tinyML® Talks

Enabling Ultra-low Power Machine Learning at the Edge

"Demoing the world's fastest inference engine for Arm Cortex-M"

Cedric Nugteren - Plumerai

January 4, 2022

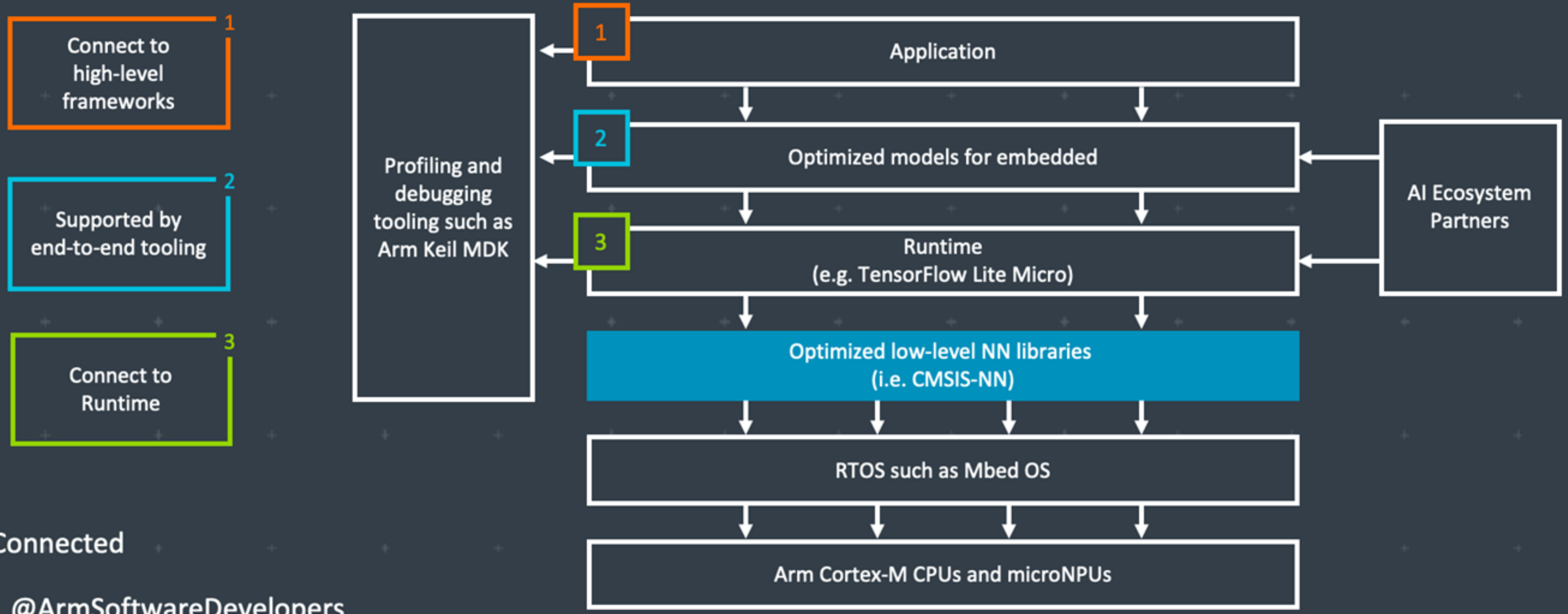
TINY ML

www.tinyML.org

tinyML Talks Strategic Partners

Additional Sponsorships available – contact Olga@tinyML.org for info

Arm: The Software and Hardware Foundation for tinyML



Connect to high-level frameworks 1
Supported by end-to-end tooling 2
Connect to Runtime 3
Profiling and debugging tooling such as Arm Keil MDK
1
Application
2
Optimized models for embedded
3
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
AI Ecosystem Partners

Stay Connected

@ArmSoftwareDevelopers

@ArmSoftwareDev

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

Deeplite

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

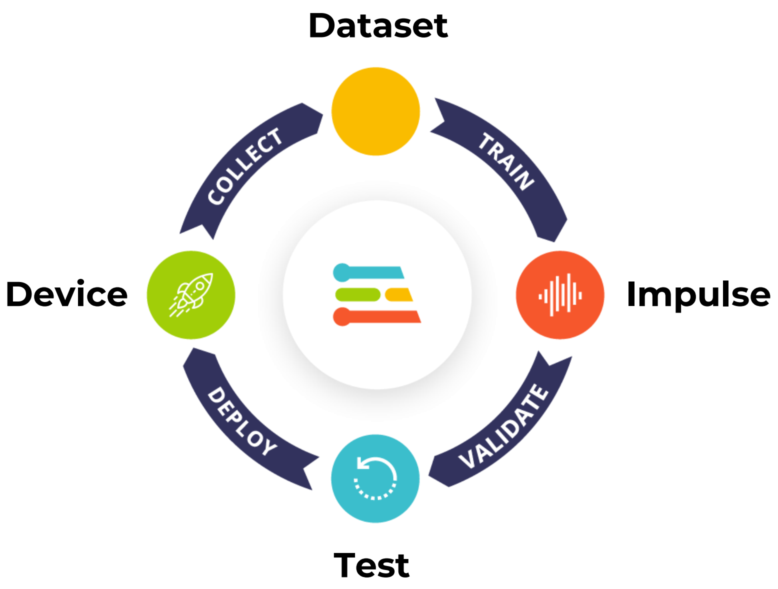
Reduce model optimization trial & error from weeks to days using Deeplite's **design space exploration**

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

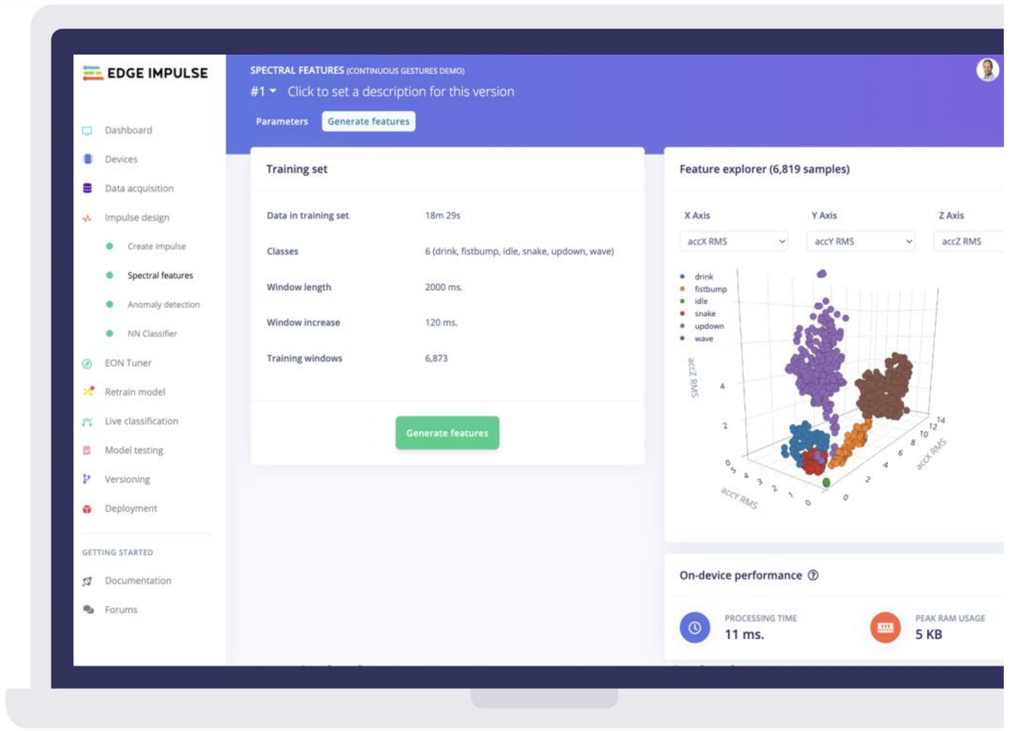
BECOME BETA USER **bit.ly/testdeeplite**

mobility**Xlab** arm AI 100 CBINSIGHTS

EDGE IMPULSE The leading edge ML platform



www.edgeimpulse.com


EDGE IMPULSE

SPECTRAL FEATURES (CONTINUOUS GESTURES DEMO)

#1 ▾ Click to set a description for this version

Parameters | Generate features

Dashboard
Devices
Data acquisition
Impulse design
- Create impulse
- Spectral features
- Anomaly detection
- NN Classifier

EON Tuner
Retrain model
Live classification
Model testing
Versioning
Deployment

GETTING STARTED
Documentation
Forums

Training set

Data in training set	18m 29s
Classes	6 (drink, fistbump, idle, snake, updown, wave)
Window length	2000 ms.
Window increase	120 ms.
Training windows	6,873

Generate features

Feature explorer (6,819 samples)

X Axis: accX RMS | Y Axis: accY RMS | Z Axis: accZ RMS

● drink
● fistbump
● idle
● snake
● updown
● wave

On-device performance

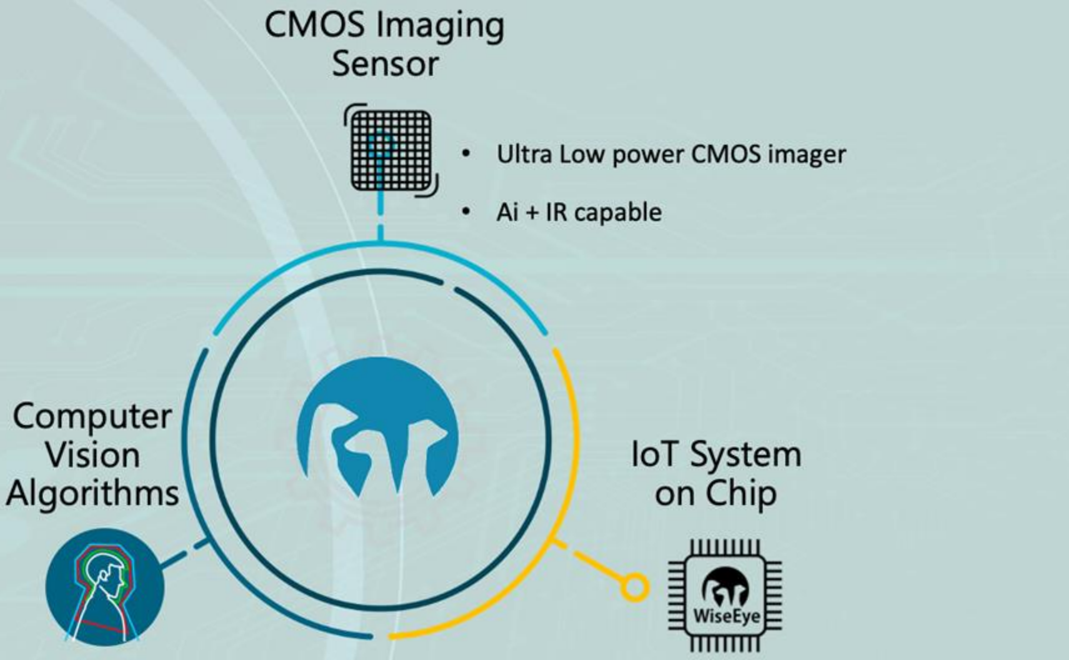
PROCESSING TIME 11 ms.

PEAK RAM USAGE 5 KB

emza
visual sense

The Eye in IoT

Edge AI Visual Sensors



CMOS Imaging Sensor

- Ultra Low power CMOS imager
- Ai + IR capable

Computer Vision Algorithms

- Machine Learning algorithm
- <1MB memory footprint
- Microcontrollers computing power
- Trained algorithm
- Processing of low-res images
- Human detection and other classifiers

IoT System on Chip

- Machine Learning edge computing silicon
- <1mW always-on power consumption
- Computer Vision hardware accelerators

info@emza-vs.com

Enabling the next generation of Sensor and Hearable products to process rich data with energy efficiency

Visible Image

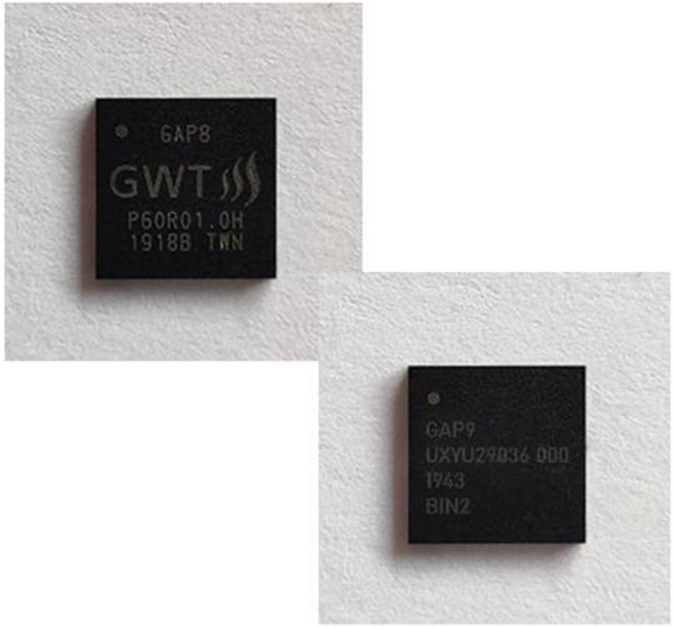
Sound

IR Image

Radar

Bio-sensor

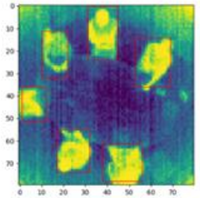
Gyro/Accel



Wearables / Hearables

Battery-powered consumer electronics

IoT Sensors



GREENWAVES TECHNOLOGIES

Grovety Inc.

SOFTWARE DEVELOPMENT SERVICES FOR TINYML SOLUTIONS

1 **Development tools**
SDK, IDE, compilers, leveraging on TVM, uTVM & LLVM

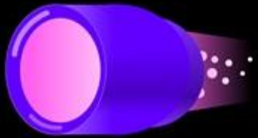
2 **Firmware**
Drivers, BSP, protocols, etc.

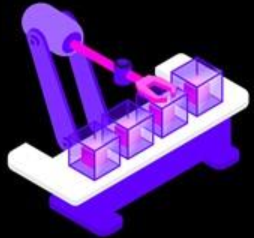
arm AI PARTNER

Distributed infrastructure for TinyML apps

HOTG

Decoupling intelligence





Develop at warp speed

Automate deployments

Device orchestration

HOTG is building the distributed infrastructure to pave the way for AI enabled edge applications

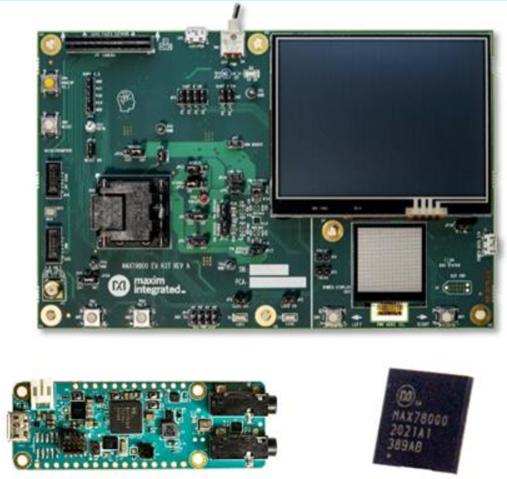
LatentAI

Adaptive AI for the Intelligent Edge

Latentai.com

Maxim Integrated: Enabling Edge Intelligence

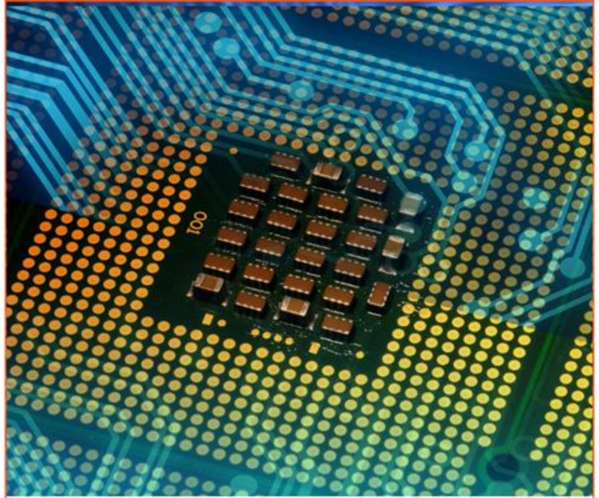
Advanced AI Acceleration IC



The new MAX78000 implements AI inferences at low energy levels, enabling complex audio and video inferencing to run on small batteries. Now the edge can see and hear like never before.

www.maximintegrated.com/MAX78000

Low Power Cortex M4 Micros



Large (3MB flash + 1MB SRAM) and small (256KB flash + 96KB SRAM, 1.6mm x 1.6mm) Cortex M4 microcontrollers enable algorithms and neural networks to run at wearable power levels.

www.maximintegrated.com/microcontrollers

Sensors and Signal Conditioning

Health sensors measure PPG and ECG signals critical to understanding vital signs. Signal chain products enable measuring even the most sensitive signals.

www.maximintegrated.com/sensors

Qeexo AutoML

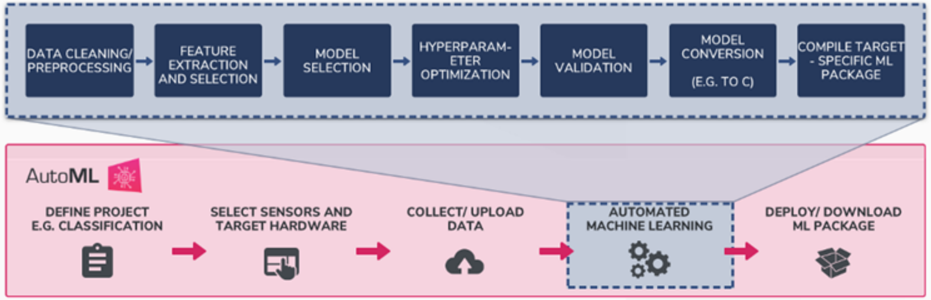
Automated Machine Learning Platform that builds tinyML solutions for the Edge using sensor data

CES INNOVATION AWARDS 2021 HONOREE

Key Features

- Supports 17 ML methods:
  - Multi-class algorithms: GBM, XGBoost, Random Forest, Logistic Regression, Gaussian Naive Bayes, Decision Tree, Polynomial SVM, RBF SVM, SVM, CNN, RNN, CRNN, ANN
  - Single-class algorithms: Local Outlier Factor, One Class SVM, One Class Random Forest, Isolation Forest
- Labels, records, validates, and visualizes time-series sensor data
- On-device inference optimized for low latency, low power consumption, and small memory footprint applications
- Supports Arm® Cortex™- M0 to M4 class MCUs

End-to-End Machine Learning Platform


DATA CLEANING/ PREPROCESSING
FEATURE EXTRACTION AND SELECTION
MODEL SELECTION
HYPERPARAM-ETER OPTIMIZATION
MODEL VALIDATION
MODEL CONVERSION (E.G. TO C)
COMPILE TARGET - SPECIFIC ML PACKAGE
AutoML
DEFINE PROJECT E.G. CLASSIFICATION
SELECT SENSORS AND TARGET HARDWARE
COLLECT/ UPLOAD DATA
AUTOMATED MACHINE LEARNING
DEPLOY/ DOWNLOAD ML PACKAGE

For more information, visit: www.qeexo.com

Target Markets/Applications

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

Qualcomm
AI research

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

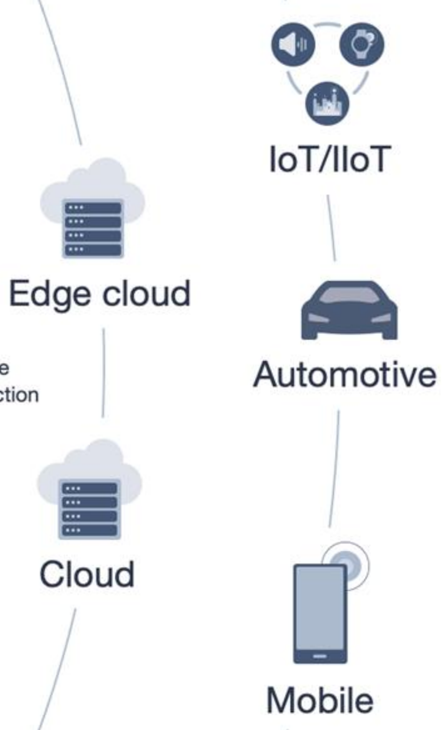
Robust learning through minimal data, unsupervised learning, on-device learning

A platform to scale AI across the industry

Perception
Object detection, speech recognition, contextual fusion

Reasoning
Scene understanding, language understanding, behavior prediction

Action
Reinforcement learning for decision making



Reality AI®

Add Advanced Sensing to your Product with Edge AI / TinyML

https://reality.ai info@reality.ai @SensorAI Reality AI

Pre-built Edge AI sensing modules, plus tools to build your own

Reality AI solutions

Prebuilt sound recognition models for indoor and outdoor use cases

Solution for industrial anomaly detection

Pre-built automotive solution that lets cars "see with sound"

Reality AI Tools® software

Build prototypes, then turn them into real products

Explain ML models and relate the function to the physics

Optimize the hardware, including sensor selection and placement

BROAD AND SCALABLE EDGE COMPUTING PORTFOLIO

Microcontrollers & Microprocessors

Arm® Core

RENESAS RA	Arm® Cortex®-M 32-bit MCUs Arm ecosystem, Advanced security, Intelligent IoT
RENESAS RZ	Arm®-based High-end 32 & 64-bit MPUs High-resolution HMI, Industrial network & real-time control
RENESAS RE	Arm® Cortex®-M0+ Ultra-low Power 32-bit MCUs Innovative process tech (SOTB), Energy harvesting
Renesas Synergy™	Arm®-based 32-bit MCUs for Qualified Platform Qualified software and tools

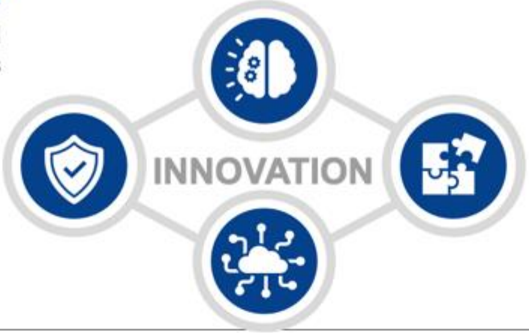
Renesas Core

RENESAS RL78	Ultra-low Energy 8 & 16-bit MCUs Bluetooth® Low Energy, SubGHz, LoRa®-based Solutions
RENESAS RX	High Power Efficiently 32-bit MCUs Motor control, Capacitive touch, Functional safety, GUI
RENESAS RH850	40nm/28nm process Automotive 32-bit MCUs Rich functional safety and embedded security features

Core technologies

AI

A broad set of high-power and energy-efficient embedded processors



Digital & Analog & Power Solution

Winning Combinations that combine our complementary product portfolios

Security & Safety

Comprehensive technology and support that meet the industry's stringent standards

Cloud Native

Cross-platforms working with partners in different verticals and organizations seeed studio

The IoT Hardware Enabler

Build Smart IoT Sensor Devices From Data

SensiML pioneered TinyML software tools that auto generate AI code for the intelligent edge.

- End-to-end AI workflow
- Multi-user auto-labeling of time-series data
- Code transparency and customization at each step in the pipeline

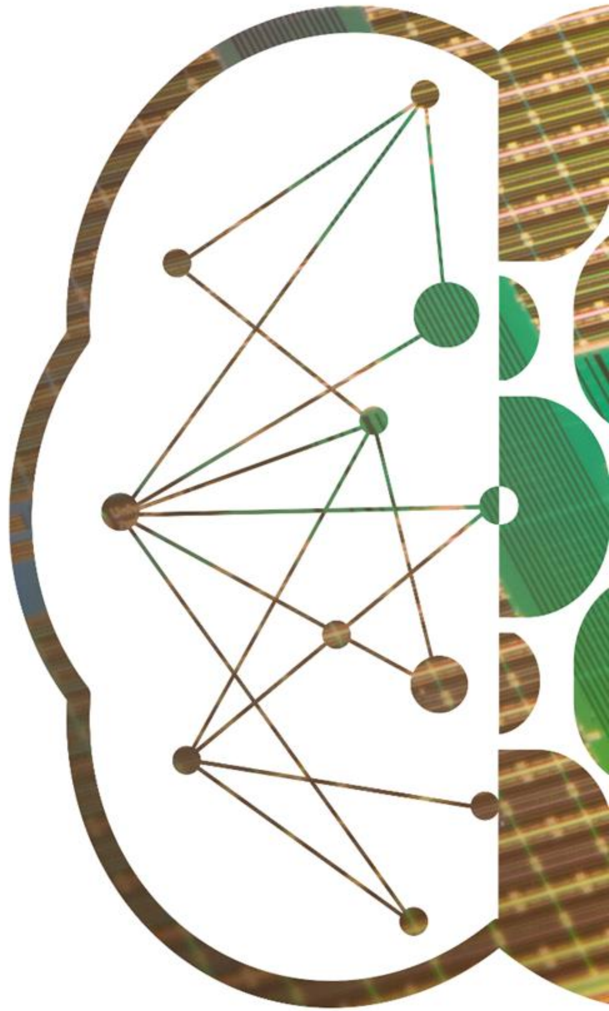
We enable the creation of production-grade smart sensor devices.

sensiml.com

SynSense

SynSense builds **sensing and inference** hardware for **ultra-low-power** (sub-mW) **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



SYNTIANT

End-to-End Deep Learning Solutions

for

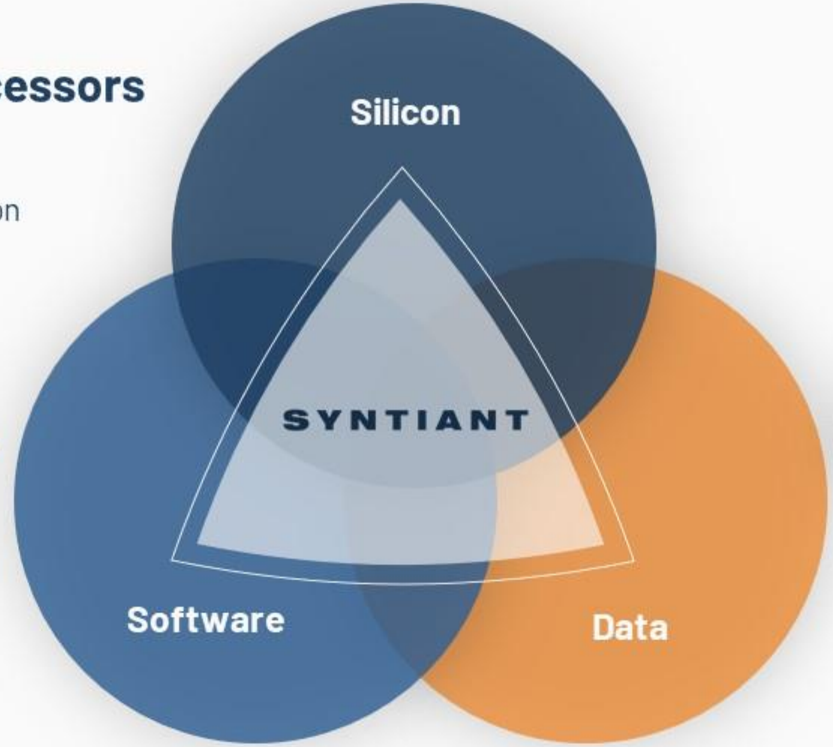
TinyML & Edge AI

Neural Decision Processors

- At-Memory Compute
- Sustained High MAC Utilization
- Native Neural Network Processing

ML Training Pipeline

- Enables Production Quality Deep Learning Deployments



Data Platform

- Reduces Data Collection Time and Cost
- Increases Model Performance

SYNTIANT

partners@syntiant.com

www.syntiant.com

Next tinyML Talks

Date	Presenter	Topic / Title
Tuesday, January 11	Tim Callahan Staff Software Engineer, Google	CFU Playground: Customize Your ML Processor for Your Specific TinyML Model

Webcast start time is 8:00 am Pacific time

Please contact talks@tinyml.org if you are interested in presenting

tinyML Summit 2022

Miniature dreams can come true…

March 28-30, 2022

Hyatt Regency San Francisco Airport

https://www.tinyml.org/event/summit-2022/

Registration will be open on **December 15**, 2021.

Deadline for poster submission is **December 17**.

*The Best Product of the Year and the Best Innovation of the Year awards are open for nominations between **November 15 and February 28.***

tinyML Research Symposium 2022

March 28, 2022

https://www.tinyml.org/event/research-symposium-2022

Call for papers – Submission deadline is **December 17**, 2021.

More sponsorships are available: sponsorships@tinyML.org

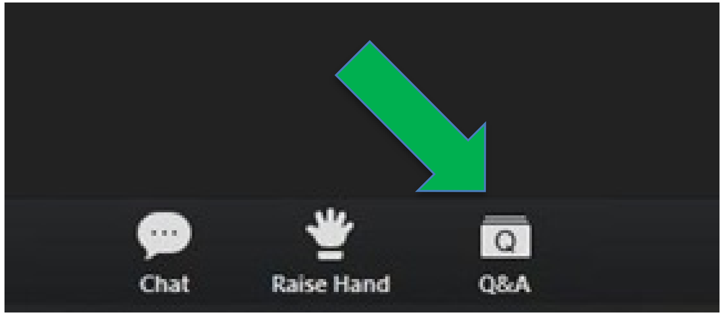
Reminders

Slides & Videos will be posted tomorrow

tinyml.org/forums

youtube.com/tinyml

Please use the Q&A window for your questions



Cedric Nugteren

Cedric Nugteren is a software engineer focussed on writing efficient code for deep learning applications. After he received his MSc and PhD from Eindhoven University of Technology he optimized GPU and CPU code for various companies using C++, OpenCL and CUDA. Then, he worked for 4 years on deep learning for autonomous driving at TomTom, after which he joined Plumerai where he is now writing fast code for the smallest microcontrollers.

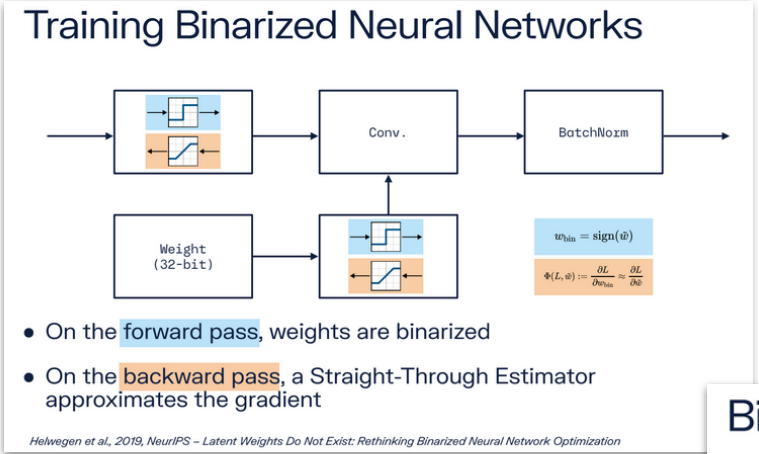
Plumerai

Demoing the world's fastest inference engine for Arm Cortex-M

Cedric Nugteren - cedric@plumerai.com

TinyML Webinar - January 4th, 2022

You might know us from: BNNs?



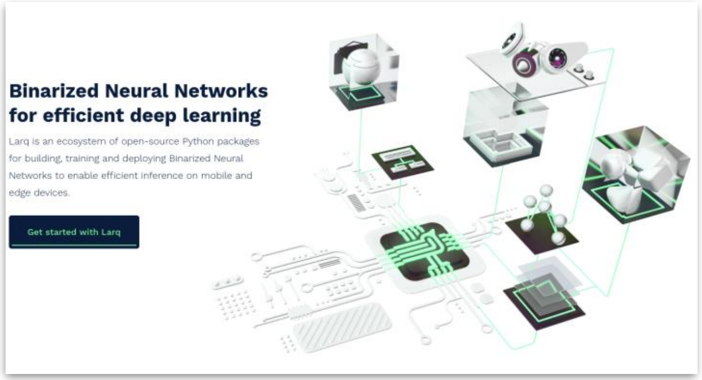
Training Binarized Neural Networks

$w_{\text{bin}} = \text{sign}(\tilde{w})$

$\Phi(L, \tilde{w}) := \frac{\partial L}{\partial w_{\text{bin}}} \approx \frac{\partial L}{\partial \tilde{w}}$

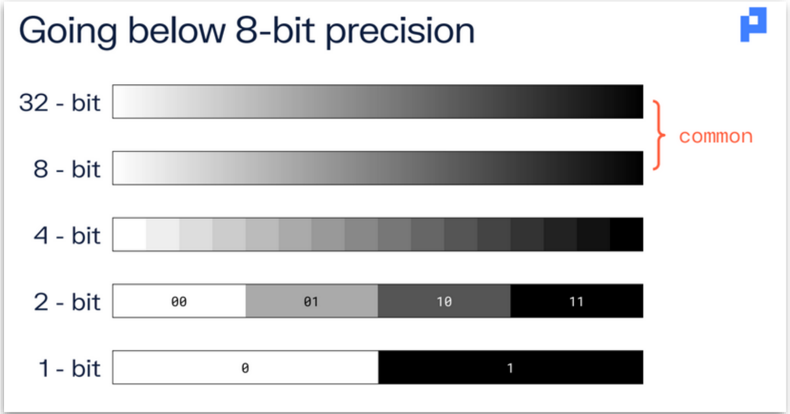
- On the forward pass, weights are binarized
- On the backward pass, a Straight-Through Estimator approximates the gradient

Helwegen et al., 2019, NeurIPS – Latent Weights Do Not Exist: Rethinking Binarized Neural Network Optimization

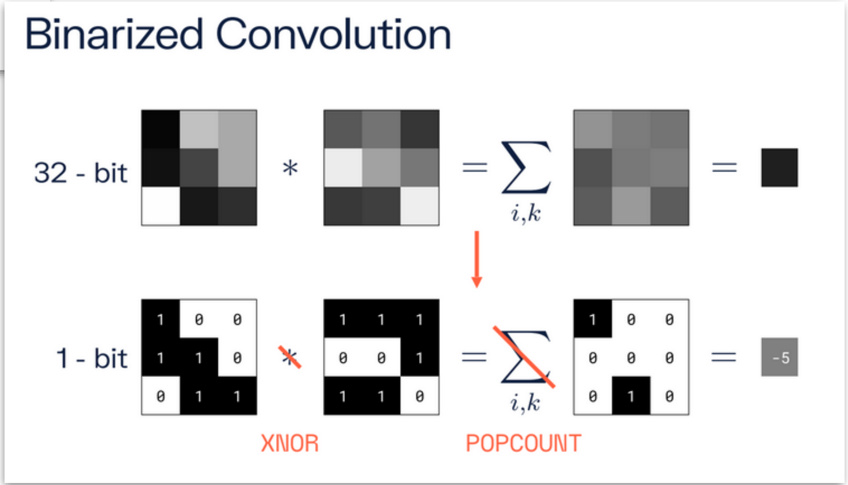


Binarized Neural Networks for efficient deep learning

Larq is an ecosystem of open-source Python packages for building, training and deploying Binarized Neural Networks to enable efficient inference on mobile and edge devices.



Going below 8-bit precision



Binarized Convolution

$$\begin{bmatrix}1&0&0\\1&1&0\\0&1&1\end{bmatrix} \begin{bmatrix}1&1&1\\0&0&1\\1&1&0\end{bmatrix} \begin{bmatrix}1&0&0\\0&0&0\\0&1&0\end{bmatrix}$$

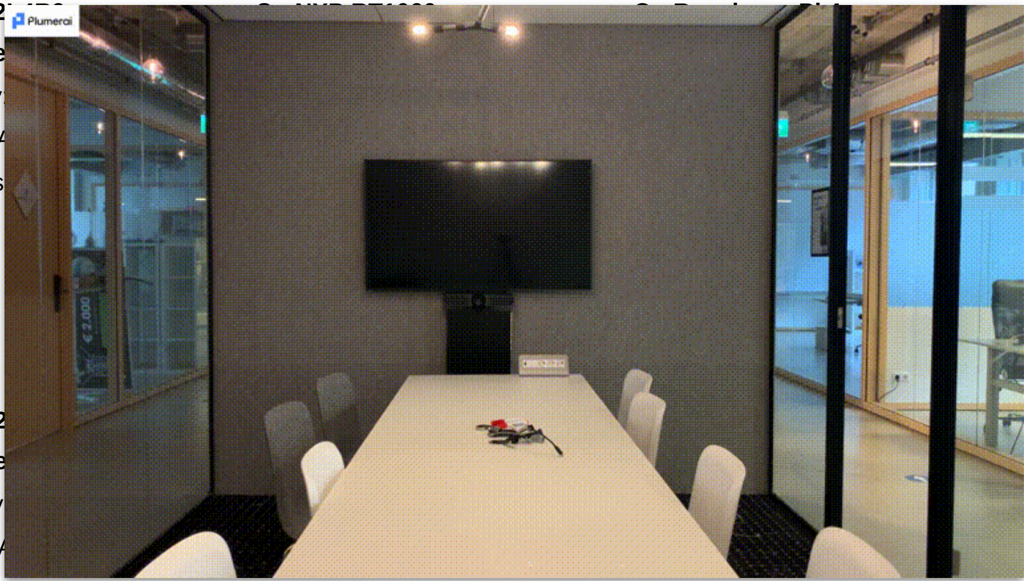
You might know us from: Person detection?

Example:
Smart doorbell

Person Presence Detection

On STM32

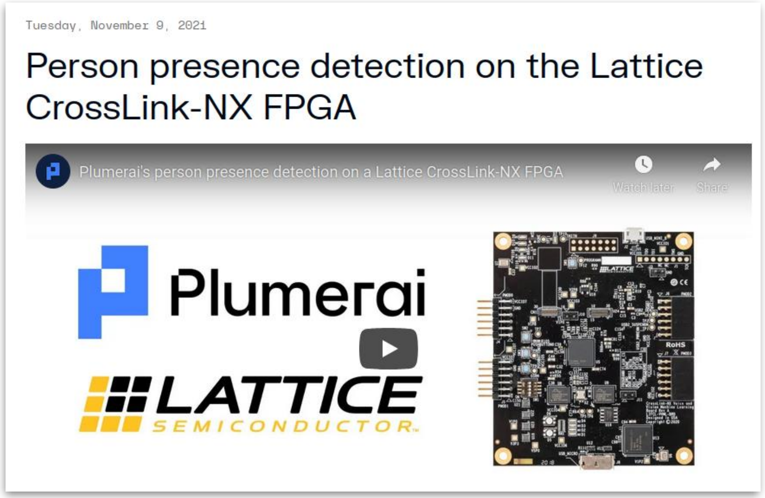
Arm Corte



- Binary size: 895 KiB
- Binary size: 895 KiB

Example:
Smart offices

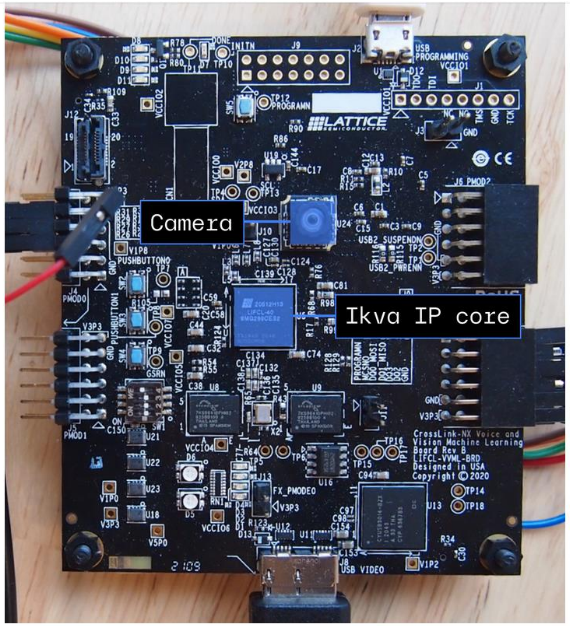
You might know us from: Our own IP core?



Tuesday, November 9, 2021

Person presence detection on the Lattice CrossLink-NX FPGA

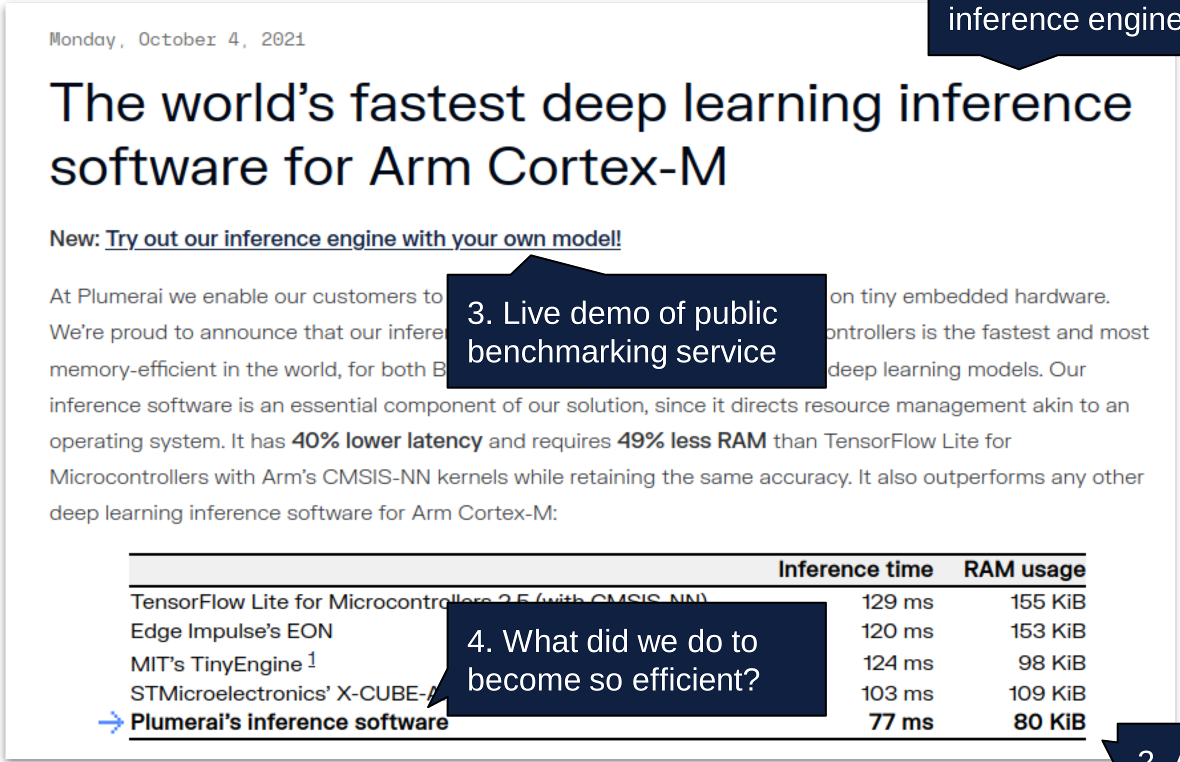
Plumerai's person presence detection on a Lattice CrossLink-NX FPGA
Watch later
Share
Plumerai
LATTICE SEMICONDUCTOR



The Lattice CrossLink-NX Voice & Vision Machine Learning Board with the CrossLink-NX LIFCL-40 FPGA

Or from: the world’s fastest Cortex-M inference?

1. What is an inference engine?



Monday, October 4, 2021

The world’s fastest deep learning inference software for Arm Cortex-M

New: Try out our inference engine with your own model!

At Plumerai we enable our customers to on tiny embedded hardware. We’re proud to announce that our infere ontrollers is the fastest and most memory-efficient in the world, for both B deep learning models. Our inference software is an essential component of our solution, since it directs resource management akin to an operating system. It has **40% lower latency** and requires **49% less RAM** than TensorFlow Lite for Microcontrollers with Arm’s CMSIS-NN kernels while retaining the same accuracy. It also outperforms any other deep learning inference software for Arm Cortex-M:

	Inference time	RAM usage
TensorFlow Lite for Microcontrollers 2.5 (with CMSIS-NN)	129 ms	155 KiB
Edge Impulse’s EON	120 ms	153 KiB
MIT’s TinyEngine [1]	124 ms	98 KiB
STMicroelectronics’ X-CUBE-A	103 ms	109 KiB
→ **Plumerai’s inference software**	**77 ms**	**80 KiB**

3. Live demo of public benchmarking service

4. What did we do to become so efficient?

2. Are we really that efficient?

Plumerai

0. How did we get here?

How did we get here?

Plumerai company timeline

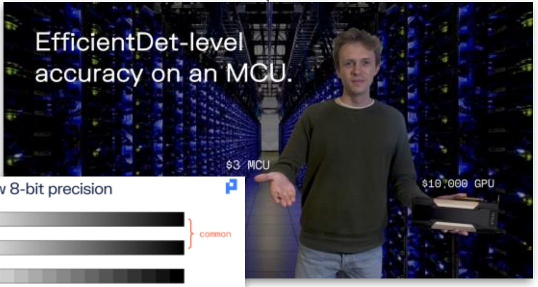
Early days of Plumerai | Quite soon after | Last year | Today

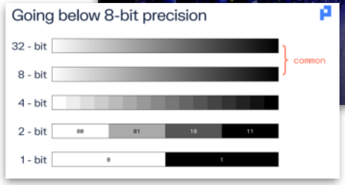
Our goal: run complex computer vision tasks on tiny devices efficiently

Need to cover the entire stack for high efficiency

BNNs do not only have binarized layers, but also INT8…

The world's fastest INT8 inference engine





```
Plumerai Data Pipeline
Plumerai BNN Models
Plumerai Inference Stack
Plumerai Hardware
```

Faster models

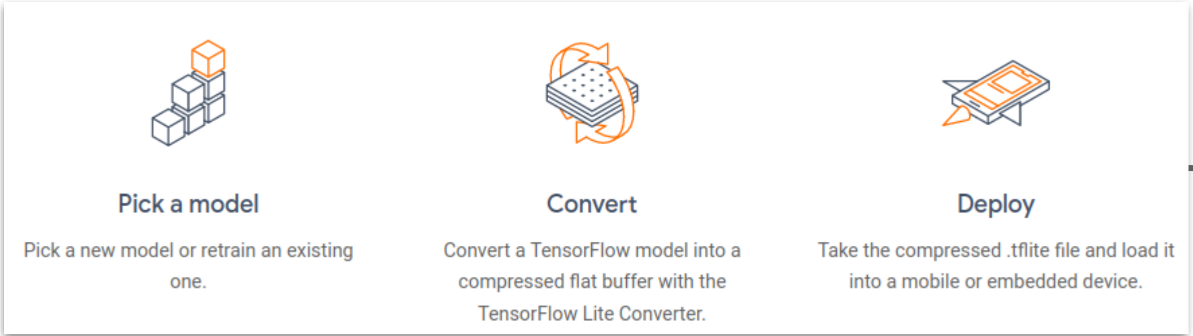
Less RAM

- Better accuracy
- Lower energy usage
- Cheaper and smaller
- Room for other apps

Plumerai

1. What is an inference engine?

The machine learning flow



Pick a model

Pick a new model or retrain an existing one.

Convert

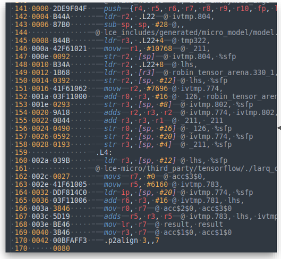
Convert a TensorFlow model into a compressed flat buffer with the TensorFlow Lite Converter.

Deploy

Take the compressed .tflite file and load it into a mobile or embedded device.

Deploy INT8 quantized model on device

Run optimized code

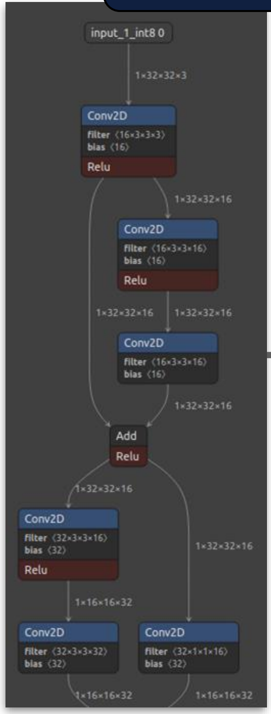


```
0000 2DE9F04F   push  {r4, r5, r6, r7, r8, r9, r10, fp,
0004 B44A       ldr r2, .L22  @ ivtmp.804,
0006 87B0       sub sp, sp, #28 @,,
                @ lce_includes/generated/micro_model/model.
0008 B44B       ldr r3, .L22+4  @ tmp322,
000a 42F61021   movw  r1, #10768  @ _211,
000e 0092       str r2, [sp]  @ ivtmp.804, %sfp
0010 B34A       ldr r2, .L22+8  @ lhs,
0012 1B68       ldr r3, [r3]  @ robin_tensor_arena.330_1,
0014 0392       str r2, [sp, #12] @ lhs, %sfp
0016 41F61062   movw  r2, #7696 @ ivtmp.774,
001a 03F11000   add r0, r3, #16 @ _126, robin_tensor_aren
001e 0293       str r3, [sp, #8]  @ ivtmp.802, %sfp
0020 9A18       adds  r2, r3, r2  @ ivtmp.774, ivtmp.802,
0022 0B44       add r3, r3, r1  @ _211, _211
0024 0490       str r0, [sp, #16] @ _126, %sfp
0026 0592       str r2, [sp, #20] @ ivtmp.774, %sfp
0028 0193       str r3, [sp, #4]  @ _211, %sfp
               .L4:
002a 039B       ldr r3, [sp, #12] @ lhs, %sfp
                @ lce-micro/third_party/tensorflow/./larq_c
002c 0027       movs  r7, #0  @ acc$3$0,
002e 41F61005   movw  r5, #6160 @ ivtmp.783,
0032 DDF814C0   ldr ip, [sp, #20] @ ivtmp.774, %sfp
0036 03F11006   add r6, r3, #16 @ ivtmp.781, lhs,
003a 3846       mov r0, r7  @ acc$2$0, acc$3$0
003c 5D19       adds  r5, r3, r5  @ ivtmp.783, lhs, ivtmp
003e BE46       mov lr, r7  @ result, result
0040 3B46       mov r3, r7  @ acc$1$0, acc$1$0
0042 00BFAFF3   .p2align 3,,7
     0080
```

Model flow image taken from: https://www.tensorflow.org/lite

The tasks of an inference engine

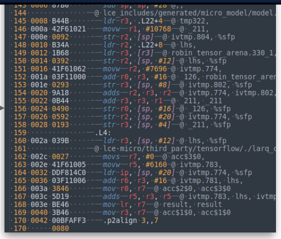
1. Execute the layers of the model in the correct order



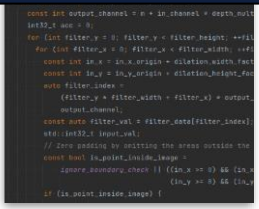
2. Plan the activations and weights in memory efficiently

ELPIDA TMN E1108ACBG -6E-E 0837A908U
ELPIDA TMN E1108ACBG -6E-E 0837A908U
ELPIDA TMN E1108ACBG -6E-E 0837A908U
ELPIDA TMN E1108ACBG -6E-E 0837A908U

3. Provide optimized INT8 code for each layer type (e.g. convolution, fully connected)



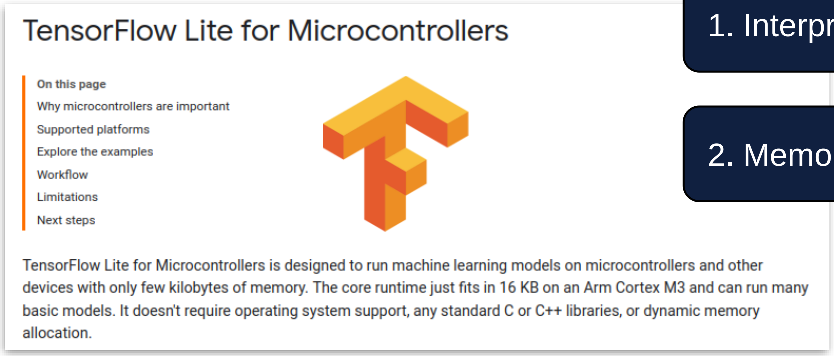
```
                        @ lce_includes/generated/micro_model/model.
0008 B44B           ldr  r3, .L22+4   @ tmp322,
000a 42F61021       movw  r1, #10768  @ _211,
000e 0092           str  r2, [sp]   @ ivtmp.804, %sfp
0010 B34A           ldr  r2, .L22+8   @ lhs,
0012 1B68           ldr  r3, [r3]   @ robin_tensor_arena.330_1,
0014 0392           str  r2, [sp, #12]  @ lhs, %sfp
0016 41F61062       movw  r2, #7696  @ ivtmp.774,
001a 03F11000       add  r0, r3, #16  @ _126, robin_tensor_aren
001e 0293           str  r3, [sp, #8]  @ ivtmp.802, %sfp
0020 9A18           adds  r2, r3, r2  @ ivtmp.774, ivtmp.802,
0022 0B44           add  r3, r3, r1  @ _211, _211
0024 0490           str  r0, [sp, #16]  @ _126, %sfp
0026 0592           str  r2, [sp, #20]  @ ivtmp.774, %sfp
0028 0193           str  r3, [sp, #4]  @ _211, %sfp
                    .L4:
002a 039B           ldr  r3, [sp, #12]  @ lhs, %sfp
                        @ lce-micro/third_party/tensorflow/./larq_c
002c 0027           movs  r7, #0  @ acc$3$0,
002e 41F61005       movw  r5, #6160  @ ivtmp.783,
0032 DDF814C0       ldr  ip, [sp, #20]  @ ivtmp.774, %sfp
0036 03F11006       add  r6, r3, #16  @ ivtmp.781, lhs,
003a 3846           mov  r0, r7  @ acc$2$0, acc$3$0
003c 5D19           adds  r5, r3, r5  @ ivtmp.783, lhs, ivtmp
003e BE46           mov  lr, r7  @ result, result
0040 3B46           mov  r3, r7  @ acc$1$0, acc$1$0
0042 00BFAFF3       .p2align 3,,7
     0080
```



```
const int output_channel = m + in_channel * depth_mult
int32_t acc = 0;
for (int filter_y = 0; filter_y < filter_height; ++fil
  for (int filter_x = 0; filter_x < filter_width; ++fi
    const int in_x = in_x_origin + dilation_width_fact
    const int in_y = in_y_origin + dilation_height_fac
    auto filter_index =
        (filter_y * filter_width + filter_x) * output_
        output_channel;
    const auto filter_val = filter_data[filter_index];
    std::int32_t input_val;
    // Zero padding by omitting the areas outside the
    const bool is_point_inside_image =
        ignore_boundary_check || ((in_x >= 0) && (in_x
                                  (in_y >= 0) && (in_y
    if (is_point_inside_image) {
```

Memory image from: https://commons.wikimedia.org/wiki/File:Swissbit_2GB_PC2-5300U-555.jpg

An inference engine example: TFLM



TensorFlow Lite for Microcontrollers

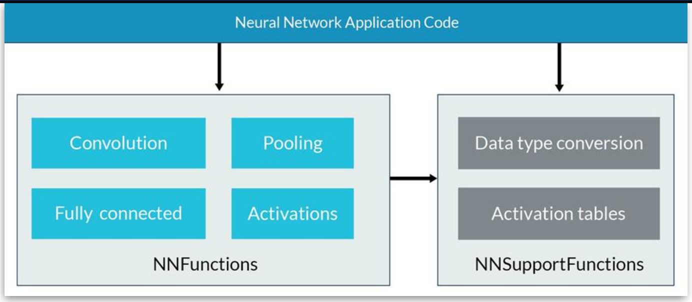
On this page
Why microcontrollers are important
Supported platforms
Explore the examples
Workflow
Limitations
Next steps

TensorFlow Lite for Microcontrollers is designed to run machine learning models on microcontrollers and other devices with only few kilobytes of memory. The core runtime just fits in 16 KB on an Arm Cortex M3 and can run many basic models. It doesn't require operating system support, any standard C or C++ libraries, or dynamic memory allocation.

1. Interpreter: TensorFlow Lite for Microcontrollers

2. Memory planner: TensorFlow Lite for Microcontrollers

3. Optimized INT8 code: ARM CMSIS-NN

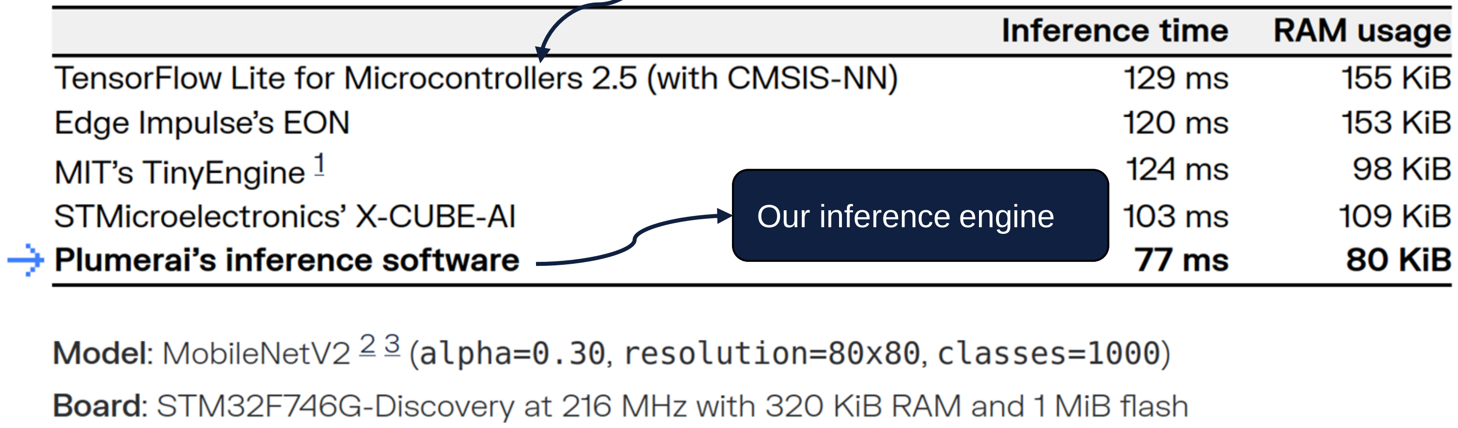


CMSIS image taken from: https://www.keil.com/pack/doc/CMSIS/NN/html/

2. Are we really that efficient?

A closer look at the results

What most people are using



	Inference time	RAM usage
TensorFlow Lite for Microcontrollers 2.5 (with CMSIS-NN)	129 ms	155 KiB
Edge Impulse's EON	120 ms	153 KiB
MIT's TinyEngine [1]	124 ms	98 KiB
STMicroelectronics' X-CUBE-AI	103 ms	109 KiB
Plumerai's inference software	**77 ms**	**80 KiB**

Model: MobileNetV2 [2] [3] (`alpha=0.30`, `resolution=80x80`, `classes=1000`)

Board: STM32F746G-Discovery at 216 MHz with 320 KiB RAM and 1 MiB flash

Our inference engine

Also tested: microTVM, but ran out of memory

No tricks: no binarization or pruning, accuracy remains the same in this table

Just good on MobileNetV2?

ite for M

CMSIS-N

EON

MIT's TinyEngine [1]

STMicroelectronics' X

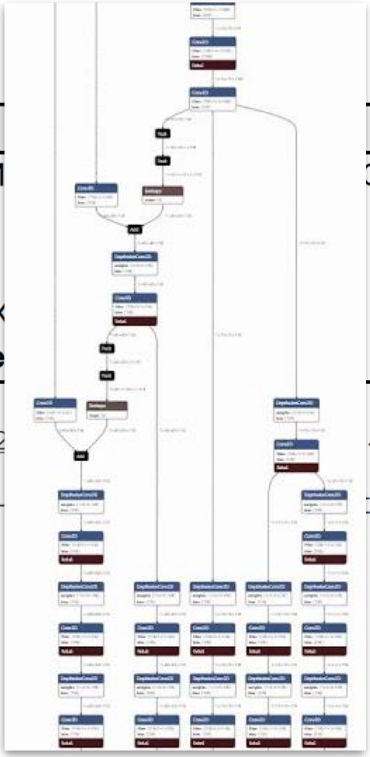
→ **Plumerai's inference**

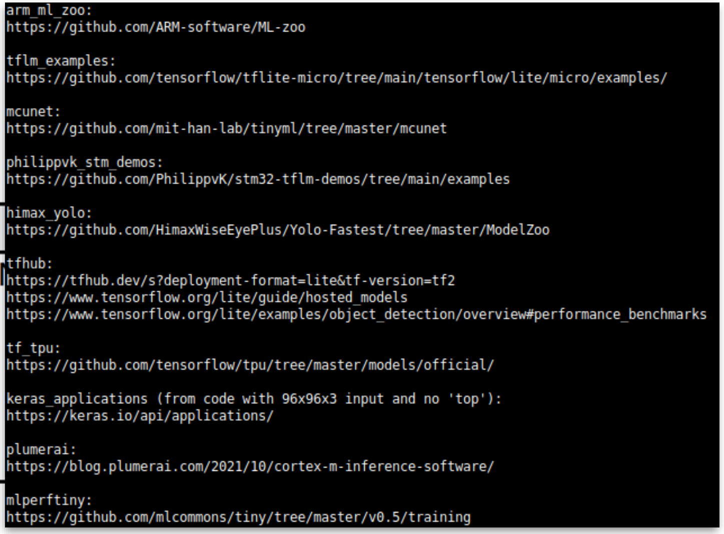
Model: MobileNetV2 [2]

tion=80x80, classes=1000)

Board: STM32F746G-

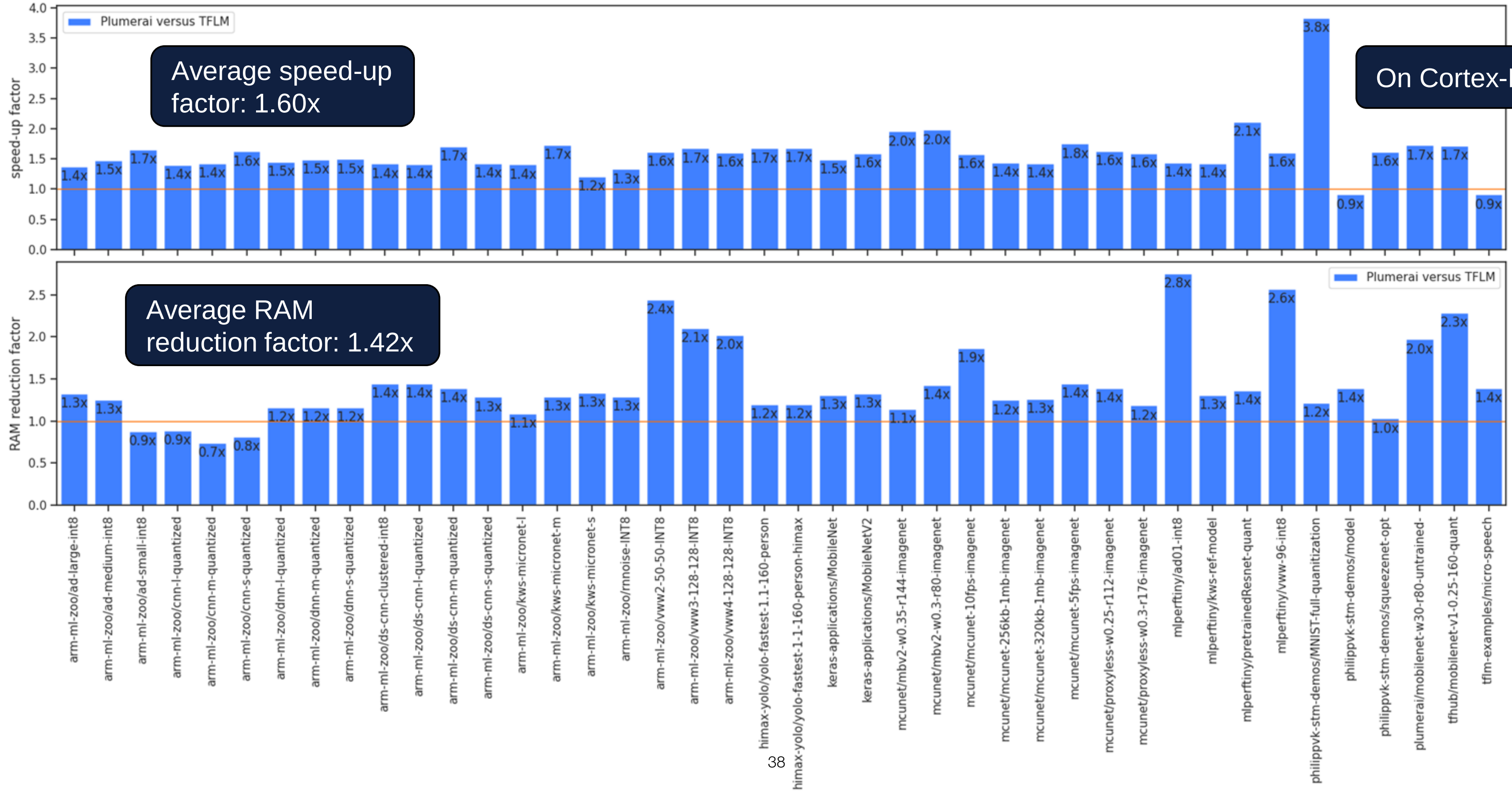
th 320 KiB RAM and 1 MiB flash





```
arm_ml_zoo:
https://github.com/ARM-software/ML-zoo

tflm_examples:
https://github.com/tensorflow/tflite-micro/tree/main/tensorflow/lite/micro/examples/

mcunet:
https://github.com/mit-han-lab/tinyml/tree/master/mcunet

philippvk_stm_demos:
https://github.com/PhilippvK/stm32-tflm-demos/tree/main/examples

himax_yolo:
https://github.com/HimaxWiseEyePlus/Yolo-Fastest/tree/master/ModelZoo

tfhub:
https://tfhub.dev/s?deployment-format=lite&tf-version=tf2
https://www.tensorflow.org/lite/guide/hosted_models
https://www.tensorflow.org/lite/examples/object_detection/overview#performance_benchmarks

tf_tpu:
https://github.com/tensorflow/tpu/tree/master/models/official/

keras_applications (from code with 96x96x3 input and no 'top'):
https://keras.io/api/applications/

plumerai:
https://blog.plumerai.com/2021/10/cortex-m-inference-software/

mlperftiny:
https://github.com/mlcommons/tiny/tree/master/v0.5/training
```

More off-the-shelf models

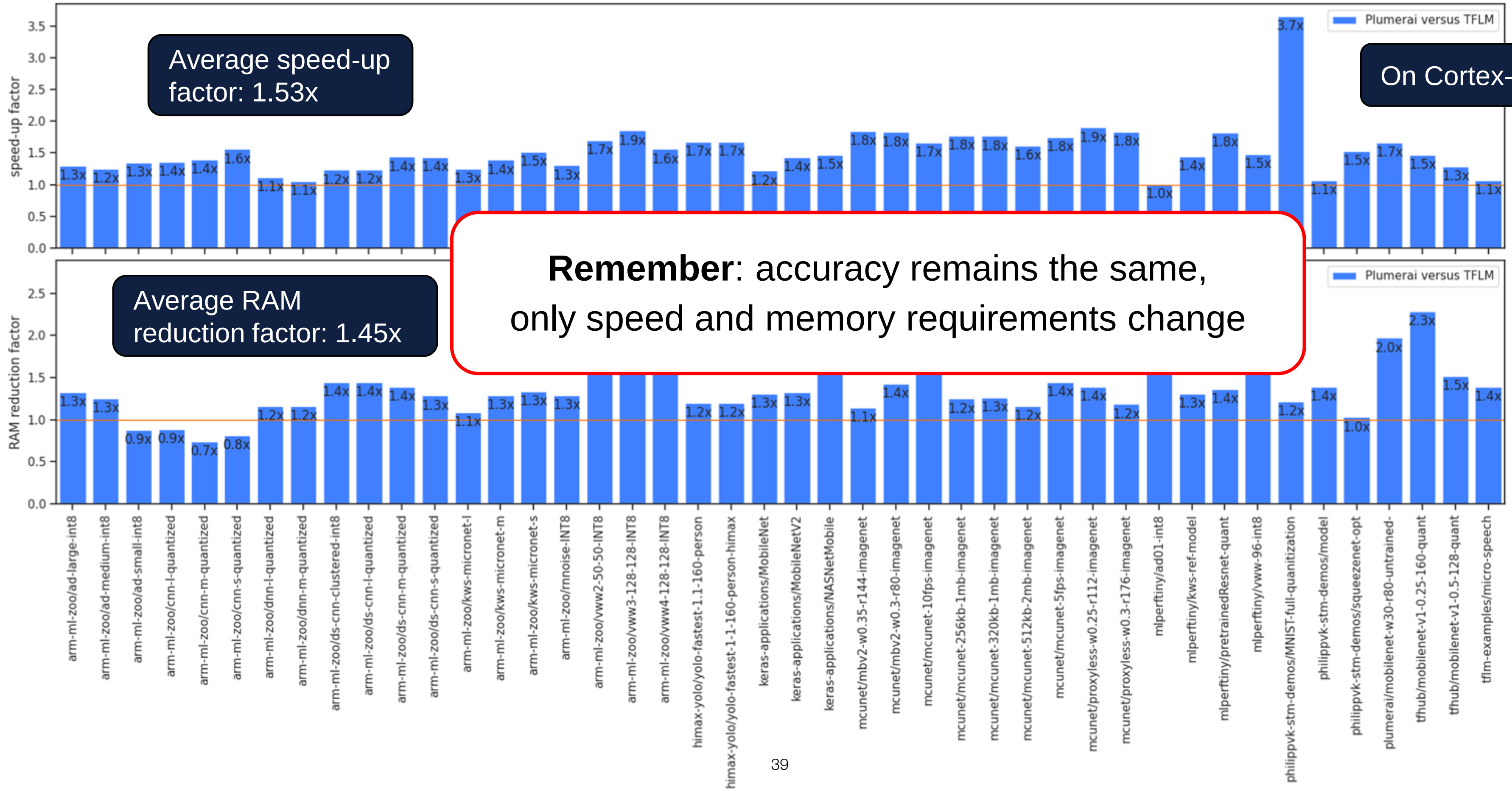


Average speed-up factor: 1.60x

On Cortex-M4

Average RAM reduction factor: 1.42x

More off-the-shelf models



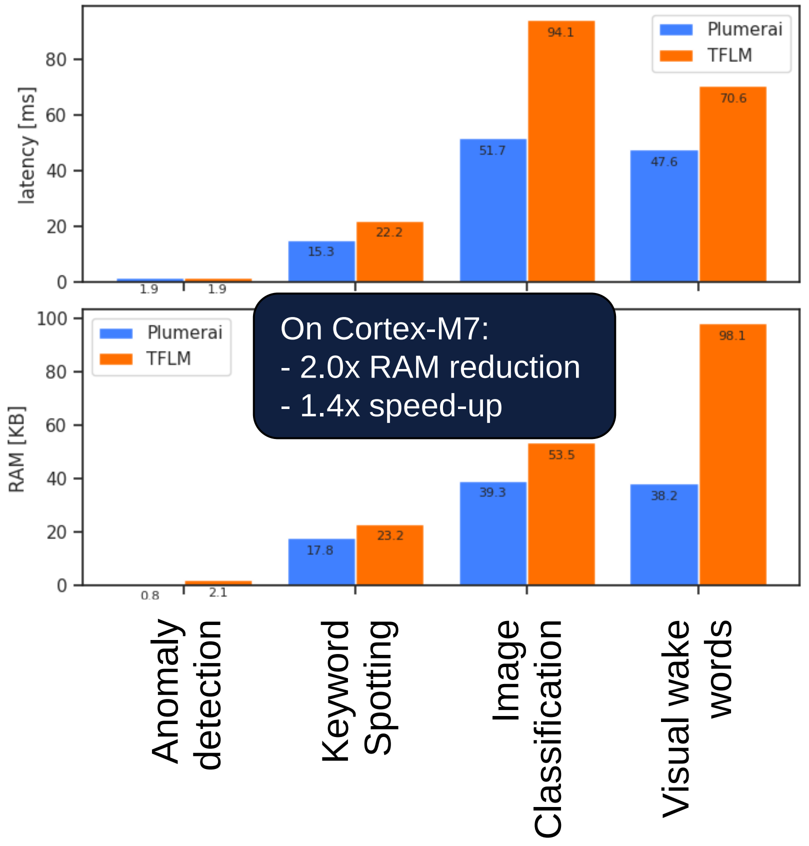
Average speed-up factor: 1.53x

On Cortex-M7

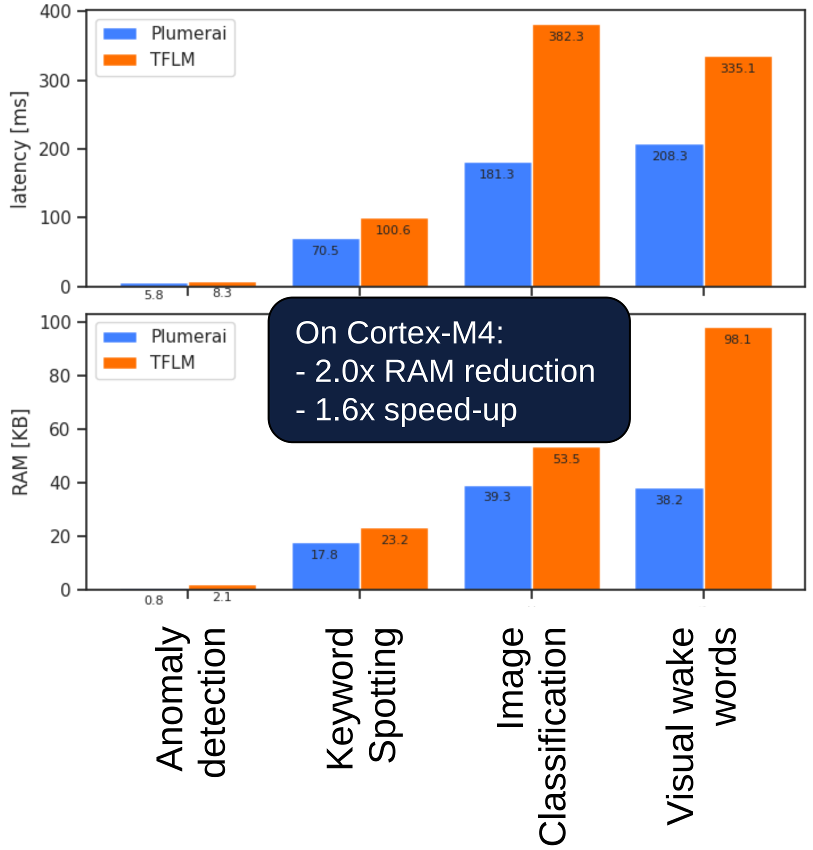
Average RAM reduction factor: 1.45x

Remember: accuracy remains the same, only speed and memory requirements change

A closer look at the MLPerf Tiny models



On Cortex-M7:
- 2.0x RAM reduction
- 1.4x speed-up

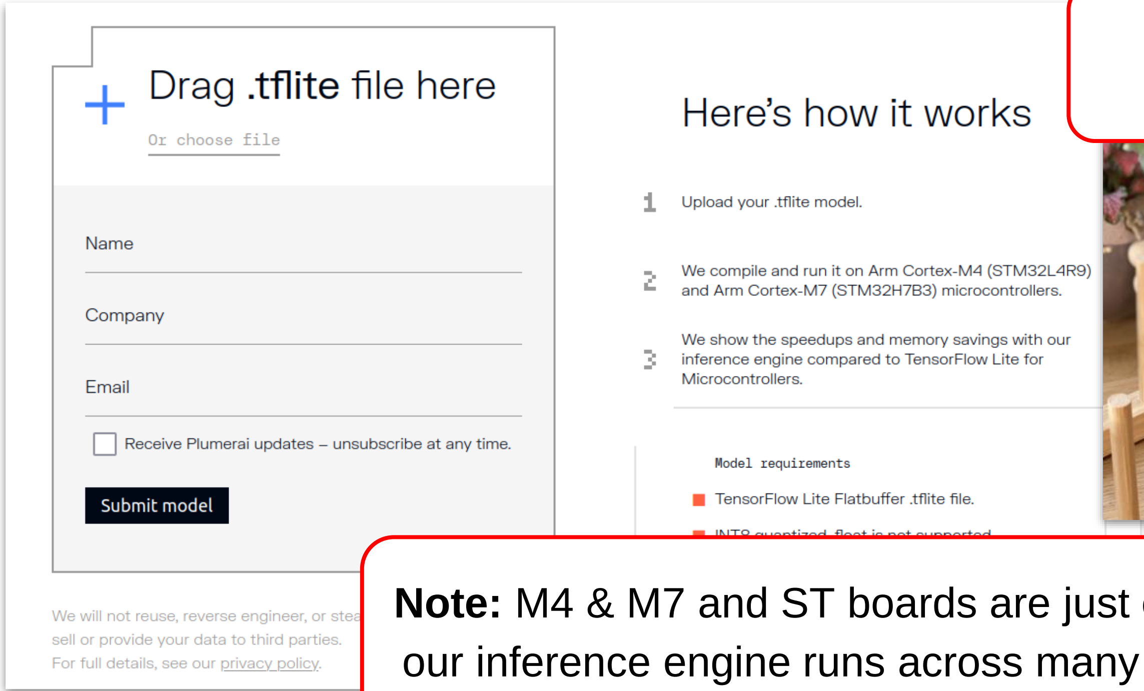


On Cortex-M4:
- 2.0x RAM reduction
- 1.6x speed-up

Plumerai

3. Live demo of public benchmarking service

Public benchmarking service: try it yourself!



Drag .tflite file here

Or choose file

Name

Company

Email

☐ Receive Plumerai updates – unsubscribe at any time.

Submit model

We will not reuse, reverse engineer, or stea
sell or provide your data to third parties.
For full details, see our privacy policy.

Here's how it works

1 Upload your .tflite model.

2 We compile and run it on Arm Cortex-M4 (STM32L4R9) and Arm Cortex-M7 (STM32H7B3) microcontrollers.

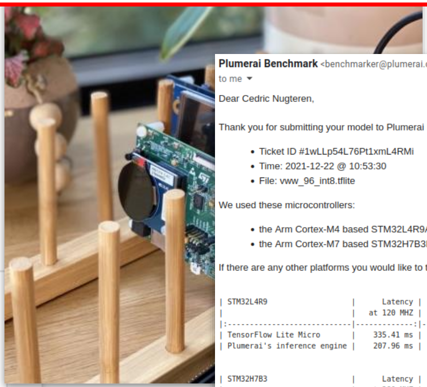
3 We show the speedups and memory savings with our inference engine compared to TensorFlow Lite for Microcontrollers.

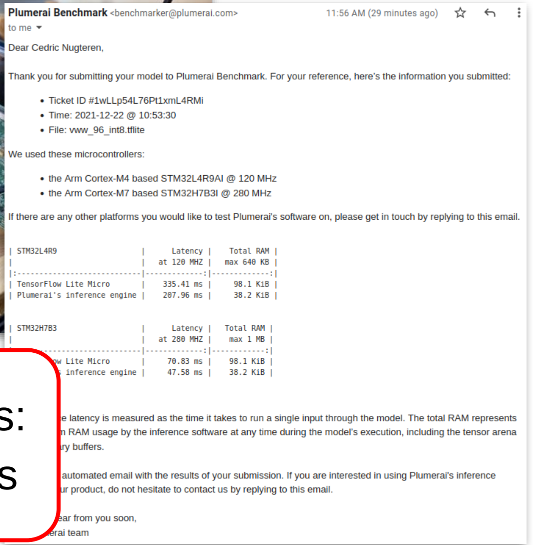
Model requirements

🟥 TensorFlow Lite Flatbuffer .tflite file.

🟥 INT8 quantized, float is not supported

Any model that runs with TFLM is supported!





Plumerai Benchmark <benchmarker@plumerai.com> 11:56 AM (29 minutes ago)
to me

Dear Cedric Nugteren,

Thank you for submitting your model to Plumerai Benchmark. For your reference, here's the information you submitted:

- Ticket ID #1wLLp54L76Pt1xmL4RMi
- Time: 2021-12-22 @ 10:53:30
- File: vww_96_int8.tflite

We used these microcontrollers:

- the Arm Cortex-M4 based STM32L4R9AI @ 120 MHz
- the Arm Cortex-M7 based STM32H7B3I @ 280 MHz

If there are any other platforms you would like to test Plumerai's software on, please get in touch by replying to this email.

```
| STM32L4R9                   |      Latency |    Total RAM |
|                             |   at 120 MHZ |   max 640 KB |
|:----------------------------|-------------:|-------------:|
| TensorFlow Lite Micro       |    335.41 ms |     98.1 KiB |
| Plumerai's inference engine |    207.96 ms |     38.2 KiB |


| STM32H7B3                   |      Latency |   Total RAM |
|                             |   at 280 MHZ |    max 1 MB |
  -------------------------|-------------:|------------:|
  ow Lite Micro            |     70.83 ms |    98.1 KiB |
  s inference engine |     47.58 ms |    38.2 KiB |
```

e latency is measured as the time it takes to run a single input through the model. The total RAM represents
n RAM usage by the inference software at any time during the model's execution, including the tensor arena
ry buffers.

automated email with the results of your submission. If you are interested in using Plumerai's inference
ur product, do not hesitate to contact us by replying to this email.

ear from you soon,
erai team

Note: M4 & M7 and ST boards are just examples: our inference engine runs across many platforms

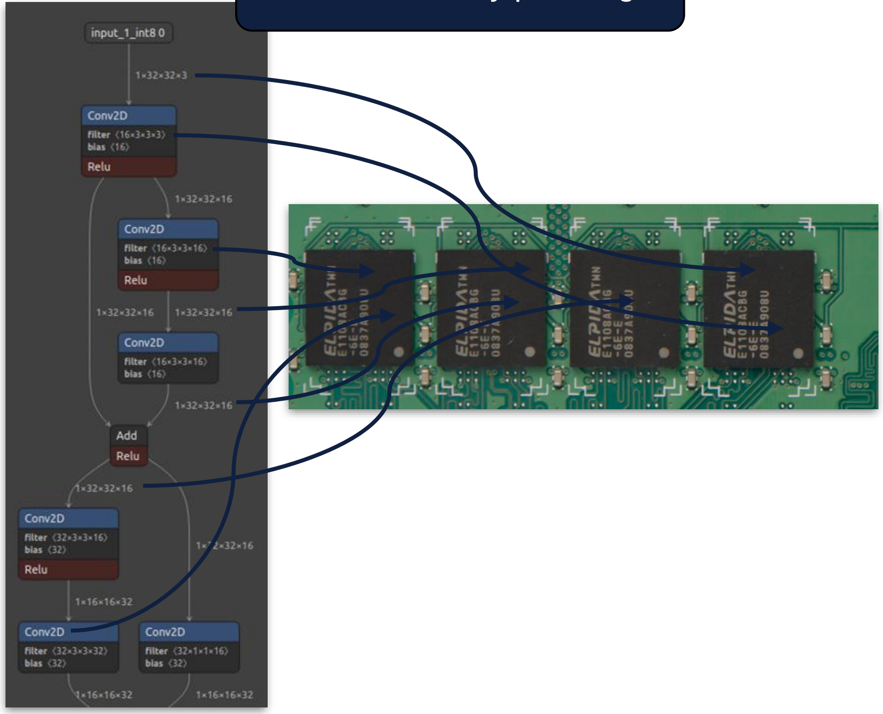
Visit https://plumerai.com/benchmark to try it with your own model

Plumerai

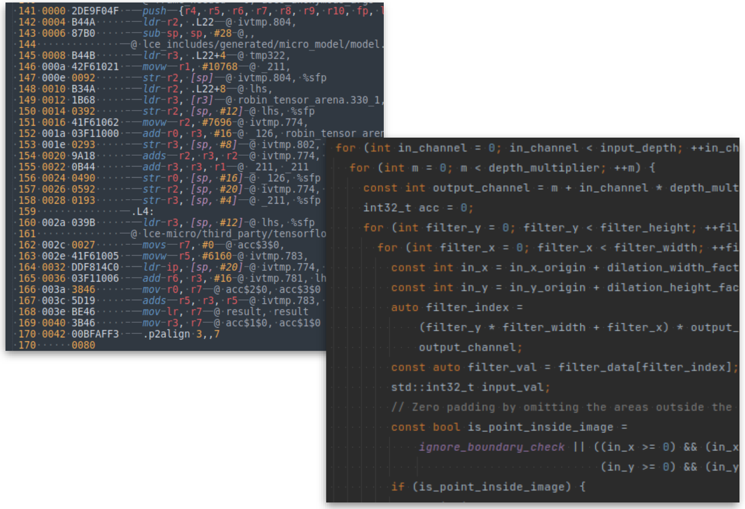
4. What did we do to become so efficient?

How to beat the competition?

1. Better memory planning



2. Optimized and model-specific INT8 code for Cortex-M



```
0000 2DE9F04F   push {r4, r5, r6, r7, r8, r9, r10, fp,
0004 B44A       ldr r2, .L22 @ ivtmp.804,
0006 87B0       sub sp, sp, #28 @,,
                @ lce_includes/generated/micro_model/model.
0008 B44B       ldr r3, .L22+4 @ tmp322,
000a 42F61021   movw r1, #10768 @ _211,
000e 0092       str r2, [sp] @ ivtmp.804, %sfp
0010 B34A       ldr r2, .L22+8 @ lhs,
0012 1B68       ldr r3, [r3] @ robin_tensor_arena.330_1,
0014 0392       str r2, [sp, #12] @ lhs, %sfp
0016 41F61062   movw r2, #7696 @ ivtmp.774,
001a 03F11000   add r0, r3, #16 @ _126, robin_tensor_aren
001e 0293       str r3, [sp, #8] @ ivtmp.802,
0020 9A18       adds r2, r3, r2 @ ivtmp.774,
0022 0B44       add r3, r3, r1 @ _211, _211
0024 0490       str r0, [sp, #16] @ _126, %sfp
0026 0592       str r2, [sp, #20] @ ivtmp.774,
0028 0193       str r3, [sp, #4] @ _211, %sfp
                .L4:
002a 039B       ldr r3, [sp, #12] @ lhs, %sfp
                @ lce-micro/third_party/tensorflo
002c 0027       movs r7, #0 @ acc$3$0,
002e 41F61005   movw r5, #6160 @ ivtmp.783,
0032 DDF814C0   ldr ip, [sp, #20] @ ivtmp.774,
0036 03F11006   add r6, r3, #16 @ ivtmp.781, lh
003a 3846       mov r0, r7 @ acc$2$0, acc$3$0
003c 5D19       adds r5, r3, r5 @ ivtmp.783,
003e BE46       mov lr, r7 @ result, result
0040 3B46       mov r3, r7 @ acc$1$0, acc$1$0
0042 00BFAFF3   .p2align 3,,7
     0080
```

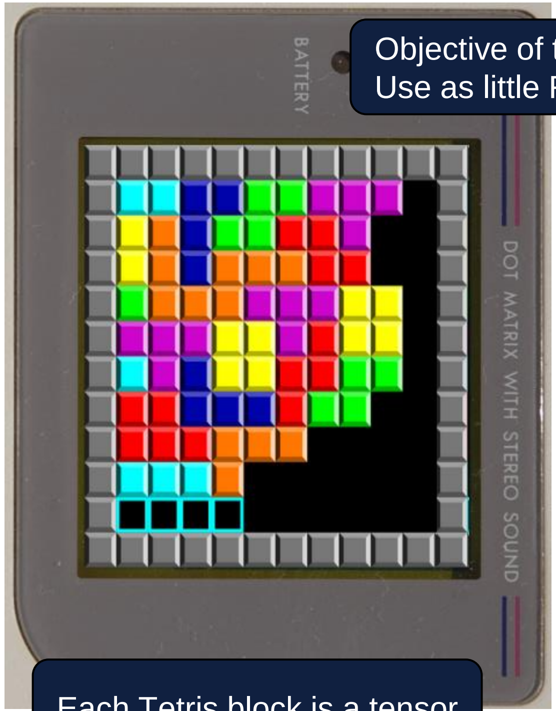
```
for (int in_channel = 0; in_channel < input_depth; ++in_ch
  for (int m = 0; m < depth_multiplier; ++m) {
    const int output_channel = m + in_channel * depth_mult
    int32_t acc = 0;
    for (int filter_y = 0; filter_y < filter_height; ++fil
      for (int filter_x = 0; filter_x < filter_width; ++fi
        const int in_x = in_x_origin + dilation_width_fact
        const int in_y = in_y_origin + dilation_height_fac
        auto filter_index =
            (filter_y * filter_width + filter_x) * output_
            output_channel;
        const auto filter_val = filter_data[filter_index];
        std::int32_t input_val;
        // Zero padding by omitting the areas outside the
        const bool is_point_inside_image =
            ignore_boundary_check || ((in_x >= 0) && (in_x
                                      (in_y >= 0) && (in_y
        if (is_point_inside_image) {
```

Memory image from: https://commons.wikimedia.org/wiki/File:Swissbit_2GB_PC2-5300U-555.jpg

Memory planning: a (rotated) game of Tetris

RAM size →

Time (layer execution) →

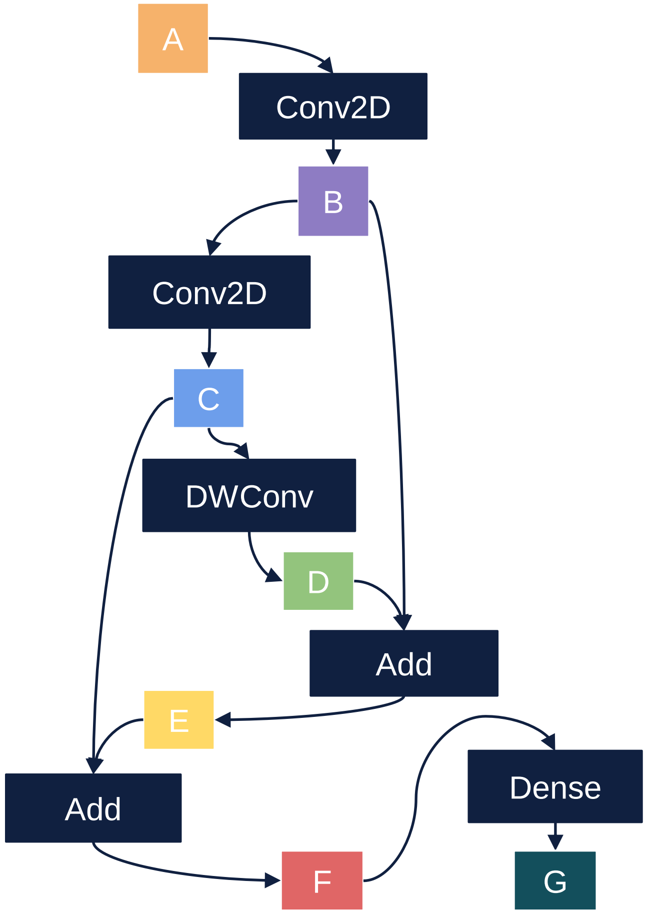


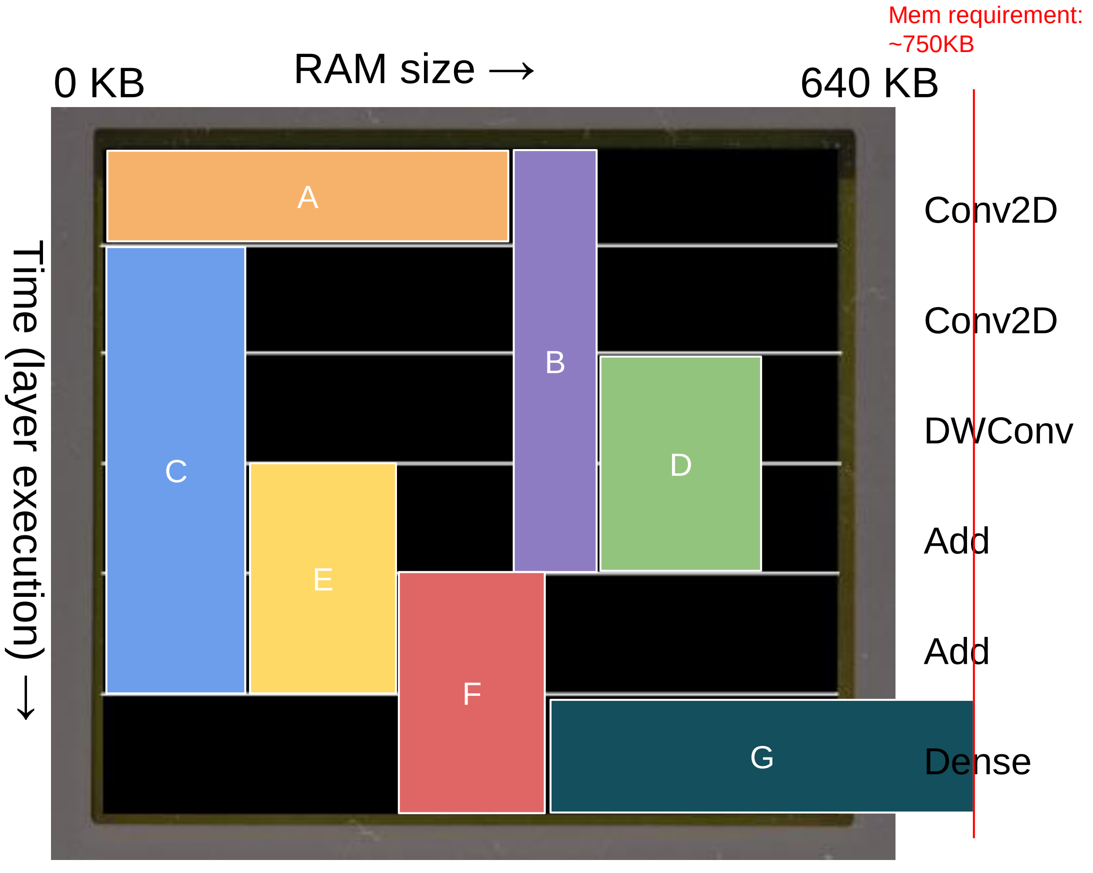
Objective of the game:
Use as little RAM as possible

Each Tetris block is a tensor

Image taken from: https://commons.wikimedia.org/wiki/File:Gameboy.jpg

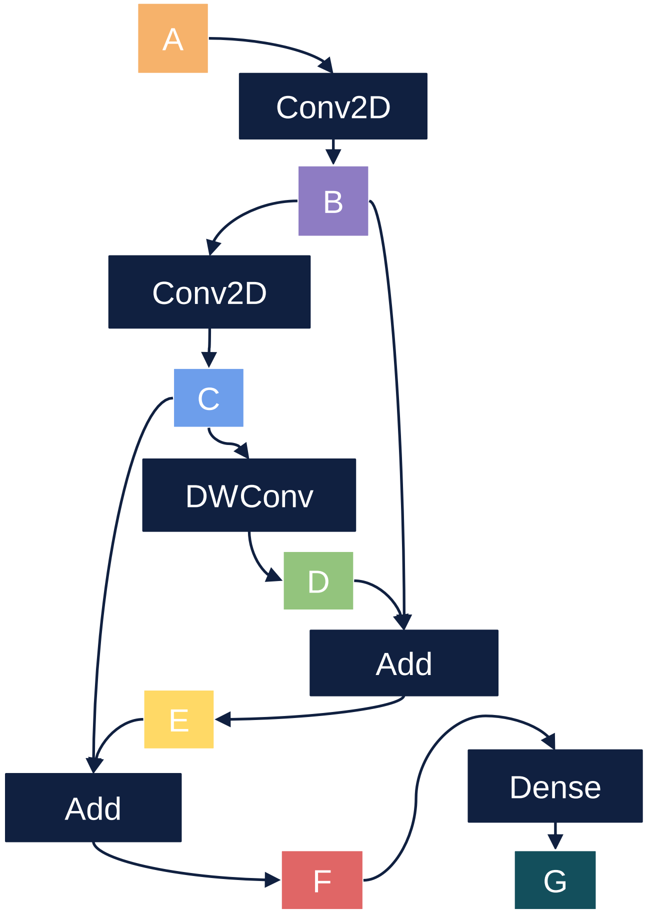
Memory planning for an example model

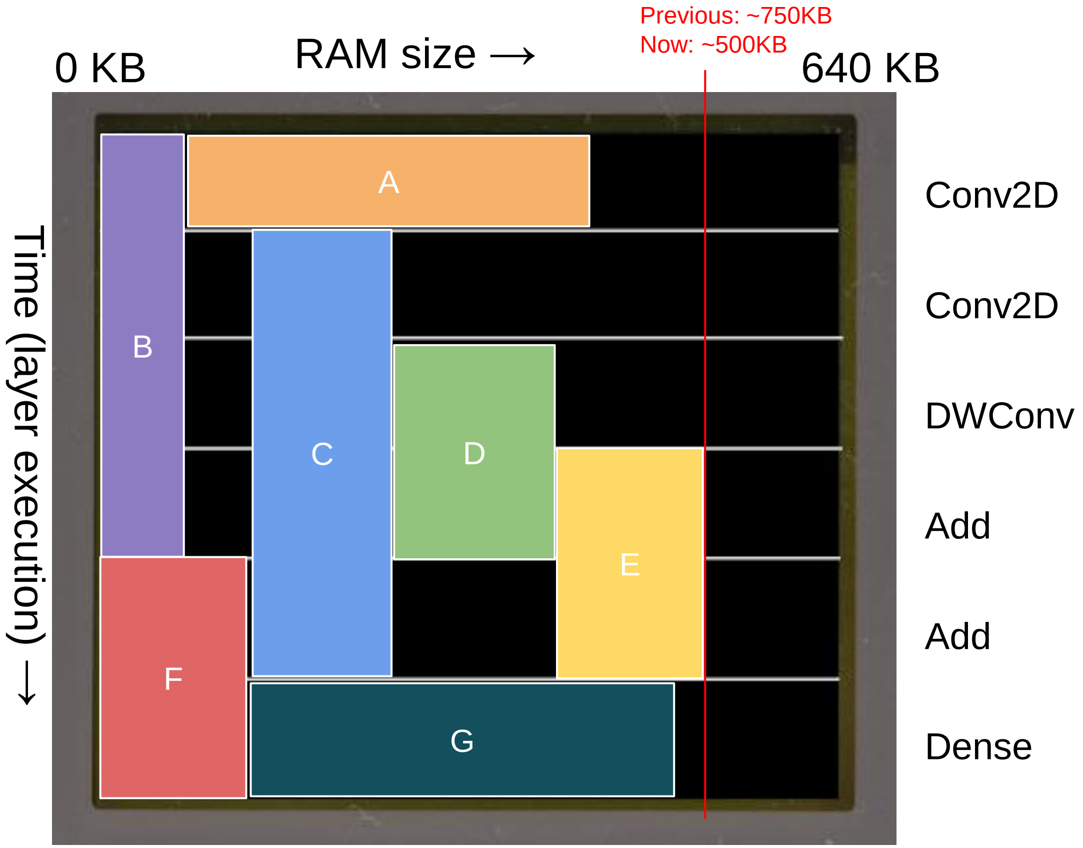




(simplified view: only activations are planned here, not weights)

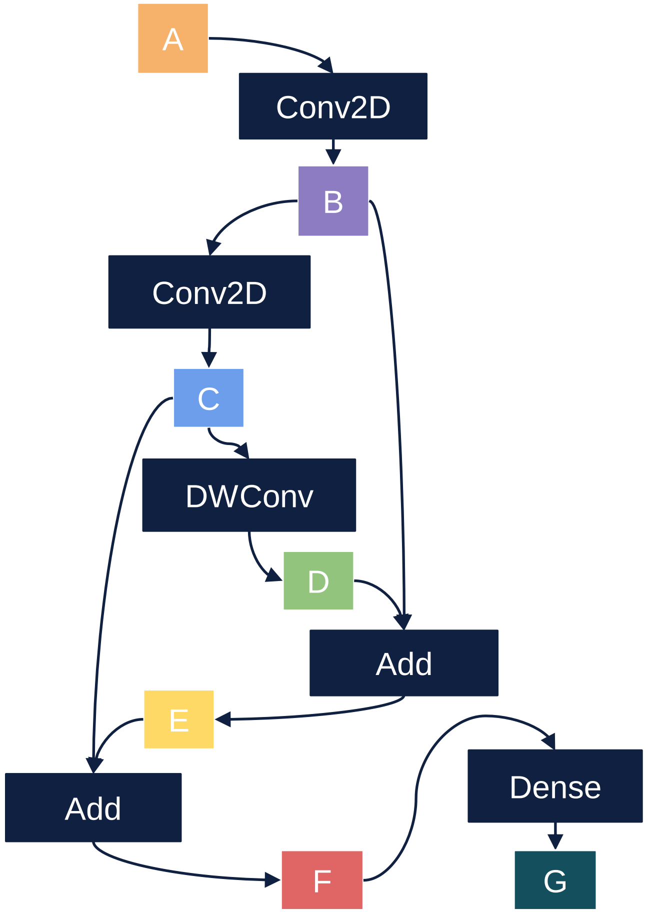
A much better memory plan

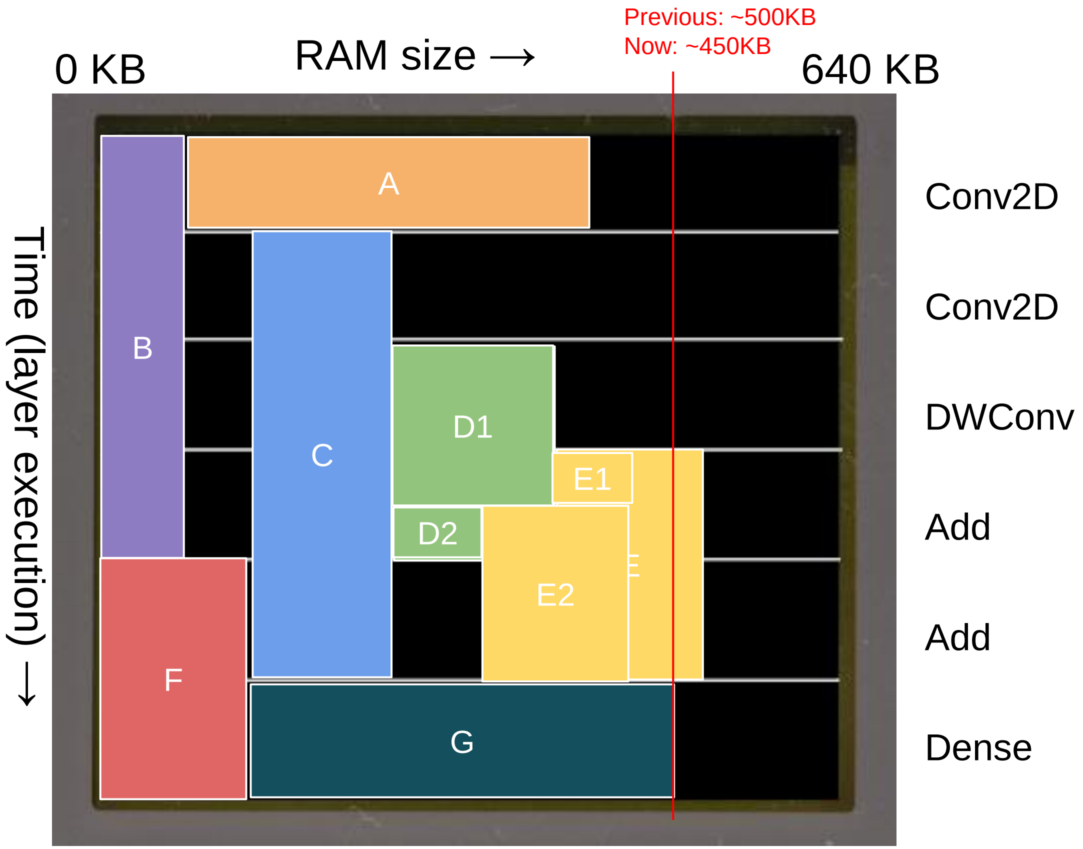




(simplified view: only activations are planned here, not weights)

Even better: lower granularity planning





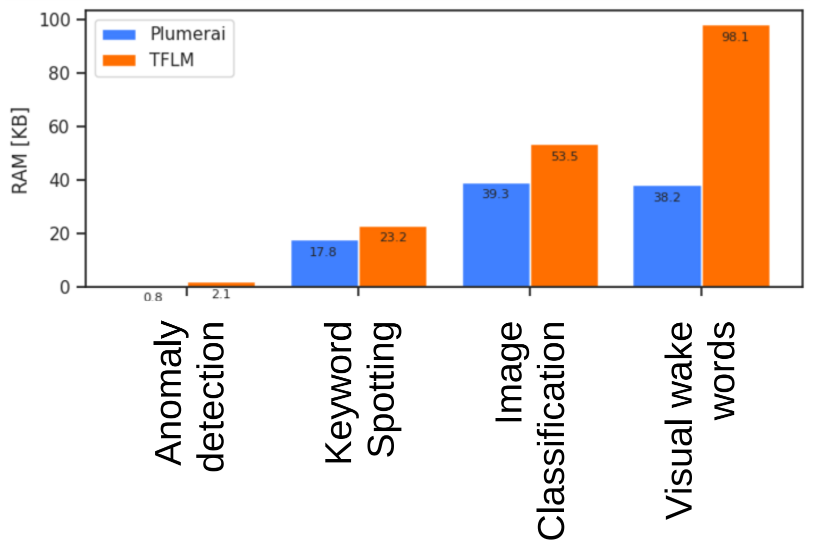
(simplified view: only activations are planned here, not weights)

Memory planning at Plumerai: summary

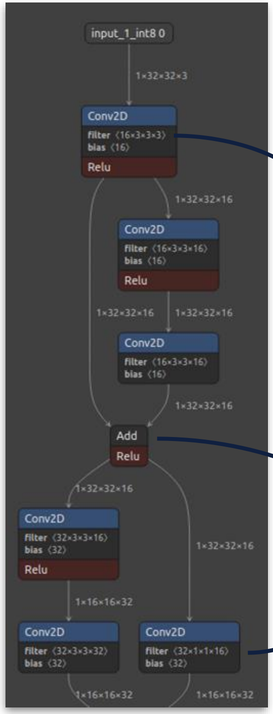
Lower RAM usage:

1. Smarter tensor placement
2. Lower granularity planning

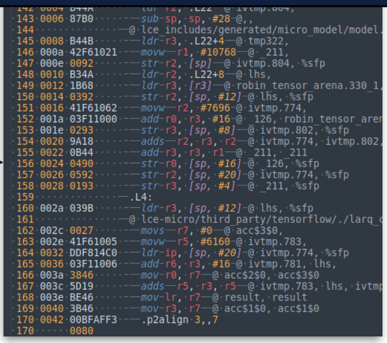
RAM savings highly model dependent!



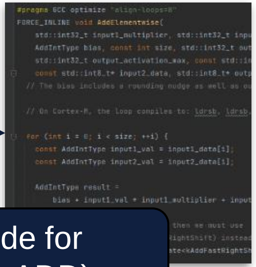
Optimized INT8 code for speed



Optimized Conv2D code:
im2col + GEMM



```
0004 B44A        ldr r2, .L22  @ ivtmp.804,
0006 87B0        sub sp, sp, #28 @,,
                 @ lce_includes/generated/micro_model/model.
0008 B44B        ldr r3, .L22+4  @ tmp322,
000a 42F61021    movw  r1, #10768  @ _211,
000e 0092        str r2, [sp]  @ ivtmp.804, %sfp
0010 B34A        ldr r2, .L22+8  @ lhs,
0012 1B68        ldr r3, [r3]  @ robin_tensor_arena.330_1,
0014 0392        str r2, [sp, #12] @ lhs, %sfp
0016 41F61062    movw  r2, #7696 @ ivtmp.774,
001a 03F11000    add r0, r3, #16 @ _126, robin_tensor_aren
001e 0293        str r3, [sp, #8]  @ ivtmp.802, %sfp
0020 9A18        adds  r2, r3, r2  @ ivtmp.774, ivtmp.802,
0022 0B44        add r3, r3, r1  @ _211, _211
0024 0490        str r0, [sp, #16] @ _126, %sfp
0026 0592        str r2, [sp, #20] @ ivtmp.774, %sfp
0028 0193        str r3, [sp, #4]  @ _211, %sfp
               .L4:
002a 039B        ldr r3, [sp, #12] @ lhs, %sfp
                 @ lce-micro/third_party/tensorflow/./larq_c
002c 0027        movs  r7, #0  @ acc$3$0,
002e 41F61005    movw  r5, #6160 @ ivtmp.783,
0032 DDF814C0    ldr ip, [sp, #20] @ ivtmp.774, %sfp
0036 03F11006    add r6, r3, #16 @ ivtmp.781, lhs,
003a 3846        mov r0, r7  @ acc$2$0, acc$3$0
003c 5D19        adds  r5, r3, r5  @ ivtmp.783, lhs, ivtmp
003e BE46        mov lr, r7  @ result, result
0040 3B46        mov r3, r7  @ acc$1$0, acc$1$0
0042 00BFAFF3    .p2align 3,,7
     0080
```



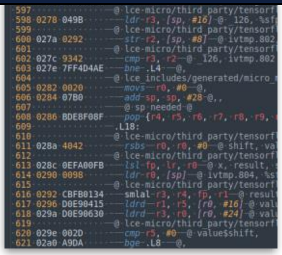
```
#pragma GCC optimize "align-loops=8"
FORCE_INLINE void AddElementwise(
    std::int32_t input1_multiplier, std::int32_t inpu
    AddIntType bias, const int size, std::int32_t out
    std::int32_t output_activation_max, const std::in
    const std::int8_t* input2_data, std::int8_t* outp
  // The bias includes a rounding nudge as well as ou

  // On Cortex-M, the loop compiles to: ldrsb, ldrsb,

  for (int i = 0; i < size; ++i) {
    const AddIntType input1_val = input1_data[i];
    const AddIntType input2_val = input2_data[i];

    AddIntType result =
        bias + input1_val * input1_multiplier + input

                            then we must use
                            RightShift) instead
                            ate<kAddFastRightSh
```

Optimized code for
other op (e.g. ADD):

Example code optimizations:

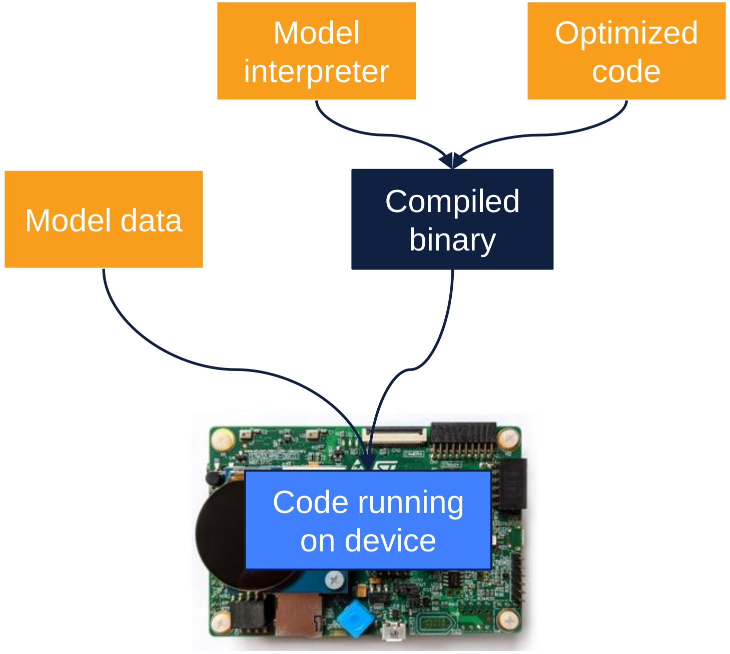
- Hand-written assembly (if needed)
- Specialization for Cortex-M4 or M7 capabilities
- Register-count aware optimisations
- Template-based loop unrolling
- Weight memory layout pre-processing

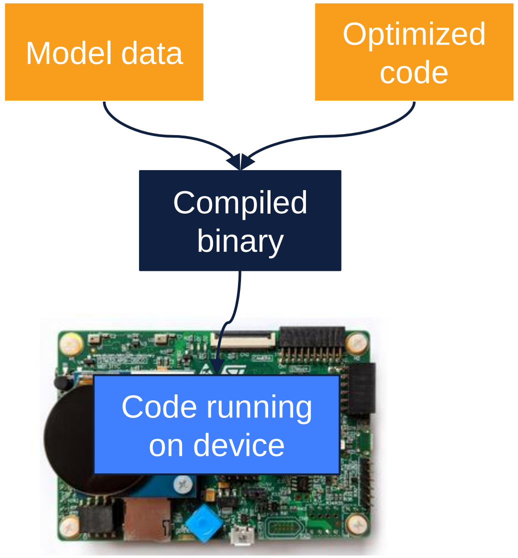
Optimized code for special
cases, e.g. 1x1 Conv2D



```
                 @ lce-micro/third_party/tensorf
0278 049B        ldr r3, [sp, #16] @ _126, %sf
                 @ lce-micro/third_party/tensorf
027a 0292        str r2, [sp, #8]  @ ivtmp.802
                 @ lce-micro/third_party/tensorf
027c 9342        cmp r3, r2  @ _126, ivtmp.802
027e 7FF4D4AE    bne .L4  @,
                 @ lce_includes/generated/micro_
0282 0020        movs  r0, #0  @,
0284 07B0        add sp, sp, #28 @,,
                 @ sp needed @
0286 BDE8F08F    pop {r4, r5, r6, r7, r8, r9,
               .L18:
                 @ lce-micro/third_party/tensorf
028a 4042        rsbs  r0, r0, #0  @ shift, val
                 @ lce-micro/third_party/tensorf
028c 0EFA00FB    lsl fp, lr, r0  @ x, result,
0290 0098        ldr r0, [sp]  @ ivtmp.804, %sf
                 @ lce-micro/third_party/tensorf
0292 CBFB0134    smlal r3, r4, fp, r1  @ result
0296 D0E90415    ldrd  r1, r5, [r0, #16] @ valu
029a D0E90630    ldrd  r3, r0, [r0, #24] @ valu
                 @ lce-micro/third_party/tensorf
029e 002D        cmp r5, #0  @ value$shift,
02a0 A9DA        bge .L8  @,
```

Model-agnostic - vs - model-specific





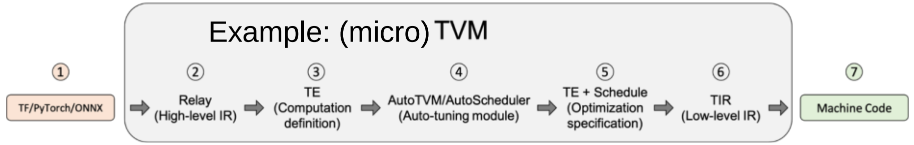
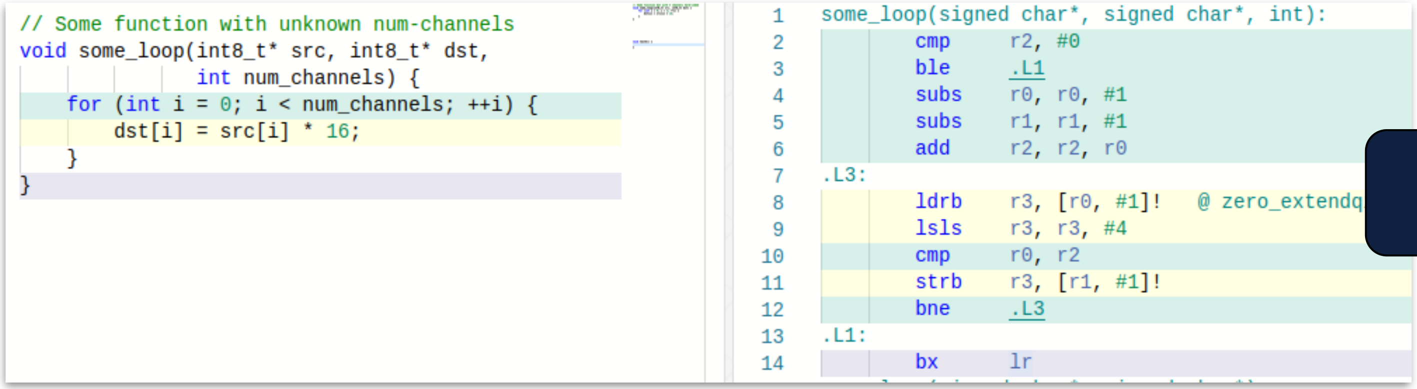

Example: (micro) TVM
① TF/PyTorch/ONNX
② Relay (High-level IR)
③ TE (Computation definition)
④ AutoTVM/AutoScheduler (Auto-tuning module)
⑤ TE + Schedule (Optimization specification)
⑥ TIR (Low-level IR)
⑦ Machine Code

Image taken from: https://tvm.apache.org/docs/tutorial/introduction.html

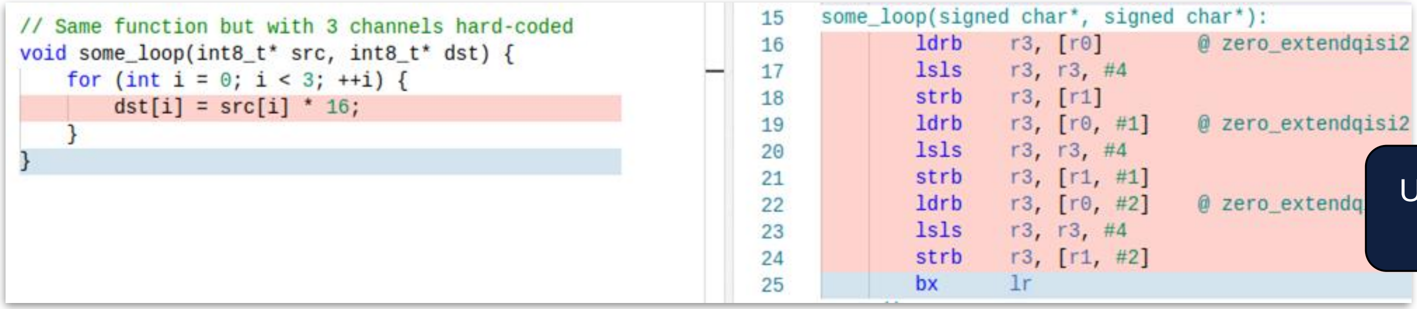
Model-specific code generation



```c
// Some function with unknown num-channels
void some_loop(int8_t* src, int8_t* dst,
               int num_channels) {
    for (int i = 0; i < num_channels; ++i) {
        dst[i] = src[i] * 16;
    }
}
```

```
some_loop(signed char*, signed char*, int):
        cmp     r2, #0
        ble     .L1
        subs    r0, r0, #1
        subs    r1, r1, #1
        add     r2, r2, r0
.L3:
        ldrb    r3, [r0, #1]!    @ zero_extendq
        lsls    r3, r3, #4
        cmp     r0, r2
        strb    r3, [r1, #1]!
        bne     .L3
.L1:
        bx      lr
```

Generic code with compare and branch instructions



```c
// Same function but with 3 channels hard-coded
void some_loop(int8_t* src, int8_t* dst) {
    for (int i = 0; i < 3; ++i) {
        dst[i] = src[i] * 16;
    }
}
```

```
some_loop(signed char*, signed char*):
        ldrb    r3, [r0]         @ zero_extendqisi2
        lsls    r3, r3, #4
        strb    r3, [r1]
        ldrb    r3, [r0, #1]     @ zero_extendqisi2
        lsls    r3, r3, #4
        strb    r3, [r1, #1]
        ldrb    r3, [r0, #2]     @ zero_extendq
        lsls    r3, r3, #4
        strb    r3, [r1, #2]
        bx      lr
```

Unrolled code with only add, load and stores

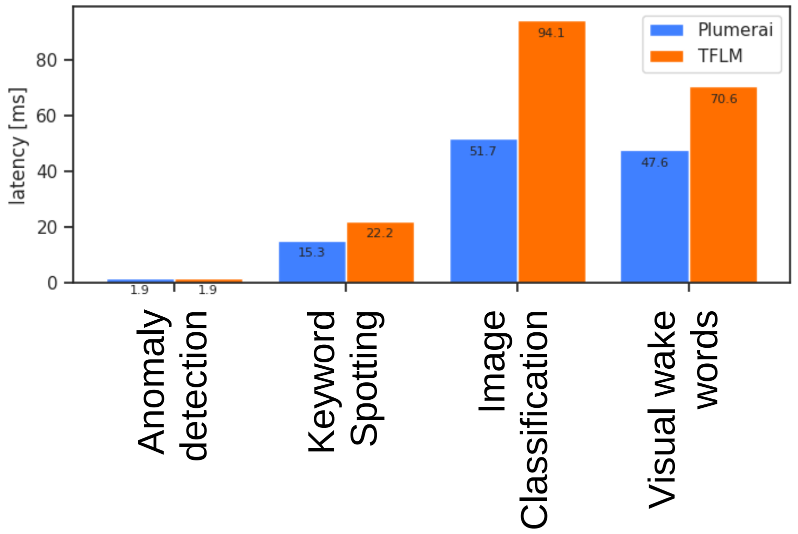
From the Godbolt compiler explorer at: https://godbolt.com

Better speed at Plumerai: summary

Lower latency, better speed:

1. Optimized code for Cortex-M
2. Model-specific code generation

Latency savings dependent on the layers and layer configurations

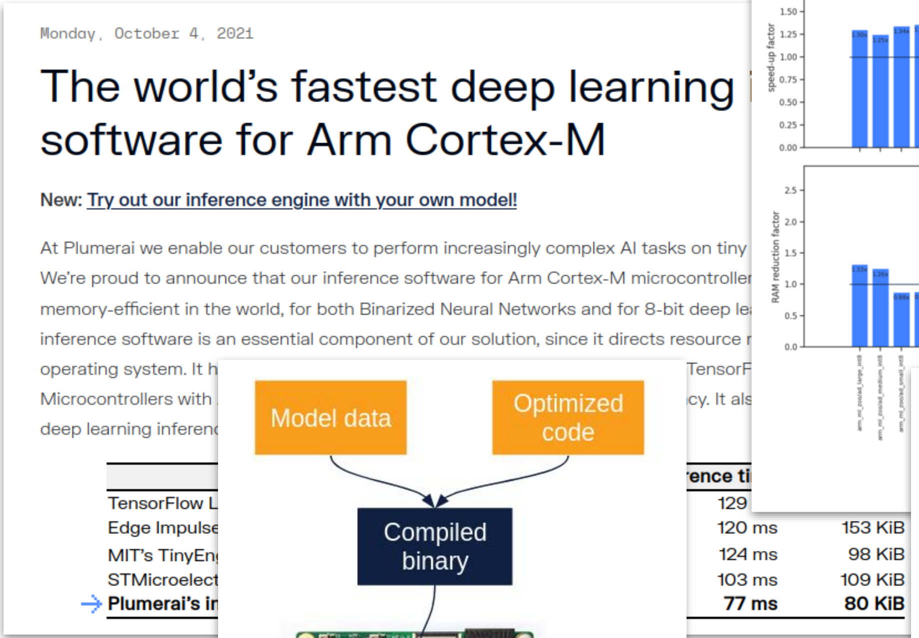


You’ve got mail!

Plumerai

5. Conclusion

The world’s fastest Cortex-M inference



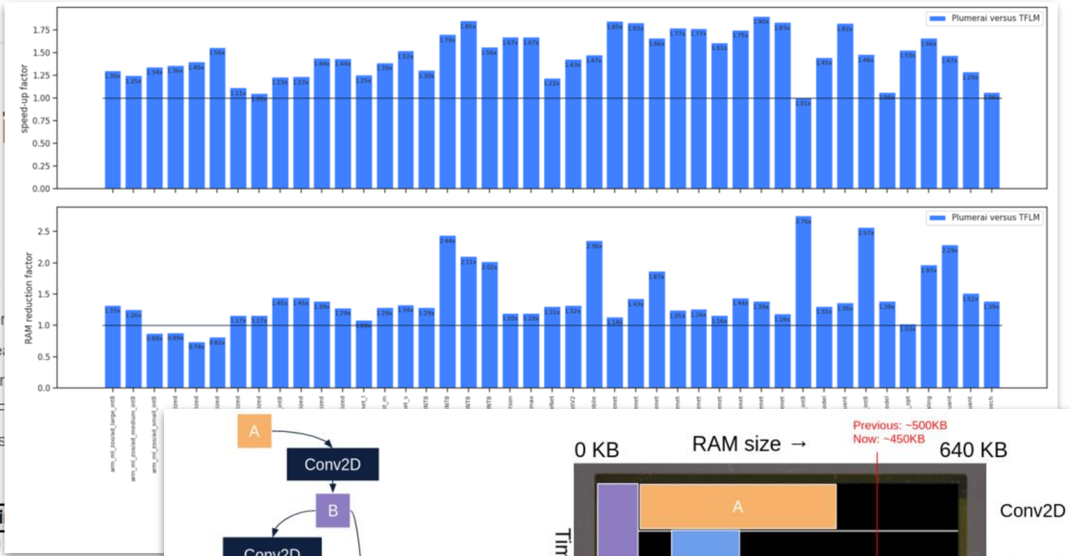
Monday, October 4, 2021

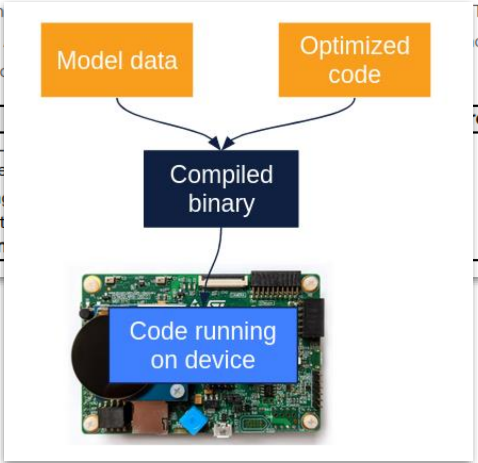
The world’s fastest deep learning i
software for Arm Cortex-M

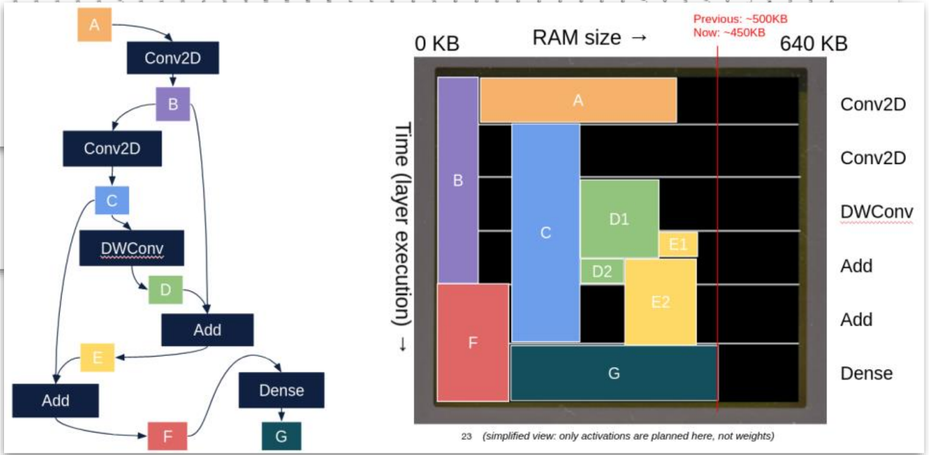
New: Try out our inference engine with your own model!

At Plumerai we enable our customers to perform increasingly complex AI tasks on tiny
We’re proud to announce that our inference software for Arm Cortex-M microcontroller
memory-efficient in the world, for both Binarized Neural Networks and for 8-bit deep le
inference software is an essential component of our solution, since it directs resource r
operating system. It h TensorF
Microcontrollers with cy. It als
deep learning inferen

	ence ti	
TensorFlow L	129	
Edge Impulse	120 ms	153 KiB
MIT’s TinyEn	124 ms	98 KiB
STMicroelect	103 ms	109 KiB
Plumerai’s in	**77 ms**	**80 KiB**

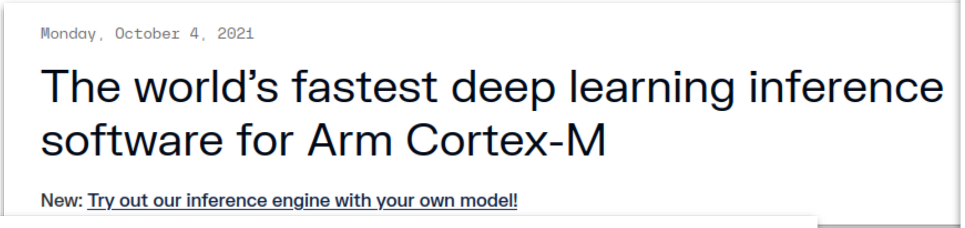






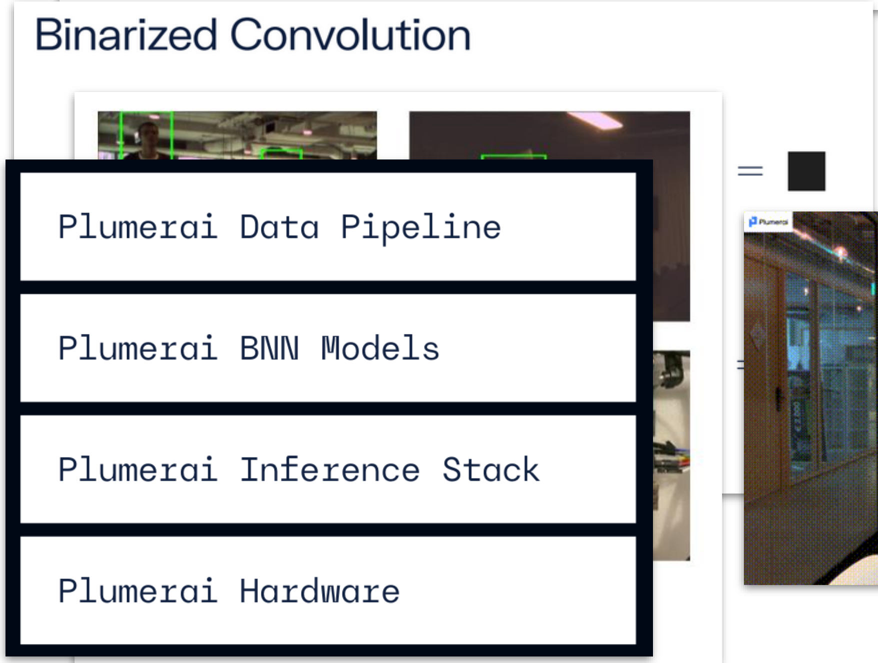
23 (simplified view: only activations are planned here, not weights)

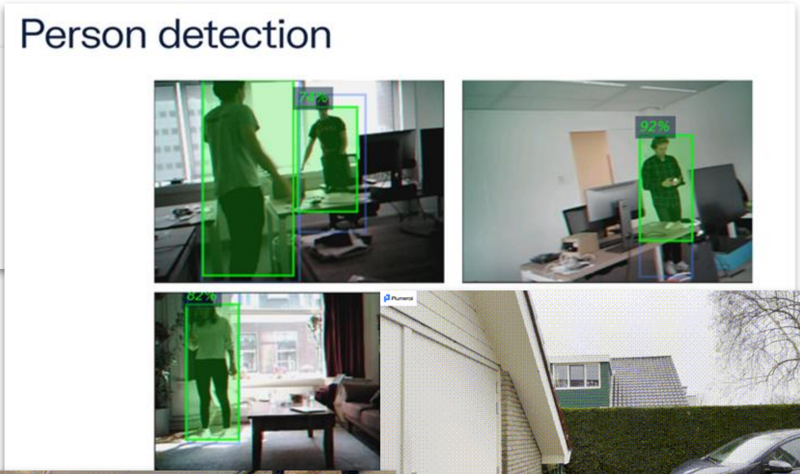
What can Plumerai mean for you?

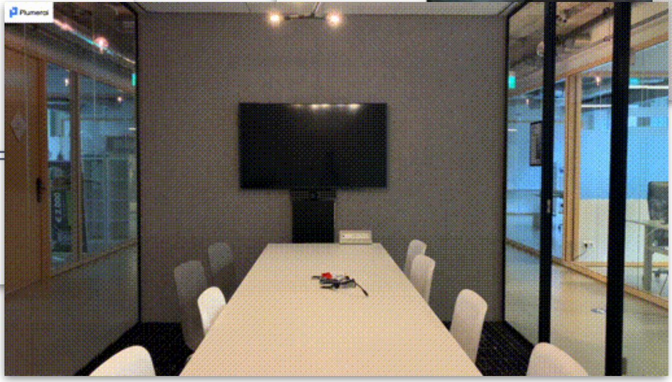

Monday, October 4, 2021

The world's fastest deep learning inference software for Arm Cortex-M

New: Try out our inference engine with your own model!


Binarized Convolution


Person detection



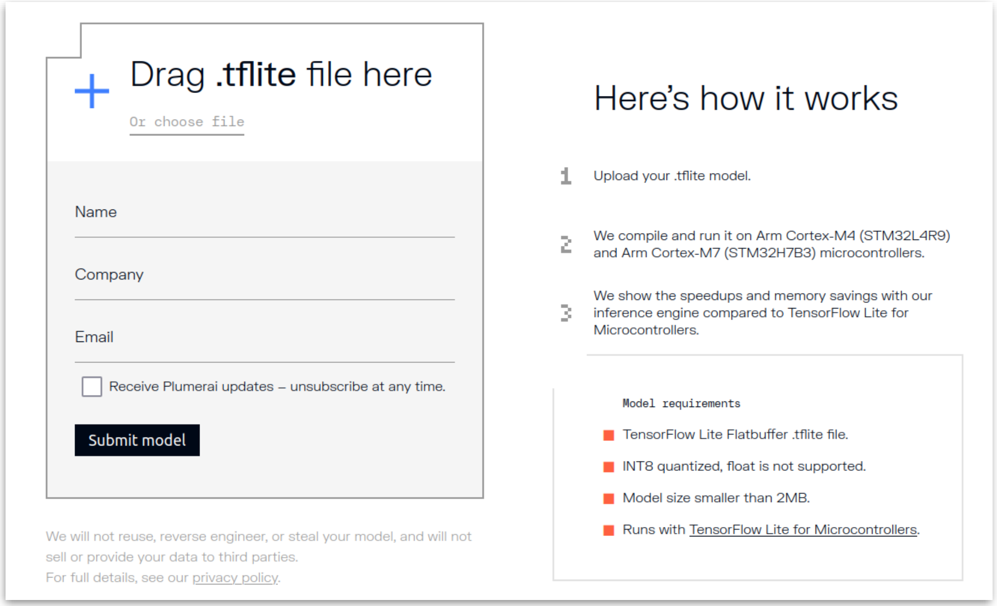
Plumerai Data Pipeline

Plumerai BNN Models

Plumerai Inference Stack

Plumerai Hardware

Public benchmarking service: try it yourself!



Drag .tflite file here

Or choose file

Name

Company

Email

☐ Receive Plumerai updates – unsubscribe at any time.

Submit model

We will not reuse, reverse engineer, or steal your model, and will not sell or provide your data to third parties.
For full details, see our privacy policy.

Here's how it works

1 Upload your .tflite model.

2 We compile and run it on Arm Cortex-M4 (STM32L4R9) and Arm Cortex-M7 (STM32H7B3) microcontrollers.

3 We show the speedups and memory savings with our inference engine compared to TensorFlow Lite for Microcontrollers.

Model requirements

- TensorFlow Lite Flatbuffer .tflite file.
- INT8 quantized, float is not supported.
- Model size smaller than 2MB.
- Runs with TensorFlow Lite for Microcontrollers.

Visit plumerai.com/benchmark to try it with your own model

Contact hello@plumerai.com for help or other questions

TINY ML TALKS webcast

Copyright Notice

www.tinyML.org