“Introduction to optimization algorithms for compressing neural networks”

Marcus Rüb - Hahn-Schickard Research Institute
[German Area Group] - November 4, 2020
tinyML Strategic Partner

arm

Additional Sponsorships available – contact Bette@tinyML.org for info
Arm: The Software and Hardware Foundation for tinyML

1. Connect to high-level frameworks
2. Supported by end-to-end tooling
3. Connect to Runtime

Application

Optimized models for embedded

Runtime (e.g. TensorFlow Lite Micro)

Optimized low-level NN libraries (i.e. CMSIS-NN)

RTOS such as Mbed OS

Arm Cortex-M CPUs and microNPUs

Profiling and debugging tooling such as Arm Keil MDK

Stay Connected

@ArmSoftwareDevelopers
@ArmSoftwareDev

Resources: developer.arm.com/solutions/machine-learning-on-arm

© 2020 Arm Limited (or its affiliates)
tinyML Talks Sponsors

Additional Sponsorships available – contact Bette@tinyML.org for info
WE USE AI TO MAKE OTHER AI FASTER, SMALLER AND MORE POWER EFFICIENT

Automatically compress SOTA models like MobileNet to <200KB with little to no drop in accuracy for inference on resource-limited MCUs

Reduce model optimization trial & error from weeks to days using Deeplite's design space exploration

Deploy more models to your device without sacrificing performance or battery life with our easy-to-use software

BECOME BETA USER bit.ly/testdeeplite
TinyML for all developers

Dataset
- Acquire valuable training data securely
- Enrich data and train ML algorithms

Edge Device
- Real sensors in real time
- Open source SDK

Impulse
- Embedded and edge compute deployment options
- Test impulse with real-time device data flows

Test

Get your free account at [http://edgeimpulse.com](http://edgeimpulse.com)
Health sensors measure PPG and ECG signals critical to understanding vital signs. Signal chain products enable measuring even the most sensitive signals.

The biggest (3MB flash and 1MB SRAM) and the smallest (256KB flash and 96KB SRAM) Cortex M4 microcontrollers enable algorithms and neural networks to run at wearable power levels.

The new MAX78000 implements AI inferences at over 100x lower energy than other embedded options. Now the edge can see and hear like never before.
Qeexo AutoML for Embedded AI
Automated Machine Learning Platform that builds tinyML solutions for the Edge using sensor data

Key Features

- Wide range of ML methods: GBM, XGBoost, Random Forest, Logistic Regression, Decision Tree, SVM, CNN, RNN, CRNN, ANN, Local Outlier Factor, and Isolation Forest
- Easy-to-use interface for labeling, recording, validating, and visualizing time-series sensor data
- On-device inference optimized for low latency, low power consumption, and a small memory footprint
- Supports Arm® Cortex™- M0 to M4 class MCUs
- Automates complex and labor-intensive processes of a typical ML workflow – no coding or ML expertise required!

Target Markets/Applications

- Industrial Predictive Maintenance
- Smart Home
- Wearables
- Automotive
- Mobile
- IoT

QEEXO AUTOML: END-TO-END MACHINE LEARNING PLATFORM

For a limited time, sign up to use Qeexo AutoML at automl.qeexo.com for FREE to bring intelligence to your devices!
is for building products

Reality AI Tools® software
- Automated Feature Exploration and Model Generation
- Bill-of-Materials Optimization
- Automated Data Assessment
- Edge AI / TinyML code for the smallest MCUs

Reality AI solutions
- Automotive sound recognition & localization
- Indoor/outdoor sound event recognition
- RealityCheck™ voice anti-spoofing

https://reality.ai  info@reality.ai  @SensorAI  Reality AI
**SynSense** (formerly known as aiCTX) 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, bio-signals and more.

https://SynSense.ai
# Next tinyML Talks

<table>
<thead>
<tr>
<th>Date</th>
<th>Presenter</th>
<th>Topic / Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tuesday, November 10</td>
<td><strong>Ehsan Saboori</strong></td>
<td>Networks within Networks: Novel CNN design space exploration for resource limited devices</td>
</tr>
<tr>
<td></td>
<td>Co-founder and CTO, Deeplite</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Alexander Samuelsson</strong></td>
<td>How to build advanced hand-gestures using radar and tinyML</td>
</tr>
<tr>
<td></td>
<td>CTO and co-founder, Imagimob</td>
<td></td>
</tr>
</tbody>
</table>

Webcast start time is 8 am Pacific time
Each presentation is approximately 30 minutes in length

Please contact [talks@tinyml.org](mailto:talks@tinyml.org) if you are interested in presenting
Local German Committee

Alexis Veynachter,
Master Degree in Control Engineering, Senior Field Application Engineer
Infineon 32bits MCUs for Sensors, Fusion & Control

Carlos Hernandez-Vaquero
Software Project Manager, IoT devices
Robert Bosch

Prof. Dr. Daniel Mueller-Gritschneder
Interim Head - Chair of Real-time Computer Systems
Group Leader ESL - Chair of Electronic Design Automation
Technical University of Munich
Reminders

Slides & Videos will be posted tomorrow

tinyml.orgforums  youtube.com/tinyml

Please use the Q&A window for your questions
Marcus Rüb studied electrical engineering at Furtwangen University. After completing his bachelor's degree, he worked as a scientific assistant for AI at Hahn-Schickard while completing his master's degree. His main interest is in embedded AI. This often involves the implementation of machine learning algorithms on embedded devices and the compression of ML models. Furthermore Marcus is one of the federal funded AI trainers and supports companies in integrating AI into their processes.
Introduction to optimization algorithms to compress neural networks

Marcus Rüb

Marcus.rueb@hahn-schickard.de
https://www.linkedin.com/in/marcus-rüb-3b07071b2
Hahn-Schickard Villingen-Schwenningen
Agenda

- What is tinyML and why do we need this?
- Quantization
- Knowledge distillation
- Pruning
- Other methods
- Take away
What is tinyML (Edge AI) and why do I need this?

**Benefits:**

**Cloud AI**
- Privacy
- Low latency

**Edge AI**
- Energy saving
- Less communication
Fields of application

- Mobile applications
e.g. pacemaker
- Predictive Maintenance
- Robotic
- IOT
- Production
- Medical Technology
- End products
- Mechanical Engineering
Compress neural networks

- The problems get complexer
- The models get bigger
- Solution: to compress the model
- Problem of compression: we get a trade-off

Compression rate | Accuracy
Quantization

- **Quantization** is the process of constraining an input from a continuous or otherwise large set of values (such as the real numbers) to a discrete set (such as the integers).*

*Wikipedia
Quantization

Double ➔ Float ➔ Fixedpoint ➔ Integer ➔ Binary

64x smaller

<table>
<thead>
<tr>
<th>Double</th>
<th>Float</th>
<th>Fixedpoint</th>
<th>Integer</th>
<th>Binary</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.09</td>
<td>-0.98</td>
<td>1.02</td>
<td>0.09</td>
<td></td>
</tr>
<tr>
<td>0.05</td>
<td>-0.14</td>
<td>-1.08</td>
<td>2.12</td>
<td></td>
</tr>
<tr>
<td>-0.91</td>
<td>1.92</td>
<td>0</td>
<td>-1.03</td>
<td></td>
</tr>
<tr>
<td>1.87</td>
<td>0</td>
<td>1.03</td>
<td>0.98</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Double</th>
<th>Float</th>
<th>Fixedpoint</th>
<th>Integer</th>
<th>Binary</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.09</td>
<td>-2.12</td>
<td>1.92</td>
<td>1.87</td>
<td></td>
</tr>
<tr>
<td>0.05</td>
<td>-0.14</td>
<td>0</td>
<td>0.09</td>
<td></td>
</tr>
<tr>
<td>-0.91</td>
<td>-0.98</td>
<td>-1.08</td>
<td>-1.03</td>
<td></td>
</tr>
<tr>
<td>1.02</td>
<td>1.03</td>
<td>0.98</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Double</th>
<th>Float</th>
<th>Fixedpoint</th>
<th>Integer</th>
<th>Binary</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0</td>
<td>-1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>-1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Retrain
Huffman coding

- Special case of quantization
- Make the model smaller but increase the inference time
- Can be good for Hardware implementations

Figure 3: Weight sharing by scalar quantization (top) and centroids fine-tuning (bottom).
Quantization

- **Pros:**
  - Quantization can be applied both during and after training
  - Can be applied on all layer types
  - Can improve the inference time/model size vs accuracy tradeoff for a given architecture

- **Cons:**
  - Quantized weights make neural networks harder to converge. A smaller learning rate is needed to ensure the network to have good performance.
  - Quantized weights make back-propagation infeasible since gradient cannot back-propagate through discrete neurons. Approximation methods are needed to estimate the gradients of the loss function with respect to the input of the discrete neurons.
Dive deeper?


https://www.tensorflow.org/lite/performance/post_training_integer_quant

https://github.com/google/qkeras
Knowledge distillation

Teacher model

Layer 1 → Layer 2 → ... → Layer m → Softmax \( (T = t) \) → soft labels

Student (distilled) model

Layer 1 → Layer 2 → ... → Layer n

Softmax \( (T = t) \) → soft predictions

Softmax \( (T = 1) \) → hard prediction

Loss Fn

Distillation loss

Hard label \( y \) (ground truth)

Student loss
Knowledge distillation

- The teacher network guides the student network
- Up to 20x smaller networks
Knowledge distillation

**Pros:**
- If you have a pre-trained teacher network, less training data required to train the smaller (student) network.
- If you have a pre-trained teacher network, training of the smaller (student) network is faster.
- Can downsize a network regardless of the structural difference between the teacher and the student network.

**Cons:**
- If you do not have a pre-trained teacher network, it may require a larger dataset and take more time to train it.
- A good hyper-parameter set is hard to find.
Dive deeper?


https://github.com/TropComplique/knowledge-distillation-keras
Pruning

Before the pruning

After the pruning

removed Synapses

removed Neurons
Structured pruning vs. Unstructured pruning

- Unstructured pruning: delete connections between neurons
  - Benefit: easy to implement
- Structured pruning: delete the whole neuron
  - Benefit: compress and speedup the model
Pruning process

1. Unpruned Network

2. Evaluate importance of neuron

3. Remove the least important neuron

4. Re-train

2 to 13x smaller
How to know which Connections/neurons to prune?

- L1/L2 mean
- Magnitude
- Mean activations
- The number of times a neuron was zero on some validation set
- Matrix similarity
Pros and Cons

Pros:
- Can be applied during or after training
- Can improve the inference time/model size vs accuracy tradeoff for a given architecture
- Can be applied to both convolutional and fully connected layers
- Better generalization
- Privacy preserving networks

Cons:
- Unstructured pruning does not speed up the inference
Dive deeper?


https://www.tensorflow.org/lite/performance/post_training_integer_quant

https://github.com/Hahn-Schickard/Automatic-Structured-Pruning
Low-rank factorization

- Done by an SVD
  - Singular Value Decomposition (SVD) of a matrix is a factorization of that matrix into three matrices
- The weight matrix get split into two vectors
- Con: Decomposition is a computationally expensive task
Fast-Conv

- Instead of calculating the convolution, calculate the transform of the input into the frequency-domain and calculate a multiplication.
- The filter kernel is pre-transformed.
- Special case: Winograd-convolution -> faster, but only with even number of filter kernel size.
- Good for hardware implementations.
Selective attention network

- „Divide et impera“ - divide and conquer
- Two algorithm:
  - The first select the area of interest
  - The other is the neural network
Summary

- We learned three compression methods
  - Quantization
    - Huffman coding
  - Knowledge distillation
  - Pruning
  - Low-rank factorization
  - Fast-Conv
  - Selective attention network

- Network compression work
- We can compress the model up to 20x of the size
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