tinyML® EMEA

Enabling Ultra-low Power Machine Learning at the Edge

tinyML EMEA Technical Forum 2021 Proceedings

June 7 – 10, 2021

Virtual Event

TINY ML

www.tinyML.org

ZigZag: An Architecture-Mapping Design Space Exploration (DSE) Framework for Deep Learning Accelerator

Linyan Mei, Pouya Houshmand, Arne Symons, Vikram Jain
and Marian Verhelst

MICAS Labs, ESAT, KU Leuven, Belgium

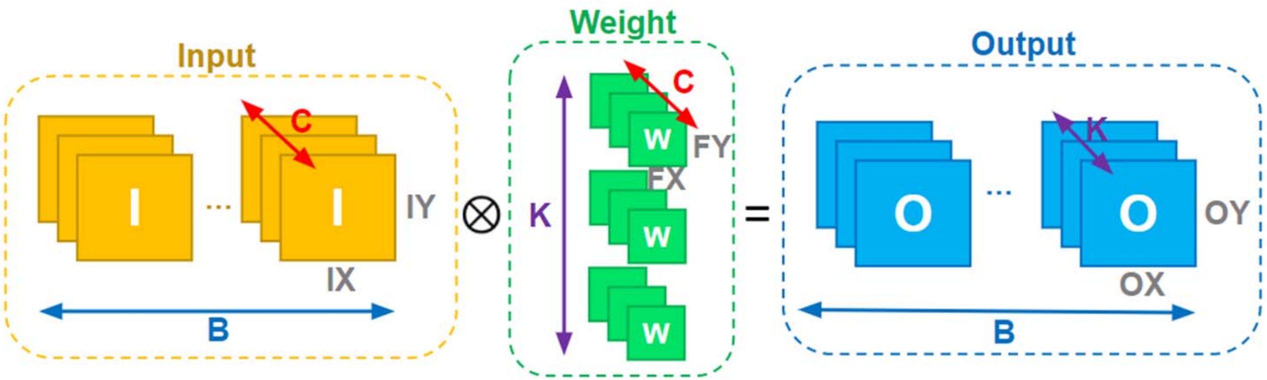
Outline

- Introduction
- Methodology
- Result
- Extension
- Conclusion & Key Takeaways

Outline

- Introduction
  - DNN Layer
  - DNN Accelerator
  - DNN Mapping
  - Co-Exploration
- Methodology
- Result
- Extension
- Conclusion & Key Takeaways

DNN Layer



```
for b = 0 to B-1                  (B: I/O batch size)
  for k = 0 to K-1                (K: O channel/W kernel)
    for c = 0 to C-1              (C: I/W channel)
      for oy = 0 to OY-1          (OY: O row)
        for ox = 0 to OX-1        (OX: O column)
          for fy = 0 to FY-1      (FY: W kernel row)
            for fx = 0 to FX-1    (FX: W kernel column)
```

I	for Input
W	for Weight
O	for Output

O[b][k][oy][ox] += **I**[b][c][oy+fy][ox+fx] × **W**[k][c][fy][fx]

	B	K	C	OY	OX	FY	FX
W	✗	✓	✓	✗	✗	✓	✓
I	✓	✗	✓	$?^{IY}$	$?^{IX}$	$?^{IY}$	$?^{IX}$
O	✓	✓	✗	✓	✓	✗	✗

✓ relevant **(r)**
✗ irrelevant **(ir)**
? partially relevant **(pr)**
$?^{IX/IY}$ partially relevant to IX/IY

A DNN Conv2D layer:

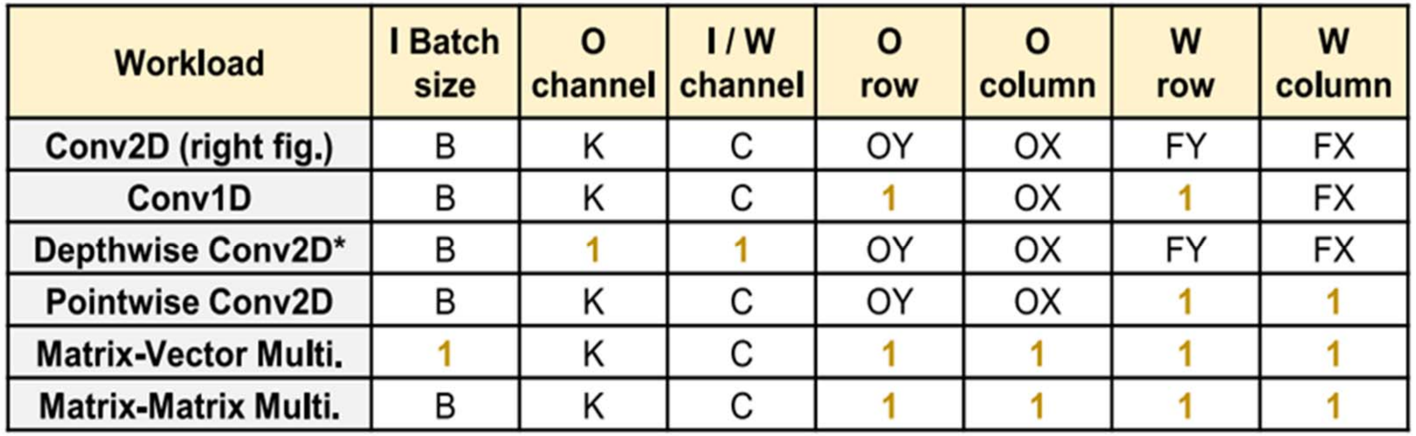
3D operand (**W/I/O**) space.

7D nested for-loop MAC operation space.

Each Operand has its own (ir)relevant loop dimensions.

- **r** loops contribute to **data size**.
- **ir** loops contribute to **data reuse**.
- **pr** loops contribute to both **data size** and **data reuse**.

DNN Layer



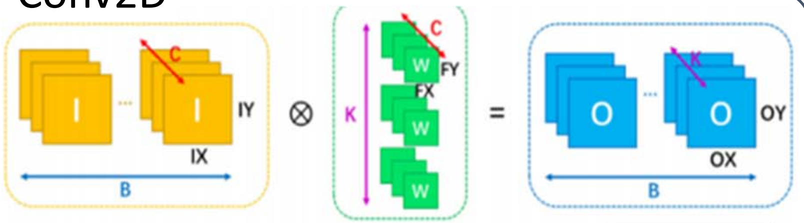
Workload	I Batch size	O channel	I / W channel	O row	O column	W row	W column
Conv2D (right fig.)	B	K	C	OY	OX	FY	FX
Conv1D	B	K	C	1	OX	1	FX
Depthwise Conv2D*	B	1	1	OY	OX	FY	FX
Pointwise Conv2D	B	K	C	OY	OX	1	1
Matrix-Vector Multi.	1	K	C	1	1	1	1
Matrix-Matrix Multi.	B	K	C	1	1	1	1

* Repeat 'C' or 'K' times to finish one Depthwise Conv2D layer (C = K).

A lot of **ML workloads** can fit into the regular **nested for-loop format**.

No data dependency between each for-loop.

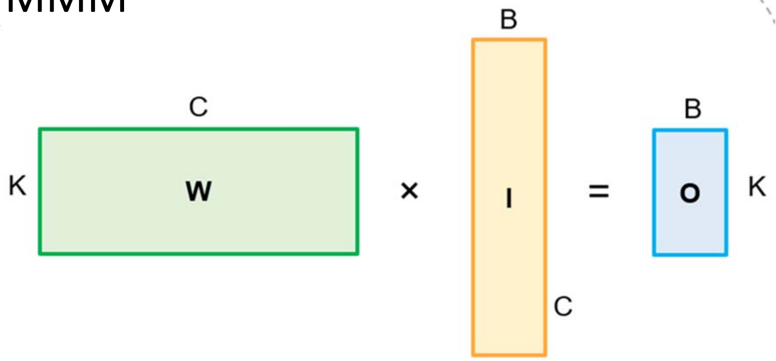
Conv2D



```
for b = 0 to B-1                      (B: I/O batch size)
 for k = 0 to K-1                     (K: O channel/W kernel)
  for c = 0 to C-1                    (C: I/W channel)
   for oy = 0 to OY-1                 (OY: O row)
    for ox = 0 to OX-1                (OX: O column)
     for fy = 0 to FY-1               (FY: W kernel row)
      for fx = 0 to FX-1              (FX: W kernel column)
       O[b][k][oy][ox] += I[b][c][oy+fy][ox+fx] × W[k][c][fy][fx]
```

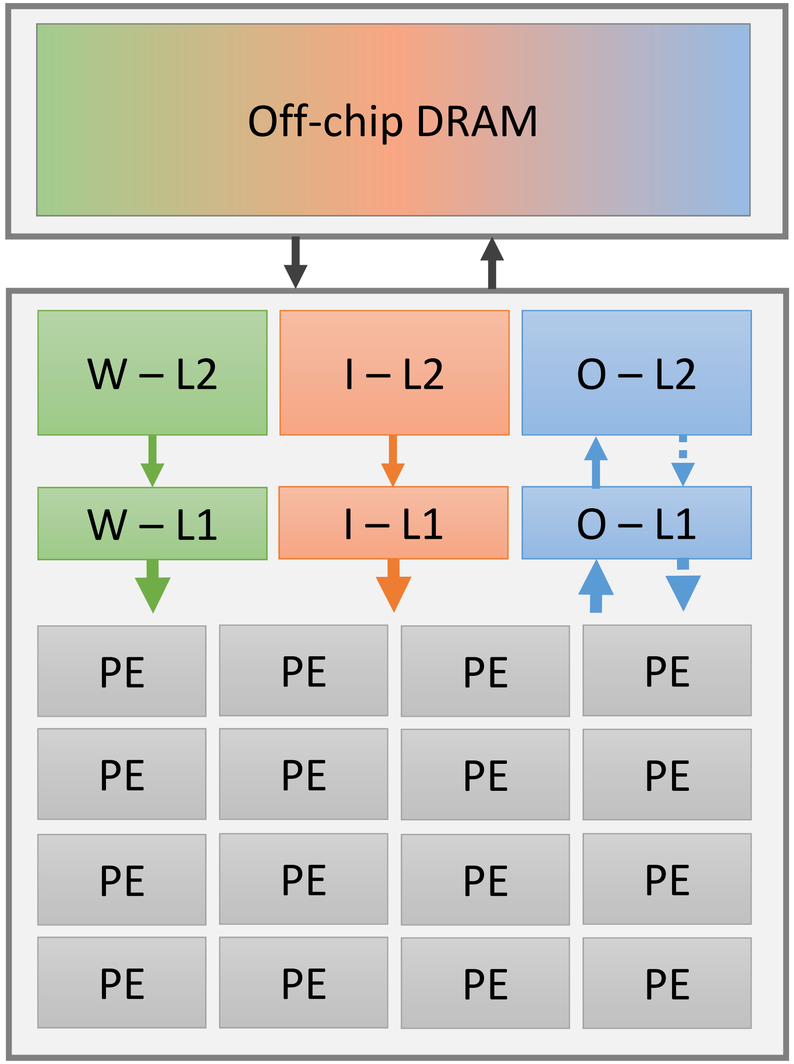
I for Input
W for Weight
O for Output

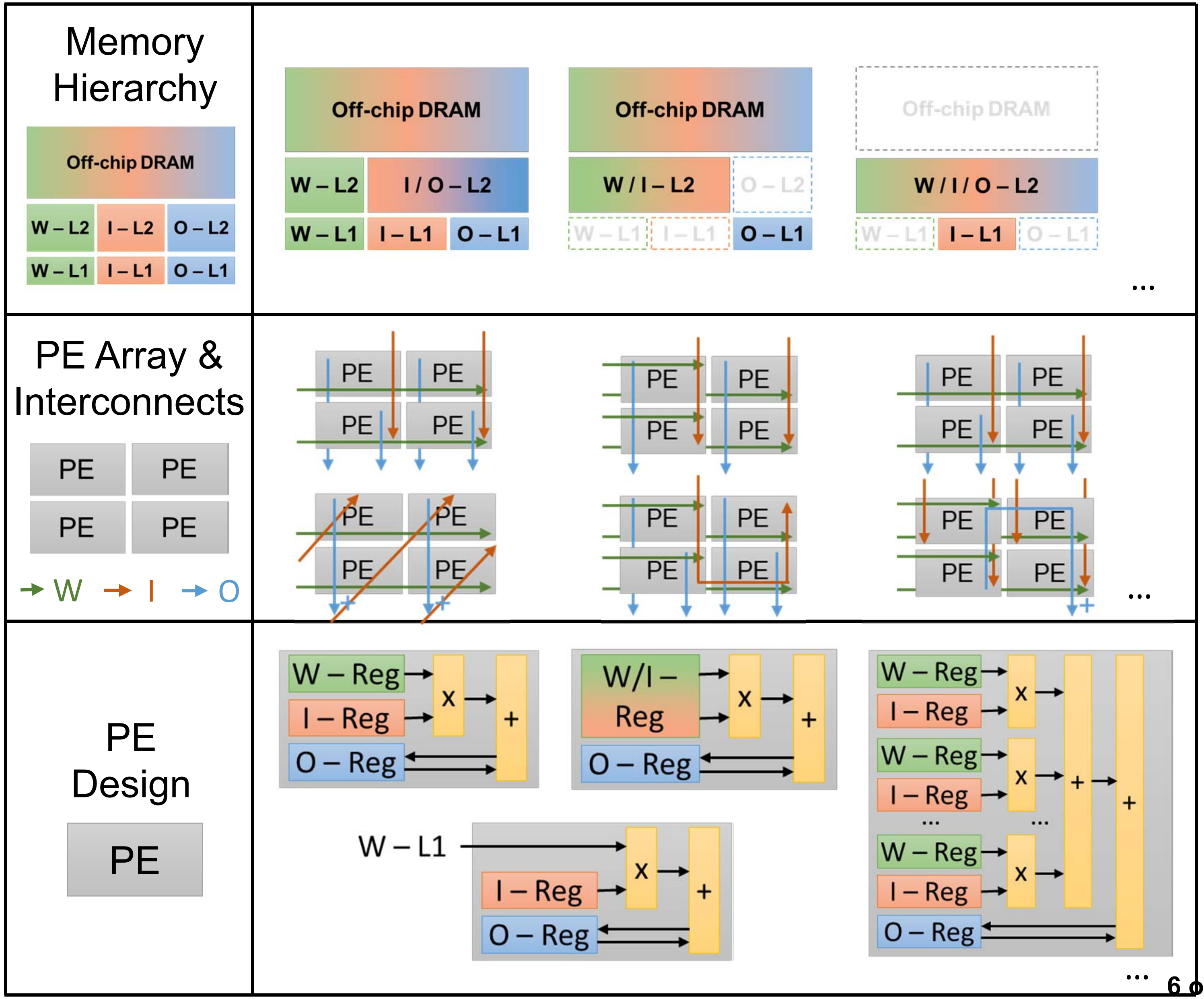
MMM



```
for b = 0 to B-1          (B: I/O col)
 for k = 0 to K-1         (K: W/O row)
  for c = 0 to C-1        (C: W col / I row)
   O[b][k] += I[b][c] × W[k][c]
```

DNN Accelerator





Large Design Degrees of Freedom!

Layer-wise Mapping (a.k.a. Dataflow)

Temporal Mapping
(Loop tilling, ordering)

Data Stationarity

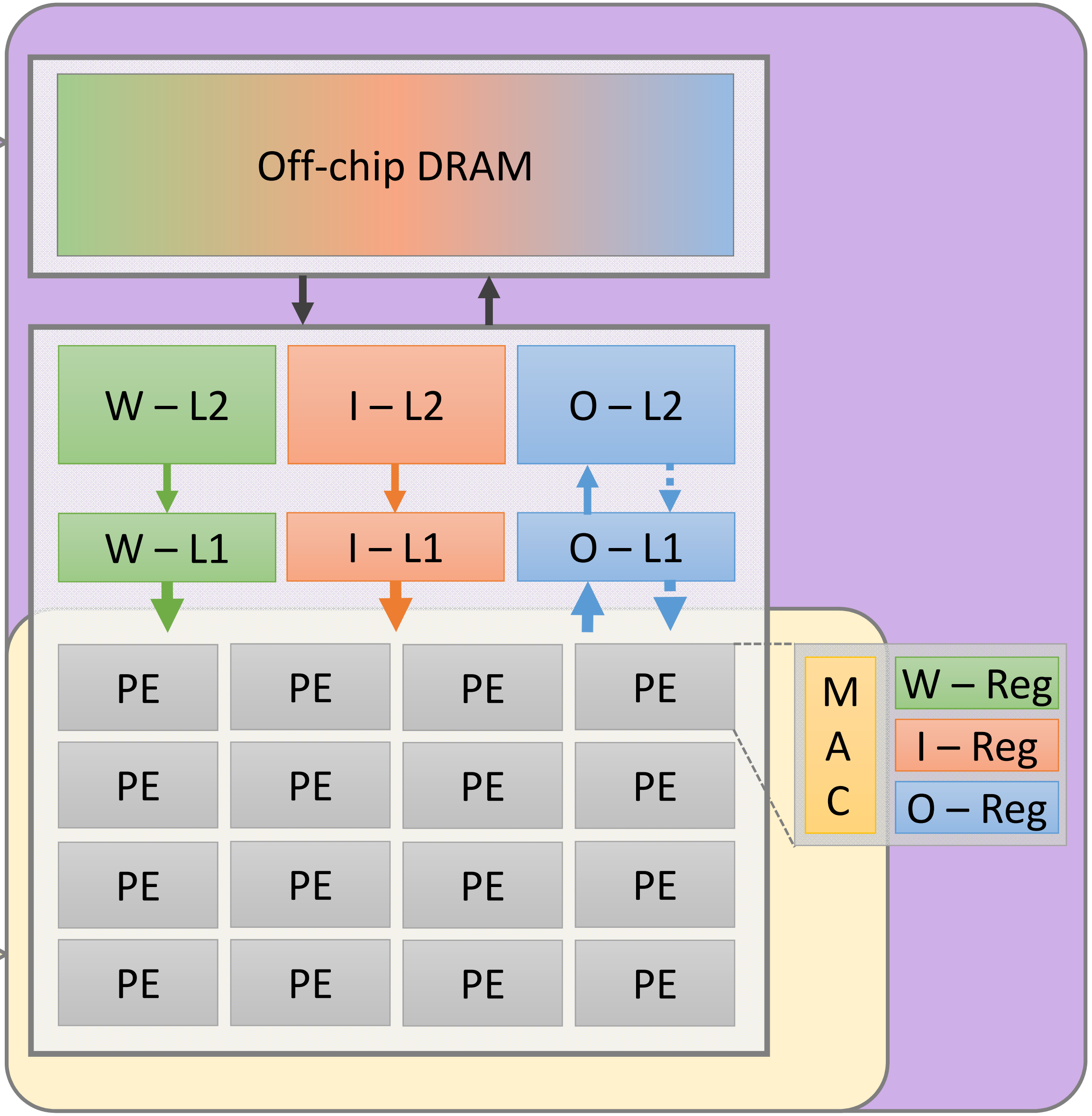
```
for b = 0 to B-1                      (B: I/O batch size)
  for k = 0 to K-1                    (K: O channel/W kernel)
    for c = 0 to C-1                  (C: I/W channel)
      for oy = 0 to OY-1              (OY: O row)
        for ox = 0 to OX-1            (OX: O column)
          for fy = 0 to FY-1          (FY: W kernel row)
            for fx = 0 to FX-1        (FX: W kernel column)

              O[b][k][oy][ox] += I[b][c][oy+fy][ox+fx] × W[k][c][fy][fx]
```

I	for Input
W	for Weight
O	for Output

Operation Parallelism

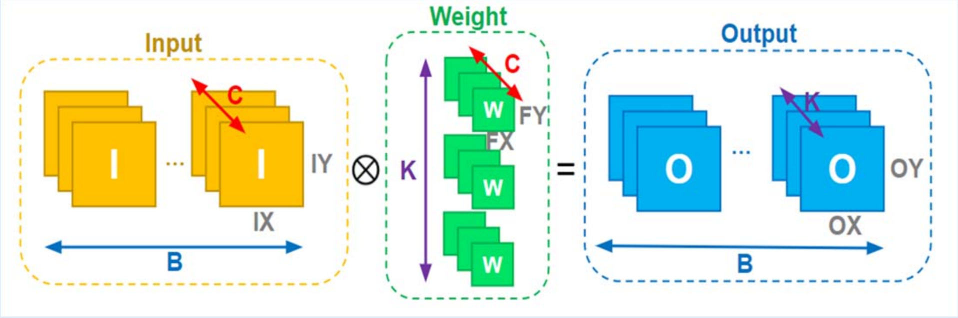
Spatial Mapping
(Loop unrolling)



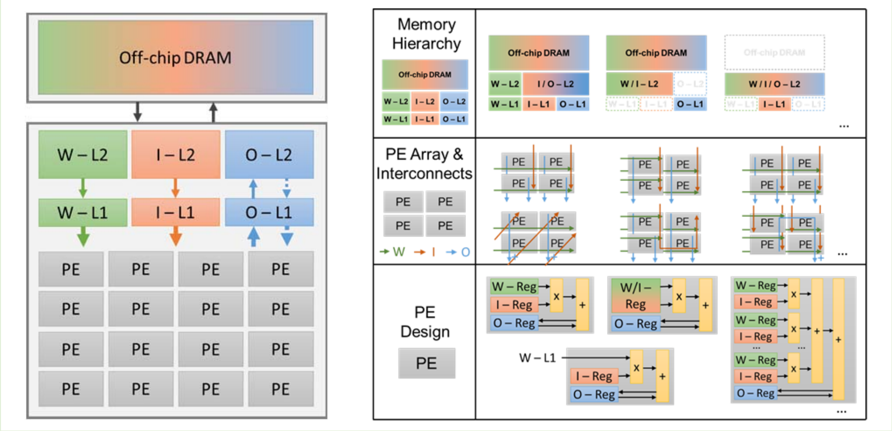
Large Design Degrees of Freedom!

Co-Exploration

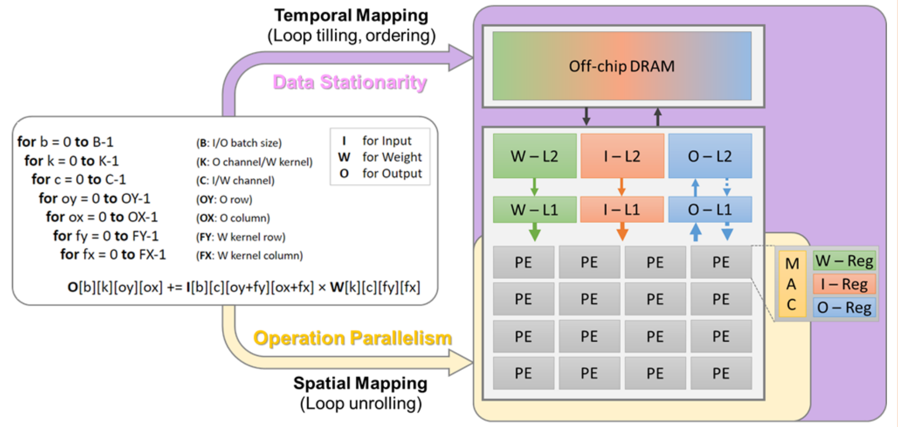
Algorithm



Hardware



Mapping



```
for b = 0 to B-1                      (B: I/O batch size)
 for k = 0 to K-1                     (K: O channel/W kernel)
  for c = 0 to C-1                    (C: I/W channel)
   for oy = 0 to OY-1                 (OY: O row)
    for ox = 0 to OX-1                (OX: O column)
     for fy = 0 to FY-1               (FY: W kernel row)
      for fx = 0 to FX-1              (FX: W kernel column)

       O[b][k][oy][ox] += I[b][c][oy+fy][ox+fx] × W[k][c][fy][fx]
```

I	for Input
W	for Weight
O	for Output

Technology and Others

Technology: 65nm/40nm/28nm/…, NVM, CIM, 3D IC, etc.

Others: Sparsity, various precisions, cross-layer execution, etc.

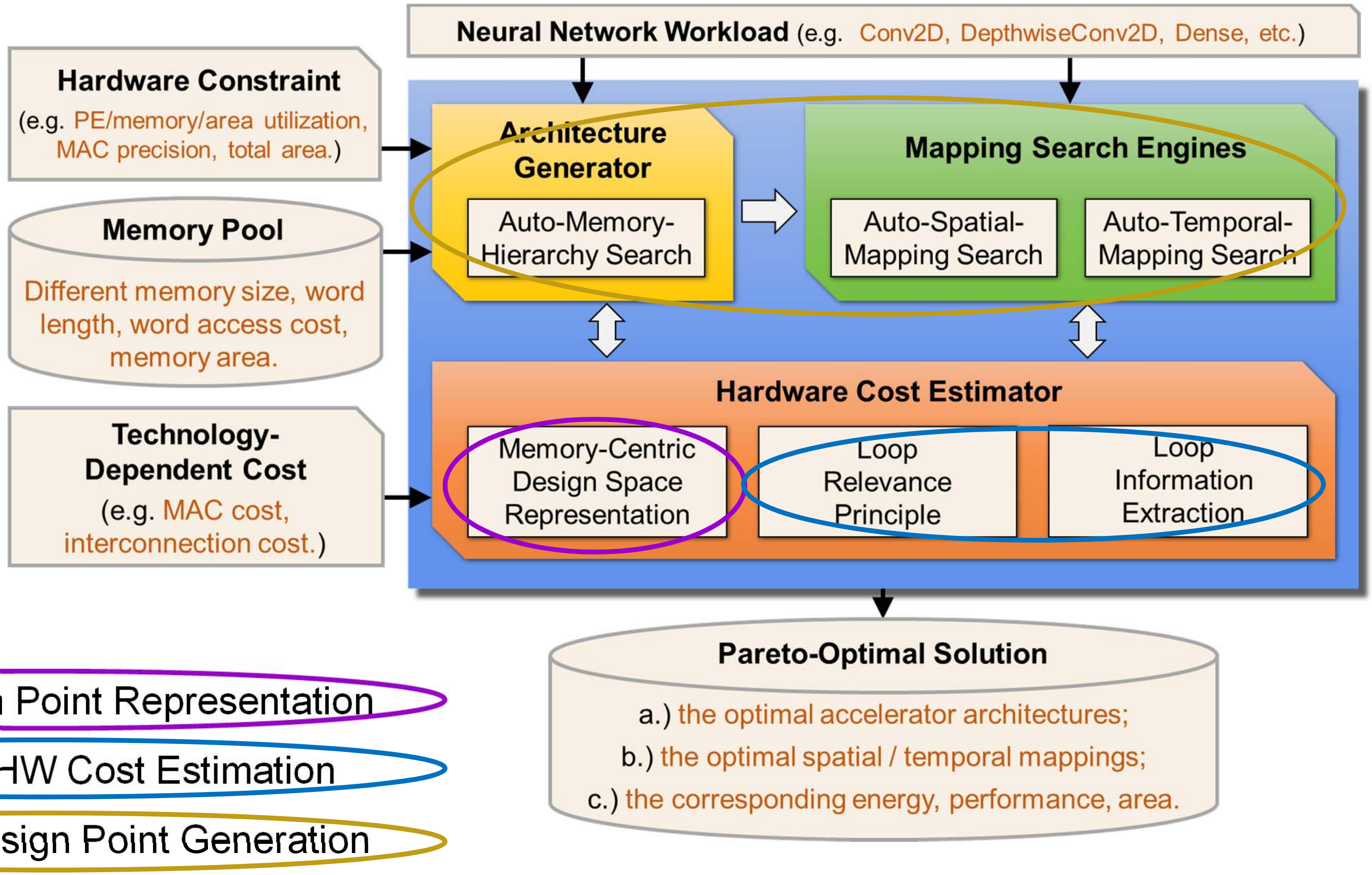
HUGE design space at **each level** & at **combined levels.**

Regular workload & **Deterministic** processing flow & **Well-defined** HW components.

Outline

- Introduction
- Methodology
  - ZigZag Overview [TC 2021]
  - Unified Design Point Representation
  - Standardized HW Cost Estimation
  - Automated Design Point Generation
- Result
- Extension
- Conclusion & Key Takeaways

ZigZag Overview



Neural Network Workload (e.g. Conv2D, DepthwiseConv2D, Dense, etc.)

Hardware Constraint
(e.g. PE/memory/area utilization, MAC precision, total area.)

Memory Pool
Different memory size, word length, word access cost, memory area.

Technology-Dependent Cost
(e.g. MAC cost, interconnection cost.)

Architecture Generator
Auto-Memory-Hierarchy Search

Mapping Search Engines
Auto-Spatial-Mapping Search
Auto-Temporal-Mapping Search

Hardware Cost Estimator
Memory-Centric Design Space Representation
Loop Relevance Principle
Loop Information Extraction

Pareto-Optimal Solution
a.) the optimal accelerator architectures;
b.) the optimal spatial / temporal mappings;
c.) the corresponding energy, performance, area.

- Unified Design Point Representation
- Standardized HW Cost Estimation
- Automated Design Point Generation

Unified Design Point Representation

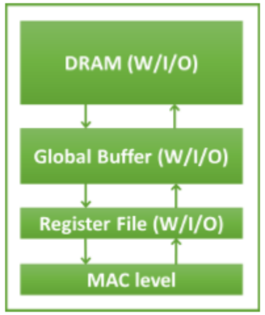
HW Architecture

Algorithm-to-HW Mapping

Even Mapping
(At all shared memory levels, excluding the top memory, W/I/O loop blocking are the same.)

Uneven Mapping
(Memory level and W/I/O loop blocking are **decoupled**.)

Balanced Memory Hierarchy
(At all memory levels, W/I/O are shared.)



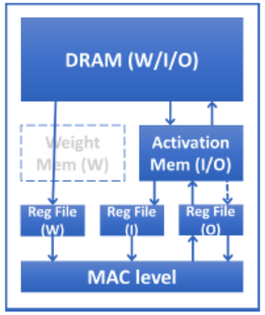
(a)

Weight			Input			Output		
DRAM	K	4	DRAM	K	4	DRAM	K	4
	C	24		C	24		C	24
Global	OX	2	Global	OX	2	Global	OX	2
Buffer	K	4	Buffer	K	4	Buffer	K	4
	OX	13		OX	13		OX	13
	OYu|FYu|OYu	13|5|2		OYu|FYu|OYu	13|5|2		OYu|FYu|OYu	13|5|2
	K	16		K	16		K	16
Register	C	2	Register	C	2	Register	C	2
File	FX	5	File	FX	5	File	FX	5
MAC Level			MAC Level			MAC Level		

(c)

Weight			Input			Output		
DRAM	K	4	DRAM	K	4	DRAM	K	4
	C	24		C	24		C	24
Global	OX	2	Global	OX	2		OX	2
Buffer	K	4	Buffer	K	4	Global	K	4
	OX	13		OYu|FYu|OYu	13|5|2	Buffer	OX	13
	K	16		OX	13		OYu|FYu|OYu	13|5|2
	OYu|FYu|OYu	13|5|2		K	16		K	16
Register	C	2	Register	C	2	Register	C	2
File	FX	5	File	FX	5	File	FX	5
MAC Level			MAC Level			MAC Level		

Unbalanced Memory Hierarchy
(Memory level and W/I/O are **decoupled**.)



(b)

Weight			Input			Output		
DRAM	K	4	DRAM	K	4	DRAM	K	4
	C	24		C	24		C	24
	OX	2	Activ.	OX	2	Activ.	OX	2
	K	4	Mem	K	4	Mem	K	4
	OX	13	(I/O)	OX	13	(I/O)	OX	13
	OYu|FYu|OYu	13|5|2		OYu|FYu|OYu	13|5|2		OYu|FYu|OYu	13|5|2
	K	16		K	16		K	16
Reg File	C	2	Reg File	C	2	Reg File	C	2
(W)	FX	5	(I)	FX	5	(O)	FX	5
MAC Level			MAC Level			MAC Level		

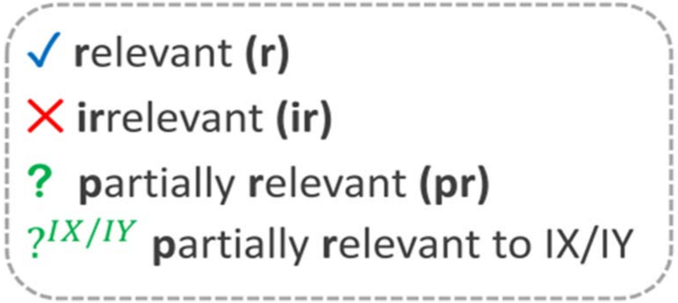
(d)

Weight			Input			Output		
DRAM	K	32		K	32	DRAM	K	32
	C	12	Activ.	C	12	Activ.	C	12
	OYu|FYu|OYu	13|5|2	Mem	OX	13	Mem (I/O)	OX	13
	OX	13	(I/O)	C	2		OYu|FYu|OYu	13|5|2
	C	2		OX	2		C	2
	OX	2		OYu|FYu|OYu	13|5|2		OX	2
	FX	5		FX	5		FX	5
Reg File	C	2	Reg File	C	2	Reg File	C	2
(W)	K	8	(I)	K	8	(O)	K	8
MAC Level			MAC Level			MAC Level		

Supporting uneven mapping opens up new mapping possibilities, thus prone to find better design points.

Standardized HW Cost Estimation

Loop Relevance Principle (LRP)

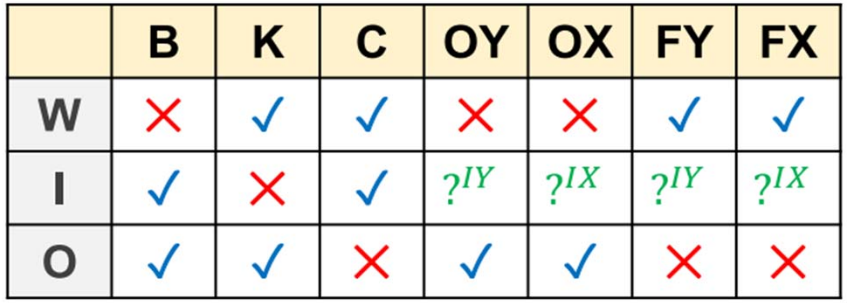


✓ relevant **(r)**

✗ irrelevant **(ir)**

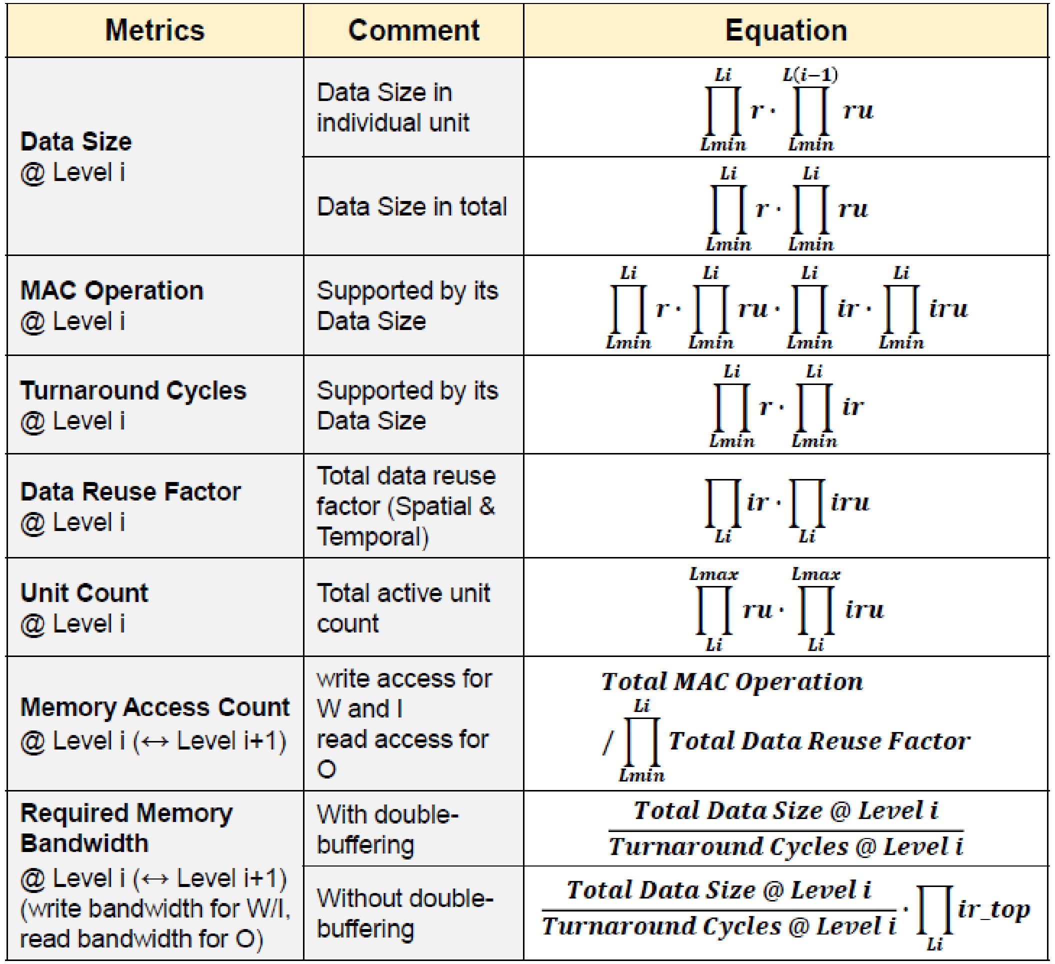
? partially relevant **(pr)**

$?^{IX/IY}$ partially relevant to IX/IY



	B	K	C	OY	OX	FY	FX
W	✗	✓	✓	✗	✗	✓	✓
I	✓	✗	✓	$?^{IY}$	$?^{IX}$	$?^{IY}$	$?^{IX}$
O	✓	✓	✗	✓	✓	✗	✗

Extracting Loop Info. based on LRP

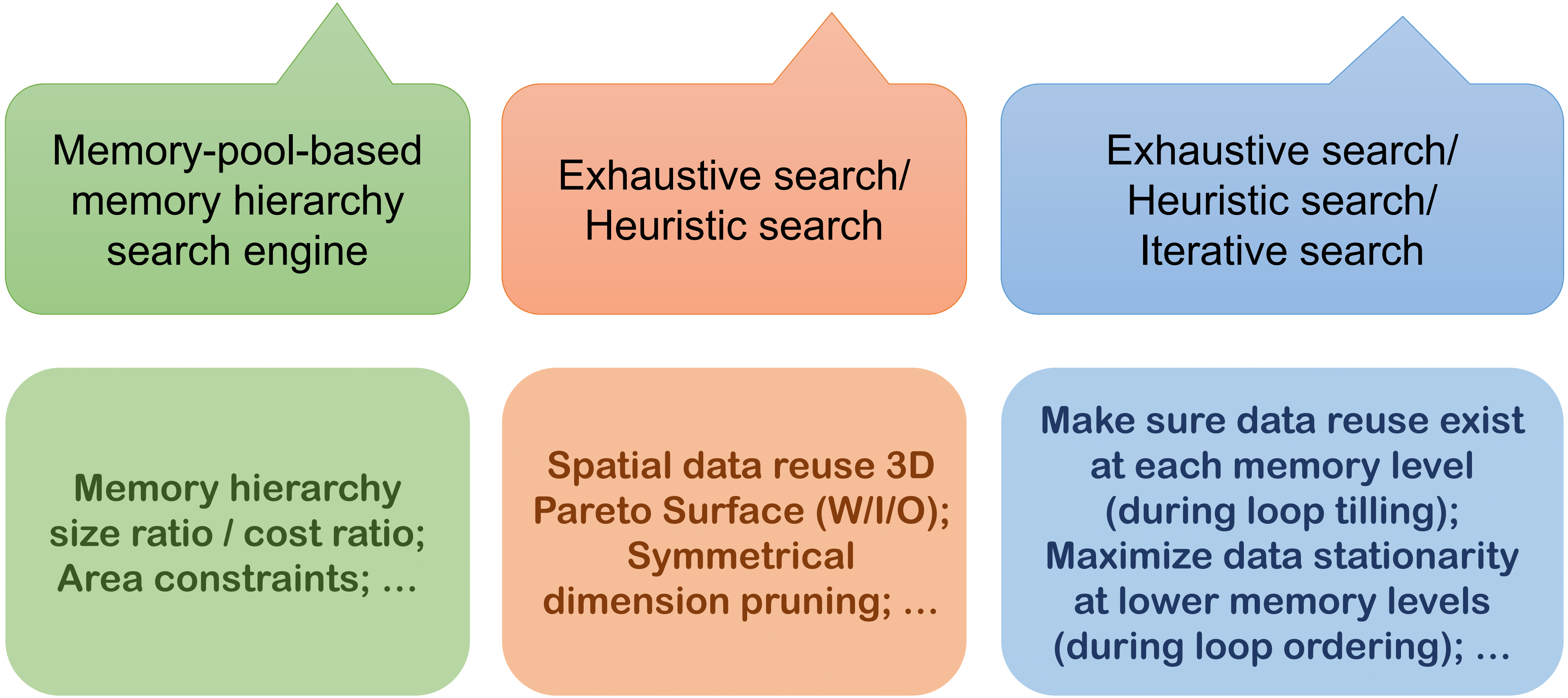


Metrics	Comment	Equation
Data Size @ Level i	Data Size in individual unit	$\prod_{Lmin}^{Li} r \cdot \prod_{Lmin}^{L(i-1)} ru$
	Data Size in total	$\prod_{Lmin}^{Li} r \cdot \prod_{Lmin}^{Li} ru$
MAC Operation @ Level i	Supported by its Data Size	$\prod_{Lmin}^{Li} r \cdot \prod_{Lmin}^{Li} ru \cdot \prod_{Lmin}^{Li} ir \cdot \prod_{Lmin}^{Li} iru$
Turnaround Cycles @ Level i	Supported by its Data Size	$\prod_{Lmin}^{Li} r \cdot \prod_{Lmin}^{Li} ir$
Data Reuse Factor @ Level i	Total data reuse factor (Spatial & Temporal)	$\prod_{Li} ir \cdot \prod_{Li} iru$
Unit Count @ Level i	Total active unit count	$\prod_{Li}^{Lmax} ru \cdot \prod_{Li}^{Lmax} iru$
Memory Access Count @ Level i (↔ Level i+1)	write access for W and I read access for O	$Total\ MAC\ Operation$ $/ \prod_{Lmin}^{Li} Total\ Data\ Reuse\ Factor$
Required Memory Bandwidth @ Level i (↔ Level i+1) (write bandwidth for W/I, read bandwidth for O)	With double-buffering	$\frac{Total\ Data\ Size\ @\ Level\ i}{Turnaround\ Cycles\ @\ Level\ i}$
	Without double-buffering	$\frac{Total\ Data\ Size\ @\ Level\ i}{Turnaround\ Cycles\ @\ Level\ i} \cdot \prod_{Li} ir\_top$

At each memory level (shared or non-shared), for each operand (W/I/O), the key matrices (e.g., memory access count) are extracted following the same procedure.

Automated Design Point Generation

A Design Point = Hardware Arch. + Spatial Mapping + Temporal Mapping



Memory-pool-based memory hierarchy search engine

Exhaustive search/ Heuristic search

Exhaustive search/ Heuristic search/ Iterative search

Pruning Principles

(no/minor optimality loss)

Memory hierarchy size ratio / cost ratio; Area constraints; …

Spatial data reuse 3D Pareto Surface (W/I/O); Symmetrical dimension pruning; …

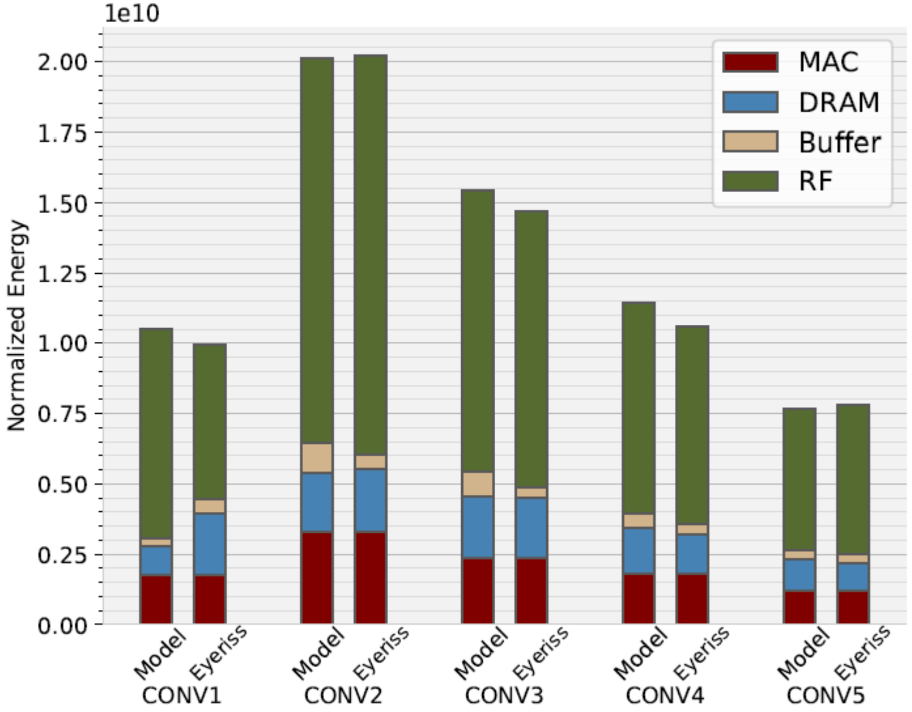
Make sure data reuse exist at each memory level (during loop tilling); Maximize data stationarity at lower memory levels (during loop ordering); …

A lot of clever search/optimization algorithms can be applied in this step.

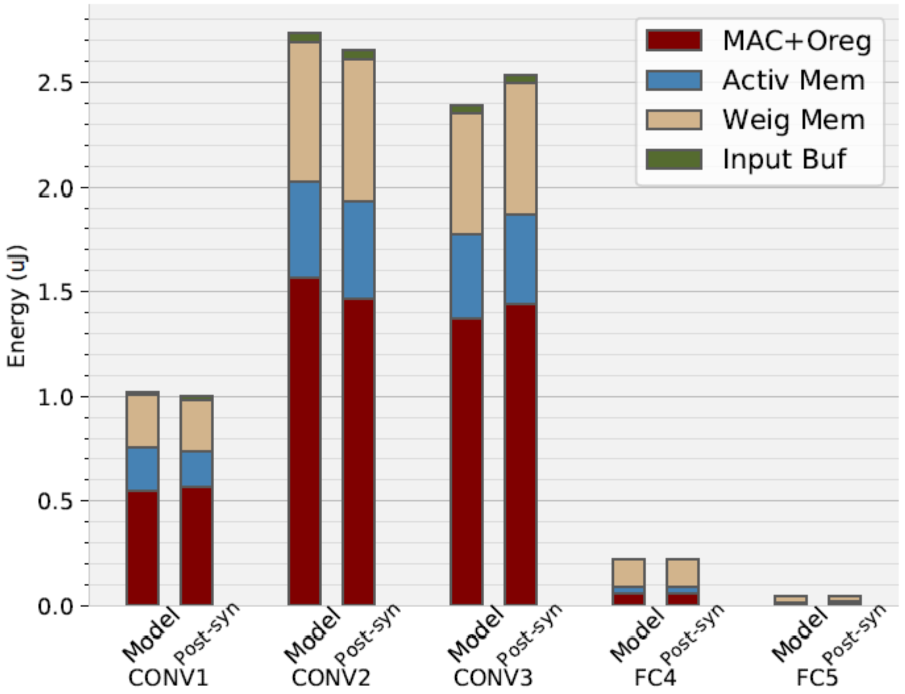
Outline

- Introduction
- Methodology
- Result
  - Validation
  - Case Study
- Extension
- Conclusion & Key Takeaways

Validation Against Real Designs



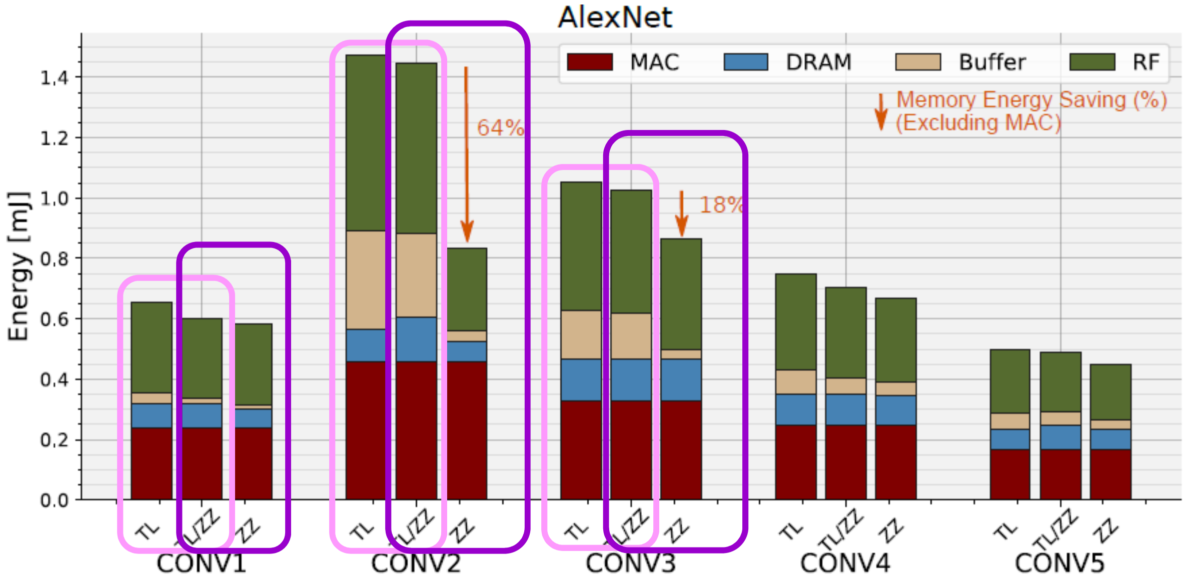
Energy validation against **Eyeriss** published data

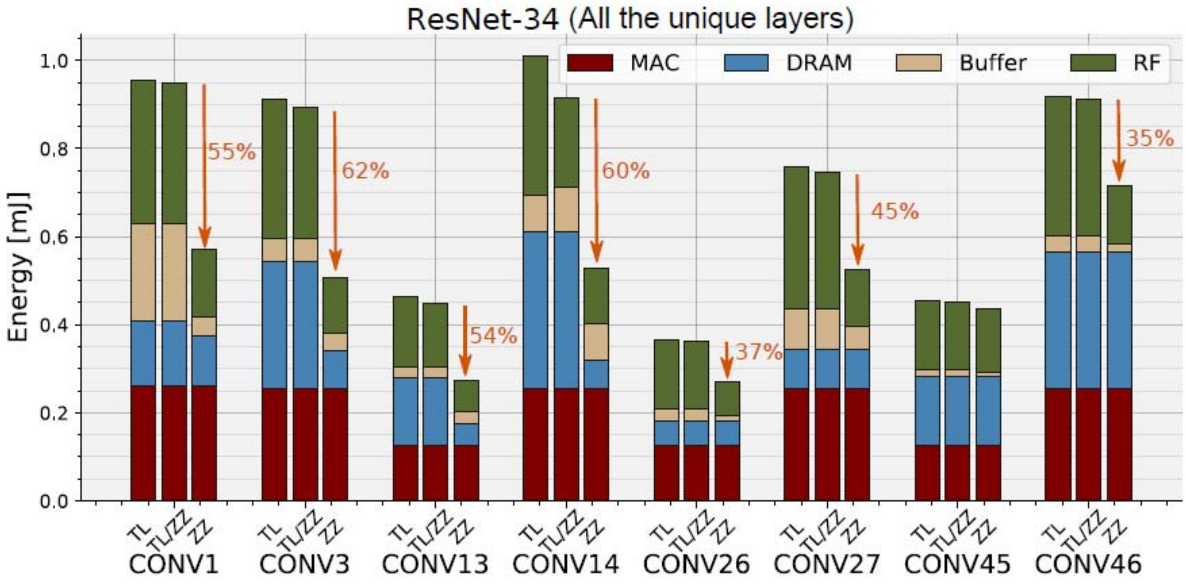


Energy validation against an **in-house accelerator**

The energy mismatches across all layers are within **7.5%**.

Validation Against SotA Framework





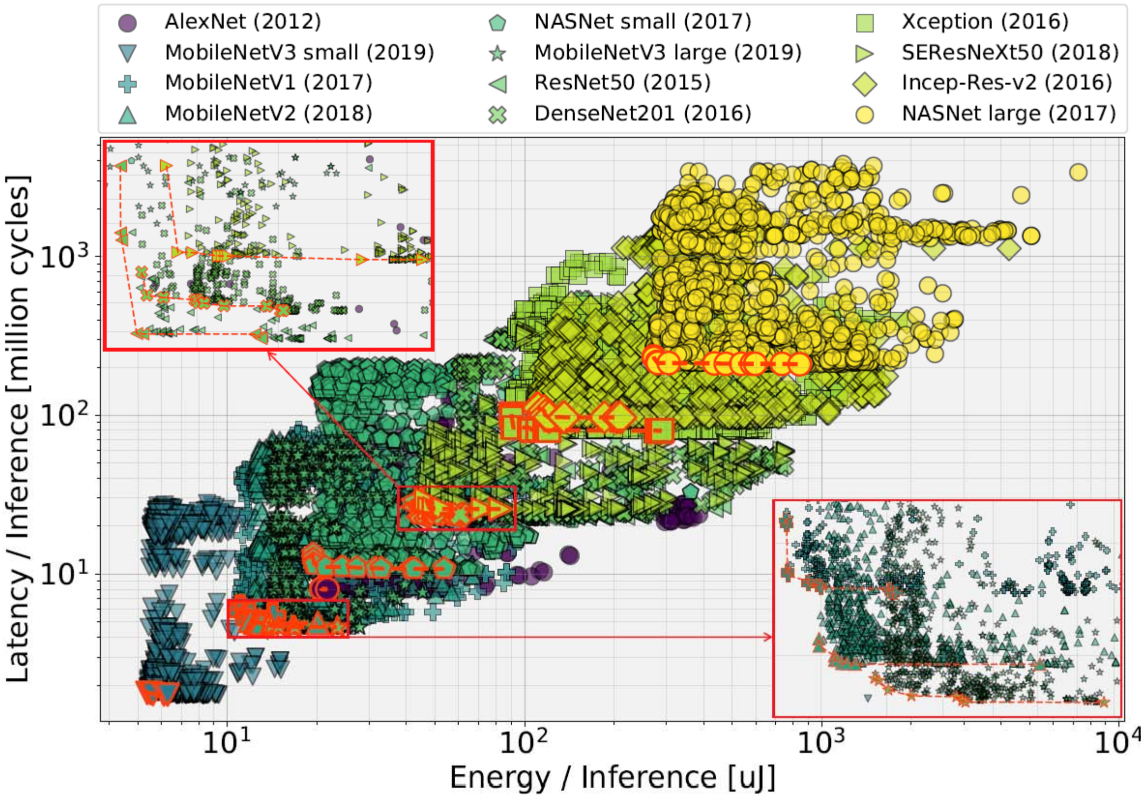
Energy validation against Timeloop+Accelergy (TL ↔ TL/ZZ).

Mapping search engine comparison against Timeloop (TL/ZZ ↔ ZZ).

ZigZag found better design points than Timeloop due to the uneven mapping support.

Case Study

Neural Network HW Cost Comparison



Memory pool @ 65 nm technology, CACTI7

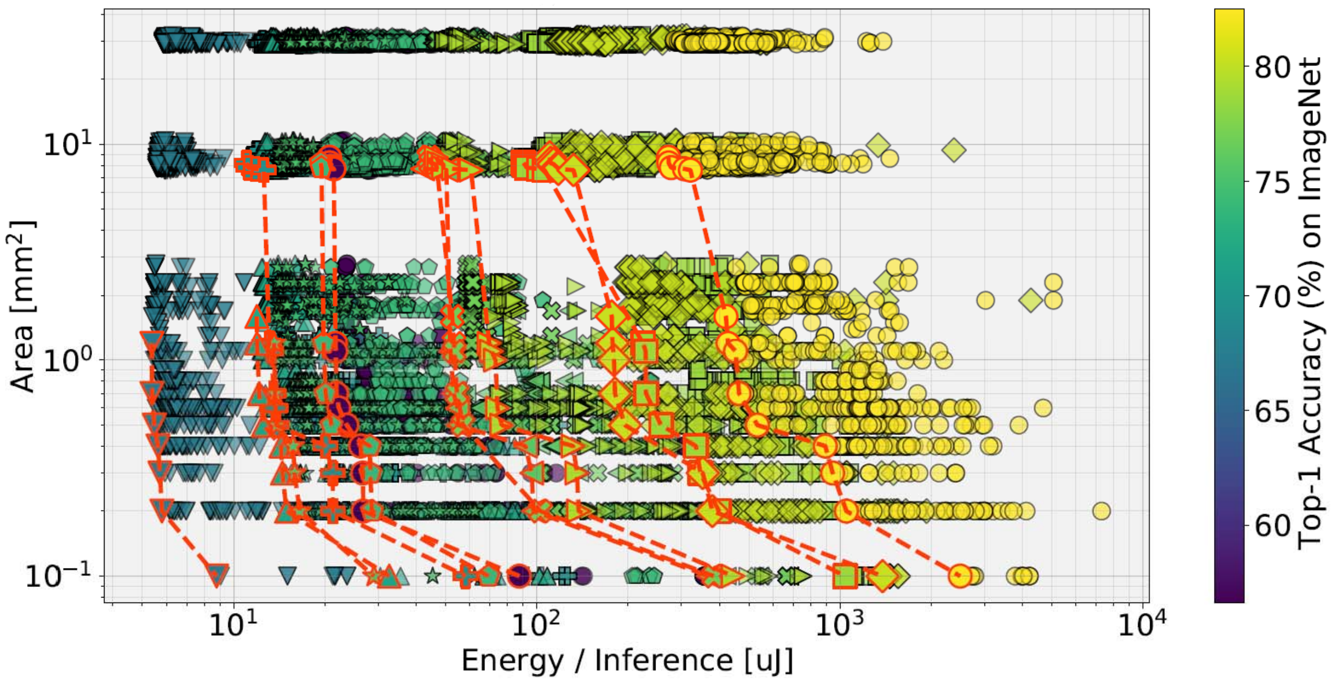
Arch. Level	Inner-PE Reg	On-chip L1	On-chip L2	Off-chip

Mem. Size Option	2 B;
	32 B;
	128 B
		8 KB;
		32 KB
			0.5 MB;
			2 MB
				DRAM

Mem. Bandwidth	16 bit/cycle (r/w)	128 bit/cycle (read/write)		

Mem. Share Option (i.e. 1/2/3 operand(s) share same memory)	All separate	All separate;
		Two shared;
		All shared
			All shared	All shared

Mem. Bypass Option	No bypass	Can bypass	Can bypass	No bypass

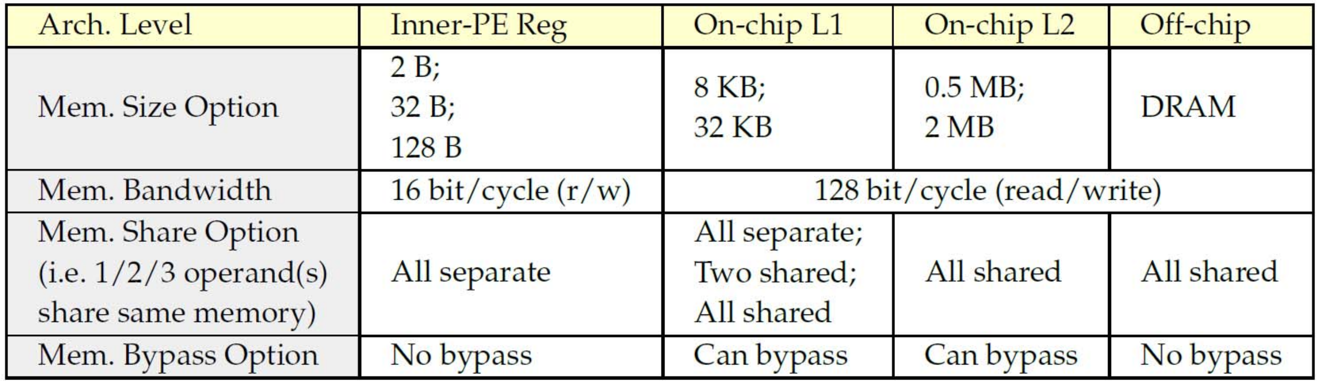


Algorithm accuracy – Energy – Latency – Area design space visualization.

Case Study

Neural Network HW Cost Comparison

Memory pool @ 65 nm technology, CACTI7



Arch. Level	Inner-PE Reg	On-chip L1	On-chip L2	Off-chip

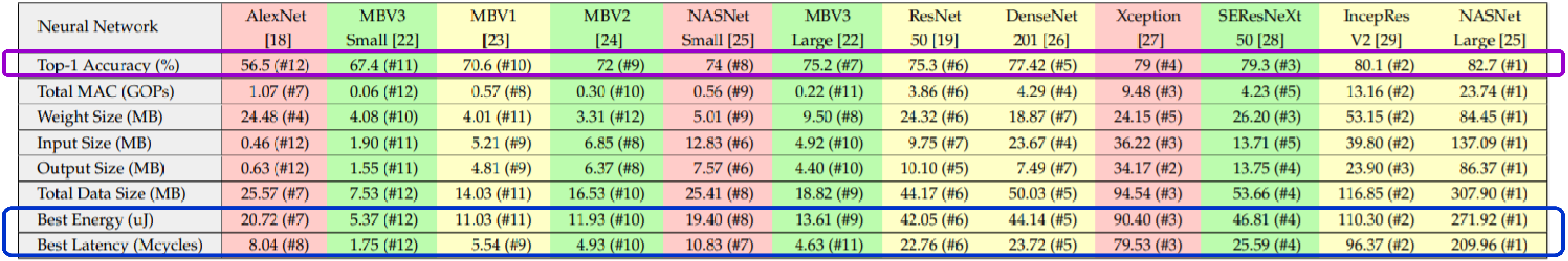
Mem. Size Option	2 B;
	32 B;
	128 B
		8 KB;
		32 KB
			0.5 MB;
			2 MB
				DRAM

Mem. Bandwidth	16 bit/cycle (r/w)	128 bit/cycle (read/write)		

Mem. Share Option (i.e. 1/2/3 operand(s) share same memory)	All separate	All separate;
		Two shared;
		All shared
			All shared	All shared

Mem. Bypass Option	No bypass	Can bypass	Can bypass	No bypass

Comparison on 12 Neural Networks' Algorithm Attribute and Hardware Performance. Weight/Input/Output Size is the accumulated size across all layers, assuming 8-bit precision on ImageNet data. '(#)' indicates value order, from high (#1) to low (#12), across all 12 NNs.



Neural Network	AlexNet [18]	MBV3 Small [22]	MBV1 [23]	MBV2 [24]	NASNet Small [25]	MBV3 Large [22]	ResNet 50 [19]	DenseNet 201 [26]	Xception [27]	SEResNeXt 50 [28]	IncepRes V2 [29]	NASNet Large [25]
Top-1 Accuracy (%)	56.5 (#12)	67.4 (#11)	70.6 (#10)	72 (#9)	74 (#8)	75.2 (#7)	75.3 (#6)	77.42 (#5)	79 (#4)	79.3 (#3)	80.1 (#2)	82.7 (#1)
Total MAC (GOPs)	1.07 (#7)	0.06 (#12)	0.57 (#8)	0.30 (#10)	0.56 (#9)	0.22 (#11)	3.86 (#6)	4.29 (#4)	9.48 (#3)	4.23 (#5)	13.16 (#2)	23.74 (#1)
Weight Size (MB)	24.48 (#4)	4.08 (#10)	4.01 (#11)	3.31 (#12)	5.01 (#9)	9.50 (#8)	24.32 (#6)	18.87 (#7)	24.15 (#5)	26.20 (#3)	53.15 (#2)	84.45 (#1)
Input Size (MB)	0.46 (#12)	1.90 (#11)	5.21 (#9)	6.85 (#8)	12.83 (#6)	4.92 (#10)	9.75 (#7)	23.67 (#4)	36.22 (#3)	13.71 (#5)	39.80 (#2)	137.09 (#1)
Output Size (MB)	0.63 (#12)	1.55 (#11)	4.81 (#9)	6.37 (#8)	7.57 (#6)	4.40 (#10)	10.10 (#5)	7.49 (#7)	34.17 (#2)	13.75 (#4)	23.90 (#3)	86.37 (#1)
Total Data Size (MB)	25.57 (#7)	7.53 (#12)	14.03 (#11)	16.53 (#10)	25.41 (#8)	18.82 (#9)	44.17 (#6)	50.03 (#5)	94.54 (#3)	53.66 (#4)	116.85 (#2)	307.90 (#1)
Best Energy (uJ)	20.72 (#7)	5.37 (#12)	11.03 (#11)	11.93 (#10)	19.40 (#8)	13.61 (#9)	42.05 (#6)	44.14 (#5)	90.40 (#3)	46.81 (#4)	110.30 (#2)	271.92 (#1)
Best Latency (Mcycles)	8.04 (#8)	1.75 (#12)	5.54 (#9)	4.93 (#10)	10.83 (#7)	4.63 (#11)	22.76 (#6)	23.72 (#5)	79.53 (#3)	25.59 (#4)	96.37 (#2)	209.96 (#1)

🟩 Accuracy order (#) < Energy/Latency order (#)

🟨 Accuracy order (#) = Energy/Latency order (#)

🟥 Accuracy order (#) > Energy/Latency order (#)

Assumes all NNs follow layer-by-layer execution (no cross-layer optimization, e.g. depth-first)

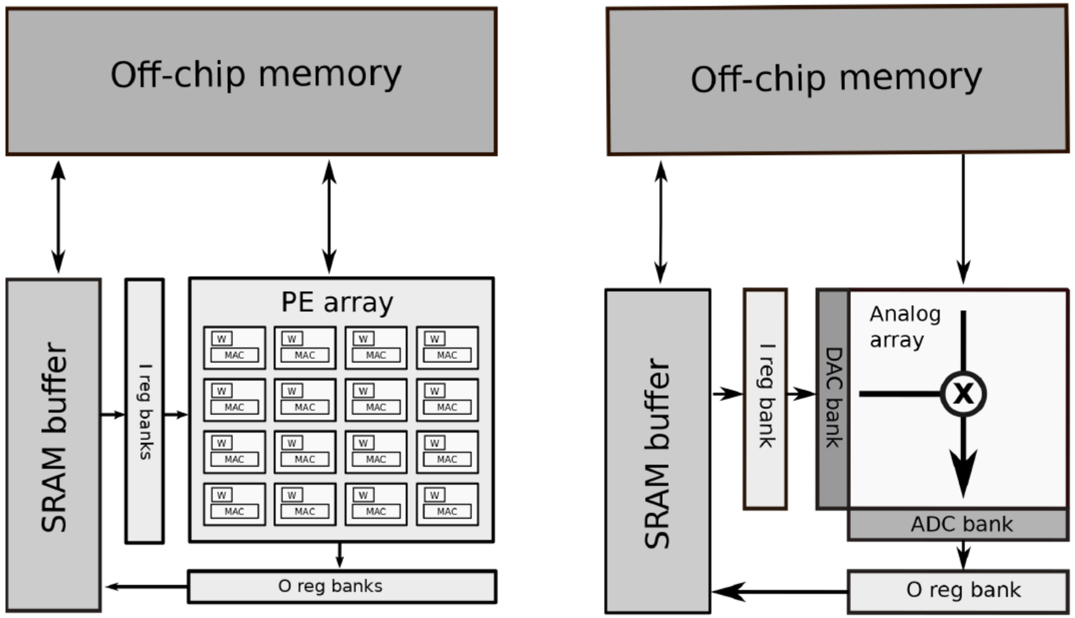
Algorithm accuracy – Energy – Latency trade-off quantification.

Outline

- Introduction
- Methodology
- Result
- Extension
  - AiMC [IEDM 2020]
  - LOMA [AICAS 2021]
- Conclusion & Key Takeaways

Extension

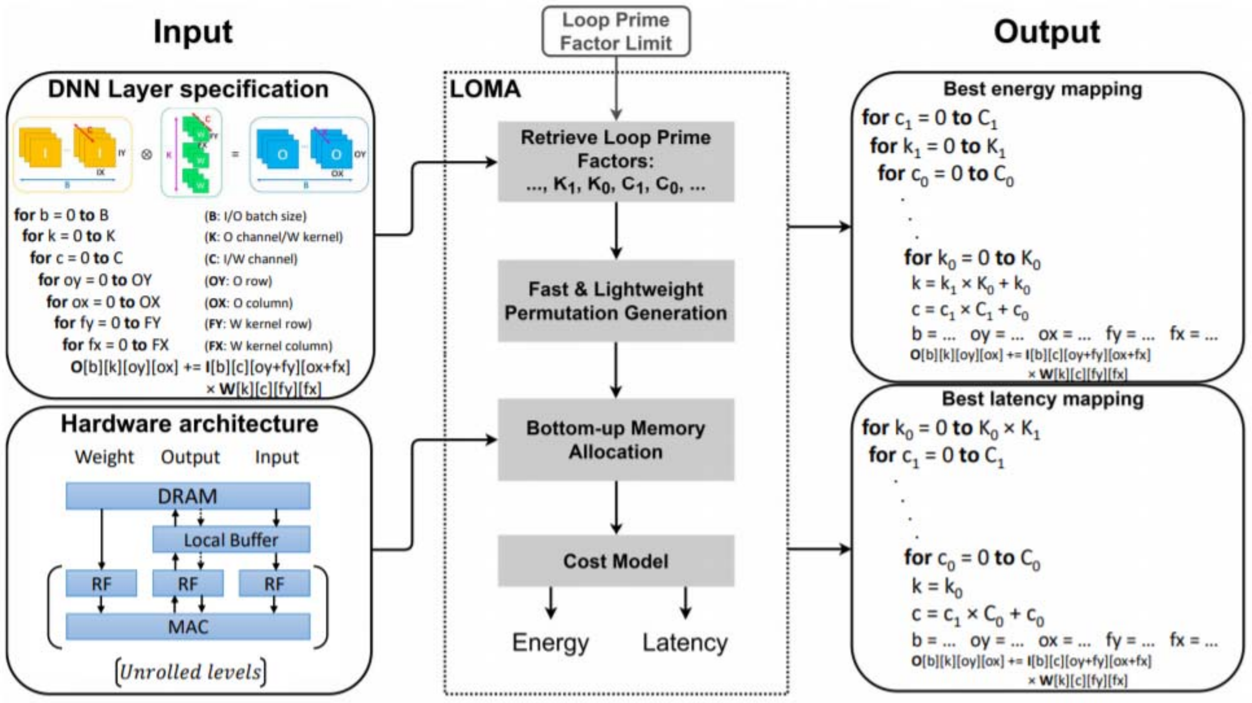
AiMC (Analog-in-Memory Computing) Modeling using ZigZag



Digital Core v.s. Analog Core

Besides focusing on optimizing the efficiency of the AiMC core itself, it is important to also assess/optimize the performance of the AiMC core in the complete processing system.

LOMA (Loop-Order-based Memory Allocation) -- A fast exhaustive temporal mapping search method



```
for b = 0 to B                          (B: I/O batch size)
 for k = 0 to K                         (K: O channel/W kernel)
  for c = 0 to C                        (C: I/W channel)
   for oy = 0 to OY                     (OY: O row)
    for ox = 0 to OX                    (OX: O column)
     for fy = 0 to FY                   (FY: W kernel row)
      for fx = 0 to FX                  (FX: W kernel column)
       O[b][k][oy][ox] += I[b][c][oy+fy][ox+fx]
                          × W[k][c][fy][fx]
```

Best energy mapping

```
for $c_1$ = 0 to $C_1$
 for $k_1$ = 0 to $K_1$
  for $c_0$ = 0 to $C_0$
   .
    .
     .
     for $k_0$ = 0 to $K_0$
      k = $k_1 \times K_0 + k_0$
      c = $c_1 \times C_1 + c_0$
      b = ...  oy = ...  ox = ...  fy = ...  fx = ...
      O[b][k][oy][ox] += I[b][c][oy+fy][ox+fx]
                         × W[k][c][fy][fx]
```

Best latency mapping

```
for $k_0$ = 0 to $K_0 \times K_1$
 for $c_1$ = 0 to $C_1$
  .
   .
    .
     .
     for $c_0$ = 0 to $C_0$
      k = $k_0$
      c = $c_1 \times C_0 + c_0$
      b = ...  oy = ...  ox = ...  fy = ...  fx = ...
      O[b][k][oy][ox] += I[b][c][oy+fy][ox+fx]
                         × W[k][c][fy][fx]
```

By combining an lightweight permutation generator with a bottom-up memory allocation, LOMA executes in near-constant and predictable CPU run-time with a small CPU memory requirement.

Outline

- Introduction
- Methodology
- Result
- Extension
- Conclusion & Key Takeaways

Conclusion & Key Takeaways

- ❑ High-level DSE is important to gain insight from the vast joint DNN-HW-Mapping design space.

- ❑ A general 3-step methodology for building a DNN accelerator DSE framework:

Unify data representation / Standardize cost extraction / Automate design point generation

- ❑ ZigZag, as a fast DSE framework for DNN accelerator, can find better design points due to its uneven mapping support.
- ❑ ZigZag can be applied/extended/improved to/in multiple directions, and we are working on it!

Related Publications

L. Mei, P. Houshmand, V. Jain, S. Giraldo and M. Verhelst, "ZigZag: Enlarging Joint Architecture-Mapping Design Space Exploration for DNN Accelerators," in *IEEE Transactions on Computers (TC)*, doi: 10.1109/TC.2021.3059962.

P. Houshmand, S. Cosemans, L. Mei, I. Papistas, D. Bhattacharjee, P. Debacker, A. Mallik, D. Verkest, and M. Verhelst. "Opportunities and Limitations of Emerging Analog in-Memory Compute DNN Architectures." In *2020 IEEE International Electron Devices Meeting (IEDM)*, pp. 29-1. IEEE, 2020.

V. Jain, L. Mei and M. Verhelst, "Analyzing the Energy-Latency-Area-Accuracy Trade-off Across Contemporary Neural Networks," *2021 IEEE International Conference on Artificial Intelligence Circuits and Systems (AICAS)* (to be present)

A. Symons, L. Mei and M. Verhelst, " LOMA: Fast Auto-Scheduling on DNN Accelerators through Loop-Order-based Memory Allocation," *2021 IEEE International Conference on Artificial Intelligence Circuits and Systems (AICAS)* (to be present)

ZigZag framework is open-source at: https://github.com/ZigZag-Project/zigzag

Premier Sponsor

Intelligent Agent
Neuton

Automated TinyML

Zero-code SaaS solution

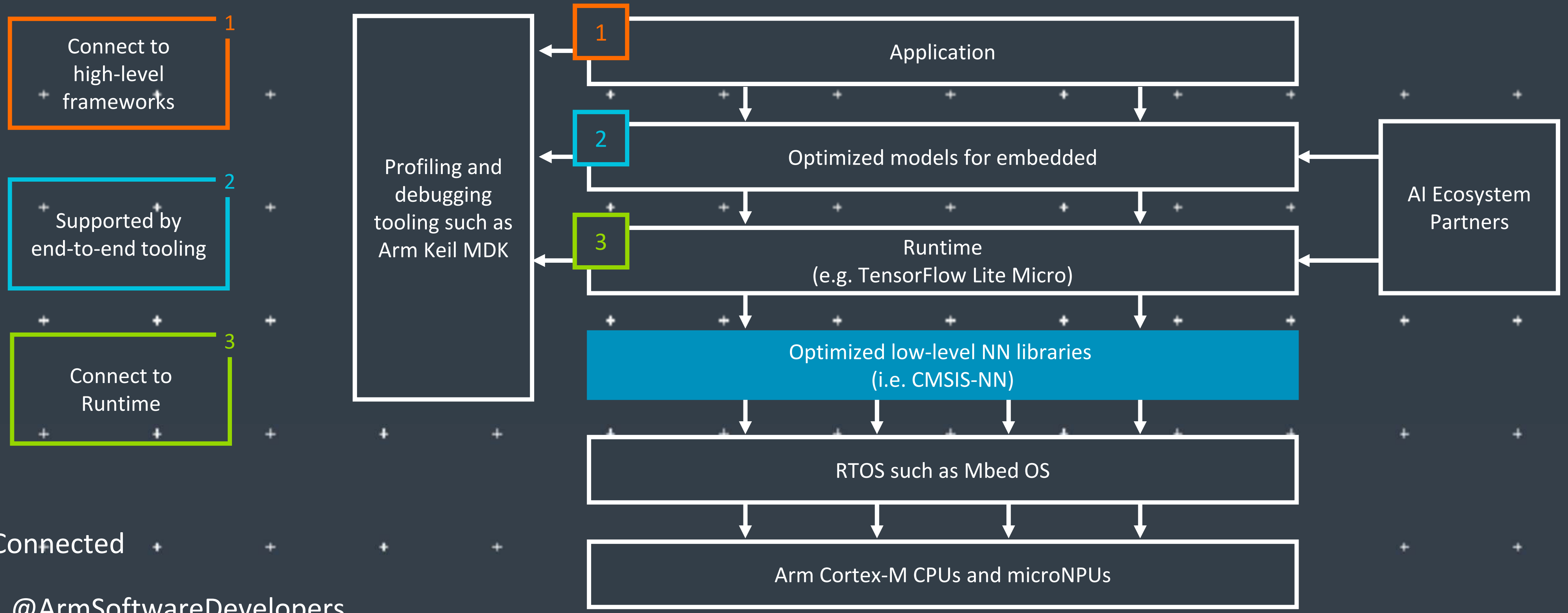
Create tiny models, ready for embedding, in just a few clicks!

Compare the benchmarks of our compact models to those of TensorFlow and other leading neural network frameworks.

Build Fast. Build Once. Never Compromise.

Executive Sponsors

Arm: The Software and Hardware Foundation for tinyML


1
Connect to high-level frameworks
2
Supported by end-to-end tooling
3
Connect to Runtime
Profiling and debugging tooling such as Arm Keil MDK
1
Application
2
Optimized models for embedded
3
Runtime (e.g. TensorFlow Lite Micro)
Optimized low-level NN libraries (i.e. CMSIS-NN)
RTOS such as Mbed OS
Arm Cortex-M CPUs and microNPUs
AI Ecosystem Partners

Stay Connected

@ArmSoftwareDevelopers

@ArmSoftwareDev

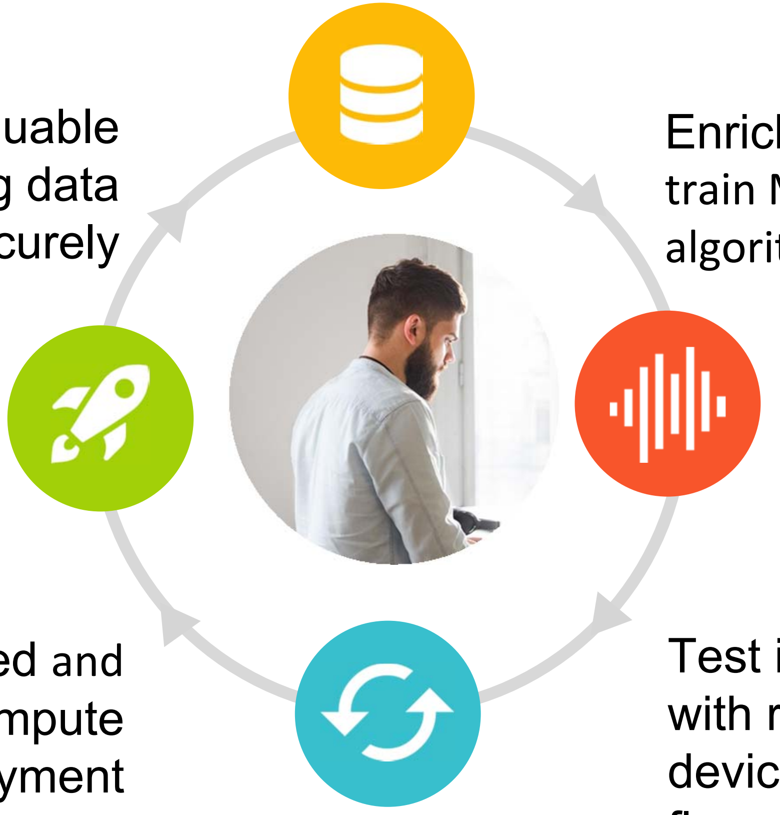
Resources: developer.arm.com/solutions/machine-learning-on-arm

TinyML for all developers

C++ library

Arduino library

WebAssembly



Dataset

Acquire valuable training data securely

Enrich data and train ML algorithms

Impulse

Edge Device

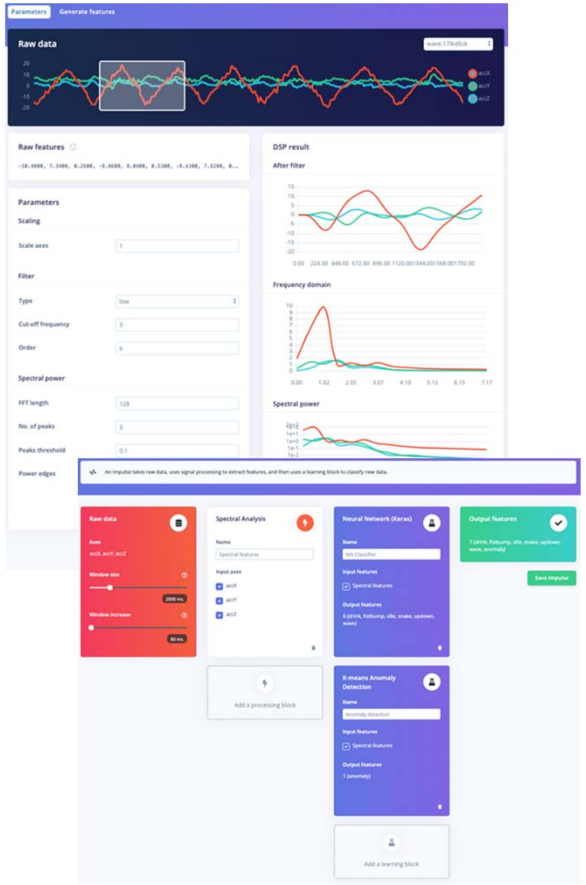
Real sensors in real time

Open source SDK

Embedded and edge compute deployment options

Test

Test impulse with real-time device data flows



Parameters | Generate features

Raw data

wave.17lkd6ck

Raw features

-18.4000, 7.3400, 0.2600, -0.0600, 0.0400, 0.5300, -5.4300, 7.6200, 0...

Parameters

Scaling

Scale axes: 1

Filter

Type: low

Cut-off frequency: 3

Order: 6

Spectral power

FFT length: 128

No. of peaks: 3

Peaks threshold: 0.1

Power edges

DSP result

After filter

Frequency domain

Spectral power

An impulse takes raw data, uses signal processing to extract features, and then uses a learning block to classify new data.

Raw data

Axes: accX, accY, accZ

Window size: 2000 ms.

Window increase: 80 ms.

Spectral Analysis

Name: Spectral features

Input axes: ☑ accX ☑ accY ☑ accZ

Add a processing block

Neural Network (Keras)

Name: NN Classifier

Input features: ☑ Spectral features

Output features: 6 (drink, fistbump, idle, snake, updown, wave)

K-means Anomaly Detection

Name: Anomaly detection

Input features: ☑ Spectral features

Output features: 1 (anomaly)

Add a learning block

Output features

7 (drink, fistbump, idle, snake, updown, wave, anomaly)

Save Impulse

www.edgeimpulse.com

Qualcomm
AI research

Advancing AI research to make efficient AI ubiquitous

Power efficiency

Model design, compression, quantization, algorithms, efficient hardware, software tool

Personalization

Continuous learning, contextual, always-on, privacy-preserved, distributed learning

Efficient learning

Robust learning through minimal data, unsupervised learning, on-device learning

A platform to scale AI across the industry

Perception

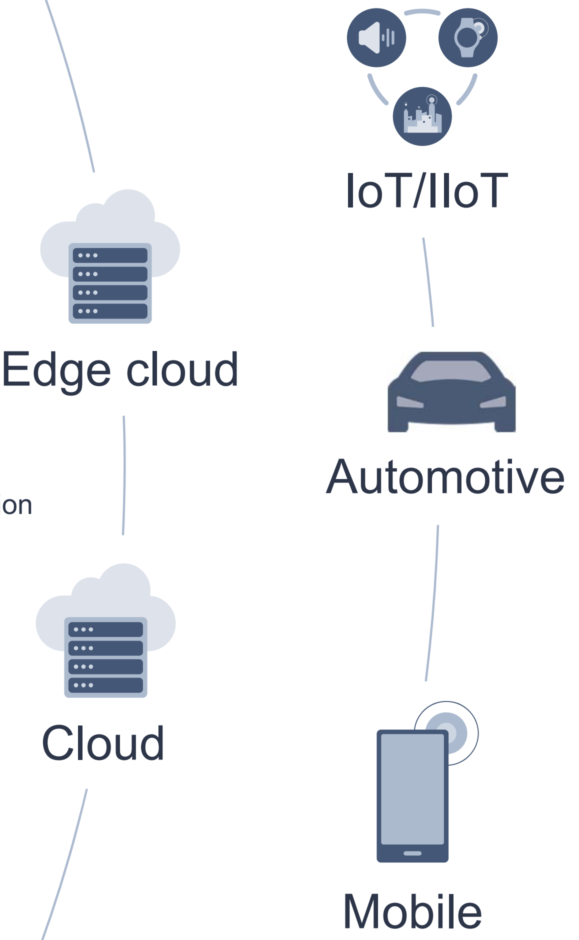
Object detection, speech recognition, contextual fusion

Reasoning

Scene understanding, language understanding, behavior prediction

Action

Reinforcement learning for decision making



Edge cloud

Cloud

IoT/IIoT

Automotive

Mobile

Qualcomm AI Research is an initiative of Qualcomm Technologies, Inc.

SYNTIANT

Syntiant Corp. is moving artificial intelligence and machine learning from the cloud to edge devices. Syntiant's chip solutions merge deep learning with semiconductor design to produce ultra-low-power, high performance, deep neural network processors. These network processors enable always-on applications in battery-powered devices, such as smartphones, smart speakers, earbuds, hearing aids, and laptops. Syntiant's Neural Decision Processors™ offer wake word, command word, and event detection in a chip for always-on voice and sensor applications.

Founded in 2017 and headquartered in Irvine, California, the company is backed by Amazon, Applied Materials, Atlantic Bridge Capital, Bosch, Intel Capital, Microsoft, Motorola, and others. Syntiant was recently named a CES® 2021 Best of Innovation Awards Honoree, shipped over 10M units worldwide, and unveiled the NDP120 part of the NDP10x family of inference engines for low-power applications.

www.syntiant.com

@Syntiantcorp

Platinum Sponsors

Part of your life. Part of tomorrow.

www.infineon.com

Reality AI®

Add Advanced Sensing to your Product with Edge AI / TinyML

https://reality.ai ✉ info@reality.ai @SensorAI Reality AI

Pre-built Edge AI sensing modules, plus tools to build your own

Reality AI solutions

Prebuilt sound recognition models for indoor and outdoor use cases

Solution for industrial anomaly detection

Pre-built automotive solution that lets cars "see with sound"

Reality AI Tools® software

Build prototypes, then turn them into real products

Explain ML models and relate the function to the physics

Optimize the hardware, including sensor selection and placement

TINY ML EMEA
Virtual
JUNE 7-10, 2021

Gold Sponsors

LatentAI

Adaptive AI for the Intelligent Edge

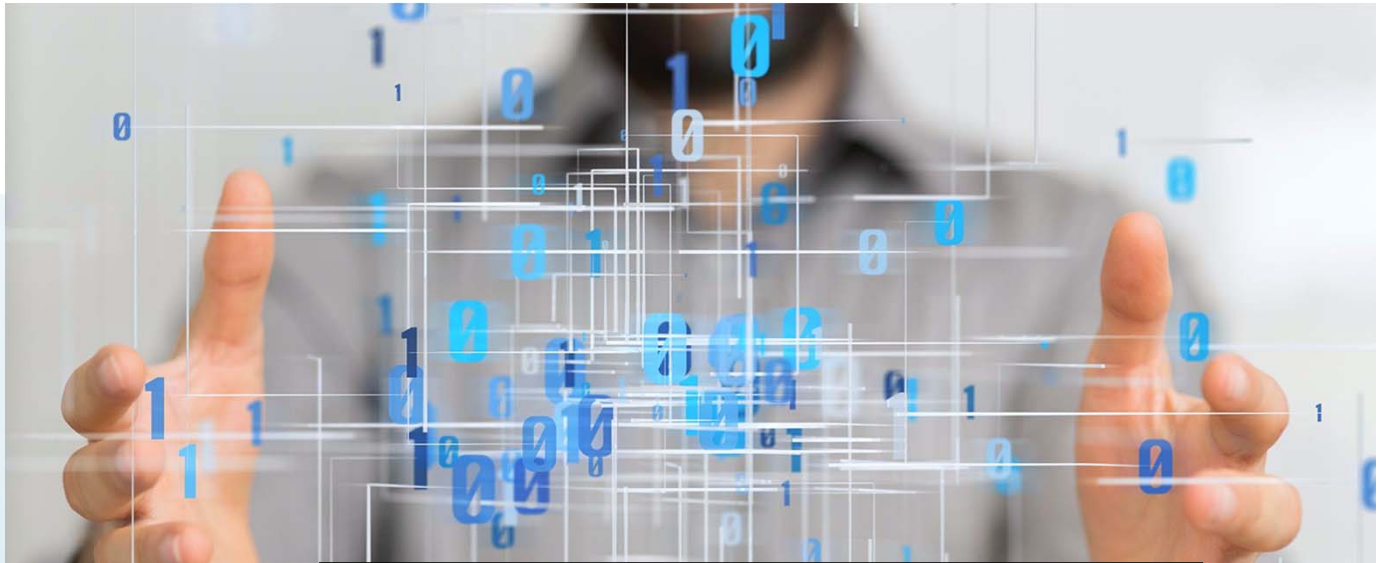
Latentai.com

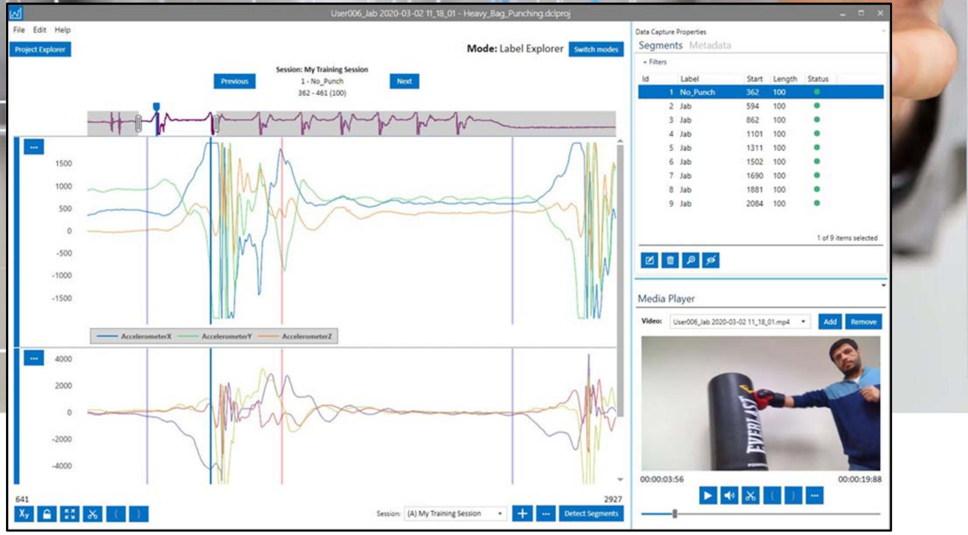
Build Smart IoT Sensor Devices From Data

SensiML pioneered TinyML software tools that auto generate AI code for the intelligent edge.

- End-to-end AI workflow
- Multi-user auto-labeling of time-series data
- Code transparency and customization at each step in the pipeline

We enable the creation of production-grade smart sensor devices.





User006_Jab 2020-03-02 11_18_01 - Heavy_Bag_Punching.dclproj

Mode: Label Explorer

Session: My Training Session
1 - No_Punch
362 - 461 (100)

641
2927

Session: (A) My Training Session

Data Capture Properties

Segments Metadata

Filters

Id	Label	Start	Length	Status
1	No_Punch	362	100	●
2	Jab	594	100	●
3	Jab	862	100	●
4	Jab	1101	100	●
5	Jab	1311	100	●
6	Jab	1502	100	●
7	Jab	1690	100	●
8	Jab	1881	100	●
9	Jab	2084	100	●

1 of 9 items selected

Media Player

Video: User006_Jab 2020-03-02 11_18_01.mp4

00:00:03:56
00:00:19:88

sensiml.com

Silver Sponsors

emza
visual sense

GREENWAVES
TECHNOLOGIES

HOTG

imagimob

Qeexo

seeed
studio
The IoT Hardware Enabler

ST
life.augmented

TINY ML EMEA
Virtual
JUNE 7-10,2021

Copyright Notice