Swordfish:

A Framework for Evaluating Deep Neural Network-based Basecalling using Computation-in-Memory with Non-Ideal Memristors

Taha Michael Shahroodi,
Gagandeep Singh, Mahdi Zahedi, Haiyu Mao,
Joel Lindegger, Can Firtina, Stephan Wong,
Onur Mutlu, Said Hamdioui

Quantum & Computer Engineering

SAFARI
SAFARI Research Group
safari.ethz.ch

TU Delft
Delft University of Technology

AMD

ETH zürich

Executive Summary

Context: Basecalling is **the first** step and **a major throughput bottleneck**
Basecallers use **deep neural networks (DNNs)**
DNN-based basecalling **accuracy** and **throughput** impact accuracy and throughput of next analysis
Prior research uses **memristor-based Computation-in-Memory (CIM)** to accelerate DNNs
Non-idealities in memristor-based CIM known to hinder **accuracy**

Problem: Prior frameworks for memristor-based CIM accelerators targeting large DNNs either

1. **overlook** existing **non-idealities**,
2. **overestimates** achievable **accuracy** by **studying non-idealities in isolation** or using **imprecise models/methodology**
3. **overlook** the effects of non-idealities mitigation techniques on the achievable throughput

Goal: Enable **accurate** and **realistic** evaluation of **accuracy** and **throughput** for DNN-based basecalling on memristor-based CIM

Key Contribution: Swordfish; the **first framework** for memristor-based CIM that uses **characterized memories** and **accurate models** to
1) accurately and **realistically** evaluate the effects of **non-idealities** on basecalling **accuracy** and **throughput**
2) comprehensively investigate the impact of **accuracy enhancement techniques** on basecalling **accuracy** and **throughput**

Key Results: Across four real datasets of varying sizes, Swordfish **realistically** provides

- **25.7× better average throughput** compared to state-of-the-art basecalling on GPU
- **12% mitigation in basecalling accuracy loss** after hardware/software co-designed enhancement techniques
- **Three** new **insights** on future research directions for **accuracy enhancement techniques**

Outline

Background & Motivation

Swordfish: Design & Implementation

Evaluation & Key Results

Takeaways & Summary

Outline

Background & Motivation

Swordfish: Design & Implementation

Evaluation & Key Results

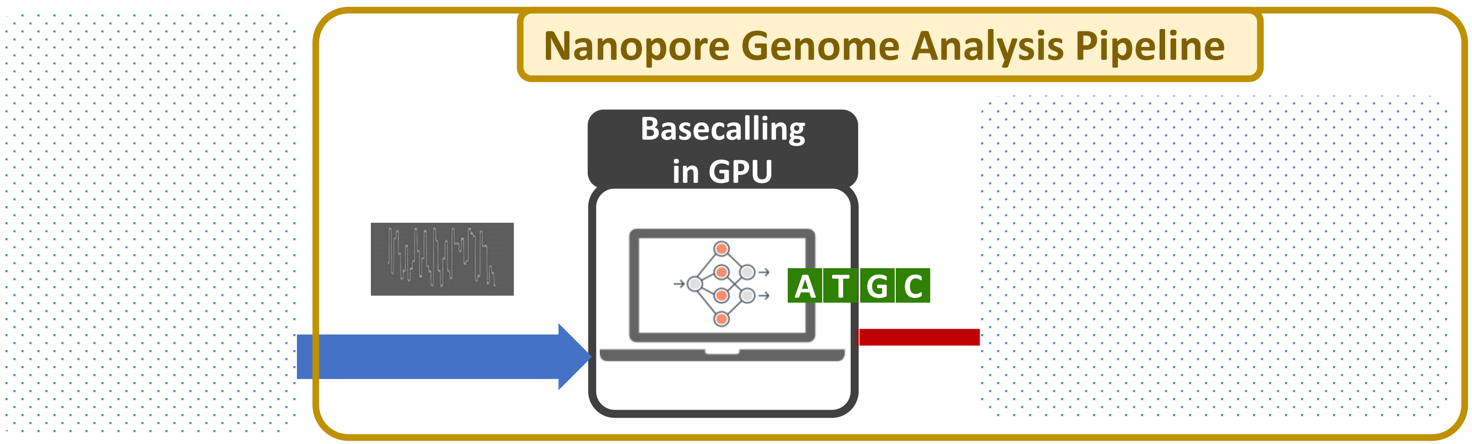
Takeaways & Summary

Nanopore Genome Sequencing and Analysis Pipeline

Genome Sequencing: Determining DNA sequence order for

1. Personalized medicine,
2. Outbreak tracing,
3. Understanding evolution

Nanopore Sequencing: A widely used sequencing technology



Basecalling consumes **up to 84.2%** of the execution time [Bowden+ 2019]

Nanopore Genome Sequencing and Analysis Pipeline

Genome Sequencing: Determining DNA sequence order for

1. Personalized medicine,
2. Outbreak tracing,
3. Understanding evolution

Nanopore Sequencing: A widely used sequencing technology

Basecalling is

1. **Accuracy-critical**
2. **Performance Bottleneck**

Basecallers are just large DNNs

DNN Hardware Acceleration

DNN execution is dominated by:

Vector-Matrix Multiplication (VMM)

Memristor-based crossbars support VMM

Data movement between memory and accelerator (e.g., GPU or TPU)

Computation in Memory (CIM) minimizes data movement

DNN Hardware Acceleration

DNN execution is dominated by:

Vector-Matrix Multiplication (VMM)

Data movement between memory and accelerator

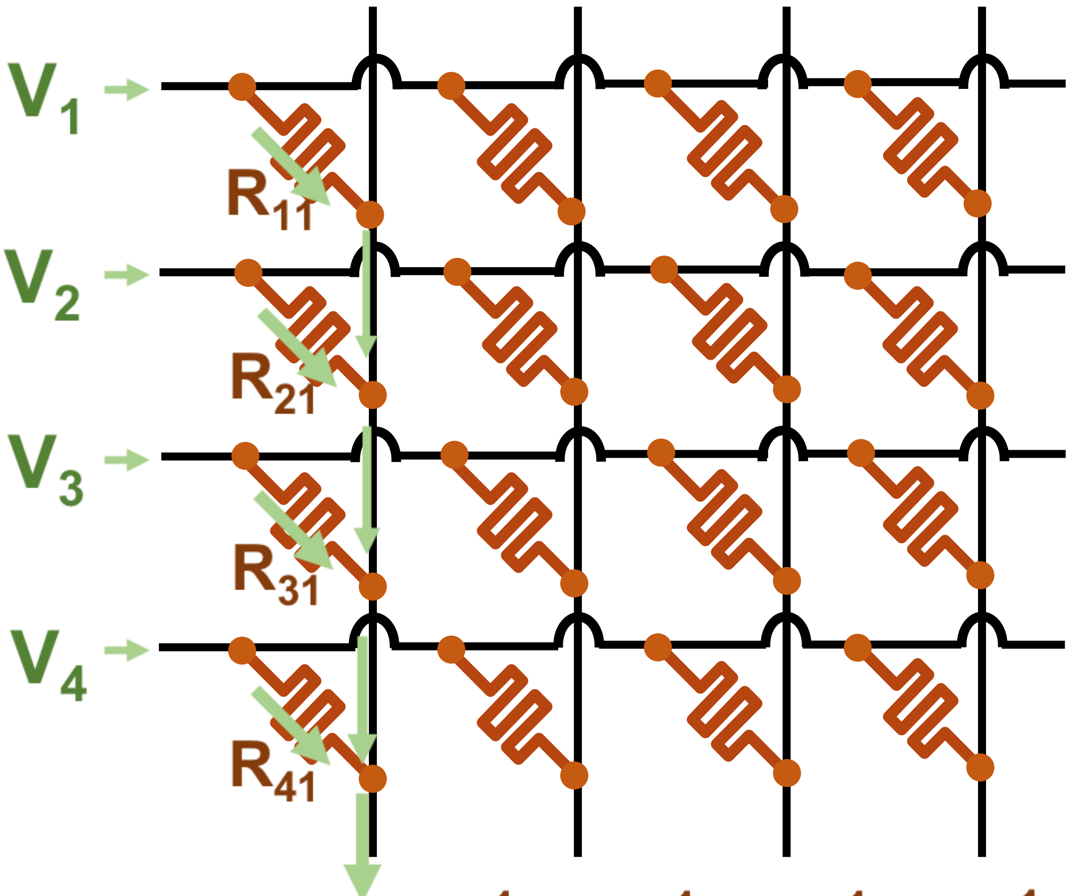
Memristor-based CIM for DNN Acceleration

Memristor-based crossbars support VMM

Computation in Memory (CIM) minimizes data movement

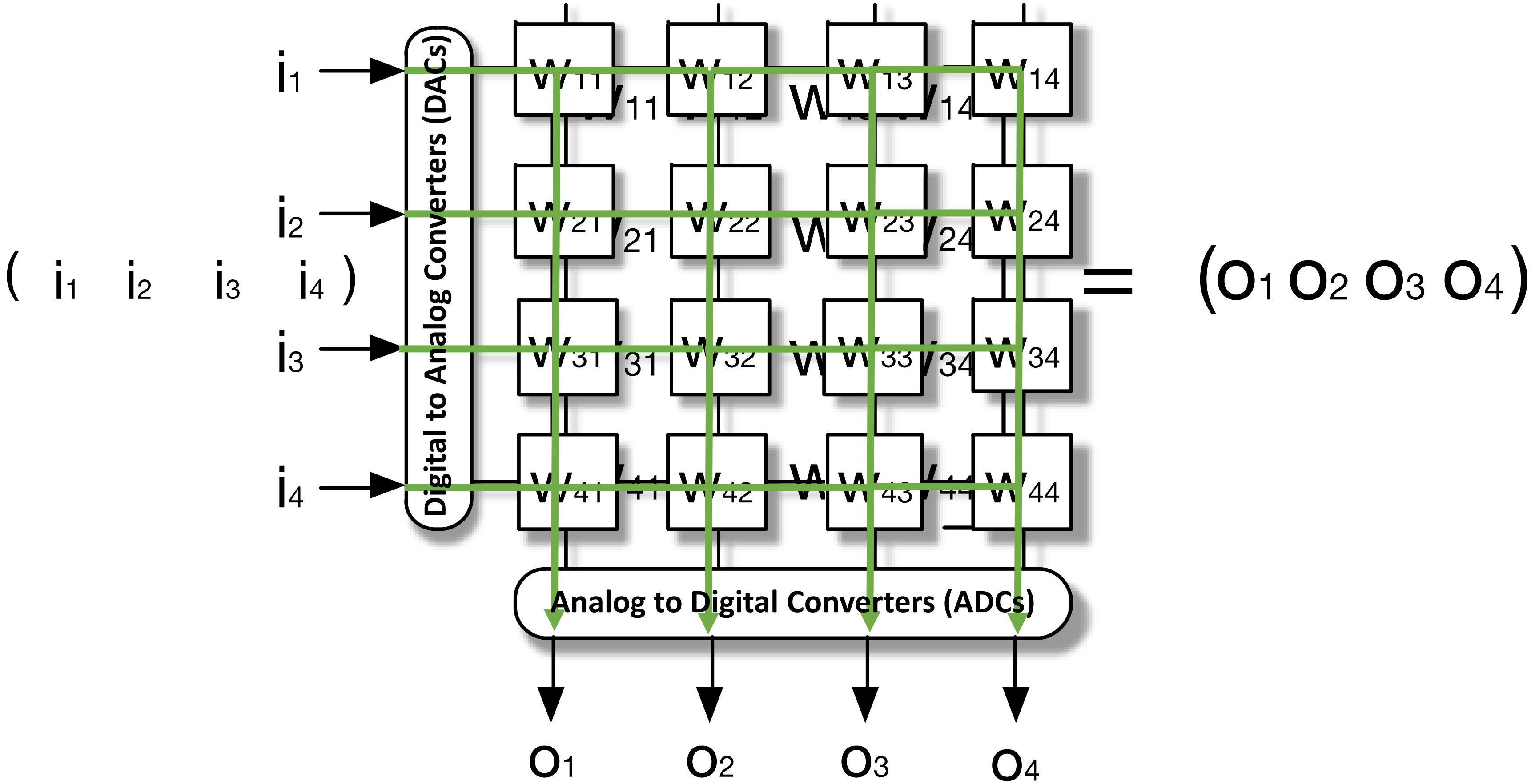
[Ankit+, ASPLOS 2019], [Chi+, ISCA 2016], [Lou+, PACT2020], [Shafiee+, ISCA 2016]

Memristor-based Crossbars



$$I_1 = \frac{1}{R_{11}}V_1 + \frac{1}{R_{21}}V_2 + \frac{1}{R_{31}}V_3 + \frac{1}{R_{41}}V_4$$

VMM in Memristor-based Crossbars



VMM in Memristor-based Crossbars

VMM in Accelerators

In Accelerators

Accurate

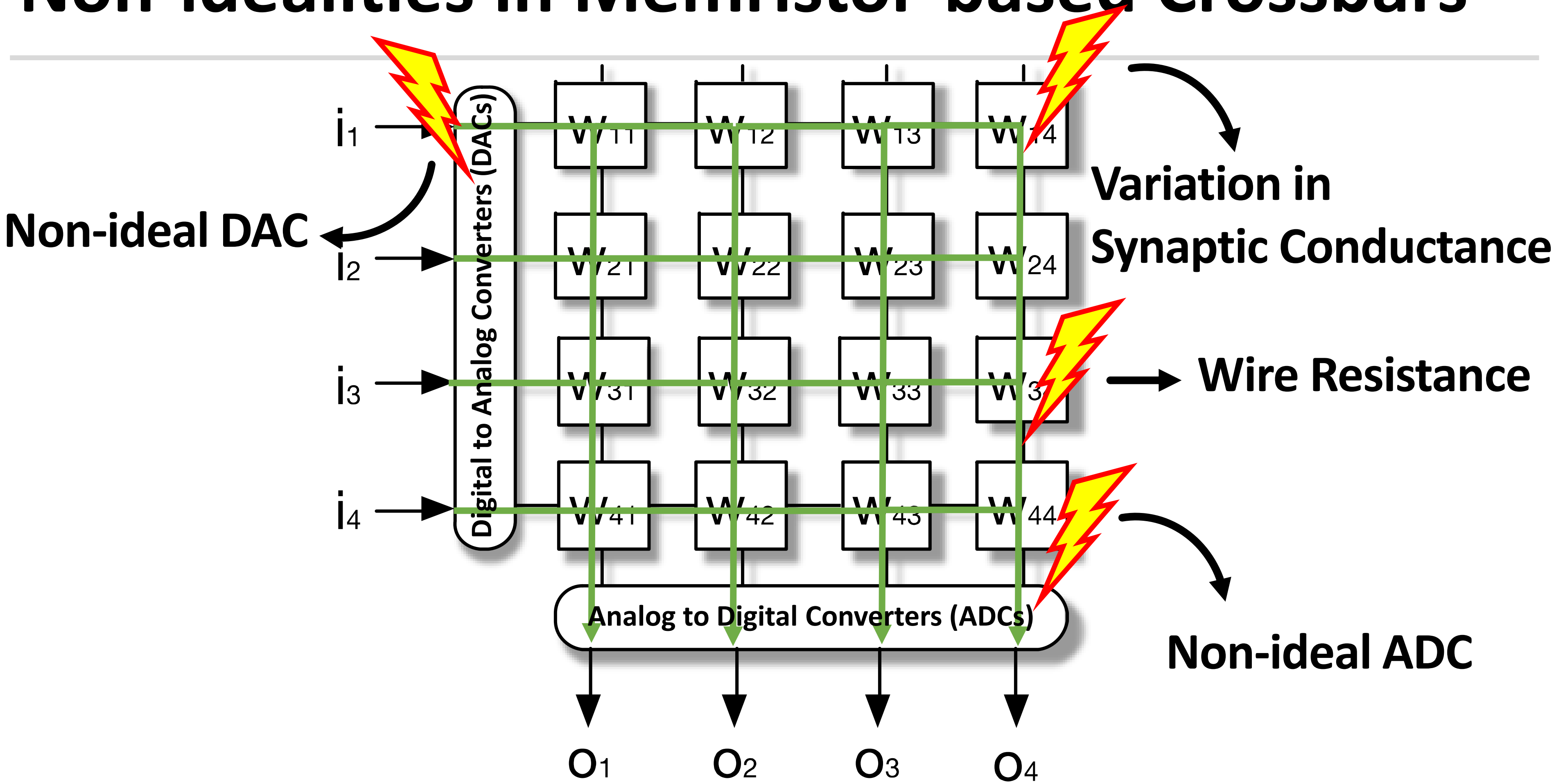
$$\begin{pmatrix} i_1 & i_2 & i_3 & i_4 \end{pmatrix} \times \begin{bmatrix} W_{11} & W_{12} & W_{13} & W_{14} \\ W_{21} & W_{22} & W_{23} & W_{24} \\ W_{31} & W_{32} & W_{33} & W_{34} \\ W_{41} & W_{42} & W_{43} & W_{44} \end{bmatrix} = \begin{pmatrix} O_1 & O_2 & O_3 & O_4 \end{pmatrix}$$

VMM in Memristor-based Crossbars

In Memory

$$\begin{pmatrix} i_1 & i_2 & i_3 & i_4 \end{pmatrix} \times \begin{bmatrix} W_{11} & W_{12} & W_{13} & W_{14} \\ W_{21} & W_{22} & W_{23} & W_{24} \\ W_{31} & W_{32} & W_{33} & W_{34} \\ W_{41} & W_{42} & W_{43} & W_{44} \end{bmatrix} = \begin{pmatrix} O_1 & O_2 & O_3 & O_4 \end{pmatrix}$$

Non-idealities in Memristor-based Crossbars



Non-idealities are **everywhere**

VMM in Memristor-based Crossbars

VMM in Memristor-based Crossbars

In Memory

$$\begin{pmatrix} i_1 & i_2 & i_3 & i_4 \end{pmatrix} \times \begin{bmatrix} W_{11} & W_{12} & W_{13} & W_{14} \\ W_{21} & W_{22} & W_{23} & W_{24} \\ W_{31} & W_{32} & W_{33} & W_{34} \\ W_{41} & W_{42} & W_{43} & W_{44} \end{bmatrix} = \begin{pmatrix} O_1 & O_2 & O_3 & O_4 \end{pmatrix}$$

VMM in Memristor-based Crossbars

VMM in Ideal Memristor-based Crossbars

In Memory

$$\begin{pmatrix} i_1 & i_2 & i_3 & i_4 \end{pmatrix} \times \begin{bmatrix} W_{11} & W_{12} & W_{13} & W_{14} \\ W_{21} & W_{22} & W_{23} & W_{24} \\ W_{31} & W_{32} & W_{33} & W_{34} \\ W_{41} & W_{42} & W_{43} & W_{44} \end{bmatrix} = \begin{pmatrix} O_1 & O_2 & O_3 & O_4 \end{pmatrix}$$

Accurate

VMM in Memristor-based Crossbars

VMM in Ideal Memristor-based Crossbars

In Memory

$$\begin{pmatrix} i_1 & i_2 & i_3 & i_4 \end{pmatrix} \times \begin{bmatrix} W_{11} & W_{12} & W_{13} & W_{14} \\ W_{21} & W_{22} & W_{23} & W_{24} \\ W_{31} & W_{32} & W_{33} & W_{34} \\ W_{41} & W_{42} & W_{43} & W_{44} \end{bmatrix} = \begin{pmatrix} O_1 & O_2 & O_3 & O_4 \end{pmatrix}$$

Accurate

VMM in Real Memristor-based Crossbars

Memory

$$\begin{pmatrix} i_1 & i_2 & i_3 & i_4 \end{pmatrix} \times \begin{bmatrix} W_{11} & W_{12} & W_{13} & W_{14} \\ W_{21} & W_{22} & W_{23} & W_{24} \\ W_{31} & W_{32} & W_{33} & W_{34} \\ W_{41} & W_{42} & W_{43} & W_{44} \end{bmatrix} = \begin{pmatrix} O_1 & O_2 & O_3 & O_4 \end{pmatrix}$$

Inaccurate

Our Goal

To **realistically evaluate** end-to-end basecalling **accuracy** and **throughput** for memristor-based CIM

Key Idea

To account for the **non-idealities** in **device**, **circuit** and **architecture** of memristor-based CIM and the **overhead** of non-idealities **mitigation techniques**

Outline

Background & Motivation

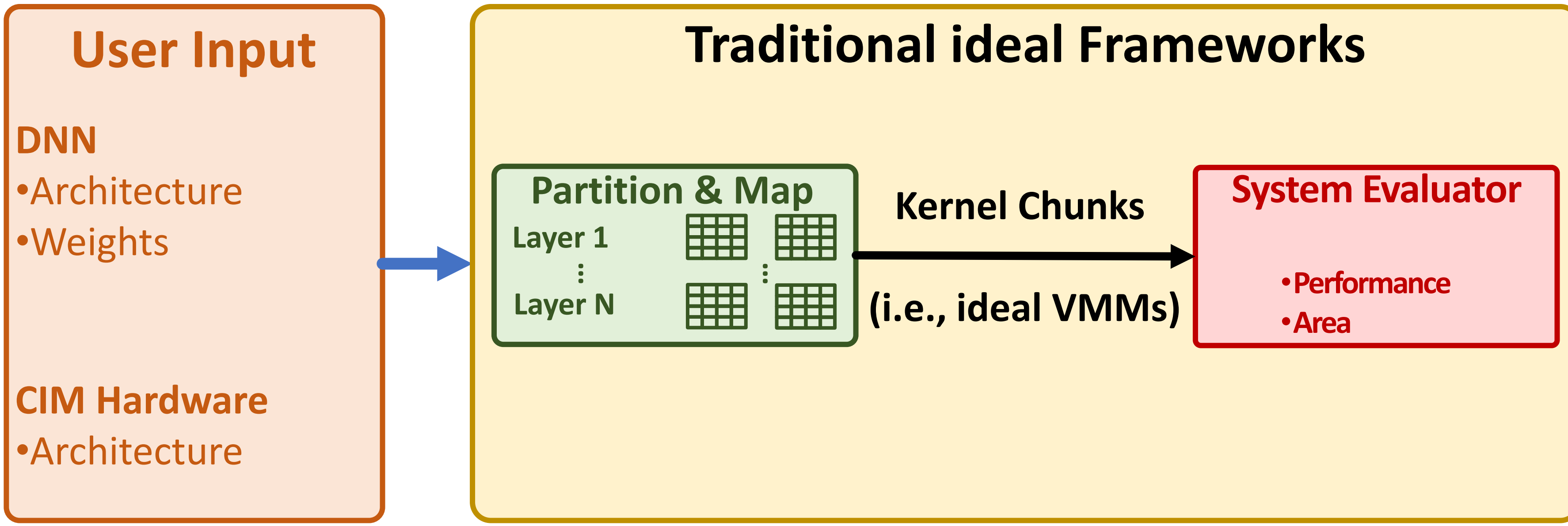
Swordfish: Design & Implementation

Evaluation & Key Results

Takeaways & Summary

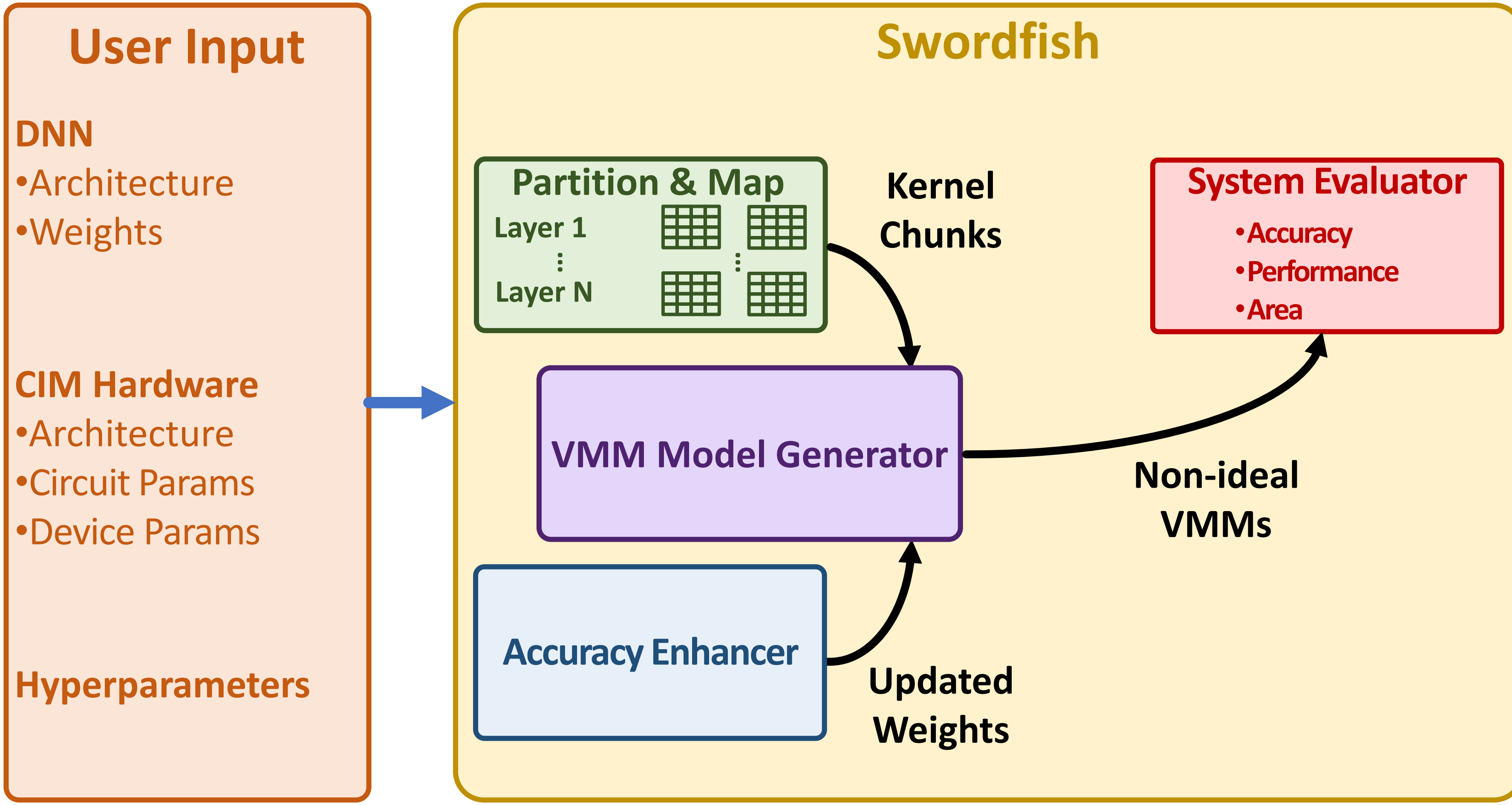
Swordfish vs Other Frameworks

Ideal Memristor-based CIM Frameworks for DNNs



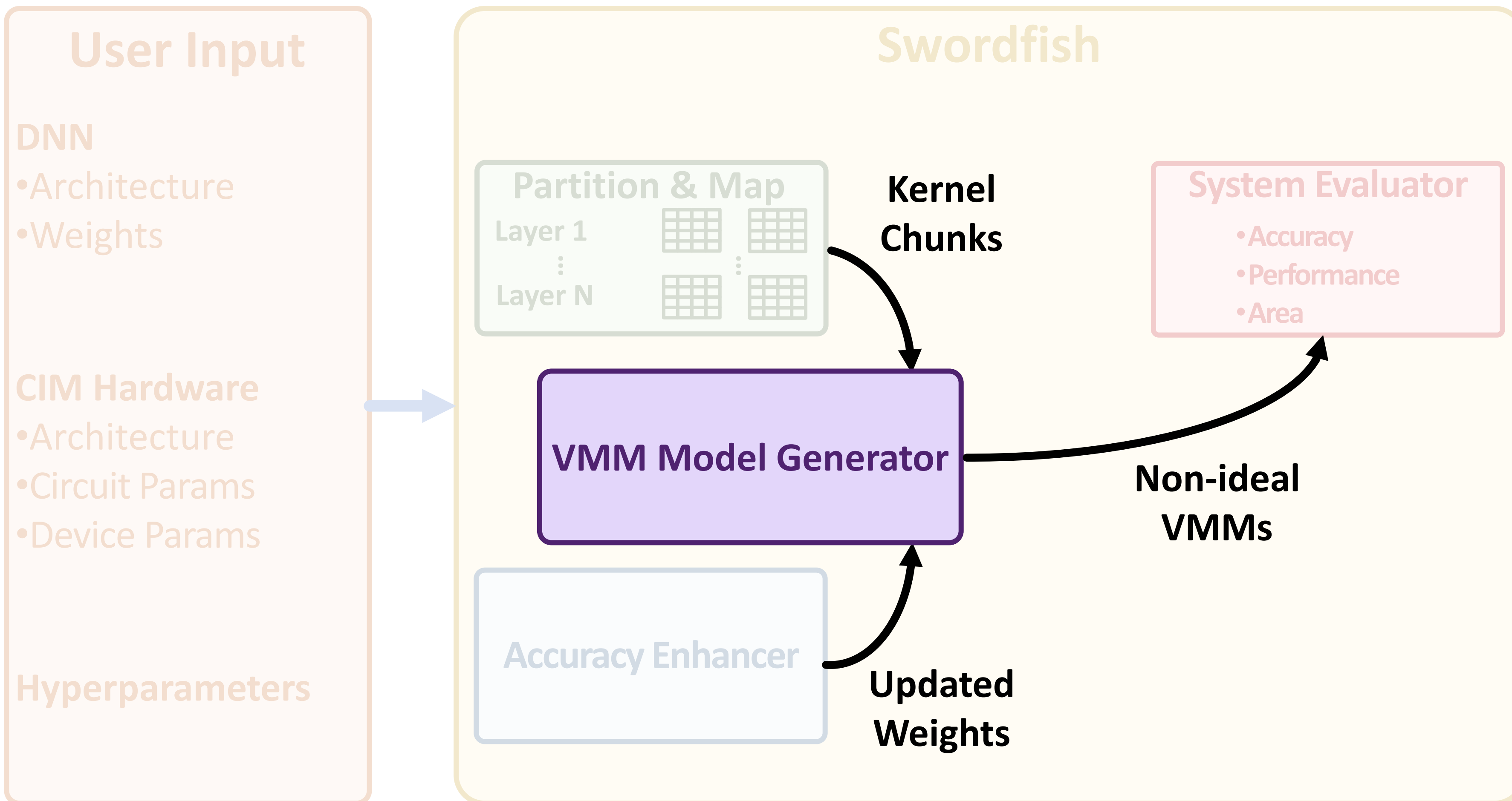
Swordfish Framework - Overview

Realistic Memristor-based CIM Frameworks for DNNs



Swordfish Framework - Overview

Realistic Memristor-based CIM Frameworks for DNNs



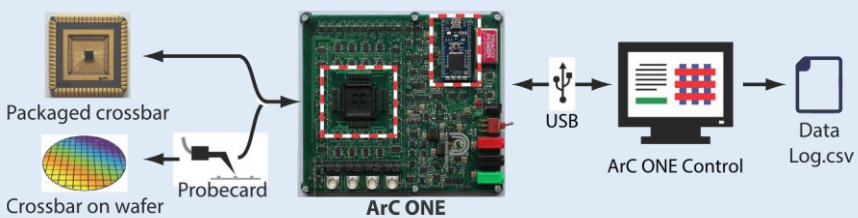
VMM Model Generator

Goal: Capture real output of VMM in presence of non-idealities

Swordfish supports two approaches:

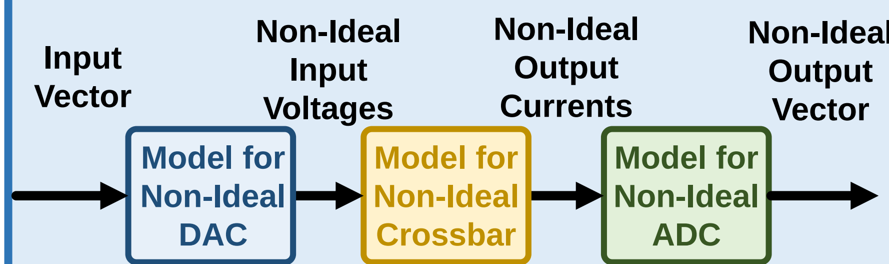
Approach 1:

Real chip **measurement** and **characterizations**



Approach 2:

Analytical Modeling using component's models



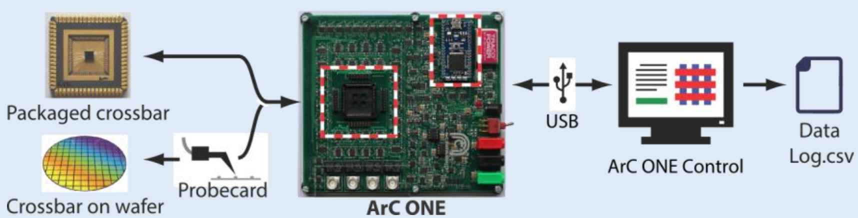
VMM Model Generator

Goal: Capture real output of VMM in presence of non-idealities

Swordfish supports two approaches:

Approach 1:

Real chip **measurement** and **characterizations**

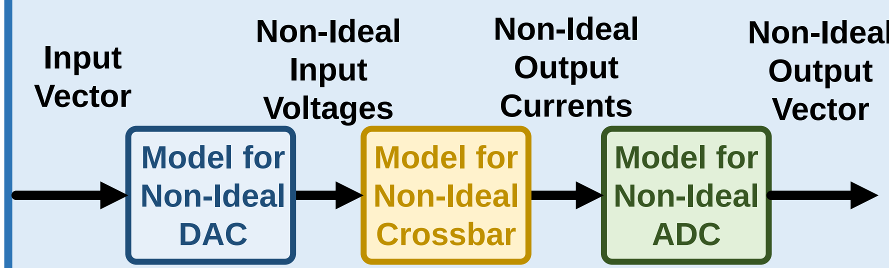


Most **Accurate**

Less **Flexible**

Approach 2:

Analytical Modeling using component's models

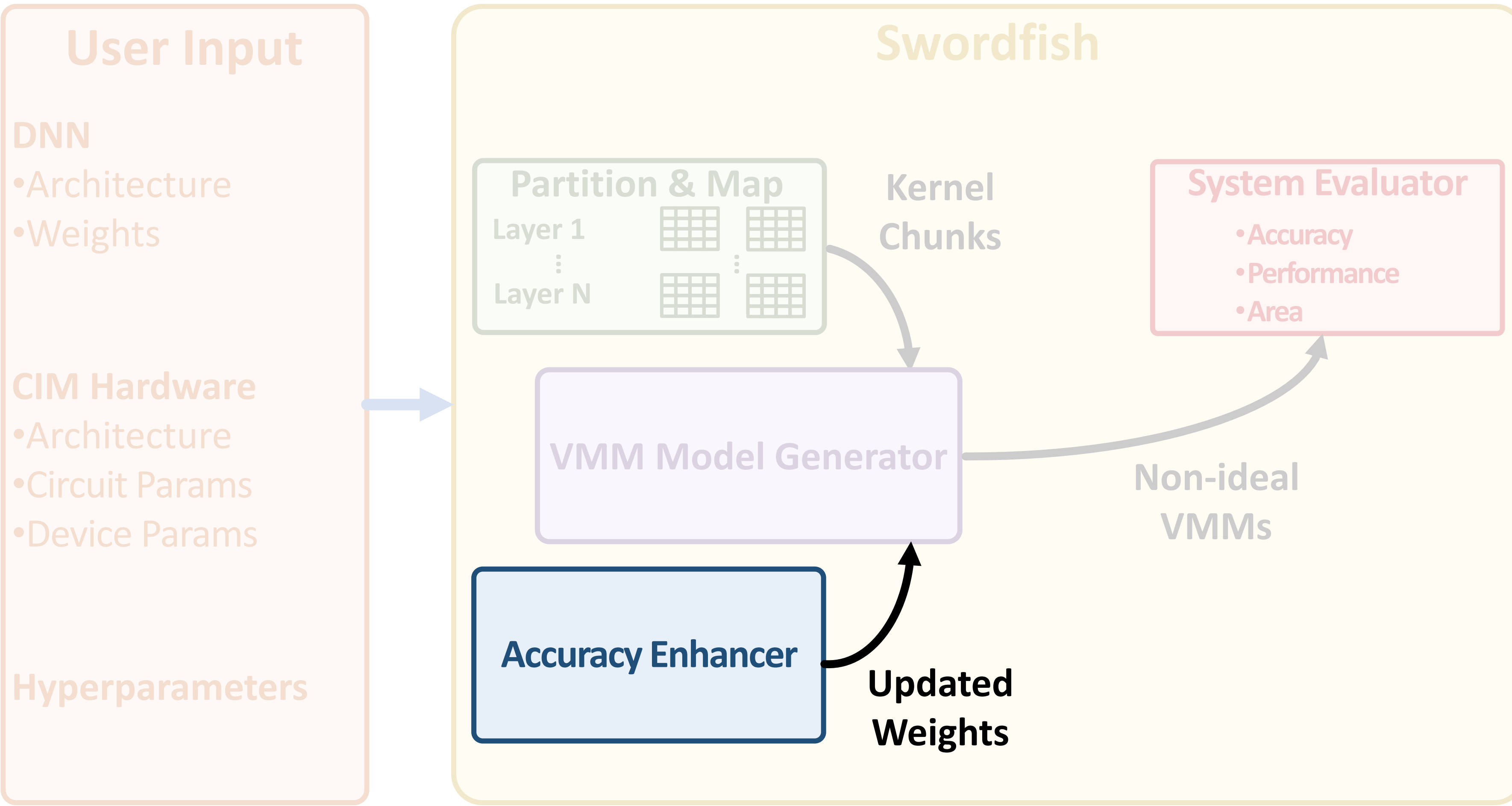


More **Flexible**

Less **Accurate**

Swordfish Framework - Overview

Realistic Memristor-based CIM Frameworks for DNNs



Accuracy Enhancement

Goal: Enhance the accuracy of a VMM by adapting input currents and resistance of memristors based on non-idealities

Swordfish supports four techniques:

1. Analytical Variation Aware Training (VAT)

2. Knowledge Distillation-based (KD) VAT

3. Read-Verify-Write (R-V-W) Training

4. Random Sparse Adaptation (RSA) Training

Example of Accuracy Enhancement

Goal: Enhance the accuracy of a VMM by adapting input currents and resistance of memristors based on non-idealities

Read more about other techniques in the paper

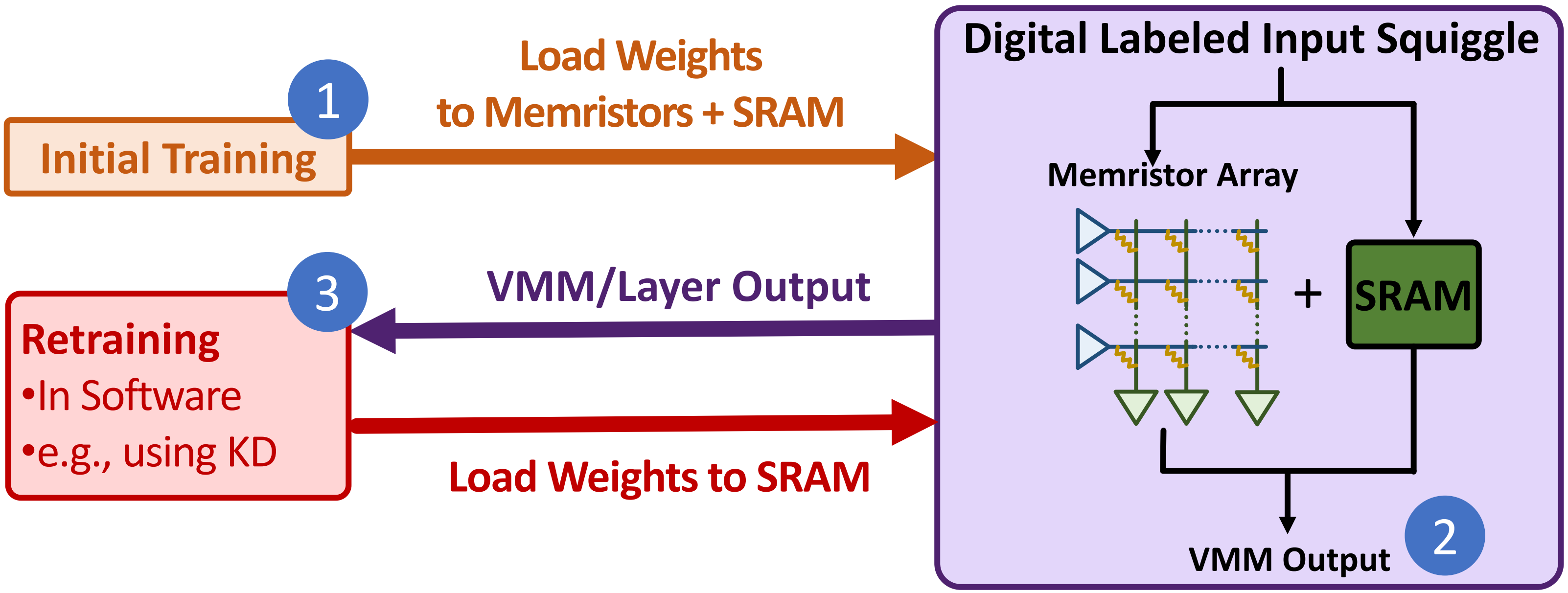
4. Random Sparse Adaptation (RSA) Training

Accuracy Enhancement via Random Sparse Adaptation

Key idea? Map the weights that otherwise would map to error-prone memristor devices to reliable SRAM cells.

RSA in 3 Steps:

1. Initial Training (one-time, on GPU) and distribution of weights
2. VMM operation using both memories
3. Retraining all weights and reload those on **SRAM (only)**



More in the Paper

- Details of **capturing non-idealities** at VMM level
- Implementation details of **Swordfish components**:
  - Partition & Map
  - Accuracy Enhancer
  - VMM Model Generator
  - System Evaluator
- Elaborations on **accuracy enhancement techniques**
  - Analytical Variation Aware Training (VAT)
  - Knowledge Distillation-based (KD) using VAT
  - Read-Verify-Write (R-V-W) Training
  - Random Sparse Adaptation (RSA) Training

Outline

Background & Motivation

Swordfish: Design & Implementation

Evaluation & Key Results

Takeaways & Summary

Evaluation Methodology: Experimental Setup

- **We evaluate**
  - **Basecaller:** Bonito [Oxford Nanopore 2023]
  - **CIM Architecture:** PUMA [Ankit+, ASPLOS 2019]

- **Infrastructure**
  - 2x AMD EPYC 7742 CPU with 500 GB DDR4 DRAM
  - 8x NVIDIA V100

- **Datasets and Workloads** [Wick+ 2019, Zook+ 2019, CADDE 2020]
  - 4 real read and reference genomes with various genome size (D1, D2, D3, and D4)

Evaluated Non-idealities & Enhancement techniques

- Non-idealities

1. Synaptic+Wires
2. Sensing+ADC Circuitry
3. DAC+Driver Circuitry
4. Combined

Analytical Models (Approach 2)

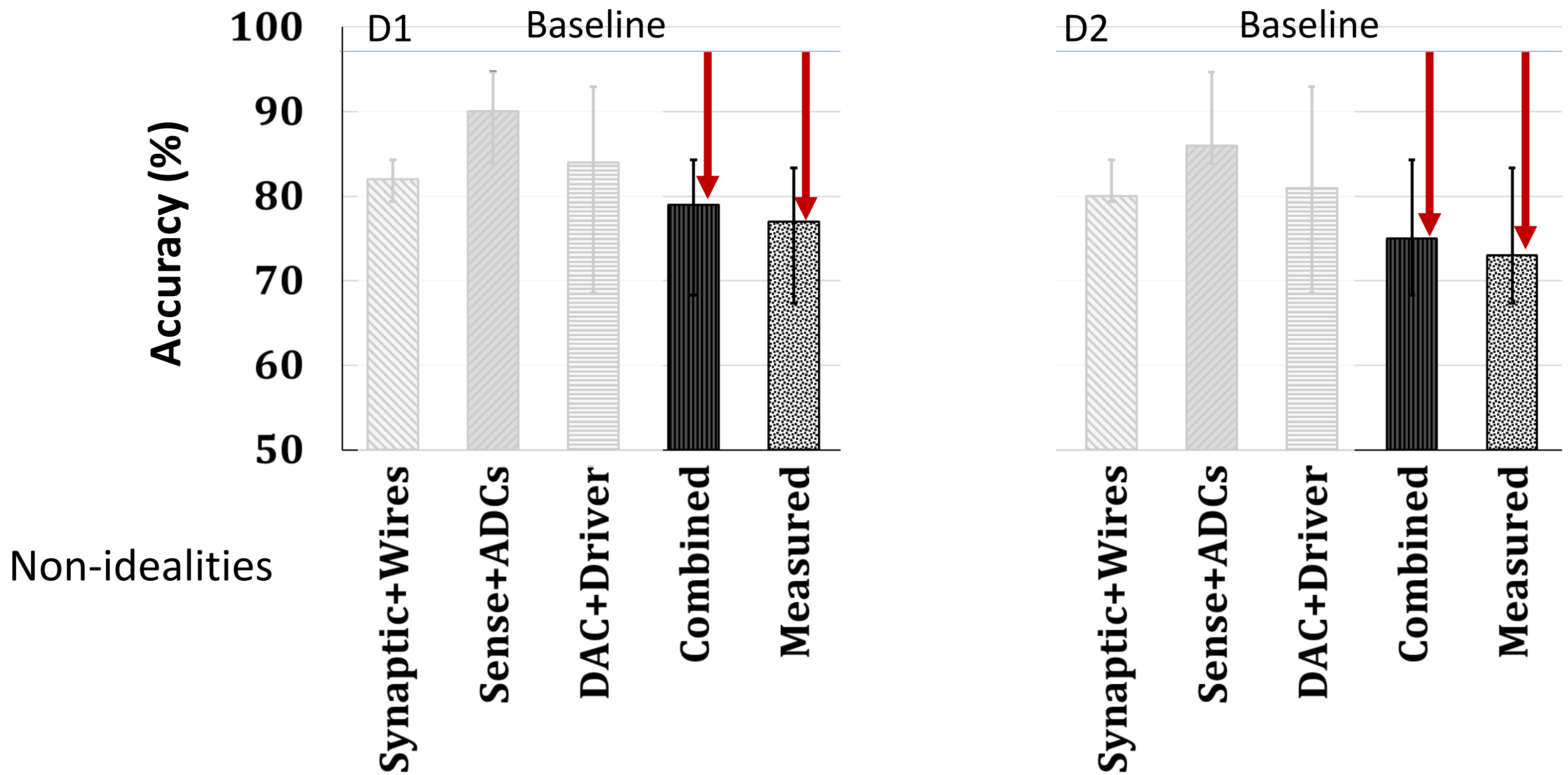
5. Measured

(Approach 1)

- Accuracy Enhancement Techniques

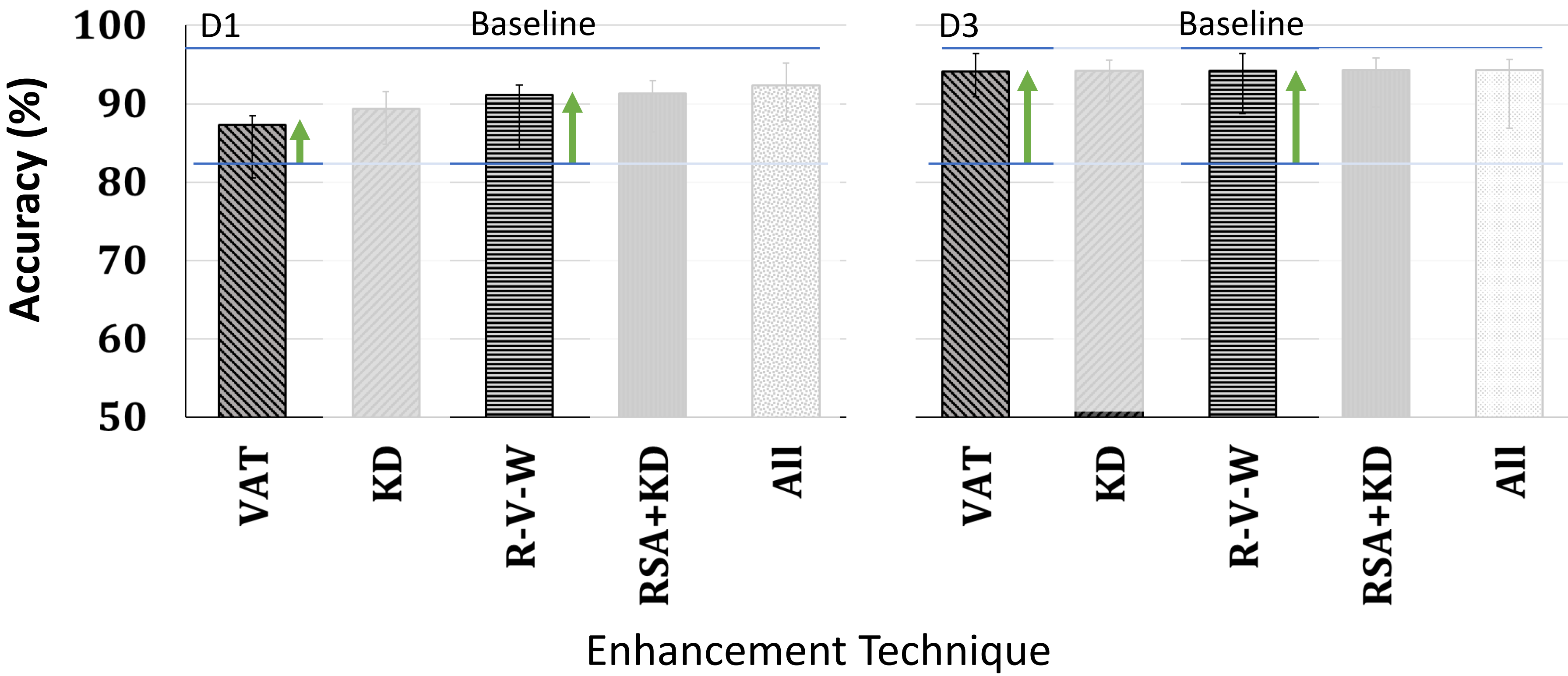
1. Variation Aware Training (VAT)
2. Knowledge Distillation (KD)
3. Read-Verify-Write (R-V-W)
4. Random Sparse Adaptation (RSA) + KD
5. All

Accuracy: All Non-idealities without Mitigation



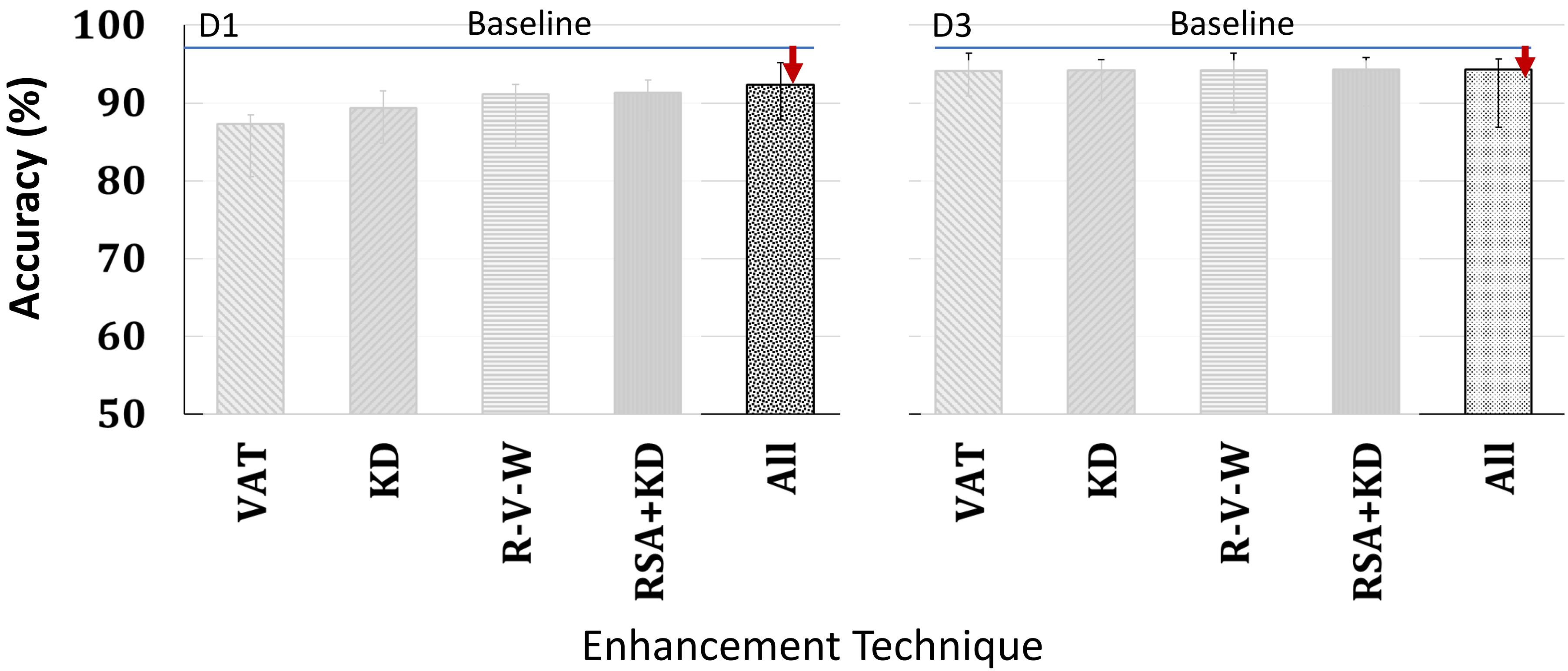
Combined non-idealities leads to significant accuracy loss (**>18%**)

Accuracy: Enhancement Techniques on All Non-idealities



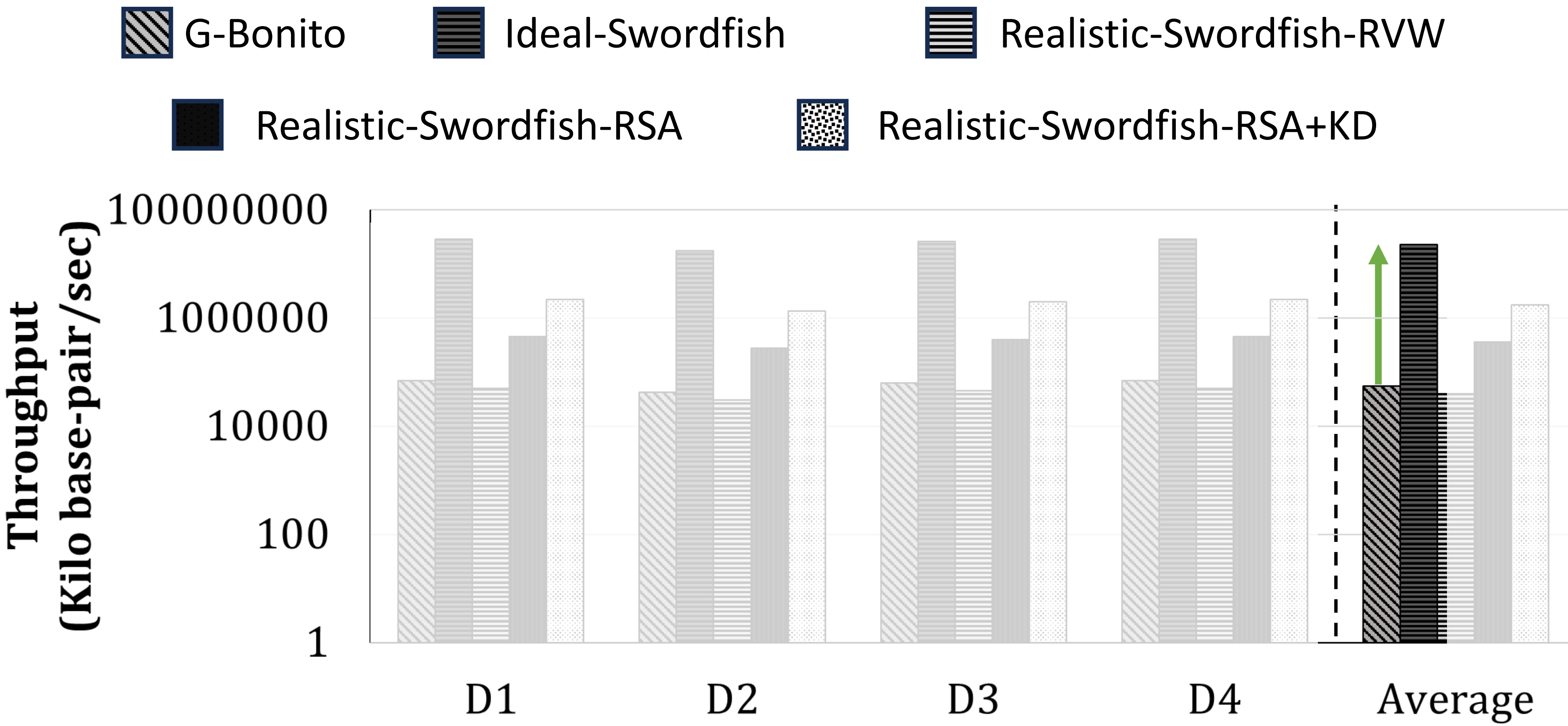
Accuracy enhancement techniques **mitigate** non-idealities,
But differently.

Accuracy: Enhancement Techniques on All Non-idealities



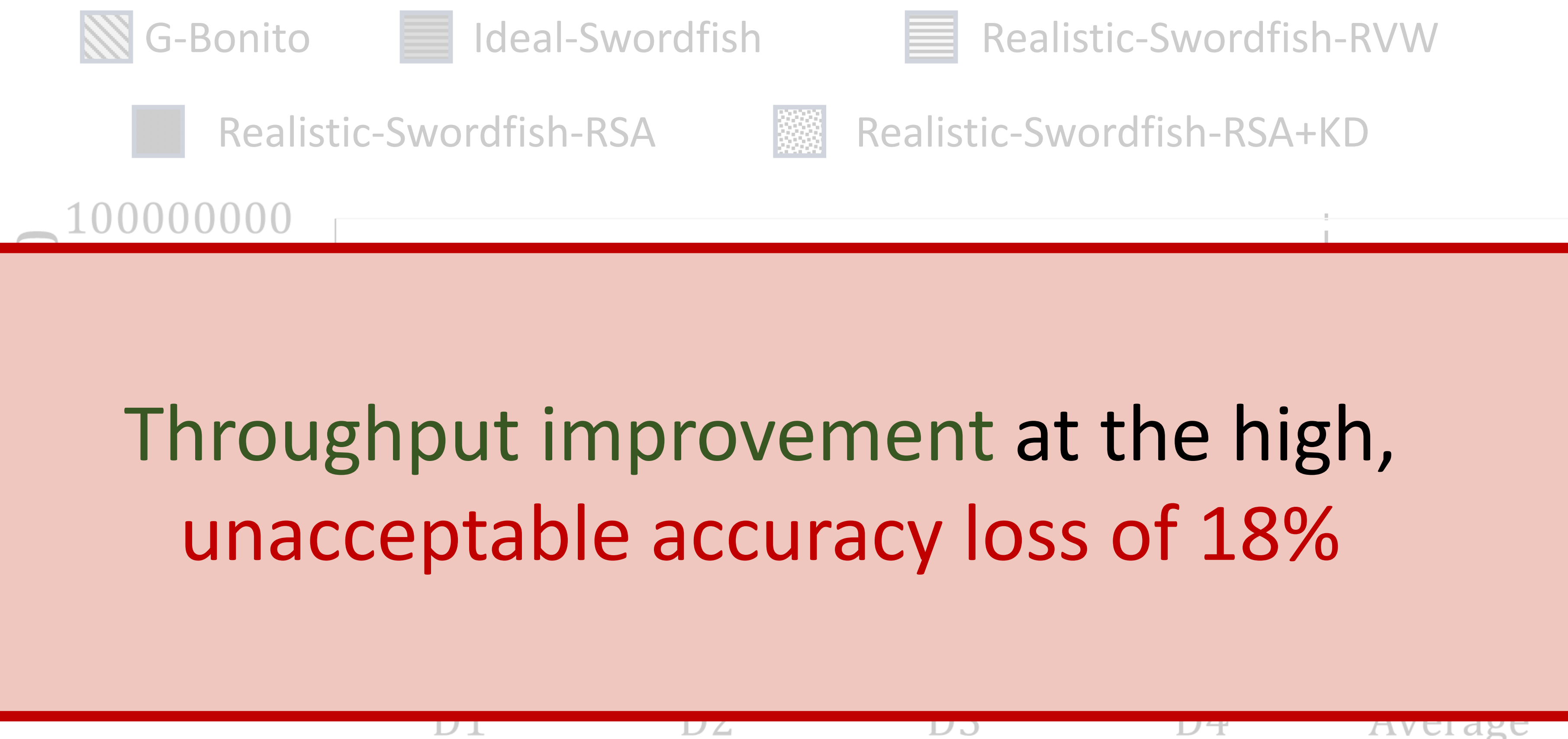
Considerable **accuracy loss** (**>6%**) **even with All** enhancement techniques.

Throughput Analysis



Ideal CIM implementation improves the basecalling throughput over Bonito-GPU by **413.6× on average**

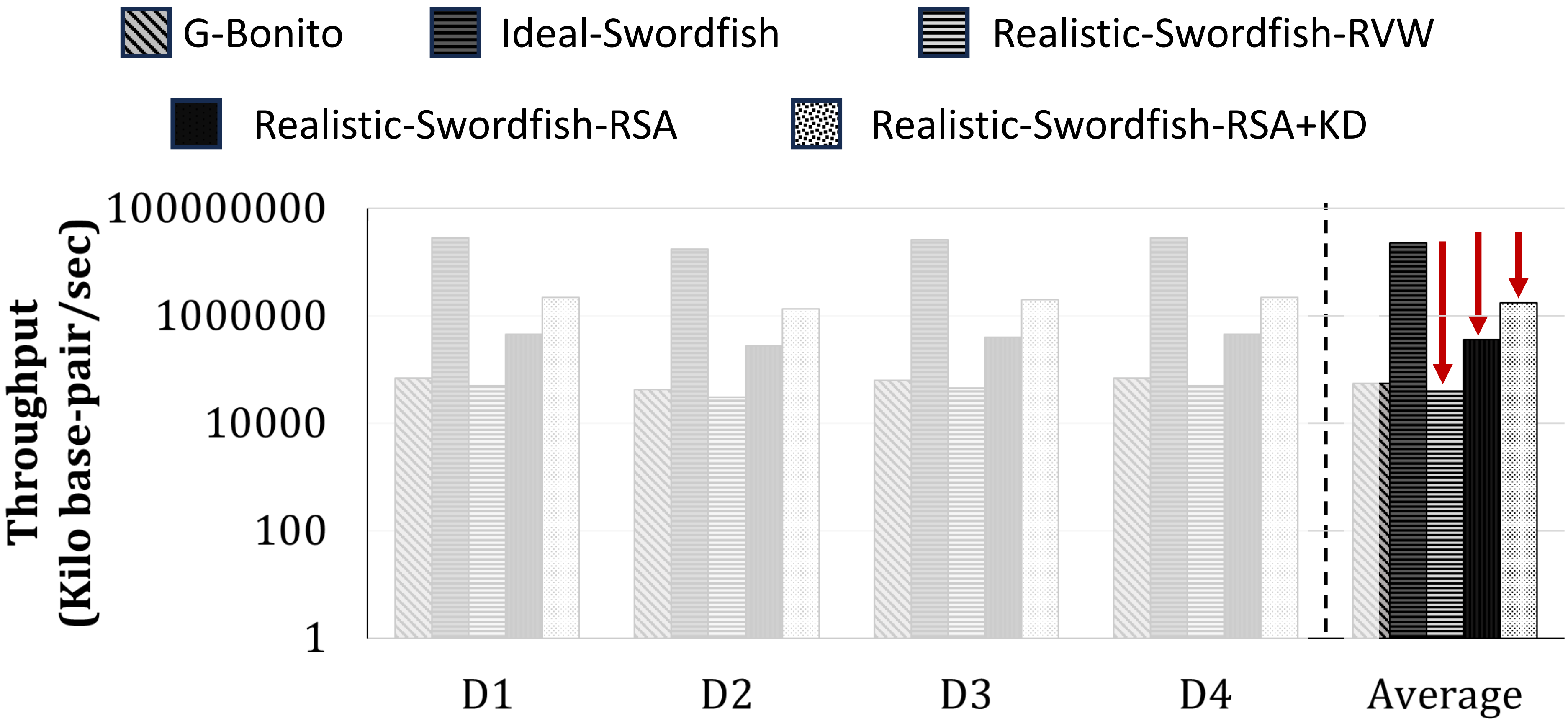
Throughput Analysis



Throughput improvement at the high, unacceptable accuracy loss of 18%

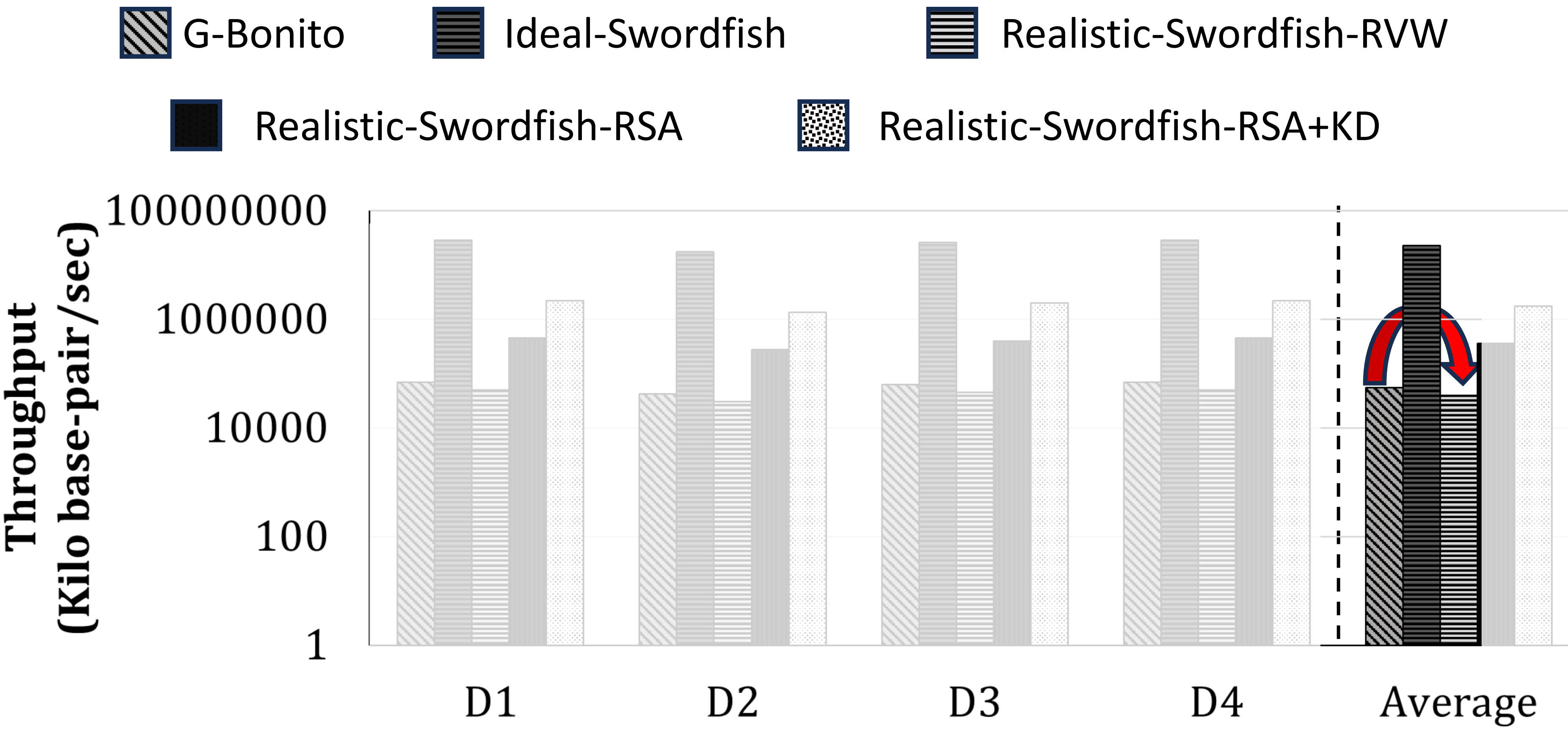
Ideal CIM implementation improves the basecalling throughput over Bonito-GPU by **413.6× on average**

Throughput Analysis



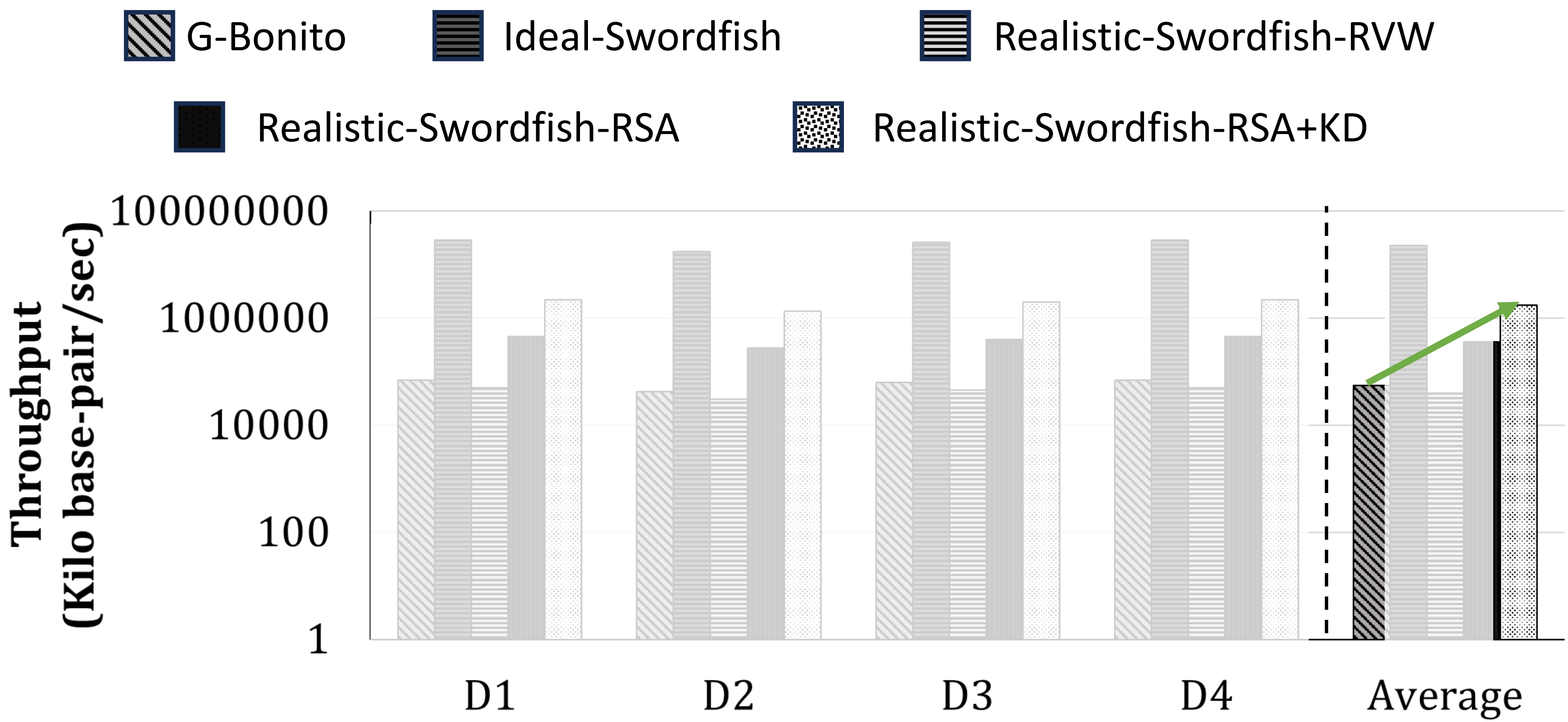
Realistic CIM designs significantly underperform ideal design

Throughput Analysis



Some **realistic CIM designs** **degrade** throughput compared to Bonito-GPU

Throughput Analysis



Realistic CIM design using RSA+KD provides on average **25.7× higher throughput** compared to Bonito-GPU

More in the Paper

- Details on **evaluation methodology**
  - Datasets
  - Array and devices
- **Evaluation results**
  - Individual non-idealities and architectural limitations on accuracy
  - Accuracy enhancements on individual and combined non-idealities and architectural limitations
  - Accuracy vs. Area analysis
  - Observations and trends from the presented figures
  - Results for 256x256 crossbar + comparison with 64x64 crossbars
- **Discussions, takeaways,** and **future work**

Outline

Background & Motivation

Swordfish: Design & Implementation

Evaluation & Key Results

Takeaways & Summary

Takeaways

The target application for memristor-based CIM **matters**

Swordfish enables **realistic** evaluation of accuracy and performance for DNN-based applications on memristor-based CIM

Non-idealities are **detrimental** to both **accuracy** and **performance**

HW/SW co-designed techniques mitigate inaccuracy the most

Summary

Key Contribution: Swordfish; the **first framework** for memristor-based CIM that uses **characterized memories** and **accurate models** to
1) accurately and **realistically** evaluate the effects of **non-idealities** on basecalling **accuracy** and **throughput**
2) comprehensively investigate the impact of **accuracy enhancement techniques** on basecalling **accuracy** and **throughput**

Key Results: Across four real datasets of varying sizes, Swordfish **realistically** provides

- **25.7× better average throughput** compared to state-of-the-art basecalling on GPU
- **12% mitigation in basecalling accuracy loss** after hardware/software co-designed enhancement techniques
- **Three** new **insights** on future research directions for **accuracy enhancement techniques**

Many opportunities for

- **Realistically evaluating** accuracy and throughput **other DNNs** on memristor-based CIM
- Developing and evaluating **novel accuracy enhancement techniques**, on software, hardware, or both
- We should remain **cautious applying known acceleration techniques** to emerging technologies, architectures, and applications

Swordfish:

A Framework for Evaluating Deep Neural Network-based Basecalling using Computation-in-Memory with Non-Ideal Memristors

Questions?

Quantum & Computer Engineering

SAFARI
SAFARI Research Group
safari.ethz.ch

TU Delft
Delft University of Technology

AMD

ETH zürich