Optimizing AI Models for Embedded Systems

Surendra Panpaliya

Agenda

- AI Model Optimization Techniques
- Techniques for Quantization,
- Pruning, and Compression
- Trade-offs between model size,
- accuracy, and performance
- in embedded systems

Agenda

- Hardware Accelerators
- Using GPUs, TPUs, and
- other accelerators in embedded systems
- Selecting appropriate hardware
- for different AI tasks

Agenda

- Power Management and Efficiency
- Techniques for reducing power consumption in
- AI applications on embedded devices
- Balancing performance and
- power efficiency for longer battery life

Agenda

- Edge AI Frameworks and Tools
- Overview of Edge Impulse,
- TensorFlow Lite, Other frameworks for edge AI
- Comparing different frameworks
- for embedded AI development

Agenda

Hands-on Lab

Optimizing and Deploying AI Models

on Embedded Systems

AI Model Optimization Techniques

- Techniques for Quantization,
- Pruning, and Compression
- Trade-offs between model size,
- Accuracy, and performance in embedded systems

AI Model Optimization Techniques

REFER TO METHODS

USED TO IMPROVE MODEL PERFORMANCE,

EFFICIENCY, AND ROBUSTNESS

DURING TRAINING AND INFERENCE.

Model Optimization Techniques

ARE ESSENTIAL FOR DEALING

WITH ISSUES SUCH AS OVERFITTING,

UNDERFITTING, SLOW CONVERGENCE, AND

INEFFICIENT USE OF COMPUTATIONAL RESOURCES.

Overfitting

Occurs when a model learns

the details and noise in the training data

to such an extent that

it negatively impacts

its performance on new data.

Overfitting

The model becomes too complex and

overly sensitive to the training data,

capturing both the signal (relevant patterns) and

the noise (irrelevant details).

Underfitting

Underfitting occurs

when a model is too simple

to capture the underlying

patterns in the data.

Underfitting

Can happen when

the model has too few parameters or

when it's not given enough training time.

Underfitting

As a result,

the model performs poorly

both on the training data and

the test data.

Slow Convergence

Training of a

machine learning model

progresses very slowly

Slow Convergence

Takes a long time for the model

to reach an optimal solution or

a satisfactory level of accuracy.

Quantization, Pruning, and Compression

Quantization, Pruning, and Compression

Essential strategies

for optimizing AI models,

particularly when deploying them

in resource-constrained environments

Quantization, Pruning, and Compression

Reduce the model's size,

Computational requirements

Energy consumption,

while maintaining accuracy.

Quantization

Quantization

Reduces the precision

of the numbers representing

model parameters (such as weights and activations)

from 32-bit floating-point numbers

to lower-bit representations

(like 16-bit, 8-bit, or even lower).

Quantization

Leads to

smaller model sizes

faster inference times.

Post-Training Quantization

- Convert
- the model
- after it has been
- fully trained.

Post-Training Quantization

Most common approach

because it doesn't require changes

to the training process.

Dynamic Range Quantization

Only weights are quantized,

typically from

32-bit floating-point

to 8-bit integers.

Full Integer Quantization

Both

weights and activations

are quantized

to 8-bit integers.

Float16 Quantization

Weights are stored in

float16 format (half precision),

which reduces the size

without impacting performance.

Quantization

```
# Example: TensorFlow Lite post-training quantization
import tensorflow as tf
converter = tf.lite.TFLiteConverter.from_saved_model('model')
converter.optimizations = [tf.lite.Optimize.DEFAULT]
tflite_model = converter.convert()
```

Pruning

Removing → Unnecessary neurons or → Parameters from a model

Pruning

Reducing its size

computation complexity

without impacting

its performance.

Pruning

Not all parameters

contribute equally

to the model’s predictions,

so redundant or less impactful ones

can be removed.

Magnitude-based Pruning

Remove weights

that are smaller than

a predefined threshold.

Magnitude-based Pruning

import tensorflow_model_optimization as tfmot

prune_low_magnitude =
tfmot.sparsity.keras.prune_low_magnitude

model = prune_low_magnitude(model)

Structured Pruning

Prune entire groups of neurons

Ex. filters in convolutional networks

rather than individual weights.

Unstructured Pruning

Remove individual weights

in the model based

on their magnitude.

Compression

AIM TO REDUCE THE OVERALL SIZE OF THE MODEL,

MAKING IT EASIER TO STORE AND DEPLOY

WITHOUT LOSING ACCURACY.

Tools for Quantization, Pruning, and Compression

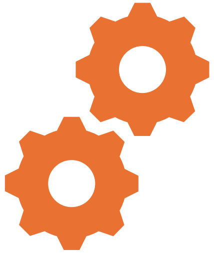
TensorFlow Lite

TensorFlow Model Optimization Toolkit

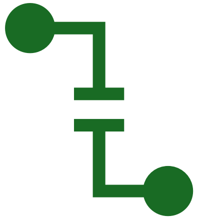
OpenVINO

- A toolkit for optimizing models,
- including quantization and compression
- for deployment on Intel hardware.

NVIDIA TensorRT



Supports model optimization,



including quantization and layer fusion,

specifically designed for NVIDIA GPUs.

Trade-offs between model size, Accuracy, and performance

- When deploying AI models in embedded systems
- there are critical trade-offs
- to consider between
- model size, accuracy, and performance.

Model Size vs. Accuracy

Smaller models tend to have

fewer parameters and

lower computational complexity.

Model Size vs. Accuracy

Pruning and quantization

can be used to reduce model size

but may result in a slight decrease

in model accuracy

Trade-off Example

REDUCING A MODEL'S SIZE BY 90%

(USING TECHNIQUES LIKE PRUNING)

CAN DECREASE ACCURACY BY 1-5%,

DEPENDING ON THE TASK.

Trade-off Example

Quantized models (8-bit)

May sacrifice a small percentage of Accuracy

compared to full precision (32-bit) models

Consideration

For lightweight applications

like voice recognition on a smartphone

slight drops in accuracy

may be acceptable.

Model Size vs. Performance

Smaller models generally perform

better in terms of

latency (response time)

throughput (number of inferences per second)

Model Size vs. Performance

Quantization can

reduce the model size

improve performance

Model Size vs. Performance

Pruned models

have fewer weights

to process can

perform better

in terms of latency,

Trade-off Example

An 8-bit quantized model

can run 3-4x faster than

its 32-bit floating-point counterpart,

but might experience a 1-3% drop in accuracy.

Consideration

Latency-sensitive applications

real-time video processing or

autonomous driving

benefit more from

smaller, faster models

Optimizing AI Models Hardware Accelerators

Optimizing AI Models Hardware Accelerators

GPUs, TPUs, and specialized hardware

crucial for achieving high performance,

low power consumption, and

efficient model execution in embedded systems.

Optimizing AI Models Hardware Accelerators

Each accelerator type

comes with

its unique advantages and

trade-offs

Optimizing AI Models Hardware Accelerators

Selecting hardware

for different AI tasks involves

considering factors such as

computational requirements,

Memory capacity, power efficiency, and cost.

General-Purpose vs. Specialized Hardware Accelerators

General-Purpose Hardware (CPUs)

Use Case:

General-purpose tasks,

lightweight AI models,

low-throughput applications.

Advantages

Highly versatile and

can handle a wide range of tasks,

including AI inference and

general computations.

Advantages

Suitable for applications

where AI is just one of many tasks

running on the system.

Limitations

Lower parallelism compared

to specialized hardware (like GPUs or TPUs),

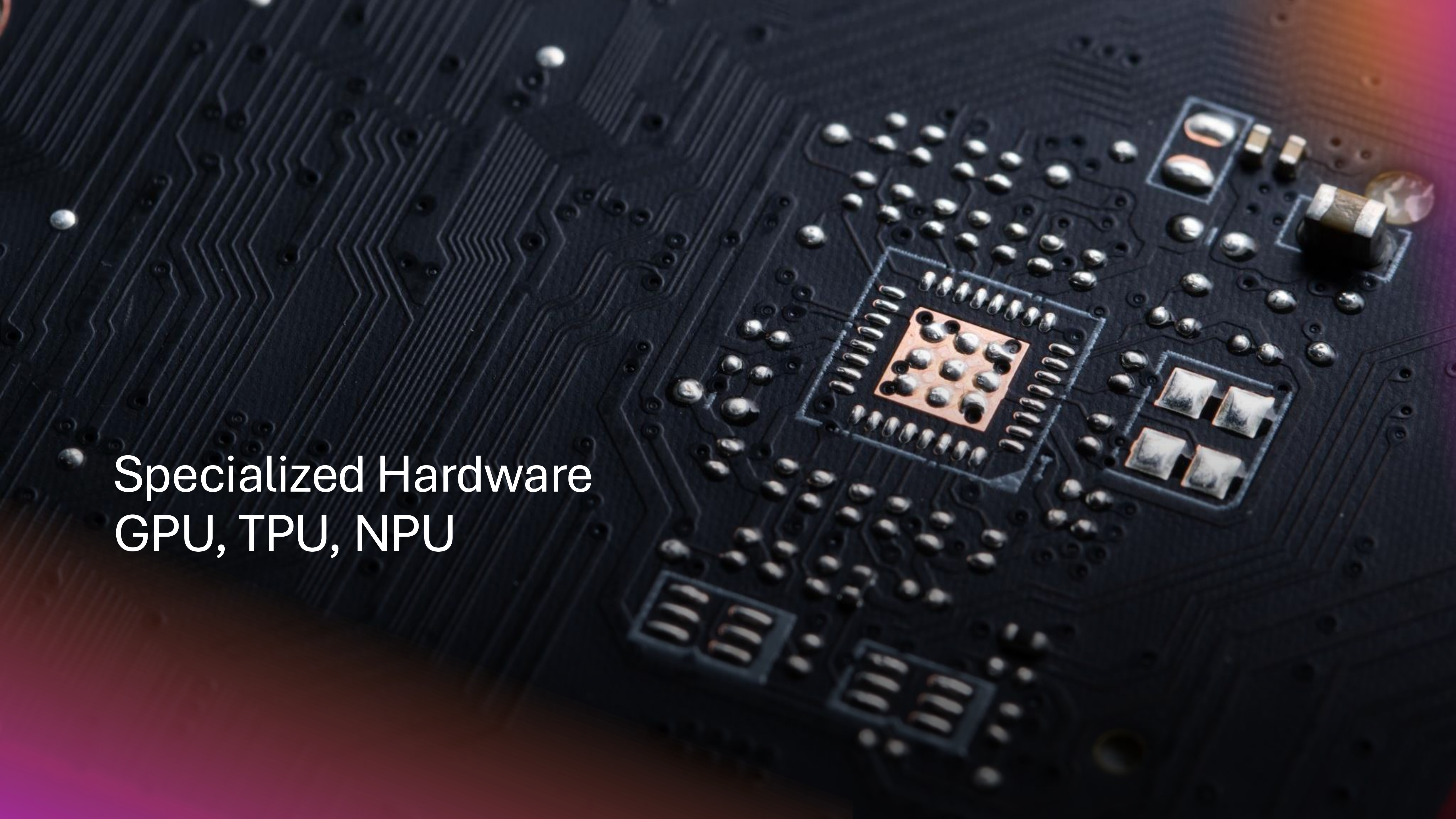
leading to slower inference speeds

for deep learning models.

Limitations

Consumes more power for AI tasks

compared to specialized accelerators.



Specialized Hardware
GPU, TPU, NPU

Specialized Hardware

Optimized for

High throughput,

Parallel computation

Efficient execution of AI models.

GPUs (Graphics Processing Units)

GPUs are designed for

parallel processing

well-suited for training and

inferencing deep neural networks.

GPUs (Graphics Processing Units)

Use Cases:

Training large models

CNNs, RNNs

Inference for high-performance applications

Real-time image classification,

Video processing

Advantages

High parallelism:

Thousands of cores handle

large-scale matrix and

tensor operations efficiently.

Advantages

Flexibility:

Can be used for

both training and

inference tasks.

Advantages

Mature software ecosystem:

Popular deep learning frameworks like

TensorFlow, PyTorch

are highly optimized for GPUs.

Limitations

HIGH POWER CONSUMPTION:

GPUS CONSUME SIGNIFICANTLY MORE POWER

LESS IDEAL FOR LOW-POWER

EMBEDDED SYSTEMS.

Best Fit

Embedded devices with

substantial power budgets

Autonomous vehicles

Drones

Best Fit

- Applications requiring
- real-time processing of
- high-dimensional data
- Video analytics,
- Autonomous driving

Examples of GPU Accelerators

NVIDIA Jetson:

A popular embedded AI platform for GPUs,

ideal for AI inference in robotics,

drones, and IoT devices.

(Jetson Nano, Jetson Xavier NX).

Examples of GPU Accelerators

Raspberry Pi 4 with Google Coral USB Accelerator:

Pairing a general-purpose device like

Raspberry Pi with an external accelerator

for GPU-level performance.

TPUs (Tensor Processing Units)

TPUS ARE AI ACCELERATORS

DESIGNED BY GOOGLE,

OPTIMIZED FOR TENSORFLOW.

TPUs (Tensor Processing Units)

Highly efficient at

running deep learning models,

particularly for inference.

Use Cases

Inference in large-scale applications:

TPUs excel at deploying

AI models at scale

Use Cases

High-performance neural network tasks:

Models requiring tensor operations

CNNs, RNNs

work well with TPUs.

Advantages

Highly efficient for matrix/tensor operations:

TPUs are optimized for

matrix multiplications,

which are the backbone of

most neural networks.

Advantages

Low power consumption compared to GPUs:

TPUs can deliver faster inference

with lower power requirements.

Advantages

Optimized for inference tasks:

TPUs are tailored more toward

high-speed inference

rather than training.

Limitations

Less flexible than GPUs:

TPUs are designed specifically for

tensor-based computations,

so they may not be as versatile

for non-tensor computations.

Limitations

Framework dependency:

Best performance is achieved with TensorFlow,

limiting flexibility for other frameworks.

Best Fit:

Edge AI: → TPUs are ideal for → deploying AI at the edge, → especially in environments → that require high-performance, → low-power inference.

Best Fit

- Large-scale inference applications:
- Suitable for tasks like real-time
- Natural Language Processing (NLP) or
- Computer Vision.

Examples of TPU Accelerators

Google Coral Edge TPU

An efficient AI inference device

for edge computing that delivers

high performance at low power consumption.

Examples of TPU Accelerators

Google Cloud TPUs:

Used for large-scale model training and

inference tasks in cloud environments.

NPUs (Neural Processing Units)

NPUS ARE SPECIALIZED HARDWARE UNITS

DESIGNED FOR ACCELERATING

NEURAL NETWORK COMPUTATIONS.

NPUs (Neural Processing Units)

NPUs are highly optimized for

low-power embedded systems and

are commonly found in

smartphones and edge devices.

Use Cases

AI ACCELERATION IN MOBILE DEVICES:

NPUS ARE COMMONLY USED IN

SMARTPHONES,

TABLETS, AND SMART CAMERAS

Use Cases

Real-time edge AI:

NPUs provide low-latency inference

for AI tasks at the edge,

without relying on cloud resources.

Advantages

Energy-efficient:

NPUs are designed to consume

very little power while delivering

high-performance AI inference.

Advantages

Optimized for edge AI:

Perfect for low-power,

high-performance inference tasks

Advantages

ON-DEVICE INFERENCE:

CAN RUN MODELS DIRECTLY ON THE DEVICE

WITHOUT REQUIRING CLOUD CONNECTIVITY,

LEADING TO LOWER LATENCY.

Limitations

- Limited flexibility:
- NPUs are specialized and
- may not handle
- non-AI tasks efficiently.

Limitations

Smaller model capacity:

May struggle with large models

compared to more powerful

GPUs or TPUs.

Best Fit

Smartphones and mobile devices:

NPUs are typically integrated into

mobile SoCs for AI tasks like

face unlocking, camera enhancements, and

speech recognition.

Best Fit

Battery-powered IoT devices:

NPUs are ideal for low-power

AI inference in applications like

smart home devices or

wearable tech.

Examples of NPU Accelerators

Apple A-series and M-series NPUs

Built into iPhones and iPads for AI tasks.

Examples of NPU Accelerators

Huawei Kirin NPU:

Optimized for mobile AI tasks

like real-time image and video processing.

Surendra Panpaliya

Founder and CEO

GKTCS Innovations

https://www.gktcs.com