An introduction to tinyML

Francesco Paissan

Energy Efficient Embedded Digital Architectures
Fondazione Bruno Kessler

fpaissan@fbk.eu

February 20, 2024

Presentation Overview

Outline

The Five (-1) Ws of tinyML

What?

- a fast-growing subfield of machine learning targeting **on-device** and **near-sensor processing**;

The Five (-1) Ws of tinyML

What?

- a fast-growing subfield of machine learning targeting **on-device** and **near-sensor processing**;

Why?

- many practical **benefits** (e.g. bandwidth reduction, infrastructure sustainability, scalability);
- **privacy** by design: enable processing on-device, thus sensitive data is never leaked;

The Five (-1) Ws of tinyML

What?

- a fast-growing subfield of machine learning targeting **on-device** and **near-sensor processing**;

Why?

- many practical **benefits** (e.g. bandwidth reduction, infrastructure sustainability, scalability);
- **privacy** by design: enable processing on-device, thus sensitive data is never leaked;

When?

- not clear, it was a continuous process, sometimes driven by necessity...

Who?

(tiny)AI researchers:

- come up with novel ML algorithms to compress and simplify NN model;
- generally approach tinyML as a ML problem;

Who?

(tiny)AI researchers:

- come up with novel ML algorithms to compress and simplify NN model;
- generally approach tinyML as a ML problem;

(AI)Embedded engineers:

- design custom NN accelerator and neuromorphic processors to speed up NN inference;
- approach tinyML as an engineering problem;

Who?

(tiny)AI researchers:

- come up with novel ML algorithms to compress and simplify NN model;
- generally approach tinyML as a ML problem;

(AI)Embedded engineers:

- design custom NN accelerator and neuromorphic processors to speed up NN inference;
- approach tinyML as an engineering problem;

But there's stuff also in the gray area...

Challenges of tinyML?

WORKSTATION

RAM: 10-100 GB
Storage: 10s of TB
Speed: 100 Billions of ops/s

PC/SBC

RAM: 1-10 GB
Storage: 10-100 GB
Speed: 1-10 Billions of ops/s

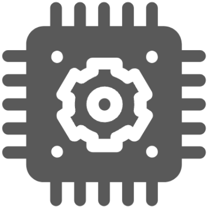
MCU

RAM: 10s - 100s of KBs
Storage: KBs - MBs
Speed: Millions of ops/s

÷10

÷10 000

Target platforms



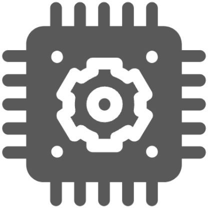
microcontrollers, SBC,

neuromorphic processors, ...

Target platforms

small parameter memory available

(kB - MB)



microcontrollers, SBC,

neuromorphic processors, ...

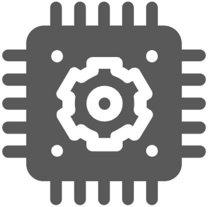
Target platforms

small parameter memory available

(kB - MB)

few operations per second

(million ops/s)



microcontrollers, SBC,

neuromorphic processors, ...

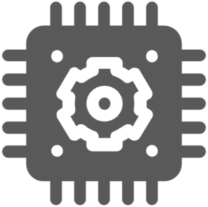
Target platforms

small parameter memory available

(kB - MB)

few operations per second

(million ops/s)



microcontrollers, SBC,

neuromorphic processors, ...

small working memory

(kB - MB)

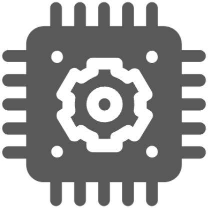
Target platforms

small parameter memory available

(kB - MB)

few operations per second

(million ops/s)



microcontrollers, SBC,

neuromorphic processors, ...

small working memory

(kB - MB)

limited operations support

(generally optimized for CNNs)

Outline

Building an intuition on NN design

It (almost) always boils down to...

$$y = \mathcal{F}(X) = (f_N \circ f_{N-1} \circ ... \circ f_1)(X)$$

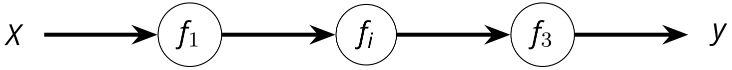
where $\mathcal{F} : \mathcal{D} \rightarrow \mathcal{Y}$, with $\mathcal{D}$, and $\mathcal{Y}$ are the input and output domains.

Building an intuition on NN design

It (almost) always boils down to...

$$y = \mathcal{F}(X) = (f_N \circ f_{N-1} \circ ... \circ f_1)(X)$$

where $\mathcal{F} : \mathcal{D} \rightarrow \mathcal{Y}$, with $\mathcal{D}$, and $\mathcal{Y}$ are the input and output domains.



Just a few examples...

- Linear: $f_i(z) = \mathbf{W}\mathbf{z} + \mathbf{b}$
- Convolution: $f_i(z) = \mathbf{z} * \mathbf{W} + \mathbf{b}$
- BatchNorm: $f_i(z) = \frac{\mathbf{z} - \mathrm{E}(\mathbf{z})}{\sqrt{\mathrm{Var}(\mathbf{z}) + \epsilon}}\alpha + \beta$
- Many other primitives and tweaks to improve performance...

Putting it in context

Let's create a use-case to get an intuition on different primitives.

Putting it in context

Let's create a use-case to get an intuition on different primitives.

$$X \longrightarrow f_1 \longrightarrow f_i \longrightarrow f_3 \longrightarrow y$$

and let's assume $X \in \mathbb{R}^{3\times64\times64}$ and $f_1(X) \in \mathbb{R}^{16\times64\times64}$.

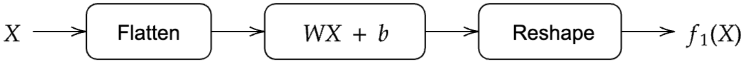
Putting it in context

Let's create a use-case to get an intuition on different primitives.

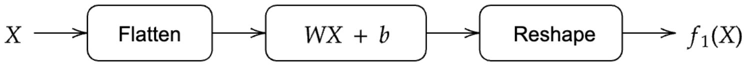
$$X \longrightarrow f_1 \longrightarrow f_i \longrightarrow f_3 \longrightarrow y$$

and let's assume $X \in \mathbb{R}^{3\times64\times64}$ and $f_1(X) \in \mathbb{R}^{16\times64\times64}$.
Let's see what happens...

A very bad idea



A very bad idea



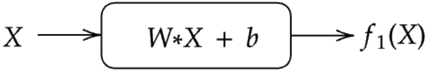
- Params: 805M
- Multiply-adds: 805M
- Heavily affected by shifts and rotations in the input space;
- Underperforms on real-life benchmarks;

A very bad idea

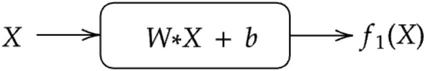
$$X \rightarrow \boxed{\text{Flatten}} \rightarrow \boxed{WX + b} \rightarrow \boxed{\text{Reshape}} \rightarrow f_1(X)$$

- Params: 805M
- Multiply-adds: 805M
- Heavily affected by shifts and rotations in the input space;
- Underperforms on real-life benchmarks;

Convolutions



Convolutions



- Params: 480
- Multiply-adds: 1.84M

$$Params = c_{in}c_{out}k^2 + c_{out} \;\;|\;\; MAC = HW(c_{in}c_{out}k^2 + c_{out})$$

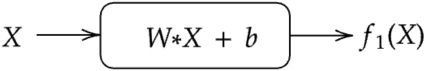
Pointwise Convolutions

$$X \longrightarrow \boxed{W * X + b} \longrightarrow f_1(X)$$

- Params: 96
- Multiply-adds: 0.28M

$$Params = c_{in}c_{out} + c_{out} \quad | \quad MAC = HW(c_{in}c_{out} + c_{out})$$

Depthwise Convolutions



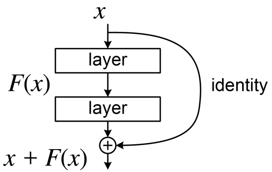
- Params: 180
- Multiply-adds: 0.61M

$$Params = c_{in}k^2 + c_{out} \;\; | \;\; MAC = HW(c_{in}k^2 + c_{out})$$

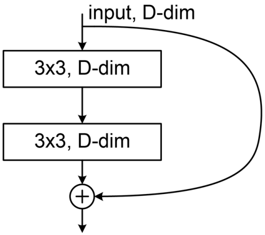
Outline

Residual Block

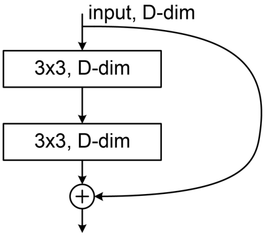
- creates a direct connection between input and output of the convolutional block;
- residual blocks follow a wide/narrow/wide structure in the number of channels;
- improves the performance by enabling deeper networks via **skip connections** Wightman, Touvron, and J'egou, "ResNet strikes back: An improved training procedure in timm";



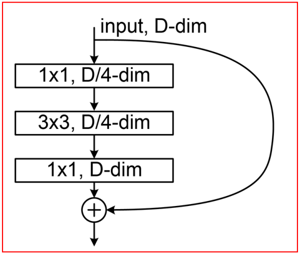
ResBlock variants



ResBlock variants



Wide-narrow-wide channel structure



Inverted Residual Block

- differently from a ResBlock, this uses a narrow/wide/narrow structure in the number of channels;

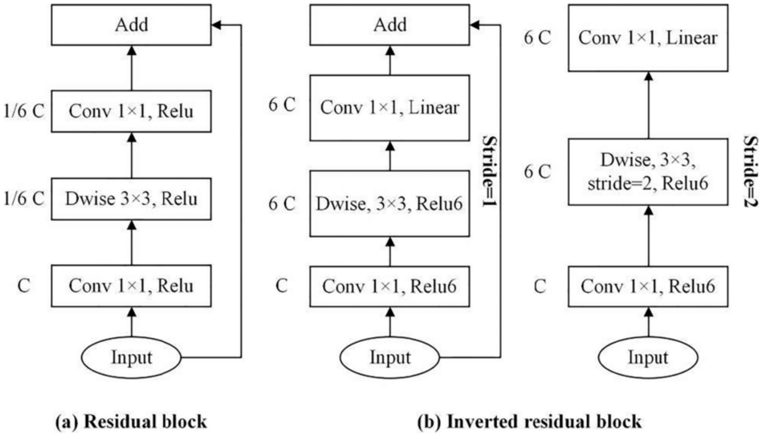
Inverted Residual Block

- differently from a ResBlock, this uses a narrow/wide/narrow structure in the number of channels;
- additionally, groups are used inside the convolutions to reduce the computational complexity (depthwise convolutions);

Inverted Residual Block

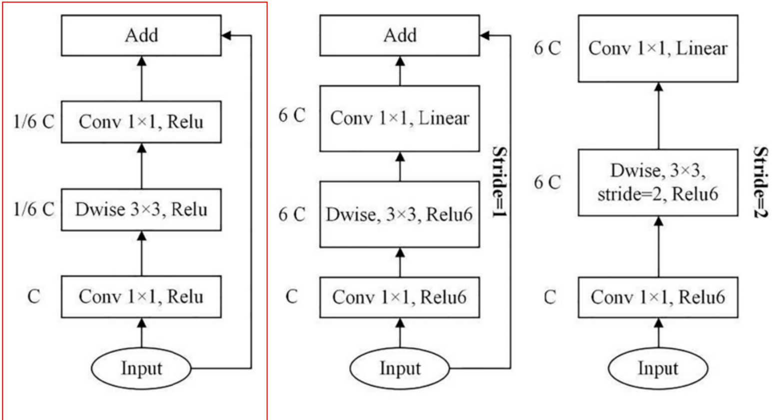
- differently from a ResBlock, this uses a narrow/wide/narrow structure in the number of channels;
- additionally, groups are used inside the convolutions to reduce the computational complexity (depthwise convolutions);
- generally used in parameter-efficient networks;

Inverted Convolutional Block



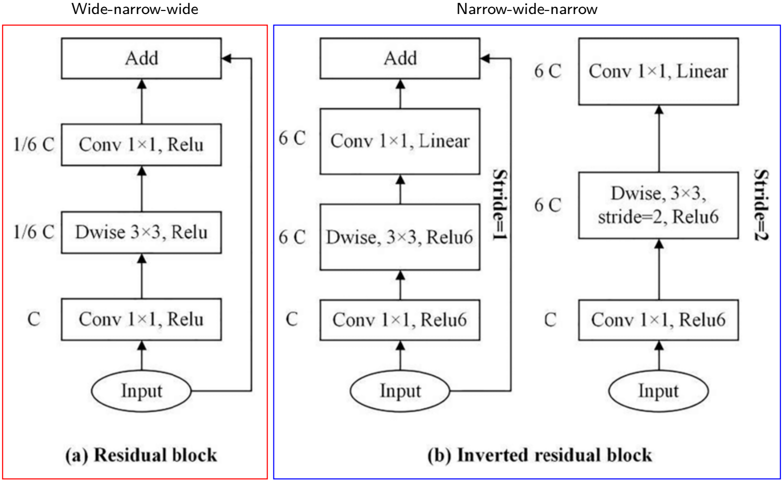
Inverted Convolutional Block

Wide-narrow-wide



(a) Residual block **(b) Inverted residual block**

Inverted Convolutional Block

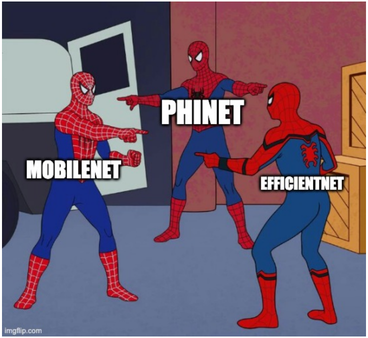


Where are these blocks used?

- Residual blocks: ResNet, UNets, and many more...
- Inverted residual blocks: MobileNet, EfficientNet, PhiNets, MCUNet, and many more...

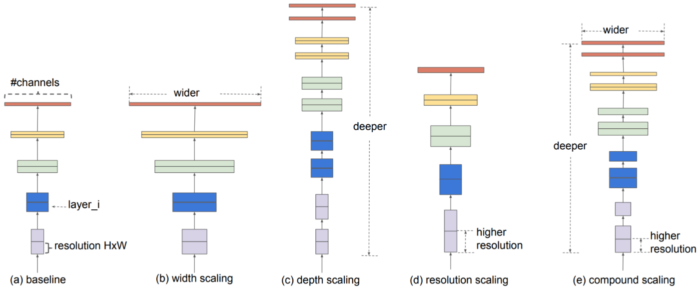
Where are these blocks used?

- Residual blocks: ResNet, UNets, and many more...
- Inverted residual blocks: MobileNet, EfficientNet, PhiNets, MCUNet, and many more...



Compound scaling

- focuses on how we ‘should’ be scaling CNNs to obtain optimal performanceTan and Le, “EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks”;
- introduces the concept of compound scaling (i.e. scaling all dimensions is better than one dimension at a time);



Shortcomings of mainstream CNNs

- **too demanding** to run on edge devices and/or compromise performance too much trying to fit;

Shortcomings of mainstream CNNs

- **too demanding** to run on edge devices and/or compromise performance too much trying to fit;
- edge devices have different capabilities some convolutional blocks **cannot exploit**;

Shortcomings of mainstream CNNs

- **too demanding** to run on edge devices and/or compromise performance too much trying to fit;
- edge devices have different capabilities some convolutional blocks **cannot exploit**;
- compound scaling changes all the computational complexities in a **coupled** way;

Ideal CNN for edge processing

- a neural network that can **scale to low computational complexity** (≤ 1 MB of FLASH, ≤ 1 MB of RAM);
- a convolutional block that is designed to **exploit the available resources** maximally;
- a scaling strategy that allows fitting neural networks on **different edge platforms** based on the applications scenarios;

PhiNets

- based on **inverted residual blocks**, modified to decouple the computational resources;

Paissan, Ancilotto, and Farella, "PhiNets: A Scalable Backbone for Low-power AI at the Edge".

PhiNets

- based on **inverted residual blocks**, modified to decouple the computational resources;
- designed and optimized for **multimedia analytics** at the edge (audio-video);

Paissan, Ancilotto, and Farella, "PhiNets: A Scalable Backbone for Low-power AI at the Edge".

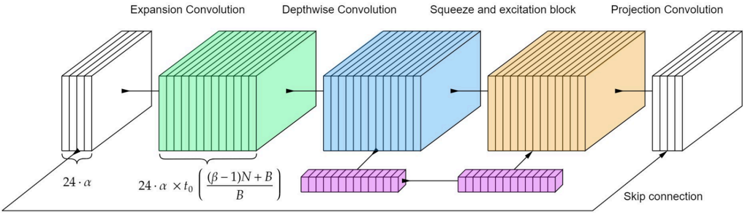
PhiNets

- based on **inverted residual blocks**, modified to decouple the computational resources;
- designed and optimized for **multimedia analytics** at the edge (audio-video);
- controls RAM (t_0), FLASH (β) and operations (α) using three hyperparamters;

Paissan, Ancilotto, and Farella, "PhiNets: A Scalable Backbone for Low-power AI at the Edge".

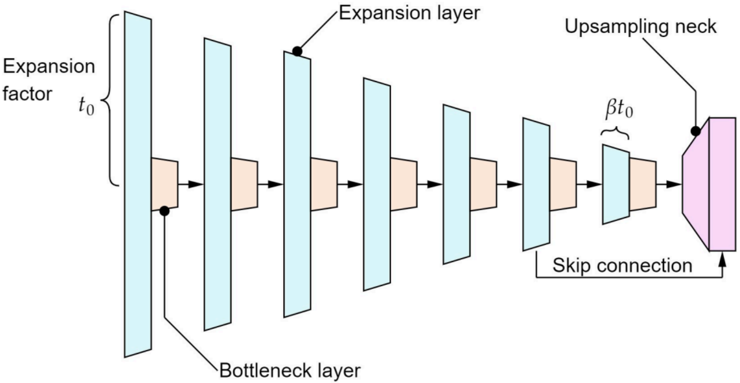
PhiNets convolutional block

Narrow-wide-narrow structure for the number of channels...



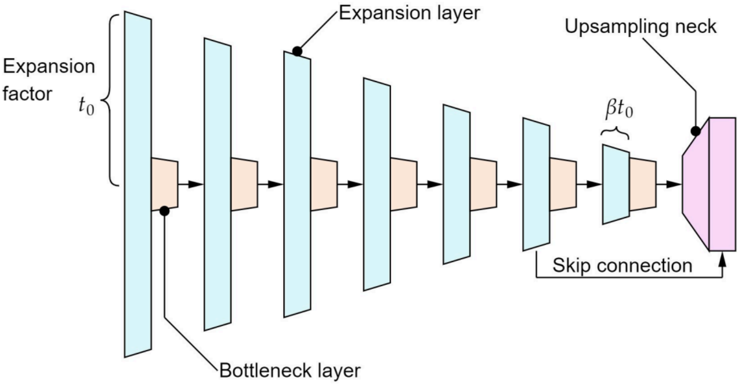
Paissan, Ancilotto, and Farella, "PhiNets: A Scalable Backbone for Low-power AI at the Edge".

The sequence of PhiNets conv blocks



Paissan, Ancilotto, and Farella, "PhiNets: A Scalable Backbone for Low-power AI at the Edge".

The sequence of PhiNets conv blocks




```
from micromind.networks import PhiNet
```

Paissan, Ancilotto, and Farella, "PhiNets: A Scalable Backbone for Low-power AI at the Edge".

Designing an optimized convolutional block

- PhiNets are designed based on **indirect efficiency metrics**, thus could be an ideal version of edge CNNs;

Designing an optimized convolutional block

- PhiNets are designed based on **indirect efficiency metrics**, thus could be an ideal version of edge CNNs;
- what happens if we try to break free of the common standards for convolutional block design and investigate from first principles?

Designing an optimized convolutional block

- PhiNets are designed based on **indirect efficiency metrics**, thus could be an ideal version of edge CNNs;
- what happens if we try to break free of the common standards for convolutional block design and investigate from first principles?

Let's see...

Formal definition of efficiency

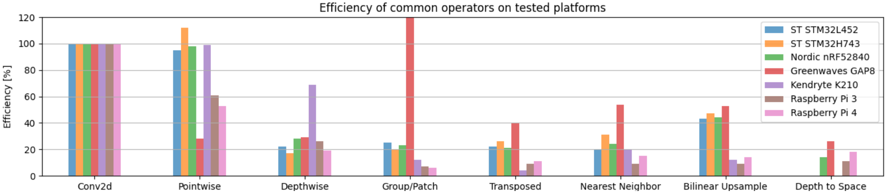
Definition 3.1

We assessed the actual efficiency of each operator (η_{op}) by calculating the ratio between the energy needed for a standard convolution (E_S) and the energy of the chosen operator (E_{op}) to perform an equivalent number of MACs.

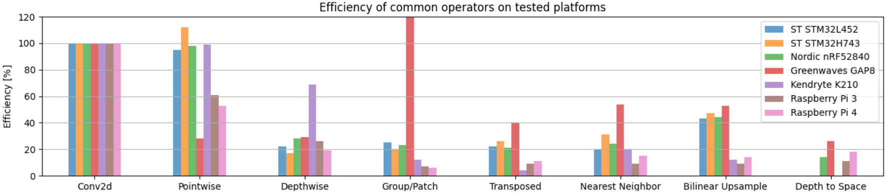
$$\eta_{op} = \frac{E_S}{E_{op}}$$

Ancilotto, Paissan, and Farella, "XiNet: Efficient Neural Networks for tinyML".

Empirical evaluation of CNN operators...

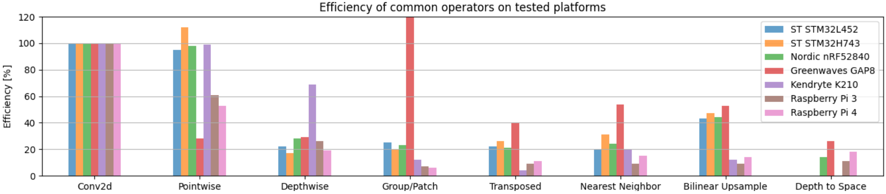


Empirical evaluation of CNN operators...



Ancilotto, Paissan, and Farella, "XiNet: Efficient Neural Networks for tinyML".

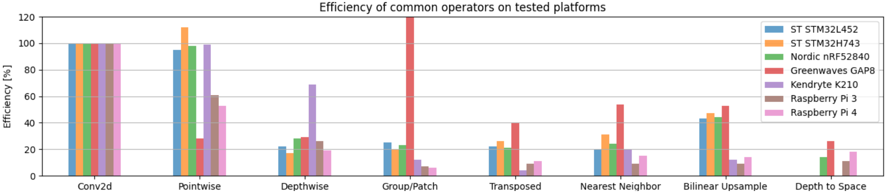
Empirical evaluation of CNN operators...



- this suggests that standard convolutions (AlexNet-style) are, on average, more efficient than other variants;

Ancilotto, Paissan, and Farella, "XiNet: Efficient Neural Networks for tinyML".

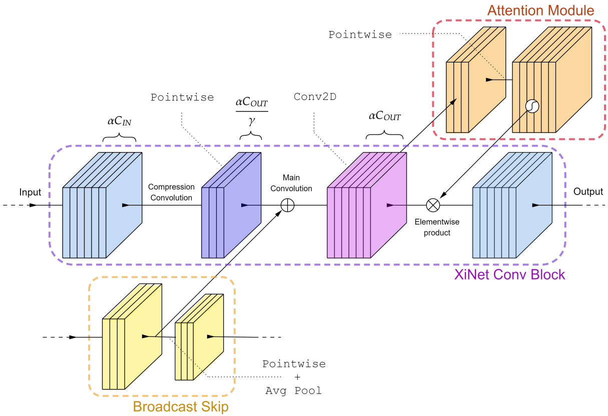
Empirical evaluation of CNN operators...



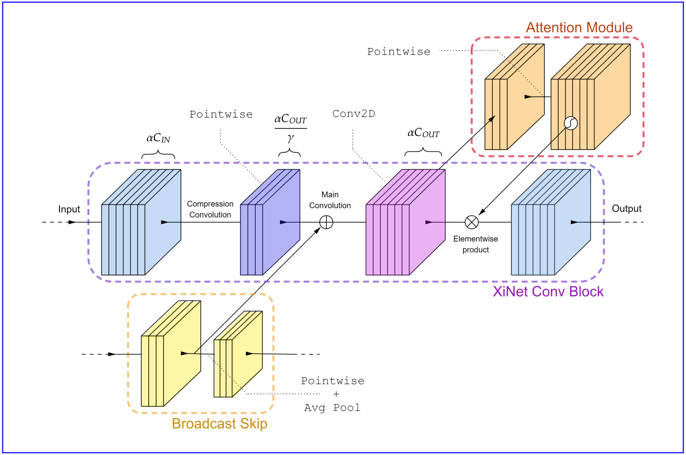
- this suggests that standard convolutions (AlexNet-style) are, on average, more efficient than other variants;
- but how do we exploit them with low parameter memory?

Ancilotto, Paissan, and Farella, "XiNet: Efficient Neural Networks for tinyML".

XiNet convolutional block

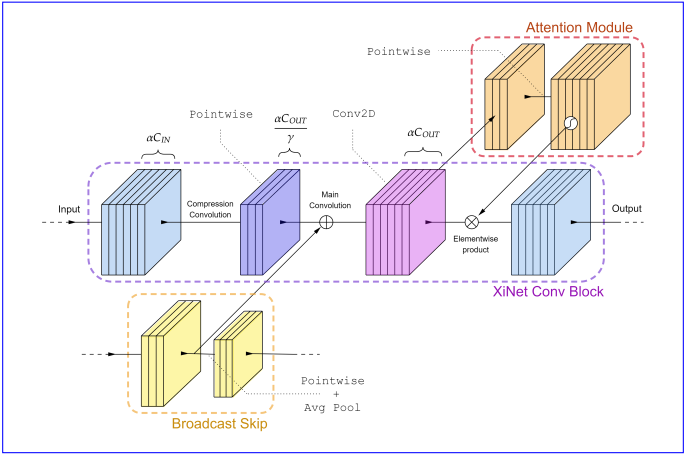


XiNet convolutional block



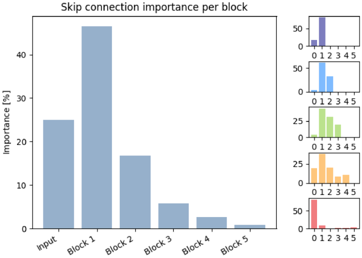
XiNet convolutional block

Wide-narrow-wide structure for channels, and much more...



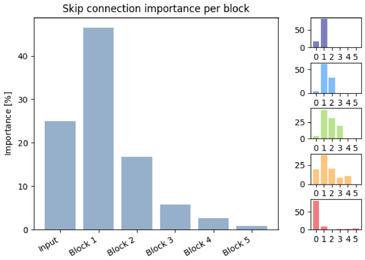
Ancilotto, Paissan, and Farella, "XiNet: Efficient Neural Networks for tinyML".

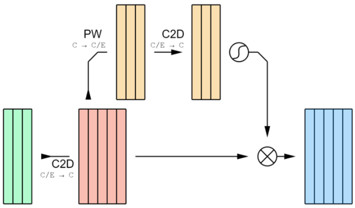
Skip connections and attention block



Ancilotto, Paissan, and Farella, "XiNet: Efficient Neural Networks for tinyML".

Skip connections and attention block





Ancilotto, Paissan, and Farella, "XiNet: Efficient Neural Networks for tinyML".

XiNet

- composed by a sequence of XiNet convolutional blocks;

Ancilotto, Paissan, and Farella, "XiNet: Efficient Neural Networks for tinyML".

XiNet

- composed by a sequence of XiNet convolutional blocks;
- similarly to PhiNets, its computational complexity is controlled using **three hyperparameters** (α, γ, β);

Ancilotto, Paissan, and Farella, "XiNet: Efficient Neural Networks for tinyML".

XiNet

- composed by a sequence of XiNet convolutional blocks;
- similarly to PhiNets, its computational complexity is controlled using **three hyperparameters** (α, γ, β);
- designed based on the **empirical benchmark** of the different operators to be very efficient;

Ancilotto, Paissan, and Farella, "XiNet: Efficient Neural Networks for tinyML".

XiNet

- composed by a sequence of XiNet convolutional blocks;
- similarly to PhiNets, its computational complexity is controlled using **three hyperparameters** (α, γ, β);
- designed based on the **empirical benchmark** of the different operators to be very efficient;

```
from micromind.networks import XiNet
```

Ancilotto, Paissan, and Farella, "XiNet: Efficient Neural Networks for tinyML".

Hardware-aware scaling

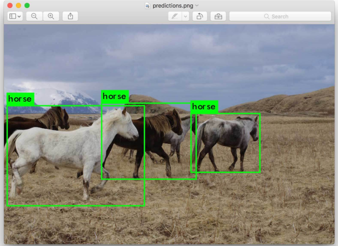
- **scaling strategy** that exploits the advanced PhiNets and XiNet architectures;
- helps deploy CNNs on a wide variety of edge platforms via its one-shot network optimization procedure;
- **inverts the mapping between computational complexity and hyperparameters** so that it can be solved with a mathematical programming toolkit for specific computational requirements;

Paissan, Ancilotto, and Farella, "PhiNets: A Scalable Backbone for Low-power AI at the Edge".

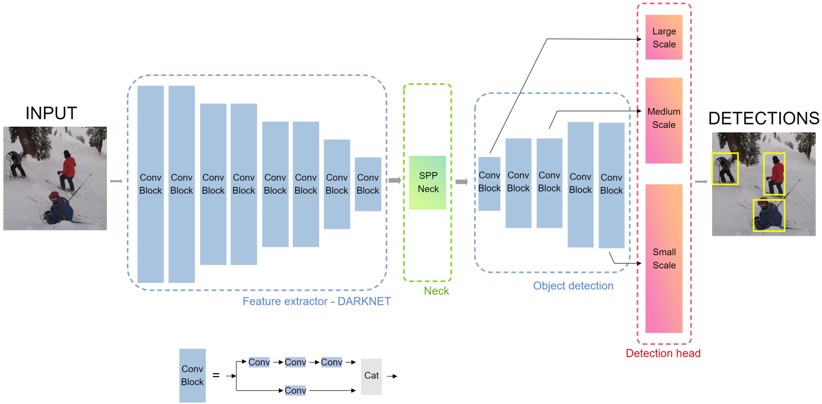
Outline

You Only Look Once (YOLO)

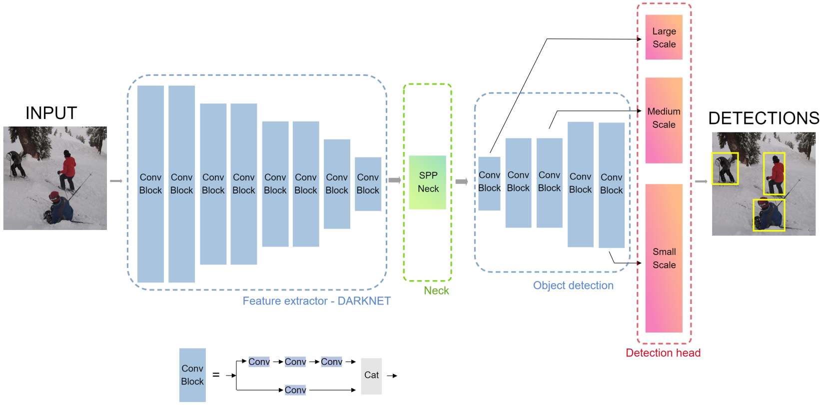
- originally proposed as an object detection pipeline;
- well known for its **good performance/complexity tradeoff**;
- mainly related to its ability to detect objects using **only one inference step** (no region proposal networks, etc...);
- recently extended to support image segmentation, keypoint detection/pose estimation;



YOLO Architecture

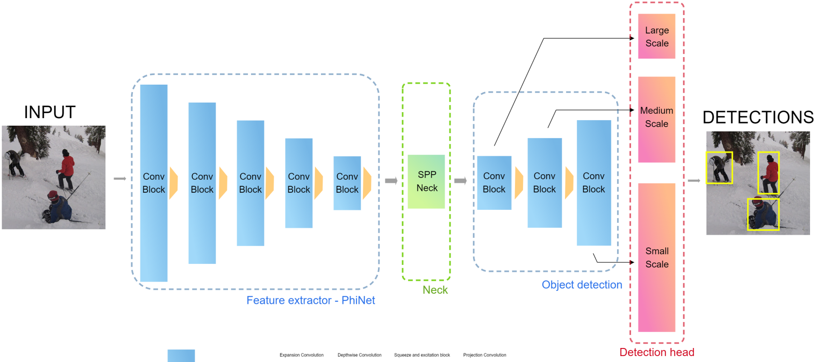


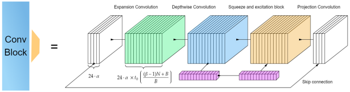
YOLO Architecture



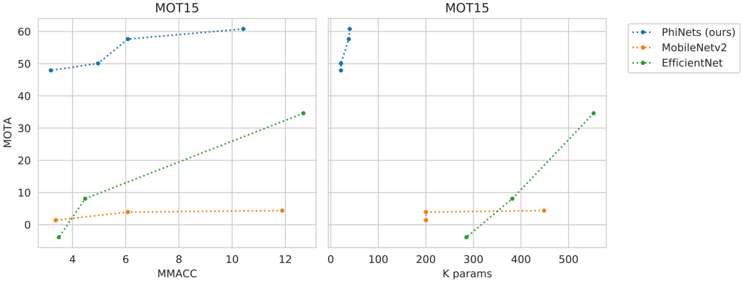
In the literature, some works propose to solve a simplified version of the object detection task; thus, reducing computational complexity... but here is what we do:

YOLOPhiNet


INPUT
Conv Block
Conv Block
Conv Block
Conv Block
Conv Block
Feature extractor - PhiNet
SPP Neck
Neck
Conv Block
Conv Block
Conv Block
Object detection
Large Scale
Medium Scale
Small Scale
Detection head
DETECTIONS


Conv Block
=
Expansion Convolution
Depthwise Convolution
Squeeze and excitation block
Projection Convolution
Skip connection

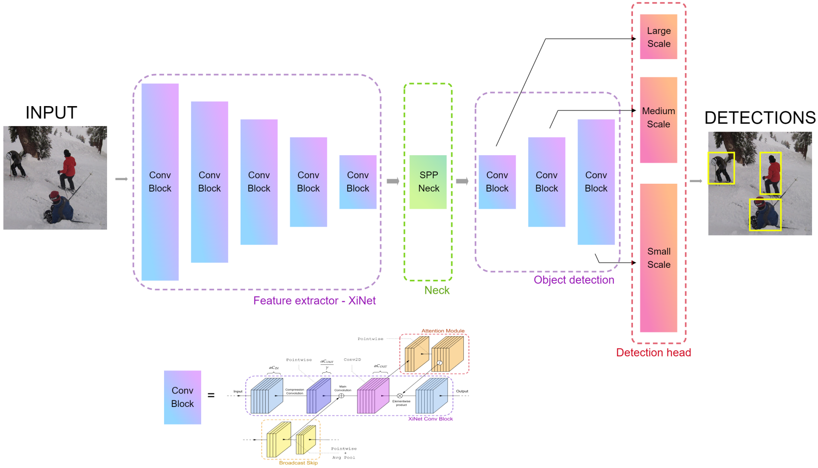
YOLOPhinet



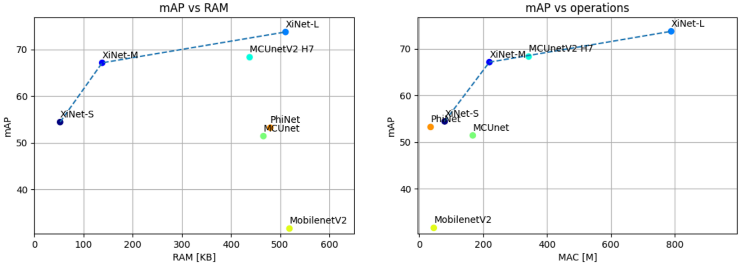
Deployed on an Arm-Cortex M7 MCU with $2\,\mathrm{MB}$ of internal Flash and $1\,\mathrm{MB}$ of RAM; achieves **power requirements in the order of** $10\,\mathrm{mW}$ @ 52% mAP on VOC2012.

`micromind/recipes/object_detection`

YOLOXiNet


INPUT
Conv Block
Conv Block
Conv Block
Conv Block
Conv Block
Feature extractor - XiNet
SPP Neck
Neck
Conv Block
Conv Block
Conv Block
Object detection
Large Scale
Medium Scale
Small Scale
Detection head
DETECTIONS
Conv Block =
Attention Module
Pointwise
Conv2D
Input
Compression Convolution
Main Convolution
Elementwise product
Output
XiNet Conv Block
Broadcast Skip
Pointwise + Avg Pool

YOLOXiNet



Deployed on an Arm-Cortex M7 MCU with $2\,$MB of internal Flash and $1\,$MB of RAM; Achieves a reduction in the **number of operations of 2**$\times$ and a reduction in **RAM usage of 9**$\times$ with respect to MCUNet, with the same performance. Achieves a **power consumption of around** $20\,$mW @ 67% mAP on VOC2012.

`micromind/recipes/object_detection`

Class-Incremental Continual Learning

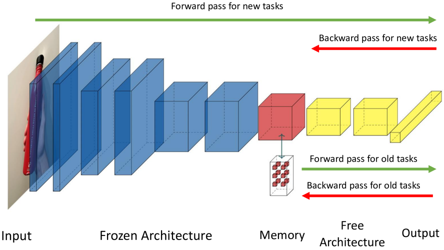
Figure: Samples of tasks from Split-CORe50

Pretty typical continual learning setting:

- stream: adding two classes for each task;
- we want to learn new classes, and not forget old ones...

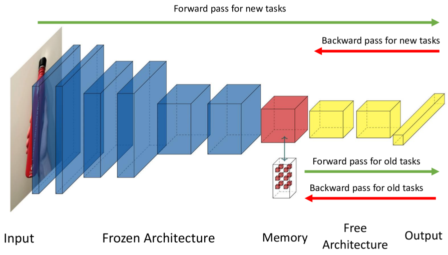
Latent Replay

An efficient CL strategy based on the replay of latent activations from previous taks.



Latent Replay

An efficient CL strategy based on the replay of latent activations from previous taks.



Of course we replace the backbone with a PhiNet, and...

CL results

TABLE II: Best results on CIFAR-10 and CORe50 for different models with a fixed replay memory of 0.5MB and 2MB.

Model	**CIFAR10**				**CORe50**			
	0.5 MB		**2 MB**		**0.5 MB**		**2 MB**	
	Avg. Acc. [%]	**MACs**	**Avg. Acc. [%]**	**MACs**	**Avg. Acc. [%]**	**MACs**	**Avg. Acc. [%]**	**MACs**
PhiNet A	50.25	3,050	72.49	3,050	51.42	3,050	70.20	3,050
PhiNet B	48.48	3,450	63.83	3,450	47.19	3,450	63.73	3,450
PhiNet B_9	51.00	3,450	66.45	3,450	48.51	3,450	65.10	3,450
PhiNet C	47.96	4,650	69.58	4,650	46.95	4,650	65.65	4,650
MobileNetV1	32.29	51,200	32.15	51,200	23.11	51,200	23.24	51,200
MobileNetV2	37.01	43,092,800	61.21	43,092,800	46.51	20,085,760	68.99	43,092,800
0.75 MobileNetV2	46.53	28,089,840	65.18	28,089,840	45.93	28,089,840	66.26	28,089,840

PhiNet is much more replay-efficient wrt to MobileNet, achieving a **+6%** average accuracy over the tasks, with only $\ll 0.1\%$ of the MACs for the update.

Tremonti et al., "An empirical evaluation of tinyML architectures for Class-Incremental Continual Learning".

But wait...

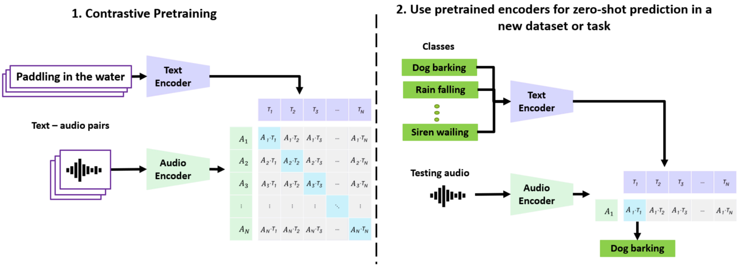
But wait...

TINYML LITERATURE
"Let's see who you really are!"
REPLACING THE BACKBONE
imgflip.com

Contrastive Language-Audio pretraining

- learns a **similarity score** between two modalities (audio and text);
- can be exploited for **zero-shot** classification;
- makes the network very **flexible** wrt the applications scenario they can be deployed to;

Zero-shot classification



tinyCLAP

- exploits the learned similarity score to learn a **more efficient audio network** (via a distillation process);

tinyCLAP

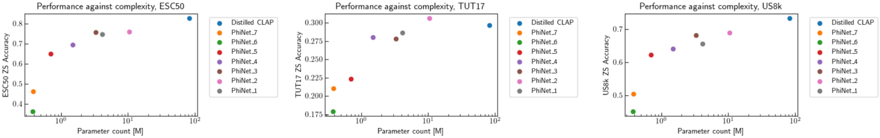
- exploits the learned similarity score to learn a **more efficient audio network** (via a distillation process);
- assumes the pre-trained **text encoder** does **not** need to be **deployed**;

tinyCLAP

- exploits the learned similarity score to learn a **more efficient audio network** (via a distillation process);
- assumes the pre-trained **text encoder** does **not** need to be **deployed**;
- achieves good performance-complexity tradeoff for ZS classification, and state-of-the-art for a benchmark;

`micromind/recipes/tinyCLAP`

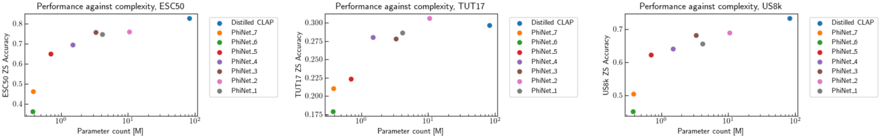
tinyCLAP: performance



- follows a common power-law scaling behaviour;

Paissan and Farella, "tinyCLAP: Distilling Constrastive Language-Audio Pretrained Models".

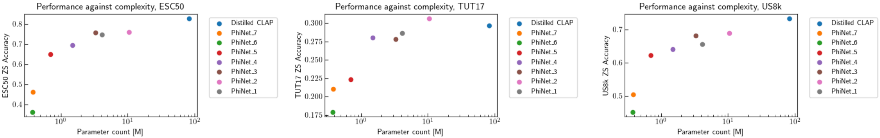
tinyCLAP: performance



- follows a common power-law scaling behaviour;
- was not yet deployed on edge platforms (WIP);

Paissan and Farella, "tinyCLAP: Distilling Constrastive Language-Audio Pretrained Models".

tinyCLAP: performance



- follows a common power-law scaling behaviour;
- was not yet deployed on edge platforms (WIP);
- **94% reduction in parameter count** wrt to original CLAP (from 82M to 4M), with a minor ZS accuracy drop (4% averaged on all benchmarks);

Paissan and Farella, "tinyCLAP: Distilling Constrastive Language-Audio Pretrained Models".

micromind: tinyML research made simple

- not a startup or a research project, just an **open-source project** for tinyML research;
- tries to provide the **full research pipeline** for model design, development, and deployment;

Checkout the project on GitHub and leave a star!

Follow me on X `@fpaissan_` for updates.

FYI

We welcome contributions on the following topics:

- anything related to interpretability for audio and speech (evaluation, post-hoc techniques, glass-box models);
- **deadline**: 4th March 2024 (non IEEE track);
- `https://xai-sa-workshop.github.io`

Additional references to our works

Following is a list of references to works related to the topics discussed in the presentation:

- Video processing: Ancilotto, Paissan, and Farella, "On the Role of Smart Vision Sensors in Energy-Efficient Computer Vision at the Edge"; Paissan, Ancilotto, and Farella, "PhiNets: A Scalable Backbone for Low-power AI at the Edge"; Ancilotto, Paissan, and Farella, "XiNet: Efficient Neural Networks for tinyML"
- Generative modeling: Ancilotto, Paissan, and Farella, "PhiNet-GAN: Bringing real-time face swapping to embedded devices"; Ancilotto, Paissan, and Farella, "XimSwap: many-to-many face swapping for TinyML"
- Audio processing: Paissan et al., "Scalable Neural Architectures for End-to-End Environmental Sound Classification"; Brutti et al., "Optimizing PhiNet architectures for the detection of urban sounds on low-end devices"; Ali et al., "Scaling strategies for on-device low-complexity source separation with Conv-Tasnet"; Paissan et al., "Improving latency performance trade-off in keyword spotting applications at the edge"
- Multimodal processing: Paissan and Farella, "tinyCLAP: Distilling Constrastive Language-Audio Pretrained Models"

The End

Questions? Comments?

Ali, Mohamed Nabih et al. "Scaling strategies for on-device low-complexity source separation with Conv-Tasnet". In: *ArXiv* abs/2303.03005 (2023). URL: https://api.semanticscholar.org/CorpusID:257364800.

Ancilotto, A., F. Paissan, and Elisabetta Farella. "XiNet: Efficient Neural Networks for tinyML". In: *ICCV2023* (2023). URL: https://openaccess.thecvf.com/content/ICCV2023/papers/Ancilotto_XiNet_Efficient_Neural_Networks_for_tinyML_ICCV_2023_paper.pdf.

Ancilotto, Alberto, Francesco Paissan, and Elisabetta Farella. "On the Role of Smart Vision Sensors in Energy-Efficient Computer Vision at the Edge". In: *2022 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops)* (2022), pp. 497–502. URL: https://api.semanticscholar.org/CorpusID:248546511.

— ."PhiNet-GAN: Bringing real-time face swapping to embedded devices". In: *2023 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops)* (2023), pp. 677–682. URL: