tinyML Talks

Enabling Ultra-low Power Machine Learning at the Edge

“tinyML Taiwan Community meet up with Gian Marco Iodice”

Gian Marco Iodice – Team and Tech Lead in the Machine Learning Group, Arm

June 9, 2022

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

*as of March 28, 2022; several more under final reviews
Executive Strategic Partners
Arm AI Virtual Tech Talks

The latest in AI trends, technologies & best practices from Arm and our Ecosystem Partners.

Demos, code examples, workshops, panel sessions and much more!

Fortnightly Tuesday @ 4pm GMT/8am PT

Find out more: www.arm.com/techtalks
EDGE IMPULSE  The leading edge ML platform

www.edgeimpulse.com
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

**Perception**
- Object detection, speech recognition, contextual fusion

**Reasoning**
- Scene understanding, language understanding, behavior prediction

**Action**
- Reinforcement learning for decision making

A platform to scale AI across the industry
SYNTIANT

Neural Decision Processors
- At-Memory Compute
- Sustained High MAC Utilization
- Native Neural Network Processing

ML Training Pipeline
- Enables Production Quality Deep Learning Deployments

Data Platform
- Reduces Data Collection Time and Cost
- Increases Model Performance

End-to-End Deep Learning Solutions for TinyML & Edge AI

partners@syntiant.com
www.syntiant.com
Platinum Strategic Partners
Add Advanced Sensing to your Product with Edge AI / TinyML

Pre-built Edge AI sensing modules, plus tools to build your own

Reality AI solutions
- Prebuilt sound recognition models for indoor and outdoor use cases
- Solution for industrial anomaly detection
- Pre-built automotive solution that lets cars “see with sound”

Reality AI Tools® software
- Build prototypes, then turn them into real products
- Explain ML models and relate the function to the physics
- Optimize the hardware, including sensor selection and placement

https://reality.ai   info@reality.ai   @SensorAI   Reality AI
BROAD AND SCALABLE EDGE COMPUTING PORTFOLIO

Microcontrollers & Microprocessors

**Arm® Core**
- Arm® Cortex®-M 32-bit MCUs
  - Arm ecosystem, Advanced security, Intelligent IoT
- Arm®-based High-end 32 & 64-bit MPUs
  - High-resolution HMI, Industrial network & real-time control
- Arm® Cortex®-M0+ Ultra-low Power 32-bit MCUs
  - Innovative process tech (SOTB), Energy harvesting

**Renesas Core**
- Ultra-low Energy 8 & 16-bit MCUs
  - Bluetooth® Low Energy, SubGHz, LoRa®-based Solutions
- High Power Efficiently 32-bit MCUs
  - Motor control, Capacitive touch, Functional safety, GUI
- 40nm/28nm process Automotive 32-bit MCUs
  - Rich functional safety and embedded security features

**Renesas Synergy™**
- Arm®-based 32-bit MCUs for Qualified Platform
- Qualified software and tools

Core technologies

**AI**
A broad set of high-power and energy-efficient embedded processors

**Security & Safety**
Comprehensive technology and support that meet the industry's stringent standards

**Digital & Analog & Power Solution**
Winning Combinations that combine our complementary product portfolios

**Cloud Native**
Cross-platforms working with partners in different verticals and organizations
Gold Strategic Partners
The new MAX78000 implements AI inferences at low energy levels, enabling complex audio and video inferencing to run on small batteries. Now the edge can see and hear like never before.

www.maximintegrated.com/MAX78000

Large (3MB flash + 1MB SRAM) and small (256KB flash + 96KB SRAM, 1.6mm x 1.6mm) Cortex M4 microcontrollers enable algorithms and neural networks to run at wearable power levels.

www.maximintegrated.com/microcontrollers

Health sensors measure PPG and ECG signals critical to understanding vital signs. Signal chain products enable measuring even the most sensitive signals.

www.maximintegrated.com/sensors
Deploy TinyML into the Real World - Plug and Play ML

Sense

- Sensors:
  - modulated and ready-to-use sensors to simplify the setup process
  - support 500+ grove modules

Train

- Codecraft:
  - no code Programming platform to Get Started With TinyML
  - supports Arduino, Python, C or JavaScript etc.
  - Edge Impulse:
    - to optimize data utilization and enable deploy a machine learning model faster than ever

Inference

- Wio Terminal
  - completed AI platform integrated with a 2.4” LCD Screen, onboard IMU (LIS3DHTR), microphone, buzz, microSD card slot, light sensor, infrared emitter (IR 940nm)

Applications

- TensorFlow Lite:
  - to easily train low memory usage machine learning models
Build Smart IoT Sensor Devices From Data

SensiML pioneered TinyML software tools that auto generate AI code for the intelligent edge.

- End-to-end AI workflow
- Multi-user auto-labeling of time-series data
- Code transparency and customization at each step in the pipeline

We enable the creation of production-grade smart sensor devices.

sensiml.com
T I N Y ML TALKS webcast

life.augmented
SynSense builds sensing and inference hardware for ultra-low-power (sub-mW) embedded, mobile and edge devices. We design systems for real-time always-on smart sensing, for audio, vision, IMUs, bio-signals and more.

https://SynSense.ai
Silver Strategic Partners
Our next tinyML Trailblazers Series
Success Stories with Mouna Elkhatib
(CEO, CTO, and Co-Founder, AONDevices Inc.)

LIVE ONLINE June 7th, 2022 at 8 am PDT

Register now!
Join Growing tinyML Communities:

**tinyML - Enabling ultra-low Power ML at the Edge**

**The tinyML Community**
https://www.linkedin.com/groups/13694488/

9.8k members in 45 Groups in 36 Countries

7.8k members & 7.3k followers
Subscribe to tinyML YouTube Channel for updates and notifications (including this video)
www.youtube.com/tinyML
# Next tinyML Talks

<table>
<thead>
<tr>
<th>Date</th>
<th>Presenter</th>
<th>Topic / Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tuesday, June 21</td>
<td>Alexander Timofeev, Founder and Chief Executive Officer, Polyn.ai</td>
<td>The new Neuromorphic Analog Signal Processor (NASP) concept and technology platform</td>
</tr>
</tbody>
</table>

Webcast start time is 8:00 am Pacific time

Please contact talks@tinyml.org if you are interested in presenting
Reminders

Slides & Videos will be posted tomorrow

tinyml.org/publications   youtube.com/tinyml

Please use the Q&A window for your questions
Gian Marco Iodice

Gian Marco Iodice is the team and tech lead in the Machine Learning Group at Arm, for the Arm Compute Library project. Gian Marco was behind the development of the Arm Compute Library from the very beginning, and with several years of experience in the field of development and optimization of machine learning and computer vision on embedded devices, Gian Marco is now leading the ML performance optimization software team on Arm Mali GPUs and Arm Cortex-A CPUs. He received the MSc degree, with honours, in electronic engineering from the University of Pisa (Italy) where he specialized in SW/HW Co-design. In the last few years Gian Marco has been a frequent speaker at Embedded Vision Summit where he presented optimization techniques and design solutions for CNNs. In 2020, Gian Marco co-founded the TinyML UK group with Dominic Binks, Alessandro Grande and Neil Cooper.
Motivation

Demonstrate how easy TinyML is for everyone, even for those with no or little familiarity with embedded programming

Target audience

ML developers/engineers interested in developing ML applications on microcontrollers through practical examples. However, embedded developers who have some basic understanding of ML can also benefit from this book
• TinyML introduction
• Overview of the TinyML ingredients
• TinyML application
TinyML Introduction
Why TinyML

Bring intelligence to objects around us with a focus on power consumption, data privacy, and cost

Make AI ubiquitous for good
What is TinyML

TinyML is the set of technologies in ML and embedded systems to make use of smart applications on extremely low-power devices. Generally, these devices have limited memory and computational capabilities, but they can sense the physical environment through sensors and act based on the decisions taken by ML algorithms.

Ingredients

Characteristics of low-power devices

System input

*TinyML Cookbook, Gian Marco Iodice – April 2022*
What is TinyML

Level of computing on top of sensors that allows smartness **in a minimal intrusive way**

Alasdair Allan, Head of documentation, Raspberry Pi

TinyML Cookbook Release party
20\textsuperscript{th} of April 2020
https://youtu.be/g6UcuCmAgwg

- **Alessandro Grande**, Edge Impulse
- **Gian Marco Iodice**, Arm
- **Allan Alasdair**, Raspberry Pi
- **Massimo Banzi**, Arduino
TinyML finds its natural home wherever a power supply from the mains is impossible or complex to have, and the application must operate with a battery for as long as possible.

Bring intelligence to battery-powered devices.
Battery-powered solutions are not limited to consumer electronics only...

There are scenarios where we might need devices to monitor environments. For example, we may consider deploying battery-powered devices running ML in a forest to detect fires and prevent fires from spreading over a large area.
Centralized versus Distributed Systems

Centralized TinyML applications
No communication with other devices

Smart watches
- Handwriting recognition
- Wake-up words ("Hey, Google!", "Siri"),
- Activity recognition

Smart assistants / Phones
- Wake-up words ("Hey, Google!", "Siri"),
- Activity recognition

Distributed TinyML applications
We may need to communicate with other devices (WSN*)

Forest
- Fire detection
- Endangered species conservation

Agriculture
- Autonomous irrigation system
- Precision/sustainable farming

*Wireless Sensor Network
tinyML Foundation (www.tinyml.org) is a non-profit professional organization supporting and connecting the TinyML world.

To do this, tinyML Foundation is growing a diverse community worldwide (including UK and Italy) between hardware, software, system engineers, scientists, designers, product managers, and businesspeople.

With several Meetup (https://www.meetup.com) groups in different countries, you can join a TinyML one near you for free.
TinyML for Good

Make positive contributions to the United Nations Sustainable Development Goals

https://www.tinyml.org/event/tinyml-for-good/
Overview of the TinyML Ingredients
Fast growing field at the intersection of machine learning (ML) and embedded systems to enable smart applications on extremely low-powered devices.
Why ML on Microcontrollers

- **Popularity**
  - They are everywhere (e.g., automotive, consumer electronics, kitchen appliances, healthcare,...)
  - With the rise of IoT, 28.1 billion microcontrollers sold in 2018! (Note: Smartphone 1.8 billion and 67 million)

- **Inexpensive**
  - From a few cents to a few dollars

- **Easy to program**
  - You can write programs in C (or Python nowadays!)
  - IDE can be FREE and web-based

- **Powerful enough to run ML**
Why Run ML Locally

- **Reducing latency**
  - Sending data back and forth from the Cloud is not instant!
- **Reducing power consumption**
  - Sending/receiving data to and from the Cloud is not power efficient
- **Privacy**
Is an application secure without internet connection?
Not necessarily. What if you have a display that continuously shows sensitive information!
Machine Learning (ML)

ML: it is the study of algorithms capable to learn automatically from experience and data how to behave

Neural Network is a common ML algorithm:

Input:
- Image
- Sound
- Accelerometer
- ...

Output:
- Image type, Object location
- Sound type,
- Action
- ....

Note: Obtained after training process and typically constant
The model is only as good as the data used for training.

Training step:
- Data
- Neural Network
- Weights
- Update

Deployment step:
- Neural Network
- Weights
  - Typically, constant
From feature engineering to convolutional neural networks (CNN)
The Deep Learning Era

Old ML fashion

It could be tailored in specific situations (e.g., sound recognition, gesture recognition)
A microcontroller is a full-fledged computer because it has a processor (CPU), a memory system (for example, RAM or ROM), and some peripherals.
However, how does it differ from a microprocessor?
Answer

Microprocessor

Microcontroller
The target applications influence their architectural design choices

<table>
<thead>
<tr>
<th>Microprocessor</th>
<th>Microcontroller</th>
</tr>
</thead>
<tbody>
<tr>
<td>• <strong>Dynamic</strong> (for example, can change with user interaction or time)</td>
<td>• <strong>Tasks are single purpose and repetitive</strong></td>
</tr>
<tr>
<td>• <strong>General-purpose</strong></td>
<td>• The device does not require strict re-programmability</td>
</tr>
<tr>
<td>• <strong>Compute intensive</strong></td>
<td>• <strong>Tasks may have real-time constraints</strong></td>
</tr>
<tr>
<td></td>
<td>• The device must be latency predictable</td>
</tr>
<tr>
<td></td>
<td>• <strong>Tasks may be battery-powered</strong></td>
</tr>
<tr>
<td></td>
<td>• RAM and ROM must be on-chip</td>
</tr>
<tr>
<td></td>
<td>• Low clock frequency</td>
</tr>
<tr>
<td></td>
<td>• Reduced computational capabilities (e.g., just integer arithmetic)</td>
</tr>
</tbody>
</table>
## Microprocessor Vs Microcontroller – 2of2

<table>
<thead>
<tr>
<th>Feature</th>
<th>Microprocessor</th>
<th>Microcontroller</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td>General-purpose</td>
<td>Single-purpose</td>
</tr>
<tr>
<td>CPU arithmetic</td>
<td>It can perform heavy mathematical calculations in floating-point or double precision</td>
<td>Mainly integer arithmetic</td>
</tr>
<tr>
<td>RAM</td>
<td>A few GB</td>
<td>A few hundred KB</td>
</tr>
<tr>
<td>ROM (or hard-drive)</td>
<td>GB or TB</td>
<td>KB or MB</td>
</tr>
<tr>
<td>Clock frequency</td>
<td>GHz</td>
<td>MHz</td>
</tr>
<tr>
<td>Power consumption</td>
<td>W</td>
<td>mW or below</td>
</tr>
<tr>
<td>Operating System (OS)</td>
<td>Required</td>
<td>Not strictly required</td>
</tr>
<tr>
<td>Cost</td>
<td>From ten to hundreds of dollars</td>
<td>From a few cents (low-end) to a few dollars (high-end)</td>
</tr>
</tbody>
</table>
In the microcontroller context, we physically dedicate two separate memories for the instructions (program) and data:

- **Program memory (ROM)**
  - Non-volatile read-only memory reserved for the program to execute
  - It can store **CONSTANT data**

- **Data memory (RAM/SRAM)**
  - Volatile memory reserved to store/read temporary data
Where can we store the weights of our ML model?
Answer

Depends on whether the model has constant weights. *If the weights are constant, so do not change during inference, it is more efficient to store them in program memory* for the following reasons:

- Program memory has more capacity than SRAM

- *It reduces memory pressure on the SRAM* since other functions require storing variables or chunks of memory at runtime
Suppose you have a processing task and you have the option to execute it on two different processors. These processors have the following power consumptions:

<table>
<thead>
<tr>
<th>Processing unit</th>
<th>Power consumption</th>
</tr>
</thead>
<tbody>
<tr>
<td>PU1</td>
<td>12</td>
</tr>
<tr>
<td>PU2</td>
<td>3</td>
</tr>
</tbody>
</table>

What processor would you use to execute the task?
Although PU1 has higher (4x) power consumption than PU2, this does not imply that PU1 is less energy-efficient. On the contrary, PU1 could be more computationally performant than PU2 (for example, 8x), making it the best choice from an energy perspective, as shown in the following formulas:

\[ E_{PU1} = 12 \cdot T_1 \]

\[ E_{PU2} = 3 \cdot T_2 = 3 \cdot 8 \cdot T_1 = 24 \cdot T_1 \]
How it Looks a TinyML Program

**Pseudo C-code**

```c
load_model();          // We load the ML to execute
initialize_model_memory();    // Allocate the memory required by the model
data = acquire_data();       // Acquire the data from the sensor/s
input = prepare_input(data);  // Prepare the input required by the model
output = run_model(input);    // Execute the model
```

// Code to interpret the output result
TinyML Resources

- [Book] TinyML, Pete Warden & Daniel Situnayake
- [Book] TinyML Cookbook, Gian Marco Iodice

- https://tinyml.seas.harvard.edu/teach/
Copyright Notice

This multimedia file is copyright © 2022 by tinyML Foundation. All rights reserved. It may not be duplicated or distributed in any form without prior written approval.

tinyML® is a registered trademark of the tinyML Foundation.

www.tinyml.org
Copyright Notice

This presentation in this publication was presented as a tinyML® Talks webcast. The content reflects the opinion of the author(s) and their respective companies. The inclusion of presentations in this publication does not constitute an endorsement by tinyML Foundation or the sponsors.

There is no copyright protection claimed by this publication. However, each presentation is the work of the authors and their respective companies and may contain copyrighted material. As such, it is strongly encouraged that any use reflect proper acknowledgement to the appropriate source. Any questions regarding the use of any materials presented should be directed to the author(s) or their companies.

tinyML is a registered trademark of the tinyML Foundation.

www.tinyml.org