Imperial College London

Imperial-X

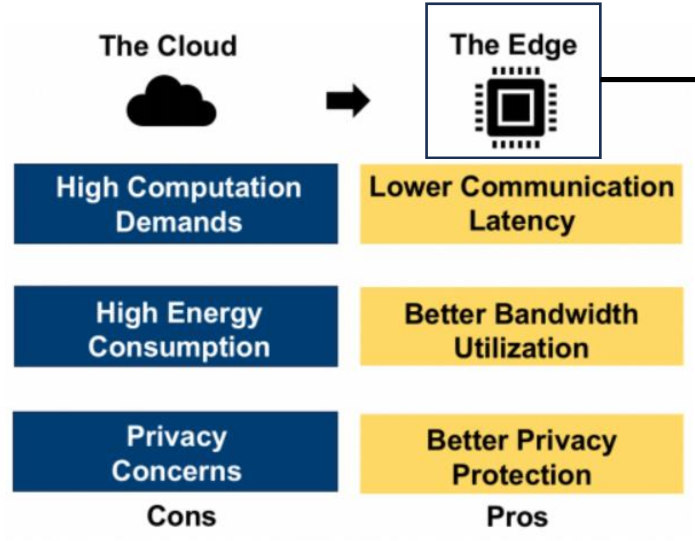
NetSys Lab

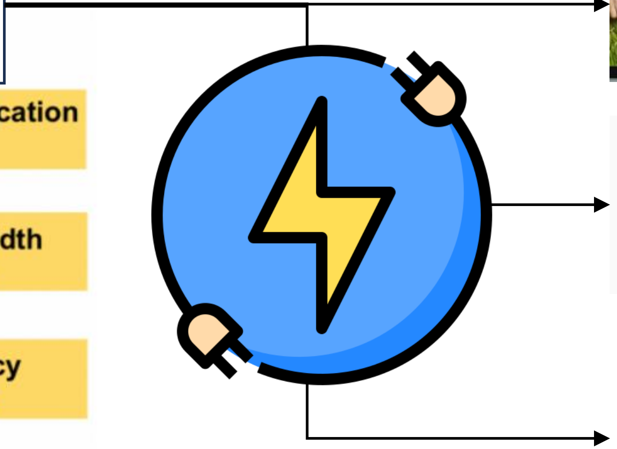
Towards Battery-Free Machine Learning and Model Personalization on MCUs

Reporter: Yushan Huang

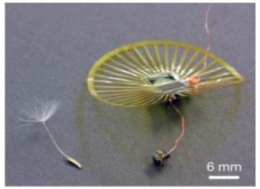
14/07/2023

Motivation


The Cloud
The Edge
High Computation Demands
Lower Communication Latency
High Energy Consumption
Better Bandwidth Utilization
Privacy Concerns
Better Privacy Protection
Cons
Pros

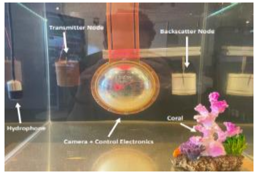


Motivation



Wind-dispersed battery-free devices [1]

Wearable energy harvesting devices [2]



Battery-free underwater camera [3]

Can we combine battery-free devices and models?

[1] Iyer, V., Gaensbauer, H., Daniel, T.L. and Gollakota, S., 2022. Wind dispersal of battery-free wireless devices. Nature, 603(7901), pp.427-433.
[2] Ylli, K., Hoffmann, D., Willmann, A., Becker, P., Folkmer, B. and Manoli, Y., 2015. Energy harvesting from human motion: exploiting swing and shock excitations. Smart Materials and Structures, 24(2), p.025029.
[3] Afzal, S.S., Akbar, W., Rodriguez, O., Doumet, M., Ha, U., Ghaffarivardavagh, R. and Adib, F., 2022. Battery-free wireless imaging of underwater environments. Nature communications, 13(1), p.5546.

Definition

Features of Resource-Constrained MCUs

- Low Power: a few mW
- Low Memory: 1 MB Flash, 128 KB SRAM
- Low Energy: a few mJ

Resource-Constrained IoT Device Concerns

- How accurate can the model be?
- How much memory is needed to run the model?
- How much energy and power is consumed to run the model?

Aim

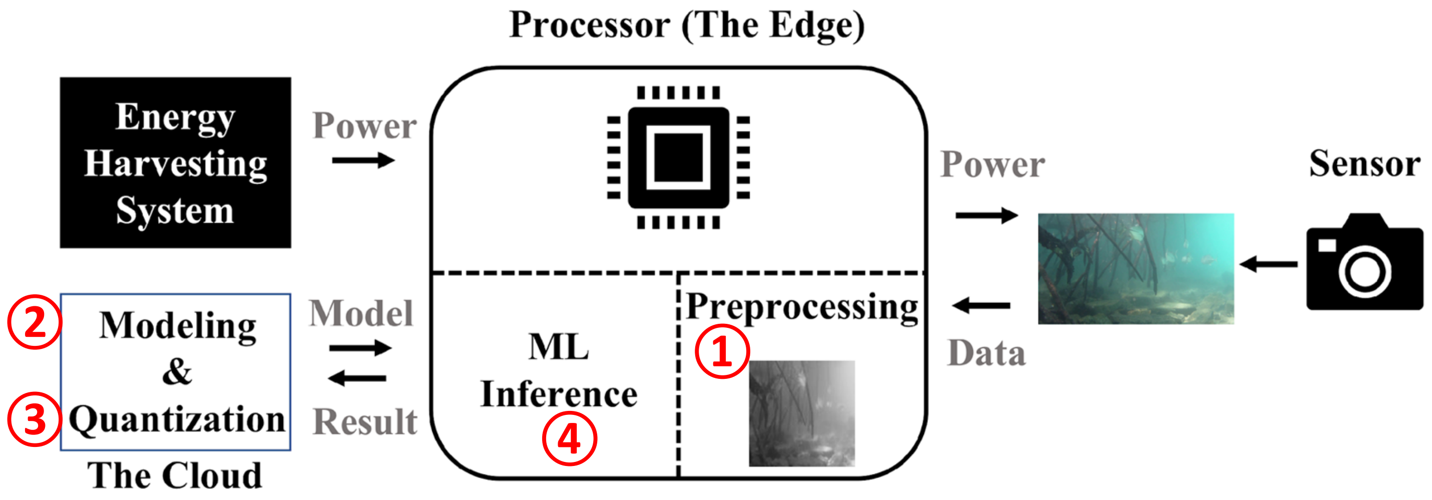
To achieve battery-free ML and model personalization on resource-constrained MCUs while optimizing power and energy consumption.

Potential Application Scenarios:

- Ambient Sensors
- In-Body Sensors
- Extreme Environmental Application [4]

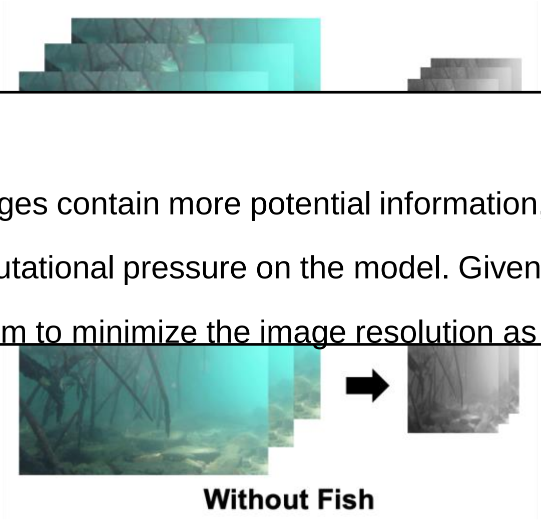
[4] Zhao, Y., Afzal, S.S., Akbar, W., Rodriguez, O., Mo, F., Boyle, D., Adib, F. and Haddadi, H., 2022, March. Towards battery-free machine learning and inference in underwater environments. In Proceedings of the 23rd Annual International Workshop on Mobile Computing Systems and Applications (pp. 29-34).

System Overview



Using underwater image recognition as an example, the system mainly has four components: ① preprocessing ② modeling ③ quantization ④ deployment and inference.

System Design - Preprocessing



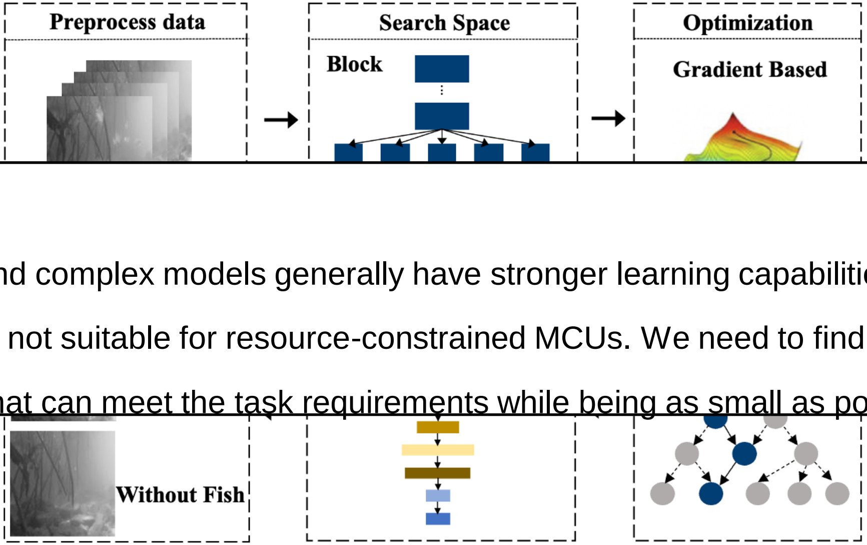
Aim?

High-resolution images contain more potential information, but they also impose more computational pressure on the model. Given the task requirements, we aim to minimize the image resolution as much as possible.

We test the pipeline on DeepFish dataset [5]. The original images are reconstructed at a resolution of 1x32x32.

[5] Saleh, A., Laradji, I.H., Konovalov, D.A., Bradley, M., Vazquez, D. and Sheaves, M., 2020. A realistic fish-habitat dataset to evaluate algorithms for underwater visual analysis. Scientific Reports, 10(1), p.14671.

System Design - Modeling



Aim?

Large and complex models generally have stronger learning capabilities, but they are not suitable for resource-constrained MCUs. We need to find a model that can meet the task requirements while being as small as possible.

We utilise a Neural Architecture Search (NAS) technique called ProxylessNAS [6] and optimize it to light version to build the model, which introduces binary gates that binarize the architecture parameters of an overparameterized network. This enables only one path to load at runtime, reducing memory consumption by not loading the entire overparameterized network to update model weights.

[6] Cai, H., Zhu, L. and Han, S., 2018. Proxylessnas: Direct neural architecture search on target task and hardware. arXiv preprint arXiv:1812.00332.

System Design - Quantization

Eq. $real\_value = (int8\_value - zero\_point) \times scale$

Aim?

Model compression techniques can reduce model size and accelerate runtime speed, thereby reducing power and energy consumption. In preliminary experiments, we employed model quantization techniques, quantizing model parameters from float32 to int8.

We utilize post-training static integer-only quantization [7]. Each real number '*real_value'* is represented in function of the quantized value '*int8_value*', a '*scale'* factor, and a '*zero_point'* parameter. Quantization scheme is an affine mapping of the integers '*int8_value'* to real numbers '*real_value'. 'zero_point'* has the same integer C-type like the *int8_value* data.

[7] Jacob, B., Kligys, S., Chen, B., Zhu, M., Tang, M., Howard, A., Adam, H. and Kalenichenko, D., 2018. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 2704-2713).

System Design - Deployment

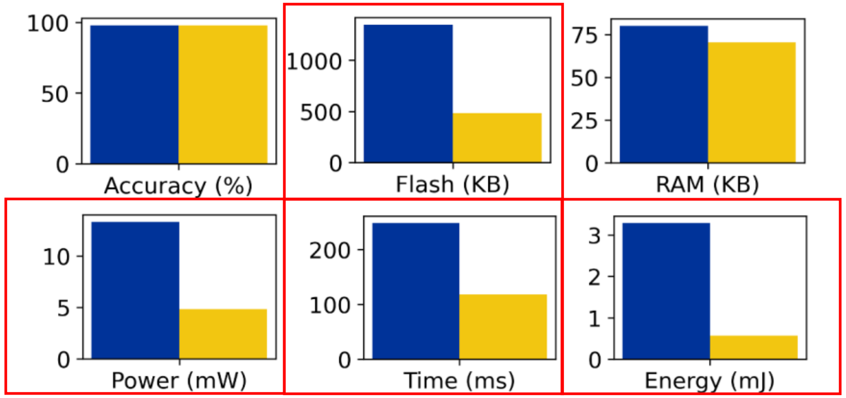
Aim?

We want to consider a more practical scenario where energy harvesting devices are not able to support edge devices like mobile phone or Raspberry Pi, but rather low-power microcontrollers.

We utilise STM32L476 as the deployment platform, which is an ultra-low-power microcontrollers with a frequency at 80 MHz, 1 MB Flash, and 128KB SRAM. We convert the model function and parameters into C language and arrays, and deploy them on the board.

Evaluation and Conclusion

		Accuracy (%)	Flash (KB)	RAM (KB)	Power (mW)	Time (ms)	Energy (mJ)
Original Model	🟦	97.88	1350.25	80.20	13.32	248	3.29
Optimized Model	🟨	97.78	483.82	70.32	4.83	118	0.57



Evaluation Results

We utilise Monzoon [8] to measure the power. Compared to the unoptimized model, we achieved an average accuracy of 97.78% with 483.82 KB Flash, 70.32 KB RAM, 118 ms inference time, 4.83 mW power, and 0.57 mJ energy consumption, which reduced by 64.17%, 12.31%, 52.42%, 63.74%, and 82.67%, respectively. The results indicate the viability of battery-free ML on MCUs, with the potential to harvest energy from certain devices.

[8] Monsoon solutions inc. https://www.msoon

Future Work

- Can we accomplish more complex tasks?
- Can we further reduce power consumption by using other model compression techniques?
- Can we adapt to energy harvesting devices with different capabilities?
- Can we accomplish fine-tune and personalisation for the models?
- How do we address the issue of missing labels on edge devices when implementing personalization?

Summary

- We propose an end-to-end solution designed for the efficient design, deployment, energy-optimizing, and execution of ML models on resource-constrained MCUs.
- We optimize the ProxylessNAS and utilised model quantization, resulting in a reduction of the model size from 1350.25 KB Flash and 80.20 KB RAM to 483.82 KB Flash and 70.32 RAM, with a slight loss of approximately 0.1% in accuracy.
- We compare the inference time, power, and energy consumption of the model before and after optimization. We achieved 97.78% accuracy with 118 ms runtime, 4.83 mW power, and 0.57 mJ energy consumption, which reduced by 52.42%, 63.74%, and 82.67% compared with the baseline.
- Our results indicate the viability of battery-free ML on MCUs with energy harvesting device.
- This work has been accepted as a poster [9] by MobiSys 2023.

[9] Huang, Y. and Haddadi, H., 2023, June. Poster: Towards Battery-Free Machine Learning Inference and Model Personalization on MCUs. In Proceedings of the 21st Annual International Conference on Mobile Systems, Applications and Services (pp. 571-572).

Personal Website
Email: yushan.huang21@imperial.ac.uk

Quantization

We utilise integer-only arithmetic during inference and floating-point arithmetic during training, with both implementations maintaining a high degree of correspondence with each other.

A basic requirement of our quantization scheme is that it permits efficient implementation of all arithmetic using only integer arithmetic operations on the quantized values (we eschew implementations requiring lookup tables because these tend to perform poorly compared to pure arithmetic on SIMD hardware). This is equivalent to requiring that the quantization scheme be an affine mapping of integers q to real numbers r, i.e. of the form:

$$r = S(q - Z) \qquad (1)$$

Where *r* is real value, *S* is scale, *q* is quantized value, and *Z* is zero point.

Equation (1) is our quantization scheme and the constants S and Z are our quantization parameters. Our quantization scheme uses a single set of quantization parameters for all values within each activations array and within each weights array; separate arrays use separate quantization parameters.

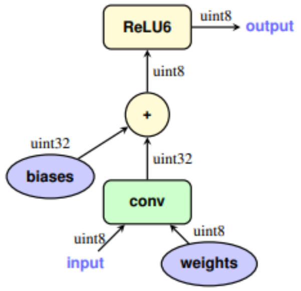
Quantization[7]

$$r = S(q - Z) \qquad (1)$$

For 8-bit quantization, q is quantized as an 8-bit integer (for B-bit quantization, q is quantized as an B-bit integer). Some arrays, typically bias vectors, are quantized as 32-bit integers.

The constant S (for “scale”) is an arbitrary positive real number. It is typically represented in software as a floating-point quantity, like the real values r. Section 2.2 describes methods for avoiding the representation of such floating-point quantities in the inference workload.

The constant Z (for “zero-point”) is of the same type as quantized values *q*, and is in fact the quantized value *q* corresponding to the real value 0. This allows us to automatically meet the requirement that the real value *r = 0* be exactly representable by a quantized value. The motivation for this requirement is that efficient implementation of neural network operators often requires zero-padding of arrays around boundaries.



Integer-arithmetic-only inference of a convolution layer

[7] Jacob, B., Kligys, S., Chen, B., Zhu, M., Tang, M., Howard, A., Adam, H. and Kalenichenko, D., 2018. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 2704-2713).

Quantization

$$r = S(q - Z) \qquad (1)$$

Consider the multiplication of two square *N* × *N* matrices of real numbers, *r1* and *r2*, with their product represented by *r3* = *r1r2*. We denote the entries of each of these matrices *rα (α = 1, 2 or 3)* as $r_\alpha^{(i,j)}$ for $1 \leqslant i, j \leqslant N$, and the quantization parameters with which they are quantized as *(Sα, Zα).* We denote the quantized entries by $q_\alpha^{(i,j)}$. Equation (1) then becomes:

$$r_\alpha^{(i,j)} = S_\alpha(q_\alpha^{(i,j)} - Z_\alpha). \qquad (2)$$

From the definition of matrix multiplication, we have

$$S_3(q_3^{(i,k)} - Z_3) = \sum_{j=1}^{N} S_1(q_1^{(i,j)} - Z_1)S_2(q_2^{(j,k)} - Z_2), \quad (3)$$

Which can be rewritten as

$$q_3^{(i,k)} = Z_3 + M \sum_{j=1}^{N} (q_1^{(i,j)} - Z_1)(q_2^{(j,k)} - Z_2), \qquad (4)$$

where the multiplier M is defined as $$M := \frac{S_1 S_2}{S_3}. \qquad (5)$$

Quantization

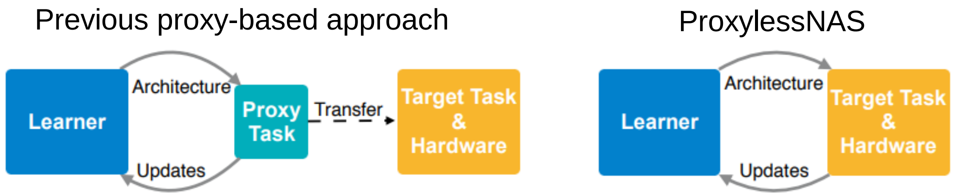
$$q_3^{(i,k)} = Z_3 + M \sum_{j=1}^{N} (q_1^{(i,j)} - Z_1)(q_2^{(j,k)} - Z_2), \qquad (4)$$

In Equation (4), the only non-integer is the multiplier *M*. As a constant depending only on the quantization scales *S1, S2, S3*, it can be computed offline. We empirically find it to always be in the interval *(0, 1),* and can therefore express it in the normalized form

$$M = 2^{-n} M_0 \qquad (6)$$

where *M0* is in the interval *[0.5, 1)* and *n* is a non-negative integer. The normalized multiplier *M0* now lends itself well to being expressed as a fixed-point multiplier (e.g. int16 or int32 depending on hardware capability). For example, if int32 is used, the integer representing *M0* is the int32 value nearest to 2^31 *M0*. Since *M0* >= *0.5*, this value is always at least 2^30 and will therefore always have at least 30 bits of relative accuracy. Multiplication by M0 can thus be implemented as a fixed-point multiplication4 . Meanwhile, multiplication by 2 ^(-n) can be implemented with an efficient bit-shift, albeit one that needs to have correct round-to-nearest behaviour.

ProxylessNAS



ProxylessNAS directly optimizes neural network architectures on target task and hardware.

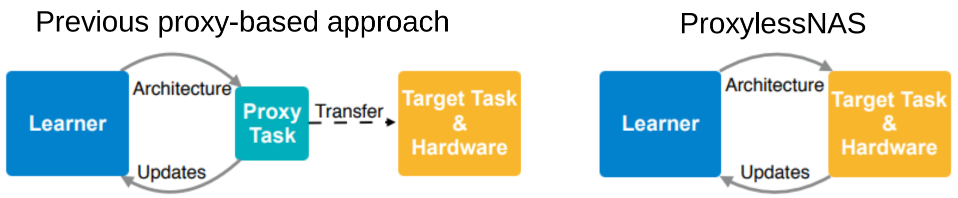
Traditional neural network search methods typically rely on proxy models to evaluate and select candidate network architectures. However, this approach suffers from discrepancies between the proxy models and the final deployed models, which can limit the performance potential of the searched architectures.

ProxylessNAS, on the other hand, bypasses the use of proxy models by directly optimizing network architectures. It evaluates and optimizes complete candidate network architectures during the search process, taking into account the requirements of the target task and hardware platform to achieve better performance.

In contrast to traditional methods with fixed candidate operations, ProxylessNAS introduces the concept of mixed operations. In the search process, each edge is no longer constrained to a specific operation but is set as a mixed operation with multiple parallel paths. This expands the search space and allows for a variety of different network architectures to be considered.

By optimizing the weights and connections of the mixed operations in the search space, ProxylessNAS can discover the best network architecture that adapts to the target task and hardware platform. This direct optimization approach provides a more accurate reflection of the final model's performance and achieves outstanding results in resource-constrained environments.

ProxylessNAS



In ProxylessNAS, the mixed operation is a mechanism used to combine different primitive operations together to form a multi-path operation. These primitive operations can include common neural network operations such as convolution, pooling, identity, zero operations, and more.

The core idea behind the mixed operation is to automatically combine different primitive operations by learning the weights and connections of each path during the search process. Each mixed operation consists of multiple parallel paths, with each path representing a specific primitive operation, such as one path representing a convolution operation and another path representing a pooling operation, and so on. This allows each mixed operation to have the characteristics of different primitive operations simultaneously, resulting in a more flexible and diverse network structure.

By optimizing the weights and connections of the mixed operations, ProxylessNAS can search for the best network architecture that adapts to the target task and hardware platform. During the search process, these weights and connections are automatically adjusted and optimized based on performance objectives to find the optimal network structure.

The introduction of mixed operations enriches and expands the search space, allowing for exploration of various network architectures. This mechanism enables ProxylessNAS to construct efficient and adaptable network models by automatically combining primitive operations, leading to improved network performance and accuracy.