

SEP 01, 2025

# NeuraLUT-Assemble and LUT-based NNs Tutorial

Marta Andronic and Oliver Cassidy  
Imperial College London

[marta.andronic18@imperial.ac.uk](mailto:marta.andronic18@imperial.ac.uk); [olly.cassidy23@imperial.ac.uk](mailto:olly.cassidy23@imperial.ac.uk)

# Agenda

---

## 1. Motivation

## 2. LUT-based NNs

- I. LogicNets - Umuroglu *et al.*, FPL 2020
- II. PolyLUT - Andronic *et al.*, FPT 2023
- III. NeuraLUT - Andronic *et al.*, FPL 2024
- IV. ReducedLUT – Cassidy *et al.*, FPGA 2025
- V. AmigoLUT – Weng *et al.*, FPGA 2025
- VI. Hardware-aware structured pruning – Andronic *et al.*, TC 2025
- VII. NeuraLUT-Assemble – Andronic *et al.*, FCCM 2025

## 3. NeuraLUT-Assemble demo and how to design a neural network

## 4. In-context results (against DWN and FINN)

## 5. FPGA demo

## 6. List of all open-source repositories

# Motivation

---

Bridging the gap between powerful computational models and real-world, on-edge processing.



**Ultra-low Latency**



**Bandwidth Savings**



**Enhanced Privacy**



**Reliability**

**USE CASES**  
PHYSICS EXPERIMENTS  
CYBERSECURITY

# Solutions and technologies

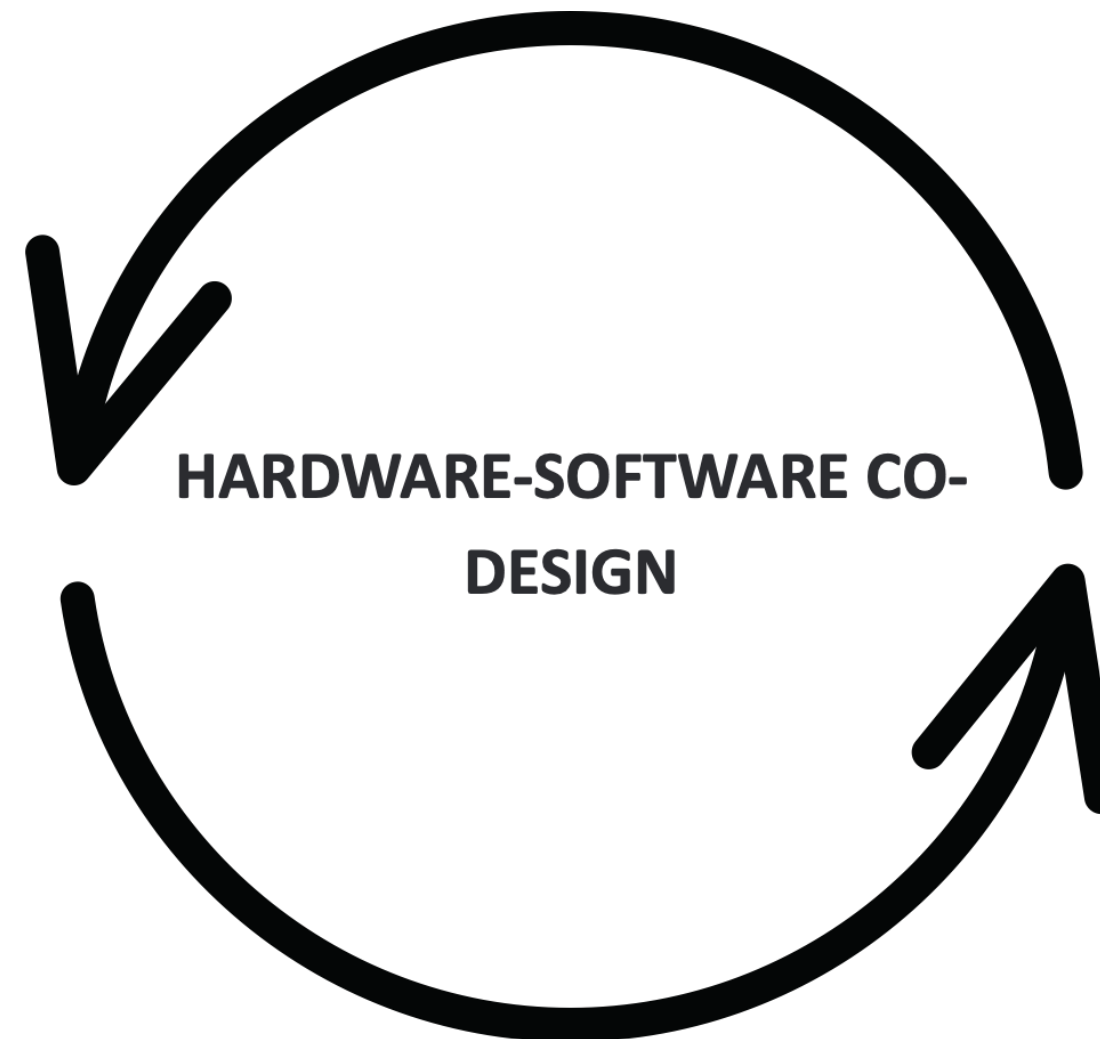
---

Edge devices are highly resource constrained.

## MODEL OPTIMIZATION

Reduce its size and computational requirements

Techniques: Quantization, Pruning, Knowledge distillation.



## EDGE-SPECIFIC HARDWARE

Specialized hardware can run DNNs more efficiently

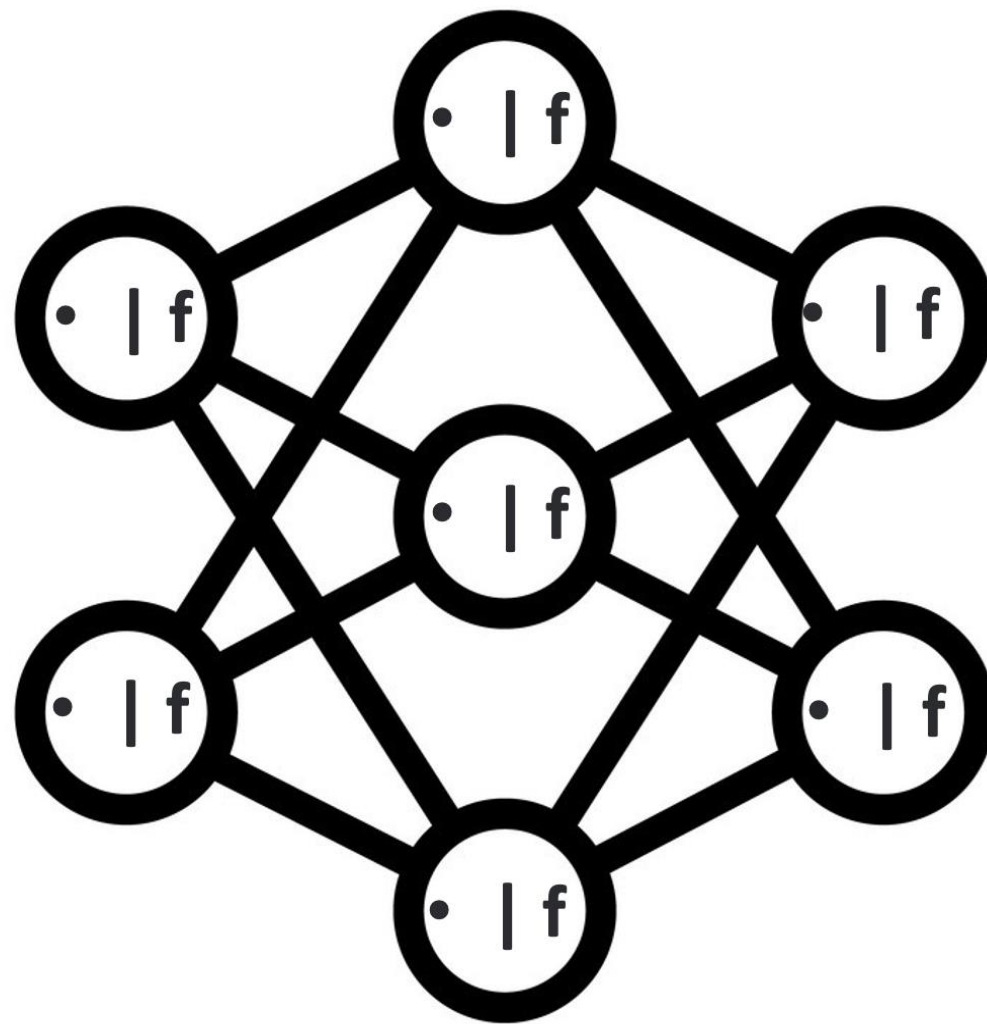
FPGAs: Highly customizable, highly parallel, rapid design iteration.

# LUT-based neural networks

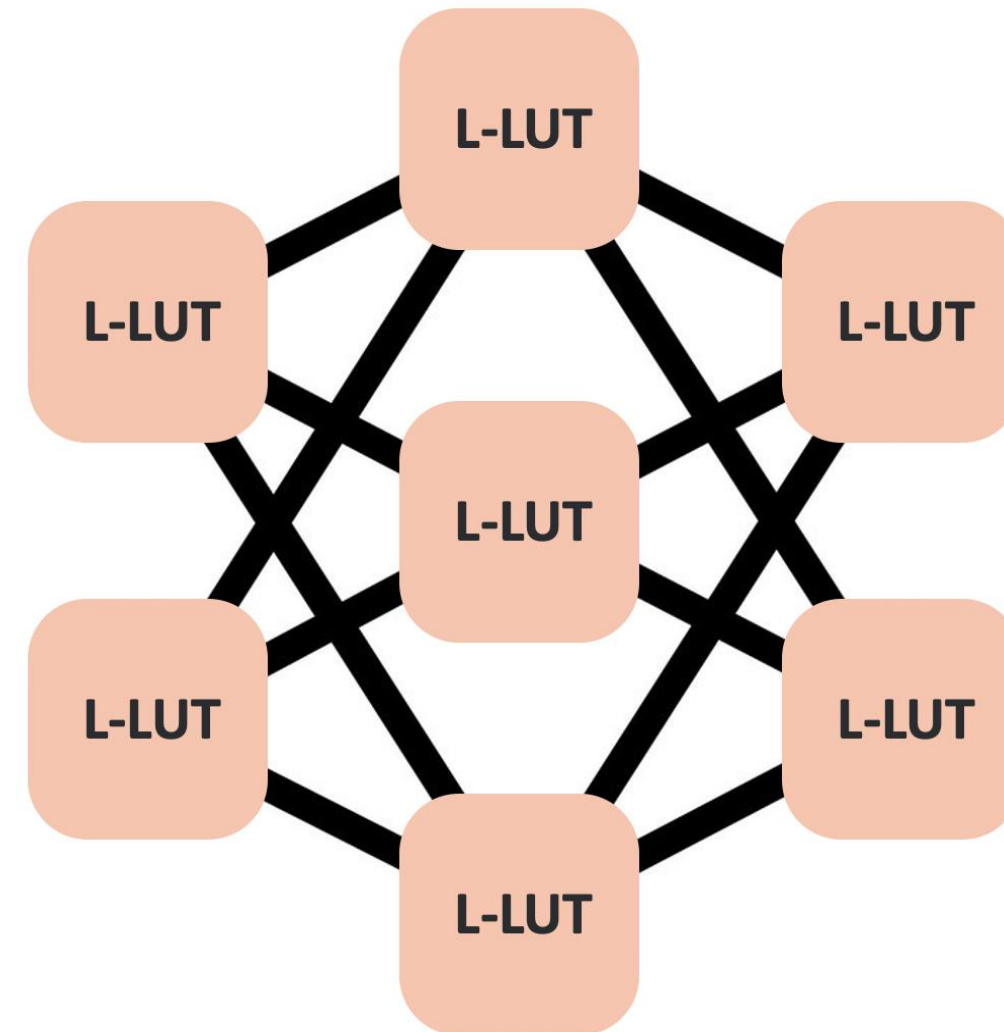
# LUT-based neural networks

---

Conventional NNs

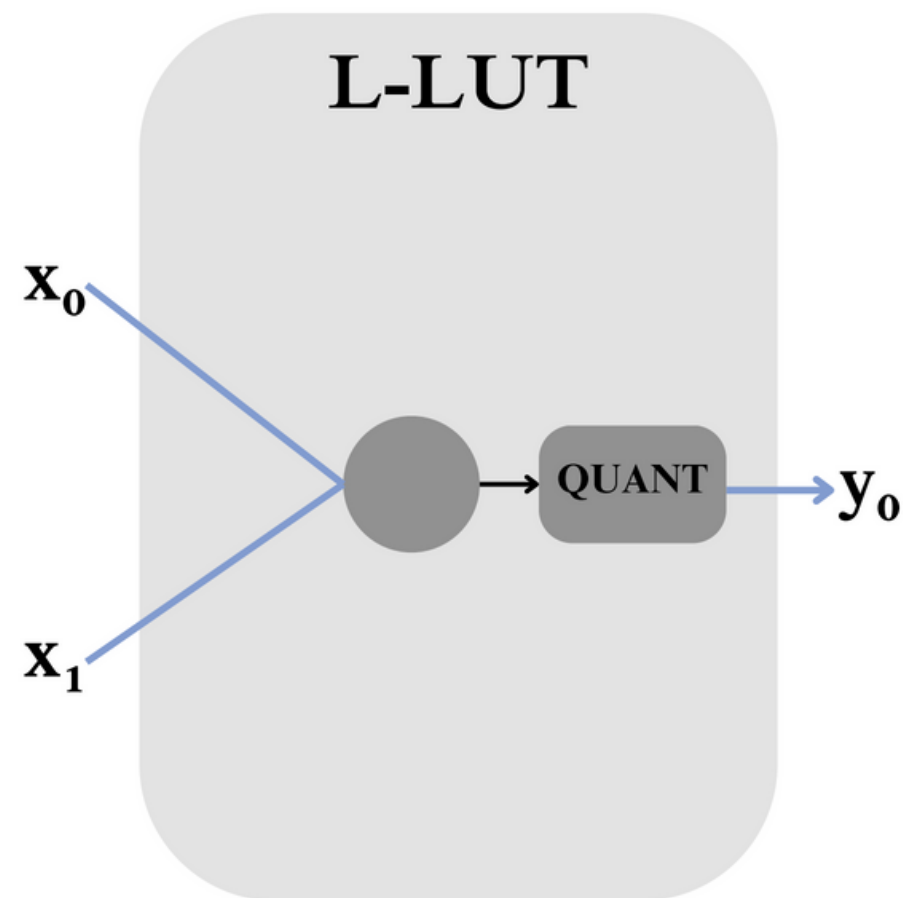


LUT-based NNs



# LogicNets

---



Neuron absorbed in a Logical LUT.

**Supports larger bit-widths**

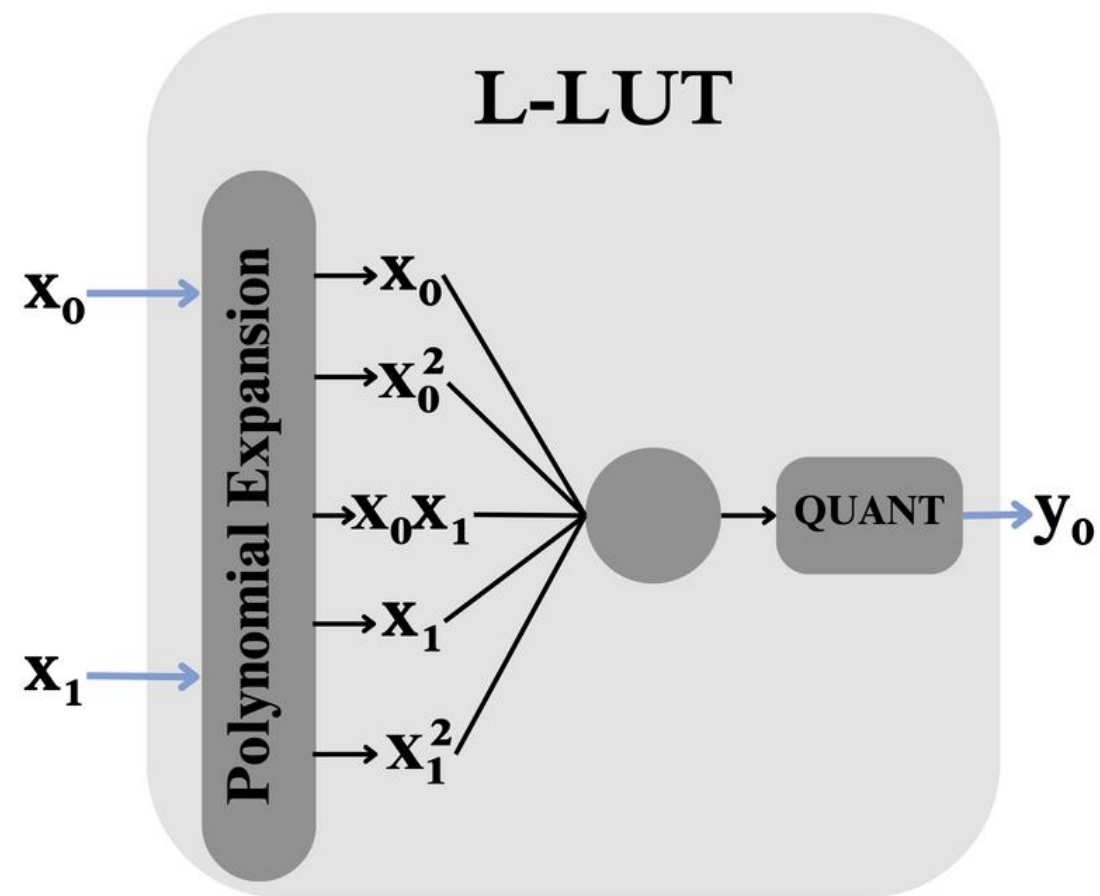
**Post training the NN gets converted to a network of L-LUTs through full enumeration.**

**Logical LUTs get mapped to one or more Physical LUTs by the synthesis tools**

**L-LUTs can hide any function**

# PolyLUT

---



L-LUT hides a multivariate polynomial.

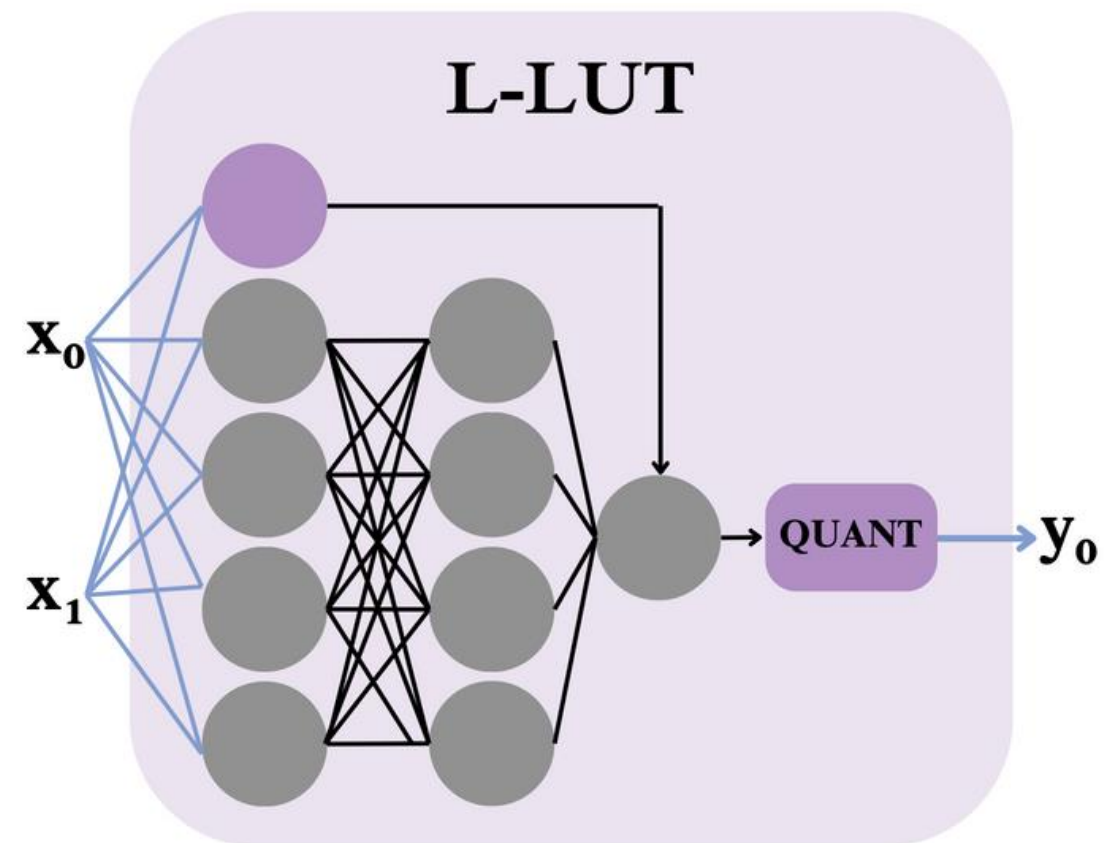
Supports larger bit-widths

Expansion of the feature vector  $x$  with all of its monomials up to a parametric degree  $D$

The number of trainable parameters can be controlled by tuning  $D$

# NeuraLUT

---



Neuron hides MLP with skip connections.

Supports larger bit-widths

Integrating skip connections facilitates the flow of gradients

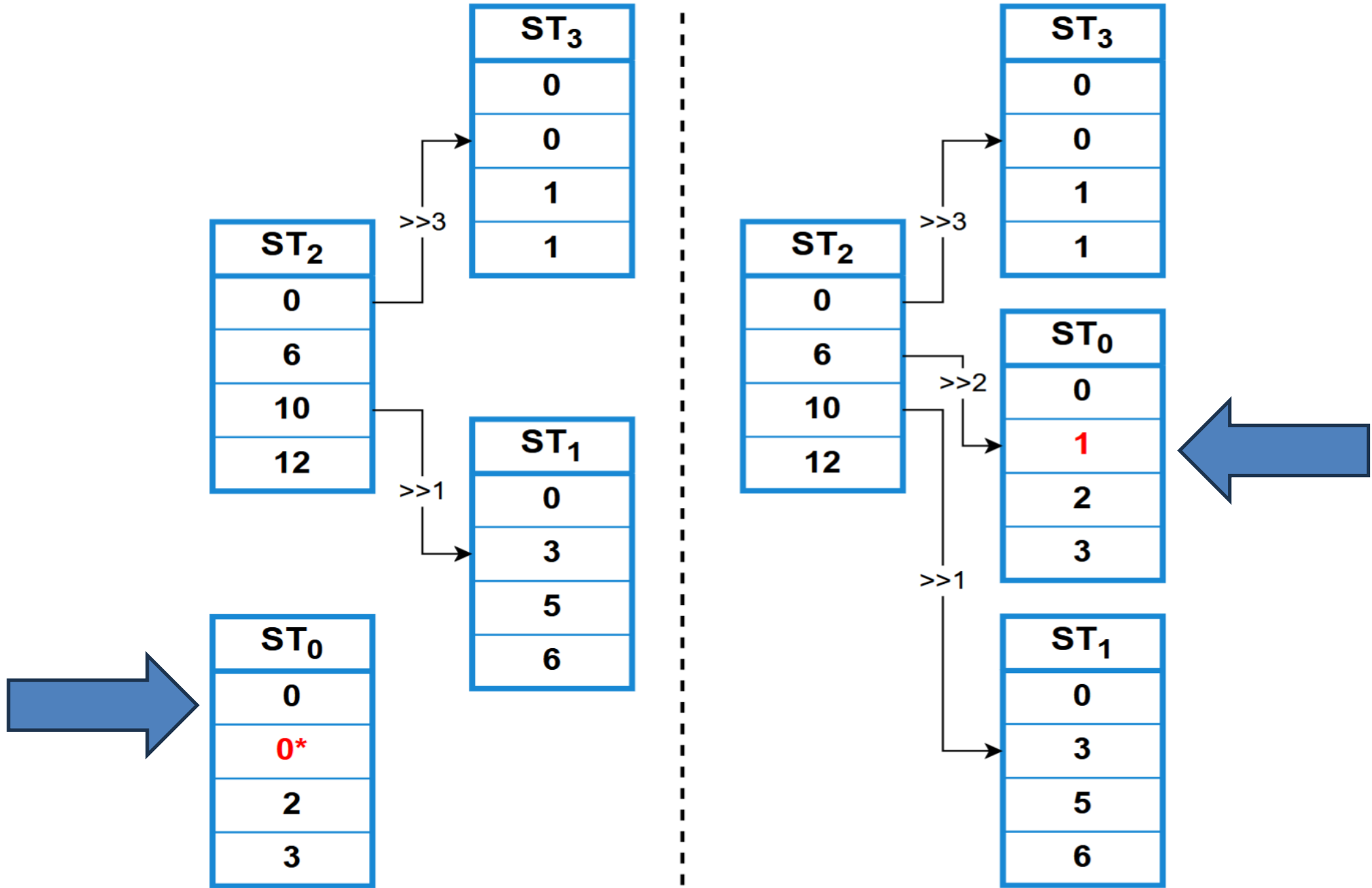
The number of trainable parameters can be controlled by tuning the width and depth of the sub-network

# Optimizations on the hardware side

# ReducedLUT

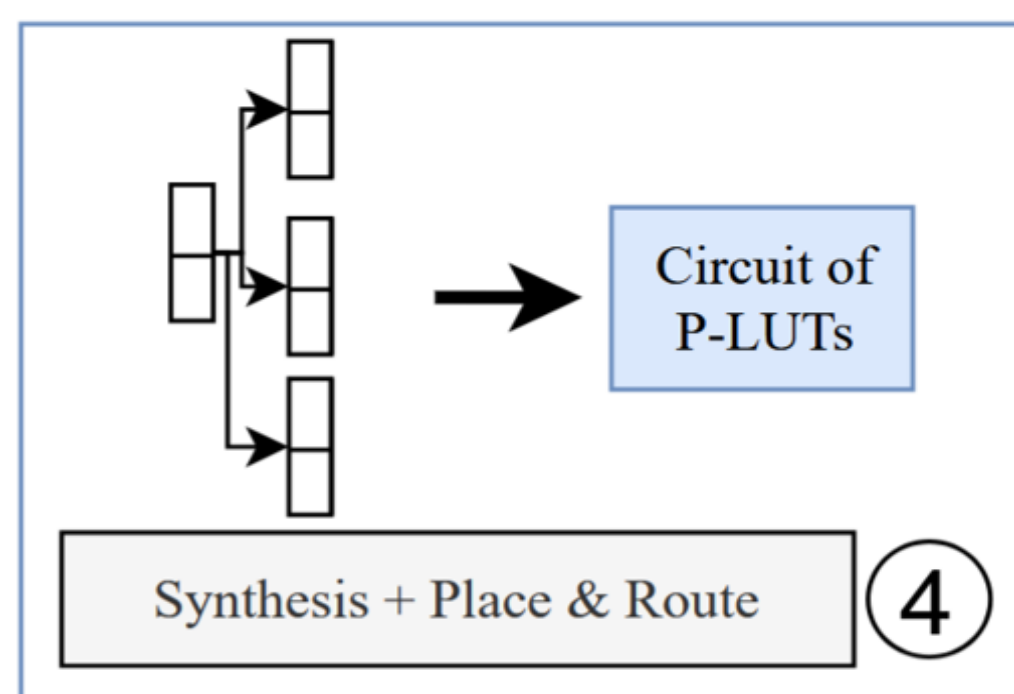
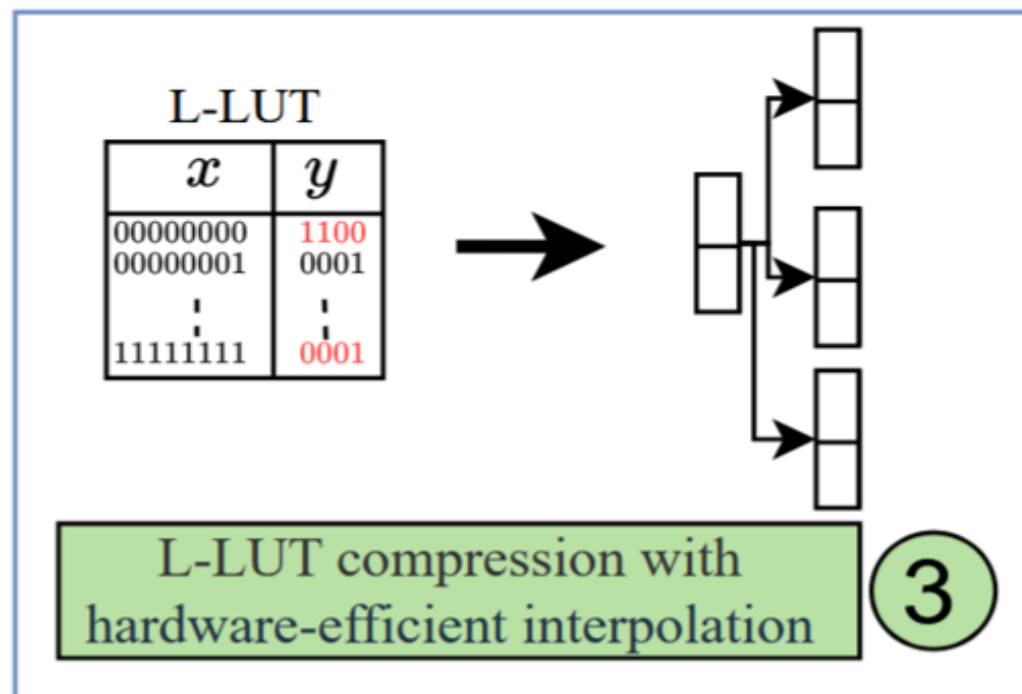
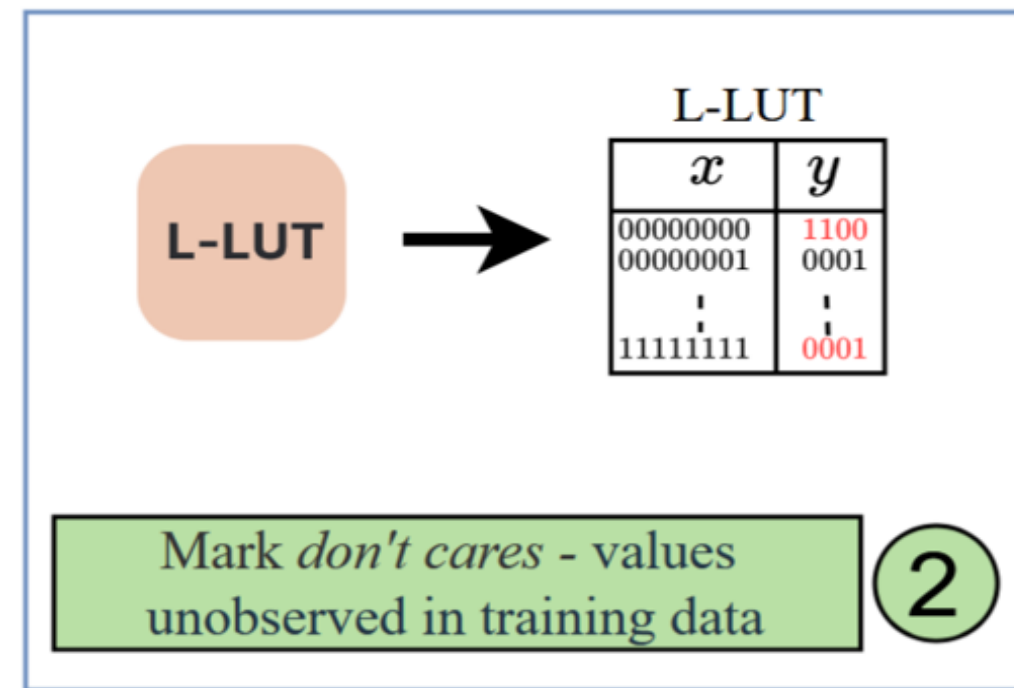
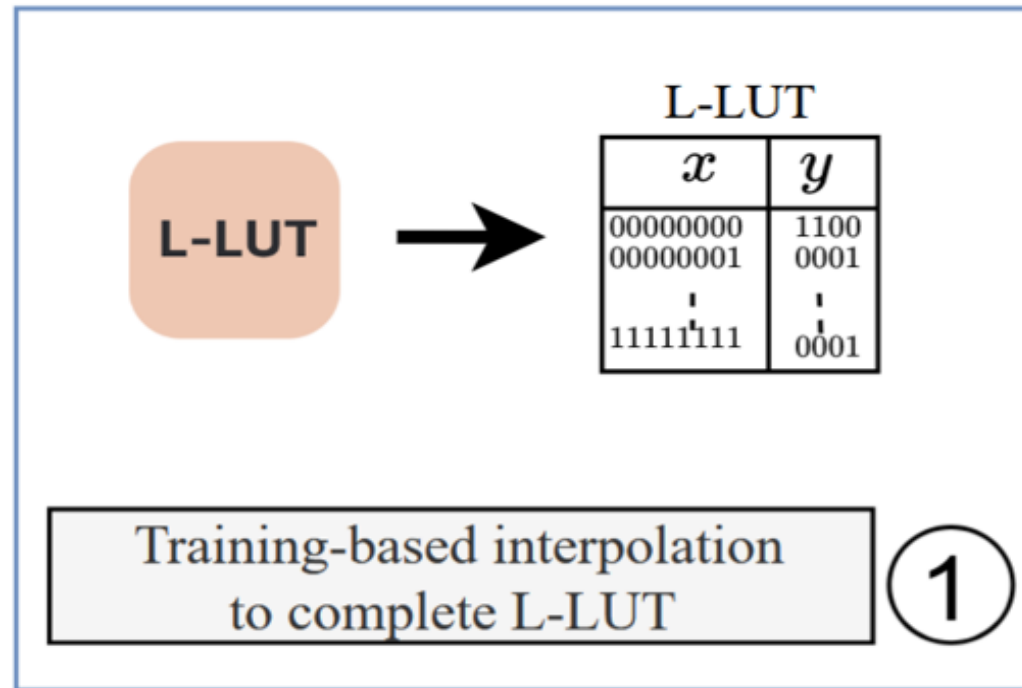
CompressedLUT

ReducedLUT



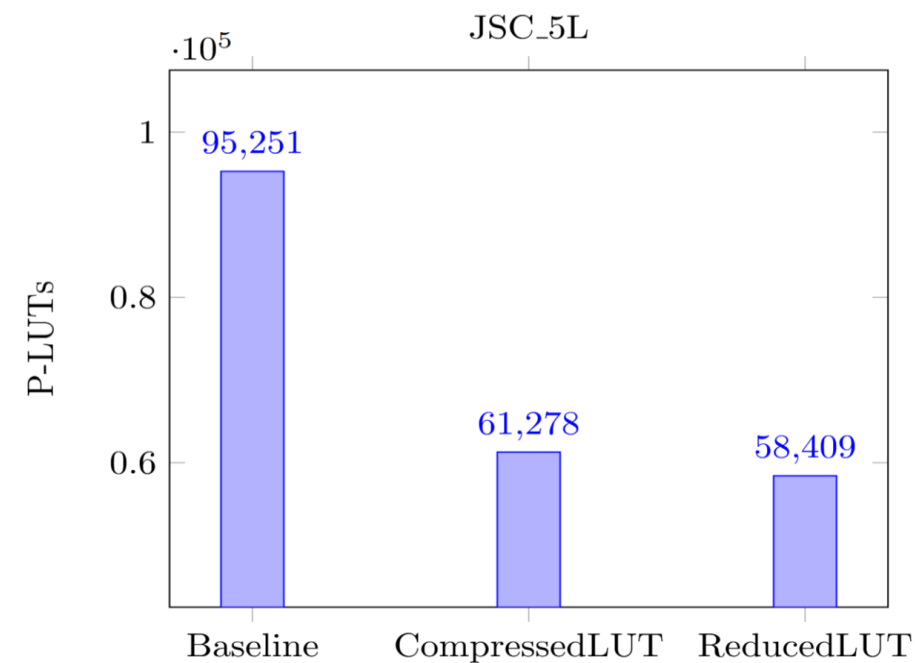
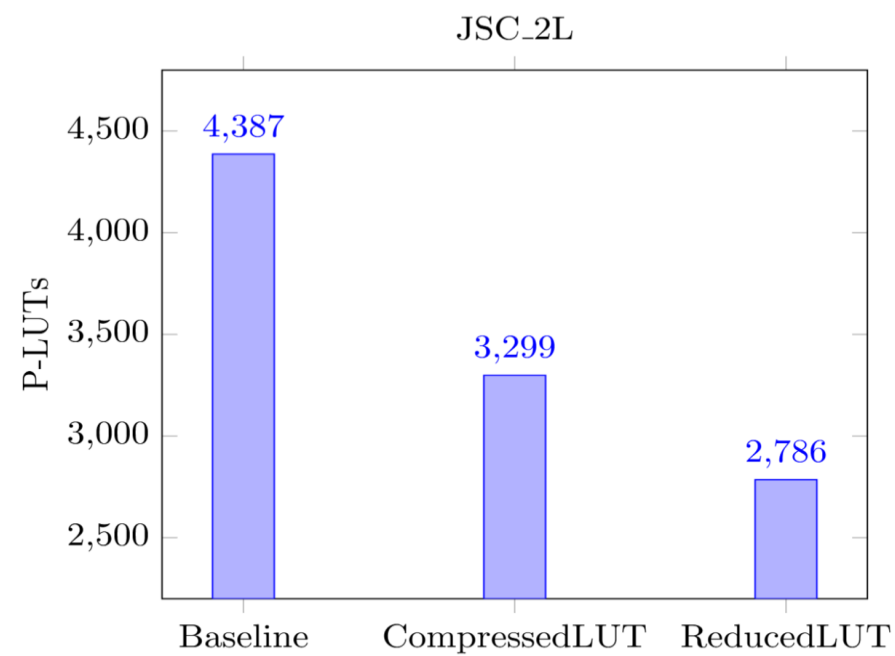
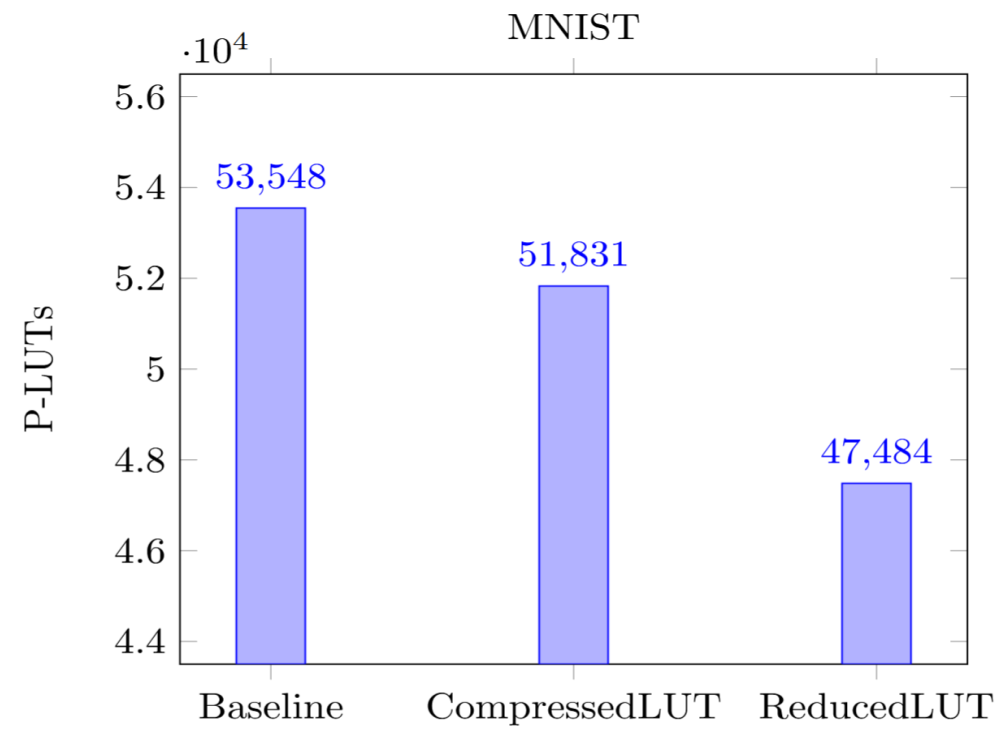
Methodology

# ReducedLUT



# Toolflow

# ReducedLUT

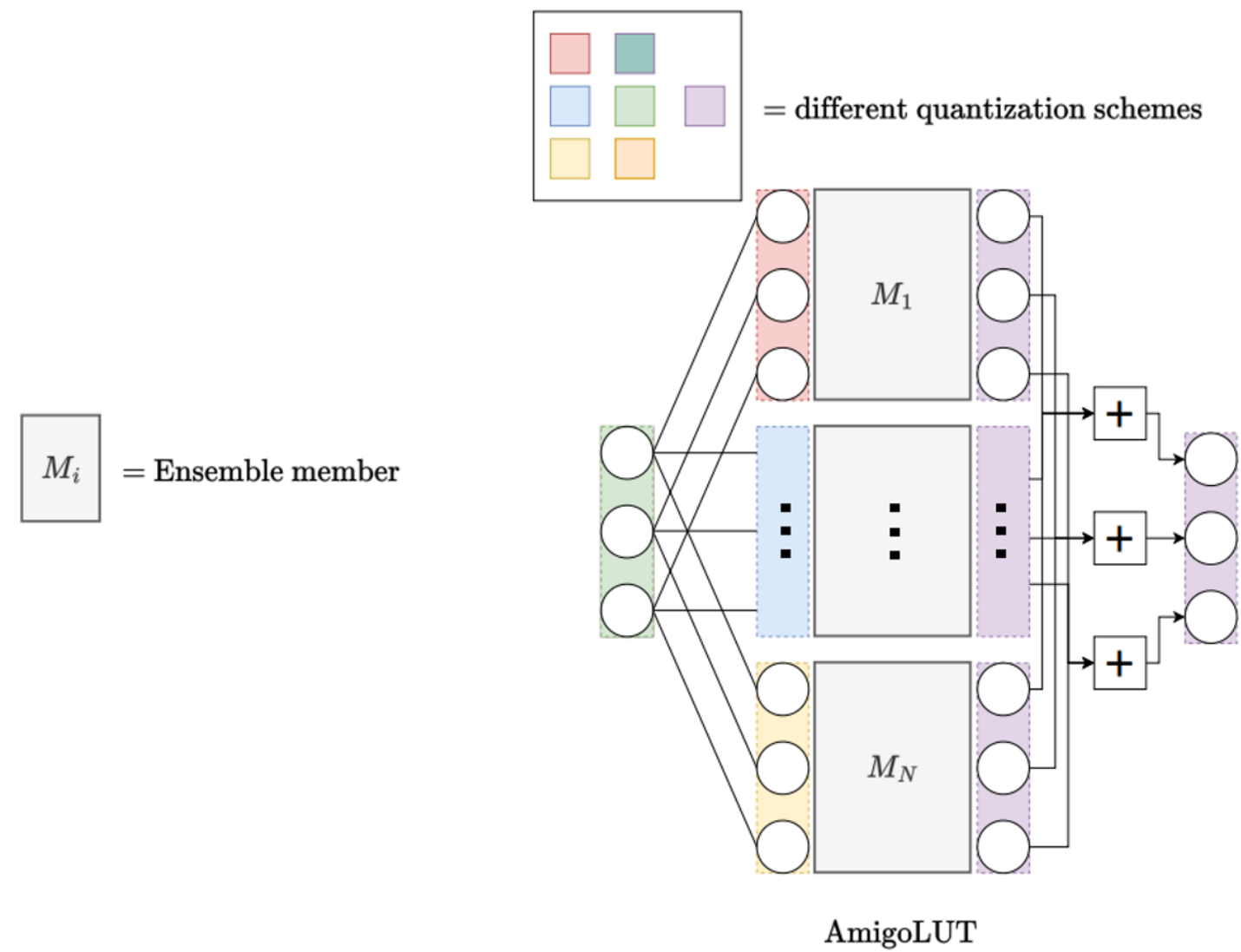


- **39%** less area utilization
- **0.01%** maximum test accuracy loss
- **Up to 50%** of maximum frequency lost recovered

# Scaling up

The slide features a dark grey background. At the bottom, there are several thin, light grey wavy lines that create a sense of motion or depth, resembling a stylized horizon or a decorative border.

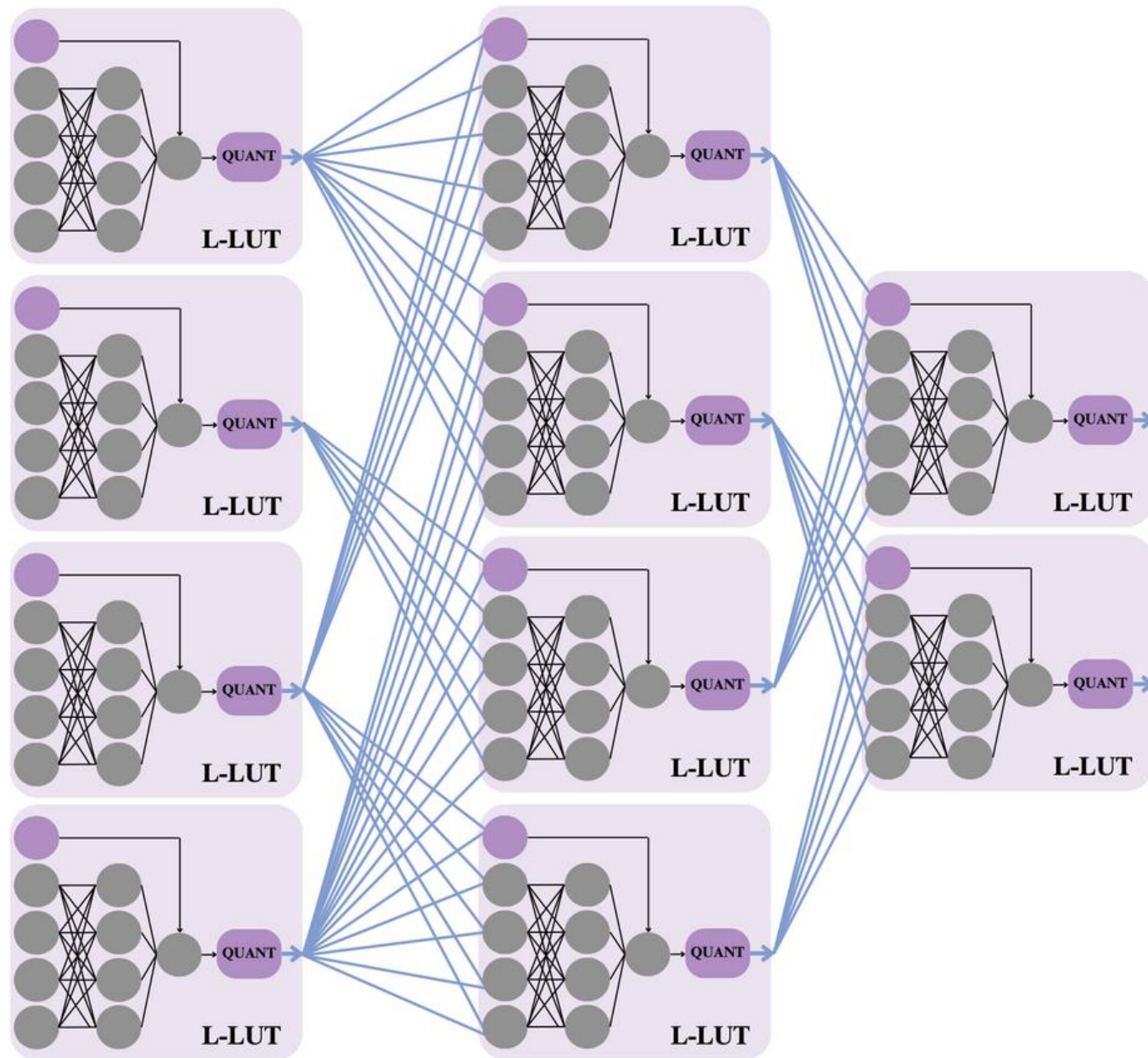
# AmigoLUT



Weak models provide good trade-offs between accuracy and size.

# Fan-in limitations

---

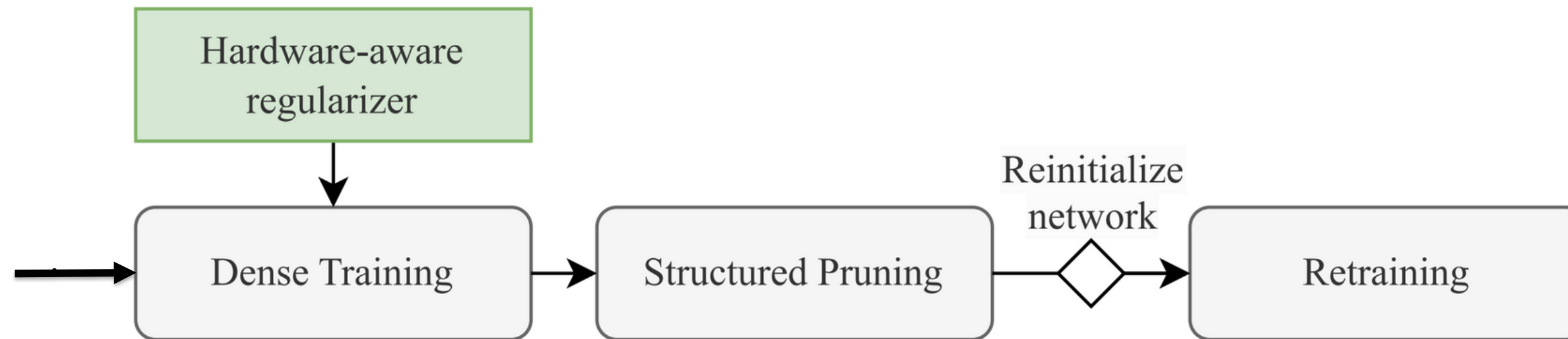


## LIMITATIONS

Complex interactions are only possible between a small number of features.

# PolyLUT (TC extension)

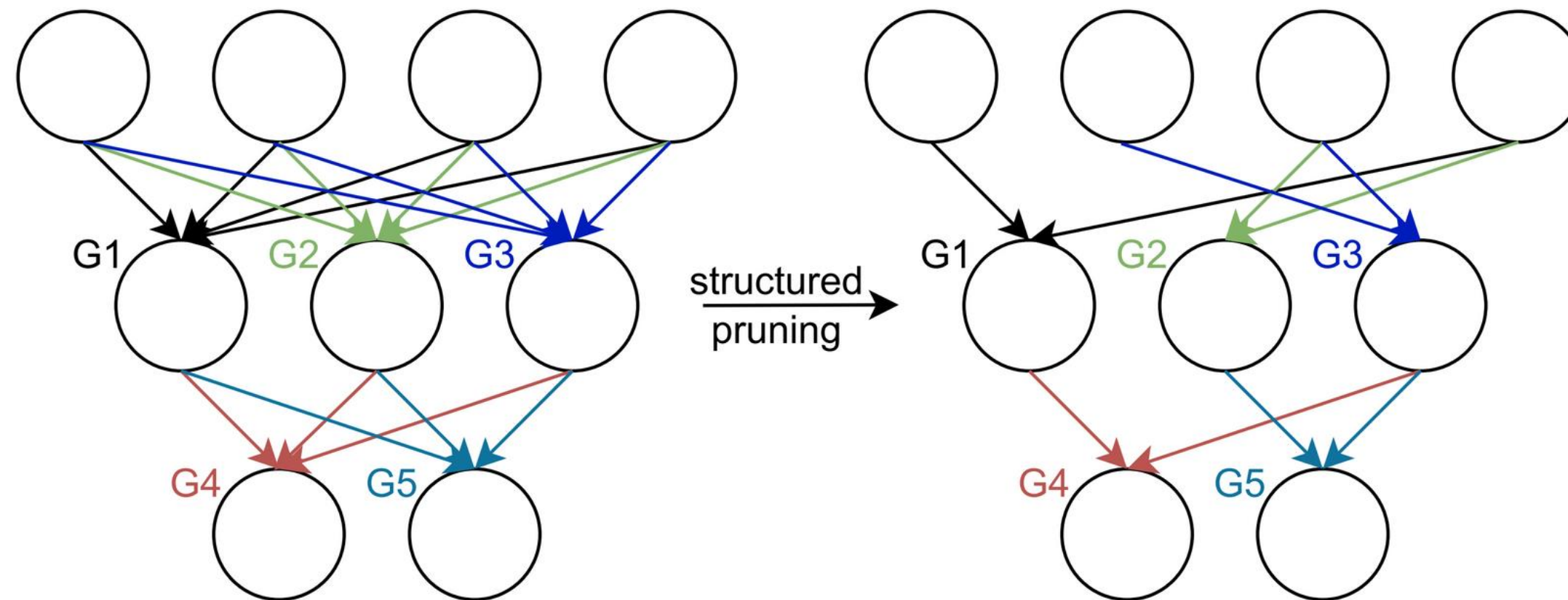
---



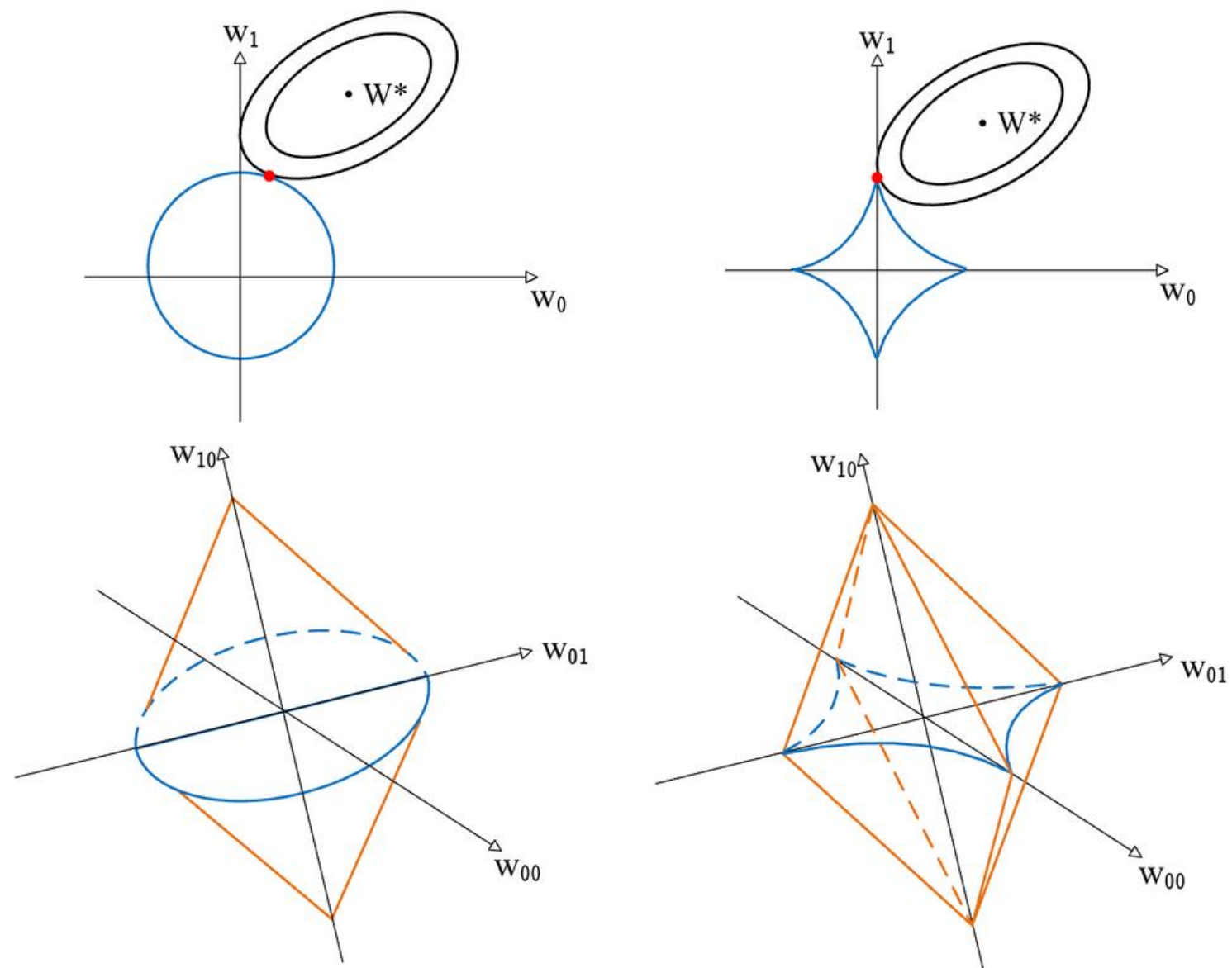
- Hardware-aware group regularizer that encourages per-neuron sparsity.
- Improves the model's accuracy compared to prior work and reduces the standard deviation of accuracy across runs by 6.7×.

# Hardware-aware structured pruning

---



# Hardware-aware regularizer



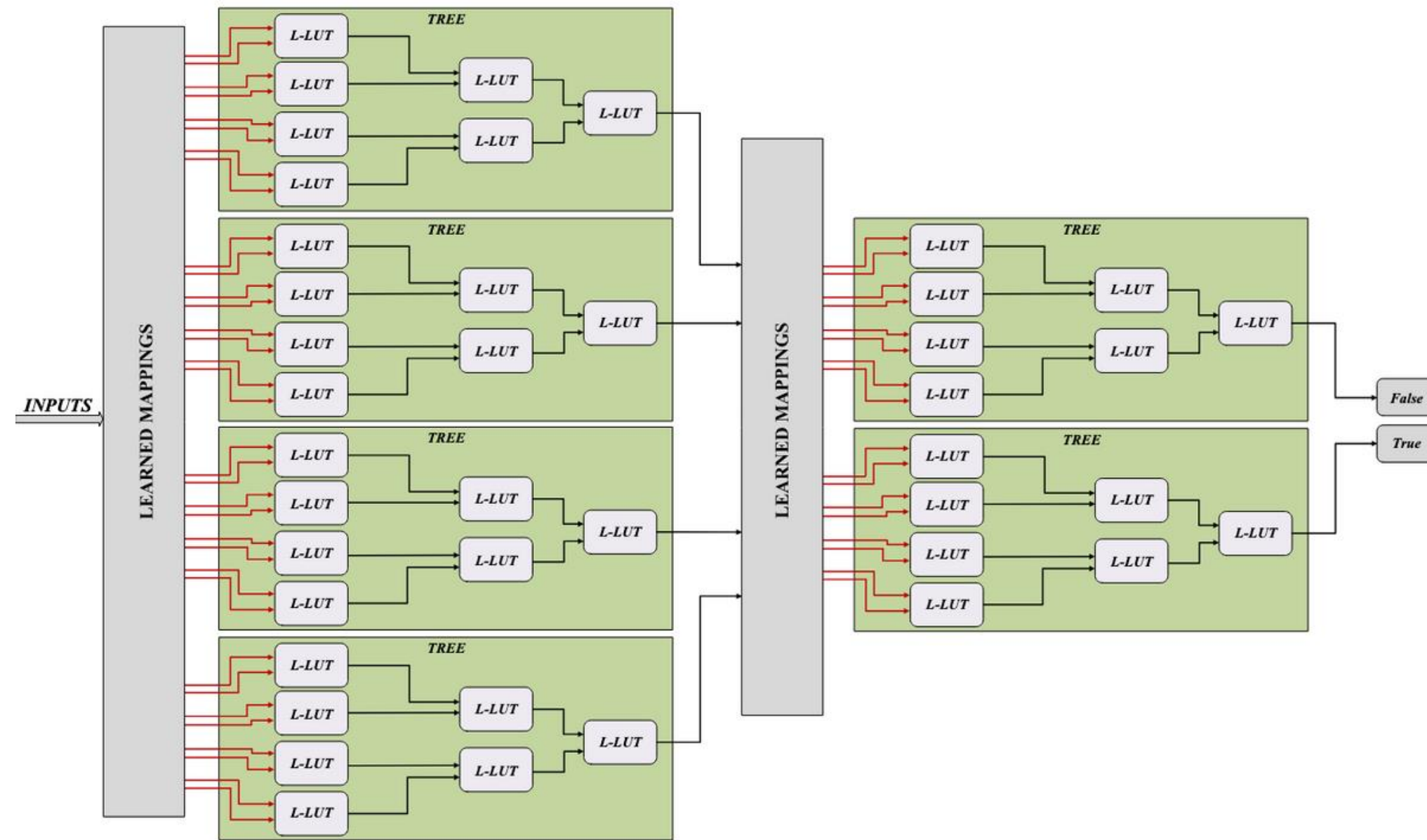
Exponential group regularizer → structured intra-group sparsity

$$\Omega(W) = \lambda \sum_{i=1}^G \|W_i\|_2^2$$

$$\Omega(W) = \lambda_1 \sum_{i=1}^G \lambda_2^{\|W_i\|_1}$$

# NeuraLUT-Assemble Methodology

# NeuraLUT-Assemble

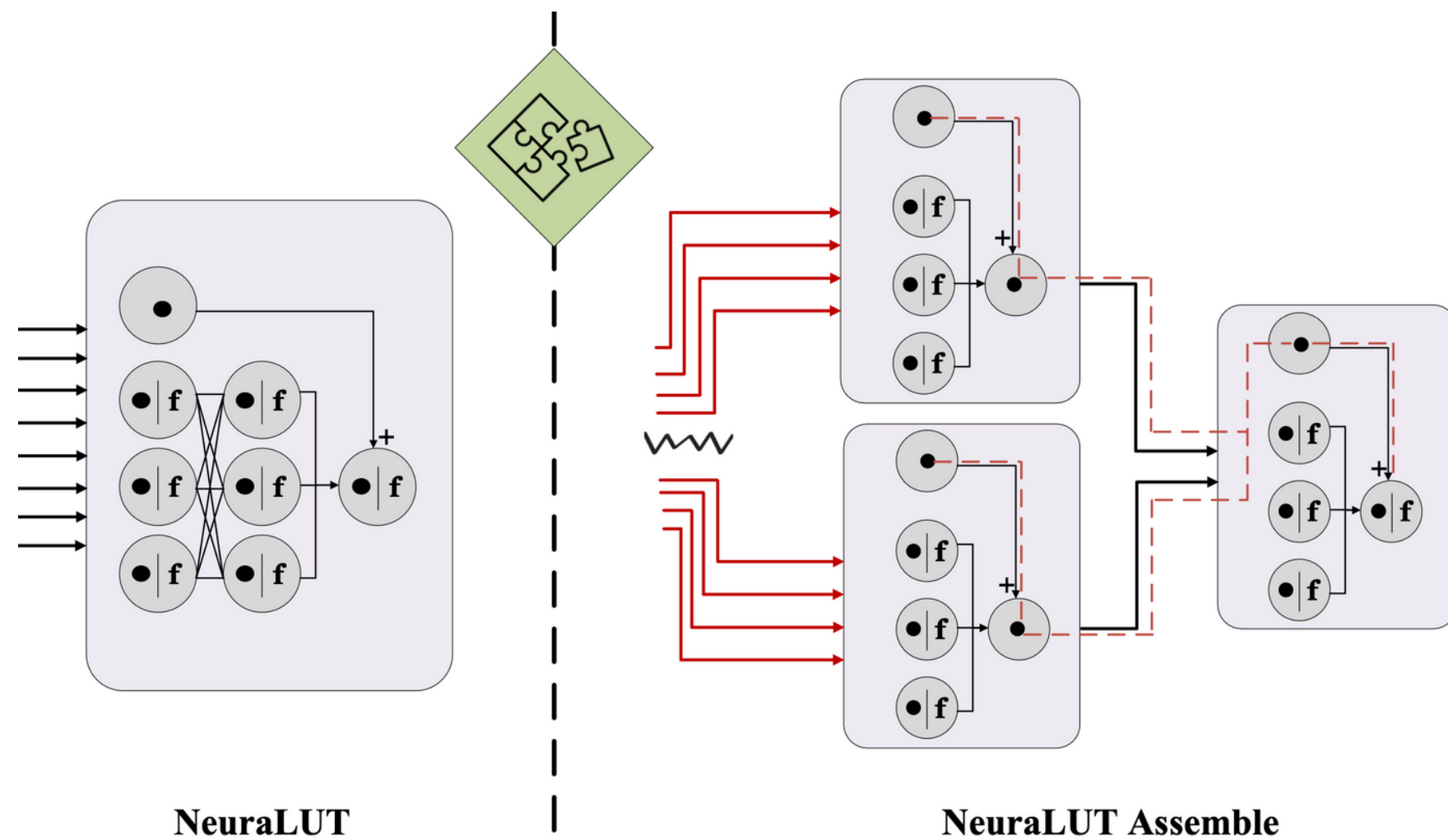


Fully-parametrizable framework that **assembles** multiple NeuraLUT neurons as **tree structures** with larger fan-in.

Directly addresses the exponential scaling challenge.

The grouping of connections at the input of the tree structure is **learned**.

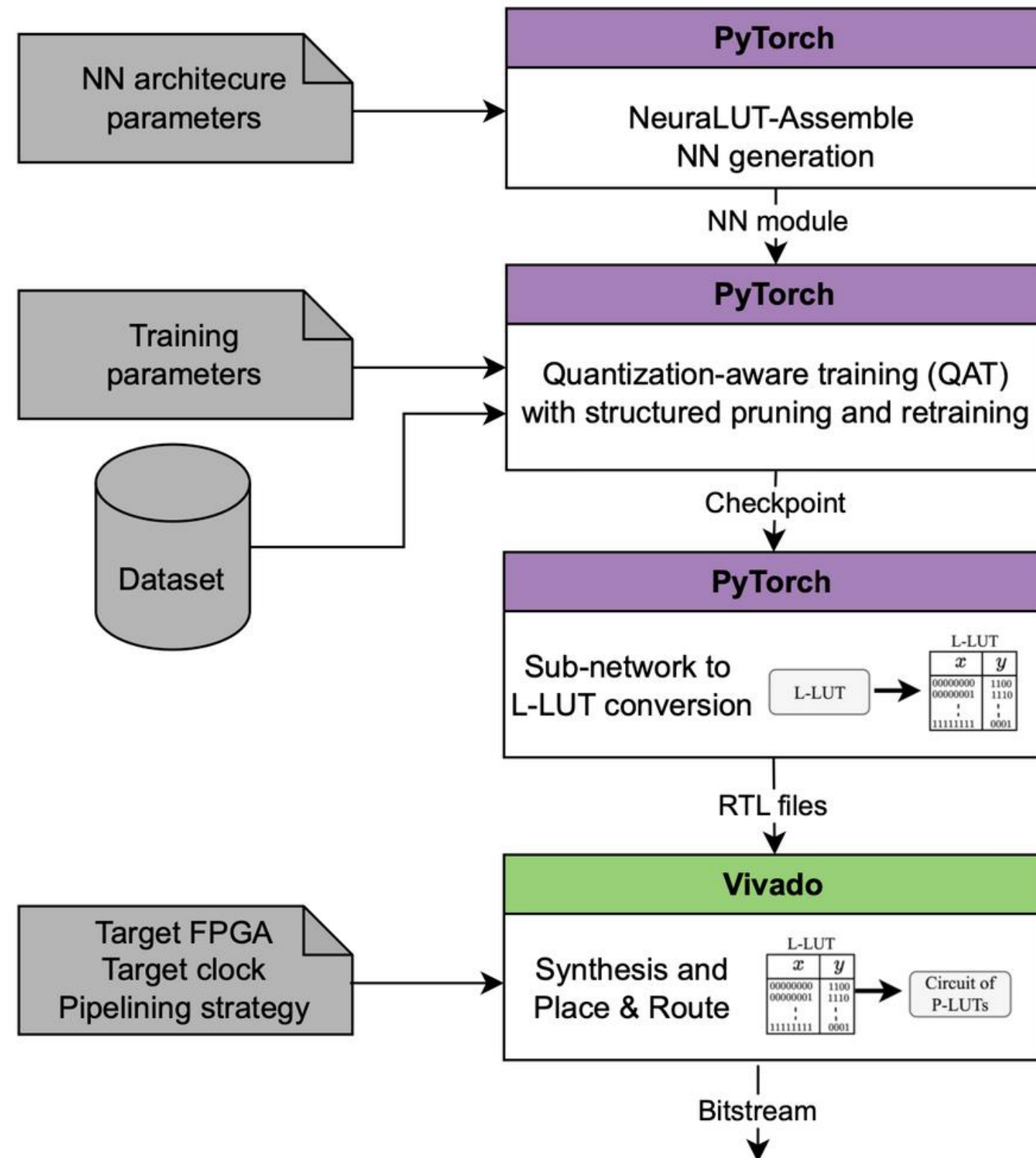
# NeuraLUT-Assemble



Skip-connections are embedded within the L-LUTs, promoting smooth gradient flow throughout the entire tree structure.

These models train fundamentally different functions, therefore the tree structure is trained from scratch.

# NeuraLUT-Assemble



## Toolflow Overview

### Quantization-Aware Training (QAT)

PyTorch-based training with quantization via Brevitas.

### Sub-network to L-LUT Conversion

Automatic transformation of trained sub-networks into L-LUTs.

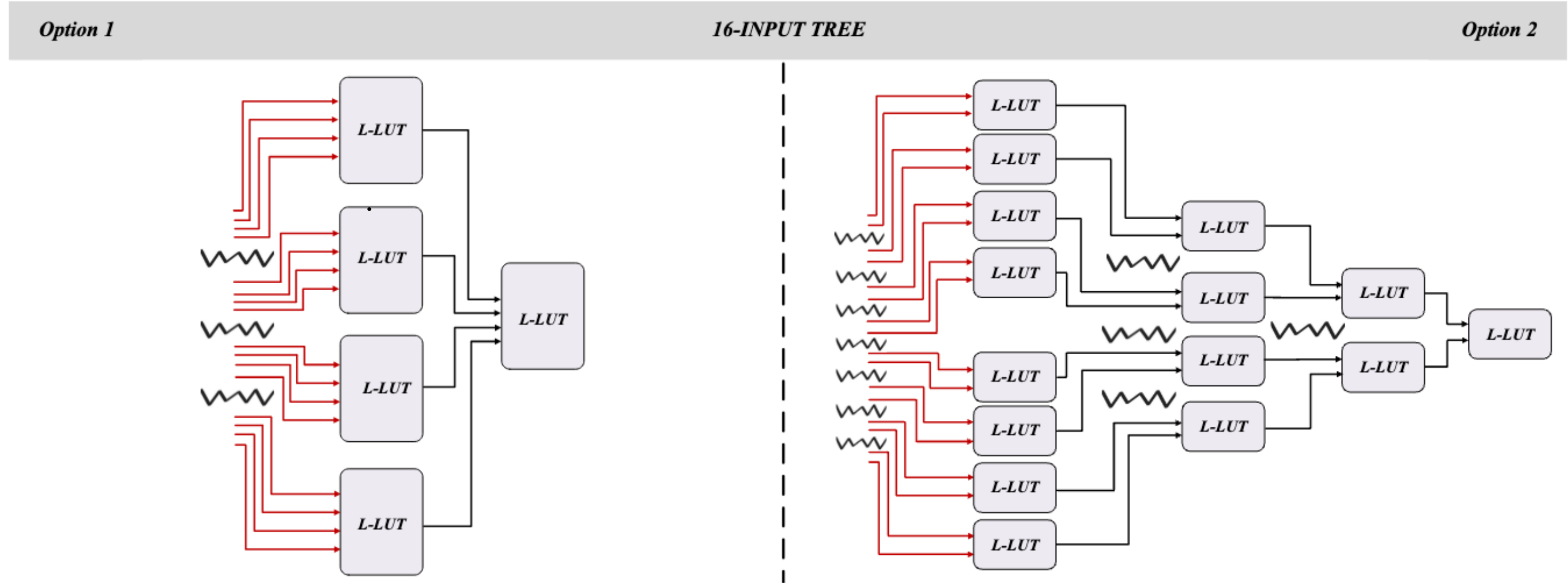
Inference-based lookup table generation.

### RTL Generation & Hardware Compilation

Verilog RTL automatically generated with L-LUTs as distributed ROMs.

Out-of-context synthesis and place & route.

# NeuraLUT-Assemble

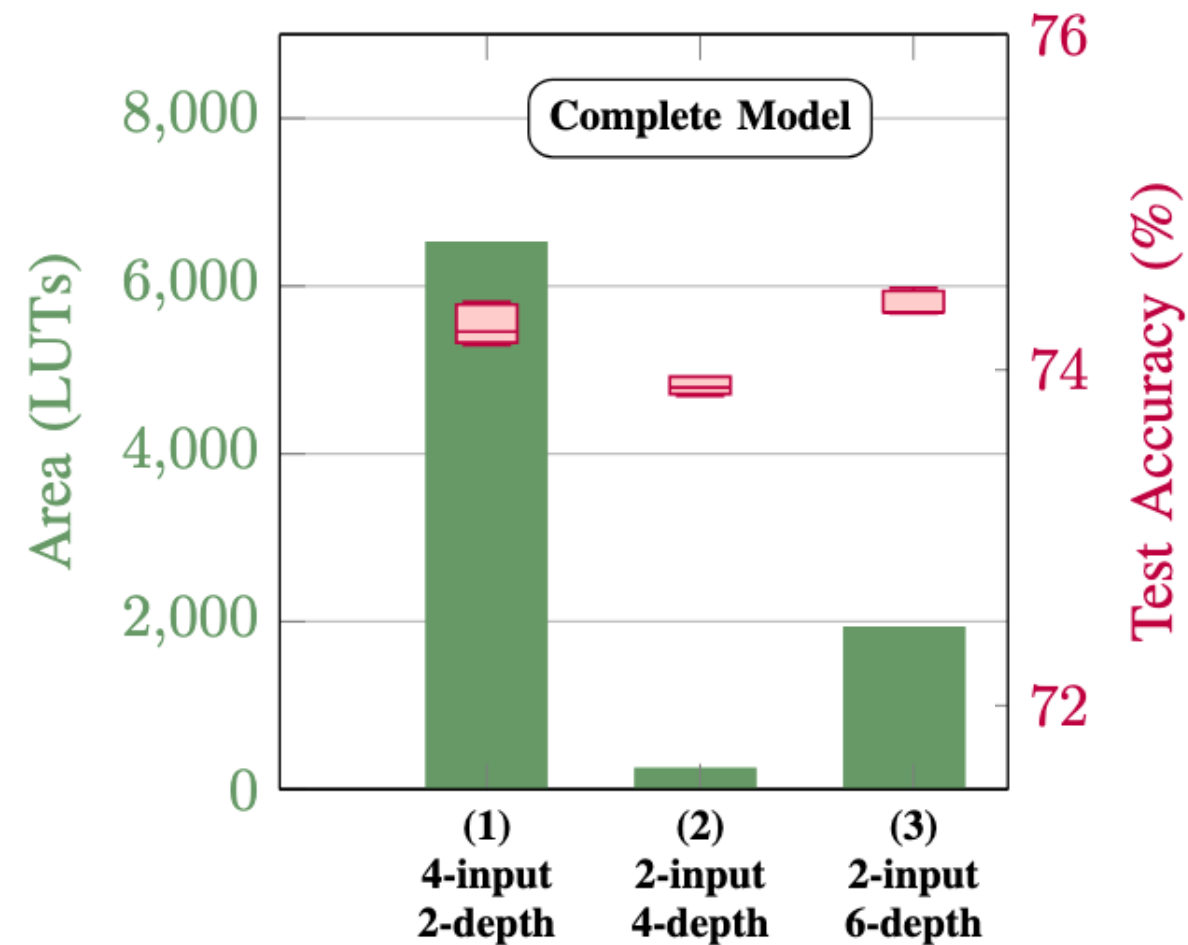


Our fully customizable framework allows users to construct tree structures tailored to their needs, balancing the trade-off between the number of L-LUTs and their size.

Two different NeuraLUT-Assemble configurations for a 16-input tree.

# NeuraLUT-Assemble

---

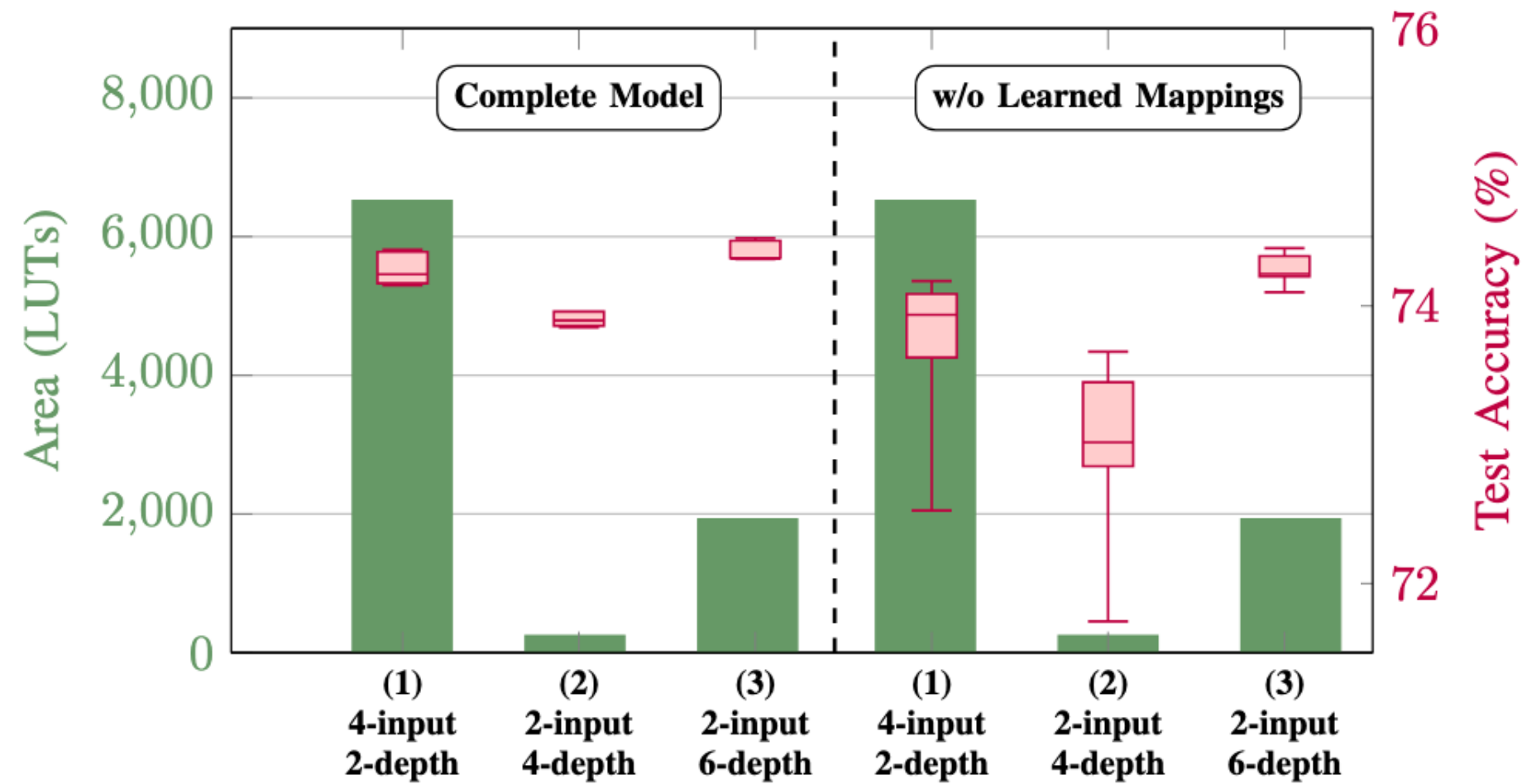


(1) to (2) reduces the area by 26× with less than a 0.5 pp drop in test accuracy.

(3) recovers the accuracy loss and even surpasses the accuracy of (1).

This demonstrates our methodology's ability to increase connectivity while achieving a significant reduction in area footprint.

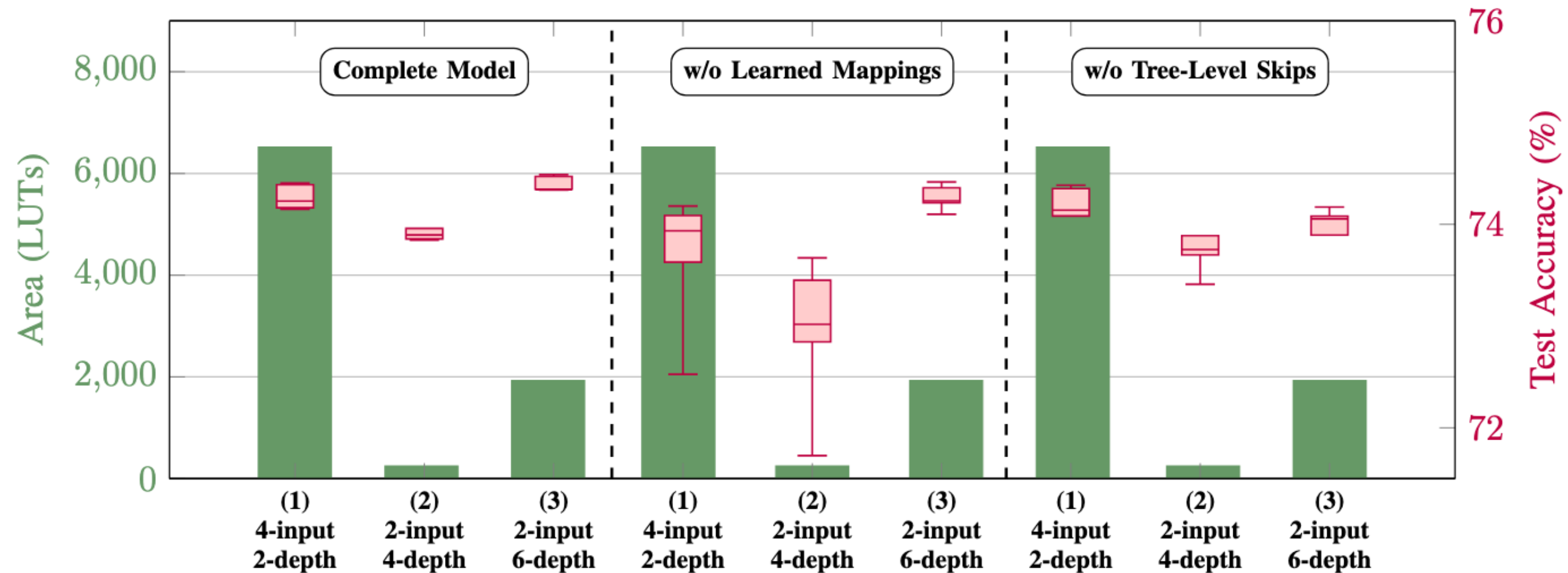
# NeuraLUT-Assemble



Without the learned mappings, the accuracy not only drops but it increases the result variability based on the random seed.

Data-driven grouping is essential for robust training.

# NeuraLUT-Assemble

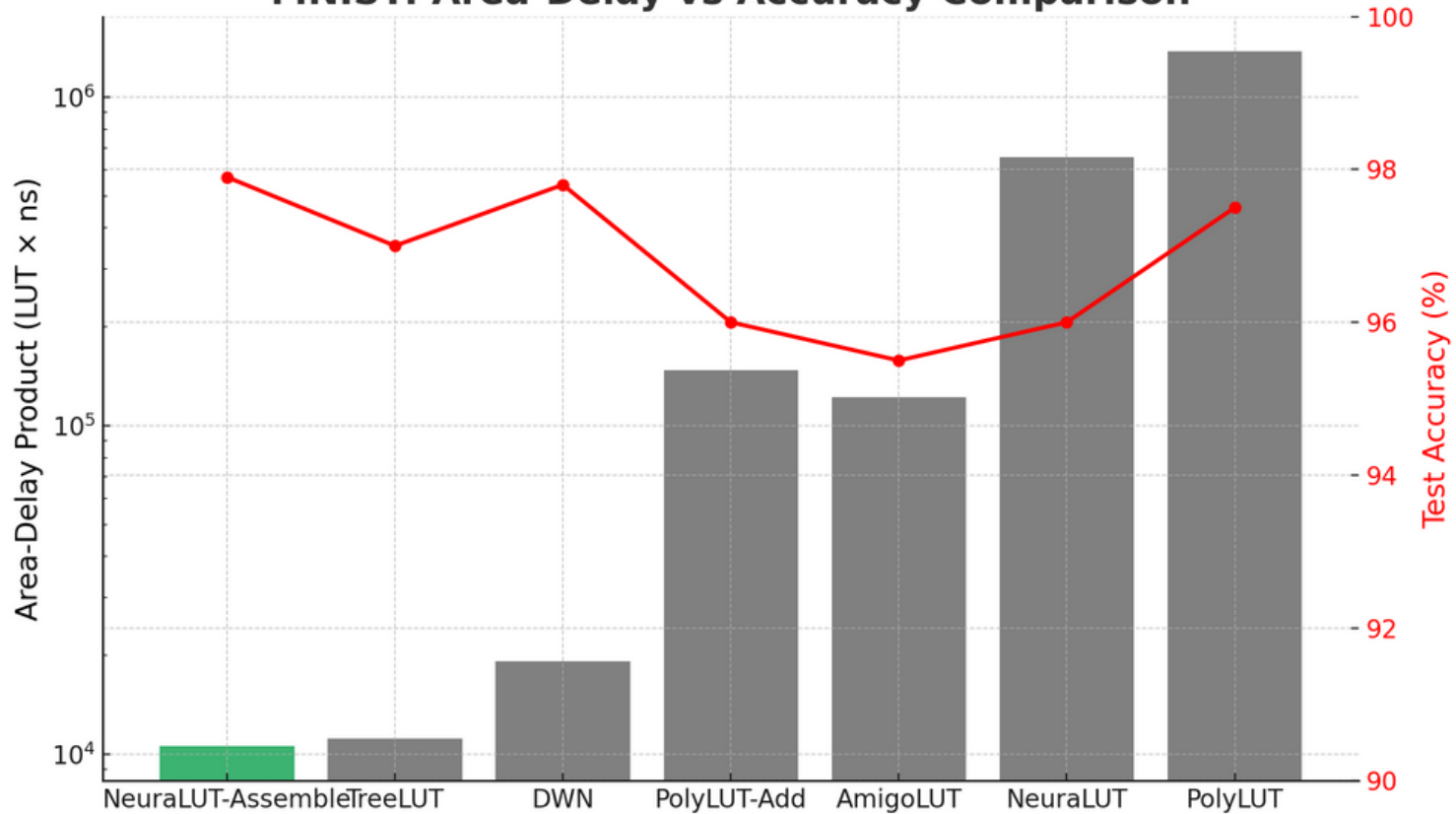


The accuracy drop caused by removing skip-connections increases with tree depth.

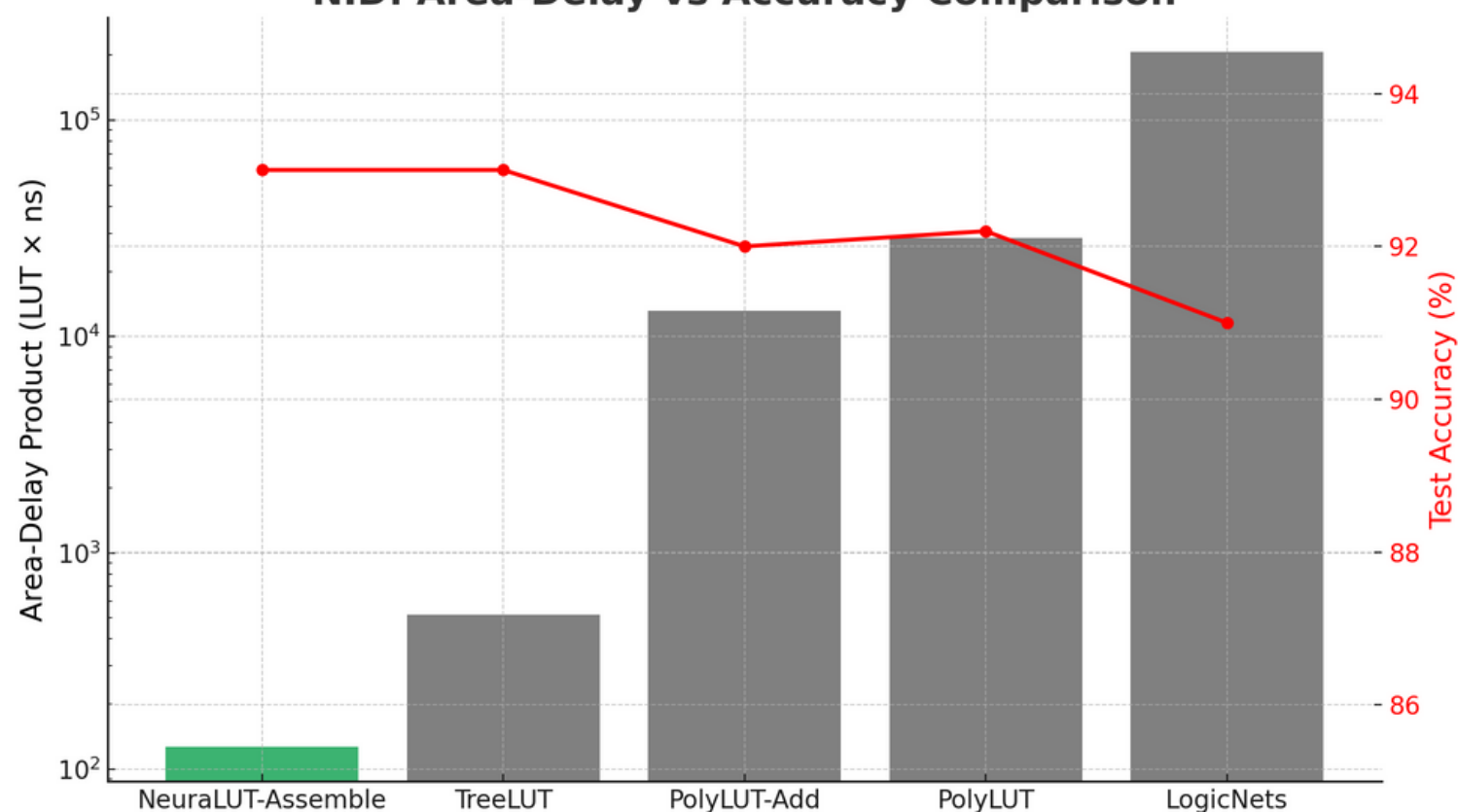
Tree-level skip connections are essential for gradient flow, especially in deeper trees.

# NeuraLUT-Assemble

**MNIST: Area-Delay vs Accuracy Comparison**



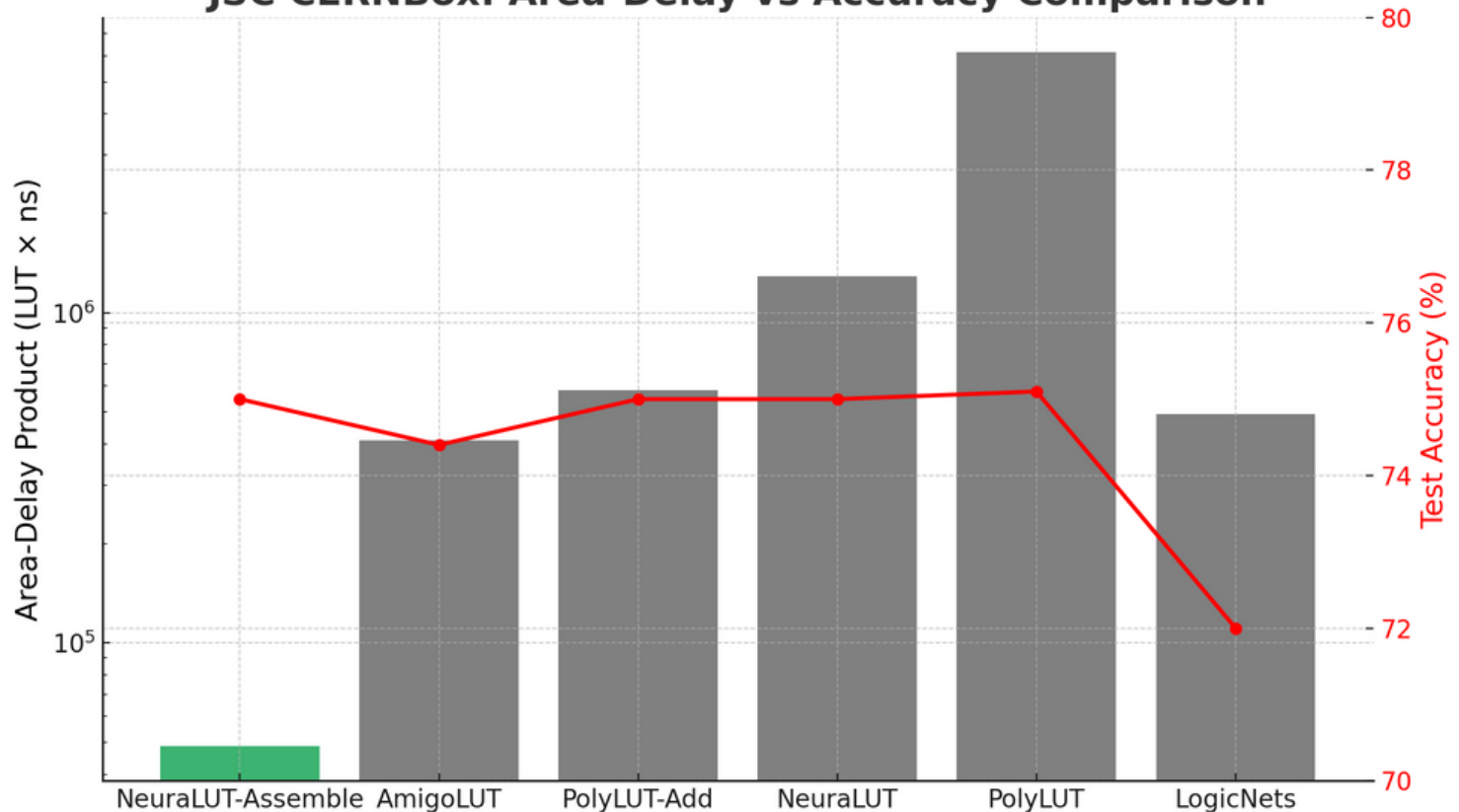
**NID: Area-Delay vs Accuracy Comparison**



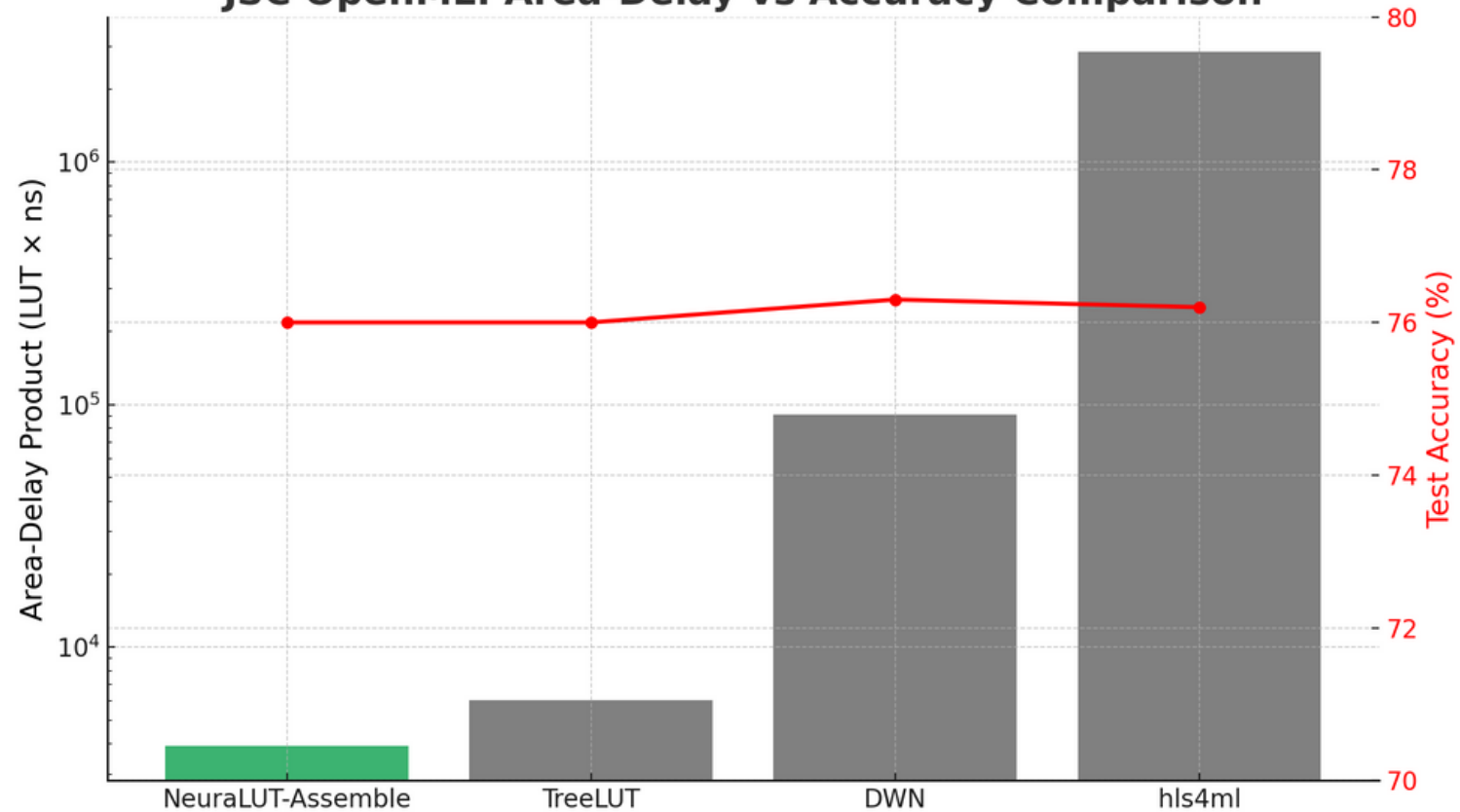
Compared to **NeuraLUT**:

- **62×** reduction in area-delay product on MNIST

**JSC CERNBox: Area-Delay vs Accuracy Comparison**



**JSC OpenML: Area-Delay vs Accuracy Comparison**



Compared to **SOTA**:

- up to **8.42×** reduction in the area-delay product

# NeuraLUT-Assemble

Dataset	Pipelining every L-LUT layer				Pipelining every 3 L-LUT layers			
	Latency (ns)	F <sub>max</sub> (MHz)	LUTs	FFs	Latency (ns)	F <sub>max</sub> (MHz)	LUTs	FFs
MNIST(+aug/-aug)	6.5	916	5040	5464	2.2	849	5037	713
	6.6	912	5089	5699	2.1	863	5070	725
JSC CERNBox	7.0	994	8535	2717	5.7	352	8539	1332
JSC OpenML	6.6	1067	1844	1983	2.1	941	1780	540
NID	3.4	1479	95	187	1.4	1471	91	24

Throughput-optimized

Latency-optimized

The user can choose to prioritize latency by adding a register after every three L-LUT layers or opt for a throughput-optimized design by placing a register after each L-LUT layer.

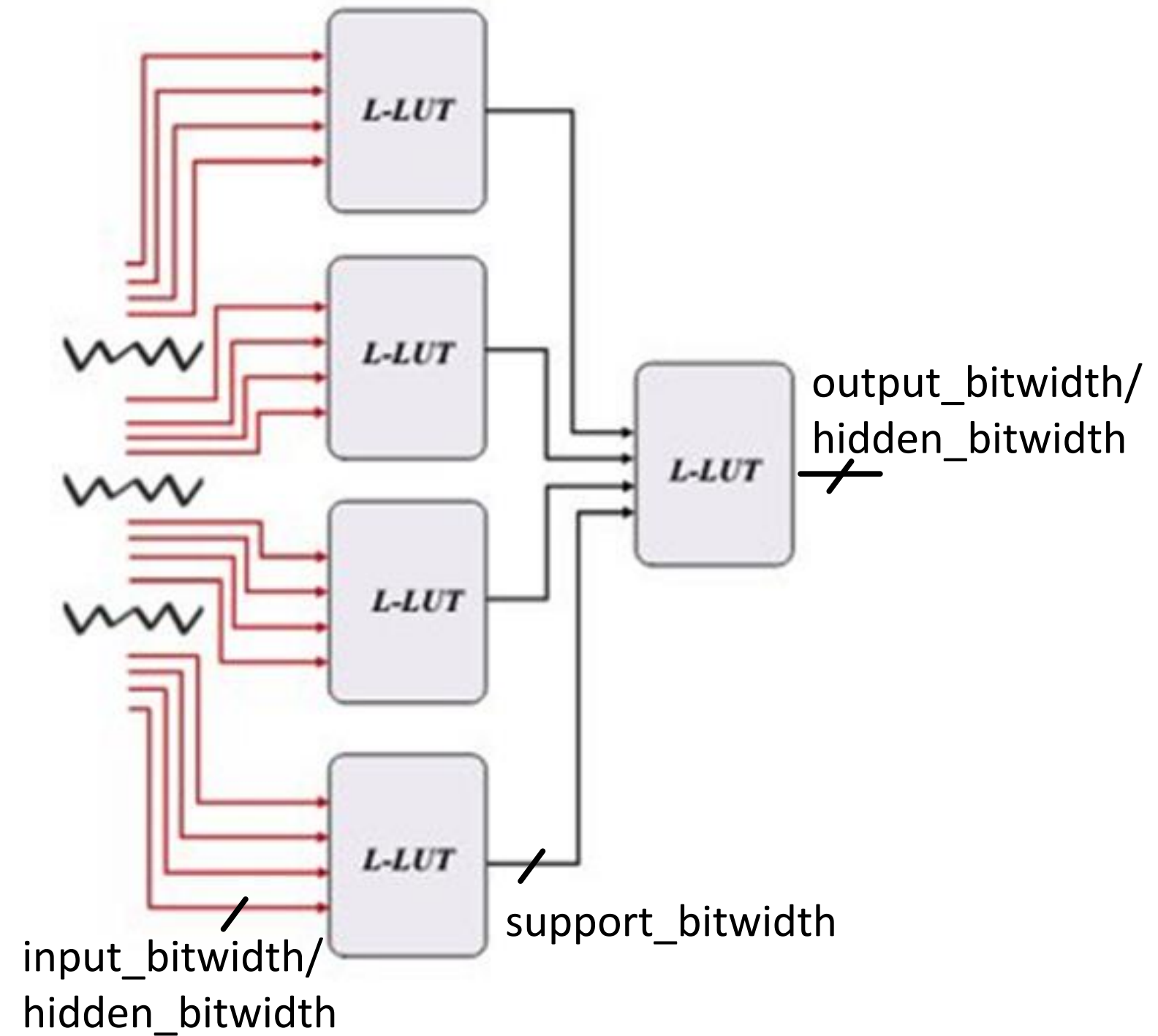
# NeuraLUT-Assemble Demo



# NeuraLUT-Assemble parameters

---

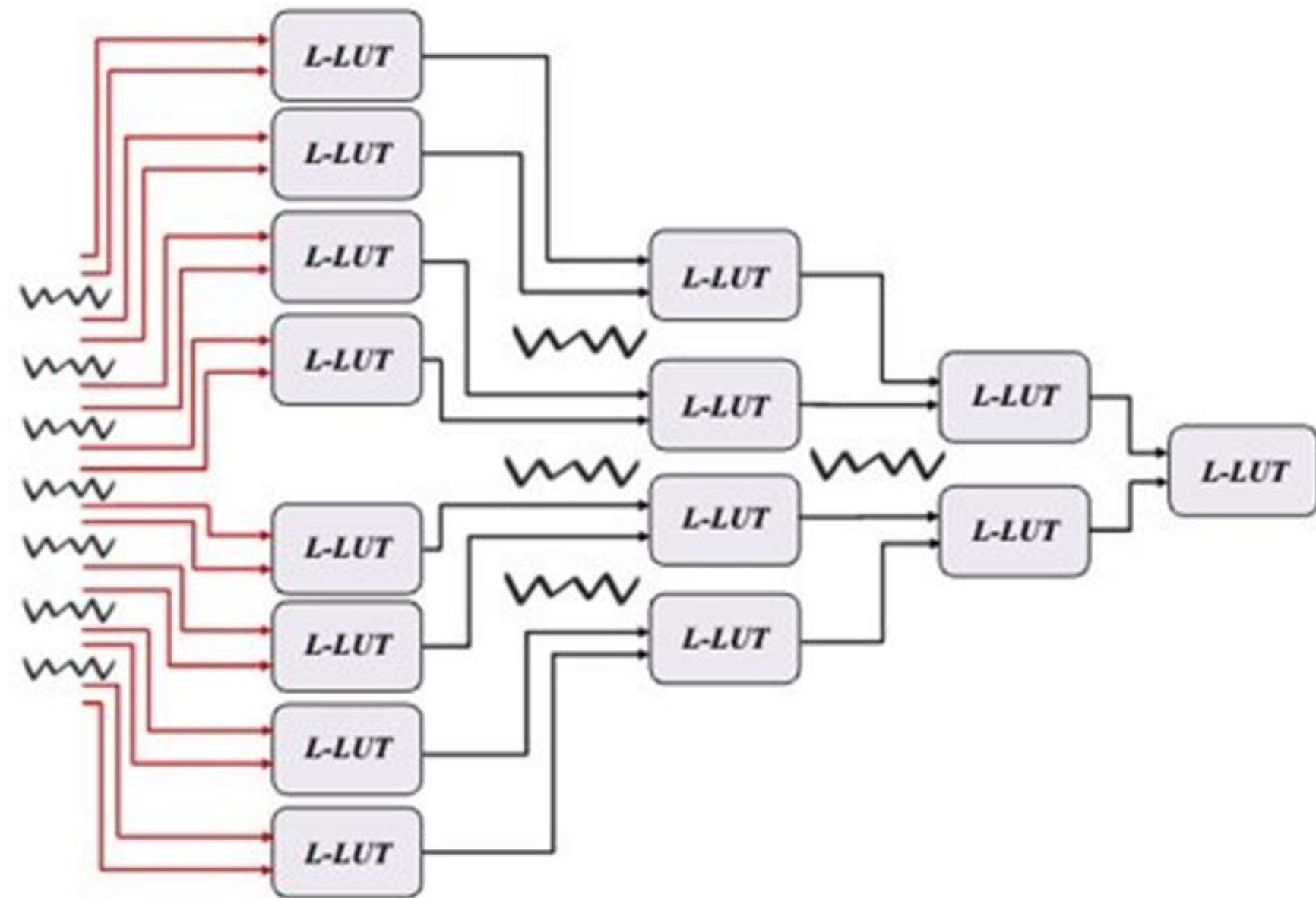
"hidden\_layers": [4,1],  
"support\_layers": [0,1],  
"input\_bitwidth": 2,  
"hidden\_bitwidth": 2,  
"output\_bitwidth": 2,  
"support\_bitwidth": 1,  
"input\_fanin": 4  
"hidden\_fanin": 4  
"support\_fanin": 4



# NeuraLUT-Assemble parameters

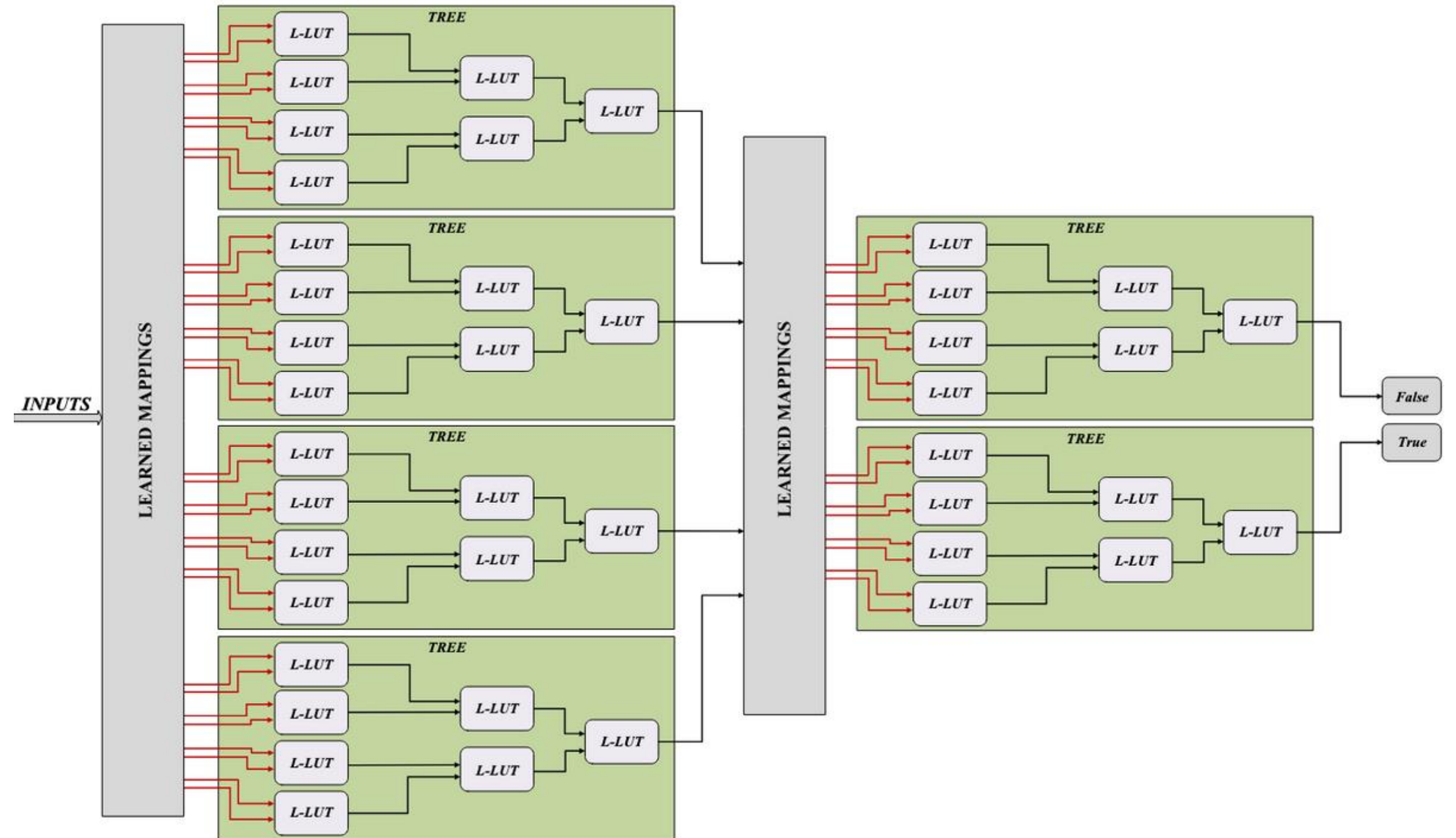
---

"hidden\_layers": [8,4,2,1],  
"support\_layers": [0,1,1,1],  
"input\_bitwidth": 2,  
"hidden\_bitwidth": 2,  
"output\_bitwidth": 2,  
"support\_bitwidth": 1,  
"input\_fanin": 2  
"hidden\_fanin": 2  
"support\_fanin": 2



# NeuraLUT-Assemble parameters

"hidden\_layers": [16,8,4,8,4,2,1],  
"support\_layers": [0,1,1,0,1,1,1],  
"input\_bitwidth": 2,  
"hidden\_bitwidth": 2,  
"output\_bitwidth": 2,  
"support\_bitwidth": 1,  
"input\_fanin": 2  
"hidden\_fanin": 2  
"support\_fanin": 2



# NeuraLUT-Assemble

---

Table 1: Implementation results for efficient inference models on a Z-7045 FPGA with input buffers and decompression implemented, with data being passed to the board using its I/O pins, which have a bandwidth of 112 bits per cycle, and the clock is limited to 200 MHz for a fair comparison to prior work.

Dataset	Model	Test Accuracy (%)	Latency (ns)	Throughput (Samples/s)	LUTs (1000s)
MNIST	FINN	98.40	2440	1.56M	83.0
	ULEEN	98.46	940	4.05M	123.1
	DiffLogicNet (xs)	96.87	90	33.3M	9.6
	DiffLogicNet (sm)*	97.62	95	33.3M	19.1
	DWN ( $n=2$ ; lg)	98.27	135	25.0M	10.3
	DWN ( $n=6$ ; sm)	97.80	60	50.0M	<b>2.1</b>
	DWN ( $n=6$ ; lg)	<b>98.31</b>	125	25.0M	4.6
	NeuraLUT-Assemble	97.87	<b>45</b>	33.3M	5.1

# NeuraLUT-Assemble

---

## NeuraLUT-Assemble Demo

```
from pynq import Overlay, MMIO
import numpy as np, struct, time
import matplotlib.pyplot as plt

overlay = Overlay('./bram_demo.bit')

print(overlay.ip_dict.keys())
```

```
.. dict_keys(['axi_gpio_0', 'processing_system7_0'])
```

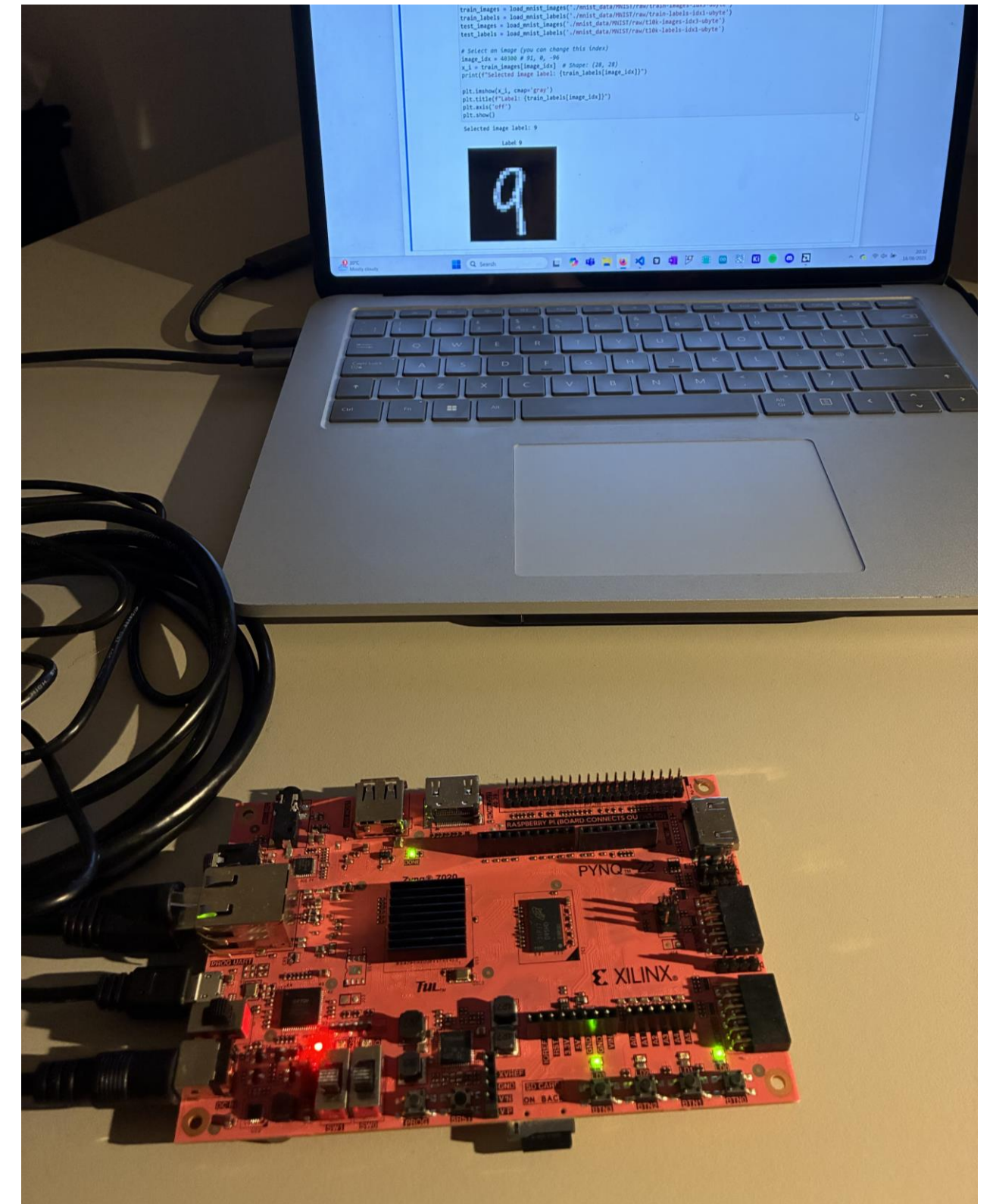
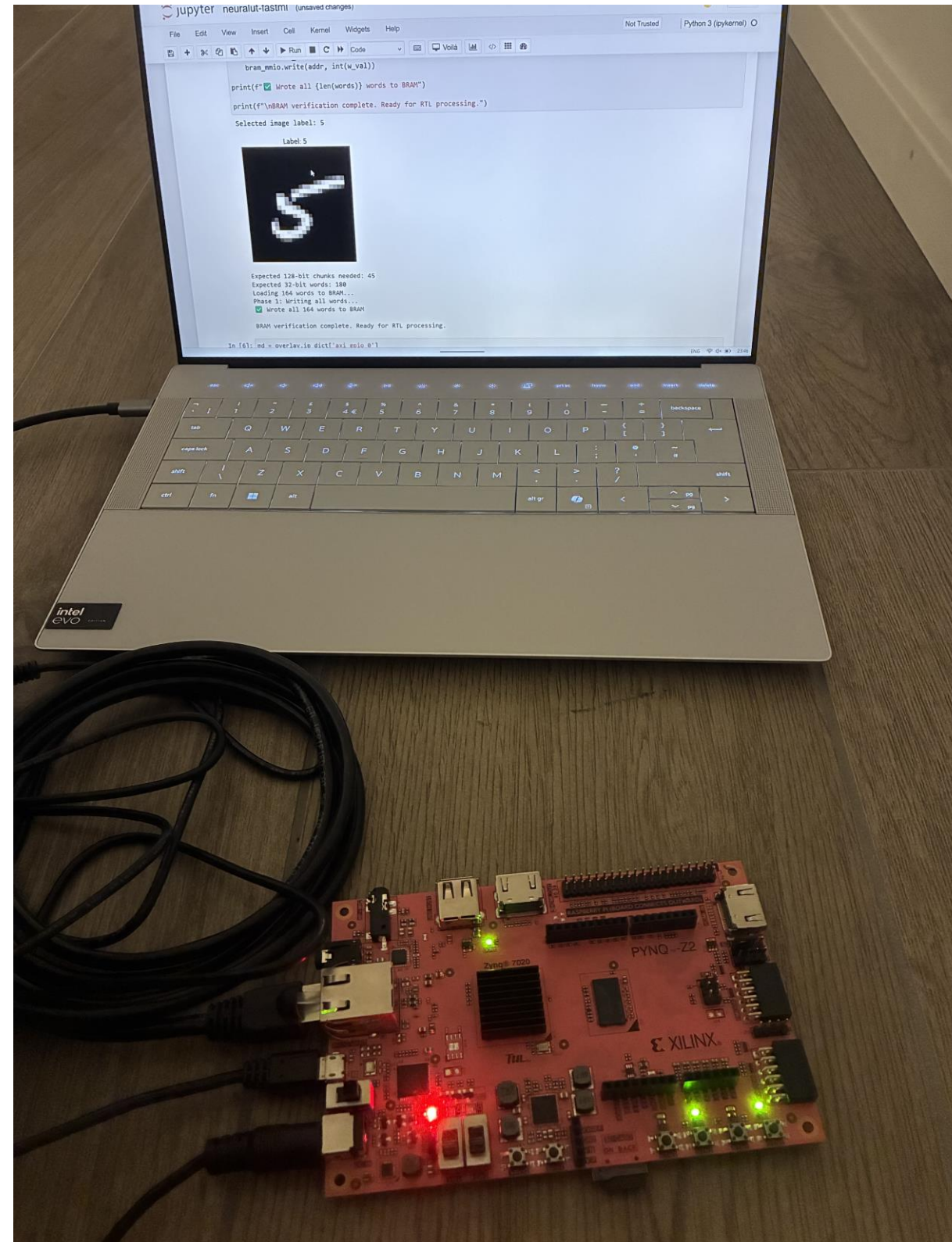
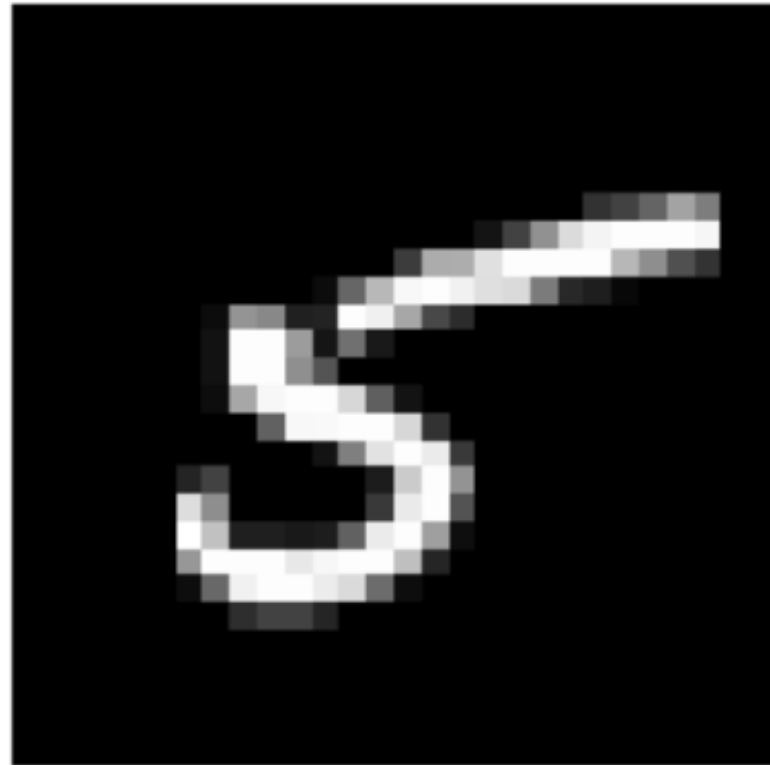
```
bram_ctrl = overlay.axi_bram_ctrl_0
bram_mmio = MMIO(bram_ctrl.mmio.base_addr, 0x2000)
```

```
feature_indices = [ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 17, 18, 19,
```

# NeuraLUT-Assemble

Selected image label: 5

Label: 5



# Open source repos

---



PolyLUT



NeuraLUT



AmigoLUT



NeuraLUT  
Assemble



ReducedLUT