Is On-Device Learning the Next Big Thing in TinyML?
Prof. Manuel Roveri, PhD

- **Associate Professor**
  Dipartimento di Elettronica, Informazione e Bioingegneria (DEIB)
  Politecnico di Milano, Italy
  Email: manuel.roveri@polimi.it
  Web: http://roveri.faculty.polimi.it

- **Research interests**: machine/deep learning, TinyML, IoT and edge computing, learning in nonstationary environments, privacy preserving computation.

- **Lecturer of the Course »Hardware Architectures for Embedded and Edge AI» @ POLIMI MSc - Spring 2023**

- **Associate Editor** of IEEE Trans. on Artificial Intelligence, Neural Networks, Trans. on Emerging Technologies in Computational Intelligence

- Chair of the IEEE CIS **Technical Activities** strategic planning committee, Chair of the IEEE CIS **Neural Network** Technical Committee
The research team (a big thank to)

- Cesare Alippi (Full Professor)
- Massimo Pavan (PhD Student)
- Alessandro Falcetta (PhD Student)
- Gabriele Viscardi (Research Assistant)
- Matteo Gambella (Research Assistant)
- Diego Riva (Research Assistant)
- Simone Disabato (Former Post-doc researcher)
- Giuseppe Canonaco (Former Post-doc researcher)
- ...
Let’s start with a provocative question …

Is On-Device Learning the Next Big Thing in TinyML?
Let’s start with a provocative question ...

Is On-Device Learning the Next Big Thing in TinyML?
The best performer, Google’s FaceNet

Ira Kemelmacher-Shlizerman, an associate professor of computer science at the University of Washington, in Seattle, and his team at Google’s Seattle office were presented at the IEEE Conference on Computer Vision and Pattern Recognition in June. (One of MegaFace’s developers also heads a computer vision team at Google’s Seattle office.)

The results presented were a mix of the intriguing and the expected. Nobody was surprised that the algorithms performed as well as they did on distractor faces increased. And the fact that algorithms had trouble identifying the same person at different ages was a known problem. However, the results also showed that algorithms trained on relatively small data sets can compete with those trained on very large ones, such as Google’s FaceNet, which was trained on more than 500 million photos of 10 million people.

For example, the FaceN algorithm from Russia’s NtechLab performed well on certain tasks in comparison with FaceNet, despite having trained on 18 million photos of 200,000 people. The SLAT MMLab algorithm, created by a Chinese team under the leadership of Yu Qiao, a professor with Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, also performed well on certain tasks.

Nevertheless, FaceNet has so far performed the best overall. It delivered the most consistent performance across all testing.

The huge drops in accuracy when scanning a million faces matter because facial recognition algorithms inevitably face such challenges in the real world. People increasingly trust these algorithms to correctly identify them in security verification scenarios, and law enforcement may also rely on facial recognition to pick suspects out of the hundreds of thousands of faces captured on surveillance cameras.

IEEE Spectrum
Aug, 2016

With that in mind, University of Washington researchers raised the bar by creating the MegaFace Challenge using 1 million Flickr images of 650,000 unique faces that are publicly available under a Creative Commons license.

The MegaFace Challenge forces facial recognition algorithms to do verification and identification, two separate but related tasks. Verification involves trying to correctly determine whether two faces presented to the facial recognition algorithm belong to the same person. Identification involves trying to find a matching photo of the same person among a million “distractor” faces. Initial results on algorithms developed by Google and four other research groups were presented at the IEEE Conference on Computer Vision and Pattern Recognition on 30 June. (One of MegaFace’s developers also heads a computer vision team at Google’s Seattle office.)

The results presented were a mix of the intriguing and the expected. Nobody was surprised that the algorithms performed as well as they did on distractor faces increased. And the fact that algorithms had trouble identifying the same person at different ages was a known problem. However, the results also showed that algorithms trained on relatively small data sets can compete with those trained on very large ones, such as Google’s FaceNet, which was trained on more than 500 million photos of 10 million people.

For example, the FaceN algorithm from Russia’s NtechLab performed well on certain tasks in comparison with FaceNet, despite having trained on 18 million photos of 200,000 people. The SLAT MMLab algorithm, created by a Chinese team under the leadership of Yu Qiao, a professor with Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, also performed well on certain tasks.

Nevertheless, FaceNet has so far performed the best overall. It delivered the most consistent performance across all testing.
TinyML: an example in smart environmental monitoring

A rock-collapse forecasting system based on intelligent IoT and Edge Units (Northern Italy)
TinyML: the “always-on” Machine Learning

Wake-word detection

Person detection

Gesture recognition

Sensors

Preprocessing

Input (x)

Machine and Deep Learning Inference

Output (y)

Actuators

Postprocessing

Software

Hardware

Software

Hardware
TinyML: the “always-on” Machine Learning

1) Changes in the environments

- Wake-word detection
- Person detection
- Gesture recognition

Diagram:
- Sensors
- Preprocessing
- Machine and Deep Learning Inference
- Postprocessing
- Actuators
- Hardware

Software
TinyML: the “always-on” Machine Learning

1) Changes in the environments
2) Changes in the users’ behaviour/interest

Wake-word detection
Person detection
Gesture recognition

Sensors
Preprocessing
Machine and Deep Learning Inference
Postprocessing
Actuators

Hardware
Software

Prof. Manuel Roveri

tinyML On Device Learning Forum 2022
TinyML: the “always-on” Machine Learning

1) Changes in the environments
2) Changes in the users’ behaviour/interest
3) Faults/malfunctioning affecting the hardware or sensors/actuators
TinyML: the “always-on” Machine Learning

1) Changes in the environments
2) Changes in the users’ behaviour/interest
3) Faults/malfunctioning affecting the hardware or sensors/actuators
4) Ageing effects or thermal drifts affecting the sensors

- Wake-word detection
- Person detection
- Gesture recognition

Diagram: Flowchart showing the interaction between hardware, sensors, actuators, preprocessing, and postprocessing.
TinyML: the “always-on” Machine Learning

Perturbed, incorrect and missing data can hence heavily affect the subsequent processing phase so as to possibly induce wrong decisions or on-the-field reactions.
Two (integrated) challenges to be addressed

- Learning in presence of concept drift
- Efficient on-device learning
Two (integrated) challenges to be addressed

- Learning in presence of concept drift
- Efficient on-device learning
Adaptation is the game changer

- Traditional assumption: stationarity hypothesis
- Adaptive solutions in a non-stationary framework:
Adaptation is the game changer

- Traditional assumption: stationarity hypothesis
- Adaptive solutions in a non-stationary framework:

The TinyML perspective
Passive vs Active approaches

Passive

Active

Environment

Sensors

Adaptation

Application / Service

User

Environment

Adaptation

Detection

Application / Service

Sensors

User
The just-in-time adaptive framework

Application

Adaptation

Detection, Information about concept drift

Paraphrase:

Nominal Concept
Nominal Concept
Nominal Concept
Reference Concept

Concept Drift Detection

Feature Extraction

Time occurrence

Learning Phase

Operational Phase

Update

Software:

Concept Library

Sensor 1
Sensor 2
Sensor 3

Samples

Temperature (°C)

0 10 20 30 40

0 2000 4000 6000 8000

0 10 20 30 40

0 2000 4000 6000 8000

Samples

Feature Extraction

Prof. Manuel Roveri

tinyML On Device Learning Forum 2022
Two (integrated) challenges to be addressed

- Learning in presence of concept drift
- Efficient on-device learning
On-device TinyML for Concept Drift

Sensors → Preprocessing → Machine and Deep Learning Inference → Postprocessing → Actuators

Hardware

Software

Feature Extractor $\varphi(\cdot)$

Dimensionality Reduction $\varsigma(\cdot)$

$k$-NN $\mathcal{K}(\cdot)$

$\mathcal{K}$'s Training Set $\mathcal{T}$

Adaptation

$\hat{y}_t = \mathcal{K} \circ \varsigma \circ \varphi(x_t)$
On-device TinyML for Concept Drift

Mathematical model:

\[ \hat{y}_t = \mathcal{K} \circ \varsigma \circ \varphi(x_t) \]
On-device TinyML for Concept Drift

- The Feature Extractor extracts features from the input $x_t$
- A pre-trained DL model (transfer learning) approximated by means of:
  - Task-Dropping
  - Precision Scaling
On-device TinyML for Concept Drift

- Reduces the dimensionality of extracted features (e.g., filter-selection mechanisms)
On-device TinyML for Concept Drift

- $k$NN classifier
- Pros: It does not require a training phase
- Cons: Memory demand and computational complexity
On-device TinyML for Concept Drift

- **kNN classifier**
- **Pros:** It does not require a training phase
- **Cons:** Memory demand and computational complexity

Condensing and Editing Algorithms for kNN
Identifying the smallest subset of training data that can correctly classify all the training sample
On-device TinyML for Concept Drift

- The adaptation module receives as input the sample $x$ and the $k$NN prediction $y$.
- When the supervised information $y$, is available, it updates the knowledge base of $k$NN.

\[ \hat{y}_t = \mathcal{K} \circ \varsigma \circ \varphi(x_t) \]
On-device TinyML for Concept Drift

- **Passive Update**: the adaptation is carried out at each new supervised sample
- **Active Update**: the adaptation is triggered by a Change-detection test
- **Hybrid Update**: integrates passive and active update
Passive update: condensing-in-time

- The adaptation is carried out at each new supervised sample

---

**Algorithm 3:** The Condensing-in-Time (Passive).

**Input:** Training Set $\mathcal{T}$, Feature Extractor $\varsigma \circ \varrho$.

**Parameters:** Maximum number of training samples $p$.

1. Compute $\mathcal{T} \leftarrow \tilde{\mathcal{T}}$ with Algorithm 2. ▷ Condense $\mathcal{T}$.
2. Initialize the $k$-NN classifier $\mathcal{K}$ with $\varsigma \circ \varrho(\mathcal{T})$.
3. Define $\mathcal{D} = \mathcal{K} \circ \varsigma \circ \varrho$.
   ▷ Loop over samples arriving at time $t$.
4. **foreach** $(x_t, y_t) \sim \mathcal{P}, t = 1, 2, \ldots$ **do**
   5. Predict $\hat{y}_t \leftarrow \mathcal{D}(x_t)$ ▷ Passive Update.
   6. **if** $\hat{y}_t \neq y_t$ **then** ▷ Window Size Check.
   7. \[ \mathcal{T} \leftarrow \mathcal{T} \cup \{(x_t, y_t)\} \]
   8. **if** $|\mathcal{T}| > p$ **then**
      \[ (x_i, y_i) \leftarrow \arg \min_{(x_i, y_i) \in \mathcal{T}} \{(x_i, y_i) \in \mathcal{T}\} \]
      \[ \mathcal{T} \leftarrow \mathcal{T} \setminus \{(x_i, y_i)\} \]
   9. Update $\mathcal{D}$ with $\mathcal{T}$.

\[ \hat{y}_t = \mathcal{K} \circ \varsigma \circ \varrho(x_t) \]
Active update: Active Tiny $k$NN

- The adaptation is triggered by a CUSUM-based change-detection mechanism

\[
s_t = \ln \frac{p_{\nu_1}(\zeta_t)}{p_{\nu_0}(\zeta_t)}, \quad S_j^t(v_1) = \sum_{i=j}^{t} s_i,
\]
\[
g_t = \vartheta(s_1, \ldots, s_t) = \max_{1 \leq j \leq t} \sup_{v_1 \in \mathcal{Y}_1} S_j^t(v_1),
\]

Estimation of the change time instant

\[
(\tilde{j}, \tilde{v}_1) = \arg\max_{1 \leq j \leq t} \sup_{v_1 \in \mathcal{Y}_1} S_j^t(v_1).
\]
Hybrid update

- Integrating passive and active updates

- The hybrid update continuously adapts $\mathcal{T}$ over time thanks to the passive adaptation.

- The active adaptation present in the hybrid update can quickly discard obsolete knowledge when a change is detected and set a bound on the memory footprint of $\mathcal{T}$.

\[ \hat{y}_t = \mathcal{K} \circ \varsigma \circ \varphi(x_t) \]
Some relevant results (Speech Command Identification)

Concept drift refers to the introduction of noise (distortion, reverb, and echoes)

<table>
<thead>
<tr>
<th>MCU</th>
<th>$t_p$</th>
<th>$t_{\xi\eta}$</th>
<th>$t_{k,10}$</th>
<th>$t_{k,50}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>STM32H743ZI</td>
<td>22.9</td>
<td>18.6</td>
<td>2.0</td>
<td>2.9</td>
</tr>
<tr>
<td>STM32F767ZI</td>
<td>53.8</td>
<td>41.5</td>
<td>2.2</td>
<td>4.0</td>
</tr>
<tr>
<td>STM32F401RE</td>
<td>34.6</td>
<td>43.1</td>
<td>2.2</td>
<td>4.6</td>
</tr>
</tbody>
</table>

Spectrogram Processing  Feature Extraction  KNN classification with 10 and 50 samples
Two current research directions

- Semi Supervised TinyML
- Automatic design, development and deployment for On-line Tiny Machine Learning
Thank you for the attention!
Copyright Notice

This presentation in this publication was presented at the tinyML® On Device Learning Forum 2022. 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.