Optimized C Code Generation for Ultra-efficient tinyML Applications

Imagimob

Johan Malm
2021-05-25
Thank you!

Tweet us: @ArmSoftwareDev

Check out our Arm Software Developers YouTube channel

Signup now for our next AI Virtual Tech Talk: developer.arm.com/techtalks

Attendees: don’t forget to fill out the survey to be in with a chance of winning an Arduino Nano 33 BLE board
<table>
<thead>
<tr>
<th>Date</th>
<th>Title</th>
<th>Host</th>
</tr>
</thead>
<tbody>
<tr>
<td>May 25th</td>
<td>Optimized C Code Generation for Ultra-efficient tinyML Applications</td>
<td>Imagimob</td>
</tr>
<tr>
<td>June 8th</td>
<td>Leveraging DarwinAI’s Deep Learning Solution to Improve Production Efficiency in Manufacturing</td>
<td>DarwinAI</td>
</tr>
<tr>
<td>June 22nd</td>
<td>Introducing NetsPresso by Nota: For Really Fast Inference on Cortex-A devices</td>
<td>Nota.AI</td>
</tr>
<tr>
<td>July 13th</td>
<td>Bringing Edge AI to Life - from PoC to Production</td>
<td>Arcturus &amp; NXP</td>
</tr>
<tr>
<td>July 20th</td>
<td>Easy TinyML with Arduino: taking advantage of machine learning right where things are happening</td>
<td>Arduino</td>
</tr>
</tbody>
</table>

Visit: developer.arm.com/techtalks
Presenter

- AI Engineer in Imagimob Product Team
- Experience from complex R&D projects in academia and industry
- ML, signal processing, numerical methods, visualization
- Ph.D. in computational physics from KTH in Stockholm
- M.Sc in Engineering Physics from Uppsala

Johan Malm
Imagimob

• Specialized in Edge AI (TinyML)
• Experience from 20+ Edge AI customer projects
• We offer
  • Imagimob AI – Software-tools-as-a-Service
  • Edge AI expertise
• Based in Stockholm, Sweden
• 15 employees
What I will talk about

• Intro: The Imagimob Vision

• Case study: Audio

• Deep dive: Optimizations
Intro:
The Imagimob Vision
Simplify The Development of Embedded AI

- Collect & annotate high-quality data
- Manage data into different datasets
- Build & train great models
- Evaluate and find the best model
- Optimize and package application
Specialized Verticals

- General development tool for e.g.
  - PIR sensors, temperature, voltage
  - Predictive maintenance
  - Around 10 different sensors integrated through our Capture System

- Specialized Verticals:
  - Radar
  - Human activity recognition
  - Audio
Case study: Audio
<table>
<thead>
<tr>
<th>Symbol</th>
<th>ID</th>
<th>Train</th>
<th>Validation</th>
<th>Test</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>down</td>
<td>1</td>
<td>0.209</td>
<td>0.143</td>
<td>0.146</td>
<td>1</td>
</tr>
<tr>
<td>up</td>
<td>2</td>
<td>0.295</td>
<td>0.055</td>
<td>0.11</td>
<td>1</td>
</tr>
</tbody>
</table>
Build a preprocessor by combining different preprocessing layers.
The preprocessor manipulates all data before it is input into models.

Preprocessor Input

<table>
<thead>
<tr>
<th>Name</th>
<th>Shape</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sliding Window</td>
<td>[512]</td>
</tr>
<tr>
<td>Hanning smoothing</td>
<td>[512]</td>
</tr>
<tr>
<td>Real Discrete Fourier Transform</td>
<td>[257, 2]</td>
</tr>
</tbody>
</table>

- Create Track from Preprocessor...
- Build Preprocessor...

- Label file contains no labels (after clipping). In 6 segments.
Deep dive:

Optimizations
C code generation

1. Flexibility
2. Portability
3. Speed
C code generation

1. Generate Tensorflow model (+ other arbitrary Python layers) from Imagimob Studio or external .h5 file.
2. All layers are implemented as efficient C code (a layer is a so-called “imunit”).
C code generation

3. A compute graph is generated from the imunits.

Graph is optimized:

- Static memory allocation (our own memory management)
- Matrices are transposed to ensure efficient memory usage
- Constants are evaluated at compile time
- Expressions are simplified \((8 + x \times (10+5)) \Rightarrow 8 + 15x\)
- Loops are optimized
C code generation

4. C code is generated from the compute graph.

C code generated for the in-ear headphone project

- 12 kB RAM
- 98 kB Flash
- ARM M4

Ideal to deploy on the tiniest MCUs on the market:

- ARM M3-M4 have been used in 90 + % of our projects
- In some projects ARM M0
Optimizations

- Quantization
  - Post-training, full integer quantization.
  - Basic idea: Replace floating point operations with integer operations – as much as possible.
  - Up to 4x smaller, 3x+ speedup (more when no FPU is used).

\[
\text{int8\_value} = \left(\frac{\text{real32\_value}}{\text{scale}}\right) + \text{zero\_point}
\]
\[
\text{real32\_value} = \left(\text{int8\_value} - \text{zero\_point}\right) \times \text{scale}
\]
Optimizations

- Hardware accelerated operations (road map).
  - Example: 65 % of the compute time used in Conv1D
    - Close to 100 % dot products → MAC
  - Dense layers for the rest
    - Close to 100 % mult-add
  - → A very large portion of the instructions can be optimized.
- ARM CMSIS-DSP, CMSIS-NN C++ API can be easily integrated.
Thank you!

Tweet us: @ArmSoftwareDev

Check out our Arm Software Developers YouTube channel

Signup now for our next AI Virtual Tech Talk: developer.arm.com/techtalks

Attendees: don’t forget to fill out the survey to be in with a chance of winning an Arduino Nano 33 BLE board
Where Hardware & Software Join Forces
Call for Papers

Learn More at devsummit.arm.com
Thank You
Danke
Merci
谢谢
ありがとうございます
Gracias
Kiitos
감사합니다
धन्यवाद
شكرًا
תודה