EDANAS: Adaptive Neural Architecture Search for Early Exit Neural Networks

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

***Abstract*—Early Exit Neural Networks (EENNs) endow neural network architectures with auxiliary classifiers to progressively process the input and make decisions at intermediate points of the network. This leads to significant benefits in terms of effectiveness and efficiency such as the reduction of the average inference time as well as the mitigation of overfitting and vanishing gradient phenomena. Currently, the design of EENNs, which is a very complex and time-consuming task, is carried out manually by experts. This is where Neural Architecture Search (NAS) comes into play by automatically designing neural network architectures focusing also on the optimization of the computational demand of these networks. These requirements are crucial in the design of machine and deep learning solutions meant to operate in devices constrained by the technology (computation, memory, and energy) such as Internet-Of-Things and embedded systems. Interestingly, few NAS solutions have taken into account the design of early exiting mechanisms. This work introduces, for the first time in the literature, a framework called Early exit aDAptive Neural Architecture Search (EDANAS) for the automatic design of both the EENN architecture and the parameters that manage its early exit mechanism in order to optimize both the accuracy in the classification tasks and the computational demand. EDANAS has proven to compete with expert-designed early exit solutions paving the way for a new era in the prominent field of NAS.**

***Index Terms*—Neural Architecture Search (NAS), Once-For-All Network (OFA), Tiny Machine Learning, Early Exit Neural Networks, Adaptive Neural Networks.**

I. Introduction

Nowadays, deep neural networks [1] are widely used in many different application scenarios. In this field, Convolutional Neural Networks (CNNs) [2] play a major role both in image classification, and recognition as well as in other forecasting [3] and detection tasks. However, deep CNN models generally suffer from many drawbacks. Firstly, even applying strong regularization, overfitting [4], and vanishing gradient [5] [6] phenomena can still occur. Secondly, deep CNN models can be affected by overthinking [7] [8], a phenomenon that points out that predictions that would be correct with smaller architectures become incorrect with deeper architectures. In order to address these problems, Early Exit Neural Networks (EENNs) have been recently introduced in the literature [9]. This novel family of neural networks is characterized by multiple early exits, called Early Exit classifiers (EECs), placed at different points of the backbone architecture to provide intermediate predictions when enough confidence in the classification is achieved [6]. This decision is taken by the so-called "selection scheme" which is a set of decision functions, one per EEC. Interestingly, Inception [10] architecture exploits these auxiliary classifiers to counteract the aforementioned drawbacks. Another fundamental point supporting EENNs is that, as shown in many works [11] [5] [12], the majority of input samples could be classified by resorting to smaller architectures [13] even in very complex datasets such as ImageNet [14]. Indeed, the majority of input samples are classified by the EENNs in earlier stages of the neural network by the EECs resulting in a significant reduction of the average inference time [9]. We emphasize that currently, the design of EENNs is still carried out manually by the neural network designers. Such a design phase is a time-consuming task and requires high expertise in this field.

The need to have automatic tools for the design of neural networks has been recently addressed in the literature by Neural Architecture Search (NAS) [15]. This technique aims at automating the design of a neural network exploring different architectural configurations given a neural network topology, a task to be accomplished, and a dataset used for training and validation to provide the optimal architecture. The majority of works in this field focus on the design of these architectures while taking into account the target hardware, hence belonging to a family of NAS solutions named Hardware-Aware NAS [16] [17]. While most of these solutions exploit model compression techniques such as pruning and quantization [18] [19], interestingly, there are very few examples of NAS for adaptive neural networks in the current literature [20] [21] [22] [23] [24]. In particular, the current literature is still lacking NAS solutions for jointly designing the EENN architecture and the parameters that manage its early exit mechanism.

The goal of this study is to develop *Early exit aDAptive Neural Architecture Search* (EDANAS), which is a novel NAS solution able to optimize jointly the designed architecture of an EENN and the selection scheme that progressively selects the EEC that has enough confidence in the classification in the input. The proposed EDANAS, based on CNAS [17], handles the selection scheme by introducing a new dimension to the existing exploration directions (kernel size, depth, expansion rate, and resolution). This new dimension expresses the thresholds used in the EECs to evaluate the confidence and the placement of the EECs. In addition, the proposed EDANAS jointly optimizes the classification accuracy and a

novel figure of merit, called ADA_MACS, accounting for the number of Multiply-and-Accumulates (MACs) of the inference of a candidate EENN. ADA_MACS is defined as the sum of the MACs of each early exit weighted by the utilization (i.e., the percentage of input samples exiting at that EEC) of the corresponding exit. In addition, EDANAS introduces a MACs predictor surrogate model to estimate the ADA_MACS of an EENN during the NAS search.

The proposed EDANAS has been shown to be competitive with EENNs designed by experts in the field in terms of accuracy and computational demands suggesting the validity of this research direction.

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

- a novel NAS solution able to co-design the network architecture and the selection scheme in EENNs;
- the automatic placement of EECs in the candidate backbone of an EENN network;
- a novel figure of merit accounting for the average number of MACs of an EENN and a predictor surrogate model to estimate this figure of merit during the NAS exploration.

The source code of EDANAS is 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 EENNs and NAS. Section III presents the problem formulation of EDANAS. Section IV introduces the EDANAS framework developed in this study, while in Section V the experimental campaign aiming at evaluating the effectiveness of EDANAS is proposed. Conclusions are finally drawn in Section VI.

II. Related literature

A. Early Exit Neural Networks

In this section, the main families of EENN solutions present in the literature are discussed. These solutions can be categorized according to the techniques used for training, inference, and optimization of the selection scheme performed [6].

As regard the training of EENNs we can classify the possible training approaches as follows:

1) *Joint Training* (JT): jointly training all EECs by combining the losses of the classifiers [9]. A variant of JT concerns the computation of a single loss on top of the combination of all exit outputs [25].
2) *Layer-wise Training* (LT) aims at separately optimizing each layer in the NN architecture. At each iteration, a single EEC is trained jointly with the layers of the backbone preceding it. This approach is used in Deep Cascade Learning [12].
3) *Knowledge Distillation (KD)*: the backbone network is trained separately from each of the EECs. In this approach, the backbone network is initially trained. Then, the EECs on top of it are trained by relying on KD [26], i.e., from the backbone (teacher) to the EECs (students).

[1]https://github.com/matteogambella/NAS

Differently, the inference in EENNs can be performed in two ways:

- the most trivial approach is to aggregate the outputs of all the EECs so as to provide a joint prediction [25];
- the most interesting approach is to process the input up to a given EEC and, then, stop the forward propagation. This scheme shows to be effective in reducing inference time, overfitting, and overthinking [9].

Another key point regards the optimization of the selection scheme. For classification problems, a common choice is to measure the level of confidence of the neural network on its own prediction and exploit it to decide whether to early exit the processed input. A popular approach to trigger the early exit is to compute the score margin (SM) and compare it to a threshold to evaluate whether enough confidence has been achieved [27] [20] as follows:

$$SM = 1^{st}score–2^{nd}score \tag{1}$$

where 1^{st} and 2^{nd} score represents the largest and the second largest score value at the classifier layer, respectively. In this setup, a prediction computed by the n-th EEC is considered to be enough confident if SM_n, i.e., the SM computed by the n-th EEC, is greater or equal than a threshold θ_n. A specific θ_n, with $n = 1, .., N$, is selected for each EEC. All in all, the decision functions of the *selection scheme* rely on the set of hyperparameters θ = { θ_1, θ_2 ... θ_N }, with N the number of EECs. The θ_n values can be user-defined or estimated by using approaches such as *static thresholding* or *dynamic thresholding adjustment* [27].

In this field, BranchyNet [9] is the first example of EENN. The training is performed in a JT approach by computing the weighted sum of the loss functions of each exit branch. In the inference phase, when an EEC has enough confidence about the classification of an input sample, the decision is made and no further computation is performed by the network.

The solution suggested in [28] relies on a training phase combining the outputs of all the exits in a recursive manner and replacing each EEC with a soft conditional version. The decision functions $g_n(x)$, with n are binary functions implementing the logic to decide whether to early exit or not. The formulation of the soft conditional output $\widetilde{y_n}$ of an EENN x at layer n is the following:

$$\widetilde{y_n} = g_n(x)y_n(x) + (1 - g_n(x))\widetilde{y_n} \tag{2}$$

where $g_n(x)$ and $y_n(x)$ are the decision function and the output of the n-th exit respectively. The training of the neural network is performed on top of a single loss function.

More recent works propose advanced solutions for the optimization of thresholds. For example, EENet [29] proposes a framework, leveraging multi-objective learning, to optimize both the allocation of input samples among the EECs and the selection scheme in order to maximize accuracy while not violating a given threshold in inference time.

B. Neural Architecture Search

This section details and comments the main approaches of Hardware-Aware NAS addressing the problem of energy efficiency of deep neural networks. In particular, all these works deal with efficient deep learning inference on resource-constrained hardware. The approach described in [17] is completely hardware-agnostic exploiting constraints on network parameters and other aspects related to the neural architecture itself, whereas [19] exploits other model compression techniques to retain the model size and uses the real energetic parameters of the architecture on the hardware. In the last part of this section, we investigate some recent works proposing NAS solutions to optimize adaptive neural networks.

One-shot NAS algorithms leverage weight sharing among models in neural architecture search space to train a Once-For-All (OFA) supernet [30]. The supernet, which encompasses many configurations of networks, is trained only once on a big dataset such as Imagenet [14]. Then, the subnetworks of the supernet are evaluated without re-training them on a target dataset, hence significantly reducing the search time by exploiting fine-tuning on the pre-trained supernet for their training.

In this direction, Constrained Neural Architecture Search [17], based on MSuNAS [31], is an example of a One-Shot NAS. It performs a bi-objective optimization on many CNN-based OFAs (e.g., MobileNetV3). In particular, this solution is able to jointly optimize the accuracy of the model and a secondary objective. The possible secondary objectives are the number of parameters of the network, the number of Multiply-and-Accumulate (MAC) operations, or the number of activations in the CNN. This work extends MSUNAS by providing an estimate of the number of activations, the possibility of optimizing all the secondary objectives jointly, and imposing functional or technological constraints. The functional constraints refer to the type of processing layers and operations that are carried out in the designed neural network, while the technological constraints limit the computational and memory demand of the designed neural network. This NAS exploits a genetic algorithm NSGA-II [32] for the search strategy and an accuracy predictor surrogate model to predict the accuracy of the candidate networks. The accuracy predictor in MSuNAS is obtained through a selection mechanism that constructs four types of surrogate models at every iteration and adaptively selects the best model via cross-validation. The four considered surrogates are RBF [33], Multi-Layer Perceptron (MLP) [34], Classification And RegressionTrees (CARTS) [35], and Gaussian Processes (GP) [36].

Joint Search for Network Architecture, Pruning, and Quantization Policy (APQ) [19] is a framework to jointly optimize the neural architecture, pruning policy, and mixed precision quantization policy. Compared to the separate optimization approach [37] [38] [39], this work achieves significant benefits in terms of energy and accuracy. The approach is similar to CNAS since it relies on OFA and uses an accuracy predictor to speed up the evaluation, and genetic evolution for the search strategy. However, this work presents a novel quantization-aware accuracy predictor that allows to populate a lookup-table of quantized models with entries consisting of the architecture, the bit-width precision of each layer, the accuracy, and the energetic parameters. On top of this lookup-table, the genetic algorithm is able to search for the optimal candidate quantized architectures.

Some recent works propose NAS solutions for optimizing the design of neural networks for adaptive inference. Recall that EENNs perform adaptive inference in a layer-wise fashion, i.e., placing the EECs between layers of the same network. Differently, some of these works are focused on a different mechanism of adaptive inference. The work [20] is a NAS framework for neural networks performing early exit in a network-wise fashion meaning that EECs are placed between different networks in cascade. S2DNAS [21] presents a NAS for neural networks performing adaptive inference in a channel-wise fashion, i.e. splitting the network along the channel width. Other works focus on NAS solutions specifically for EENNs. EExNAS [22] is a simple first attempt to optimize an EENN. It accounts for only one EECC at a fixed position. ESAI [23] is a NAS for Early Exit in a Cloud setting. The network is divided into two sections, the lighter one processed locally and the heavier one on the server. The early exit mechanism is used to decide where to send the samples. HADAS [24] proposes a framework to optimize the backbone network and EECs, varying their number and placement, on two levels with two different genetics. So, this NAS casts the problem of designing an EENN not as a joint optimization but as a 2-step optimization. On the outer level, HADAS chooses the optimal backbone networks by sampling from an OFA supernet and taking into consideration the hardware constraints, while on the inner level, takes the best networks optimized by the outer level, mounts the EECs on top of the backbone networks, trains the obtained EENNs, and then selects the optimal EENNs. The EENNs are trained using the KD approach, explained in Section II-A, by keeping the backbone weights frozen. We emphasize that the current literature is still lacking NAS solutions for the co-design, on the same level, of neural networks with many (possible) EECs with different hyperparameters guaranteeing a sufficient degree of generality.

III. Problem formulation

The problem addressed by EDANAS consists in selecting the architecture of an EENN aiming at providing a high accuracy on a given task and reducing the MACs associated to the EE. Formally, EDANAS can be defined as the following joint optimization problem:

$$\begin{aligned} &\text{minimize } \mathcal{G}\left(\mathcal{ACC}(\widetilde{x}), \mathcal{ADA\_MACS}(\widetilde{x})\right) \\ &\text{s. t. } \widetilde{x} \in \Omega_{\widetilde{x}} \end{aligned} \tag{3}$$

where $\mathcal{G}$ is a bi-objective optimization function, $\widetilde{x}$ and $\Omega_{\widetilde{x}}$ represent a candidate EENN architecture and the corresponding

search space consisting of the architectural parameters and the hyperparameters of the selection scheme, respectively, $\mathcal{ACC}(\widetilde{x})$ is the classification accuracy of $\widetilde{x}$, and $\mathcal{ADA\_MACS}(\widetilde{x})$ is the number of MACs in the EE inference of $\widetilde{x}$. We emphasize that the backbone network of an EENN $\widetilde{x}$ is referred to x in the next sections.

The optimization problem described in Eq. 3 will be tackled by the proposed EDANAS framework that is introduced in the next Section.

IV. The proposed EDANAS

This section introduces the proposed EDANAS, which is a NAS solution able to jointly optimize the design of EENN and its selection scheme.

An overall description of the EDANAS framework is given in Fig. 1. In more detail, EDANAS 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 ADA set Ω_θ from which the selection scheme parameters are obtained. We emphasize that this work focuses on the image classification task but what here proposed can be extended to other classification and recognition tasks.

At the end of the search, EDANAS returns the set of the k optimal EENN architectures $X^\circ = \{x_1^\circ, \ldots, x_k^\circ\}$, being k a value specified by the user. We emphasize that the output of EDANAS is a set of networks enhanced with EECs, while the OFA given in input consists of single exit neural networks (i.e., no EECs).

We now detail the main parts of EDANAS. Section IV-A explains the training phase of the EENNs. Section IV-B presents the inference phase of the EENNs. Section IV-C provides the definition of the novel figure of merit ADA_MACS. Lastly, Section IV-D introduces the EDANAS framework.

A. Training phase of the EENNs

The optimization objective in classification tasks of EDANAS relies on the softmax cross-entropy loss function. The design goal of each EEC is to minimize this loss function. In line with the JT approach, described in Section II-A, to aim at training the entire EENN, we design the joint optimization problem as a weighted sum of the loss functions of the classifiers of the EENN (including the final exit):

$$L_{EENN}(\hat{y}, y, \theta) = \sum_{n=1}^{N} w_n L(\hat{y}_n, y, \theta_n) + w_{N+1} L(\hat{y}_{N+1}, y) \tag{4}$$

where N is the total number of EECs, w_n is the weight assigned to the n-th loss term, $\hat{y}$ is the set of predictions computed by each classifier of the EENN (including the final exit), y is the set of true values, and θ_n is the threshold of the n-th EEC. We emphasize that the term $L(\hat{y}_{N+1}, y)$ does not depend on θ since it refers to the last exit of the EENN.

The steps performed to train EENNs in the EDANAS framework are the following ones:

- Sample a neural network x from Ω_x and a selection scheme θ from Ω_θ;
- Load the available pre-trained weights on x;
- Use x as a backbone network and add EECs to x following the specifications of the selection scheme θ. We refer to the resulting EENN as $\widetilde{x}$.
- Train $\widetilde{x}$ in a JT approach, combining the losses of each classifier, for a user-defined number of epochs.

B. Inference phase of the EENNs

Once trained, the EENN can be used for EE inference by classifying samples as soon as enough confidence is achieved during the processing. As presented in Section II-A, such a decision is made by using the Score Margin (SM) approach. At the n-th EEC, given the vector P_n containing the estimated posterior probabilities (score values) for all possible class labels, we compute SM_n as the difference between the first and the second largest score value and compare it with the corresponding threshold θ_n which is the n-th entry of the set of thresholds used for the selection mechanism. If SM_n is greater or equal than θ_n, the output of the EENN $\hat{y}$ becomes the output of the n-th EEC $\bar{y}_n$, hence skipping the execution of the remaining M-n blocks. This mechanism is shown in Fig. 2.

C. Definition of ADA_MACS

The novel metric ADA_MACS proposed in this work is defined as the sum of the number of MACs of each EEC weighted by the utilization of the corresponding EEC. This metric is computed as follows:

$$\begin{aligned} ADA\_MACS(\widetilde{x}) = &\sum_{n=1}^{N} MACS(x_n) W_n(\theta_n) \\ &+ MACS(x_{N+1}) W_{N+1}(\theta_n) \end{aligned} \tag{5}$$

where N is the total number of EECs, x_n is the subnetwork consisting of the backbone network up to the n-th exit followed by the auxiliary classifier of the n-th exit, x_{N+1} is the backbone network, θ_n is the threshold of the n-th EEC[2], and $W_n(\cdot)$ is accounting for the percentage of samples classified by the n-th EEC (intuitively $W_{N+1}(\cdot)$ refers to the final exit of the EENN). We emphasize that $\sum_{n=1}^{N} W_n(\theta_n) + W_{N+1} = 1$. We can reformulate the secondary objective of EDANAS as the problem of increasing the gap between $MACS(x)$ and $ADA\_MACS(\widetilde{x})$ with $ADA\_MACS(\widetilde{x}) <= MACS(x)$ to stress the reduction in inference time of an early exit neural network with respect to a single exit one.

D. The EDANAS framework

In EDANAS, the optimization problem defined in Eq.6 has been reformulated as follows:

$$\begin{aligned} &\text{minimize } \mathcal{G}\left(\mathcal{S}_{\mathcal{ACC}}(\widetilde{x}), \mathcal{S}_{\mathcal{ADA\_MACS}}(\widetilde{x})\right) \\ &\text{s. t. } \widetilde{x} \in \Omega_{\widetilde{x}} \end{aligned} \tag{6}$$

[2] $L(\cdot)$ and $W_n(\cdot)$ depends on θ_n as well as on all the terms θ_1, θ_2, .., θ_{n-1}. In the formulas, we wrote only θ_n for simplicity.

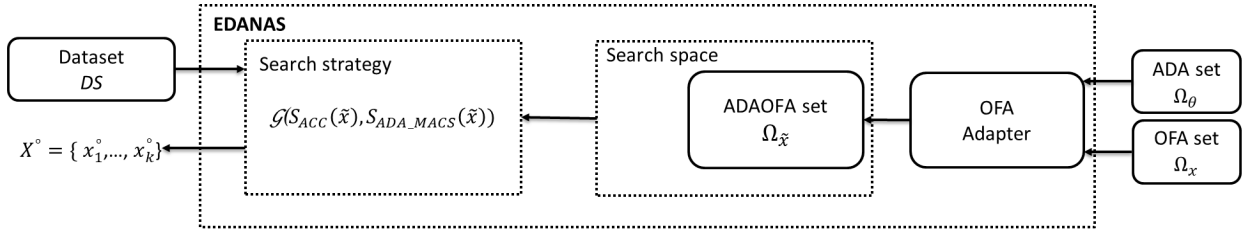


Fig. 1: The proposed EDANAS framework, composed of an OFA Adapter and a Search Strategy module.

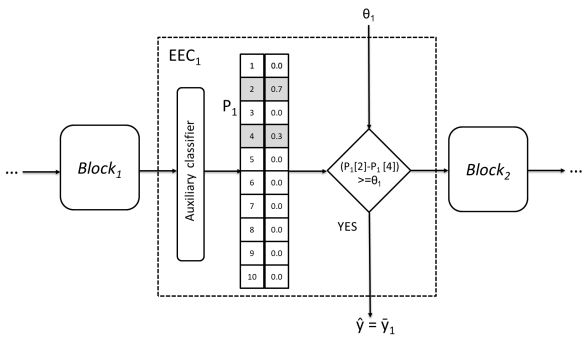


Fig. 2: Early Exit Inference: a detail of the 1^{st} EEC.

where $\mathcal{S}_{\mathcal{ACC}}(\widetilde{x})$ is the predictor of the classification accuracy of $\widetilde{x}$ introduced in MSuNAS and $\mathcal{S}_{\mathcal{ADA\_MACS}}(\widetilde{x})$ is the predictor of the number of MACS in the EE inference of $\widetilde{x}$ introduced in Section IV-C.

In more detail, the *OFA Adapter* module takes in input the ADA set Ω_θ and the OFA set Ω_x and enhances each x, candidate network of the OFA, with a possible selection scheme configuration θ by the ADA set. The output is a new OFA set $\Omega_{\widetilde{x}}$, called ADAOFA, consisting of EENNs. More specifically, given M building blocks of the backbone network, the OFA Adapter can place up to a maximum of N = M - 1 EECs, one between each of these building blocks. To handle the exit EECs placement, we imposed that for θ_n equal to 1, with $n = 1, .., N$ the EEC is not placed for that intermediate point[3]. This makes sense since θ_n equal to 1 means that a decision is made by the EEC only when the EEC has 100% confidence, resulting in no samples exiting from that EEC (i.e., that EEC is switched off). We emphasize that the possible position of the EECs is fixed given the building blocks of the backbone network, while whether using or not the EEC in that specific position depends on the values of the thresholds.

The fundamental part of EDANAS is the *Search Strategy* module. It adopts the genetic algorithm NSGA-II [32] to solve the bi-objective problem introduced in Eq. 6, by jointly optimizing the objectives $\mathcal{S}_{ACC}(\widetilde{x})$ and $\mathcal{S}_{ADA\_MACS}(\widetilde{x})$ on the dataset $\mathcal{DS}$, and by evaluating network architectures candidates from $\Omega_{\widetilde{x}}$. Before starting the search, a representative set of the OFA $\Omega_{\widetilde{x}}$, is sampled (usually around 100 samples) and the EENNs are trained for a few epochs and their classification accuracy and ADA_MACS are computed. We call this set the *archive*. The number of samples is a parameter chosen by the user. The search process is iterative: at each iteration, the two surrogates are chosen by means of a mechanism called "adaptive-switching", the same method used by MSUNAS [31], which selects the best surrogate model according to a correlation metrics (i.e., Kendall' s Tau [40]) and training the possible surrogate models using the archive as the dataset. Then, a ranking of the candidate networks, based on their predicted accuracy and predicted ADA_MACS, is computed and a new set of candidates is obtained by NSGA-II and added to the archive, available for evaluation in the next iteration.

[3]After the placement of the EECs, θ_n values equal to 1 are removed from the selection scheme since they are used only in the EENN definition. In this way, the cardinality of the set θ is equal to the number of the EECs of the corresponding EENN.

V. Experimental results

This section shows the results of the experimental campaign aiming at assessing the effectiveness of EDANAS. In more detail, the purpose of this section is to point out that EDANAS is able to design network architectures that are effective (i.e., providing high accuracy for the considered task) and characterized by a reduced computational demand (i.e., a low number of MACs in EE inference) with respect to state-of-the-art EENNs that have been manually designed by experts.

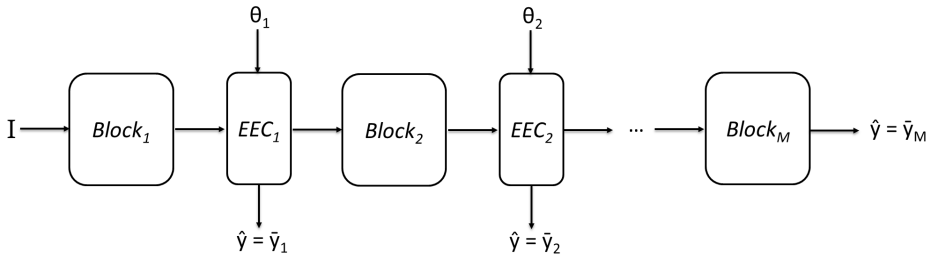
A. Dataset

The dataset used in the experimental campaign is the CIFAR-10 [41], which consists of 32x32 color images representing 10 classes of objects. The training set comprises 60000 images, while the testing set comprises 10000 images.

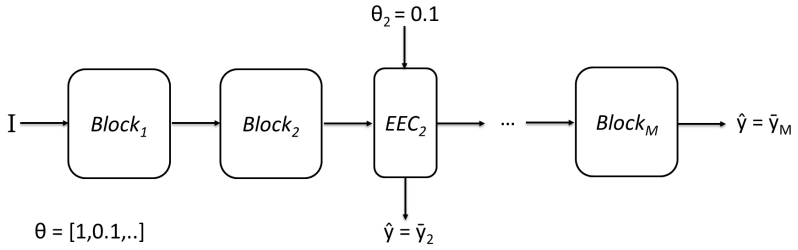
B. Experimental setup

The setup of the experiments is the following:

- dataset is CIFAR-10;
- the 1^{st} objective is top1 accuracy and the 2^{nd} objective is ADA_MACS;
- the weights w_i of the training loss function are set to 1;
- the initial number of samples populating the archive is 100;
- the training epochs for the candidate networks is 5;
- the Ω_x OFA supernet is MobileNetV3;
- the number of exit points (i.e., the points of the network where EECs are possibly placed) is set to 4;
- the possible values of ADA set Ω_θ for the thresholds of the selection scheme are 0.1, 0.2, and 1. Recall that θ_n



(a) General architecture of an EENN designed by EDANAS.



(b) Architecture of an EENN designed by EDANAS with no EEC after the first block and an EEC with $\theta_2 = 0.1$ placed after the second block.

Fig. 3: Early Exit Neural Network in the EDANAS framework.

equal to 1 results in no EEC placed at that n-th exit point of the network;
- the number of EDANAS iterations is 30;
- the number k of optimal architecture in output is 4.

The output set of optimal architectures X° contains one EENN for each possible number of EECs, ranging from 1 to 4 EECs. These archictectures are pareto optimal using the best trade-off between top1 accuracy and ADA_MACS as selection criterion. In particular, each optimal EENN is selected among a set of EENNs with the same number of EECs.

As a comparison, we considered the following state-of-the-art EENNs.

1) an EENN [42] with an auxiliary classifier consisting of two fully-connected layers with 1024 and 10 neurons placed after the first max-pooling. We refer to this network as *EECNN*;
2) a CNN network (designed from scratch) [42] consisting of a convolutional layer with 32 filters 5x5, a 2x2 max-pooling with stride 2, a convolutional layer with 64 filters 5x5, a 2x2 max-pooling with stride 2 and two fully-connected layers with 1024 and 10 neurons. We refer to this network as *standalone CNN*;
3) AlexNet, VGG-13, and ResNet-18 modified to handle 32x32 images with multiple EECs [28]. The networks are enriched with 5 EECs for AlexNet and 9 for the other two architectures. Each EEC consists of a convolutional layer with 128 filters, a max-pooling, and a fully-connected layer. We refer to these networks as *EEAlexNet*, *EEVGG-13*, and *EEResNet-18*;
4) AlexNet, VGG-13 and ResNet-18 modified to handle 32x32 images [28]. We refer to these networks as *AlexNet*, *VGG-13*, and *ResNet-18*;
5) the standalone backbone networks of the optimal networks found by EDANAS, which are networks belonging to the MobileNetV3-based OFA supernet Ω_x. We refer to these networks as NAS_n being n the number of EECs in their EENN counterpart.

C. Analysis of the results

Experimental results are presented in Table I. The optimal networks searched by EDANAS are referred to as $EDANAS_n$ being n the number of EECs. These networks are the EENN counterpart of NAS_n being n the number of EECs. The column *NAS* specifies if the network is designed by a NAS procedure (yes) or not (no). The column *Exits* specifies the number of EECs (including the final exit). The column θ specifies the thresholds of the selection scheme. The column *Utils* specifies the utilization of the EECs (including the final exit) in terms of percentage. Note that for a network with a single exit (i.e., no EECs), ADA_MACS coincides with the number of MACs, θ is not present, and the Utils is [1.0].

The reduction of MACs of the EENNs produced by EDANAS with respect to their single exit counterpart is 0.2x, 0.37x, 0.47x, and 0.7x. The drop in accuracy of the EENNs with respect to the single exit counterpart is 0.011x, 0.066x, 0.044x, and 0.2x. Interestingly, the reduction in MACs of $EDANAS_n$ shows an increasing monotonic trend with respect to n, while the drop in accuracy does not follow a monotonic trend with respect to n showing a significant drop in accuracy of $EDANAS_4$.

The number of MACs is not explicitly given in the work [42] since the complexity in inference is given as a number of multiplies. However, we can approximately state

TABLE I: Results for the experimental campaign of EDANAS.

Model	*NAS*	*Exits*	*ADA_MACS* (M)	*Accuracy*	θ	*Utils*
NAS_1 [17]	yes	1	21.67	81.8%	-	[1.0]
$EDANAS_1$ (our)	yes	2	17.31	80.9%	[1.0,1.0,1.0,0.1]	[0.90, 0.10]
NAS_2 [17]	yes	1	37.21	**86.8%**	-	[1.0]
$EDANAS_2$ (our)	yes	3	23.52	81.1%	[0.2,1.0,1.0,0.1]	[0.04, 0.86, 0.10]
NAS_3 [17]	yes	1	18.14	81.3%	-	[1.0]
$EDANAS_3$ (our)	yes	4	9.65	77.7%	[0.2,1.0,0.2,0.2]	[0.03, 0.70, 0.14, 0.13]
NAS_4 [17]	yes	1	10.64	78.2%	-	[1.0]
$EDANAS_4$ (our)	yes	5	**3.15**	62.3%	[0.1,0.1,0.1,0.1]	[0.15, 0.36, 0.35, 0.09, 0.05]
Standalone CNN [42]	no	1	25.14	77.2%	-	[1.0]
EECNN [42]	no	2	7.48	76.5%	[0.1]	[1.0, 0.0]
AlexNet [28]	no	1	-	73.2%	-	[1.0]
EEAlexNet [28]	no	6	-	79.9%	-	[0.01,0.82,0.12,0,05]
VGG-13 [28]	no	1	-	75.4%	-	[1.0]
EEVGG-13 [28]	no	10	-	86.7%	-	-
ResNet-18 [28]	no	1	-	81.6%	-	[1.0]
EEResNet-18 [28]	no	10	-	84.9%	-	[0.004,0.22,0.02,0.52,0.14,0.096]

that the number of MACS is two times the number of multiplies. *EECNN*, with respect to $EDANAS_1$, performs the same in terms of reduction of MACs (0.7x) but better in terms of the drop in accuracy (0.007x). Interestingly, *EECNN* exits all the samples in the auxiliary classifier.

The number of MACs is not reported in the work [28] but the speed-up in inference time with respect to the baseline (the single exit counterpart) is given. The θ is not reported in the table since, as mentioned in II-A, the work [28] uses a different strategy for the EE so we cannot compare directly the value of the thresholds with the other EENNs. The value of *Utils* of *EEVGG-13* is not reported in the table due to the inconsistency of this result since the total utilization, obtained by summing the utilization of each EEC and the final exit given in the work [28], is greater than 1. The hand-made *EEAlexNet*, *EEVGG-13*, and *EEResNet-18* achieves brilliant results in terms of accuracy outperforming their single exit counterpart and shows a speed-up of 0.12x, 0.44x, and 0.58x in inference time.

Interestingly, most early-exit solutions found with EDANAS exit only very late into the architecture. The NAS procedure seems to favor the selection of more complex EENNs since they tend to perform significantly better in terms of accuracy with respect to smaller networks on average. It is evident that these smaller networks would benefit from better optimization, especially in the training techniques.

All in all, the results point out that EDANAS is able to provide automatically EENNs comparable with the expert-designed EENNs in the current literature.

VI. Conclusions

This work introduced Early exit aDAptive Neural Architecture Search (EDANAS), a framework that allows to co-design both the network architecture and the selection scheme to perform adaptive inference to achieve optimal accuracy and number of MACs. The software implementation, built on top of a state-of-the-art NAS, is released to the scientific community as a public repository. The framework has proven to develop EE solutions able to compete with expert-designed models. Future works will encompass the use of constraints (which can be enabled in CNAS) in relation to the selection scheme, a proper setting of the weights of the training loss function, different multi-objective optimization algorithms, more application scenarios, new OFAs, and the combination of EDANAS with other AutoML techniques.

Acknowledgment

This paper is supported by PNRR-PE-AI FAIR project funded by the NextGeneration EU program.

References

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

[2] S. Albawi, T. A. Mohammed, and S. Al-Zawi, "Understanding of a convolutional neural network," in *2017 International Conference on Engineering and Technology (ICET)*. Antalya: IEEE, Aug. 2017, pp. 1–6. [Online]. Available: https://ieeexplore.ieee.org/document/8308186/

[3] S. Bai, J. Z. Kolter, and V. Koltun, "An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling," Apr. 2018, arXiv:1803.01271 [cs]. [Online]. Available: http://arxiv.org/abs/1803.01271

[4] X. Ying, "An Overview of Overfitting and its Solutions," *Journal of Physics: Conference Series*, vol. 1168, p. 022022, Feb. 2019. [Online]. Available: https://iopscience.iop.org/article/10.1088/1742-6596/1168/2/022022

[5] C.-Y. Lee, S. Xie, P. Gallagher, Z. Zhang, and Z. Tu, "Deeply-Supervised Nets," Sep. 2014, arXiv:1409.5185 [cs, stat]. [Online]. Available: http://arxiv.org/abs/1409.5185

[6] S. Scardapane, M. Scarpiniti, E. Baccarelli, and A. Uncini, "Why Should We Add Early Exits to Neural Networks?" *Cognitive Computation*, vol. 12, no. 5, pp. 954–966, Sep. 2020. [Online]. Available: https://link.springer.com/10.1007/s12559-020-09734-4

[7] Y. Kaya, S. Hong, and T. Dumitras, "Shallow-Deep Networks: Understanding and Mitigating Network Overthinking," May 2019, arXiv:1810.07052 [cs, stat]. [Online]. Available: http://arxiv.org/abs/1810.07052

[8] X. Wang, Y. Luo, D. Crankshaw, A. Tumanov, F. Yu, and J. E. Gonzalez, "IDK Cascades: Fast Deep Learning by Learning not to Overthink," Jun. 2018, arXiv:1706.00885 [cs]. [Online]. Available: http://arxiv.org/abs/1706.00885

[9] S. Teerapittayanon, B. McDanel, and H. T. Kung, "BranchyNet: Fast Inference via Early Exiting from Deep Neural Networks," Sep. 2017, arXiv:1709.01686 [cs]. [Online]. Available: http://arxiv.org/abs/1709.01686

[10] C. Szegedy, Wei Liu, Yangqing Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Boston, MA, USA: IEEE, Jun. 2015, pp. 1–9. [Online]. Available: http://ieeexplore.ieee.org/document/7298594/

[11] E. Belilovsky, M. Eickenberg, and E. Oyallon, "Greedy Layerwise Learning Can Scale to ImageNet," Apr. 2019, arXiv:1812.11446 [cs, stat]. [Online]. Available: http://arxiv.org/abs/1812.11446

[12] E. S. Marquez, J. S. Hare, and M. Niranjan, "Deep Cascade Learning," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 11, pp. 5475–5485, Nov. 2018. [Online]. Available: https://ieeexplore.ieee.org/document/8307262/

[13] J. Frankle and M. Carbin, "The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks," Mar. 2019, arXiv:1803.03635 [cs]. [Online]. Available: http://arxiv.org/abs/1803.03635

[14] J. Deng, W. Dong, R. Socher, L.-J. Li, Kai Li, and Li Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*. Miami, FL: IEEE, Jun. 2009, pp. 248–255. [Online]. Available: https://ieeexplore.ieee.org/document/5206848/

[15] 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.

[16] H. Benmeziane, K. El Maghraoui, H. Ouarnoughi, S. Niar, M. Wistuba, and N. Wang, "Hardware-Aware Neural Architecture Search: Survey and Taxonomy," in *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*. Montreal, Canada: International Joint Conferences on Artificial Intelligence Organization, Aug. 2021, pp. 4322–4329. [Online]. Available: https://www.ijcai.org/proceedings/2021/592

[17] M. Gambella, A. Falcetta, and M. Roveri, "CNAS: Constrained Neural Architecture Search," in *2022 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. Prague, Czech Republic: IEEE, Oct. 2022, pp. 2918–2923. [Online]. Available: https://ieeexplore.ieee.org/document/9945080/

[18] H. Bai, M. Cao, P. Huang, and J. Shan, "BatchQuant: Quantized-for-all Architecture Search with Robust Quantizer."

[19] T. Wang, K. Wang, H. Cai, J. Lin, Z. Liu, and S. Han, "APQ: Joint Search for Network Architecture, Pruning and Quantization Policy," Jun. 2020, arXiv:2006.08509 [cs, stat]. [Online]. Available: http://arxiv.org/abs/2006.08509

[20] D. Stamoulis, T.-W. Chin, A. K. Prakash, H. Fang, S. Sajja, M. Bognar, and D. Marculescu, "Designing Adaptive Neural Networks for Energy-Constrained Image Classification," Aug. 2018, arXiv:1808.01550 [cs, stat]. [Online]. Available: http://arxiv.org/abs/1808.01550

[21] Z. Yuan, B. Wu, G. Sun, Z. Liang, S. Zhao, and W. Bi, "S2DNAS: Transforming Static CNN Model for Dynamic Inference via Neural Architecture Search," in *Computer Vision – ECCV 2020*, A. Vedaldi, H. Bischof, T. Brox, and J.-M. Frahm, Eds. Cham: Springer International Publishing, 2020, vol. 12347, pp. 175–192, series Title: Lecture Notes in Computer Science. [Online]. Available: https://link.springer.com/10.1007/978-3-030-58536-5_11

[22] M. Odema, N. Rashid, and M. A. A. Faruque, "EExNAS: Early-Exit Neural Architecture Search Solutions for Low-Power Wearable Devices," in *2021 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*. Boston, MA, USA: IEEE, Jul. 2021, pp. 1–6. [Online]. Available: https://ieeexplore.ieee.org/document/9502503/

[23] B. Zeinali, D. Zhuang, and J. M. Chang, "ESAI: Efficient Split Artificial Intelligence via Early Exiting Using Neural Architecture Search," Jun. 2021, arXiv:2106.12549 [cs]. [Online]. Available: http://arxiv.org/abs/2106.12549

[24] H. Bouzidi, M. Odema, H. Ouarnoughi, M. A. A. Faruque, and S. Niar, "HADAS: Hardware-Aware Dynamic Neural Architecture Search for Edge Performance Scaling," Dec. 2022, arXiv:2212.03354 [cs]. [Online]. Available: http://arxiv.org/abs/2212.03354

[25] J. Kim, J. K. Lee, and K. M. Lee, "Deeply-Recursive Convolutional Network for Image Super-Resolution," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Las Vegas, NV, USA: IEEE, Jun. 2016, pp. 1637–1645. [Online]. Available: http://ieeexplore.ieee.org/document/7780550/

[26] S. Abbasi, M. Hajabdollahi, N. Karimi, and S. Samavi, "Modeling teacher-student techniques in deep neural networks for knowledge distillation," *CoRR*, vol. abs/1912.13179, 2019. [Online]. Available: http://arxiv.org/abs/1912.13179

[27] E. Park, D. Kim, S. Kim, Y.-D. Kim, G. Kim, S. Yoon, and S. Yoo, "Big/little deep neural network for ultra low power inference," in *2015 International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS)*. Amsterdam, Netherlands: IEEE, Oct. 2015, pp. 124–132. [Online]. Available: http://ieeexplore.ieee.org/document/7331375/

[28] S. Scardapane, D. Comminiello, M. Scarpiniti, E. Baccarelli, and A. Uncini, "Differentiable Branching In Deep Networks for Fast Inference," in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Barcelona, Spain: IEEE, May 2020, pp. 4167–4171. [Online]. Available: https://ieeexplore.ieee.org/document/9054209/

[29] F. Ilhan, L. Liu, K.-H. Chow, W. Wei, Y. Wu, M. Lee, R. Kompella, H. Latapie, and G. Liu, "EENet: Learning to Early Exit for Adaptive Inference," Jan. 2023, arXiv:2301.07099 [cs]. [Online]. Available: http://arxiv.org/abs/2301.07099

[30] H. Cai, C. Gan, T. Wang, Z. Zhang, and S. Han, "Once-for-All: Train One Network and Specialize it for Efficient Deployment," Apr. 2020, arXiv:1908.09791 [cs, stat]. [Online]. Available: http://arxiv.org/abs/1908.09791

[31] Z. Lu, K. Deb, E. Goodman, W. Banzhaf, and V. N. Boddeti, "NSGANetV2: Evolutionary Multi-Objective Surrogate-Assisted Neural Architecture Search," Jul. 2020, arXiv:2007.10396 [cs]. [Online]. Available: http://arxiv.org/abs/2007.10396

[32] 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, Apr. 2002. [Online]. Available: http://ieeexplore.ieee.org/document/996017/

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

[34] 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.

[35] 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.

[36] X. Dai, P. Zhang, B. Wu, H. Yin, F. Sun, Y. Wang, M. Dukhan, Y. Hu, Y. Wu, Y. Jia, P. Vajda, M. Uyttendaele, and N. K. Jha, "ChamNet: Towards Efficient Network Design through Platform-Aware Model Adaptation," Dec. 2018, arXiv:1812.08934 [cs]. [Online]. Available: http://arxiv.org/abs/1812.08934

[37] H. Cai, L. Zhu, and S. Han, "ProxylessNAS: Direct Neural Architecture Search on Target Task and Hardware," Feb. 2019, arXiv:1812.00332 [cs, stat]. [Online]. Available: http://arxiv.org/abs/1812.00332

[38] Y. He, J. Lin, Z. Liu, H. Wang, L.-J. Li, and S. Han, "AMC: AutoML for Model Compression and Acceleration on Mobile Devices," Jan. 2019, arXiv:1802.03494 [cs]. [Online]. Available: http://arxiv.org/abs/1802.03494

[39] K. Wang, Z. Liu, Y. Lin, J. Lin, and S. Han, "HAQ: Hardware-Aware Automated Quantization with Mixed Precision," Apr. 2019, arXiv:1811.08886 [cs]. [Online]. Available: http://arxiv.org/abs/1811.08886

[40] 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

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

[42] S. Disabato and M. Roveri, "Reducing the Computation Load of Convolutional Neural Networks through Gate Classification," in *2018 International Joint Conference on Neural Networks (IJCNN)*. Rio de Janeiro: IEEE, Jul. 2018, pp. 1–8. [Online]. Available: https://ieeexplore.ieee.org/document/8489276/