“Suitability of Forward-Forward and PEPITA Learning to MLCommons-Tiny benchmarks”

Danilo Pau – Technical Director, IEEE, AAIA & ST Fellow STMicroelectronics

September 19, 2023
Thank you, tinyML Strategic Partners, for committing to take tinyML to the next Level, together.
Executive Strategic Partners
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

Qualcomm AI Research is an initiative of Qualcomm Technologies, Inc.
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
Witness potential made possible at analog.com.
Build the Future of tinyML on Arm
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
NEUROMORPHIC INTELLIGENCE FOR THE SENSOR-EDGE
Renesas is enabling the next generation of AI-powered solutions that will revolutionize every industry sector.
STMicroelectronics provides extensive solutions to make tiny Machine Learning easy
ENGINEERING
EXCEPTIONAL
EXPERIENCES

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

www.synaptics.com
Silver Strategic Partners
Join Growing tinyML Communities:

tinyML - Enabling ultra-low Power ML at the Edge

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

16.8k members in 49 Groups in 41 Countries

4k members & 13k followers
Subscribe to tinyML YouTube Channel for updates and notifications (including this video)
www.youtube.com/tinyML
tinyML Asia
Technical Forum

November 16, 2023
Seoul, South Korea

Register now
https://www.tinyml.org/event/asia-2023/
2023 Edge AI Technology Report

The guide to understanding the state of the art in hardware & software in Edge AI.

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

Slides & Videos will be posted tomorrow

tinyml.org/forums   youtube.com/tinyml

Please use the Q&A window for your questions
Danilo Pau

Danilo PAU (h-index 26, i10-index 69) graduated in 1992 at Politecnico di Milano, Italy. One year before his graduation, he joined SGS-THOMSONS (now STMicroelectronics) as interns on Advanced Multimedia Architectures, and he worked on memory reduced HDMAC HW design. Then MPEG2 video memory reduction. Next, on video coding, transcoding, embedded 2/3D graphics, and computer vision. Currently, his work focuses on developing solutions for tiny machine learning tools.

Since 2019 Danilo is an IEEE Fellow and AAIA on 2023; he served as Industry Ambassador coordinator for IEEE Region 8 South Europe, was vice-chairman of the “Intelligent Cyber-Physical Systems” Task Force within IEEE CIS, was IEEE R8 AfI member in charge of internship initiative. Today he is a Member of the Machine Learning, Deep Learning and AI in the CE (MDA) Technical Stream Committee CESoc. He was AE of IEEE TNNLS. He wrote the IEEE Milestone on Multiple Silicon Technologies on a chip, 1985 which was ratified by IEEE BoD in 2021 and IEEE Milestone on MPEG Multimedia Integrated Circuits, 1984-1993 which was ratified in 2022. He served as TPC member to TinyML EMEA forum and is the chair of the TinyML on Device Learning working group. He serves as 2023 IEEE Computer Society Fellow Evaluating Committee Members

With 78 and 68 respectively European and US application patents, 157 publications, 113 ISO/IEC/MPEG authored documents and 67 invited talks/seminars at various Universities and Conferences, Danilo’s favorite activity remains supervising undergraduate students, MSc engineers and PhDs.
Fabrizio Aymone

Fabrizio M. Aymone is currently pursuing a Bachelor degree in Electronics Engineering at Politecnico di Milano. He is also intern at the System Research and Applications department of STMicroelectronics, where he is studying solutions for On-Device Learning in the domain of tiny devices. His research interests focus on reducing memory usage and computational complexity of AI algorithms and exploring alternative learning rules to backpropagation.
Suitability of Forward-Forward and PEPITA Learning to MLCommons-Tiny benchmarks

Danilo P. Pau and Fabrizio M. Aymone

Sept, 19 2023
1. Back-propagation
2. On-Device Learning
3. Forward-Forward and PEPITA
4. The research question
5. Methodology and Results
6. Takeways
7. Future works
8. Q&A
Reduce activations, not trainable parameters for efficient on-device learning

Reduce activations, not trainable parameters for efficient on-device learning

Backpropagation

- The loss is calculated w.r.t. the ground truth and the final output;
- The gradient of the loss is then computed;
- Then, the derivative of the output activations $z_l$;
- Finally, the derivative of the loss function for the output layer is computed as Hadamard product.

Activations are computed for each layer and stored into memory.

In reverse pipeline order, the derivative of the loss function of the previous layer is computed as Hadamard product of the derivative of the previous step is multiplied to the input activations of the previous layer $a_{l-1}$ to compute the variation of the weights that are added to update the weights of the previous layer.

---

Algorithm 1 Backpropagation

**Forward Pass**

```
for $i = 1, ..., L$ do
    $a_i = \sigma_i(W_i a_{i-1} + b_i)
end for
```

**Backward pass layer L**

```
\delta_L = \nabla_{a_L} L(a_L, target) \odot \sigma'(z_L)
W_L = W_L - \delta_L a^T_{\text{previous}}
```

**Backward pass previous layers**

```
for $l = \text{previous}, ...$ do
    $\delta_l = W_{l+1}^T \delta_{l+1} \odot \sigma'(z_l)$
    Weight update
    $W_l = W_l - \delta_l a^T_{l-1}$
end for
```

GT = ground truth
NL = non-linearity
L = Loss
Backpropagation
Backpropagation
Sparse Update: 
- updates only **some** parameters of the model.
- these parameters are selected **offline** according to how much they contribute to reduce the error during training.
- it needs to store only the *intermediate activations* of such parameters.


\[ 335\text{ KB} \xrightarrow{8.8x} \quad (\text{Sec } 2.2) \]
<table>
<thead>
<tr>
<th>Application</th>
<th>Hardware accelerator</th>
<th>Acronym</th>
</tr>
</thead>
</table>
| Computer graphics                               | • General-purpose computing on GPU  
• CUDA architecture  
• Ray-tracing hardware                  | •GPGPU  
• CUDA  
• RTX |
| Digital signal processing                       | Digital signal processor                                                                | DSP     |
| Analog signal processing                        | • Field-programmable analog array  
• Field-programmable RF                  | • FPAAFPRF |
| Sound processing                                | Sound card and sound card mixer                                                        | N/A     |
| Computer networking on a chip                   | • Network processor and network interface controller Network on a chip               | NPU and NICNoC |
| Cryptography Encryption                         | • Cryptographic accelerator and secure hardware-based encryption  
• Custom hardware attack  
• Hardware random number generator       | N/A     |
| Attack                                           |                                                                                       |         |
| Random number generation                        |                                                                                       |         |
| Artificial intelligence/Machine vision/computer vision | • AI accelerator  
• Vision processing unit  
• Physical neural network  
• Neuromorphic engineering               | • N/A/VPU  
• PNN  
• N/A |
| Neural networks                                 |                                                                                       |         |
| Brain simulation                                |                                                                                       |         |
| Multilinear algebra                             | Tensor processing unit                                                                 | TPU     |
| Physics simulation                              | Physics processing unit                                                                | PPU     |
| Regular expressions[16]                         | Regular expression coprocessor                                                        | N/A     |
| Data compression[17]                            | Data compression accelerator                                                          | N/A     |
| In-memory processing                            | Network on a chip and Systolic array                                                  | NoC; N/A |
| Data processing                                 | Data processing unit                                                                  | DPU     |
| Any computing task                              | • Computer hardware Field-programmable gate arrays[18]  
• Application-specific integrated circuits[18]  
• Complex programmable logic devices  
• Systems-on-Chip                           | • HW (sometimes)FPGA  
• ASIC  
• CPLD  
• SoC |

* https://en.wikipedia.org/wiki/Hardware_acceleration

**Hardware acceleration**

Hardware acceleration is the use of computer hardware designed to perform specific functions more efficiently when compared to software running on a general-purpose central processing unit (CPU).
Edge AI drives hardware for on-device machine learning inference.

Tiny ML hardware (today) is optimized for forward workloads.

Forward-Forward and PEPITA
Learning: biological plausibility

- The brain **learns** by modifying the synaptic individual connections between neurons\(^3\)

- It’s **not known how** the single modifications are coordinated to achieve a global’s goal

- **Loop-based** neuron circuits seems used to get error signals and credits (i.e. how much each synapse contributes to the error) assigned to other synapsis of neurons

---

Our brain does not use backpropagation\textsuperscript{4}

Our brain does not use backpropagation\(^4\)

1. No weight symmetry
   a) error is not projected back using the same weights of the forward pass

Our brain does not use backpropagation\(^4\)

1. No weight symmetry
2. No neural activity freeze
   a) intermediate activations are not stored

Backpropagation vs bio-plausibility?

Our brain does not use backpropagation\(^4\)

1. No weight symmetry
2. No neural activity freeze
3. No locality of the loss function
   a) Neurons do exchange error signals and credits within loops
   b) Synapsis learn from local signals

Backpropagation vs bio-plausibility?

Our brain does not use backpropagation\(^4\)

1. No weight symmetry
2. No neural activity freeze
3. No locality of the loss function
4. No Update-locking
   a) No need to wait the end of the backward pass to update the weights of the layers
   b) There is no backward pass

## A lot of work in 30 years…

<table>
<thead>
<tr>
<th>Biologically plausible method</th>
<th>Citation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Biologically plausible method</td>
<td>Citation</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>----------</td>
</tr>
</tbody>
</table>

Positive weight update of each layer independently from the next ones (locality)

Negative weight update of each layer independently from the next ones (locality)
2 variants: supervised and unsupervised → same learning procedure, different forward procedures

Supervised variant: \( n \) passes, for \( n \) classes

- The **goodness function** \( G(a_l) \) is defined at each layer; e.g. it can be the sum of the squared \( a_l \) activations

- The probability of being classified as a positive data is

\[
p(\text{positive}) = \sigma(G(a_L) - \theta)
\]

- Then maximize the \( \log(p) \) for positive data and minimize it for negative

---

Algorithm 3 PEPITA

Given: Features($x$) and label(target)

Standard Pass

$$a_0 = x$$
for $\ell = 1, ..., L$ do
  $$a_\ell = \sigma_\ell(W_\ell a_{\ell-1})$$
end for
$$e = a_L - \text{target}$$

Modulated pass

$$a_0^{err} = x + F e$$
for $\ell = 1, ..., L$ do
  $$a_\ell^{err} = \sigma_\ell(W_\ell a_{\ell-1}^{err})$$
  Weight update
  $$W_\ell = W_\ell - (a_\ell - a_\ell^{err}) \cdot (a_{\ell-1}^{err})^T$$
end for

Compute the error at the output

Random matrix F, with zero mean and small variance, to project the error on the inputs

Weights update by using the forward pass computed activations (stored in memory #) with the modulated ones

---

\[ W_\ell = W_\ell - (a_\ell - a_\ell^{err}) \cdot (a_{\ell-1}^{err})^T \]

Standard pass

Modulated pass

PEPITA
Research question

At which computational and memory cost would FF and PEPITA learning algorithms compare to BP if applied to MLCommons/Tiny benchmarks?
Contributions today

Introducing **MEMPEPITA** to not store intermediate activations. Memory savings expected!

Computational complexity and memory footprint of FF, PEPITA and MEMPEPITA for the MLCommons/Tiny benchmarks.
MEMPEPITA

Algorithm 2 MEMPEPITA
Given: Features($x$) and label($target$)

Standard Pass

\[ a_0 = x \]
\[ \text{for } \ell = 1, \ldots, L \text{ do} \]
\[ a_\ell = \sigma_\ell(W_\ell a_{\ell-1}) \]
\[ \text{end for} \]
\[ e = a_L - target \]

Error projection

\[ a_0^{err} = x + Fe \]
\[ \text{for } \ell = 1, \ldots, L \text{ do} \]
\[ \text{Standard pass} \]
\[ a_\ell = \sigma_\ell(W_\ell a_{\ell-1}) \]
\[ \text{Modulated pass} \]
\[ a_\ell^{err} = \sigma_\ell(W_\ell a_{\ell-1}^{err}) \]
\[ \text{Weight update} \]
\[ W_\ell = W_\ell - (a_\ell - a_\ell^{err}) \cdot (a_{\ell-1}^{err})^T \]
\[ \text{end for} \]

Forward inference

Forward inferences thus recomputing activations instead of storing them into memory

Compute the error at the output

Random matrix $F$, with zero mean and small variance, to project the error onto the inputs

Introduces a second standard pass which runs simultaneously along with modulated pass

Weights update by using the forward pass re-computed activations with the modulated ones
MEMPEPITA

Standard Pass

Error projection

Standard Pass

Modulated Pass

Parameters Update
## Summary of the **learning** procedures

<table>
<thead>
<tr>
<th>Method</th>
<th>BP (number)</th>
<th>FF (number)</th>
<th>PEP (number)</th>
<th>MPE (number)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Forward passes</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>Backward passes</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Weight update</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Loss function</td>
<td>Global</td>
<td>Local</td>
<td>Global</td>
<td>Global</td>
</tr>
<tr>
<td>Activations</td>
<td>all</td>
<td>current</td>
<td>all</td>
<td>current</td>
</tr>
</tbody>
</table>

**PEP** = PEPITA  
**MPE** = MEMPEPITA  
*Local* = loss function per layer  
*Global* = loss function at the output layer  
*All* = all layers  
*Current* = current layer
Methodology
<table>
<thead>
<tr>
<th>Use Case</th>
<th>Description</th>
<th>Dataset</th>
<th>Model</th>
<th>Quality Target (Closed Division)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Audio Wake Words</td>
<td>Small vocabulary keyword spotting</td>
<td>Speech Commands</td>
<td>DS-CNN</td>
<td>90% (Top1)</td>
</tr>
<tr>
<td>Visual Wake Words</td>
<td>Image classification (2 classes)</td>
<td>Person Detection</td>
<td>MobileNet</td>
<td>80% (Top1)</td>
</tr>
<tr>
<td>Image Classification</td>
<td>32x32 tiny Images Classification (10 classes)</td>
<td>Cifar10</td>
<td>ResNet</td>
<td>85% (Top1)</td>
</tr>
<tr>
<td>Anomaly Detection</td>
<td>Detecting anomalies in machine operating sounds</td>
<td>ToyADMOS</td>
<td>Deep AutoEncoder</td>
<td>0.85 (AUC)</td>
</tr>
</tbody>
</table>
ML Commons framework specifies number of samples and epochs
- Weights, biases, activations represented in INT8
- Softmax layer represented in FLOAT32
- MACCs represented in INT8
- No layer memory overwrite
- Batch normalization not considered.
- Cycles/MACC and processor’s frequency
- Results validated with STM32Cube.AI Developer Cloud

STM32H735G-DK 550MHz
STM32G474RE-NUCLEO 170MHz
https://stm32ai-cs.st.com/home
Complexity analysis: Assumptions

<table>
<thead>
<tr>
<th>Computations required</th>
<th>Forward pass</th>
<th>Error at output layer</th>
<th>Backward pass</th>
<th>Weight update</th>
<th>Goodness function</th>
<th>Normalization</th>
<th>Error projection</th>
</tr>
</thead>
<tbody>
<tr>
<td>BP</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FF</td>
<td>2</td>
<td></td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>PEP</td>
<td>2</td>
<td>1</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>MPE</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>
## RAM estimation

<table>
<thead>
<tr>
<th>Learning procedure</th>
<th>Activations RAM during training</th>
</tr>
</thead>
<tbody>
<tr>
<td>BP and PEP</td>
<td>Sum of the activation buffers of all layers</td>
</tr>
<tr>
<td>FF</td>
<td>Max value of the sum of the activation buffers of two consecutive layers + the input sample</td>
</tr>
<tr>
<td>MPE</td>
<td>Max value of the sum of the activation buffers of two consecutive layers + the largest activation buffer between these two layers</td>
</tr>
</tbody>
</table>

Total RAM = *Activations RAM* + footprint *(weights + biases)*
Results
Learning procedure: the analysis

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Learning method</strong></td>
<td>BP</td>
<td>FF</td>
<td>PEP</td>
<td>MPE</td>
</tr>
<tr>
<td>MAC (C (M))</td>
<td>7.7</td>
<td>+43%</td>
<td>+4%</td>
<td>+39%</td>
</tr>
<tr>
<td>ACT (KiB)</td>
<td>+253%</td>
<td>21</td>
<td>+253%</td>
<td>+37%</td>
</tr>
<tr>
<td>PAR/ACT</td>
<td>0.3</td>
<td>1.1</td>
<td>0.3</td>
<td>0.8</td>
</tr>
<tr>
<td>RAM (KiB)</td>
<td>+120%</td>
<td>43</td>
<td>+120%</td>
<td>+17%</td>
</tr>
</tbody>
</table>

**averaged**

<table>
<thead>
<tr>
<th>Architecture</th>
<th>CNN</th>
<th>FC(AE)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Learning method</strong></td>
<td>FF</td>
<td>PEP</td>
</tr>
<tr>
<td>MACC</td>
<td>+40%</td>
<td>+3%</td>
</tr>
<tr>
<td>ACT</td>
<td>-65%</td>
<td>0%</td>
</tr>
<tr>
<td>RAM</td>
<td>-41%</td>
<td>0%</td>
</tr>
</tbody>
</table>
## Inference procedure: the analysis

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Inference method</strong></td>
<td>BP</td>
<td>FF</td>
<td>PEP</td>
<td>MPE</td>
</tr>
<tr>
<td>MACC (M)</td>
<td>3</td>
<td>+1167 %</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>RAM (KiB)</td>
<td>20</td>
<td>+2.4%</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>ROM (KiB)</td>
<td>22.604</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Inference method: averaged

<table>
<thead>
<tr>
<th>Inference</th>
<th>CNN</th>
<th>FC(AE)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Inference method</strong></td>
<td>FF</td>
<td>PEP</td>
</tr>
<tr>
<td>MACC</td>
<td>+731%</td>
<td>0%</td>
</tr>
<tr>
<td>RAM</td>
<td>+20%</td>
<td>0%</td>
</tr>
</tbody>
</table>
Latency per input sample on MCUs

H7 = STM32H735G-DK @ 550 MHz
G4 = NUCLEO-G474RE @ 170 MHz

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>BP</td>
<td>FF</td>
<td>PEP</td>
<td>MPE</td>
</tr>
<tr>
<td>Learning method</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H7 Training (ms)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H7</td>
<td>42</td>
<td>+43%</td>
<td>+4%</td>
<td>+39%</td>
</tr>
<tr>
<td>G4</td>
<td>203</td>
<td>+43%</td>
<td>+4%</td>
<td>+39%</td>
</tr>
<tr>
<td>Inference (ms)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H7</td>
<td>14.5</td>
<td>+1165%</td>
<td>14.5</td>
<td>42</td>
</tr>
<tr>
<td>G4</td>
<td>71</td>
<td>+1165%</td>
<td>71</td>
<td>202</td>
</tr>
</tbody>
</table>
- **LEARNING → FF** and MEMPEPITA
  - reduced activations (ACT) storage on average 40% to 65% (w.r.t. BP),
  - increased MACCs 40% to 100%
  - **PEPITA** (same memory as BP) increased MACCS (CNN) 3%, (FC) 69%.

- Total **RAM reduction** is noticeable if the topology has low parameters/activations (FC vs CNN). → e.g. for DS-CNN is 0.3 and RAM reduction is around -100%

- **INFERENCE → MEMPEPITA, PEPITA** and BP featured 1 forward pass, while supervised FF adds 2-3x more computation due to N forward passes, for N classes
About training → MEMPEPITA reduced total RAM, (CNN) 33%, (FC) 0.3%, at the expense of a third more MACCs. Inference complexity was unchanged.

**Suitability of Forward-Forward and PEPITA Learning to MLCommons-Tiny benchmarks**

This repository contains the spreadsheet ("analysis.ods") of the quantitative analysis performed for the paper "Suitability of Forward-Forward and PEPITA Learning to MLCommons-Tiny benchmarks".

**Brief description**

The objective of the analysis is to evaluate the performances in terms of memory usage and complexity of a learning procedure $X$ on a certain model $Y$ tackling a task $T$ on a dataset $D$. The learning procedures are Backpropagation (BP), Forward-Forward (FF), PEPITA (PEP), MEMPEPITA (MPE). The models evaluated were respectively named DS-CNN, MobileNet, ResNet and AutoEncoder (AE). The datasets used were Speech Commands (SC), Visual Wake
Further questions? Please contact: danilo.pau@st.com
Our technology starts with You

Find out more at www.st.com
Copyright Notice

This multimedia file is copyright © 2023 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