Teaching old sensors new tricks

Jan Jongboom
Co-founder and CTO
Jan Jongboom
CTO and co-founder, Edge Impulse
jan@edgeimpulse.com
柯洁 KE JIE
00:17:05

ALPHAGO
01:51:38
Machine learning?
Fast forward to 2020...

http://www.reactiongifs.com/tim-eric-mind-blown.gif
An ML model is still an algorithm

Is water boiling?

Temperature sensor

if temperature > 100°C then
  Yes
else
  No
An ML model is still an algorithm

Is water boiling?

Temperature sensor (102°C)

Water boiling?

Algorithm: Yes

Observed: Yes
An ML model is still an algorithm

Is water boiling?

Algorithm: No
Observed: No

Temperature sensor (98°C)
But now we go mountain climbing...

Is water boiling?

Temperature sensor (98°C)

Water boiling?

Algorithm: No

Observed: Yes
Machine learning finds hidden correlation

Step 1 - Collect data

<table>
<thead>
<tr>
<th>Temperature</th>
<th>Altitude</th>
<th>Boiling?</th>
</tr>
</thead>
<tbody>
<tr>
<td>98°C</td>
<td>0 meter</td>
<td>No</td>
</tr>
<tr>
<td>102°C</td>
<td>0 meter</td>
<td>Yes</td>
</tr>
<tr>
<td>98°C</td>
<td>2200 meter</td>
<td>Yes</td>
</tr>
<tr>
<td>91°C</td>
<td>2200 meter</td>
<td>No</td>
</tr>
</tbody>
</table>

Step 2 - Train ML model

Find function that most accurately maps data => label
Takes lots of trial & error

boiling = temp + (altitude / 304) > 100
#1 - Stick to inferencing

It's just math

Microcontrollers are really good at math

Training ML algorithms is (typically) outside the scope of TinyML

Need some training? Push to cloud, re-train, push model back to device.
Large formulas with lots of parameters

Input signal
16,000 data points

Model architecture
100,000 parameters
#2 - Quantization

Finding the mathematical formula requires lots of trial & error

Need high precision to make small adjustments

But... after model is trained can reduce precision (e.g. float32 => int8)

4x less RAM, speed up w/ SIMD instructions, vector extensions

## Quantization in practice

<table>
<thead>
<tr>
<th></th>
<th>RAM USAGE</th>
<th>LATENCY</th>
<th>ROM USAGE</th>
<th>ACCURACY</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Quantized (int8)</strong></td>
<td>8.8K</td>
<td>36 ms</td>
<td>31.8K</td>
<td>93.08%</td>
</tr>
<tr>
<td><strong>Unoptimized (float32)</strong></td>
<td>27.7K</td>
<td>169 ms</td>
<td>41.9K</td>
<td>93.92%</td>
</tr>
</tbody>
</table>

Typical speedup between 2x and 5x

Larger model? Larger error. Up to 15%!

Quantization-aware training might help

*It's a trade-off, measure it!*
Not every parameter is as important
#3 - Pruning

Certain parameters barely influence the outcome (esp. after quantization)

Find all parameters that are ~0, remove them

Large network might have subnetwork that performs similarly (https://arxiv.org/abs/2002.00585)
Lending your ML algorithm a hand...

Apply low-pass filter...
Lending your ML algorithm a hand...

Map to time/frequency domain...

16,000 features

3,185 features
Lending your ML algorithm a hand...

Mel Frequency Cepstral Coefficients...
#4 - Leverage signal processing

On-device intelligence is not new

Neural networks are inefficient, if you can preprocess? Do so!

Significantly reduce input features, leading to smaller networks.

Cleans up input
Also gives early insight in dataset quality

UMAP view of Keyword dataset (yes/no) after MFCC
Constraints matter

- Sensor suite
- Available flash
- Power budget
- Available RAM
- Hardware acceleration
- Inferences / sec
#5 - Design with constraints in mind

New hardware? >2 year cycle

The BOM matters

Signal processing parameters are hyperparameters

Find the best model within Latency/memory/accuracy constraints

Visualizing constraints

MFCC, FFT Length 128

On-device performance

- INFERENCING TIME: 212 ms.
- PEAK RAM USAGE: 13 KB

You can set the EISDP_QUANTIZE_FILTER macro to save ~30% in inferencing time, at the cost of 10K RAM.

MFCC, FFT Length 256

On-device performance

- INFERENCING TIME: 306 ms.
- PEAK RAM USAGE: 17 KB

You can set the EISDP_QUANTIZE_FILTER macro to save ~30% in inferencing time, at the cost of 10K RAM.

1D CNN

On-device performance

- INFERENCING TIME: 4 ms.
- PEAK RAM USAGE: 5.3K
- ROM USAGE: 35.7K

2D CNN

On-device performance

- INFERENCING TIME: 36 ms.
- PEAK RAM USAGE: 8.8K
- ROM USAGE: 31.8K
Most of the time, nothing changes...
#6 - Exploit sparsity

Changes are often localized, e.g. only part of the image changes.

Parameters in formula are 'grouped' and arranged in layers (layers of connected neurons).

Only re-calculate the groups of neurons affected.
#7 - There's still a lot to be found

State-of-the-art models don't translate well to small devices

Explore, think outside the box

Think about signal processing alongside ML architectures

Edge Impulse's best anomaly detection model is basic K-means clustering!
The world is moving fast!

Time per inference (vibration monitoring, Cortex-M4F)

- June 2019: 472ms.
- Now: 16ms.

RAM required (96x96 vision model)

- August 2020: 440K
- Now: 297K
My stages of TinyML

1. Summer 2017: This is impossible!
2. Summer 2018: Perhaps for basic and small models.
3. Summer 2019: OK for realtime audio, but that's probably it.
4. Summer 2020: Holy shit, we can do realtime image classification.
5. Summer 2021: ????? But it's going to be amazing.
Getting started 🚀
Get some hardware

https://docs.edgeimpulse.com/docs

Get data in from any device with the 'Data forwarder'

Deploy to any device that has a C++ compiler

Or use your phone!
End-to-end tutorials on vibration, audio, vision: edgeimpulse.com
Recap

ML + sensors = perfect fit

Significant effort is constantly pushing boundaries

Your experience + ML = unique opportunity

forums.tinyml.org
Questions?

edgeimpulse.com

tinyml.org
Sponsors

Premier Sponsor & tinyML Strategic Partner

arm

Gold Sponsor

EDGE CORTIX®

Silver Sponsor

SynSense
Arm: The Software and Hardware Foundation for tinyML

1. Connect to high-level frameworks
   - Profiling and debugging tooling such as Arm Keil MDK
2. Supported by end-to-end tooling
   - Optimized models for embedded
     - Runtime (e.g. TensorFlow Lite Micro)
3. Connect to Runtime
   - Optimized low-level NN libraries (i.e. CMSIS-NN)
   - RTOS such as Mbed OS
   - Arm Cortex-M CPUs and microNPUs

Stay Connected
- @ArmSoftwareDevelopers
- @ArmSoftwareDev

Resources: developer.arm.com/solutions/machine-learning-on-arm
Dynamic Neural Accelerator™

- 10x more compute with single DNA engine
- More than 20x better energy-efficiency
- Ultra-low latency
- Fully-programmable with INT 8bit support

www.edgecortix.com

TARGET MARKETS
- Automotive
- Robotics
- Drones
- Smart Cities
- Industry 4.0

Tight coupling between software & hardware with automated co-design
SynSense builds ultra-low-power (sub-mW) sensing and inference hardware for 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
Questions?
Or to join tinyML WeChat Group

添加工作人员进官方微信群(注明tinyML)
Please add staff to join our official tinyML WeChat Group
Copyright Notice

This presentation in this publication was presented at tinyML® Asia 2020. 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