On-Device Neural Architecture Search

Andrea Mattia Garavagno*†‡, Edoardo Ragusa*, Paolo Gastaldo*, Antonio Frisoli†‡, and Claudio Loconsole§‡
* Department of Electrical, Electronic, Telecommunication Engineering and Naval Architecture, DITEN
University of Genoa, Genoa 16145, Italy
† Department of Excellence in Robotics & AI, Scuola Superiore Sant'Anna, Piazza Martiri della Libertà 33, Pisa 56127, Italy
‡ Institute of Mechanical Intelligence, Scuola Superiore Sant'Anna, Ghezzano, 56010 Pisa, Italy
§ Faculty of Technological and Innovation Sciences, Universitas Mercatorum, 00186 Rome, Italy
e-mail: AndreaMattia.Garavagno@{edu.unige.it, santannapisa.it}

2025 IEEE International Conference on Consumer Electronics (ICCE) | 979-8-3315-2116-5/25/$31.00 ©2025 IEEE | DOI: 10.1109/ICCE63647.2025.10929801

***Abstract*—This paper proposes a new approach to near-sensor computing, in which a lightweight Neural Architecture Search (NAS) is performed directly on the deployment device to find the best tiny neural architecture for analyzing the real-time data acquired through sensors. This new adaptation capability can be particularly useful in the case of human-machine interfaces for which the neural network analyzing the biometrical data can be re-designed each time the user changes, after a guided data collection procedure, fighting the typical data variations between individuals on a new level. To implement the proposed approach a new NAS has been designed and then validated on the Italian Sign Language dataset (ISL), a collection of surface electromyography (sEMG) signals of the signs of the Italian alphabet, using several embedded systems. Moreover, further validation on the Case Western Reserve University dataset (CWRU), a benchmark for intelligent fault diagnosis, is presented to suggest another possible application of the proposed approach. When run on a Raspberry Pi 4, the proposed NAS performs beyond the state of the art proposing a tiny neural architecture having 0.63 times less RAM occupancy and 5.96 percentage points of more accuracy in the case of the ISL dataset; and 0.44 times less RAM occupancy and 0.2 percentage points of more accuracy in the case of the CWRU dataset.**

***Index Terms*—Neural Architecture Search, Human-Machine Interfaces, Intelligent Fault Diagnosis, TinyML**

I. Introduction

The present work proposes a new approach to near-sensor computing [1], in which a lightweight Neural Architecture Search (NAS) is performed directly on the deployment device to find the best tiny neural architecture for analyzing the real-time data acquired through sensors. In this scenario, the sensor node can collect labelled data from human feedback, find a tiny neural architecture tailored to the collected data thanks to a lightweight search procedure, and then use it to analyze the data from the attached sensors in real-time. This novel near-sensor computing approach has the potential to enhance continual learning [2], allowing not only weights but also the architecture itself to co-adapt with the data drift. The new adaptive capability has the potential to enhance a next-generation of Human-Machine Interfaces (HMIs) that analyze real-time data from biometric sensors. Each time the user changes, the tiny neural architecture analyzing biometric data is redesigned after a guided data collection procedure, addressing the typical data variations between individuals [3] on a new level.

The same approach could support applications typical of near-sensor computing such as e-health, personalized advertisement, structural health monitoring, and intelligent fault diagnosis. In the latter use case, an application-agnostic intelligent fault diagnosis system (IFDS) could be designed in-house, using the proposed approach, and then deployed on the field. Personnel could manually tamper the machine to be monitored to gather fault data with the available sensors; then collected data feed the application-agnostic IFDS, which automatically designs a custom neural architecture for classifying faults in real-time. In this way, a single IFDS can satisfy the requirements of multiple applications, as the neural architecture can adapt to the specific set of sensors used to acquire fault data.

II. Related Work

The quest towards implementing deep learning on edge devices started with the inference phase [4]–[6], proceeded with the training phase [7]–[9], and is leaning towards the design phase [10], [11] given the recent results provided by hardware-aware NAS (HW-NAS) [12]–[15].

Garavagno et al. [10] reformulated the optimization problem of HW-NAS by including in the set of constraints the resources available in the device performing the search procedure. Such constraints add to the constraints imposed by the deployment device, which usually characterize the classical versions of HW-NAS; this in turn further reduces the search space. The goal is to avoid the evaluation of architectures not fitting the resources available in the device performing the search procedure; as such, one can enable practical implementation of HW-NAS on consumer devices. In [11], Garavagno et al. further developed the idea of running the search procedure on embedded devices, proposing an adaptive search strategy, which tunes the length of the evaluation process according to the available computational resources.

The objective of [10] and [11] was to automatically design custom neural architectures inside a gateway, one for each type of sensor node connected to it. Such approach preserves privacy, as each device cab exploit the locally acquired data, without involving any third party. Conversely, this paper aims to propose a different approach, where the NAS is expected to run on the deployment device. Another main difference is that the NAS implemented in [10] and [11] produce models meant

to analyze images, while the NAS proposed in the present document produces models meant to analyze time series.

III. Proposed Neural Architecture Search

A. Search Space

The proposed NAS organizes the input time series from different sensors as a bi-dimensional matrix where each row contains the samples belonging to one sensor. Hence obtaining a matrix having $n_{sensors}$ rows and $n_{samples}$ columns. In such a way the resulting matrix can be analyzed by a bi-dimensional convolutional neural network built according to the following rules: the input is a base cell, composed of a bi-dimensional convolutional layer having k kernels, followed by a batch normalization layer and a ReLu activation. Then c cells, composed of a bi-dimensional max pooling layer followed by a bi-dimensional convolutional layer, a batch normalization layer, and a ReLu activation, are stacked upon the base cell. The number of kernels used in each cell is computed using the formula presented in (1). The extracted features are then reduced by a global average pooling layer and finally classified by a fully connected layer, having Softmax as the activation function.

$$n_c = \begin{cases} k & if \quad c = 0 \\ \left\lceil (2 - \sum_{i=1}^{c-1} 2^{-i}) \cdot n_{c-1} \right\rceil & if \quad c \geq 1 \end{cases} \tag{1}$$

To preserve the input matrix height, kernels of $n_{sensors}$ height are used, and stride 1 is always applied to the height dimension. Instead, on the rows dimension, a stride equal to two and a kernel as wide as two samples are always applied in the case of pooling layers, this is done to halve the time series length each time, whereas, in the case of convolution, a stride equal to one is applied alongside a kernel width of three. All the convolutional layers employ zero padding to maintain the input's resolution intact.

B. Optimization Problem

Classical HW-NAS aims to find the best neural architecture for the target hardware in the search space. This leads to a constrained optimization problem where the objective function describes the metric adopted to evaluate candidate solutions and the constraints set the boundaries of the search according to the deployment hardware chosen. Garavagno et al. [10] proposed to include in the constraints the resources available on the hardware performing the search, which usually were considered unbounded. As a result, HW-NAS can run on a resource-constrained device.

In the case of the approach proposed in the present paper, the device performing the search procedure coincides with the device used for deployment. Therefore, the constraints refer to a single target hardware. Indeed, as the goal is to deal with real-time applications (see Sec. I), a constraint on the inference time is also added. The network's input size is considered fixed during the search since it depends on the user's needs. Hence, the search variable $x = (k, c)$ is defined without including the network's input size. This leads to problem formulation P

$$P : \begin{cases} \max f(x) \\ \phi_M(x) \leq \xi_M, \phi_T(x) \leq \xi_T \\ \xi_M, \xi_T > 0 \end{cases} \tag{2}$$

where function f returns the maximum validation accuracy obtained during the evaluation phase, function ϕ_M returns the memory occupancy during training and function ϕ_T returns the time consumed by one inference. Parameters ξ_M and ξ_T respectively represent the upper bounds for the candidate's memory occupancy during training and time per inference.

C. Search Strategy

In the proposed approach the developed NAS runs on constrained devices; thus, there is a need for a lightweight search strategy. Accordingly, the derivative-free search strategy proposed in [16] has been reformulated to solve the optimization problem presented in section III-B. In detail, the phase in which a candidate network is evaluated has been reorganized as a child process. As such, its parent can monitor if the child process has been killed by the operative system because of memory depletion; this in turn means that the candidate network violated the constraint on the available memory. Moreover, the evaluation phase now measures the inference time of the candidate network to determine if such solution satisfies the time constraint.

IV. Experimental Validation

To provide experimental validation of the proposed approach, the developed NAS has been run on three different embedded devices, namely: the Raspberry Pi 4, the Raspberry Pi 3, and the Raspberry Pi Zero 2 W, which are often used in wearable and intelligent systems [17]. Table I provides the hardware specifications for each device.

Embedded Device Raspberry Pi	SoC Broadcom	RAM [GiB]
4 Model B	BCM2711	4
3 Model B	BCM2837	1
Zero 2 W	BCM2710A1	0.5

TABLE I
SPECIFICATIONS OF THE EMBEDDED DEVICES TARGETED BY THIS WORK.

Two datasets have been selected according to the use cases proposed in section I: the Italian Sign Language (ISL) dataset [20] to demonstrate that the proposed approach can be effectively used to automatically design neural architectures for human-machine interfaces, and the Case Western Reserve University (CWRU) dataset [21] to demonstrate that the proposed approach can be also applied to design neural architectures for intelligent fault diagnosis systems.

The ISL dataset [20] contains surface electromyography (sEMG) and inertial measurements regarding the 26 letters of the hand sign Italian alphabet. 8 electrodes of the Myo Gesture Control Armband have been used to acquire the sEMG data at 200 Hz for 2 seconds. 30 samples for each sign have been collected, for a total of 780. To compare the resulting architecture with the one proposed by Pau et al. [18], only

dataset	device	search cost		resulting architecture					reference architecture				
		time [hh]:[mm]	energy [Wh]	(k, c)	RAM [kiB]	Flash [kiB]	test acc. [%]	latency [ms]	work	RAM [kiB]	Flash [kiB]	test acc. [%]	latency [ms]
ISL	RPi 4	8:11	37.2	(16, 4)	81.5	224.5	99.9%	15.3 ± 0.18	[18]	128.5	96.7	93.94	0.6 ± 0.04
	RPi 3	14:42	54.4	(8, 4)	42.5	62.5	92.4%	10.2 ± 0.4					1.2 ± 0.08
	RPi Zero 2	12:03	23.9	(4,0)	17.5	2.7	44.3%	1.2 ± 0.34					1.3 ± 0.3
CWRU	RPi 4	3:42	14.6	(16, 4)	29.5	39.3	99.5%	0.7 ± 0.01	[19]	66.5	163.39	99.3	0.2 ± 0.02
	RPi 3	7:58	27.9	(16, 3)	28.5	24.5	99.2%	1.1 ± 0.04					0.5 ± 0.05
	RPi Zero 2	5:03	10.2	(4, 4)	10	9.45	94.6%	0.4 ± 0.17					0.6 ± 0.23

TABLE II
RESULTING ARCHITECTURES VS REFERENCE ARCHITECTURES

the sEMG data regarding A, C, H, J, K, M, N, P, S, W, X signs were used, for a total of 11 classes and 330 samples. The test split of 20% which brought the best results in terms of accuracy score in [18] has been adopted.

The CWRU dataset [21] contains vibration data for normal and faulty bearings. To compare the resulting architecture with the work proposed by Chen et al. [19], the vibration data from the acceleration sensor at the driving end were selected at 12 kHz sampling frequency. Absence of faults and 9 types of faults, namely ball and inner raceway at 0.007, 0.014, 0.021 inches, and outer raceway oriented at 3, 6, and 12 o'clock at 0.007 inches were selected at 1797 rpm, i.e. at 0 HP. 100 of samples of length 1024 are collected for each class, for a total of 1000 samples.

In both cases, an upper bound of 0.33 seconds, equivalent to 30 inferences per second has been set to the latency, as the constraint of real-time. All models are trained using quantization-aware training and then quantized to 8-bit. Architectures are evaluated for 500 epochs with a learning rate of 10^{-3} and a batch size of 256.

Table II compares the resulting architectures with the state-of-the-art work for each dataset, and also reports the time and the energy needed by the NAS to obtain the result. RAM and Flash occupancies have been measured using X-CUBE-AI's stm32tflm executable, which reports resource usage when using Tensorflow Lite for Microcontrollers as runtime. Whereas, latency is measured by running the model 1000 times on the chosen hardware using Tensorflow Lite runtime.

As can be seen, the resulting architectures' footprint decrease according to the resources available to the embedded device performing the search. The main cause is the decrease in the memory available for training, which further restricts the search space, not allowing the algorithm to explore solutions with larger footprints. The latter decrease also causes a loss in test accuracy. However, it does not cause an increase in the latency: even if the processor's computational power decreases, the footprint reduction overcomes the decrease, providing lower latencies. In any case, the proposed approach provides significantly smaller footprints than the reference work for both datasets, whatever execution platform is considered. Still, the test accuracy provided surpasses the one offered by the reference works in the case of the Raspberry Pi 4 as an execution platform while remaining comparable in the case of the Raspberry Pi 3. Only in the case of Raspberry Pi Zero 2, the accuracy score obtained is drastically lower, especially in the case of the ISL dataset, in which no candidate architecture provides satisfactory results within such a tight memory constraint for training. Even if the total power consumption could be provided by a common laptop battery, the search cost of the proposed approach is high, suggesting a run overnight.

V. CONCLUSION

The present document proposed a new approach to NAS where the search procedure is performed directly on the deployment device. This novel approach enables a new paradigm where an edge device can collect labelled data from human feedback, find a neural architecture tailored to the collected data, and use it to perform inference in real time. The latter paradigm has the potential to enhance continual learning, allowing not only weights but also architectural changes to co-adapt with the data drift.

The proposed approach has been validated on two different datasets: the Italian Sign Language dataset and the Case Western Reserve University dataset; using three different embedded devices: the Raspberry Pi 4 Mobel B, the Raspberry Pi 3 Mobel B, and the Raspberry Pi Zero 2 W; demonstrating a promising capacity of delivering state-of-the-art architectures not only in the domain of biometrical signals for human-machine interfaces, but also in the domain of intelligent fault diagnosis.

The search cost measured during experiments suggests overnight runs. In the latter scenario, the human-machine interface can adapt to the new owner overnight, delivering a tailored neural architecture for the real-time analysis of the user's biometrical signals. However, considering that the amount of data collected from a single user would be less than the one of the datasets adopted as benchmark, the search cost could be lower. For the latter use case, further experiments are needed the asses the real search cost.

ACKNOWLEDGMENT

Project funded under the National Recovery and Resilience Plan (NRRP), Mission 4 Component 2 Investment 1.1 - Call for tender No. 1409 published on Sept 14, 2022 by the Italian Ministry of University and Research (MUR) funded by the European Union – NextGenerationEU - Project Title "LEARN - muLtimodal Edge computing-bAsed weaRable exoskeletoNs for assistance in daily life" – CUP: D53D23016190001, D53D23016200001, J53D23014090001 - Grant Assignment Decree No. 1383 adopted on September 01, 2023 by the Italian Ministry of University and Research (MUR).

REFERENCES

[1] F. Zhou and Y. Chai, "Near-sensor and in-sensor computing," Nature Electronics, vol. 3, no. 11, pp. 664–671, 2020.

[2] R. Hadsell, D. Rao, A. A. Rusu, and R. Pascanu, "Embracing change: Continual learning in deep neural networks," Trends in cognitive sciences, vol. 24, no. 12, pp. 1028–1040, 2020.

[3] J. Ortega-Garcia, J. Bigun, D. Reynolds, and J. Gonzalez-Rodriguez, "Authentication gets personal with biometrics," IEEE signal processing magazine, vol. 21, no. 2, pp. 50–62, 2004.

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

[5] J. Lin, W.-M. Chen, Y. Lin, C. Gan, S. Han et al., "Mcunet: Tiny deep learning on iot devices," Advances in Neural Information Processing Systems, vol. 33, pp. 11 711–11 722, 2020.

[6] J. Lin, W.-M. Chen, H. Cai, C. Gan, and S. Han, "Mcunetv2: Memory-efficient patch-based inference for tiny deep learning," arXiv preprint arXiv:2110.15352, 2021.

[7] J. Lin, L. Zhu, W.-M. Chen, W.-C. Wang, C. Gan, and S. Han, "On-device training under 256kb memory," Advances in Neural Information Processing Systems, vol. 35, pp. 22 941–22 954, 2022.

[8] J. Lee and H.-J. Yoo, "An overview of energy-efficient hardware accelerators for on-device deep-neural-network training," IEEE Open Journal of the Solid-State Circuits Society, vol. 1, pp. 115–128, 2021.

[9] Y. Yang, G. Li, and R. Marculescu, "Efficient on-device training via gradient filtering," in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2023, pp. 3811–3820.

[10] A. M. Garavagno, E. Ragusa, A. Frisoli, and P. Gastaldo, "Running hardware-aware neural architecture search on embedded devices under 512mb of ram," in 2024 IEEE International Conference on Consumer Electronics (ICCE). IEEE, 2024, pp. 1–2.

[11] ——, "An affordable hardware-aware neural architecture search for deploying convolutional neural networks on ultra-low-power computing platforms," IEEE Sensors Letters, 2024.

[12] E. Ragusa, F. Zonzini, P. Gastaldo, and L. De Marchi, "Combining compressed sensing and neural architecture search for sensor-near vibration diagnostics," IEEE Transactions on Industrial Informatics, 2024.

[13] M. Risso, A. Burrello, F. Conti, L. Lamberti, Y. Chen, L. Benini, E. Macii, M. Poncino, and D. J. Pagliari, "Lightweight neural architecture search for temporal convolutional networks at the edge," IEEE Transactions on Computers, vol. 72, no. 3, pp. 744–758, 2022.

[14] J. Rala Cordeiro, A. Raimundo, O. Postolache, and P. Sebastião, "Neural architecture search for 1d cnns—different approaches tests and measurements," Sensors, vol. 21, no. 23, p. 7990, 2021.

[15] L. L. Zhang, Y. Yang, Y. Jiang, W. Zhu, and Y. Liu, "Fast hardware-aware neural architecture search," in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, 2020, pp. 692–693.

[16] A. M. Garavagno, D. Leonardis, and A. Frisoli, "Colabnas: Obtaining lightweight task-specific convolutional neural networks following occam's razor," Future Generation Computer Systems, vol. 152, pp. 152–159, 2024.

[17] S. E. Mathe, H. K. Kondaveeti, S. Vappangi, S. D. Vanambathina, and N. K. Kumaravelu, "A comprehensive review on applications of raspberry pi," Computer Science Review, vol. 52, p. 100636, 2024.

[18] D. P. Pau and M. D. Randriatsimiovalaza, "Electromyography gestures sensing with deeply quantized neural networks," in 2023 IEEE International Conference on Metrology for eXtended Reality, Artificial Intelligence and Neural Engineering (MetroXRAINE). IEEE, 2023, pp. 711–716.

[19] C.-C. Chen, Z. Liu, G. Yang, C.-C. Wu, and Q. Ye, "An improved fault diagnosis using 1d-convolutional neural network model," Electronics, vol. 10, no. 1, p. 59, 2020.

[20] P. Sernani, I. Pacifici, N. Falcionelli, S. Tomassini, A. F. Dragoni et al., "Italian sign language alphabet recognition from surface emg and imu sensors with a deep neural network." in RTA-CSIT, 2021, pp. 74–83.

[21] W. A. Smith and R. B. Randall, "Rolling element bearing diagnostics using the case western reserve university data: A benchmark study," Mechanical systems and signal processing, vol. 64, pp. 100–131, 2015.