Hands-on TinyML for IoT, Bringing Intelligence to the Edge

Francesco Paissan, Alberto Ancilotto, Elisabetta Farella

Outline

- Why porting AI algorithms at the edge is important
- Challenges of AI at the edge
- Bringing intelligence to the edge:
    - From the bottom up;
    - From the top down;
- Hands-on tutorials:
    - Neural network design and training;
    - Model conversion;
    - Model deployment

Housekeeping rules

- Keep your mic muted (initially muted by default) unless you are asking questions
- Q&A - there will be dedicated moment. However, you can also…
  - …use the chat (I'll review and answer questions from time to time)
  - …raise your hand (feel free to interrupt particularly during the hands-on sessions)
  - …reach out offline at fpaissan@fbk.eu OR @fpaissan_ on X
- Turn on the camera (and the mic, of course!) when asking questions

Fondazione Bruno Kessler

PROFILE

- Fondazione Bruno Kessler (FBK) is a not-for-profit public research center, the result of a history that is more than half a century old.

MISSION

- FBK aims to excellence in science and technology with particular emphasis on interdisciplinary approaches and to the applicative dimension.

FONDAZIONE BRUNO KESSLER

FUTURE BUILT ON KNOWLEDGE

FBK at a glance

400+
researchers

140+
PhD students from 25 different Countries

700+
students involved in the FBK activities

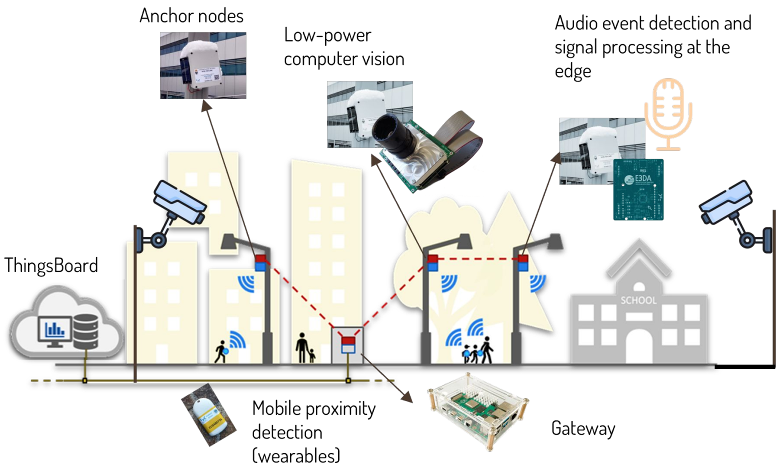
200+
thesis students, visiting professor, visitors

3,500 sq m
labs for scientific research

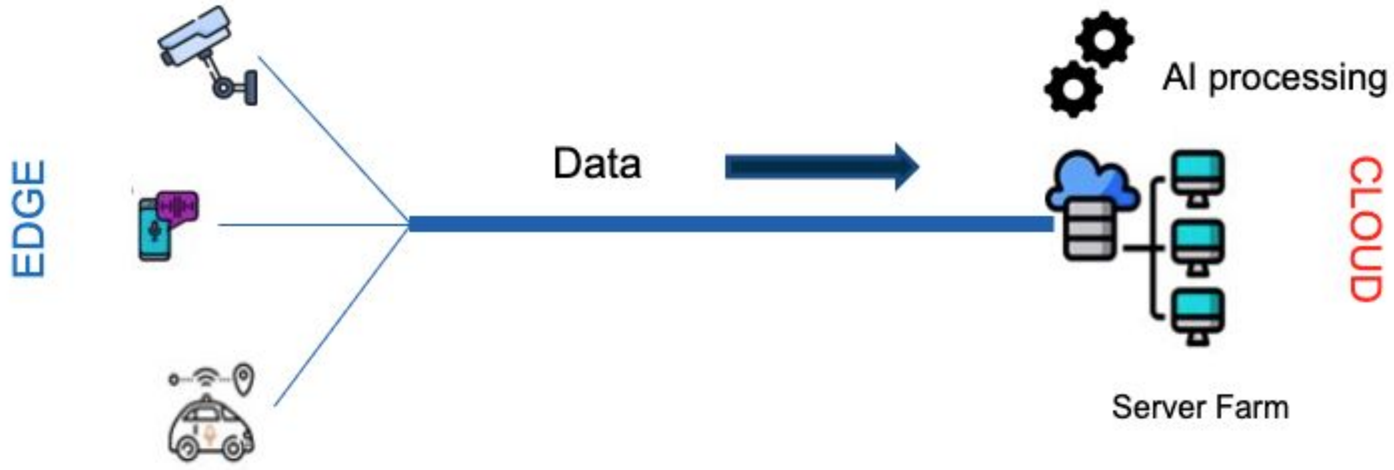
230,000

E3DA research unit

- **AI at the very edge**: artificial intelligence and advanced signal processing on resource-constrained wireless sensing platforms (MCU-based end-devices);
- **Energy efficient wireless embedded systems**: Wireless Sensor Networks, Wearable electronics, Internet of Things;
- Application in HCI, **smart cities** (always-on and event-based audio and vision sensing), rehabilitation, context-aware scenarios and **ambient intelligence**.



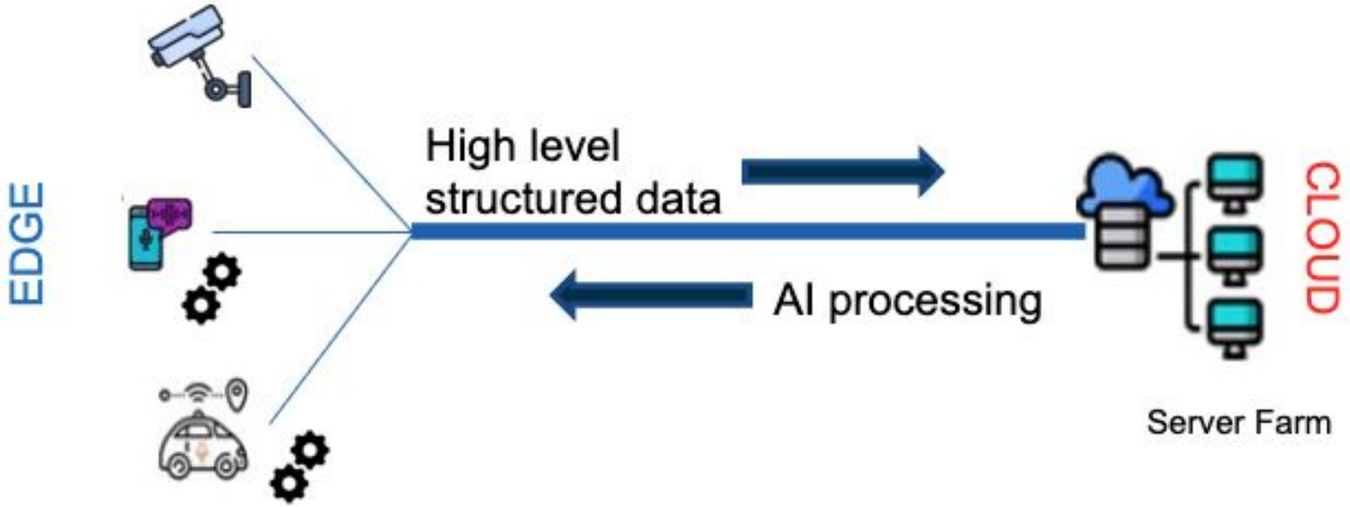
Centralized intelligence



Advantages
Computational resources, management of models and services

Issues
Energy and bandwidth for data transfer, latency, security and privacy of the data

Moving AI to the edge



Privacy ✅
Bandwidth/Energy/Latency ✅
Resources ❌
Device heterogeneity ❌

Scientific challenges

Edge-devices are typically characterized by:

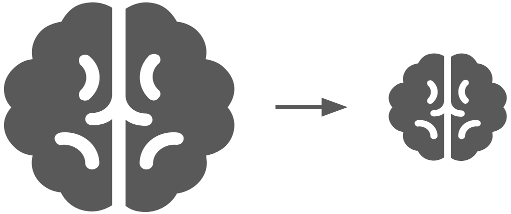
- Limited computational power (extremely limited in case of micro-controllers)
- Limited memory
- Fixed point representation
- Limited operation per second (GFLOPS – MMAC)
- Limited size -> limited energy budget/battery operated

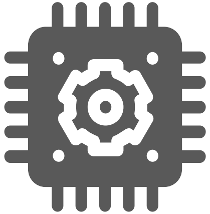
Edge nodes are not suitable for current AI algorithms and models

Current AI algorithms are not suitable for edge nodes

Resource and Energy Aware AI algorithms

TinyML: how?





<u>Reducing SoTA algorithms</u>:

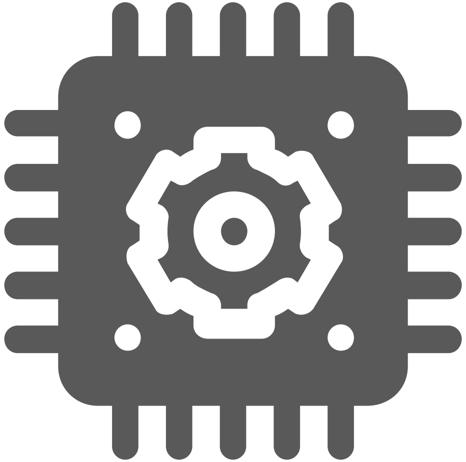
- Showcases drawbacks of current techniques;
- Generally focuses on **model compression**

<u>Build ad-hoc algorithms for MCUs</u>:

- Typically reduces to building simple pipelines based on **classical ML** algorithms;
- Applications are usually very specific to their domain;

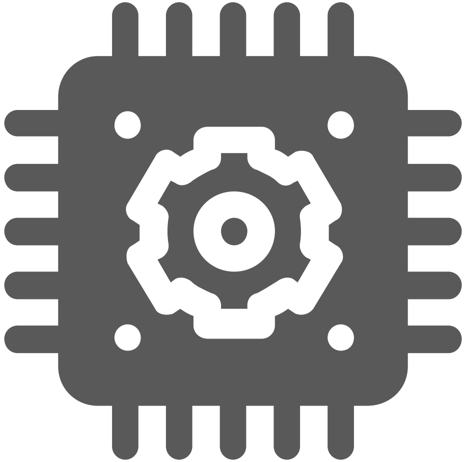
Target of TinyML

small memory footprint
(KBs - MBs)



Target of TinyML

small memory footprint
(KBs - MBs)

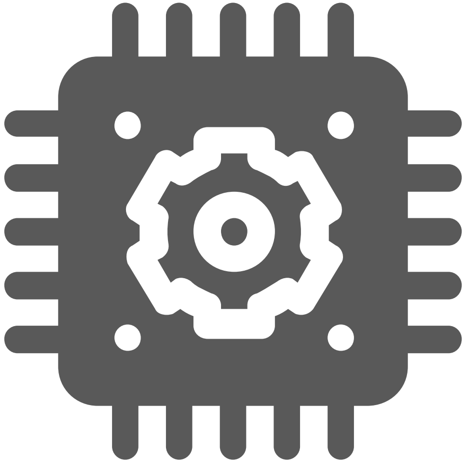


low complexity
(millions of ops/s)

Target of TinyML

small memory footprint
(KBs - MBs)

low complexity
(millions of ops/s)

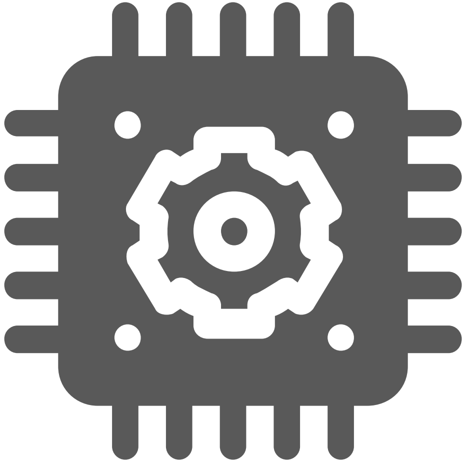


low resolution
(complexity and RAM)

Target of TinyML

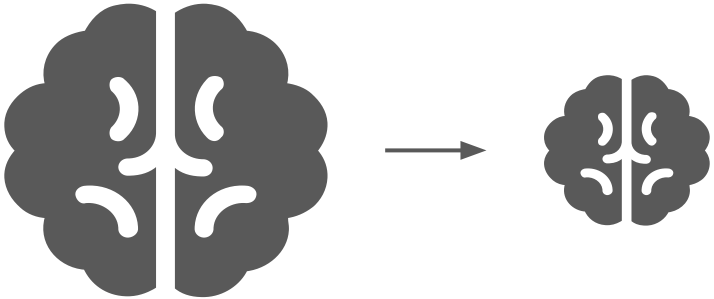
small memory footprint
(KBs - MBs)

low complexity
(millions of ops/s)



low resolution
(complexity and RAM)

low working memory
(KBs)



Reducing SoTA deep learning algorithms

Distillation, pruning, (quantization)

Knowledge distillation

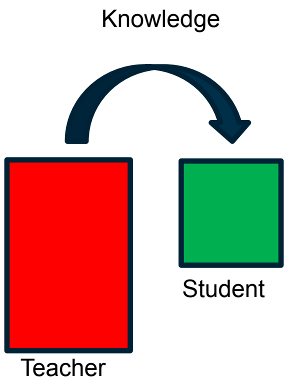
2015

Distilling the Knowledge in a Neural Network

Geoffrey Hinton[*†]
Google Inc.
Mountain View
geoffhinton@google.com

Oriol Vinyals[†]
Google Inc.
Mountain View
vinyals@google.com

Jeff Dean
Google Inc.
Mountain View
jeff@google.com



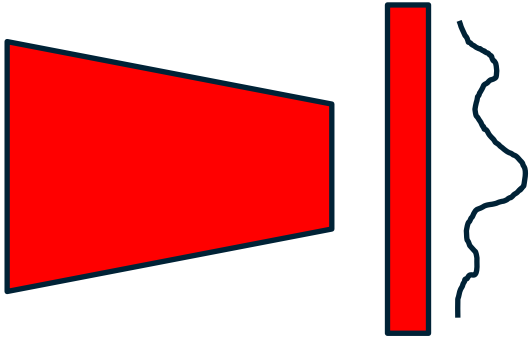
Soft labels (as opposed to hard labels)

$$q_i = \frac{exp(z_i/T)}{\sum_j exp(z_j/T)}$$

Softmax output of the teacher model

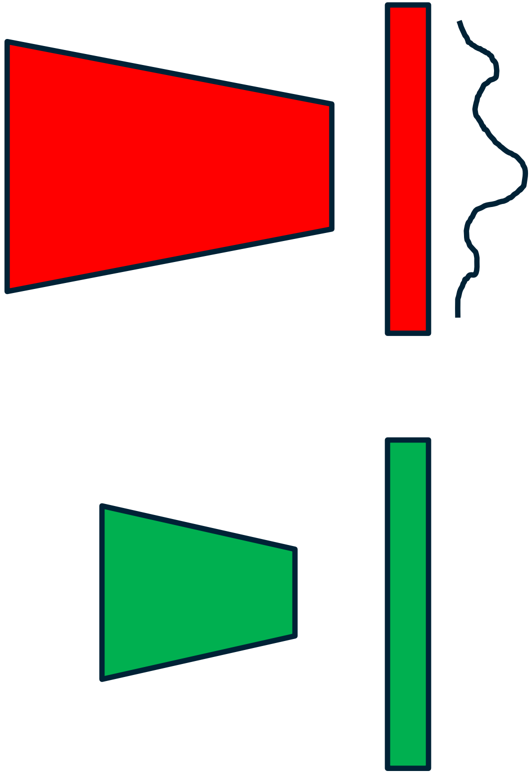
- **Exploits knowledge** from pre-trained, big neural network to facilitate the task of learning;
- Generally achieves better performance wrt to training from scratch;

Knowledge Distillation – Student Teacher



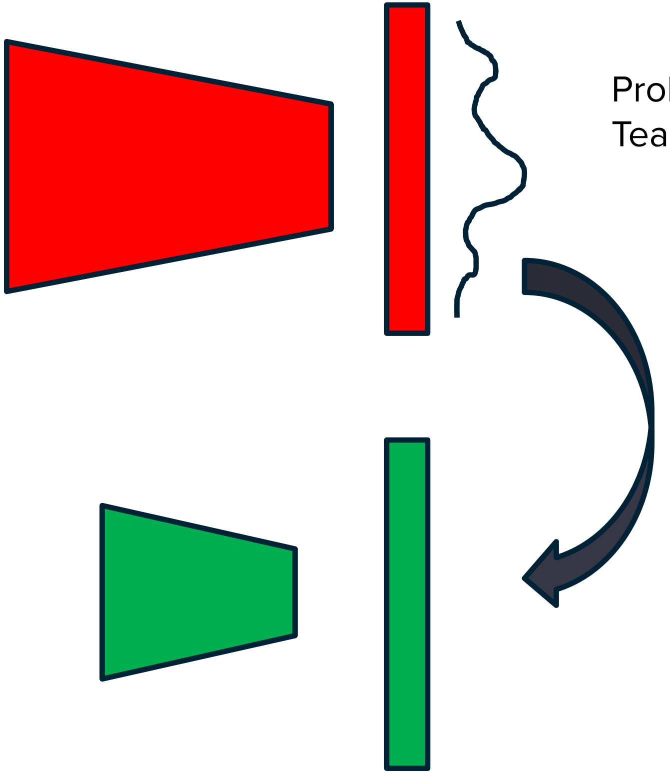
Probability distribution of the classes estimated by the Teacher model

Knowledge Distillation – Student Teacher



Probability distribution of the classes estimated by the Teacher model

Knowledge Distillation – Student Teacher



Probability distribution of the classes estimated by the Teacher mode

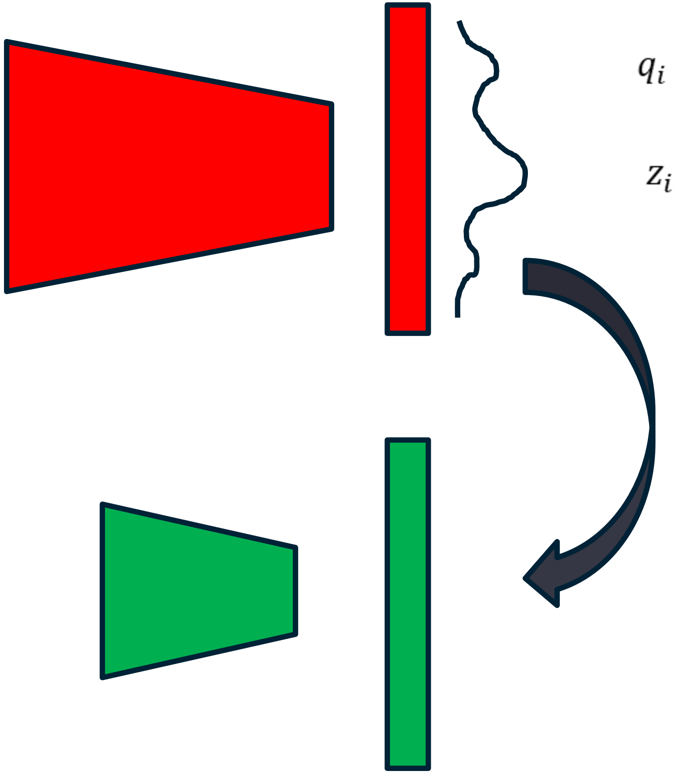
Kullbach Leibler divergence

$$\mathcal{L}_{KD} = \sum_{n} \sum_{i} q_i \log(z_i) \,/\log(q_i)$$

Cross entropy

$$\mathcal{L}_{KD} = \sum_{n} \sum_{i} q_i \log(z_i)$$

Knowledge Distillation – Student Teacher



q_i Probability distribution of the classes estimated by the Teacher mode

z_i Output of the student network for class *i*

Kullbach Leibler divergence

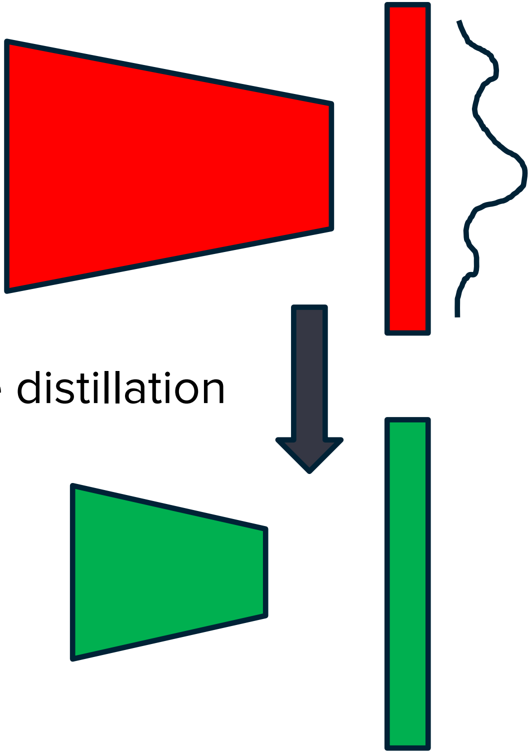
$$\mathcal{L} = \sum_n \sum_i q_i \log(z_i) \,/ \log(q_i) + \sum_n z_{i=y}$$

Cross entropy

Hard labels

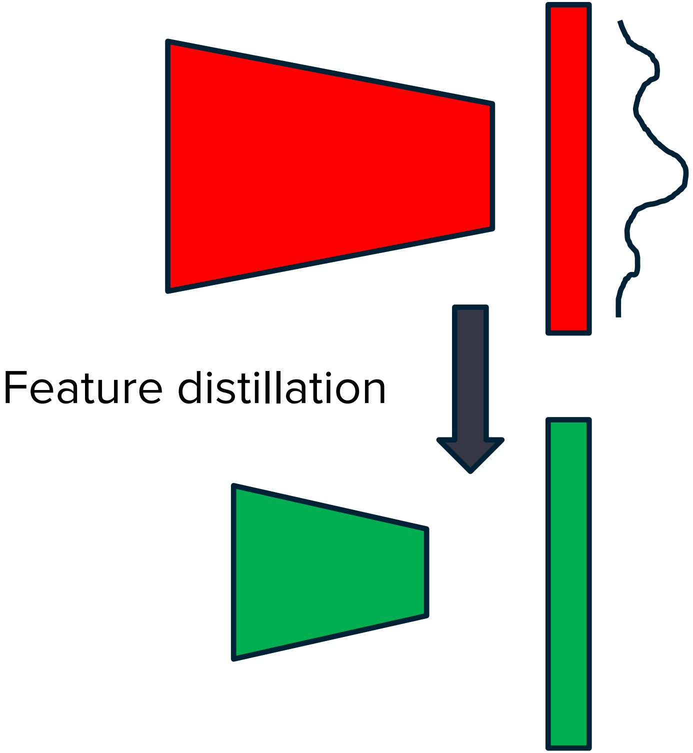
$$\mathcal{L} = \sum_n \sum_i q_i \log(z_i) + \sum_n z_{i=y}$$

Knowledge Distillation – Student Teacher



Feature distillation

Knowledge Distillation – Student Teacher



Embedding similarity loss

$$\mathcal{L}_{\mathrm{e}}(\mathbf{X}) = \sum_{n=1}^{N} ||v_t(\mathbf{x}_n) - v_s(\mathbf{x}_n)||^2$$

Euclidean distance
Cosine similarity
....
Etc.

$$\mathcal{L} = \sum_n \sum_i q_i \log(z_i) + \alpha \sum_n z_{i=y} + \beta \mathcal{L}_e$$

Gou, J., Yu, B., Maybank, S.J. *et al.* Knowledge Distillation: A Survey. *Int J Comput Vis* 129, 1789–1819 (2021). https://doi.org/10.1007/s11263-021-01453-z

Knowledge Distillation – Student Teacher

- Large varieties of distillation strategies have been introduced at different levels and for different architectures and tasks;
- Important: KD requires training data and the training process, thus cannot be applied as a post-hoc complexity reduction technique;

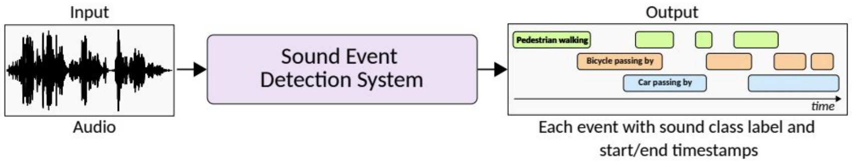
Knowledge Distillation: A Survey

Jianping Gou[1] · Baosheng Yu [1] · Stephen J. Maybank [2] · Dacheng Tao[1]

Gou, J., Yu, B., Maybank, S.J. et al. Knowledge Distillation: A Survey. Int J Comput Vis 129, 1789–1819 (2021). https://doi.org/10.1007/s11263-021-01453-z

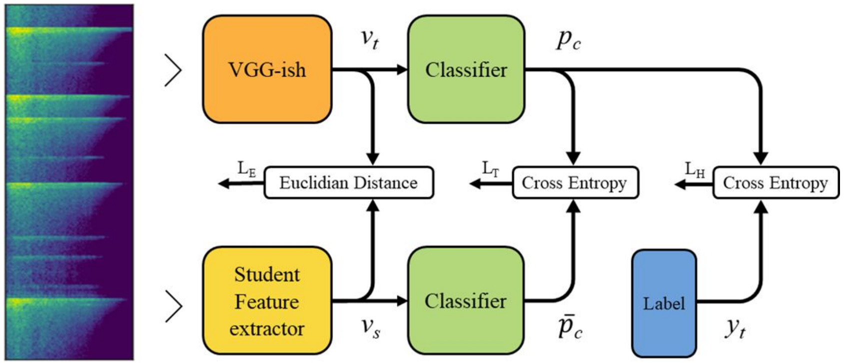
KD Example: Sound event detection

- Audio processing task, used in smart cities applications and more;
- Detects audio events from monaural audio signals in real-time;



A. Brutti, F. Paissan, A. Ancilotto and E. Farella, "Optimizing PhiNet architectures for the detection of urban sounds on low-end devices," EUSIPCO 2022
G. Cerutti, et al., "Compact recurrent neural networks for acoustic event detection on low-energy low-complexity platforms", IEEE Journal on Selected Topics on Signal Processing, 2020

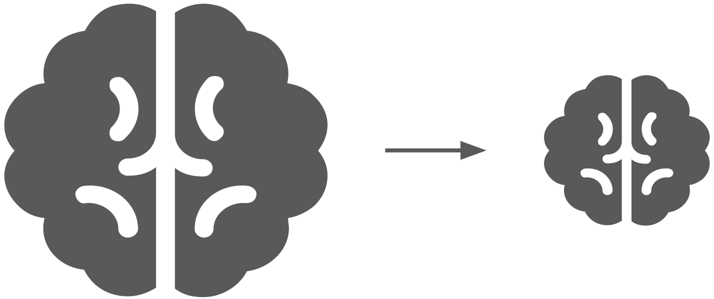
KD for sound event detection



STM32L496ZG
1MB flash
320KB RAM
100 MOPS
10 mW

	VGGish	Proposed
#Params	~72.1M	~18.0M (4x)
#Ops	~1.72G	~608M
Accuracy (US8k)	75%	70%

G. Cerutti, et al., "Compact recurrent neural networks for acoustic event detection on low-energy low-complexity platforms", IEEE Journal on Selected Topics on Signal Processing, 2020



Reducing SoTA deep learning algorithms

Distillation, pruning, (quantization)

Pruning

Learning from one-hot hard-labels is hard

We need large neural models

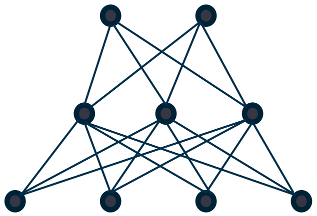
Neural models are redundant

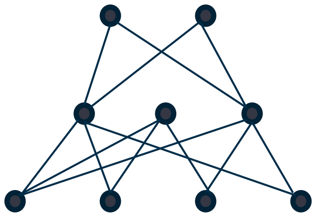
Not all weights contribute in the same way

$$y = xW + b$$

Zeroing some (the smallest) parameters

$$\tilde{y} = x\widetilde{W} + b \cong y$$

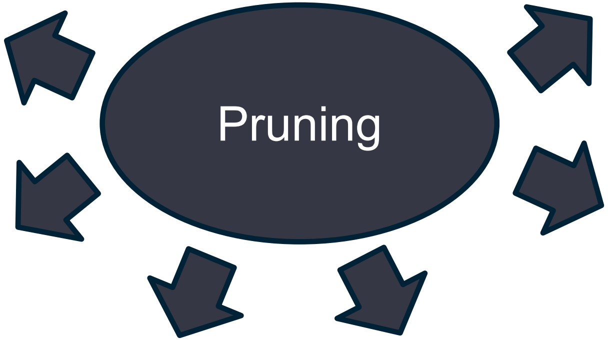




Pruning

Structured
Block of parameters (i.e.) rows of W

Soft
Zeroed weights can be restored



Unstructured
Zeroing sparse parameters

Hard
Zeroed weights cannot be restored

Post Training
Pruning applied on a trained model

Training
Pruning is performed during training

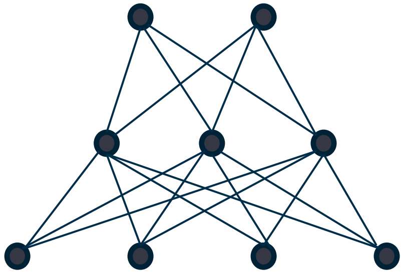
Pruning – post training

Naïve unstructured pruning
In inference

1) $$w_{ij} = 0 \; if \, w_{ij} < \theta$$

Set to 0 all parameters smaller that a given threshold θ: given the target reduction to be achieved

2) Incrementally zeroing parameters from the smallest as long as the target performance is maintained



Pruning during training

Soft/Hard pruning during training

$$\mathcal{L} = \sum c \log(p(x)) + \|W\|_0$$

Not differentiable

- The trick is how to approximate the non differentiable term
- Dynamic management of the batch size

Dynamic hard pruning of Neural Networks at the edge of the internet

Lorenzo Valerio [b], Franco Maria Nardini [a], Andrea Passarella [b], Raffaele Perego [a]

LayerDrop (for Transfomer Models)

Structured drop out

- Drop layers or group of layers during training
- In inference the model is robust to layer removal

REDUCING TRANSFORMER DEPTH ON DEMAND WITH STRUCTURED DROPOUT

Angela Fan
Facebook AI Research/LORIA
angelafan@fb.com

Edouard Grave
Facebook AI Research
egrave@fb.com

Armand Joulin
Facebook AI Research
ajoulin@fb.com

Pruning: example of LayerDrop for ASR

WavLM-Large pretrained model with 24 encoder layers

- WER on Librispeech
- Random layer drop of a model trained with LayerDrop
- Finetuning of the model after layer removal

FINE-TUNING STRATEGIES FOR FASTER INFERENCE USING SPEECH SELF-SUPERVISED MODELS: A COMPARATIVE STUDY

Salah Zaiem[⋆,∓] *Robin Algayres*[◇] *Titouan Parcollet*[†] *Slim Essid*[⋆] *Mirco Ravanelli*[∓]

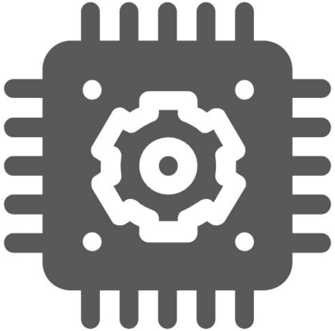
⋆ Telecom Paris, Palaiseau, France, ◇ ENS, INRIA, INSERM, UPEC, PSL Research University, France
† Samsung AI Center, Cambridge, United-Kingdom,
∓ Mila-Quebec AI Institute, Université de Montréal, Concordia University, Canada

Technique		WER ↓	GPU (s)	CPU (s)	WER-LM ↓	GPU-LM (s)	CPU-LM (s)	MACs (G)
Baseline	**Full Model**	4.09	134	1121	3.31	152	1128	386.53
Layer Drop	**Drop Prob**							
	0.5	11.28	96	721	5.89	156	776	244.19
Structured soft layer pruning	0.4	8.32	102	816	4.58	145	844	272.28
	0.3	6.56	109	888	3.84	157	913	300.98
	0.25	5.91	113	932	3.72	148	950	314.24
Layer Removal	**Num. Kept Layers**							
	12	14.39	93	726	8.64	127	739	236.64
Structured hard layer pruning	16	8.16	109	852	5.53	131	861	286.60
	20	5.14	117	988	3.62	142	989	336.57

Pruning: final remarks

- Zeroing parameters **does not reduce** memory footprint or flops: zeros have to be stored and processed
- Only the **actual implementation** of the model skipping the zeroed connections will reduce the memory footprint and the flops
- Unstructured pruning sparsifies the matrices
  This could allow applying efficient methods for **sparse matrices**

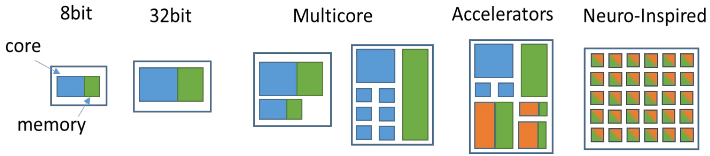
A. Ancilotto, F. Paissan and E. Farella "XiNet-pose: Extremely lightweight pose detection for microcontrollers”



Build ad-hoc algorithms for MCUs and inference engines

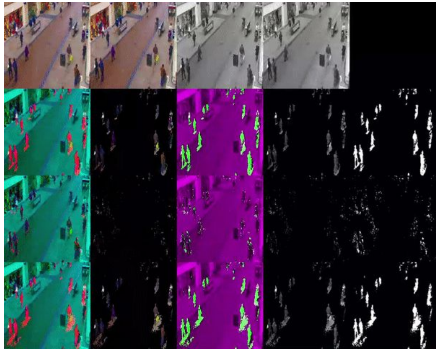
Building platform-, application-specific algorithms

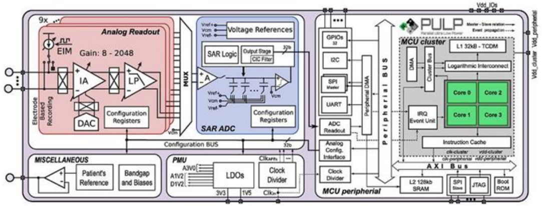
- Requires a different skill set:
  - knowledge of **bare-metal programming**;
  - Classical machine learning, computer vision and signal processing techniques;
- Usually, these solutions **don't generalize** to new application domains and hardware;
- Exploit specialized computing platforms, smart vision sensors to pre-process the images and reduce redundancy in the data, thus the amount of processing time.

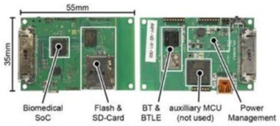


PMIC and readout IC in sensor hub of the SimBand

Two variants of the sensing module in the SimBand

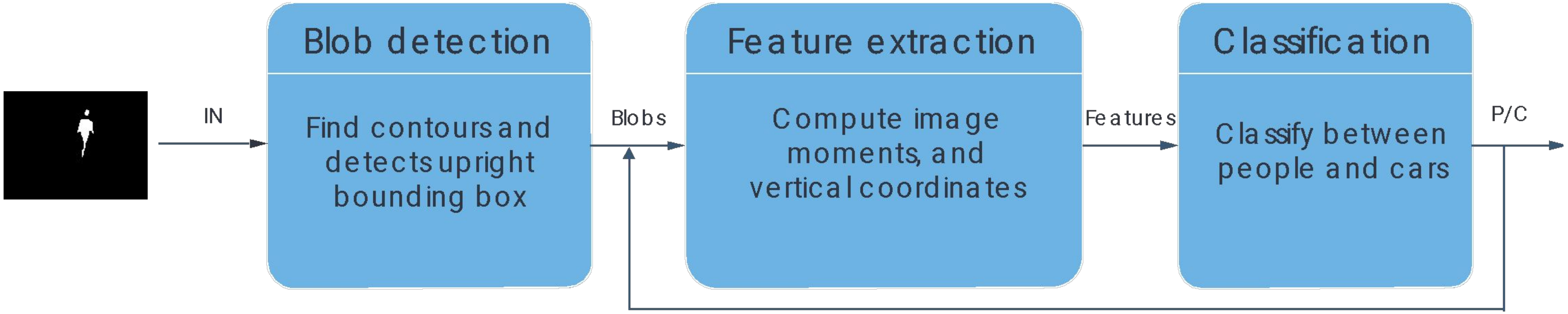






A. Ancilotto, F. Paissan, E. Farella "On the Role of Smart Vision Sensors in Energy-Efficient Computer Vision at the Edge", Accepted at IEEE PerconAI Workshop 2022

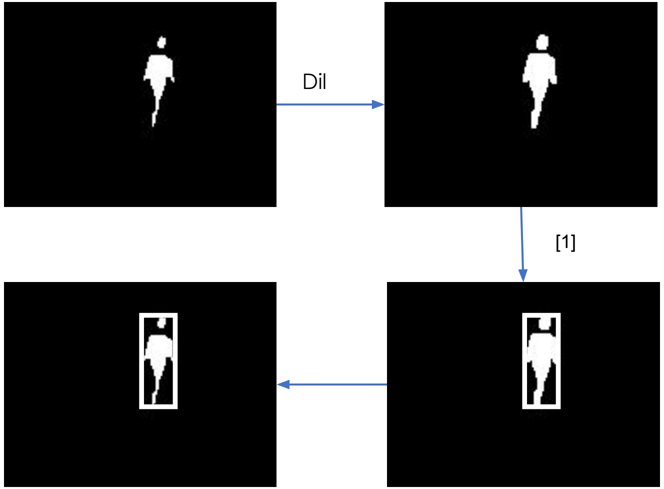
People/car classification using low-power SVS



F. Paissan, G. Cerutti, M. Gottardi, E. Farella "People/car classification using an ultra-low-power smart vision sensor", IEEE IWASI 2019

Blob detection

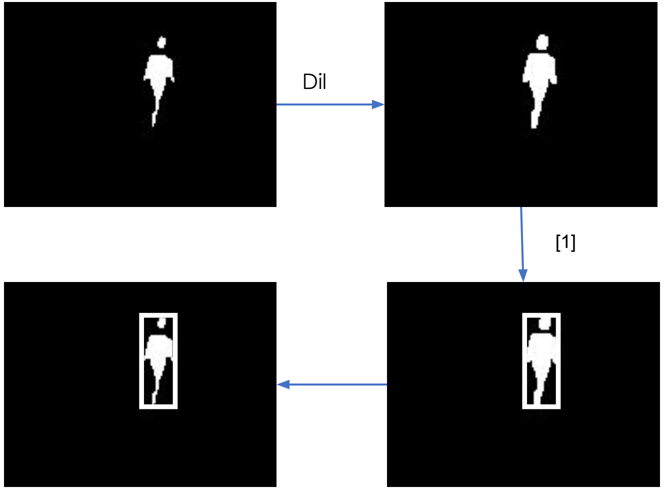
1. Border following algorithm [1]
2. Detect contours with hierarchical structure;
3. Creates bounding boxes;
4. For each blob:
    - Compute statistical features of the bounding box (area, image moments);
5. Classify bounding boxes with an SVM;



[1] Suzuki, et al. Topological Structural Analysis of Digitized Binary Images by Border Following. CVGIP 30 1, pp 32-46 (1985)

Blob detection

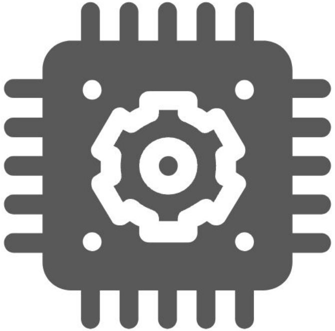
1. Border following algorithm [1]
2. Detect contours with hierarchical structure;
3. Creates bounding boxes;
4. For each blob:
    - Compute statistical features of the bounding box (area, image moments);
5. Classify bounding boxes with an SVM;



Sub-mW Keyword Spotting on an MCU: Analog Binary Feature Extraction and Binary Neural Networks

Gianmarco Cerutti, Lukas Cavigelli, Renzo Andri, Michele Magno, Elisabetta Farella, Luca Benini

[1] Suzuki, et al. Topological Structural Analysis of Digitized Binary Images by Border Following. CVGIP 30 1, pp 32-46 (1985)



Build ad-hoc algorithms for MCUs and <u>inference engines</u>

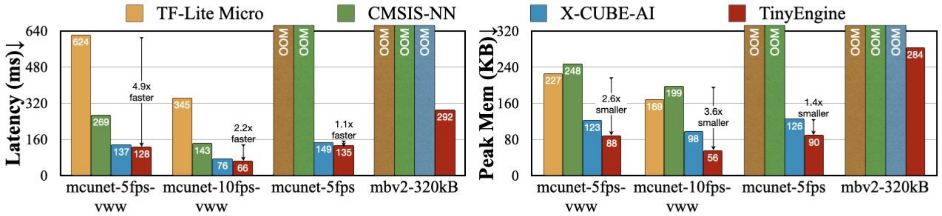
Custom inference engines

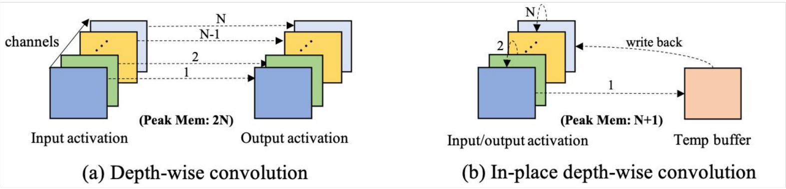
- Specialized solutions for specific hardware platforms;
- Outperforms common toolchains for running NNs on MCUs;

- Supports fewer operations;
- Harder to run with custom neural networks;

MCUNet: Tiny Deep Learning on IoT Devices

Ji Lin[1] Wei-Ming Chen[1,2] Yujun Lin[1] John Cohn[3] Chuang Gan[3] Song Han[1]
[1]MIT [2]National Taiwan University [3]MIT-IBM Watson AI Lab
https://tinyml.mit.edu





J. Lin et al. "MCUNet: Tiny Deep Learning on IoT Devices", NeurIPS 2020

Toward tinyML

Reducing SoTA

- Generally more "elastic": can be adapted to different application domains (audio, video, multimodal);
- Achieves good performance by exploiting a bigger network;

- Generally comes with a computational overhead given by the deployment toolchains;
- Is not always an option and is not guaranteed to work;

Ad-hoc algorithms

- Can be more computationally efficient, as the are super specific on a single application and device;

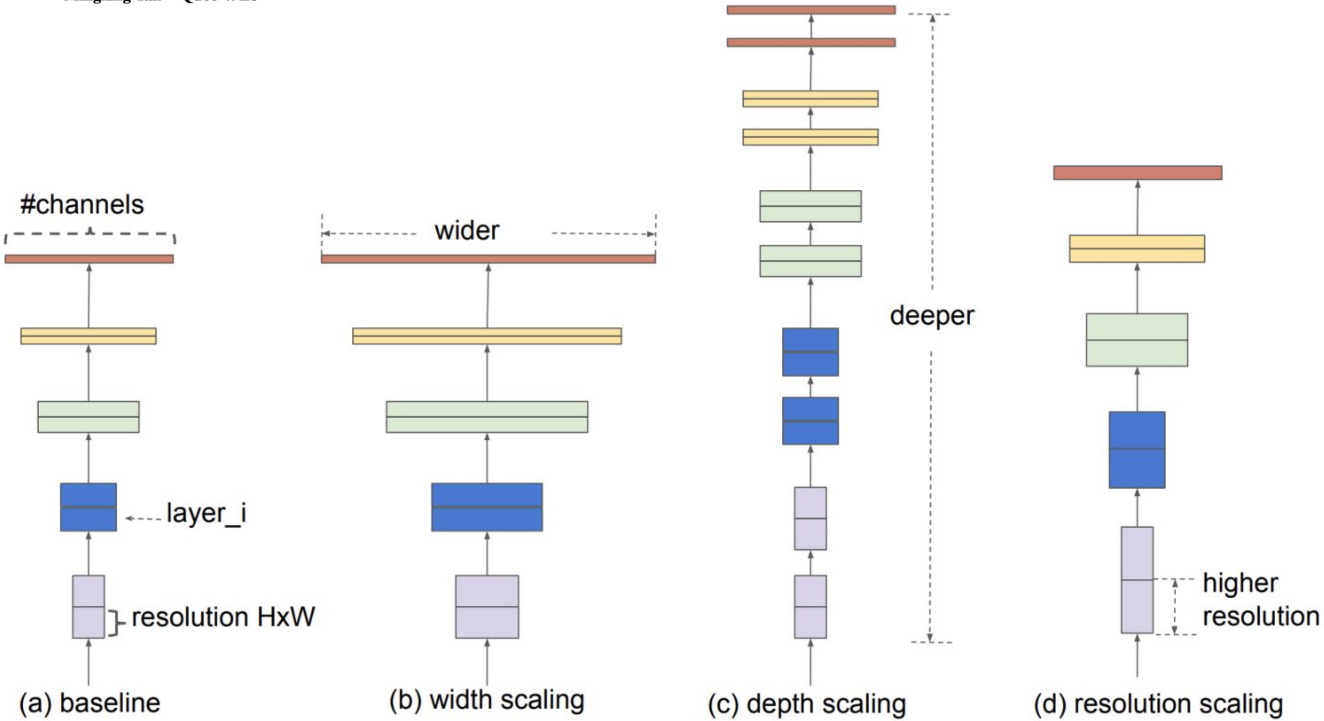
- It is too device-dependent and application-specific;
- Comes with all the limitations of classical ML algorithms;

Putting it all together...

Specialize solutions for each device ✅
Scales to different tasks ✅
Runs on low resources ✅

EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks

Mingxing Tan [1] Quoc V. Le [1]



Tan, Mingxing, and Quoc Le. "Efficientnet: Rethinking model scaling for convolutional neural networks." International conference on machine learning. PMLR, 2019.

PhiNets

What?

- Backbone family that exploit HAS paradigm;
- Based on slightly modified inverted residual blocks;

Goal and Target

- Designed and optimized for multimedia analytics at the edge;
- Has advanced scalability principles to comply with hardware varying constraints;

PhiNets: a scalable backbone for low-power AI at the edge

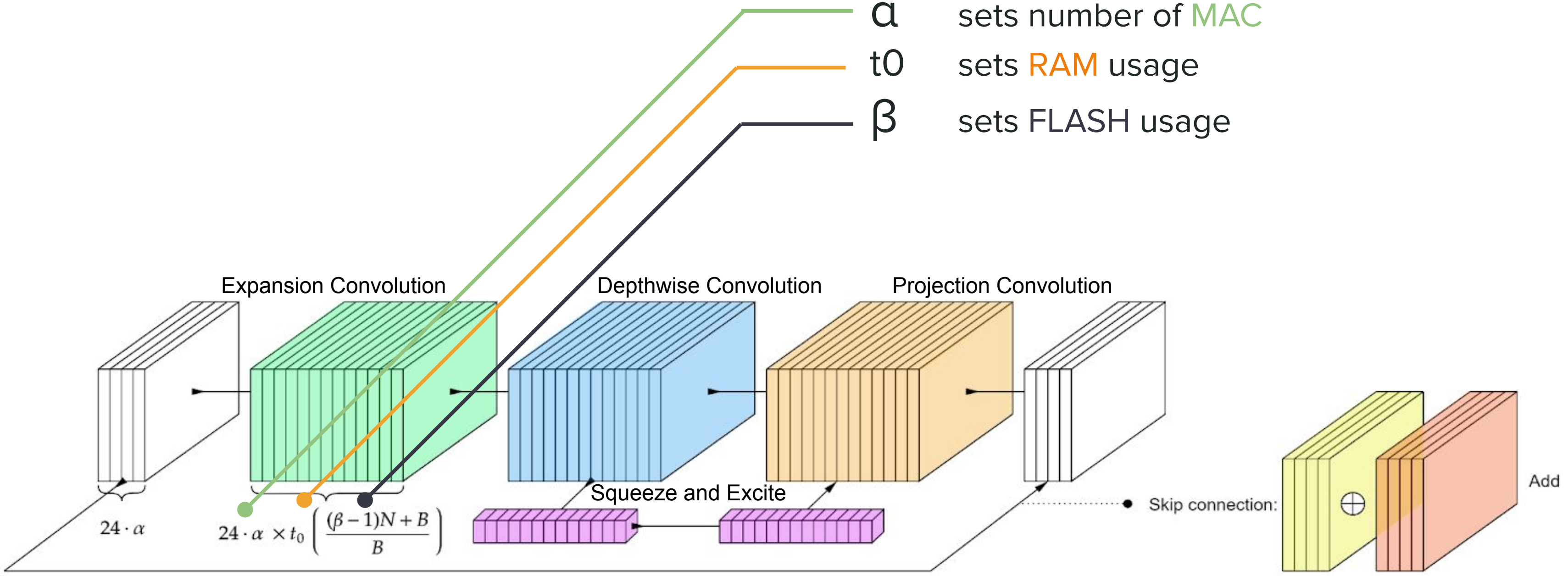
FRANCESCO PAISSAN*, ALBERTO ANCILOTTO*, and ELISABETTA FARELLA, E3DA Unit, Digital Society Center - Fondazione Bruno Kessler (FBK)

Paissan, Francesco, Alberto Ancilotto, and Elisabetta Farella. "PhiNets: a scalable backbone for low-power AI at the edge." ACM TECS

PhiNets convolutional block

Three parameters:

α	sets number of MAC
t0	sets RAM usage
β	sets FLASH usage



Hardware aware scaling

$$Params = \sum_{B=1}^{N} [2(\alpha^2 \cdot t_B \cdot C_B^2) + (\alpha \cdot t_B \cdot C_B \cdot K^2)]$$

$$MAC = \sum_{B=1}^{N} (W_B \cdot H_B)[2(\alpha^2 \cdot t_B \cdot C_B^2) + (\alpha \cdot t_B \cdot C_B \cdot K^2)]$$

$$RAM = H_0 \times W_0 \times (t_0 + \alpha \cdot t_0 + C_0)$$

1.

Params = f (α, β, t0)
RAM = g (α, t0)
MAC = h (α, β, t0)

2.

α = f (Params, RAM, MAC)
β = g (Params, RAM, MAC)
t0 = h (Params, RAM, MAC)

3.

Solve for
(Params, RAM, MAC)
of target hardware

Hardware aware scaling

- One-shot network scaling that enables to run the best performing network on a **target platform**;
- Derive resources needed from **network parameters**;
- **Invert equations** to find network hyperparameters;

MCU specs

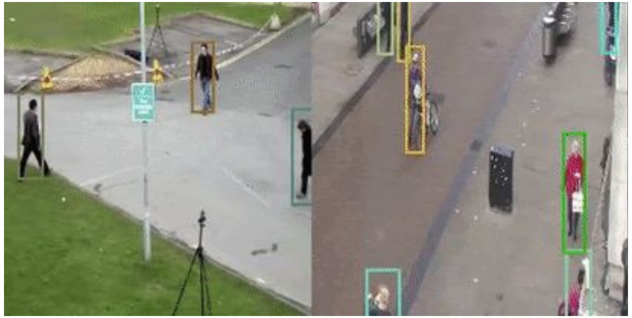
- **Flash:** 1 MB
- **RAM:** 392 KB
- **MACC:** 63 M

- **α:** 1.0
- **β:** 0.66
- **t0:** 4.0

NET parameters

Other examples of MCU applications...

Object detection and tracking



Running on:

- < 2MB of FLASH
- < 1MB of RAM
- power requirements in the order of 10 mW

Video anonymization

Original

Anonymized face

Running on:

- 984 K parameters
- 62 M MACC
- 392 KB RAM
- 28 mJ / frame on K210 @ 9fps

And many more...

Hands-on 1: Designing and training NNs

Hands-on 2: Converting and deploying NNs