

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

Yarib Nevarez

Universität Bremen

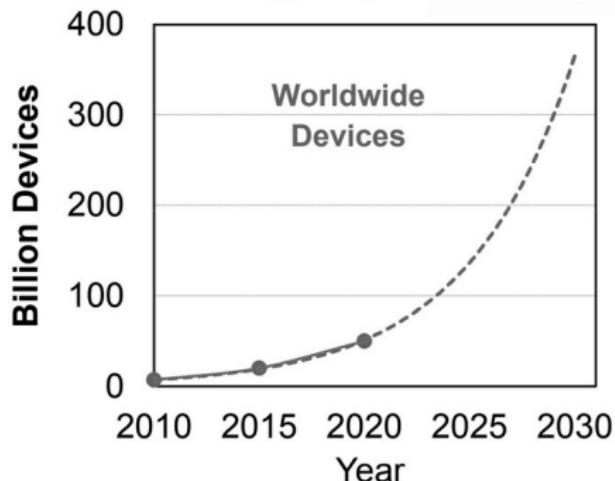
May 22, 2024

# 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



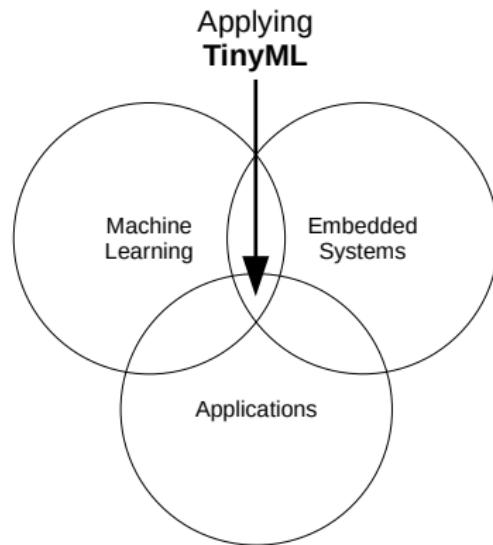
Loh, Kou-Hung Lawrence. "1.2 Fertilizing AIoT from roots to leaves." In 2020 IEEE International Solid-State Circuits Conference-(ISSCC), pp. 15-21. IEEE, 2020.

# Introduction

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

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

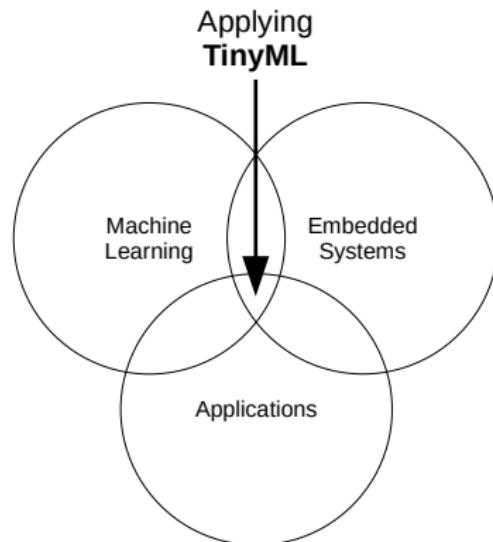
Any device    Anybody    Anywhere    Any business    Any network    Anytime



## 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

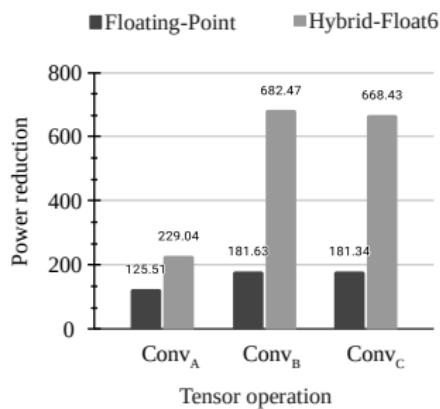
Quality preservation

Application versatility

Platform compatibility

On-device training

#### Power reduction relative to CPU



## 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:

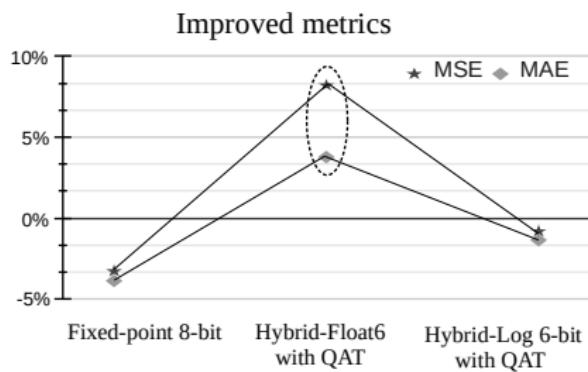
Energy and resource efficiency

**Quality preservation**

Application versatility

Platform compatibility

On-device training



# Introduction

## 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

#### Model versatility

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 ( $F \times T \times S$ )

## 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

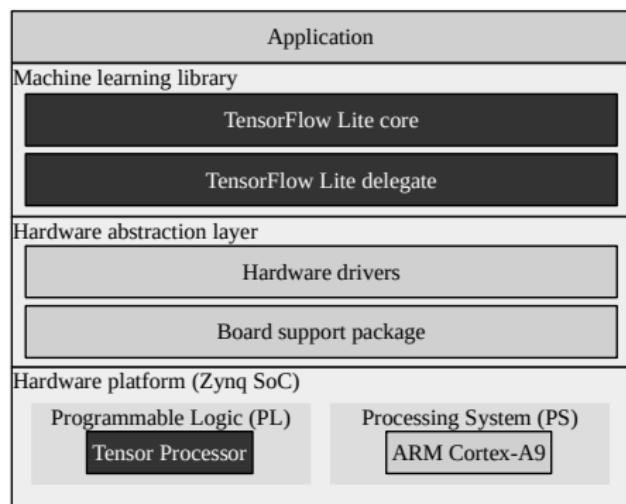
Quality preservation

Application versatility

**Platform compatibility**

On-device training

### HW/SW co-design framework



## 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:

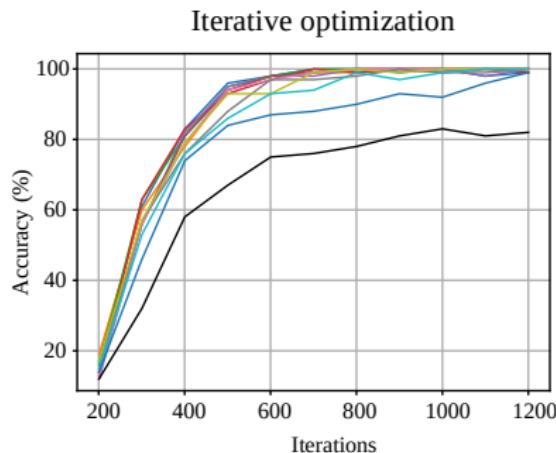
Energy and resource efficiency

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**  
Fails to adequately meet fundamental aspects, particularly in **complex problems** and **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**  
Fails to adequately meet fundamental aspects, particularly in **complex problems** and **mission-critical applications**.
- Fixed precision**  
Fails to adequately adapt to the ongoing technological shift towards **on-device training**.

## Goal and Objectives

- **Goal:** To establish a sustainable, efficient, and universally applicable neural network acceleration technique that supports inference and facilitates on-device training in extreme edge devices, ensuring future-proof and broad applicability. This approach integrates hybrid custom floating-point (FP) computation.

## Goal and Objectives

- **Goal:** To establish a sustainable, efficient, and universally applicable neural network acceleration technique that supports inference and facilitates on-device training in extreme edge devices, ensuring future-proof and broad applicability. This approach integrates hybrid custom floating-point (FP) computation.
- **Objectives:**

## Goal and Objectives

- **Goal:** To establish a sustainable, efficient, and universally applicable neural network acceleration technique that supports inference and facilitates on-device training in extreme edge devices, ensuring future-proof and broad applicability. This approach integrates hybrid custom floating-point (FP) computation.
  
- **Objectives:**
  - Investigate custom FP computation

## Goal and Objectives

- **Goal:** To establish a sustainable, efficient, and universally applicable neural network acceleration technique that supports inference and facilitates on-device training in extreme edge devices, ensuring future-proof and broad applicability. This approach integrates hybrid custom floating-point (FP) computation.
- **Objectives:**
  - Investigate custom FP computation
  - Conduct performance evaluation

## Goal and Objectives

- **Goal:** To establish a sustainable, efficient, and universally applicable neural network acceleration technique that supports inference and facilitates on-device training in extreme edge devices, ensuring future-proof and broad applicability. This approach integrates hybrid custom floating-point (FP) computation.
- **Objectives:**
  - Investigate custom FP computation
  - Conduct performance evaluation
  - Analyze impact

## Goal and Objectives

- **Goal:** To establish a sustainable, efficient, and universally applicable neural network acceleration technique that supports inference and facilitates on-device training in extreme edge devices, ensuring future-proof and broad applicability. This approach integrates hybrid custom floating-point (FP) computation.
- **Objectives:**
  - Investigate custom FP computation
  - Conduct performance evaluation
  - Analyze impact
  - Ensure cross-platform compatibility

# Outline

- 1 Methodology
- 2 Hybrid 8-bit Floating-Point and 4-bit Logarithmic Computation
- 3 Hybrid 6-bit Floating-Point Computation
- 4 Conclusions and Future Research

# Methodology

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

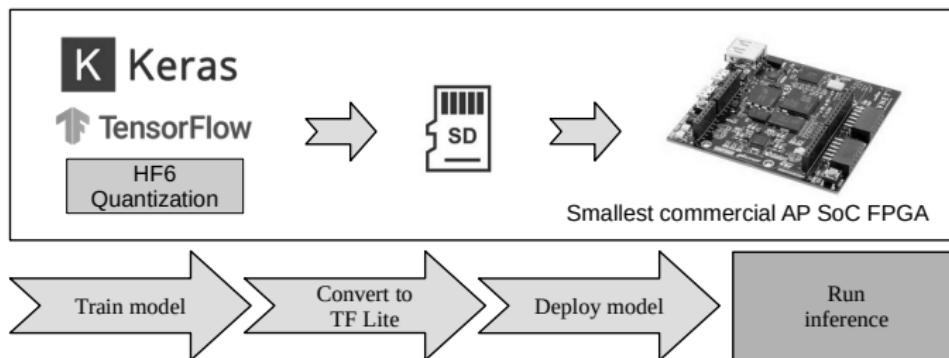
The methodology efficiently deploys and accelerates floating-point neural networks on embedded systems, optimizing performance, energy consumption, and hardware utilization.

Characteristics:

Streamlined hardware architecture

Custom floating-point arithmetic

Neural network execution without quantization processes



# Methodology

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

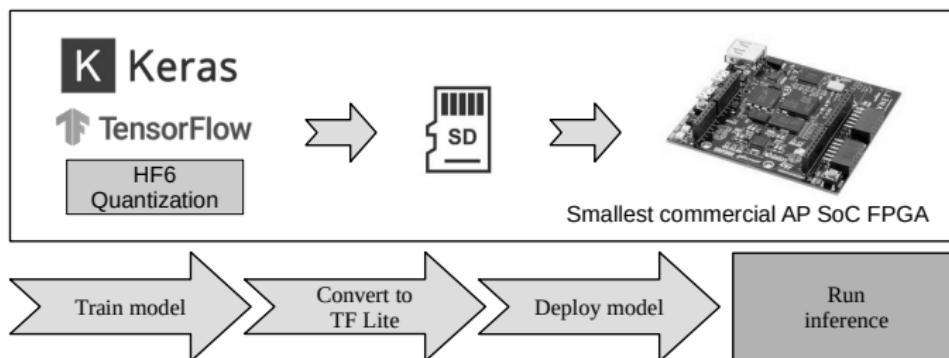
The methodology efficiently deploys and accelerates floating-point neural networks on embedded systems, optimizing performance, energy consumption, and hardware utilization.

### Characteristics:

Streamlined hardware architecture

Custom floating-point arithmetic

Neural network execution without quantization processes



# Methodology

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

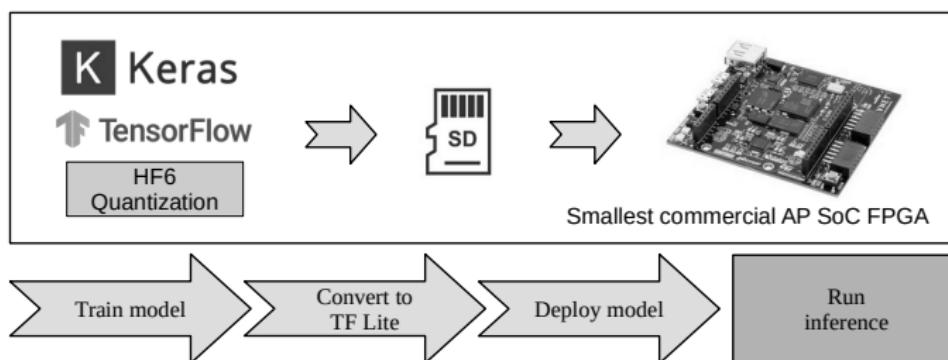
The methodology efficiently deploys and accelerates floating-point neural networks on embedded systems, optimizing performance, energy consumption, and hardware utilization.

### Characteristics:

Streamlined hardware architecture

Custom floating-point arithmetic

Neural network execution without quantization processes



# Methodology

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

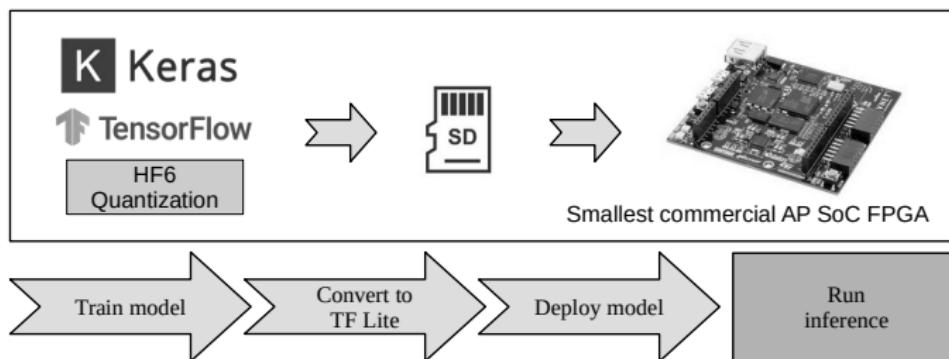
The methodology efficiently deploys and accelerates floating-point neural networks on embedded systems, optimizing performance, energy consumption, and hardware utilization.

### Characteristics:

Streamlined hardware architecture

Custom floating-point arithmetic

Neural network execution without quantization processes



# Methodology

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

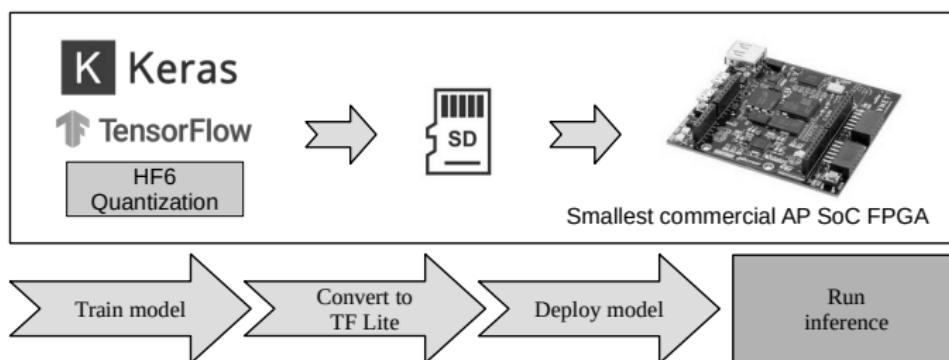
The methodology efficiently deploys and accelerates floating-point neural networks on embedded systems, optimizing performance, energy consumption, and hardware utilization.

### Characteristics:

Streamlined hardware architecture

Custom floating-point arithmetic

Neural network execution without quantization processes



## Methodology

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

The methodology efficiently deploys and accelerates floating-point neural networks on embedded systems, optimizing performance, energy consumption, and hardware utilization.

#### **Abstraction levels:**

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

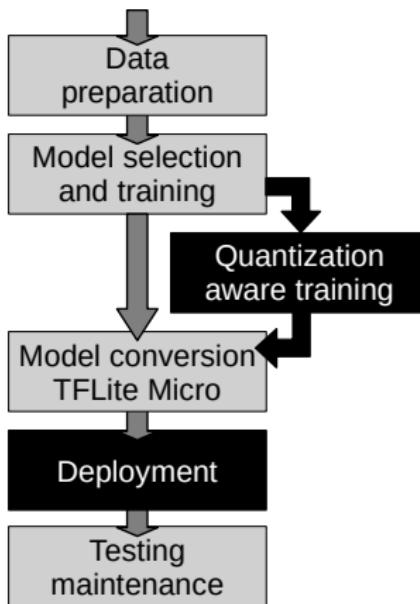
# Methodology

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

The methodology efficiently deploys and accelerates floating-point neural networks on embedded systems, optimizing performance, energy consumption, and hardware utilization.

### Abstraction levels:

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



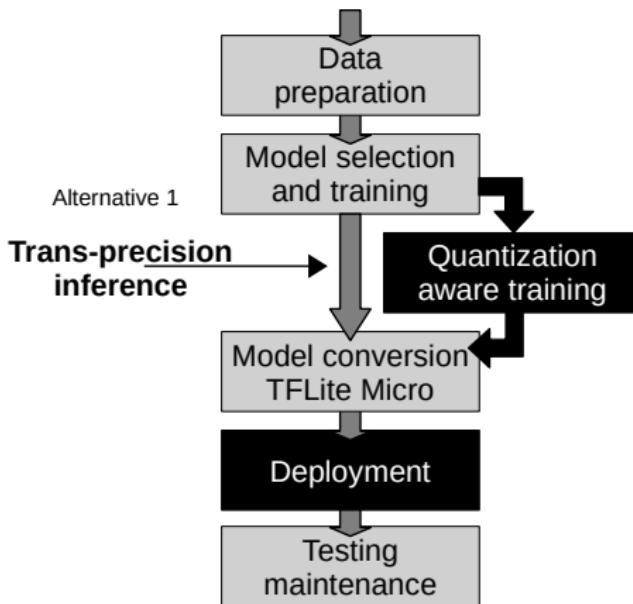
# Methodology

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

The methodology efficiently deploys and accelerates floating-point neural networks on embedded systems, optimizing performance, energy consumption, and hardware utilization.

### Abstraction levels:

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



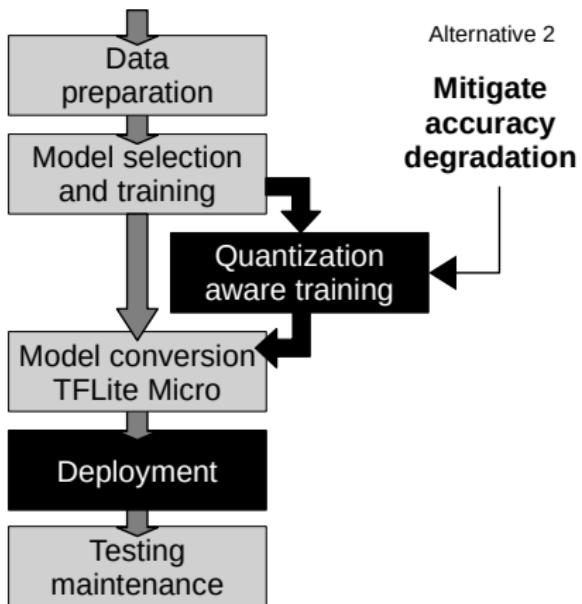
# Methodology

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

The methodology efficiently deploys and accelerates floating-point neural networks on embedded systems, optimizing performance, energy consumption, and hardware utilization.

### Abstraction levels:

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



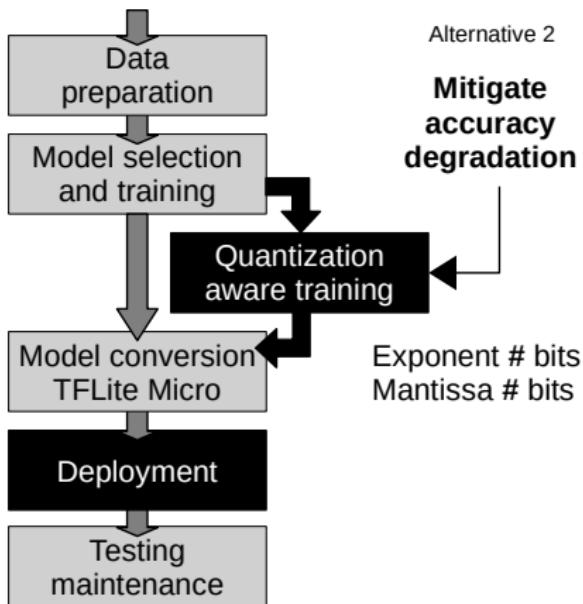
# Methodology

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

The methodology efficiently deploys and accelerates floating-point neural networks on embedded systems, optimizing performance, energy consumption, and hardware utilization.

### 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

The methodology efficiently deploys and accelerates floating-point neural networks on embedded systems, optimizing performance, energy consumption, and hardware utilization.

### Abstraction levels:

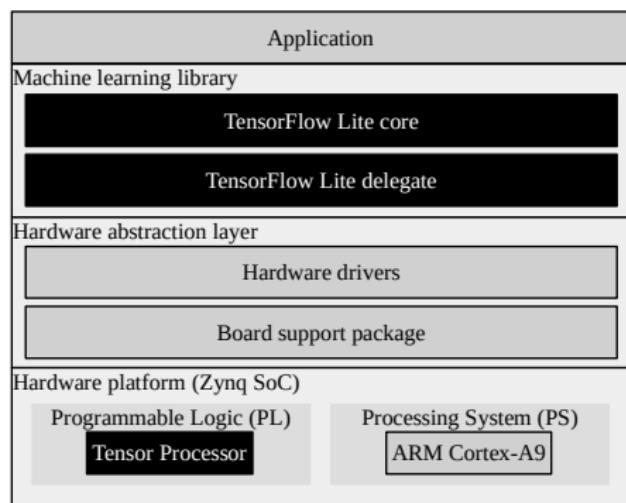
1. Model deployment

### 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

The methodology efficiently deploys and accelerates floating-point neural networks on embedded systems, optimizing performance, energy consumption, and hardware utilization.

### Abstraction levels:

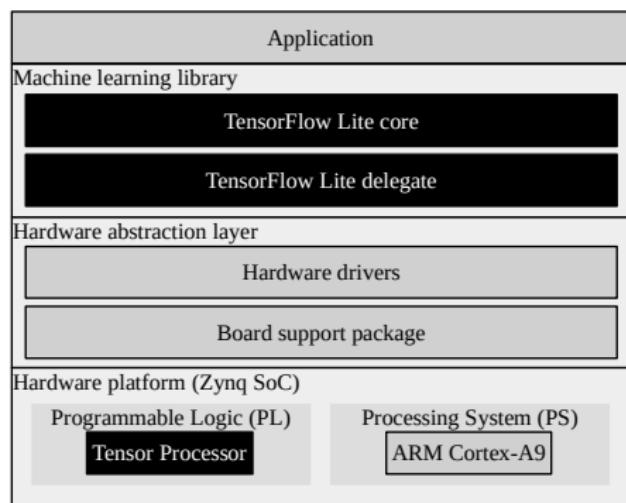
1. Model deployment

**2. System infrastructure**

3. Streamlined acceleration

4. Optimized processing

### HW/SW co-design framework



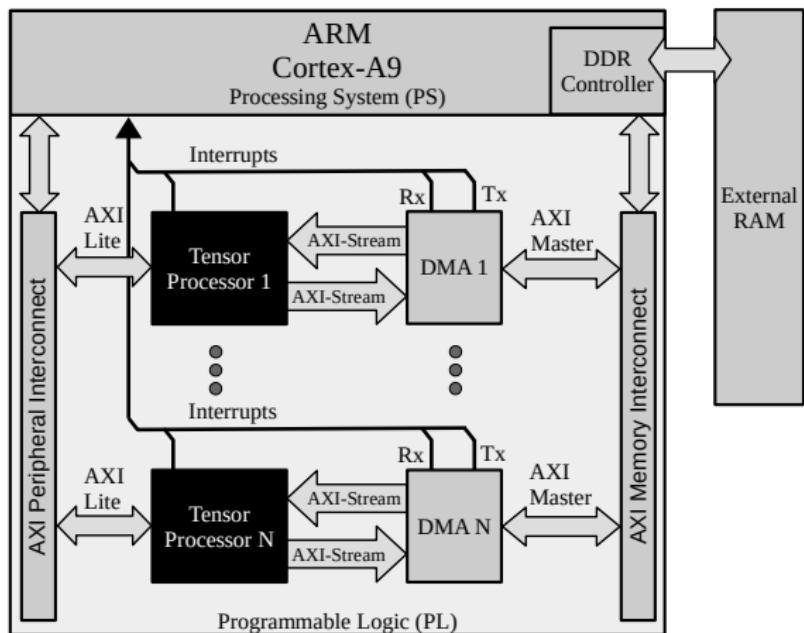
# Methodology

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

The methodology efficiently deploys and accelerates floating-point neural networks on embedded systems, optimizing performance, energy consumption, and hardware utilization.

### Abstraction levels:

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



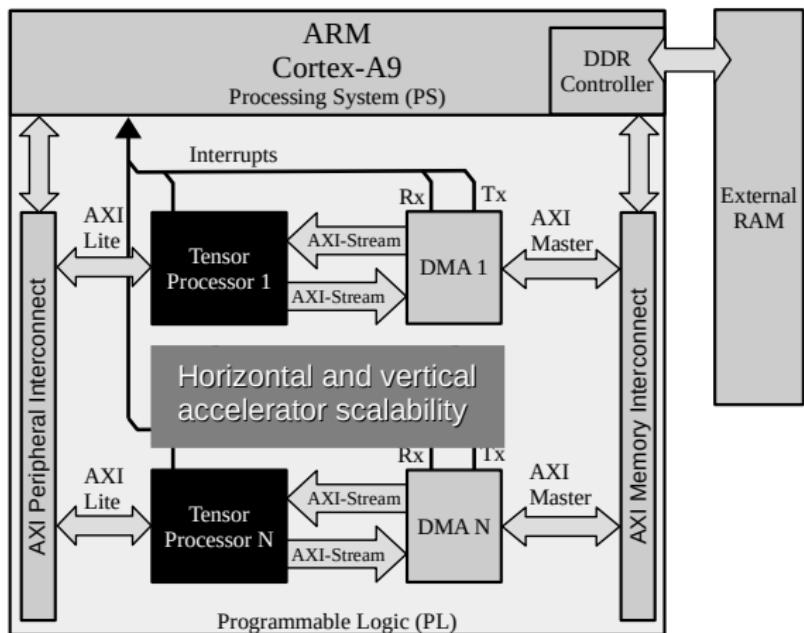
# Methodology

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

The methodology efficiently deploys and accelerates floating-point neural networks on embedded systems, optimizing performance, energy consumption, and hardware utilization.

### 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

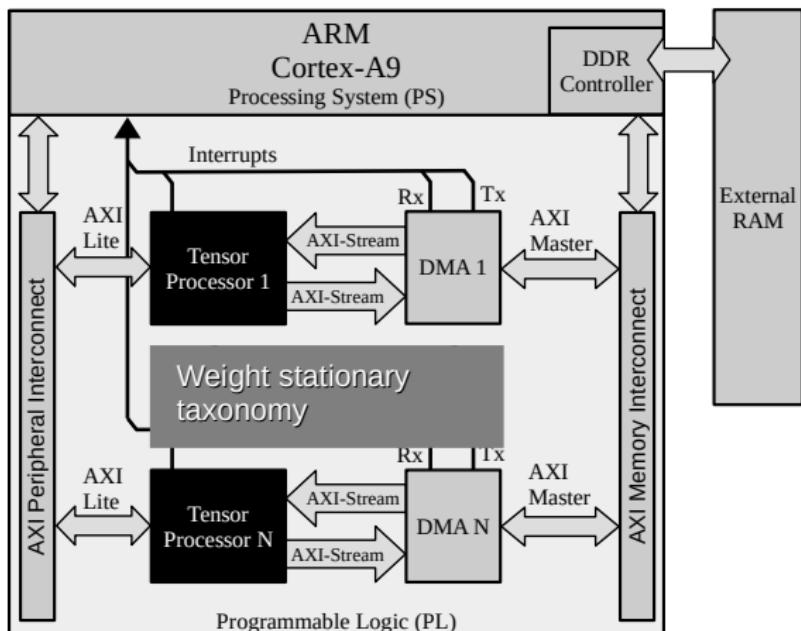
The methodology efficiently deploys and accelerates floating-point neural networks on embedded systems, optimizing performance, energy consumption, and hardware utilization.

### 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

The methodology efficiently deploys and accelerates floating-point neural networks on embedded systems, optimizing performance, energy consumption, and hardware utilization.

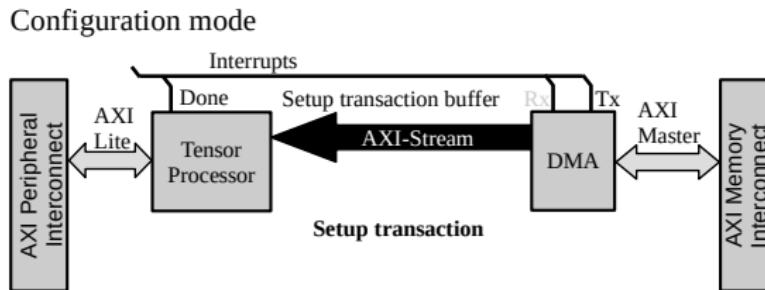
### Abstraction levels:

1. Model deployment

2. System infrastructure

### 3. Streamlined acceleration

4. Optimized processing



# Methodology

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

The methodology efficiently deploys and accelerates floating-point neural networks on embedded systems, optimizing performance, energy consumption, and hardware utilization.

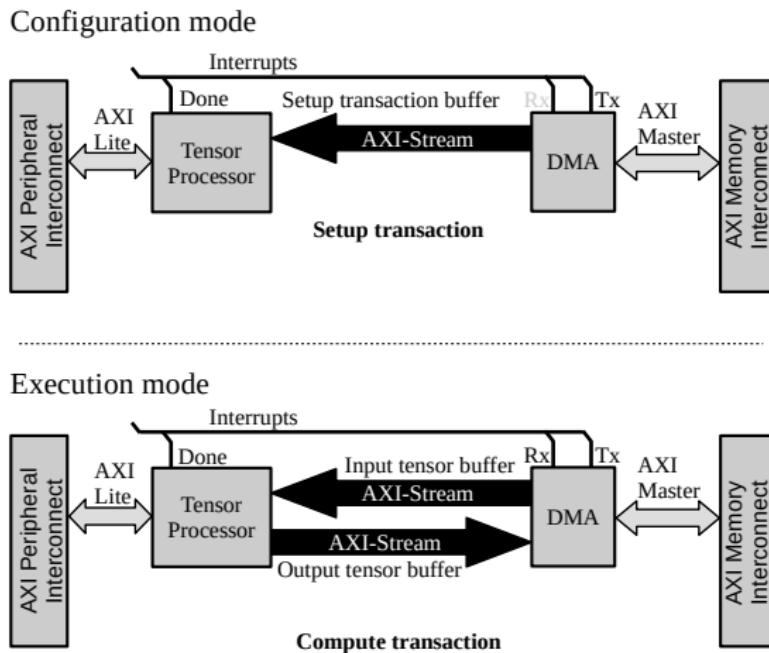
### Abstraction levels:

1. Model deployment

2. System infrastructure

### 3. Streamlined acceleration

4. Optimized processing



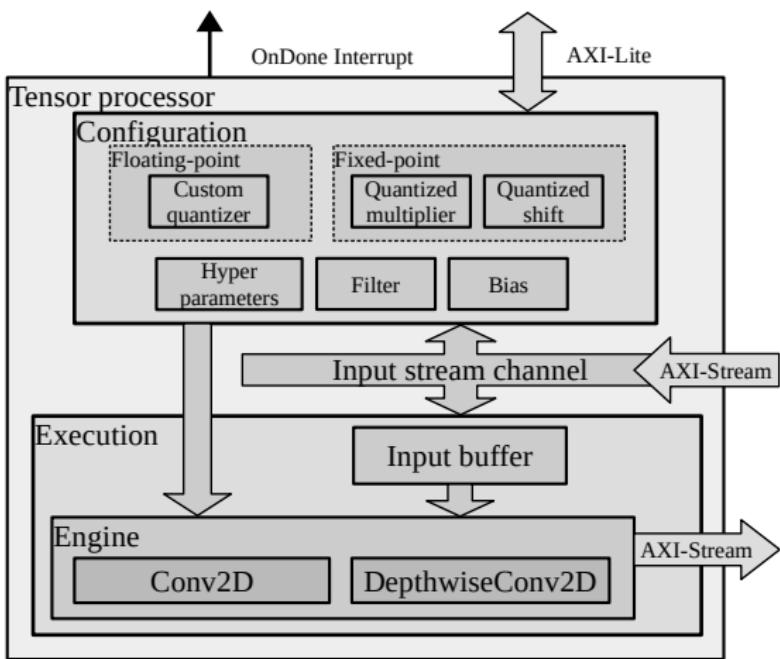
# Methodology

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

The methodology efficiently deploys and accelerates floating-point neural networks on embedded systems, optimizing performance, energy consumption, and hardware utilization.

### 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

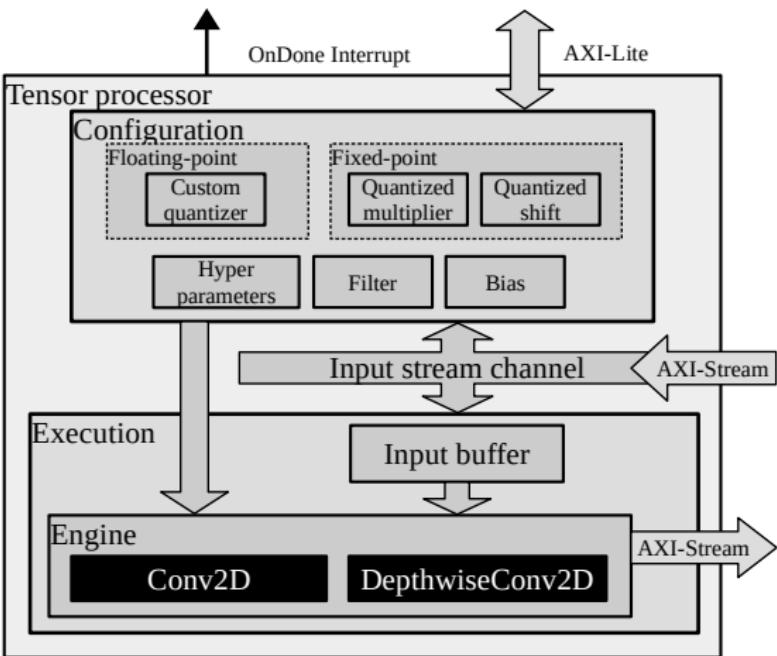
The methodology efficiently deploys and accelerates floating-point neural networks on embedded systems, optimizing performance, energy consumption, and hardware utilization.

### Abstraction levels:

1. Model deployment
2. System infrastructure

**3. Streamlined acceleration**

4. Optimized processing



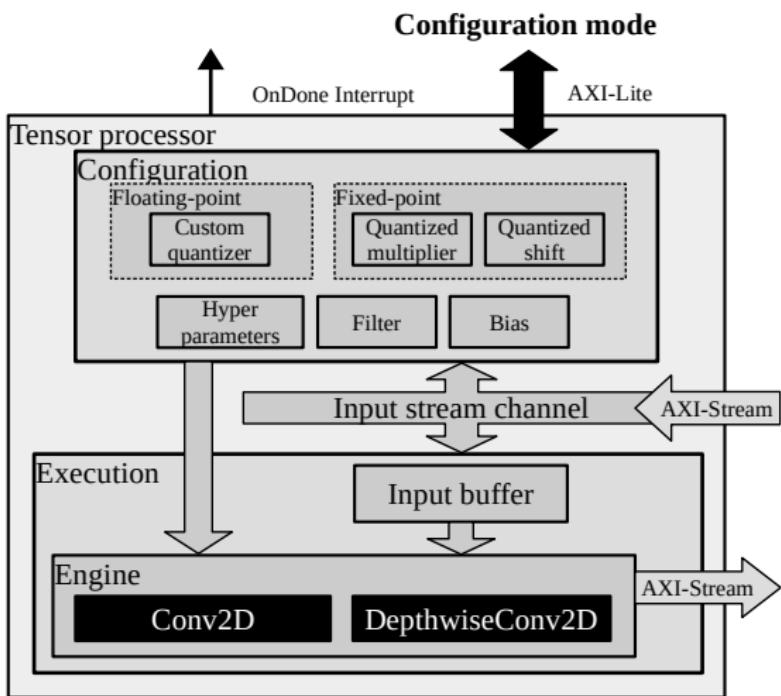
# Methodology

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

The methodology efficiently deploys and accelerates floating-point neural networks on embedded systems, optimizing performance, energy consumption, and hardware utilization.

### 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

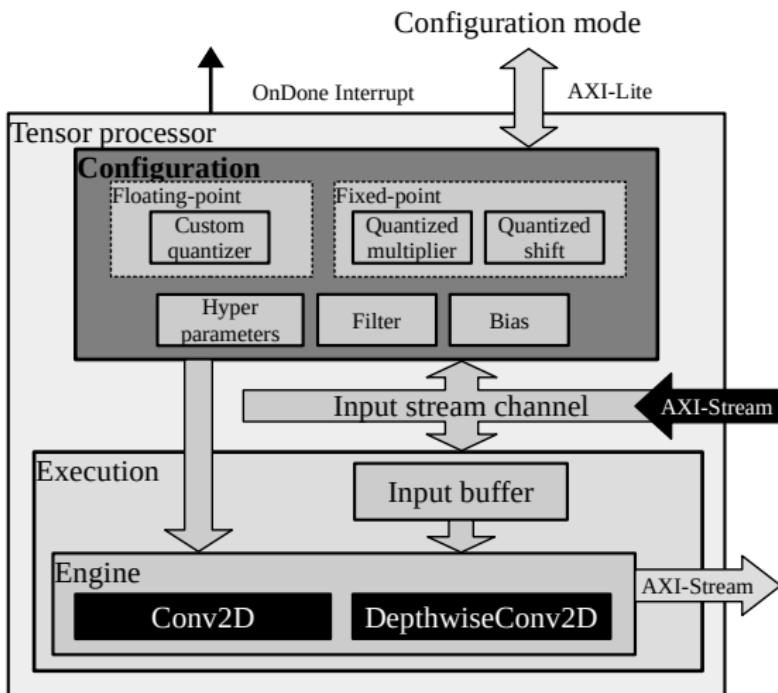
The methodology efficiently deploys and accelerates floating-point neural networks on embedded systems, optimizing performance, energy consumption, and hardware utilization.

### 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

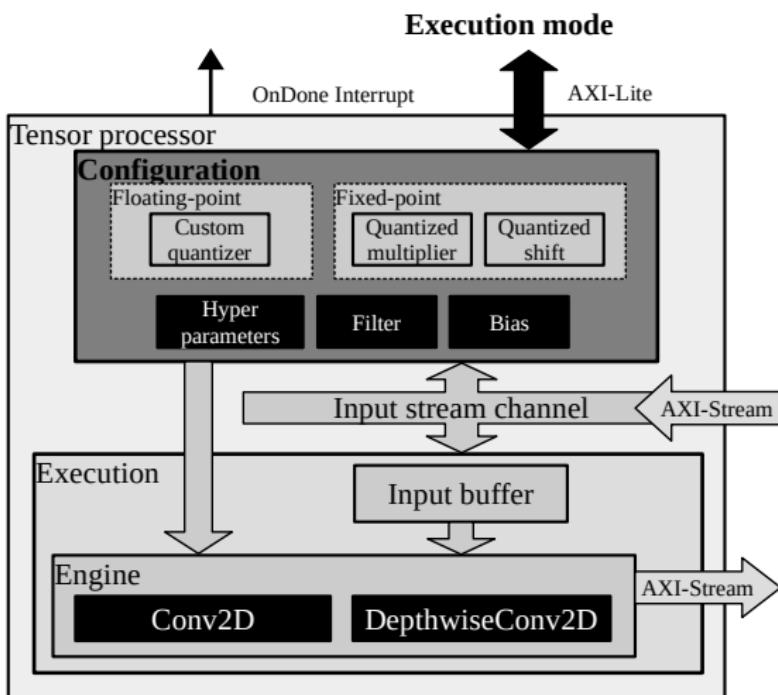
The methodology efficiently deploys and accelerates floating-point neural networks on embedded systems, optimizing performance, energy consumption, and hardware utilization.

### Abstraction levels:

1. Model deployment
2. System infrastructure

**3. Streamlined acceleration**

4. Optimized processing



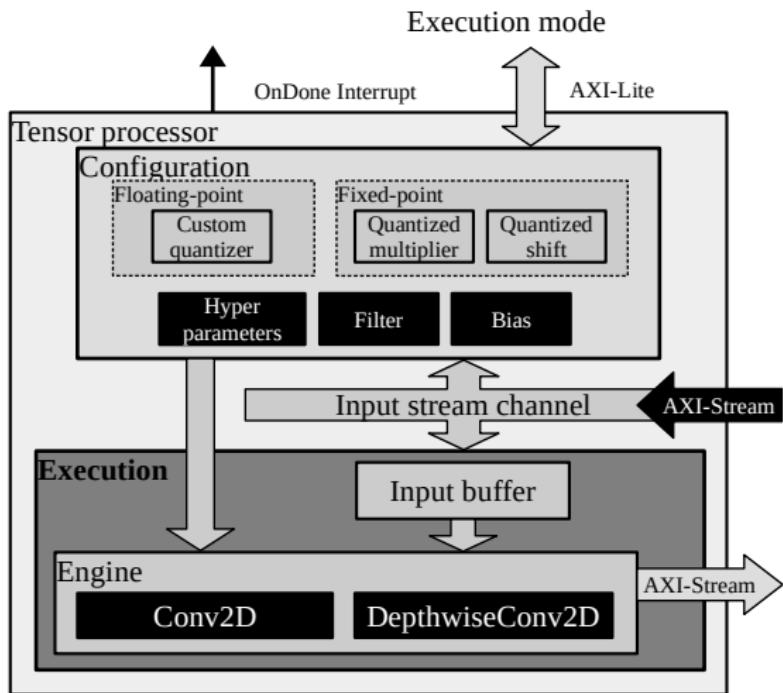
# Methodology

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

The methodology efficiently deploys and accelerates floating-point neural networks on embedded systems, optimizing performance, energy consumption, and hardware utilization.

### Abstraction levels:

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



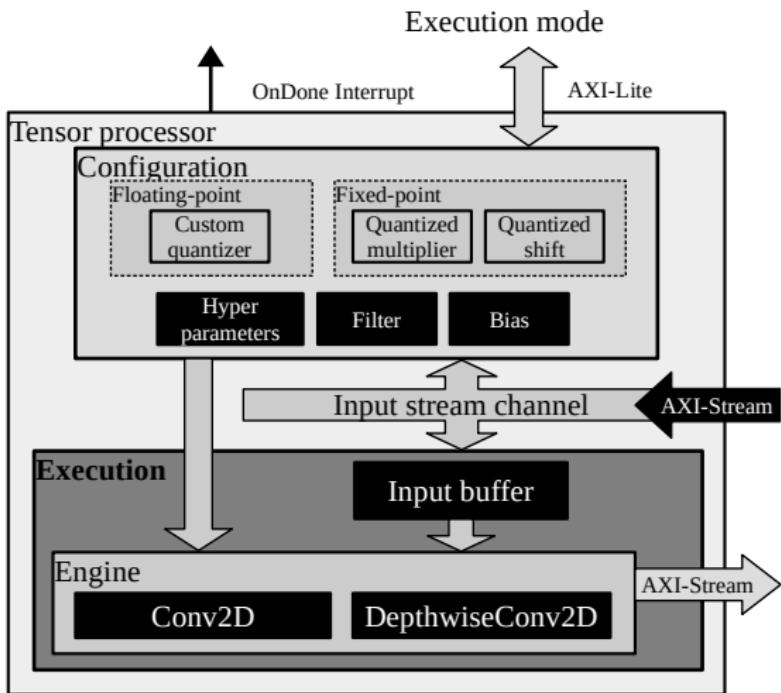
# Methodology

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

The methodology efficiently deploys and accelerates floating-point neural networks on embedded systems, optimizing performance, energy consumption, and hardware utilization.

### 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

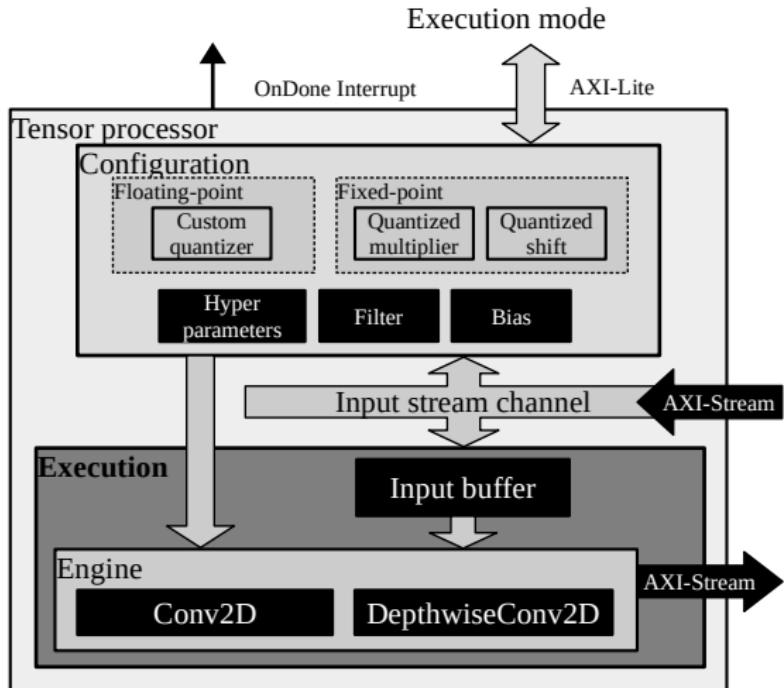
The methodology efficiently deploys and accelerates floating-point neural networks on embedded systems, optimizing performance, energy consumption, and hardware utilization.

### Abstraction levels:

1. Model deployment
2. System infrastructure

**3. Streamlined acceleration**

4. Optimized processing



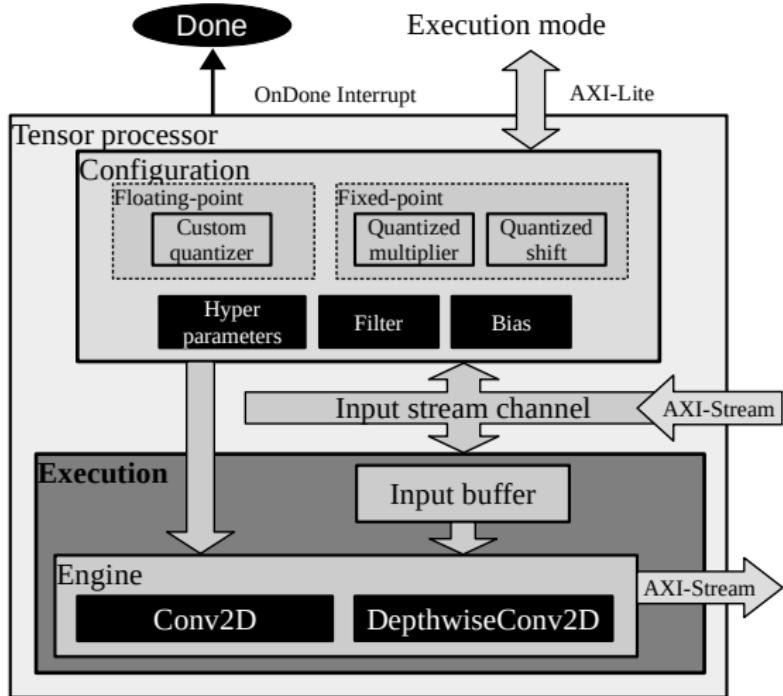
# Methodology

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

The methodology efficiently deploys and accelerates floating-point neural networks on embedded systems, optimizing performance, energy consumption, and hardware utilization.

### Abstraction levels:

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



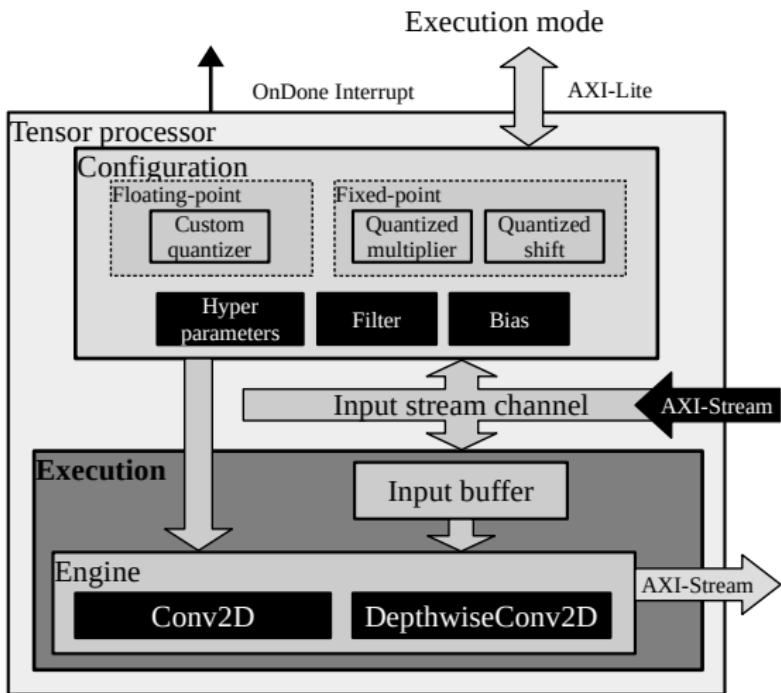
# Methodology

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

The methodology efficiently deploys and accelerates floating-point neural networks on embedded systems, optimizing performance, energy consumption, and hardware utilization.

### Abstraction levels:

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



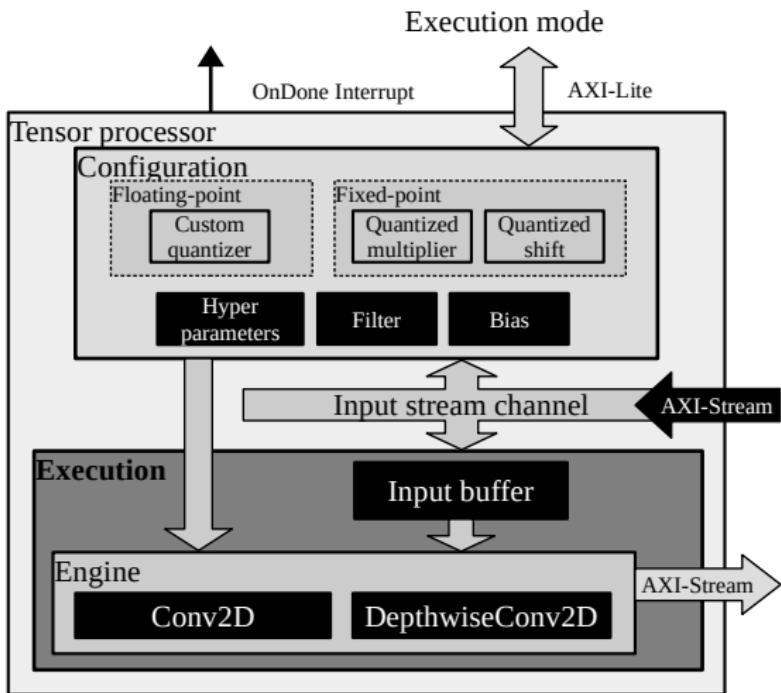
# Methodology

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

The methodology efficiently deploys and accelerates floating-point neural networks on embedded systems, optimizing performance, energy consumption, and hardware utilization.

### Abstraction levels:

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



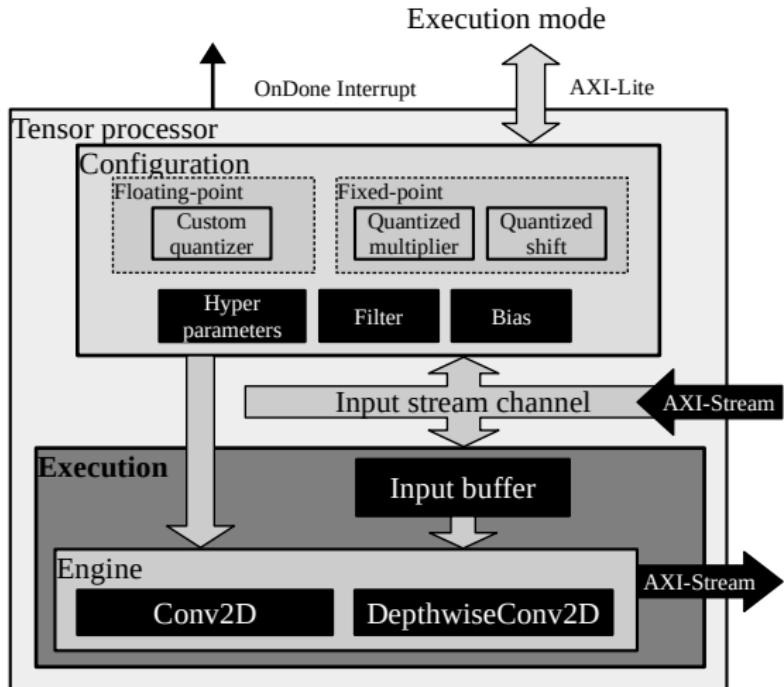
# Methodology

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

The methodology efficiently deploys and accelerates floating-point neural networks on embedded systems, optimizing performance, energy consumption, and hardware utilization.

### Abstraction levels:

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



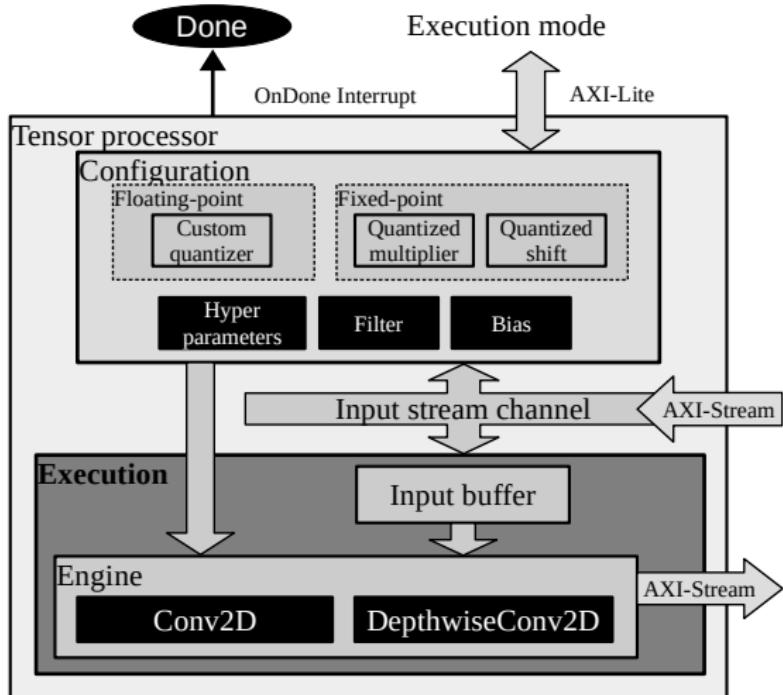
# Methodology

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

The methodology efficiently deploys and accelerates floating-point neural networks on embedded systems, optimizing performance, energy consumption, and hardware utilization.

### Abstraction levels:

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



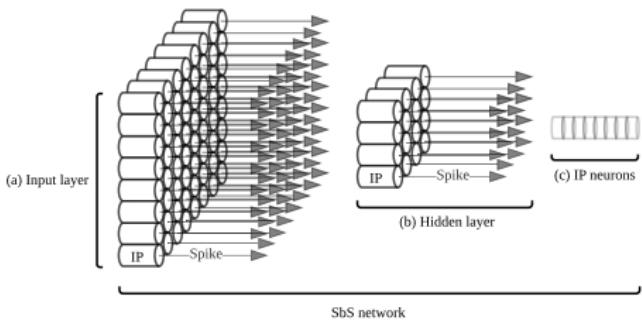
1 Methodology

2 Hybrid 8-bit Floating-Point and 4-bit Logarithmic Computation

3 Hybrid 6-bit Floating-Point Computation

4 Conclusions and Future Research

# Spike-by-Spike Neural Network



**Figure:** SbS inference

# Spike-by-Spike Neural Network

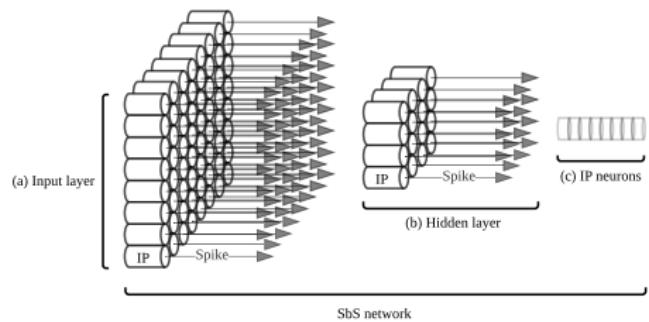


Figure: SbS inference

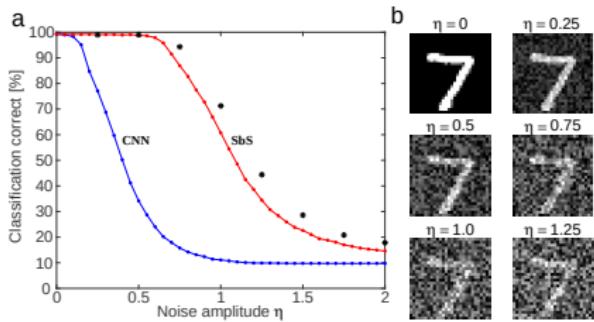


Figure: Robustness of SbS NN versus CNN

# Spike-by-Spike Neural Network

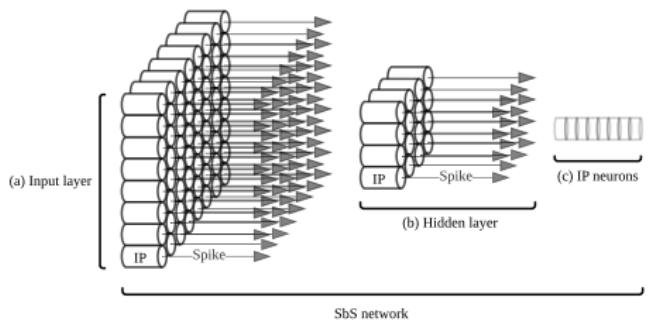


Figure: SbS inference

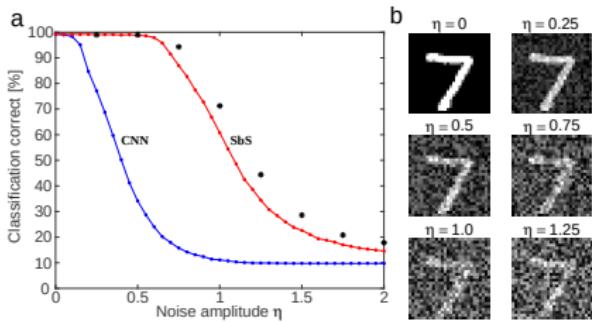
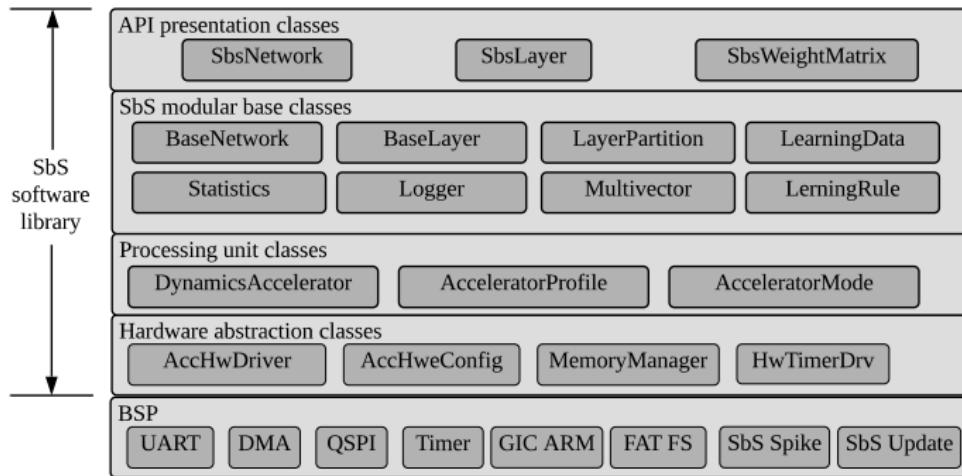


Figure: Robustness of SbS NN versus CNN

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

# HW/SW Co-Design and Deployment Framework



**Figure:** System-level overview of the embedded software architecture

1 Methodology

2 Hybrid 8-bit Floating-Point and 4-bit Logarithmic Computation

3 Hybrid 6-bit Floating-Point Computation

4 Conclusions and Future Research

# Convolution Operation

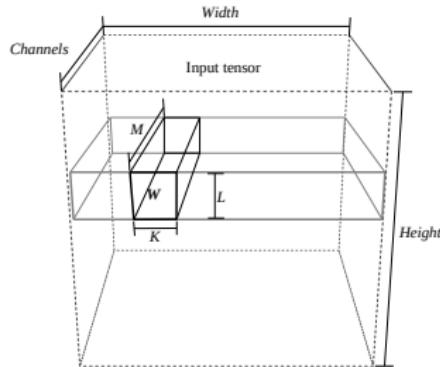
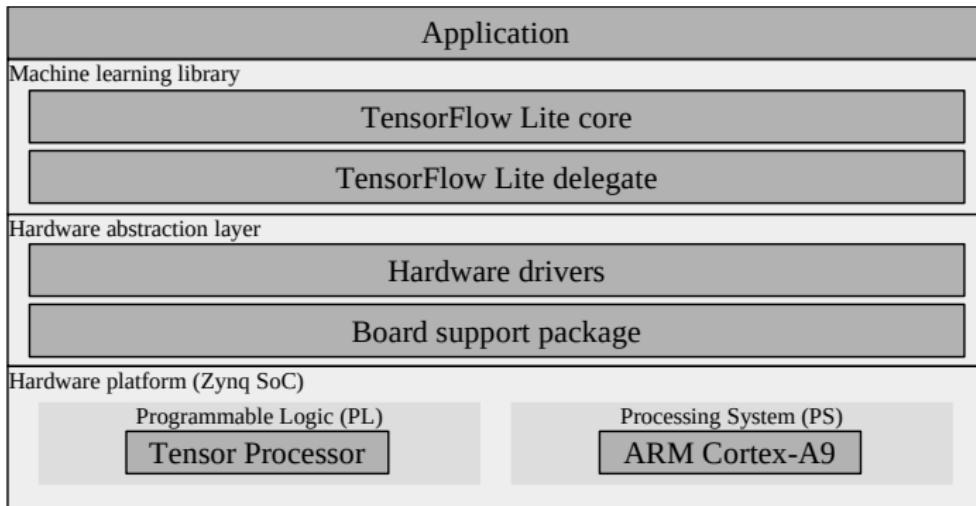


Figure: 2D convolution operation

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

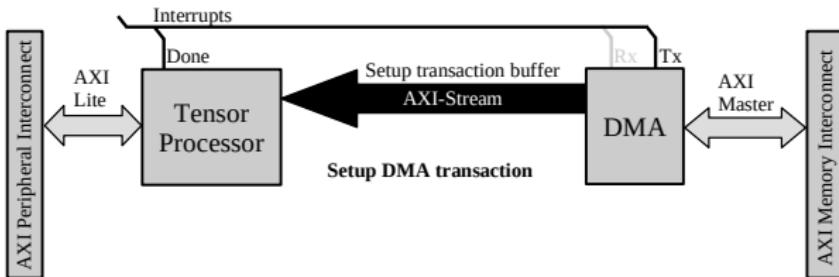
# HW/SW Co-Design and Deployment Framework



**Figure:** High level embedded software architecture

# Tensor Processor Operations and Data Flow Overview

## a) Configuration mode



## a) Execution mode

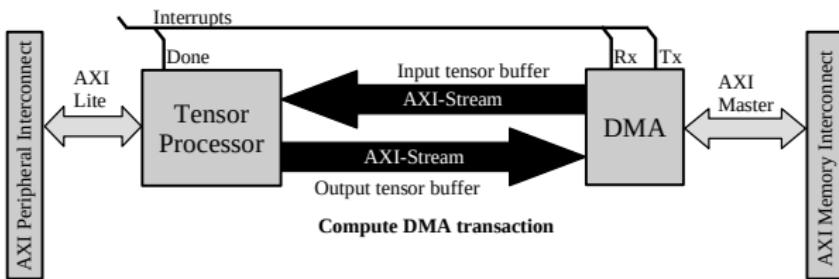


Figure: Tensor processor task execution

# On-Chip Memory

$$TP_M = TP_B + V_M \quad (1)$$

$$TP_B = Input_M + Filter_M + Bias_M \quad (2)$$

$$Input_M = K_H W_I C_I BitSize_I \quad (3)$$

$$Filter_M = C_I K_W K_H C_O BitSize_F \quad (4)$$

$$Bias_M = C_O BitSize_B \quad (5)$$

$$C_O = \frac{TP_M - V_M - K_H W_I C_I BitSize_I}{C_I K_W K_H BitSize_F + BitSize_B} \quad (6)$$

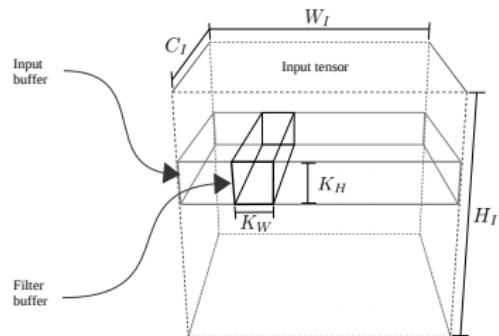


Figure: On-chip memory buffers

# Dot-Product with Hybrid Floating-Point 6-bit

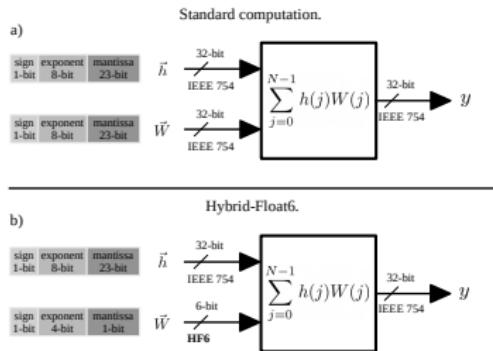


Figure: Dot-product

# Dot-Product with Hybrid Floating-Point 6-bit

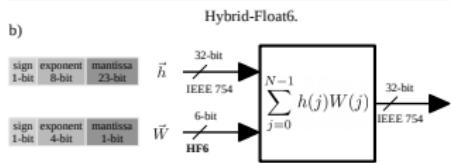
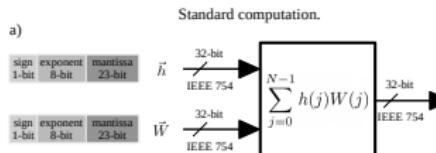


Figure: Dot-product

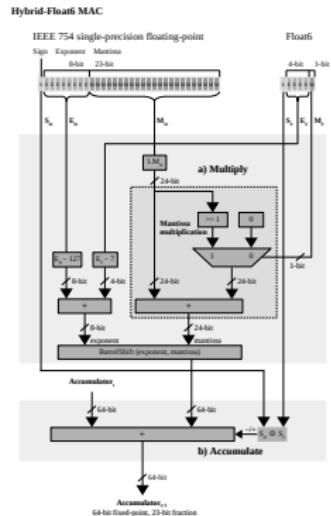


Figure: Multiply-accumulate unit

$$L_{hf} = N + 7$$

# Custom Floating-Point Quantization

---

**Algorithm 1:** Custom floating-point quantization

---

```
Input: MODEL as the CNN
Input:  $E_{size}$  as the target exponent bit size
Input:  $M_{size}$  as the target mantissa bits size
Input:  $STD M_{size}$  as the IEEE 754 mantissa bit size
Output: MODEL as the quantized CNN

1 foreach layer in MODEL do
2   if layer is Conv2D or SeparableConv2D then
3     filter, bias ← GetWeights(layer)
4     foreach x in filter and bias do
5       sign ← GetSign(x)
6       exp ← GetExponent(x)
7       fullexp ←  $2^{E_{size}-1} - 1$  // Get full range value
8       cman ← GetCustomMantissa(x,  $M_{size}$ )
9       leftman ← GetLeftoverMantissa(x,  $M_{size}$ )
10      if exp < fullexp then
11        | x ← 0
12      else
13        | if exp > fullexp then
14          | | x ←  $(-1)^{sign} \cdot 2^{fullexp} \cdot (1 + (1 - 2^{-M_{size}}))$ 
15        | | else
16          | | | if  $2^{STD M_{size} - M_{size} - 1} - 1 < leftman$  then
17          | | | | cman ← cman + 1 // Above halfway
18          | | | | if  $2^{M_{size}} - 1 < cman$  then
19          | | | | | cman ← 0 // Correct mantissa overflow
20          | | | | | exp ← exp + 1
21          | | | | x ←  $(-1)^{sign} \cdot 2^{exp} \cdot (1 + cman \cdot 2^{-M_{size}})$ 
22   SetWeights(layer, filter, bias)
```

---

# Case Study: Applying Sensor Analytics to Structural Health Monitoring

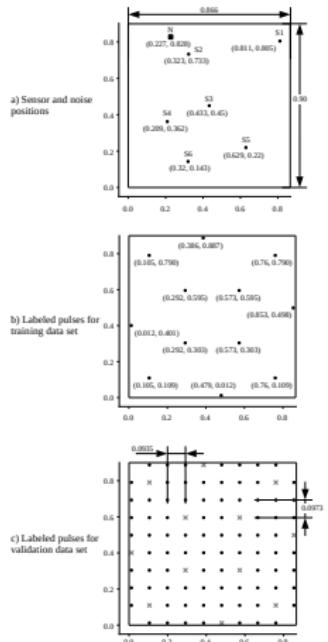


Figure: Structural health monitoring, all lengths are in meters (m)

# Case Study: Applying Sensor Analytics to Structural Health Monitoring

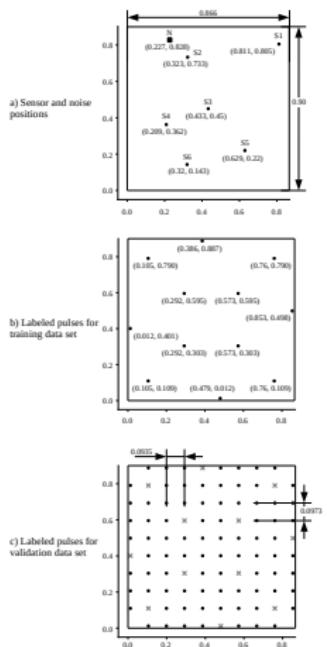


Figure: Structural health monitoring, all lengths are in meters (m)

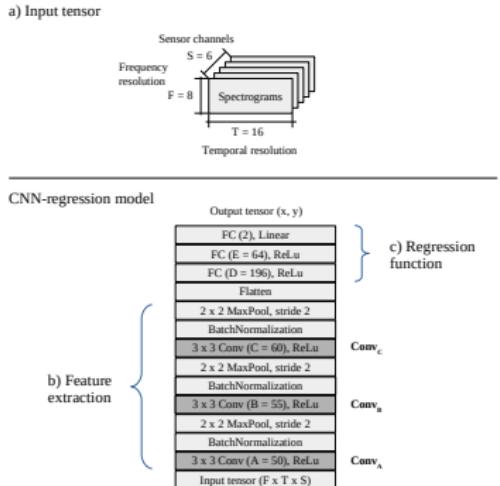


Figure: CNN-regression model for sensor analytics

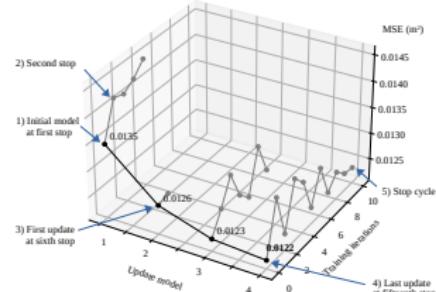
# Quantized Aware Training with Custom Floating-Point

**Algorithm 2:** Training with iterative early stop cycle.

```
Input: MODEL as the input model
Input:  $D_{train}$  as the training data set
Input:  $D_{val}$  as the validation data set
Input:  $N_f$  as the stop patience for iterative training cycle
Input:  $N_E$  as the early stop patience (epochs) for training
Input:  $B_{size}$  as the mini-batch size
Output: MODEL as the full-precision output model

// Initial training and evaluation
1 Train(MODEL,  $D_{train}, D_{val}, N_E, B_{size}$ )
2  $mse_i \leftarrow Evaluate(MODEL, D_{val})$ 
3  $n_f \leftarrow 0$ 
4 while  $n_f < N_f$  do
    // Iterative early stop cycle
    5 Train(MODEL,  $D_{train}, D_{val}, N_E, B_{size}$ )
    6  $mse_v \leftarrow Evaluate(MODEL, D_{val})$ 
    7 if  $mse_v < mse_i$  then
        8     | Update(MODEL)
        9     |  $mse_i \leftarrow mse_v$ 
    10    end
    11    else
    12        | MODEL  $\leftarrow LoadPreviousWeights()$ 
    13        |  $n_f \leftarrow n_f + 1$ 
    14    end
15 end
```

Training with iterative early stop



# Quantized Aware Training with Custom Floating-Point

**Algorithm 2:** Training with iterative early stop cycle.

```
Input: MODEL as the input model
Input:  $D_{train}$  as the training data set
Input:  $D_{val}$  as the validation data set
Input:  $N_t$  as the stop patience for iterative training cycle
Input:  $N_E$  as the early stop patience (epochs) for training
Input:  $B_{size}$  as the mini-batch size
Output: MODEL as the full-precision output model

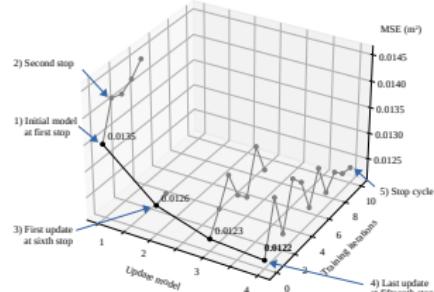
// Initial training and evaluation
1 Train(MODEL,  $D_{train}$ ,  $D_{val}$ ,  $N_E$ ,  $B_{size}$ )
2  $mse_i \leftarrow Evaluate(MODEL, D_{val})$ 
3  $n_t \leftarrow 0$ 
4 while  $n_t < N_t$  do
    // Iterative early stop cycle
    5 Train(MODEL,  $D_{train}$ ,  $D_{val}$ ,  $N_E$ ,  $B_{size}$ )
    6  $mse_e \leftarrow Evaluate(MODEL, D_{val})$ 
    7 if  $mse_e < mse_i$  then
        8     | Update(MODEL)
        9     |  $mse_i \leftarrow mse_e$ 
    10 end
    11 else
    12     | MODEL  $\leftarrow LoadPreviousWeights()$ 
    13     |  $n_t \leftarrow n_t + 1$ 
    14 end
15 end
```

**Algorithm 3:** OnMiniBatchUpdate\_Callback.

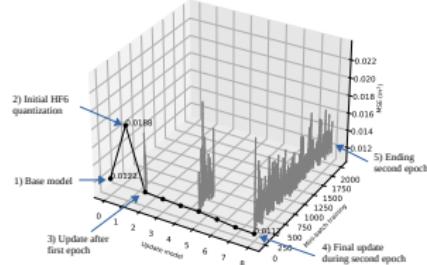
```
Input: MODEL as the full-precision input model
Input:  $E_{size}$  as the target exponent bits size
Input:  $M_{size}$  as the target mantissa bits size
Input:  $D_{train}$  as the training data set
Input:  $D_{val}$  as the validation data set
Input:  $N_{ep}$  as the number of epochs
Input:  $B_{size}$  as the mini-batch size
Output: MODEL as the quantized output model

// Quantize
1 MODEL  $\leftarrow QuantizeTraining(MODEL, E_{size}, M_{size})$ 
2 if  $1 < epoch$  then
    // Update model after first epoch
    3  $mse_v \leftarrow Evaluate(MODEL, D_{val})$ 
    4 if  $mse_v < mse_i$  then
        5     | Update(MODEL)
        6     |  $mse_i \leftarrow mse_v$ 
    7 end
8 end
```

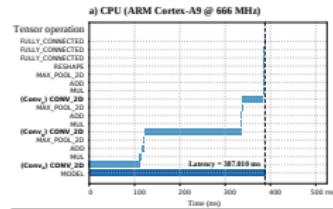
Training with iterative early stop



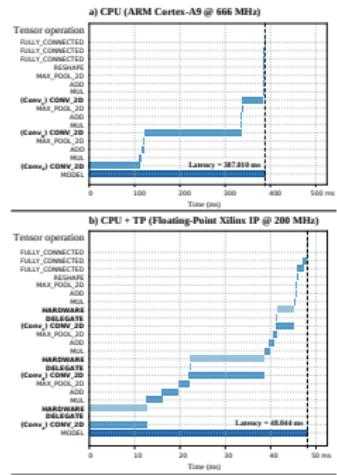
Quantization-aware training



# Evaluating Floating-Point Configurations: Speed, Power, and Hardware Utilization

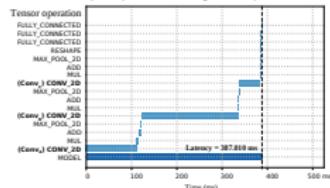


# Evaluating Floating-Point Configurations: Speed, Power, and Hardware Utilization

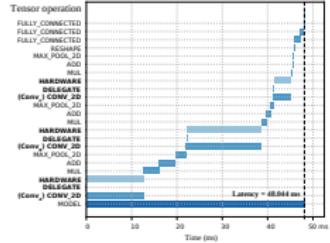


# Evaluating Floating-Point Configurations: Speed, Power, and Hardware Utilization

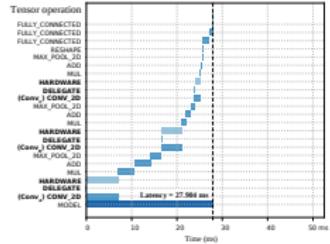
a) CPU (ARM Cortex-A9 @ 666 MHz)



b) CPU + TP (Floating-Point Xilinx IP @ 200 MHz)

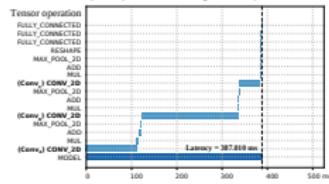


c) CPU + TP (Hybrid-Float6 @ 200 MHz)

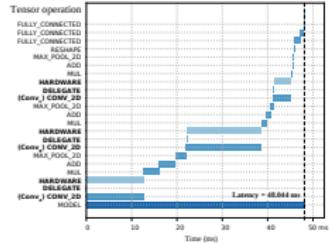


# Evaluating Floating-Point Configurations: Speed, Power, and Hardware Utilization

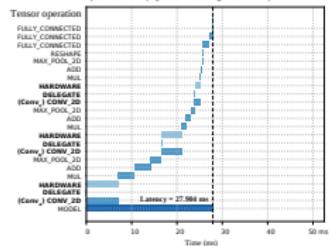
a) CPU (ARM Cortex-A9 @ 666 MHz)



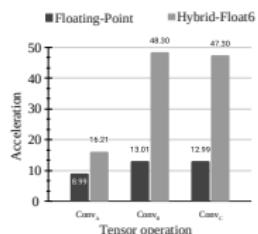
b) CPU + TP (Floating-Point Xilinx IP @ 200 MHz)



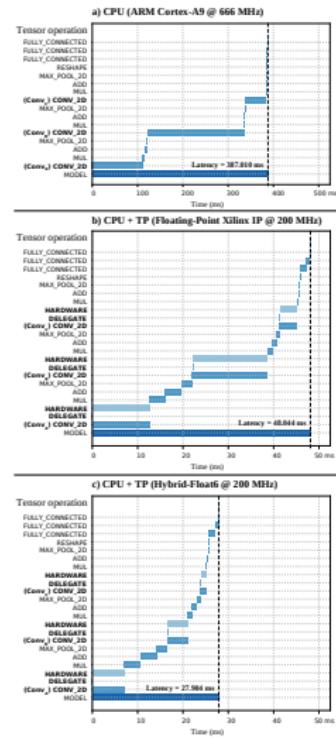
c) CPU + TP (Hybrid-Float6 @ 200 MHz)



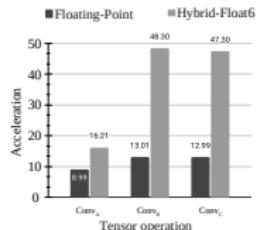
Acceleration vs. CPU



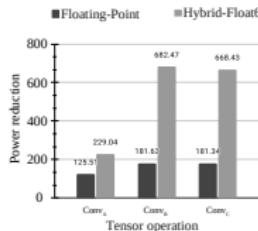
# Evaluating Floating-Point Configurations: Speed, Power, and Hardware Utilization



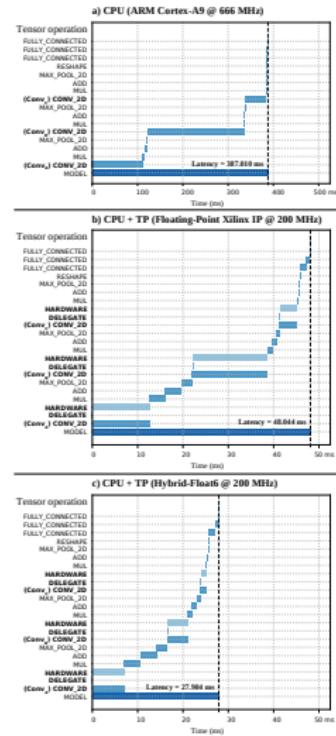
Acceleration vs. CPU



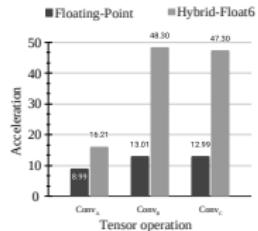
Power reduction vs. CPU



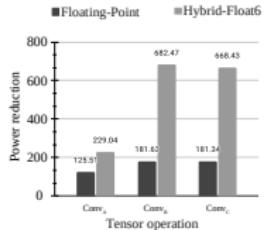
# Evaluating Floating-Point Configurations: Speed, Power, and Hardware Utilization



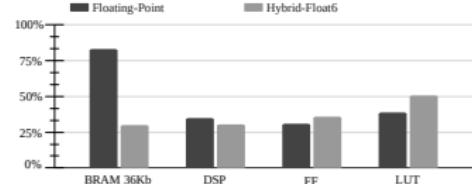
Acceleration vs. CPU



Power reduction vs. CPU

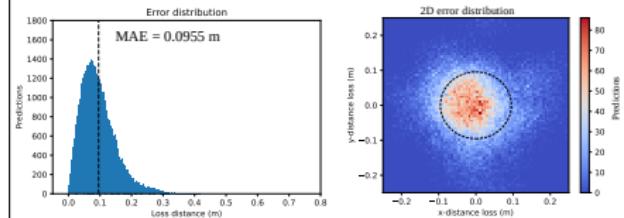


Hardware resource utilization.



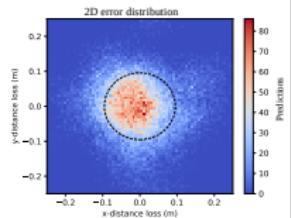
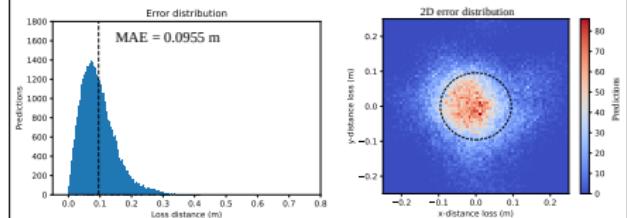
# Quantization Impact: Error Histograms in Position Prediction

a) Floating-Point 32-bit

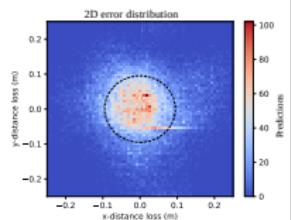
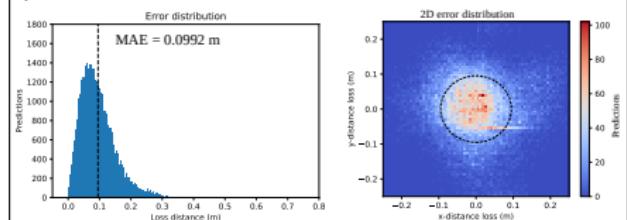


# Quantization Impact: Error Histograms in Position Prediction

a) Floating-Point 32-bit

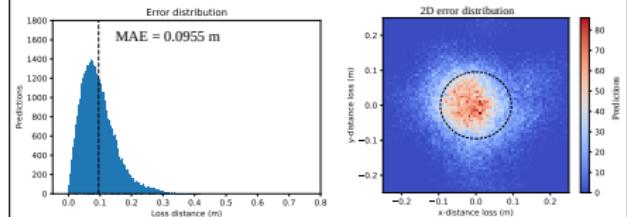


b) Fixed-Point 8-bit

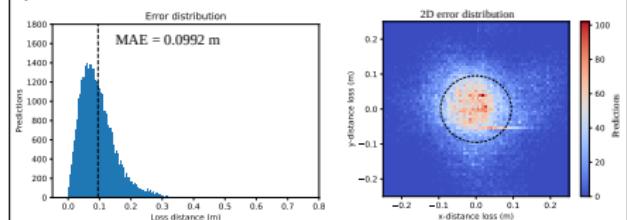


# Quantization Impact: Error Histograms in Position Prediction

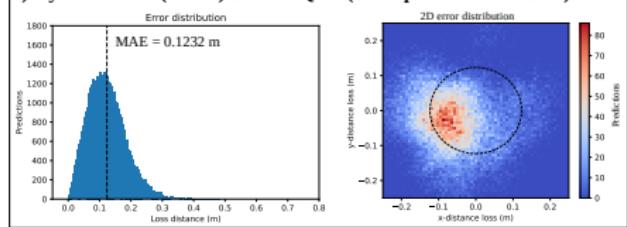
a) Floating-Point 32-bit



b) Fixed-Point 8-bit

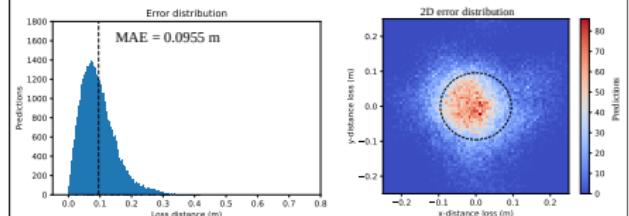


c) Hybrid-Float6 (E4M1) without QAT (trans-precision inference)

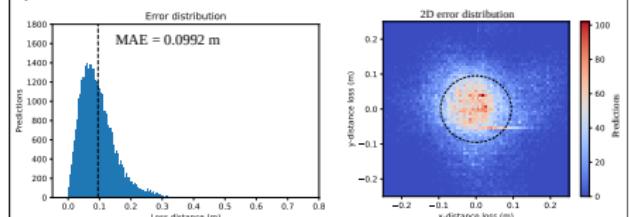


# Quantization Impact: Error Histograms in Position Prediction

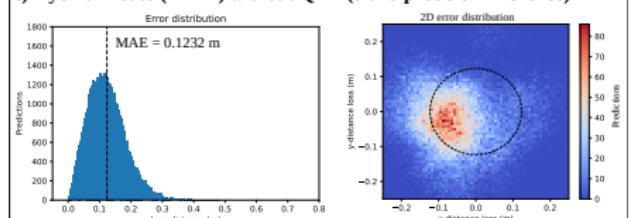
a) Floating-Point 32-bit



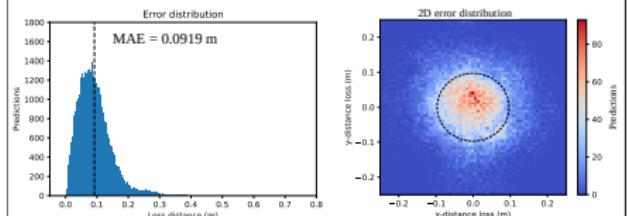
b) Fixed-Point 8-bit



c) Hybrid-Float6 (E4M1) without QAT (trans-precision inference)

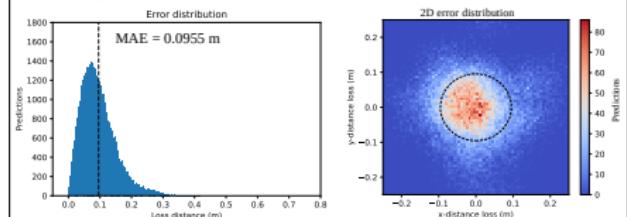


d) Hybrid-Float6 (E4M1) with QAT

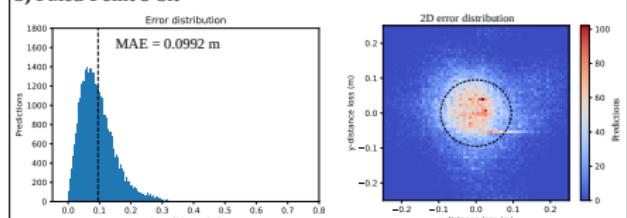


# Quantization Impact: Error Histograms in Position Prediction

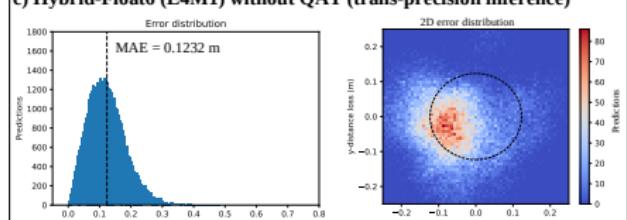
a) Floating-Point 32-bit



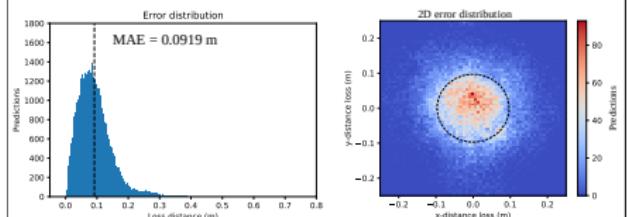
b) Fixed-Point 8-bit



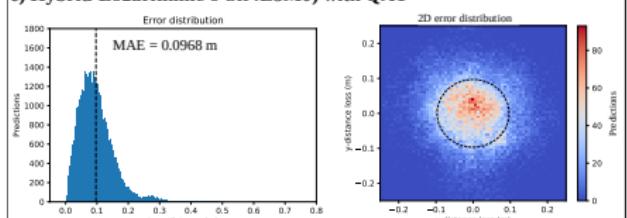
c) Hybrid-Float6 (E4M1) without QAT (trans-precision inference)



d) Hybrid-Float6 (E4M1) with QAT

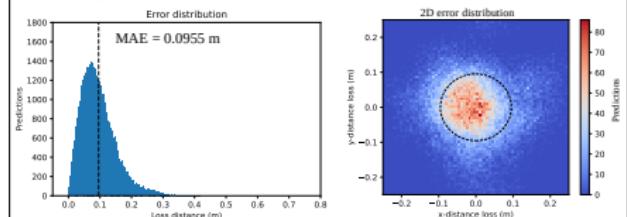


e) Hybrid-Logarithmic 6-bit (E5M0) with QAT

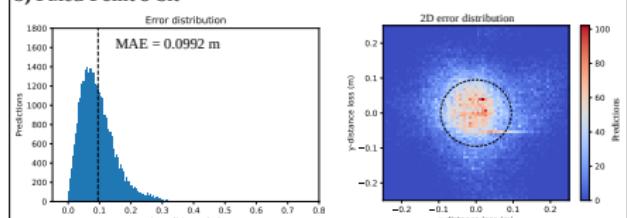


# Quantization Impact: Error Histograms in Position Prediction

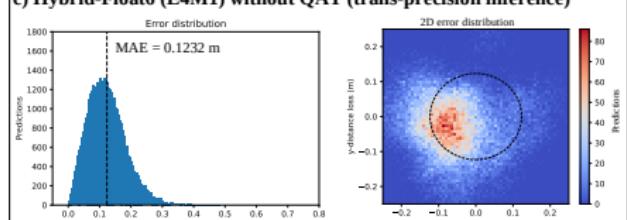
a) Floating-Point 32-bit



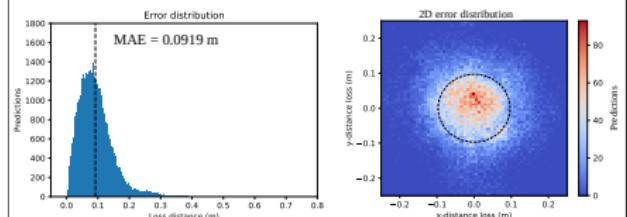
b) Fixed-Point 8-bit



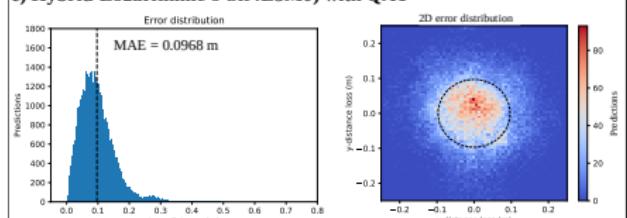
c) Hybrid-Float6 (E4M1) without QAT (trans-precision inference)



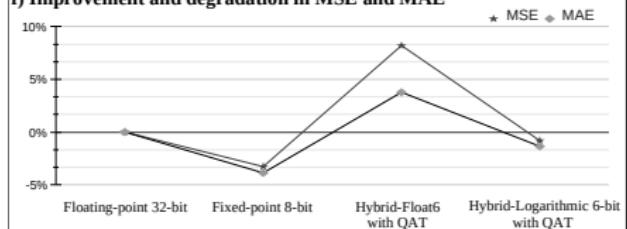
d) Hybrid-Float6 (E4M1) with QAT



e) Hybrid-Logarithmic 6-bit (E5M0) with QAT



f) Improvement and degradation in MSE and MAE



1 Methodology

2 Hybrid 8-bit Floating-Point and 4-bit Logarithmic Computation

3 Hybrid 6-bit Floating-Point Computation

4 Conclusions and Future Research

## Conclusions

- It was presented a design methodology for low-power Spike-by-Spike neural network accelerators with **hybrid 8-bit floating-point and 4-bit logarithm computation**

### Conclusions

- It was presented a design methodology for low-power Spike-by-Spike neural network accelerators with **hybrid 8-bit floating-point and 4-bit logarithm computation**
- It was presented a design methodology for low-power custom floating-point CNN accelerators showcasing the potential of **hybrid 6-bit quantization**

### Conclusions

- It was presented a design methodology for low-power Spike-by-Spike neural network accelerators with **hybrid 8-bit floating-point and 4-bit logarithm computation**
- It was presented a design methodology for low-power custom floating-point CNN accelerators showcasing the potential of **hybrid 6-bit quantization**

# Conclusions and Future Research

## Conclusions

- It was presented a design methodology for low-power Spike-by-Spike neural network accelerators with **hybrid 8-bit floating-point and 4-bit logarithm computation**
- It was presented a design methodology for low-power custom floating-point CNN accelerators showcasing the potential of **hybrid 6-bit quantization**

## Future Research

- Low-power neural network accelerators for **on-device training** with hybrid custom floating-point computation

# 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

## Limitations of State-of-the-Art

- **Spike-by-Spike Neural Networks:**

## Limitations of State-of-the-Art

- **Spike-by-Spike Neural Networks:**
  - **Quantization Challenges:** Struggles with reduced number representations (8-bit and 4-bit)

## Limitations of State-of-the-Art

- **Spike-by-Spike Neural Networks:**

- **Quantization Challenges:** Struggles with reduced number representations (8-bit and 4-bit)
- **Deployment Frameworks:** Lack of deployment strategies for IoT and TinyML applications

# Limitations of State-of-the-Art

- **Spike-by-Spike Neural Networks:**

- **Quantization Challenges:** Struggles with reduced number representations (8-bit and 4-bit)
- **Deployment Frameworks:** Lack of deployment strategies for IoT and TinyML applications
- **Design Methodologies:** Lack of effective low-power accelerator design methodologies

## Limitations of State-of-the-Art

- **Spike-by-Spike Neural Networks:**
  - **Quantization Challenges:** Struggles with reduced number representations (8-bit and 4-bit)
  - **Deployment Frameworks:** Lack of deployment strategies for IoT and TinyML applications
  - **Design Methodologies:** Lack of effective low-power accelerator design methodologies
- **Low-Power CNN Accelerators:**

# Limitations of State-of-the-Art

- **Spike-by-Spike Neural Networks:**
  - **Quantization Challenges:** Struggles with reduced number representations (8-bit and 4-bit)
  - **Deployment Frameworks:** Lack of deployment strategies for IoT and TinyML applications
  - **Design Methodologies:** Lack of effective low-power accelerator design methodologies
- **Low-Power CNN Accelerators:**
  - **Floating-Point Design:** Literature gap in floating-point methodologies for IoT and TinyML applications

# Limitations of State-of-the-Art

- **Spike-by-Spike Neural Networks:**

- **Quantization Challenges:** Struggles with reduced number representations (8-bit and 4-bit)
- **Deployment Frameworks:** Lack of deployment strategies for IoT and TinyML applications
- **Design Methodologies:** Lack of effective low-power accelerator design methodologies

- **Low-Power CNN Accelerators:**

- **Floating-Point Design:** Literature gap in floating-point methodologies for IoT and TinyML applications
- **Reproducibility:** Deficiency in reproducible research, limiting validation and adoption

# Limitations of State-of-the-Art

- **Spike-by-Spike Neural Networks:**
  - **Quantization Challenges:** Struggles with reduced number representations (8-bit and 4-bit)
  - **Deployment Frameworks:** Lack of deployment strategies for IoT and TinyML applications
  - **Design Methodologies:** Lack of effective low-power accelerator design methodologies
- **Low-Power CNN Accelerators:**
  - **Floating-Point Design:** Literature gap in floating-point methodologies for IoT and TinyML applications
  - **Reproducibility:** Deficiency in reproducible research, limiting validation and adoption
- **Low-Power Accelerators with Aggressive Quantization:**

# Limitations of State-of-the-Art

- **Spike-by-Spike Neural Networks:**
  - **Quantization Challenges:** Struggles with reduced number representations (8-bit and 4-bit)
  - **Deployment Frameworks:** Lack of deployment strategies for IoT and TinyML applications
  - **Design Methodologies:** Lack of effective low-power accelerator design methodologies
- **Low-Power CNN Accelerators:**
  - **Floating-Point Design:** Literature gap in floating-point methodologies for IoT and TinyML applications
  - **Reproducibility:** Deficiency in reproducible research, limiting validation and adoption
- **Low-Power Accelerators with Aggressive Quantization:**
  - **Computational Efficiency vs. Accuracy:** Enhances efficiency but often at the expense of significant accuracy degradation in complex tasks

# Limitations of State-of-the-Art

- **Spike-by-Spike Neural Networks:**
  - **Quantization Challenges:** Struggles with reduced number representations (8-bit and 4-bit)
  - **Deployment Frameworks:** Lack of deployment strategies for IoT and TinyML applications
  - **Design Methodologies:** Lack of effective low-power accelerator design methodologies
- **Low-Power CNN Accelerators:**
  - **Floating-Point Design:** Literature gap in floating-point methodologies for IoT and TinyML applications
  - **Reproducibility:** Deficiency in reproducible research, limiting validation and adoption
- **Low-Power Accelerators with Aggressive Quantization:**
  - **Computational Efficiency vs. Accuracy:** Enhances efficiency but often at the expense of significant accuracy degradation in complex tasks
  - **Mission-Critical Applications:** Often unsuitable for applications requiring high reliability, safety, and quality-of-result

# Limitations of State-of-the-Art

- **Spike-by-Spike Neural Networks:**
  - **Quantization Challenges:** Struggles with reduced number representations (8-bit and 4-bit)
  - **Deployment Frameworks:** Lack of deployment strategies for IoT and TinyML applications
  - **Design Methodologies:** Lack of effective low-power accelerator design methodologies
- **Low-Power CNN Accelerators:**
  - **Floating-Point Design:** Literature gap in floating-point methodologies for IoT and TinyML applications
  - **Reproducibility:** Deficiency in reproducible research, limiting validation and adoption
- **Low-Power Accelerators with Aggressive Quantization:**
  - **Computational Efficiency vs. Accuracy:** Enhances efficiency but often at the expense of significant accuracy degradation in complex tasks
  - **Mission-Critical Applications:** Often unsuitable for applications requiring high reliability, safety, and quality-of-result
  - **Compatibility and Portability:** Faces challenges in compatibility and portability across different computing platforms and ML frameworks

# Spike-by-Spike Neural Network

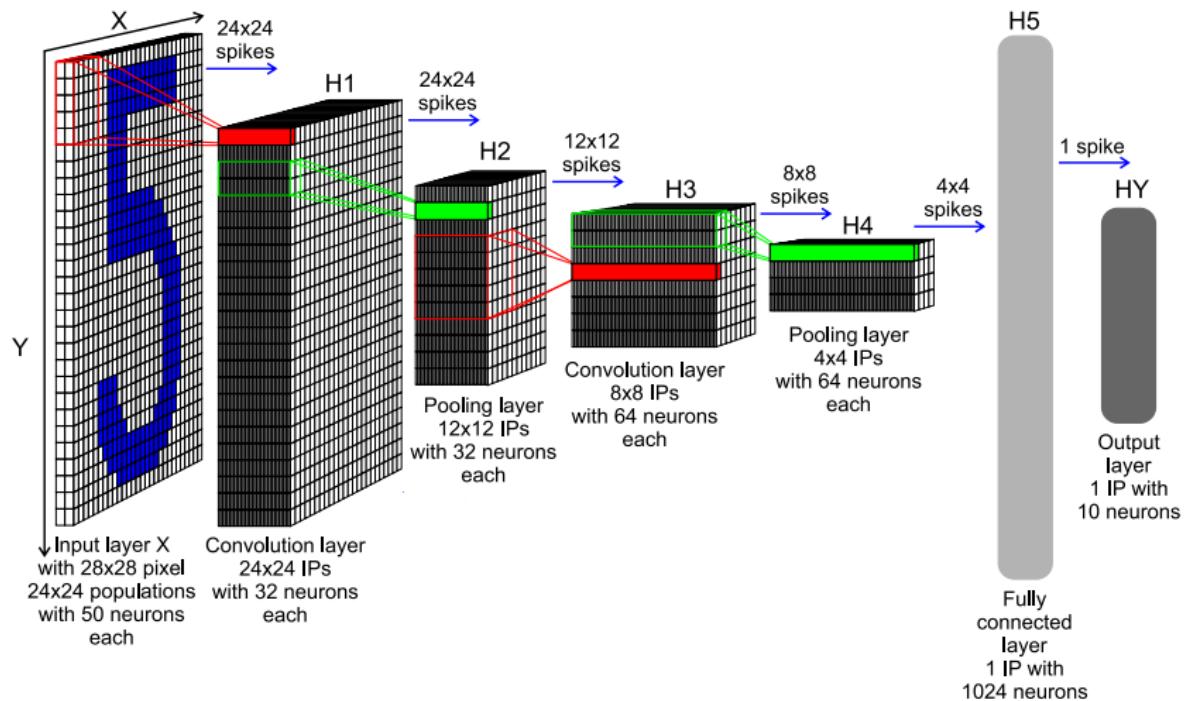


Figure: Spike-by-Spike (SbS) neural network architecture for handwritten digit classification task

# SbS Processing Unit

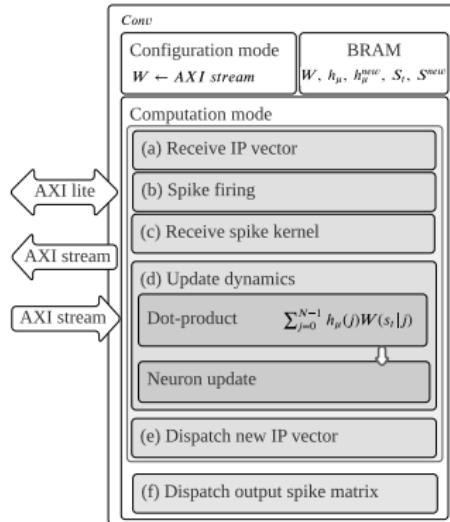


Figure: Convolution processing unit

# SbS Processing Unit

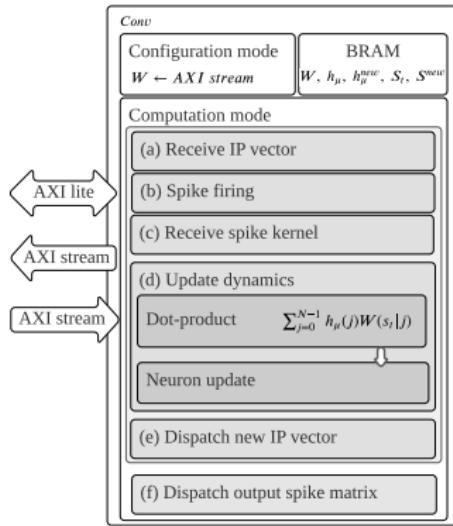


Figure: Convolution processing unit

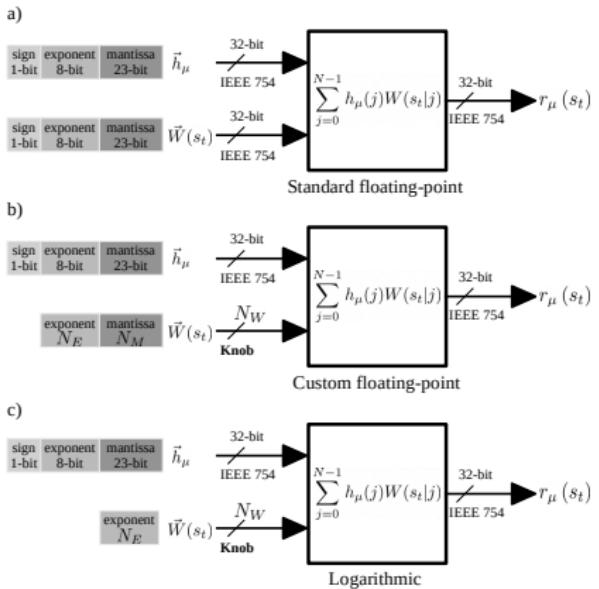


Figure: Dot-product hardware module

# SbS HW/SW Co-Design and Deployment Framework

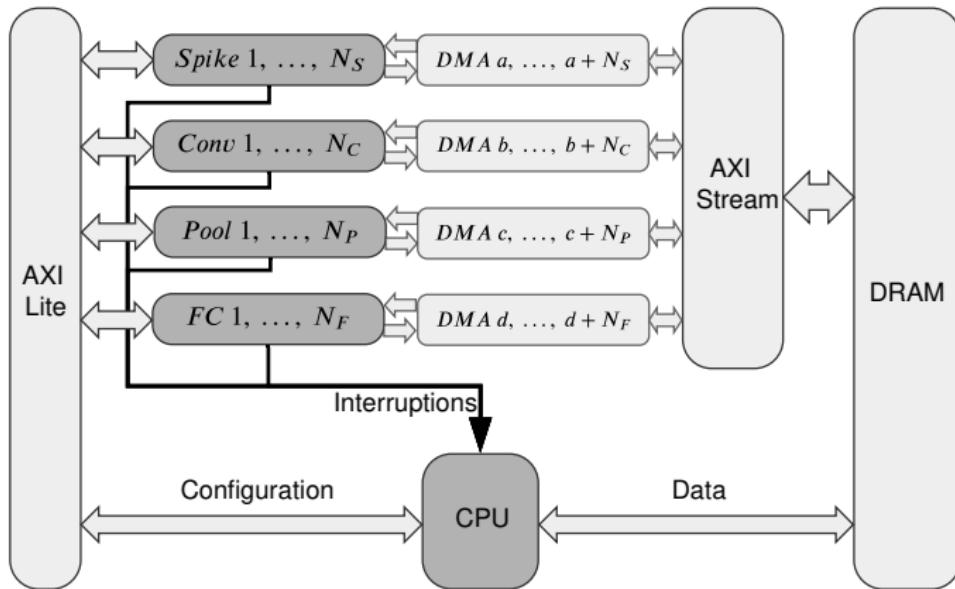


Figure: System-level hardware architecture with scalable number of heterogeneous processing units

# Hybrid Dot-Product Approximation

$$r_\mu(s_t) = \sum_{j=0}^{N-1} h_\mu(j) W(s_t|j) \quad (7)$$

$$E_{\min} = \log_2(\min_{\forall i}(W(i))) \quad (8)$$

$$N_E = \lceil \log_2(|E_{\min}|) \rceil \quad (9)$$

$$N_W = N_E + N_M \quad (10)$$

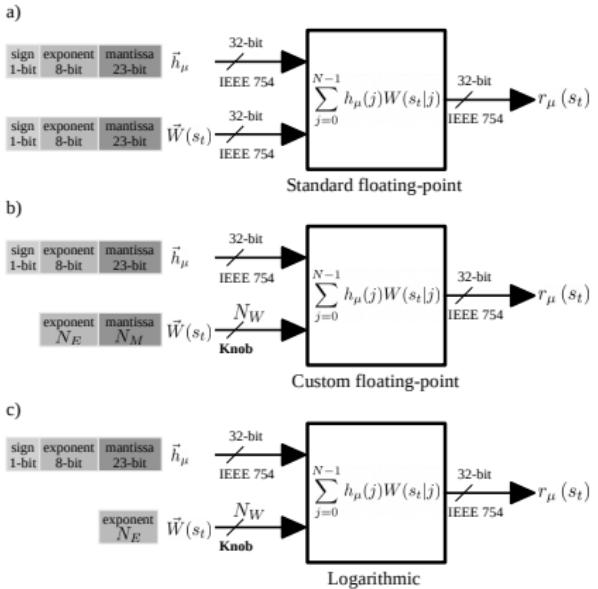


Figure: Dot-product hardware module

# Dot-Product with Standard Floating-Point (IEEE 754)

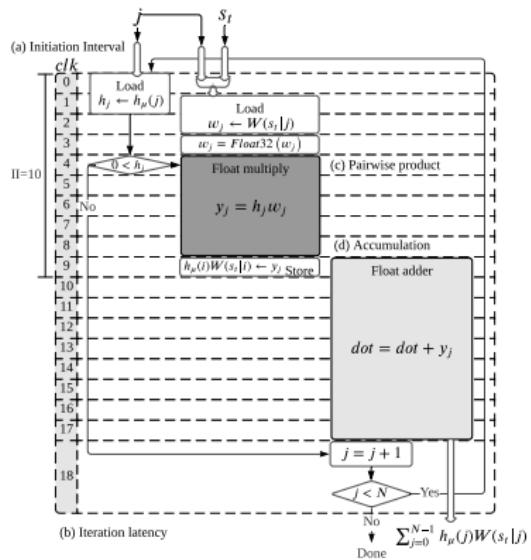
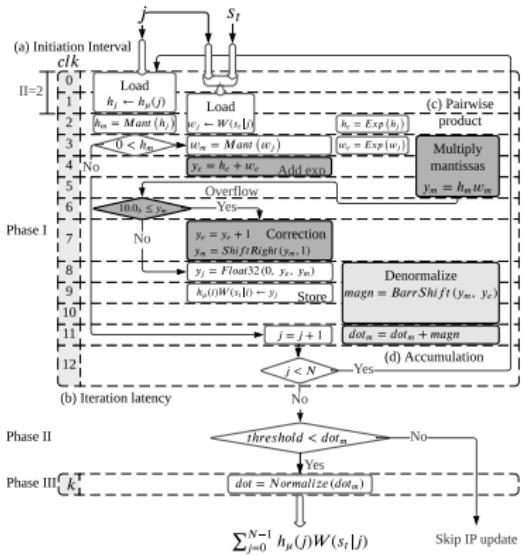


Figure: Dot-product hardware module with standard floating-point computation

$$L_{f32} = 10N + 9$$

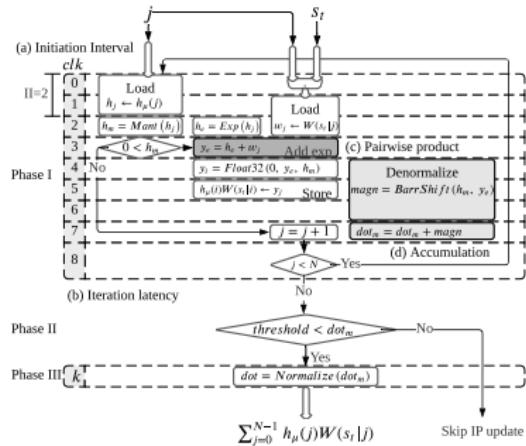
# Dot-Product with Hybrid Custom Floating-Point Approximation



**Figure:** Dot-product hardware module with hybrid custom floating-point approximation

$$L_{\text{custom}} = 2N + 11$$

# Dot-product with Hybrid Logarithmic Approximation



**Figure:** Dot-product hardware module with hybrid logarithmic approximation

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

## Deployment with Standard Floating-Point

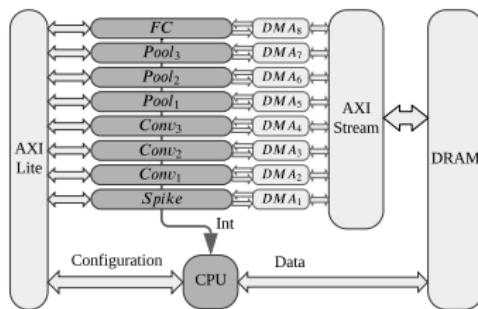


Figure: System overview of the top-level architecture with 8 processing units

# Deployment with Standard Floating-Point

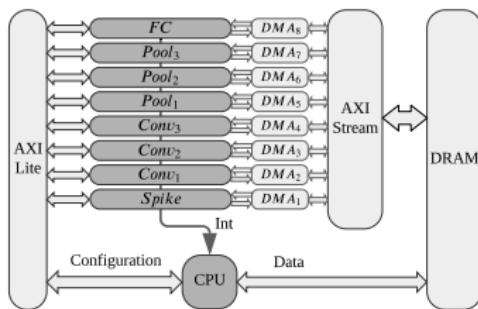


Figure: System overview of the top-level architecture with 8 processing units

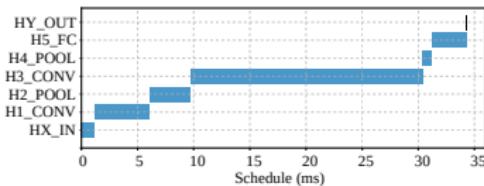


Figure: Computation on embedded CPU

# Deployment with Standard Floating-Point

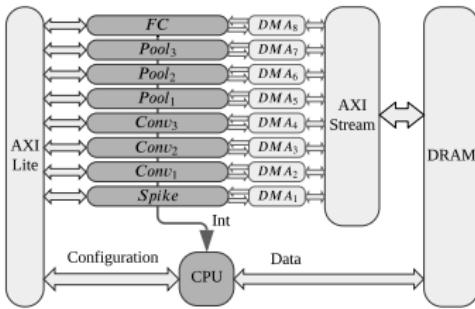


Figure: System overview of the top-level architecture with 8 processing units

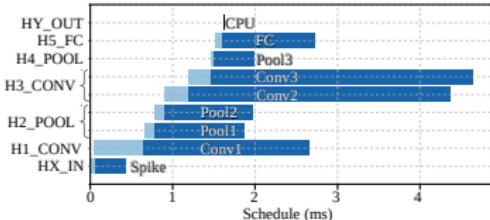


Figure: Performance of processing units with standard floating-point with **acceleration of 10.7X**

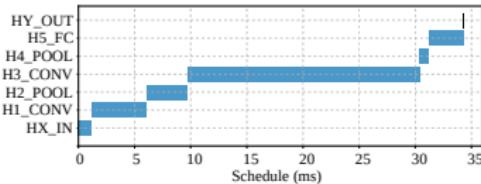


Figure: Computation on embedded CPU

# Deployment with Standard Floating-Point

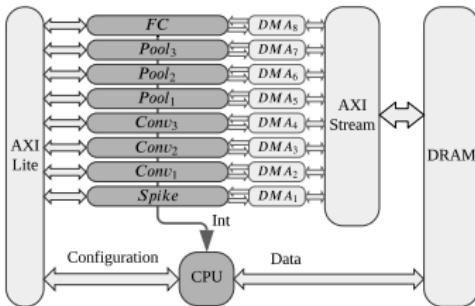


Figure: System overview of the top-level architecture with 8 processing units

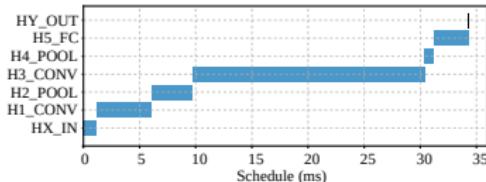


Figure: Computation on embedded CPU

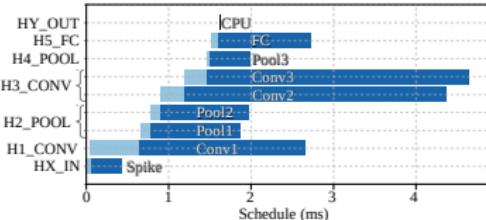


Figure: Performance of processing units with standard floating-point with acceleration of 10.7X

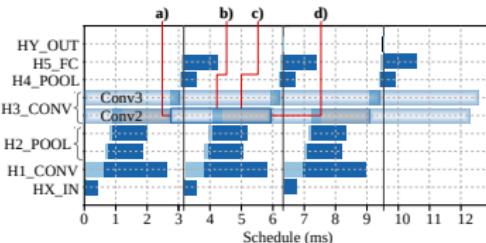


Figure: Performance bottleneck of cyclic computation on processing units with standard floating-point

# Deployment with Custom Floating-Point

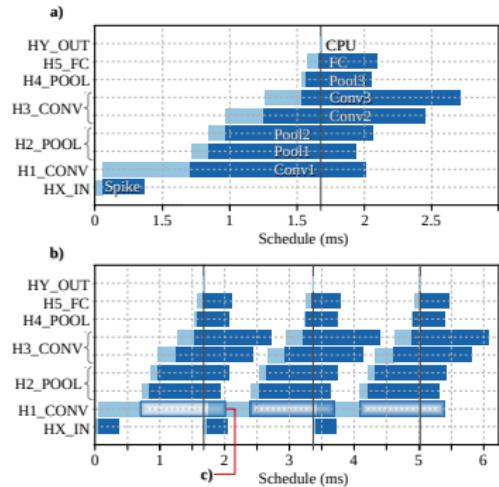


Figure: Performance on processing units with hybrid 8-bit floating-point with acceleration of 20.5X

# Deployment with Custom Floating-Point

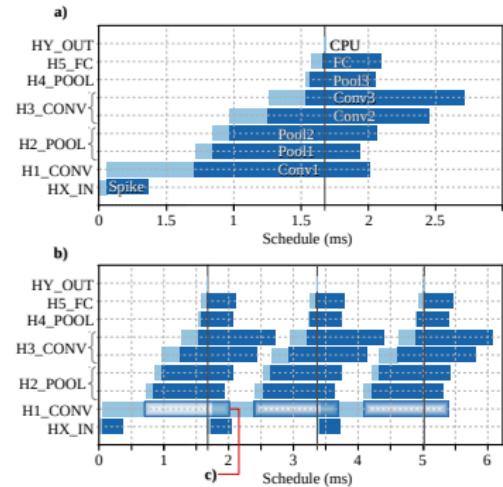


Figure: Performance on processing units with hybrid 8-bit floating-point with acceleration of 20.5X

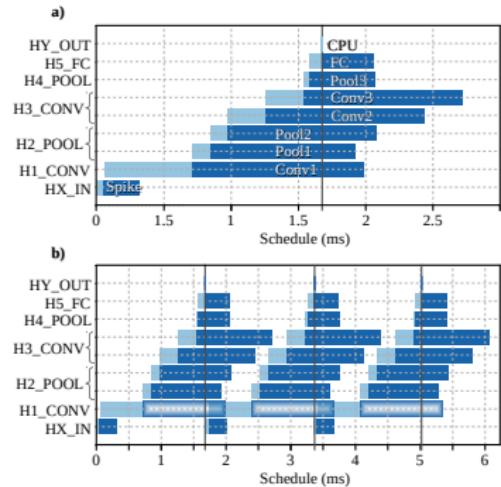


Figure: Performance of processing units with hybrid 4-bit logarithmic with acceleration of 20.5X

# Quantization Impact: Noise Tolerance

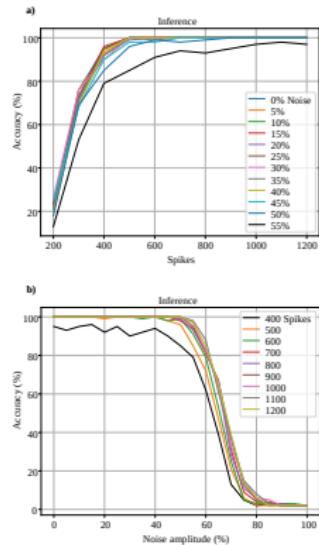


Figure: Noise tolerance with 32-bit floating-point

# Quantization Impact: Noise Tolerance

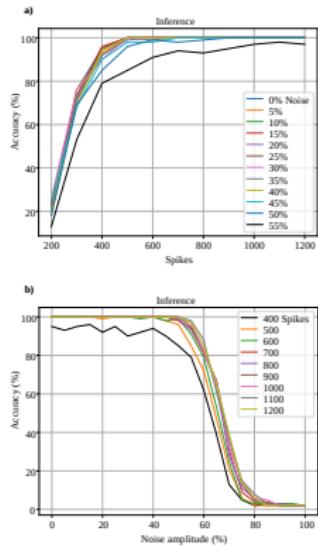


Figure: Noise tolerance with 32-bit floating-point

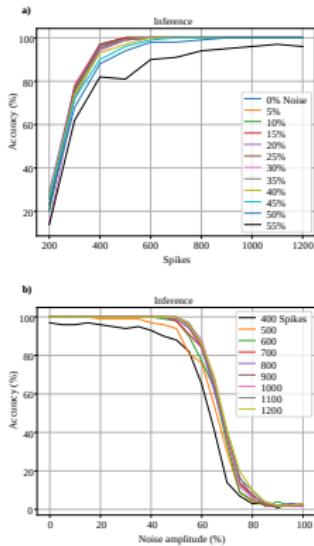


Figure: Noise tolerance with hybrid 8-bit floating-point

# Quantization Impact: Noise Tolerance

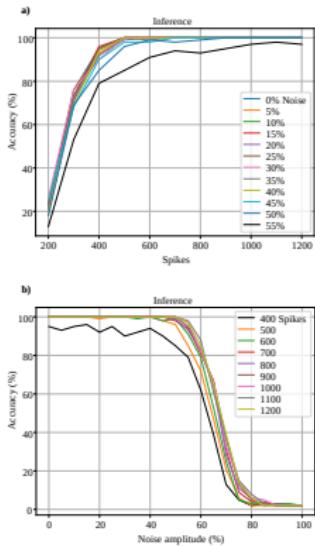


Figure: Noise tolerance with 32-bit floating-point

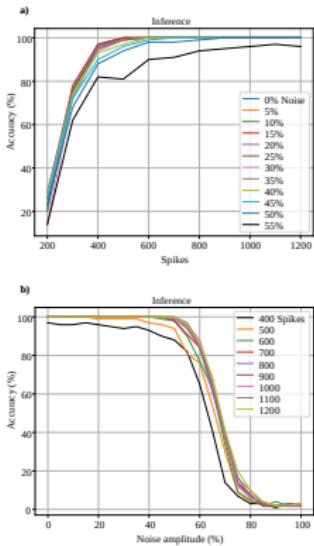


Figure: Noise tolerance with hybrid 8-bit floating-point

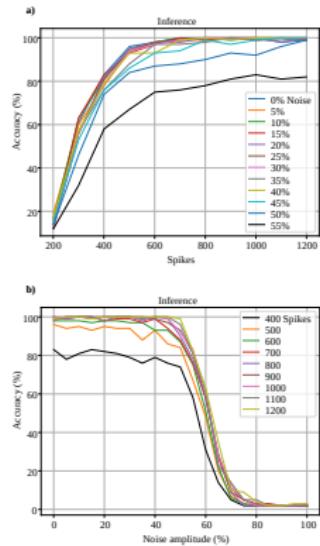


Figure: Noise tolerance with hybrid 4-bit logarithmic

# Conv2D Tensor Processor

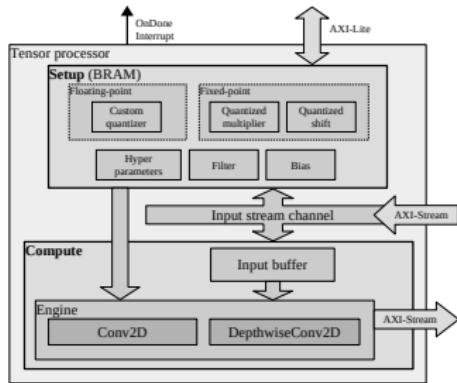


Figure: High level architecture of tensor processor

# Conv2D Tensor Processor

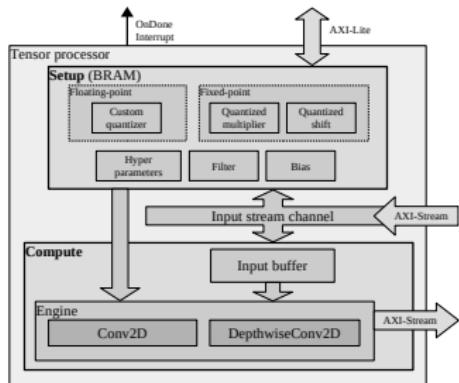


Figure: High level architecture of tensor processor

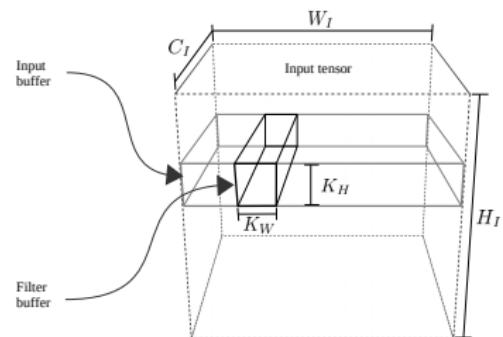


Figure: On-chip memory buffers

# Tensor Processor Setup Data Frame

## Setup transaction buffer stream

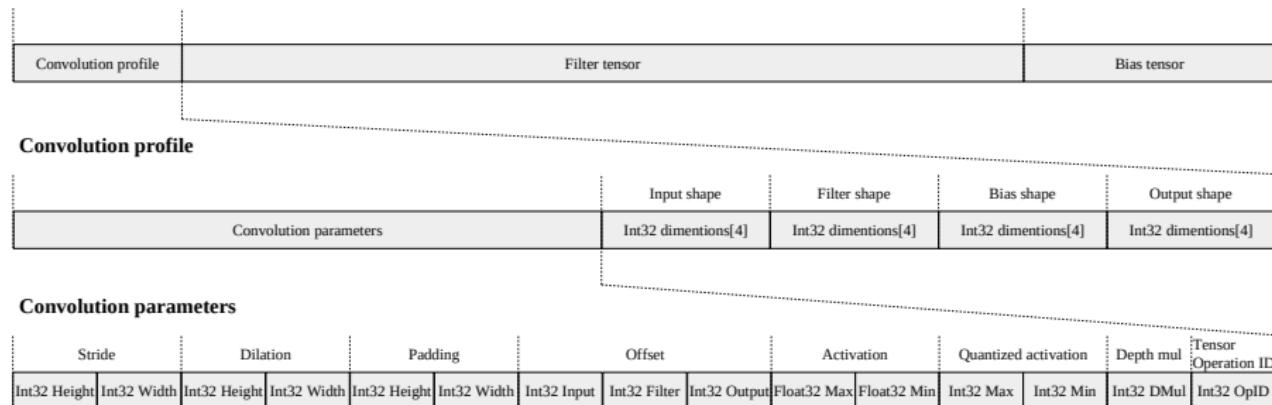


Figure: Setup transaction buffer stream

# Embedded System Architecture with Tensor Processor

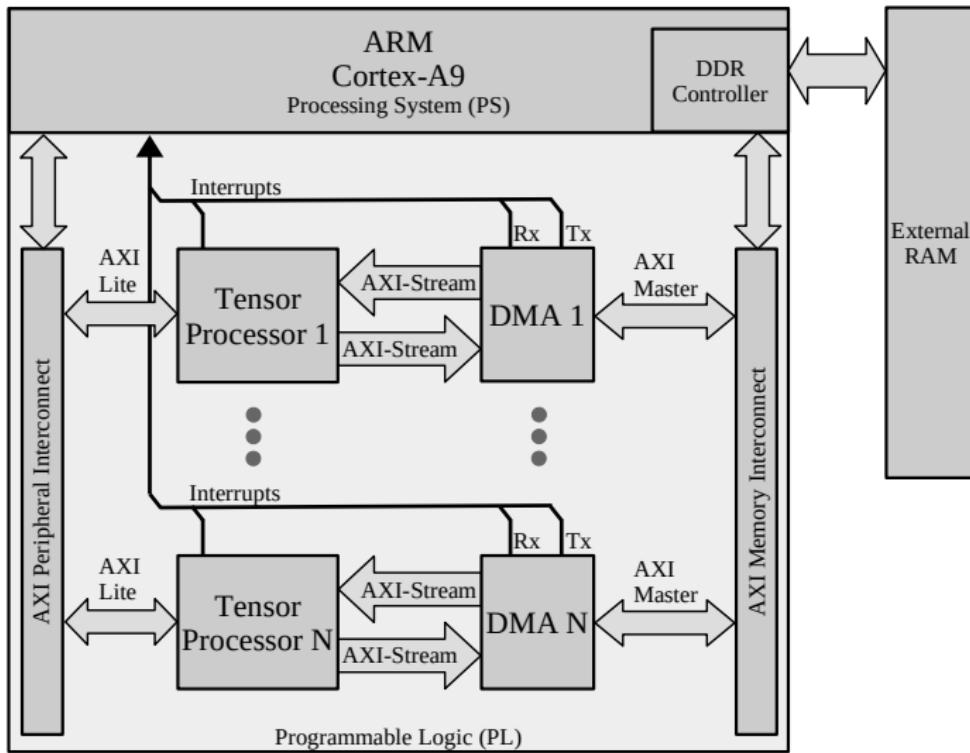


Figure: Embedded system architecture

# Comparison with Related Work with Tensor Processor

Comparison with related work with floating-point and fixed-point.

Platform	Chunsheng <i>et al.</i> [1]	Chen <i>et al.</i> [2]	BFP [3]	Paolo <i>et al.</i> [4]	This work
Device	XC7VX690T	XC7K325T	XC7VX690T	XC7Z007S	XC7Z007S
Year	2017	2019	2019	2019	2023
Dev. kit cost	\$7,494	\$1,299	\$7,494	\$89	\$89
Format (activation/weight)	FP 16-bit	FP 8-bit / 8-bit	FP 16-bit / 8-bit	INT 16-bit	FP 32-bit / 6-bit
Frequency (MHz)	200	200	200	80	200
Peak power efficiency (GFLOP/s/W)	18.72	115.40	82.88	2.98	5.74
Peak throughput (GFLOP/s)	202.42	1086.8	760.83	10.62	0.482
Wall plug power (W)	10.81	9.42	9.18	2.5	2.3
BRAM 36Kb utilization	196.5	234.5	913	44	15
DSP utilization	1728	768	1027	54	20

- [1] Chunsheng Mei, Zhenyu Liu, Yue Niu, Xiangyang Ji, Wei Zhou, and Dongsheng Wang. A 200mhz 202.4 gflops@ 10.8 w vgg16 accelerator in xilinx vx690t. In *2017 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, pages 784–788. IEEE, 2017.
- [2] Chen Wu, Mingyu Wang, Xinyuan Chu, Kun Wang, and Lei He. Low-precision floating-point arithmetic for high-performance fpga-based cnn acceleration. *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, 15(1):1–21, 2021.
- [3] Xiaocong Lian, Zhenyu Liu, Zhourui Song, Jiwu Dai, Wei Zhou, and Xiangyang Ji. High-performance fpga-based cnn accelerator with block-floating-point arithmetic. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 27(8):1874–1885, 2019.
- [4] Paolo Meloni, Antonio Garufi, Gianfranco Deriu, Marco Carreras, and Daniela Loi. Cnn hardware acceleration on a low-power and low-cost apsoc. In *2019 Conference on Design and Architectures for Signal and Image Processing (DASIP)*, pages 7–12. IEEE, 2019.