“TinyDenoiser: RNN-based Speech Enhancement on a Multi-Core MCU with Mixed FP16-INT8 Post-Training Quantization”

Manuele Rusci – MSCA Post-Doc Fellow, the Katholieke Universiteit Leuven

January 26, 2023
Thank you, tinyML Strategic Partners, for committing to take tinyML to the next Level, together
Executive Strategic Partners
The Leading Development Platform for Edge ML

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
Accelerate Your Edge Compute

Making Edge AI A Reality

www.syntiant.com
Platinum Strategic Partners
Fastest Video Analytics Solutions on Arm CPUs
KLIKA • TECH
GLOBAL IOT SOLUTIONS
Renesas is enabling the next generation of AI-powered solutions that will revolutionize every industry sector.
Sony Semiconductor Solutions Corporation
Gold Strategic Partners
Witness potential made possible at analog.com.

Where what if becomes what is.
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
FOTAHUB

Making Over-the-Air Firmware and ML models Updates Simple and Accessible!

- Securely update your IoT devices regardless of their Hardware Platform (Silicon) Provider and physical location.
- Unlock TinyML business value through OTA Firmware and ML models update.
- Pay-as-you-go

www.fotahub.com
contact@fotahub.com
TOGETHER, WE ACCELERATE THE BREAKTHROUGHS THAT ADVANCE OUR WORLD

www.nxp.com/ai
The Right Edge AI Tools Can Make or Break Your Next Smart IoT Product

Analytics Toolkit Suite

AutoML

Data Collection

Test & Validation

Code Generation

Version Control and Model Management

Team Collaboration

sensiml.com/tinyML
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
Join Growing tinyML Communities:

[Image: meetup logo]

tinyML - Enabling ultra-low Power ML at the Edge

[Image: LinkedIn logo]

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

12.8k members in 46 Groups in 37 Countries

3.3k members & 10.8k followers
Subscribe to tinyML YouTube Channel for updates and notifications (including this video)
www.youtube.com/tinyML
The tinyML Research Symposium 2023 will be held in conjunction with the tinyML Summit. The Research Symposium is the premier annual gathering of senior level technical experts and decision makers representing fast growing global tinyML community.

Venue: Hyatt Regency San Francisco Airport
1333 Bayshore Highway, Burlingame, CA 94010
Dates: March 27-29, 2023
- + Exhibition
- + tinyML Research Symposium
Registration is open: https://www.tinyml.org/event/summit-2023/

Call for Summit 2023 nomination awards is open now!
Nominations - Best Product of the Year, Best Innovation of the Year, Best Paper
https://www.tinyml.org/news/summit-2023-awards

- Sponsorships@tinyml.org
- In-person event
- Contact: bette@tinyml.org
tinyML EMEA Innovation Forum 2023

Connect, Unify, and Grow the tinyML EMEA Community
June 26 - 28, 2023

https://www.tinyml.org/event/

Event will be held in person in Amsterdam, Netherlands

EMEA 2023 Call for Presentations is open now:
https://www.tinyml.org/news/emea-2023-call-for-presentations
Abstract due - February 28, 2023
Author notification - March 10, 2023

More sponsorships are available: sponsorships@tinyML.org
Reminders

Slides & Videos will be posted tomorrow

tinyml.org/forums  youtube.com/tinyml

Please use the Q&A window for your questions
Dr. Manuele Rusci received the Ph.D. degree in electronic engineering from the University of Bologna in 2018. He is currently holding a MSCA Post-Doctoral Fellowship at the Katholieke Universiteit Leuven, after being Post-Doc at the University of Bologna. His main research interests include low-power AI-powered smart sensors and on-device continual learning.
TinyDenoiser: RNN-based Speech Enhancement on a Multi-Core MCU with Mixed FP16-INT8 Post-Training Quantization

Manuele Rusci, Marie Curie Post-Doc at KU Leuven
manuele.rusci@esat.kuleuven.be

In collaboration with: Marco Fariselli, Martin Croome, Francesco Paci, Eric Flamand (GreenWaves Technologies)

Low-Power Consumption to gain an extended battery lifetime

- size constraints demand small batteries (e.g. 60mAh)
- 20h if avg power of 10mW for sensing, processing and actuation

✓ Active Power < 10s mW
✓ Sleep Power of 10s uW
✓ Single-core RISC CPU,
  ▪ max freq < 200MHz
✓ Few MBs of on-chip memory

Integration of complex Machine Learning pipelines extremely challenging
Speech Enhancement (or Denoising)

Noisy Speech Signal

Windowed Audio Frame \( t \)

STFT

Deep Learning model

\( \text{iSTFT} \)

Real-Time Digital Signal Processing

Denoised Speech Signal

feed

overlap and add

M. Rusci - TinyMLTalk 26Jan 2023
Speech Enhancement (or Denoising)

Noisy Speech Signal

Real-Time Digital Signal Processing

Denoised Speech Signal

Hop Size Real-Time Constraint

Windowed Audio Frame $t$

feed

STFT

Deep Learning model

iSTFT

overlap and add
Embedding RNNs on MCUs is hard

Deep Learning model

Input Frequency Feature vector

Conv Layer

Input Embedding vector

Recurrent Layer (LSTM, GRU)

Output Embedding vector

Conv Layer

Output Frequency Feature vector

LSTM

\[
\begin{align*}
i_t &= \sigma(W_i \cdot [x_t, h_{t-1}] + b_i) \\
f_t &= \sigma(W_f \cdot [x_t, h_{t-1}] + b_f) \\
g_t &= \tanh(W_g \cdot [x_t, h_{t-1}] + b_g) \\
a_t &= \sigma(W_o \cdot [x_t, h_{t-1}] + b_o) \\
c_t &= f_t \cdot c_{t-1} + i_t \cdot g_t \\
h_t &= a_t \cdot \tanh(c_t)
\end{align*}
\]

GRU

\[
\begin{align*}
i_t &= \sigma(W_i \cdot [x_t, h_{t-1}] + b_i) \\
f_t &= \sigma(W_f \cdot [x_t, h_{t-1}] + b_f) \\
g_t &= \tanh(W_g \cdot [x_t, h_{t-1}] + b_g) \\
c_t &= f_t \cdot c_{t-1} + i_t \cdot g_t \\
h_t &= (1 - z_t) \cdot c_{t-1} + z_t \cdot h_{t-1}
\end{align*}
\]

Model | Type | MAC | Params | QUANT | OAT | MCU
--- | --- | --- | --- | --- | --- | ---
RNNoise [1] | GRU | 0.208 M | 0.208 M | INT8 | yes | STM32L476
TinyLSTM [2] | LSTM | 0.33 M | 0.33 M | INT8 | yes | STM32F746VE
TinyLSTM [2] w/ skip update | LSTM | 0.185 M | 0.46 M | INT8 | yes | STM32F746VE

Embedding RNNs on MCUs is hard

**Deep Learning model**

- Single Core CPU
  - 140 MHz (L4) - 216 MHz (F7)
  - Active Power < 10s mW, Sleep Power of 10s uW
  - INT16 Vectorized MAC in a single clock cycle
  - CMSIS-NN backend for the acceleration of NN inference

- Few MBs of on-chip memory
  - STM32L4: 128kB RAM + 2 MB FLASH
  - STM32FT: 320kB RAM + 0.5 MB FLASH

**Model | Type | MAC | Params | QUANT | QAT | MCU**
---|---|---|---|---|---|---
RNNoise [1] | GRU | 0.208 M | 0.208 M | INT8 | yes | STM32L476
TinyLSTM [2] | LSTM | 0.33 M | 0.33 M | INT8 | yes | STM32F746VE
TinyLSTM [2] w/ skip update | | 0.188 M | 0.46 M | INT8 | yes | STM32F746VE


**LSTM**
\[
i_t = \sigma(W_i \cdot [x_t, h_{t-1}] + b_i) \\
f_t = \sigma(W_f \cdot [x_t, h_{t-1}] + b_f) \\
g_t = \tanh(W_g \cdot [x_t, h_{t-1}] + b_g) \\
a_t = \sigma(W_o \cdot [x_t, h_{t-1}] + b_o) \\
c_t = f_t \cdot c_{t-1} + i_t \cdot g_t \\
h_t = a_t \cdot \tanh(c_t)
\]

**GRU**
\[
r_t = \sigma(W_r \cdot [x_t, h_{t-1}] + b_r) \\
z_t = \sigma(W_z \cdot [x_t, h_{t-1}] + b_z) \\
n_t = \tanh(W_n \cdot x_t + r_t \cdot (W_n h_{t-1} + b_n)) \\
h_t = (1 - z_t) \cdot n_t + z_t \cdot h_{t-1}
\]

**Input Frequency Feature vector**

**Input Embedding vector**

**Recurrent Layer** (LSTM, GRU)

**Output Frequency Feature vector**

**Output Embedding vector**

**Conv Layer**

\[>2\text{x faster than FP32 (if FPU 1clk MAC)}\]
\[4\text{x memory compression than FP32}\]
Embedding RNNs on MCUs is hard

- Training produces **FP32** weights and activations
- Quantization to **INT8** is lossy

**QAT** (quantization-aware training)

- Fine-tuning training the model with quantization error modeling
- Recover accuracy
- Need data
- Computationally expensive
- DL tool support

**PTQ** (post-training Quantization)

- Lightweight but not as effective as QAT
- Need low data
- Computationally inexpensive
- Some accuracy drop

---

**Deep Learning model**

- **Input Frequency Feature vector**
- **Conv Layer**
- **Input Embedding vector**
- **Recurrent Layer** (LSTM, GRU)
- **Output Frequency Feature vector**
- **Conv Layer**
- **Output Embedding vector**

**LSTM**

\[
i_t = \sigma(W_i \cdot [x_t, h_{t-1}] + b_i) \\
f_t = \sigma(W_f \cdot [x_t, h_{t-1}] + b_f) \\
g_t = \tanh(W_g \cdot [x_t, h_{t-1}] + b_g) \\
a_t = \sigma(W_o \cdot [x_t, h_{t-1}] + b_o) \\
c_t = f_t \cdot c_{t-1} + i_t \cdot g_t \\
h_t = a_t \cdot \tanh(c_t)
\]

**GRU**

\[
i_t = \sigma(W_i \cdot [x_t, h_{t-1}] + b_i) \\
f_t = \sigma(W_f \cdot [x_t, h_{t-1}] + b_f) \\
g_t = \tanh(W_g \cdot [x_t, h_{t-1}] + b_g) \\
n_t = \tanh(W_n \cdot x_t + i_t \cdot (W_n \cdot h_{t-1})) \\
h_t = (1 - z_t) \cdot n_t + z_t \cdot h_{t-1}
\]

---

**Model**

<table>
<thead>
<tr>
<th>Model</th>
<th>Type</th>
<th>MAC</th>
<th>Params</th>
<th>QUANT</th>
<th>QAT</th>
<th>MCU</th>
</tr>
</thead>
<tbody>
<tr>
<td>RNNoise [1]</td>
<td>GRU</td>
<td>0.208 M</td>
<td>0.208 M</td>
<td>INT8</td>
<td>yes</td>
<td>STM32L476</td>
</tr>
<tr>
<td>TinyLSTM [2]</td>
<td>LSTM</td>
<td>0.33 M</td>
<td>0.33 M</td>
<td>INT8</td>
<td>yes</td>
<td>STM32F746VE</td>
</tr>
<tr>
<td>TinyLSTM [2] w/ skip update</td>
<td>LSTM</td>
<td>0.188 M</td>
<td>0.46 M</td>
<td>INT8</td>
<td>yes</td>
<td>STM32F746VE</td>
</tr>
</tbody>
</table>

We present an optimized **HW/SW** design for **LSTM and GRU-based Speen Enhancement (SE)** models for **multi-core MCU** systems with limited memory space.

We propose an **almost lossless Mixed-Precision FP16-INT8 Post-Training Quantization** scheme to accelerate RNN-based SE on MCUs.

We provide a detailed analysis of **latency** and **HW/SW efficiency** on a 22-nm 1+8 RISC-V cores MCU.
TinyDenoisers

Noisy Speech Signal

Denoised Speech Signal

RNN-based SE
TinyDenoiser

Output Spectral mask

[ 0.01, 0.52, 0.92, 0.00, ...
0.00, 0.01 ]

1x257 STFT
Magnitude features

Linear + ReLU
RNN_0
Linear + ReLU
RNN_1
Linear + Sigm

LSTM or GRU layers

25 msec windows (hop length 6.25 msec)

Time
Frequency bins

LSTM256 | GRU256 | LSTM128 | GRU128
---|---|---|---
**k** | 256 | 256 | 128 | 128
**RNN_0** | LSTM(257,256) | GRU(257, 256) | LSTM(257,128) | GRU(257, 128)
**RNN_1** | LSTM(257,256) | GRU(257, 256) | LSTM(128, 128) | GRU(128, 128)
**Params** | 1.24 M | 0.985 M | 0.493 M | 0.411 M
**% rnn params** | 84% | 80% | 66.5% | 59.8%

M. Rusci - TinyMLTalk 26Jan 2023
RISC-V MultiCore MCU Platform (GAP9)

- **1 Cluster Controller (CC) Core + 8 Computes Cores**
  - The extended RISC-V ISA includes vectorized INT8 MAC and float16 (FP16) MAC

<table>
<thead>
<tr>
<th>L3 FLASH Memory</th>
<th>8MB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mem IF</td>
<td></td>
</tr>
<tr>
<td>FLASH 2MB</td>
<td></td>
</tr>
<tr>
<td>Micro DMA</td>
<td></td>
</tr>
<tr>
<td>L2 RAM Memory</td>
<td>1.5MB</td>
</tr>
<tr>
<td>L1 Cluster RAM Memory (128kB)</td>
<td></td>
</tr>
<tr>
<td>Octa Core Cluster</td>
<td></td>
</tr>
</tbody>
</table>

```c
void foo()
{
    // some computation here
    // parallelized over 8 cores
    ...
}

int main_cc_core()
{
    ...
    task_offload_compute_cores( foo )
    ...
}
```
- **8 Computes Cores** + 1 **Cluster Controller** (CC) Core
  - The extended RISC-V ISA includes vectorized INT8 MAC and **float16 (FP16) MAC**
- **128kB** of fast-access **L1** and **1.5 MB L2** memories
  - CC program the **DMA** from L2 to L1 to copy data between L2 and L1

```c
void foo()
{
    // some computation here
    // parallelized over 8 cores
    ...
}

int main_cc_core()
{
    ...
    task_offload_compute_cores( foo )
    ...
}
```
Computes Cores + 1 Cluster Controller (CC) Core
- The extended RISC-V ISA includes vectorized INT8 MAC and float16 (FP16) MAC

128kB of fast-access L1 and 1.5 MB L2 memories
- CC program the DMA from L2 to L1 to copy data between L2 and L1

2MB of on-chip L3 memory OR 8MB of off-chip L3 memory
- CC program the MicroDMA to copy data between L3 and L1

Octa Core Cluster
RNN Mapping on HW

- Model Coefficient stored in non-volatile FLASH memory (on-chip, if fitting)
- Execution layer-by-layer
  - Data copied from L3 to L1
  - Parallel Execution on 8cores

\[ T = \sum_{i=1}^{N_{layer}} T_i = T_i^{L3-L2} + T_i^{L2-L1} + T_i^{cores} \]

**LSTM based kernels**
\[
xs = [x_{in}, h_{state}]
\]
\[
\text{parallel for } j \text{ in size}(h_{state}):
\]
\[
\begin{align*}
\text{for } i \text{ in size}(xs): \\
\text{acc}_f &= \text{mac} (xs[i], w_f); \\
\text{acc}_i &= \text{mac} (xs[i], w_i); \\
\text{acc}_g &= \text{mac} (xs[i], w_g); \\
\text{acc}_o &= \text{mac} (xs[i], w_o); \\
\end{align*}
\]
\[
\text{Of} = \text{Sigmoid}(Of); \\
\text{Oi} = \text{Sigmoid}(Oi); \\
\text{Og} = \text{Tanh}(Og); \\
\text{Oo} = \text{Sigmoid}(Oo); \\
\text{next}_c_{state}[j] = c_{state}[j] \times \text{Of} + ((\text{Oi} \times \text{Og})); \\
\text{next}_h_{state}[j] = \text{Tanh}(c_{state}[j]) \times \text{Oo}; \\
\text{h\_state} = \text{next}_h_{state} \\
\text{c\_state} = \text{next}_c_{state}
\]

**GRU based kernels**
\[
xs = [x_{in}, h_{state}]
\]
\[
\text{parallel for } j \text{ in size}(h_{state}):
\]
\[
\begin{align*}
\text{for } i \text{ in size}(xs): \\
\text{acc}_r &= \text{mac} (xs[i], w_r); \\
\text{acc}_z &= \text{mac} (xs[i], w_z); \\
\text{acc}_h &= \text{mac} (xs[i], w_h); \\
\end{align*}
\]
\[
\text{Or} = \text{Sigmoid}(Or); \\
\text{Oz} = \text{Sigmoid}(Oz); \\
\text{Oh} = \text{Tanh}(Oh); \\
\text{next}_h_{state}[j] = (1-\text{Oz})\times\text{Oh} + \text{Oz} \times h_{state}[j]; \\
\text{h\_state} = \text{next}_h_{state}
\]

Problem: coefficients may not fit L1 memory
Need to split a weight tensor into $N_{tile}$ sub-tensors: tiles

\[ T_i = N_{tile} \cdot (T_{i,tile}^{L3-L2} + T_{i,tile}^{L2-L1} + T_{i,tile}^{cores}) \]
Execution of RNN layers

\[
T_i = N_{tile} \cdot (T_{L3-L2} \cdot T_{i,tile} + T_{L2-L1} \cdot T_{i,tile} + T_{cores} \cdot T_{i,tile})
\]

RNN layers features
\#params = \#MAC

1 - 8 MAC/clk
0.5 - 4 MAC /clk
L3 L2

INT8: 1 - 8 params/clk
FP16: 0.5 - 4 params/clk
L3 L2

Inner-loop as the upper-bound of the computation
- 2x slowdown due to outer-loop overhead

**LSTM inner loop**

```python
for i in size(xs):
    acc_f += mac (xs[i], w_f);
    acc_i += mac (xs[i], w_i);
    acc_g += mac (xs[i], w_g);
    acc_o += mac (xs[i], w_o);
```

9 instr.
(5 vect LD + 4 vect vMAC) per-core

**GRU inner loop**

```python
for i in size(xs):
    acc_f += mac (xs[i], w_f);
    acc_i += mac (xs[i], w_i);
    acc_z += mac (xs[i], w_z);
    acc_h += mac (xs[i], w_h);
```

7 instr.
(4 vect LD + 3 vect vMAC) per-core

**Theoretical peak perf on 8 cores**

INT8: 14.2 MAC / cyc
FP16: 7.1 MAC/cyc

FP16 and INT8 execution ~1-2x L2 Memory BW

- L3 time can be predominant: >10x slower than compute
\[ T_i = N_{tile} \cdot (T_{i,tile}^{L3-L2} + T_{i,tile}^{L2-L1} + T_{i,tile}^{cores}) \]
Weight Tensor Promotion before inference

- Smaller models feature a higher promotion rate and a faster execution

High compression brings more promotion opportunities!

\[ T_i = N_{tile} \cdot (T_i^{L_3-L_2} + T_{i,tile}^{L_2-L_1} + T_{i,tile}^{cores}) \]
CC core interleaves memory transfers and compute tasks

Memory copies and computation happens **concurrently**

\[ T_i = N_{tile} \cdot \max(T_{i, tile}^{L3-L2}, T_{i, tile}^{L2-L1}, T_{i, tile}^{cores}) \]

M. Rusci - TinyMLTalk 26Jan 2023
FP16, casting from FP32 to FP16, lossless
(and data-free)

TinyDenoiser models trained on Valentini dataset: FP32 baseline

<table>
<thead>
<tr>
<th>Quantized</th>
<th>LSTM256</th>
<th>GRU256</th>
<th>LSTM128</th>
<th>GRU128</th>
</tr>
</thead>
<tbody>
<tr>
<td>PESQ</td>
<td>STOI</td>
<td>Mem</td>
<td>PESQ</td>
<td>STOI</td>
</tr>
<tr>
<td>MixFP16-INT8</td>
<td>FP32</td>
<td>2.785</td>
<td>0.942</td>
<td>4.750</td>
</tr>
<tr>
<td>MixFP16-INT8</td>
<td>FP16</td>
<td>2.785</td>
<td>0.942</td>
<td>2.37</td>
</tr>
<tr>
<td>MixFP16-INT8</td>
<td>FP16</td>
<td>2.419</td>
<td>0.922</td>
<td>2.484</td>
</tr>
<tr>
<td>MixFP16-INT8</td>
<td>INT8</td>
<td>2.732</td>
<td>0.930</td>
<td>1.370</td>
</tr>
</tbody>
</table>

Mem in MB
**Post-Training Quantization**

- **INT8** is lightweight but **lossy**
  - avg PESQ loss: -0.3, STOI loss: 0.015
  - 2x mem compression

<table>
<thead>
<tr>
<th>Quantized</th>
<th>LSTM256</th>
<th>GRU256</th>
<th>LSTM128</th>
<th>GRU128</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PESQ</td>
<td>STOI</td>
<td>Mem</td>
<td>PESQ</td>
</tr>
<tr>
<td>FP32</td>
<td>2.785</td>
<td>0.942</td>
<td>4.750</td>
<td>2.782</td>
</tr>
<tr>
<td>FP16</td>
<td>2.785</td>
<td>0.942</td>
<td>2.37</td>
<td>2.780</td>
</tr>
<tr>
<td>INT8</td>
<td>2.419</td>
<td>0.922</td>
<td>1.180</td>
<td>2.484</td>
</tr>
<tr>
<td>MixFP16-INT8</td>
<td>2.732</td>
<td>0.930</td>
<td>1.370</td>
<td>2.720</td>
</tr>
</tbody>
</table>

TinyDenoiser models trained on Valentini dataset: FP32 baseline

**Mem in MB**

The majority of the weights are due to RNN layers!
Post-Training Quantization

TinyDenoiser models trained on Valentini dataset: FP32 baseline

<table>
<thead>
<tr>
<th>Quantized</th>
<th>LSTM256</th>
<th>GRU256</th>
<th>LSTM128</th>
<th>GRU128</th>
</tr>
</thead>
<tbody>
<tr>
<td>PESQ</td>
<td>STOI</td>
<td>Mem</td>
<td>PESQ</td>
<td>STOI</td>
</tr>
<tr>
<td>FP32</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FP16</td>
<td>2.785</td>
<td>0.942</td>
<td>1.875</td>
<td>2.758</td>
</tr>
<tr>
<td>INT8</td>
<td>2.419</td>
<td>0.922</td>
<td>0.930</td>
<td>2.512</td>
</tr>
<tr>
<td>MixFP16-INT8</td>
<td>2.732</td>
<td>0.930</td>
<td>1.370</td>
<td>2.687</td>
</tr>
</tbody>
</table>

MixFP16-INT8 present low-accuracy degradation (PESQ: -0.06, STOI: 0.007) while 1.4 – 1.7x mem compression vs FP16

- no need for expensive QAT

M. Rusci - TinyMLTalk 26Jan 2023
Benchmark on 1+8 core 22nm chip (GAP9)

- $V_{DD} = 0.8V$, $f_{max} = 370$ MHz
- $V_{DD} = 0.65V$, $f_{max} = 240$ MHz

Metrics

- MAC/cyc to measure code efficiency
- power consumption (mW)

$$\rho^{L3} = \frac{\# L3\ params}{\#\ params}$$
Latency & Power on target HW/SW

eMRAM as internal Flash Memory

Benchmark on 1+8 core 22nm chip (GAP9)
- $V_{DD} = 0.8V$, $f_{max} = 370$ MHz
- $V_{DD} = 0.65V$, $f_{max} = 240$ MHz

Metrics
- MAC/cyc to measure code efficiency
- power consumption (mW)
- $\rho_{L3} = \frac{\# L3\ params}{\#\ params}$

- FP16 LSTM256 and GRU256 are L3 memory bound given high $\rho_{L3}$
  - extra 40-50 mW of ext L3

- FP16 GRU256 cut power memory costs by storing data in 2MB on-chip FLASH

- Thanks to promotion, FP16 LSTM128 and GRU128 have low $\rho_{L3}$
  - efficiency up to 2.2 MAC/cyc
Latency & Power on target HW/SW

Benchmark on 1+8 core 22nm chip (GAP9)
- $V_{DD} = 0.8V$, $f_{max} = 370$ MHz
- $V_{DD} = 0.65V$, $f_{max} = 240$ MHz

Metrics
- MAC/cyc to measure code efficiency
- power consumption (mW)
- $\rho_{L3} = \frac{\# L3 \ params}{\# \ params}$

- LSTM256 and GRU256 decrease $\rho_{L3}$ thanks to Mixed-Precision
  - avg. eff. up to 2.2MAC/cyc
  - higher power cost because of higher operation density

- LSTM128 and GRU128 improves MAC/cyc thanks to Mixed Precision
Latency & Power on target HW/SW

Benchmark on 1+8 core 22nm chip (GAP9)
- \( V_{DO} = 0.8V, f_{max} = 370 \text{ MHz} \)
- \( V_{DO} = 0.65V, f_{max} = 240 \text{ MHz} \)

Metrics
- MAC/cyc to measure code efficiency
- power consumption (mW)
- \( \rho_{L3} = \frac{# L3 \text{ params}}{# \text{ params}} \)

- Scaling down the voltage reduces the power by 2-2.5x
  - Efficiency slightly improves (7-8%) because on-chip FLASH increases BW at low freq

- Real-time constraint (T=6.25msec) matched
- Assuming negligible power in sleep state, the avg power can reduce up to 3mW

\( eMRAM \) as internal Flash Memory

\( \bullet \) FP16 ExFlash VDD0.8
\( \bullet \) FP16 eMRAM VDD0.8
\( \bullet \) MixFP16INT8 eMRAM VDD0.8
\( \bullet \) MixFP16INT8 eMRAM VDD0.65

\( \rho_{L3} = 0.45 \)
\( \rho_{L3} = 0.79 \)
\( \rho_{L3} = 0.13 \)
\( \rho_{L3} = 0.0 \)
\( \rho_{L3} = 0.0 \)

Latency:
- LSTM256: 2.5 msec
  - Energy: 71.1 \( \mu \text{J} \)
- GRU256: 1.7 msec
  - Energy: 56.7 \( \mu \text{J} \)
- LSTM128: 0.67 msec
  - Energy: 20.6 \( \mu \text{J} \)
- GRU128: 0.6 msec
  - Energy: 18.3 \( \mu \text{J} \)
## Comparison w/ SoA

<table>
<thead>
<tr>
<th>Model</th>
<th>Mpar</th>
<th>Quant</th>
<th>QAT</th>
<th>Device</th>
<th>Deployment</th>
<th>msec/inf</th>
<th>MAC/cyc</th>
<th>MOSP/W</th>
</tr>
</thead>
<tbody>
<tr>
<td>TinyLSTM [2]</td>
<td>0.33</td>
<td>INT8</td>
<td>yes</td>
<td>STM32F746VE</td>
<td>N/A</td>
<td>4.26</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TinyLSTM w/ SkipUpdate [2]</td>
<td>0.46</td>
<td>INT8</td>
<td>yes</td>
<td>STM32L476</td>
<td>NNoM w/ CMSIS-NN</td>
<td>2.39</td>
<td>0.38</td>
<td>0.14</td>
</tr>
<tr>
<td>RNNoise [1]</td>
<td>0.21</td>
<td>INT8</td>
<td>yes</td>
<td>STM32L476</td>
<td>NNoM w/ CMSIS-NN</td>
<td>3.28</td>
<td>0.45</td>
<td>1.84</td>
</tr>
<tr>
<td>TD LSTM256</td>
<td>1.24</td>
<td>MixFP16</td>
<td>no</td>
<td>8-core RISC-V</td>
<td>GAPFlow</td>
<td>2.50</td>
<td>2.11</td>
<td>17.78</td>
</tr>
<tr>
<td>TD GRU156</td>
<td>0.96</td>
<td>INT8</td>
<td>no</td>
<td>8-core RISC-V</td>
<td>GAPFlow</td>
<td>1.70</td>
<td>2.41</td>
<td>17.46</td>
</tr>
</tbody>
</table>

This work: 2.6-6x more params

no QAT but Mixed-Precision

More efficient HW/SW design

More energy-efficient

---


Conclusion

Our optimized design for Speech Enhancement on MCUs consists of multiple elements:

- Multi-core acceleration with vector FP16 and INT8 ISA support
- SW pipeline with interleaved memory transfer and compute calls
- Tensor promotion mechanism to speed-up execution
- FP16-INT8 precision to gain INT8 speed and FP16 accuracy

>10x more energy efficient design than SoA
Thanks for the attention!

manuele.rusci@esat.kuleven.be
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