A quantitative review of automated neural search and on-device learning for tiny devices

Danilo Pau, Prem Kumar Ambrose
System Research and Applications
STMicroelectronics, Agrate Brianza, Italy
Corresponding author: danilo.pau@st.com

Abstract

This paper presents the state-of-the-art review of the different approaches for Neural Architecture Search targeting resource constrained devices such as microcontrollers. As well as the implementations of On-Device learning techniques for such devices. The approaches like MCUNet have been able to design a tiny neural architecture with low memory and computational requirements which can be deployed on microcontrollers. TinyOL and TML-CD are the state of the solutions for On-device learning to address concept drift and cope with the accuracy drop on real time data.

1 Automated Machine Learning

Automated Machine Learning (AutoML) tools automatically design a learning algorithm and simultaneously set its hyperparameters to optimize its empirical performances on a given dataset that shapes an application problem. AutoML sets a field which helps the ML and embedded C developer experts, who don't have years of knowledge and experiences, to apply ML to their problems with ease being more productive. AutoML focuses on the automation of several problems associated with extraction, transformation & loading of data, training and deployment of the models which need to be deployed on resource un-constrained processors. Optimization of the hyperparameters (HPO) is one of the major focuses of the AutoML. When dealing with tiny devices, resource constraint is a major problem. In most cases resource constraints are not factored into AutoML tools.

1.1 Neural Architecture Search

When a ML algorithm to optimize is an Artificial Neural Network (ANN), AutoML specializes into Neural Architecture Search (NAS). It is therefore a focused subset of AutoML to ANN. NAS aims to find the best architecture with better performance for a neural network. It takes the task carried out by human experts manually adjusting an ANN (topology and associated hyper parameters) and automates this task to find out more complex architectures which performs even better than manually handcrafted networks. It is comprised of a set of tools and methods which explores a large hyper dimensional search space to train, evaluate and test using a certain optimization strategy and select the resulting ANN which performs accurately for the given target by maximizing an objective function. Although NAS seems to be a relatively different field, the under lying problem is similar as that of the hyperparameter optimization. Designing an optimal, accurate and light weight ANN to fit the target devices with limited resources is a problem which is addressed by the research community. NAS can be made to design a light and accurate network automatically by optimizing over a search space of given requirements. NAS can be made to optimize several metrics during the design process like memory requirements, FLOPs, MACCs, latency, inference per second etc.

2 NAS approaches for tiny devices

These solutions are mainly focused on memory constrained edge devices and the search space for the architecture search is optimized to find such architectures.
In Hard Constrained diffeRentiable NAS [1], a continuous probability distribution is induced over the search space, and this makes the search space continuous. This helps creating a sample sub-network using Gumbel SoftMax Trick [2]. The search space is divided into Micro and Macro spaces, where the Micro space is used to control the internal structures of each building blocks of the network. These blocks are the elastic versions of the MBInvRes block [3]. Whereas the Macro search space is used to control how these blocks relate to each other and how these blocks are interconnected.. This solution mainly focuses on the latency. It performed on ImageNet dataset with Top-1 accuracy of 77.3% under a short latency of 32ms.
The Structural wired Neural architecture search for internet of things (MSNet) [4] is graph-based NAS. This focuses on reducing the size of the model to as low as 200KB of peak memory usage and 42M MACCs (multiply and accumulate operations) on Visual Wake Words and a 250KB peak memory usage for ImageNet-1000. On top of that structure level pruning method is used to explore a compact architecture with higher the pruning level to lower the MACCs and the performance also decreases with the increased level of pruning.

Co-Design NAS [5] is a framework which enables the joint exploration of the space of neural architectures, hardware implementation and quantization. It's a combination of Pure software NAS and Hardware aware NAS and the search is made to find a Pareto frontier between hardware efficiency and accuracy. The search process is computationally heavy considering the joint exploration on CIFAR10 with a LUT of 30,000 this method provided a network with an accuracy of 82.98% under 460 Kbits of parameter size even after quantization. This method is more flexible and robust compared to traditional design using fixed architecture.

E-DNAS [6], is a differentiable architecture search method for designed light weight networks. This method finds networks with low latency and better performing DNN which can be deployed on memory constrained devices. The three main ideas behind this approach are a depth aware convolution to compute high resolution feature maps then parallel architecture search pipeline on the feature maps and learns the optimal size and parameters of the convolution kernels. This optimization process is driven by a multi-objective differentiable loss function of accuracy and latency. Lastly to increase the architecture search speed a novel block is used which connects the learned meta kernels during training. The results came out with ImageNet top-1 accuracy of 76.9% with 5.9M parameters and a latency of 38ms.

Unfortunately, this method is not tested on MCU level memory constraints. [7] Proposes a framework which is based on Bayesian Optimization to optimize the hyperparameters of ANN which can be deployed on MCUs as the computational requirements for the optimization of hyperparameters and structure of ANN is much higher than that available from a MCU.

2.1 Approaches for Microcontrollers

AutotinyML [7] proposes a framework, based on Bayesian Optimization (BO), to optimize the hyperparameters of a Convolutional Neural Network by dealing with black box deployable constraints (memory occupation) extracted from STM32Cube.AI tool. It is composed of two different phases. In the first phase, a non-linear SVM classifier is used to approximate the feasible region of the search space associated to hyperparameters values most probably leading to DNNs models deployable on MCU. In the second phase, a BO is focused to the estimated feasible region with the aim to optimize the loss function. Moreover, a probabilistic regression model, specifically a Random Forest is used to approximate the objective function by using the Lower Confidence Bound. Results shown comparable accuracy w.r.t handcrafted baseline with a remarkable reduction of RAM, ROM and MACs.

Solutions	Approach	Target Device	Constraints	Dataset	Acc. %	GPU hours	Tested Applications
HardCoRe-NAS [1]	Differentiable search space + one shot	Edge-GPU/CPU	Latency	ImageNet	78.0	400	Image classificaiton
MSNet[4]	Evolution Search	100-320K SRAM, (256KB-1MB) Flash	Peak memory usage	CIFAR-10	89.09	8	Image classification, VWW
Co-Design NAS[5]	RL (architecture & quantization space)	(0.5-3.5MB) Flash AIoT/ Mobile embedded	Peak memory usage & througput	CIFAR-10	82.93	NA	Image Classification
E-DNAS[6]	Gradient	SoC (ARM cortex-A15)	Low Latency, Memory	ImageNet	76.9	70	Image classification
AutotinyML [7]	Bayesian Optimization	MCU	Memory	User Identification from Walking Activity	92.93	200	Human Activity recognition
MCUNet[8]	One shot + Evolution Search	MCU	Memory (model size + peak memory usage), Latency	ImageNet	70.7	300	Image classification, Object detection, VWW
uNAS[18]	Aging Evolution	MCU	Model size, RAM usage	MNIST	99.19	30	Image classification

Table 1: Overview of different resource constrained NAS approaches and their performances

MCUNet [8] Tiny Deep Learning on IoT Devices is a framework of system-algorithm co-design that jointly optimizes the neural architecture with TinyNAS and the scheduling of the inference with TinyEngine in a same loop. TinyEngine offloads redundant operations from runtime to compile time and only generates the code that will be executed by the TinyNAS which helps in reducing the memory requirements of the inference and allowing more memory for the model size. TinyNAS takes advantage of the memory reduced by TinyEngine and finds a high accuracy model compared to existing frameworks. It is a combination of one-shot NAS giving a super network with all subnetworks. This optimized space of subnetworks undergoes evolution search to find the best architecture. MCUNet achieved a record ImageNet accuracy of 70%+ on a STM32H743 MCU with an SRAM usage of 490KB and

Flash usage of 1.9MB. Large datasets like ImageNet can be used by this method. Supports different memory constraints. Table 1 shows the overview of all the above-mentioned approaches, target devices and their performances.

μNAS [18] is a combination of highly granular search space which almost takes into consideration every aspect of a network such as layer's kernel size, stride, channels, pooling size, fully connected layer's output dimension, connection between each layer, etc. Next the accurate resource use of computation is considered where the peak memory usage, model size, latency are the different aspects driving the loss function. Aging evolution and Bayesian Optimization were the two search algorithms compared. The former tends to perform better than the latter. In addition to that model compressing and pruning were done to reduce the memory requirements of the microcontroller (MCU). The experimental results showed an accuracy of 77.49% on CIFAR-10 with a model size of 685KB, RAM usage of 909B and 41.2K MACs. This performance is better compared to the previous solutions in such highly constrained device like an MCU. It is still time expensive to search for architectures. Improvements can be made to reduce the search time by employing weight sharing to reduce the cost of training each candidate network and by not using the same search space throughout the entire search process by using a parameterized space granularity which can vary through the search space.

3 On Device Learning

This is an increasingly important topic in the ML community since it tries to match the availability of constrained memory and computational power. The environment in which the inference model is deployed is assumed constantly changing (being time varying) and this may cause accuracy drop of ML inferences. This is known as concept drift. Therefore, there is a need for a constant update of the inference model using fresh data to learn from them. On device learning makes shift the process from offline updating the ML model to automatically update with the real time data by ingesting new samples at run time so that the device simultaneously learns and deploys the model with constant adaptation. The accuracy needs to be maintained at higher levels and it shall also reduce the memory footprint to fit MCU hardware assets.

In the next section, this paper reviews and discusses about the different approaches and strategies carried out to implement On-device learning considering the memory footprint be severely constrained.

3.1 On Device Learning on MCUs

On device learning enables resource constrained edge devices to continuously adapt its knowledge, synthetized into its model to new data and fitting the tight memory constraints of such devices. Concept drift is one of the major reasons for constant update of the network. This may result in lower accuracies since the environment data change in real time and if handled with no constant update of the model. To overcome this, the idea of on device learning at the edge helps since it scales, enable better personalization, privacy and set the base for federated learning (FL) [9].

There are several approaches such as [10] introduces intelligent Cyber-Physical Systems (CPSs) and it is able to predict faults with autonomous behavior and self-adaptation on the CPS itself. This method helps in energy efficiency, reducing the bandwidth, autonomy can be achieved on CPSs. [11] proposes a novel solution for online learning and real time anomaly detection of pathological conditions using a low power MCU from ECG signals. The proposed system is based on Reservoir Computing followed by Principal Component Analysis (PCA) and One-Class Support Vector Machine (OC-SVM). This eliminates the need for storing ECG signals for longer periods of time and avoids the time-consuming off-line search of anomalies. [12] proposes a Block based Binary Shallow Echo State Network (BBS-ESN) which is a deeply quantized anomaly detector of oil leaks that happen in the wind turbines with fixed and minimal computational complexity. This network can be deployed on an off-the shelf MCU and the power consumption is greatly reduced. This is achieved by binarization of images and one bit quantization of network's weights and activations. [13] proposes a novel Field oriented Control algorithm by means of extreme learning to modify the behavior and performance of electromechanical systems which are highly nonlinear and needs to adapt itself continuously over time. The Semi Binary Deep Echo State Networks (SB-DESN) proposed achieves a good control accuracy with less complexity. On top of that the paper also proposes a novel complexity optimization which reduces the memory footprint very low, such that it can be deployed even on MCUs. Interesting results were obtained on STM32H7 with an inference time of 20us. [14] proposes a highly accurate, less complex online learning anomaly detection Deep Echo State Network for water distribution systems which adapts to the time varying data distribution and can be deployed on an MCU. In this approach the online learning can be made in two different ways: single iteration and batch decomposition as per the memory constraint of the device. Tiny-Transfer-Learning (TinyTL) [15]proposes freezing of weights and learning the bias modules which eliminates the need for storing the intermediate activations. The results claim to have reduced the memory with little loss in the accuracy compared to fine tuning the full network. Compared to fine tuning only the last layer, TinyTL performs with better accuracy with little memory overhead. The memory can be further saved by feature extraction adaptation without losing the accuracy. Other's state-of the art solutions for On-device learning on MCUs exist.

TinyML with Online learning on MCUs (TinyOL) [16] is implemented in C++ and can be attached to an arbitrary existing network as an additional layer in MCUs. The last additional layer learns from the new data and updates its weights. As the network learns incrementally there is no need for storing the historical data for training reducing the memory requirements. New classes can be added by the last layer and can be trained upon user request. This method is

like transfer learning in which fine tuning happens in the last few layers. Only one data pair of the real time stream is stored in the memory, thus reducing the memory footprint. It is flexible to modify the layer structure on the fly. It has been proven to be robust against concept drift. Yet this approach is limited as the models are trained offline and do not have support for training with 8-bit MCUs. The experimental results on fine tuning a network as well as on classification performed better with the stream of data on a device with less than 256KB SRAM.

The other solution Tiny Machine Learning for Concept Drift (TML-CD) [17] is mainly focused on overcoming the accuracy drop due to concept drift with less memory requirements. This approach is composed of a feature extractor, Dimensionality Reduction operator, kNN Classifier and an adaptation module. The adaptation module adapts the training dataset for a kNN classifier with the new data. The adaptation mechanism is carried out in three different ways: active, passive and hybrid among which the latter has been proved to perform better than the other two. The hybrid adaptation is a combination of both passive and active methods. It continuously adapts over time and at the same time it discards the obsolete knowledge when a change is detected in the stream of data due to concept drift and this sets a bound-on memory footprint. This approach has been tested with MCUs with RAM as low as 96Kb to 512KB and the memory footprint is kept almost constant. It performs a faster recovery when a change is detected. The experimental results showed that the hybrid approach outperforms all the other adaptation mechanisms and recovers faster when there is concept drift. This approach can be further improved by implying learning mechanisms for the feature extractor block and by exploration of Sparse and Quantized solution for the TinyML algorithms.

4 Conclusions

We discussed several approaches of Neural Architecture Search for severely constrained devices to automate the design process of accurate ML architectures with consideration of the technological constrains imposed by the devices and On-device learning to eliminate concept drift and accuracy drop in edge devices. We further discussed on how these approaches can be improved in the future for better performing and robust solutions. Both these research fields are important for the growth of TinyML, a worldwide level community focused on the machine learning ecosystem for mW (and below) power consumption envelope devices, and deployment of Machine Learning on edge devices.

References

[1] N. Nayman, Y. Aflalo, A. Noy and L. Zelnik-Manor, "HardCoRe-NAS: Hard Constrained diffeRentiable Neural Architecture Search," *ICML,* 2021.

[2] E. Jang, S. Gu, and B. Poole, ‘‘Categorical reparameterization with Gumbel-Softmax,’’ in Proc. 5th Int. Conf. Learn. Represent. (ICLR), 2017, pp. 1–12.

[3] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov and L.-C. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," CVPR, 2018, pp. 4510-4520

[4] H.-P. Cheng, T. Zhang, Y. Yang, F. Yan, H. Teague, Y. Chen and H. H. Li, "MSNet: Structural Wired Neural Architecture Search for Internet of Things," *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW),* pp. 2033-2036, 2019.

[5] Lu, Qing & Jiang, Weiwen & Xu, Xiaowei & Shi, Yiyu & Hu, Jingtong. (2019). On Neural Architecture Search for Resource-Constrained Hardware Platforms.

[6] J. G. López, A. Agudo and F. Moreno-Noguer, "E-DNAS: Differentiable Neural Architecture Search for Embedded Systems," *2020 25th International Conference on Pattern Recognition (ICPR),* pp. 4704-4711, 2021.

[7] R. Perego, A. Candelieri, F. Archetti and D. Pau, "Tuning Deep Neural Network’s Hyperparameters Constrained to Deployability on Tiny Systems," ICANN, 2020.

[8] J. Lin, W.-M. Chen, Y. Lin, J. Cohn, C. Gan and S. Han, "MCUNet: Tiny Deep Learning on IoT Devices," NeurIPS 2020, Vancouver, Canada.

[9] Wikipedia contributors, "Federated learning — Wikipedia," 2022.

[10] D. Cogliati, M. Falchetto, D. Pau, M. Roveri and G. Viscardi, "Intelligent Cyber-Physical Systems for Industry 4.0," *2018 First International Conference on Artificial Intelligence for Industries (AI4I),* pp. 19-22, 2018.

[11] N. Abdennadher, D. Pau and A. Bruna, "Fixed complexity tiny reservoir heterogeneous network for on-line ECG learning of anomalies," *2021 IEEE 10th Global Conference on Consumer Electronics (GCCE),* pp. 233-237, 2021.

[12] M. Cardoni, D. P. Pau, L. Falaschetti, C. Turchetti and M. Lattuada, "Online Learning of Oil Leak Anomalies in Wind Turbines with Block-Based Binary Reservoir," *Electronics,* 2021.

[13] N. Federici, D. Pau, N. Adami and S. Benini, "Tiny Reservoir Computing for Extreme Learning of Motor Control," *2021 International Joint Conference on Neural Networks (IJCNN),* pp. 1-8, 2021.

[14] D. Pau, A. Khiari and D. Denaro, "Online learning on tiny micro-controllers for anomaly detection in water distribution systems", IEEE ICCE Berlin 2021.

[15] H. Cai, C. Gan, L. Zhu and S. Han, "TinyTL: Reduce Activations, Not Trainable Parameters for Efficient On-Device Learning," NeurIPS 2020.

[16] H. Ren, D. Anicic and T. A. Runkler, "TinyOL: TinyML with Online-Learning on Microcontrollers," *2021 International Joint Conference on Neural Networks (IJCNN),* pp. 1-8, 2021.

[17] S. Disabato and M. Roveri, "Tiny Machine Learning for Concept Drift," arXiv:2107.14759, 2021.

[18] E. Liberis, L. Dudziak and N. D. Lane, "µNAS: Constrained Neural Architecture Search for Microcontrollers," *Proceedings of the 1st Workshop on Machine Learning and Systems,* 2021.