CNAS: Constrained Neural Architecture Search

Matteo Gambella, Alessandro Falcetta, and Manuel Roveri
Dipartimento di Elettronica, Informazione e Bioingegneria,
Politecnico di Milano, Milano, Italy
Email: {matteo.gambella,alessandro.falcetta,manuel.roveri}@polimi.it

***Abstract*—Neural Architecture Search (NAS) paves the way for the automatic definition of neural networks architectures. The research interest in this field is steadily growing with several solutions available in the literature. This study introduces, for the first time in the literature, a NAS solution, called Constrained NAS (CNAS), able to take into account constraints on the search of the designed neural architecture. Specifically, CNAS is able to consider both functional constraints (i.e., the type of operations that can be carried out in the neural network) and technological constraints (i.e., constraints on the computational and memory demand of the designed neural network). CNAS has been successfully applied to Tiny Machine Learning and Privacy-Preserving Deep Learning with Homomorphic Encryption being two relevant and challenging application scenarios where functional and technological constraints are relevant in the neural network search.**

***Index Terms*—Neural Architecture Search (NAS), Once-For-All Network (OFA), Constrained optimization, Tiny Machine Learning, Homomorphic Encryption.**

I. Introduction

Neural Architecture Search (NAS) [1] is a novel and promising research field whose goal is to automate the definition of neural network architectures for a specific task. Solutions in this field are becoming very popular in recent years due to the widespread diffusion of deep learning models [2]. Indeed, the definition of deep neural network architectures is typically a complex and time-consuming task [3] often requiring a high expertise in the field [3]. Therefore, NAS solutions became relevant assets for experts and practitioners.

From the technical point of view, NAS solutions typically comprise optimization algorithms able to automatically select and combine neural network layers, activation functions, and connections (just to name a few), given a task to be addressed and a corresponding dataset (used to train and validate the neural networks).

The state of the art NAS solutions belong to a family of NAS dubbed One-Shot NAS [4]–[8]. This class relies on the Once-For-All (OFA) [9] supernets (also called one-shot models), which encompass many configurations of networks. The main advantage of using a supernet is the decoupling of search and training, making it computationally feasible to perform the search by training a supernet once and evaluating the subnetworks through inference.

Typically, NAS solutions based on OFAs aim at optimizing the classification accuracy of the designed neural networks [4], [5]. Unfortunately, these solutions cannot take into account other figures of merit such as the computational complexity or the memory demand of the designed neural networks. Recently, the research started focusing on bi-objective NAS solutions (e.g., [6]–[8]) where the network search aims at jointly optimizing accuracy and a figure of merit related to the neural network efficiency (e.g., the number of layers, the number of parameters, etc.).

Interestingly, a promising but mostly left unattended research field concerns NAS with constraints [8] [10]. Adding constraints to NAS paves the way for the automatic design of neural networks that must be executed on technologically-constrained devices (e.g., characterized by constraints on the memory or computational demand) and/or that must comprise only given sets of operations (i.e., only linear or polynomial operations must be considered).

The goal of this study is to develop the *Constrained Neural Architecture Search* (CNAS) that is a novel NAS solution able to take into account constraints on the neural network search. More specifically, CNAS is able to manage *functional* and *technological constraints*. The former refers to constraints on the type of processing layers and operations that are carried out in the designed neural network; the latter consists in constraints on the computational and memory demand of the designed neural network. The proposed CNAS, which is based on MSuNAS [6], comprises an *Once-For-All Approximation* module to manage the functional constraints, and a *Search Strategy module* which uses the NSGA-II [11] genetic algorithm to manage the technological constraints, through an ad-hoc objective function.

The proposed CNAS has been applied to two relevant application scenarios: *TinyML* (Section IV-D) and *Privacy-Preserving Deep Learning with Homomorphic Encryption* (Section IV-E). The former refers to the design of deep learning solutions able to be executed on tiny devices such as Internet-of-Things or Edge computing units [12]. In this scenario, severe technological constraints (on the memory and computational demand) bound the search phase of the designed neural networks. The latter refers to deep learning solutions able to operate on encrypted data, hence guaranteeing the privacy of users [13], thanks to Homomorphic Encryption (HE). Unfortunately, one of the drawbacks of HE is that privacy-preserving deep learning solutions must be redesigned both to encompass only polynomial operations (e.g., nonlinear operations such as ReLU and sigmoid cannot be used with HE) and limit the depth of the neural processing pipeline, due to high memory and computational demand of HE. Hence, in this setting, both functional (i.e., only polynomial operations are allowed) and technological constraints (on memory and

computational demand) are considered.

To sum up, the novel contributions introduced in this study are the following:

- a novel NAS solution able to manage constraints on the search procedure;
- the introduction of functional and technological constraints in NAS;
- a new Once-for-All (OFA) network for Privacy-Preserving Deep Learning with Homomorphic Encryption.

The source code of CNAS, as well as the new OFA for Privacy-Preserving Deep Learning with Homomorphic Encryption, are released to the scientific community as a public repository.[1]

The remainder of this paper is organized as follows. Section II introduces the existing solutions for NAS. Section III presents the problem formulation of constrained NAS. Section IV introduces the CNAS framework developed in this study, while in Section V experiments on the applications of CNAS to the two aforementioned case studies are presented. Conclusions are finally drawn in Section VI.

II. Related literature

The research interest in the field of NAS is steadily growing in recent years and several solutions are available in the literature. This section details and comments on these solutions, highlighting advantages and disadvantages. We emphasize that NAS is an important subfield of AutoML even if the former focuses on the definition of the network architecture, while the latter focuses on the optimization of both the hyperparameters of a given network architecture and the training process. For this reason, this analysis of the related literature focuses only on NAS solutions and, in particular, on One-Shot NAS (i.e., relying on a OFA supernet), being the state of the art in this field.

Block-wisely Self-supervised Neural Architecture Search (BossNAS) [5] is one of the most important NAS solutions in the literature. Its main novelty is the use of a diversified search space consisting of both Convolutional Neural Networks (CNNs) [14] and Transformers [15]. This allows BossNAS to produce neural networks combining CNNs and Transformers able to outperform both pure Transformers and pure CNNs. The BossNAS is a One-Shot NAS paired with block factorization of the search space and a pre-trained teacher model to provide block-wise supervision [16]. The supervision is implemented through an ensemble bootstrapping scheme [17] to ensure unsupervised evaluation. This approach is fundamental in a hybrid scenario where the nature of the teacher may bias the search process. The BossNet-T1+ model, produced by BossNAS, achieves 82.5% of accuracy on ImageNet [18], surpassing Efficient-Net [19] by 2.4%.

Differently, AlphaNet [4] is a NAS introducing a more general divergence figure of merit that extends the Kullback Leibler (KL) one used in the standard student-teacher approach (known as Knowledge Distillation [20]). This enlarges the set of divergence measures by introducing, in the traditional formulation of KL divergence, an hyperparameter that permits to simultaneously penalize both over-estimation and under-estimation of the teacher model uncertainty. The accuracy obtained on ImageNet with AlphaNetA5 is 80.6% of accuracy in its base version, and of 80.4% in the small version.

Neural Architecture Transfer (NAT) [7] is a multi-objective NAS algorithm based on Evolutionary Algorithms. NAT is designed to efficiently generate task-specific custom models that are competitive under multiple conflicting objectives. The main contribution of NAT is the introduction of an accuracy predictor, which is an ensemble of Radial Basis Functions (RBF) [21] models. This allows the search procedure not to validate each candidate architecture on a validation set, saving time and resources. The NAT-M4 model produced by NAT shows a 80.5% accuracy on ImageNet. BossNAS, AlphaNet, and NAT aim at only optimizing the accuracy of the obtained models.

Differently, Evolutionary Multi Objective Surrogate-Assisted NAS (MSuNAS) [6] is a NAS procedure able to adopt a bi-objective optimization for the search of CNNs. In particular, MSuNAS is able to jointly optimize the accuracy of the model (i.e., the final accuracy of the selected optimal architecture) and a secondary objective. The possible secondary objectives are: number of parameters of the network, number of Multiply and Accumulate (MAC) operations, and latency. MSuNAS supports different CNN OFAs, such as MobileNetV3 [22] and ProxylessNAS [23]. Its search procedure explores four important dimensions of CNNs, including depth (number of layers), width (number of channels), kernel sizes, and input resolution. The structure of NAT and MSuNAS are quite similar, being both based on Evolutionary Multi Objective Surrogate-Assisted NAS methods. The main difference regards the accuracy predictor. Indeed, the accuracy predictor in MSuNAS is obtained through a selection mechanism which constructs four types of surrogate models at every iteration and adaptively selects the best model via cross-validation. These four different surrogates are RBF, Multi Layer Perceptron (MLP) [24], Classification And Regression Trees (CART) [25], and Gaussian Processes (GP) [26]. The NSGANetV2-xl model produced by MSuNAS search shows a 80.4% accuracy on ImageNet, but with a slight reduction in terms of computational demand with respect to NAT-M4.

Hard Constrained diffeRentiable Neural Architecture Search (HardCore-NAS) [8] is able to limit the latency of the obtained models, through a differentiable search space trained with the block coordinate stochastic Frank-Wolfe (BC-SFW) [27] algorithm. The HardcoreNAS_E_KD model selected by HardCore-NAS shows a promising result in terms of MACs (i.e., 330M MACs), the lowest value among the state of the art architectures. At the same time, it achieves a 80.1% accuracy on ImageNet.

Constrained Neural Architecture Search For Microcron-

[1]The link to the GitHub repository will be made available in the final version of the paper.

tollers (μNAS) [10], although not belonging to the SOTA architectures on top1 Imagenet accuracy, shows remarkable results in terms of model size retention. Indeed, the model selected by μNAS has 11.4K parameters and 384K MACs proving that NAS can be compliant even with extremely resource-scarce devices like microcontrollers units (MCUs).

We emphasize that, while HardCore-NAS and μNAS focuses only on model size and latency constraints, the CNAS framework proposed in this paper introduces more general figures of merits (see Section IV-B), as well as an entirely different family of constraints (see Section IV-A).

III. Problem formulation

The problem of NAS with constraints consists in selecting a network architecture providing a high accuracy on a given task, while respecting constraints on its architecture or its processing. These constraints may concern the maximum values of some figures of merit (e.g., the maximum number of MACs is limited) of the candidate network architectures, or the type of operations composing the neural network (e.g., the division operation is not allowed). Formally, the NAS with constraints can be defined as the following constrained optimization problem:

$$\begin{aligned} &\text{minimize } \mathcal{G}\left(\mathcal{S}(\widetilde{x}), F_1(\widetilde{x}), \ldots, F_N(\widetilde{x})\right) \\ &\text{s. t. } F_i(\widetilde{x}) < \overline{F}_i,\ i = 1, \ldots, N \\ &\widetilde{x} \in \Omega_{\widetilde{x}} \end{aligned} \tag{1}$$

where $\mathcal{G}$ is a multi-objective optimization function, $\widetilde{x}$ and $\Omega_{\widetilde{x}}$ represent a candidate neural network architecture and the search space of the search, respectively, $\mathcal{S}(\widetilde{x})$ is the classification accuracy of $\widetilde{x}$, $F_i(\widetilde{x})$ and $\overline{F}_i$ with $i = 1, \ldots, N$ account for the i-th figures of merit and the corresponding constraint, respectively, and N is the number of constraints. The optimization problem described in Eq. 1 will be tackled by the proposed CNAS framework that is introduced in the next Section.

IV. The proposed Constrained NAS

This study introduces the proposed CNAS, which is a NAS solution able to take into account constraints in the search of neural network architectures.

An overall description of the CNAS framework is given in Fig. 1. In detail, CNAS receives as input a dataset $\mathcal{DS}$, comprising a training part (used to train the candidate networks) and a validation part (used to validate the candidate networks), an OFA set Ω_x from which the candidate neural networks are obtained, and the two classes of constraints which can be considered in the search procedure, namely the functional constraints $\mathcal{FC}$ (detailed in Section IV-A) and the technological constraints $\mathcal{TC}$ (detailed in Section IV-B). Moreover, a vector of weights $W = [w_P, w_A, w_M]$, used to balance the different search objectives, can be specified: the role of these weights is detailed in Section IV-C.

At the end of the search, CNAS returns the set of the k optimal network architectures $X^\circ = \{x_1^\circ, \ldots, x_k^\circ\}$. These network architectures are guaranteed to respect both the functional and technological constraints. The value of k can be specified by the user of CNAS.

A. Functional constraints $\mathcal{FC}$

The functional constraints $\mathcal{FC}$ allow the CNAS to exclude from the neural network search the layers and activation functions which include given operations. In other words, if there is a functional constraint on a layer/activation function τ, CNAS guarantees that τ is never used in the obtained optimal network architectures $X^\circ = \{x_1^\circ, \ldots, x_k^\circ\}$. This is a crucial ability, for example, in the field of Privacy-Preserving Deep Learning with Homomorphic Encryption (PPDL-HE) where only additions and multiplications are allowed in the processing. This particular scenario is presented in Section IV-E.

B. Technological constraints $\mathcal{TC}$

Considering technological constraints in the search process is useful when the designed neural network has to be deployed on a device with a limited amount of memory and computational resources. In this study, the following three technological constraints $\mathcal{TC}$ have been defined:

- $F_P(\widetilde{x}) < \overline{F}_P$, where $F_P(\widetilde{x})$ accounts for the total number of weights present in $\widetilde{x}$ and $\overline{F}_P$ is the corresponding technological constraint;
- $F_A(\widetilde{x}) < \overline{F}_A$, where $F_A(\widetilde{x})$ accounts for the size of the activations (i.e., the number of floating point values composing the feature maps) present in $\widetilde{x}$ and $\overline{F}_A$ is the corresponding technological constraint;
- $F_M(\widetilde{x}) < \overline{F}_M$, where $F_M(\widetilde{x})$ accounts for the total number of MAC operations[2] in $\widetilde{x}$ and $\overline{F}_M$ is the corresponding technological constraint.

We emphasize that $F_P(\widetilde{x})$ and $F_M(\widetilde{x})$ are available in MSuNAS only as secondary objectives for the search and not as constraints to be considered during the search. In addition, $F_A(\widetilde{x})$ has been introduced in CNAS and represents a novel outcome of this study. Remarkably, $F_P(\widetilde{x})$ and $F_A(\widetilde{x})$ account for the total memory demand of $\widetilde{x}$, while $F_M(\widetilde{x})$ accounts for its computational demand.

C. The CNAS framework

In CNAS, the constrained optimization problem described in Eq.1 has been reformulated in its non-constrained version as follows:

$$\begin{aligned} &\text{minimize } \mathcal{G}\left(\mathcal{S}_f(\widetilde{x}), \Phi_{CNAS}(\widetilde{x})\right) \\ &\text{s. t. } \widetilde{x} \in \Omega_{\widetilde{x}} \end{aligned} \tag{2}$$

where $\mathcal{S}_f$ is the predictor of the classification accuracy of $\widetilde{x}$ introduced in MSuNAS and $\Phi_{CNAS}(\widetilde{x})$ is defined as follows:

$$\begin{aligned} \Phi_{CNAS}(\widetilde{x}) = &w_P \cdot \left(F_P(\widetilde{x}) + \alpha \cdot \max\left(0, \left(F_P(\widetilde{x}) - \overline{F}_P\right)\right)\right) + \\ &w_A \cdot \left(F_A(\widetilde{x}) + \alpha \cdot \max\left(0, \left(F_A(\widetilde{x}) - \overline{F}_A\right)\right)\right) + \\ &w_M \cdot \left(F_M(\widetilde{x}) + \alpha \cdot \max\left(0, \left(F_M(\widetilde{x}) - \overline{F}_M\right)\right)\right) \end{aligned}$$

[2]We can approximately state that $1MAC \simeq 2FLOPS$.

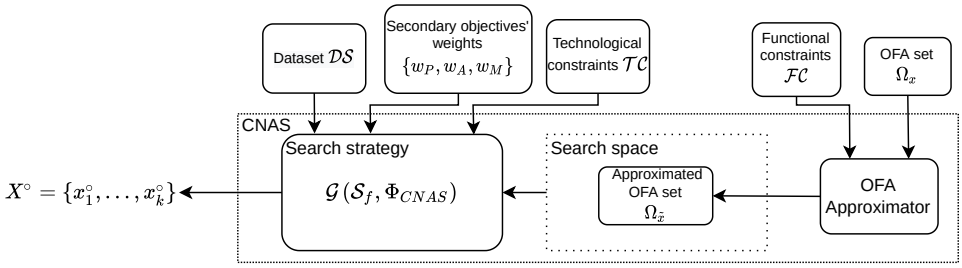

Fig. 1. The proposed CNAS framework, composed of an OFA Approximator and a Search Strategy module.

where α is a *penalty* factor, which is a constant positive term. Its role is to assign a high value to $\Phi_{CNAS}(\widetilde{x})$ when a technological constraint is violated in $\widetilde{x}$. The weights $\{w_P, w_A, w_M\}$ are used to optimize the different figures of merit. By default, $\{w_P, w_A, w_M\} = \{1.0, 1.0, 1.0\}$ which means that all the figures of merit are equally important.[3]

The functional constraints $\mathcal{FC}$ are managed in CNAS by means of the *OFA Approximator* module, which enforces them on the OFA set Ω_x received in input. In particular, the OFA Approximator implements a "search-and-replace" mechanism on the code definition of each OFA in Ω_x. Formally, let $\Omega_x = \{\Theta_1, \ldots, \Theta_N\}$ be the set of OFAs which constitute the search space of the NAS, and let τ be a layer or activation function that is forbidden in Ω_x, representing a functional constraint (i.e., τ can not be used in the network candidates). The *OFA approximation* modifies Ω_x into $\Omega_{\widetilde{x}} = \{\widetilde{\Theta}_1, \ldots, \widetilde{\Theta}_N\}$, where:

$$\widetilde{\Theta}_k = \begin{cases} \Theta_k & \text{if } \Theta_k \text{ does not use } \tau \\ \Theta_k^{\tau,\widetilde{\tau}} & \text{otherwise} \end{cases} \tag{3}$$

for k in $\{1, \ldots, N\}$ where $\Theta_k^{\tau,\widetilde{\tau}}$ refers to Θ_k where the layer τ is replaced with the layer $\widetilde{\tau}$ (i.e., $\Theta_k^{\tau,\widetilde{\tau}}$ does not use τ). The rules used to replace τ with $\widetilde{\tau}$ are specified by the user. The module returns the Approximated OFA set $\Omega_{\widetilde{x}}$, which is the search space of CNAS.

The core of CNAS is the *Search Strategy* module. It uses the genetic algorithm NSGA-II [11] to solve the bi-objective problem introduced in Eq. 2, by jointly optimizing the objectives $\mathcal{S}_f(\widetilde{x})$ and $\Phi_{CNAS}(\widetilde{x})$ on the dataset $\mathcal{DS}$, evaluating network architectures candidates from $\Omega_{\widetilde{x}}$. The search process is iterative: at each iteration the surrogate accuracy predictor $\mathcal{S}_f$ is chosen with a mechanism called "adaptive-switching", the same method adopted by MSUNAS [6], which selects the best surrogate model according to a correlation metrics (Kendall' s Tau [28]). Then, the candidates are ranked according to their predicted accuracy, and a new set of candidates—to be evaluated in the next round—is obtained by NSGA-II.

D. The TinyML scenario

One of the most representative application scenarios to show the usefulness of CNAS is the one of TinyML. This is because TinyML typically requires the deployment of deep learning models on severely-constrained hardware, which ranges from IoT-devices (e.g., Arduino [29]) to mini-PCs (e.g., Raspberry [30]). The objective function presented in Eq. 2 is well suited to be used in this case. As anticipated in Section IV-A, $F_P(\widetilde{x})$, $F_A(\widetilde{x})$, and $F_M(\widetilde{x})$ are related to the technological aspects of the considered device. In fact, it is possible to estimate the values of $\overline{F}_P$ and $\overline{F}_A$ from the available memory on the considered device. On the other hand, $\overline{F}_M$ can be estimated given the CPU clock of the considered device. In Section V, the experimental campaign will explore technological constraints introduced due to the need to design neural networks to be used on a Raspberry device. The details about the corresponding $\mathcal{TC}$s are given in Section V-B.

E. Privacy-Preserving DL with HE

PPDL-HE is an interesting example of application of CNAS with functional constraints. In fact, HE schemes represent a particular type of encryption schemes which support the computation of a restricted set of operations directly on encrypted data [31]. This is possible because the algebraic structure of the plain data is maintained also when the data are encrypted. However, many HE schemes support only additions and multiplications. For these reasons, PPDL-HE is particularly relevant in CNAS from two different perspectives.

First, using NAS to design network architectures compatible with HE requires the introduction of functional constraints. In particular, all the layers and blocks containing non-polynomial operations (e.g., ReLU and maximum pooling) should not be considered. In this scenario, for instance, a single OFA, based on ResNet [32], is considered. Formally, $\Omega_x = \{\Theta_{ResNet}\}$ and $\tau = ReLU$.

Given that Θ_{ResNet} contains τ, it should be approximated. A polynomial approximation of ReLU [33] is used. Formally:

$$\widetilde{\tau}(x) = 0.01x^2 + 0.5x.$$

As a result, $\widetilde{\Theta}_{ResNet} = \Theta_{ResNet}^{\tau,\widetilde{\tau}}$, which is the Θ_{ResNet} architecture with ReLU activations replaced with the polynomial presented above. The approximated OFA set $\Omega_{\widetilde{x}} = \{\widetilde{\Theta}_{ResNet}\}$ does not contain τ (i.e., it is HE-compliant).

Second, HE introduces an important overhead in terms of memory and time for the computation of operations. Similarly to the TinyML scenario presented in Section IV-D, technological constraints can be used to tackle this problem.

[3]The figures of merit are normalized to take into account their different orders of magnitude.

Usually, PPDL-HE models are run on the Cloud where, even if a high amount of resources is available, the costs may be important. This justifies the use of technological constraints also in this scenario. Further details about the $\mathcal{TC}$s are given in Section V-C.

It is stressed that the CNAS is used on plain, unencrypted data: the details on the deployment of the obtained network architectures on encrypted data are outside of the scope of this work (the reader can refer to [13] for details).

V. Experimental results

This section details the results of the experimental campaign. Here, the goal is to show that CNAS is able to design network architectures that are effective (i.e., providing a high accuracy for the considered task) and that are able to satisfy functional and technological constraints. In all the experiments, the weights w_P, w_M, and w_A are set to $[1.0, 1.0, 1.0]$, while the penalty factor is set to infinite.

A. Dataset

The dataset considered in this analysis is the CIFAR-10 [34], which comprises 32x32 color images representing 10 classes of objects. The training set comprises 60000 images, while the testing set comprises 10000 images.

B. The TinyML scenario

In the TinyML scenario, the goal is to identify the best network architecture which can be deployed on a device with severe technological constraints. As reference, the Raspberry Pi 3 Model B+ [35], which is endowed with 1GB RAM memory and a 1.4GHz CPU, is used. The technological constraints $\mathcal{TC}$ on the number of parameters, the number of MACs and the size of activations correspond to the 1% of the computational resources available on the Raspberry Pi 3 Model B+, and they are consequently defined as follows:

- $\overline{F}_P = 2.2M$;
- $\overline{F}_M = 7M$;
- $\overline{F}_A = 0.3M$.

The OFA considered in this experimental scenario is based on the MobileNetV3 [22], while the following deep learning models are used for comparison: MSuNAS [6], μNAS [10], MUXNET [36], HCGNet-A1 [37], CCT-2/3x2 [38] and CCT-7/3x1 [38]. MSuNAS is the NAS solution described in Section II, whereas the remaining models[4] are detailed in the Image Classification ranking on CIFAR-10 presented in [39].

Experimental results are presented in Table I. Some comments can be made. First, as expected, CNAS is able to design a neural network satisfying the technological constraints $\mathcal{TC}$. Differently, MSuNAS identifies a neural network characterized by a higher classification accuracy, but with values of $F_P(\widetilde{x})$, $F_M(\widetilde{x})$ and $F_A(\widetilde{x})$ far beyond $\mathcal{TC}$. Similarly, the manually-designed models (i.e., HCGNET-A1, CCT-2/3x2, and CCT-7/3x1), being designed to optimize only the accuracy, do not satisfy the constraints $\mathcal{TC}$. Interestingly, CCT-2/3x2, which satisfies the constraint on $F_P(\widetilde{x})$ but not the ones on $F_M(\widetilde{x})$ and $F_A(\widetilde{x})$, provides an accuracy that is lower than CNAS. Lastly, μNAS outperforms CNAS in terms of model size retention but achieves lower top1 accuracy.

[4]We considered models with comparable values of number of parameters and number of MACs.

TABLE I
Results for TinyML scenario. The column NAS specifies if the network is designed by a NAS procedure (yes) or not (no).

Model	NAS	Params (P)	MACs (M)	Act. Sizes (A)	Accuracy
$\mathcal{TC}$		2.2M	7M	0.3M	
CNAS (our)	yes	2.14M	6.85M	**0.23M**	89.9%
MSUNAS	yes	2.53M	194.16M	8.2M	97.75%
μNAS	yes	**11.4K**	**384K**	-	86.49%
MuxNet	yes	2.1M	200M	-	**98.0%**
HCGNet-A1	no	1.1M	100M	-	96.9%
CCT-2/3x2	no	0.28M	30M	-	89.2%
CCT-7/3x1	no	3.76M	950M	-	**98.0%**

C. The Privacy-Preserving Deep Learning scenario

In this scenario, the goal is to identify a neural network architecture which provides a high accuracy for the classification of encrypted CIFAR-10 images. The functional constraint $\mathcal{FC}$ forces the neural network to comprise only polynomial operations. In addition, the following technological constraints $\mathcal{TC}$, selected in order to obtain a model with less parameters than the one used in the TinyML case, have been introduced:

- $\overline{F}_P = 0.5M$;
- $\overline{F}_M = 150M$;
- $\overline{F}_A = 5M$.

As a comparison, we considered the privacy-preserving version [13] of the well known LeNet-5 neural network and a simple neural network with a single Linear Layer (called SingleLayerNN). Experimental results are shown in Table II. The main comment is that CNAS is able to provide a network architecture that respects the functional constraint $\mathcal{FC}$, meanwhile guaranteeing an accuracy far higher than the one of LeNet-5 and SingleLayerNN. It is also worth noting that the neural network architecture also satisfies the technological constraints $\mathcal{TC}$. A more thorough analysis of this scenario can be found in [13].

VI. Conclusions

This work proposed Constrained Neural Architecture Search (CNAS), an extension of NAS that allows one to specify

TABLE II
Results for PPDL-HE scenario. The column NAS specifies if the network is designed by a NAS procedure (yes) or not (no).

Model	NAS	Params	MACs	Act. Sizes	Accuracy
$\mathcal{TC}$		0.5M	150M	5M	
CNAS (our)	yes	0.49M	137.51M	1.19M	**74.7%**
LeNet-5	no	0.14M	1.5M	0.018M	68.0%
SingleLayerNN	no	**0.05M**	**0.05M**	**0.005M**	41.0%

constraints on the candidate neural network architectures. The constraints can either be on the computational and memory demand of the networks, or on the presence of particular operations or layers. The software implementation, built on top of a state of the art NAS, is released to the scientific community as a public repository. Lastly, experiments are provided on two real use-cases to prove the efficiency and effectiveness of the solution. Future works will encompass the introduction of new OFAs able to extend the range of application of CNAS, new types of constraints, different multi-objective optimization algorithms, more application scenarios, and the combination of CNAS with other AutoML techniques.

ACKNOWLEDGMENT

This work has been partially funded by Luxottica Group SpA and by Dhiria Srl for the application of CNAS to TinyML and to Privacy-preserving deep learning, respectively.

REFERENCES

[1] P. Ren, Y. Xiao, X. Chang, P.-Y. Huang, Z. Li, X. Chen, and X. Wang, "A comprehensive survey of neural architecture search: Challenges and solutions," *arXiv preprint arXiv:2006.02903*, 2020.

[2] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–44, 05 2015.

[3] X. He, K. Zhao, and X. Chu, "Automl: A survey of the state-of-the-art," *Knowledge-Based Systems*, vol. 212, p. 106622, 2021.

[4] D. Wang, C. Gong, M. Li, Q. Liu, and V. Chandra, "Alphanet: Improved training of supernets with alpha-divergence," in *International Conference on Machine Learning*. PMLR, 2021, pp. 10 760–10 771.

[5] C. Li, T. Tang, G. Wang, J. Peng, B. Wang, X. Liang, and X. Chang, "Bossnas: Exploring hybrid cnn-transformers with block-wisely self-supervised neural architecture search," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 12 281–12 291.

[6] Z. Lu, K. Deb, E. Goodman, W. Banzhaf, and V. N. Boddeti, "Nsganetv2: Evolutionary multi-objective surrogate-assisted neural architecture search," in *European Conference on Computer Vision*. Springer, 2020, pp. 35–51.

[7] Z. Lu, G. Sreekumar, E. Goodman, W. Banzhaf, K. Deb, and V. N. Boddeti, "Neural architecture transf er," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 9, pp. 2971–2989, 2021.

[8] N. Nayman, Y. Aflalo, A. Noy, and L. Zelnik-Manor, "Hardcore-nas: Hard constrained differentiable neural architecture search," in *ICML*, 2021.

[9] H. Cai, C. Gan, T. Wang, Z. Zhang, and S. Han, "Once for all: Train one network and specialize it for efficient deployment," in *International Conference on Learning Representations*, 2020. [Online]. Available: https://arxiv.org/pdf/1908.09791.pdf

[10] E. Liberis, Ł. Dudziak, and N. D. Lane, "μnas: Constrained neural architecture search for microcontrollers," in *Proceedings of the 1st Workshop on Machine Learning and Systems*, 2021, pp. 70–79.

[11] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," *IEEE transactions on evolutionary computation*, vol. 6, no. 2, pp. 182–197, 2002.

[12] C. Alippi and M. Roveri, "The (not) far-away path to smart cyber-physical systems: An information-centric framework," *Computer*, vol. 50, no. 4, pp. 38–47, 2017.

[13] A. Falcetta and M. Roveri, "Privacy-preserving deep learning with homomorphic encryption: An introduction," *IEEE Computational Intelligence Magazine*, vol. 17, no. 3, 2022.

[14] S. Albawi, T. A. Mohammed, and S. Al-Zawi, "Understanding of a convolutional neural network," in *2017 International Conference on Engineering and Technology (ICET)*, 2017, pp. 1–6.

[15] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[16] C. Li, J. Peng, L. Yuan, G. Wang, X. Liang, L. Lin, and X. Chang, "Block-wisely supervised neural architecture search with knowledge distillation," *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1986–1995, 2020.

[17] Z. Shen, Z. He, and X. Xue, "Meal: Multi-model ensemble via adversarial learning," *ArXiv*, vol. abs/1812.02425, 2019.

[18] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.

[19] M. Tan and Q. V. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," *ArXiv*, vol. abs/1905.11946, 2019.

[20] S. Abbasi, M. Hajabdollahi, N. Karimi, and S. Samavi, "Modeling teacher-student techniques in deep neural networks for knowledge distillation," in *2020 International Conference on Machine Vision and Image Processing (MVIP)*. IEEE, 2020, pp. 1–6.

[21] B. Baker, O. Gupta, R. Raskar, and N. Naik, "Accelerating neural architecture search using performance prediction," *arXiv preprint arXiv:1705.10823*, 2017.

[22] A. G. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan, Q. V. Le, and H. Adam, "Searching for mobilenetv3," *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 1314–1324, 2019.

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

[24] C. Liu, B. Zoph, M. Neumann, J. Shlens, W. Hua, L.-J. Li, L. Fei-Fei, A. Yuille, J. Huang, and K. Murphy, "Progressive neural architecture search," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.

[25] Y. Sun, H. Wang, B. Xue, Y. Jin, G. G. Yen, and M. Zhang, "Surrogate-assisted evolutionary deep learning using an end-to-end random forest-based performance predictor," *IEEE Transactions on Evolutionary Computation*, vol. 24, no. 2, pp. 350–364, 2020.

[26] X. Dai, P. Zhang, B. Wu, H. Yin, F. Sun, Y. Wang, M. Dukhan, Y. Hu, Y. Wu, Y. Jia *et al.*, "Chamnet: Towards efficient network design through platform-aware model adaptation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 11 398–11 407.

[27] S. Lacoste-Julien, M. Jaggi, M. Schmidt, and P. Pletscher, "Block-coordinate Frank-Wolfe optimization for structural SVMs," in *Proceedings of the 30th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, S. Dasgupta and D. McAllester, Eds., vol. 28, no. 1. Atlanta, Georgia, USA: PMLR, 17–19 Jun 2013, pp. 53–61.

[28] M. G. Kendall, "A new measure of rank correlation," *Biometrika*, vol. 30, no. 1/2, pp. 81–93, 1938. [Online]. Available: http://www.jstor.org/stable/2332226

[29] "Arduino," https://www.arduino.cc/, accessed: 2022-03-26.

[30] "Raspberry," https://www.raspberrypi.com/, accessed: 2022-03-26.

[31] F. Boemer, Y. Lao, R. Cammarota, and C. Wierzynski, "ngraph-he: a graph compiler for deep learning on homomorphically encrypted data," in *Proceedings of the 16th ACM International Conference on Computing Frontiers*, 2019, pp. 3–13.

[32] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[33] R. Podschwadt, D. Takabi, and P. Hu, "Sok: Privacy-preserving deep learning with homomorphic encryption," *ArXiv*, vol. abs/2112.12855, 2021.

[34] A. Krizhevsky, "Learning multiple layers of features from tiny images," 2009.

[35] "Raspberry pi 3 model b+," https://www.raspberrypi.com/products/raspberry-pi-3-model-b-plus/, accessed: 2022-03-26.

[36] Z. Lu, K. Deb, and V. N. Boddeti, "Muxconv: Information multiplexing in convolutional neural networks," *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 12 041–12 050, 2020.

[37] C. Yang, Z. An, H. Zhu, X. Hu, K. Xu, C. Li, B. Diao, and Y. Xu, "Gated convolutional networks with hybrid connectivity for image classification," in *AAAI*, 2020.

[38] A. Hassani, S. Walton, N. Shah, A. Abuduweili, J. Li, and H. Shi, "Escaping the big data paradigm with compact transformers," *ArXiv*, vol. abs/2104.05704, 2021.

[39] "Nas-imagenet," https://paperswithcode.com/sota/neural-architecture-search-on-imagenet, accessed: 2022-02-03.