A Hardware-Aware Energy Estimator for μNAS

Topology-aware per-inference energy prediction for TinyML on microcontrollers

Martynas Šmitas | Supervisors: Hao Liu, Marco Zuñiga Zamalloa | EEMCS, Delft University of Technology

1. Motivation & Goal

TinyML runs deep learning directly on microcontrollers (MCUs) — in wearables, wildlife collars, and field sensors that live on battery or harvested solar power, where **energy is a hard constraint**, not an afterthought.

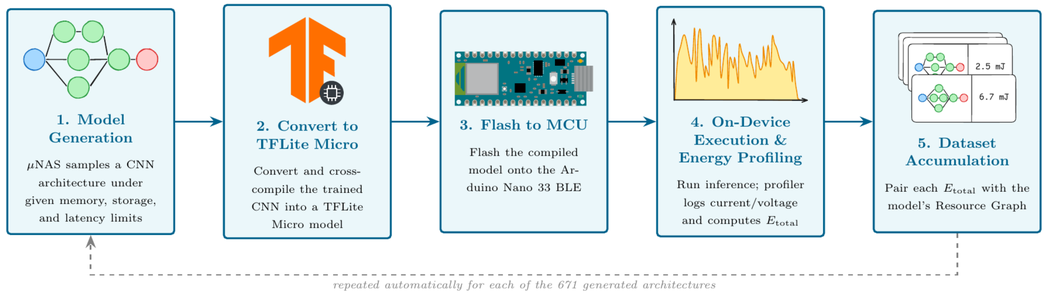
Frameworks such as μ**NAS** search CNNs that fit an MCU's memory, storage, and latency limits, scoring each candidate by its **Multiply–Accumulate (MAC) count** as a cheap proxy for hardware cost.

But MACs assume every operation costs the same. On real MCUs, memory access, caching, scheduling, and the graph rewrites **TensorFlow Lite Micro** applies at compile time (e.g. operator fusion) make equal-MAC networks behave very differently.

Goal. Create an **accurate, hardware-specific energy estimator** compatible with μNAS search loop — so energy is optimised alongside memory, storage, and latency.

2. An automated Hardware-in-the-Loop energy profiling pipeline

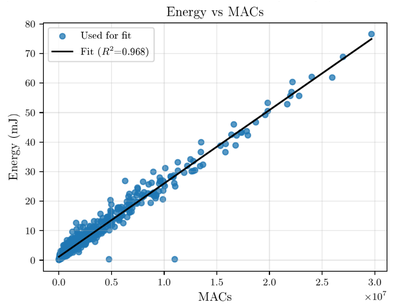
No public MCU energy dataset exists with a large range of MCUs -> built an automated MCU-agnostic pipeline. Fully automated, hardware- and task-agnostic: it runs the complete model lifecycle (generate → convert → flash → profile → accumulate) with no manual measurement, repeated for each model.



Each candidate's measured per-inference energy is paired with its *Resource Graph* (the structured operator/data-flow description μNAS already produces) to form one training example.

3. Baseline: MAC-only regression

A linear fit on μNAS MAC counts — the same proxy μNAS itself relies on.



Strong *macro* fit (R^2 = 0.985), matching μNAS's reported latency correlation. **Yet the per-network error is large:** a global R^2 hides that MACs miss memory movement, execution context, and TFLite optimisations.

Mean error 85.4%, with a catastrophic tail — off by >3× on its worst predictions.

4. Our estimator: a Directed Acyclic Graph NN

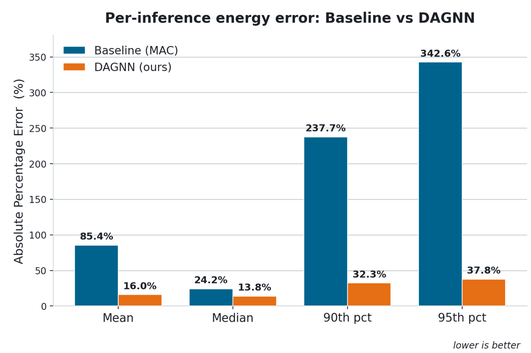
The estimator consumes the network's **Resource Graph** (nodes = operations, edges = data dependencies) — the *same* graph μNAS already builds.

Topology-aware by construction. Unlike a standard GCN, the DAGNN respects edge *direction*: it visits nodes in **topological order**, each aggregating its predecessors — exactly how tensors flow at inference, so it learns how energy accumulates along the path. Every operator (Conv2D, DWConv2D, Dense, Pool, Add) is encoded as one fixed **21-dim node vector**:

Dim	Feature	Applies to
0–5	operator type (one-hot)	all
6–8	log(1+x): H, W, C/units	all
9–12	filters, kernel, stride, pad	Conv, DWConv
13–15	batch-norm, activation, bias	Conv/DW/Dense
16–17	log(1+units), flatten	Dense
18–20	pool size/type, equal inputs	Pool, Add

Model: 2 DAGNN layers, hidden dim 128 → graph pooling → FC head → scalar energy. Trained 500 epochs (lr 10^{-3}), 5-fold CV.

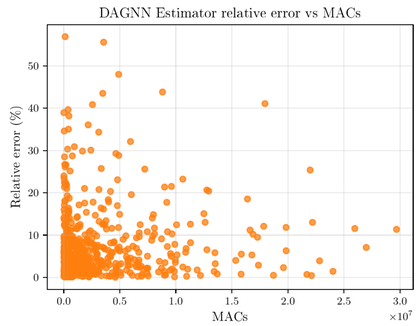
5. Results: accurate across the whole search space



Absolute Percentage Error of per-inference energy (lower is better).

The DAGNN cuts **mean error 85.4%** → **16.0%** and **collapses the tail** (95th pct 342.6% → 37.8%) — staying accurate across the *whole* search space, not just on average. This robustness is what matters for NAS: a single severe misprediction can promote a wasteful candidate or reject an efficient one.

6. Error stays tight everywhere



Relative error vs. MACs for the DAGNN: **low and bounded** across both small and large networks — a tight tail is what makes energy a *trustworthy* search signal.

7. Plugging back into μNAS

- **Shared interface:** reuses the Resource Graph μNAS already builds — no extra conversion.
- **Offline once, online cheap:** trained once, then frozen; each query is **one forward pass (ms)**, cheaper than μNAS's accuracy estimate.
- **Energy as a criterion:** a *hard constraint* (discard over-budget candidates) and/or part of the *multi-objective fitness*.
- **Retargetable:** for a new MCU, re-run the pipeline, retrain, swap weights — search code untouched.

8. Conclusion & future work

An automated HIL pipeline plus a **topology-aware DAGNN** turns per-inference energy into a **practical, accurate NAS objective** on microcontrollers — 5.3× lower mean error than the MAC proxy, with a 9× smaller error tail.

Next:

- **Cross-hardware** transfer to other MCUs.
- **Multi-task** validation beyond CIFAR-10 (e.g. Keyword Spotting).
- **Smarter datasets** — guided, non-random model generation.

Dataset and pipeline are **publicly available**.