Efficient Adaptive Inference Leveraging Bag-of-Features-based Early Exits

Nikolaos Passalis[1], Jenni Raitoharju[2], Moncef Gabbouj[3] and Anastasios Tefas[1]
[1]Department of Informatics, Aristotle University of Thessaloniki, Thessaloniki, Greece
[2]Programme for Environmental Information, Finnish Environment Institute, Jyväskylä, Finland
[3]Faculty of Information Technology and Communication Sciences, Tampere University, Tampere, Finland
E-mails: passalis@csd.auth.gr, jenni.raitoharju@tuni.fi, moncef.gabbouj@tuni.fi, tefas@csd.auth.gr

***Abstract*—Early exits provide an effective way of implementing adaptive computational graphs over deep learning models. In this way it is possible to adapt them on-the-fly to the available computational resources or even to the difficulty of each input sample, reducing the energy and computational power requirements in many embedded and mobile applications. However, performing this kind of adaptive inference also comes with several challenges, since the difficulty of each sample must be estimated and the most appropriate early exit must be selected. It is worth noting that existing approaches often lead to highly unbalanced distributions over the selected early exits, reducing the efficiency of the adaptive inference process. At the same time, only a few resources can be devoted to the aforementioned process, in order to ensure that an adequate speedup will be obtained. The main contribution of this work is to provide an easy to use and tune adaptive inference approach for early exits that can overcome some of these limitations. In this way, the proposed method allows for a) obtaining a more balanced inference distribution among the early exits, b) relying on a single and interpretable hyper-parameter for tuning its behavior (ranging from faster inference to higher accuracy), and c) improving the performance of the networks (increasing the accuracy and reducing the time needed for inference). Indeed, the effectiveness of the proposed method over existing approaches is demonstrated using four different image datasets.**

***Index Terms*—Early Exits, Adaptive Inference, Bag-of-Features, Adaptive Computational Graphs**

I. Introduction

Deep Learning (DL) led to a number of impressive applications, ranging from autonomous cars [1] and robotics [2] to accurate medical diagnosis and disease prognosis [3]. However, despite its success in these areas, state-of-the-art DL models are computationally intensive, requiring an enormous amount of resources in order to be successfully deployed in most embedded and mobile applications. These limitations led to the development of various methods for training lightweight DL models, ranging from quantization [4] and pruning methods [5] to knowledge distillation approaches [6], [7]. These methods were indeed capable of providing faster and more lightweight models. However, this was typically achieved at the expense of the accuracy of the final models, since lightweight models tend to perform worse than the larger ones. At the same time, the constructed models are unable to dynamically adapt to the difficulty of the input samples, leading to a constant cost for feed-forwarding through the network. However, it has been demonstrated that easier samples can be often classified correctly even with significantly smaller models [8], [9]. This hints at using models that can switch between using more layers/computational resources for harder examples, while keeping the computations to the minimum required for easier examples. In this way, it is possible to reduce the computational requirements and energy consumption, as well as improve inference speed, often without significantly affecting the accuracy of the model.

Indeed, models with *adaptive computational graphs*, such as [8], [9], [10], have been proposed to this end. These models provide an effective way to dynamically alter the number of computations to match the available computational resources by following a different path on the model's graph. Perhaps the most straightforward way to provide such adaptive models is by using *early exits* at various layers of the network [8], [9], [10]. In this way it is possible to estimate the final output of a model, without feed-forwarding through the whole computational graph. However, even though early exits proved to be a valuable tool for adapting the models to the available computational resources and different inference scenarios, e.g., using the same model across different mobile devices, using them for adapting to the difficulty of each sample is usually not straightforward. There are two main reasons for this: a) early exits do not provide a way for directly estimating the difficulty of an input sample, while b) designing and employing a meaningful exit strategy that utilizes all the available exits in the optimal way is non-trivial. In fact, as we also experimentally demonstrate, naive methods tend to use only a small number of the exits that are available, often skipping intermediate exits, leading to sub-optimal results and reducing inference speed and accuracy.

To better understand these limitations, we need to consider the way existing methods perform adaptive inference. Most of them estimate the difficulty of a sample indirectly, by measuring the *confidence* of the network at a specific early exit. This can be done either by measuring the strongest activation or by measuring the entropy of the output probability

This work was supported by the European Union's Horizon 2020 Research and Innovation Program (OpenDR) under Grant 871449. This publication reflects the authors' views only. The European Commission is not responsible for any use that may be made of the information it contains.

978-1-7281-9320-5/20/$31.00 ©2020 IEEE

distribution [8]. After that, they employ a *fixed* threshold at each early exit to decide whether the specific exit should be used or not. When the confidence level is above a certain threshold, the computation ends at the specific early exit. It is also worth noting that even though more sophisticated methods can be developed for estimating the difficulty of each sample and/or the confidence of the network, it is critical to keep the required computations to the minimum, since spending a significant amount of time deciding whether the computations should end at a specific exit, might end up consuming more time than feed-forwarding through the whole network.

At the same time, the aforementioned process typically involves manually fine-tuning these thresholds for each early exit in order to achieve specific inference goals, either regarding the speed or the accuracy of the network. However, these thresholds are typically application specific and there is no easy way to select and interpret them in order to tune the behavior of the network. Finally, it is worth noting that sub-optimally tuning these thresholds, e.g., by simply selecting the average confidence of a layer, often leads to a significantly skewed exit distribution, since the confidence of the network is typically not normally distributed. This behavior is indeed confirmed in Section III.

The main contribution of this work is to provide an easy to use and tune adaptive inference approach for early exits. To this end, the proposed method combines several methodological advances to overcome a number of limitations of existing methods. First, instead of using naive early exits that discard a significant amount of information regarding the input distribution [9], we employ a powerful Bag-of-Features (BoF)-based early exits strategy. This, allows us to provide compact and normalized histogram-based representations of the features extracted from each layer, where an early exit is placed, which, in turns, enables the efficient estimation of the difficulty of each sample. Then, a fast and robust way to estimate the confidence of the network at each early exit is employed and combined with an easy to use, yet effective methodology for tuning the behavior of the method according to different inference scenarios. In this way, the proposed method allows for a) obtaining a more balanced inference distribution among the early exits, b) relying on a single and interpretable hyper-parameter for tuning its behavior (ranging from faster inference to higher accuracy), and c) improving the performance of the networks (increase the accuracy and require less time for inference). To the best of our knowledge, this paper presents the first experimental study that demonstrates the importance of employing methods that lead to a balanced exit distribution, while also exploiting these observations to provide a dynamic inference scheme for BoF-based exits that can be easily adjusted to the needs of each application. The effectiveness of the proposed approach over existing approaches is demonstrated using four different image datasets.

The rest of the paper is structured as follows. The proposed method is introduced in Section II, while the experimental evaluation is provided in Section III. Finally, conclusions are drawn in Section IV.

II. Proposed Method

The proposed method is presented in this Section. First, the notation and the employed early exit strategy are described. Then, the proposed adaptive inference approach is introduced and discussed. The notation $f_{\mathbf{W}}(\mathbf{x}, i)$ is used to refer to the response of the i-th layer of a neural network that is composed of a total of m layers. The trainable parameters of the network are denoted by $\mathbf{W}$, while the input to the neural network is denoted by $\mathbf{x}$. In this paper, we focus on convolutional neural networks. Therefore, the input to the network is an image $\mathbf{x} \in \mathbb{R}^{W \times H \times C}$, where its width is denoted by W, its height by H, and the number of channels by C. It is worth noting that the proposed method can be also applied for networks operating on different kinds of signals. Furthermore, let $\mathbf{y}^{(i)} = f_{\mathbf{W}}(\mathbf{x}, i) \in \mathbb{R}^{W_i \times H_i \times C_i}$ denote the output of the i-th convolutional layer. Similarly, the notation W_i, H_i and C_i is used to refer to the width, height and number of channels of the corresponding feature map. Finally, the output of the network, which estimates the probability of each sample belonging to a different class (out of a total of N_C classes), is denoted by $\mathbf{y} = f_{\mathbf{W}}(\mathbf{x}, m) \in \mathbb{R}^{N_C}$.

Also, let $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$ denote a training set of N images. Each image is also annotated by a target (ground truth annotation) vector $\mathbf{t}_i \in \mathbb{R}^{N_C}$. The network can be then trained using back-propagation to minimize a loss function $\mathcal{L}$:

$$\mathbf{W}' = \mathbf{W} - \eta \sum_{j=1}^{N} \frac{\partial \mathcal{L}(f_{\mathbf{W}}(\mathbf{x}_j, m), \mathbf{t}_j)}{\partial \mathbf{W}}. \tag{1}$$

The notation $\mathbf{W}'$ is used to refer to the parameters of the network after an update, while η denotes the learning rate. For this paper, the cross-entropy loss function is used:

$$\mathcal{L}(\mathbf{y}, \mathbf{t}) = -\sum_{i=1}^{N_C} [\mathbf{t}]_i \log([\mathbf{y}]_i), \tag{2}$$

where the notation $[\mathbf{y}]_i$ is used to refer to the i-th element of a vector $\mathbf{y}$.

In order to efficiently estimate the output of the network at various points of its computational graph, we employ an additional estimator $g^{(i)}_{\mathbf{W}_i}(\cdot)$ on top of the feature maps extracted at the i-th layer as:

$$g^{(i)}_{\mathbf{W}_i}\left(\mathbf{y}^{(i)}\right) = g^{(i)}_{\mathbf{W}_i}(f_{\mathbf{W}}(\mathbf{x}, i)). \tag{3}$$

In this way, it is possible to estimate the final output of the network, without having to feed-forward through the whole architecture. We use the notation $\mathbf{W}_i$ to refer to the parameters of the employed early exit. Typically, an early exit is composed of a feature aggregation approach, e.g., Global Average Pooling, followed by a fully connected classification layer. Note that the effectiveness and efficiency of the feature aggregation approach is crucial for the successful deployment of the resulting network. For example, the feature maps extracted from the first layers are typically very large and close to the

size of the input image, requiring a way to efficiently compress them into a compact representation that can be used to rapidly take a classification decision.

To overcome this limitation, in this work we employ a Bag-of-Features (BoF)-based aggregation approach [9]. Therefore, each feature vector extracted from each spatial location of feature map is first quantized using a set of N_K codewords, each one denoted by $\mathbf{v}_{ij}$, where i refers to the layer on which the BoF layer is used and j to the codewords. The codewords are used to represent the prototypical concepts captured by the corresponding layer [11], while a different set of codewords is used for different exits. The membership vector for each feature vector extracted from the i-th early exit is calculated as:

$$[\mathbf{u}_{ikl}]_j = \frac{K([\mathbf{y}^{(i)}]_{kl}, \mathbf{v}_{ij})}{\sum_{m=1}^{N_K} K([\mathbf{y}^{(i)}]_{kl}, \mathbf{v}_{im})} \in [0,1], \quad (4)$$

where (k,l) is the location from which the feature vector was extracted and $K(\cdot)$ is a kernel function that measures the similarity between a codeword and an input vector. Then, the membership vectors $\mathbf{u}_{ikl}$ are aggregated, leading to the final constant length histogram representation of the i-th object as:

$$\mathbf{s}^{(i)} = \frac{1}{W_i H_i} \sum_{k=1}^{W_i} \sum_{l=1}^{H_i} \mathbf{u}_{ikl} \in \mathbb{R}^{N_K}, \quad (5)$$

Note that this histogram vector provides a compact semantic summary of the features extracted from the corresponding layer, allowing for efficiently performing classification tasks, regardless of the actual size of the input feature map.

Furthermore, BoF can be efficiently implemented in DL models by measuring the similarity between each codeword and each feature vector using a hyperbolic (sigmoid) kernel:

$$K([\mathbf{y}^{(i)}]_{kl}, \mathbf{v}_{ij}) = \frac{1}{2}\left(\tanh(c_1 [\mathbf{y}^{(i)}]_{kl}^T \mathbf{v}_{ij} + c_2) + 1\right) \quad (6)$$

where $\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$, and c_1 and c_2 are the kernel parameters (typically set to $c_1 = 1$ and $c_2 = 0$). Then, BoF can be formulated as an inner product-based layer, followed by a recurrent accumulator [12]. The representation extracted from each early exit is then fed into a fully connected layer to obtain the final output of the i-th early exit. Each early exit is trained by minimizing the same loss as the main network:

$$\mathbf{W}_i' = \mathbf{W}_i - \eta \sum_{j=1}^{N} \frac{\partial \mathcal{L}\left(g_{\mathbf{W}_i}^{(i)}(\mathbf{y}_j), \mathbf{t}_j\right)}{\partial \mathbf{W}_i}. \quad (7)$$

Also, early exits can be either trained at the same time with the main network or separately, after first fixing the weights of the main network [9]. However, back-propagating gradients from the early exits to the main network can potentially negatively affect its performance in some cases by forcing the earlier layers to be more discriminative, harming in this way the granularity of the analysis performed by the DL model. Therefore, even though both approaches can be used, in this paper we follow the second approach by keeping the parameters of the main network frozen when training the early exits.

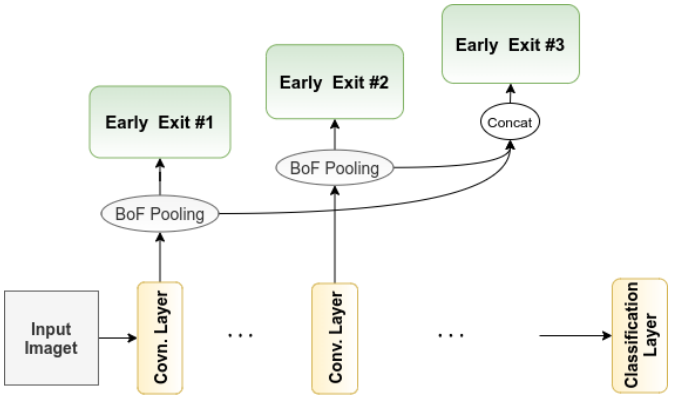

Fig. 1: Hierarchical Early Exits using Bag-of-Features. Note that any intermediate early exit can be selected to reduce the computational complexity of the inference process, given that the network is confident enough for the category of the current input sample.

The representations extracted from the previous layers can be also readily re-used by concatenating them with the current representation, building in this way a hierarchical inference structure, as shown in Fig. 1. Therefore, the representation $\mathbf{s}^{(i,h)}$ extracted from the i-th layer is calculated as:

$$\mathbf{s}^{(i,h)} = \begin{cases} \mathbf{s}^{(i)} & \text{if } i = 1 \\ \mathbf{s}^{(i)} \frown \mathbf{s}^{(i-1,h)} & \text{if } i > 1 \end{cases}, \quad (8)$$

where $\mathbf{a} \frown \mathbf{b}$ denotes the concatenation of vectors $\mathbf{a}$ and $\mathbf{b}$. This approach allows for increasing the classification accuracy, with virtually zero additional cost, since the representations extracted from the previous layers can be readily cached and re-used.

In order to perform adaptive inference using a network equipped with early exits, an appropriate criterion must be used to decide whether inference should stop at a specific exit or continue until the next one. To this end, the difficulty of the input sample must be first estimated. In this work, we employ a simple, yet robust approach: the uncertainty of the network/difficulty $r(\mathbf{x}, i)$ of an input sample $\mathbf{x}$, given the i-th exit, is estimated based on the confidence of the network on the class that corresponds to the neuron with the highest activation:

$$r(\mathbf{x}, i) = [g_{\mathbf{W}_i}^{(i)}(\mathbf{y}^{(i)})]_k, \quad (9)$$

where

$$k = \arg_{k'} \max [g_{\mathbf{W}_i}^{(i)}(\mathbf{y}^{(i)})]_{k'}. \quad (10)$$

Then, the most straightforward approach is to calculate a threshold for the i-th early exit based on the mean activation of the winning neurons during inference:

$$\mu_i = \frac{1}{N} \sum_{j=1}^{N} [g_{\mathbf{W}_i}^{(i)}(\mathbf{y}_j)]_k, \quad (11)$$

where $g_{\mathbf{W}_i}^{(i)}$ is the output of the i-th early exit and again $k = \arg\max g_{\mathbf{W}_i}^{(i)}(\mathbf{y}_j)$. Then, the inference process can stop at the i-th exit whenever the confidence is higher than this threshold, i.e.,

$$r(\mathbf{x}, i) > \alpha\mu_i, \tag{12}$$

where α is a hyper-parameter that allows for adapting the behavior of the inference process to the needs of each application. That is, using larger values for α enables us to get more accurate result (at the expense of higher inference time, since the corresponding early exit will be chosen less frequently), while using smaller values allows for increasing the speed, but possibly leading to less accurate classification results.

Note that the value of α is not bounded, since any positive number can be used. We have experimentally found out that this approach is also especially sensitive, since even small changes in the value of α can lead to significant changes in the behavior of the inference process, as we also experimentally demonstrate in Section III. Furthermore, this behavior seems to be also related to the specific dataset/network used for training, with some datasets/networks requiring vastly different values compared to others. To overcome this limitation, we propose calculating the threshold based both on the mean activation of the current layer μ_i (lower bound), as well as on the last layer of the network μ_C (upper bound). Therefore, the threshold for selecting a specific early exit is calculated as:

$$r(\mathbf{x}, i) > (1 - \alpha)\mu_i + \alpha\mu_C, \tag{13}$$

for $\alpha \in [0, 1]$. In this way, α is bounded between 0 and 1, while its value can be easily interpreted: as the value of α increases from 0 to 1, less samples are using the current exit. Indeed, as demonstrated in Section III, the proposed way for calculating the threshold allows for acquiring significantly better performance, avoiding collapse phenomena where only a few of the early exits are actually used.

III. Experimental Evaluation

The experimental evaluation of the proposed approach is provided in this Section. Four datasets were used for evaluating the proposed method: a) the MNIST image classification dataset [13], b) the Fashion MNIST fashion product classification dataset [14], as well as the more challenging c) CIFAR-10 object recognition dataset [15], and d) FER-2013 facial expression dataset [16].

We used three different convolutional neural network architectures for the conducted experiments:

- **CNN-1**, which was used for the MNIST dataset,
- **CNN-2**, which was used for the Fashion MNIST dataset,
- **MobileNet v.2** [17], which was used for the CIFAR-10 and FER-2013 datasets.

CNN-1 employs a 3×3 convolution layer with 32 filters, which is followed by 2×2 max pooling layer, a 3×3 convolution layer with 64 filters, another 2×2 max pooling layer, a fully connected layer with 1024 neurons, dropout with rate $p = 0.5$, and a final fully connected classification layer. CNN-2 uses a 3×3 convolution layer with 64 filters, which is followed by a 2×2 max pooling layer, a 3×3 convolution layer with 128 filters, another 2×2 max pooling layer, a fully connected layer with 1024 neurons layer, dropout with rate $p = 0.5$, and a final fully connected classification layer. All the layers (except from the final one) employ the ReLU activation function. For the MobileNet the filter size for the first convolutional layer was appropriately tuned, according to the image size of different datasets.

TABLE I: Classification error and MFLOPs for the backbone networks (without using any early exit)

Dataset	Error	MFLOPs
MNIST	0.68	4.10
Fashion MNIST	7.82	12.66
CIFAR-10	7.81	94.61
FER-2013	38.84	211.52

All networks were trained using the categorical cross-entropy loss. The Adam algorithm was used for the optimization using the default hyper-parameters [18]. The optimization ran for 50 epochs for the CNN-1 and CNN-2 models (a learning rate of 0.001 was used). On the other hand, MobileNet models were trained for 100 epochs (50 epochs with a learning rate of 0.001 and the remaining ones using a reduced learning rate of 0.0001). Two early exits and one hierarchical exit (combining the previous two ones) were used. For the CNN-1 and CNN-2 models, the first early exit was placed after the first convolutional layer, while the second one was placed after the second convolutional layer. The 5th and 7th convolutional layers were used for the MobileNet architecture. Finally, for the MNIST and FashionMNIST datasets, a spatial segmentation scheme into 4 regions was used for the BoF model employed for the early exits [11].

The proposed method is evaluated in Table II. Four different inference settings were used, ranging from faster, yet less accurate settings 1 and 2 to the slower, yet more accurate settings 3 and 4. These settings correspond to altering the value of the hyper-parameter α used to calculate the inference threshold. For the baseline method, as described by (12), the value of α was set to 1, 1.05, 1.07, and 1.095 for the four different settings. There values were obtained after carefully fine-tuning the upper limit in order to not collapse the inference process into using only some of the employed early exits. For the proposed method, the corresponding values were set to 0, 0.5, 0.7, and 0.95. Note that selecting these values is much easier for the proposed method, since α always ranges between 0 and 1, while this value is not expected to be dataset-specific, compared to the baseline one. Also, two different BoF settings were used: BoF-1 and BoF-2. For the MNIST and FashionMNIST datasets, 8 codewords were used for each early exit, while for BoF-2 16 codewords were used (32 for the MNIST dataset). Also, 16/32 codewords were used for BoF-1/BoF-2 for the experiments involving the MobileNet.

Several interesting conclusions can be drawn based on the results reported in Table II. Note that for the first setting, both methods provide the same results, since setting $\alpha = 1$ for

TABLE II: Classification error (%) and MFLOPs for different adaptive inference settings

Dataset	Method	Setting 1		Setting 2		Setting 3		Setting 4		Class. Cost
		Error (%)	MFLOPs	Error (%)	MFLOPs	Error (%)	MFLOPs	Error (%)	MFLOPs	
MNIST										
Baseline	BoF-1	1.32	0.69 ± 1.03	0.89	1.17 ± 1.65	0.79	1.54 ± 1.83	0.68	4.16 ± 0.00	4.0449
Proposed	BoF-1	1.32	0.69 ± 1.03	0.97	0.86 ± 1.20	0.82	1.02 ± 1.31	0.70	1.68 ± 1.55	2.3989
Baseline	BoF-2	1.09	0.68 ± 0.83	0.68	4.34 ± 0.00	0.68	4.34 ± 0.00	0.68	4.34 ± 0.00	10.5819
Proposed	BoF-2	1.09	0.68 ± 0.83	0.95	0.77 ± 0.94	0.86	0.85 ± 1.02	0.76	1.22 ± 1.25	4.2937
Fashion MNIST										
Baseline	BoF-1	9.54	3.62 ± 4.94	9.26	4.10 ± 5.21	9.12	4.31 ± 5.32	8.83	4.62 ± 5.45	10.4730
Proposed	BoF-1	9.54	3.62 ± 4.94	8.55	5.01 ± 5.52	8.26	5.92 ± 5.70	7.84	8.29 ± 5.49	4.8555
Baseline	BoF-2	8.95	3.51 ± 4.85	8.59	4.05 ± 5.16	8.42	4.28 ± 5.28	8.27	4.64 ± 5.46	8.7200
Proposed	BoF-2	8.95	3.51 ± 4.85	8.29	4.63 ± 5.38	8.10	5.34 ± 5.58	7.84	7.34 ± 5.70	6.6399
CIFAR-10										
Baseline	BoF-1	10.24	53.17 ± 16.49	9.52	55.59 ± 18.90	9.21	58.24 ± 21.52	9.05	61.45 ± 23.83	61.7983
Proposed	BoF-1	10.23	53.18 ± 16.50	9.06	56.93 ± 18.78	8.55	59.32 ± 19.76	7.87	65.71 ± 20.66	37.2700
Baseline	BoF-2	9.07	52.72 ± 16.32	8.52	58.93 ± 22.81	8.42	59.78 ± 23.20	8.21	61.09 ± 23.71	90.0544
Proposed	BoF-2	9.06	52.74 ± 16.34	8.24	55.59 ± 18.26	8.07	57.50 ± 19.19	7.78	62.06 ± 20.41	58.1173
FER-2013										
Baseline	BoF-1	43.58	131.27 ± 45.54	42.94	135.89 ± 47.59	42.57	137.69 ± 48.33	42.30	139.97 ± 49.09	152.8386
Proposed	BoF-1	43.58	131.27 ± 45.54	38.87	172.93 ± 47.77	38.67	183.43 ± 43.72	38.95	199.48 ± 31.20	39.0866
Baseline	BoF-2	41.74	131.74 ± 45.49	41.46	136.34 ± 47.42	41.18	138.33 ± 48.22	40.87	140.27 ± 48.80	299.9848
Proposed	BoF-2	41.74	131.74 ± 45.49	38.79	167.02 ± 48.87	38.84	178.49 ± 46.01	38.59	195.66 ± 35.08	60.0961

TABLE III: Distribution over the early exits for different inference settings

Dataset	Method	Setting 1	Setting 2	Setting 3	Setting 4	Unused/Least Used Exits
MNIST						
Baseline	BoF-1	0.84 - 0.10 - 0.02 - 0.04	0.77 - **0.00** - **0.00** - 0.23	0.67 - **0.00** - **0.00** - 0.33	**0.00** - **0.00** - **0.00** - 1.00	7
Proposed	BoF-1	0.84 - 0.10 - 0.02 - 0.04	0.78 - 0.12 - 0.02 - 0.07	0.73 - 0.15 - 0.03 - 0.09	0.51 - 0.23 - 0.08 - 0.19	**0**
Baseline	BoF-2	0.89 - 0.08 - **0.01** - 0.02	**0.00** - **0.00** - **0.00** - 1.00	**0.00** - **0.00** - **0.00** - 1.00	**0.00** - **0.00** - **0.00** - 1.00	10
Proposed	BoF-2	0.89 - 0.08 - **0.01** - 0.02	0.85 - 0.10 - **0.01** - 0.03	0.83 - 0.12 - 0.02 - 0.03	0.68 - 0.23 - 0.03 - 0.06	**2**
Fashion MNIST						
Baseline	BoF-1	0.71 - 0.11 - 0.02 - 0.16	0.67 - 0.11 - 0.02 - 0.20	0.66 - 0.11 - **0.01** - 0.21	0.63 - 0.11 - **0.01** - 0.24	2
Proposed	BoF-1	0.71 - 0.11 - 0.02 - 0.16	0.59 - 0.12 - 0.03 - 0.26	0.52 - 0.12 - 0.03 - 0.33	0.32 - 0.11 - 0.05 - 0.52	**0**
Baseline	BoF-2	0.73 - 0.11 - 0.02 - 0.14	0.69 - 0.11 - 0.02 - 0.19	0.67 - 0.11 - 0.02 - 0.21	0.64 - 0.11 - **0.01** - 0.24	1
Proposed	BoF-2	0.73 - 0.11 - 0.02 - 0.14	0.63 - 0.12 - 0.03 - 0.22	0.57 - 0.12 - 0.03 - 0.27	0.41 - 0.11 - 0.05 - 0.43	**0**
CIFAR-10						
Baseline	BoF-1	0.73 - 0.15 - **0.01** - 0.11	0.70 - 0.13 - **0.00** - 0.17	0.68 - 0.07 - **0.00** - 0.25	0.66 - **0.00** - **0.00** - 0.34	5
Proposed	BoF-1	0.73 - 0.15 - **0.01** - 0.11	0.63 - 0.19 - **0.01** - 0.17	0.57 - 0.22 - **0.01** - 0.21	0.39 - 0.30 - 0.02 - 0.30	**3**
Baseline	BoF-2	0.76 - 0.13 - **0.01** - 0.11	0.71 - **0.00** - **0.00** - 0.29	0.70 - **0.00** - **0.00** - 0.30	0.67 - **0.00** - **0.00** - 0.33	7
Proposed	BoF-2	0.76 - 0.13 - **0.01** - 0.11	0.68 - 0.16 - **0.01** - 0.15	0.62 - 0.19 - **0.01** - 0.18	0.49 - 0.25 - **0.01** - 0.24	**4**
FER-2013						
Baseline	BoF-1	0.60 - 0.17 - **0.01** - 0.22	0.56 - 0.17 - **0.01** - 0.26	0.54 - 0.17 - **0.01** - 0.27	0.52 - 0.17 - **0.01** - 0.29	4
Proposed	BoF-1	0.60 - 0.17 - **0.01** - 0.22	0.22 - 0.18 - 0.02 - 0.58	0.15 - 0.15 - 0.02 - 0.68	0.05 - 0.08 - **0.01** - 0.85	**2**
Baseline	BoF-2	0.59 - 0.18 - **0.01** - 0.22	0.55 - 0.18 - **0.01** - 0.25	0.53 - 0.18 - **0.01** - 0.27	0.52 - 0.18 - 0.02 - 0.29	3
Proposed	BoF-2	0.59 - 0.18 - **0.01** - 0.22	0.26 - 0.21 - 0.02 - 0.51	0.18 - 0.17 - 0.02 - 0.63	0.07 - 0.11 - **0.01** - 0.80	**2**

The percentage of the samples that use each of the 4 available exits is reported for each different setting. The first number corresponds to the first early exit, the second number to the second early exit, the third number to the hierarchical exit, while the last number to the final exit of the network.

the baseline is equivalent to setting $\alpha = 0$ for the proposed method. First, note that all methods lead to a significant reduction in the number of required FLOPs compared to the full networks, as reported in Table I. At the same time, the proposed method allows for a more fine-grained control over the behavior of the proposed method, in virtually every case, as demonstrated by the number of FLOPs for different settings. For example, note that for MNIST dataset (BoF-2), baseline collapses to using the final exit even for the second setting. At the same time, the proposed method does not lead to this behavior in any of the evaluated cases, even for the last setting. These results are also further validated in Table III, where the distribution of the used early exits are reported. The proposed method leads to a smoother distribution, avoiding phenomena where an early exit is rarely selected. Indeed, this can be easily verified by measuring the number of cases where an exit is selected for less than 1% of the time (last column of Table III).

Classification cost, which is calculated by dividing the mean number of FLOPs by the absolute classification error improvement over the first setting, is also reported in Table II. The mean classification cost calculated using the three remaining settings (2 to 4) is reported. Note that smaller values indicate better utilization of the resources, since the cost essentially measures the number of FLOPs required for each percentage of accuracy improvement. Again, the ability of the proposed method to significantly improve the inference results is verified, with some cases, e.g., FER-2013 dataset, leading to an almost 5-fold increase in efficiency.

Finally, note that, in many cases, the employed hierarchical exit (third exit) is only rarely selected, as shown in Table III. This is expected, since it generally provides small accuracy improvements over the second early exit, as also noted in [9]. This could possibly hint into designing architectures that only employ early exits on selected paths of the inference graph, allowing for avoiding unnecessary calculations that do not contribute to increasing the overall performance of the network. It is also worth noting that the proposed method allows for identifying such exits, since it leads to a smoother distribution over the used exits. Therefore, any exit that is below a certain threshold can be, in most of the cases, safely discarded, providing an efficient way for selecting the most appropriate points of the computational graph to add early exits.

IV. Conclusions

In this paper, we presented an easy to use and tune, yet effective adaptive inference approach for networks equipped with BoF-based early exits. The proposed method employs a fast confidence estimation approach, along with a bounded way to calculate the exit thresholds for the various exits. As it was experimentally demonstrated, this allows for obtaining a more balanced inference distribution among the early exits increasing the accuracy and requiring less time for inference. At the same time, the proposed method paves the way for more advanced adaptive inference approaches for models equipped with early exits. For example, the proposed method provides an implicit way to estimate the layers on which early exits should be placed, avoiding the need for more expensive neural architecture search methods, more advanced approaches for estimating the confidence for each sample, while neural network distillation methods [19] can be employed to further improve the performance of the proposed method.

References

[1] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436, 2015.

[2] Niko Sünderhauf, Oliver Brock, Walter Scheirer, Raia Hadsell, Dieter Fox, Jürgen Leitner, Ben Upcroft, Pieter Abbeel, Wolfram Burgard, Michael Milford, et al., "The limits and potentials of deep learning for robotics," *The International Journal of Robotics Research*, vol. 37, no. 4-5, pp. 405–420, 2018.

[3] Baris Gecer, Selim Aksoy, Ezgi Mercan, Linda G Shapiro, Donald L Weaver, and Joann G Elmore, "Detection and classification of cancer in whole slide breast histopathology images using deep convolutional networks," *Pattern Recognition*, vol. 84, pp. 345–356, 2018.

[4] Song Han, Huizi Mao, and William J Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," in *Proceedings of the International Conference on Learning Representations*, 2016.

[5] Jian-Hao Luo, Jianxin Wu, and Weiyao Lin, "Thinet: A filter level pruning method for deep neural network compression," in *Proc. IEEE International Conference on Computer Vision*, 2017, pp. 5058–5066.

[6] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[7] Nikolaos Passalis and Anastasios Tefas, "Learning deep representations with probabilistic knowledge transfer," in *Proc. European Conference on Computer Vision*, 2018, pp. 268–284.

[8] Surat Teerapittayanon, Bradley McDanel, and Hsiang-Tsung Kung, "Branchynet: Fast inference via early exiting from deep neural networks," in *Proc. International Conference on Pattern Recognition*, 2016, pp. 2464–2469.

[9] Nikolaos Passalis, Jenni Raitoharju, Anastasios Tefas, and Moncef Gabbouj, "Adaptive inference using hierarchical convolutional bag-of-features for low-power embedded platforms," in *Proc. IEEE International Conference on Image Processing*, 2019, pp. 3048–3052.

[10] Yue Bai, Shuvra S Bhattacharyya, Antti P Happonen, and Heikki Huttunen, "Elastic neural networks: A scalable framework for embedded computer vision," in *Proceedings of the European Signal Processing Conference*, 2018, pp. 1472–1476.

[11] Nikolaos Passalis and Anastasios Tefas, "Learning bag-of-features pooling for deep convolutional neural networks," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 5766–5774.

[12] Nikolaos Passalis, Anastasios Tefas, Juho Kanniainen, Moncef Gabbouj, and Alexandros Iosifidis, "Temporal logistic neural bag-of-features for financial time series forecasting leveraging limit order book data," *arXiv preprint 1901.08280 (2019)*.

[13] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[14] Han Xiao, Kashif Rasul, and Roland Vollgraf, "Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms," *arXiv preprint 1708.07747 (2017)*.

[15] Alex Krizhevsky, "Learning multiple layers of features from tiny images," Tech. Rep., University of Toronto, 2009.

[16] Octavio Arriaga, Matias Valdenegro-Toro, and Paul Plöger, "Real-time convolutional neural networks for emotion and gender classification," *arXiv preprint 1710.07557 (2017)*.

[17] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4510–4520.

[18] Diederik P Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[19] Nikolaos Passalis, Maria Tzelepi, and Anastasios Tefas, "Heterogeneous knowledge distillation using information flow modeling," in *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 2339–2348.