tinyML® Talks

Enabling Ultra-low Power Machine Learning at the Edge

"Enabling on-device learning on STM32 microcontrollers"

Beatrice Rossi – Research Scientist, STMicroelectronics
Michele Craighero - PhD Student, Politecnico di Milano

September 5, 2023

TINY ML

www.tinyML.org

Thank you, **tinyML Strategic Partners**, for committing to take tinyML to the next Level, together

Executive Strategic Partners

Qualcomm
AI research

Advancing AI research to make efficient AI ubiquitous

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

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

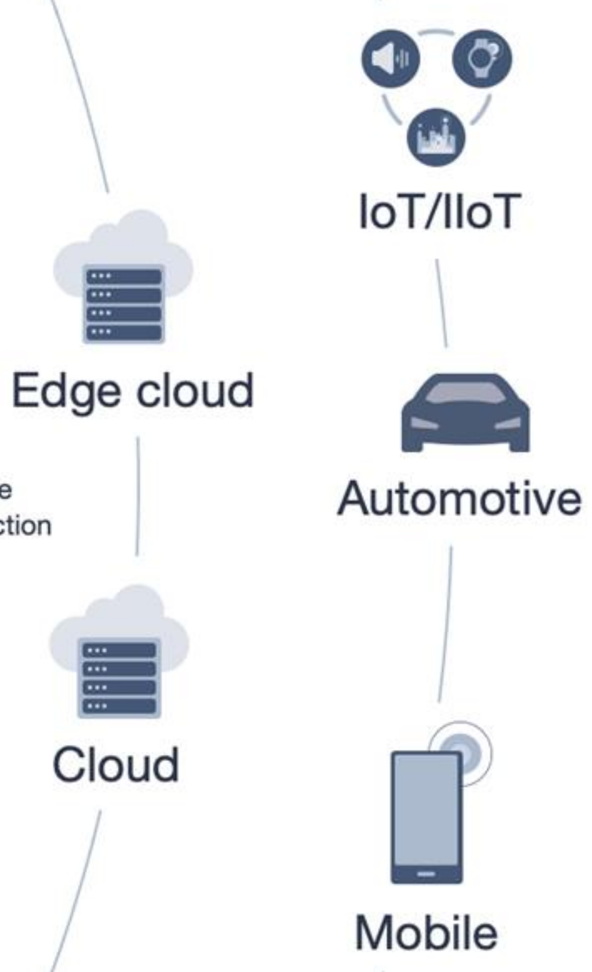
Efficient learning
Robust learning through minimal data, unsupervised learning, on-device learning

A platform to scale AI across the industry

Perception
Object detection, speech recognition, contextual fusion

Reasoning
Scene understanding, language understanding, behavior prediction

Action
Reinforcement learning for decision making



Accelerate Your Edge Compute

SYNTIANT

Making Edge AI A Reality

www.syntiant.com

Platinum Strategic Partners

DEPLOY VISION AI AT THE EDGE AT SCALE

Gold Strategic Partners

AHEAD OF WHAT'S POSSIBLE™

AHEAD OF WHAT'S POSSIBLE™

Where what if becomes what is.

Witness potential made possible at analog.com.

Build the Future of tinyML

on arm

TINY ML TALKS webcast

EDGE IMPULSE

The Leading Development Platform for Edge ML

edgeimpulse.com

Driving decarbonization and digitalization. Together.

Infineon serving all target markets as
Leader in Power Systems and IoT

www.infineon.com

Infineon

innatera

NEUROMORPHIC INTELLIGENCE FOR THE SENSOR-EDGE

www.innatera.com

Microsoft

Renesas is enabling the next generation of AI-powered solutions that will revolutionize every industry sector.

renesas.com

STMicroelectronics provides extensive solutions to make tiny Machine Learning easy

life.augmented

synaptics

ENGINEERING EXCEPTIONAL EXPERIENCES

We engineer exceptional experiences for consumers in the home, at work, in the car, or on the go.

www.synaptics.com

TINY ML TALKS webcast

Silver Strategic Partners

AIZIP
ARDUINO
brainchip
EFFICIENT
GREENWAVES TECHNOLOGIES
Grovety Inc.
Himax
IBM
imagimob
Nota AI
NXP
POLYN TECHNOLOGY
Qeexo
Schneider Electric
SensiML
SILICON LABS
TDK

Join Growing tinyML Communities:

16.5k members in
49 Groups in 41 Countries

tinyML - Enabling ultra-low Power ML at the Edge

https://www.meetup.com/tinyML-Enabling-ultra-low-Power-ML-at-the-Edge/

4k members
&
13k followers

The tinyML Community

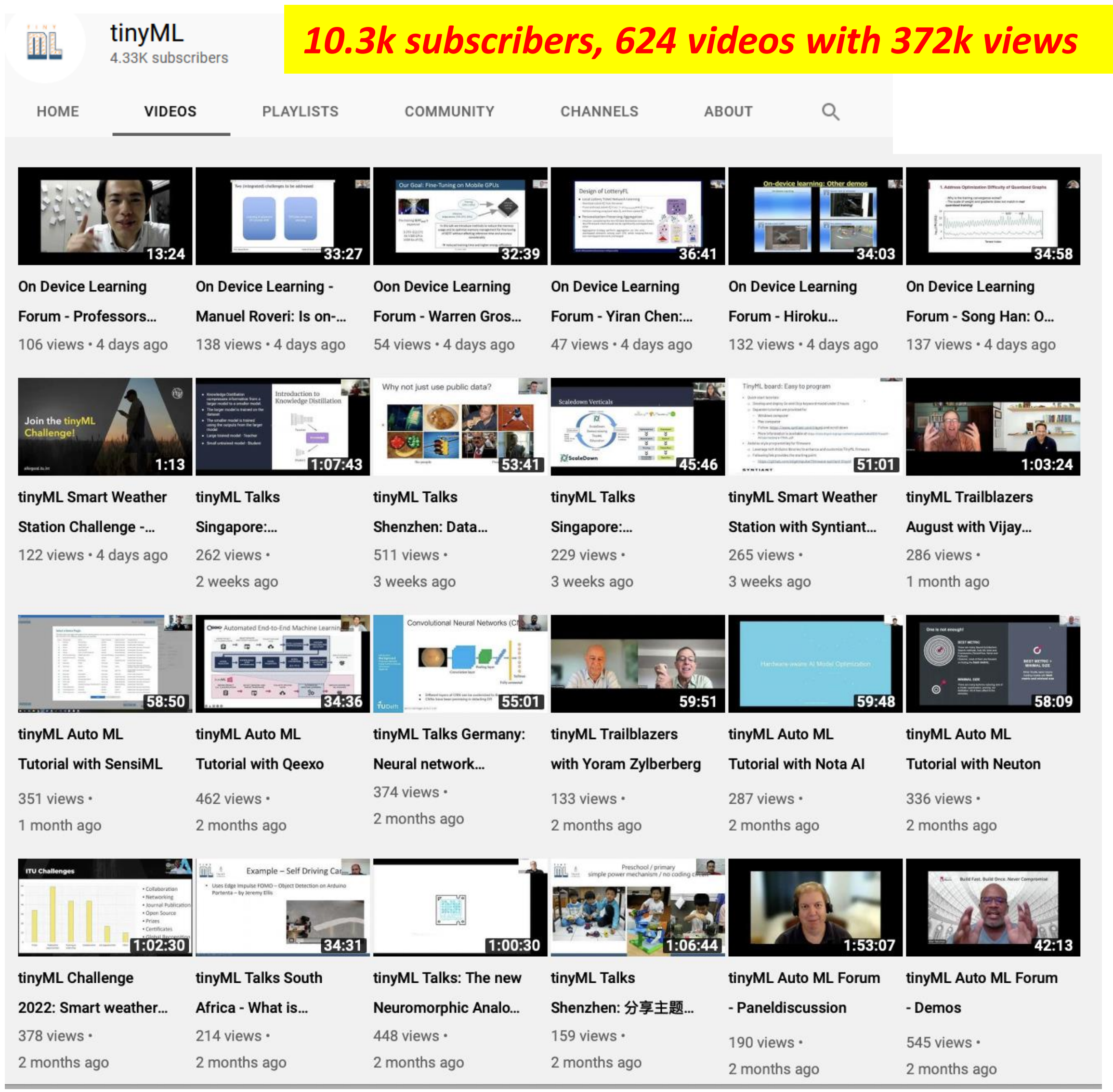
https://www.linkedin.com/groups/13694488/

TINY ML TALKS webcast

Subscribe to tinyML YouTube Channel for updates and notifications *(including this video)*

www.youtube.com/tinyML

YouTube



Call for Presentations and Posters – Deadline August 7
https://www.tinyml.org/event/asia-2023/

WEVOLVER

https://www.wevolver.com/article/2023-edge-ai-technology-report

TINY ML FOUNDATION

Reminders

Slides & Videos will be posted tomorrow

tinyml.org/forums

youtube.com/tinyml

Please use the Q&A window for your questions

Beatrice Rossi

Beatrice Rossi graduated in Mathematics and Applications at Università degli Studi di Milano Bicocca in 2008. Since then, she has been working in STMicroelectronics, System Research and Applications. Her research interests include Edge AI, Tiny Machine and Deep Learning, and Distributed Ledger Technology for the IoT.

Michele Craighero

Michele Craighero graduated in Computer Science and Engineering at Politecnico di Milano in 2022 and he is currently in the first year of his PhD. His research project is titled “Learning and Adaptation in Distributed Environments” and it is a collaboration between Politecnico di Milano and STMicroelectronics. His research interests include Machine Learning techniques for Time Series Classification, Change Detection and Unsupervised Domain Adaptation.

Enabling On-Device Learning on STM32 microcontrollers

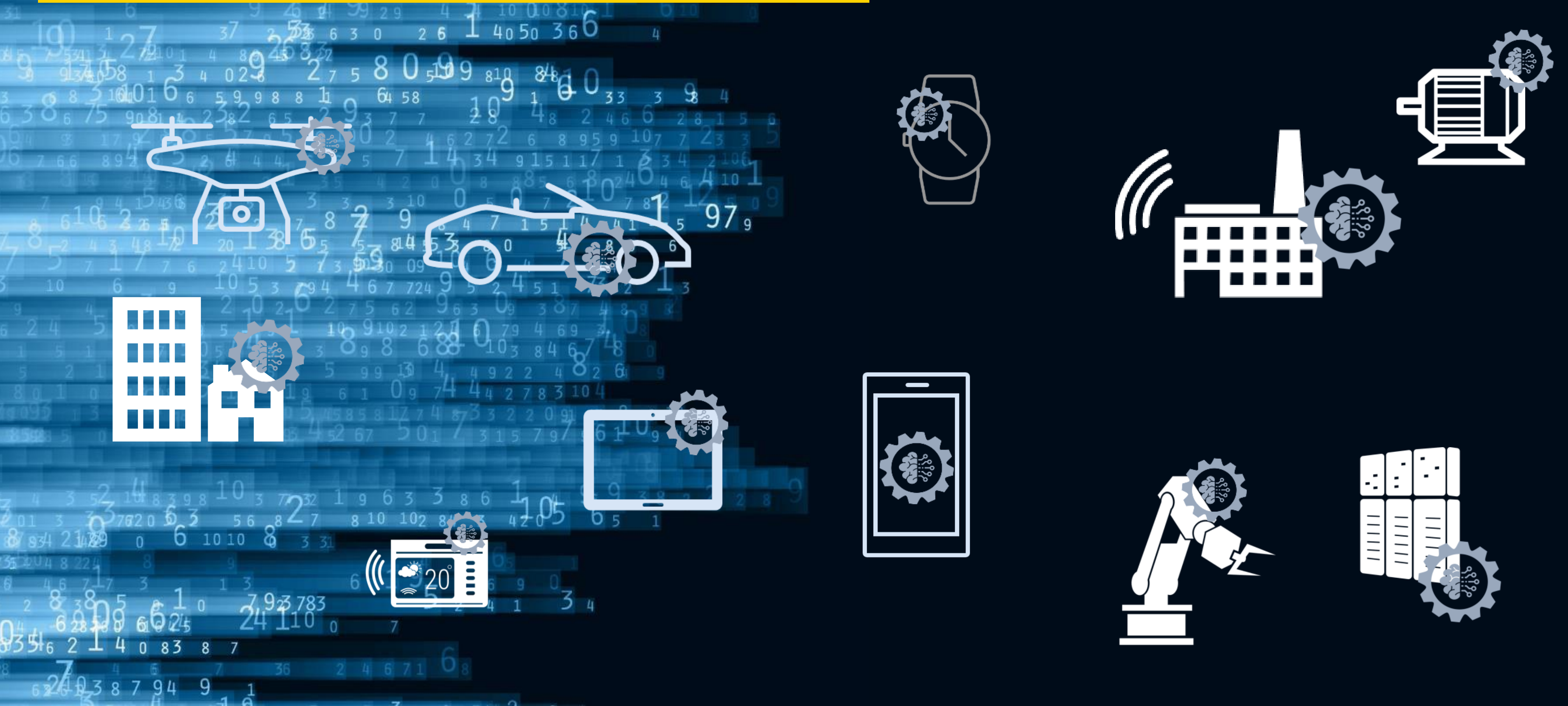
Beatrice Rossi, Michele Craighero

System Research and Applications, STMicroelectronics

DEIB, Politecnico di Milano

The rise of Edge AI

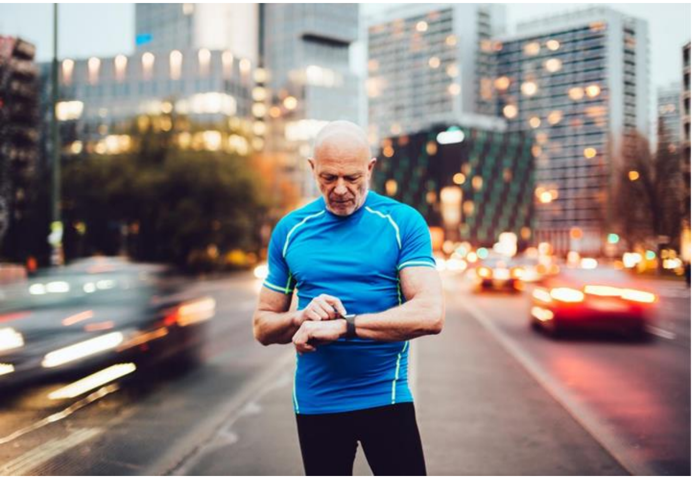
We are surrounded of smart devices



Focus Applications

Industrial Maintenance
Condition monitoring, Predictive maintenance.

Internet of Things (IoT)
Smart cities, Smart buildings, Connected homes and things



Healthcare and Wellbeing Systems
Monitoring through wearables, Remote care.

Automotive
Enhanced safety, efficiency, overall driving experience; BMS.

Example: Human Activity Recognition

HAR is a time series classification task identifying the specific movement or action of a person based on sensor data.

Approach
– Exploits 3-axis accelerometer data
– Classes: stationary, walking, running, biking, driving…

1D - Convolutional Neural Network model

OPTIMIZED WITH

STM32 Cube.AI

MODEL CREATED WITH

AI Function Pack

FP-AI-SENSING1

RUNNING ON

STEVAL-STLKT01V1

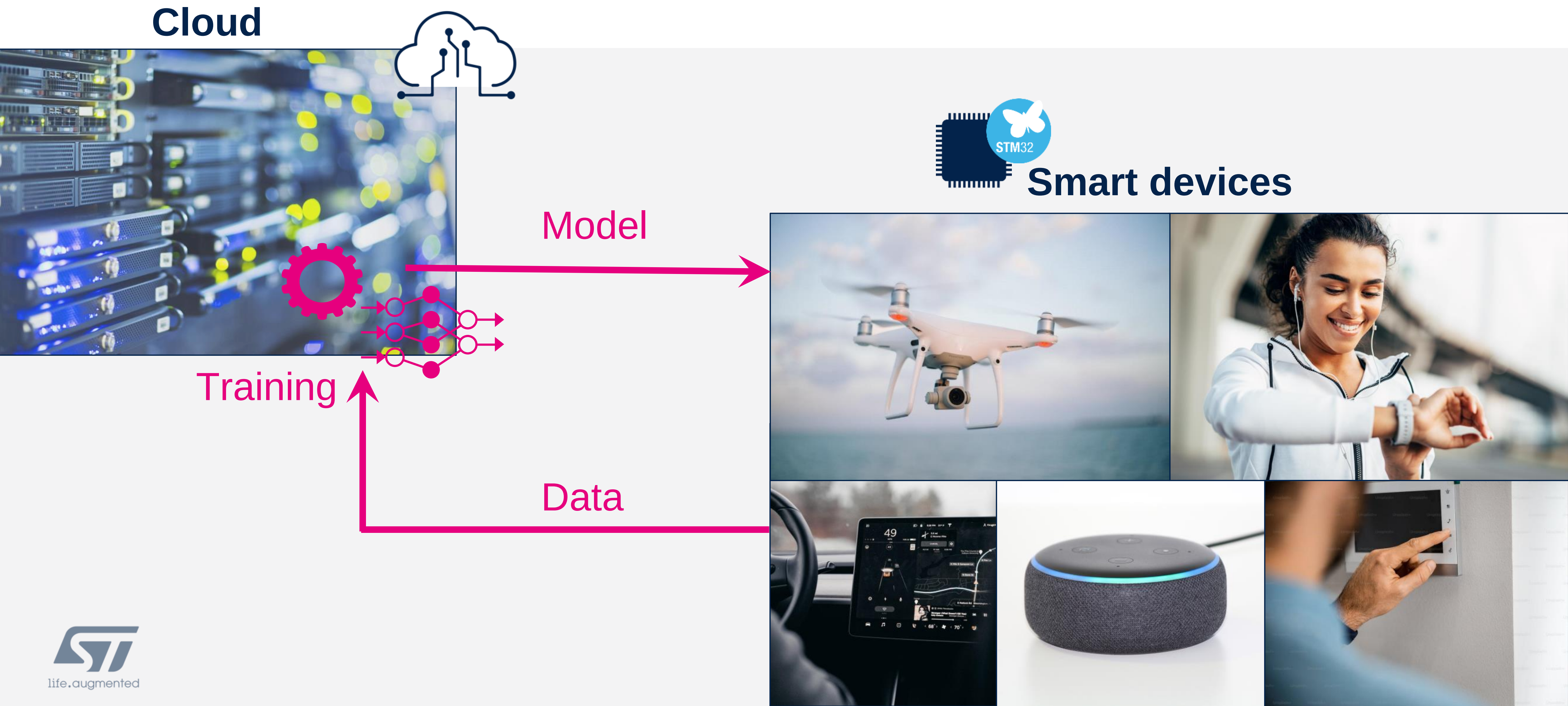
COMPATIBLE WITH

STM32

STM32L4 SERIES

life.augmented

From on-cloud to on-device learning



From on-cloud to on-device learning

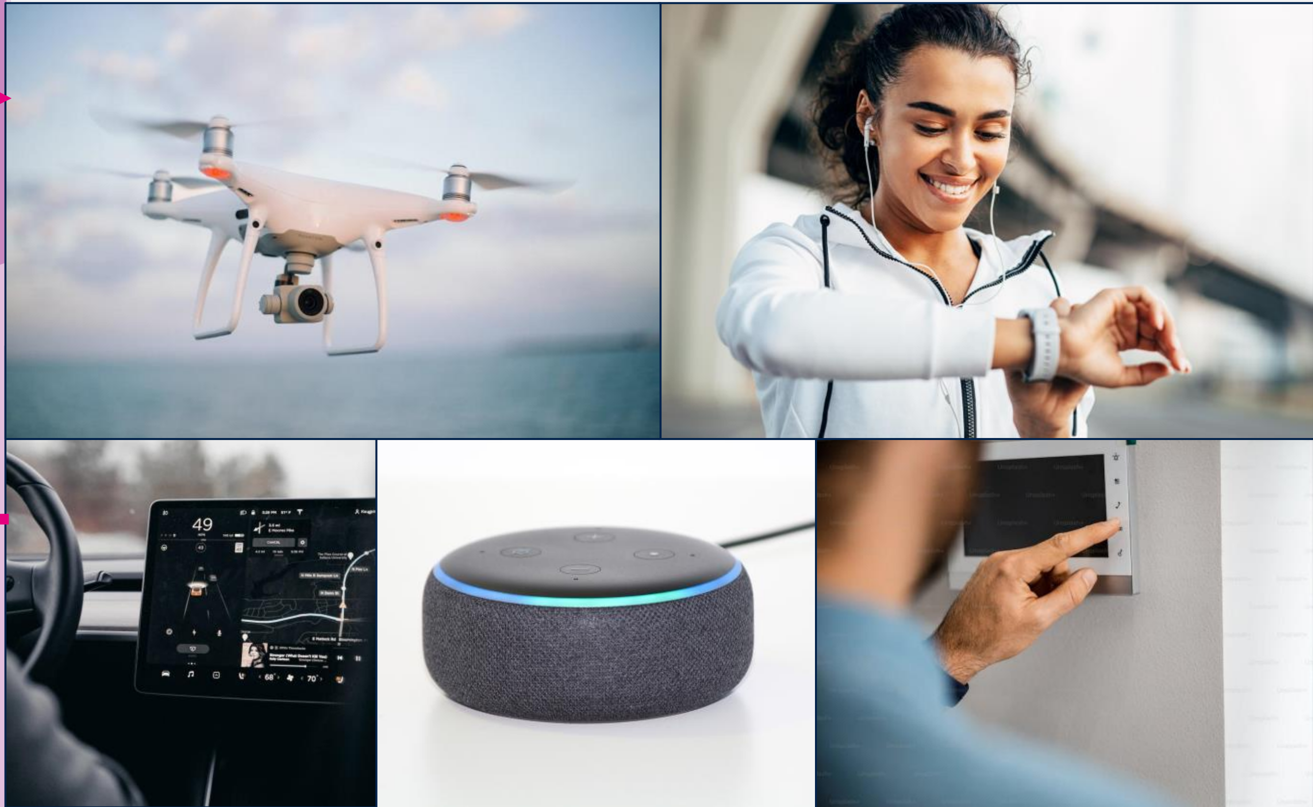
Cloud

Model

Training

Data

Smart devices



Drawbacks

- Low privacy and security of data
- High latency
- Low personalization

From on-cloud to on-device learning

Cloud

Pretrained Model

Smart devices

On-Device Learning (ODL): Adapt a pretrained model after deployment based on user’s interaction and newly acquired data.

The benefits of ODL

Enhanced privacy and security

Lower latency

Improved accuracy:
- By **Personalization**
- By more sophisticated learning schemes as **Federated Learning**

Improve AI-powered experiences

Enable key features for our products

Where we are

Cloud-based learning

On-device learning

Data and labels

On-device inference

Frameworks

TensorFlow Lite

STM32 Cube.AI

Model optimization
Model quantization
Code generation

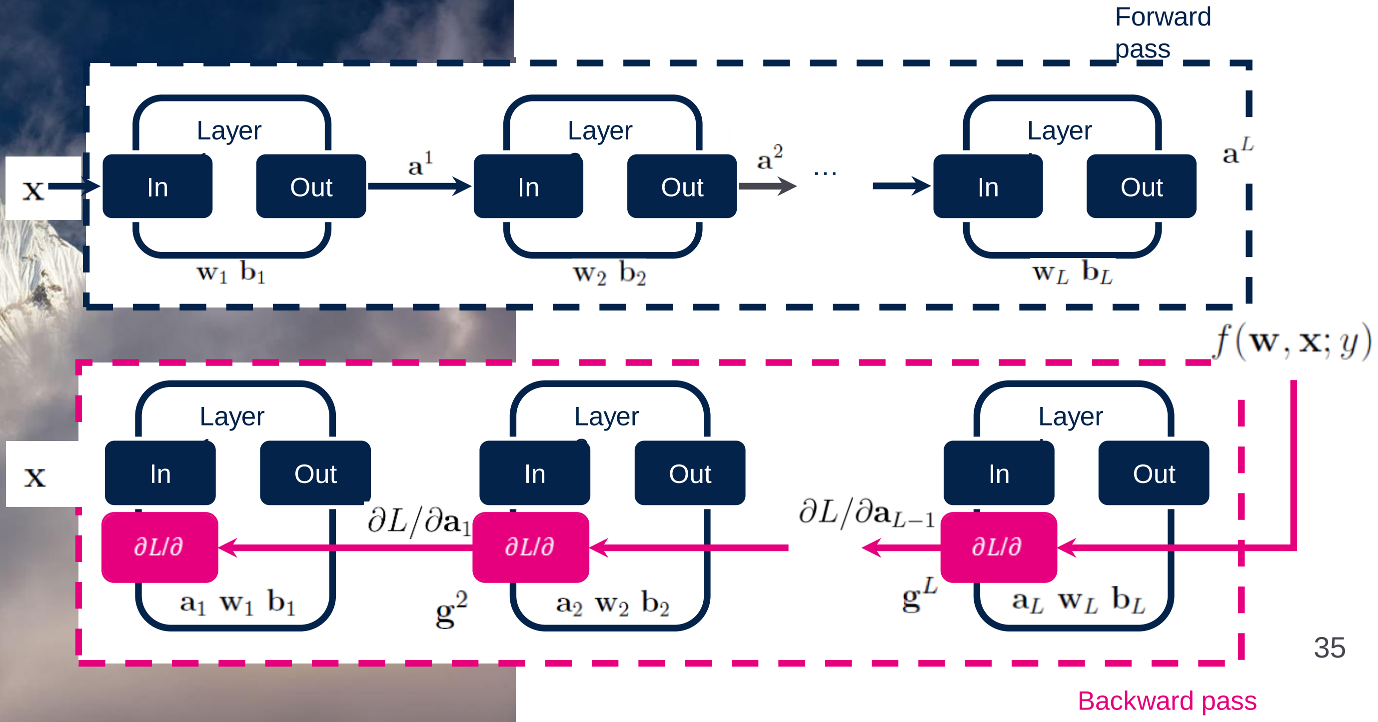
Solutions for efficient **on-device inference** are **mature**

those of **learning are at their early stages (especially on MCUs!)**

The Challenge

Enable ODL functionalities on STM32 microcontrollers

Performing **backpropagation** on MCUs is highly challenging due to the strict memory and computational constraints.



Our Contributions

Seminal work presented at TinyML Summit 2023

CRAIGHERO, Michele, et al.
On-Device Personalization for Human Activity Recognition on STM32.
IEEE Embedded Systems Letters, 2023.

1. **SW framework** to train 1D-CNNs on STM32 MCUs;
2. **Explicit gradient computation** for the several common network layers;
3. **Memory footprint and CPU loads estimates**;
4. **Case study**: personalization for HAR.

life.augmented

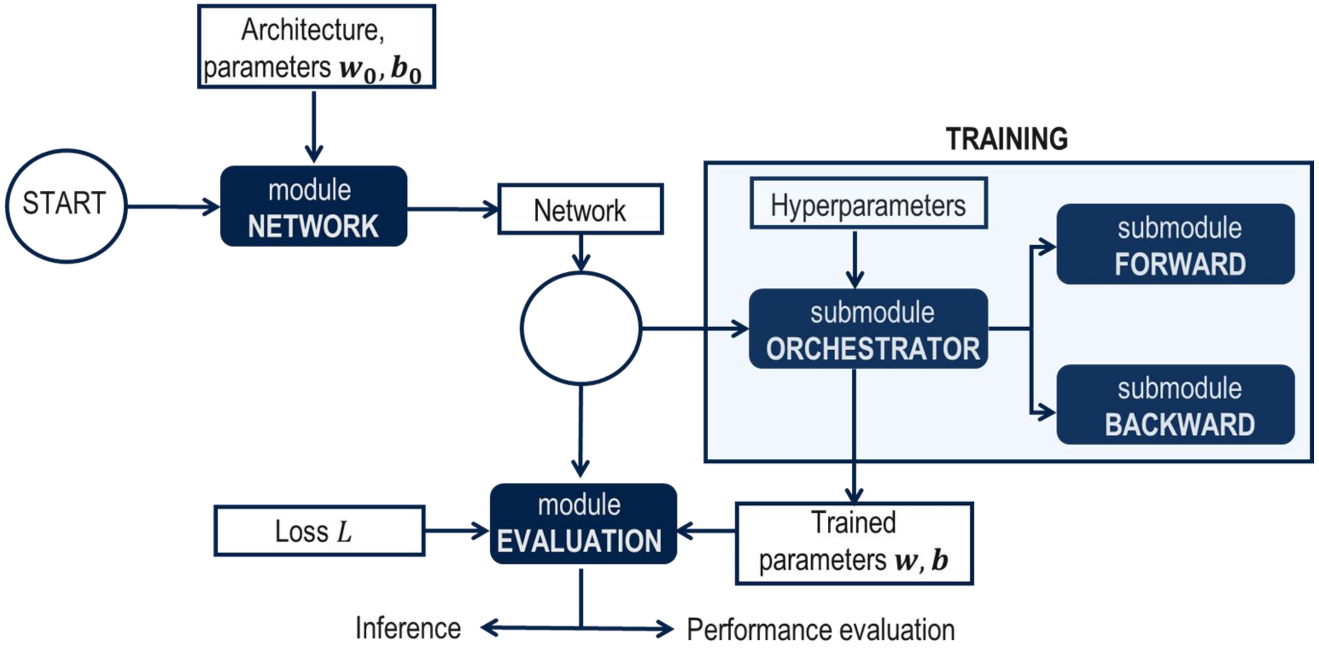
SW Framework

SW Framework

Network:

Instantiates the network from topology specs;

Initializes network parameters (randomly set or imported from a pre-trained model).



Training:

Orchestrator: governs the training procedure by invoking alternatively the Forward and Backward modules;

Forward: performs the FW pass implementing the forward expressions;

Backward: performs the BW pass implementing the backward expressions

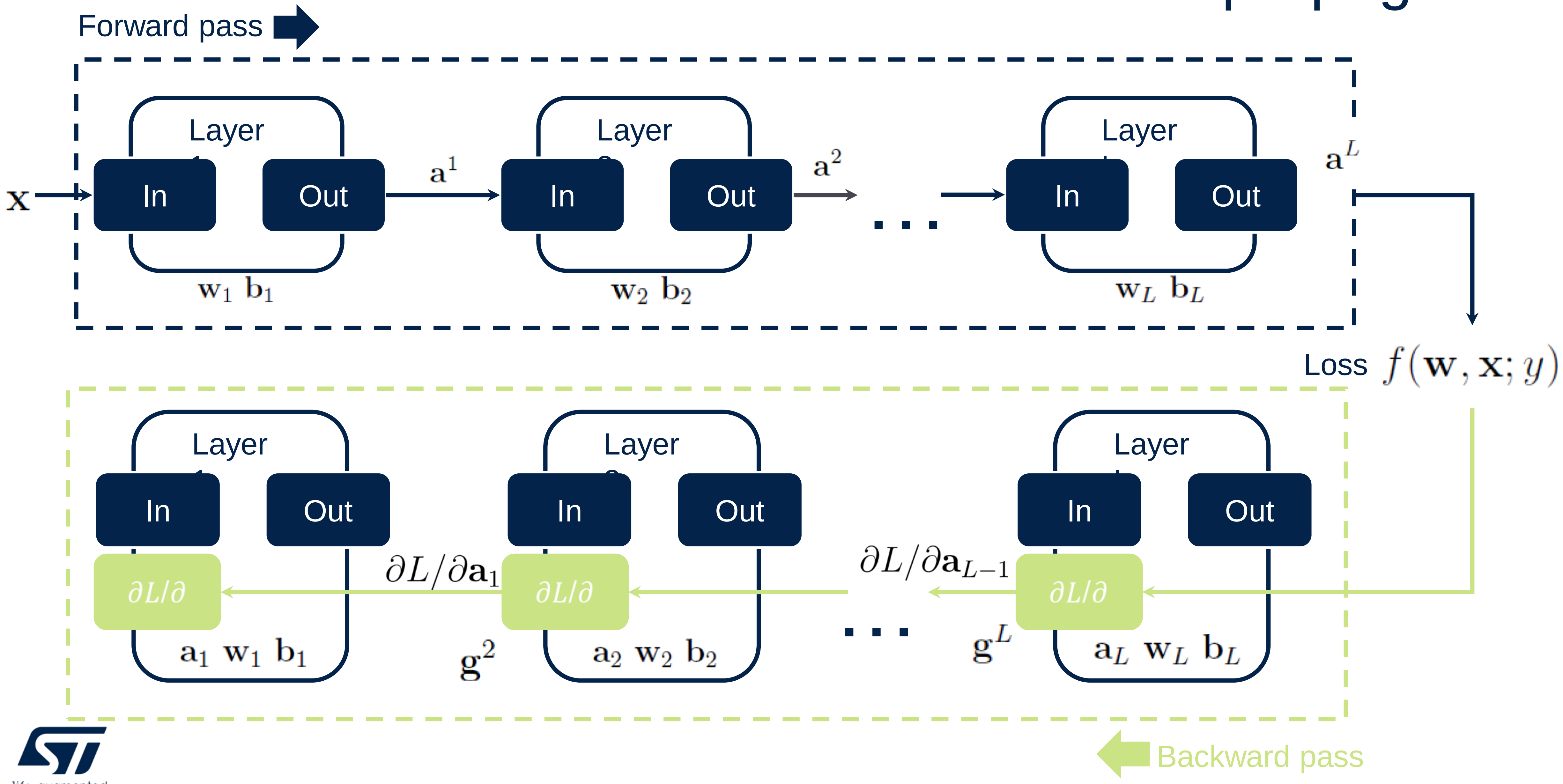
Evaluation:

Performs inference and computes the loss

life.augmented

Gradient Computation

Backpropagation



Gradient Computation

$$\frac{\partial L}{\partial \boldsymbol{w}} = \sum_{i=1}^{M} \frac{\partial L}{\partial a_i} \frac{\partial f_i}{\partial \boldsymbol{w}}$$

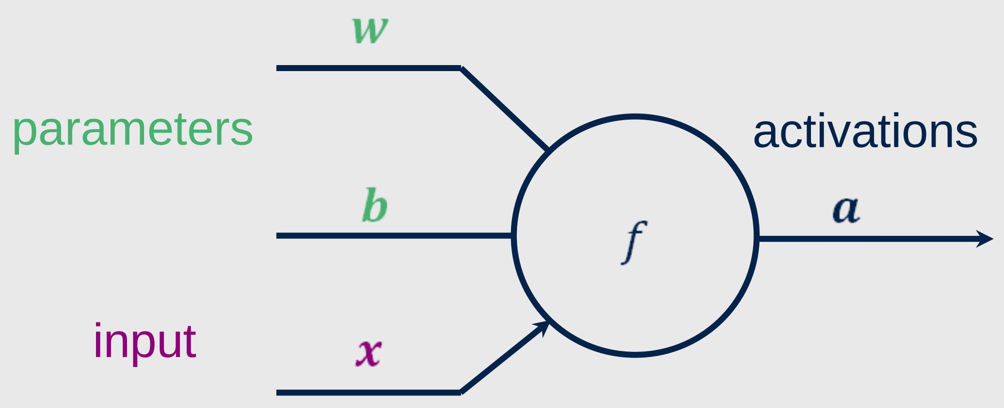
$$\frac{\partial L}{\partial \boldsymbol{b}} = \sum_{i=1}^{M} \frac{\partial L}{\partial a_i} \frac{\partial f_i}{\partial \boldsymbol{b}}$$

$$\frac{\partial L}{\partial \boldsymbol{x}} = \sum_{i=1}^{M} \frac{\partial L}{\partial a_i} \frac{\partial f_i}{\partial \boldsymbol{x}}$$

Multivariate chain rule

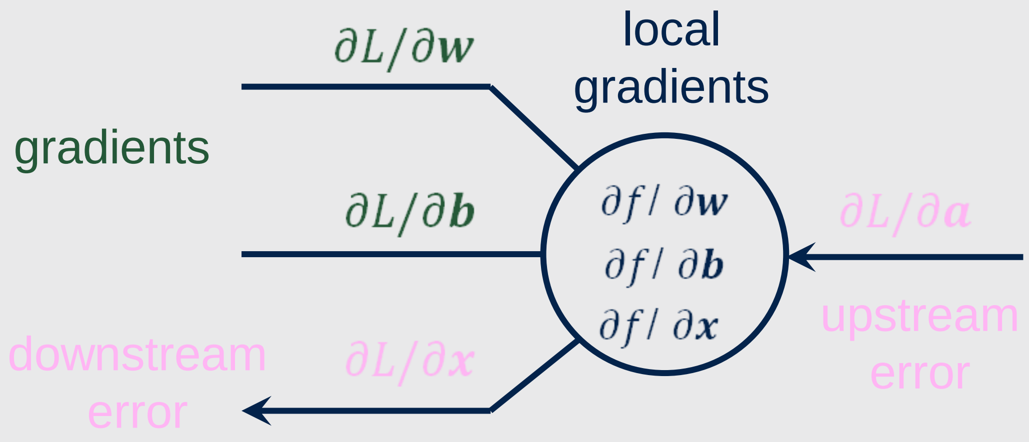
Forward pass

$$\boldsymbol{a} = f(\boldsymbol{x}, \boldsymbol{w}, \boldsymbol{b})$$



The function f depends on the type of layer

Backward pass



Derivatives are computed using the Multivariate Chain Rule

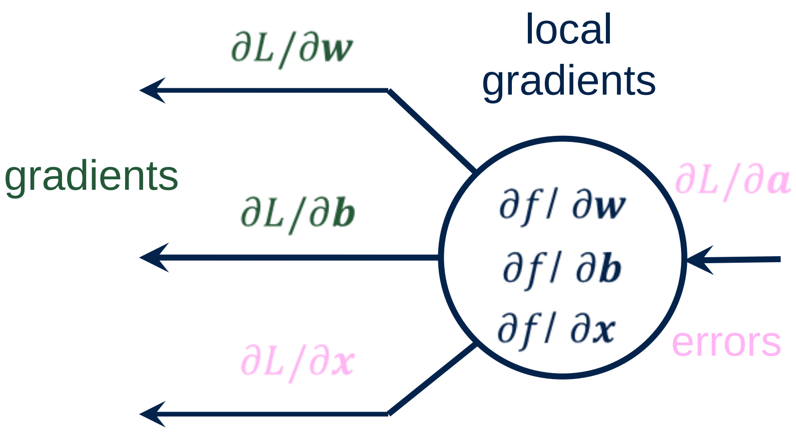
life.augmented

Expressions for Layers of 1D-CNNs

Layer	Forward pass	Backward pass			Parameters	
		Input	Weights	Bias	Weights	Bias
Dense	$\mathrm{a}=\mathrm{w}^T \cdot \mathbf{x}+\mathbf{b}$	$\frac{\partial L}{\partial \mathbf{x}}=\mathbf{w}\cdot\frac{\partial L}{\partial \mathbf{a}}$	$\frac{\partial L}{\partial \mathbf{w}}=\mathbf{x}\cdot\left(\frac{\partial L}{\partial \mathbf{a}}\right)^T$	$\frac{\partial L}{\partial \mathbf{b}}=\frac{\partial L}{\partial \mathbf{a}}$	$M \cdot N$	N
Conv1D	a=conv(x,w)+b	$\frac{\partial L}{\partial \mathbf{x}}=\mathrm{conv}\left(\frac{\partial L}{\partial \mathbf{a}}, \mathrm{flip}(\mathbf{w}), \mathrm{full}\right)$	$\frac{\partial L}{\partial \mathbf{w}}=\mathrm{conv}\left(\frac{\partial L}{\partial \mathbf{a}}, \mathbf{x}\right)$	$\frac{\partial L}{\partial \mathbf{b}}=\frac{\partial L}{\partial \mathbf{a}}$	$F \cdot C \cdot K$	F
Activation	$\mathrm{a}_i=\frac{e^{x_i}}{\sum_{i=1}^{N} e^{x_i}}$	$\frac{\partial L}{\partial \mathbf{x}}=\mathbf{a}-\mathbf{t}$	-	-	-	-
AvgPool1D	$\mathrm{a}_{jm}=\frac{1}{p}\sum_{i=0}^{p} x_{ji}$	$\frac{\partial L}{\partial x_{in}}=\frac{1}{p}\frac{\partial L}{\partial a_{ij}}$	-	-	-	-
GlobalAvgPool1D	$\mathrm{a}_j=\frac{1}{N}\sum_{i=0}^{N} x_{ji}$	$\frac{\partial L}{\partial x_{ij}}=\frac{1}{N}\frac{\partial L}{\partial a_{in}}$	-	-	-	-
Flatten	a = vec(x)	$\frac{\partial L}{\partial \mathbf{x}}=reshape(\frac{\partial L}{\partial \mathbf{a}})$	-	-	-	-

life.augmented

Example: Conv1D



The layer function

$$a_{j,m} = \sum_{c=1}^{C} \sum_{k=1}^{K} x_{c,m+k-1} w_{j,c,k} + b_j$$

$$j \in \{1, \dots, F\} \quad m \in \{1, \dots, M\}$$
$$M = N - K + 1$$

Derivatives

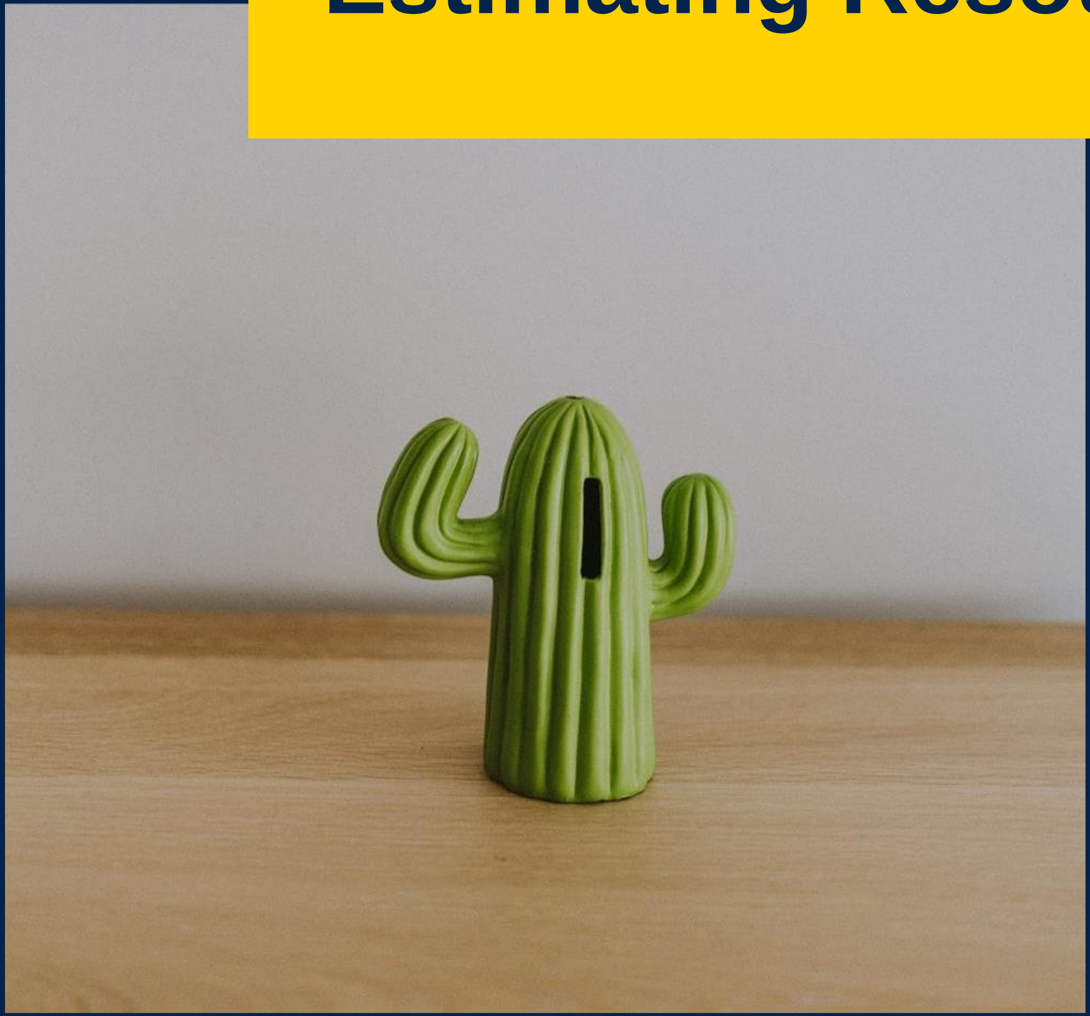
$$\frac{\partial L}{\partial \boldsymbol{x}} = \sum_{j=1}^{F} \sum_{k=1}^{K} \frac{\partial L}{\partial a_{j,m}} \frac{\partial f_{j,m}}{\partial \boldsymbol{x}} \qquad \frac{\partial L}{\partial x_{i,n}} = \sum_{j=1}^{F} \sum_{k=1}^{K} \frac{\partial L}{\partial a_{j,n-k+1}} w_{j,i,k}$$

$$\frac{\partial L}{\partial \boldsymbol{x}} = conv\left(pad\left(\frac{\partial L}{\partial \boldsymbol{a}}\right), flip(\boldsymbol{w})\right)$$

Hints

- $n - k + 1$ ranges from $2 - K$ to N ($N + K - 1$ terms for each channel j)
- $\frac{\partial L}{\partial \boldsymbol{a}}$ has size $F \times (N - K + 1)$. If $K > 1$, we apply a zero-padding to $\frac{\partial L}{\partial \boldsymbol{a}}$ by adding $F \cdot (K - 1)$ zeros to both sides of $\frac{\partial L}{\partial \boldsymbol{a}}$ along its second dimension
- Index k has opposite signs in the two terms of the convolution ($-k$ in $\frac{\partial L}{\partial \boldsymbol{a}}$ and $+k$ in w), thus a flipped kernel is obtained

Estimating Resources



life.augmented

Memory Footprint

Our SW framework is equipped with a tool that estimates the memory footprint and the CPU loads needed to train a given neural network.

Memory Footprint

- Model memory
- Activations memory
- Optimizer memory:
- Stochastic Gradient Descent: requires storing the first order momentum of each parameter, thus occupies the same memory as the model.
- Adam: uses the first and second order momenta of each parameters. It occupies 2x the size of the model.

life.augmented

Model Memory

Memory used to store network's parameters (weights & biases and other hyperparameters)

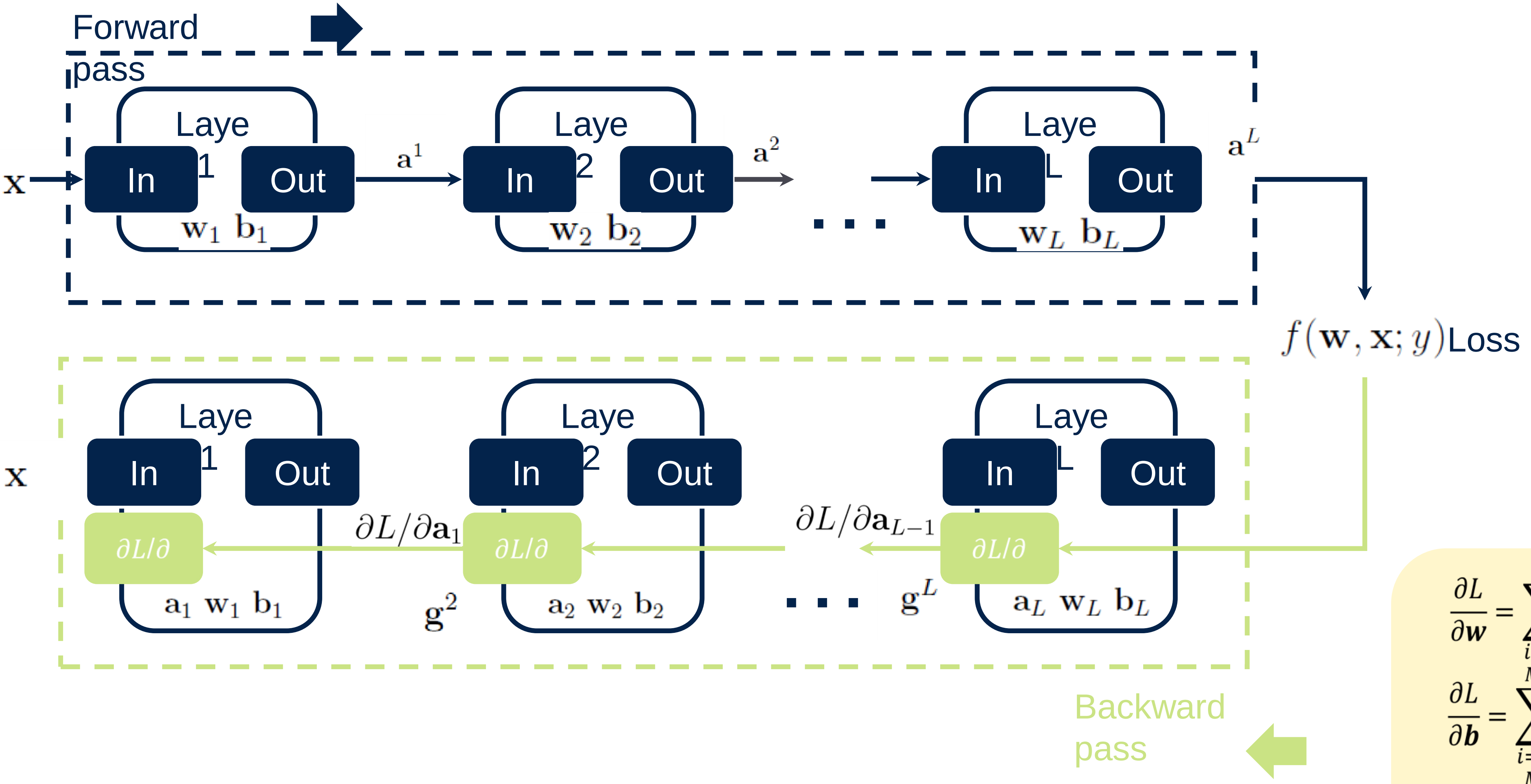
Depends on the network's architecture and remains constant regardless of the input or batch size

#parameters * bit precision
Not quantized, at least in this seminal work

Not optimized, at least in this seminal work

life.augmented

Activations Memory



Multivariate chain rule

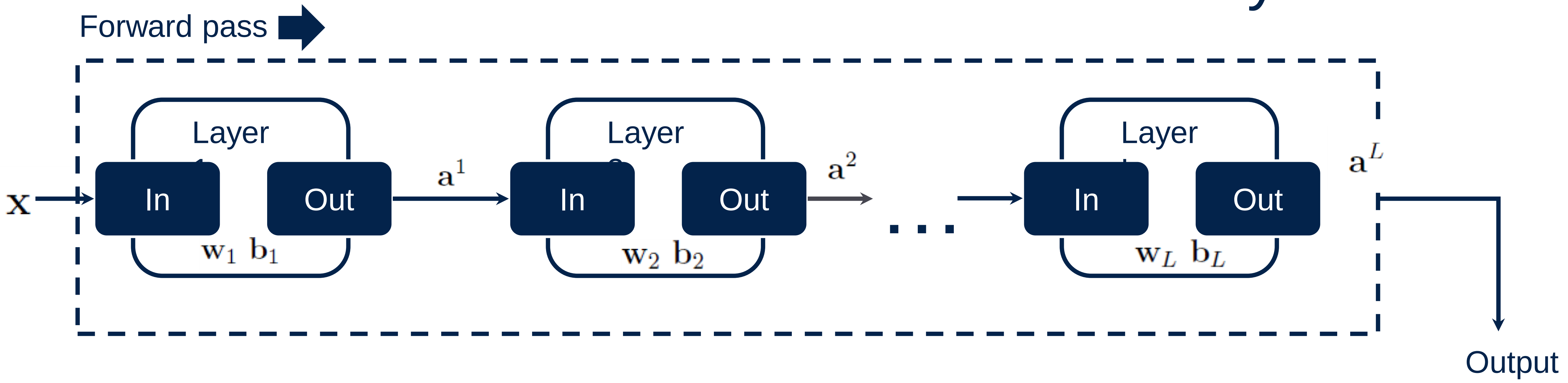
$$\frac{\partial L}{\partial \boldsymbol{w}} = \sum_{i=1}^{M} \frac{\partial L}{\partial a_i} \frac{\partial f_i}{\partial \boldsymbol{w}}$$

$$\frac{\partial L}{\partial \boldsymbol{b}} = \sum_{i=1}^{M} \frac{\partial L}{\partial a_i} \frac{\partial f_i}{\partial \boldsymbol{b}}$$

$$\frac{\partial L}{\partial \boldsymbol{x}} = \sum_{i=1}^{M} \frac{\partial L}{\partial a_i} \frac{\partial f_i}{\partial \boldsymbol{x}}$$

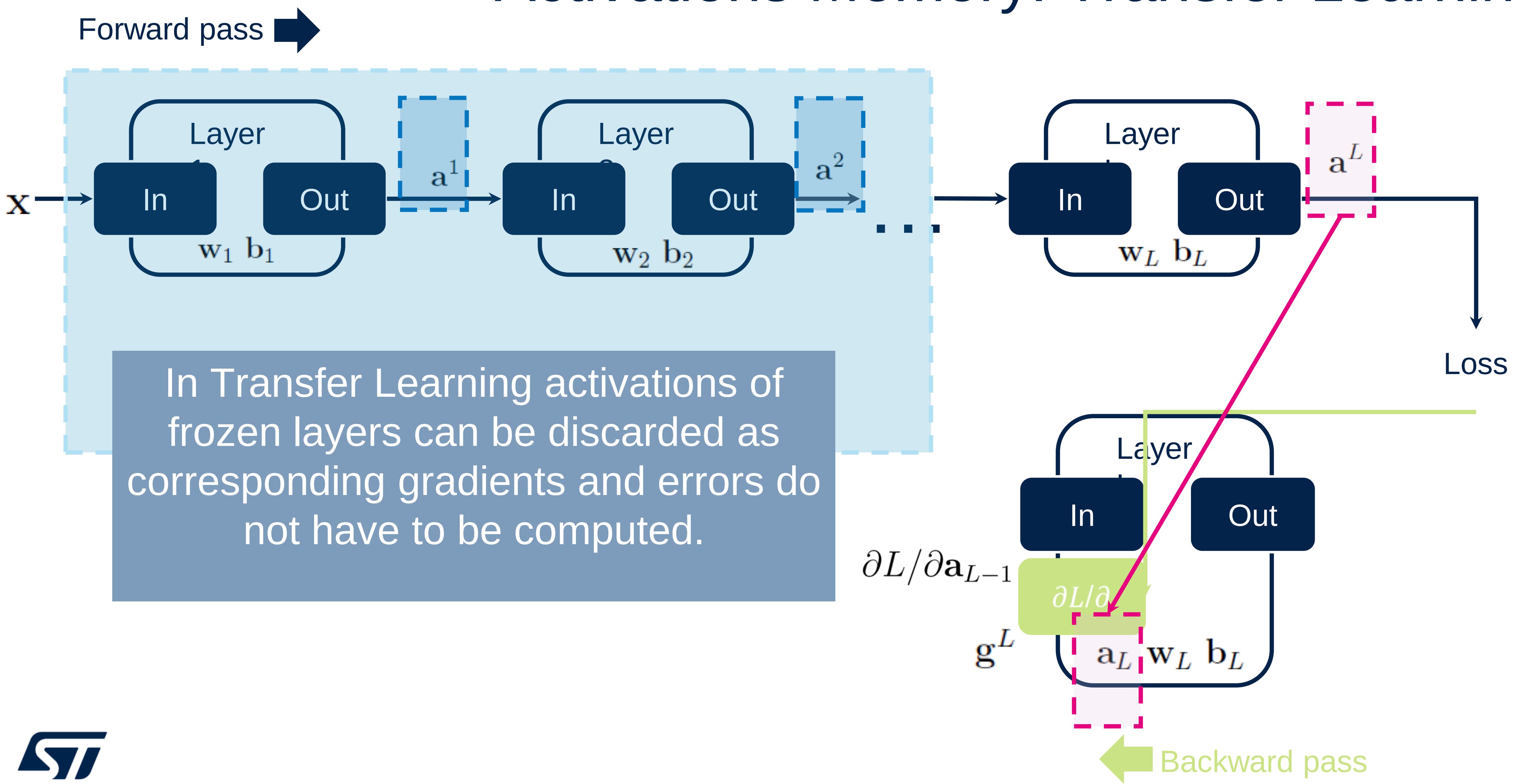
life.augmented

Activations Memory: Inference



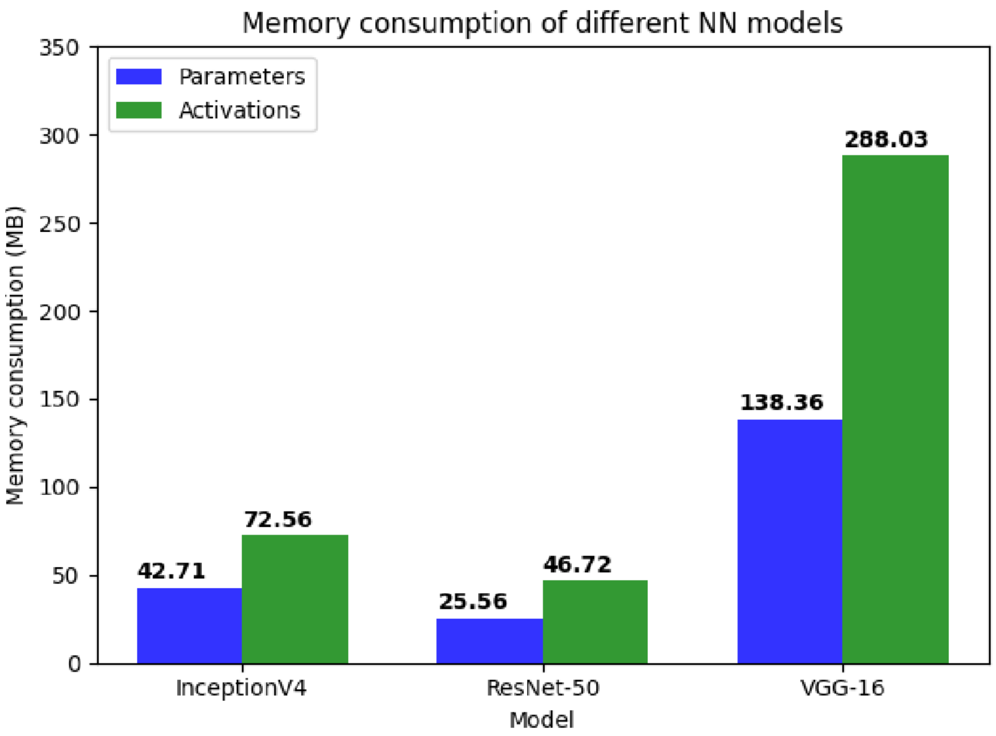
In **on-device inference**, activations of previous layers can be discarded while the computation goes forward (they are stored temporarily in an over-writable buffer)

Activations Memory: Transfer Learning



In Transfer Learning activations of frozen layers can be discarded as corresponding gradients and errors do not have to be computed.

Activations Memory



Memory used to store the activations corresponding to the layers we want to train on-device.

Depends on the network's architecture, input and batch size.

#activations * bit precision

Not quantized, at least in this seminal work

CPU Load

Our SW framework is equipped with a tool that estimates the memory footprint and the CPU loads needed to train a given neural network.

Layer	Forward pass	Backward pass			Parameters	
		Input	Weights	Bias	Weights	Bias
Dense	$a = w^T \cdot x + b$	$\frac{\partial L}{\partial x} = w \cdot \frac{\partial L}{\partial a}$	$\frac{\partial L}{\partial w} = x \cdot \left(\frac{\partial L}{\partial a}\right)^T$	$\frac{\partial L}{\partial b} = \frac{\partial L}{\partial a}$	$M \cdot N$	N
Conv1D	$a = \mathrm{conv}(x, w) + b$	$\frac{\partial L}{\partial x} = \mathrm{conv}\left(\frac{\partial L}{\partial a}, \mathrm{flip}(w), \mathrm{full}\right)$	$\frac{\partial L}{\partial w} = \mathrm{conv}\left(\frac{\partial L}{\partial a}, x\right)$	$\frac{\partial L}{\partial b} = \frac{\partial L}{\partial a}$	$F \cdot C \cdot K$	F
Activation	$a_i = \frac{e^{x_i}}{\sum_{i=1}^{N} e^{x_i}}$	$\frac{\partial L}{\partial x} = a - t$	-	-	-	-
AvgPool1D	$a_{jm} = \frac{1}{p} \sum_{i=0}^{p} x_{ji}$	$\frac{\partial L}{\partial x_{in}} = \frac{1}{p} \frac{\partial L}{\partial a_{ij}}$				
GlobalAvgPool1D	$a_j = \frac{1}{N} \sum_{i=0}^{N} x_{ji}$	$\frac{\partial L}{\partial x_{ij}} = \frac{1}{N} \frac{\partial L}{\partial a_{in}}$				
Flatten	$a = \mathrm{vec}(x)$	$\frac{\partial L}{\partial x} = reshape(\frac{\partial L}{\partial a})$				

CPU Load

- Depends on the type of layers we want to train
- Is derived from the explicit expressions in Table.
- Operations are performed using floating point numbers, thus we refer to them as FLOPs.

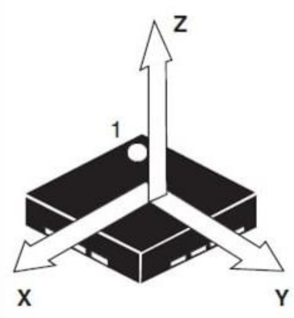
Case Study: Personalization for HAR

Case study: Human Activity Recognition

HAR is a time series classification task which aims to identify the specific movement or action of a person based on inertial sensor data.

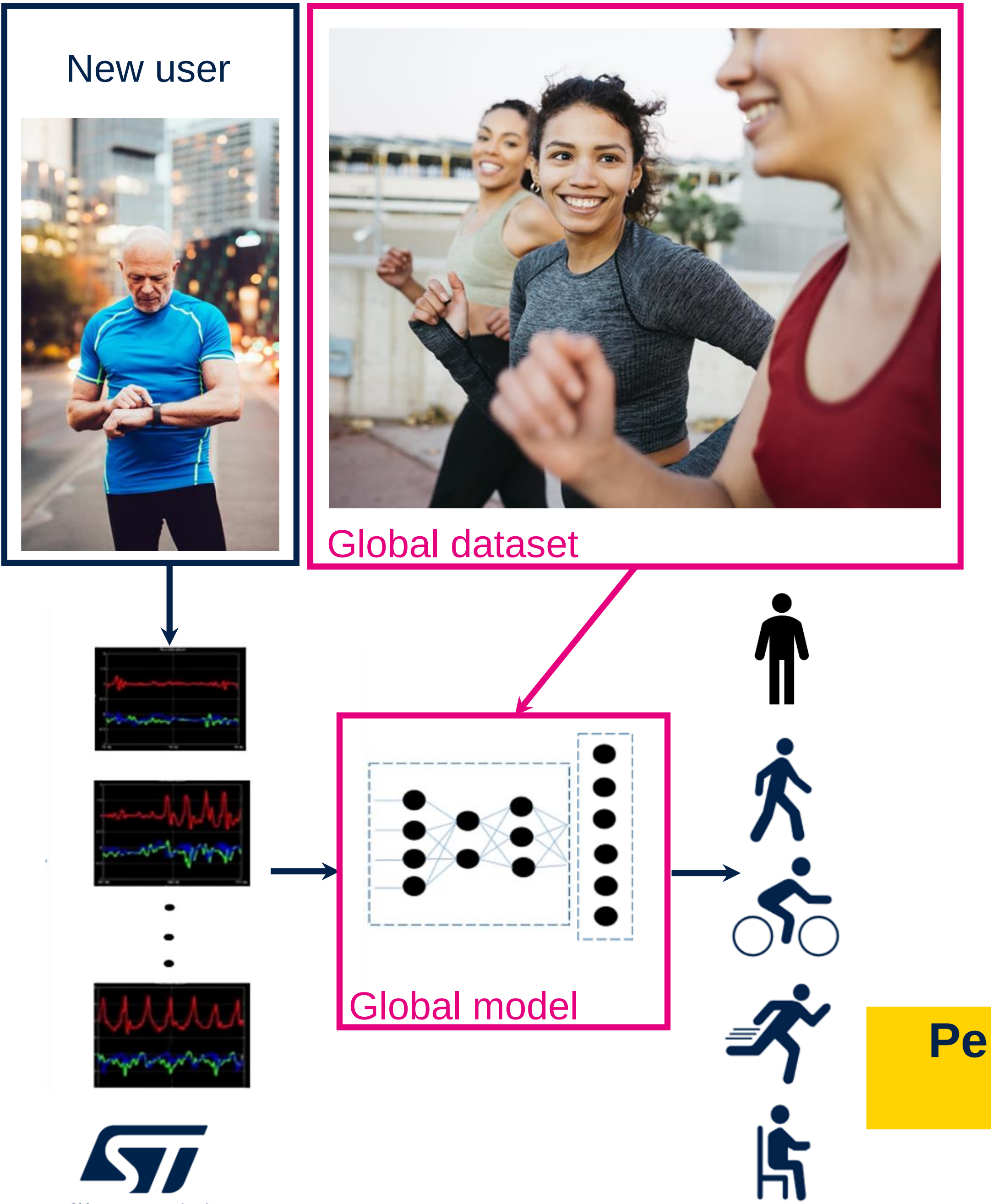
Approach:

- Tri-axial accelerometer data (IMU)
- Activities: standing, sitting, walking, running, biking, …
- Deep learning models



HAR has great potential for daily life tracking, athletic training, healthcare and physiotherapy.

Personalization in HAR



In typical HAR scenarios a vendor trains a global model by recruiting a large numbers of subjects and then delivers it to clients who target real-life applications.

The global model successfully performs HAR assuming the same distribution between subjects and customers' data.

However, there always exist differences in those distributions due to the **heterogeneity of subjects.**
The global model will overfit the global dataset and cannot generalize on data from new users.

Personalize the global model with new user's data is essential to improve the classification accuracy.

Personalization fine-tunes a pre-trained global model using data from new users …**on device**!

Datasets

WISDM: WIreless Sensor Data Mining

https://www.cis.fordham.edu/wisdm/dataset.php

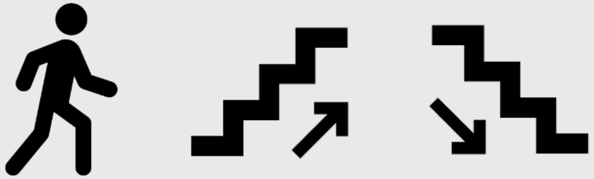
- 36 users, around 30k samples for each user
- 6 activities: walking, jogging, upstairs, downstairs sitting, standing
- Sampling frequency 20 Hz

Used to pretrain a global model

ST dataset

https://github.com/ausilianapoli/HAR-CNN-Keras-STM32

- Collected using SensorTile.box
- 3 users, around 30k samples for each user
- 3 activities: walking, upstairs, downstairs (a subset of those of WISDM)
- Sampling frequency 27 Hz (subsampled at 20 Hz)



Used to represent a domain shift from WISDM dataset

Model Architecture

Layer	Output Shape
Input(20,3)	(20,3)
Conv1D(F=32,K=3)	(18,32)
AvgPool1D(p=2)	(9,32)
Conv1D(F=64,K=3)	(7,64)
AvgPool1D(p=2)	(3,64)
GlobalAvgPool1D()	(64)
Dense(50)	(50)
Dense(6)	(6)

Inputs: from 1s to 5s of recordings from tri-axial accelerometer corresponding to inputs of shape: (20,3),(40,3),(60,3),(80,3),(100,3)

Model architecture: 1D-CNN
1) Conv1D with F = 32 filters and kernel size K = 3 with ReLU, AvgPool1D
2) Conv1D of F = 64 filters and kernel size K = 3 with ReLU, AvgPool1D, GlobalAvgPool1D
3) Dense of M = 50units with ReLU
4) Dense of M = 6 units with Softmax

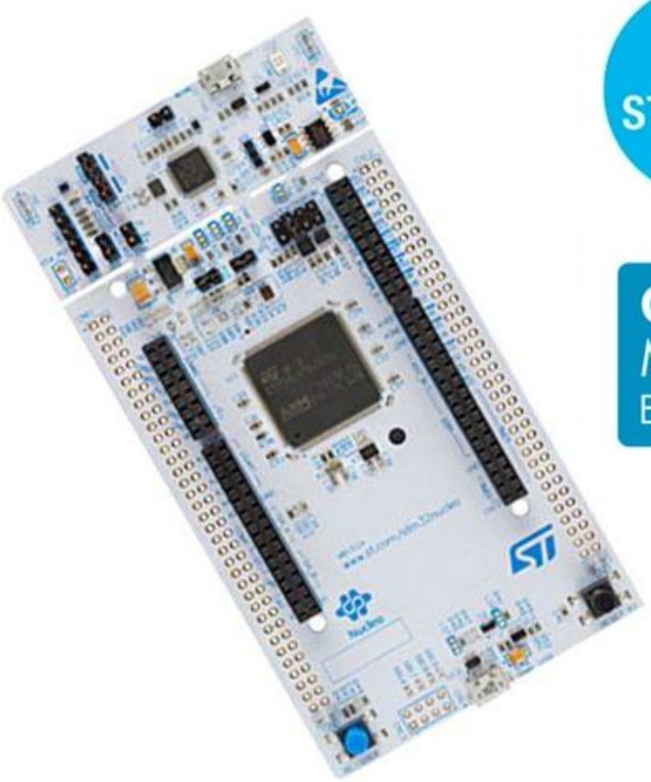
Total number of parameters: around 10k

SGD optimizer, learning rate 0.01 and **batch size 32**
Categorical Cross-Entropy as training loss

life.augmented

Target Board

Nucleo-L496ZG

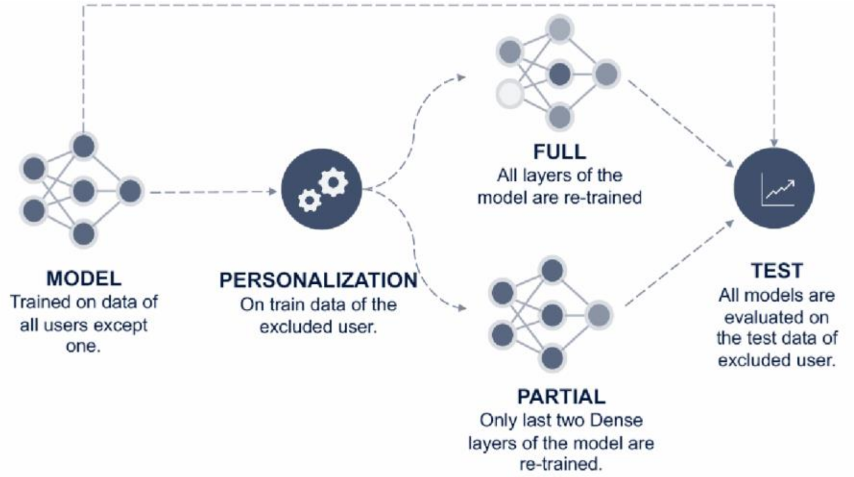


- Ultra low-power MCU STM32L496ZG
- Arm® Cortex®-M4 32-bit RISC core operating at a frequency of up to 80 MHz.
- **320-Kbyte sRAM**.

Firmware is compiled and flashed on the board using STM32CubeIDE.

Experiment 1: WISDM Dataset

Does personalization on user-specific data improve the accuracy of a pretrained model?



Input size	WISDM Dataset No Pers.	TL	Full
(100,3)	0.813	0.955	**0.969**
(80,3)	0.815	0.963	**0.973**
(60,3)	0.819	0.962	**0.978**
(40,3)	0.808	0.958	**0.974**
(20,3)	0.804	0.948	**0.967**

F1 Score (averaged on WISDM users)

Leave-One-Subject-Out (LOSO) Approach

For each user $i = 1, \ldots, 36$:

- We define a **training set** $\boldsymbol{TR_i}$ and **test set** $\boldsymbol{TS_i}$ and we pretrain a **global classifier** $\boldsymbol{C}$ on the other 35 users
- **Personalization strategies**:
  - **Full**: we personalize C_i by retraining all the layers of C using TR_i
  - **Partial (Transfer Learning)**: we personalize C_i by retraining only the last 2 dense layers of C using TR_i
- We **assess** $\boldsymbol{C_i}$ **on** $\boldsymbol{TS_i}$
- We also consider No Pers. as the performance of the global classifier C.

Full personalization reaches the highest F1-score for all the input sizes.

life.augmented

Experiment 2: ST Dataset

Does personalization of a pretrained model outperform a classifier trained only on the target user?

Input size	ST Dataset No Pretrain.	TL	Full
(100,3)	0.936	0.938	**0.960**
(80,3)	0.944	0.939	**0.962**
(60,3)	0.938	0.931	**0.966**
(40,3)	0.951	0.929	**0.965**
(20,3)	0.945	0.911	**0.959**

F1 Score (averaged on ST users)

We consider a global classifier C trained on the WISDM dataset

For each user $i = 1, 2{,}3$ of the ST dataset

- We define a **training set TR_i** and **test set TS_i** and we fine tune the last layer of C using the other 2 users (2 epochs).
- **Personalization strategies**:
  - **Full**: we personalize C_i by retraining all the layers of C using TR_i
  - **Partial (Transfer Learning)**: we personalize C_i by retraining only the last 2 dense layers of C using TR_i
- We **assess C_i on TS_i**
- We train a user specific classifier (No Pretrain) from TR_i starting from a random initialization.

Full personalization reaches the highest F1-score for all the input sizes.
Enabling the retraining of all the network's layers is highly beneficial even when personalization is performed on data from a different dataset, which is common in HAR scenarios.

Estimating Resources

The memory footprint and the time needed for both Full and TL personalization are estimated using the tool we developed.

Input size	Memory Footprint TL	Memory Footprint Full	Time per batch TL	Time per batch Full
(100,3)	115KB	189 KB	14.25 s	48.10 s
(80,3)	102KB	165 KB	11.27 s	38.05 s
(60,3)	91KB	131 KB	8.29 s	28.00 s
(40,3)	79KB	122 KB	5.31 s	17.95 s
(20,3)	63KB	98 KB	2.33 s	7.90 s

Memory Footprint

- Increasing the input size results in larger memory footprint
- All the tested cases are within the memory limitations of our selected device (less than 320 kB)

Time per batch (of 32 samples)

- Increasing the input size results in a larger execution time
- However, an input size of (20, 3) is enough for reaching a very high accuracy

life.augmented

Power Consumption

X-NUCLEO-LPM01A power shield
Measures the current absorbed during the training

Target board
STM32L496ZG

- Consumed power is obtained by multiplying the measured current by the voltage provided (3.3 V);
- Consumed power is practically the same for TL and Full personalization procedures for any input size;
- Energy (power × time) required to process a single batch is higher for the Full personalization, since it scales linearly with the time.

Input size	Time per batch TL	Time per batch Full	Power per batch TL	Power per batch Full
(100,3)	14.25 s	48.10 s	2.67 mW	2.68 mW
(80,3)	11.27 s	38.05 s	2.68 mW	2.69 mW
(60,3)	8.29 s	28.00 s	2.66 mW	2.68 mW
(40,3)	5.31 s	17.95 s	2.64 mW	2.67 mW
(20,3)	2.33 s	7.90 s	2.65 mW	2.65 mW

life.augmented

Conclusions and Future Works

We developed a SW framework to fine-tune and personalize 1D-CNN on STM32 MCUs.

Our experiments on HAR showed that the Full personalization of the model achieves better accuracy than traditional Transfer Learning, although it requires more energy.

Future work concerns extending our framework to support more layers and different optimization strategies.

We are also working on quantized training and on reducing the memory footprint needed to store activations.

life.augmented

Our technology starts with You

beatrice.rossi@st.com
michele.craighero@polimi.it

Find out more at www.st.com

life.augmented

Copyright Notice

www.tinyml.org

Copyright Notice

www.tinyml.org