AdaEE: Adaptive Early-Exit DNN Inference Through Multi-Armed Bandits

Roberto G. Pacheco*, Mark Shifrin†, Rodrigo S. Couto*, Daniel S. Menasché*,
Manjesh K. Hanawal‡ and Miguel Elias M. Campista*
* Universidade Federal do Rio de Janeiro (Brazil), † Ben-Gurion University (Israel),
‡ Indian Institute of Technology Bombay (India)
Email: {pacheco, rodrigo, miguel}@gta.ufrj.br, markshi@post.bgu.ac.il, sadoc@dcc.ufrj.br, mhanawal@iitb.ac.in

***Abstract*—Deep Neural Networks (DNNs) are widely used to solve a growing number of tasks, such as image classification. However, their deployment at resource-constrained devices still poses challenges related to energy consumption and delay overheads. Early-Exit DNNs (EE-DNNs) address the challenges by adding side branches through their architecture. Under an edge-cloud co-inference, if the confidence at a side branch is larger than a fixed confidence threshold, the inference is performed completely at the edge device, saving computation for more difficult observations. Otherwise, the edge device offloads the inference task to the cloud, incurring overhead. Despite its success, EE-DNNs for image classification have to cope with distorted images. The baseline distortion level depends on the environmental context, e.g., time of the day, lighting, and weather conditions. To cope with varying distortion, we propose Adaptive Early-Exit in Deep Neural Networks (AdaEE), a novel algorithm to dynamically adjust the confidence threshold based on context, leveraging the Upper Confidence Bound (UCB) for that matter. AdaEE provably achieves logarithmic regret under mild conditions. We experimentally verify that 1) convergence occurs after collecting a few thousand observations for images with different distortion levels and overhead values, and 2) AdaEE obtains a lower cumulative regret when compared against alternatives using the Caltech-256 dataset subject to varying distortion.**[1]

I. Introduction

Deep Neural Networks (DNNs) have presented significant advances in their performance [1], [2], especially for computer vision tasks, such as image classification. However, DNNs require high processing power, which may prevent their use for inference on mobile devices. One alternative to overcome these constraints is offloading inference tasks to a cloud computing infrastructure equipped with Graphical Processing Units (GPUs) [3]. In a cloud-only DNN application, the mobile devices gather data and offload it to the cloud to run the entire DNN model. However, this scenario may introduce an overhead due to excessive communication delay between mobile devices and a cloud server. To circumvent this issue, edge computing emerges by allocating computing resources closer to mobile devices [3], motivating DNN model partitioning to perform edge-cloud co-inference [4]–[6]. Under DNN partitioning, edge devices run the layers before the partitioning layer while cloud servers process the remaining layers.

This paper considers a classification task in which the DNN labels objects present in an image. We consider an adaptive offloading scenario where early-exit DNNs (EE-DNNs) are combined with model partitioning under an edge-cloud co-inference framework [7]–[9]. EE-DNNs are designed with side branches (i.e., intermediary classifiers) inserted into intermediary layers (Fig. 1). The idea behind these EE-DNNs is that the features extracted by the first convolutional layers can provide sufficiently confident predictions.

When the edge device gathers an image, it is processed until a side branch. If the classification confidence is greater than a threshold, inference ends. Otherwise, the edge device offloads data to the cloud that runs the remaining layers. Hence, the inference task is conditionally concluded based on the input classification difficulty, saving network resources, computation and energy, and reducing inference time for easier inputs. However, early classification may entail sacrificing confidence, leading to a latency-confidence tradeoff.

Previous works, such as BranchyNet [7] and SPINN [8], use a fixed threshold to decide whether to terminate an inference. These proposals implicitly assume that input images always are pristine (i.e., undistorted), which may not bode well with real-world applications. For example, an edge device can gather images with several distortion types, such as blur [10] and different distortion levels. Dodge and Karam [11] show that DNN models are sensitive to image distortion, which severely reduces their performance in terms of accuracy. Moreover, Pacheco *et al.* [12] experimentally demonstrate that, in EE-DNNs, side branches' offloading decisions are significantly impacted by image distortions. Thus, the distortion presented in an image, which we refer to as context, influences inference outcomes. Indeed, our experiments show that context significantly impacts the probability of classifying at the side branches and overall accuracy.

Our key insight consists in noting that the distribution of confidence estimates changes as a function of context. Then, the threshold determining if a given confidence is large enough to cause an early exit should be adapted accordingly. Our key contribution is Adaptive Early-Exit in Neural Networks (AdaEE) based on multi-armed bandits (MABs), an reinforcement learning algorithm. AdaEE learns the optimal

[1]©2023 IEEE. Personal use of this material is permitted. Permission from IEEE must obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

confidence threshold by employing MABs, accounting for i) known confidence at side branches, ii) envisioned confidence gains when using all layers, and iii) the corresponding overhead. Although MABs have already been considered in EE-DNNs [13]–[15], our proposal makes the early-exit decision on-the-fly using the confidence information provided by side branches. Our evaluation suggests that leveraging confidence levels at intermediary layers, which are readily available at decision time, is crucial for adaptively adjusting to context.

This paper is organized as follows. Section II reviews related work. Section III explains the concepts of EE-DNNs through MABs, and Section IV presents our problem formulation and the proposed algorithm. Next, Section V analyzes the impact of context on early-exit decisions and evaluates AdaEE in choosing the optimal threshold. Finally, Section VI concludes this paper and indicates future steps.

II. Related Work

Next, we discuss related work on early-exit DNNs and the use of multi-armed bandits for these DNNs.

A. Early-Exit in Deep Neural Networks

BranchyNet [7] is an early-exit DNN (EE-DNN) that decides whether a sample can be classified earlier on a side branch, using the classification entropy. SPINN [8] and SEE [16] make the decision to exit at a side branch based on the estimated classification confidence given by the probability of the most likely class. Besides BranchyNet and SPINN, other works also employ EE-DNNs to reduce inference time. FlexDNN [17] and Edgent [18] use EE-DNNs to select the most appropriate DNN depth.

Some works focus on deploying EE-DNNs in hardware. Dynexit [9] trains and deploys an EE-DNN on Field-Programmable Gate Array (FPGA) hardware. Paul *et al.* [19] show that implementing an early-exit DNN on the FPGA board can reduce inference time and energy consumption.

Pacheco *et al.* [20] combine EE-DNN and DNN partitioning to offload mobile devices via early-exit DNNs. This offloading scenario is also considered in [12], which proposes a robust EE-DNN against image distortion. Similarly, EPNet [21] learns when to exit early, accounting for the tradeoff between overhead and accuracy, but the learning occurs offline.

All above works assume a fixed [12], [17]–[20] application-defined threshold. Instead, we propose an adaptive exit policy that sets the threshold using reinforcement learning according to context. In this paper, we consider EE-DNNs where the exit at a side branch occurs whenever the classification confidence is above a threshold learned in an online fashion.

B. Multi-Armed Bandits in Early-Exit DNNs

Unlike previous works that employ a fixed threshold, LEE [13], DEE [14], and UEE-UCB [15] are notable exceptions that learn the optimal exit in an EE-DNN using MABs.

Our work differs from [13]–[15] in the following ways: *First,* our paper employs MABs to learn optimal thresholds that, in turn, characterize optimal exits, while previous work

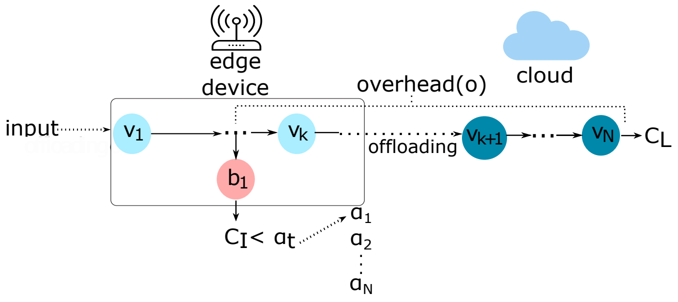


Fig. 1. Early-exit MobileNetV2 with one side branch under an adaptive offloading scenario.

sets the optimal exit explicitly as the control. *Second,* in LEE and UEE-UCB, early-exit decisions are taken before the image goes through multiple DNN layers. Instead, in DEE and in our work, these decisions occur on-the-fly at an intermediary DNN layer. This distinction allows us to leverage information about the confidence level at intermediary layers and then decide whether to continue the inference with additional layers. Since we have a different set of assumptions about the information available to make a decision (e.g., when compared against LEE and UEE-UCB) and on the decision variables (e.g., when compared against DEE), the threshold tuning proposed in this work is simpler and significantly differs from the class of policies considered in previous works. *Third,* LEE and DEE aim to enable an efficient DNN inference task for mobile devices, e.g., during service outages and network disconnections. To this end, they consider an edge-only scenario, in which the entire early-exit DNN model is processed at an edge device. *Fourth,* LEE and DEE focus on image classification tasks, and UEE-UCB tackles a natural language processing task employing a trained Elastic BERT model. Also, UEE-UCB assumes that the error rates at the exit points are decreasing and satisfy 'strong dominance.' We do not make any such assumptions. Our work, in contrast, uses convolutional DNNs for an image classification task adapting to different levels of image distortion. Image distortion dynamically changes according to context but, to the best of our knowledge, has not been analyzed by the aforementioned works.

We conclude this section by noting that as the control variables (actions) and objective function (regret) in our proposal are different from previous work, a direct comparison is infeasible. The works can be indirectly contrasted on real world deployments, that we leave as subject for future work.

III. Early-exit DNN through Multi-armed Bandits

Fig. 1 shows an EE-DNN model with a threshold α_t. This threshold is used to decide whether an early exit will be taken and is continuously adjusted by the proposed algorithm. Given an observation $\boldsymbol{x}$, EE-DNN processes it layer-by-layer until it reaches the side branch. Let $\mathcal{C}$ denote the set of target classes. Leveraging information available at the side branch, let $p_{\mathrm{I}}(c)$ denote the estimated probability that $\boldsymbol{x}$ belongs to class c, $c \in \mathcal{C}$. The intermediary confidence refers to the maximum of those probabilities, $C_{\mathrm{I}} = \max_{c \in \mathcal{C}} p_{\mathrm{I}}(c)$.

TABLE I
TABLE OF NOTATIONS.

Notation	Description
o	Overhead to use all layers as opposed to intermediary layer
C_{I}	Classification confidence at intermediary layer
C_{L}	Classification confidence at last layer
ΔC	Confidence gain from not performing early exit
α	Confidence threshold at intermediary layer
α_t	Confidence threshold at intermediary layer at round t
r_t	Instantaneous reward after processing observation t
$N(\alpha)$	Number of times action α was chosen
$Q(\alpha)$	Average reward of action α
$R_t(\pi)$	Instantaneous regret at round t using policy π
$R(\pi, n)$	Cumulative regret using policy π over n rounds

If confidence C_{I} meets the threshold α_t, i.e., $C_{\text{I}} \geq \alpha_t$, the side branch can provide a prediction assumed to be accurate, and the input is classified as $\hat{y} = \arg\max_{c \in \mathcal{C}}(p_{\text{I}}(c))$. In this case, the input is not further processed.

Otherwise, i.e., $C_{\text{I}} < \alpha_t$, the next layers process the input until the final output layer. The last layer produces confidence C_{L}, following a similar approach as the one used to produce C_{I} at the intermediary layer. This further processing entails an overhead o, as illustrated in Fig. 1. For instance, the overhead can be high due to an adaptive offloading scenario with DNN partitioning. In this case, the overhead is the communication time to offload data from the edge device to the cloud server, plus the time needed at the cloud to process the remaining layers. Alternatively, the overhead can also include the energy consumption of the edge device or the processing time to run the subsequent layers at the edge device. This paper considers an EE-DNN with one side branch, as in Fig. 1.

In this work, we propose models and algorithms to adapt EE-DNNs to different contexts, e.g., to account for dynamic image blur levels, using MABs. One of our key insights consists in learning a *conditional threshold* to determine whether to follow an early exit or not. The choice of the conditional threshold is also referred to as the *action* to be taken, and the threshold is *conditional* as it depends on the confidence observed at the side branch. The following section details our solution for the adaptive threshold choice.

IV. ADAPTIVE EARLY-EXIT IN NEURAL NETWORKS

This section presents our problem formulation and AdaEE.[2]

A. Problem Setup

For each incoming observation (image) to be classified, the intermediary-layer and last-layer confidences C_{I} and C_{L} are unknown until the observation passes through the corresponding DNN layers. Let $\mathcal{A}$ denote the threshold set containing the possible thresholds to be chosen (the notation is summarized in Table I). For each observation $\boldsymbol{x}$, the learner selects an action $\alpha_t \in \mathcal{A}$ corresponding to a threshold chosen at round t based on past observations. Then, the learner checks if $C_{\text{I}} \geq \alpha_t$. If so, an early exit will be taken. Otherwise, the input is processed until the last layer, which will provide the last-layer confidence C_{L}.

[2]Our code is available at https://github.com/pachecobeto95/AdaEE

Algorithm 1: AdaEE

```
1  Input: $\tilde{c} > 0$ (learning rate), $o$ (overhead), $\mathcal{A}$ (set of
     thresholds), $K$ (number of thresholds)
2  Initialize Try each threshold; set
     $Q_1(\alpha_1) = r_1(\alpha_1), \ldots, Q_K(\alpha_K) = r_K(\alpha_K)$
3  for $t = K+1, K+2, \ldots,$ do
4      Receive a sample
         $\alpha_t \leftarrow \text{argmax}_{\alpha \in \mathcal{A}} \left(Q_{t-1}(\alpha) + \tilde{c}\sqrt{\frac{\ln(t)}{N_{t-1}(\alpha)}}\right)$;
5      Obtain confidence $C_{\text{I}}$ at the intermediary level
6      if $C_I \geq \alpha_t$ then
7          Perform early exit and set $r_t(\alpha_t) \leftarrow 0$
8      else
9          Use the last layer and observer $C_{\text{I}}$ and $C_{\text{L}}$
10         $r_t(\alpha_t) \leftarrow \max(C_{\text{L}} - C_{\text{I}}, 0) - o$;
11     end
12     $N_t(\alpha_t) = N_{t-1}(\alpha_t) + 1$;
13     $Q_t(\alpha_t) = \sum_{u=1}^{t} r_u(\alpha_u)\mathbb{1}\{\alpha_u = \alpha_t\}/N_t(\alpha_t)$;
14 end
```

Let ΔC denote the confidence gain from the intermediary to the last layer, $\Delta C = \max\{C_{\text{L}} - C_{\text{I}}, 0\}$. Recall that o denotes the overhead due to processing up to the last layer. We assume that o is between $[0, 1]$, e.g., scaling an overhead measured in units of energy or time by a factor, such that ΔC and o are commensurable. For any threshold $\alpha_t \in \mathcal{A}$, let $r(\alpha_t)$ denote its associated instantaneous reward. Then,

$$r(\alpha_t) = \begin{cases} 0 & \text{if } C_{\text{I}} \geq \alpha_t \text{ (early exit)}, \\ \Delta C - o & \text{otherwise.} \end{cases} \quad (1)$$

Without loss of generality, we assume that random variables C_{I} and C_{L} have the same support: both vary in the range $[0, 1]$. Then, the mean reward for choosing threshold α_t is

$$\mathbb{E}[r(\alpha_t)] = \mathbb{E}[\Delta C - o | C_{\text{I}} < \alpha_t] \cdot P[C_{\text{I}} < \alpha_t]. \quad (2)$$

The learner's goal is to find a threshold that maximizes the expected reward. The optimal threshold is denoted by α^*,

$$\alpha^* = \arg\max_{\alpha \in \mathcal{A}} \mathbb{E}[r(\alpha)]. \quad (3)$$

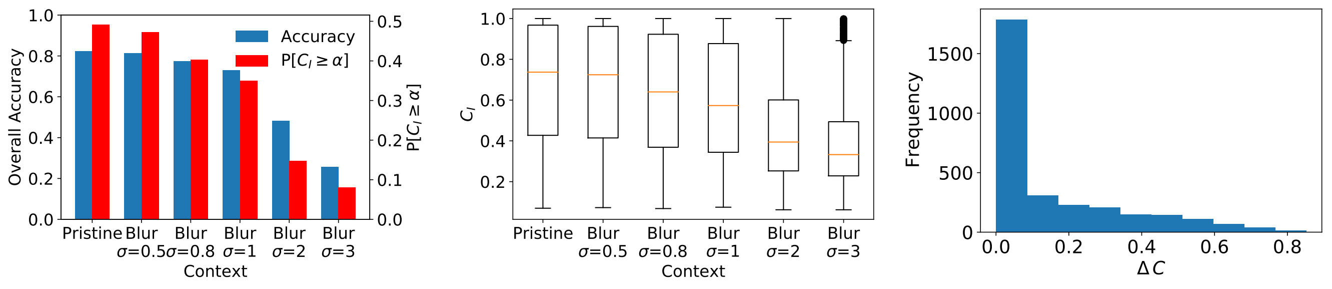
The instantaneous regret $R_t(\pi)$ is given by

$$R_t(\pi) = r(\alpha^*) - r(\alpha_t). \quad (4)$$

We define the performance of the learner that uses policy π over $n \in \mathbb{N}$ rounds in terms of expected regret as

$$R(\pi, n) = n \cdot \mathbb{E}[r(\alpha^*)] - \sum_{t=1}^{n} \mathbb{E}[r(\alpha_t)|\pi] = \sum_{t=1}^{n} \mathbb{E}[R_t(\pi)]. \quad (5)$$

Our goal is to develop a learning algorithm based on multi-armed bandits to find a policy π^* that reaches a sub-linear cumulative regret $R(\pi^*, n)/n \to 0$.



(a) Performance under threshold α=0.8 (b) Intermediary-layer confidence, $C_{\rm I}$ (c) Confidence gain, ΔC

Fig. 2. Using Caltech-256 dataset, the plots show (a) the overall accuracy for different contexts (image distortion levels) using a fixed threshold of $\alpha = 0.8$; (b) boxplot of intermediary-layer classification confidence $C_{\rm I}$, under $\alpha = 0.8$; and (c) a histogram of confidence gain ΔC.

B. Algorithm

To address the problem of setting confidence thresholds for an early exit, we develop AdaEE. AdaEE is an online learning algorithm to find the optimal confidence threshold based on the Upper Confidence Bound (UCB) algorithm [22]. Algorithm 1 presents the pseudo-code of the AdaEE algorithm. The inputs to the algorithm are the exploration rate, $\tilde{c}$, and the set of thresholds $\mathcal{A}$. Let K denote the number of thresholds, $K = |\mathcal{A}|$. Each threshold is applied once for the first K inputs, producing one sample for each action (line 2). In the next rounds, the learner selects a threshold with the highest UCB index (line 4).

Key idea. If $C_{\rm I} \geq \alpha_t$, an early exit occurs (lines 6-7). Otherwise, all subsequent layers are processed. In this case, the prediction from the layer with the highest confidence is taken as the final output (lines 8-10). Next, we update the statistics $N_t(\alpha_t)$ and $Q_t(\alpha_t)$ (lines 12-13). $N_t(\alpha_t)$ is the number of times the action α_t is chosen at the round t. $Q_t(\alpha_t)$ is the average reward of action α_t at the round t.

Logarithmic regret. For $\tilde{c} \geq 2$ and any n, the regret of the proposed algorithm can be upper bounded as follows:

$$R(AdaEE, n) \leq 8 \sum_{\alpha \in \mathcal{A} \setminus \alpha^*} \frac{\log(n)}{\Delta_\alpha} + (\pi^2/3 + 1) \sum_{\alpha \in \mathcal{A} \setminus \alpha^*} \Delta_\alpha, \tag{6}$$

where $\Delta_\alpha = \mathbb{E}[r(\alpha^*)] - \mathbb{E}[r(\alpha)]$.

The above bound follows from the analysis of UCB1 [22]. Hence, when using UCB1, our policy can achieve logarithmic regret [22]. The next section provides experiments motivating a threshold conditioned on the context. Then, we experimentally evaluate the convergence of AdaEE, illustrating that logarithmic regret is achievable in realistic settings.

V. Evaluation

A. Context Matters

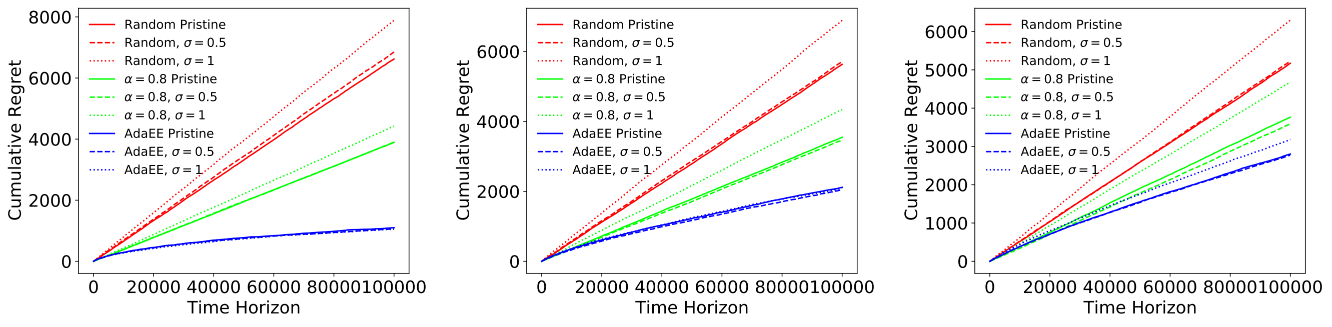
Next, we show that the context impacts the performance of EE-DNNs, by evaluating the classification probability and accuracy on a side branch. Our analysis considers that the context is the image distortion. However, context can be defined by other environmental aspects, such as the predominance of a given class over others in the gathered images. In our analysis, we train an early-exit MobileNetV2 [23] with one side branch on the Caltech-256 dataset [24], following the training procedure described in [7], [13]. The Caltech-256 dataset contains undistorted images of ordinary objects, such as motorbikes and shoes. This dataset has 257 classes. We split the dataset into 80% of images for training, 10% for validation, and 10% for testing. Next, we apply Gaussian blur as image distortion on each image from the testing dataset.

Gaussian blur occurs when the camera is out of focus or the image is gathered in a foggy environment [11]. To obtain a blurred image, we apply a Gaussian kernel with standard deviation σ to an undistorted image. The standard deviation σ defines the blur level so that a higher σ corresponds to a more blurred image. We employ $\sigma \in \{0.5, 0.8, 1, 2, 3\}$. For each blur level, the kernel size is $4\sigma + 1$ [25]. A pristine (undistorted) image is an image with blur level $\sigma = 0$.

Impact of image distortion. To assess the impact of distortion on model performance, Fig. 2(a) shows the early-exit probability $P[C_{\rm I} \geq \alpha]$ and overall accuracy. The early-exit probability is the fraction of inputs classified on the side branch, and the overall accuracy is the fraction of inputs correctly classified. Fig. 2(a) shows that for a fixed threshold $\alpha = 0.8$, both metrics decrease as blur increases. The decrease is steeper when blur increases from 1 to 2, suggesting that for such high levels of distortion it may be advantageous to increase the early-exit threshold.

Challenge: high intra-context variance. Fig. 2(b) presents a boxplot with the observations' intermediary-layer confidences $C_{\rm I}$ provided for pristine ($\sigma = 0$) and blurred images with different blur levels σ. Recall that we interchangeably refer to *context* and image distortion level. First, this figure corroborates that context impacts EE-DNNs performance, affecting the intermediary-layer confidence $C_{\rm I}$. Moreover, this figure shows the high variance in the observations' confidence, even when sampled from the same context, further motivating the need for adaptive thresholds [22].

Confidence gain analysis. Next, we analyze the confidence gain ΔC obtained by running early-exit inference on pristine (undistorted) images from the testing set of Caltech-256. Fig. 2(c) presents a histogram of confidence gain ΔC. This figure shows that the confidence gain concentrates on



(a) Overhead: 0.0 (b) Overhead: 0.05 (c) Overhead: 0.1

Fig. 3. Cumulative regret obtained by AdaEE, under several overhead values and contexts, compared against two alternative strategies.

small values, around 0 and 0.1. Recall that our model receives as input normalized overhead values in agreement with confidence gains. In our evaluation, we let $o \in \{0, 0.05, 0.1\}$.

B. AdaEE Evaluation

Is logarithmic regret feasible? We experimentally demonstrate the convergence of AdaEE to select the optimal threshold α^* that adapts to context. To this end, we run the AdaEE algorithm using test images from the Caltech-256 dataset. For each round, we randomly draw an image from the test set with replacement. Then, we perform an adaptive early-exit inference, using AdaEE to choose the threshold (action) $\alpha_t \in \mathcal{A}$, where $\mathcal{A}$ denotes the action set. In our evaluation we let $\mathcal{A} = \{0, 0.1, \cdots, 1.0\}$.

Fig. 3 shows cumulative regret as a function of the time horizon for a specific overhead value and several contexts, represented by pristine and blurred images with different blur levels. Each line corresponds to a particular context, while each plot shows the results for a specific overhead value. For example, the curve labeled as "AdaEE, $\sigma = 0.1$" represents the cumulative regret obtained by AdaEE using blurred images with $\sigma = 0.1$. The cumulative regret is computed as the sum of the instantaneous regret obtained up to round t.

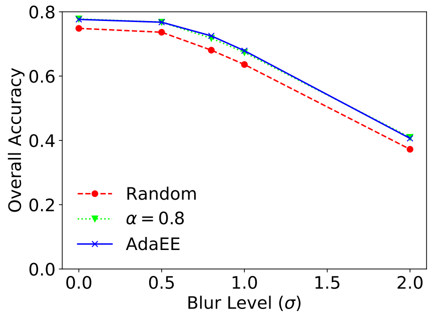
Fig. 3 experimentally confirms that logarithmic regret is achievable, i.e., convergence typically occurs after collecting a few thousand observations for different contexts and overhead values [26]. Indeed, for all the considered contexts, AdaEE reaches sub-linear cumulative regret before 100,000 observations. In Fig. 3(a), when overhead is zero, the optimal threshold is the maximum allowed threshold, hence never performing an early exit. Fig. 3 shows that as the overhead increases from 0 to 0.1, the difficulty of finding the optimal threshold α^* also increases, taking more rounds to achieve logarithmic convergence. Moreover, Fig. 3 compares the cumulative regret of AdaEE against two alternatives, namely 1) a random threshold strategy and 2) a fixed threshold, $\alpha = 0.8$. Fig. 3 shows that AdaEE outperforms the considered alternatives, as it yields a lower cumulative regret and achieves sub-linear growth of regret earlier than its alternatives.

AdaEE's Performance: Figs. 4(a) and 4(b) compare the overall accuracy of AdaEE against 1) a random threshold strategy and 2) a fixed threshold, $\alpha = 0.8$. Each plot presents the results considering a particular overhead value. The results show that the proposed AdaEE outperforms the overall accuracy obtained by the randomly-chosen threshold by about 4% and that AdaEE matches the overall accuracy obtained with a fixed threshold value of $\alpha = 0.8$. Recall from Fig. 3 that the fixed threshold yields a higher cumulative regret when compared to AdaEE. Together, Fig. 3 and Figs. 4(a) and 4(b) indicate that a fixed threshold yields high performance penalty due to inefficient decision-making and that AdaEE is able to circumvent such penalty through adaptive thresholds.

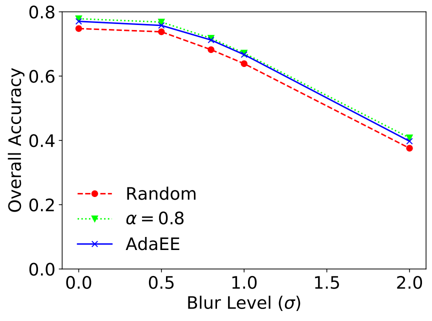
Next, we assess the adaptive behavior of AdaEE over time under dynamic contexts. To this aim, Fig. 4(c) presents the evolution of the overall accuracy of AdaEE (blue), compared against a random-threshold strategy (red) and fixed threshold with $\alpha = 0.8$ (green). The points of discontinuity in the lines correspond to context changes that occur at $t = 5,000$, $t = 10,000$, and $t = 15,000$. Different line styles also characterize different contexts: solid lines present results for pristine images, while dashed, dotted, and dashed-dotted lines correspond to blurred images with $\sigma = 0.5$, $\sigma = 0.8$, and $\sigma = 1$, respectively. First, this figure demonstrates the ability of AdaEE to adapt after context transitions. Second, the results indicate that AdaEE rapidly converges to a stable accuracy level after switching between different contexts. Third, they also show that AdaEE outperforms randomly-chosen thresholds in terms of overall accuracy for the analyzed contexts. The curve of AdaEE starts with a lower overall accuracy than $\alpha = 0.8$ due to its need to converge. However, as AdaEE converges, we can notice that AdaEE matches the performance of $\alpha = 0.8$, as shown in Figs. 4(a) and 4(b).

VI. Conclusion

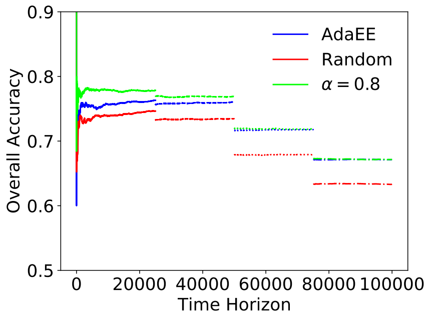
Despite the success of DNNs in addressing an increasingly large number of tasks, their adoption is still challenging in resource-constrained devices. EE-DNNs can extend the applicability of DNNs to an edge-cloud co-inference framework. However, their optimal parameterization is context-dependent. Indeed, our evaluation found that the fraction of observations that can be early classified varies widely across different contexts (Fig. 2). This indicates the need for adaptive thresholds. We presented AdaEE, an adaptive online scheme to parameterize the confidence thresholds of EE-DNNs using



(a) Overhead: 0.0



(b) Overhead: 0.1



(c) Overhead: 0.1

Fig. 4. The leftmost and middle plots compare AdaEE with randomly-chose threshold and fixed threshold in terms of overall accuracy, using overhead of (a) 0.0 and (b) 0.1. The rightmost plot show evolution of accuracy over the time horizon by running AdaEE algorithm for different contexts.

MABs. Our key metric of interest is the cumulative regret, that is derived from the instantaneous reward (Equation (1)), and acts as a proxy to the impact of wrong early-exit decisions. In particular, it depends on the overhead to process additional layers of EE-DNNs. In a setting where overhead captures communication delay, AdaEE can reduce the inference time, e.g., by decreasing the offloading frequency to the cloud.

We show that AdaEE can converge towards efficient solutions whose cumulative regret grows sub-linearly. Moreover, AdaEE shows stable behavior across the considered contexts and achieves a sub-linear cumulative regret faster than randomly-chosen and fixed thresholds (Fig. 3). Also, AdaEE outperforms, in terms of accuracy, randomly-chosen and fixed thresholds, rapidly converging after context changes (Fig. 4).

As the next steps, we can consider more complex EE-DNN models, such as ResNet, also accounting for multiple side branches. We also plan to establish tighter bounds on the convergence of MABs for the parametrization of early-exit DNNs and conduct a realistic deployment of AdaEE, accounting for the sudden and dynamic environmental changes.

Acknowledgement

This study was financed in part by Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Finance Code 001, CNPq, PR2/UFRJ, FAPERJ Grants E-26/203.211/2017, E-26/010.002174/2019, and E-26/201.300/2021, and FAPESP Grant 15/24494-8.

References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, 2012.

[2] K. Bochie, M. S. Gilbert, L. Gantert, M. S. M. Barbosa, D. S. V. Medeiros, and M. E. M. Campista, "A survey on deep learning for challenged networks: Applications and trends," *Journal of Network and Computer Applications*, vol. 194, p. 103213, 2021.

[3] M. Satyanarayanan, "The emergence of edge computing," *Computer*, vol. 50, no. 1, pp. 30–39, 2017.

[4] Y. Kang, J. Hauswald, C. Gao, A. Rovinski *et al.*, "Neurosurgeon: Collaborative intelligence between the cloud and mobile edge," in *ACM Computer Architecture News*, vol. 45, 2017, pp. 615–629.

[5] C. Hu, W. Bao, D. Wang, and F. Liu, "Dynamic adaptive DNN surgery for inference acceleration on the edge," in *IEEE Conf. on Computer Communications (INFOCOM)*, 2019, pp. 1423–1431.

[6] R. G. Pacheco and R. S. Couto, "Inference time optimization using branchynet partitioning," in *IEEE Symposium on Computers and Communications (ISCC)*, 2020, pp. 1–7.

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

[8] S. Laskaridis, S. I. Venieris, M. Almeida, I. Leontiadis, and N. D. Lane, "SPINN: synergistic progressive inference of neural networks over device and cloud," in *Int. Conf. on Mobile Computing and Networking (MobiCom)*, 2020, pp. 1–15.

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

[10] F. Secci and A. Ceccarelli, "On failures of rgb cameras and their effects in autonomous driving applications," in *IEEE International Symposium on Software Reliability Engineering (ISSRE)*, 2020, pp. 13–24.

[11] S. Dodge and L. Karam, "Understanding how image quality affects deep neural networks," in *IEEE International Conference on Quality of Multimedia Experience (QoMEX)*, 2016, pp. 1–6.

[12] R. Pacheco, F. R. Oliveira, and R. Couto, "Early-exit deep neural networks for distorted images: providing an efficient edge offloading," in *IEEE Global Communications Conf. (GLOBECOM)*, 2021, pp. 1–6.

[13] W. Ju, W. Bao, D. Yuan, L. Ge, and B. B. Zhou, "Learning early exit for deep neural network inference on mobile devices through multi-armed bandits," in *IEEE/ACM Int. Symposium on Cluster, Cloud and Internet Computing (CCGrid)*, 2021, pp. 11–20.

[14] W. Ju, W. Bao, L. Ge, and D. Yuan, "Dynamic early exit scheduling for deep neural network inference through contextual bandits," in *ACM Int. Conf. on Information & Knowledge Management*, 2021, pp. 823–832.

[15] H. Narayan, M. K. Hanawal, and A. Bhardwaj, "Unsupervised early exit in dnns with multiple exits," in *ACM Int. Conf. AI-ML Systems*, 2022. [Online]. Available: https://doi.org/10.48550/arXiv.2209.09480

[16] Z. Wang, W. Bao, D. Yuan, L. Ge, N. H. Tran, and A. Y. Zomaya, "SEE: Scheduling early exit for mobile dnn inference during service outage," in *ACM Int. Conf. on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, 2019, pp. 279–288.

[17] B. Fang, X. Zeng, F. Zhang, H. Xu, and M. Zhang, "Flexdnn: Input-adaptive on-device deep learning for efficient mobile vision," in *IEEE/ACM Symposium on Edge Computing (SEC)*, 2020, pp. 84–95.

[18] E. Li, L. Zeng, Z. Zhou, and X. Chen, "Edge ai: On-demand accelerating deep neural network inference via edge computing," *IEEE Transactions on Wireless Communications*, vol. 19, no. 1, pp. 447–457, 2019.

[19] G. Kim and J. Park, "Low cost early exit decision unit design for cnn accelerator," in *IEEE Int. SoC Design Conf.*, 2020, pp. 127–128.

[20] R. Pacheco, R. Couto, and O. Simeone, "Calibration-aided edge inference offloading via adaptive model partitioning of deep neural networks," in *IEEE Int. Conf. Communications (ICC)*, 2021, pp. 1–6.

[21] X. Dai, X. Kong, and T. Guo, "EPNet: Learning to exit with flexible multi-branch network," in *ACM Int. Conf. on Information & Knowledge Management*, 2020, pp. 235–244.

[22] P. Auer *et al.*, "Finite-time analysis of the multiarmed bandit problem," *Machine Learning*, vol. 47, pp. 235–256, 2002.

[23] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convo-

lutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[24] G. Griffin, A. Holub, and P. Perona, "Caltech-256 object category dataset," California Institute of Technology, Tech. Rep., 2007.

[25] T. S. Borkar and L. J. Karam, "Deepcorrect: Correcting dnn models against image distortions," *IEEE Transactions on Image Processing*, vol. 28, no. 12, pp. 6022–6034, 2019.

[26] H. Tibrewal, S. Patchala, M. K. Hanawal, and S. J. Darak, "Distributed learning and optimal assignment in multiplayer heterogeneous networks," in *IEEE Conf. on Computer Communications (INFOCOM)*, 2019, pp. 1693–1701.