

The Benefits of TinyAI for CAM Control Systems

James Dickie

What we typically associate with AI



Chatbots (e.g., ChatGPT)



Computer vision in ADAS



10^6 - 10^{12}
Parameters

GB to TB
Memory

GFLOPS to TFLOPS
Performance

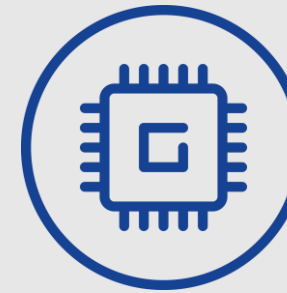
What if we could bring AI to real-time embedded systems?



Smart sensors



Smart actuators



1K-10K
Parameters

MB
Memory

MFLOPS
Performance

Tiny AI for Safety-Critical Embedded Systems

Example: Virtual Sensors

Use Case

AI-based Hands On/Off-Detection

Benefits of AI

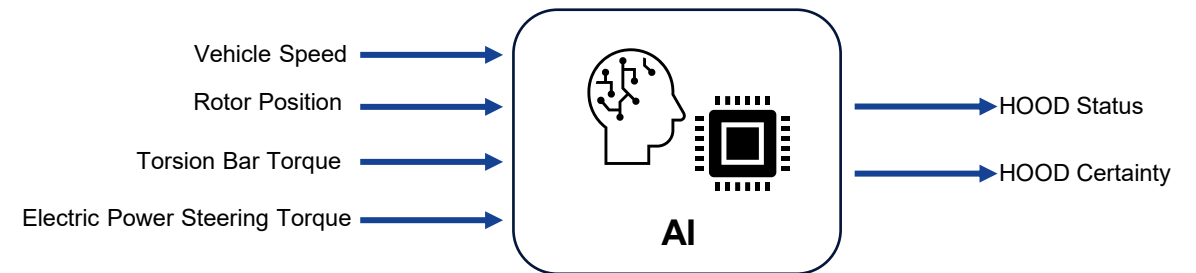


Cost savings by replacement of physical capacitive sensor on steering wheel



Reduced mechanical **complexity**

Future **updates** and improvements



Tiny AI for Safety-Critical Embedded Systems

Example: Enhanced functions

Use Case

AI-based object height classification with ultrasound sensors in parking systems

Benefits of AI



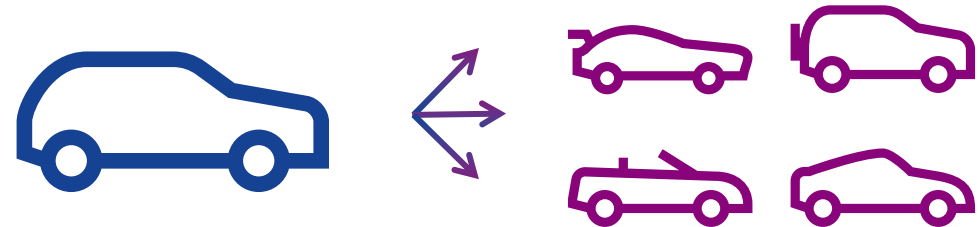
Robust detection of objects and **fast response**



Efficient adaption to many variants

AI functions developed for one product

Scaled to many similar products



Tiny AI for Safety-Critical Embedded Systems

Bringing AI to CAM Control Systems

Embedded deployment is one of the major pain points of making AI in products a reality

AI world



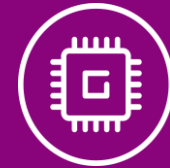
Data

AI model definition
& training

Trained neural
network



Embedded world



Embedded
C-code

Integration in
target hardware

Tiny AI for Safety-Critical Embedded Systems

Requirements for Embedded AI Deployment

Generate **C-code** from trained **neural networks** for **embedded systems**

AI without new costly hardware



Functional safety support



Easy to use



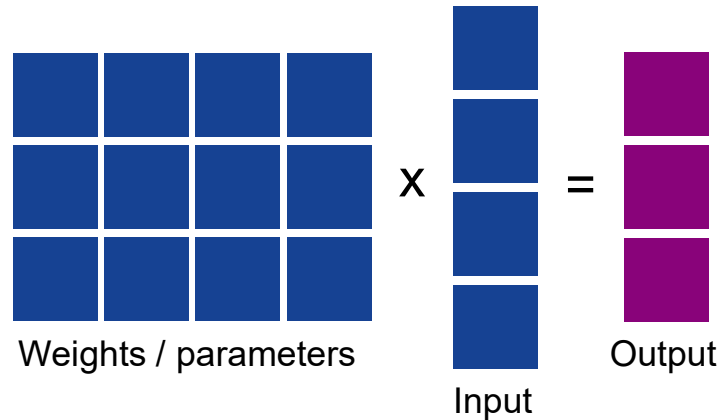
Low resource usage (runtime & memory)



Deep Dive: Resource Efficiency

Neural Networks Mathematics

Matrix-Vector Multiplication / Dot products

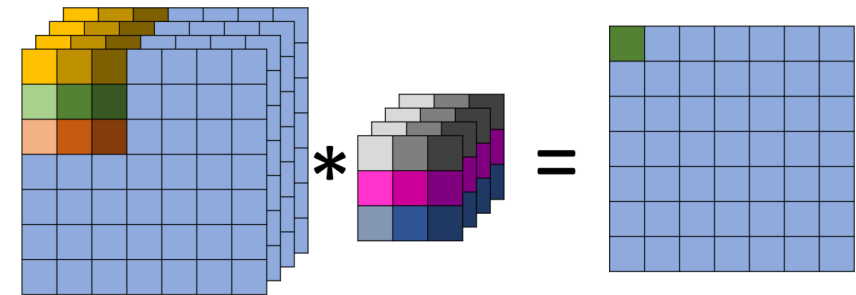


Requirements:

- Allocate memory for weights, inputs & outputs
- Compute dot products (multiply & add)

Variations of Neural Networks

- Convolutional neural networks (for image-like data)



- Recurrent neural networks (for time-series data)
- Many more

Important: The most important operation is the dot-product. Still, different networks need different handling.

In Embedded Systems, the targets are: Minimize runtime & memory consumption

Runtime Optimization with Code Generation vs. Library

Specific vs. Flexible Implementations:

- Specific implementation can be more efficient than flexible implementations

Example:

Clamping of values in quantized int8 networks

- Library implementation:

```
value = MAX(v_min, MIN(v_max, value));
```

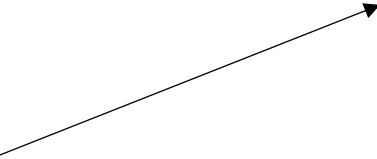
→ Works flexibly for any v_min, v_max

- Code generator implementation:

```
value = SSAT(value);
```

→ Clamps efficiently to full int8 range

Assembler results:



```
cmp    r0, r1
it     lt
movlt  r0, r1
cmp    r0, r2
it     ge
movge  r0, r2
```



```
ssat   r0, #8, r0
```

Runtime Optimization Using Interleaved Convolutions

Interleaved Convolutions:

- Loading data from memory is expensive
- Executing convolution patches one by one leads to data loading overheads

Optimization Approach:

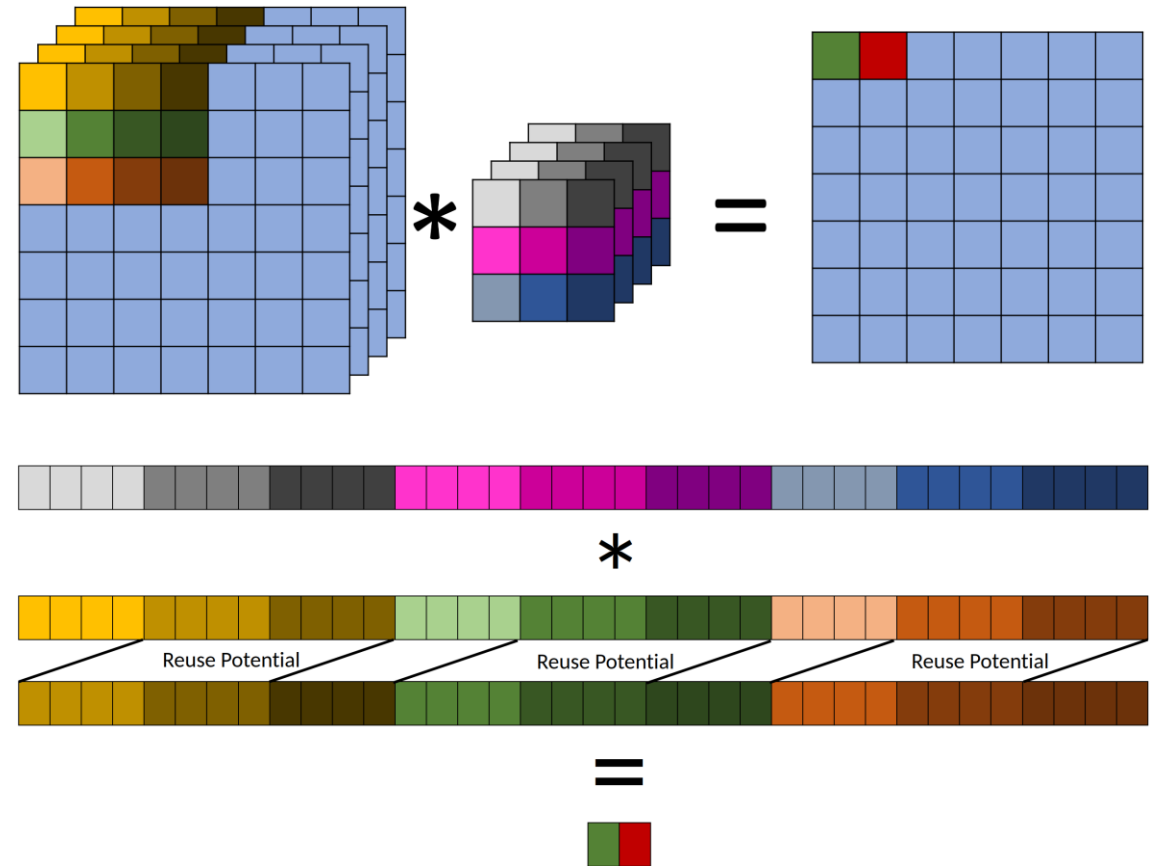
- Interleaved implementation that stores and loads each element only once

Impact:

- Reduced memory loads (~33%)
- Significant runtime improvement (~17% *), but mathematically equal

* ARM® Cortex®-M4, MLPerf™ Tiny image classification ResNet

Example: Data re-use when computing two pixels



Memory Optimization with Overlapping

Background:

- Neural networks on microcontrollers are computed serially
- Input data which has been processed fully is not needed anymore

Optimization:

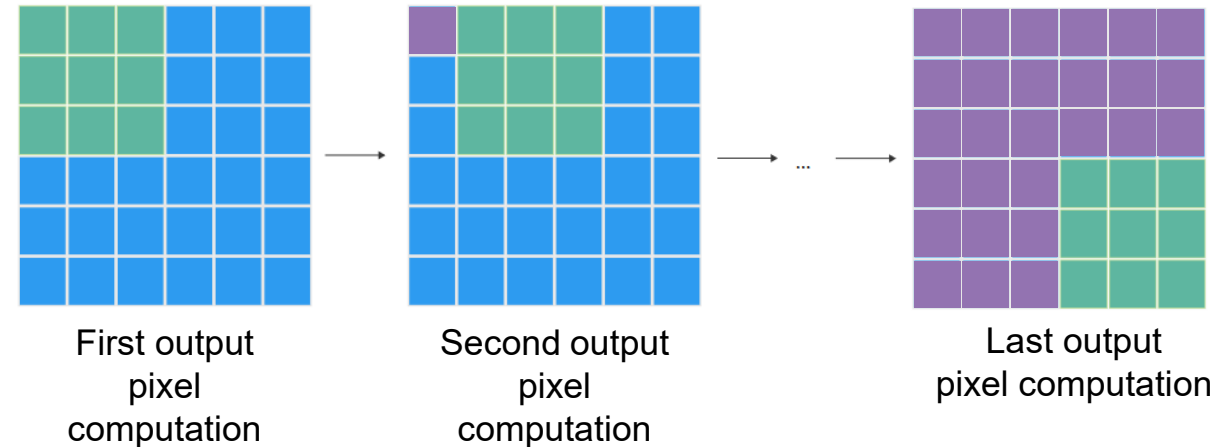
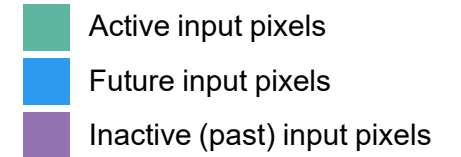
- Overwrite inputs that are not needed anymore with outputs

Impact:

- Working memory savings of up to 50%

Example:

- 2D Convolution
- 3x3 kernel, stride 1



Pareto-optimal Deployment

Background:

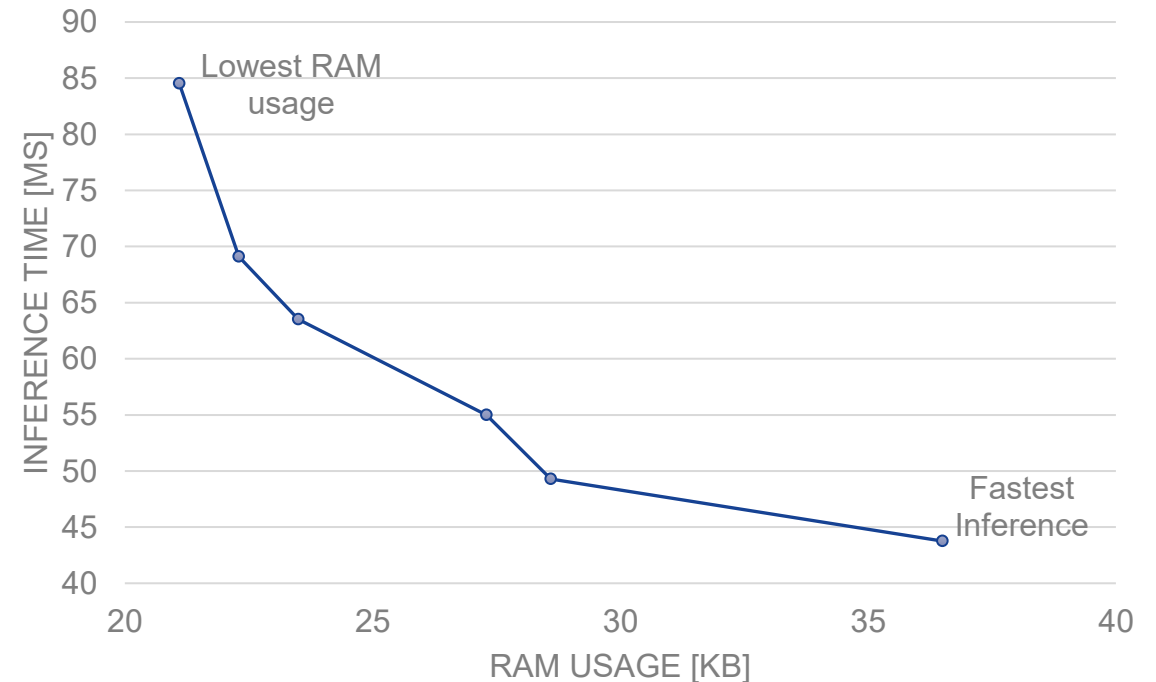
- Optimization methods yield trade-off between memory and runtime
- Optimization target is application-dependent
 - Some are runtime-critical (e.g. braking systems)
 - Some are memory-critical (e.g. condition monitoring)

Optimization:

- Generate code with various optimization approaches
- Choose the final implementation from the Pareto Front that shows different resource trade-offs (but **equal mathematical behavior**)

Example:

- MLPerf™ Tiny image classification ResNet
- Hardware: ARM® Cortex®-M7, 280 MHz

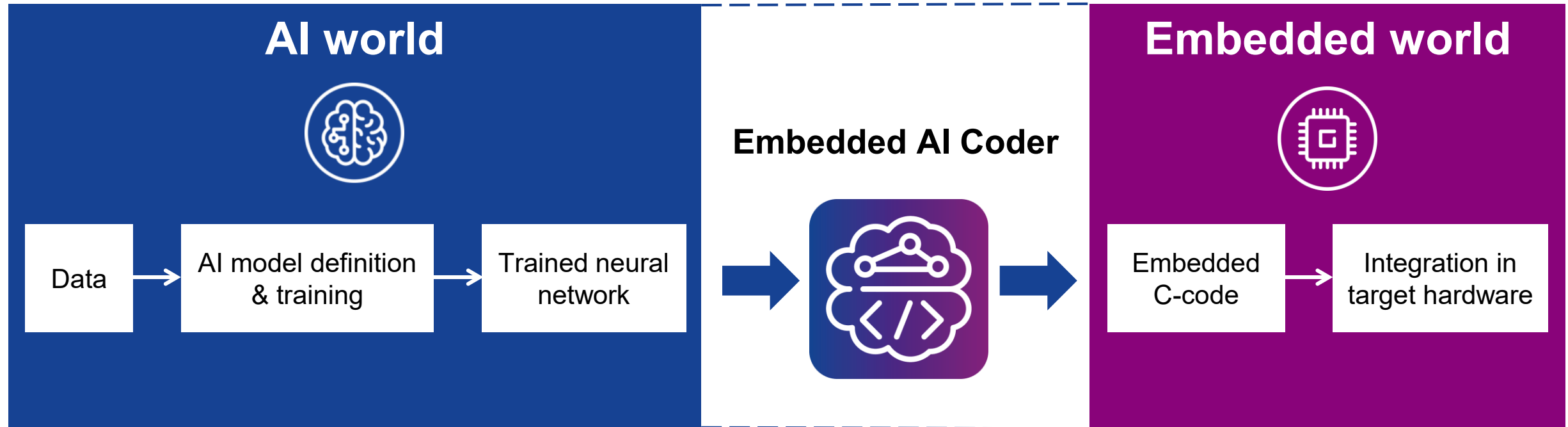


Summary

Tiny AI for Safety-Critical Embedded Systems

Bringing AI to CAM Control Systems

Embedded deployment is one of the major pain points of making AI in products a reality



www.etas.com/embeddedaicoder

Supporting deployment of embedded AI code in CAM



Safe code

- Numerically correct behaviour of generated code
- Memory safety
- MISRA compliance



Automatic verification

- Integrated test suite for automated testing of generated code
- Provisioning of test report and safety manual for qualification



Reparameterisation

- Build once and calibrate easily for multiple variants
- Supports standard automotive formats like DCM

Thank you

james.dickie@etas.com