TinyML Compression

Portability **Trade-offs**

Sacrifice **portability** across systems for **efficiency**.

ⓧ	✓

Specific HW Implementation of a Library

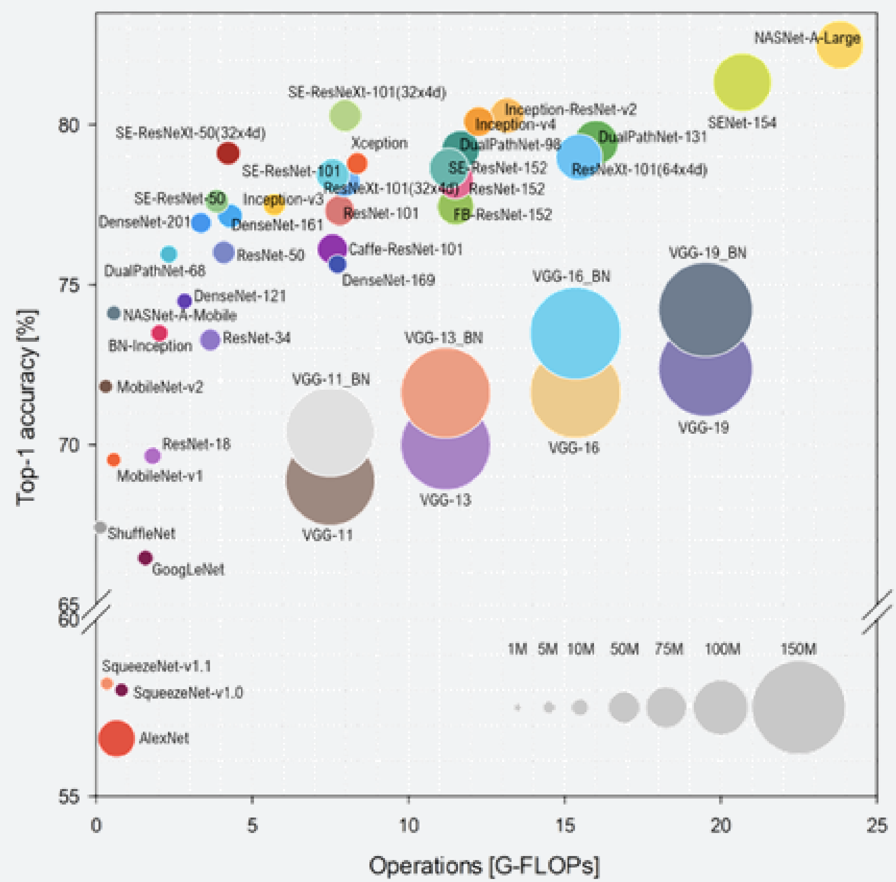
Option 1

- Universal Code Portability/Compatibility ✓
- Cost ($) ⓧ
- Power Consumption (W) ⓧ
- Engineering Effort ⓧ

Option 2

- Lower Code Portability ⓧ
- Cost ($) ✓
- Power (W) ✓
- Eng. Effort ✓

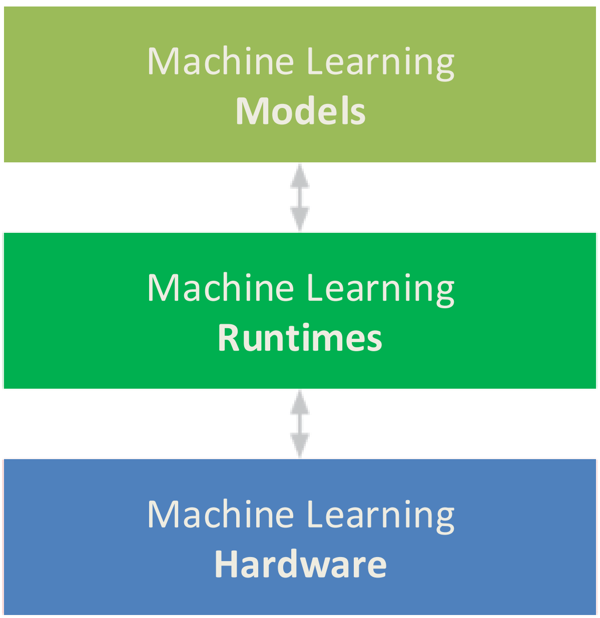
ML Model Evolution



Source: S. Bianco, R. Cadene, L. Celona, and P. Napoletano, "Benchmark analysis of representative deep neural network architectures," *IEEE Access*, vol. 6, pp. 64 270–64 277, 2018

Model Comparisons

Model	Version / Variant	Top-1 Accuracy (ImageNet)	Approx. Model Size / Parameters	Comments
AlexNet	AlexNet (2012)	~ 63 % top-1 reported historically. (AI Summer)	~ 60 M parameters, ~ hundreds of MB model size. (Wikipedia)	One of the first deep CNNs, now mostly a teaching tool rather than state-of-the-art.
VGGNet	VGG16 / VGG19	VGG16: ~ 71.3 % top-1 in Keras Applications table. (keras.io)	~ 138 M parameters (VGG16); ~ 528 MB size in Keras table. (keras.io)	Simple, uniform architecture; large size & heavy compute.
ResNet	ResNet152V2 (as a recent version)	ResNet152V2: ~ 78.0 % top-1 (Keras table) (keras.io)	~ 60.4 M parameters (ResNet152/152V2) (keras.io)	Deep residual networks; very good accuracy vs older nets with more efficient size than VGG.
MobileNet	MobileNetV3 (Large)	~ 75.2 % top-1 for MobileNetV3-Large in original paper. (arXiv)	~ 5.48 M parameters for V3-Large (source) (GitHub)	Designed for mobile / edge devices: very efficient size and latency trade-off.



Model Compression Techniques

Pruning

Quantization

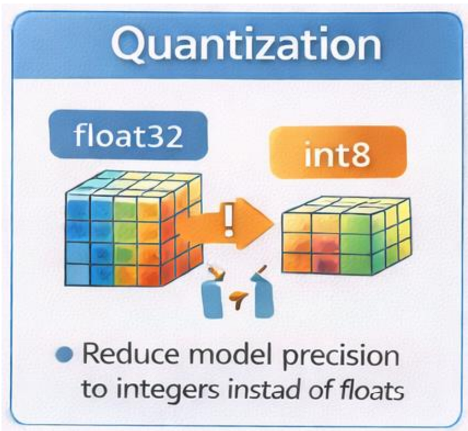
Knowledge

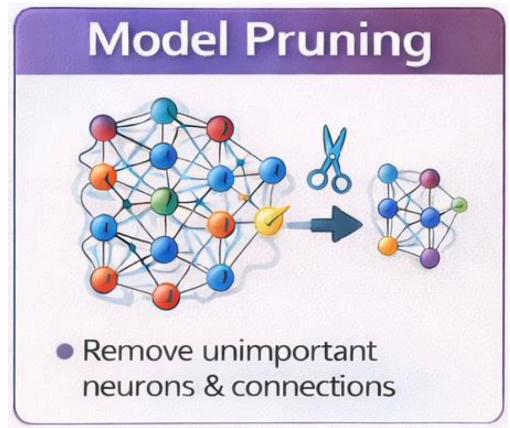
Distillation

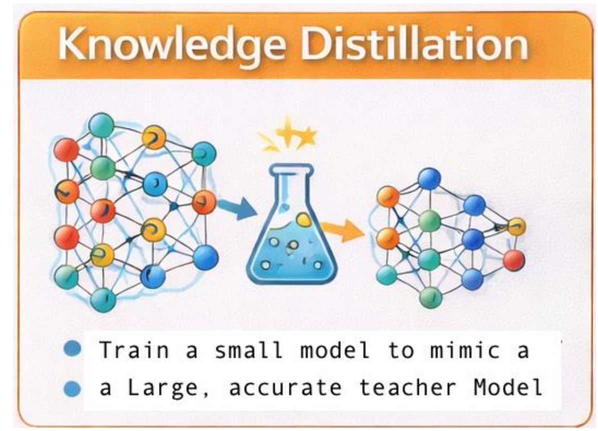
...

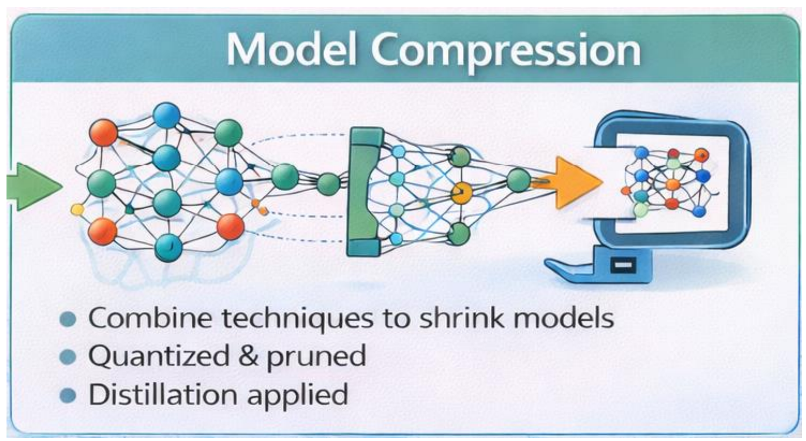
Optimization in Tiny Devices

Embedded ML





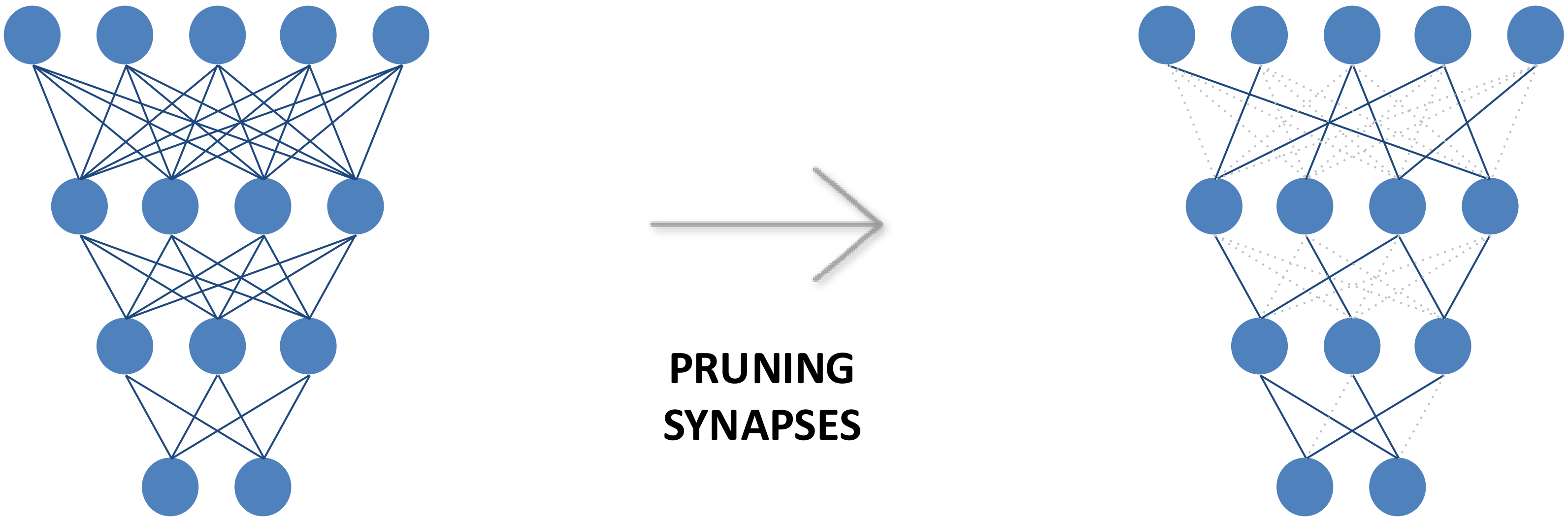




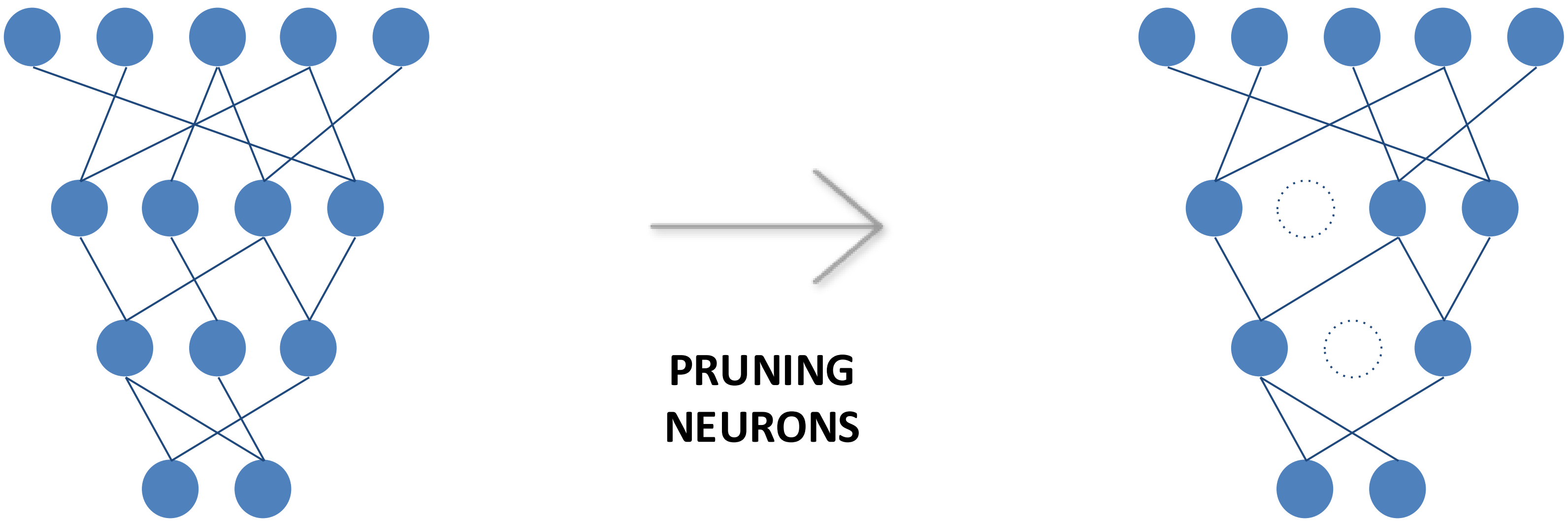
“Smaller models run faster, use less memory, and consume less power"

Pruning

Pruning is one model compression technique that allows the model to be optimized for real-time inference for resource-constrained devices.



Pruning



Quantization

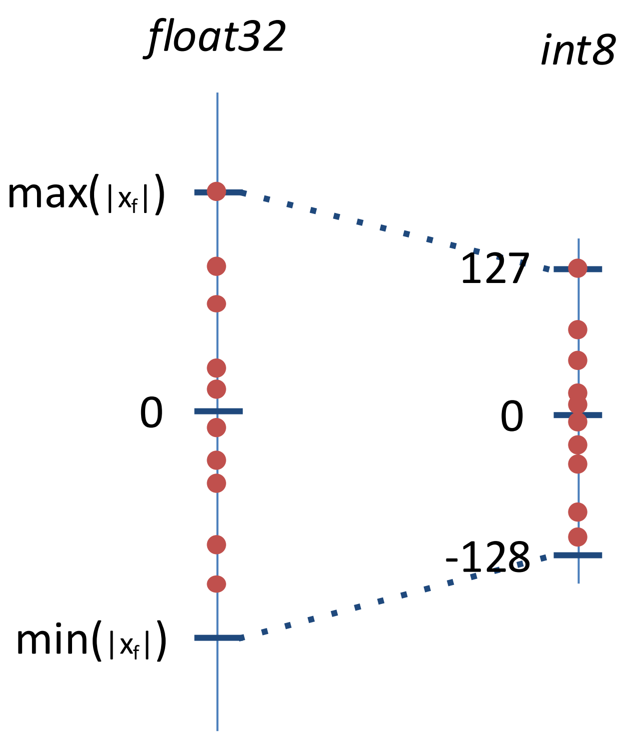
- Post-training Quantization
- Quantization-aware Training

Post-training Optimization Tool (POT) is the fastest and easiest way to get a quantized model. A conversion technique that can reduce model size while also improving CPU and hardware accelerator latency, with little degradation in model accuracy.

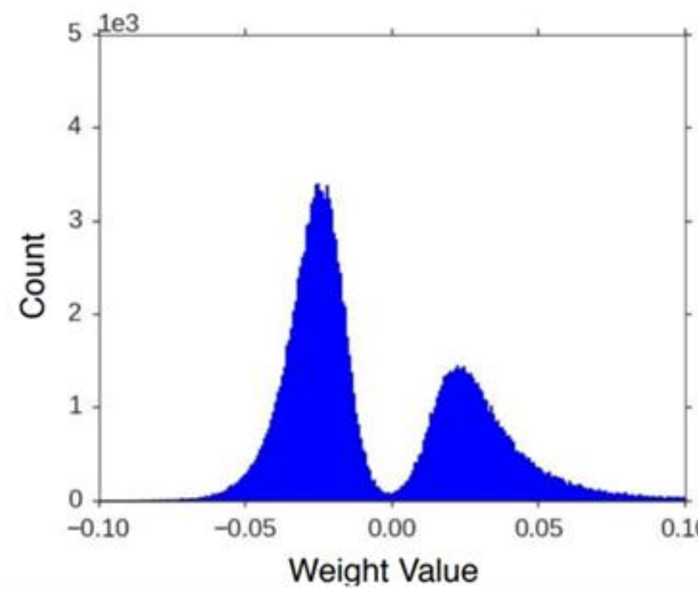
Quantization aware training ***emulates inference-time quantization***, creating a model that downstream tools will use to produce actually quantized models. The quantized models use lower-precision (e.g. 8-bit instead of 32-bit float), leading to benefits during deployment.

Quantization

Quantization is an optimization that works by reducing the precision of the numbers used to represent a model's parameters, which by default are 32-bit floating point numbers. This results in a smaller model size, better portability and faster computation.



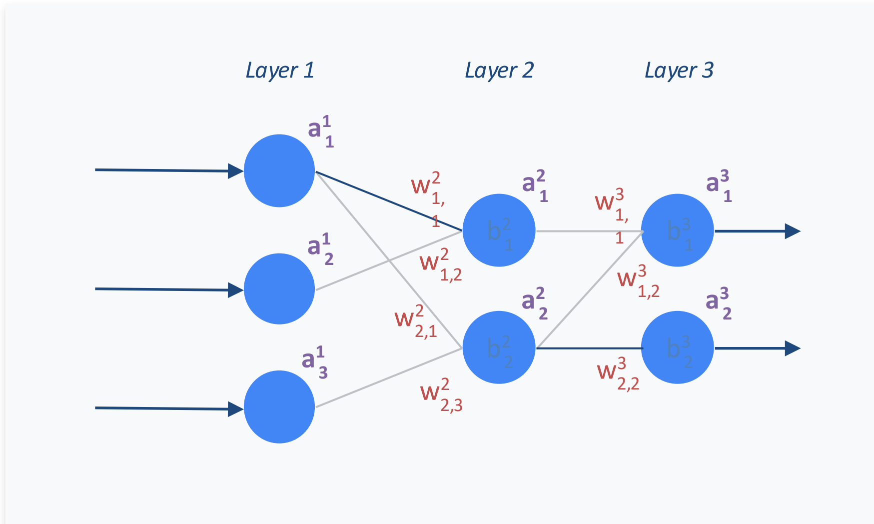
Why it works?
Weight distribution for AlexNet shows how most weight values are concentrated in a small range.



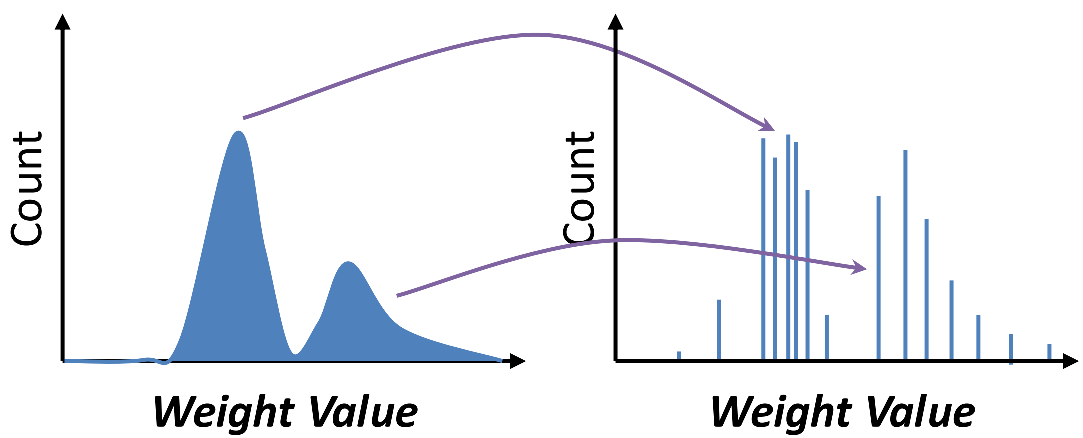
What to Quantize?

Quantize

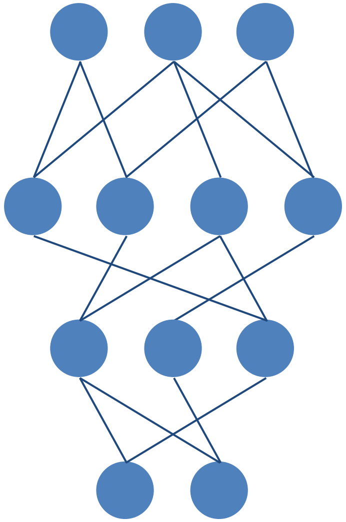
- Weights
- Biases
- Activations

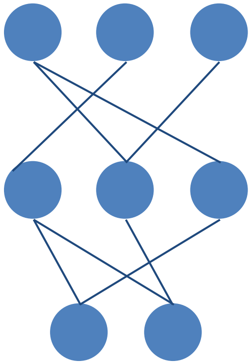


Reduce Precision (Discretize)



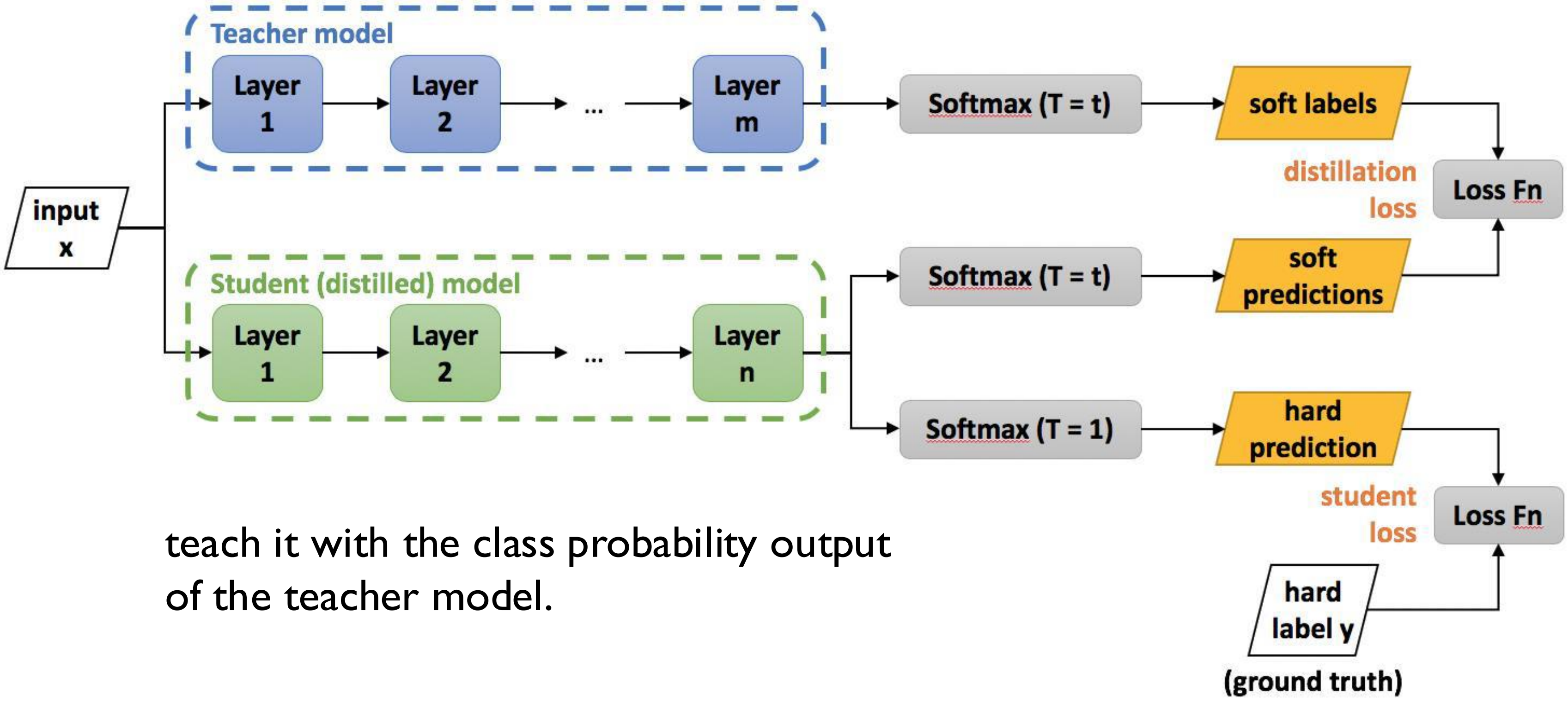
Knowledge Distillation





knowledge distillation is the process of transferring knowledge from a large model to a smaller one.

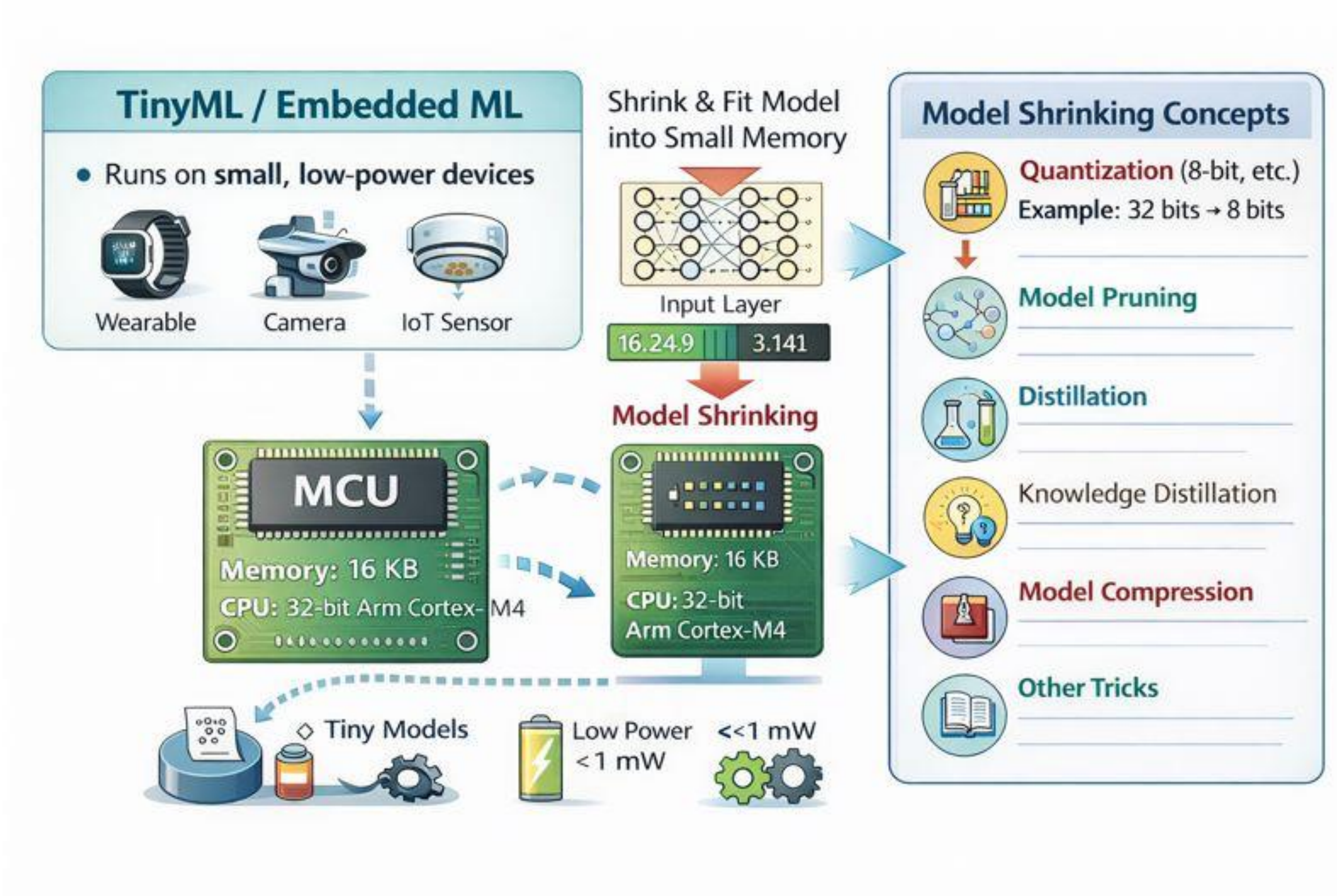
Transferring Knowledge



teach it with the class probability output of the teacher model.

TinyML

Embedded ML



Deploy in the Tiny Device

Embedded ML