Anticipate, Ensemble and Prune: Improving Convolutional Neural Networks via Aggregated Early Exits

Simone Sarti
DEIB, Politecnico di Milano
Milan, Italy
simone.sarti@mail.polimi.it

Eugenio Lomurno
DEIB, Politecnico di Milano
Milan, Italy
eugenio.lomurno@polimi.it

Matteo Matteucci
DEIB, Politecnico di Milano
Milan, Italy
matteo.matteucci@polimi.it

***Abstract*—Today, artificial neural networks are the state of the art for solving a variety of complex tasks, especially in image classification. Such architectures consist of a sequence of stacked layers with the aim of extracting useful information and having it processed by a classifier to make accurate predictions. However, intermediate information within such models is often left unused. In other cases, such as in edge computing contexts, these architectures are divided into multiple partitions that are made functional by including early exits, i.e., intermediate classifiers, with the goal of reducing the computational and temporal load without extremely compromising the accuracy of the classifications. In this paper, we present Anticipate, Ensemble and Prune (AEP), a new training technique based on a weighted ensemble of early exits, which aims at exploiting the information in the structure of networks to maximise their performance. Through a comprehensive set of experiments, we show how the use of this approach can yield average accuracy improvements of up to 15% over traditional training. AEP's internal pruning operation also allows reducing the number of parameters by up to 41%, lowering the number of multiplications and additions by 18% and the latency time to make inference by 16%. By using AEP, it is also possible to learn weights that allow early exits to achieve better accuracy values than those obtained from single-output reference models. The code will be available on GitHub after acceptance of the paper.**

***Index Terms*—Early Exits, Ensemble, Pruning, AEP, Image Classification, Convolutional Neural Networks**

I. Introduction

Over the last decade, deep learning has emerged as one of the dominant disciplines in computer science, thanks to the research conducted and the remarkable results obtained. Among the main fields of application, that of visual recognition, and in particular that of image classification, has undoubtedly been the catalyst for a veritable revolution, rooted in the development and refinement of architectures known as convolutional neural networks (ConvNets). Such artificial neural networks involve numerous convolutional layers that extract the information contained in input images and process it to obtain high-level features.

The development of the first top performing ConvNet, called AlexNet [1], was made possible by the increasing production and storage of data and, in particular, by the public release of the ImageNet benchmark [2]. From then on, the succession of discoveries of increasingly high-performance models has accelerated. Among the major milestones, architectures such as VGG [3], Inception [4], ResNet [5], DenseNet [6], MobileNet [7], EfficientNet [8] and recently ConvNeXt [9] excelled in setting new levels in terms of accuracy, scalability, efficiency and design quality.

Recently, the study of neural networks with early exits has gained importance. An early exit of a neural network is a classifier placed at an intermediate level between the input layer and the traditional (single) output layer. The objectives of such a design pattern are many and varied, including exploiting the information contained in the intermediate layers of the models, streamlining their overall weight by cutting them, or for purposes related to distributed systems and edge computing [10]. The optimal number of branches with early exits and their positioning represent an important choice in this context, especially for very deep ConvNets or in architectures that are not strictly sequential. Another fundamental step lies in the choice related to the update of the weights of such architectures with multiple outputs, their individual or joint use, and the management of outputs with degraded performance.

This work shows the analysis of the behaviour of the main ConvNets in the literature modified through a newly proposed early exits technique named Anticipate, Ensemble and Prune (AEP). In particular, AEP is introduced for the first time as an early exits weighted ensemble technique. Outputs aggregation strategies for both loss function and inference are discussed, in order to understand which conditions favour an improvement or lead to a loss in classification performance compared to the basic single-output version of the neural networks examined. Finally, through the adoption of a pruning step, it is shown how it is possible to reduce the number of parameters, operations and network latency, further increasing accuracy through the extraction of an optimal sub-ensemble network. The experiments are conducted on a large set of ConvNets and datasets and both in a traditional training context and through the tuning of pre-trained architectures. Unlike the main reference works in the literature, which aim to reduce latency as much as possible without sacrificing model accuracy

or to develop techniques for edge computing [11]–[13], this work aims to quantify and maximise the accuracy gain that the use of an early exits' ensemble can provide.

The document is divided into five sections. Section II summarises related works that have been proposed in the literature to better understand the remaining of the contribution. Section III describes the steps composing the AEP technique, while Section IV describes the details of the experiments and the configurations with which they were performed. Section V presents and discusses the results of the experiments, and finally, Section VI concludes the work and suggests some possible research directions.

II. Related Works

The use of early exits is a deep learning technique that aims to improve the efficiency of neural network models by allowing them to make predictions before the input data is processed by all the available layers. This is done by training multiple sub-models within a backbone model, each with a different level of complexity. One of the first works in which the early exits technique is used was carried out by Panda *et al.* and it was applied to convolutional neural networks (ConvNets) [14]. In particular, the aim of that study was to create an algorithm capable of identifying the optimal depth within the classification network under examination, so that the computational expense could be dynamically adjusted without losing accuracy. In parallel, Teerapittayanon *et al.* presented their work on early exits aimed at demonstrating the predictive properties of classifiers placed in the intermediate layers of ConvNets, such that the easiest samples to predict are processed by fewer hidden layers, while the most difficult ones traverse the entire architecture [15].

A more recent approach aimed at reducing the energy cost and complexity of single-output ConvNets has been proposed by Wang *et al.* achieving an extremely beneficial trade-off between accuracy and FLOPS, i.e., floating point operations [16]. The technique implemented by the authors involves the use of early exits and weighted loss functions applied to architectures containing skip connections. Pacheco *et al.* demonstrated how the use of early exit architectures is incredibly beneficial in the context of edge computing [17]. In particular, they have shown how the ability to classify non-anomalous samples at the shallow levels of a ConvNet allows not losing performance compared to a single-exit classification.

Early exits provide additional classifiers for a model. By having several available classifiers, some of them not necessarily high-performing, the most intuitive step to leverage on the all of them is to identify an intelligent aggregation strategy to exploit their joint potential. Ensembling, largely seen as the natural solution in many classification problems, is a machine learning technique that consists of training several different models to solve the same task and then exploiting the knowledge derived from all of them at inference time to make the best choice. The ensemble technique works because the different models have different weaknesses that are compensated by the others' strength points. Ensembling of early exits consists in classifiers sharing part of their structure and parameters, but still working on different features given the same input data. This technique has been exploited by Wołczyk *et al.* to produce an early exit-based approach in which each prediction is reused by subsequent exits, combining previous results in an ensemble-like manner [11]. The goal of that work was to minimise the prediction latency without sacrificing the accuracy performance of the proposed models.

For what concerns the training of early exit networks, joint training is the most common approach and consists in formulating a single optimization problem whose loss depends on all branches, in particular the network loss is often calculated as the weighted sum of all branches' loss [15], [18], [19]. Strategies have also been devised to dynamically adjust the exits losses weights during training [16] and to improve the ensemble by combining the prediction loss with a diversity loss [12], [20]. When early exits are trained jointly with the backbone network, they favor the learning of more discriminative features at each layer and lead to faster convergence, while also acting as regularization [15], [21].

Also the output of early exits ensemble can be computed in a multitude of ways, e.g., as arithmetic mean of the predictions [19], [20], via geometric ensemble [11] or through voting strategies [12]. The effectiveness of early exits ensembles is not limited to image classification, as they've also recently been used for image captioning [22], natural language processing [12], for uncertainty quantification and biosignal classification [13], [20] and to improve robustness against adversarial attacks [19]. Moreover, early exits ensembles were employed to produce a teacher-free knowledge distillation technique by treating the aggregated predictions as the teacher predictions [23].

The ensemble technique based on anticipated outputs proposed in this work, and called Anticipate, Ensemble and Prune (AEP), has been fully tested in the field of computer vision to solve image classification tasks, but can easily be extended to other types of data and purposes. Compared to work in the literature, AEP aims to maximise the accuracy performance of the designated model, and through the final pruning step, further optimise this metric by going in parallel to minimise the number of parameters, number of operations and inference time.

III. Method

Given a ConvNet, it is practically always possible to identify the presence of repeating blocks or stages stacked within it. Regardless of the architecture in question, it is therefore possible to replicate the classifier present at the end of the ConvNet, i.e. after the last stage, immediately after each intermediate stage, as can be seen in Figure 1.

The training of an AEP model proceeds as it follows: for each batch of B images, given C target classes, the algorithm lets the images flow through the network such that every classifier outputs a tensor of B elements, each representing one image and containing C values, one for each class c probability. In this setting, the network loss is computed as the

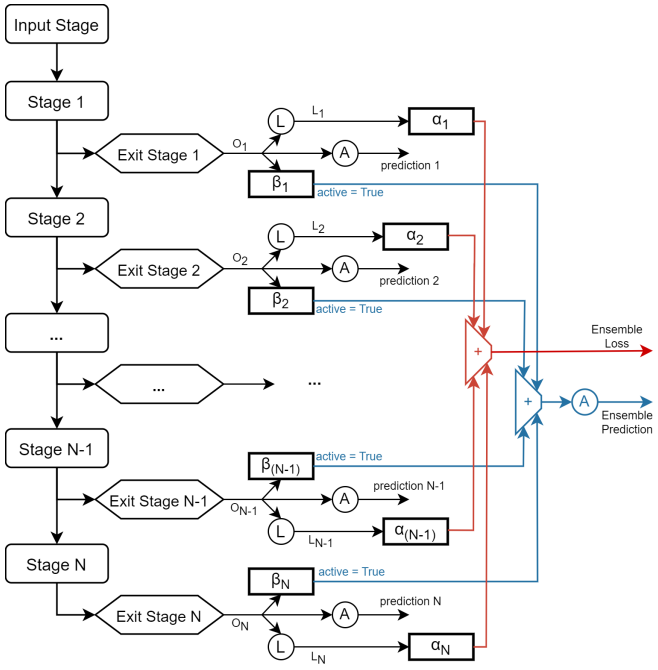

Fig. 1. Outline of the AEP technique. O_i represents the output of the i-th exit stage with linear activation. Ⓛ represents the loss function, i.e., the categorical cross-entropy, while Ⓐ represents the argmax function used to obtain the class prediction. α_i and β_i represent the weights assigned to the loss and the output of exit stage i, respectively. The activation of the outputs is relative to the pruning step with which the proposed method terminates.

weighted sum of the categorical cross entropy loss L_i of each exit $i \in [1, N]$, in a joint training fashion, as in Equation 1.

$$L_i = L_{cce,i} = -\frac{1}{B}\sum_{b=1}^{B}\sum_{c=1}^{C}(p_{b,c}\log(y_{b,c})). \tag{1}$$

Unlike other approaches to early exits, in AEP the last exit is not treated differently from the intermediate ones, since its contribution to the final loss is weighted following the same weight assignment strategy as the others. Equation 2 represents the overall network loss, where α_i is the weight assigned to the loss associated with exit i.

$$L = \sum_{i=1}^{N} \alpha_i \cdot L_i \tag{2}$$

The prediction obtained after the ensemble, from now on referred to as $\hat{y}$, is computed as a weighted sum of the outputs of each i-th output, namely O_i. The calculation of each classification metric is performed after applying the argmax operator to the vector obtained through the ensemble. Equation 3 thus represents the prediction step, where β_i is the weight assigned to the outputs associated with exit i.

$$\hat{y} = \text{argmax}(\sum_{i=1}^{N} \beta_i \cdot O_i). \tag{3}$$

The strategy for selecting weights $[\alpha_1, ..., \alpha_N]$ and $[\beta_1, ..., \beta_N]$, which are in principle different, is a crucial step in AEP. In this implementation, the weights of the loss function and the weights of the outputs were chosen to be linearly increasing/decreasing with respect to the order of the early exits. Furthermore, the values of these weights have been chosen such that they are strictly positive and with sum equal to one. In this way, it is possible to maintain the desired properties by having adaptive scales of weights regardless of the number of early exits within the neural network.

TABLE I
THE LIST OF WEIGHT CONFIGURATIONS FOR LOSS FUNCTIONS AND EXITS PREDICTIONS USED IN THE EXPERIMENTAL SECTION.

Weights Mode	Losses Weights	Outputs Weights
DESC	decreasing	decreasing
ASC	increasing	increasing
MIX	decreasing	increasing
UNIF	uniform	uniform

In the initial set of experiments, the same weights were used to compute both the network loss and the output of the ensemble. Specifically, the weights were either always increasing or always decreasing along the structure of the network. To differentiate between these two cases, the experiments in which the weights were assigned in a descending order for both the losses and the outputs were referred to as "EEdesc", while those in which the weights were assigned in an ascending order for both the losses and the outputs were referred to as "EEasc". Ablation studies in the experimental section of this paper demonstrate that, while the use of decreasing weights generally leads to better exits when considered individually, particularly for earlier exits, networks with ascending weights perform better in terms of ensemble prediction. To exploit this, the two sets of weights, i.e., losses weights and ensemble weights, have been decoupled, and "EEmix" networks are proposed, in which the exits losses weights are assigned in descending order while the output weights are assigned in ascending order. Additionally, a uniform weight mode has been tested, in which all exits are given the same weight. These experiments are referred to as "EEunif" in the experimental section. A summary of the different weight modes can be found in Table I.

Upon completion of training the multi-output network, a pruning step is applied as a post-processing technique to optimize its performance. In order to apply the proposed pruning operation, the resulting ConvNet is loaded and validated in all possible combinations of exit activation states, using the same exit weights used during the training. Then, the best sub-network is extracted and evaluated on the test set. This selection process can aid in further improving accuracy in comparison to both the full ensemble and the single-exit network, while also reducing the number of parameters, operations, and latency as entire sections of the original network can be removed.

IV. Experiments Setup

The experiments conducted to evaluate the effectiveness of the AEP technique include several ConvNet architectures and many of the major image classification benchmarks. For the

sake of completeness, all experiments have been evaluated with two different input scales, and each ConvNet has been trained both from scratch and through fine-tuning from the weights learned on ImageNet.

A. Networks

The AEP technique has been tested on 5 well known network architectures, thus encompassing many of ConvNets most relevant architectural trends and patterns. The goal was to span different network sizes, design patterns and inner components. You can refer to Figure 1 as an abstract model representing the various ConvNets considered in this study. By identifying the repeating blocks in some classical architectures, it was possible to identify the list of stages to which early exits should be attached. Each network could then be enriched with the function to extract a subnetwork by specifying which exits to preserve. Below the list of networks we have used in the experiments:

- **ResNet50** [5]: In terms of size, it is considered an average size ResNet architecture. ResNets are characterized by the use of skip/residual connections between layers, which should help in improving gradient flow through the network.
- **VGG16** [3]: It is a purely sequential ConvNet, and while smaller than other VGG models, it is still by far the most demanding in terms of operations required.
- **DenseNet169** [6]: It is considered an average size DenseNet architecture characterized by a dense blocks structure, in which features obtained after a layers are not only passed to the following ones but also concatenated to their output, so that the information in a block is better preserved and exploited.
- **MobileNetV3Small** [7]: Designed to work under mobile settings, it was a good candidate to see what would happen with tiny architectures. It is also characterized by the use of a particular type of block called the Residual Inverted Bottleneck, characterized by high memory efficiency.
- **EfficientNetB5** [8]: Part of the EfficientNet family, currently one of the best performing architecture families, this model in particular was chosen because with more than 500 layers, in contrast with MobileNetV3Small, allows to study a very deep architecture.

For each of the 5 networks, the classifier in the original single-exit architecture was extracted and replicated for each of the early exits. The only exit stage altered from the original structure was that of the VGG16 model, which by default is extremely large and over-parameterized. In this case instead of replicating the exit classifier, we use a simpler Global Average Pooling layer followed by a dense layer with a number of units equal to the number of possible classes.

The experiments conducted considered each network both with a number of early exits equal to 4, for comparison purposes, and with a number of early exits equal to the number of stages in the original model. Specifically, while the experiments on Resnet50 and DenseNet169 did not need to be repeated since 4 was already the correct number of exits to match the number of stages, VGG16 and MobileNetV3Small required also an additional set of experiments concerning 5 exits, while EfficientNetB5 required tests involving 7 exits. We refer to these 3 network variations by adding the "full" keyword to their name: **VGG16full**, **MobileNetv3Smallfull** and **EfficientNetB5full**, bringing the total number of networks tested to 8.

TABLE II
THE LIST OF DATASETS USED TO CONDUCT THE EXPERIMENTS WITH THEIR CHARACTERISTICS.

Dataset	Classes	Balanced	Channels	Train	Validation	Test
Cifar10 [24]	10	yes	RGB	45000	5000	10000
Cifar100 [24]	100	yes	RGB	45000	5000	10000
Eurosat [25]	10	no	RGB	17500	4000	5500
FMNIST [26]	10	yes	Grayscale	51000	9000	10000
GTSRB [27]	43	no	RGB	33209	6000	12630
TinyImg [28]	200	yes	RGB	90000	10000	10000

B. Datasets and Metrics

In this study, an experimental evaluation of the AEP algorithm was conducted using six different image classification datasets. These datasets were chosen to cover a range of characteristics, including variations in class imbalance and image modality, in order to assess the generalizability and robustness of the proposed algorithm. The datasets used in this study span a range of class sizes from tens to hundreds, and are summarized in Table II.

The evaluation compares the performance of baseline and ensemble networks under different scenarios, including training from scratch and fine-tuning from ImageNet weights for all networks as well as using images of sizes 224x224 and 64x64 for all datasets. For each experiment, a set of performance metrics has been collected, including Top1 accuracy, number of parameters, number of MACs (i.e., Multiply-ACcumulate operations), latency, and training time. Additionally, for multi-output networks, the Top1 accuracy for each exit has been also recorded.

C. Hyperparameters Settings

The goal of this research was to evaluate the performance of single-exit networks in comparison to early-exit ensembles rather than matching or beating state of the art results. To accomplish this, a simple standardized training approach was employed, using 100 training epochs, a batch size of 64, and a learning rate of 10^{-4} with the Adam optimizer. The other training parameters were left at their default values in the PyTorch model implementation. An early stopping technique was applied, with a patience of 12 epochs, based on the validation loss. No data augmentation or learning rate schedules were utilized. The only preprocessing applied to the images was resizing to 224 or 64 through bicubic interpolation and normalization. A small learning rate was chosen to ensure that the same hyperparameters could be used in fine-tuning too, thereby ensuring comparable results. The

models were implemented using PyTorch 1.12.1 and executed on an NVIDIA Quadro RTX 6000 GPU.

V. Results and Discussion

This section describes the results of the experiments we conducted in the present research. The analysis begins by comparing the variations in accuracy between single-output architectures and the proposed ensembled configurations. Next, the benefits of the pruning step are shown, in terms of accuracy gain, parameter reduction, optimisation of MACs and faster inference time. Finally, the accuracy performance of early exits without ensemble compared to single-output models is analysed. The symbol '*' indicates early exits experiments whose networks have been pruned according to the complete AEP procedure. In contrast, the absence of the symbol '*' represents full-ensemble networks.

A. Classification Accuracy

Table III shows the results of the experiments performed in terms of the percentage change in Top1 accuracy compared to the single-output experiments, grouped by neural network in the left table and by dataset in the right table.

With regard to the experiments performed with 224x224 images and traditional training (TRAIN-224), excellent average improvements can first of all be appreciated regardless of the strategy with which AEP was applied. For each neural network and each dataset, it is possible to identify a configuration that improves the accuracy obtained. In particular, as far as the datasets are concerned, the greatest advantages are found in CIFAR100 and TinyImageNet, which respectively obtain an average accuracy improvement of 16.23% with the EEmix* configuration and 13.94% with the EEdesc* configuration. With regard to network configurations, the improvement achieved by ResNet50 and both MobileNetV3-small versions is noteworthy too.

The general considerations made for the TRAIN-224 scenario are confirmed and underlined by the results of the experiments with 64x64 images and traditional training (TRAIN-64). In this case, with the same dataset and neural network, the problem to be solved by the algorithm is more complex due to the reduced amount of input information. In this context, it is possible to observe how beneficial AEP is overall, increasing the quality of the models regardless of their model architecture, weights configuration and dataset, in some cases making such improvements as to make the difference between a poorly performing model and a winning one. In this scenario, the EEdesc* technique seems to be definitely the favourite one, capable of improving accuracy performance by 15% on average.

Moving from the training from scratch scenario to the fine-tuning scenario, and in particular to the one with 224x224 images (FINETUNE-224), a different behavior can be observed. Since these are pre-trained algorithms on ImageNet images with similar size, the addition of early exits and ensembles seems to have a negative overall impact on average. This is reasonable in light of the fact that the starting models have an optimized set of weights available to facilitate the production of accurate predictions from the single output at the bottom of the network. This assumption is confirmed by the counterexample represented by the results obtained from the EEasc and EEasc* configurations. In fact, it can be observed from Table III how the use of ascending weights in the loss and inference phase turns out to be more suitable by favoring strongly the contribution of the outputs close to the single output of the original model.

Finally, going to the results of the last pool of experiments, i.e., those with fine-tuning and 64x64 images (FINETUNE-64), it is possible to see a different behavior from the previous case. The change of spatial domain resulted in an overall improvement in the accuracy of the AEP-based models compared to the single-output counterpart. This is probably due to the higher complexity of the problem inversely proportional to the size of the input, which thus allows better exploitation of classifiers trained on lower-level and thus less elaborate features, as it is not the case with large images that require more complex transformations. Nevertheless, the use of ascending weights turns out to be the winning move here as well, capable of improving models accuracy by an average of 4.79%.

In general, it can be argued that ensemble networks with early exits improve much more than the baseline when trained from scratch than when fine-tuned in different settings with respect to those where the models have been originally optimized. This is reasonable because when training from scratch by jointly optimising the exits, the network is able to improve the features of each layer to achieve a good classification result, producing better features in the layers closer to the network input. Fine-tuning, on the other hand, starts from the weights obtained after training single-output networks, which means that the features of the initial layers are not good classification features by themselves, but do produce good classification features in the final layers. Adding early exits may lead to contradictory behaviour, as the optimisation is trying to override the initial weights to solve a different task which is optimally solved at later stage. Indeed, by assigning higher weights to exits closer to the input, you end up assigning high classification importance to layers that should not be able to do so, which can lead to worse performance than the baseline.

Regarding the effect of the pruning step, represented by '*' in Table III, the same behaviour as described in full ensembles can be observed with slight improvements in accuracy. Table IV collects and compares the accuracy performance of pruned versions of early exits ensemble networks against their full-ensemble counterparts. Among the different types of networks, EEdesc networks seem to be the ones that benefit the most from the pruning phase, particularly in fine-tuning; noticeably, in the same learning context in which they perform the worst.

B. Parameters, Operations and Latency

The average effect, in terms of parameters, due to the use of AEP is summarised in Table V. As it can be seen from the

TABLE III

COMPARISONS BETWEEN MODELS BASED ON EARLY EXITS AND THOSE BASED ON SINGLE EXITS, CALCULATED AS AVERAGE PERCENTAGE CHANGE OF ACCURACY. THE FOUR EXPERIMENT CONFIGURATIONS ARE GROUPED IN ROWS ACCORDING TO THE TRAINING TECHNIQUE AND THE SIZE OF THE INPUT IMAGES. THE RESULTS ARE GROUPED BY ARCHITECTURE (LEFT COLUMN) AND BY DATASET (RIGHT COLUMN). IMPROVEMENTS ARE SHOWN IN GREEN, WORSENINGS IN RED. THE BEST RESULTS ARE HIGHLIGHTED IN BOLD.

TRAIN-224

Network	EEdesc	EEasc	EEmix	EEunif	EEdesc*	EEasc*	EEmix*	EEunif*
Resnet50	7.962	7.948	9.901	8.510	8.617	7.997	**10.088**	8.831
Vgg16	0.849	**4.653**	3.670	4.360	2.596	4.651	3.688	4.407
Vgg16full	0.922	4.063	3.517	4.035	3.012	4.217	3.524	**4.859**
DenseNet169	-1.335	0.804	-0.066	0.395	-0.410	0.803	0.046	**1.046**
MBV3-small	12.046	9.672	11.644	9.882	**12.289**	9.588	11.679	10.055
MBV3-smallfull	11.361	9.803	12.727	10.249	**13.661**	9.817	12.650	10.591
EffNetB5	1.630	2.100	2.108	**2.362**	1.885	2.113	2.239	2.086
EffNetB5full	3.710	3.046	4.443	4.320	4.065	2.958	**4.480**	4.359
Average	4.643	5.261	5.993	5.514	5.714	5.268	**6.049**	5.779

Dataset	EEdesc	EEasc	EEmix	EEunif	EEdesc*	EEasc*	EEmix*	EEunif*
Cifar10	3.795	3.758	5.232	4.182	4.261	3.822	**5.299**	4.365
Cifar100	13.972	11.019	16.117	12.339	15.888	11.020	**16.229**	12.680
Eurosat	3.027	2.586	2.996	2.956	3.046	2.622	**3.103**	2.993
FMNIST	0.628	0.912	0.942	1.122	0.995	0.926	0.946	**1.208**
GTSRB	-5.024	1.719	-2.914	-0.746	-3.846	**1.873**	-3.149	-0.013
TinyImg	11.462	11.573	13.586	13.232	**13.942**	11.346	13.867	13.442
Average	4.643	5.261	5.993	5.514	5.714	5.268	**6.049**	5.779

TRAIN-64

Network	EEdesc	EEasc	EEmix	EEunif	EEdesc*	EEasc*	EEmix*	EEunif*
Resnet50	20.039	10.444	16.952	14.368	**20.642**	10.655	17.790	14.402
Vgg16	6.395	4.273	5.078	5.704	**6.942**	4.252	5.376	5.607
Vgg16full	5.288	4.359	5.025	5.743	**6.277**	4.452	5.174	5.909
DenseNet169	9.852	6.805	8.222	9.117	**10.268**	6.806	8.751	9.221
MBV3-small	26.040	16.323	23.086	20.008	**26.129**	16.377	23.848	19.956
MBV3-smallfull	26.940	17.934	24.223	21.844	**28.230**	17.750	25.559	21.572
EffNetB5	7.367	2.250	6.616	4.983	7.477	2.378	**7.482**	5.861
EffNetB5full	13.607	4.323	11.213	9.007	**14.026**	5.142	12.500	9.879
Average	14.441	8.339	12.552	11.347	**14.999**	8.477	13.310	11.551

Dataset	EEdesc	EEasc	EEmix	EEunif	EEdesc*	EEasc*	EEmix*	EEunif*
Cifar10	10.510	5.643	9.763	8.233	**10.595**	5.811	10.449	8.333
Cifat100	24.093	12.637	21.284	18.241	**25.590**	12.778	22.409	18.772
Eurosat	9.491	4.462	8.317	7.316	**9.774**	5.051	9.189	7.718
FMNIST	1.793	1.071	1.727	1.772	**1.804**	1.062	1.736	1.734
GTSRB	5.309	7.208	5.881	6.853	5.196	**7.236**	5.961	6.860
TinyImg	35.450	19.012	28.340	25.665	**37.035**	18.921	30.115	25.888
Average	14.441	8.339	12.552	11.347	**14.999**	8.477	13.310	11.551

FINETUNING-224

Network	EEdesc	EEasc	EEmix	EEunif	EEdesc*	EEasc*	EEmix*	EEunif*
Resnet50	-4.199	0.288	-2.126	-0.889	-3.150	**0.352**	-2.218	-0.490
Vgg16	-0.417	1.756	0.945	1.845	0.669	1.709	0.943	**1.886**
Vgg16full	-1.635	1.895	0.987	1.550	0.255	1.977	0.932	**2.114**
DenseNet169	-6.263	-1.196	-4.153	-2.520	-4.870	**-1.004**	-3.961	-2.018
MBV3-small	-3.813	0.810	-1.274	0.119	-1.691	**0.833**	-1.255	0.593
MBV3-smallfull	-4.619	0.536	-1.274	-0.087	-0.993	**0.784**	-1.164	0.542
EffNetB5	-4.883	-0.343	-1.431	-1.186	-1.704	**-0.228**	-1.107	-0.283
EffNetB5full	-4.920	-0.604	-1.956	-1.342	-2.535	**-0.332**	-1.809	-0.649
Average	-3.844	0.393	-1.285	-0.314	-1.752	**0.511**	-1.205	0.212

Dataset	EEdesc	EEasc	EEmix	EEunif	EEdesc*	EEasc*	EEmix*	EEunif*
Cifar10	-3.247	0.070	-1.000	-0.690	-1.393	**0.260**	-0.892	-0.229
Cifar100	-6.428	0.973	-1.718	0.268	-2.540	**1.015**	-1.593	0.867
Eurosat	0.492	**0.812**	0.556	0.698	0.627	0.805	0.556	0.701
FMNIST	-0.485	**0.040**	-0.065	-0.030	-0.096	0.039	-0.090	0.030
GTSRB	-2.653	-0.207	-0.940	-0.787	-1.529	**-0.201**	-0.919	-0.354
TinyImg	-10.742	0.669	-4.546	-1.340	-5.583	**1.150**	-4.292	0.257
Average	-3.844	0.393	-1.285	-0.314	-1.752	**0.511**	-1.205	0.212

FINETUNING-64

Network	EEdesc	EEasc	EEmix	EEunif	EEdesc*	EEasc*	EEmix*	EEunif*
Resnet50	1.048	**4.252**	0.861	3.687	1.233	3.895	0.697	3.660
Vgg16	2.516	**3.500**	2.148	2.555	2.811	3.403	2.151	2.584
Vgg16full	1.805	3.714	2.108	3.454	2.738	**3.762**	2.110	3.508
DenseNet169	-0.399	2.790	0.113	1.985	-0.156	**2.979**	0.190	2.211
MBV3-small	6.060	6.084	5.949	**6.646**	6.417	5.850	6.009	6.585
MBV3-smallfull	5.097	6.119	5.773	6.233	6.418	6.102	5.719	**6.500**
EffNetB5	5.181	5.768	5.137	6.040	5.243	5.771	5.126	**6.045**
EffNetB5full	6.189	6.075	6.120	6.771	6.291	6.123	6.397	**6.792**
Average	3.437	**4.788**	3.526	4.671	3.874	4.736	3.550	4.736

Dataset	EEdesc	EEasc	EEmix	EEunif	EEdesc*	EEasc*	EEmix*	EEunif*
Cifar10	2.722	3.401	2.983	3.737	2.966	3.415	3.063	**3.768**
Cifar100	5.402	7.594	5.835	7.876	6.123	7.563	5.891	**7.954**
Eurosat	3.480	3.186	3.298	3.395	**3.512**	3.230	3.289	3.494
FMNIST	0.835	0.916	1.043	1.093	0.926	0.907	1.093	**1.138**
GTSRB	4.433	**5.640**	4.571	5.136	4.712	5.355	4.417	5.070
TinyImg	3.750	**7.989**	3.426	6.791	5.007	7.943	3.547	6.990
Average	3.437	**4.788**	3.526	4.671	3.874	4.736	3.550	4.736

first column, the addition of early exits has a minimal impact on the increase in parameters, which, on average, increase by only 2.88% compared to single-exit architectures. The key aspect is the impact of the pruning step, which in general brings considerable benefits. Focusing on the TRAINING-64 scenario, it can be seen that the EEmix* configuration results in an average parameter reduction of 41.02% and in parallel an average accuracy gain of 13.31%, as described by Table III.

With regard to the number of MACs, the average results of the experiments conducted are summarised in Table VI. Also for this metric, the addition of early exits alone has a negative impact, albeit with an average computational increase

TABLE IV

COMPARISONS BETWEEN MODELS BASED ON EARLY EXITS AFTER PRUNING AND THEIR COUNTERPARTS WITH FULL-ENSEMBLE. CALCULATED AS AVERAGE PERCENTAGE CHANGE OF ACCURACY. THE FOUR EXPERIMENT CONFIGURATIONS ARE DIVIDED BY ROW ACCORDING TO TRAINING TECHNIQUE AND INPUT IMAGE SIZE. PERCENTAGE IMPROVEMENTS DUE TO PRUNING ARE SHOWN IN GREEN, DETERIORATIONS IN RED. THE BEST RESULTS ARE HIGHLIGHTED IN BOLD. VALUES ARE AVERAGED OVER ARCHITECTURES AND DATASETS.

	EEdesc*	**EEasc***	**EEmix***	**EEunif***
TR-224	**1.006**	0.009	0.050	0.260
TR-64	0.435	0.142	**0.625**	0.193
FT-224	**2.280**	0.121	0.087	0.539
FT-64	**0.421**	-0.048	0.021	0.063

TABLE V

COMPARISONS BETWEEN MODELS BASED ON EARLY EXITS AND THEIR SINGLE-EXIT COUNTERPARTS, CALCULATED AS AVERAGE PERCENTAGE CHANGE OF PARAMETERS NUMBER. THE FOUR EXPERIMENT CONFIGURATIONS ARE DIVIDED BY ROW ACCORDING TO TRAINING TECHNIQUE AND INPUT IMAGE SIZE. IMPROVEMENTS ARE SHOWN IN GREEN, DETERIORATIONS IN RED. THE BEST RESULTS ARE HIGHLIGHTED IN BOLD. VALUES ARE AVERAGED OVER ARCHITECTURES AND DATASETS.

	EE	**EEdesc***	**EEasc***	**EEmix***	**EEunif***
TR-224	2.876	-0.566	-10.422	**-16.508**	-1.773
TR-64	2.876	-12.731	-22.549	**-41.018**	-25.300
FT-224	2.876	**-5.791**	1.773	-1.902	-0.720
FT-64	2.876	-4.260	-14.305	**-19.351**	-11.292

TABLE VI

COMPARISONS BETWEEN MODELS BASED ON EARLY EXITS AND THEIR SINGLE-EXIT COUNTERPARTS, CALCULATED AS AVERAGE PERCENTAGE CHANGE OF MACS. THE FOUR EXPERIMENT CONFIGURATIONS ARE DIVIDED BY ROW ACCORDING TO TRAINING TECHNIQUE AND INPUT IMAGE SIZE. IMPROVEMENTS ARE SHOWN IN GREEN, DETERIORATIONS IN RED. THE BEST RESULTS ARE HIGHLIGHTED IN BOLD. VALUES ARE AVERAGED OVER ARCHITECTURES AND DATASETS.

	EE	**EEdesc***	**EEasc***	**EEmix***	**EEunif***
TR-224	5.444	1.816	-1.908	**-5.438**	2.118
TR-64	5.260	-3.302	-8.549	**-17.952**	-9.078
FT-224	5.444	**-1.475**	2.830	2.514	1.115
FT-64	5.260	0.059	-1.954	**-4.085**	-1.050

TABLE VII

COMPARISONS BETWEEN MODELS BASED ON EARLY EXITS AND THEIR SINGLE-EXIT COUNTERPARTS, CALCULATED AS AVERAGE PERCENTAGE CHANGE OF INFERENCE LATENCY. THE FOUR EXPERIMENT CONFIGURATIONS ARE DIVIDED BY ROW ACCORDING TO TRAINING TECHNIQUE AND INPUT IMAGE SIZE. IMPROVEMENTS ARE SHOWN IN GREEN, DETERIORATIONS IN RED. THE BEST RESULTS ARE HIGHLIGHTED IN BOLD. VALUES ARE AVERAGED OVER ARCHITECTURES AND DATASETS.

	EE	**EEdesc***	**EEasc***	**EEmix***	**EEunif***
TR-224	5.795	3.782	0.387	**-2.729**	3.220
TR-64	7.771	-0.776	-6.061	**-16.325**	-7.482
FT-224	6.645	**0.407**	4.959	4.177	3.143
FT-64	7.877	2.264	-1.511	**-3.111**	0.122

of only 5%. On the contrary, it is possible to observe the beneficial effect of pruning, which in addition to providing the previously discussed benefits, goes as far as reducing the number of operations by up to a maximum of 17.95% in the EEmix* case.

A similar behaviour can be observed in the inference latency, as shown by the results of the experiments summarised in Table VII. Adding more outputs and calculating their weighted sum is clearly a disadvantage from a time point of view. In percentage, this phenomenon is more pronounced in architectures with input sizes 64x64. The cut of unnecessary outputs implemented by the pruning operation inevitably benefits inference times, as the image input to the networks will on average have to travel through a portion of the architecture and not entirely. Fine-tuning scenarios tend to benefit less from the pruning step precisely because, as can be deduced from Table V and Table VI, on average, only the very last outputs, or none at all, are cut out. Consequently, although mitigated, the addition of early exits is only partially beneficial, as opposed to scenarios characterised by training from scratch.

C. Early Exits Analysis

An interesting aspect is certainly that of the quality of the classifiers obtained prior to the pruning operation. In fact, the proposed training, whether from scratch or fine-tuning, has the purpose of optimising the ensemble to be pruned, but has the secondary effect of computing weights that can be used directly by the classifiers obtained thanks to early exits. For each of the scenarios considered, for the networks with 4 exits, Table VIII presents the average percentage change in accuracy of each early output compared to the average performance achieved by single-output networks.

The first notable result is the presence of early exits performing better than the single output network. This occurs in every scenario except FINETUNE-224, managing to achieve average accuracy improvements of up to 9.96%. This confirms not only the effectiveness of AEP as an ensemble-based training technique, but also opens up the possibility of realising shallower and at the same time more performant architectures, particularly useful in mobile, edge, or distributed setups. Focusing on the TRAIN-64 and TRAIN-224 scenarios, it is evident that both the third and the fourth exits represent an advantageous alternative to the traditional single output regardless of the AEP configuration used. This result is perhaps the most significant. Indeed, it shows that sometimes good human intuitions, such as increasing the complexity of a model, do not necessarily translate into improvements in performance. On the contrary, better results can be achieved by optimising the learning process and exploiting various levels of abstraction of the input data, as it is done through the use of multiple outputs.

VI. CONCLUSION

This paper presented Anticipate, Ensemble and Prune (AEP), the early exits and ensemble-based technique for improving the performance of single-output artificial neural networks. The extensive series of experiments conducted shows how this approach can be applied in the context of image classification, achieving remarkable improvements in terms

TABLE VIII

COMPARISONS IN TERMS OF PERCENTAGE CHANGE OF ACCURACY BETWEEN THE INDIVIDUAL OUTPUTS OF THE MODELS BASED ON EARLY EXITS AND THEIR SINGLE-OUTPUT COUNTERPARTS. THE FOUR EXPERIMENT CONFIGURATIONS ARE DIVIDED BY TABLE ACCORDING TO TRAINING TECHNIQUE AND INPUT IMAGE SIZE. IMPROVEMENTS ARE SHOWN IN GREEN, DETERIORATIONS IN RED. BEST RESULTS ARE HIGHLIGHTED IN BOLD. VALUES ARE AVERAGED OVER ARCHITECTURES AND DATASETS.

TR-224	**exit1**	**exit2**	**exit3**	**exit4**
EEdesc	-30.155	-4.193	2.508	**2.793**
EEasc	-42.465	-12.538	1.171	**3.041**
EEmix	-31.434	-5.444	2.884	**3.115**
EEunif	-35.224	-8.517	1.942	**2.970**

TR-64	**exit1**	**exit2**	**exit3**	**exit4**
EEdesc	-10.432	**9.515**	9.442	7.739
EEasc	-25.613	-1.243	**4.999**	4.373
EEmix	-11.434	8.350	**9.957**	8.060
EEunif	-17.433	4.527	**7.027**	5.692

FT-224	**exit1**	**exit2**	**exit3**	**exit4**
EEdesc	-37.867	-13.969	-5.713	**-3.027**
EEasc	-48.667	-19.457	-5.634	**-0.528**
EEmix	-39.664	-16.120	-5.777	**-2.574**
EEunif	-41.946	-16.633	-5.089	**-1.066**

FT-64	**exit1**	**exit2**	**exit3**	**exit4**
EEdesc	-20.932	-2.276	**1.181**	0.649
EEasc	-32.504	-7.237	1.902	**2.773**
EEmix	-22.601	-3.186	**1.173**	0.874
EEunif	-25.106	-3.932	1.891	**1.915**

of accuracy, parametric complexity, number of operations and inference time. The experiments were also replicated in the context of fitting pre-trained networks using fine-tuning techniques, with excellent results, especially in the absence of high-resolution input images. It was also shown how AEP can optimise the performance of individual early exits, exceeding the accuracy of single-exit models even without using the ensemble technique. We believe this can be an important step towards optimising the efficiency of neural architectures, especially for mobile and edge scenarios. In a follow-up study, AEP will include an automatic optimisation process to find the best loss weights and a search process to find the best output weights, which is likely to lead to further improvements and potentially reveal undiscovered design patterns.

REFERENCES

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.

[2] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.*, "Imagenet large scale visual recognition challenge," *International journal of computer vision*, vol. 115, no. 3, pp. 211–252, 2015.

[3] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[4] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[5] 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.

[6] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708.

[7] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan *et al.*, "Searching for mobilenetv3," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 1314–1324.

[8] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *International conference on machine learning*. PMLR, 2019, pp. 6105–6114.

[9] Z. Liu, H. Mao, C.-Y. Wu, C. Feichtenhofer, T. Darrell, and S. Xie, "A convnet for the 2020s," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 11 976–11 986.

[10] Y. Matsubara, M. Levorato, and F. Restuccia, "Split computing and early exiting for deep learning applications: Survey and research challenges," *ACM Computing Surveys*, vol. 55, no. 5, pp. 1–30, 2022.

[11] M. Wołczyk, B. Wójcik, K. Bałazy, I. T. Podolak, J. Tabor, M. Śmieja, and T. Trzcinski, "Zero time waste: Recycling predictions in early exit neural networks," *Advances in Neural Information Processing Systems*, vol. 34, pp. 2516–2528, 2021.

[12] T. Sun, Y. Zhou, X. Liu, X. Zhang, H. Jiang, Z. Cao, X. Huang, and X. Qiu, "Early exiting with ensemble internal classifiers," *arXiv preprint arXiv:2105.13792*, 2021.

[13] A. Campbell, L. Qendro, P. Liò, and C. Mascolo, "Robust and efficient uncertainty aware biosignal classification via early exit ensembles," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022, pp. 3998–4002.

[14] P. Panda, A. Sengupta, and K. Roy, "Conditional deep learning for energy-efficient and enhanced pattern recognition," in *2016 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2016, pp. 475–480.

[15] S. Teerapittayanon, B. McDanel, and H.-T. Kung, "Branchynet: Fast inference via early exiting from deep neural networks," in *2016 23rd International Conference on Pattern Recognition (ICPR)*. IEEE, 2016, pp. 2464–2469.

[16] M. Wang, J. Mo, J. Lin, Z. Wang, and L. Du, "Dynexit: A dynamic early-exit strategy for deep residual networks," in *2019 IEEE International Workshop on Signal Processing Systems (SiPS)*. IEEE, 2019, pp. 178–183.

[17] R. G. Pacheco, K. Bochie, M. S. Gilbert, R. S. Couto, and M. E. M. Campista, "Towards edge computing using early-exit convolutional neural networks," *Information*, vol. 12, no. 10, p. 431, 2021.

[18] 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, 2020.

[19] L. Qendro and C. Mascolo, "Towards adversarial robustness with early exit ensembles," in *2022 44th Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC)*. IEEE, 2022, pp. 313–316.

[20] L. Qendro, A. Campbell, P. Lio, and C. Mascolo, "Early exit ensembles for uncertainty quantification," in *Machine Learning for Health*. PMLR, 2021, pp. 181–195.

[21] C.-Y. Lee, S. Xie, P. Gallagher, Z. Zhang, and Z. Tu, "Deeply-supervised nets," in *Artificial intelligence and statistics*. PMLR, 2015, pp. 562–570.

[22] Z. Fei, X. Yan, S. Wang, and Q. Tian, "Deecap: Dynamic early exiting for efficient image captioning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 12 216–12 226.

[23] H. Lee and J.-S. Lee, "Students are the best teacher: Exit-ensemble distillation with multi-exits," *arXiv preprint arXiv:2104.00299*, 2021.

[24] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.

[25] P. Helber, B. Bischke, A. Dengel, and D. Borth, "Eurosat: A novel dataset and deep learning benchmark for land use and land cover classification," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 12, no. 7, pp. 2217–2226, 2019.

[26] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," *arXiv preprint arXiv:1708.07747*, 2017.

[27] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel, "Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition," *Neural networks*, vol. 32, pp. 323–332, 2012.
[28] Y. Le and X. Yang, "Tiny imagenet visual recognition challenge," *CS 231N*, vol. 7, no. 7, p. 3, 2015.