# Embedded AI

## Guide to AI and Data Courses · Part VI: Systems and Applications

Resource limits | Quantization | Pruning and distillation | Efficient architectures | Devices and accelerators | Neighbouring courses

Sercan KÜLCÜ · Giresun University · Computer Engineering

© Sercan Külcü. All rights reserved. · Interactive HTML version of these slides: https://sercankulcu.github.io/files/ai_courses/slides/Chapter_36_Embedded_AI.html

INTRODUCTION

# What Question Does It Ask?

> **How do AI models run on small, low-power devices?**

Embedded AI studies running trained models on devices with limited memory, compute and energy, such as phones, cameras or microcontrollers. The course covers shrinking a model by **quantization**, pruning and distillation, moving it onto the device, and measuring latency and energy; the extreme form is called TinyML.

## At a glance

| | | |
|---|---|---|
| **APPROACH**<br>resource-constrained inference | **INPUT**<br>trained model, device constraints | **OUTPUT**<br>small, fast model |
| **SUCCESS MEASURE**<br>latency, energy, memory | **MATH LOAD**<br>medium | **DATA NEEDS**<br>training data already exists |

# A Short History

**2015**
Hinton, Vinyals and Dean introduce knowledge distillation: a large model's knowledge passes to a small one.

**2016**
Han et al. combine pruning and quantization in Deep Compression, shrinking networks by tens of times.

**2017**
MobileNets is published and TensorFlow Lite announced: deep networks on phones become routine.

**2018**
Google announces the Edge TPU accelerator for edge devices.

**2019**
The first tinyML Summit is held; Warden and Situnayake publish the book TinyML.

**2020**
MCUNet runs ImageNet classification on a microcontroller with over 70% accuracy.

THE COURSE

# Core Concepts

**Quantization**
8-bit or smaller integers instead of 32-bit floats.

**Pruning**
Removing unimportant weights, channels or layers.

**Knowledge distillation**
A small student model mimics a large teacher.

**Efficient architectures**
MobileNet, depthwise separable convolution.

**Inference engine**
The thin software layer that runs the model on the device.

**Accelerators**
NPU, DSP, Edge TPU: hardware built for neural networks.

**Memory budget**
A few hundred KB of SRAM and flash for weights and activations.

**Latency and energy**
Two performance metrics as important as accuracy.

# A Typical Workflow




**Example:** A small keyword-spotting network is trained on a computer with 32-bit floats, quantized to 8-bit integers and loaded onto a microcontroller with a few hundred kilobytes of memory. The device listens all the time, never sends audio to the cloud and uses very little power.

THE COURSE

# What Does the Course Cover?

1. Embedded hardware: MCUs, memory and energy budgets
2. Sensor data collection and preprocessing
3. Small neural networks and efficient architectures
4. Post-training quantization
5. Quantization-aware training
6. Pruning and sparse networks
7. Knowledge distillation
8. Model conversion: TensorFlow Lite and ONNX
9. Inference on microcontrollers: TFLite Micro, CMSIS-NN
10. Accelerators: NPUs, DSPs and the Edge TPU
11. Measuring latency, memory, energy; MLPerf Tiny
12. On-device learning and over-the-air model updates

## Skills you will gain

converting models with TensorFlow Lite | applying 8-bit quantization | deploying a model to Arduino | an end-to-end project in Edge Impulse | measuring latency, memory and energy

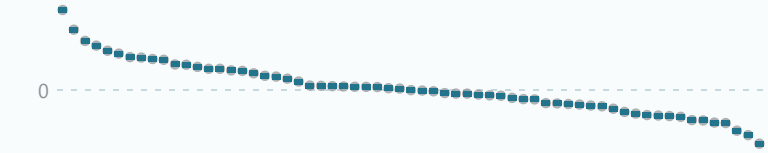
A typical semester; the order and coverage vary between universities.

TRY IT

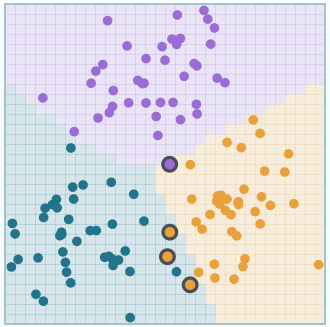
# Fit the Model on a Microcontroller

A vibration sensor on a motor must recognize faults by itself, but its processor has 512 KB of memory. Choose how many bits to store each weight in; size, speed and accuracy change together.

Weights (largest first): grey exact, blue 32-bit




Test data: accuracy 95.0%



Memory: **1172 KB**

device limit 512 KB

Latency: **120 ms**

Energy: **4.8 mJ**

latency and energy are estimates

Bits per weight: **32**

**32** | 16 | 8 | 4 | 2

32 bit: model **1172 KB** → does not fit
Accuracy **95.0%** (95.0% at 32 bits)
Estimated latency 120 ms, energy 4.8 mJ

● normal ● imbalance ● bearing fault. The network is trained here (63 weights); size is scaled to a real 300k-weight model.

Embedded AI treats shrinking the model as part of the job: 8-bit **quantization** usually keeps accuracy and cuts memory to a quarter. At 2 bits the decision boundaries fall apart.

CONTEXT

# Prerequisites

Courses worth knowing before this one, and why they are needed.

| Course | Why it is needed |
| --- | --- |
| Deep Learning | quantization, pruning, small models |
| Microprocessors | memory, registers, interrupts and peripherals: understanding device limits |
| Programming with C | the language of embedded software; pointers and static memory management |

CONTEXT

# Are You Ready for This Course?

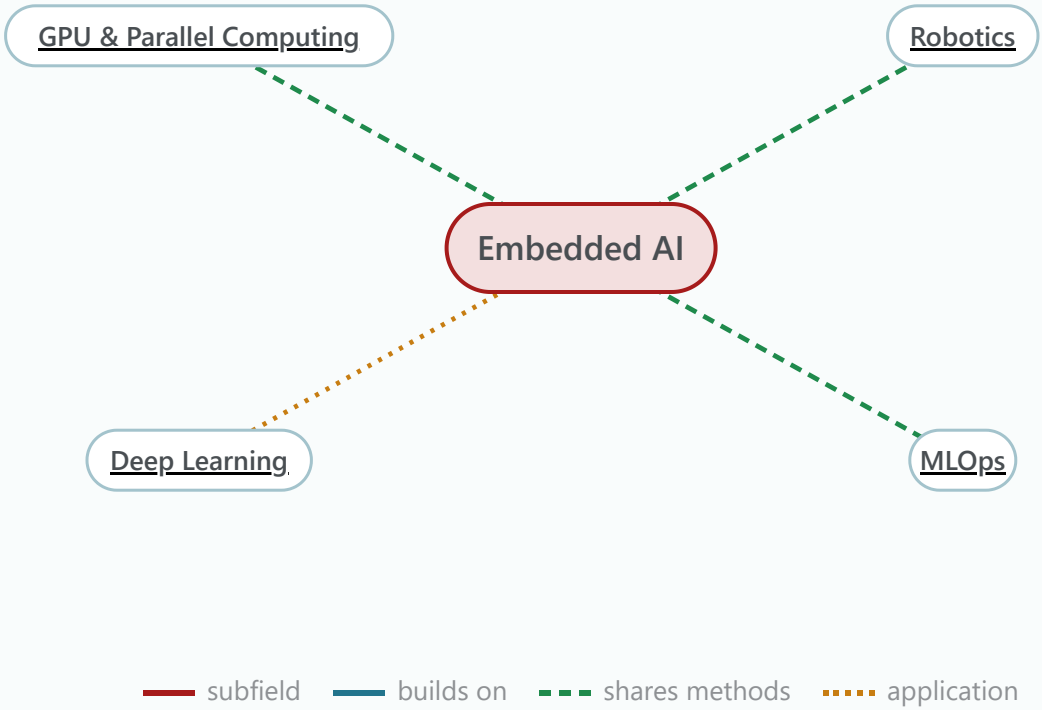
Ask yourself; for every "no", the right-hand column shows where to start.

| Ask yourself | If not, start with |
|---|---|
| Can you train a convolutional network and report its validation accuracy? | Deep Learning |
| Can you write C code with pointers and without dynamic memory allocation? | A C programming course |
| Can you flash a program to a microcontroller and read data from a sensor? | A microprocessors course and an Arduino starter kit |
| Can you explain how a number is represented as an 8-bit integer? | Binary numbers and fixed point from digital design |

If you can say "yes" to most of these, you are ready; one or two gaps can be closed during the semester.

CONTEXT

# Neighbouring Courses



OVERLAP

# Overlapping Courses

| Relation | Course | Shared ground |
|---|---|---|
| application | This course applies the methods of Deep Learning | quantization, pruning, small models |
| shares methods | Shares methods with Robotics and Robot Vision | real-time on-device inference |
| shares methods | Shares methods with Machine Learning Systems and MLOps | deploying and updating models on devices |
| shares methods | Shares methods with GPU Programming and Parallel Computing | hardware accelerators |

DIFFERENCES

# Where It Differs (1/2)

| Criterion | Embedded AI | Deep Learning | GPU & Parallel Computing |
|---|---|---|---|
| **Core question** | How do AI models run on small, low-power devices? | How do many-layered networks learn useful representations from raw data? | How do thousands of cores share the same computation? |
| **Approach** | resource-constrained inference | multilayer representation learning | data-parallel computation |
| **Input** | trained model, device constraints | raw data: images, text, audio | parallelizable algorithm |
| **Output** | small, fast model | learned representation and prediction | accelerated program |
| **Success measure** | latency, energy, memory | accuracy, loss, scaling | speedup, memory bandwidth |
| **Math load** | medium | high: derivatives, matrices, probability | low-medium |
| **Data needs** | training data already exists | large; needs GPUs | none |

- **Deep Learning:** Deep learning grows the model for accuracy; this course shrinks the same model while losing little accuracy.
- **GPU & Parallel Computing:** GPU programming gets peak performance from large accelerators; this course runs inference on devices drawing a few milliwatts.

DIFFERENCES

# Where It Differs (2/2)

| Criterion | Embedded AI | MLOps | Robotics |
|---|---|---|---|
| **Core question** | How do AI models run on small, low-power devices? | How is a model taken from the lab into production and kept running there? | How does a machine sense the physical world, know where it is and move? |
| **Approach** | resource-constrained inference | software engineering meets ML | sense-plan-act loop |
| **Input** | trained model, device constraints | trained model, data pipeline | sensor measurements |
| **Output** | small, fast model | a running, monitored service | motor commands, path |
| **Success measure** | latency, energy, memory | latency, uptime, data drift | task success, safety |
| **Math load** | medium | low | high: transformations, control |
| **Data needs** | training data already exists | streaming production data | physical experiments and simulation |

- **MLOps:** MLOps deploys and monitors models on servers; this course embeds the model in devices that may have no connection.
- **Robotics:** Robotics treats perception, localization and motion as a whole; this course makes the robot's model run in real time.

PRACTICE

# Applications

**Wake words**
An always-listening model in speakers and phones.

**Smart cameras**
Object detection without sending images to the cloud.

**Wearables**
Fall, step and heart rhythm analysis on a watch.

**Predictive maintenance**
Catching fault signs in motor vibration.

**On-phone AI**
Photo enhancement, offline translation.

**Farming and environment**
Sound and pest detection on battery sensors.

PRACTICE

# Risks and Ethics

**The hidden cost of shrinking**

Quantization and pruning barely move average accuracy, yet rare classes and minority groups can lose far more.

**Devices that cannot be updated**

A faulty or vulnerable model spread over thousands of devices in the field is hard to recall.

**Unnoticed surveillance**

Always-listening microphones and cameras can monitor bystanders unawares, even when data stays on the device.

**Model theft**

The model file on a device can be read and copied, and reverse engineering opens it to adversarial attacks.

In depth: Explainable AI and AI Ethics.

PRACTICE

# Tools and Datasets

## Tools and libraries

TensorFlow Lite | ONNX Runtime | ExecuTorch | Edge Impulse | CMSIS-NN | Arduino | NVIDIA Jetson

## Well-known datasets

Speech Commands | Visual Wake Words | MNIST | CIFAR-10 | ImageNet | MLPerf Tiny

Success is not measured by accuracy alone: model size, peak memory, latency and energy go in the same table.

PRACTICE

# Common Misconceptions

| Common belief | In fact |
| --- | --- |
| Models are trained on the small device. | Training is mostly done on a server; only inference runs on the device. |
| Quantization always ruins accuracy. | 8-bit quantization costs most models very little accuracy; quantization-aware training reduces the loss further. |
| Fewer parameters means a faster model. | Speed depends on memory access, operation types and hardware support; parameter count is not enough. |
| Embedded AI is only needed without internet. | For privacy, latency and energy, on-device processing is preferred even when a connection exists. |

NEXT STEPS

# After This Course

## Courses that build on or extend this one

This course is mostly an end point: it builds on the others.

## Related careers

Embedded software engineer | Edge AI engineer | Machine learning engineer | Hardware-software co-designer | IoT solutions architect

Embedded AI connects deep learning to hardware: from here you move on to robotics, autonomous vehicles and Internet of Things applications. For more depth in hardware, accelerator design and compilers come next; on the software side, MLOps and updating fleets of devices.

# Further Reading


- Warden and Situnayake, *TinyML*, O'Reilly, 2019.
- Sze, Chen, Yang and Emer, *Efficient Processing of Deep Neural Networks*, Morgan & Claypool, 2020.
- Han, Mao and Dally, "Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding", *ICLR*, 2016.
- Howard et al., "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications", *arXiv:1704.04861*, 2017.
- Hinton, Vinyals and Dean, "Distilling the Knowledge in a Neural Network", *arXiv:1503.02531*, 2015.

# The End

## Chapter 36: Embedded AI

← Previous chapter · Map of the field · Next chapter →