A 772µJ/frame ImageNet Feature Extractor Accelerator on HD Images at 30FPS

Ivan Miro-Panades, Vincent Lorrain, Lilian Billod, Inna Kucher, Vincent Templier, Sylvain Choisnet, Nermine Ali, Baptiste Rossigneux, Olivier Bichler, Alexandre Valentian

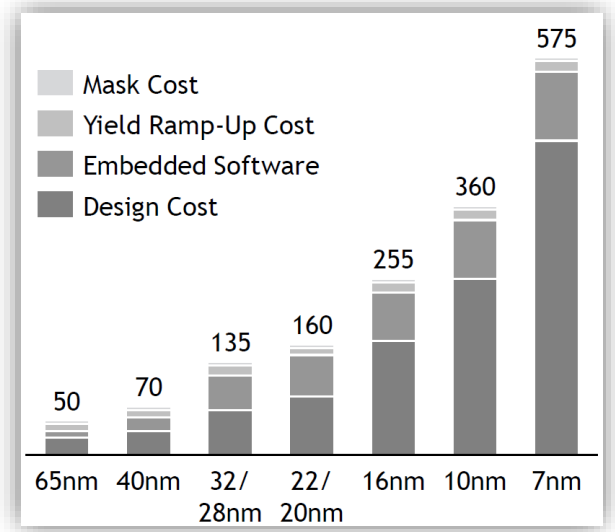
APCCAS, November 7-9, 2024

Outline

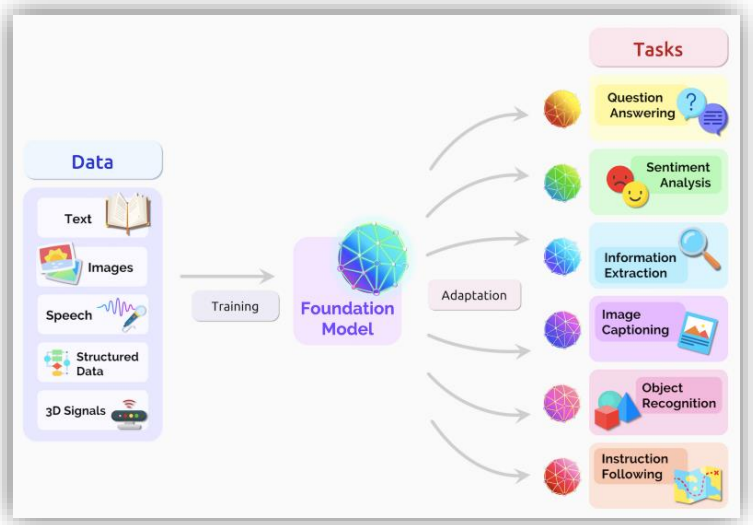
- Introduction and Challenges
- NeuroCorgi
  - Architecture
  - Computing modes
  - Semantic segmentation use case
- NeuroCorgi ImageNet measures
- Conclusion

Introduction

- AI is everywhere !
- Embedding AI models in edge devices is challenging
- Advance node technology required to meet the target energy efficiency
  - However, design and fabrication cost are too high on these advanced nodes
- Foundation Models are emerging
  - Trained with a massive amount of data
  - Adaptation (transfer learning) to multiple tasks
  - The backbone remains fixed while last layers are tuned

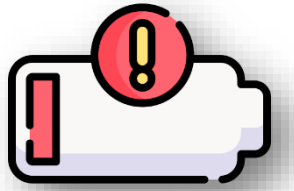


Source: Yunsup Lee, Andrew Waterman, VLSI 2020



Source: https://arxiv.org/abs/2108.07258

Facing challenges on edge devices



Low power operation

- Battery powered devices
- Energy efficiency is an imperative

High frame rate

- Detect and track fast moving objects

Low processing latency

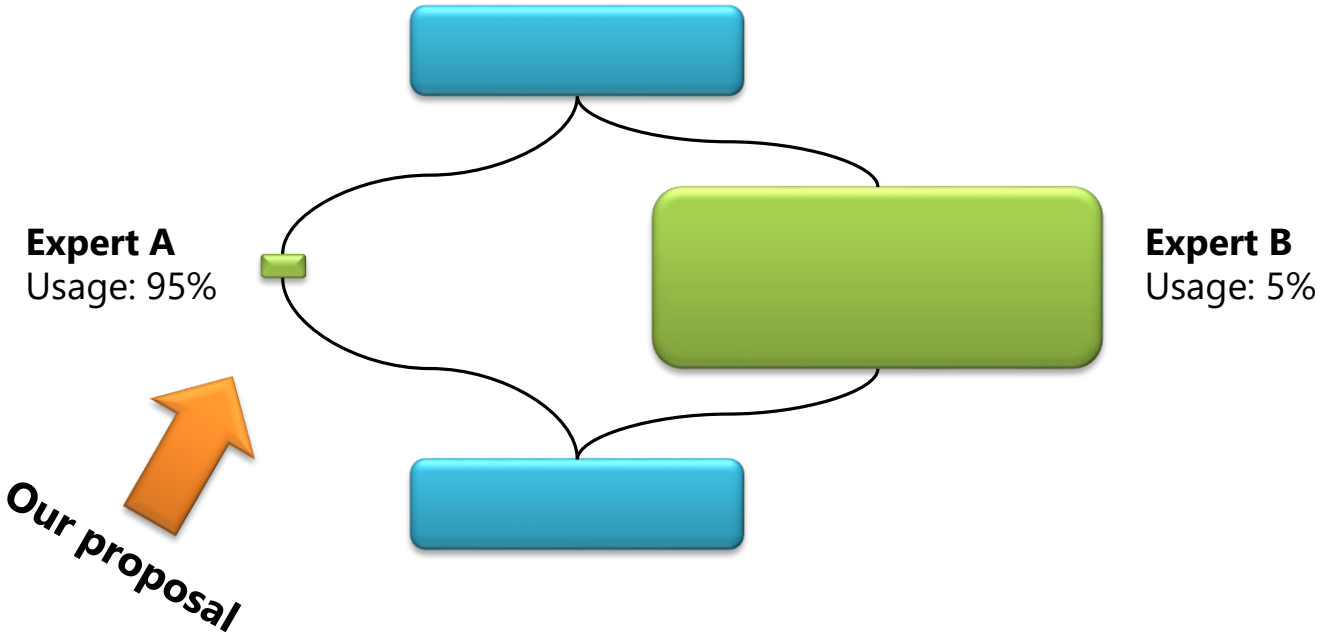
- Batch size of 1 become mandatory

High resolution images

- Moving from 224x224 images to HD images

Occam's razor

- Should I always use a large and energy-consuming model for all inferences?
  - Probably not!
  - For 95% of the cases, a low-energy expert model may obtain the same results
  - Combine different models to manage accuracy and energy efficiency tradeoff

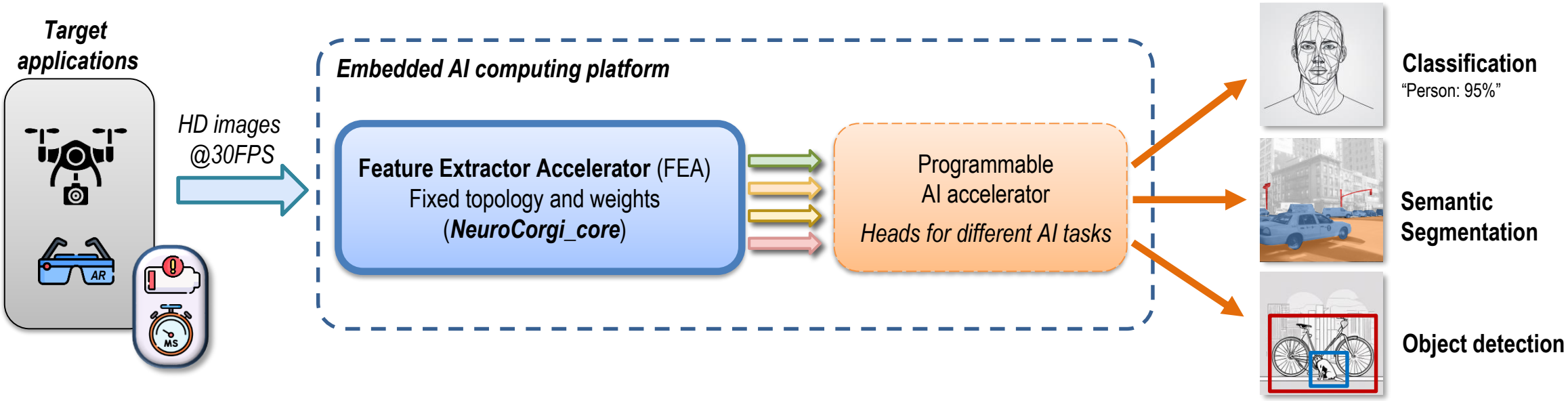


Outline

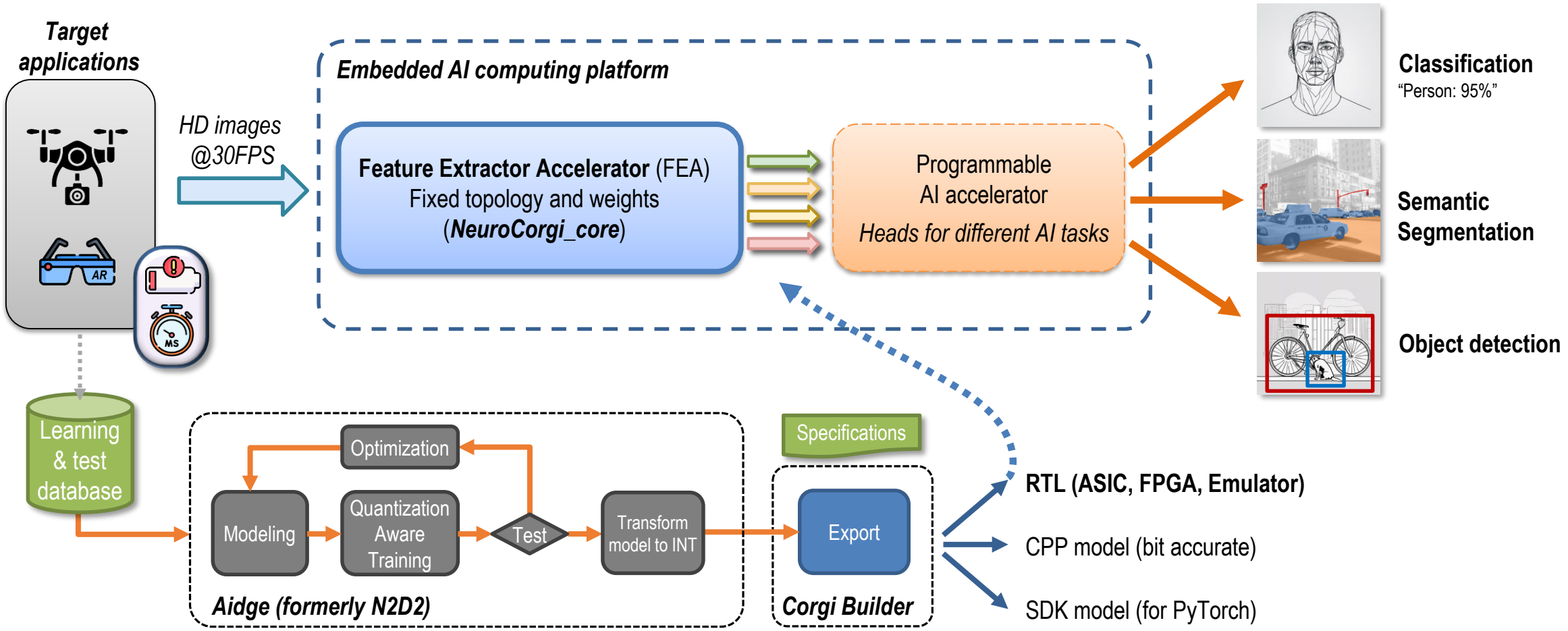
- Introduction and Challenges
- NeuroCorgi
  - Architecture
  - Computing modes
  - Semantic segmentation use case
- NeuroCorgi ImageNet measures
- Conclusion

NeuroCorgi introduction

- A high energy efficiency Feature Extractor Accelerator (FEA)
- Supporting up-to HD images at 30FPS with reduced processing latency
- By integrating an external *Programmable AI accelerator*, the system can compute specific AI tasks based on FEA results
  - The *Programmable AI accelerator* is not included in current NeuroCorgi circuit
- As in Foundation models, the FEA part is pre-trained and fixed at design-time
- FEA uses a lightweight backbone to obtain high energy efficiency and low latency

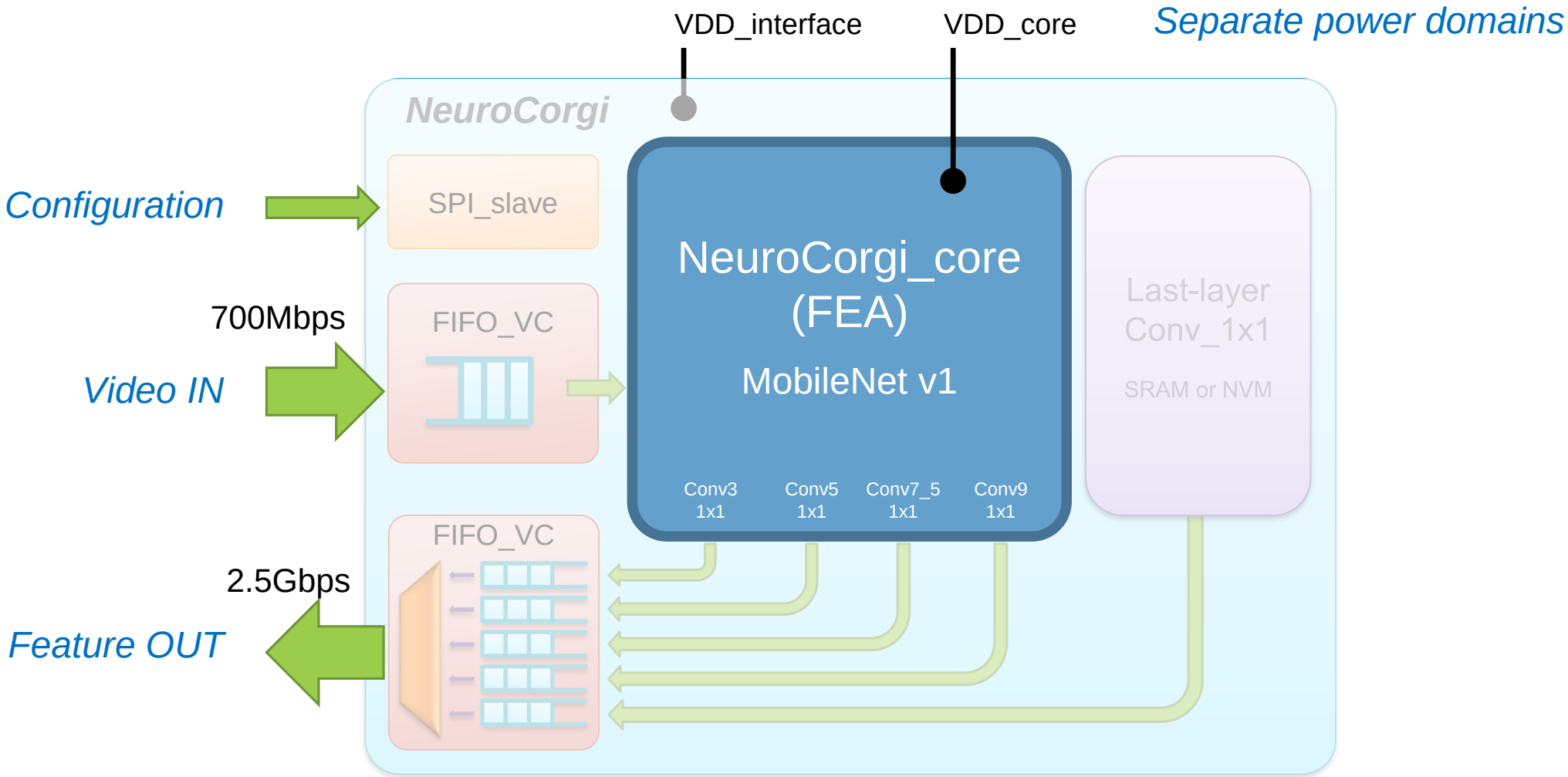


NeuroCorgi introduction



- NeuroCorgi comes with a set of tools to train and quantize a model and generate a new RTL

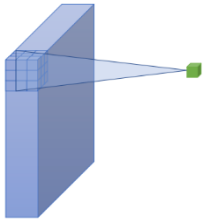
NeuroCorgi circuit



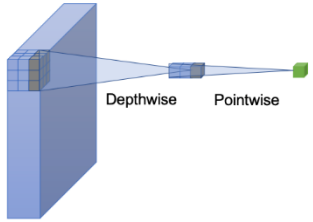
No external memory for activations and weights
Only Video IN and Feature OUT

Network topology

- NeuroCorgi backbone uses MobileNet v1 topology
  - Tradeoff between network complexity and operations per inference
  - Uses Depth-Wise (DW) and Point-Wise (PW) convolutions to reduce the computing complexity
    - Lower MACs per images
    - Lower energy per inference

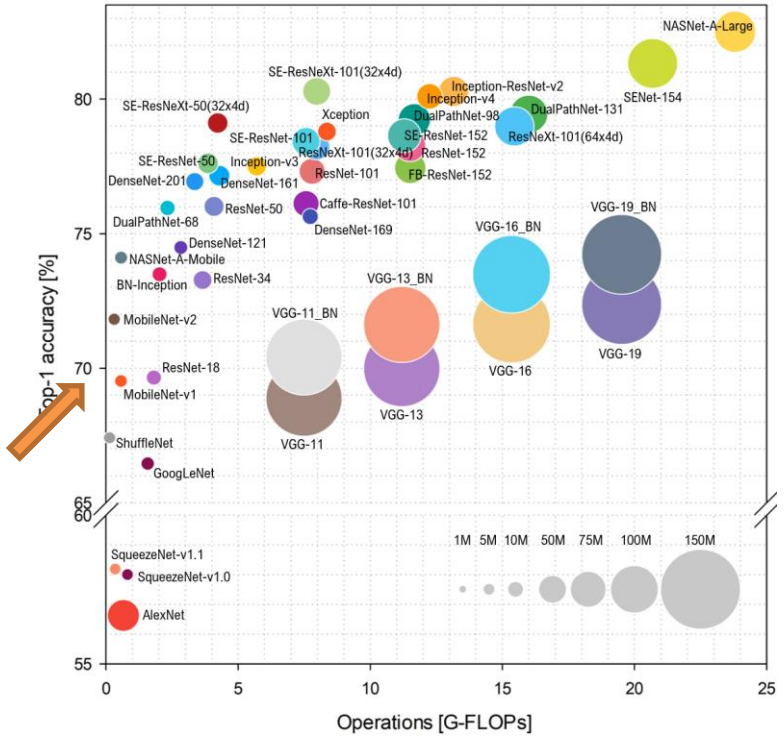


Standard convolution



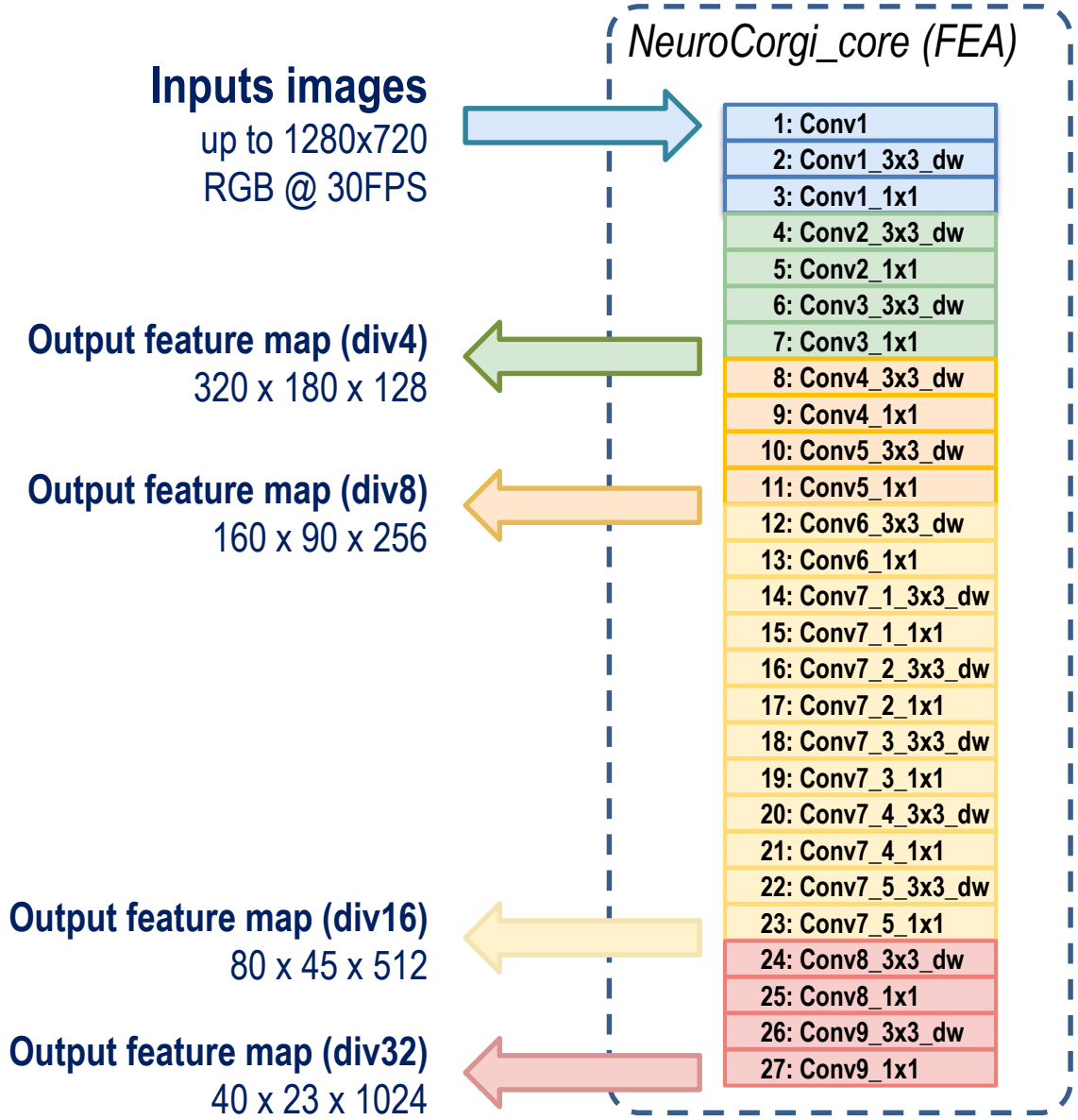
Depth-wise (DW) + Point-wise (PW) convolutions

- Leverage on **fixed topology** and **fixed weights**
  - Fixed topology optimizes the buffering and the inter-layer communication throughput
  - Fixed weights allows to fix within the ASIC the weight values
    - A ROM of weights can be optimized at design time



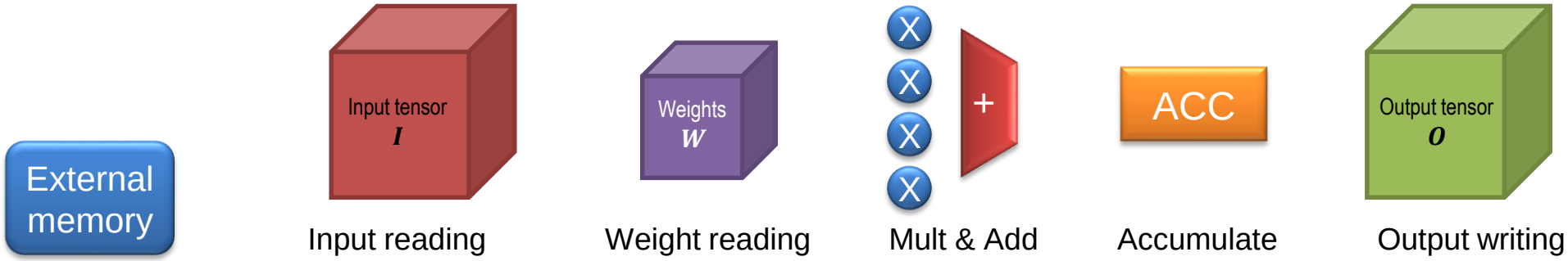
S. Bianco, "Benchmark Analysis of Representative Deep Neural Network Architectures,"

NeuroCorgi layers

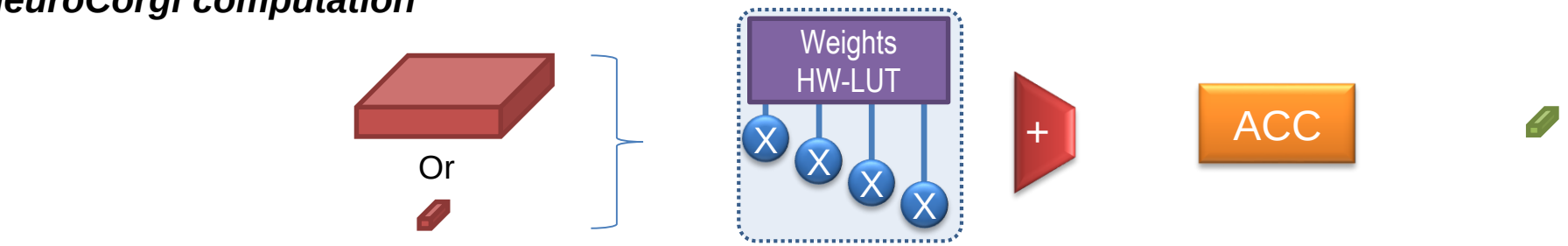


NeuroCorgi computing approach

Conventional DNN computation (layer-wise)



NeuroCorgi computation



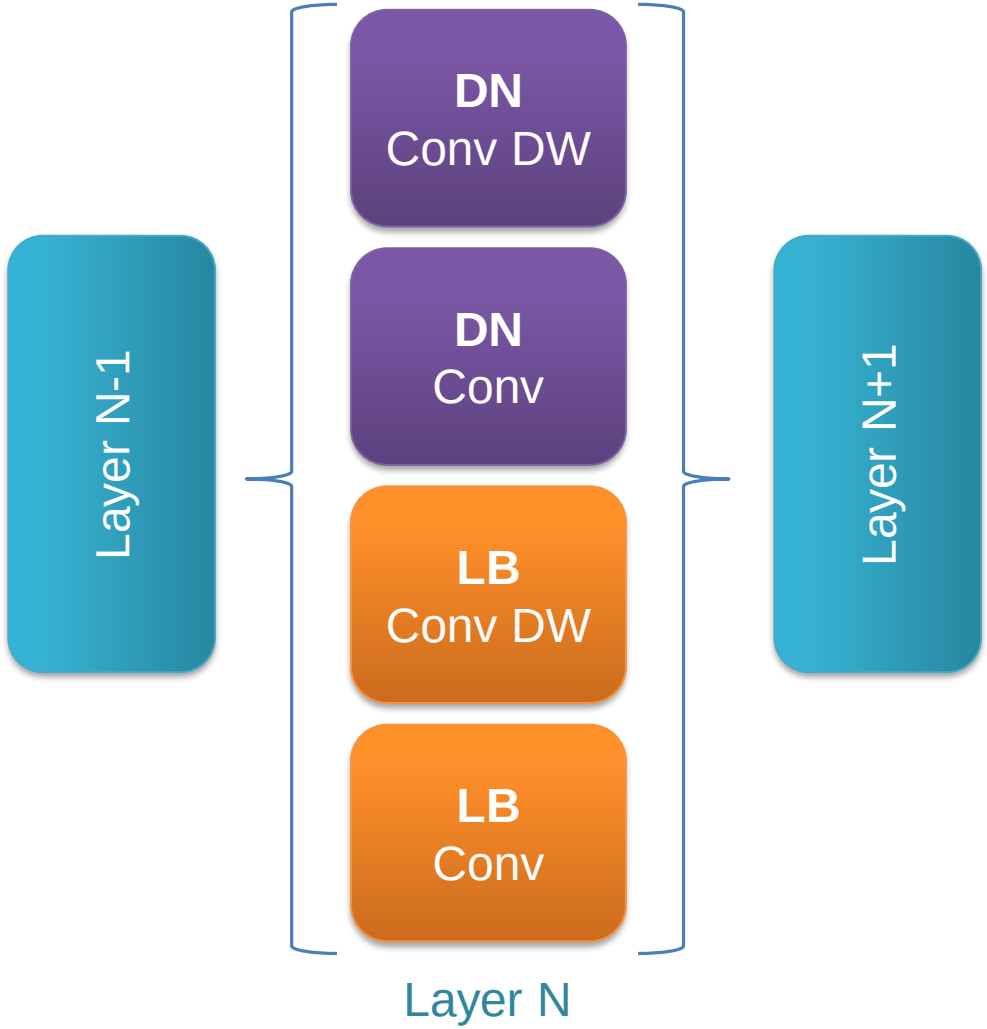
External memory

Streaming approach
Reduced input data
Low storage area
Lower reading power

Multi-constant multiplier (MCM)
Optimized multipliers and add tree
at design time
Lower reading power
Lower computing power

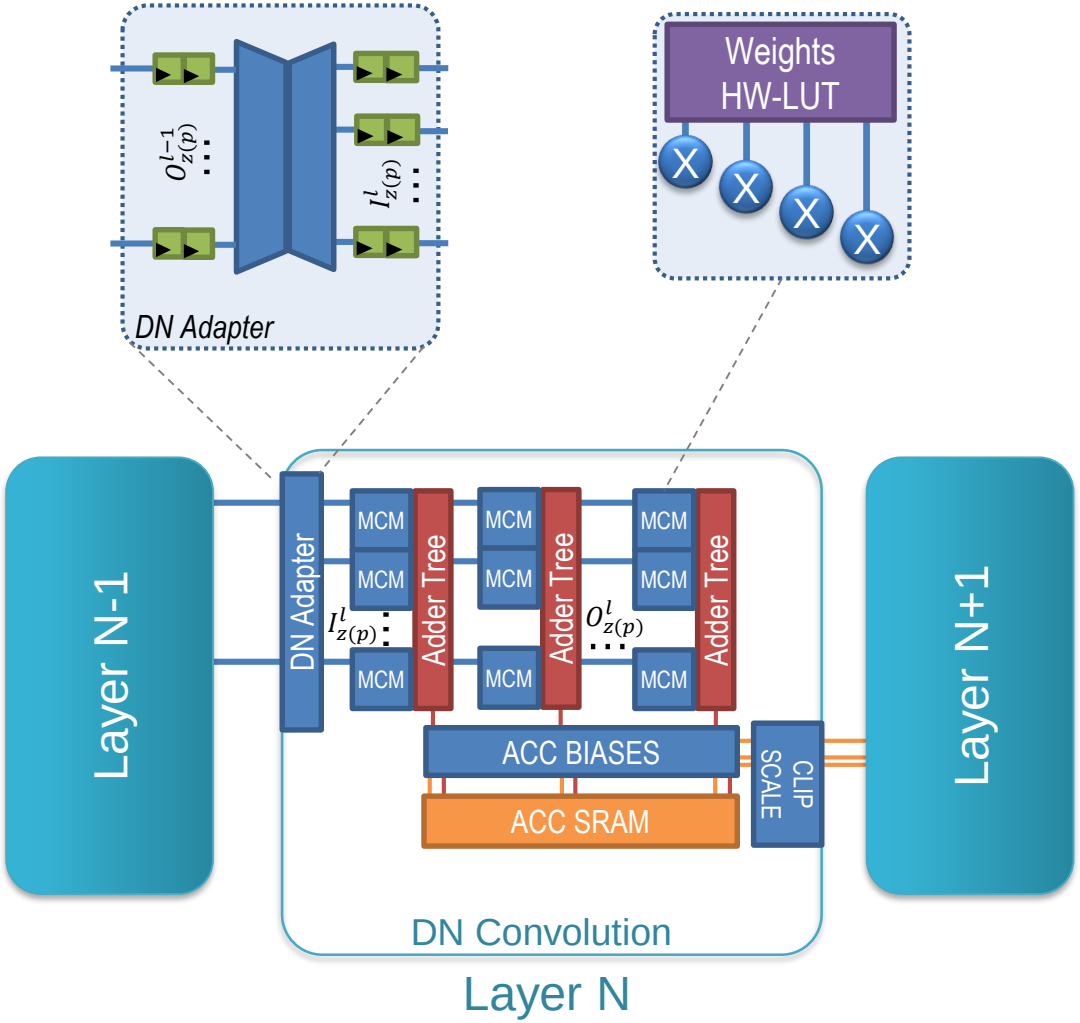
Reduced output data
No buffering between layers
Streaming approach
Lower writing power

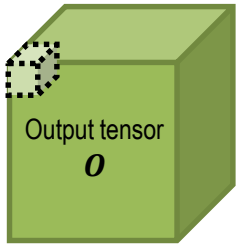
Compute modes



- Streaming architecture with specialized compute modules
- Four computation modes for convolutions
  - DN and LineBuffer (LB) approaches
  - 3D and Depth-Wise(DW) Convolutions
- Stride and padding supported
- Optimized streaming interfaces
  - FIFO-less, only elastic registers
  - Clock frequency adapted to the layer performance

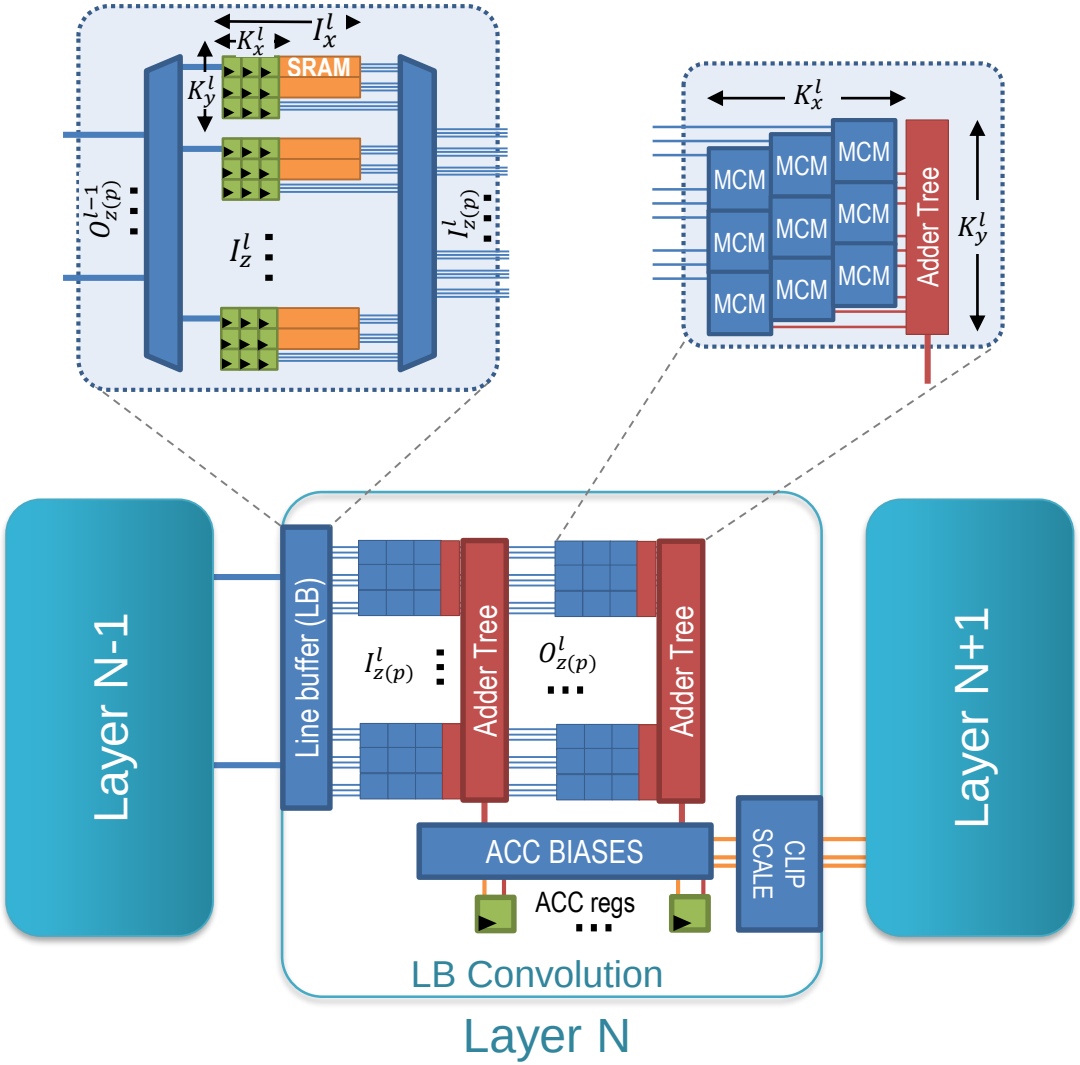
DN Convolution





- Input tensor is used to compute partial accumulations
- Area efficient architecture
  - Very few registers and low wire routing
  - No input FIFO. Inputs are directly used
  - ACC data in SRAM
- Energy efficiency
  - High energy efficiency thanks to MCM
  - But high number of ACC operations due to partial accumulations

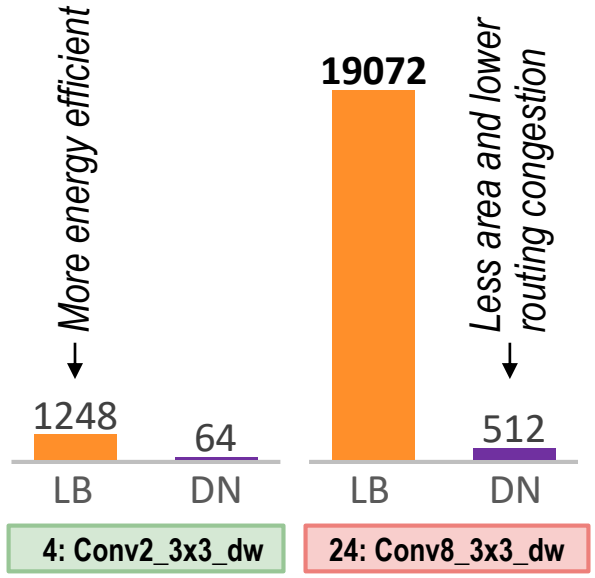
LB Convolution



- Store input tensor and then compute kernel in parallel
- Energy efficient architecture
  - LB is more energy efficient than DN
  - Convolution is computed as parallel as possible
  - Input data is reused on multiple convolutions
  - Tradeoff between convolution computation and input reuse
  - Optimized MCM increases the energy efficiency
- Line buffer requires N-1 lines of input features
  - E.g. 2 lines of input activations on a 3x3 convolution
  - Uses SRAM memory for density and registers for parallel read

Compute modes selection

Layer	Compute mode
1: Conv1	LB Conv
2: Conv1_3x3_dw	LB Conv DW
3: Conv1_1x1	LB Conv
4: Conv2_3x3_dw	LB Conv DW
5: Conv2_1x1	LB Conv
6: Conv3_3x3_dw	LB Conv DW
7: Conv3_1x1	LB Conv
8: Conv4_3x3_dw	LB Conv DW
9: Conv4_1x1	LB Conv
10: Conv5_3x3_dw	LB Conv DW
11: Conv5_1x1	LB Conv
12: Conv6_3x3_dw	DN Conv DW
13: Conv6_1x1	LB Conv
14: Conv7_1_3x3_dw	DN Conv DW
15: Conv7_1_1x1	LB Conv
16: Conv7_2_3x3_dw	DN Conv DW
17: Conv7_2_1x1	LB Conv
18: Conv7_3_3x3_dw	DN Conv DW
19: Conv7_3_1x1	LB Conv
20: Conv7_4_3x3_dw	DN Conv DW
21: Conv7_4_1x1	LB Conv
22: Conv7_5_3x3_dw	DN Conv DW
23: Conv7_5_1x1	LB Conv
24: Conv8_3x3_dw	DN Conv DW
25: Conv8_1x1	LB Conv
26: Conv9_3x3_dw	DN Conv DW
27: Conv9_1x1	LB Conv



Number of registers using LB or DN approaches in function of the layer

- LB is more energy efficient than DN
- However LB suffers from wire congestion on 3D and DW convolutions
  - For layers with many channels, the number of registers is too high
- Routing congestion is not an issue on point-wise layers

Quantization results on ImageNet

MobileNet v1 topology on ImageNet

- Floating point model (original paper)
  - Accuracy: 70.6%

- 4bit quantized model (first layer on 8bits)
  - Accuracy: 70.54%
  - NeuroCorgi embeds this model

aidge https://projects.eclipse.org/projects/technology.aidge

Semantic segmentation use case

Image segmentation

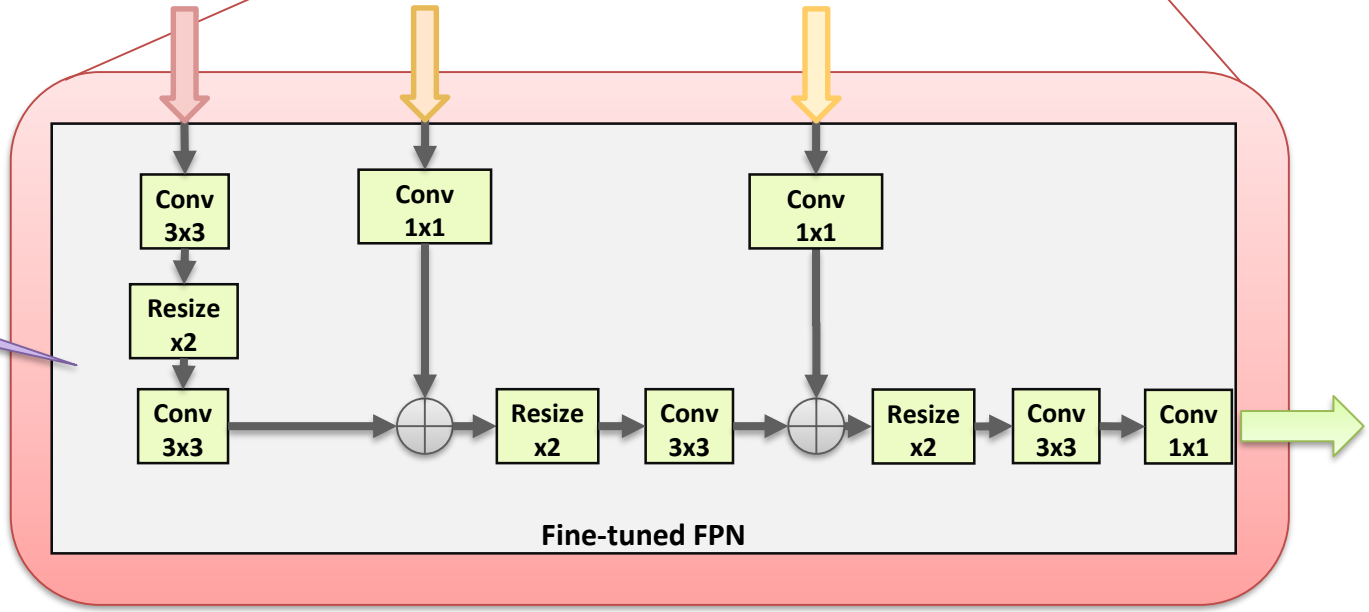
NeuroCorgi

96.7% MAC/frame

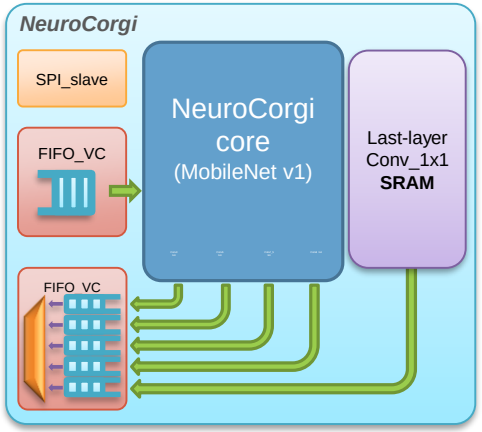
Programmable AI accelerator

3.3% MAC/frame

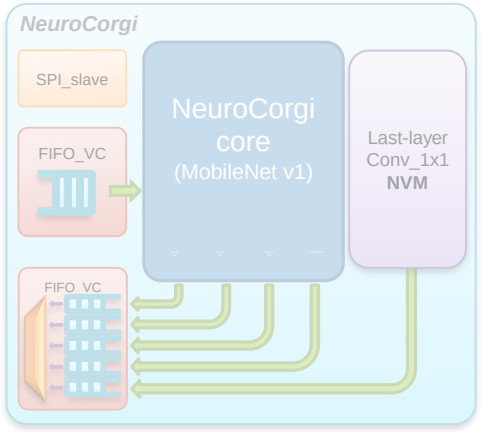
Computing complexity:
314M MAC/frame
3.3% of total MAC/frame
mIoU: 73.3% on Cityscapes



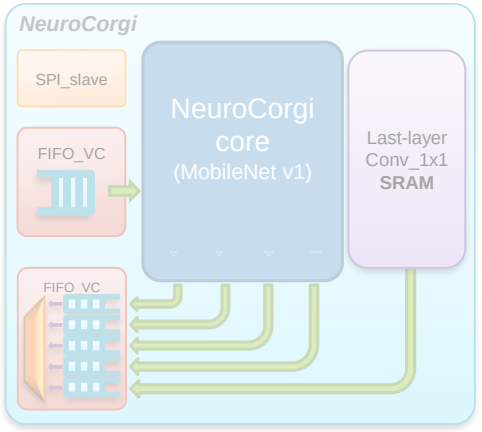
NeuroCorgi implementation variants



ImageNet database
Last Layer: SRAM



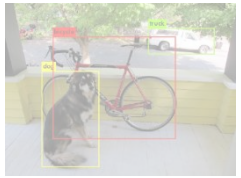
ImageNet database
Last Layer: NVM



COCO database
Last Layer: SRAM

Classification and Semantic segmentation tasks

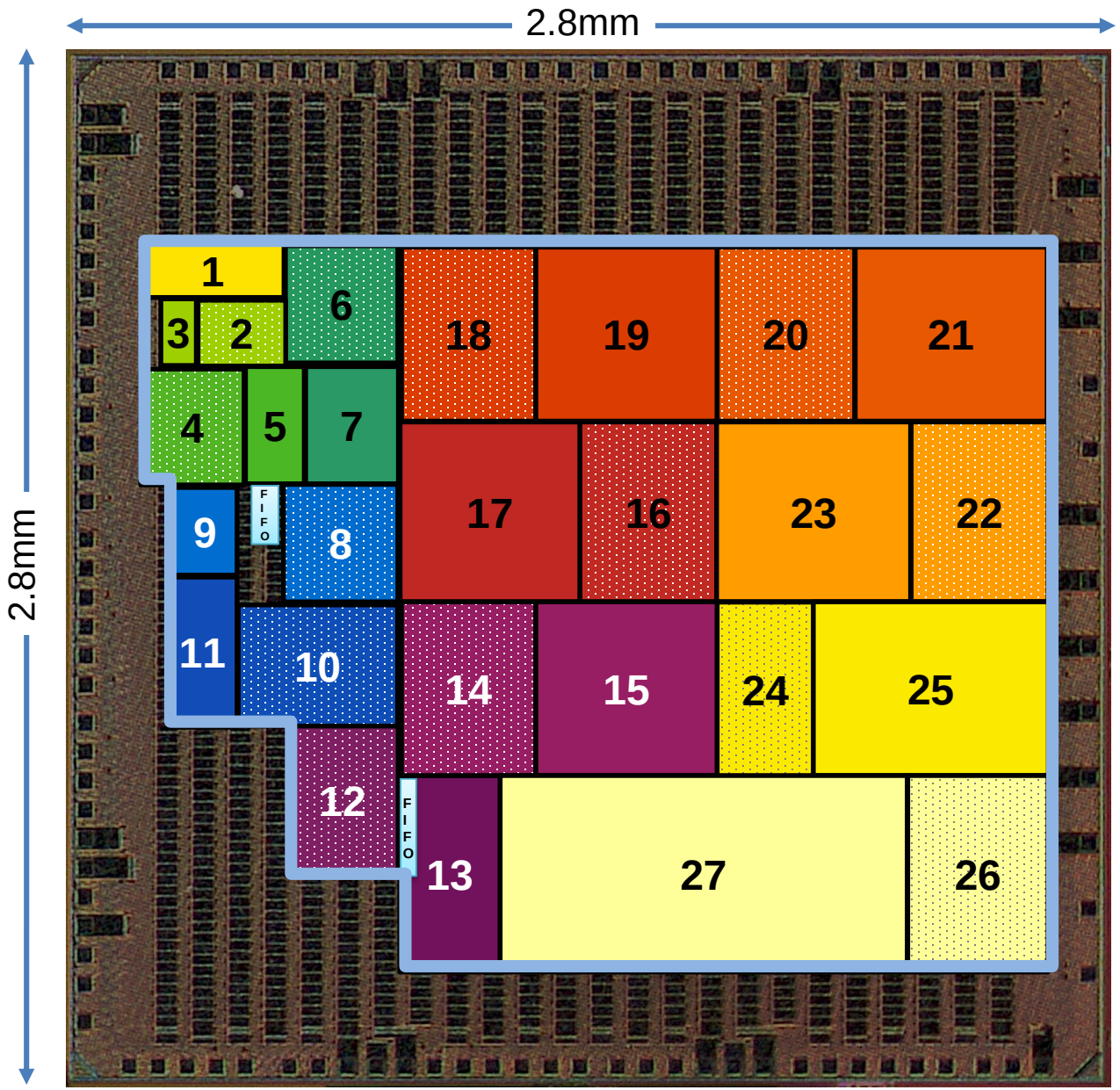
Object detection tasks



Outline

- Introduction and Challenges
- NeuroCorgi
  - Architecture
  - Computing modes
  - Semantic segmentation use case
- NeuroCorgi ImageNet measures
- Conclusion

NeuroCorgi ImageNet



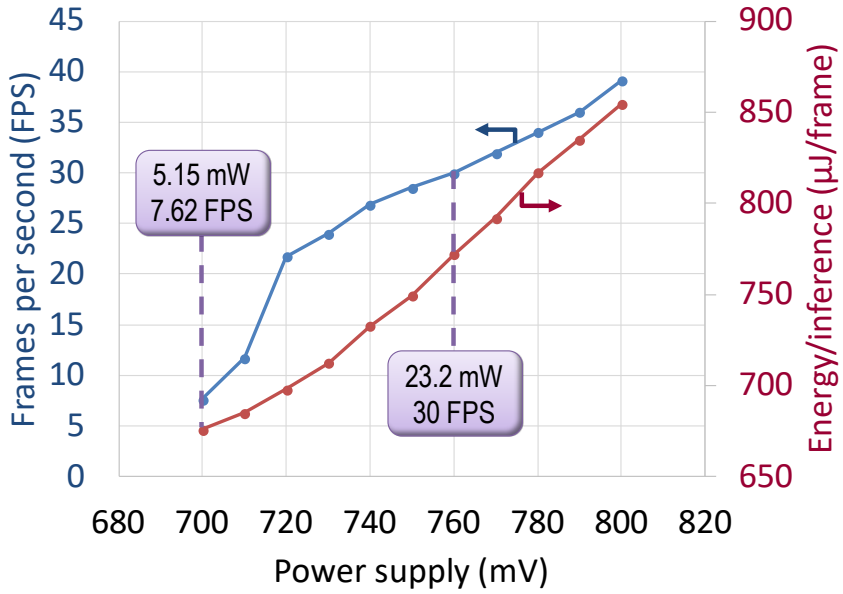
1: Conv1
2: Conv1_3x3_dw
3: Conv1_1x1
4: Conv2_3x3_dw
5: Conv2_1x1
6: Conv3_3x3_dw
7: Conv3_1x1
8: Conv4_3x3_dw
9: Conv4_1x1
10: Conv5_3x3_dw
11: Conv5_1x1
12: Conv6_3x3_dw
13: Conv6_1x1
14: Conv7_1_3x3_dw
15: Conv7_1_1x1
16: Conv7_2_3x3_dw
17: Conv7_2_1x1
18: Conv7_3_3x3_dw
19: Conv7_3_1x1
20: Conv7_4_3x3_dw
21: Conv7_4_1x1
22: Conv7_5_3x3_dw
23: Conv7_5_1x1
24: Conv8_3x3_dw
25: Conv8_1x1
26: Conv9_3x3_dw
27: Conv9_1x1

Chip summary	
Technology	GF 22FDX
Chip area	7.86mm²
FEA area	4.45mm²
# multipliers	42k
# SRAM memories	186
SRAM memory	1.1MB
Main clock	59MHz
AI model	MobileNet v1
Training dataset	ImageNet
Batch size	1

NeuroCorgi ImageNet - Energy efficiency

On HD images (1280x720) at 30FPS, 0.76V

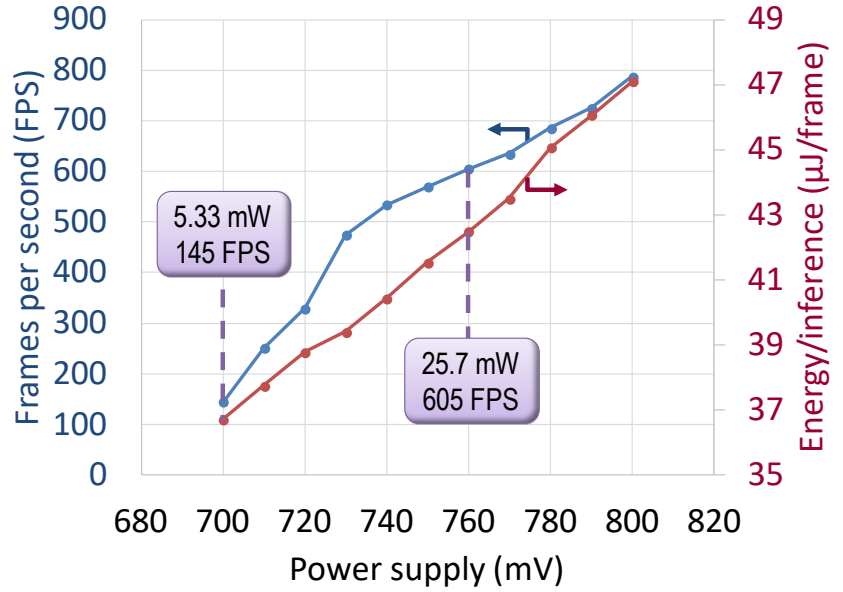
- Main clock 59MHz
- 23.2mW @ 30FPS
- 772µJ/frame
- 837pJ/pixel/frame
- Leakage 0.96%: 224µW



An HD camera @ 30FPS consumes ~100mW

On 224x224 images at 0.76V

- Main clock 59MHz
- 25.7mW @ 605 FPS
- 42.4µJ/frame
- 846pJ/pixel/frame
- Leakage 0.88%: 224µW



1.5mW for 224x224 @30FPS

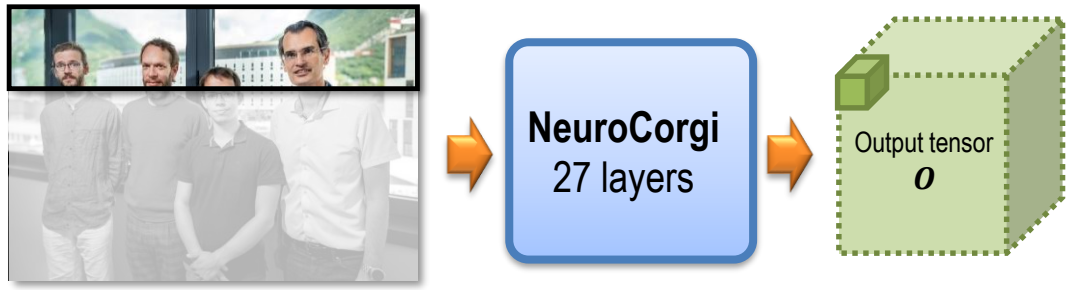
Processing latency

On HD images @ 59MHz

Output	Latency (µs)	% of image
Conv3_1x1	391	1.17%
Conv5_1x1	916	2.75%
Conv7_5_1x1	4790	14.37%
Conv9_1x1	6902	20.71%

On 224x224 images @ 59MHz

Output	Latency (µs)	% of image
Conv3_1x1	69	4.06%
Conv5_1x1	166	9.78%
Conv7_5_1x1	893	52.60%
Conv9_1x1	1288	75.86%



- The first features are generated with only 21% of the input image
- 6.9ms latency on HD mages at 30FPS

Comparison with SoA

			ISSCC'20 Y. Jiao et al.	JSSC'23 J. -S. Park et al.	JSSC'23 DIANA	JSSC'24 Marsellus	JSSC'24 DynaPlasia	ISSCC'22 Hiddenite	This work NeuroCorgi	
Technology			12nm	4nm	22nm	22nm FDX	28nm	40nm	22nm FDX	
Application			Server	Mobile	Edge	AI-IoT	Embedded	Embedded	Embedded	
Area (mm²)			709 (chip)	4.74 (core)	3.3 (core)	18.7 (chip) 2.42 (core)	20.25 (chip)	9 (chip) 4.36 (core)	7.86 (chip) 4.45 (FEA)	
Programmability			Programmable	Programmable	Programmable	Programmable	Programmable	Configurable	Fixed	
Power consumption (mW)			25W – 276W	381 – 5133	–	12.8 – 123	261	85.4 – 534.7	5 – 37	
ImageNet use case	Training dataset		ImageNet	ImageNet	ImageNet	ImageNet	ImageNet	ImageNet	ImageNet	
	AI model		ResNet50 v1	MobileNet TPU	ResNet18	ResNet18	ResNet18	ResNet50	MobileNet v1	
	Precision (bits)		8	8	Analog + digital	RBE 4x4b	9w, 8a	ternary(w), 8a	4w, 4a	
	Top-1 accuracy (%)		74.93	–	64.1	68.5	70.4	70.09	**70.42**	
	Inferences/second (FPS)	224x224	78563	3433	277	20.8	776$^{\gamma}$	169.7$^{\alpha}$	**788**	
		1280x720	–	–	–	–	–	–	**39**	
	FEA latency (ms)	224x224	0.2$^{\beta\delta}$	0.29$^{\delta}$	3.6$^{\delta}$	48$^{\delta}$	1.29$^{\gamma\delta}$	5.92$^{\alpha\delta}$	**1.23**	
		1280x720	–	–	–	–	–	–	**6.90**	
	TOPS/W	224x224	4.14	11.59	5.52$^{\alpha}$	5.83	10.8$^{\gamma}$	16$^{\alpha}$	**30.9$^{\phi}$**	1.9×
		1280x720	–	–	–	–	–	–	**30.9$^{\phi}$**	
	Best FEA Energy/inference (µJ/frame)	224x224	2000$^{\delta}$	340$^{\delta}$	659$^{\alpha\delta}$	557$^{\delta}$	336$^{\gamma\delta}$	503$^{\alpha\delta}$	**36.7**	9.2×
		1280x720	–	–	–	–	–	–	**676**	

1 MAC = 2 Ops. Zero skipping included as MACs $^{\alpha}$Without considering off-chip memory accesses. $^{\beta}$Latency reported on Inception v3. $^{\phi}$Only feature extraction (no FC layer)
$^{\delta}$Estimated feature extraction part. Assuming 1Conv OP = 1FC OP for latency and energy. FC is 0.18% (0.03%) of total MACs/frame on 224x224 images for MobileNet v1 (ResNet18)
$^{\gamma}$Power & latency off-chip weight loading from external DRAM / external host CPU / on-chip network / on-chip memory access / refresh are not included.

>9.2× better energy per inference at 224x224 images with similar accuracy

Outline

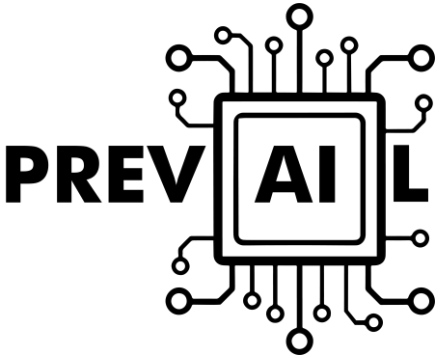
- Introduction and Challenges
- NeuroCorgi
  - Architecture
  - Computing modes
  - Semantic segmentation use case
- NeuroCorgi ImageNet measures
- Conclusion

Conclusion

- NeuroCorgi is a Feature Extractor Accelerator targeting EdgeAI devices
- The streaming architecture leverages on fixed topology and fixed weights to achieve high energy efficiency
- Transfer learning technique is used to address multiple AI tasks
- An implementation flow from application to circuit design is proposed
- NeuroCorgi has been fabricated in three different variants
- ImageNet SRAM variant show outstanding performances
  - 23.2mW (772µJ/frame) with HD images at 30FPS
  - 1.5mW with 224x224 images at 30FPS
  - Processing latency of 6.9ms with HD at 30FPS
  - At least 9.2× energy efficiency over prior ASICs

Acknowledgements

This project has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 876925. The JU receives support from the European Union's Horizon 2020 research and innovation program and France, Belgium, Germany, Netherlands, Portugal, Spain, Switzerland



This project has received funding from PREVAIL project under grant agreement No 101083307 (DIGITAL-2021-CLOUD-AI-01)

The authors would like to thank David Briand, Johannes Christian Thiele and Marc Duranton for their contribution