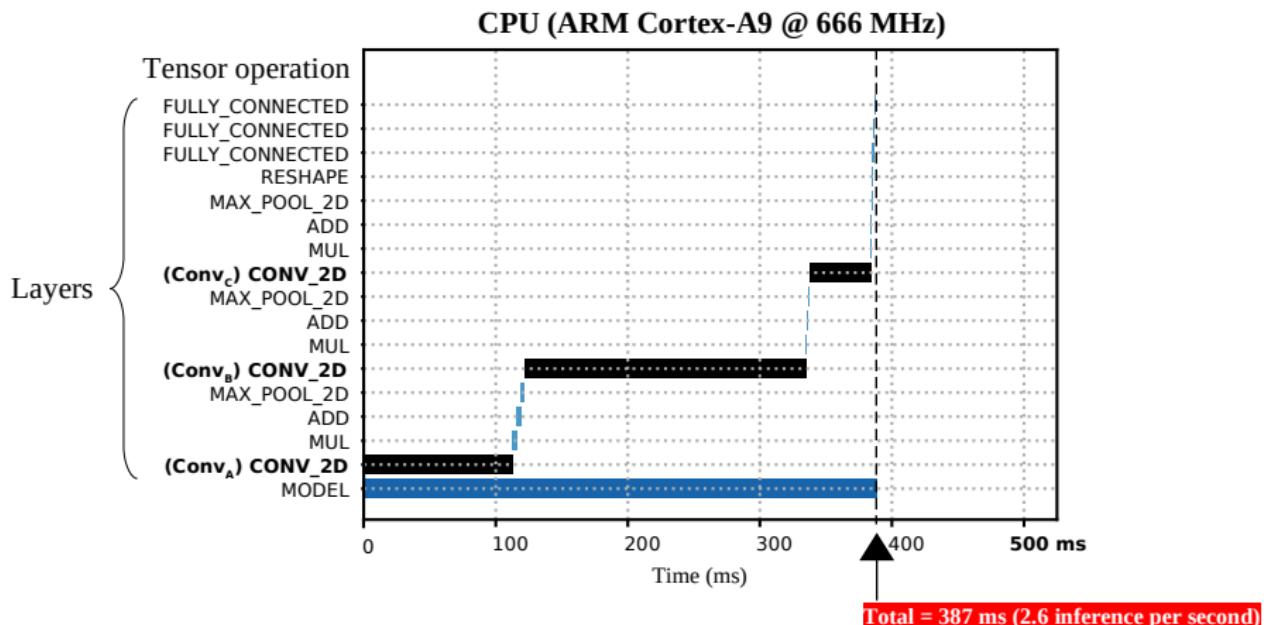
Tensor Op	CPU (ms)	TP FP32 (ms)	Gain	TP HF6 (ms)	Gain
Conv _A	112.24				
Conv _B	213.13				
Conv _C					



Accelerating 2D Convolution with Hybrid Floating-Point 6-bit

Runtime Profiling: Latencies and Scheduling on Floating-Point and Hybrid Float 6 Accelerators

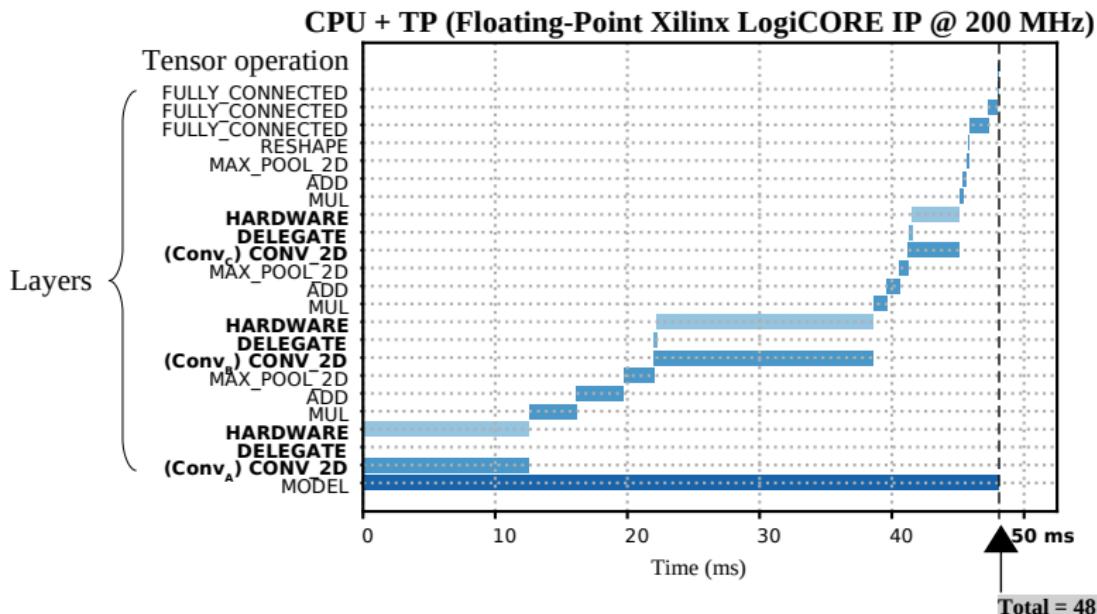
Tensor Op	CPU (ms)	TP FP32 (ms)	Gain	TP HF6 (ms)	Gain
Conv _A	112.24				
Conv _B	213.13				
Conv _C	46.59				



Accelerating 2D Convolution with Hybrid Floating-Point 6-bit

Runtime Profiling: Latencies and Scheduling on Floating-Point and Hybrid Float 6 Accelerators

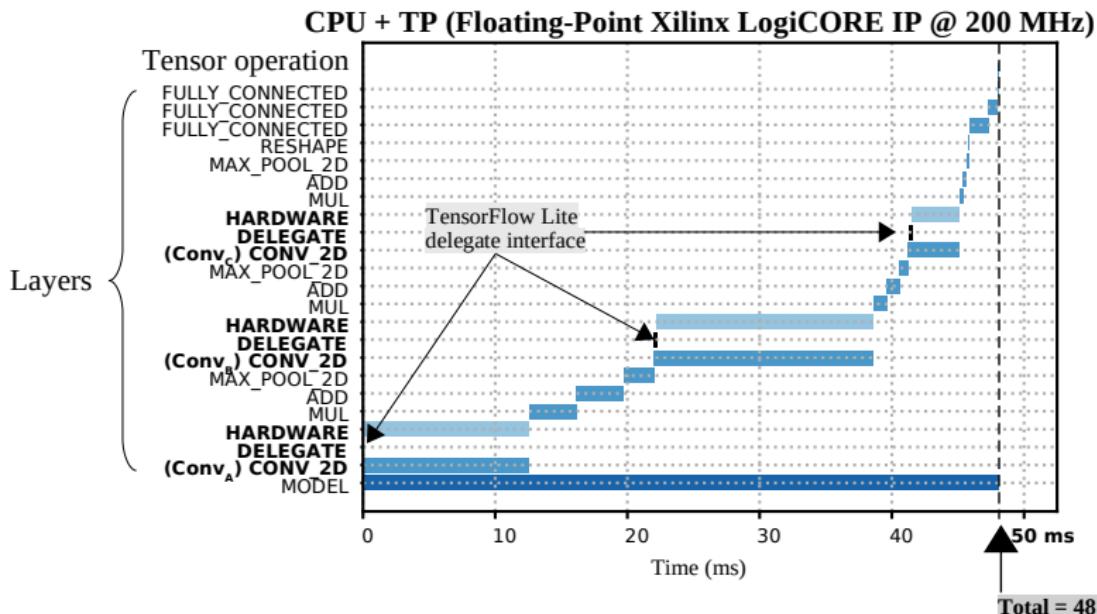
Tensor Op	CPU (ms)	TP FP32 (ms)	Gain	TP HF6 (ms)	Gain
Conv _A	112.24				
Conv _B	213.13				
Conv _C	46.59				



Accelerating 2D Convolution with Hybrid Floating-Point 6-bit

Runtime Profiling: Latencies and Scheduling on Floating-Point and Hybrid Float 6 Accelerators

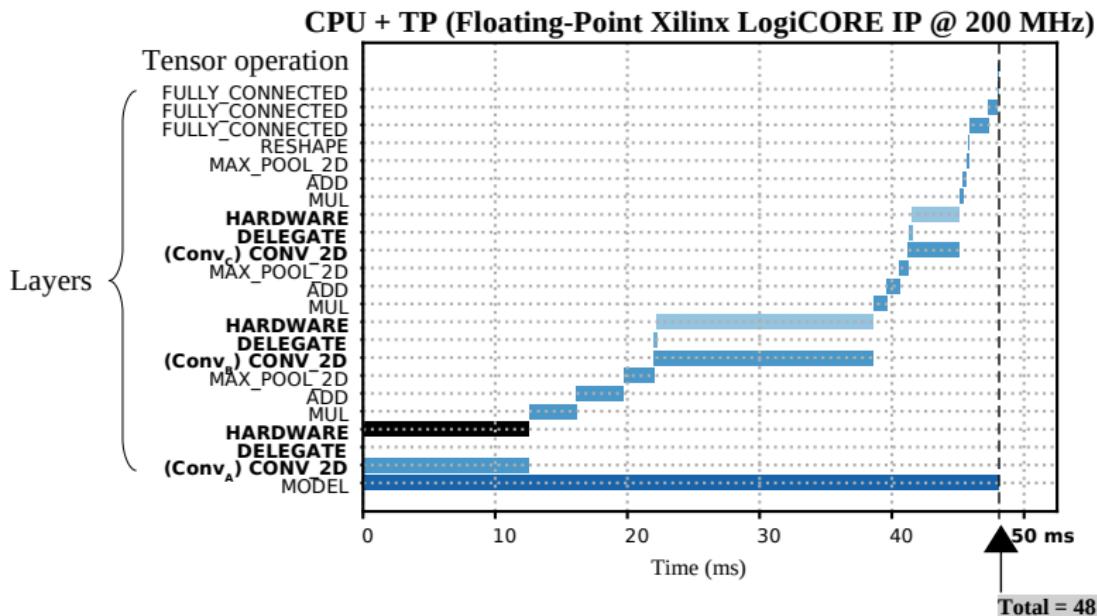
Tensor Op	CPU (ms)	TP FP32 (ms)	Gain	TP HF6 (ms)	Gain
Conv _A	112.24				
Conv _B	213.13				
Conv _C	46.59				



Accelerating 2D Convolution with Hybrid Floating-Point 6-bit

Runtime Profiling: Latencies and Scheduling on Floating-Point and Hybrid Float 6 Accelerators

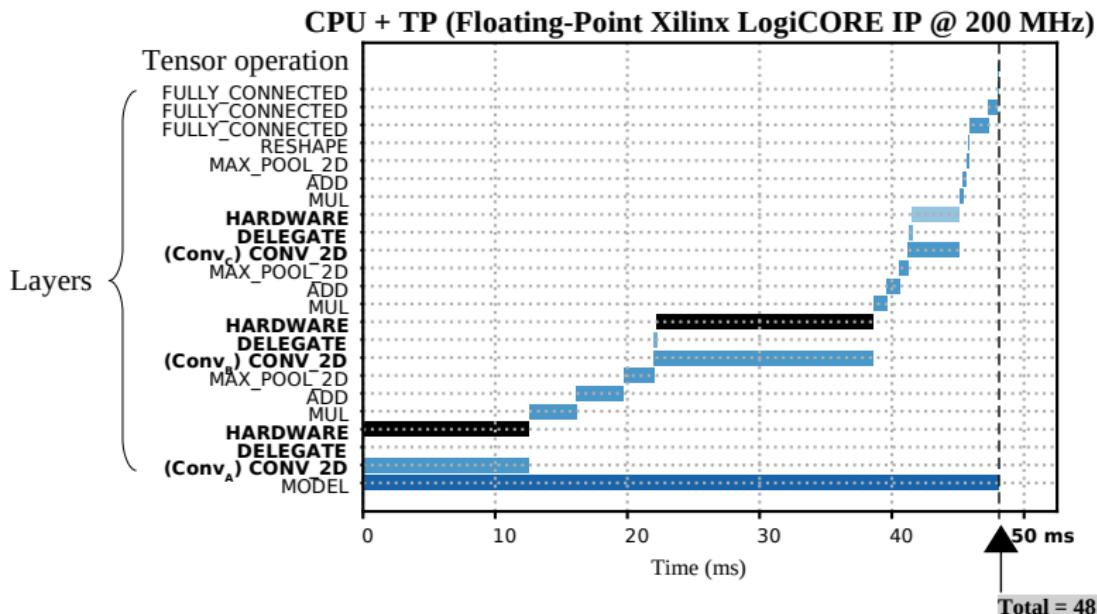
Tensor Op	CPU (ms)	TP FP32 (ms)	Gain	TP HF6 (ms)	Gain
Conv _A	112.24	12.49	8.9		
Conv _B	213.13				
Conv _C	46.59				



Accelerating 2D Convolution with Hybrid Floating-Point 6-bit

Runtime Profiling: Latencies and Scheduling on Floating-Point and Hybrid Float 6 Accelerators

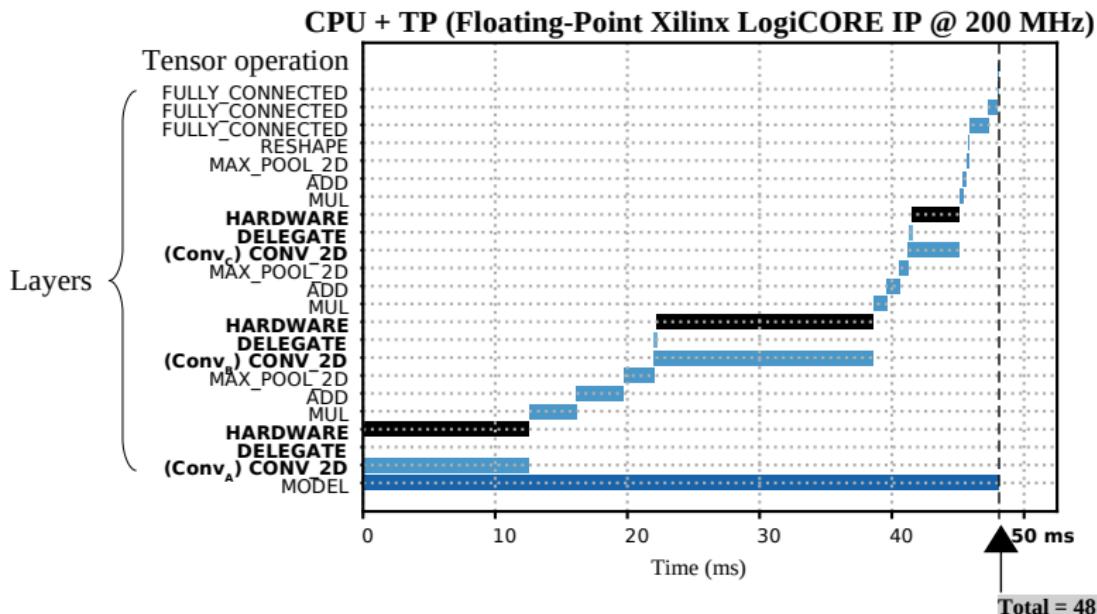
Tensor Op	CPU (ms)	TP FP32 (ms)	Gain	TP HF6 (ms)	Gain
Conv _A	112.24	12.49	8.9		
Conv _B	213.13	16.39	13.0		
Conv _C	46.59				



Accelerating 2D Convolution with Hybrid Floating-Point 6-bit

Runtime Profiling: Latencies and Scheduling on Floating-Point and Hybrid Float 6 Accelerators

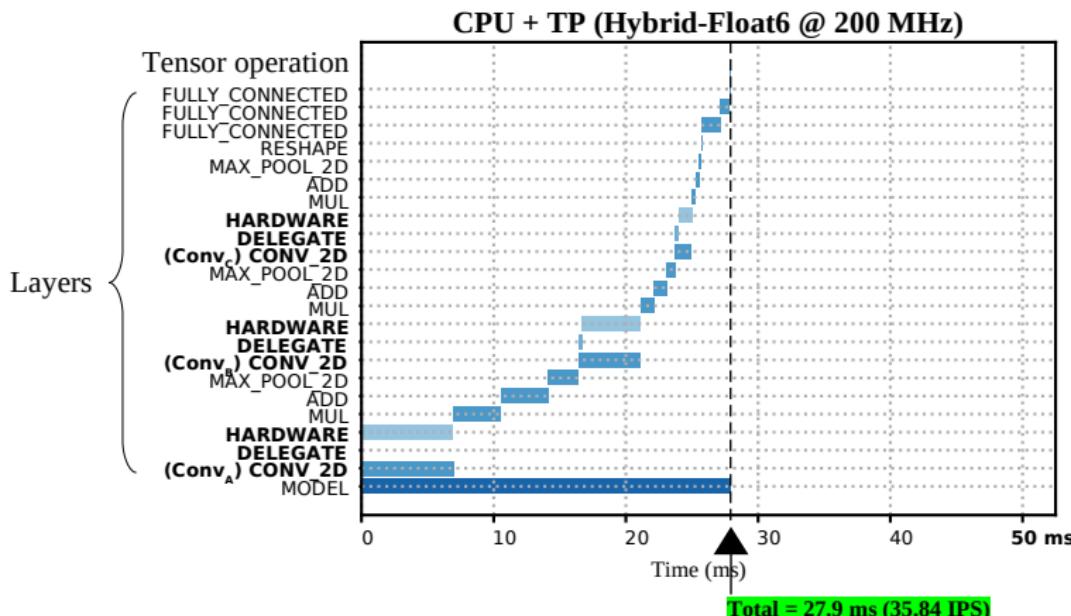
Tensor Op	CPU (ms)	TP FP32 (ms)	Gain	TP HF6 (ms)	Gain
Conv _A	112.24	12.49	8.9		
Conv _B	213.13	16.39	13.0		
Conv _C	46.59	3.59	12.9		



Accelerating 2D Convolution with Hybrid Floating-Point 6-bit

Runtime Profiling: Latencies and Scheduling on Floating-Point and Hybrid Float 6 Accelerators

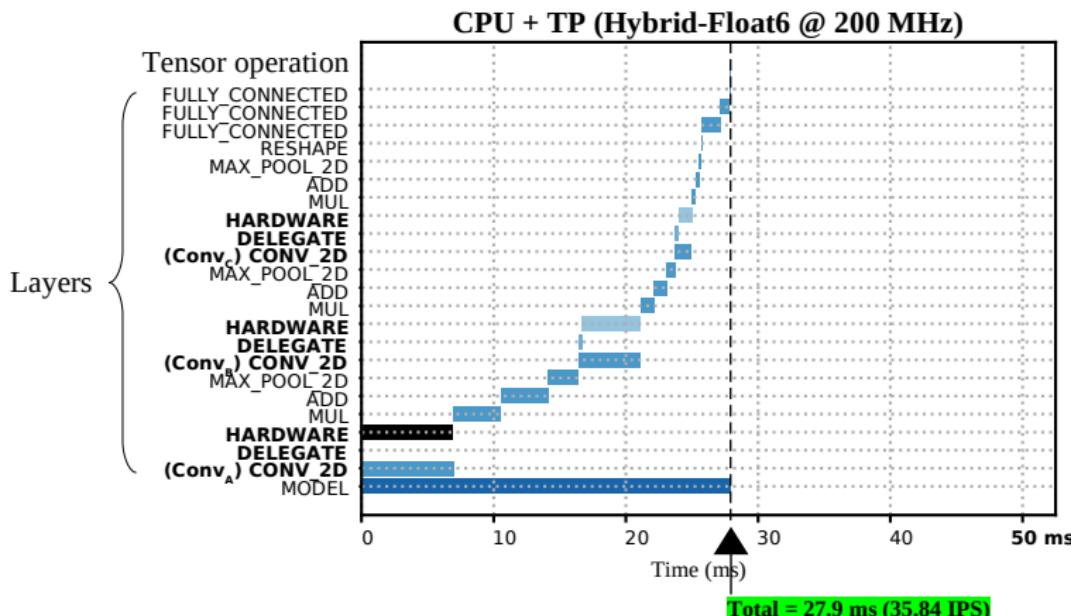
Tensor Op	CPU (ms)	TP FP32 (ms)	Gain	TP HF6 (ms)	Gain
Conv _A	112.24	12.49	8.9		
Conv _B	213.13	16.39	13.0		
Conv _C	46.59	3.59	12.9		



Accelerating 2D Convolution with Hybrid Floating-Point 6-bit

Runtime Profiling: Latencies and Scheduling on Floating-Point and Hybrid Float 6 Accelerators

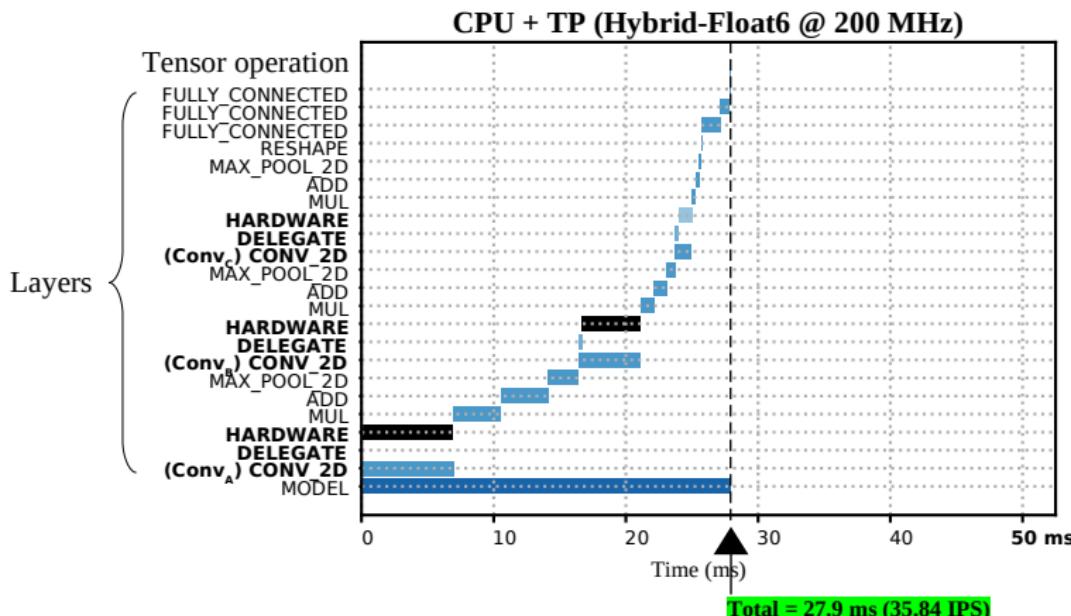
Tensor Op	CPU (ms)	TP FP32 (ms)	Gain	TP HF6 (ms)	Gain
Conv _A	112.24	12.49	8.9	6.92	16.2
Conv _B	213.13	16.39	13.0		
Conv _C	46.59	3.59	12.9		



Accelerating 2D Convolution with Hybrid Floating-Point 6-bit

Runtime Profiling: Latencies and Scheduling on Floating-Point and Hybrid Float 6 Accelerators

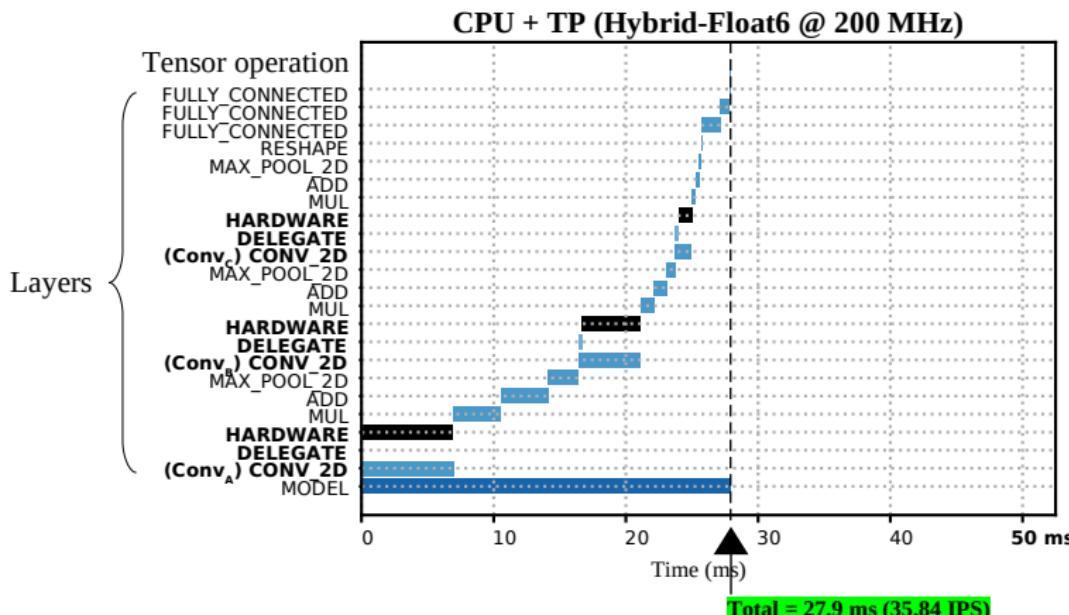
Tensor Op	CPU (ms)	TP FP32 (ms)	Gain	TP HF6 (ms)	Gain
Conv _A	112.24	12.49	8.9	6.92	16.2
Conv _B	213.13	16.39	13.0	4.41	48.3
Conv _C	46.59	3.59	12.9		



Accelerating 2D Convolution with Hybrid Floating-Point 6-bit

Runtime Profiling: Latencies and Scheduling on Floating-Point and Hybrid Float 6 Accelerators

Tensor Op	CPU (ms)	TP FP32 (ms)	Gain	TP HF6 (ms)	Gain
Conv _A	112.24	12.49	8.9	6.92	16.2
Conv _B	213.13	16.39	13.0	4.41	48.3
Conv _C	46.59	3.59	12.9	0.99	47.0

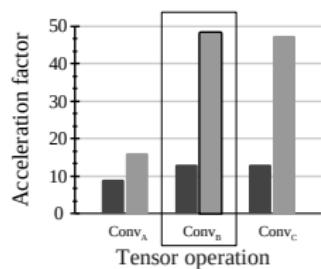


Accelerating 2D Convolution with Hybrid Floating-Point 6-bit

Final Benchmarking Insights: Accelerators vs. CPU in Performance and Power Efficiency

■ Floating-Point
■ Hybrid-Float6

Acceleration vs. CPU

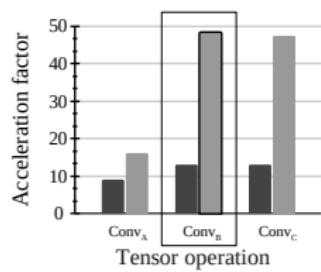


Accelerating 2D Convolution with Hybrid Floating-Point 6-bit

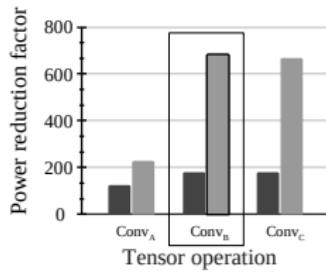
Final Benchmarking Insights: Accelerators vs. CPU in Performance and Power Efficiency

Floating-Point
Hybrid-Float6

Acceleration vs. CPU



Power reduction vs. CPU

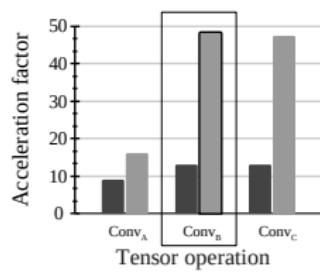


Accelerating 2D Convolution with Hybrid Floating-Point 6-bit

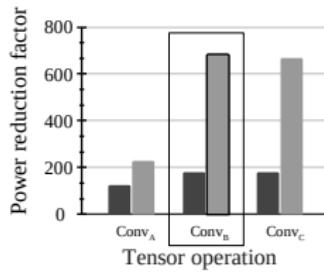
Final Benchmarking Insights: Accelerators vs. CPU in Performance and Power Efficiency

Floating-Point
Hybrid-Float6

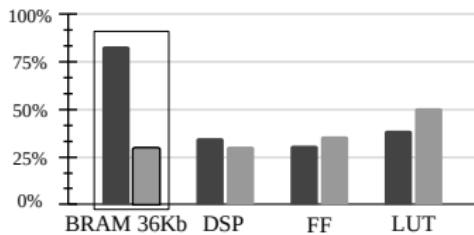
Acceleration vs. CPU



Power reduction vs. CPU



Hardware resource utilization.



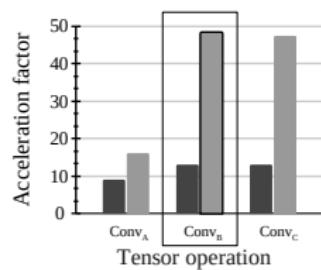
Accelerating 2D Convolution with Hybrid Floating-Point 6-bit

Final Benchmarking Insights: Accelerators vs. CPU in Performance and Power Efficiency

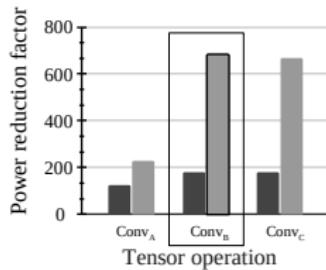
■ Floating-Point
■ Hybrid-Float6

Smallest 2D CNN floating-point accelerator in the literature

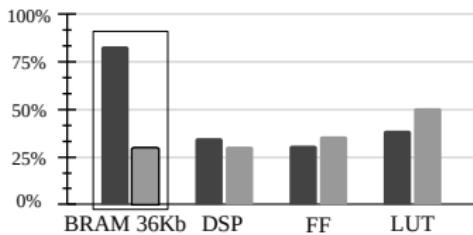
Acceleration vs. CPU



Power reduction vs. CPU



Hardware resource utilization.



1 Methodology

2 Floating-Point Multiply-Accumulate Unit

3 Case Studies

4 Conclusions

Conclusion

- Presentation of HW/SW design methodology

Conclusion

- Presentation of HW/SW design methodology
- Advancements in low-precision floating-point techniques

Conclusion

- Presentation of HW/SW design methodology
- Advancements in low-precision floating-point techniques
- Demonstration of iterative on-device optimization

Conclusion

- Presentation of HW/SW design methodology
- Advancements in low-precision floating-point techniques
- Demonstration of iterative on-device optimization
- Presentation of TinyML application

Publications

Journal Articles

- **Yarib Nevarez**, David Rotermund, Klaus R Pawelzik, and Alberto Garcia-Ortiz, "Accelerating Spike-by-Spike Neural Networks on FPGA With Hybrid Custom Floating-Point and Logarithmic Dot-Product Approximation," *IEEE Access*, vol. 9, pp. 80603–80620, May 2021, doi: [10.1109/ACCESS.2021.3085216](https://doi.org/10.1109/ACCESS.2021.3085216).
- **Yarib Nevarez**, Andreas Beering, Amir Najafi, Ardalan Najafi, Wanli Yu, Yizhi Chen, Karl-Ludwig Krieger, and Alberto Garcia-Ortiz, "CNN Sensor Analytics With Hybrid-Float6 Quantization on Low-Power Embedded FPGAs," *IEEE Access*, vol. 11, pp. 4852–4868, January 2023, doi: [10.1109/ACCESS.2023.3235866](https://doi.org/10.1109/ACCESS.2023.3235866).

Conference Proceedings

- **Yarib Nevarez**, Alberto Garcia-Ortiz, David Rotermund, and Klaus R Pawelzik, "Accelerator framework of spike-by-spike neural networks for inference and incremental learning in embedded systems," 2020 9th International Conference on Modern Circuits and Systems Technologies (MOCAST), Bremen, 2020, pp. 1–5, doi: [10.1109/MOCAST49295.2020.9200288](https://doi.org/10.1109/MOCAST49295.2020.9200288).
- Wanli Yu, Ardalan Najafi, **Yarib Nevarez**, Yanqiu Huang and Alberto Garcia-Ortiz, "TAAC: Task Allocation Meets Approximate Computing for Internet of Things," 2020 IEEE International Symposium on Circuits and Systems (ISCAS), Sevilla, 2020, pp. 1-5, doi: [10.1109/ISCAS45731.2020.9180895](https://doi.org/10.1109/ISCAS45731.2020.9180895).

Publications

- Amir Najafi, Ardalan Najafi, **Yarib Nevarez** and Alberto Garcia-Ortiz, "Learning-Based On-Chip Parallel Interconnect Delay Estimation," 2022 11th International Conference on Modern Circuits and Systems Technologies (MOCAST), Bremen, 2022, pp. 1–5, doi: 10.1109/MOCAST49295.2020.9200288.
- Yizhi Chen, **Yarib Nevarez**, Zhonghai Lu, and Alberto Garcia-Ortiz, "Accelerating Non-Negative Matrix Factorization on Embedded FPGA with Hybrid Logarithmic Dot-Product Approximation," 2022 IEEE 15th International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSoC), Malaysia, 2022, pp. 239–246, doi: 10.1109/MCSoC57363.2022.00070.
- Ardalan Najafi, Wanli Yu, **Yarib Nevarez**, Amir Najafi, Andreas Beering, Karl-Ludwig Krieger, and Alberto Garcia-Ortiz, "Acoustic Emission Source Localization using Approximate Discrete Wavelet Transform," 2023 12th International Conference on Modern Circuits and Systems Technologies (MOCAST), Bremen, 2023, pp. 1–5, doi: 10.1109/MOCAST57943.2023.10176952.

Thank You for Your Attention