

Edge Intelligence: Time for the Edge to Grow Up!

Or: Edge AI – From Models to Hardware!



Theocharis Theocharides

Professor and Department Head

Department of Electrical and Computer Engineering

Director of Research, KIOS Research and Innovation Center of Excellence

University of Cyprus

Humans are Lazy!

Humans Could Be Prone To Laziness

According to **Michael In** Toronto, we're lazy and measured given equal o

"Does it mean we're lazy effort," says Inzlicht. "We maximize the amount of

An October 2018 study may in fact, be wired for benefits of exercise, our

It's not completely clear there is an evolutionary able to go to the grocer calorie we burned mear

And we're equally econ for an hour and get an / spend an hour.

Using brain power does burn calories, and, in fact, the brain is among the body's biggest usurpers of **caloric energy**. Some researchers even contend that the higher your IQ, the more calories your brain burns.

It's true that we often look for the easy option, the path of least resistance, the shortcut to success. If you have a remote control, why get up and switch channels on the TV market? If you can get away with doing t?



THE SCIENCES | MIND | TECHNOLOGY | HEALTH | ENVIRONMENT | PLANET EARTH

WE HAVE COMPLETED MAINTENANCE ON DISCOVERMAGAZINE.COM AND ACTION MAY BE REQUIRED ON YOUR ACCOUNT. LEARN MO

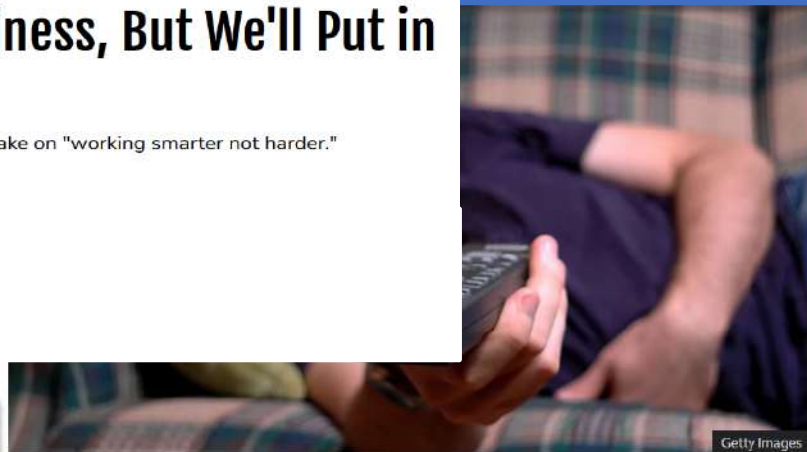


MIND

Humans Could be Prone to Laziness, But We'll Put in Effort for a High Reward

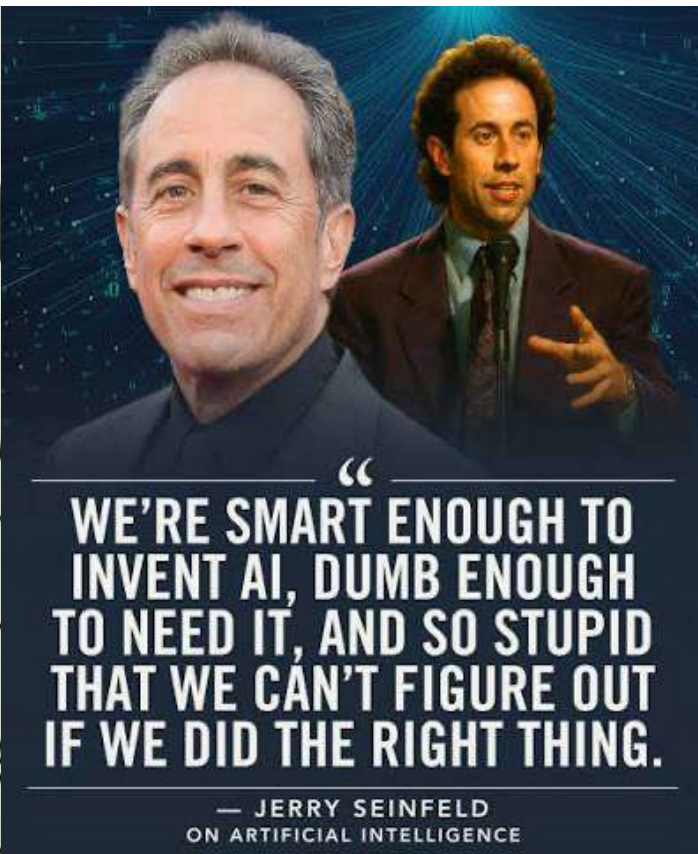
What makes us lazy? Find out how laziness could be a whole new take on "working smarter not harder."

By Sara Novak
Feb 26, 2024 6:00 PM

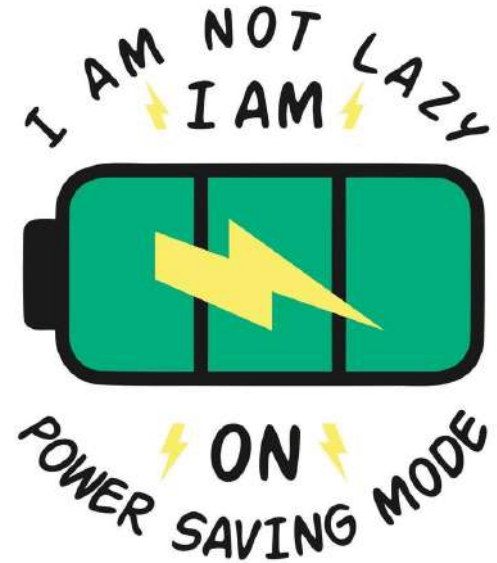


Getty Images

We created AI to help us do boring tasks...



Doing something, does not mean doing it efficiently!



"I choose a lazy person to do a hard job. Because a lazy person will find an easy way to do it"
– (attributed to Bill Gates, although its origin maybe much longer in time...)

So, why is this relevant to today's talk?



The Circle of Life...



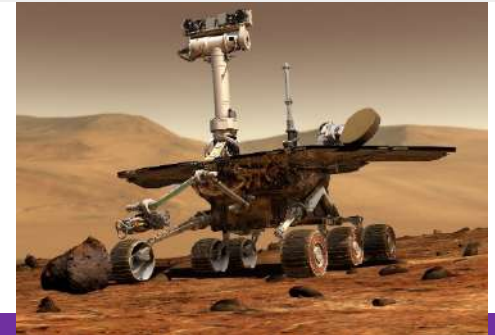
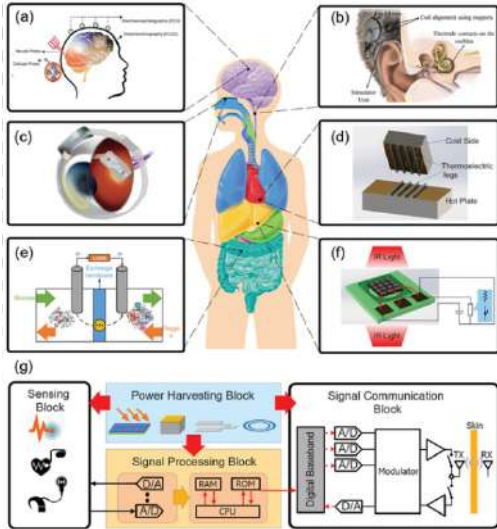
"Edge Intelligence: Time for the Edge to Grow Up!"

1964: Stanley Kubrick's "Connected World"



- **Before** ARPANET
- **Before** the Internet
- **Before** Minski's & Papert's Vision Project (1966)

Well, not science fiction anymore!



"Edge Intelligence: Time for the Edge to Grow Up!"

The “Birth” of Edge AI

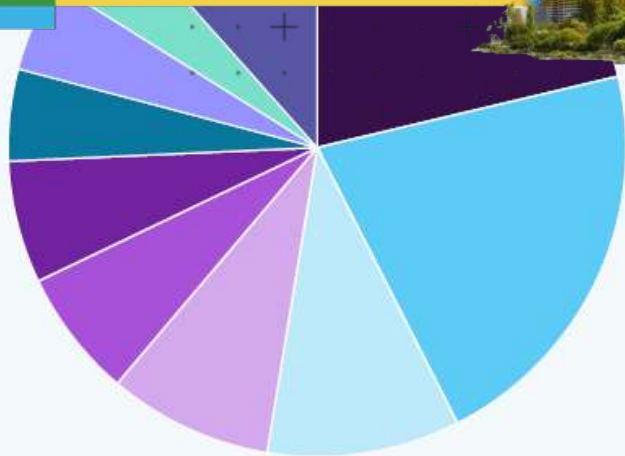


Step 1: Communications (the “IoT” Buzz)

Edge AI Market

There will be **2.5 billion tiny edge devices** across industrial, smart city and consumer electronics by 2030

2.5 Billion
tiny edge devices



- Healthcare
- Manufacturing
- Smart Cities
- Retail
- Energy
- Government
- Others



\$24.9B

Global Market Size,
2025

Source:
www.grandviewresearch.com

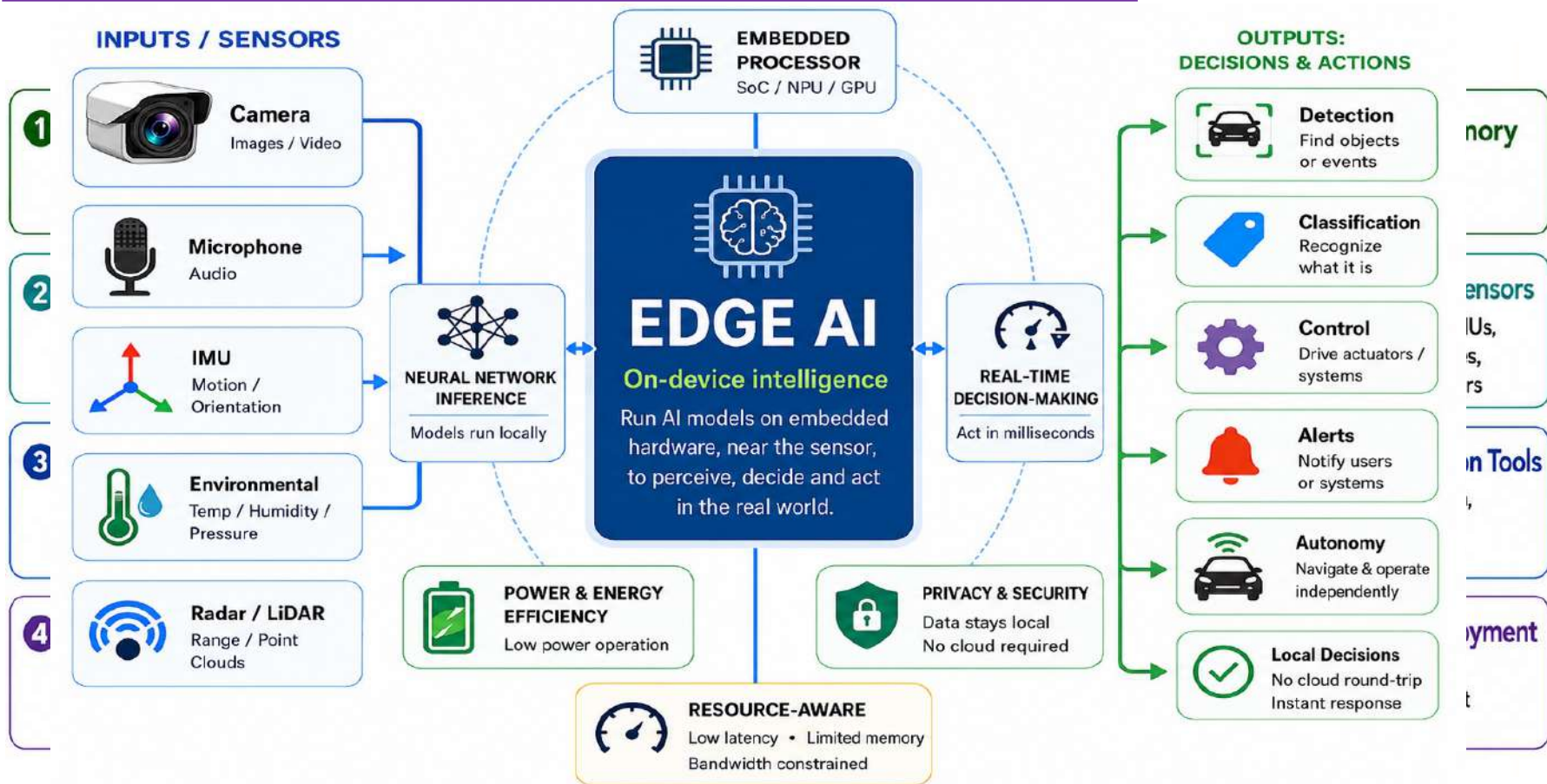
Key Challenge

Robust (*Reliable & Secure*), Resource-Constrained, and **Real-Time** technologies for *Intelligence @ Edge*

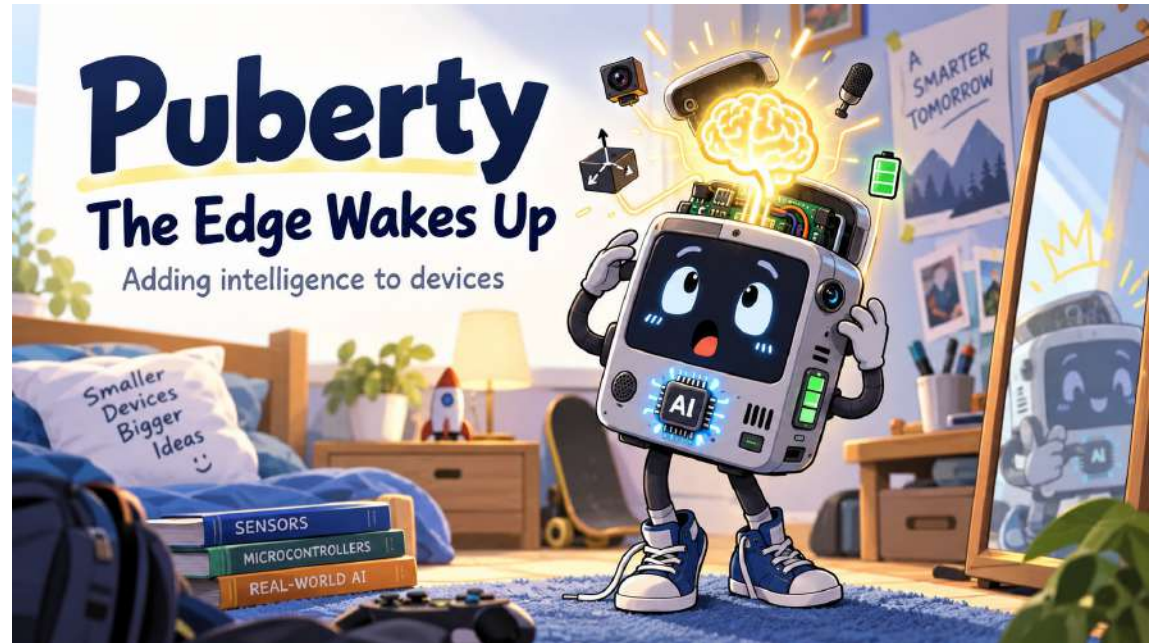
Sense → Analyze & Decide → React



Edge AI Illustrated



Part 1: The Puberty Era



Sensing and Communicating

It starts with the Sensors!

50 MP

Ultra Wide

50 MP

Telephoto

5x Optical Zoom & 10x Optical Quality Zoom

200 MP

Wide-angle

2x Optical Quality Zoom

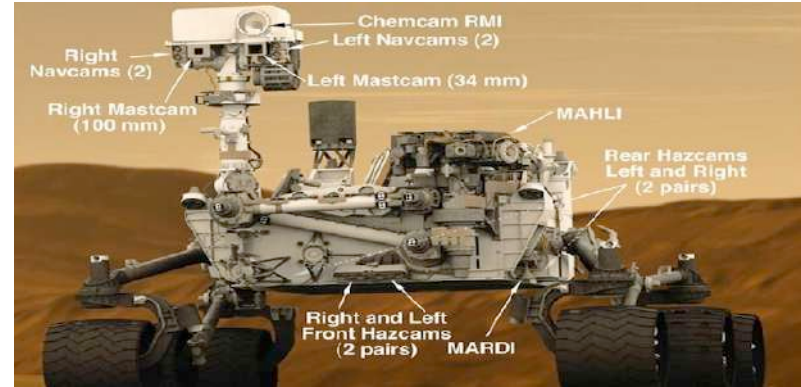
10 MP

Telephoto

3x Optical Zoom

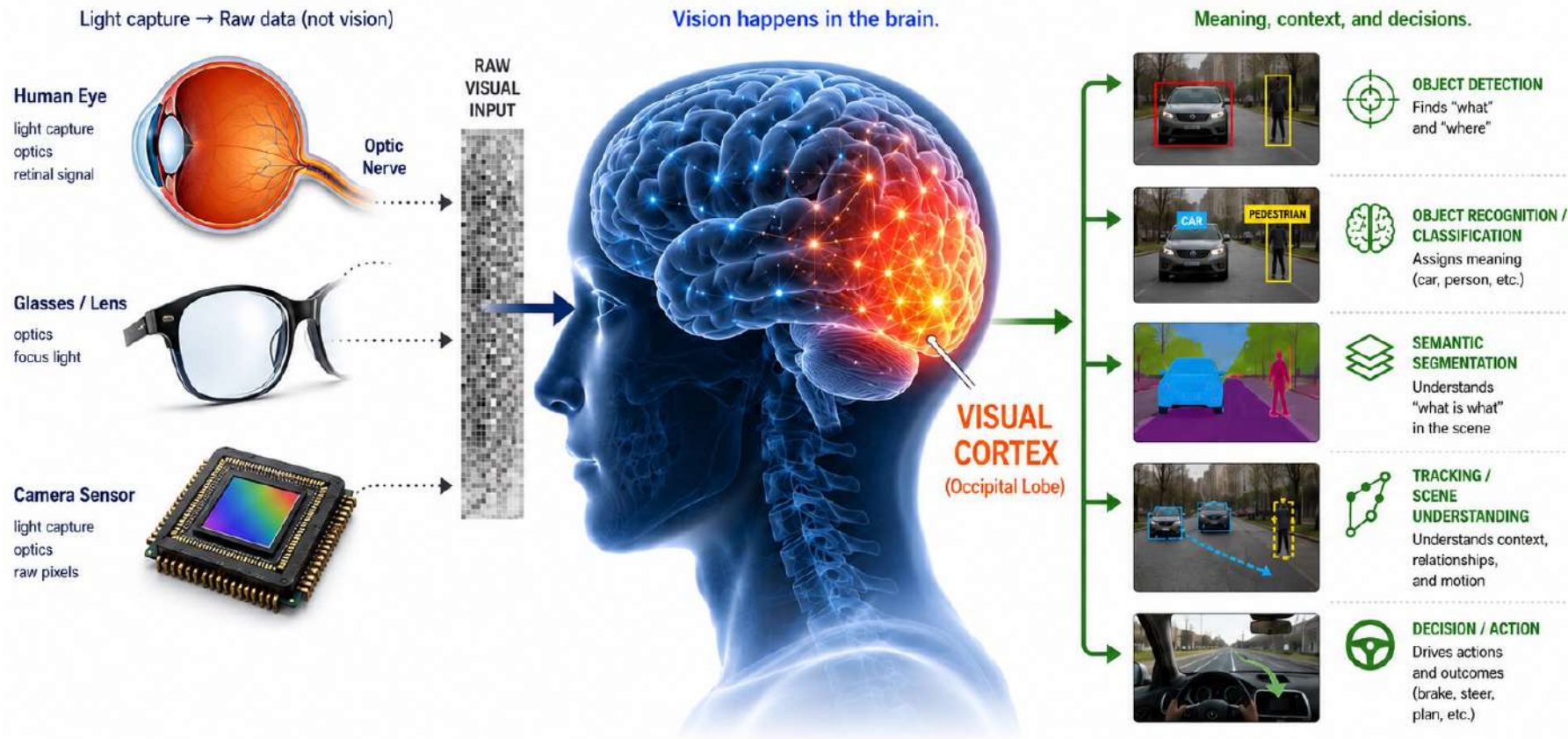
Remote Sensing, Sensor Fusion, Sensing beyond Biology

Photo Credit: Wired, BeamPro, NASA



"Edge Intelligence: Time for the Edge to Grow Up!"

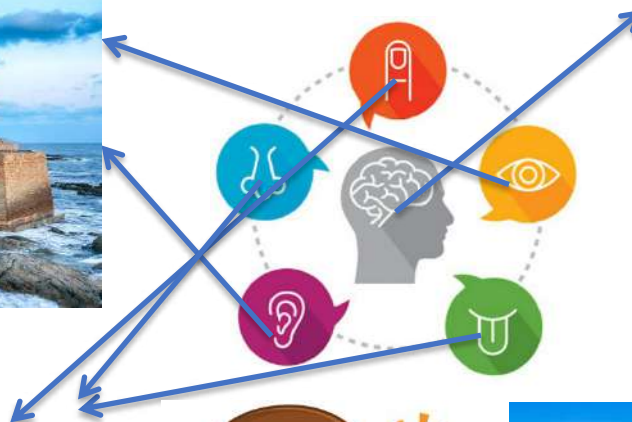
Intelligence is the BRAIN, NOT the sensor



Thus...the senses (and sensors!) need a brain!



Church? Space Shuttle?



CONTEXT!!!



Alghero!!!

Edge AI → Building the Edge Brains!



"Edge Intelligence: Time for the Edge to Grow Up!"

Typical Edge Brains (aka Processors :-))

1

Edge GPU



Programmable parallel processor for high-throughput edge vision

Advantages

- High parallelism
- Model flexibility
- Strong throughput

Disadvantages

- Higher power
- Thermal limits
- Memory demand



Robotics, UAVs, mobile vision

2

Edge TPU / dedicated AI accelerator



Specialized inference engine for efficient neural-network execution

Advantages

- Energy efficient
- Fast inference
- Compact module

Disadvantages

- Restricted operators
- Quantized focus
- Less flexible



Smart cameras, gateways, compact edge nodes

3

NPU



Integrated neural engine inside edge SoCs and embedded platforms

Advantages

- Low latency
- On-chip integration
- Good efficiency

Disadvantages

- Vendor lock-in
- Tool dependence
- Limited portability



Phones, embedded SoCs, automotive edge

4

MCU / TinyML device



Ultra-low-power microcontroller for tiny always-on intelligence

Advantages

- Very low power
- Low cost
- Tiny form factor

Disadvantages

- Limited memory
- Small models
- Lower accuracy



Wearables, sensors, battery nodes

5

FPGA



Reconfigurable hardware for custom low-latency AI pipelines

Advantages

- Custom dataflow
- Low latency
- Reconfigurable

Disadvantages

- Long development
- Tool complexity
- Lower usability



Industrial edge, prototyping, adaptive systems

6

DSP / vision processor



Signal-centric processor for efficient vision and multimedia tasks

Advantages

- Signal efficient
- Low power
- Good streaming

Disadvantages

- Narrow scope
- Limited flexibility
- Programming effort



Vision sensing, multimedia edge, smart perception

Efficiency / low power



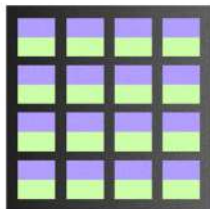
Programmability / throughput

Industry is racing!



Mythic Analog Computing

Compute-in-memory



Compared to digital high-bandwidth memory:

18,000x

Faster

1,500x

More energy-efficient

>1,000x

More parallel

Top Hardware options for Edge AI Deployment

Hailo-8 AI Accelerator



Kinara AI Accelerator



RPI + AI Accelerators



NVIDIA Jetson

AI on Boards, Edge Box



RK3588 + Hailo-8

AI on Modules & Boards, Edge Box

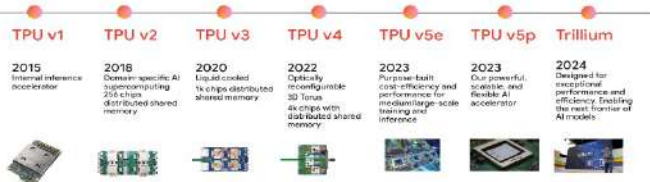


i.MX 8M Plus + Kinara Ara-2

AI on Modules & Boards, Edge Box



TPU AI accelerators

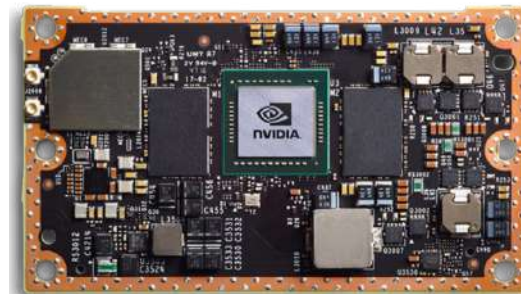
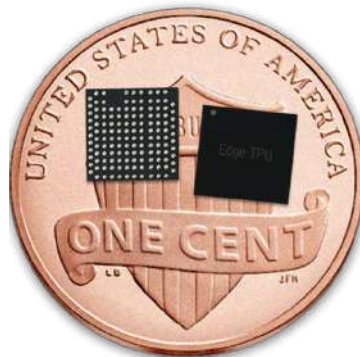
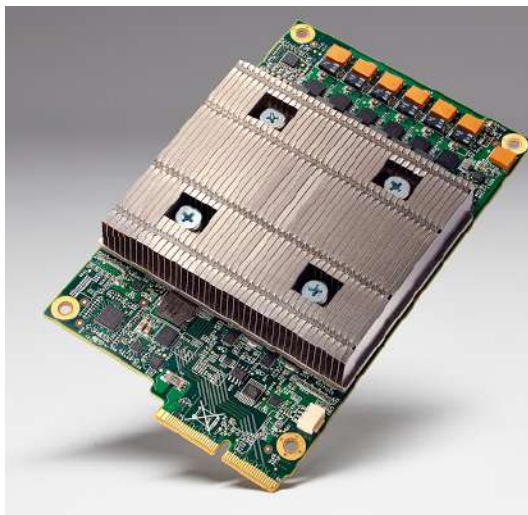


Source: AMD, Intel, Google, Tenstorrent

"Edge Intelligence: Time for the Edge to Grow Up!"

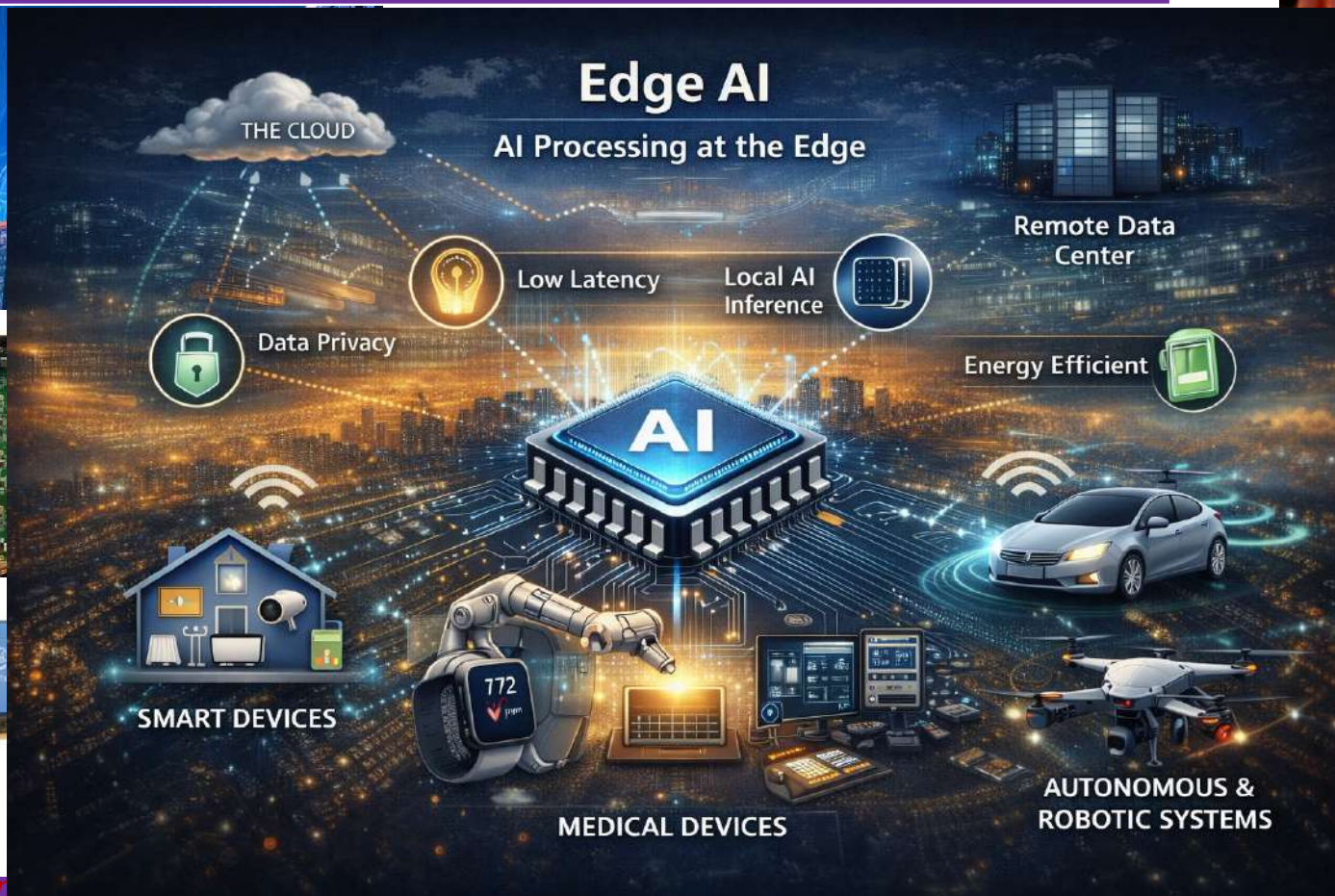
Some Early Edges...

Google's Edge TPU and NVIDIA Jetson TX2



The Edge TPU is the little brother of the regular Tensor Processing Unit, which Google uses to power its own AI, and which is available for other customers to use via Google Cloud. Credit: Google

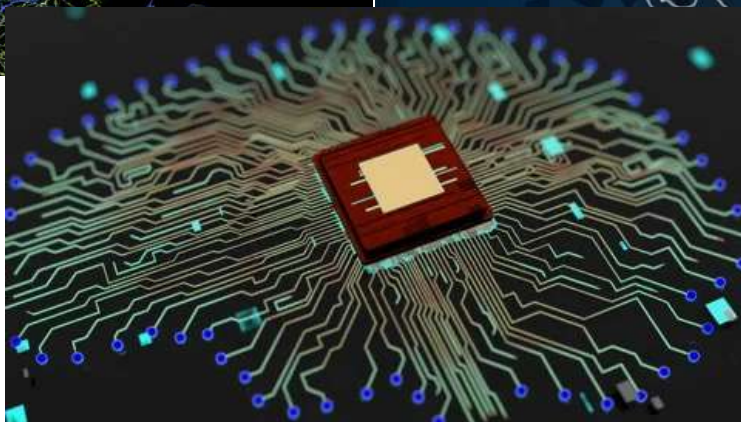
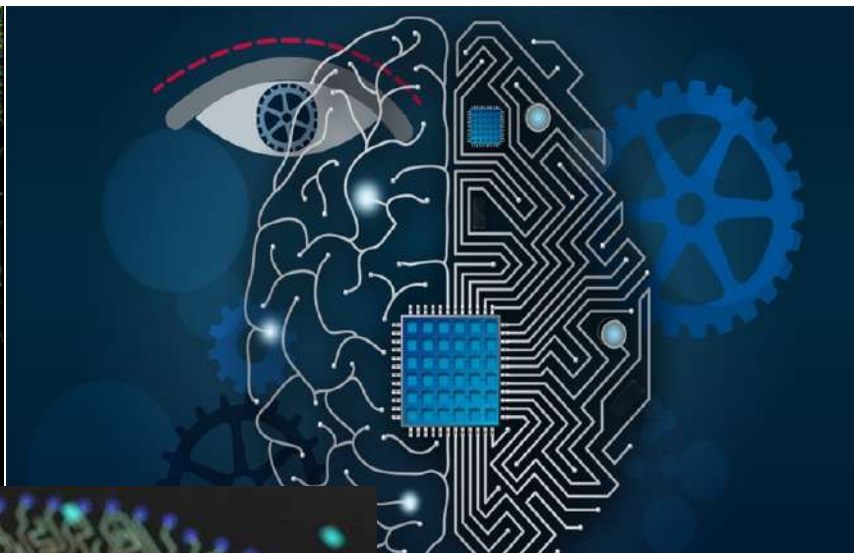
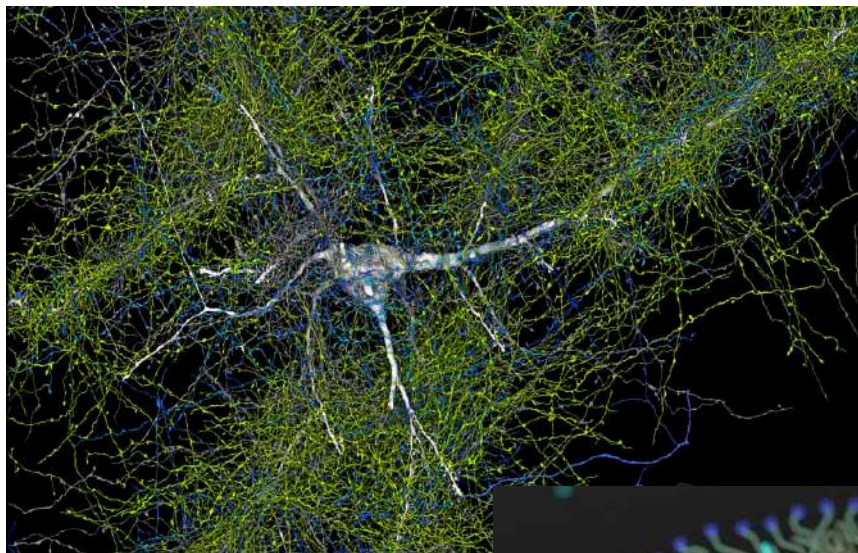
Edge AI Off—The Shelf



Sources: In

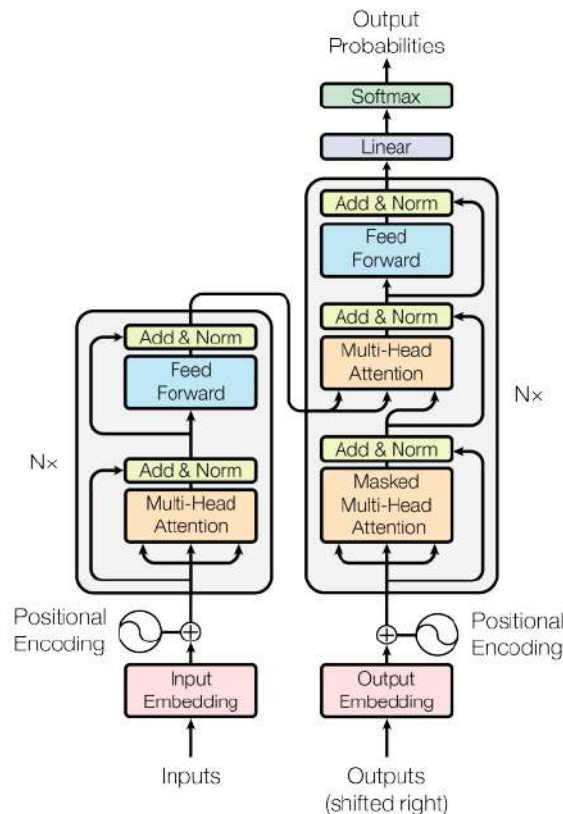
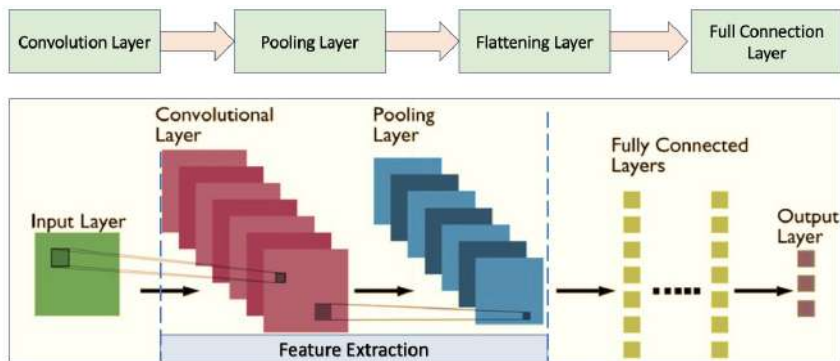
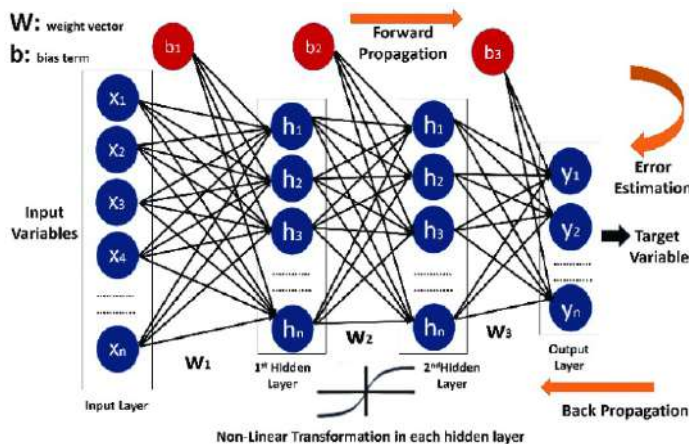
Edge Intelligence. Time for the Edge to Grow Up:

The “Edge” Brain



“Edge Intelligence: Time for the Edge to Grow Up!”

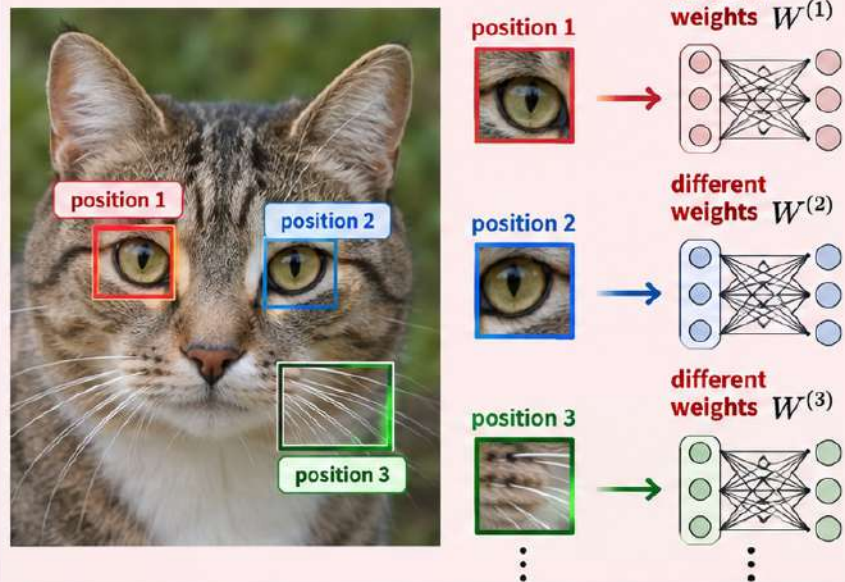
The Brain Models (MLPs, CNNs, Transformers...)



MLP and CNNs

MLP (fully connected)

Each location needs its own detector
(separate weights)

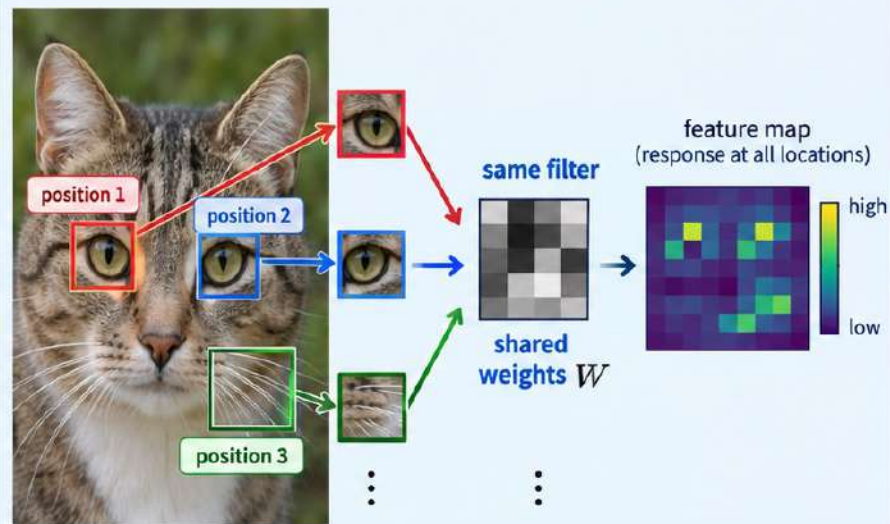


MLP: separate weights at each location

✗ **Inefficient: many parameters** (one detector per location)

CNN (convolution)

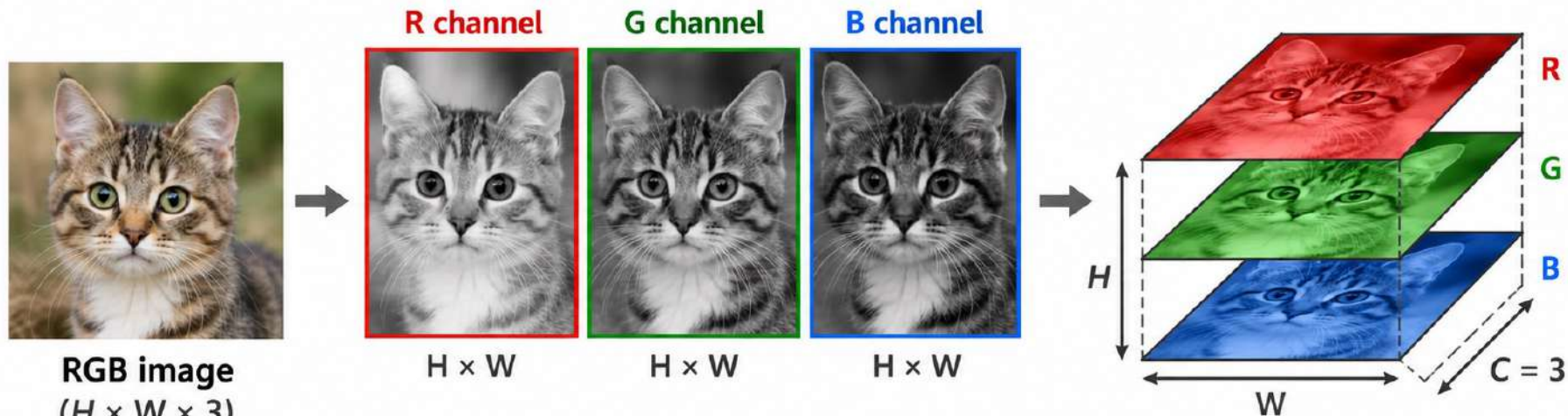
One filter is reused at all locations
(shared weights)



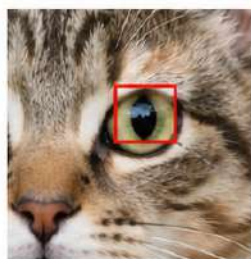
CNN: same weights reused everywhere

✓ **Efficient: far fewer parameters** (one filter for all locations)

Data (e.g. images)



One pixel has 3 values (one per channel):



R = 120

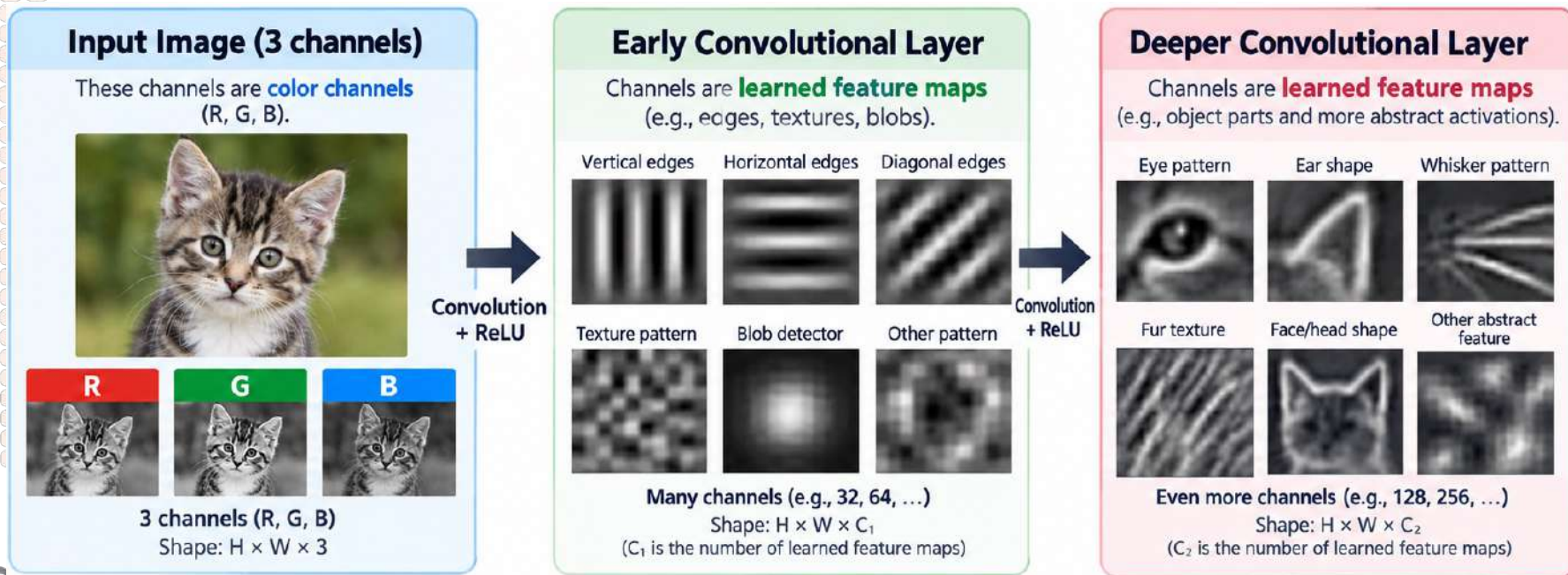
G = 200

B = 80

Pixel value:
[120, 200, 80]

Input tensor: $H \times W \times 3$
(for RGB, $C = 3$)

Channel Representation



Pixels $\rightarrow$ Edges $\rightarrow$ Object Parts $\rightarrow$ Learned Representations

Data Movement: Stride

Stride = 1

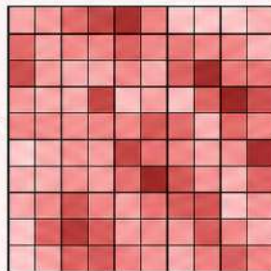
filter moves one pixel at a time

- more activations
- higher spatial resolution



Input image
(grayscale)

3 × 3
filter



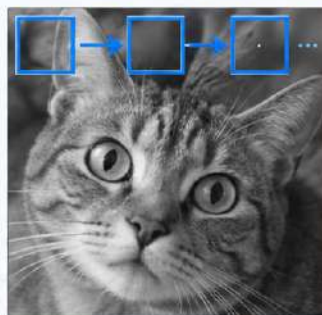
Output feature map
(many activations)

Output is sampled more densely
(one position at a time).

Stride = 2

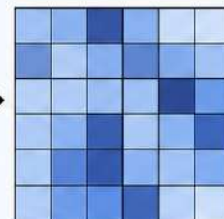
filter skips every second position

- fewer activations
- lower spatial resolution
- lower computation



Input image
(grayscale)

3 × 3
filter



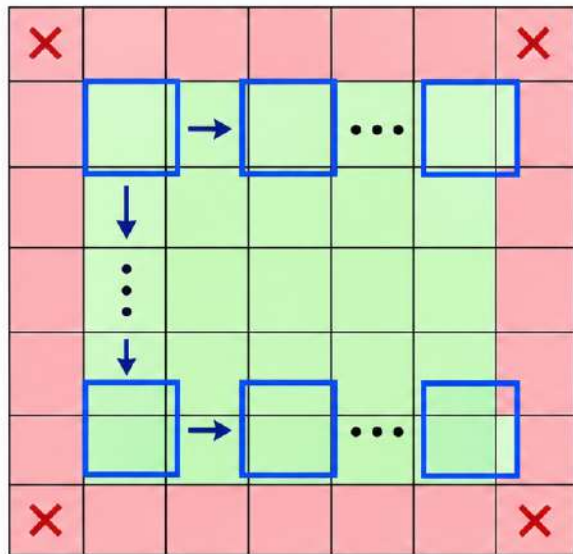
Output feature map
(fewer activations)

Output is sampled more coarsely
(every second position).

Stride is a sampling / design choice that controls how far the filter(s) moves each step.

Data Movement: Padding

Input (7×7)
 $N = 7$



Valid 3×3 positions
(kernel fully inside)



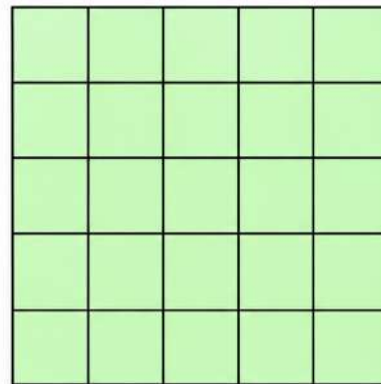
Invalid positions
(kernel goes outside)

3×3 kernel
 $K = 3$

w_{11}	w_{12}	w_{13}
w_{21}	w_{22}	w_{23}
w_{31}	w_{32}	w_{33}

Stride
 $S = 1$

Output (5×5)
 $N_{\text{out}} = 5$



$$\begin{aligned} N_{\text{out}} &= \frac{(N - K)}{S} + 1 \\ &= \frac{(7 - 3)}{1} + 1 \\ &= 5 \end{aligned}$$

7×7 input $\rightarrow 5 \times 5$ output

**No padding $\Rightarrow$
output shrinks.**

Data Movement: Pooling



local responses
(feature map patch)

0.2	0.9
0.4	0.1

Stronger response at the cat's whisker (edge).

max pooling

take the maximum value
in the local region

strongest response kept
(pooled activation)

0.9

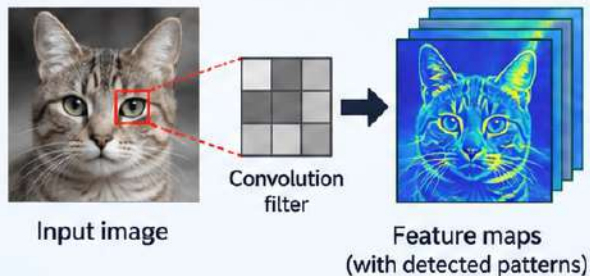
Keep 0.9 (strongest local activation) and discard weaker neighboring responses (0.2, 0.4, 0.1).

Max pooling downsamples the feature map by taking the maximum value from each 2×2 region.

A “Modern” Conv Block

Convolution

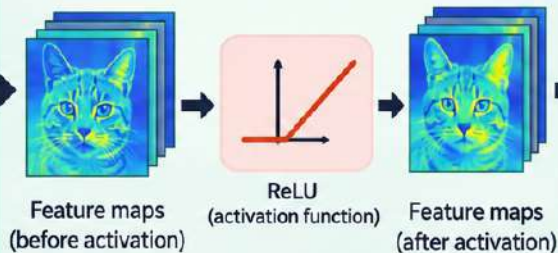
Extracts local patterns from the input (e.g., edges, textures).



$3 \times 3 \times 3$

Nonlinearity

Applies an activation function (e.g., ReLU) to introduce nonlinear behavior.

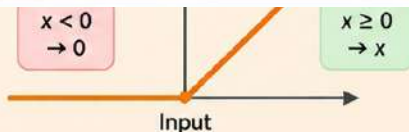
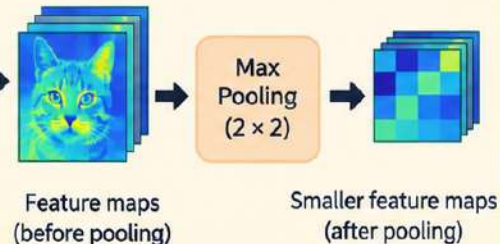


mean: -0.5
std: 1.9

mean: 0.0
std: 1.0

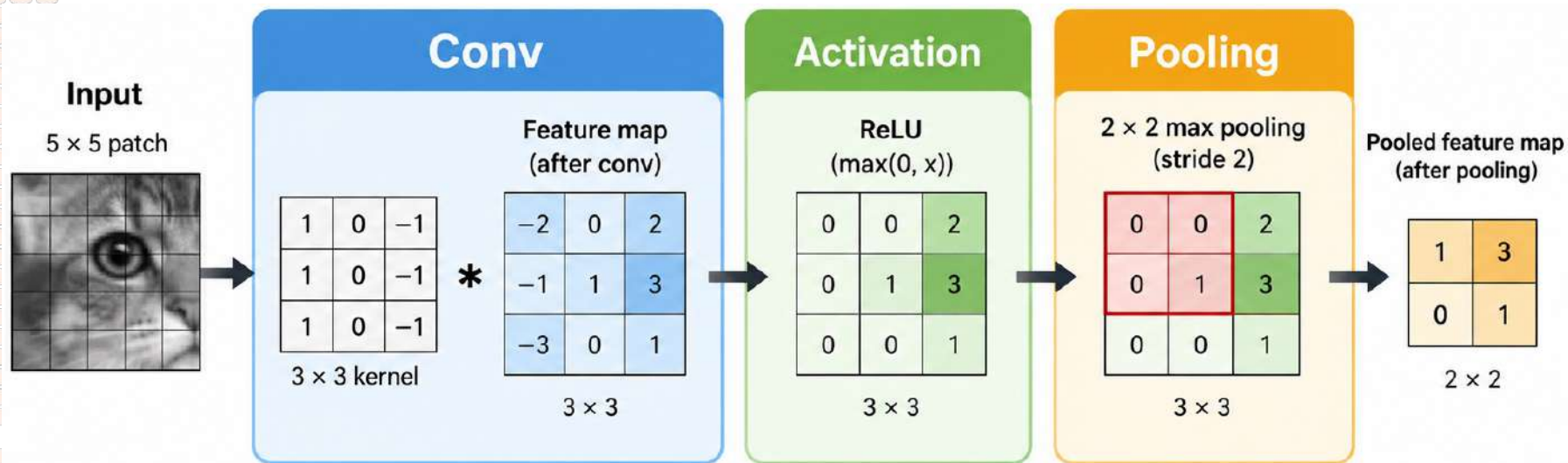
Spatial reduction

Uses pooling or downsampling to create a smaller feature map (e.g., max pooling).



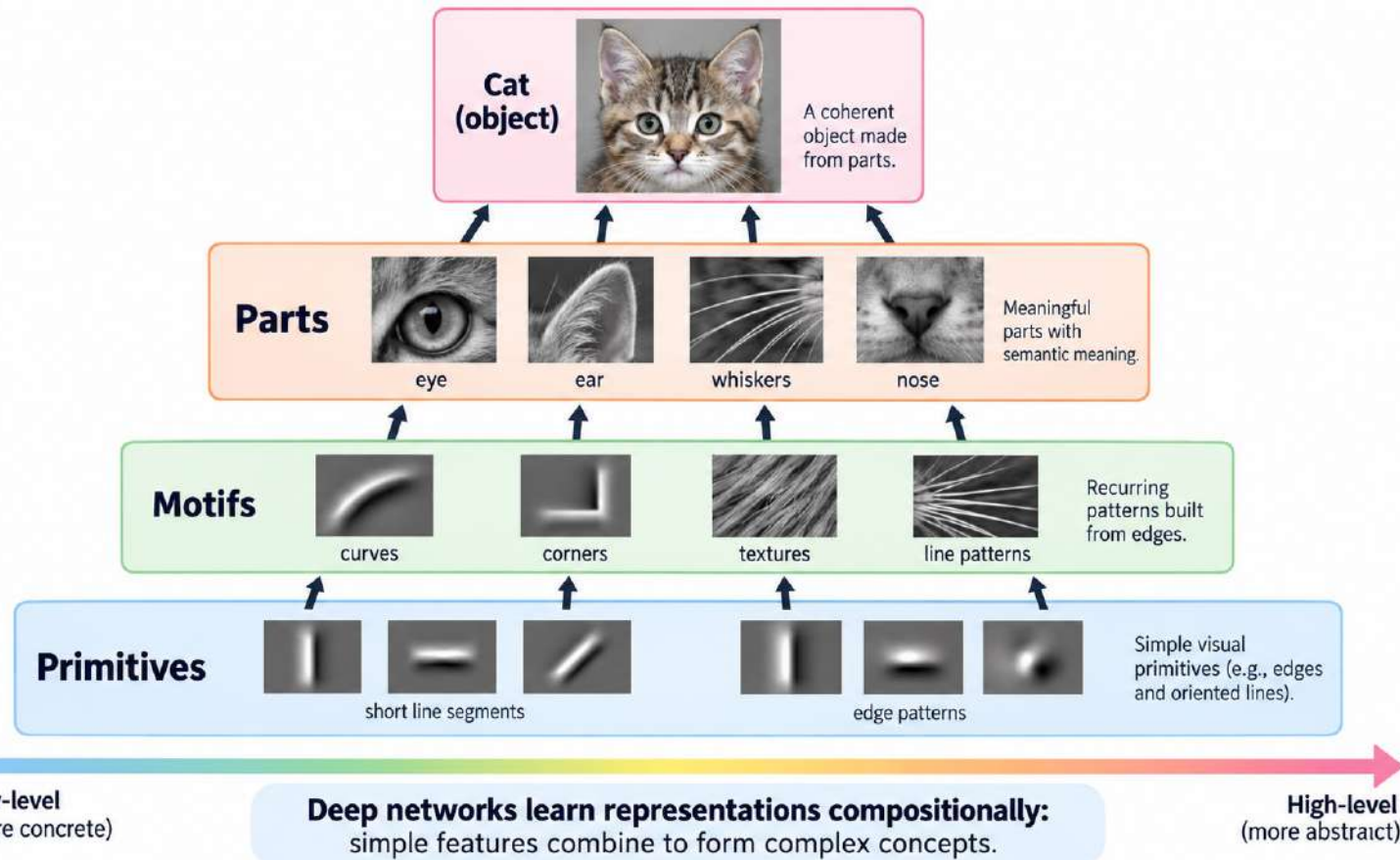
Convolution → Feature Extraction
BatchNorm → Used for Efficient Training
ReLU → Non-Linearity

The Convolution Block - Datapath

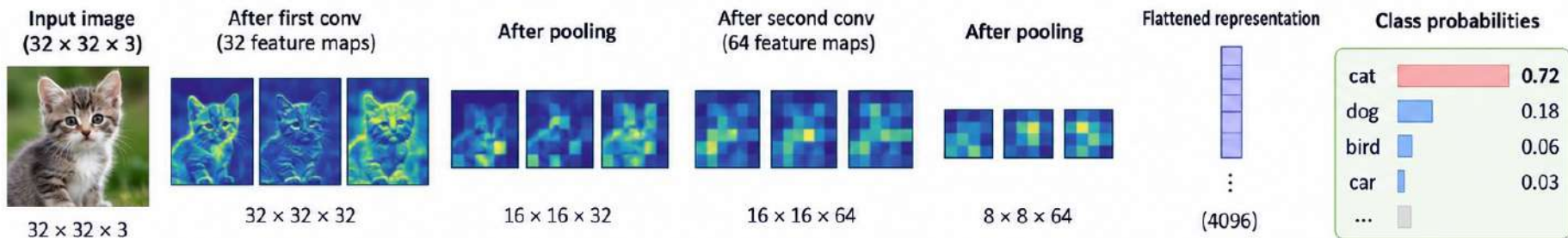
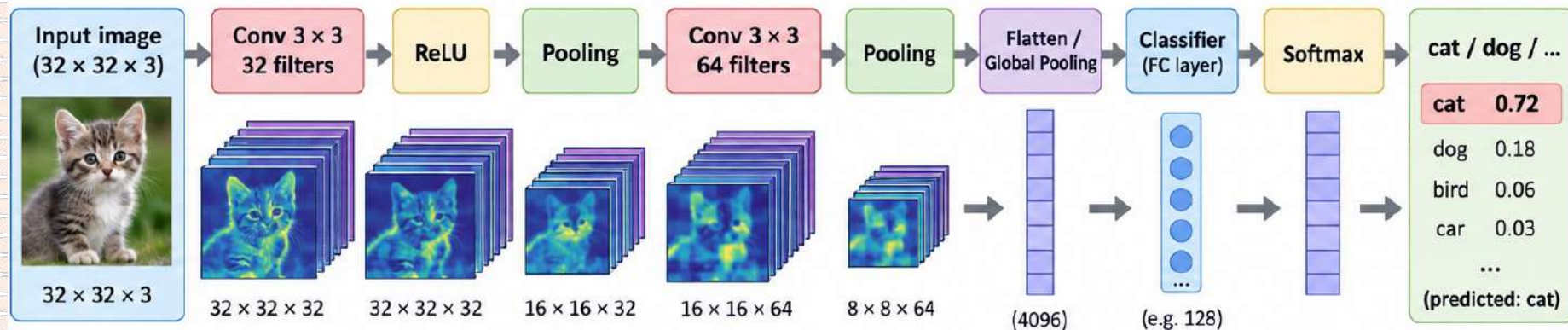


Extract Features → Apply Non-Linearity → Reduce Spatial Size

As a side note: Meaning of Depth (and DNNs)



A CNN Model – Computation Datapath



High spatial resolution
(32×32)

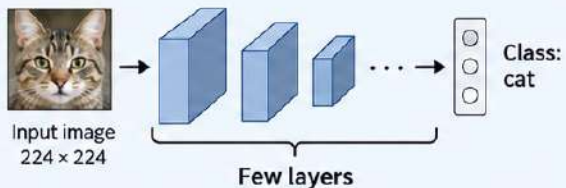
Learn hierarchical features
(spatial size ↓, channels ↑)

Compact representation
for classification

Increasing Depth – More Data Operations

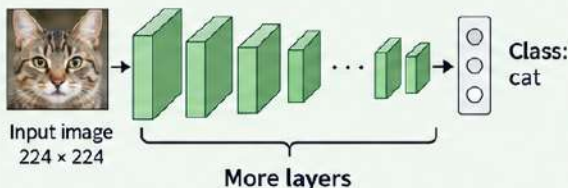
Shallow CNN

(e.g., 3–5 layers)



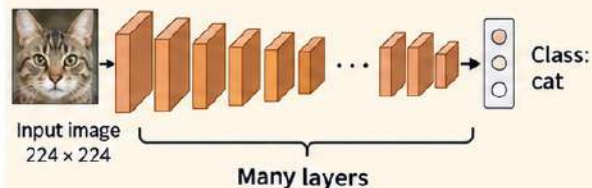
Deep CNN

(e.g., 10–30 layers)



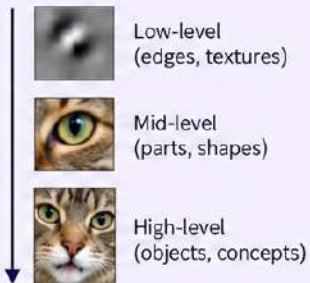
Very Deep CNN

(e.g., 50–200+ layers)

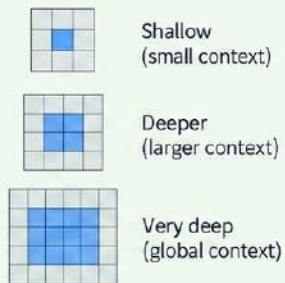


What changes when networks become truly deep?

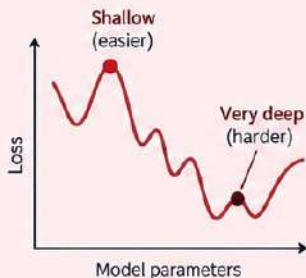
Richer hierarchical features



Larger receptive fields

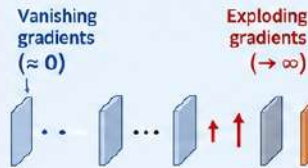


Harder optimization



Vanishing / exploding gradients

As depth increases, gradients can become very small or very large.



Need for architectural ideas

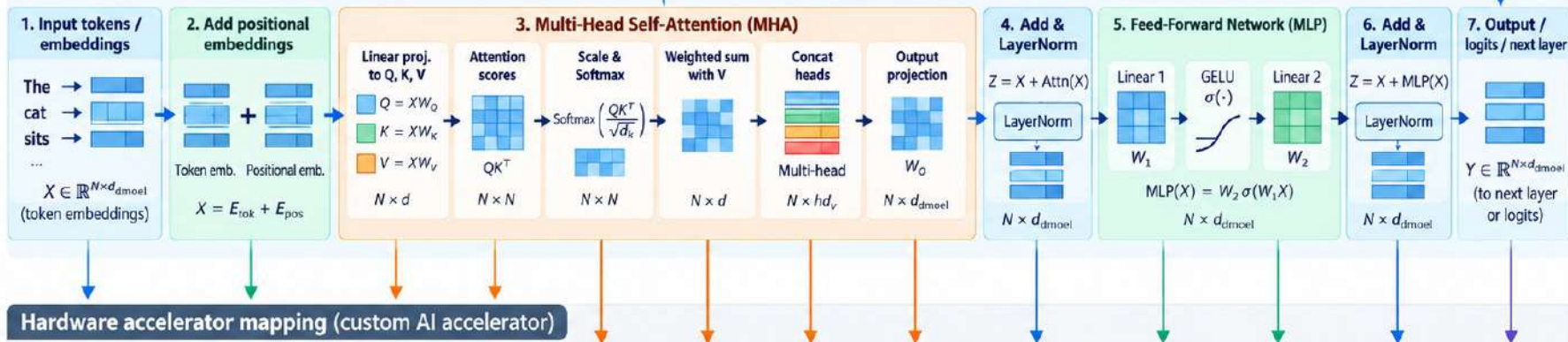
To train very deep networks successfully, we rely on:

- **Normalization** (e.g., BatchNorm, LayerNorm)
- **Skip connections** (e.g., ResNet)
- **Careful initialization**
- **Modern building blocks** (e.g., bottlenecks, residual blocks)
- **Regularization and training tricks** (e.g., dropout, learning rate schedules)

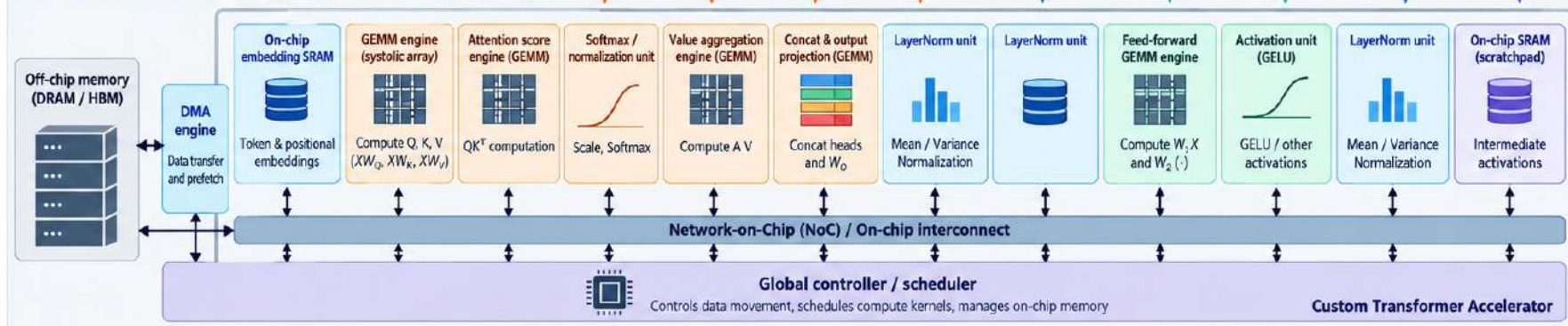
One Net to Rule them All...

Transformer computation flow (one Transformer block)

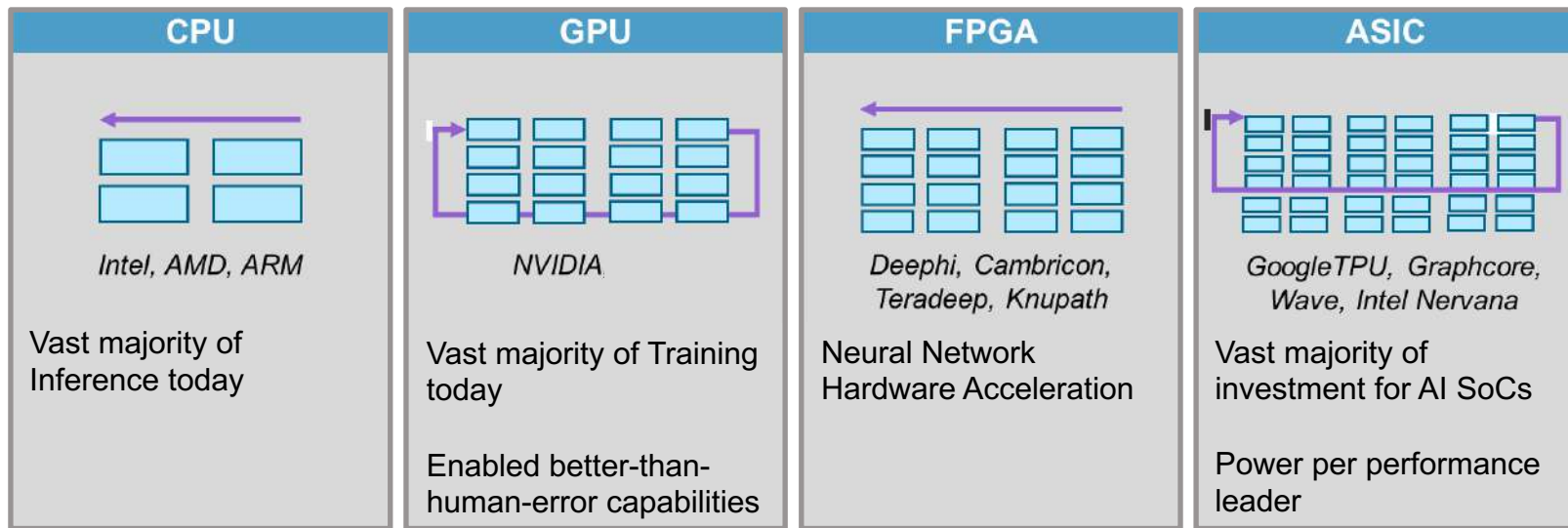
Repeat N times (stack of Transformer blocks)



Hardware accelerator mapping (custom AI accelerator)

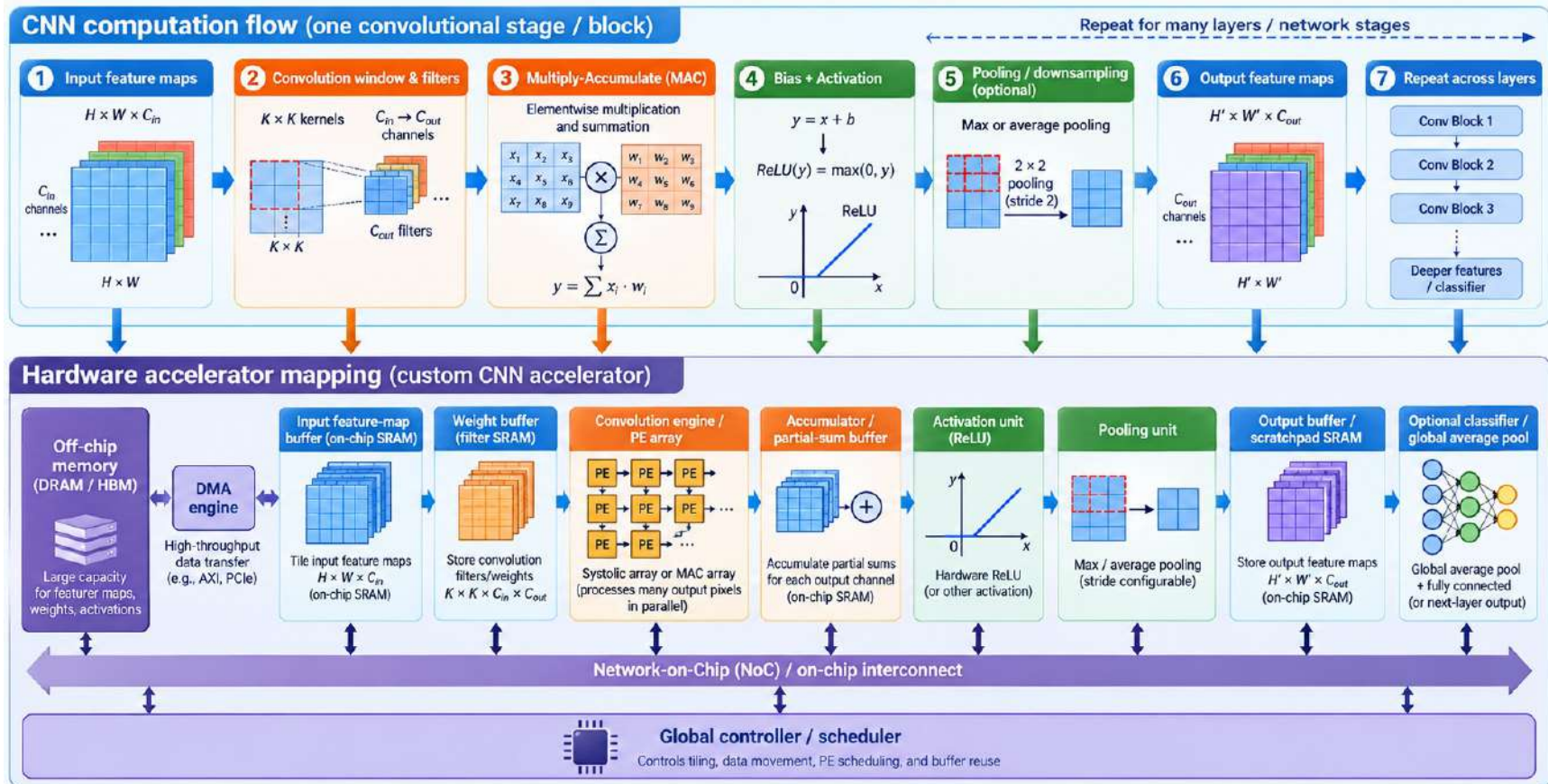


AI Hardware



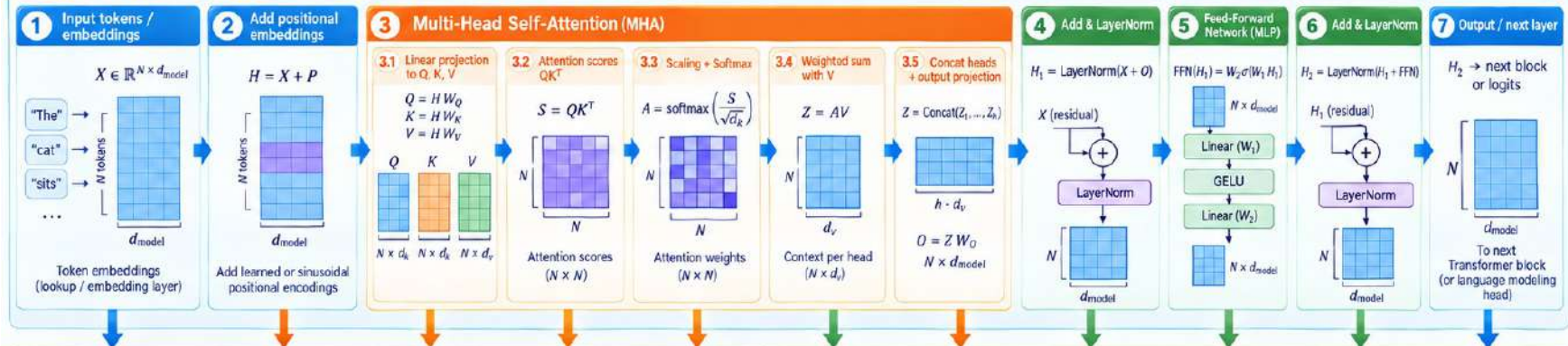
- Most investment dedicated for CNNs & Transformers
- Solutions include Software Development Kit (SDK) for mapping AI Graphs to hardware
- Most competitive include Neural Network Hardware Acceleration (MAC operation Circuits)

Accelerating Deep CNNs...

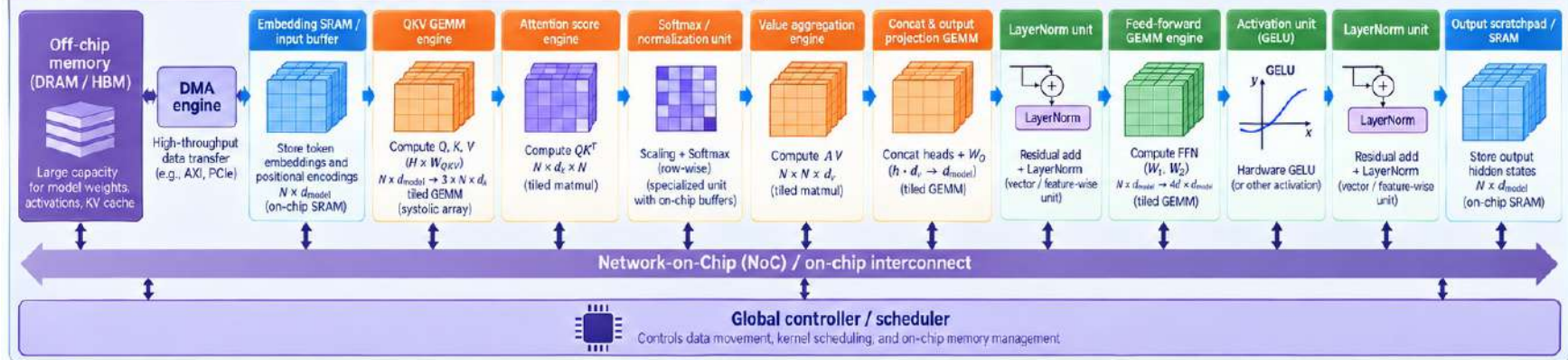


Accelerating Transformers...

Transformer computation flow (one Transformer block)

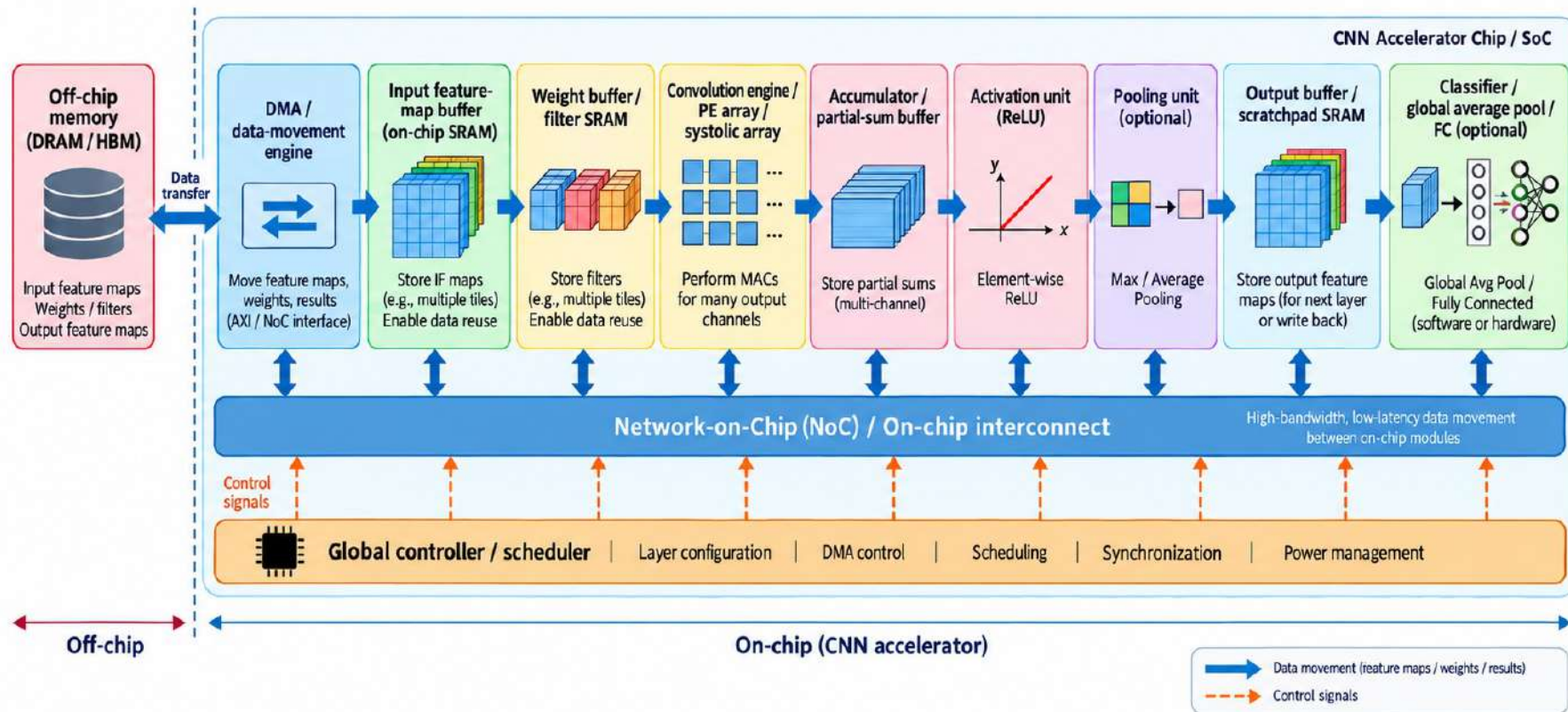


Hardware accelerator mapping (custom Transformer accelerator)

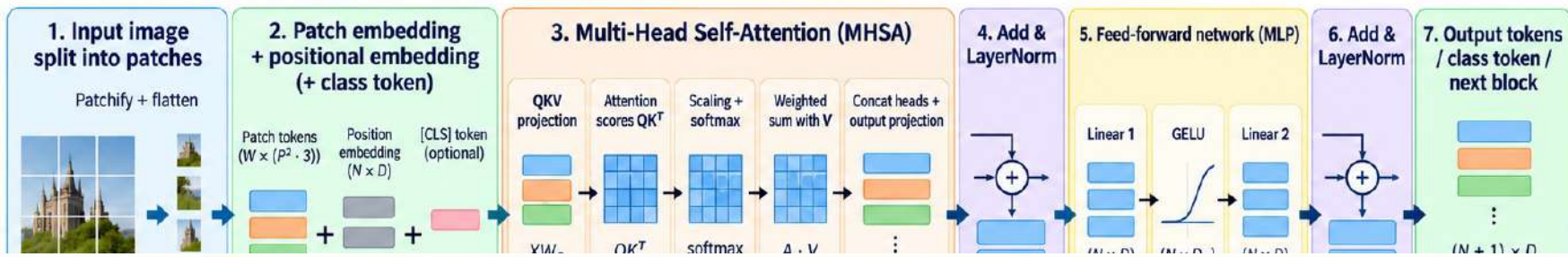


"Edge Intelligence: Time for the Edge to Grow Up!"

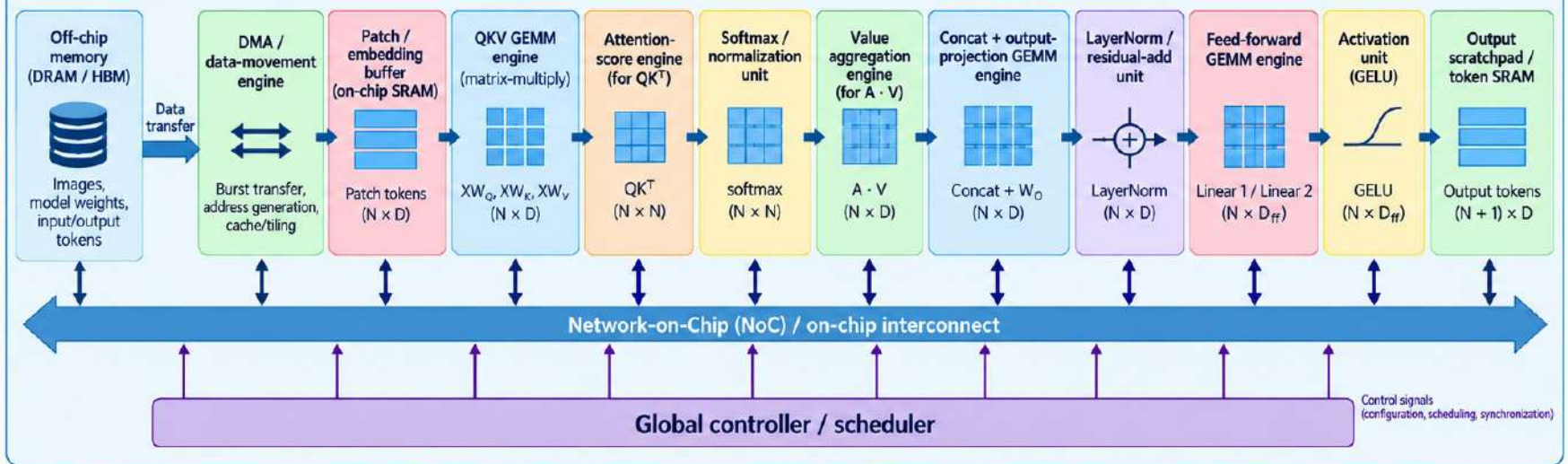
CNN Computation → H/W



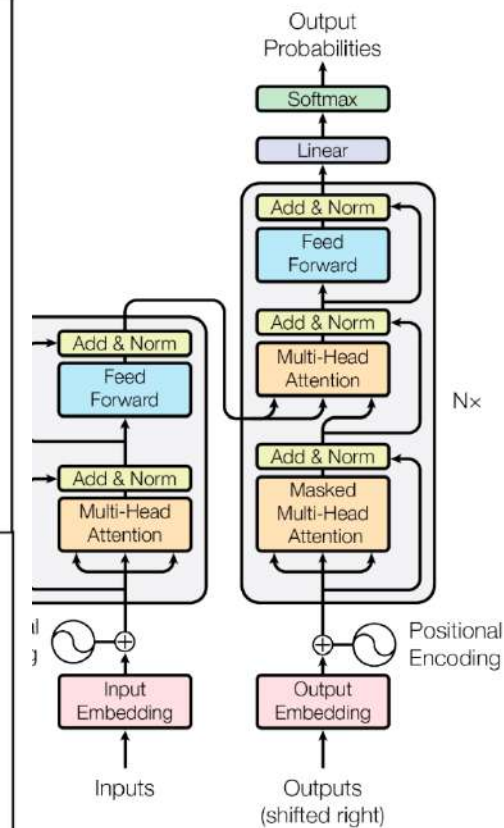
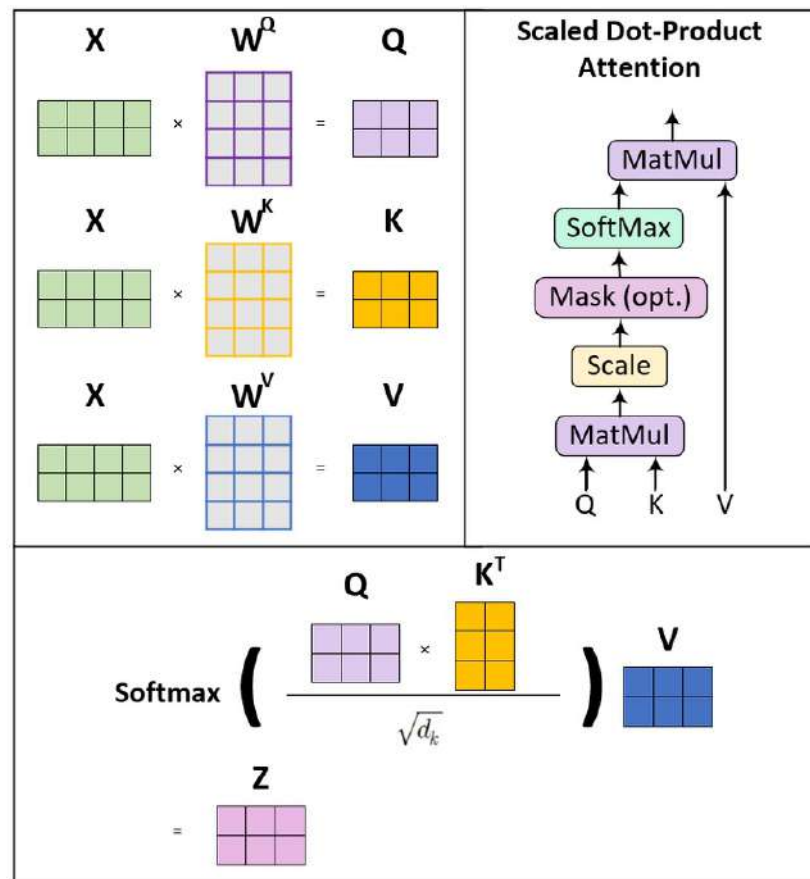
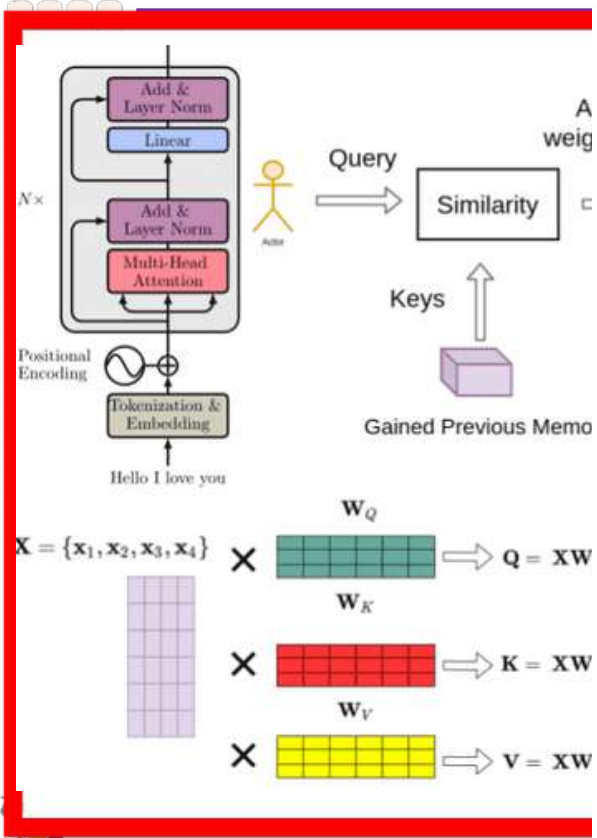
Transformer Computation → H/W



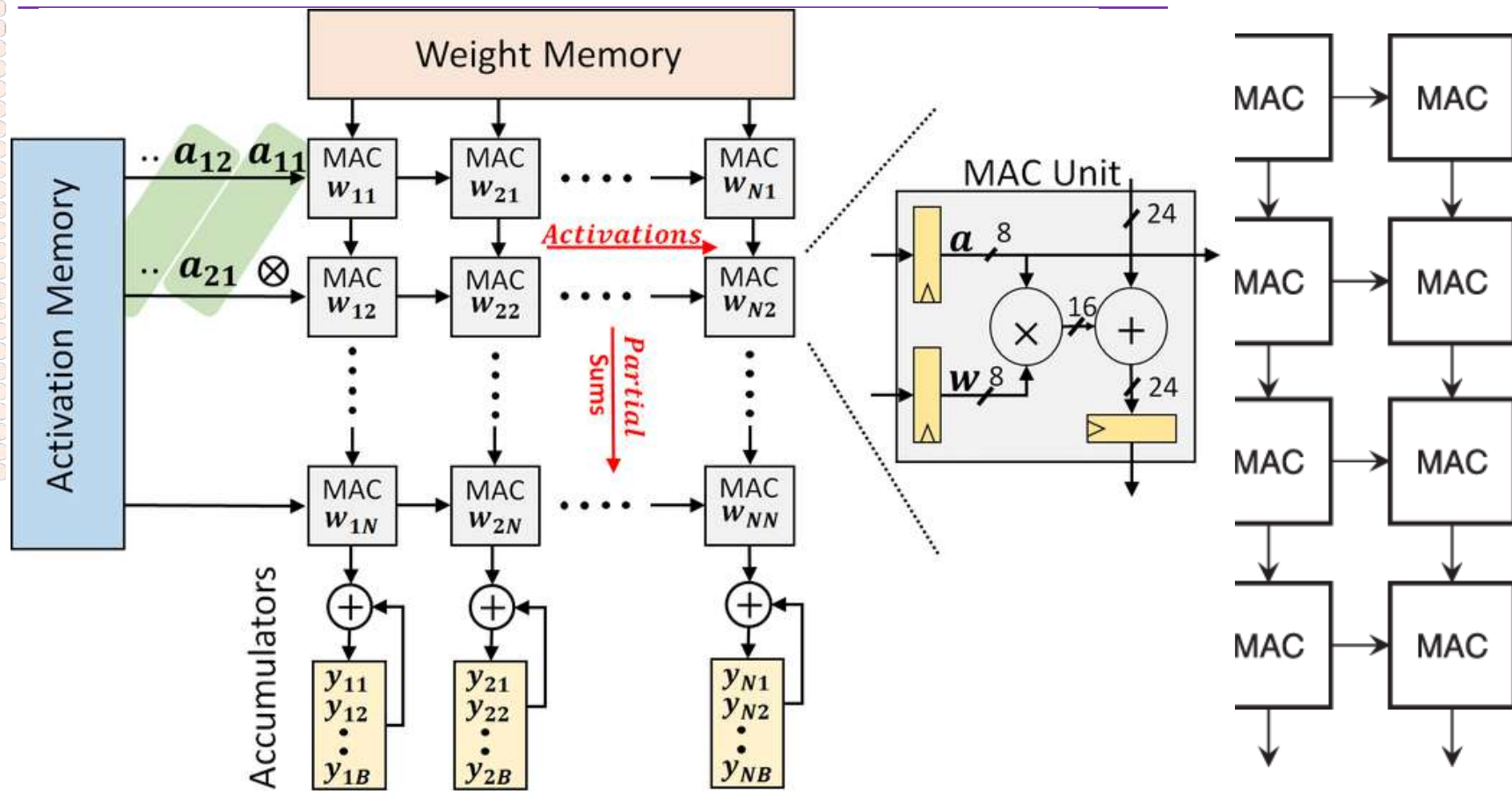
ViT Accelerator Chip



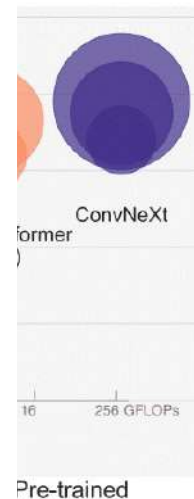
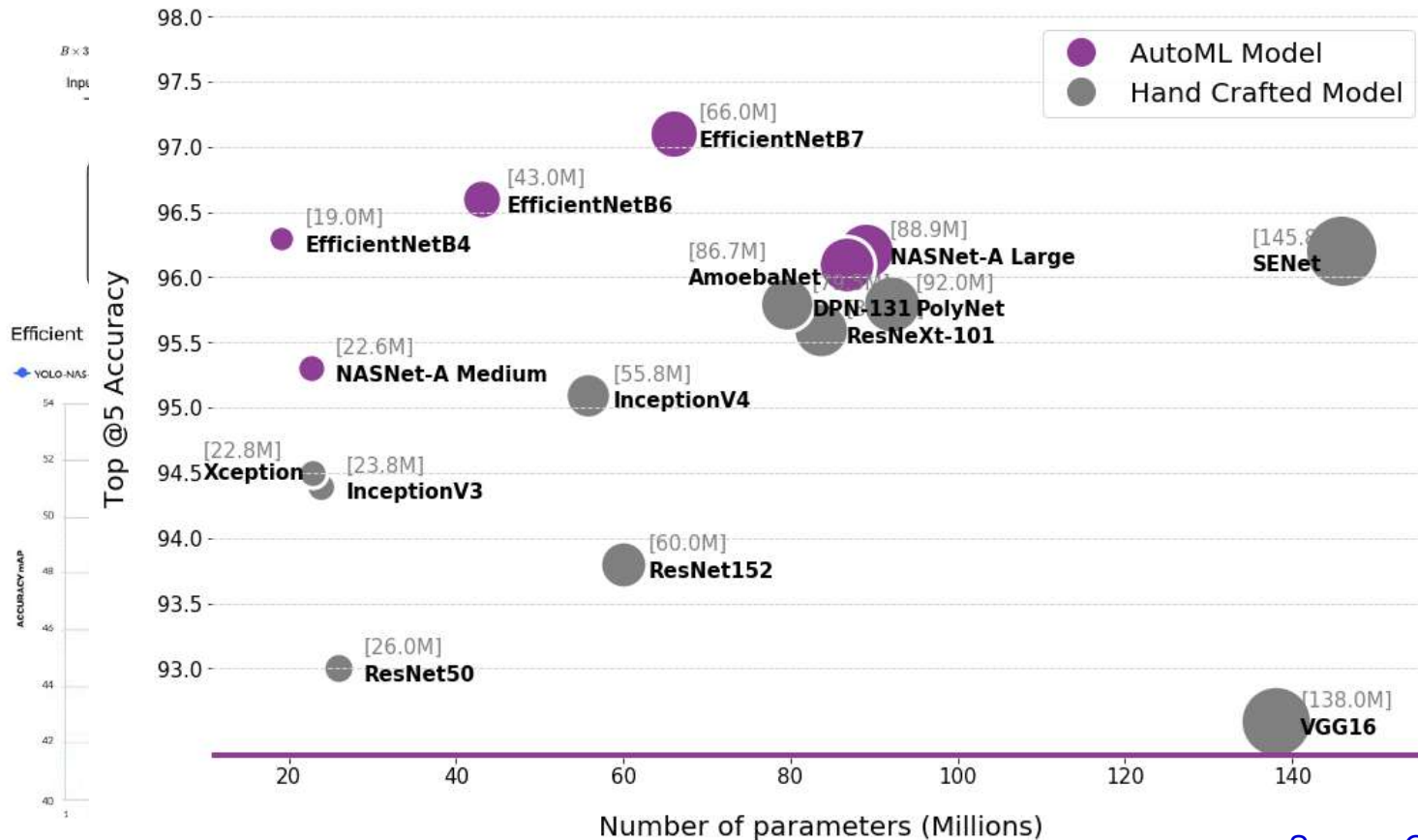
Inference (i.e. when the brain computes)?



Typical Processing



Alas, Intelligence Needs MEMORY!



Source: Open Data Science

Some Metrics

Model	Example Variant	Parameters	Approx Model Size (FP32)
ResNet	ResNet50	~25 million	~100 MB
MobileNet	MobileNetV2	~3.4 million	~14 MB
YOLO	YOLOv11-small (approx)	~10–20 million	~40–80 MB
Transformer	BERT-base	~110 million	~420 MB
Transformer	GPT-2 small	~124 million	~500 MB
LSTM	2-layer LSTM (typical NLP)	~10–30 million	~40–120 MB

Precision	Memory per parameter
FP32	4 bytes
FP16	2 bytes
INT8	1 byte

ResNet50 with INT8 quantization = $25\text{M} \times 1 \text{ byte}$
 $\approx 25 \text{ MB}$

Edge devices typically 1M – 20M parameters

Model	Parameters
MobileNet	~3M
ResNet50	~25M
BERT-base	~110M
GPT models	Billions!!!!

Some numbers to consider...

Component	Count	Key spec	Power draw	Source
Grace-Blackwell compute trays	18	~6.5 kW each	117 kW total	Supermicro 2025
NVLink-5 switch trays	9	130 TB/s aggregate fabric	3.6 kW total	Supermicro 2025
Power shelves	8	132 kW total DC output	0.8 kW overhead	Supermicro 2025
Bluefield-3 DPUs	18	Storage & security offload	Included in compute	The Register 2024

AI's Insatiable Need for Compute, Memory, & Electricity



ELECTRICITY

“AI workloads could consume 80% of all compute cycles and 10% of global electricity use by 2030”*

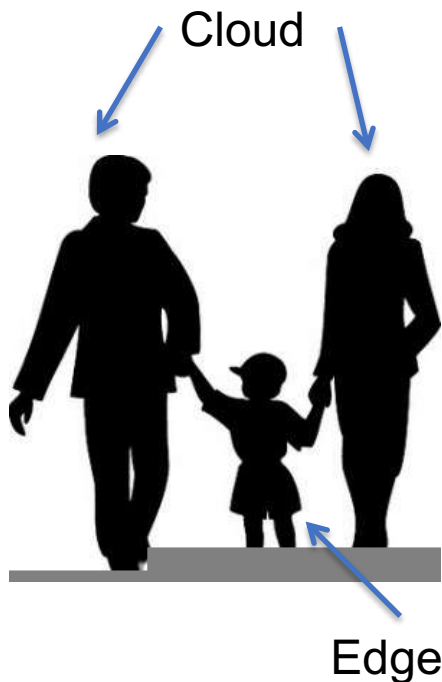
Model	Example Variant	Parameters	~Size
ResNet	ResNet50	~25 million	~100 MB
MobileNet	MobileNetV2	~3.4 million	~14 MB
YOLO	YOLOv11-small (approx)	~10–20 million	~40–80 MB
Transformer	BERT-base	~110 million	~420 MB
Transformer	GPT-2 small	~124 million	~500 MB
LSTM	2-layer LSTM (typical NLP)	~10–30 million	~40–120 MB

Model	Parameters
MobileNet	~3M
ResNet50	~25M
BERT-base	~110M
GPT models	Billions!!!!

* Source: International Energy Agency, (www.iea.org)

But, it IS time for the edge to grow up!

The Past...



The Future...



Part 2: The Education Era



Lets start building the basics!

Helping Edge Intelligence Grow Up – The C's and O's!

- Shift computation towards the edge of the network
 - improve time-to-action and reduce latency
 - greater privacy and security
 - lower cost
- Challenges on the edge
 - limited compute resources (low-power devices)
 - low-memory
 - limited storage
- There is a need for optimization of the model and the structure based on the:
 - device H/W (embedded, FPGA, CPU, GPU) ,
 - algorithm,
 - task (e.g. pattern recognition, classification, decision support, control output, etc.)

Still Embedded Systems Design 101!

Device related constraints:

- **Power Efficiency**
- **Cooling**
- **Battery**
- **Cost (overheads)**
- **Robustness**
- **Speed (latency)**
- **Physical Size...**
- **Implementation**
- **Prototyping**
- **Real-World Deployment**
- **Validation**
- **Evaluation**
- **...**

Designing Neural Networks for Edge Intelligence

- **Deep Neural Networks**

- decision making and data analytics
- **high** computation and memory requirements

- **Current Trends**

- **Design & Train a CNN to achieve max. accuracy & THEN Optimize for Edge:**
 - reduce the bit-width precision of the network parameters and the processed data
 - use binary values
 - use compression techniques with quantization and pruning to reduce the network demand for memory and storage

¹M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "Xnor-net:Imagenet classification using binary convolutional neural networks,"

²Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural network with pruning, trained quantization and Huffman coding"

Energy-Efficient Deep Learning

Accelerators, ISAs,
Architectures, Post
Von-Neuman
Computing, etc.

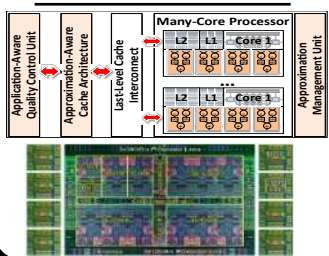
SW/Algo Level



Pruning, Quantization,
Knowledge Distillation, Low
Rank Factorization, etc.

Post-CMOS functional units,
Memristive X-Bars, Analog & Mixed-
Signal, hybrid processing units,
Neuromorphic, etc.

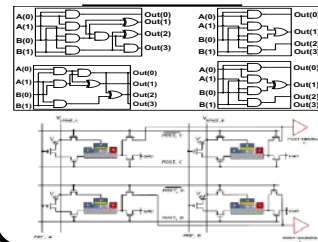
Architecture



Accuracy vs. Energy
Trade-Offs

Credit: M. Shafique

Circuits



Trends

Credit: M. Shafique

1 μ J/neuron



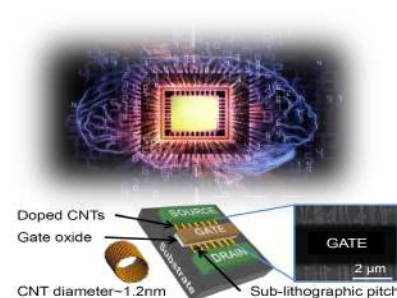
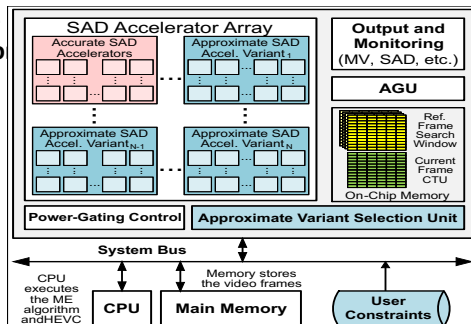
Neuromorphic
Architectures
and *Beyond*

Post-CMOS
Technologies

Deep Learning
Architectures

Approximate
Computing

Software
(Multi-Co
GPUs)

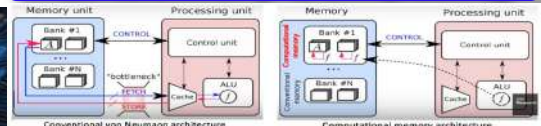


Source: S. Mitra, P. Wong (Stanford), C. Mackin (MIT), J. Zhang (Google)

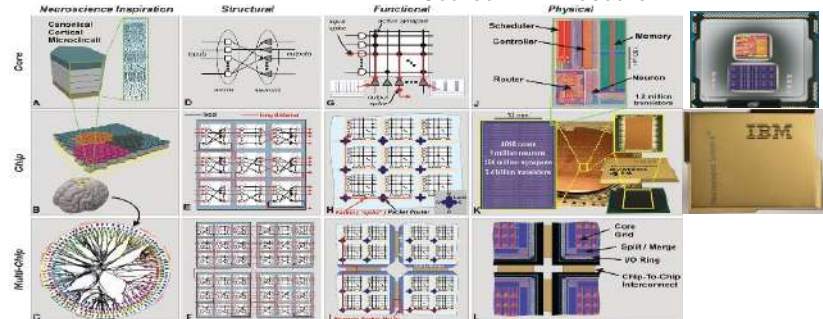
In-Memory Computing



Source: Huawei Kirin Chip



Source: IBM Research



Source: IBM, TrueNorth Chip

"Edge Intelligence: Time for the Edge to Grow Up!"

Energy and Performance-Efficient DNNs

Software (algorithmic) techniques

- Pruning
- Knowledge Distillation
- Low-Ranked Factorization
- Quantization

Hardware based techniques

- Numerical representation
- Approximate Computing
- Post Von Neumann
- Event-Triggered Computing

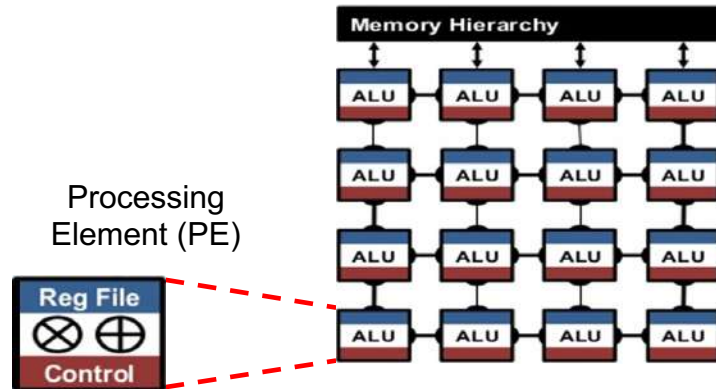
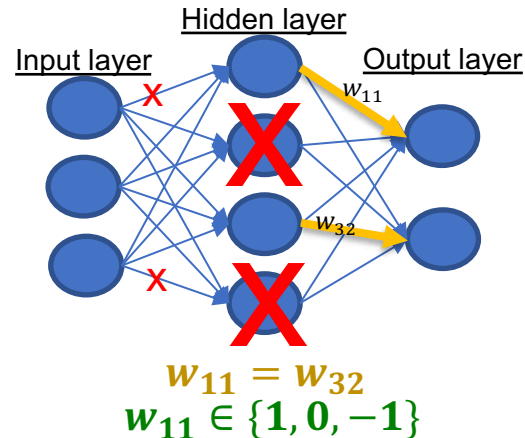
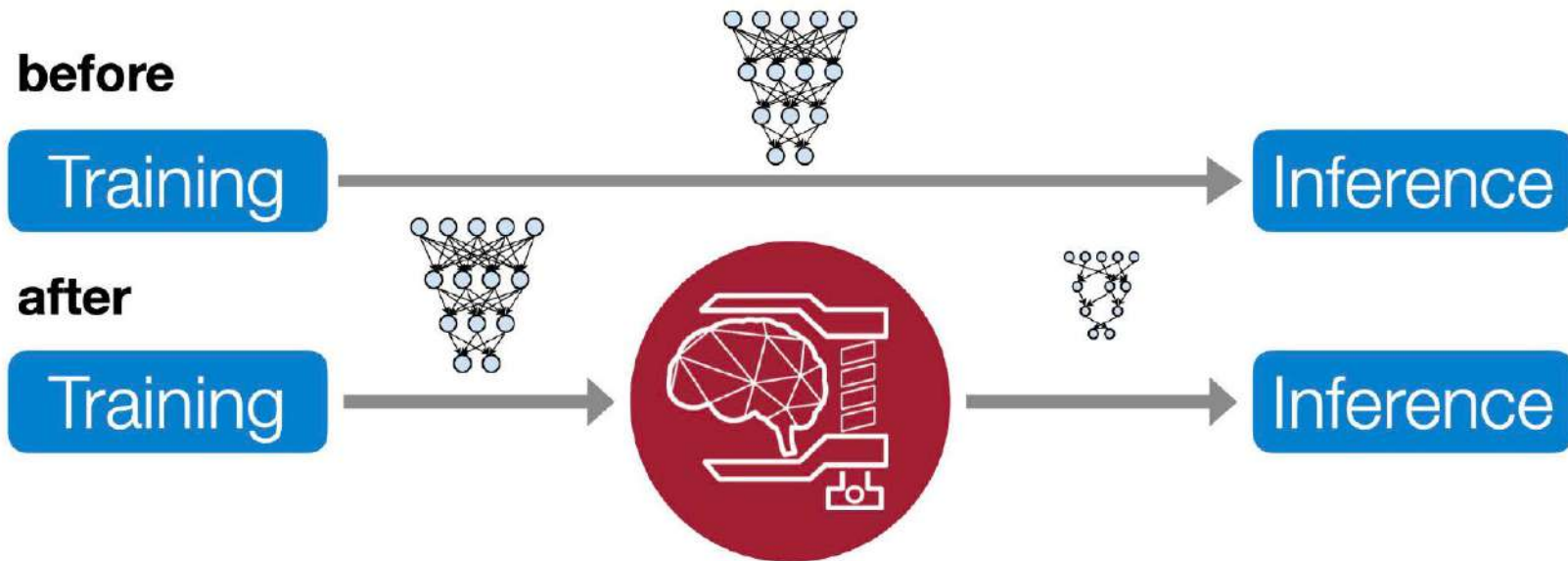


Image Source: Eyeriss@MIT

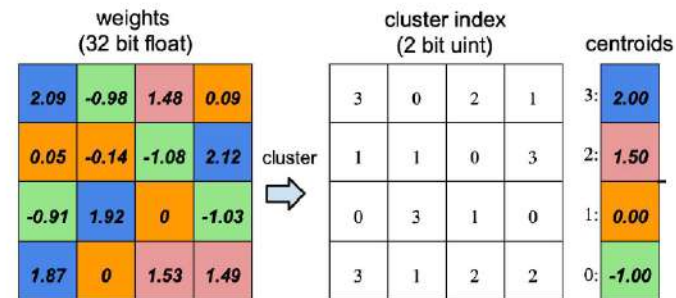
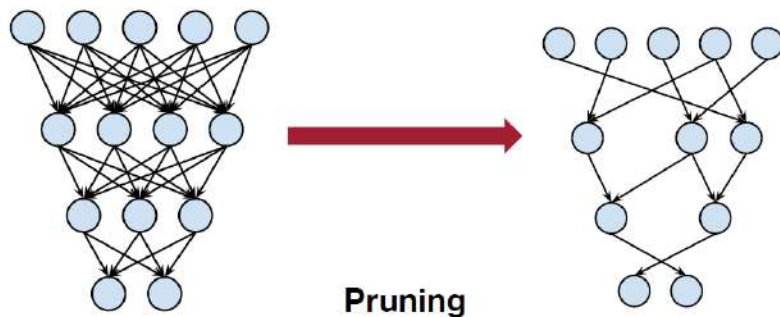
Model Compression



Model compression:
Pruning, sparsity, quantization, etc

Credit: Song Han, MIT

Compression Techniques



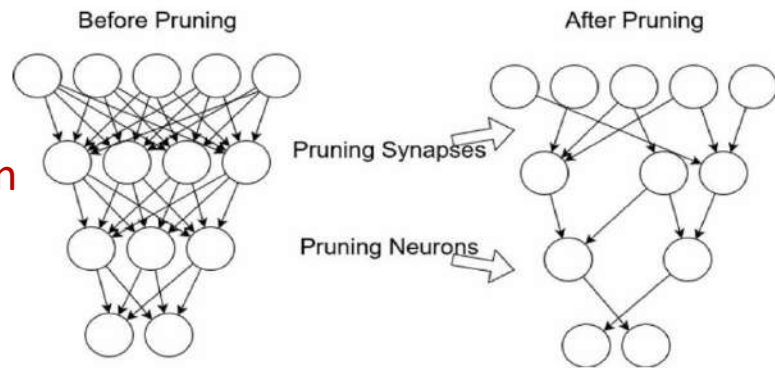
Quantization



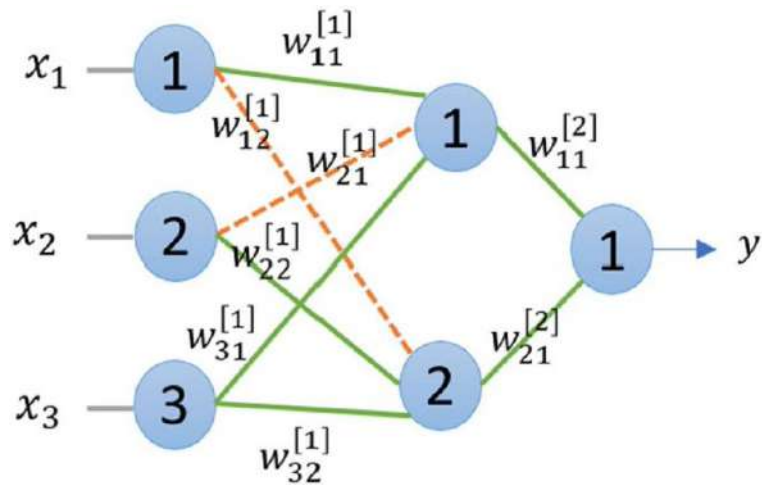
Credit: Song Han, MIT

Pruning - I

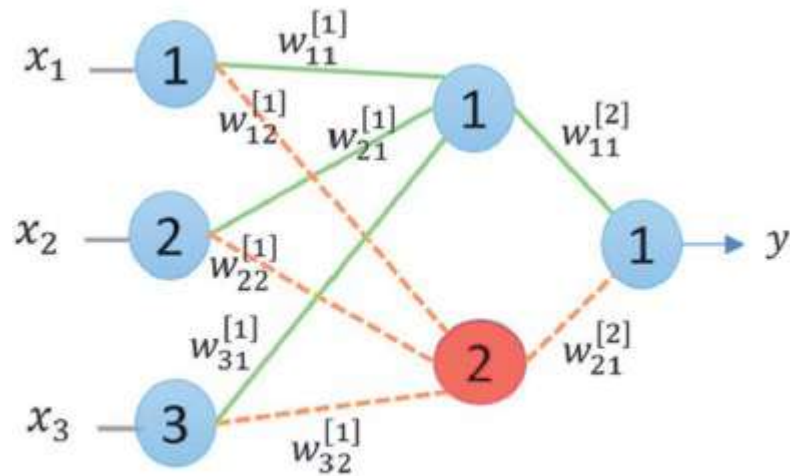
- **Compresses** the model to a smaller size with zero or marginal loss of accuracy.
- **Reduces the number of parameters** by removing redundant, and unimportant connections that do not affect model accuracy.
- Results in **compressed neural networks that run faster**.
- **Reduces the computational cost** involved in network training.
- Reduces the overall model size, saves computation time and energy.



Pruning - II

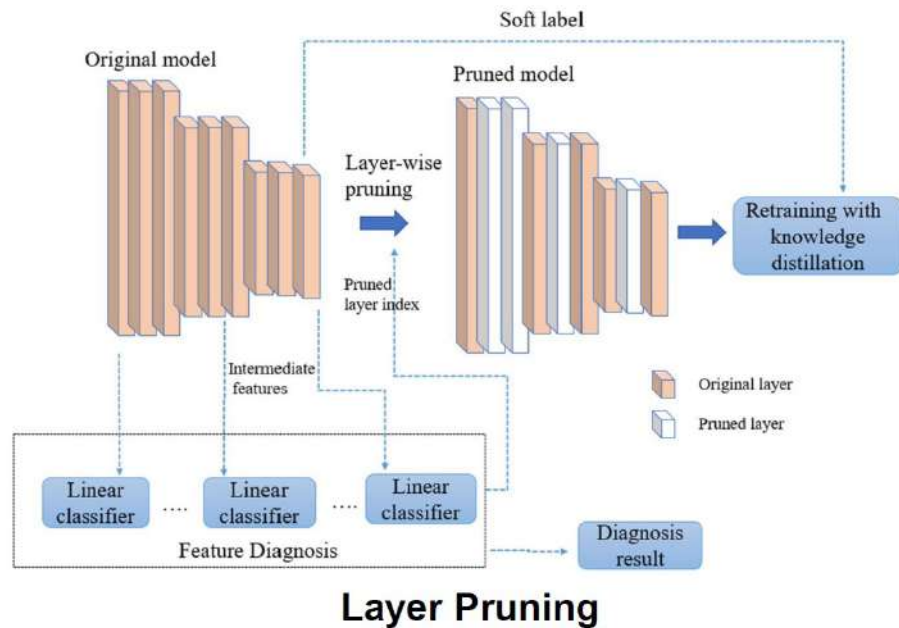
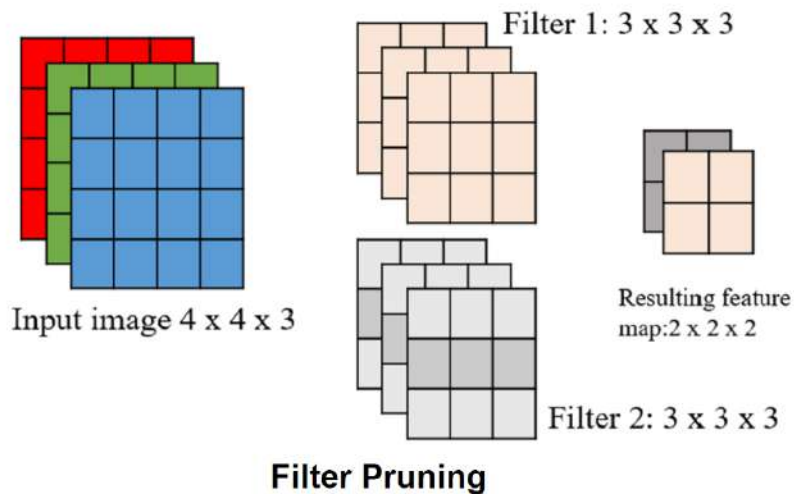


Weight Pruning



Neuron Pruning

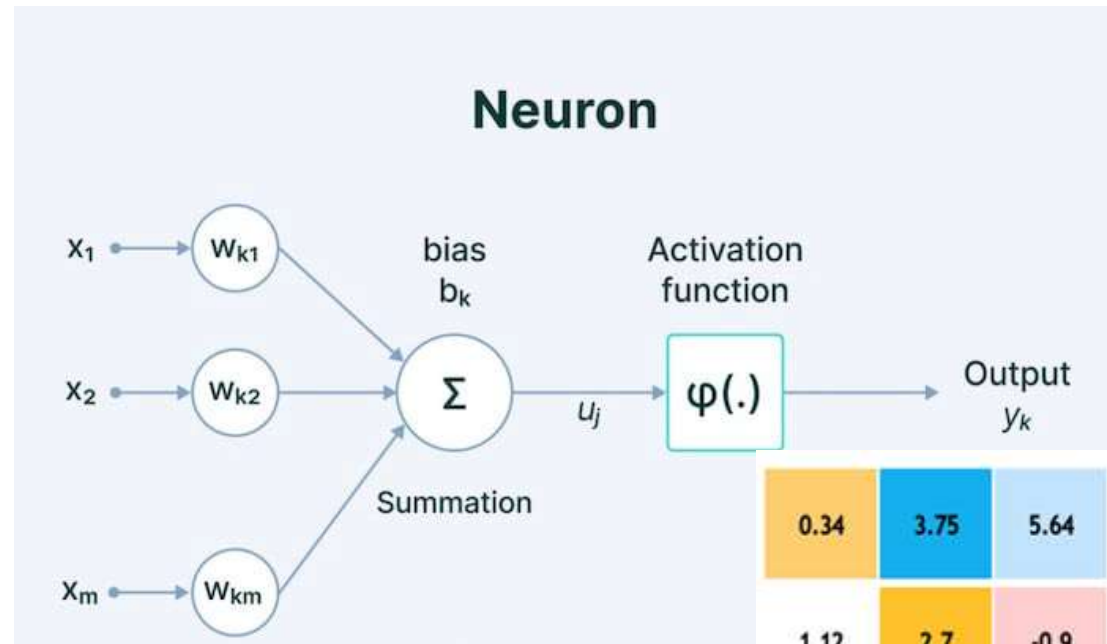
Pruning - III



Pruning - Summary

- ❑ Can be applied during or after training.
- ❑ Can be applied to both convolutional and fully connected layers.
- ❑ Both the original and pruned model have the same architecture, with the pruned model being sparser (weights with the low magnitude being set to zeros).
- ❑ Pruned models sometimes outperform the original architecture but rarely outperform a better architecture.

Quantization - I



0.34	3.75	5.64
1.12	2.7	-0.9
-4.7	0.68	1.43

FP32

Floating point

3452.3194

Integer

3452

32 bit

01010101	01010101
01010101	01010101

8 bit

01010101



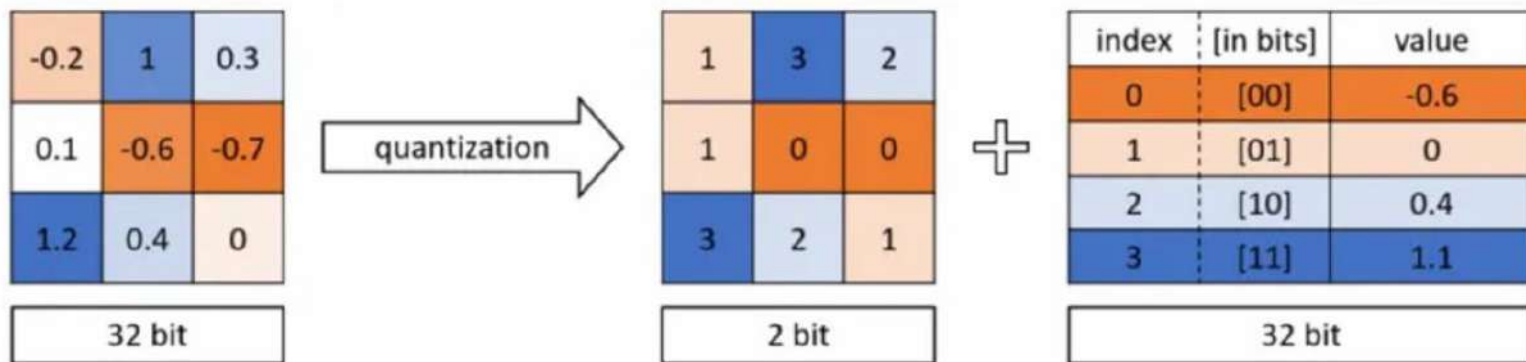
Quantization

64	134	217
76	119	21
3	81	99

INT8

Quantization - II

- Compresses the original network by **reducing the number of bits required to represent each weight**.
- The weights can be quantized to 16-bit, 8-bit, 4-bit or even with 1-bit.
- Per-channel quantization and/or per-layer quantization of weights.
- By reducing the number of bits used, the size of the deep neural network can be significantly reduced.
- Reduces inference time for hardware platforms supporting lower precision arithmetic



Quantization - III

- ❑ Two types of quantization:
 - Post-training quantization
 - Quantization aware training

Model	Non-quantized Top-1 Accuracy	8-bit Quantized Accuracy
MobilenetV1 224	71.03%	71.06%
Resnet v1 50	76.3%	76.1%
MobilenetV2 224	70.77%	70.01%

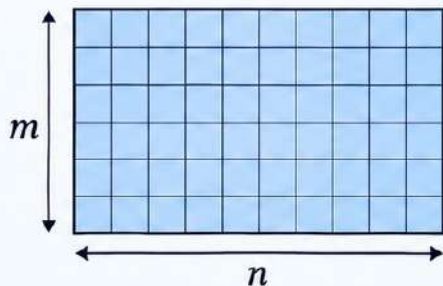
Quantization - Summary

- ❑ Quantization can be applied both **during** and **after** training.
- ❑ Can be applied to both **convolutional** and **fully connected** layers.
- ❑ Quantized weights **make neural networks harder to converge**.
A **smaller learning rate is needed** for the network to have good performance. ← ***Training Issue, but affects inference!***
- ❑ Quantized weights make back-propagation infeasible since gradient cannot back-propagate through discrete neurons.
Approximation methods are needed to estimate the gradients of the loss function with respect to the input of the discrete neurons.

Low Rank Factorization (PCA-Alike)

1. Original weight matrix

$$W \in \mathbb{R}^{m \times n}$$

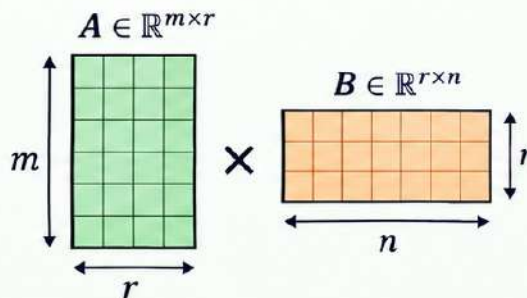


Number of parameters:

$$m \cdot n$$

2. Low-rank factorization

$$W \approx A \times B$$



Number of parameters:

$$r(m + n)$$

3. Choose a small rank

Choose $r \ll \min(m, n)$

With r much smaller than $\min(m, n)$:

$$r(m + n) \ll m \cdot n$$

Compression intuition

- ✓ A and B capture the most important (low-rank) structure of W .
- ✓ Reduces storage from $m \cdot n$ to $r(m + n)$ parameters.
- ✓ Also reduces compute, since we can compute $(A \times B)x$ more efficiently than using the full W .

Trade-off: rank r

Smaller r

- More compression (fewer parameters)
- Lower compute
- May lose more accuracy

Choose r based on your storage, compute, and accuracy needs.

Larger r

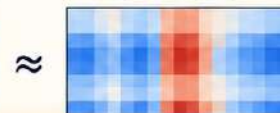
- Better accuracy (closer to original W)
- Less compression
- Higher compute

Visual intuition: capturing the most important structure

Original weight matrix W
($m \times n$)



Low-rank approximation $A \times B$
($m \times n$, rank r)



≈

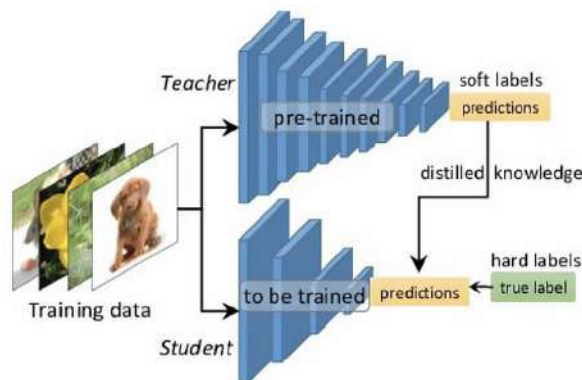
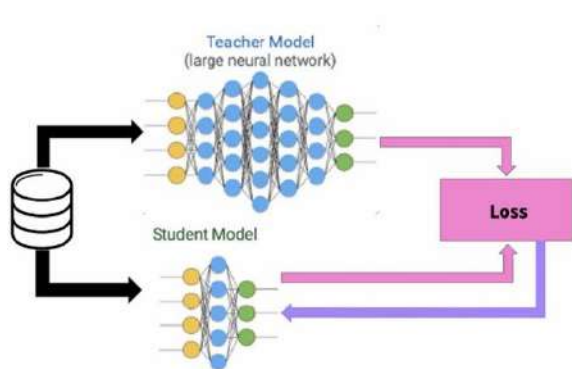
The factorized form captures the dominant patterns of W (e.g., key features and correlations) while discarding less important details, leading to a much smaller, more efficient model.

Low Rank Factorization

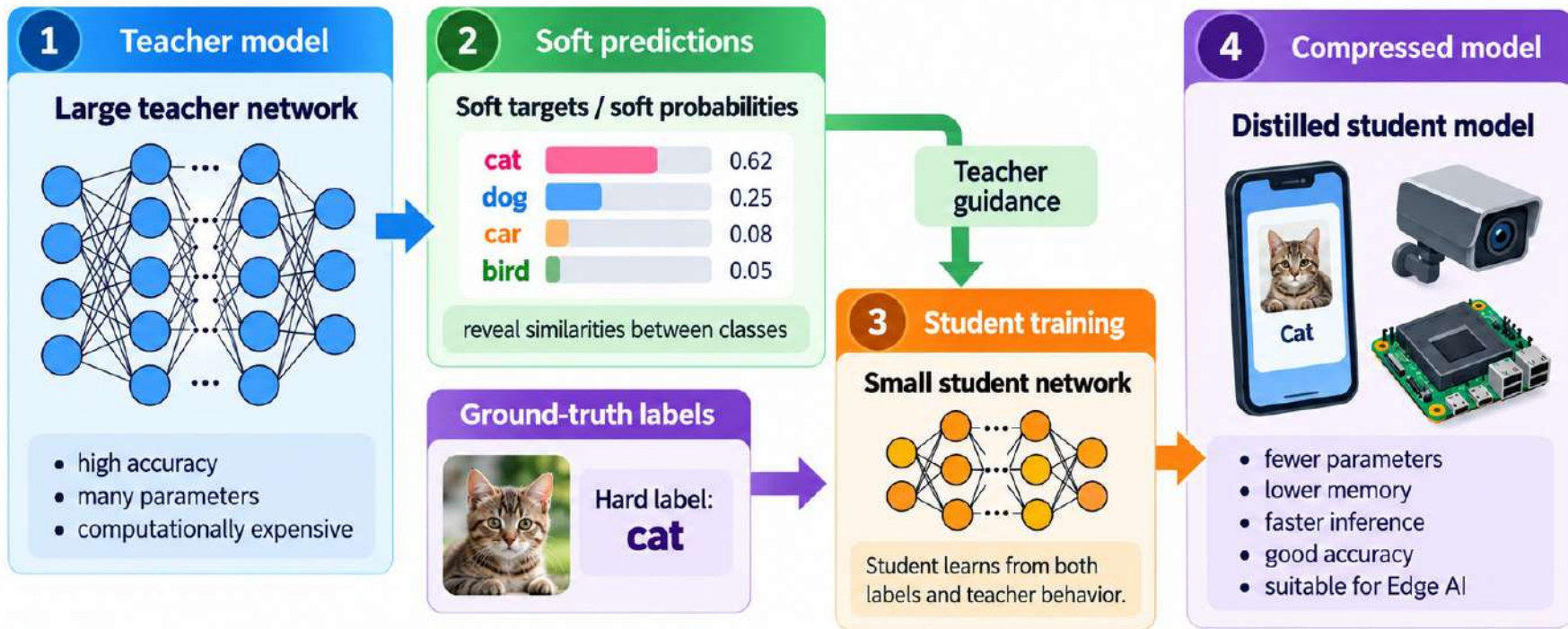
- ❑ Low-rank factorization identifies redundant parameters of deep neural networks by employing the matrix and tensor decomposition
- ❑ A weight matrix A with $m \times n$ dimension and having a rank r is replaced by smaller dimension matrices.
- ❑ This technique helps by factorizing a large matrix into smaller matrices.
- ❑ Can be applied during or after training.
- ❑ Can be applied to both convolutional and fully connected layers.
- ❑ When applied during training, can reduce training time.
- ❑ The factorization of the dense layer matrices reduces model size.
- ❑ Improves the speed up-to 30-50% compared to the full-rank matrix representation.

Knowledge Distillation - I

- A large, complex model is trained on a large dataset.
- When this large model can generalize and perform well on unseen data, the knowledge is transferred to a smaller network.
 - The larger model is also known as the teacher model
 - Smaller network is also known as the student network.
- Knowledge types, distillation strategies and the teacher-student architectures play the crucial role in the student learning.



Knowledge Distillation - II



Knowledge distillation transfers knowledge from a large teacher to a compact student.

Model Compression - Summary

- ❑ Model compression techniques are **complementary** to each other.
- ❑ Can be applied *to pre-trained models, as a post-processing step to reduce your model size and increase inference speed.*
- ❑ Can be applied during training time as well.
- ❑ Pruning and quantization have now been integrated into frameworks such as Tensorflow and PyTorch.
- ❑ There are many more compression approaches (weight sharing-based model compression, structural matrix, transferred filters, and compact filters)



Energy and Performance-Efficient DNNs

Software based techniques

- Pruning

- Knowledge Distillation

- Low-Ranked Factorization

- Quantization

Hardware based techniques

- Fixed point representation

- Limited fixed point precision

- System array based designs

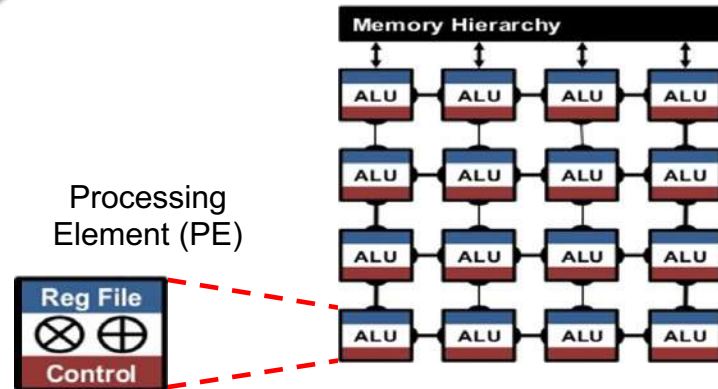
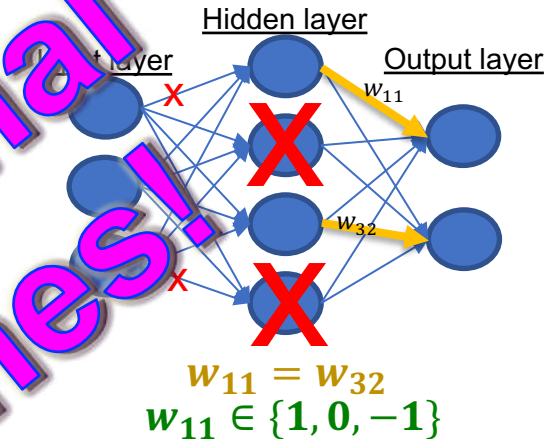


Image Source: Eyeriss@MIT

4K TRAFFIC SCENE

3840 x 2160 | ~8.3 Million Pixels

Massive visual information
Every region may contain
objects to detect



ONBOARD EDGE PLATFORM



Limited Compute



Limited Memory



Limited Energy



Thermal Constraint

EDGE UAV CONSTRAINTS



COMPUTE LOAD

Very High



MEMORY PRESSURE

High



DATA VOLUME

Very High



ENERGY CONSUMPTION

High



LATENCY REQUIREMENT

Tight



RAW DATA VOLUME

~24-30 MB per frame (4K)

At 30 FPS → ~0.7-0.9 GB/s

Exhaustive CNN processing
on edge is computationally
and energetically expensive

Part 3: The Entering the Real World Era!



Deployment and Cyber-Physical Integration

Computational Efficiency Gap: AI vs. Humans

Deep Blue vs. Garry Kasparov



IBM RS/6000 SP high-performance computer

900W + Power for 480
dedicated ASICs

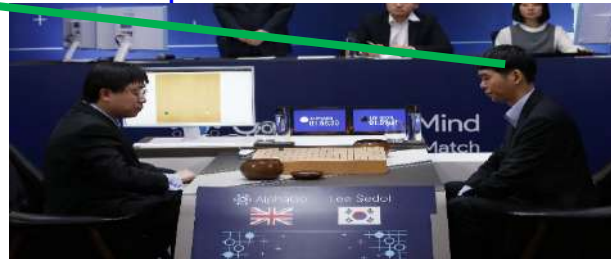
30 P2SC nodes +
480 dedicated
ASICs to play
chess

20W

1MW

1,920 CPUs and
280 GPUs

AlphaGO vs. Lee Sedol



Google's cloud computing

1996

Chess

2011

Jeopardy

2016

GO

IBM Watson vs. Brad Rutter and Ken Jennings

200KW

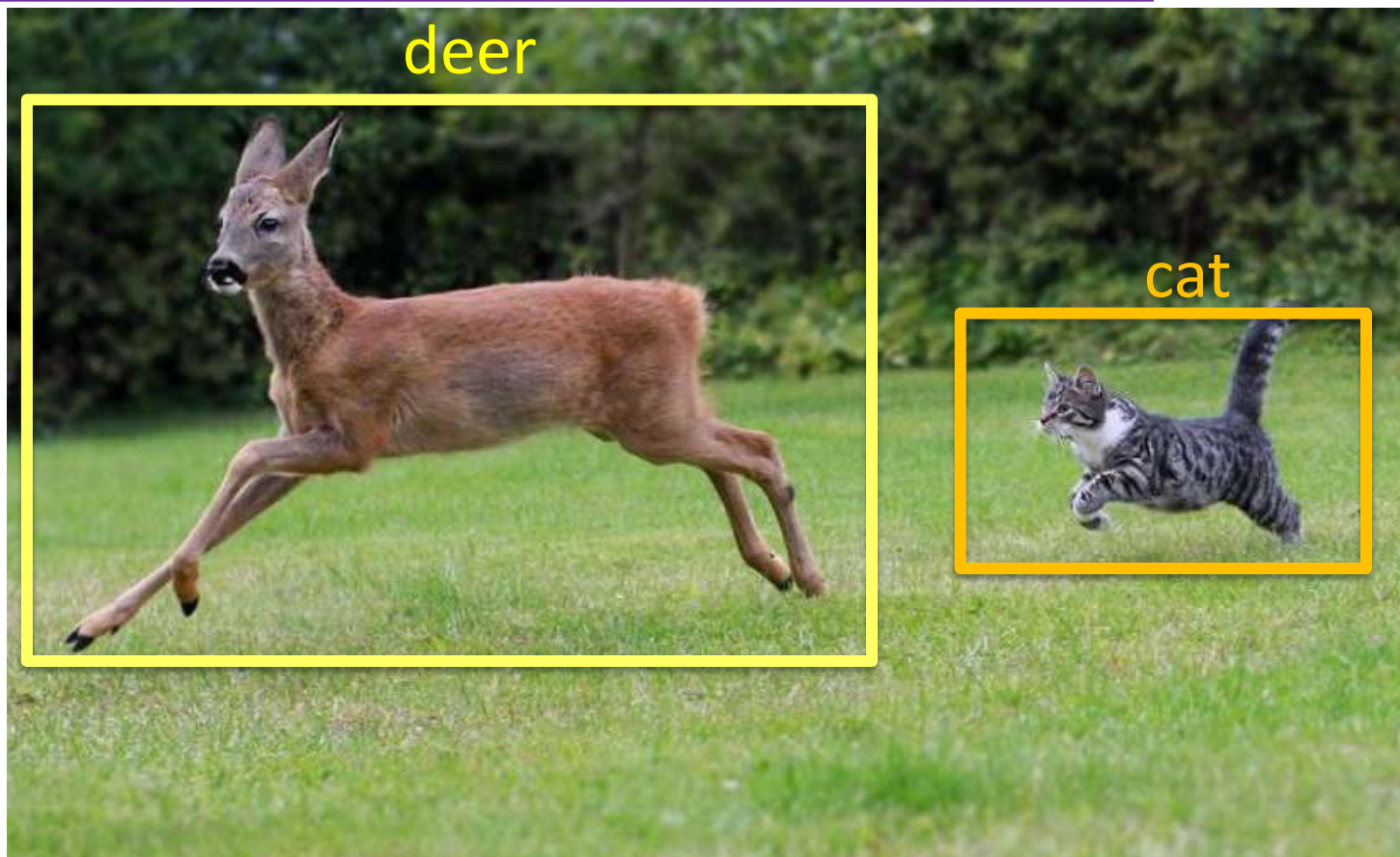
2,880 POWER7 processor
threads and 16 terabytes of
RAM



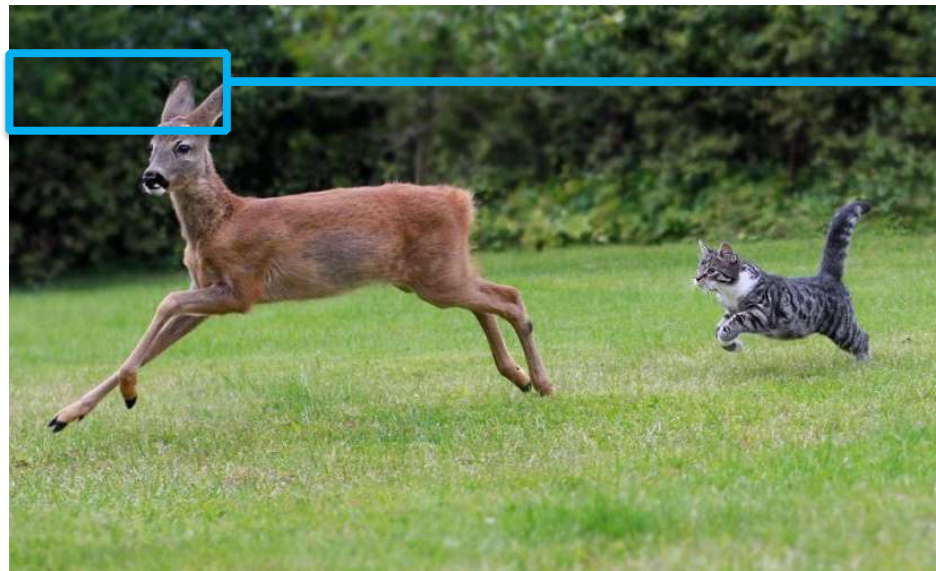
IBM Blue Gene supercomputer

"Edge Intelligence: Time for the Edge to Grow Up!"

Object Detection – A Closer Look



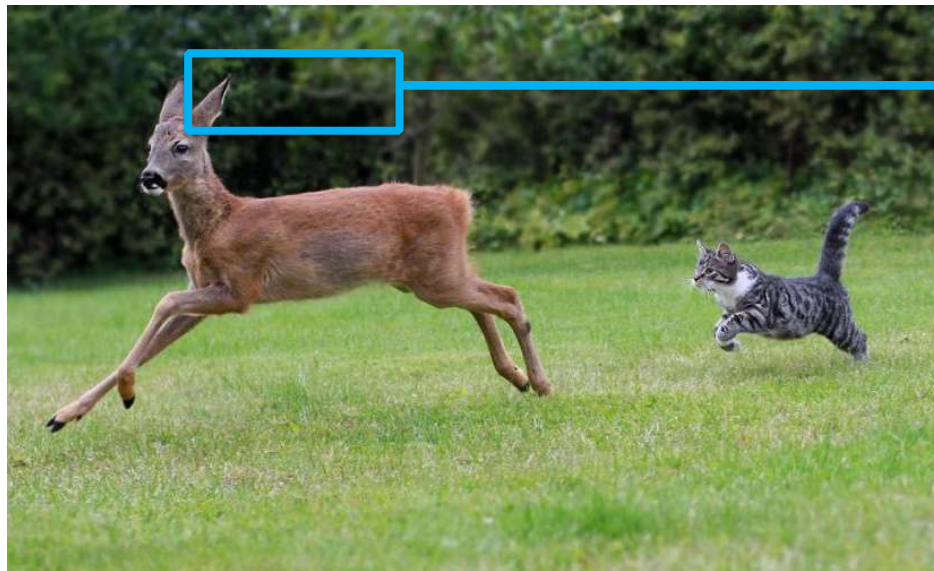
Object Detection as Classification



O.D.

deer?
cat?
background?

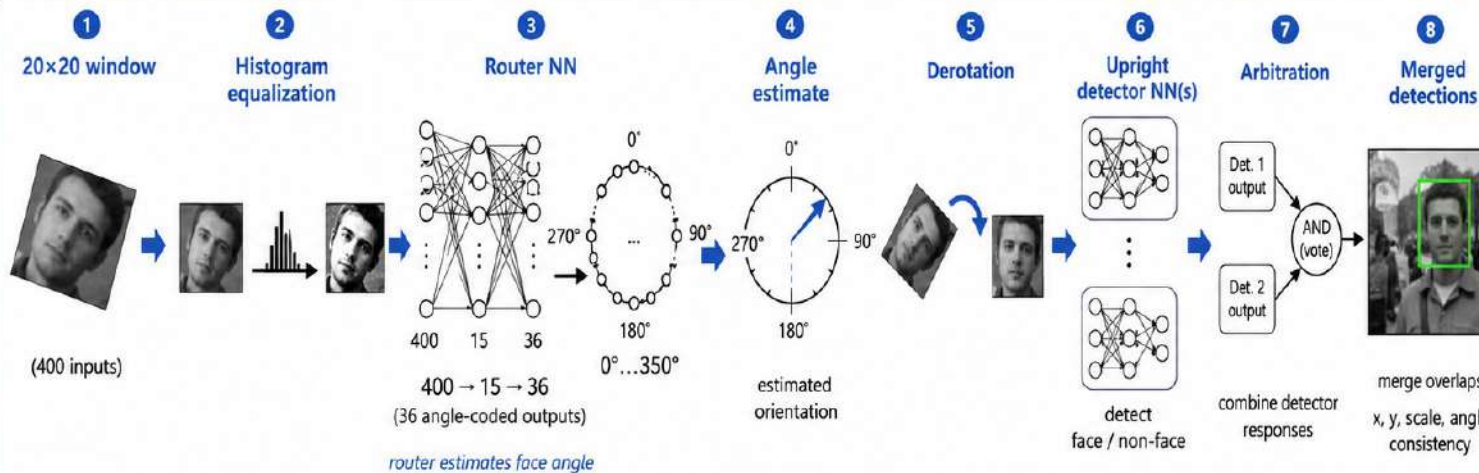
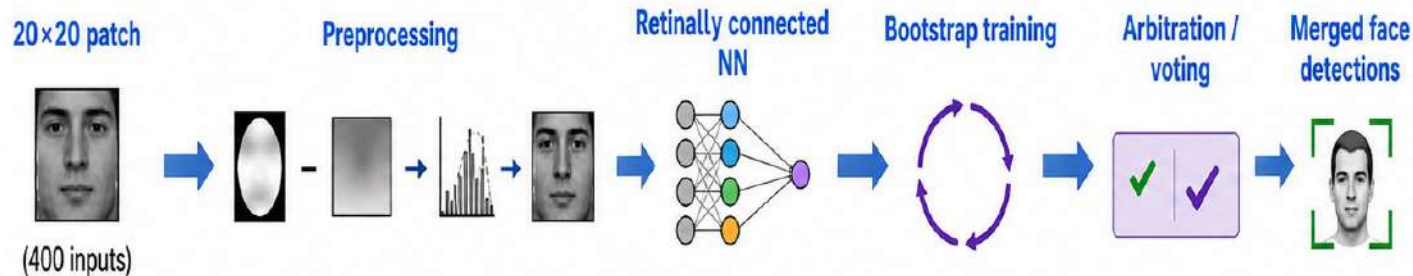
Object Detection as Classification with Sliding Window



O.D.

deer?
cat?
background?

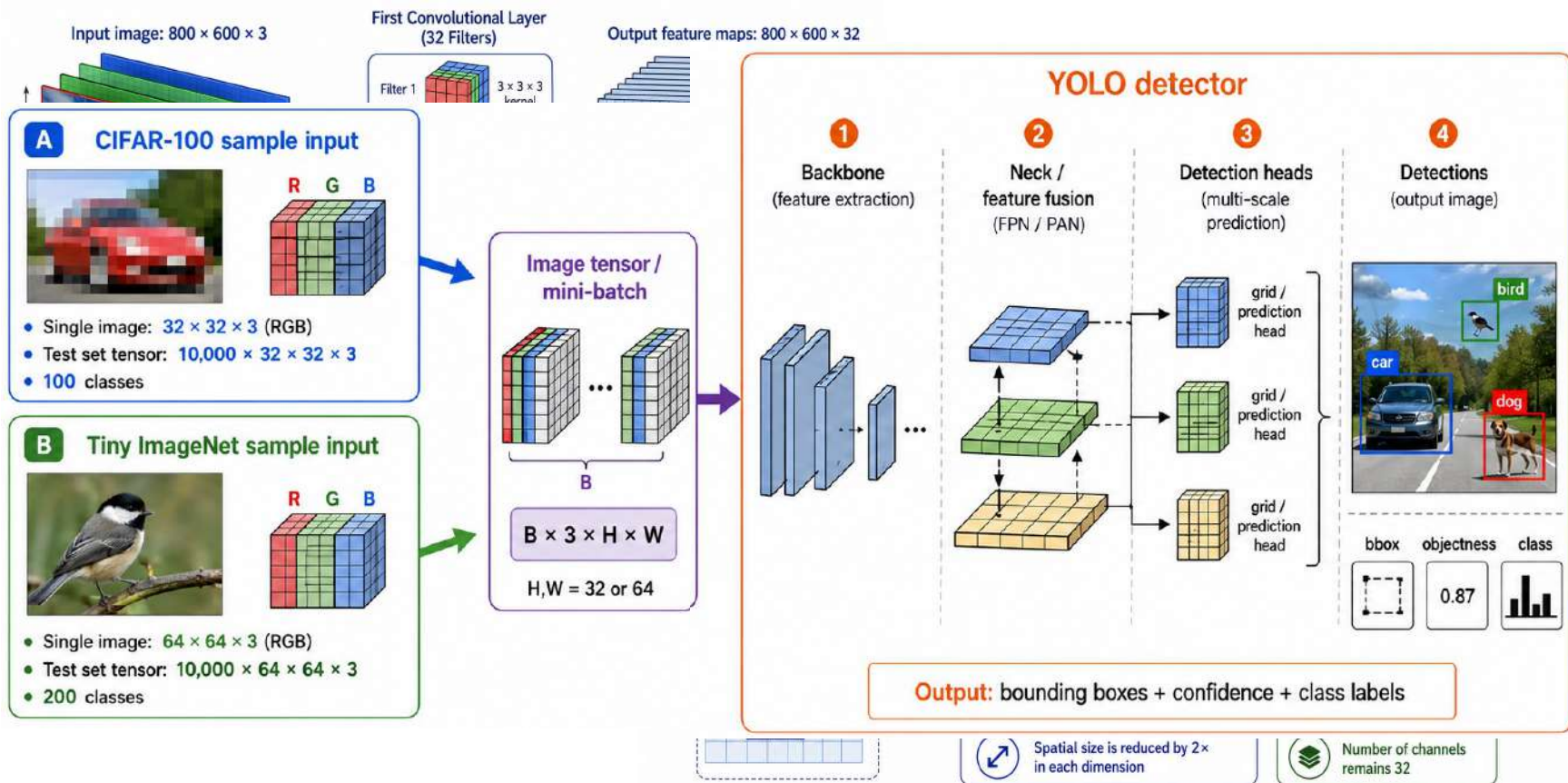
Throwback: Rowley and Kanade (PAMI, 1998)



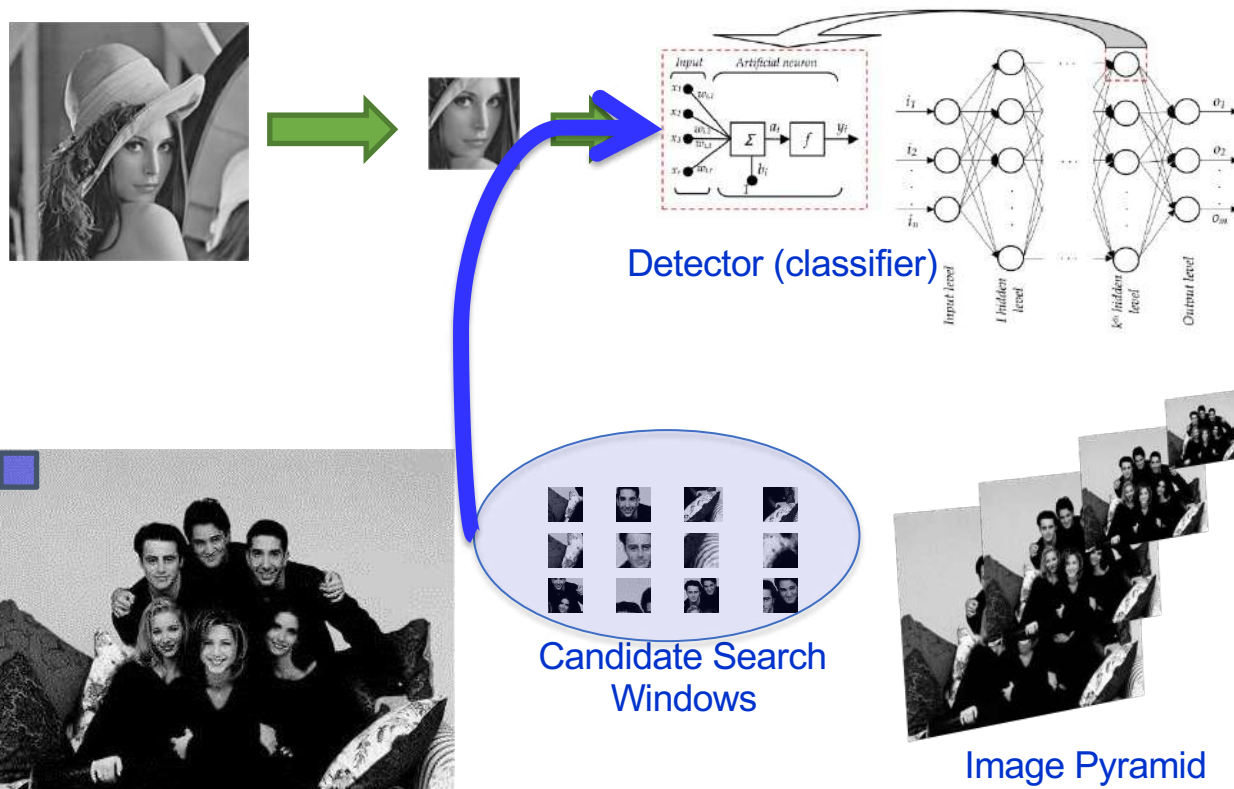
Rotation-invariant face detector (Rowley-Baluja-Kanade, 1998):

router estimates angle → derotate window → apply upright detector(s) → arbitrate and merge detections.

As engineers, we like to build bottom-up!



Object Detection & Sliding Window Search



Candidate Search Windows → Detector

What the ANN/CNN detector sees...

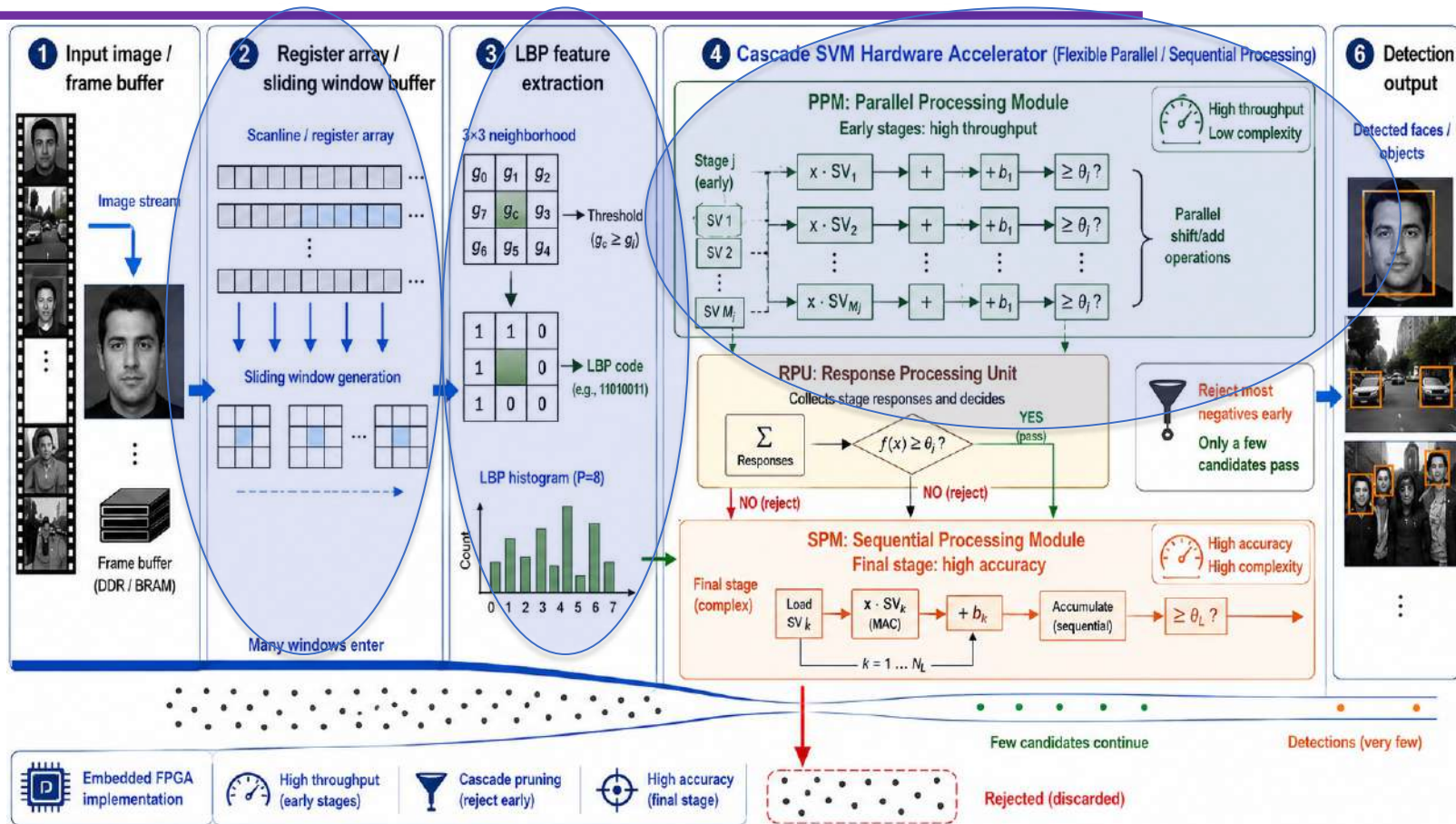


"Edge Intelligence: Time for the Edge to Grow Up!"

Can we speed things up?



H/W Friendly Approaches



Introducing Cascades

Baseline early cascade stage

Support Vectors (SVs)

SV index	x_1	x_2	...	x_d
1	0.3125	-0.8750	...	0.5313
2	-0.6250	0.4375	...	-0.3125
$\vdots$	$\vdots$	$\vdots$	...	$\vdots$
N_{SV}	0.7188	-0.1563	...	0.2813

Stored as ordinary fixed-point values

Alpha Coefficients (α)

SV index	α (fixed-point)
1	0.187500
2	-0.562500
$\vdots$	$\vdots$
N_{SV}	0.093750

Stored as ordinary fixed-point values

SVM Decision (Linear Kernel Example)

Input Feature Vector

$$x = [x_1 \ x_2 \ \dots \ x_d]^T$$

x_1
 x_2
 $\vdots$
 x_d

For each support vector i

$$\begin{aligned} \times \quad x_1 \times SV_{i1} \\ \times \quad x_2 \times SV_{i2} \\ \vdots \\ \times \quad x_d \times SV_{id} \end{aligned}$$

Σ
(or
(d terms)

$\times$

Σ

Decision
 $f(x) = \sum_i \alpha_i (SV_i \cdot x) + b$

Accumulate
over i

$\times$ = Multiplier

Multiplication-heavy operations



More Multipliers



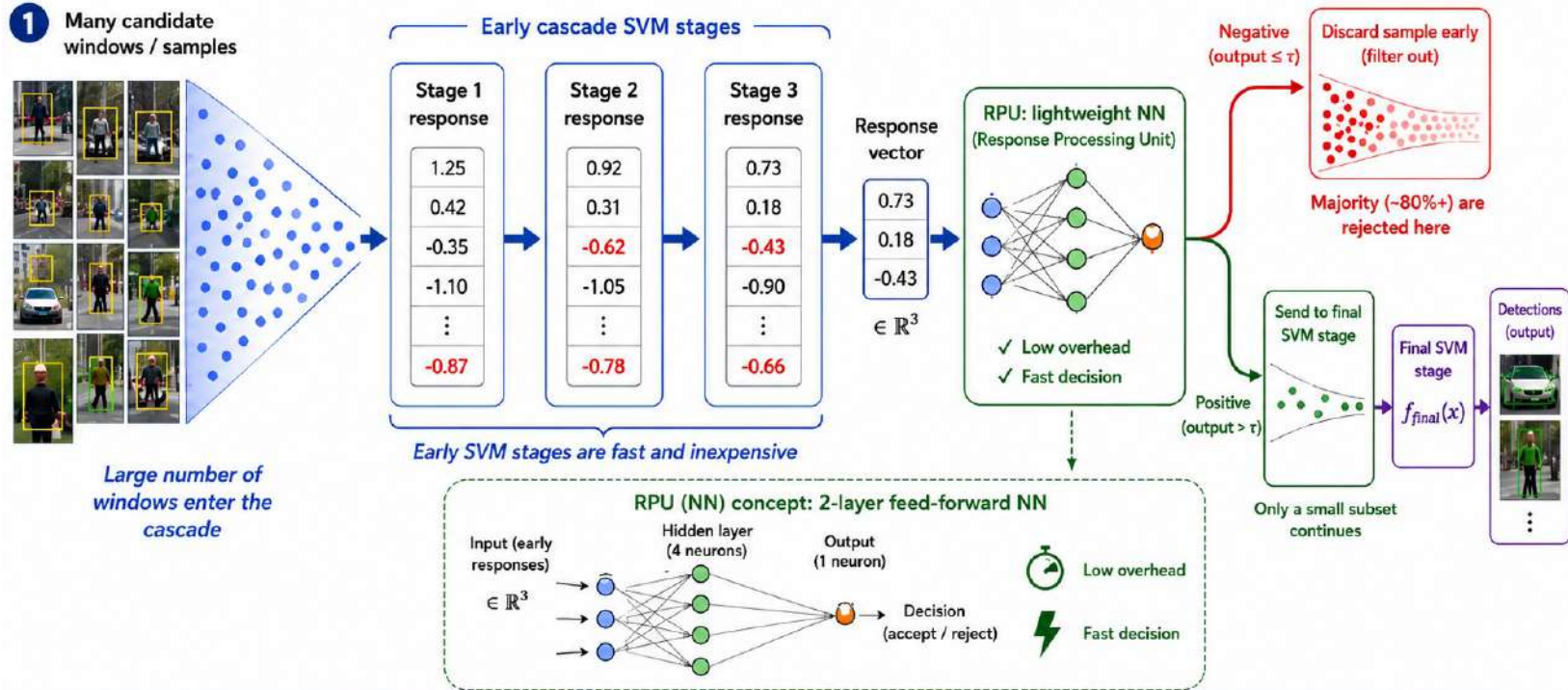
Larger Memory



Higher Power

Round SVs and
alpha coefficients
to nearest
power of two

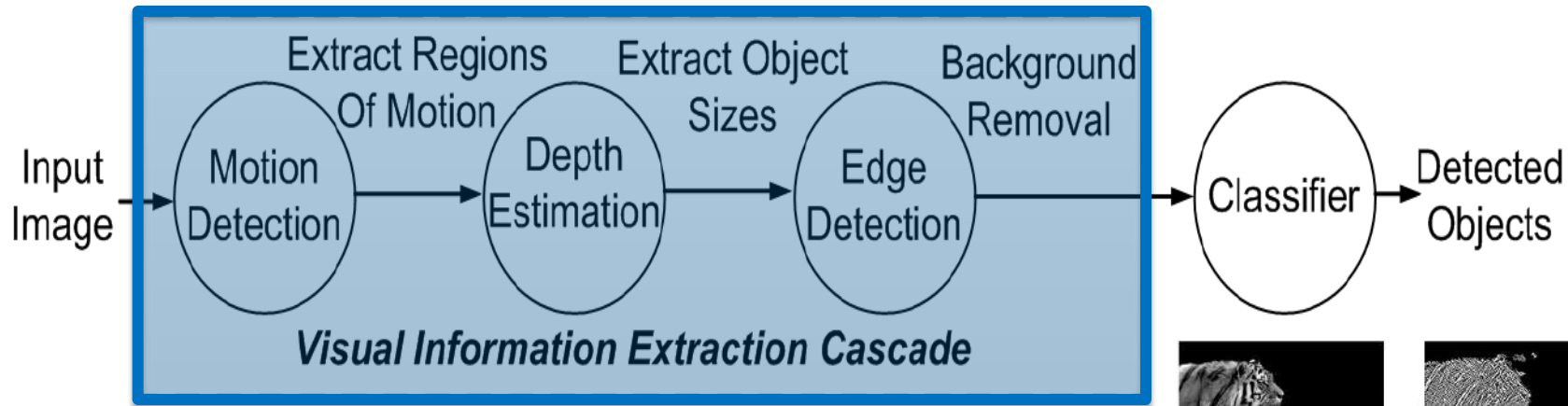
Cascades work well in non ANN-based models!



Kyrkou, C., Theodoridis, T., et al. Embedded hardware-efficient real-time classification with cascade support vector machines. *IEEE Transactions on Neural Networks and Learning Systems*, Vol 27, Issue 1, pp. 99-112.

Cascading is a form of laziness!

Visual Information Extraction Cascade



- ❑ Step 1: **motion** (focus on changes) (**Algorithm**)
- ❑ Step 2: **depth** (focus on object size) (**Sensor**)
- ❑ Step 3: **edge** (focus only where's a lot of information) (**Sensor+Algorithm**)
- ❑ Step 4: **Classify** *only what's necessary!* (**ANN Model**)

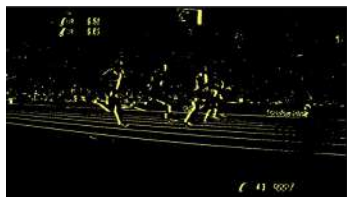
Process only what's necessary!



- ❑ A (very) high-speed 100m race!

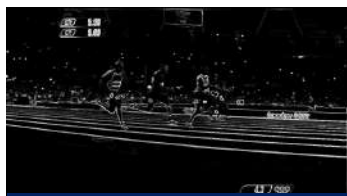


- ❑ Cluttered background



- ❑ Various illumination conditions

- ❑ 20-76% overall reduction from motion



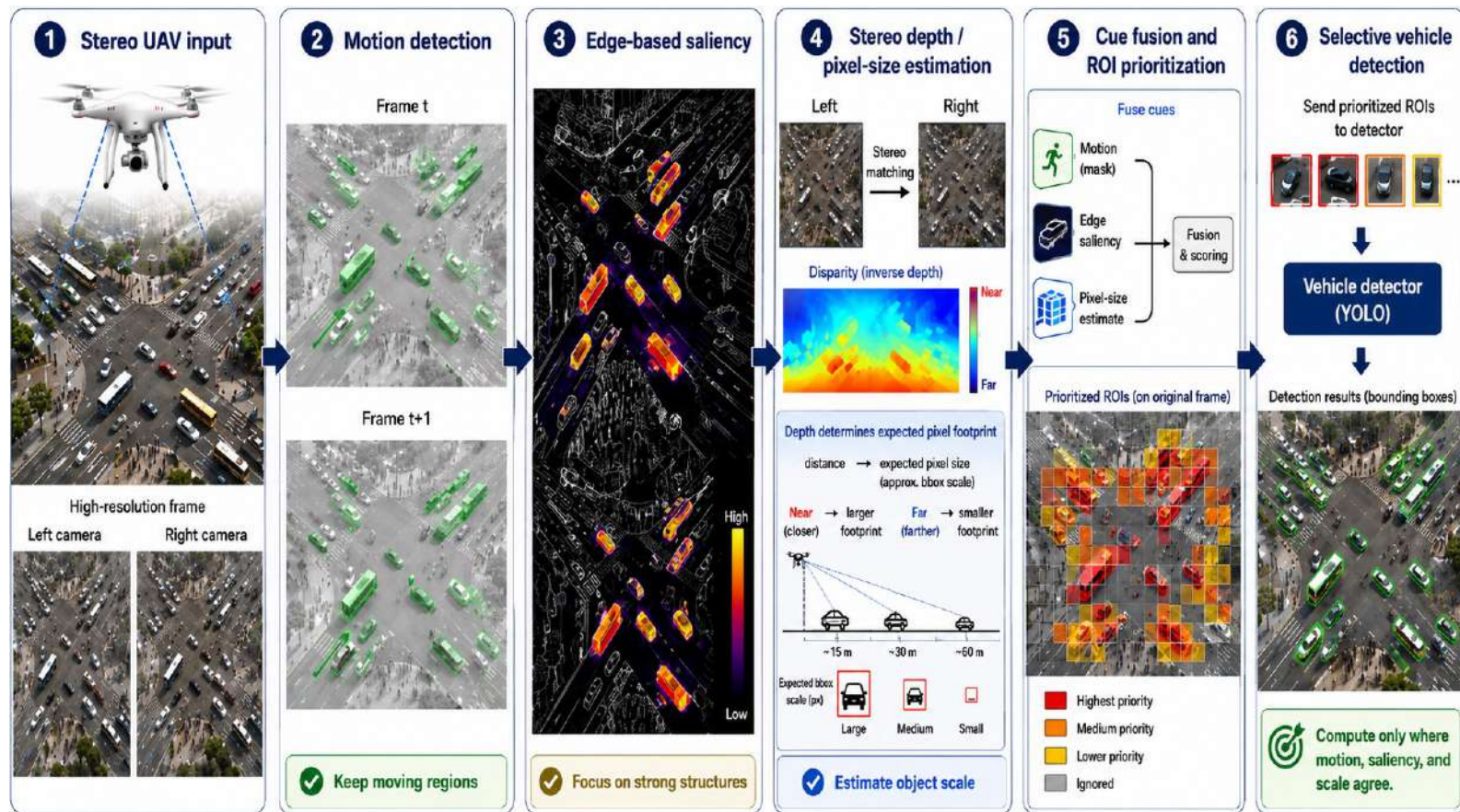
- ❑ 12-30% reduction from depth

- ❑ 15-20% reduction from edge

- ❑ ~1% of useful data reach the classifier

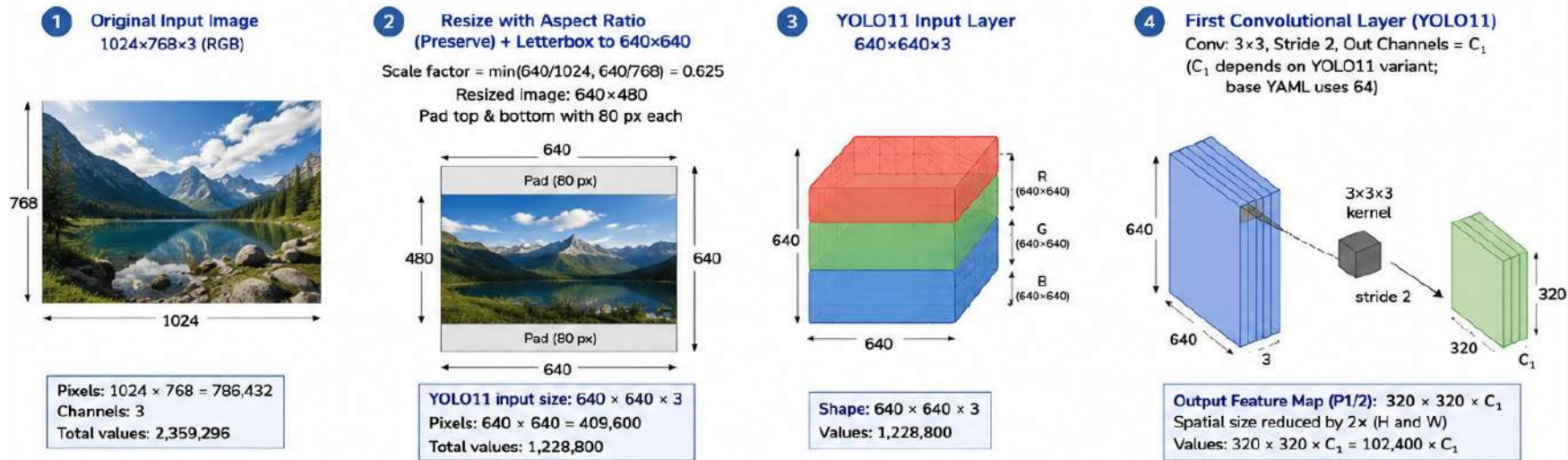
C. Kyrkou, T. Theodoridis "Accelerating object detection via a visual-feature-directed search cascade: algorithm and field programmable gate array implementation," *Journal of Electronic Imaging* 25(4), 041013 (3 Aug 2016) <https://doi.org/10.1117/1.JEI.25.4.041013>

Applied easily in our example problem as well!



Alternatively – Scale (modern ANN)Models Up!!!

YOLO – Single Shot Detector



Stride-2 convolution halves the spatial dimensions:
 $640 \times 640 \rightarrow 320 \times 320$

J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You Only Look Once: Unified, Real-Time Object Detection,” in Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR), Jun. 2016.

How can we handle increasing resolution?



Full HD

1920 × 1080



8K

7680 × 4320

Images grow!!! (SVGA → 8K)



	1024 × 768	1920 × 1080 (Full HD)	2560 × 1440 (2K / QHD)	3840 × 2160 (4K / UHD)	7680 × 4320 (8K / UHD-2)
Input size	1024 × 768 × 3	1920 × 1080 × 3	2560 × 1440 × 3	3840 × 2160 × 3	7680 × 4320 × 3
Input pixels	786,432	2,073,600	3,686,400	8,294,400	33,177,600

Σ MACs Formula

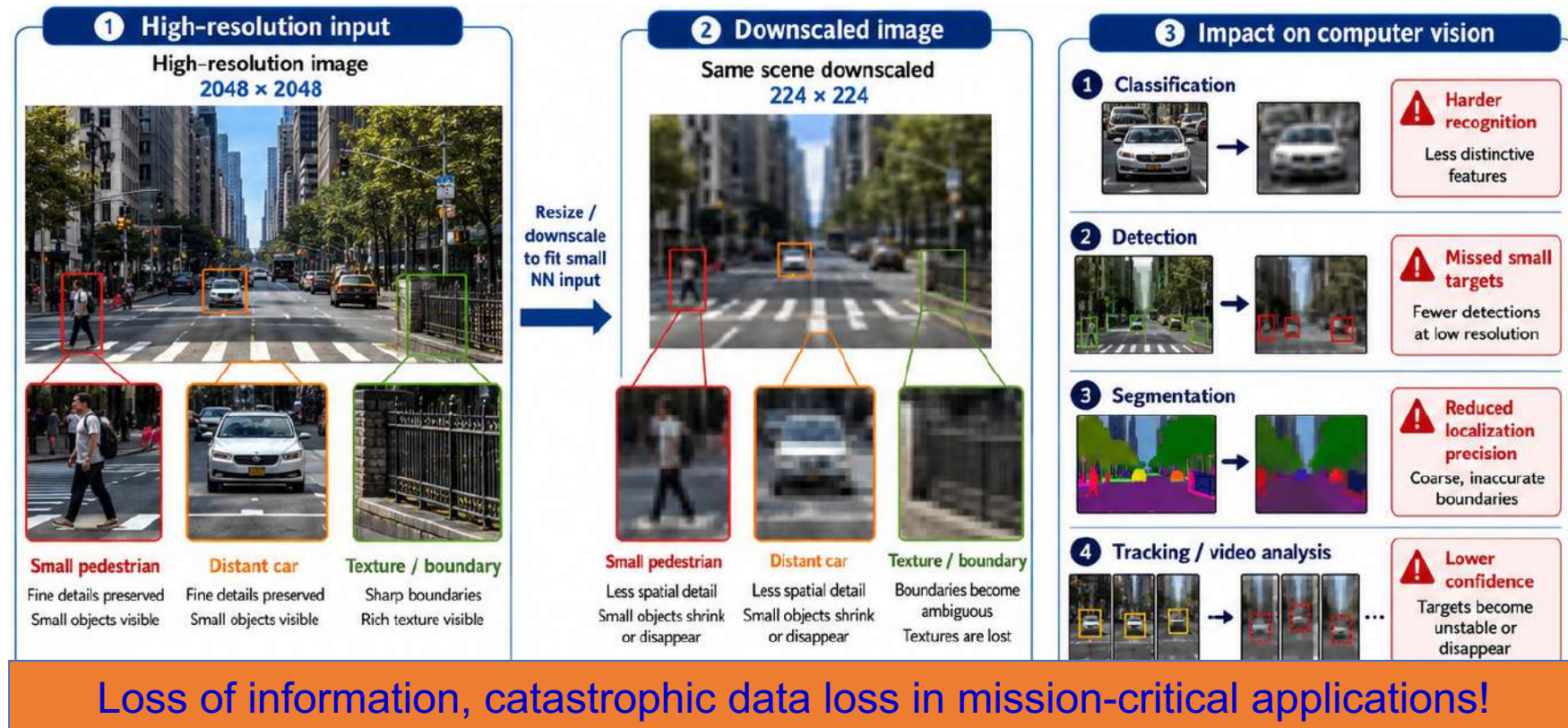
For a 3×3×3 first layer,

Going from 1024x768 to 8K has a **~42x** increase in computation, memory traffic → Latency, Power, Memory Bandwidth!!!

Relative first-layer complexity	1x	2.64x	4.69x	10.55x	42.19x
Relative activation memory	1x	2.64x	4.69x	10.55x	42.19x

Because stride = 2 and C_1 is fixed, complexity scales linearly with image resolution.

Downscaling causes data loss...



The challenges of high-resolution

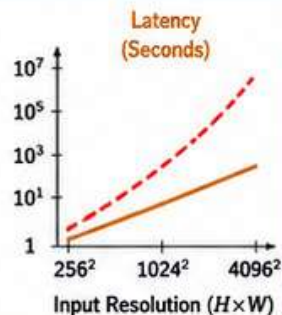
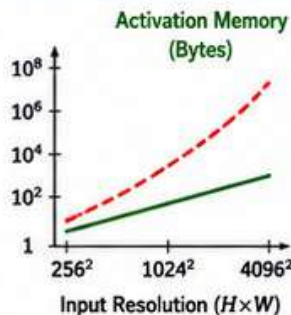
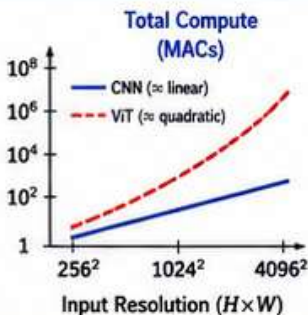
1 Input resolution increases

256×256



How cost grows with input resolution

Relative Cost (arbitrary scale)



CNN: cost $\approx O(H \times W)$ (per layer)

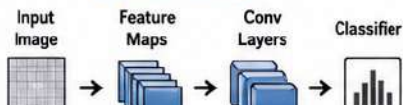
- - - ViT: cost $\approx O(N^2)$ (self-attention)

$N = \text{token count} \propto H \times W$

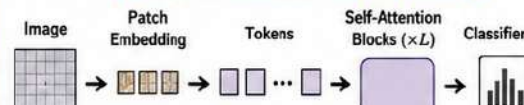
2

Impact on CNNs and Vision Transformers

A) CNN scaling



B) Vision Transformer (ViT) scaling

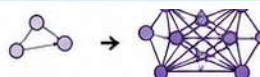


Data movement / memory traffic increases



Longer latency

Computation for attention grows quadratically



Poor scalability at very high resolution

CNN cost scales roughly with spatial size ($H \times W$) per layer ($\approx$ linear with number of pixels)

ViT token count grows with image size
Self-attention cost can grow quadratically with token count ($O(N^2)$)

3 What this means on edge devices



More computation (higher MAC count)



More energy consumption



More on-chip / off-chip memory pressure



Higher latency (end-to-end delay)



Bigger models or deeper hierarchies may be needed

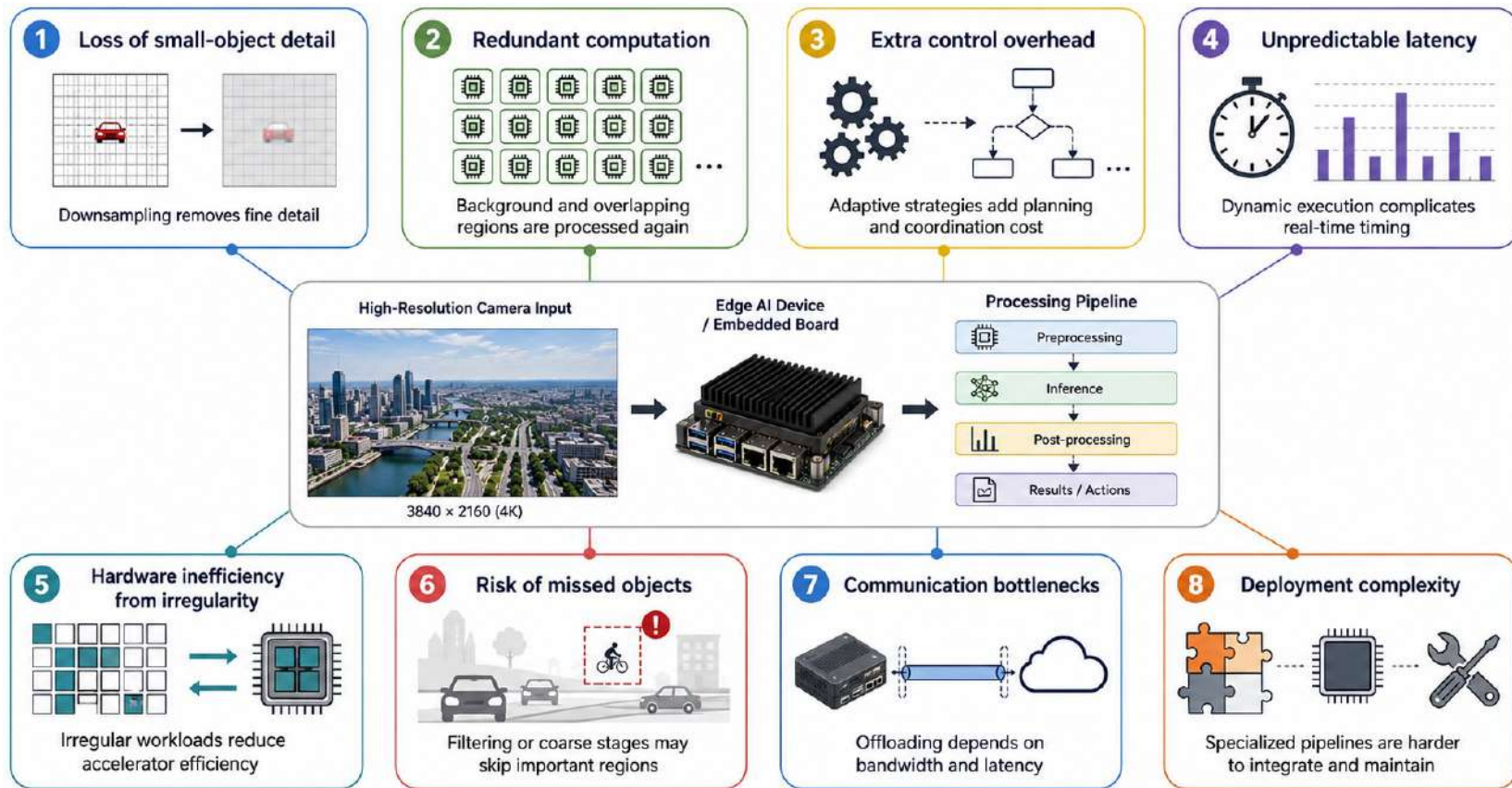


Naive full-image processing becomes inefficient



Motivation: process only the most informative image regions

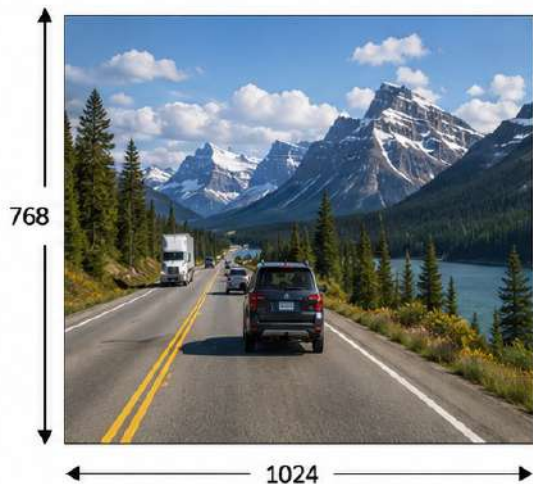
Challenges for Edge Vision & High-Res Images



Apply Engineering 101 - Partition the problem!

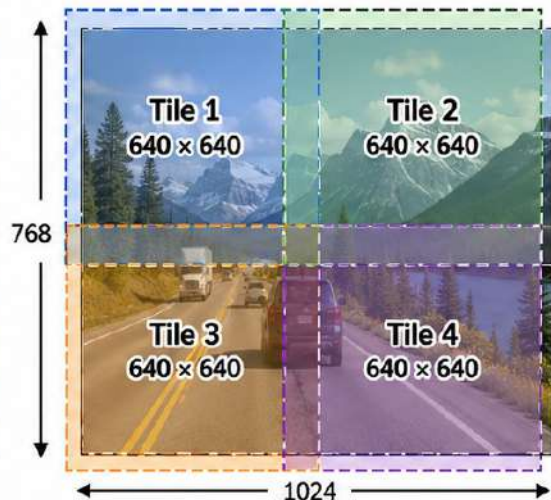
1 Original image

Original image: $1024 \times 768 \times 3$



2 2×2 Overlapping Tile Partition

Same image with overlapping 640×640 tiles



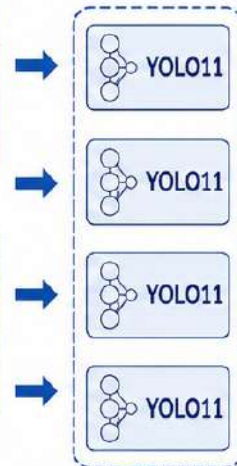
Overlapping tiles preserve coverage at image boundaries.

3 Tiles as YOLO11 Inputs

Extracted $640 \times 640 \times 3$ tiles



YOLO11 input
 $640 \times 640 \times 3$



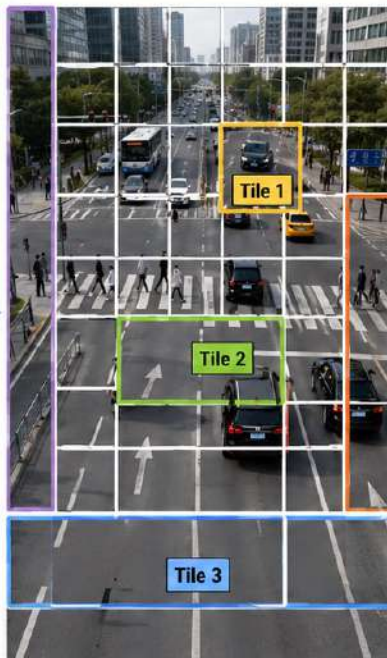
Each tile is processed independently.

Partitioning (or tiling)

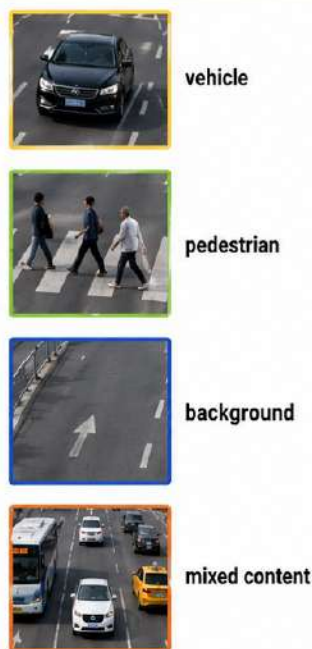
1. Input traffic frame



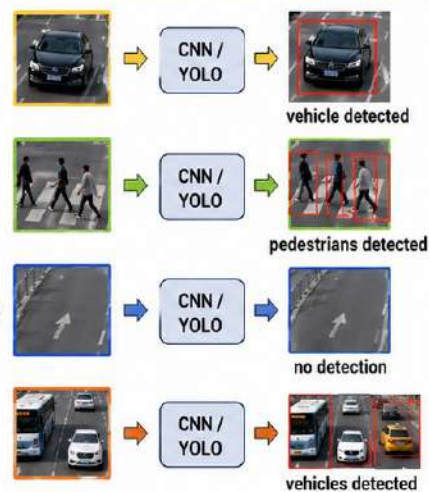
2. Partition into tiles



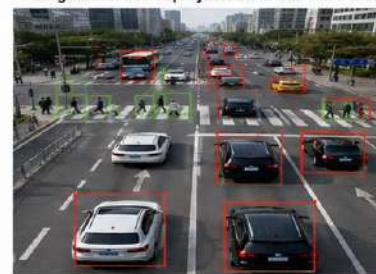
3. Tile extraction



4. Per-tile processing



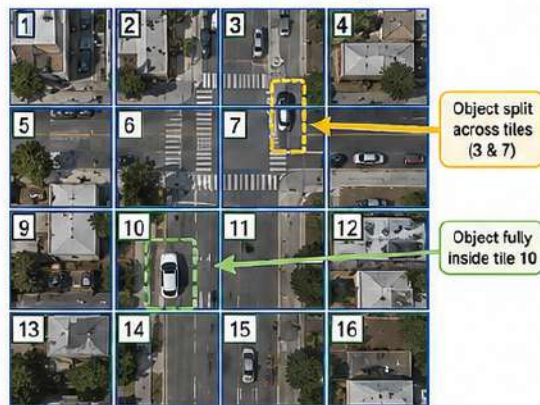
Merged detections projected back to full frame



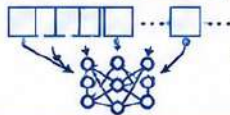
Tiling divides a large frame into smaller regions so each tile can be processed independently, reducing memory footprint and enabling selective processing on edge devices.

Tiling Issues – Tile Size, Tile Overlap

A) Zero overlap



tiles: medium
example: 16



Each tile
→ NN

Impact

- No redundant coverage
- Boundary splitting can hurt detection
- Moderate compute

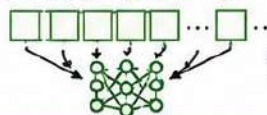
Detection result (example)



B) Small overlap



tiles: higher
example: 25



Each tile
→ NN

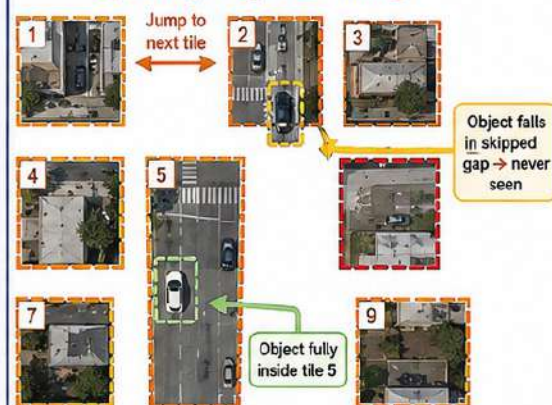
Impact

- Better object continuity
- Fewer border misses
- Higher compute / more redundancy

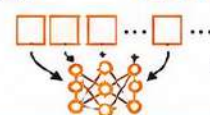
Detection result (example)



C) Jump / sparse tiling



tiles: lowest
example: 9



Each sampled
tile → NN

Impact

- Lowest compute
- Gaps in coverage
- Highest miss risk

Detection result (example)



Again – Do only what's necessary!



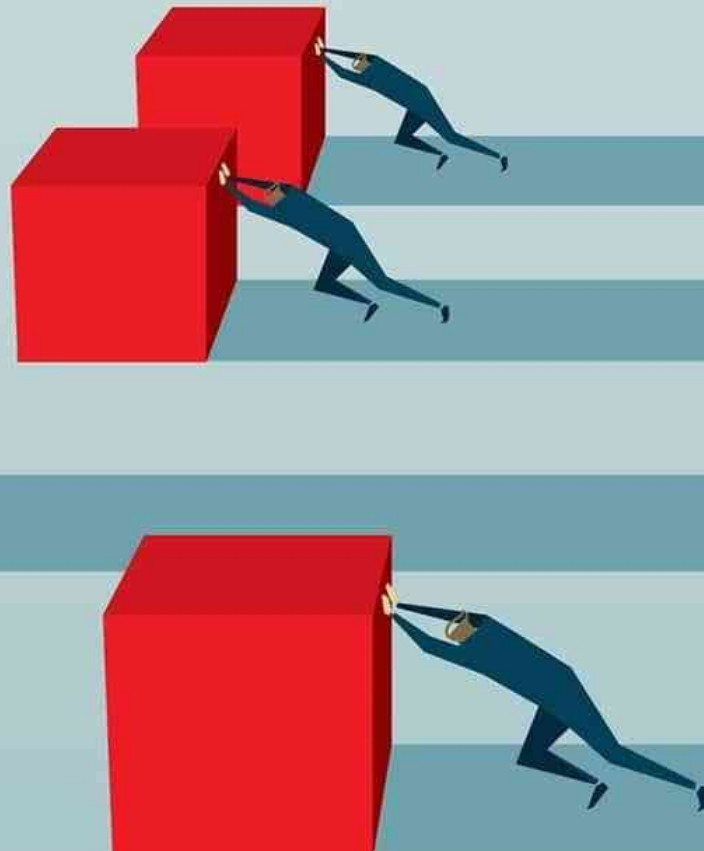
“Edge Intelligence: Time for the Edge to Grow Up!”

Selective Tile Processing

Spend time crafting circle

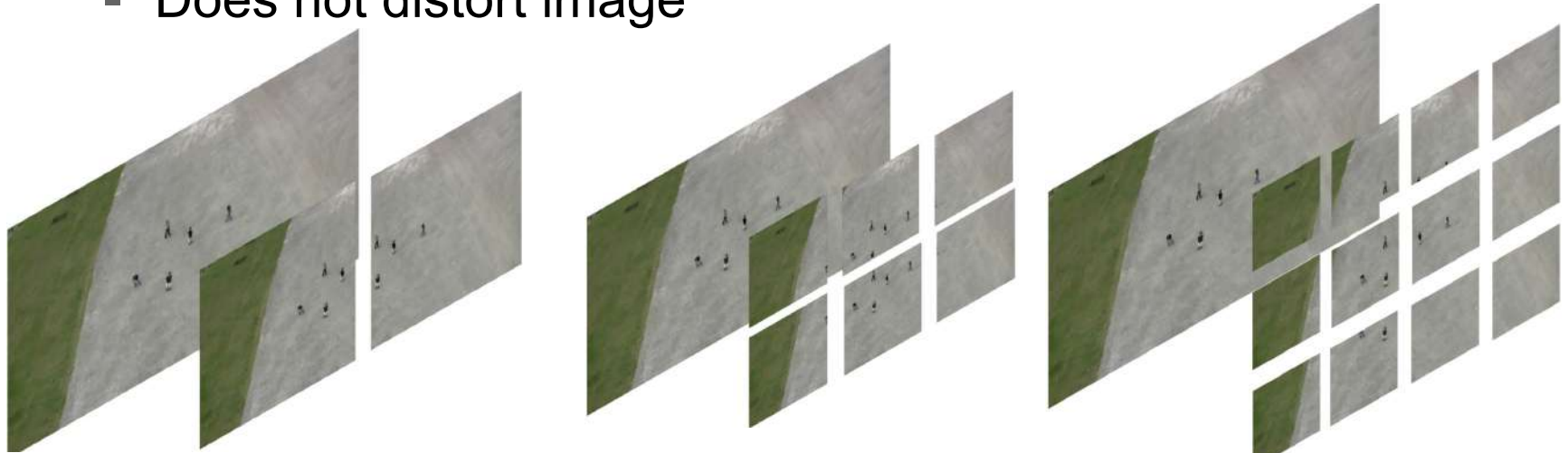


Enjoy the benefits!



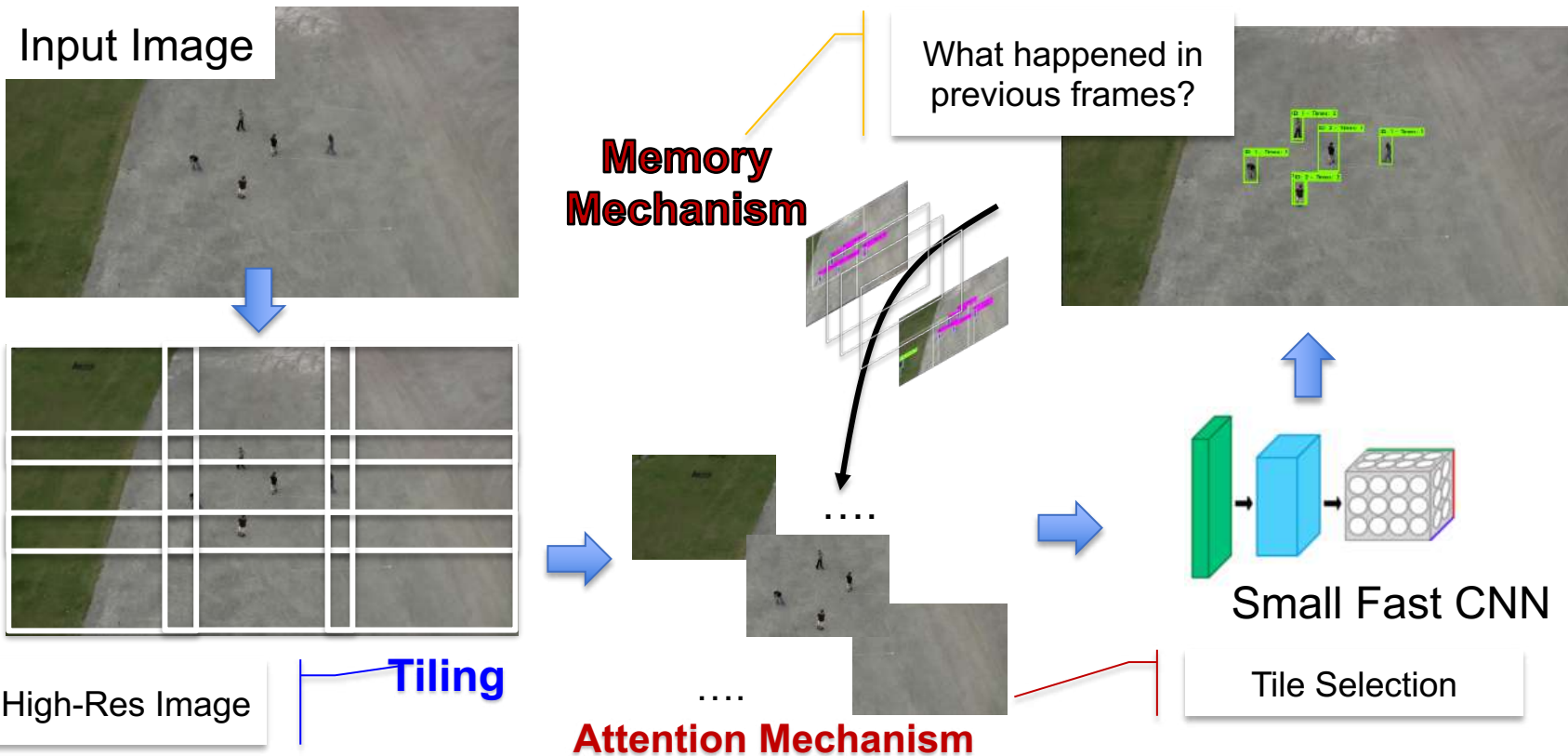
DroNet v1 – Efficient CNN Object Detector for UAV

- ❑ **Tiling**: Uniformly partition input image to achieve complete coverage of the image - maintain a constant overlap
 - Maintain object resolution
 - Does not distort image



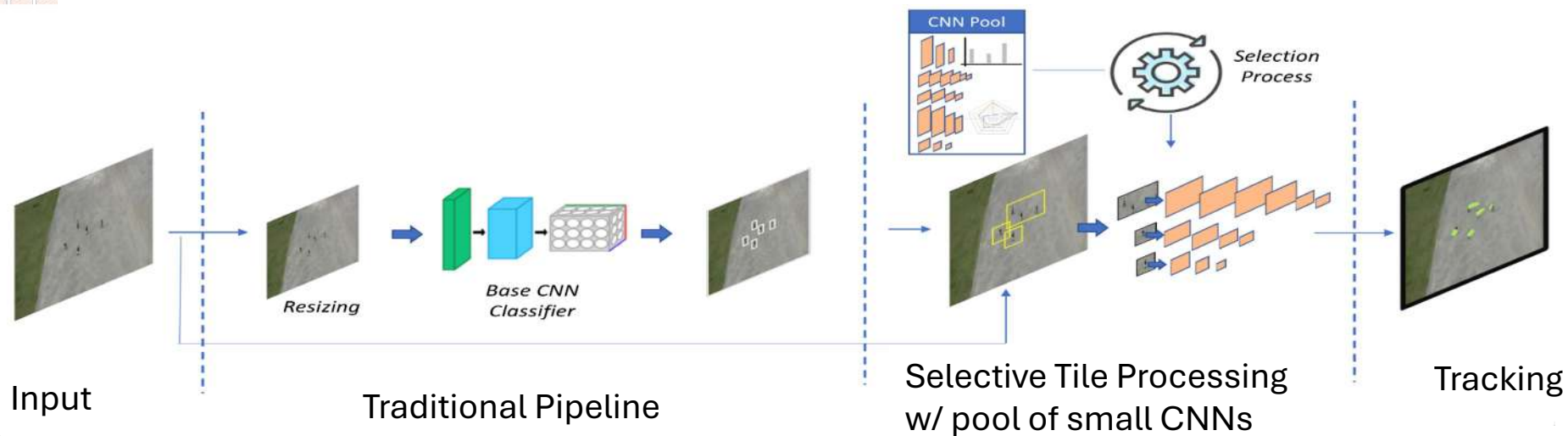
C. Kyrkou, G. Plastiras, T. Theocharides, S. I. Venieris and C. Bouganis, "DroNet: Efficient convolutional neural network detector for real-time UAV applications," *2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, Dresden, Germany, 2018, pp. 967-972, doi: 10.23919/DATE.2018.8342149.

Improvement w/ intro of human-like response?



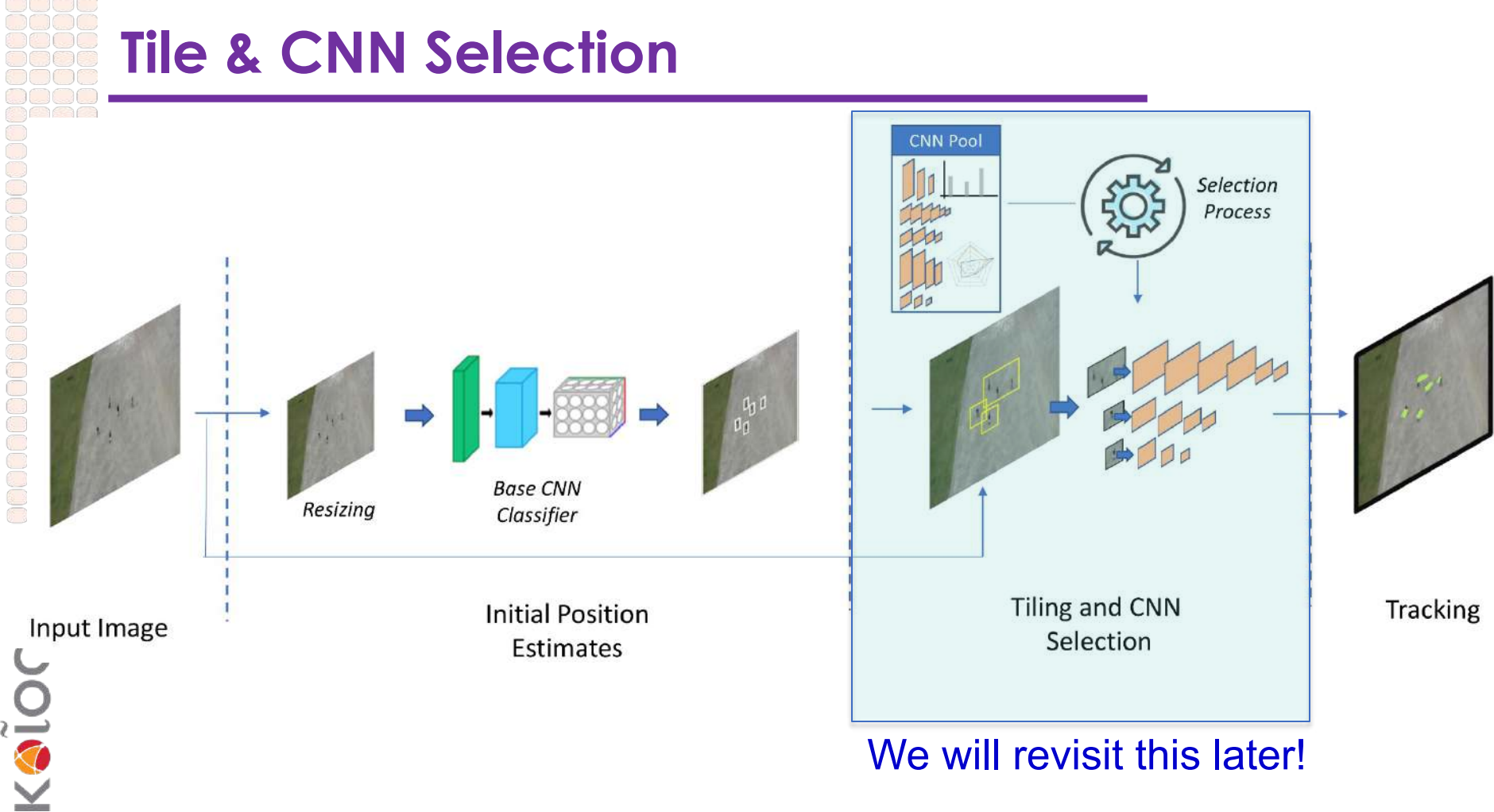
¹George Plastiras, Christos Kyrkou, Theocharis Theocharides, "Efficient ConvNet-based Object Detection for Unmanned Aerial Vehicles by Selective Tile Processing", In International Conference on Distributed Smart Cameras (ICDSC), September 2018.

Dronet → EdgeNet – Putting it all together!

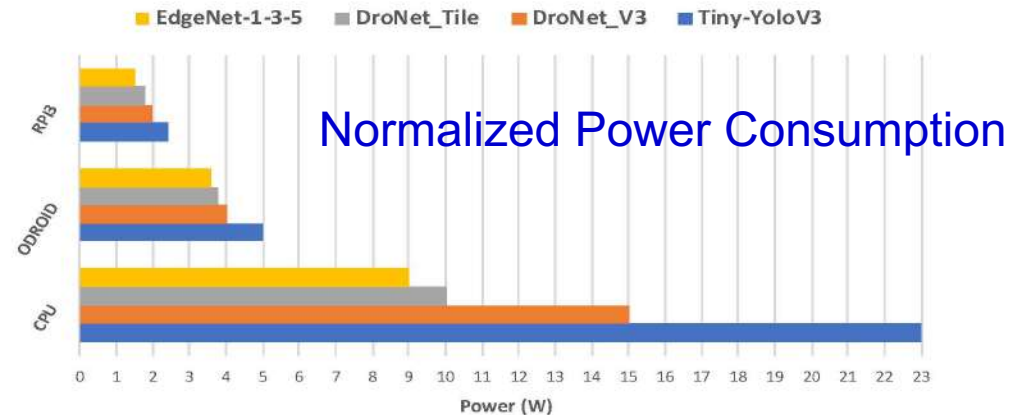
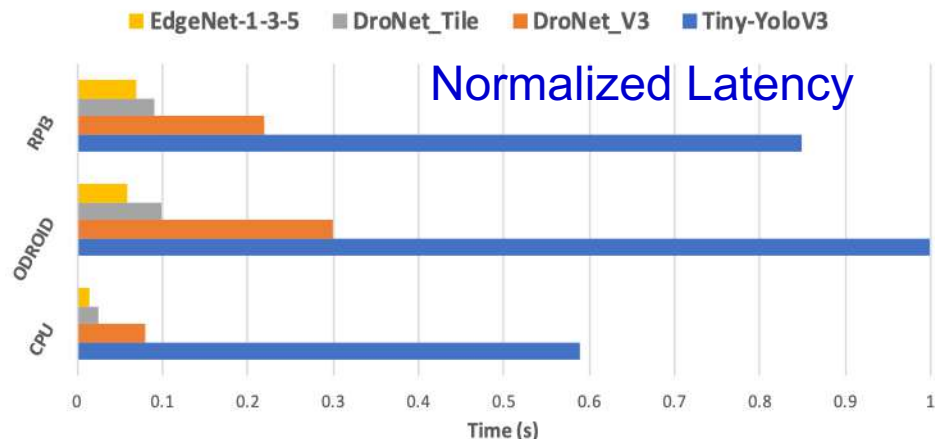
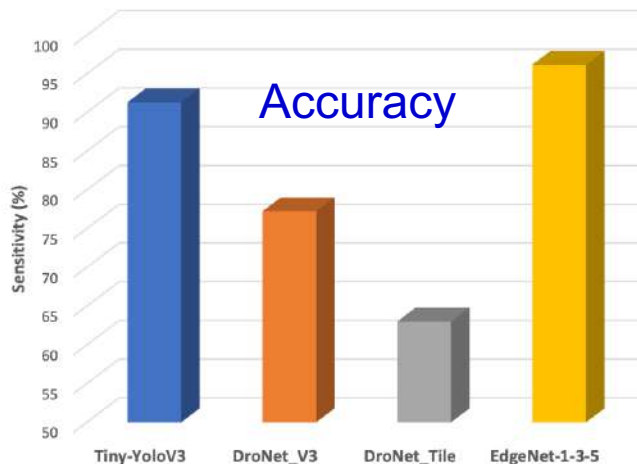


G. Plastiras, C. Kyrkou, and T. Theocharides, "EdgeNet: Balancing accuracy and performance for edge-based convolutional neural network object detectors," in *Proc. 13th Int. Conf. Distributed Smart Cameras (ICDSC 2019)*, Trento, Italy, 2019, Art. no. 8, pp. 1–6, doi: 10.1145/3349801.3349809.

Tile & CNN Selection



Accuracy, Performance, Energy



Experiments run on:

- Lightweight laptop
- Raspberry Pi 3
- Odroid (Samsung Exynos)

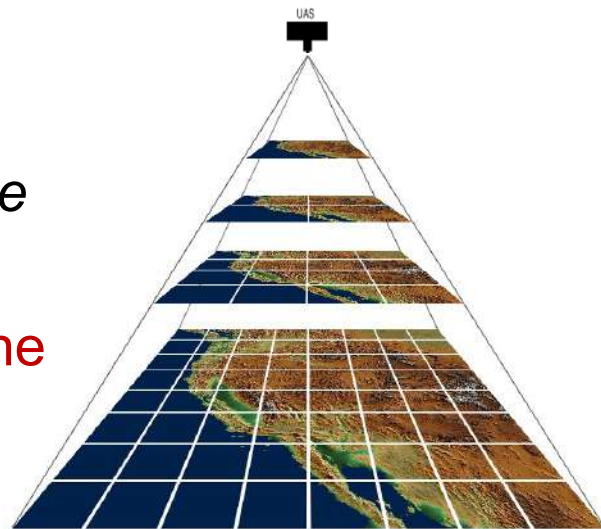
EdgeNet in Action (car counting - high altitude)



"Edge Intelligence: Time for the Edge to Grow Up!"

Impact of Tiling, Memory & Attention

- ☐ Improved Accuracy
- ☐ Balanced Performance
- ☐ *Model Agnostic*
- ☐ *Further parameter tuning and **selection criteria** to achieve even higher performance*
- ☐ **Tiles are static** → Dynamically adjusting the number, the size, and positioning based on recorded activity
- ☐ **A lot of room for improvement**

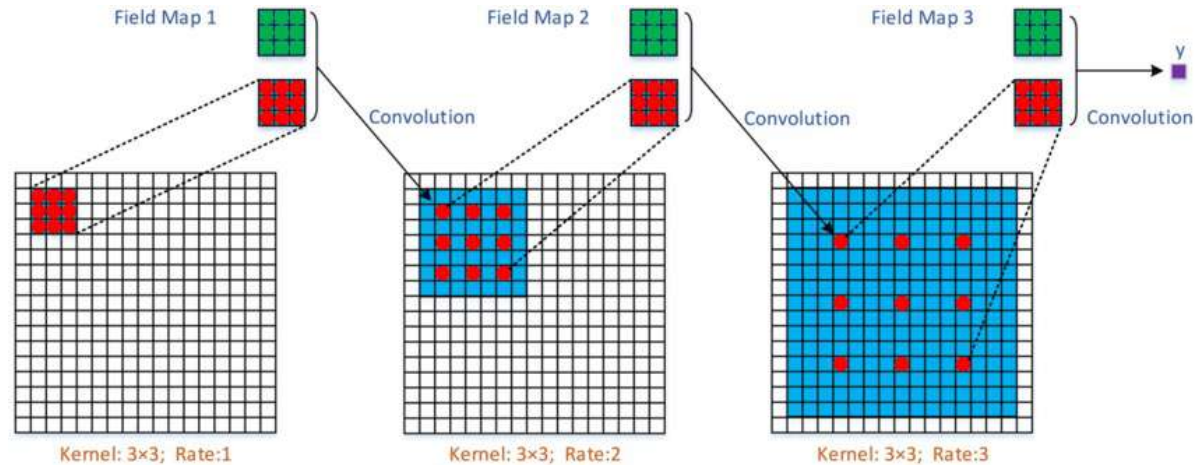
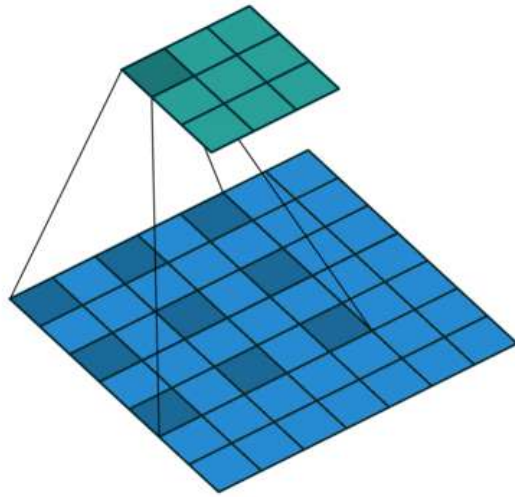


What about other selection methods?



Alternatively – don't tile the image, tile the Conv!

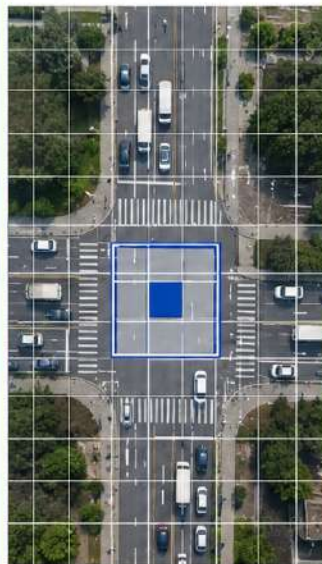
Atrous (Dilated Convolution)



C. Kyrkou and T. Theodoridis, "EmergencyNet: Efficient Aerial Image Classification for Drone-Based Emergency Monitoring Using Atrous Convolutional Feature Fusion," in *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 13, pp. 1687-1699, 2020

Atrus Convolution and its Impact

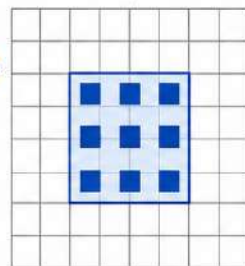
1 Input feature map



Example: UAV view of a busy intersection (grid overlay represents feature map locations)

2 Standard convolution (dilation = 1)

3×3 kernel

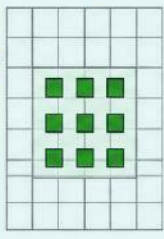
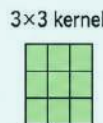


Dense sampling

Effective field: 3×3

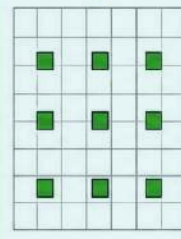
3 Atrous convolution

(1) dilation rate = 1



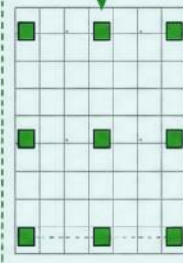
Effective field: 3×3

(2) dilation rate = 2



Effective field: 5×5

(3) dilation rate = 3

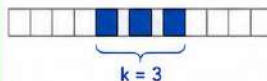


Effective field: 7×7

Same number of weights, larger context

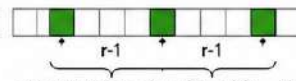
1D intuition

Standard convolution (r = 1)
samples contiguous elements



k = 3

Atrous convolution (rate = r)
samples every r-th element



effective size = $k + (k - 1)(r - 1)$

4 Key idea / output effect



Larger receptive field



Preserves resolution



Same parameter count



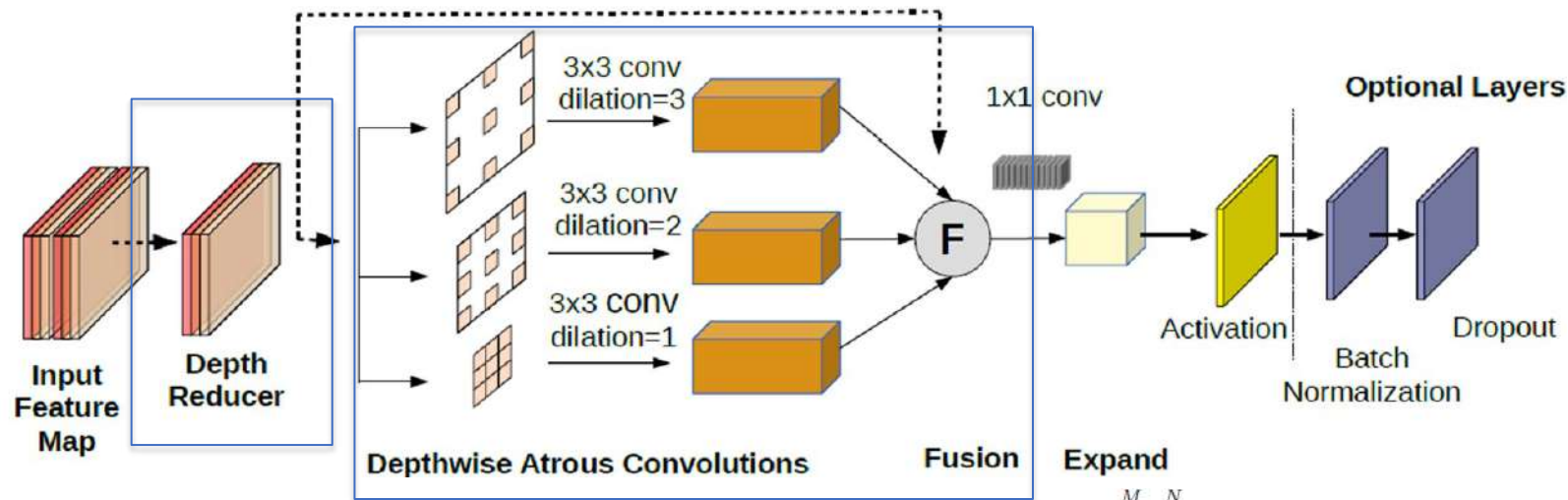
Useful for dense prediction / segmentation

effective kernel size

$$= k + (k - 1)(r - 1)$$

for $k = 3, 5, 7, \dots$

Atrous Convolutional Feature Fusion



Expand

$$U_d = \sum_{i=1}^M \sum_{j=1}^N X(m + d \times i, n + d \times j) \odot w_d(i, j) \quad (1)$$

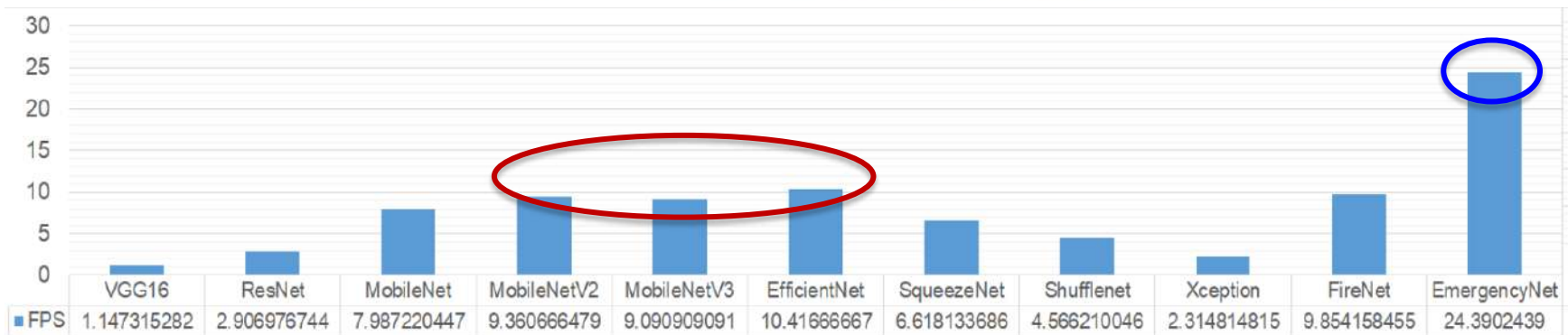
$$Z = F_{\substack{1 \leq m \leq M \\ 1 \leq n \leq N \\ 1 \leq d \leq N_d}}(\{U_d(m, n)\})$$

$$d \in [1, \dots, N_d]. \quad (2)$$

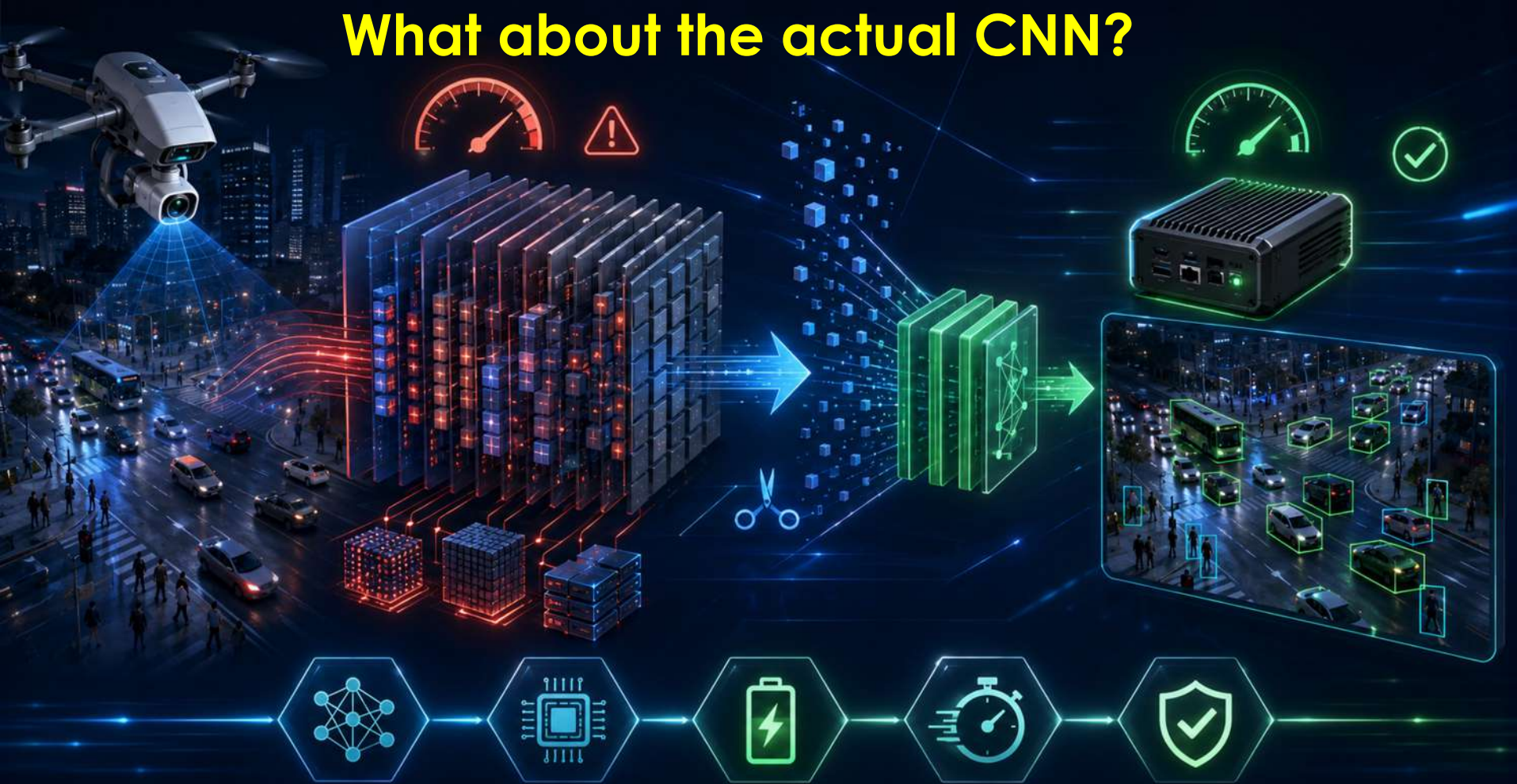
C. Kyrkou and T. Theocharides, "EmergencyNet: Efficient Aerial Image Classification for Drone-Based Emergency Monitoring Using Atrous Convolutional Feature Fusion," in IEEE JSTARS, vol. 13, pp. 1687-1699, 2020

C. Kyrkou, T. Theocharides "Deep-Learning-Based Aerial Image Classification for Emergency Response Applications using Unmanned Aerial Vehicles", CVPR 3d International Workshop in Computer Vision for UAVs, Long Beach, CA, 16-20 June, 2019, pp. 517-525.

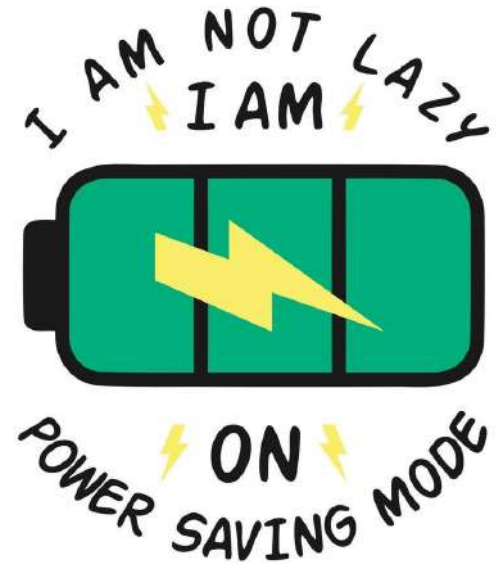
Performance Evaluation



What about the actual CNN?



Doing something, does not mean doing it efficiently!



"I choose a lazy person to do a hard job. Because a lazy person will find an easy way to do it"
– (attributed to Bill Gates, although its origin maybe much longer in time...)

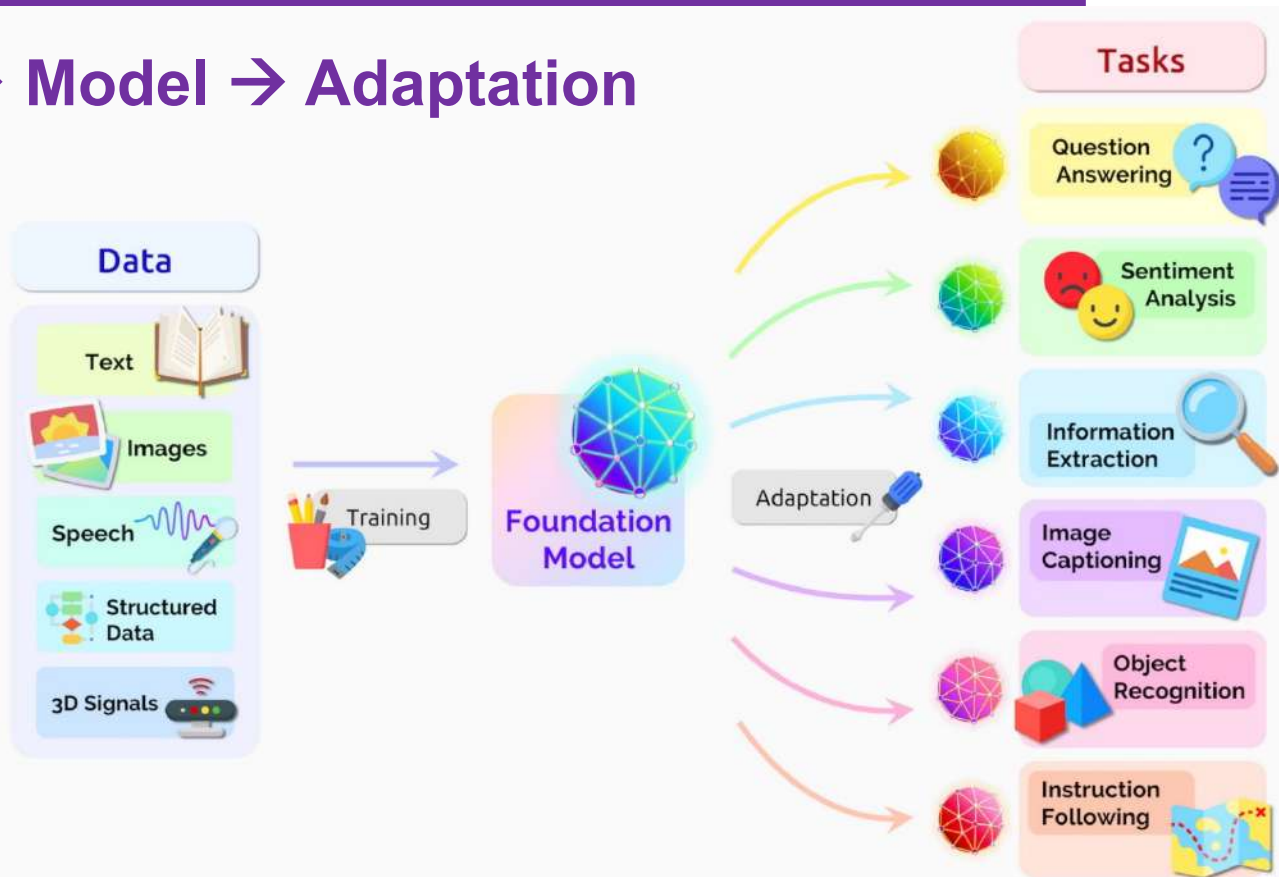
Part 4: The Adulthood Era



Becoming Efficient (and having a drink here and there!)

What do we typically do today...

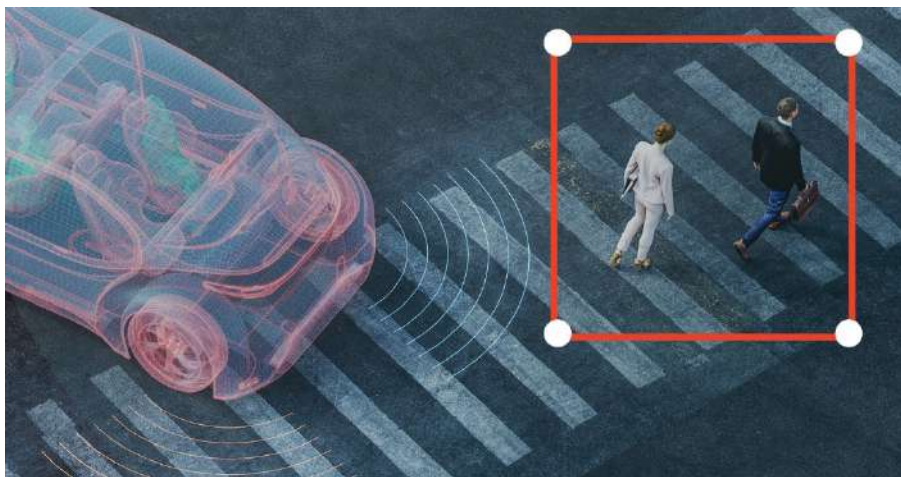
Data → Model → Adaptation



Well, we are doing a lot!

... in fact, way more than what is necessary!

- ❑ Aiming to design a model which performs good across all data!
- ❑ Benchmarking – ImageNet, CIFAR100, MNIST, etc.
- ❑ This is TOO MUCH!



I need the car to detect pedestrians.
I don't care if it's a man or a woman!



How do we (humans) do it?

❑ Multi-modal sensing

- I hear a sound, I turn and focus there!

❑ Saliency (visual focus)

- cognitive ability to quickly differentiate from background

❑ Short and Long Term Spatiotemporal Memory

- Focus on current context
- Quick context switch

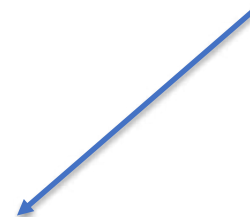
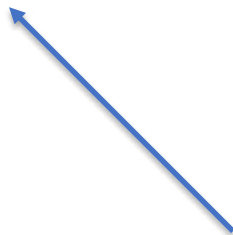
❑ Multi-Task

- Most tasks done “mechanically”
– i.e. set-it-and-forget-it.
- Event-Driven Thinking!

❑ Occasional Refresh



Act on demand (and on high stakes!)



Do only what and when you have to!

❑ Data-Specific Training

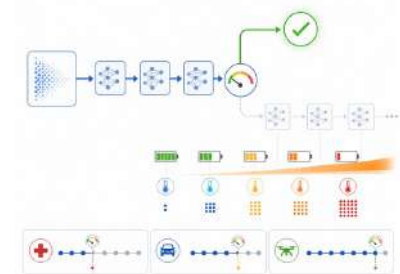
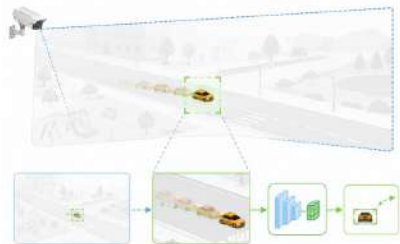
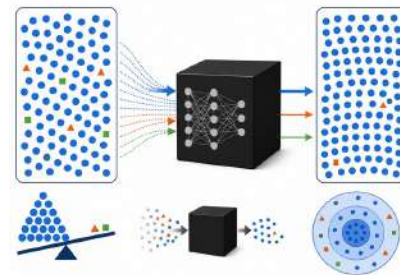
- Model building relies on a black-box approach
- Datasets over-represent!
- Mathematical models will regress to whatever datapoints given

❑ Do **only** what's necessary

- Focus only where there's a high probability of something to happen
- Use short-term memory (remember the principle of spatiotemporal locality from Caches?)

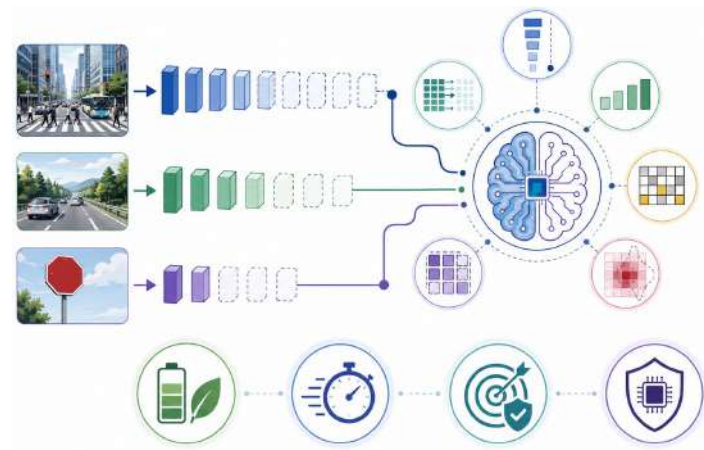
❑ **Stop** when you have reached a relatively safe conclusion!

- More work → More Power → More Computation
- Application-Dependent Criteria!

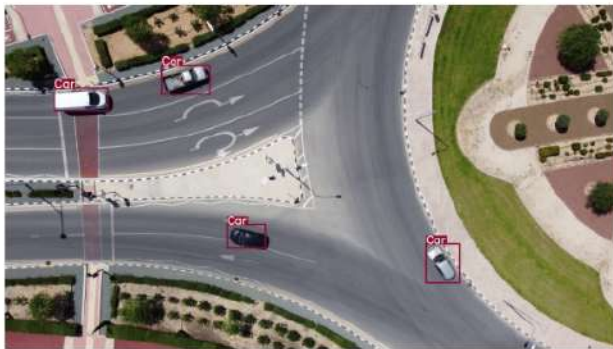


How? → Dynamic Deep Neural Networks!

- ❑ Improved Efficiency:
Computation and energy reduction by *adapting the model's depth or parameters to each input.*
- ❑ Faster Inference:
Early exit strategies for example allow *quick predictions* for easy inputs, lowering latency.
- ❑ Resource Awareness:
Dynamic models can *adjust operations* to handle resource constraints
- ❑ Maintained Accuracy:
Often preserve accuracy by allocating full model capacity *only when needed.*



Case-Study: UAV-based Vehicle Detection



(a) 50m



(b) 200m

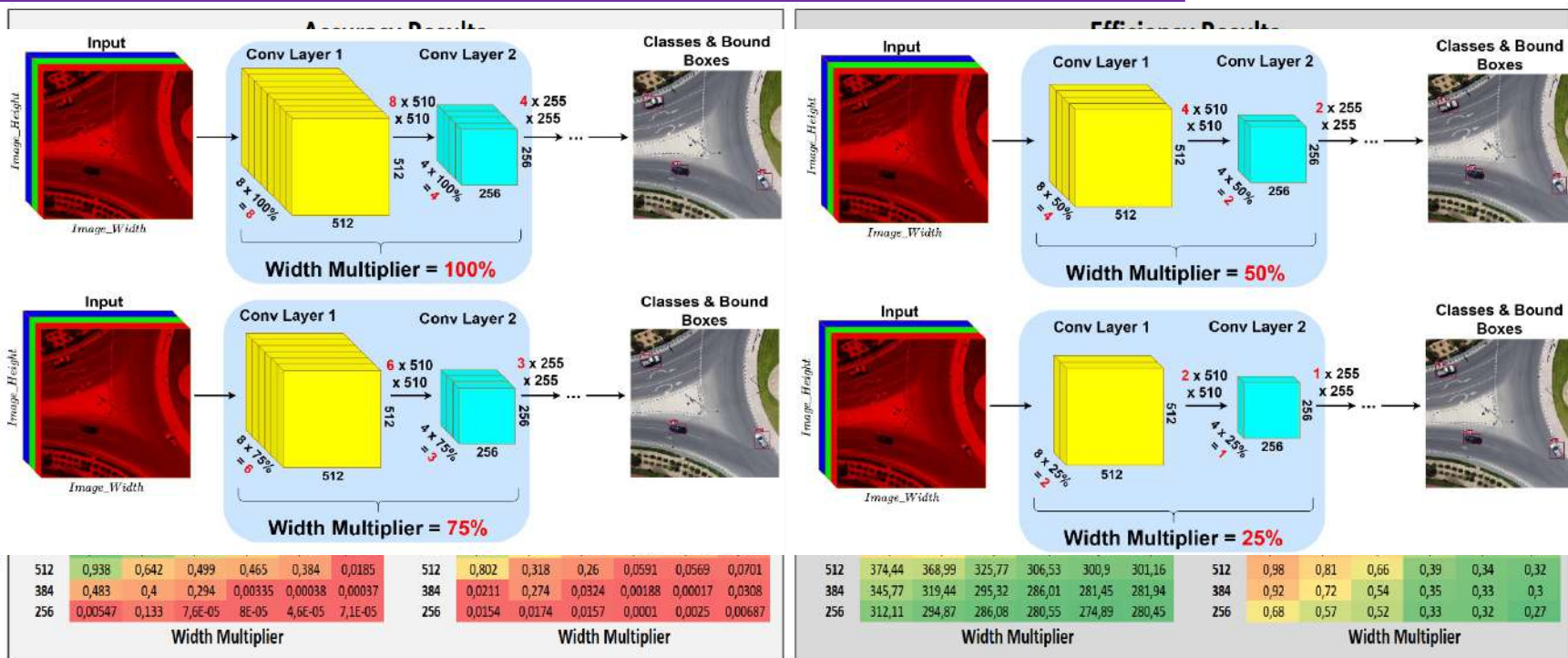


(c) 350m



(d) 400m

Experimenting with various Deep CNNs



Channels	2	4	8	16	32	64	128
Power (mW)	486	487	487	491	494	529	580
LUTs	151	238	271	888	1506	10014	16668

Altitude-Aware Observations



(a) 50m



(b) 200m



(c) 350m



(d) 400m

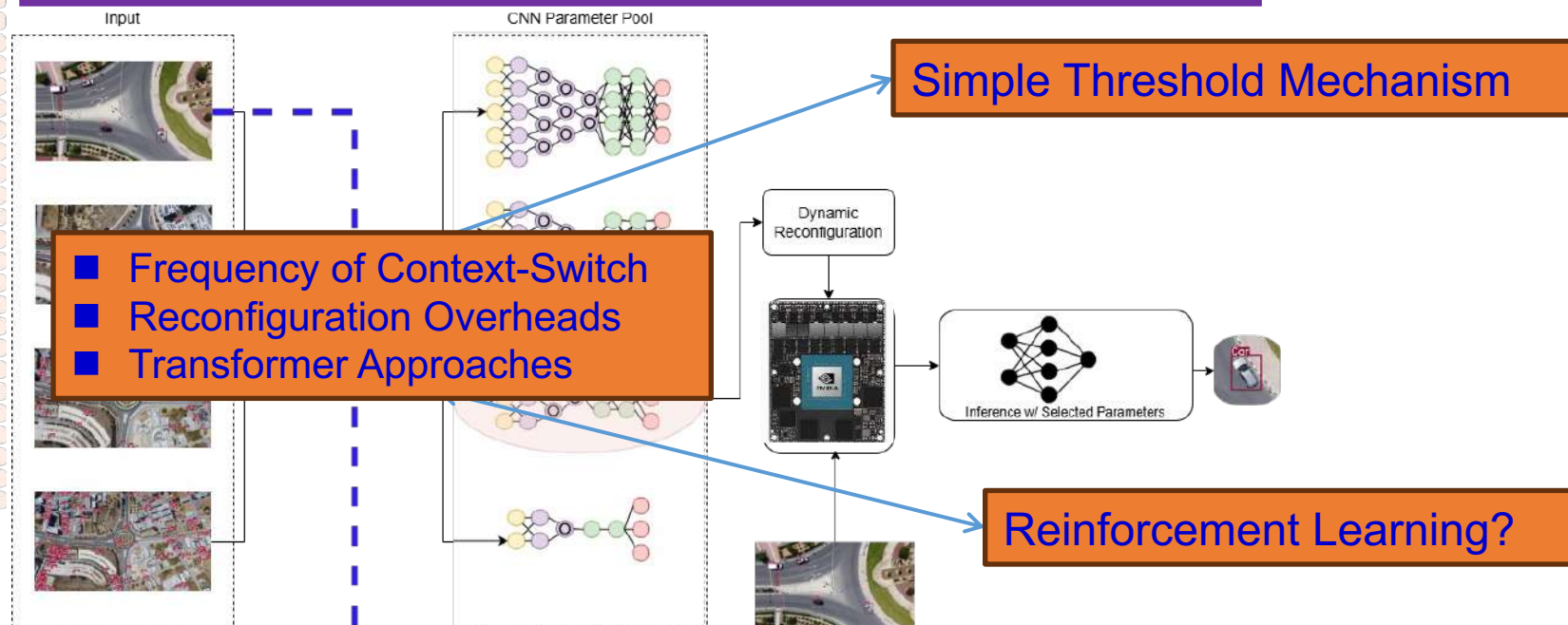
		MAC Operations (M)					
		1	0.75	0.5	0.25	0.1	0.01
Resolution	1536	89,03	66,9	44,77	22,65	11,38	4,67
	1408	75,92	57,05	38,18	19,31	9,71	3,98
	1280	61,19	45,98	30,77	15,57	7,83	3,21
	1088	45,42	34,13	22,84	11,55	5,81	2,38
	896	31,99	24,04	16,09	8,14	4,09	1,68
	768	22,71	17,07	11,42	5,78	2,9	1,19
	640	16,35	12,29	8,22	4,16	2,09	0,86
	512	11,03	8,29	5,55	2,81	1,41	0,58
	384	6,75	5,07	3,39	1,72	0,86	0,35
	256	2,92	2,19	1,47	0,74	0,37	0,15

		Size (MB)					
		1	0.75	0.5	0.25	0.1	0.01
Resolution	1536	22,55	17,29	13,53	11,27	10,69	10,54
	1408	21,01	15,75	11,99	9,73	9,15	9
	1280	19,27	14,01	10,25	7,99	7,41	7,26
	1088	17,41	12,15	8,39	6,13	5,55	5,4
	896	15,83	10,57	6,81	4,55	3,97	3,82
	768	14,73	9,47	5,71	3,45	2,87	2,72
	640	13,98	8,72	4,96	2,7	2,12	1,97
	512	13,35	8,09	4,33	2,07	1,49	1,34
	384	12,85	7,59	3,83	1,57	0,99	0,84
	256	12,4	7,14	3,38	1,12	0,54	0,39

Width Multiplier

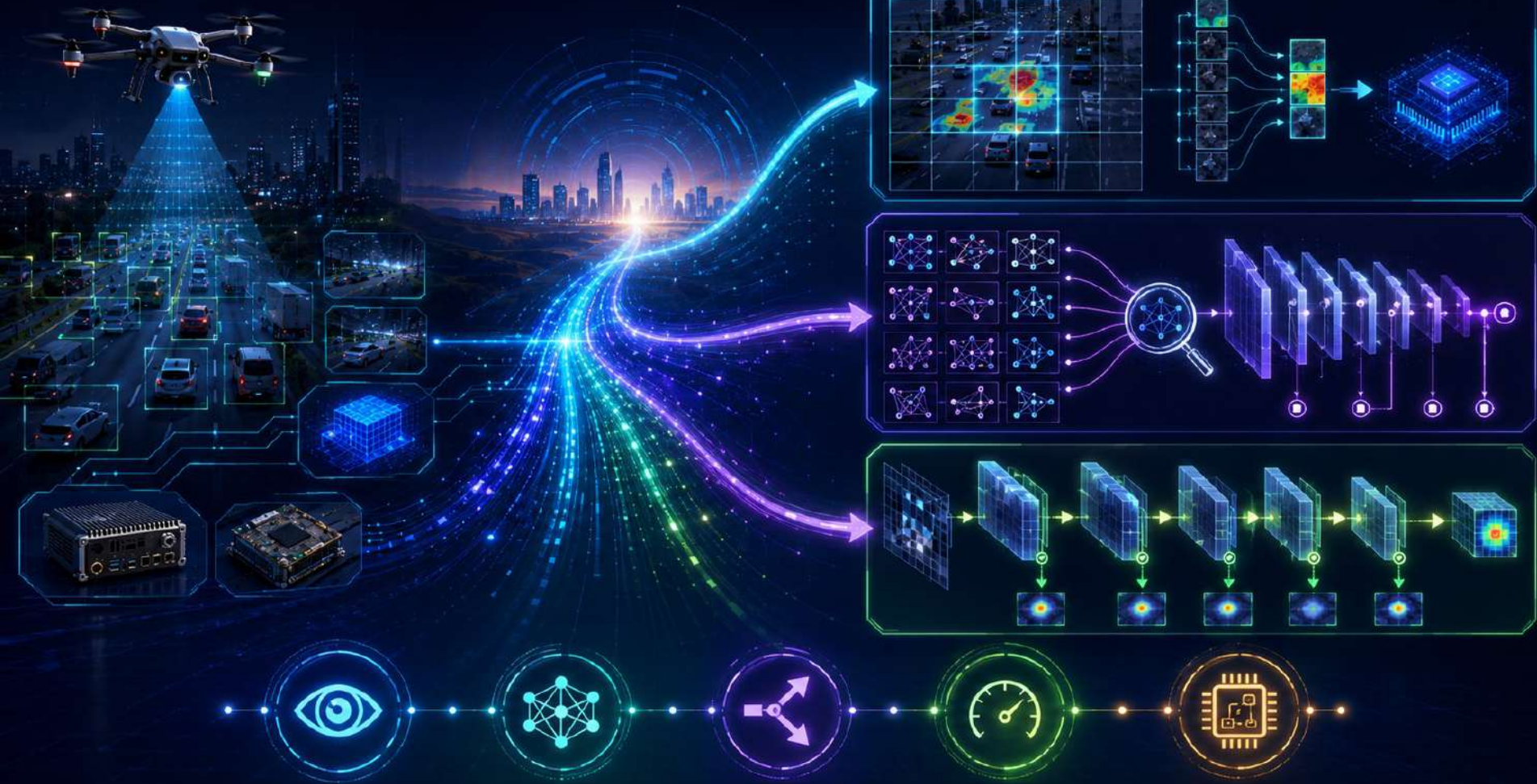
M. Piponidis and T. Theocharides, "Dynamic CNN Parameter Exploration for Multi-Altitude UAV Object Detection," 2025 11th International Conference on Control, Automation and Robotics (ICCAR), Kyoto, Japan, 2025, pp. 510-515, doi: 10.1109/ICCAR64901.2025.11072991.

Context-Aware Dynamic Convolution Selector

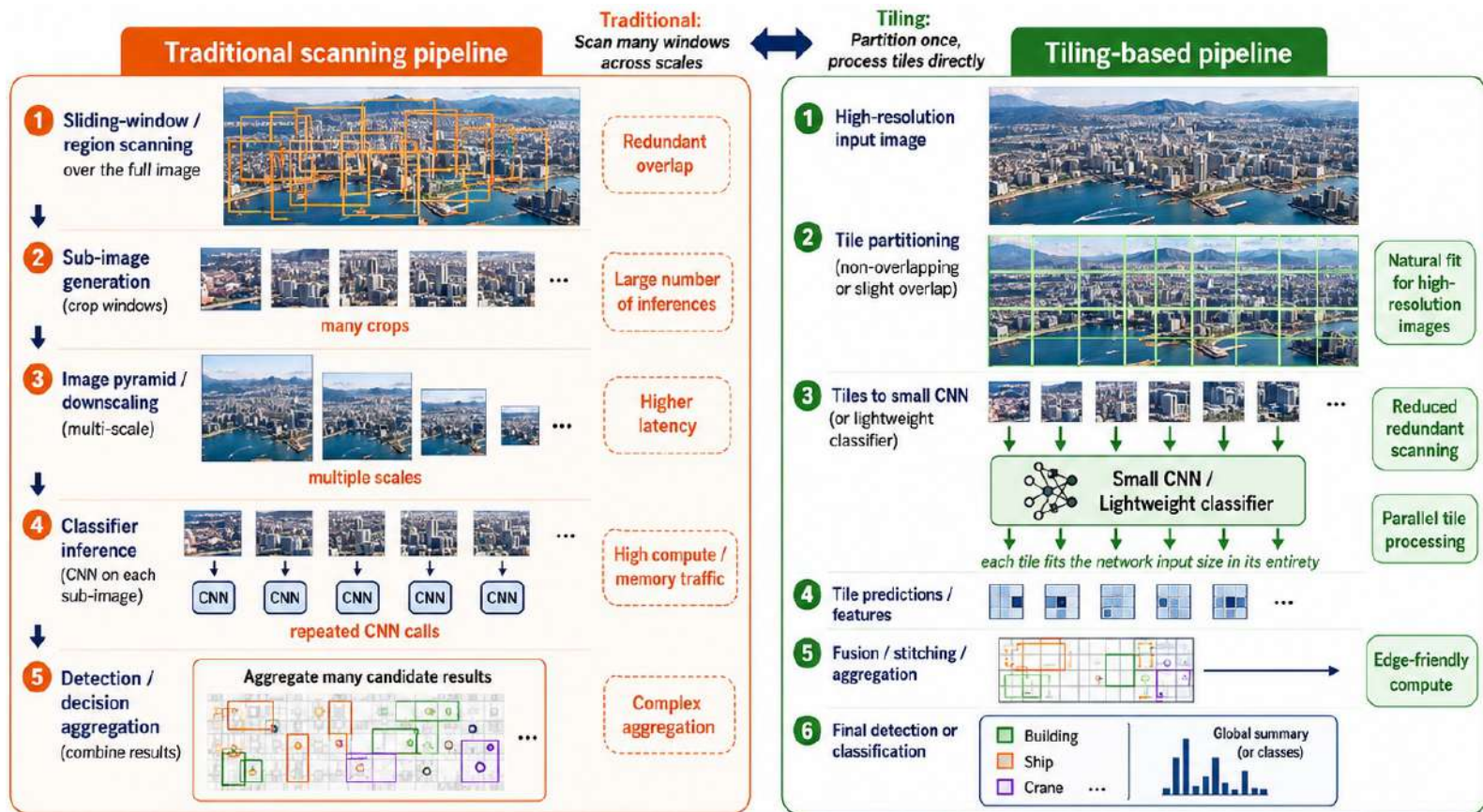


Model Configuration	Accuracy (mAP@.5)	CPU Usage (%)	Memory Usage (MB)	Power Usage (W)	Latency (ms)
Static	91.9	64.23	507.99	1.02	976.1
High Altitudes	92.6	52.68	429.36	0.75	805.6
Low Altitudes	93.5	43.51	359.08	0.56	373.23

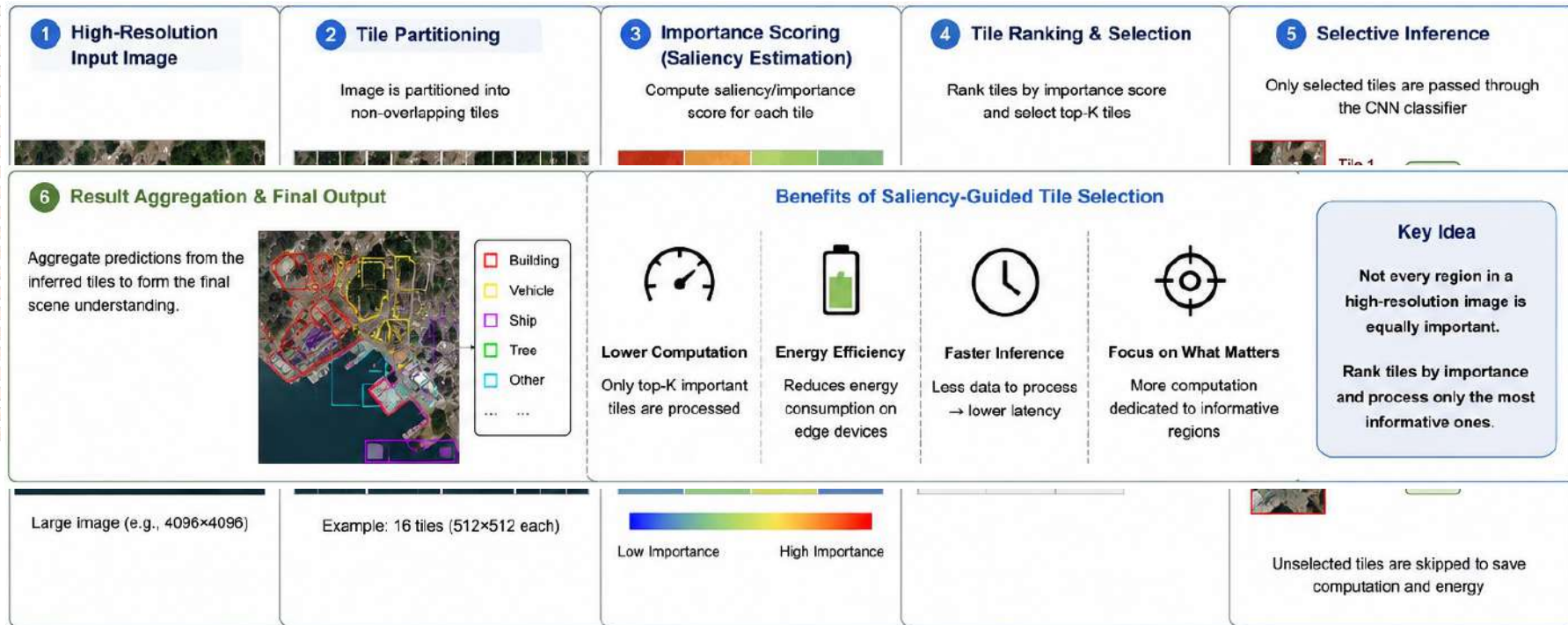
So...what else can we do?



Selective Region/Tile Processing (Recall)

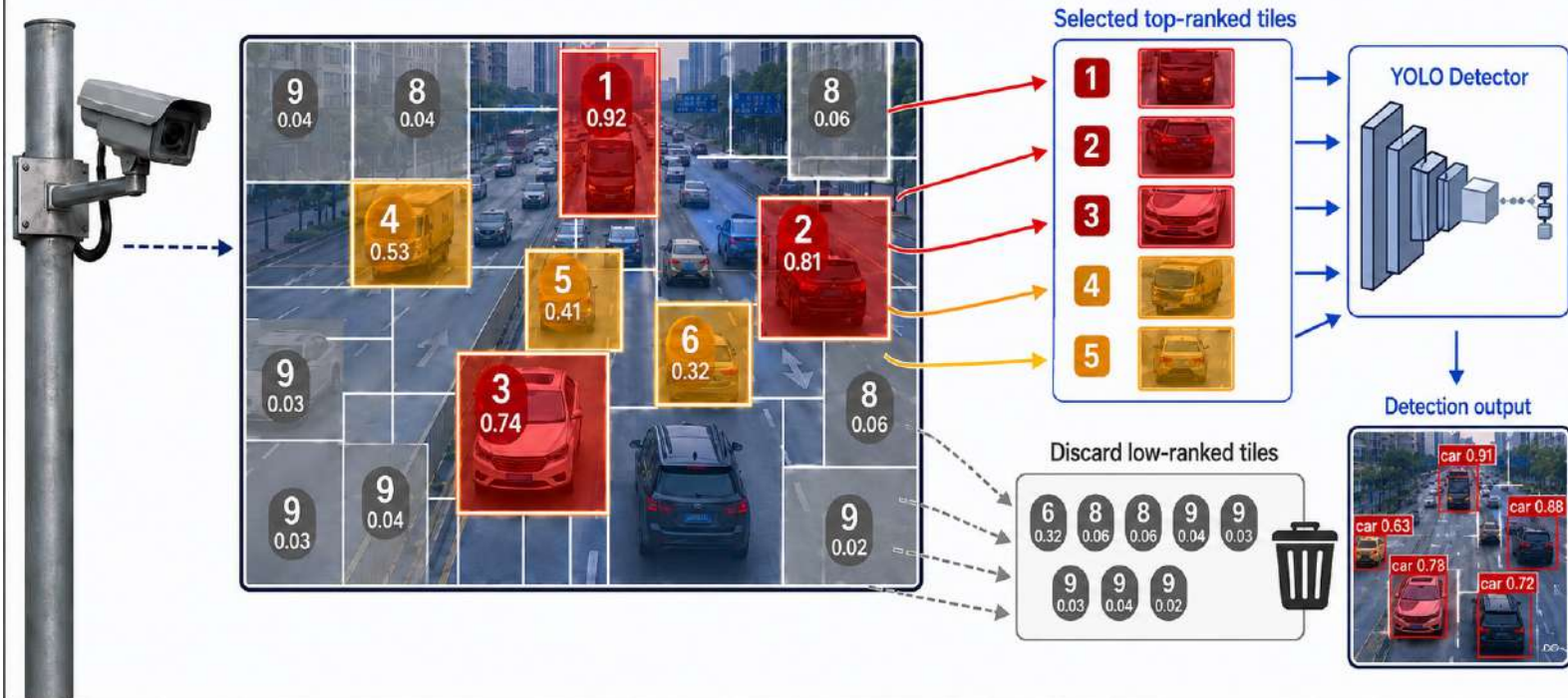


Saliency (Bioinspired focus!)



Example - Setup

- 1 Capture frame
- 2 Compute saliency-based ranking
- 3 Establish ranking
- 4 Route selected vs discarded tiles
- 5 Run YOLO on selected tiles



Saliency across CPU and Embedded GPU

Input frames
(**t+1**, **t**)



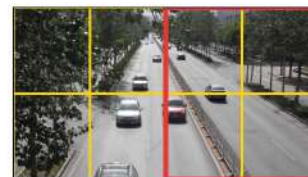
Saliency mask



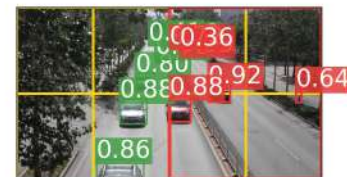
Tile scores (2x4)



Supertile (2x2)



AG detections
(**cached** + **supertile**)



CPU (Laptop)

1. Frame Differencing



40.0 ms (adopted)

2. Hybrid



147.4 ms

3. FD + k-means



245.1 ms

4. Temporal Phase



1.1 s

5. Optical Flow



1.9 s

eGPU - Jetson

1. DEVA



31.5 ms

2. U²-Net



116.9 ms

3. PoolNet



1.1 s

4. YOLO-Seg + Flow



1.3 s

5. InSPyReNet

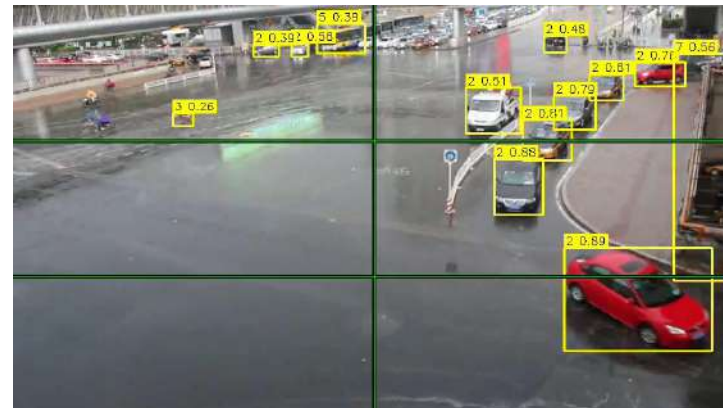
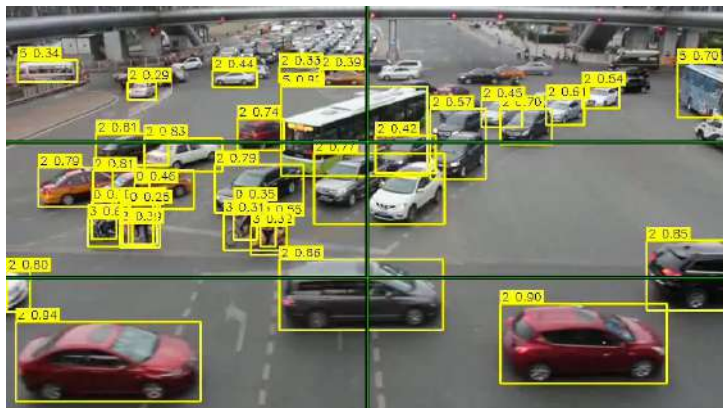
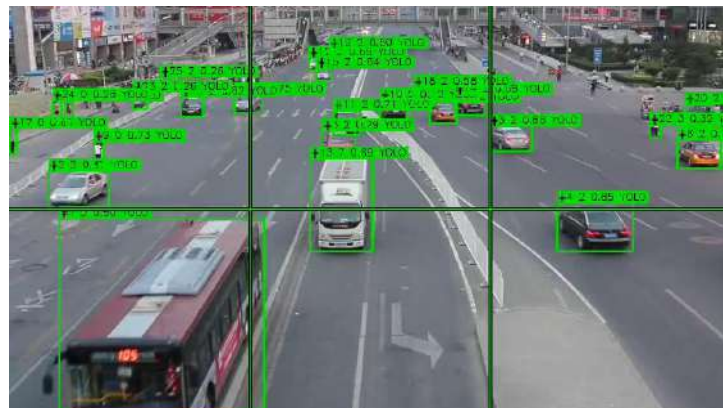


16.0 s

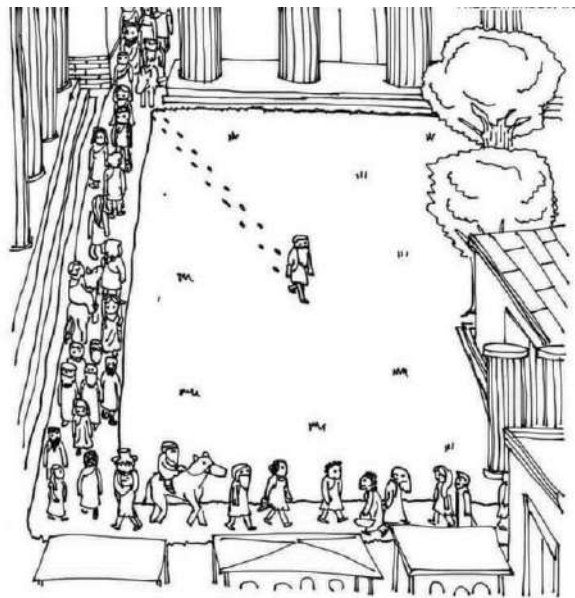
Results w/ Jetson Nano

Benchmark	Camera	Model	FPS	mAP	Power (W)	Utilization %	Unique Objects Detected
MOT17-02	Static	Baseline	4.2	0.613	15.41	49.7	52/56
		SIGHT	8.4	0.566	9.92	33.3	50/56
MOT17-04	Static	Baseline	2.4	0.748	11.34	63.6	81/81
		SIGHT	6.4	0.720	9.92	33.3	81/81
MOT17-05	Moving	Baseline	16.5	0.850	11.12	51.2	120/124
		SIGHT	18.6	0.661	8.03	35.9	114/124
MOT17-09	Static	Baseline	2.4	0.925	11.40	69.3	26/26
		SIGHT	6.6	0.862	10.15	36.3	26/26
MOT17-10	Moving	Baseline	2.2	0.709	10.98	60.7	57/57
		SIGHT	6.9	0.658	9.31	26.1	57/57
MOT17-11	Moving	Baseline	2.1	0.741	10.60	52.1	58/73
		SIGHT	4.6	0.654	11.55	46.0	59/73
MOT17-13	Moving	Baseline	2.2	0.598	10.62	43.4	104/108
		SIGHT	6.7	0.540	9.31	26.1	103/108

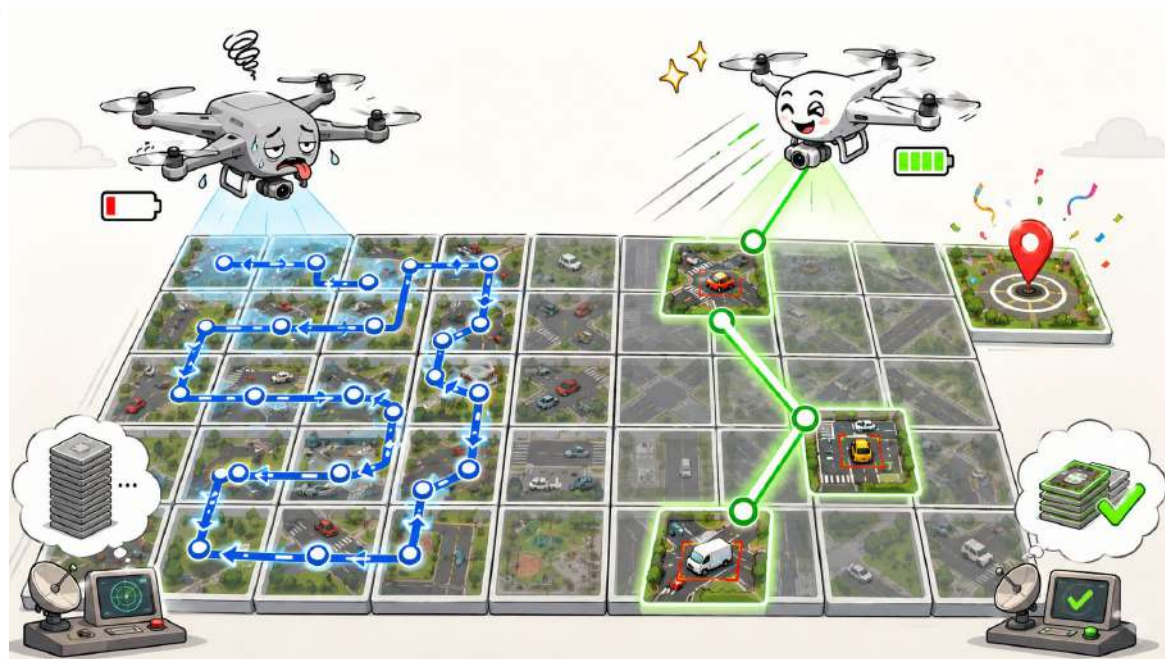
Jetson Nano Example



The Student's Favorite: Path of Least Resistance!

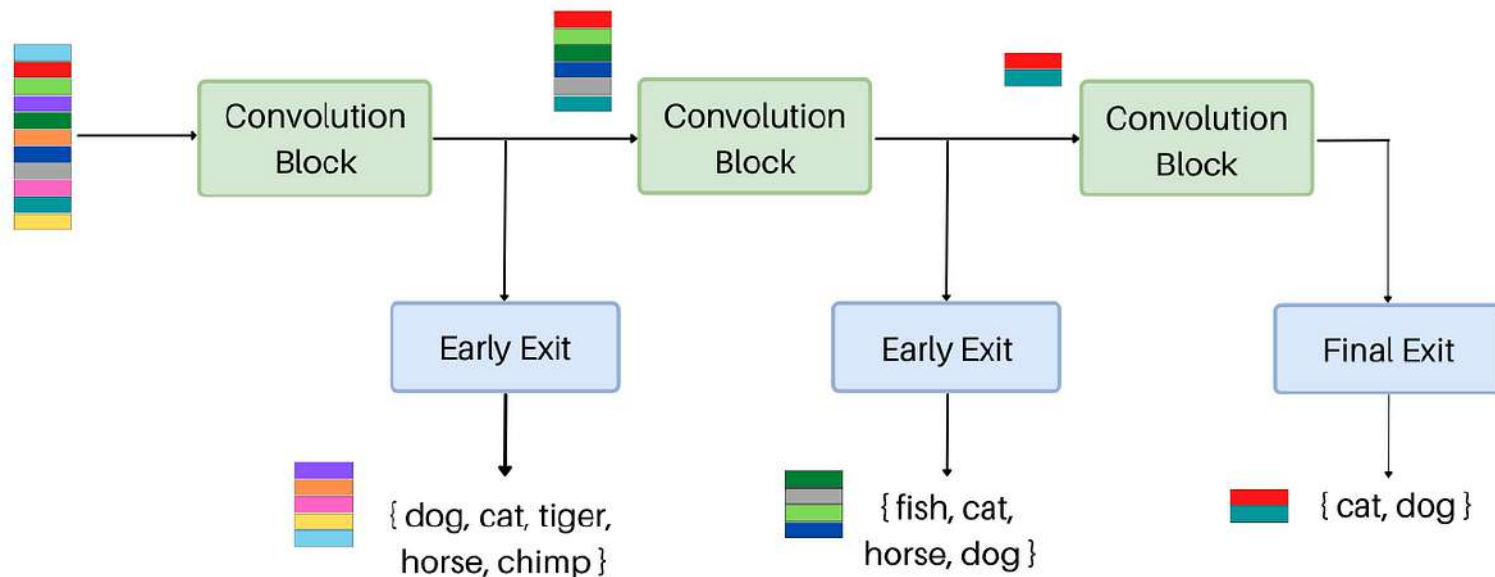


"UGH, THERE GOES THAT PYTHAGORAS GUY"



(Trust me, it remains like that in adulthood!)

Early Exit Deep Neural Networks



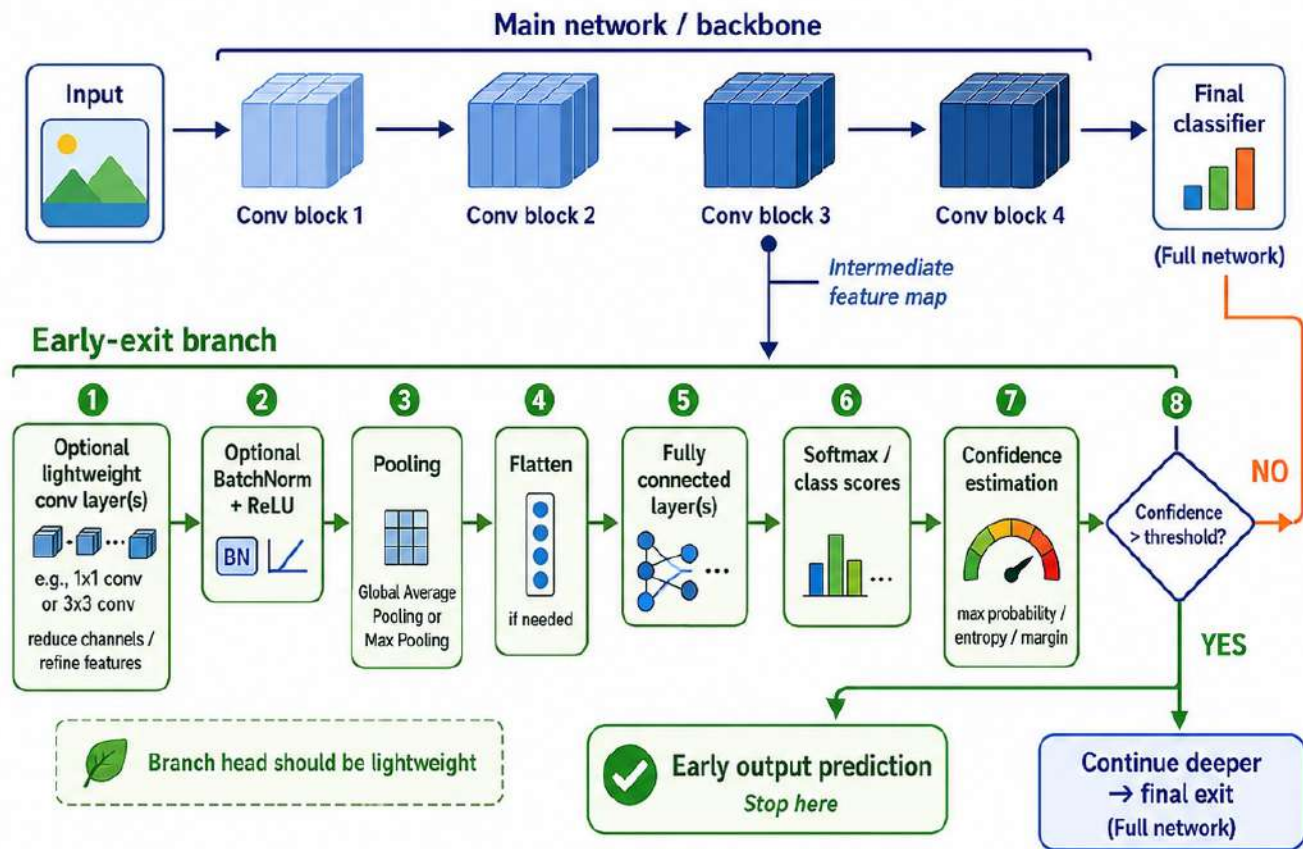
Throwback: Adaboost (Viola & Jones, 2001)

Remember the Cascade?

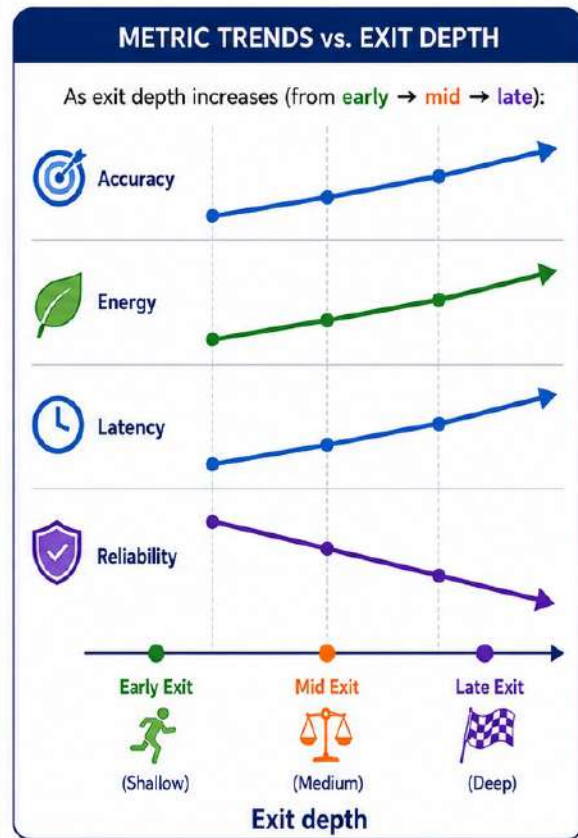
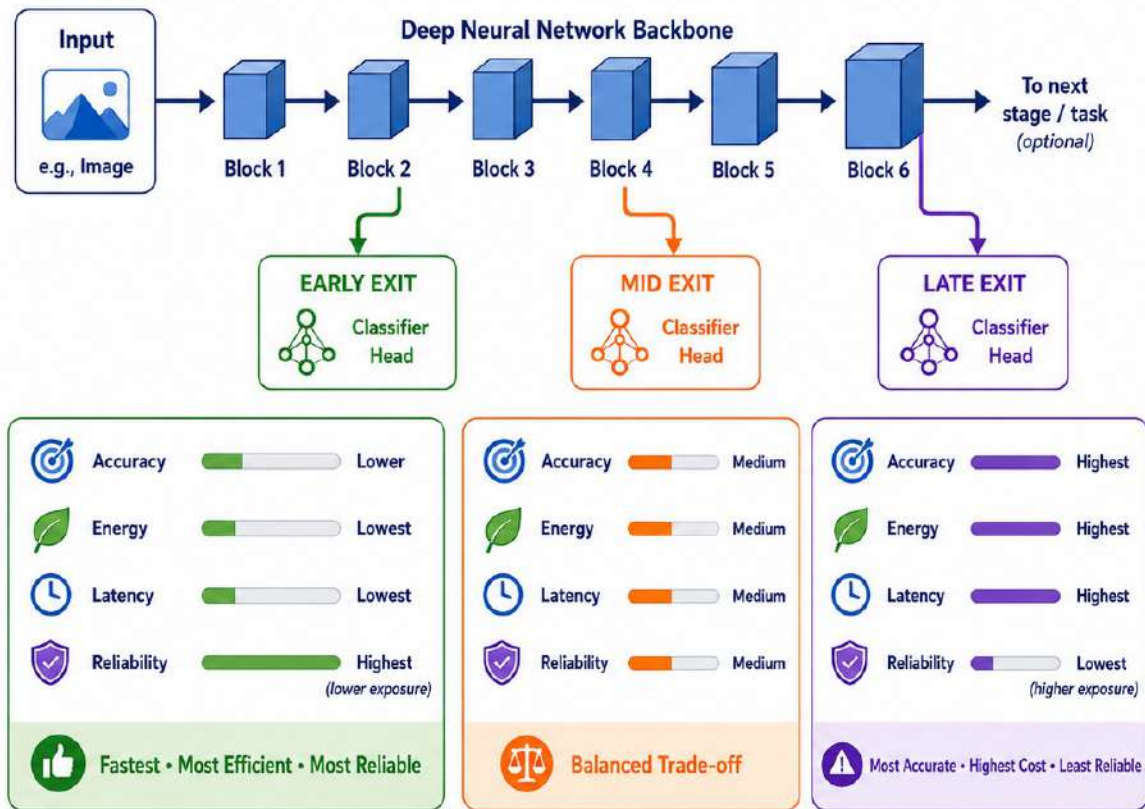


An early exit is implemented by attaching a small classifier head to an intermediate feature map. If the branch is confident enough, the network stops early; otherwise, inference continues through the deeper layers.

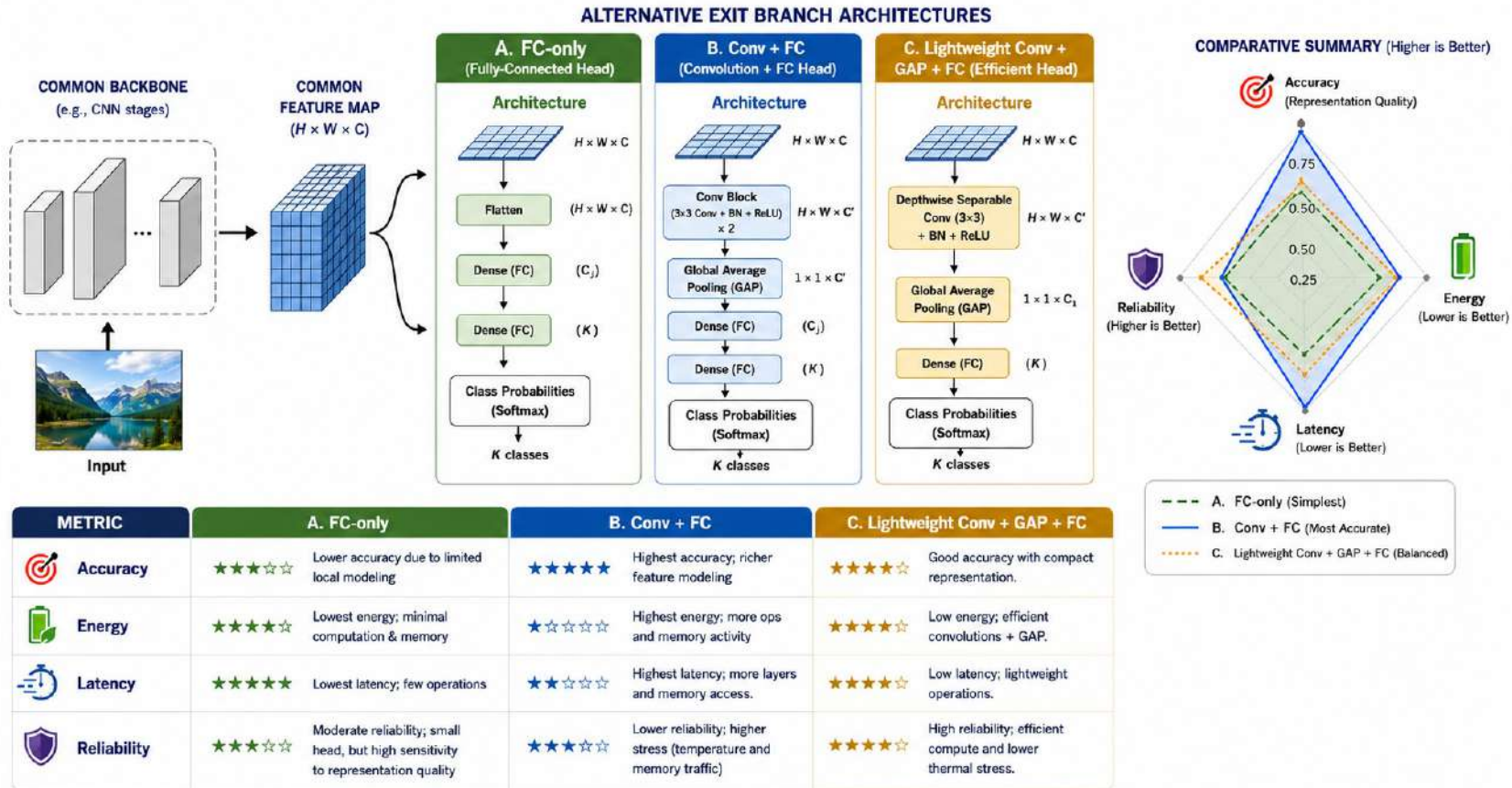
Early Exit Branch - Implementation



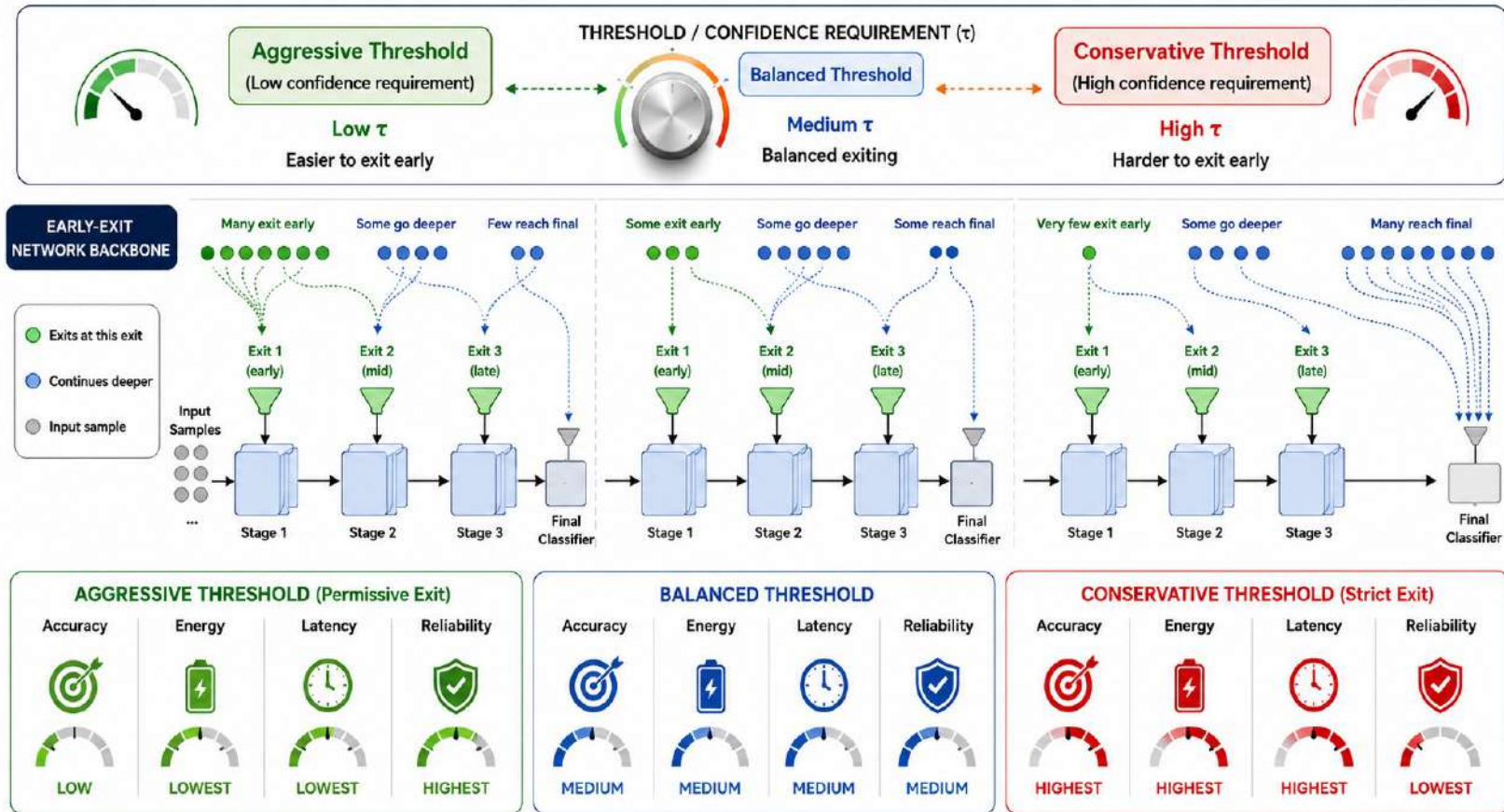
Exit Placement Impact



Exit Architecture Impact



Exit Strategy (i.e. Threshold) Impact



Success!



Success, of course, is in the eye of the beholder...

How do we evaluate success in our Adulthood?



- ☐ Family?
- ☐ Social Relationships?
- ☐ Money?
- ☐ A roof over our heads?
- ☐ Having a better car than our neighbor?

How do we evaluate Edge AI?

- ❑ Computation Savings --> Power/Energy

- Lesser # of MAC operations

Every design
Max # of Memory Accesses
only when necessary

- ❑ Latency (Performance)

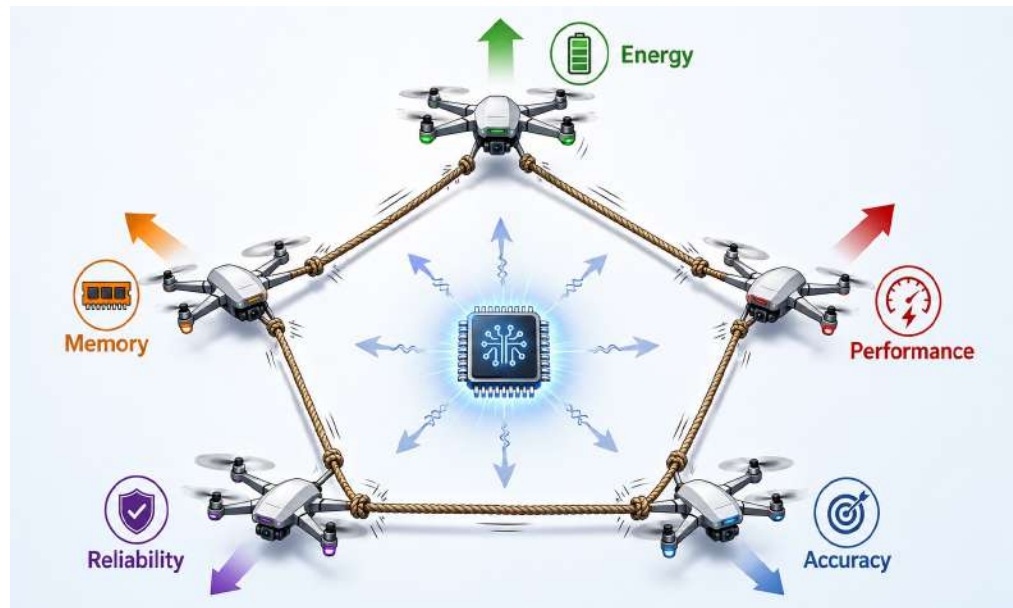
- Even in conservative approaches, average latency is smaller

- ❑ Reliability

- SEUs, Aging, other n/w faults

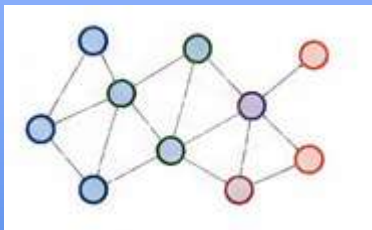
- ❑ Model Size

- Smaller amount of computational components
 - Hardware Reuse

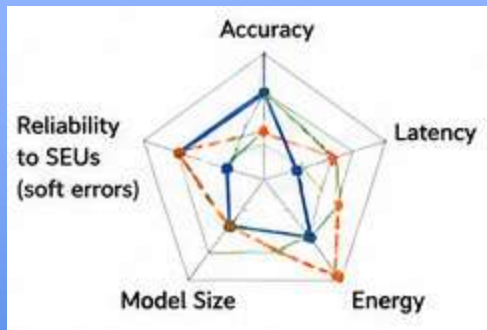


The Need for Design Automation

Huge Design Space



Multi-Objective Trade-Offs



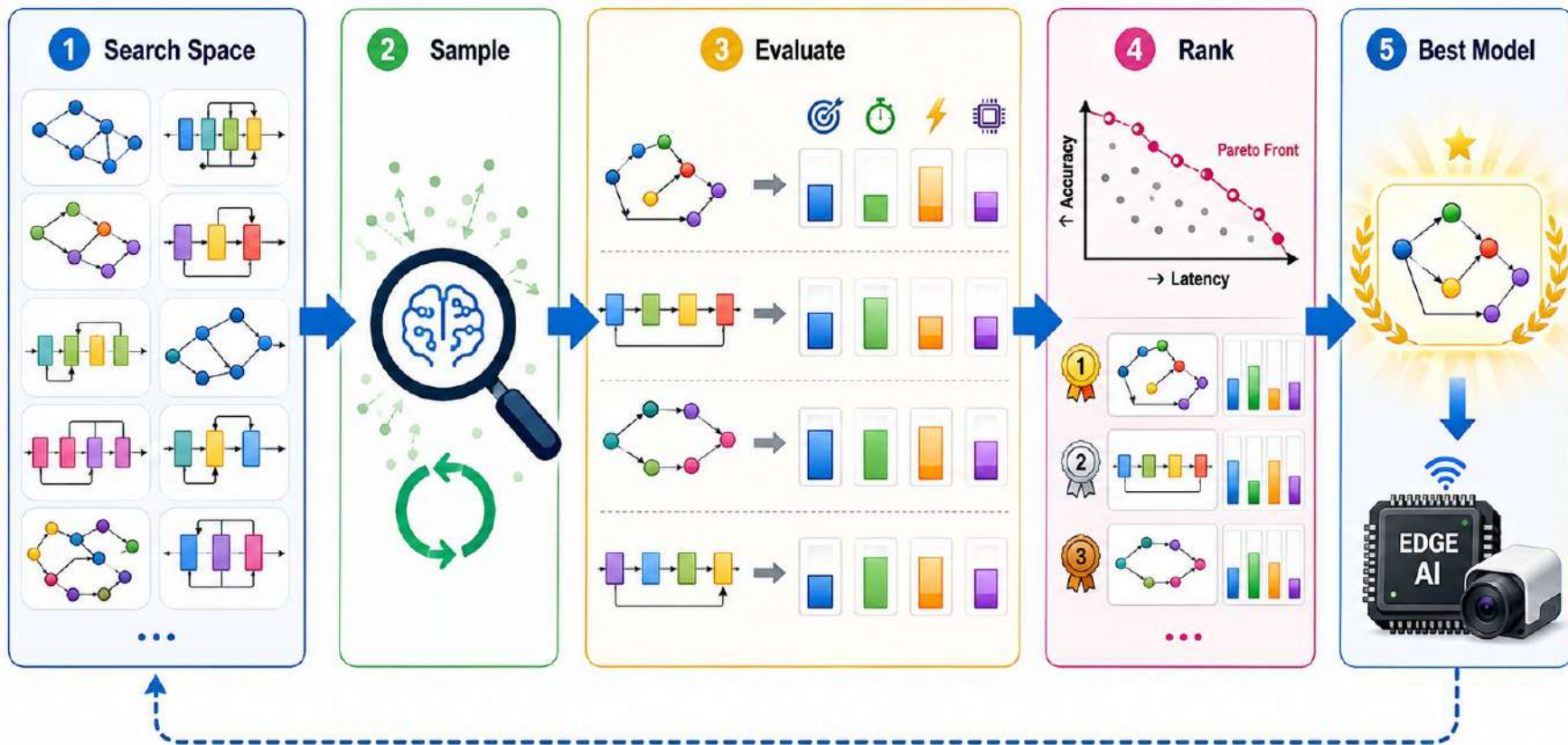
No single solution
optimal for all objectives

Costly Manual Evaluation



Exhaustive Search
Impractical

Neural Architecture Search



Part 5: The Senior (wise) Era

Knowledge is Power!



Knowledge is Power!

What about learning?



How can training help Edge AI?

Training

- **End-to-end models trained with Backprop!**
- Overfitting, local minima and vanishing/exploding gradients.

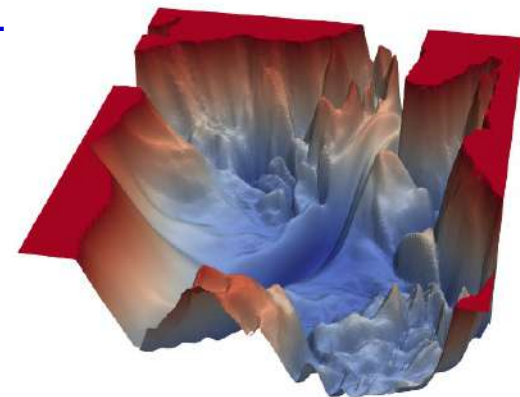
Backpropagation

- “Black-Box” approach compared to layer-wise techniques.

Alternatives to Backpropagation

- Predictive Coding
- Evolutionary Algorithms
- Competitive Hebbian Learning

VGG-56



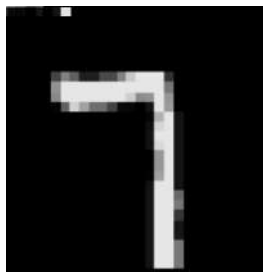
[1] Visualizing the Loss Landscape of Neural Nets, H. Li et al., 2018, 1712.09913-arXiv

Forward-Forward (FF) Algorithm

Advantages

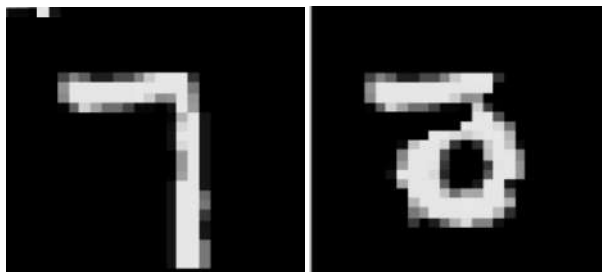
- +Computational efficiency – Low memory consumption.
- +Ability to work with unknown forward computation details.
- +Independent training of each layer.
- +**Layer-wise transparency.**

7 - True Class



x_i^+

3 - False Class



x_i^-

Input: Positive x_i^+ and Negative x_i^- data representation.

Geoffrey Hinton. The forward-forward algorithm: Some preliminary investigations, 2022

Focus on optimizing each layer!

Layer-wise Loss Function with Channel-wise Competitive Learning

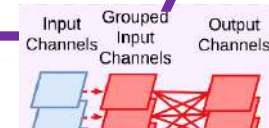
- Reformulates the goodness function to avoid negative data construction.
- Enables each CNN layer to act as an independent classifier.

Channel-wise Feature Separator and Extractor (CFSE) Block

- Incorporates channel-wise grouped convolutional layers to partition feature space.
- Facilitates learning of compositional features via standard non-separable convolutional layers.

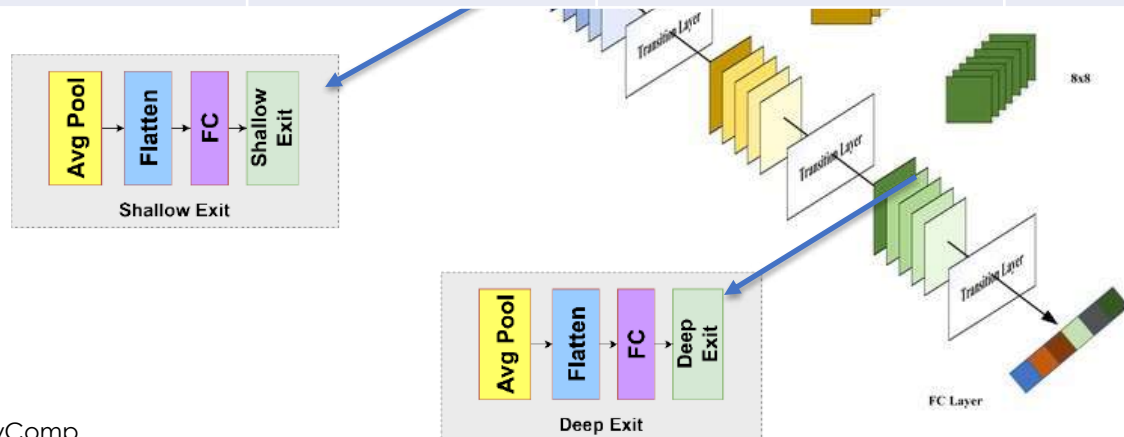
A. Papachristodoulou, C. Kyrkou, S. Timotheou & T. Theodoridis, "[Convolutional Channel-Wise Competitive Learning for the Forward-Forward Algorithm](#)", Proceedings of the AAAI Conference on Artificial Intelligence, 2024

Allows for Promising Performance w/ Early Exits!



Performance

Model	Accuracy	MAC Operations	Energy Consumption / Sample	Latency / Sample
End-to-End	85.4%	100%	100%	100%
Deep Exit	85.33%	92.08%	77.28%	80.45%
Shallow Exit	85.13%	76.54%	62.40%	65.50%

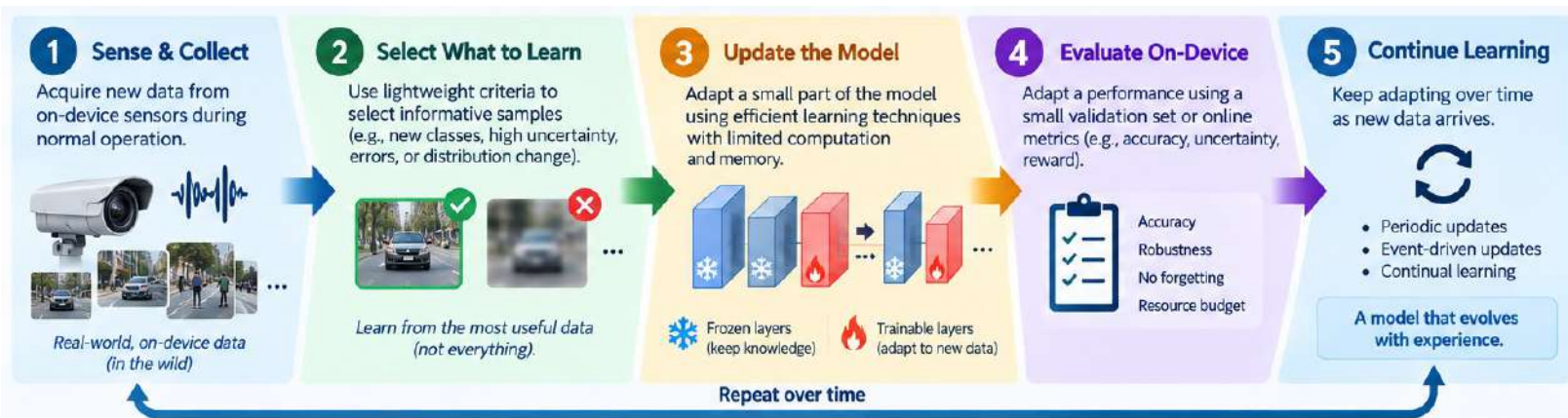


<https://github.com/andreaspapac/CwComp>

What about On-Device Learning? – Why?



On Device Learning - How

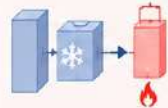


Efficient On-Device Learning Techniques

Update only what is necessary to meet resource constraints.

1 Train only the head

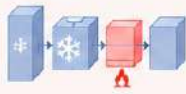
Freeze backbone, update classifier/head.



Low cost, effective for many tasks.

2 Parameter-efficient adaptation

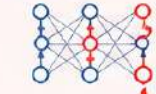
Use adapters, LoRA, prompts, or small trainable modules.



Small number of parameters to update.

3 Selective / sparse learning

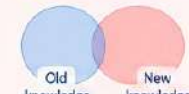
Update only certain layers, channels, or neurons.



Focus updates where change is needed.

4 Continual / incremental learning

Learn new data without forgetting old knowledge (e.g., rehearsal, distillation).



Maintain performance over time.

Resource-Aware Training on the Edge

Optimized for the constraints of edge devices.



Low memory footprint

Update a small subset of parameters. Use memory-efficient optimizers.



Low energy consumption

Lightweight updates, few training steps, quantized training (when possible).



Real-time operation

Learn in the background without interrupting primary tasks.



Hardware-aware implementation

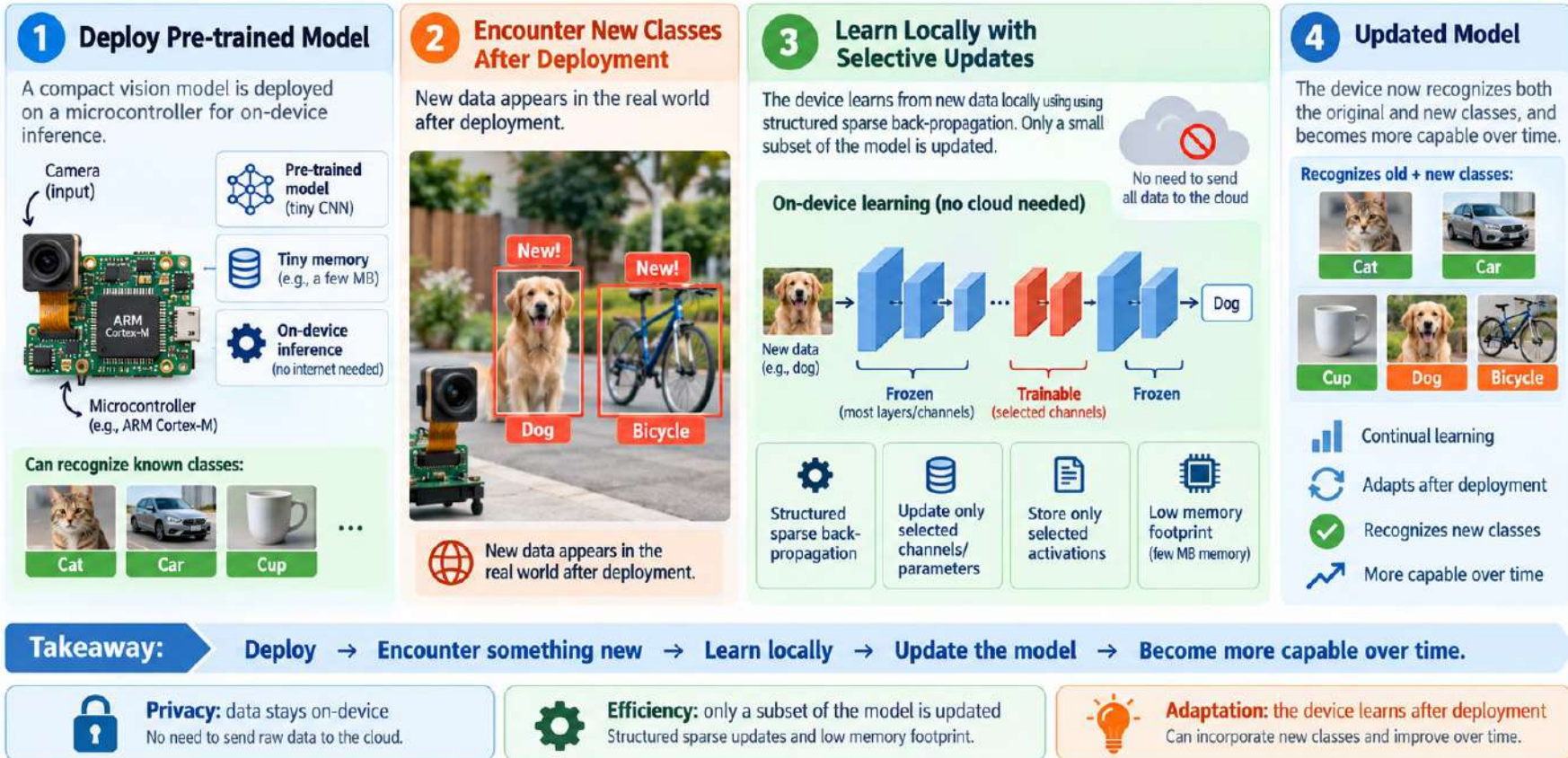
Leverage NPUs/GPUs, mixed precision, and efficient kernels.

Updated Model (Improved & Adapted)

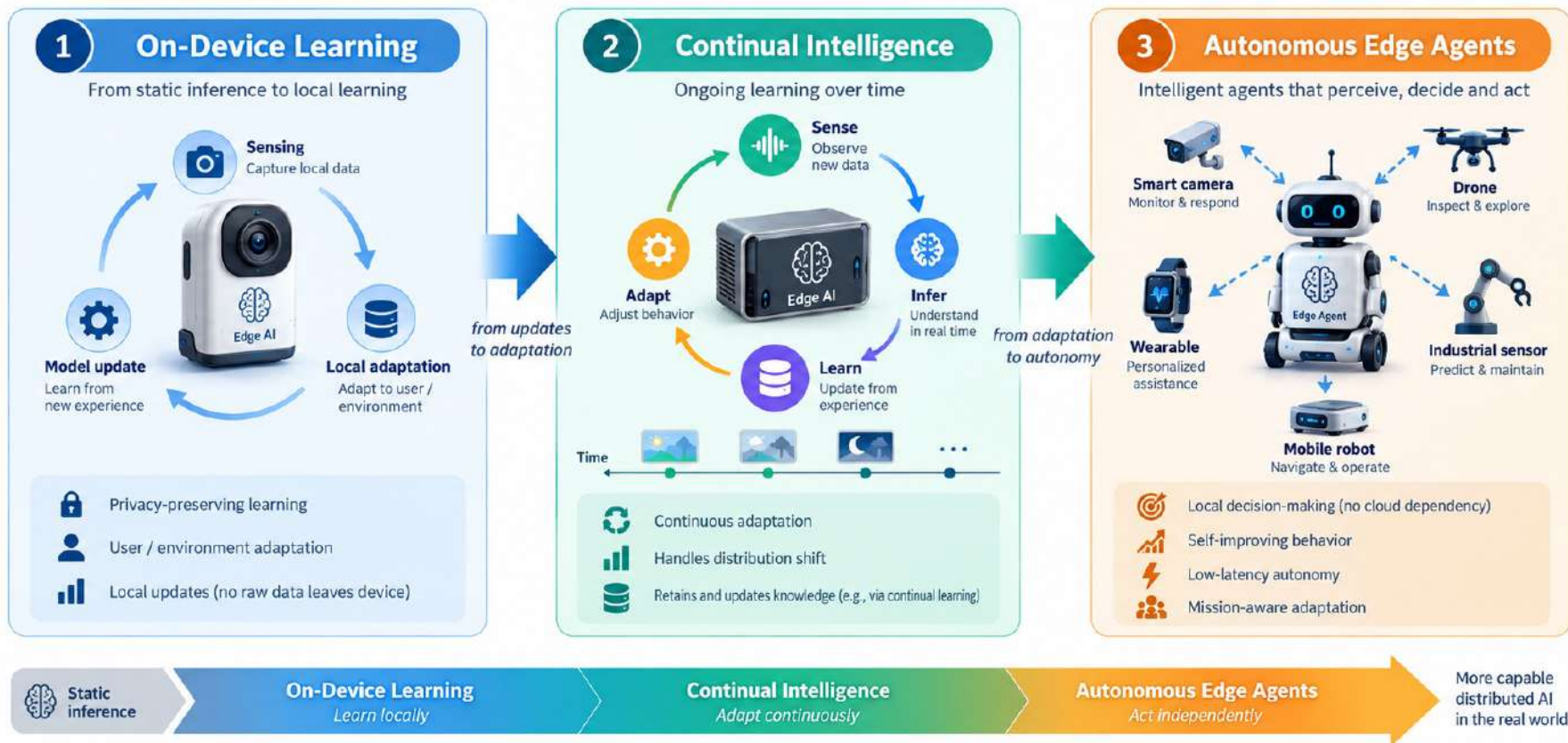


- ✓ Adapts to new data and conditions
- ✓ Improves over time
- ✓ More robust and personalized
- ✓ Works without constant cloud access

Example: MCU Vision



On-Device Learning

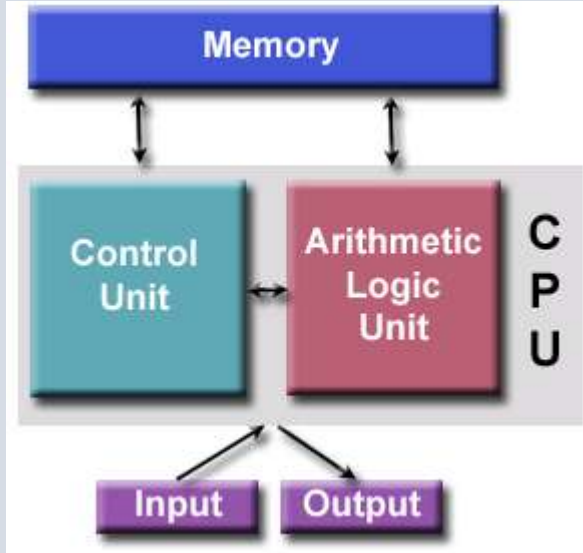


On-Device Learning - Challenges



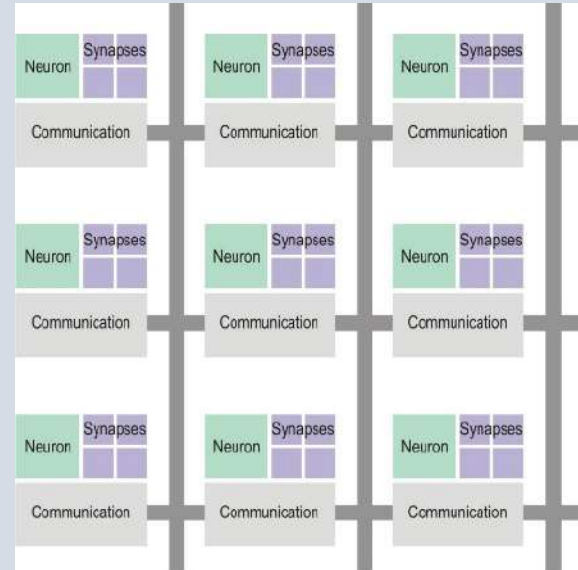
Post Von-Neumann Computing

Von Neumann Architecture



Source: <https://www.quora.com/How-does-the-Von-Neumann-architecture-provide-flexibility-for-program-development>

Non-Von Neumann Architecture



Source: <https://www.extremetech.com/extreme/187612-ibm-cracks-open-a-new-era-of-computing-with-brain-like-chip-4096-cores-1-million-neurons-5-4-billion-transistors>

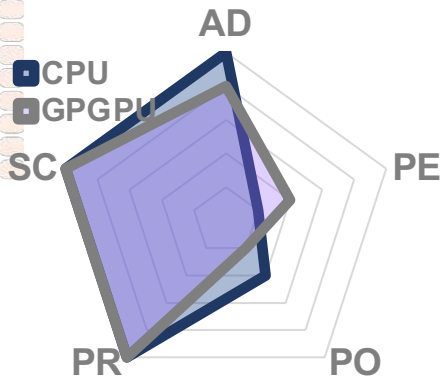
Technologies?

Application Specific IC

Misra et al, Neurocomputing, 2010

Adaptivity (AD)
Performance (PE)
Power Efficiency (PO)
Programmability (PR)
Scalability (SC)

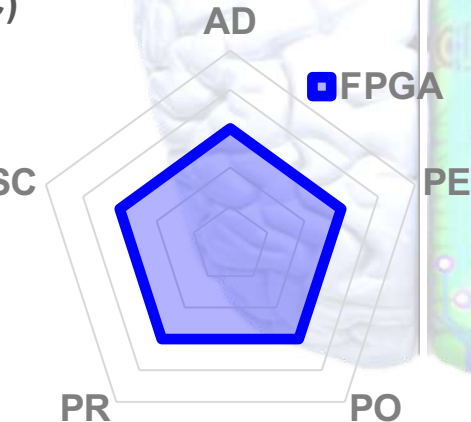
Ack: Helen Li and Yiran Chen



P. J. Fox, Tech. Report, 2013

Graf et al, NIPS, 2009

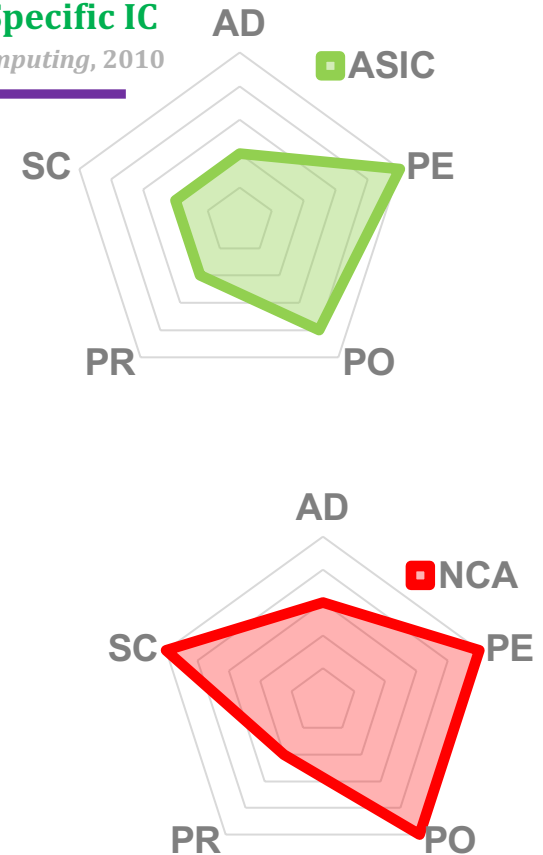
General Purpose Platform



Programmable Hardware

Graf et al, NIPS, 2009

Misra et al, Neurocomputing, 2010



Memristor Based Reconfigurable Design

H. Li, HPEC, 2010 4, DAC, 2015

How about CMOS/EDA?

❑ Are conventional CMOS and EDA technologies capable to support long-term research and development of AI?

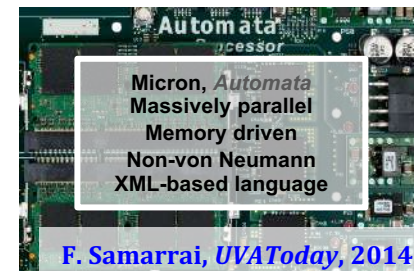
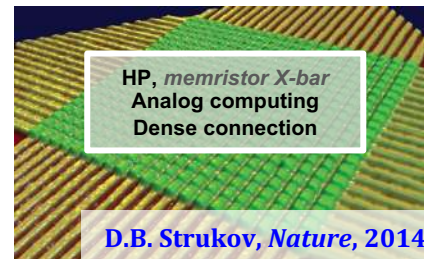
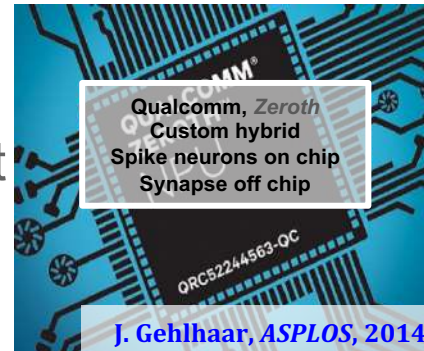
▪ Debates

- Analog or Digital?
- Spiking-based or level-based?
- Synchronous or asynchronous?
- CMOS or Post-Silicon?

▪ Other Challenges

- Programmability
- Reliability
- Scalability
- Security

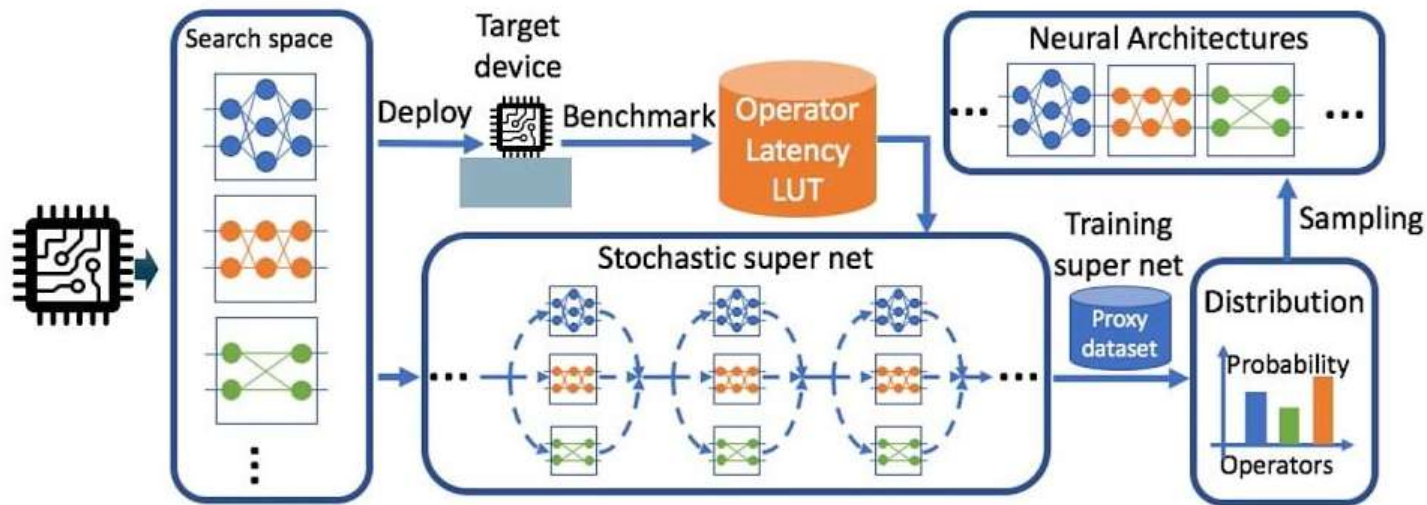
Ack: Helen Li and Yiran Chen



How About Neural Architecture Search & EDA?

Desirable features

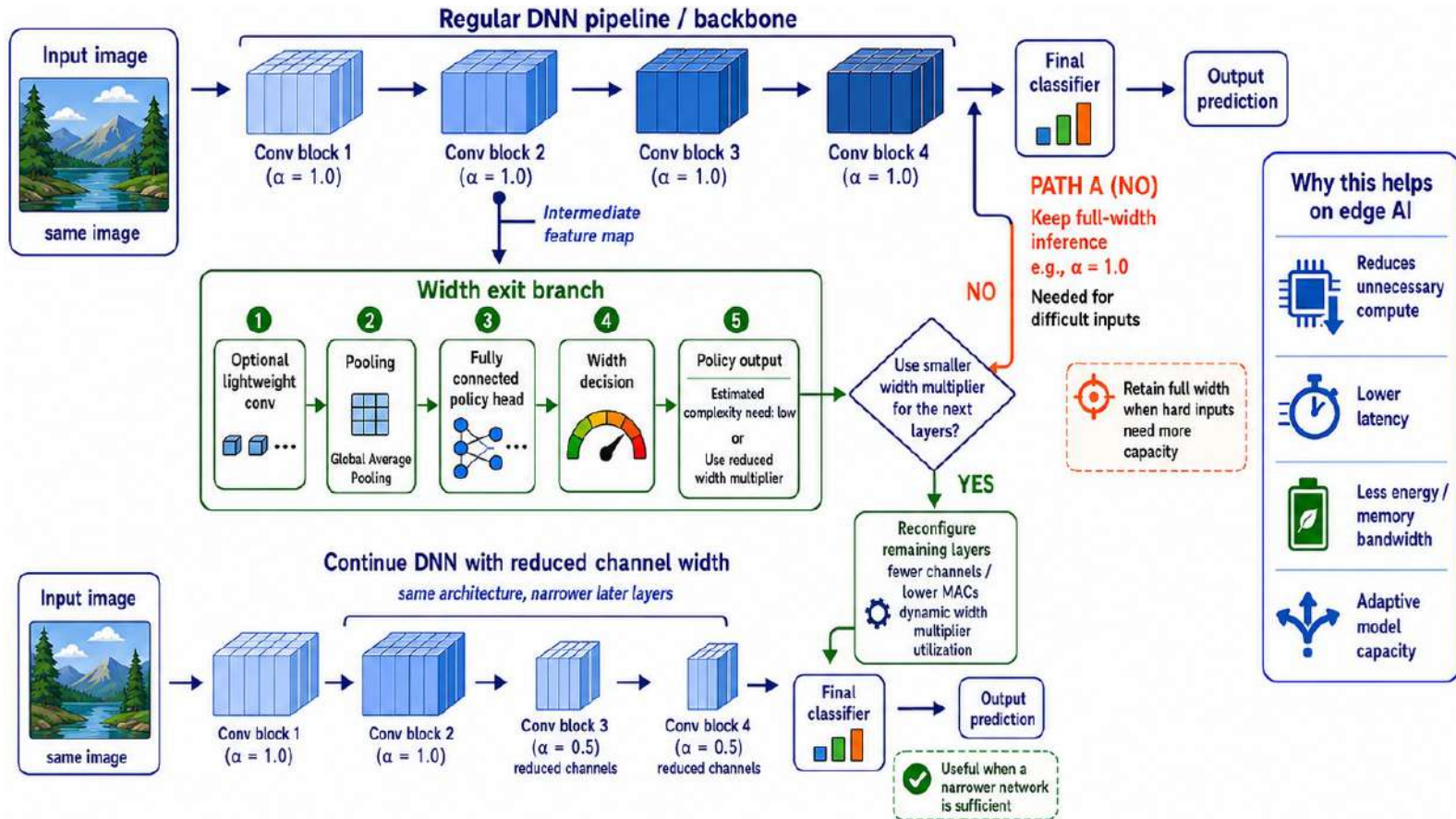
- A general framework to support arbitrary design spaces
- Optimize for actual efficiency metrics (such as latency)
- Reasonable search cost



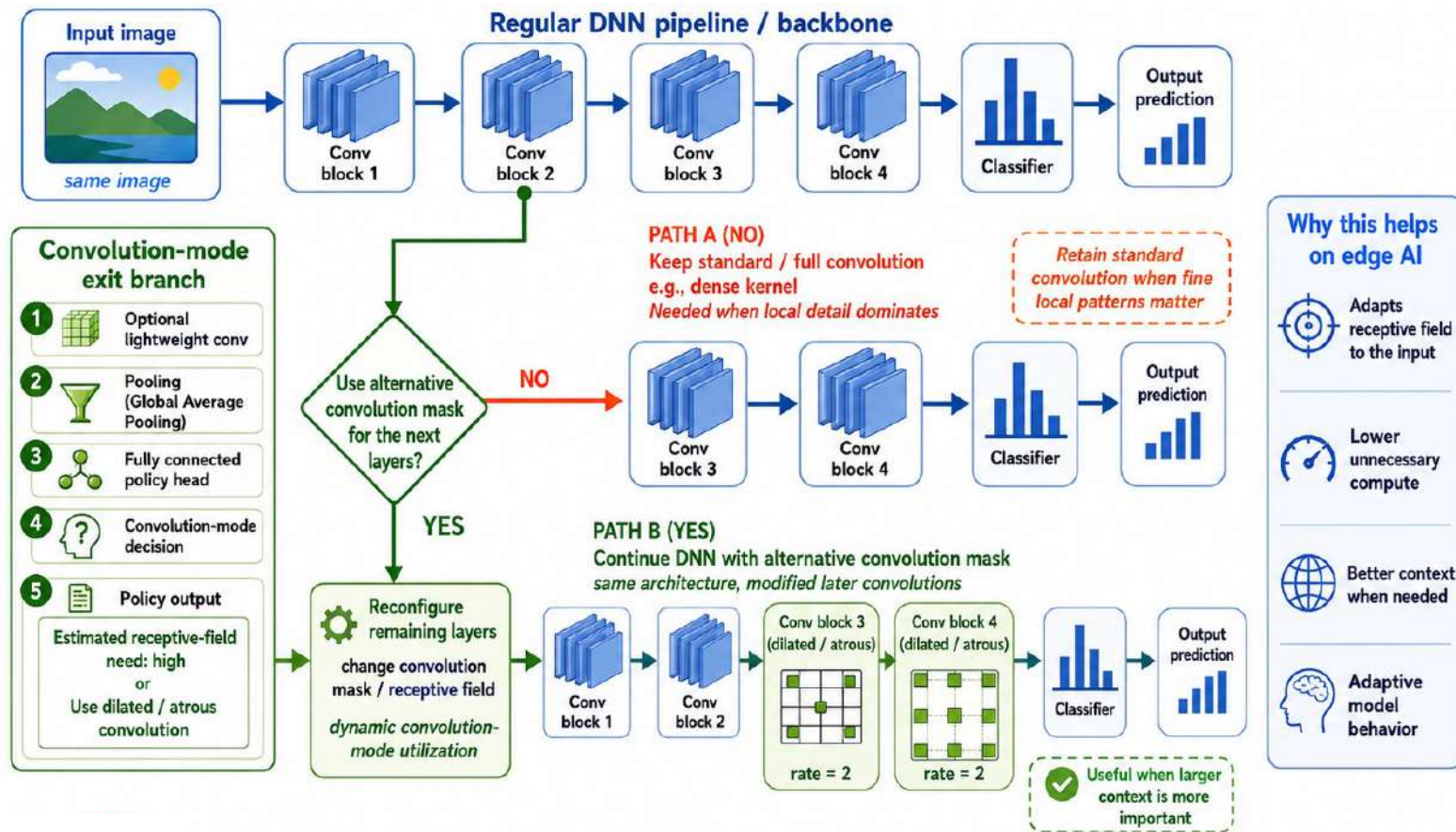
Opportunities and Challenges for our Senior Friend



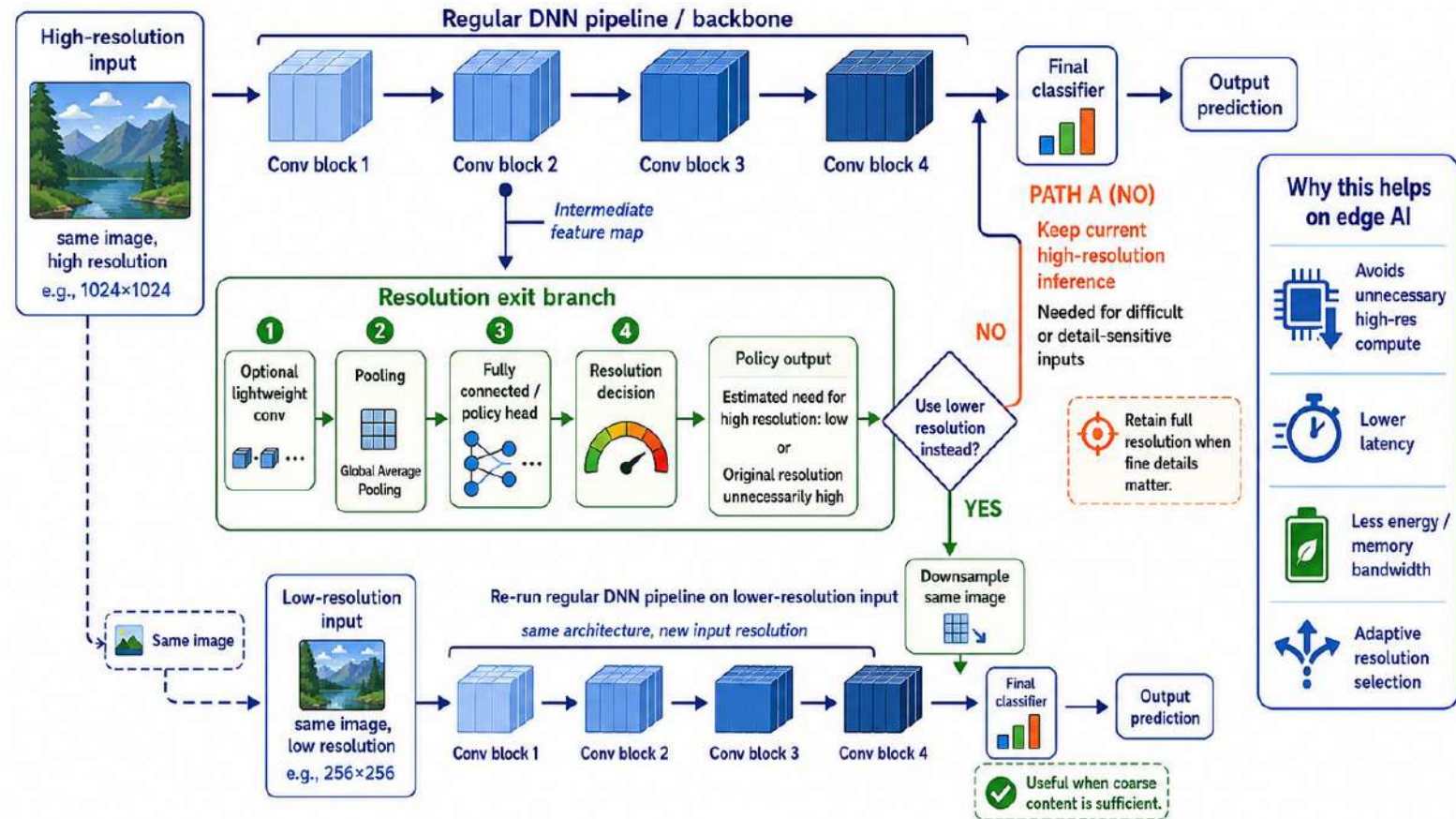
Dynamic Scenarios I – Conv Width



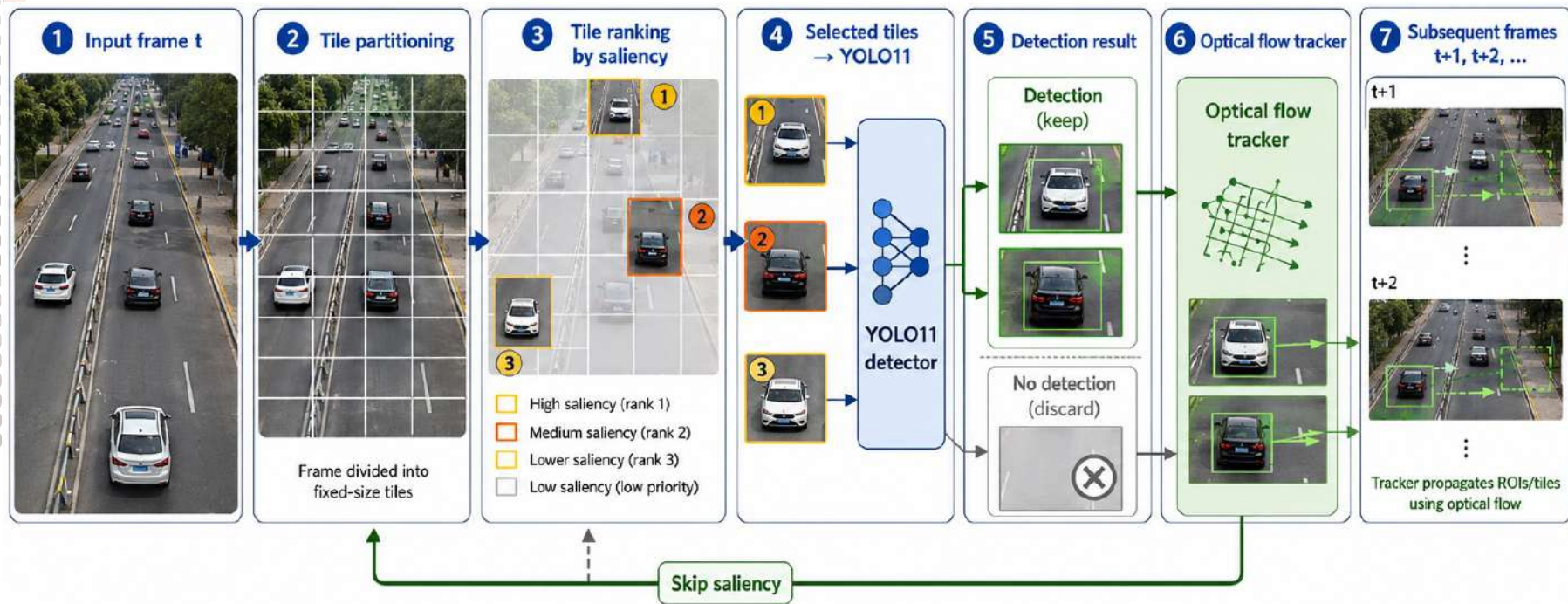
Dynamic Scenarios I – Kernel Mask



Dynamic Scenarios II - Resolution

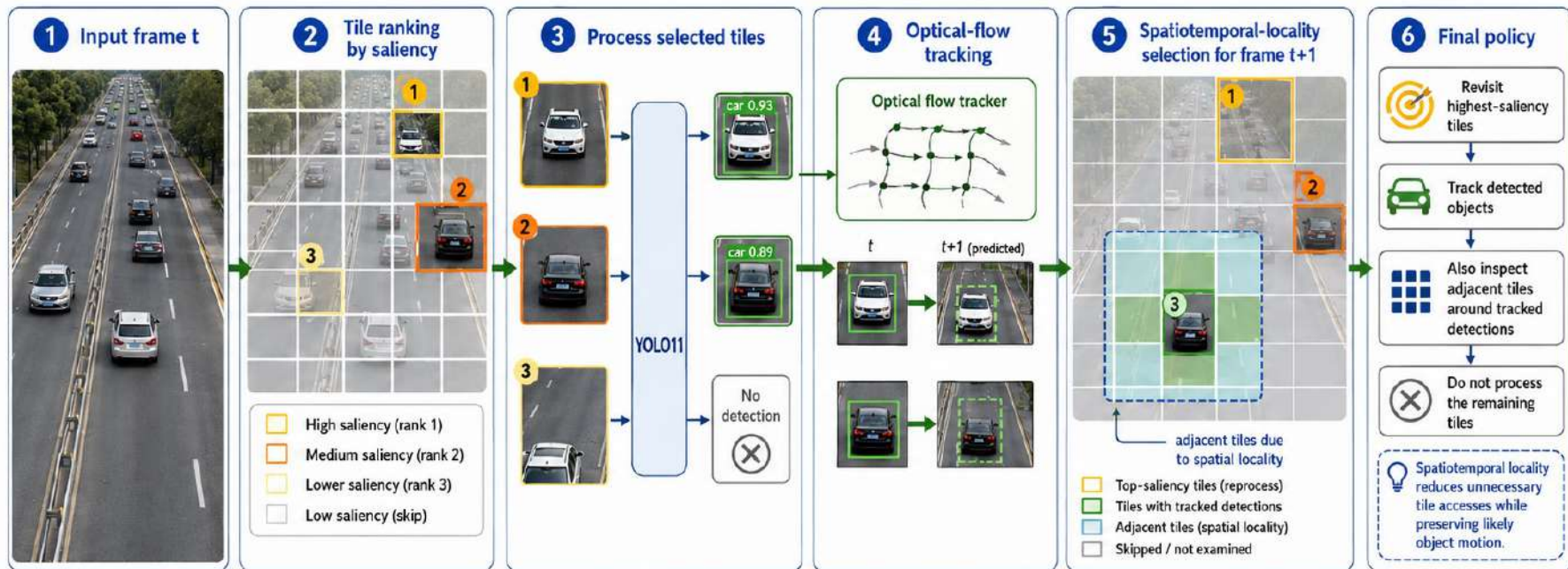


Dynamic Saliency based Tile Processing I



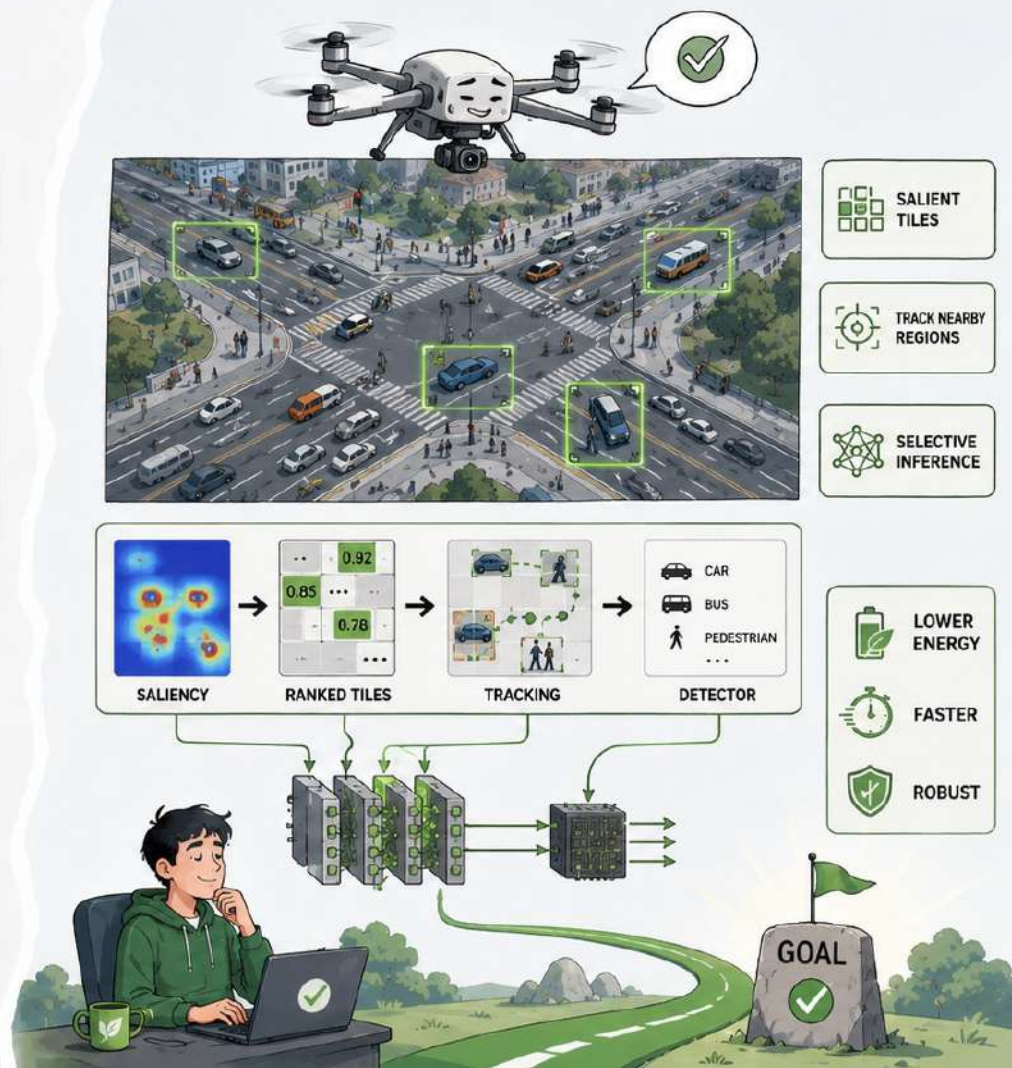
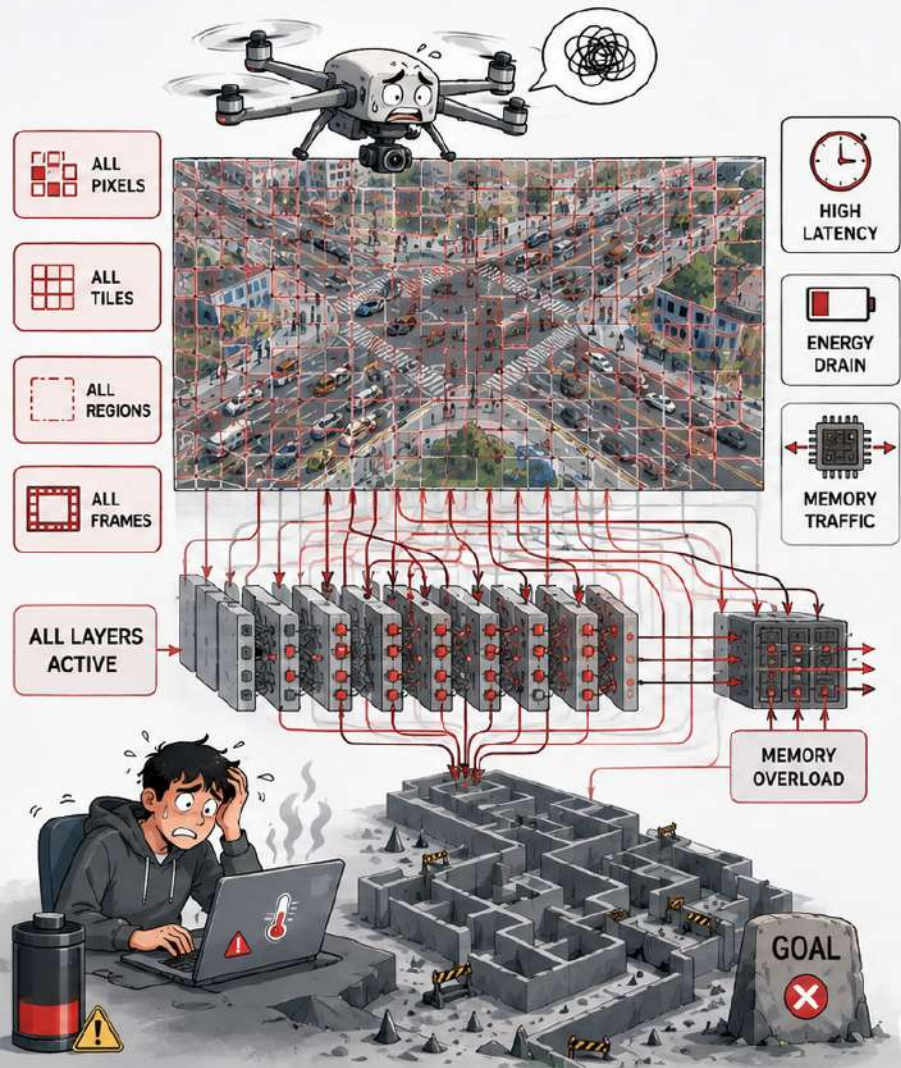
Tracking (human-like approach!)

Dynamic Saliency based Tile Processing II



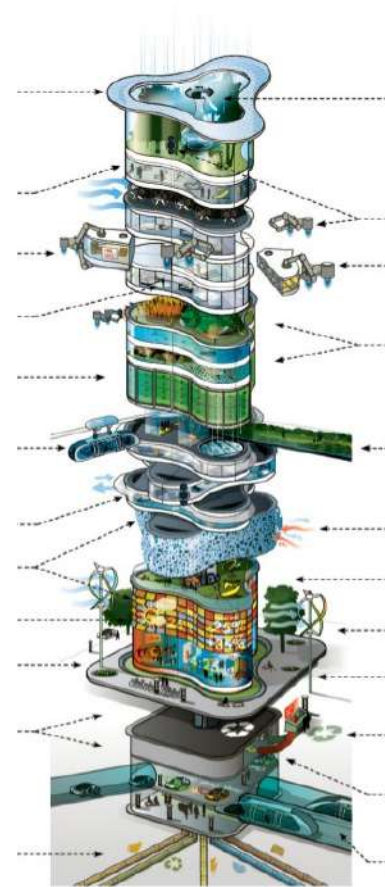
Spatiotemporal Locality (Cache-Like)





Conclusions

- Edge AI vital as data generation moves toward edge!
- Conventional DL models too large and energy-intensive for edge
- Model compression, quantization, pruning, dynamic networks, and efficient architectures enable practical deployment
- Bioinspired Approaches Provide Vast Options
- Edge AI is inherently a co-design problem, **requiring joint optimization across algorithms, models, hardware accelerators, and system architecture.**
- Future research will focus on adaptive models, specialized accelerators, and collaborative edge–cloud intelligence to enable scalable and reliable AI at the edge.





As you set out for Ithaca, hope the voyage is a long one...



Full of Adventure...



Full of Discovery.



ITHACA
Destination

COMPUTE WISELY

Learn along the way
Arrive with meaning

"The greatest
teacher,
failure is"
~Yoda



@MrSorensen805



Explore
with Purpose



Focus on
What Matters



Learn, Share,
Elevate Others



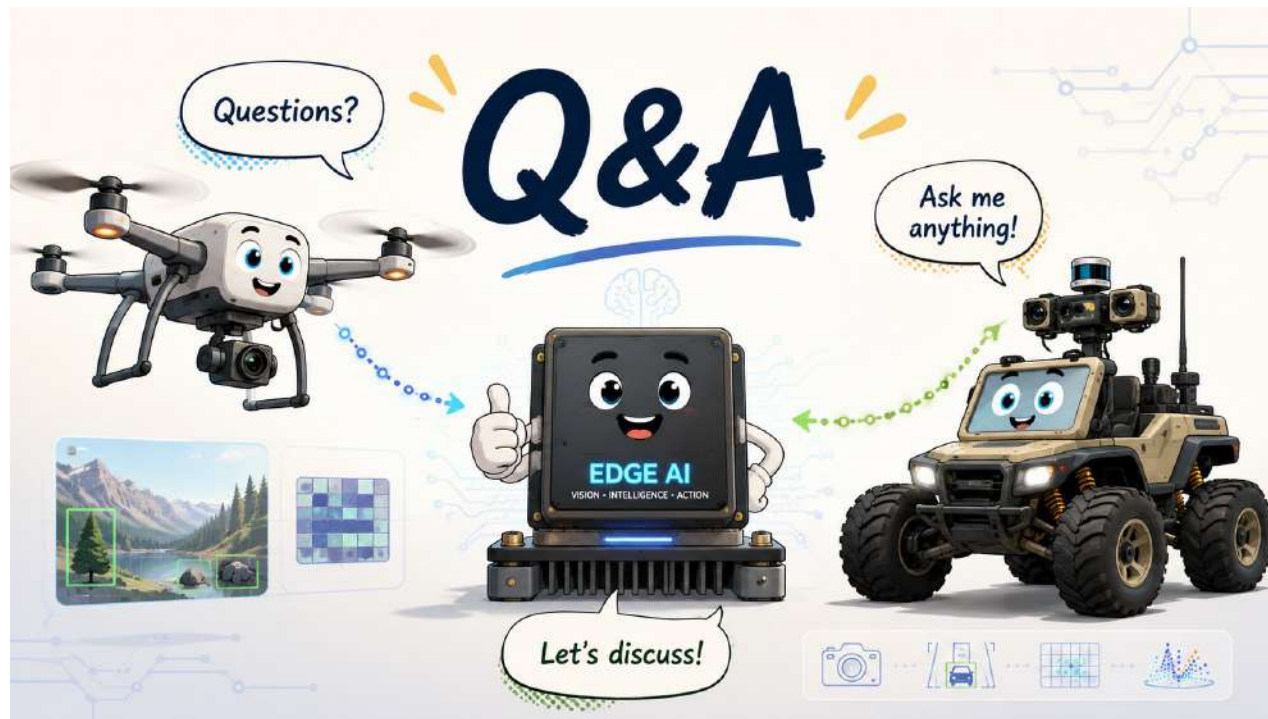
Discover
What's Next



Leave Things
Better

"Edge Intelligence: Time for the Edge to Grow Up!"

Now, your turn to NOT be lazy and ask questions!



Funded by:



KIOS



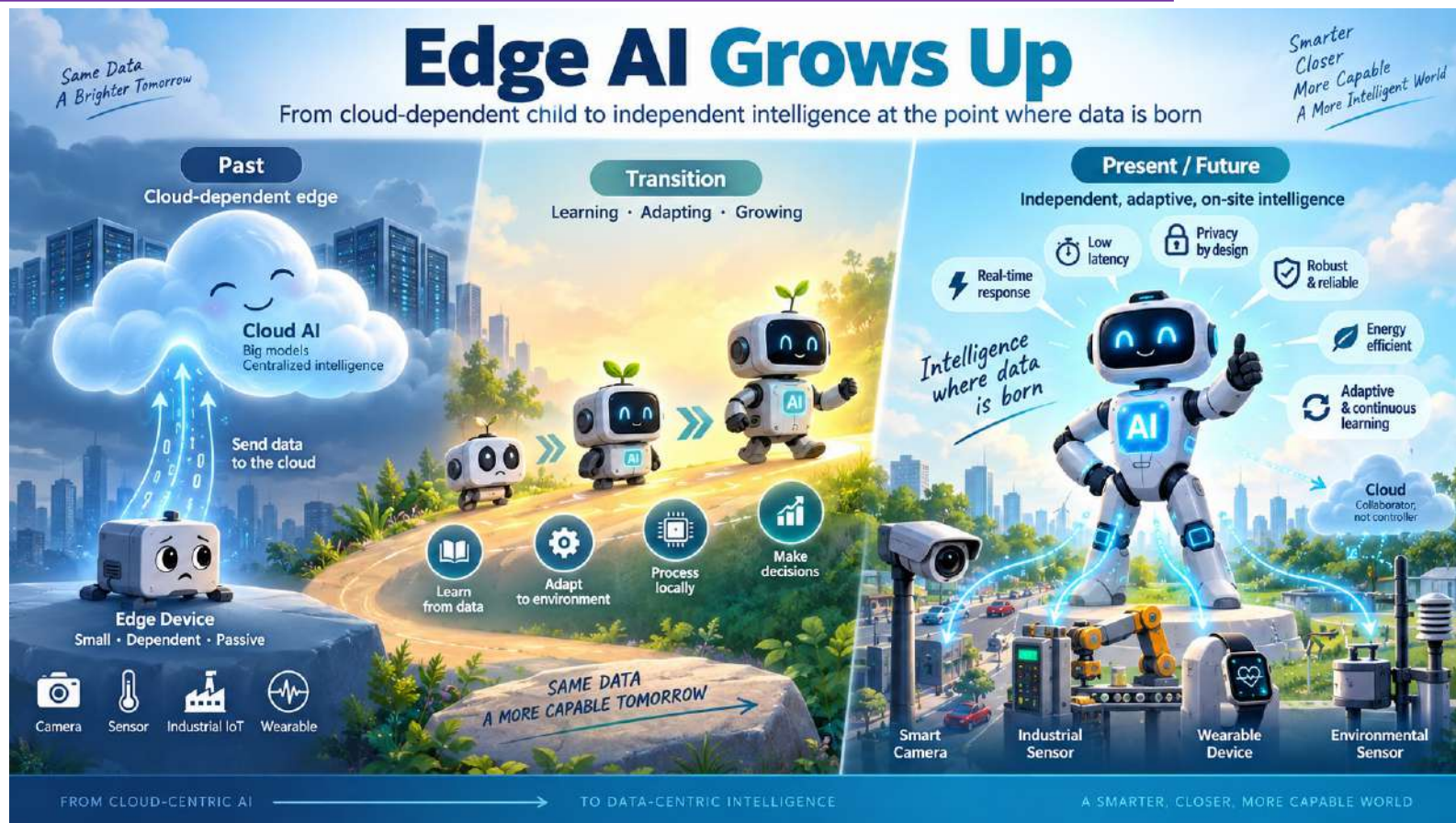
TIRAMISU

Acknowledgement

Parts of this work were supported by the European Union (i. Horizon 2020 Teaming, KIOS CoE, No. 739551, and from the Government of the Republic of Cyprus through the Deputy Ministry of Research, Innovation, and Digital Policy.

Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council Executive Agency. Neither the European Union nor the granting authority can be held responsible for them.

THANK YOU!



“Edge Intelligence: Time for the Edge to Grow Up!”