Stockholms universitet

Adaptive AI Task Partitioning and Offloading in Heterogeneous Edge-Cloud Networks

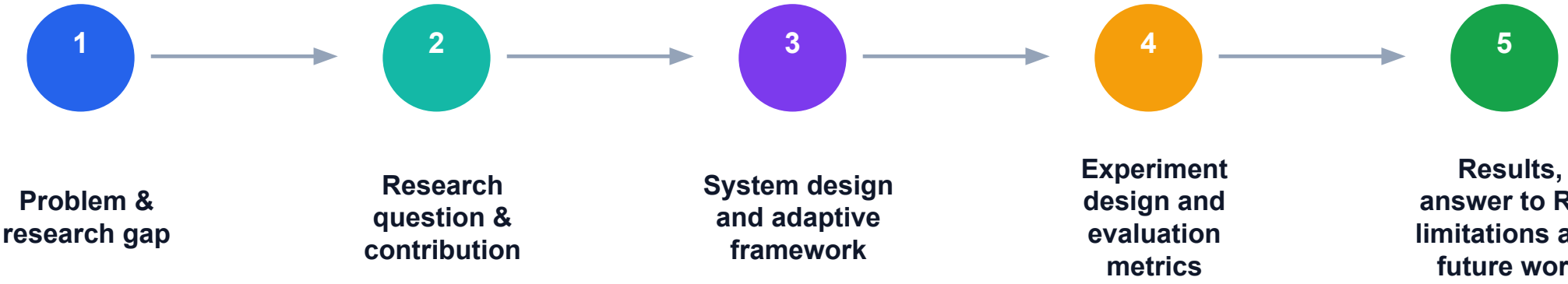
Akuen Akoi Deng

Eimantas Butkus

Supervisor: Dr. Praveen Kumar Donta

2026-05-18

Presentation roadmap



Problem: Static AI partitioning and offloading does not fit dynamic edge-cloud conditions

Stockholms universitet

- **IoT devices are resource-constrained**
  - Limited battery, compute power, and memory make running full AI models locally impractical
- **Static partitioning cannot adapt**
  - Most existing solutions use fixed splits that don't respond to runtime changes in network or compute load

Gap

- **Simulation-dominated evaluation**
  - Most frameworks are tested in simulated environments that assume ideal conditions and miss real-world variability
- **Focused mainly on latency, throughput, or inference speed**
  - Several related works optimise end-to-end latency, inference time, throughput, or accuracy, but give limited attention to energy consumption on resource-constrained devices.

Research Question

- “To what extent can an adaptive model partitioning algorithm reduce the energy consumption of resource-constrained devices without violating real-time latency constraints in a heterogeneous edge-cloud network?”

Contribution

1. Artefact

A Python framework that dynamically selects CNN split points across Pi, laptop and GPU PC.

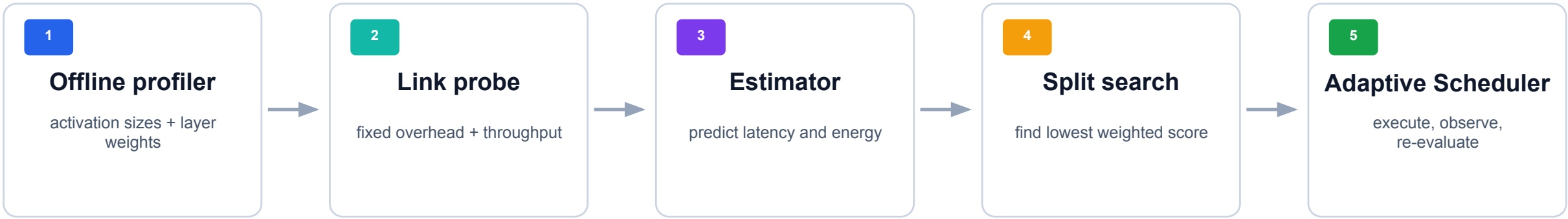
2. Testbed

A real 3-tier physical setup instead of a pure simulation.

3. Evidence

Quantitative comparison against static partitioning using latency and energy metrics.

Proposed Framework: five-stage decision pipeline



Objective: minimise edge-device energy without violating real-time latency constraints.

Decision Policy: Scoring Function

Stockholms universitet

Weighted objective

For every valid split (i, j), the scheduler computes:

$$S(i,j) = w_{PiE}\frac{E_{Pi}(i,j)}{E_{Pi,norm}} + w_{TotE}\frac{E_{tot}(i,j)}{E_{tot,norm}} + w_{Lat}\frac{L(i,j)}{L_{norm}}$$

Weight constraint:

$$w_{PiE} + w_{TotE} + w_{Lat} = 1$$

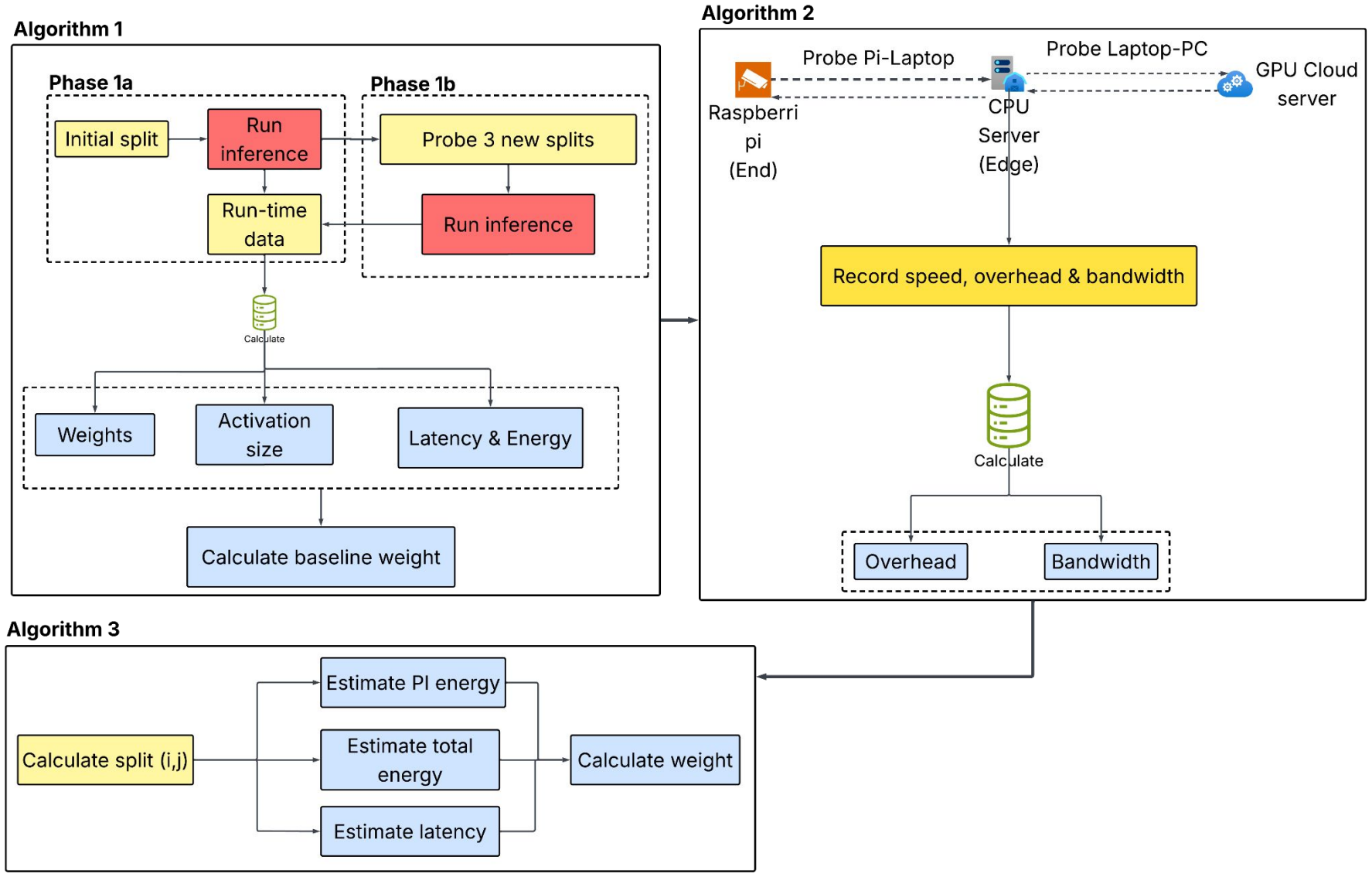
Latency feasibility condition

A split is considered feasible only when:

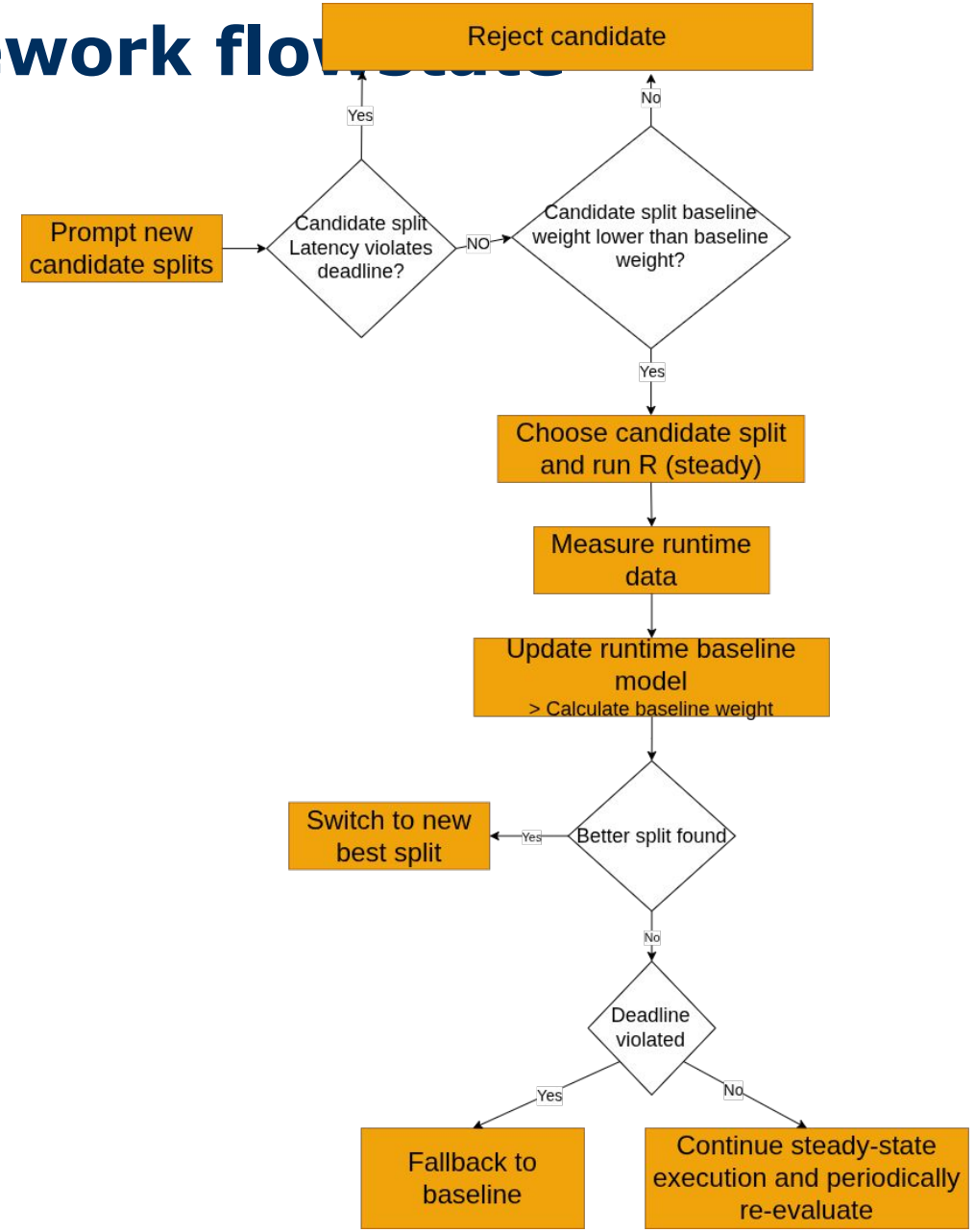
$$L(i,j) \le L_{max}$$

Only feasible candidates are ranked by the score S(i, j).

Proposed Framework lifecycle



Proposed Framework flowstate



Experiment Design

• Hardware Testbed

Node	Device	Processor	RAM
End	Raspberry Pi 5	4-core ARM Cortex-A76 2.4 GHz	8 GB LPDDR4X
Edge	Laptop	4-core i7-10510U 1.8–4.9 GHz	16 GB DDR4
Cloud	Desktop PC	RTX 4070 Ti (7680 CUDA cores)	32 GB DDR5

• DNN Models

Model	Parameters	GFLOPs	Size	Why chosen
VGG-16	138 M	15.5	528 MB	Heavy, dense architecture
AlexNet	61 M	0.7	234 MB	Mid-range, classical CNN
MobileNetV2	2.2 M	0.3	8.8 MB	Lightweight, edge-optimised

Energy Measurement

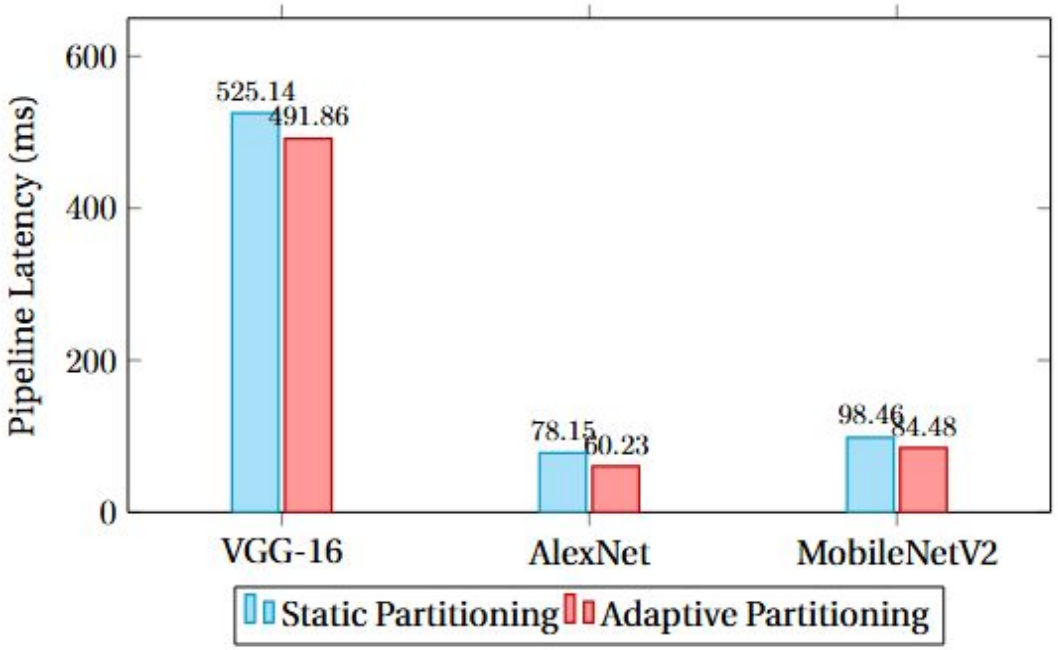
500 inferences × 10 repetitions • Dummy input: 1×3×224×224

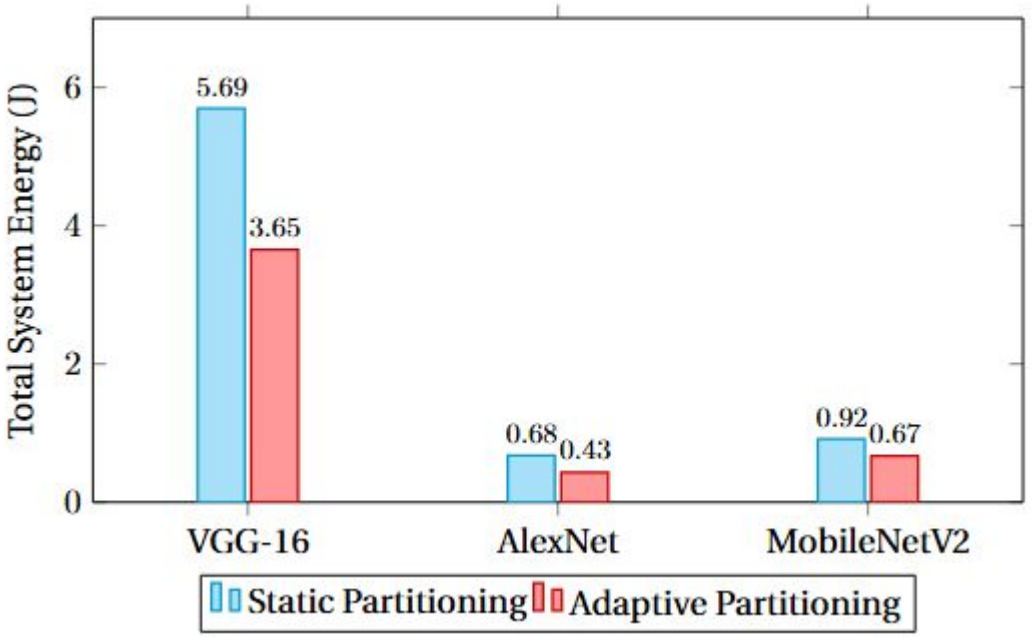
Pi: $E = P \times t$ (fixed 12W power model)

Laptop: $E = e_1 - e_0$ (Intel RAPL via Linux powercap)

PC: $E = (P_0+P_1)/2 \times t$ (NVIDIA NVML)

Results: Static vs Adaptive Partitioning





Energy reduced by 27–36% • Latency reduced by 6–23% • All models improved on both metrics

Results: Percentage Improvement & Consistency

Stockholms universitet

VGG-16	AlexNet	MobileNetV2
35.82% energy reduction	**35.70%** energy reduction	**27.09%** energy reduction
6.34% latency reduction	**22.92%** latency reduction	**14.20%** latency reduction

Statistical Consistency (mean ± stdev across runs)

Model	Approach	Latency (ms)	Energy (J)
VGG-16	Static	525.14 ± 11.31	5.693 ± 0.158
	Adaptive	491.86 ± 17.00	3.654 ± 0.202
AlexNet	Static	78.15 ± 11.81	0.675 ± 0.091
	Adaptive	60.23 ± 4.65	0.434 ± 0.082
MobileNetV2	Static	98.46 ± 1.65	0.919 ± 0.025
	Adaptive	84.48 ± 2.05	0.670 ± 0.063

Energy distributions: no overlap for any model → improvements are real. AlexNet adaptive stdev 2.5× lower than static.

Discussion

Per-model analysis

VGG-16 — **Largest energy saving (2.04 J), smallest latency gain (6.34%)**

Large activation tensors (3.2 MB at first block) — pushing cut point earlier saves compute but increases transfer cost

AlexNet — **Best on both metrics (35.70% energy, 22.92% latency)**

11×11 stride-4 first conv aggressively downsamples → small activation tensors at cut points → low transfer overhead

MobileNetV2 — **Smallest gains (27.09% energy), latency still above Pi-only**

Already optimised for edge — Pi runs it in 71.9 ms at 0.86 J. Network transfer overhead exceeds compute savings from offloading

Adaptivity evidence: network throttling test

Throttled to ~5 Mbit/s
Shifted from 1.6 MB → 98 KB WAN payload

→

Restored to ~40 MB/s
Migrated back to original split within 1 re-eval window (~60s)

Conclusion

Answer to RQ

An adaptive model partitioning algorithm can reduce the energy consumption of resource-constrained devices by 27% to 36% without violating real-time latency constraints. In all three models tested, latency was also reduced (6–23%), demonstrating that energy optimisation and latency preservation are not mutually exclusive in a real heterogeneous edge-cloud network.

Key takeaways

- Real hardware testbed — not simulation. Results reflect actual thermal throttling, OS noise, network variability.
- Periodic re-evaluation makes the framework genuinely adaptive — demonstrated via live network throttling.
- Lightweight approach — single objective function, no RL, no heavy computational overhead for routing decisions.
- Results comparable to recent literature (Li et al. ~27%, Chen et al. ~35–43%) despite different optimisation objectives.

Limitations & Future Work

Limitations

- Only CNN models tested — no transformers, RNNs, or encoder-decoders
- No direct head-to-head comparison with reimplemented state-of-the-art (EdgeCI, DeepThings)
- Energy on Pi and Laptop are estimates (fixed power model, RAPL), not hardware power meters
- Dummy input tensors rather than real image datasets (CIFAR, ImageNet)
- Single network topology only (Pi → Laptop → PC chain)

Future Work

- Extend framework to transformers and encoder-decoder architectures
- Use hardware power meters (USB power monitors) for precise energy readings
- Multi-objective scoring function incorporating temperature, memory usage, current GPU/CPU load
- Larger-scale deployment with more edge nodes and network topologies
- Real image datasets to validate under production-like data pipelines

Stockholms universitet

Thank you!
Questions
…

2026-05-18