Hands-on tinyML

A (very) short guide to deployment on MCUs

Francesco Paissan

University of Trento, Mila - Québec AI Institute

April 23, 2025

francescopaissan.it/tinyml-tutorial

Table of Contents

Modelling

Inference

Learning

Modelling

Inference

Learning

Previously on...

- We design a convolutional neural network for CIFAR10;
- We validated its performance on a representative test set;

Previously on...

- We design a convolutional neural network for CIFAR10;
- We validated its performance on a representative test set;

Now we want to deploy it!

Overview of the deployment

1. Export the model out of PyTorch (either ONNX, or TFLite)[1]
2. Visualize the model and make sure everything looks good;
3. Start the on-device deployment... but how?

[1] https://github.com/micromind-toolkit/micromind/

Depends on the hardware and vendors

Different vendors have different tools, and sometimes this should guide the model design![2]

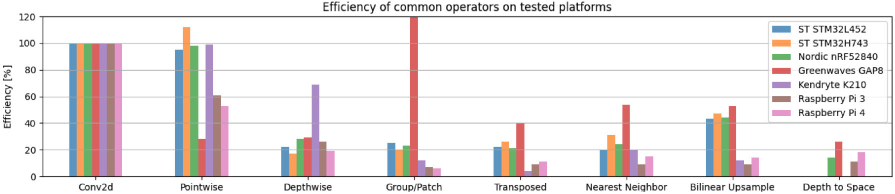

Figure 2. Measured real-world efficiency on different platforms for the benchmarked operators. Each color corresponds to a different hardware platform, each bar corresponds to the efficiency (defined in equation 4) of the tested operator, indicated in the horizontal axis.

[2] Ancilotto, Paissan, Farella - "XiNet: Efficient Neural Networks for tinyML", ICCV'23

Depends on the hardware and vendors

Different vendors have different tools, and sometimes this should guide the model design![2]

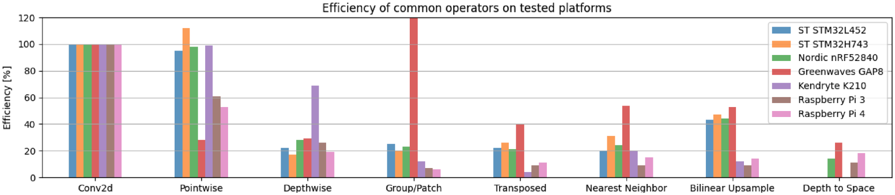

Figure 2. Measured real-world efficiency on different platforms for the benchmarked operators. Each color corresponds to a different hardware platform, each bar corresponds to the efficiency (defined in equation 4) of the tested operator, indicated in the horizontal axis.

...and of course, you can always write plain C code for all operators. But are you sure you could write very optimized kernels?

[2] Ancilotto, Paissan, Farella - "XiNet: Efficient Neural Networks for tinyML", ICCV'23

A couple of examples

Depending on how strong your computing background is, you could:

- Write your own kernel
- Use optimized kernel libraries (https://github.com/ARM-software/CMSIS-NN)
- Use automated deployment tools (CubeAI, EdgeImpulse, ...)

It depends on what you want to optimize.

A very easy recipe

When approaching a new problem, try to:

- Understand the art for that task: what kind of architecture is used? How big are these networks? Where is the computational bottleneck?
- Favor the use of pre-trained neural nets, and eventually distill those;[3]
- Iterate very quickly on the hardware you are targeting
- Don't forget about the impact of quantization and of the compromises you are doing on the way

[3] tinyCLAP