tinyML® Research Symposium

Enabling Ultra-low Power Machine Learning at the Edge

April 22, 2024

www.tinyML.org

SCHOOL OF ELECTRICAL AND INFORMATION ENGINEERING

Simulating Battery-Powered TinyML Systems Optimised using Reinforcement Learning in Image-Based Anomaly Detection

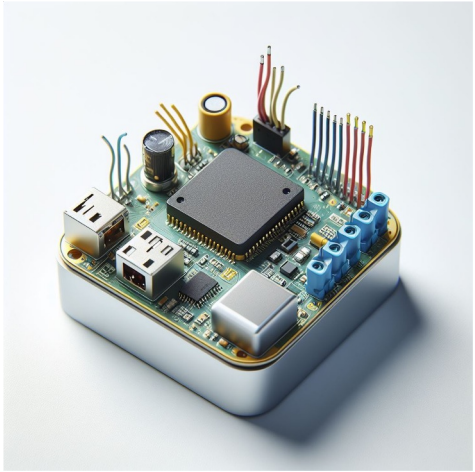
UNIVERSITY OF THE WITWATERSRAND, JOHANNESBURG

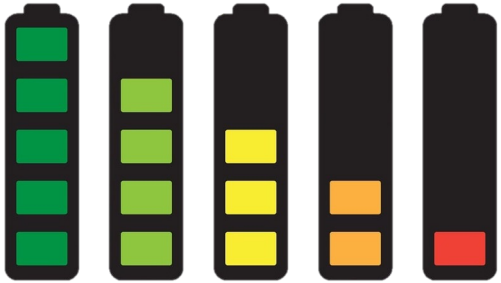
Jared Ping

TINY ML

Motivation

- Many IoT solutions rely on battery or unreliable supply, such as solar.
- Increased ML capabilities = increased compute = faster battery drain.
- Broader and complex action set requires improved optimisation.
- How to improve feasibility of battery-powered tinyML IoT solutions?



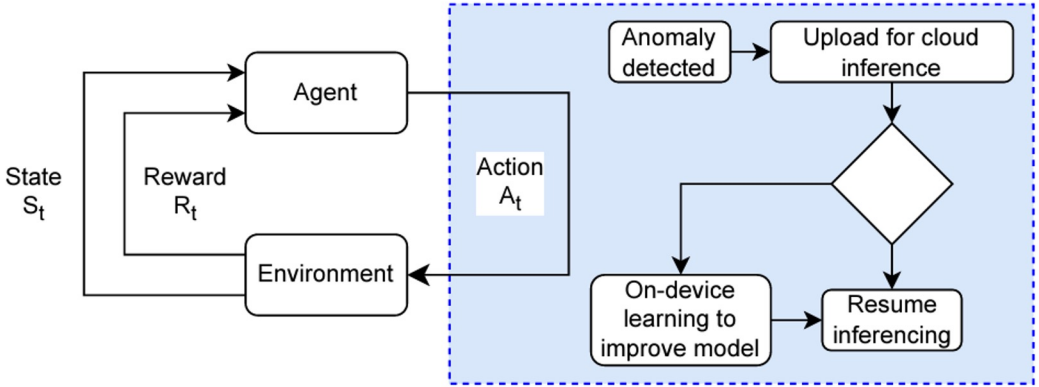


Problem definition (1)

- TinyML research has enabled inferencing and training on resource-constrained hardware.
- Challenge of balancing image-based anomaly uploads versus performing on-device training to prolong deployment battery life.
- Utilising cellular medium (NB-IoT), sufficient bandwidth for small image upload.
- Cost of uploads and training are most costly.

Problem definition (2)

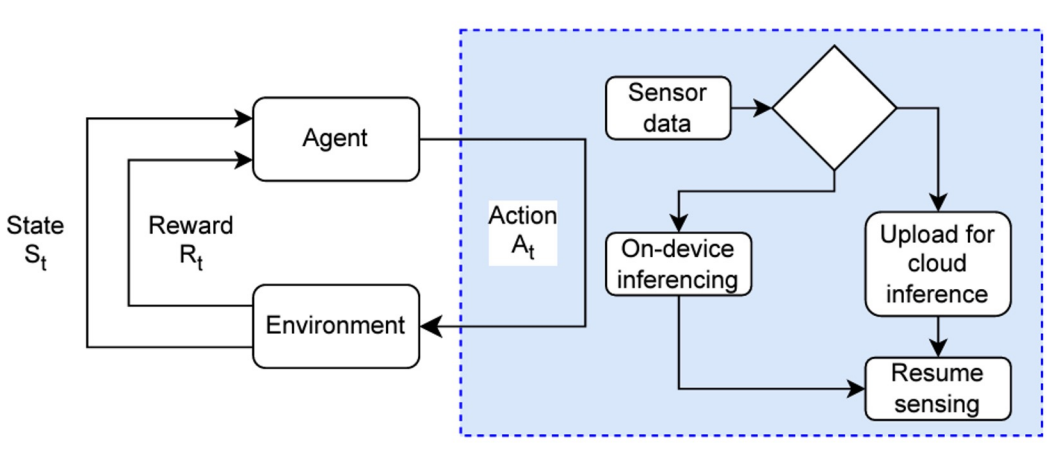
- Using RL to balance the action selection.
- Avoid unsuccessful training attempts.
- Compare against predefined schemes.



Related work (1)

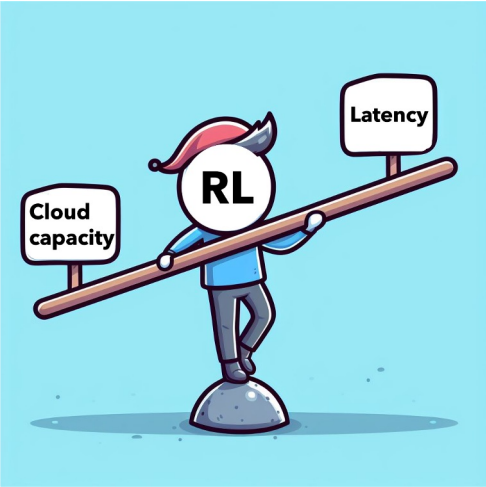
- Liu et al. in 2019 proposes **Q-greedy reinforcement learning** (RL) to optimise the power vs latency trade-off.

- System supports on-device inference and cloud offloading capabilities.

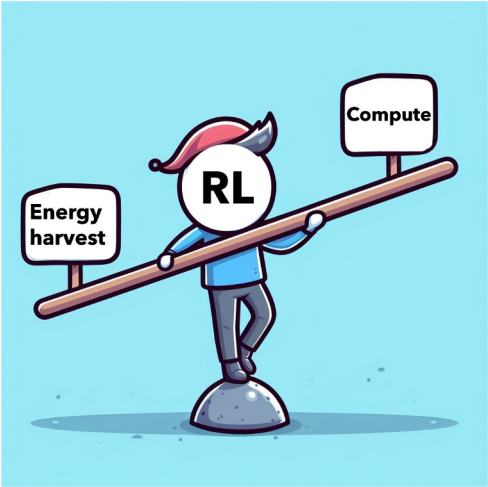


Related work (2)

- Ren et al. in 2019 further considers transmission delay vs cloud compute capacity optimisation using RL.



- Basaklar et al. in 2022 considers RL in scope of tinyML system optimisation when performing compute in a variable energy environment.



Optimisation schemes (1)

Static	Dynamic	Autonomous
Determined at compile or load time.	Behaviour adoption required or known conditions vary.	Dynamic optimisation whereby optimal action is self-learned.
Minimal compute required on system.	Increased compute required on system.	Potential for large compute requirements on system.
System cannot adjust once deployed.	System adopts, within defined constraints, to environment.	Systems adopts through its own learning without constraints.

Optimisation schemes (2)

- Autonomous solution implements reinforcement learning, specifically **decayed epsilon-greedy Q-learning**.

- Yields optimal results within a small dataset and is well suited to a small discrete environment.

- Well-suited to single-agent environment, and moderate number of state-action pairs yield small memory footprint that many resource-constrained MCUs can support.

- Can be pre-trained in the cloud for initial deployment. Supports ongoing learning post-deployment.

Optimisation schemes (3)

Static approach

- Pre-defined training threshold.
- Once threshold reached, train on each anomaly until successful.
- Unaware of any environmental conditions.

Algorithm 1: Static Optimisation Algorithm

```
N ← 0 ;        /* Number of classified anomalies */
T ← 35 ;         /* Required training threshold */
V ← 0.85 ;     /* Training validation threshold */
if N >= T then
    R ← TrainAndValidate() ;   /* Validation acc. */
    if R >= V then
        N ← 0;
```

Optimisation schemes (4)

Dynamic approach

- Initially defined training threshold.
- Upon training failure, increment training threshold.
- Upon consecutive training successes, decrement training threshold.
- Reacts to environment in pre-defined way.

Algorithm 2: Dynamic Optimisation Algorithm

```
N ← 0 ;          /* Number of classified anomalies */
T ← 10 ;           /* Required training threshold */
V ← 0.85 ;       /* Training validation threshold */
S ← 0 ;          /* Successful training iterations */
Z ← 5 ;            /* Data set reduction threshold */
if N >= T then
    R ← TrainAndValidate() ;   /* Validation acc. */
    if R >= V then
        N ← 0;
        S ← S + 1;
        if S >= Z then
            T ← T − 1;
    else
        S ← 0;
        T ← T + 1;
```

Optimisation schemes (5)

Autonomous approach

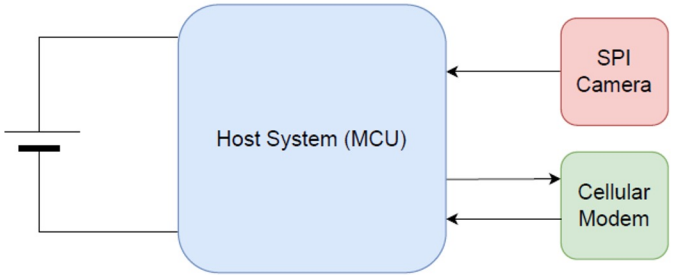
- No pre-/initially defined training threshold.
- Training thresholds determined through self-learning of varying environment conditions.
- Focuses on learning pre-deployment, before eventually exploiting what is learned post-deployment.
- Action triggered by anomaly count determined from learned Q-table.

Algorithm 3: Initial Q-Table Training Algorithm

```
α ← 0.2 ;                                   /* Learning rate */
γ ← 0.95 ;                                  /* Discount factor */
ε ← 1 ;                                     /* Initial exploration rate */
Initialise Q-table Q(s, a) state-action pairs (s, a) to zero;
for each episode do
    Initialise state s;
    while s is not terminal do
        α ← α × 0.995;
        Pick action a for state s via decayed ε-greedy policy:
            ε ← ε × 0.99;
            rnd ← random number between 0 and 1;
        if rnd < ε then
            a ← choose random action from possible
              actions;
        else
            a ← argmax over a from Q(s, a);
        Do action a, observe reward r and next state s′;
        Update Q-value for state-action pair (s, a):
            Q(s, a) ←
              Q(s, a) + α (r + γ max_a′ Q(s′, a′) − Q(s, a));
        s ← s′;
```

System simulation (1)

- Simulate core functional blocks required to implement a battery-powered tinyML anomaly detection system.
- Consumption profile modelled using low cost, easily sourced components selected for possible deployable solution.
- System exposed to simulated environments with varying anomaly occurrence ratios.
- Determine the average deployment battery life per optimisation scheme.



System simulation (2)

- Simulated system emulates a single-use ideal battery offering 5V over a battery capacity of 3.5 Ah, yielding 17.5 Wh of deliverable energy.

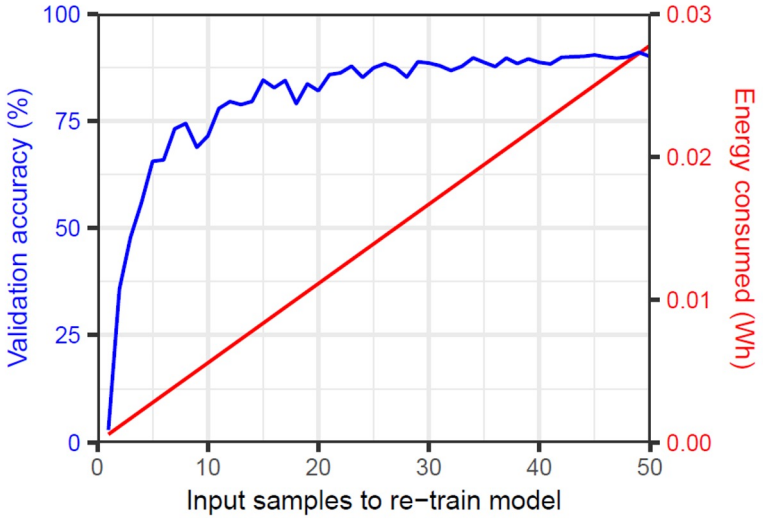
- Mitigates variance in discharge profiles of different cell chemistries.

State/Action	Responsible Component	Simulated Hardware	E_{avg}
System Active	MCU	STM32F746xx	400 mWh
System Sleep	MCU	STM32F746xx	50 µWh
Infer Result	MCU	STM32F746xx	17 µWh
Re-training	MCU	STM32F746xx	556 µWh/img
Image Capture	SPI camera	OV2640	180 µWh
Anomaly Upload	Cell modem	SIM7000E	3 mWh

System simulation (3)

- On-device training behaviour is modelled to cater for variance in training, assuming relative consistency in data quality.

- Simulating MCUNet VWW model requiring 478 KB and 190 KB of MCU flash and RAM, respectively.



Benchmarks and results (1)

- Four benchmark environments with varied anomaly ratios: **5%**, **10%**, **20%** and **40%**

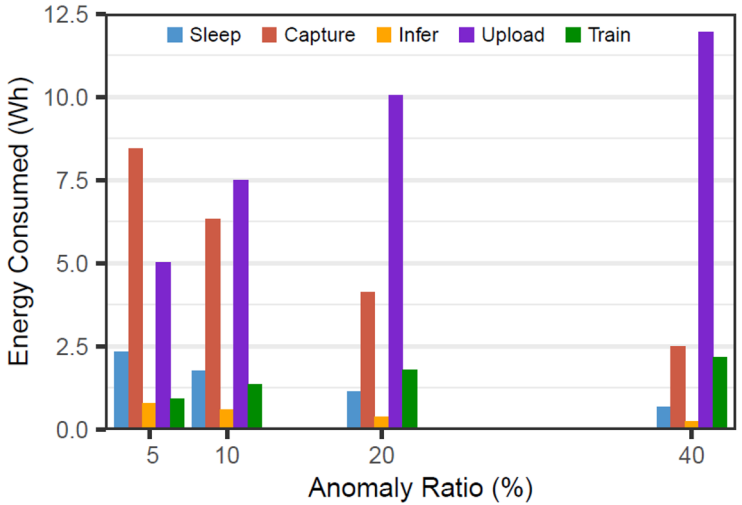
- Anomalies are artificially inserted probabilistically to ensure the target ratios are achieved.

- System is configured to wake hourly to capture an image, thereafter feeding the captured image into the onboard neural network for anomaly detection. Termed **sampling iteration**.

- If anomaly is detected, an upload is performed, yielding an emulated classification from the server.

Benchmarks and results (2)

- Optimisation algorithm determines if sufficient anomaly classifications have been obtained to attempt re-training the on-device neural network before returning to sleep.
- Repeat sampling iterations until available battery energy has been depleted.
- Consumption is accumulated per available action and state.
- Final results measure split between each action and state, and total simulated deployment battery life.

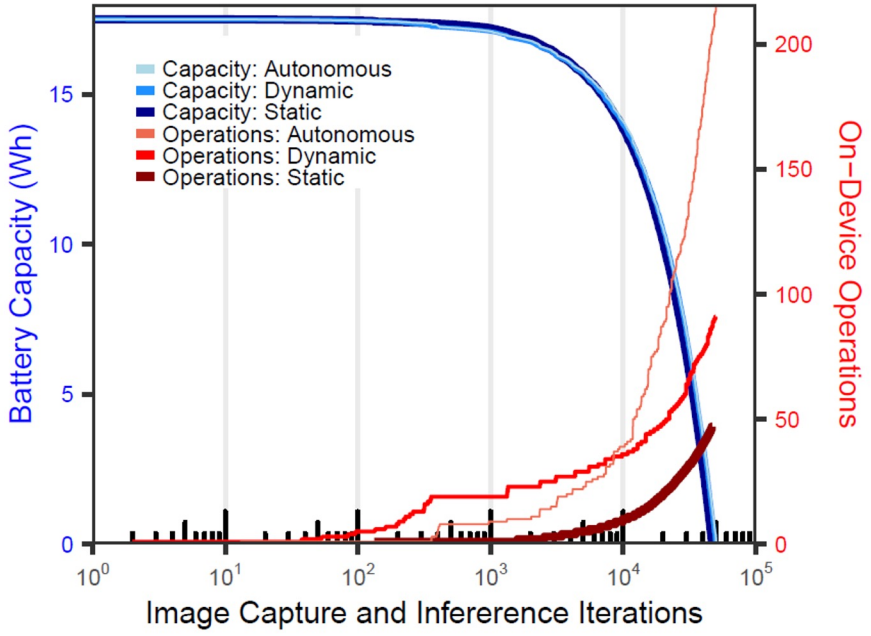


Benchmarks and results (3)

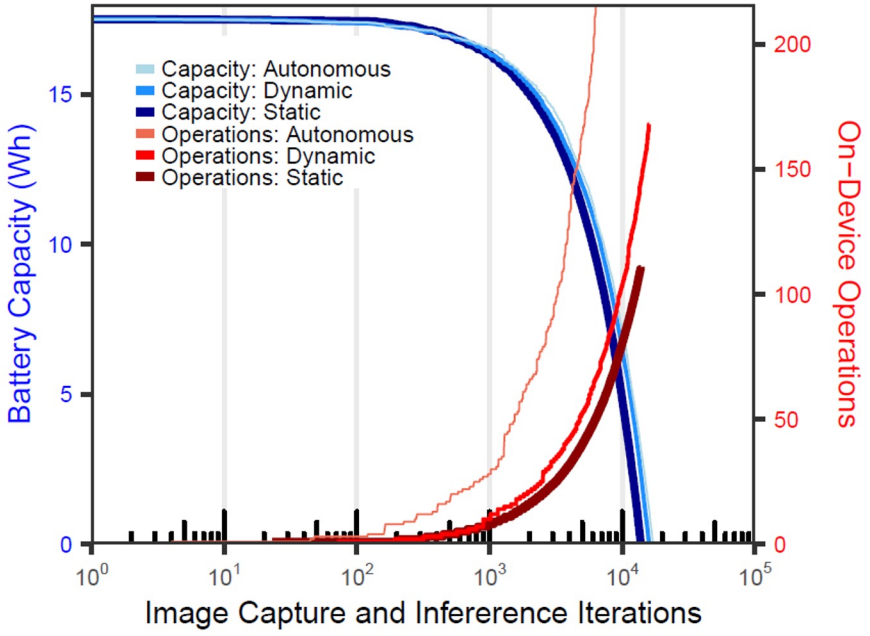
- Autonomous optimisation yields **22.86%** and **10.86%** deployment battery life **improvement** versus static and dynamic respectively.
- Further improvements through ongoing learning were mitigated by the compute requirement.
- Sleep consumption plays larger role in longer deployment durations.

Anomaly Ratio (%)	Static Battery Life (hr)	Dynamic Battery Life (hr)	Autonomous Battery Life (hr)
5	45 956	49 728	53 292
10	34 741	38 162	41 988
20	22 909	26 304	29 809
40	13 781	15 909	19 137
Avg.	29 347	32 526	36 057

Benchmarks and results (4)



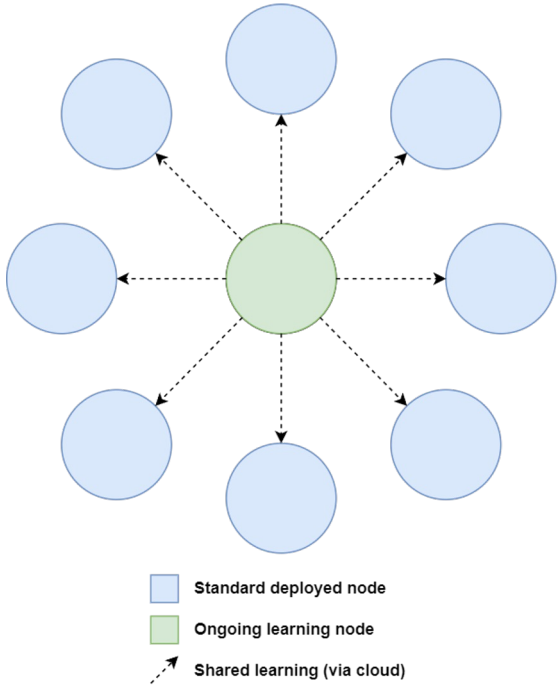
(a) Anomaly ratio of 5%.



(b) Anomaly ratio of 40%.

Deployability

- Small memory footprint of **800 B** to support the simulated Q-table.
- Ideal deployment would utilise a pilot system to collect initial training data in different conditions to enable pre-learning on the cloud rather than device.
- Ongoing learning can be utilised, and selectively enabled on desired systems, after which updated Q-table can be shared with other deployment systems.
- Upload mediums and encryption should be carefully considered for anomaly data uploads.



Conclusion and future work

- TinyML functionality continues to grow, and enhance value in several sectors.
- Power consumption of such capabilities always need to be considered and optimised, particularly for battery-powered solutions.
- RL can be a powerful tool in achieving this, with **decayed epsilon-greedy Q-learning** proving a capable solution in balancing anomaly uploads and on-device training.
- Future work includes physical experiment to validate real-world performance of the simulated system.

Thank you

Any questions/comments?

Copyright Notice

www.tinyml.org