EARLY EXIT DEEP NEURAL NETWORKS FOR DISTORTED IMAGES ON EDGE ENVIRONMENTS

Roberto G. Pacheco[1, 2], Fernanda D.V.R. Oliveira[1], Rodrigo S. Couto[1]

[1]Universidade Federal do Rio de Janeiro (UFRJ) - GTA/PADS/PEE-COPPE/DEL-Poli, [2]Universidade Federal Fluminense (UFF)

NOTE: Corresponding author: Roberto G. Pacheco, pacheco@gta.ufrj.br

Abstract – Deep Neural Networks (DNNs) are widely used for image classification but can struggle with distorted images, leading to reduced accuracy. Moreover, these applications often demand meeting a strict deadline. To this end, an alternative involves employing an adaptive offloading based on early exit DNNs (EE-DNNs). EE-DNNs have branches inserted into their middle layers at the edge device. These branches provide confidence estimates. If the classification is sufficiently confident, the inference terminates at the edge device. Otherwise, the edge offloads the inference task to the cloud which runs the remaining layer. Moreover, multiple branches can compose an ensemble that collectively produces a more accurate inference. This work analyzes whether EE-DNN-based solutions can make DNN inference more robust against image distortion, while also satisfying deadline requirements in an adaptive offloading. Our results show that, in terms of accuracy, EE-DNNs and the ensemble approach are as sensitive as conventional DNNs for blurred images. However, given the edge usage, these EE-DNNs approaches can reduce the inference time, enabling them to better fulfill deadline requirements compared to conventional DNNs (i.e., with no branches).

Keywords – adaptive offloading, branchynet, early exit DNNs, edge computing, image distortion

NOTE: Part of this work is based on our paper published in Portuguese in the *XLI Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos* (SBRC 2023), available at https://sol.sbc.org.br/index.php/sbrc/article/view/24563. This utilization is permitted by the Brazilian publisher, as seen in https://sol.sbc.org.br/index.php/indice/conduta .

1. INTRODUCTION

Deep Neural Networks (DNNs) used in computer vision faces multiple challenges, such as the presence of distortion in the image inputs. In this case, studies have experimentally shown that the performance of DNNs is highly sensitive to image distortions [1, 2, 3, 4], which can arise from external conditions, such as camera movement during capture or environmental factors [4]. As a result, DNNs suffer a significant impact on their performance when classifying images with various types of distortions. Therefore, it is essential to develop techniques and strategies to enhance the robustness of DNNs against image distortions. By doing so, we can improve their ability to maintain high accuracy even when faced with distorted images.

Another challenge in computer vision applications is ensuring that DNN inference operates with high responsiveness, to meet strict deadlines. In this context, Early Exit Deep Neural Networks (EE-DNNs) have gained attention for its ability to make predictions at multiple stages of DNN depth, allowing for quicker decisions, which reduces the inference time compared to the conventional DNN (with no side branches).

This work aims to analyze whether solutions based on EE-DNNs can make inference more robust against distortions in images compared to conventional DNNs, while meeting deadline requirements. EE-DNNs consist of DNNs with additional branches inserted throughout their architecture [5, 6]. Given an input image, these branches can estimate the confidence of the image classification. If the confidence value exceeds a given threshold, the image is considered sufficiently confident and can

be classified early on the side branch. EE-DNNs are based on the idea that captured images vary in classification difficulty, and a significant portion of images can be classified early using these side branches.

Another approach involves using multiple side branches to compose an ensemble [7]. In this case, the inferences provided by each individual side branch are combined collaboratively to produce a single overall inference, which is then used to classify the image. We refer to this approach as an ensemble-based strategy.

This work investigates whether EE-DNNs and ensemble-based EE-DNNs with side branches can make inference more robust against images with Gaussian blur compared to conventional DNNs (i.e., with no side branches). To achieve this, EE-DNNs and the ensemble-based strategy are implemented in an adaptive offloading scenario, where one part of the EE-DNN is processed on the edge device, while the other part is run at the cloud server. This work experimentally demonstrates that the use of EE-DNNs and an ensemble-based EE-DNNs can achieve an accuracy level equivalent to conventional DNNs for images with various levels of blur. Moreover, in an adaptive offloading scenario aided by EE-DNNs, this study shows that EE-DNNs and ensemble-based strategy can significantly reduce inference time and the number of Floating-point Operations (FLOPs) by classifying a substantial number of images early at the edge.

This work is structured as follows. Section 2 presents the related work existing in the literature. Next, Section 3 introduces the concept of EE-DNNs. Section 4 analyzes the performance of EE-DNNs solutions for when receiving distorted images and compares this to the performance of conventional DNNs, while Section 5 analyzes the EE-DNNs solutions considering application-level metrics. Finally, Section 6 concludes this work and proposes the next steps.

2. RELATED WORK

There are several articles in the literature about EE-DNNs. BranchyNet [5] is an EE-DNN proposal that employs entropy to decide whether a certain input can be classified early. On the other hand, SPINN [6] makes this decision based on a classification confidence estimate. BranchyNet and SPINN experimentally show that, in different image classification datasets, a significant portion of images can be classified early at the side branches because achieves a given threshold. As a result, the primary goal of this work is to reduce the average inference time. Other proposals, such as Dong *et al.* [8] and LEE [9], develop optimization-based strategies to dynamically select the number of DNN layers processed at the edge or at the cloud to achieve requirements. On the other hand, another direction of research focuses on implementing EE-DNNs on various hardware setups. For example, Que *et al.* [10] implements the early exit strategy on Field-Programmable Gate Array (FPGA). Nevertheless, previous work does not account for the presence of images with potential distortions, which represents a more realistic assumption in real-world applications.

Other studies have begun to investigate the influence of image distortion on DNN inference performance. Dodge and Karam experimentally demonstrate that DNNs are sensitive to image distortion [1]. As a result, distorted images significantly degrade the performance of DNNs. Therefore, several studies propose strategies to enhance the robustness of conventional DNNs against distorted images. DeepRepair [11] proposes a strategy based on data augmentation to turn a running DNN model more robust against inputs with distortion. Meanwhile, AdaEE [12] and Split-EE [13] develop strategies to adjust the early exit decision based on blur level. Collabar [4] also trains a specialized DNN model for each distortion type. However, Collabar implements a distortion classifier to identify the distortion type present in the image and select the appropriate specialized DNN. Nevertheless, the drawback of these proposals is the need to store and execute a specialized model for each potential distortion type in the image. Consequently, this approach may be impractical, particularly for edge devices with constrained computational resources. To address this issue, Pacheco *et al.* propose an architecture of EE-DNNs with expert side branches, where only the side branches are trained specifically for each distortion type [2]. Therefore, EE-DNNs with expert side branches can scale the number of distortion types considered, without the need to store and execute separate models for each distortion type.

Unlike previous proposals, this work does not focus on developing new strategies to make EE-DNNs specialized in specific distortion types. Instead, our

goal is to compare the use of EE-DNNs to conventional DNNs to improve performance in classifying distorted images. Therefore, EE-DNN models in this work are not trained for a specific distortion type, such as in [4] and [3], nor the side branches are trained to be expert in a particular distortion type, such as in [2]. This article analyzes the inherent natural robustness of EE-DNNs to handle distorted images. Therefore, this article can recommend that applications that consider of receiving images should employ EE-DNN solutions, instead of conventional DNNs to achieve a better performance. Moreover, unlike previous proposals, except for [2], this work implements an adaptive offloading scenario to reduce inference time compared to a conventional DNN processed solely in the cloud.

Our previous paper [14] investigates EE-DNN solutions for enhancing inference robustness against blurred images compared to conventional DNNs in an adaptive offloading scenario. The results demonstrate significant improvements in inference robustness with EE-DNNs and ensemble. This article focuses on evaluating EE-DNNs and ensemble-based solutions at application-level metrics, such as *inference outage probability* and *missed deadline probability*, which is crucial to assessing performance in adaptive offloading scenario [15]. This work finds that EE-DNNs and ensemble-based approaches improve inference robustness against blurred images, particularly in meeting deadline requirements.

3. EARLY EXIT DEEP NEURAL NETWORKS

Deep Neural Networks (DNNs) can be described as a sequence of neural layers capable of extracting features from input images and learning from these features. Generally, the initial layers of DNNs extract simpler features, while the deeper layers can extract more complex features. Early exit Deep Neural Networks (EE-DNNs) are based on the assumption that different inputs may require features with varying levels of complexity to be adequately classified [5]. In other words, there are inputs in the dataset that can be easily classified using features extracted by the initial neural layers. Hence, such samples can be classified right after processing these early layers. To achieve this, EE-DNNs have side branches inserted in their intermediate layers. On the other hand, there are other samples that need more complex features obtained from the deeper neural layers. In this case, these inputs must be processed by deeper layers to obtain an accurate classification.

Fig. 1 illustrates an EE-DNN with multiple side branches inserted throughout its architecture. In this figure, the gray vertices $L_1, \cdots, L_N$ represent the neural layers of the DNN's main backbone. The side branches are depicted by vertices $b_1, \cdots, b_k$. These side branches b_i can classify specific inputs using features extracted by the neural layers of the main backbone up to the corresponding side branch $L_1, \cdots, L_i$. This article follows the approach developed by SPINN [6] and inserts three side branches equally spaced in terms of FLOPs along the architecture of the DNN's main backbone. Consequently, EE-DNN comprises four exits, including the output layer present in the main backbone represented by the vertex L_N. For training, we employ the traditional training procedure proposed by [5] and [6].

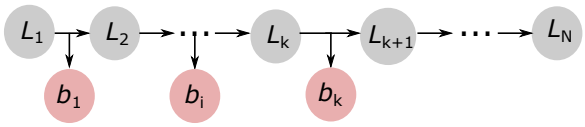


Figure 1 – Generic EE-DNN architecture. This image is extracted from our previous work [14].

3.1 Adaptive offloading

This work implements an EE-DNN in an adaptive offloading scenario [15]. Fig. 2 illustrates this scenario, in which a part of the EE-DNN is implemented on the edge device, while the remaining layers are processed on the cloud server. First, we define k as the number of side branches executed on the edge device. The output of layer L_k is sent to the cloud in case the side branches $b_1, \cdots, b_k$ cannot perform the classification.

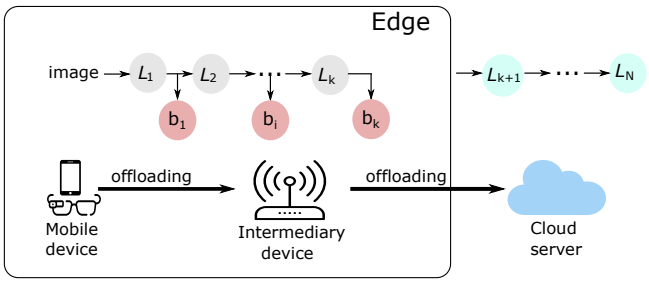


Figure 2 – An adaptive offloading scenario. This image is extracted from our previous work [14].

Once the EE-DNN is trained, its model receives an input image and processes it to perform inference and classify the object present in the image. The edge device processes the image through each neural layer until it reaches the side branch b_i, where $i = 1, \cdots, k$, and k is the number of side branches in the edge device. At this stage, the side branch b_i generates the output vector $\mathbf{z}_i$. From this vector $\mathbf{z}_i$, we can compute the probability vector $\boldsymbol{p}_i = \text{softmax}(\boldsymbol{z}_i) \propto \exp(\boldsymbol{z}_i)$, where each element corresponds to the probability of the image belonging to a certain class. Next, the classification confidence f_i is calculated as $f_i = \max_{c \in \mathcal{C}} p_{i,c}$, where $p_{i,c}$ is the c-th element of vector $\boldsymbol{p}_i$ and $\mathcal{C}$ is the set of possible classes. If the confidence value f_i is greater than or equal to a certain threshold α (i.e., $f_i \geq \alpha$), the edge device concludes the inference and the side branch b_i classifies the image as $\hat{y} = \arg\max_{c \in \mathcal{C}}(p_{i,c})$, where $\hat{y}$ denotes the label classified by the side branch. Otherwise, i.e., $f_i < \alpha$, the input is processed through the next layers of the main backbone until it reaches the next side branch b_{i+1}, where the described procedure is repeated. If none of the side branches at the edge reach the threshold α, then the edge device sends the inference to the cloud, where the remaining neural layers of the EE-DNN's main backbone are processed.

3.2 Ensemble of side branches with adaptive offloading

The ensemble-based EE-DNN strategy aims to use the individual inferences provided by each side branch to collectively produce a more accurate final inference [7]. In other words, the inference process does not end at the side branches; instead, their results are accumulated to compose an ensemble. Thus, after processing the k side branches, we obtain the set of probability vectors given by $\mathcal{K} = \{\boldsymbol{p}_1, \boldsymbol{p}_2, \cdots, \boldsymbol{p}_k\}$.

After processing the k side branches, Qendro *et al.* obtain the total inference $\boldsymbol{p}_{\mathcal{E}}(\mathcal{K})$ produced by the ensemble of side branches by taking the arithmetic mean of the probability vector p_i from each side branch [7]. However, in this work, we apply a weighted mean, in which the weighting is based on the accuracy of each branch b_i, denoted by Acc_i. The purpose of employing the weighted mean is to make the influence of a branch on the total inference, produced by the ensemble, proportional to its accuracy obtained on the validation set. It is worth noting that, in this paper, the total inference given by the ensemble does not consider the inference provided by the output layer of the main backbone, as it is implemented in the cloud.

During inference, each neural layer processes the input image, including the side branches k implemented at the edge, resulting in the set of inferences $\mathcal{K}$. The total inference produced by the ensemble of side branches $\boldsymbol{p}_{\mathcal{E}}(\mathcal{K})$ is computed as a weighted mean given by

$$\boldsymbol{p}_{\mathcal{E}}(\mathcal{K}) = \sum_{i=1}^{k} \text{Acc}_i \cdot \boldsymbol{p}_i. \qquad (1)$$

Thus, Equation *(1)* computes the probability vector $\boldsymbol{p}_{\mathcal{E}}(\mathcal{K})$ via a weighted mean by using the accuracy of each side branch Acc_i as a weight. As in Section 3.1, this vector is used to obtain the classification confidence $f_{\mathcal{E}}(\mathcal{K}) = \max_{c \in \mathcal{C}} p_{\mathcal{E},c}(\mathcal{K})$, where $p_{\mathcal{E},c}$ is the c-th element of the vector $\boldsymbol{p}_{\mathcal{E}}(\mathcal{K})$. Then, it is checked if $f_{\mathcal{E}}(\mathcal{K}) \geq \alpha$. In this case, the edge device classifies the image as $\hat{y} = \arg\max_{c \in \mathcal{C}}(p_{\mathcal{E},c}(\mathcal{K}))$. Otherwise, the edge device sends the inference to the cloud to process the main backbone, as explained in Section 3.1.

4. EXPERIMENTS

The experiments in this work evaluate the impact of adopting solutions based on EE-DNNs to classify images with different levels of Gaussian blur compared to a conventional DNN (i.e., with no side branches). For this purpose, the classification performance using the early exit ensemble is compared to the performance obtained by a conventional DNN implemented solely on the cloud server. The use of a conventional DNN corresponds to a traditional strategy, where the mobile device captures the image and transmits it to the cloud for further processing. This section describes the dataset and the experimental setup employed and then presents the experiments. All the developed code is available in an open repository[1].

4.1 Dataset

This work uses the Caltech-256 dataset [16], which contains 30 607 color images of everyday objects, such as motorcycles, flags, and sneakers, without distortion. The dataset consists of 257 categories

[1] `https://github.com/pachecobeto95/DR_EENN`

and has been divided into 80% of the images for training, 10% for validation, and 10% for testing.
To perform this evaluation, a procedure is necessary to introduce different levels of distortion into the images. The distortion evaluated in this work is a Gaussian blur. An image with Gaussian blur can be modeled as follows: $g(\boldsymbol{x}|\sigma_{GB}) = f(\boldsymbol{x}) * h(\boldsymbol{x}|\sigma_{GB})$, where $\boldsymbol{x} = (x_1, x_2)$ are the pixel coordinates. The term $g(\boldsymbol{x}|\sigma_{GB})$ represents the image with Gaussian blur, while $f(\boldsymbol{x})$ denotes the undistorted image, and $h(\boldsymbol{x}|\sigma_{GB})$ is the Gaussian filter. The standard deviation of this Gaussian filter is denoted by σ_{GB}. The level of blur present in the image depends on the standard deviation of the Gaussian filter σ_{GB}. Thus, the higher the σ_{GB} value, the more the image is distorted with blur. This work employs $\sigma_{GB} \in \{0, 0.1, 0.2, 0.5, 0.8, 0.9, 1, 1.2, 1.5, 1.8\}$. Images without distortion correspond to $\sigma_{GB} = 0$.

4.2 Experimental setup

This work reproduces a realistic adaptive offloading scenario, as shown in Fig. 2. To this end, a Nvidia Jetson Nano board[2] is used as the edge device, running an Ubuntu 18.04 Operating System (OS) and located in Rio de Janeiro, Brazil. The edge device is equipped with an Nvidia Maxwell GPU with 128 cores and an ARM A57 CPU with four cores running at 1.43 GHz. Additionally, the experiment utilizes an Amazon AWS EC2[3] (Elastic Cloud Computing) virtual machine as the cloud server, instantiated in São Paulo, Brazil. São Paulo is chosen as it is the closest AWS location to Rio de Janeiro, providing the best cloud use case. The cloud machine is a `g4dn.xlarge` instance with four Intel Cascade Lake vCPUs and an NVIDIA Tesla T4 GPU, running Ubuntu 20.04 LTS. To implement the scenario, including the application on the cloud server, Python Flask[4] is used. The edge device is connected to the Internet through a Gigabit Ethernet network interface and communicates with the cloud server using HTTP.
Before conducting the experiments, the network conditions between the edge device (Nvidia Jetson Nano) and the cloud server (Amazon EC2 `g4dn.xlarge` instance) are checked. This is done by measuring the Round Trip Time (RTT) and throughput using `ping` and `iPerf3`, respectively. The average throughput obtained is 93.9 Mbps, with a standard deviation of 3.8 Mbps. The average RTT is 11.91 ms, with a standard deviation of 0.66 ms. These values may vary during the experiment.
This article employs MobileNetV2 [17] as the architecture for the conventional DNN, since it is a lightweight model designed to run on devices with limited computational resources. For EE-DNN, three branches are inserted into intermediate layers along the MobileNetV2 architecture, following the strategy proposed by SPINN [6]. As mentioned in Section 3, EE-DNNs have a confidence threshold α to decide whether an image should be classified early. In the experiments, α is set to $\{0.8, 0.9\}$. The DNN and EE-DNNs are calibrated using the methodology described in [15].
For each experimental round, we add Gaussian blur to each image in the dataset with a given level, varying according to σ_{GB}. In the execution of the experiment with EE-DNNs and the ensemble, we first define the number of active side branches (i.e., branches that can terminate the inference) at the edge and the confidence threshold α. Then, the edge device starts a timer to measure the inference time and begins processing the neural layers at the edge. If the inference meets the confidence criteria to terminate the classification at the edge, as described in Section 3, the timer is stopped. Otherwise, the edge device sends the data to the cloud server, which in turn processes the remaining neural layers of the DNN. In this case, the edge device waits for an HTTP response from the cloud server to stop the timer.
In the execution of the experiment with conventional DNN, a timer is first started and the image is immediately sent to the cloud server without processing any of the layers at the edge. Upon receiving the image, the cloud processes all the neural layers of the conventional DNN. As described earlier, the edge device waits for an HTTP response from the cloud server to stop the timer.
After completing an experimental round, it is possible to obtain the inference time required to classify an input and evaluate whether the classification performed is correct. This procedure is performed for each of the images in the test set. Therefore, the overall accuracy measured by the model and the average inference time are obtained. Additionally, in the case of EE-DNN and the ensemble of side branches, the probability of early classification

[2] `https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/`
[3] `https://aws.amazon.com/ec2/`
[4] `https://flask.palletsprojects.com/en/2.2.x/`

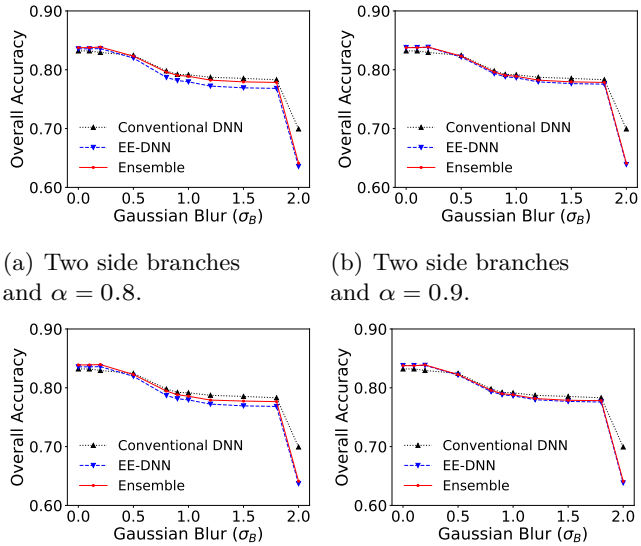
at the edge is also evaluated. The next sections present these results, comparing the EE-DNN without the ensemble, referred to simply as "EE-DNN", with the ensemble of side branches, referred to as "Ensemble", and the conventional DNN.

4.3 Accuracy

In this section, we assess the overall accuracy achieved by the EE-DNN, the ensemble of side branches, and the conventional DNN for different levels of Gaussian blur. Given a distortion level, the overall accuracy is calculated as the number of correctly classified images divided by the total number of images, considering both the images classified on the edge device and on the cloud server.

Fig. 3 illustrates the overall accuracy as a function of the Gaussian blur level, considering different confidence thresholds α and the number k of active branches at the edge. Note that the results for the conventional DNN are the same across all figures, as they are not dependent on α or the number of branches. At first glance, the figure reveals that the accuracy of the evaluated models decreases as the level of blur in the images increases. This finding supports the conclusion made by Dodge and Kara [1], indicating that image distortion leads to a degradation in DNN accuracy. Furthermore, the figure demonstrates that for undistorted images or those with a low level of blur (i.e., $\sigma_{GB} \in \{0, 0.1, 0.2\}$), the accuracy of both the EE-DNN and the ensemble of side branches can even surpass the performance of the conventional DNN. This outcome aligns with the expected behavior of an EE-DNN, as it adapts to images with varying levels of classification complexity within the same dataset [5]. We can observe that the ensemble of side branches achieved a performance comparable to that of EE-DNN and maintained a better performance than conventional DNN after introducing a low level of blur. Therefore, these results recommend the adoption of EE-DNN-based solutions over conventional DNNs, even for undistorted or mildly blurred images.

When comparing the performance of the EE-DNN and the ensemble of side branches in Fig. 3, it becomes evident that the ensemble achieved a slight accuracy gain over EE-DNN. This behavior arises from the fact that the ensemble of side branches employs multiple branches to obtain a more accurate estimation of the classification confidence $f_{\mathcal{E}}(\mathcal{K})$ compared to the confidence estimates of the individual branches. As a result, the ensemble enhances decisions regarding samples that should be classified at the edge. Consequently, the ensemble performs fewer early classifications at the edge, as will be demonstrated in Section 4.4, and relies more often on the more robust DNN in the cloud. These observations turn the ensemble into an intermediary solution between the EE-DNN and the conventional DNN. Despite the slight increase in accuracy compared to the EE-DNN, the ensemble does come with certain disadvantages relative to EE-DNN, which will be elaborated upon in the forthcoming experiments. Given the similarity in accuracy behavior between three and two branches, subsequent experiments in this study assume the activation of all three branches.



(a) Two side branches and $\alpha = 0.8$.

(b) Two side branches and $\alpha = 0.9$.

(c) Three side branches and $\alpha = 0.8$.

(d) Three side branches and $\alpha = 0.9$.

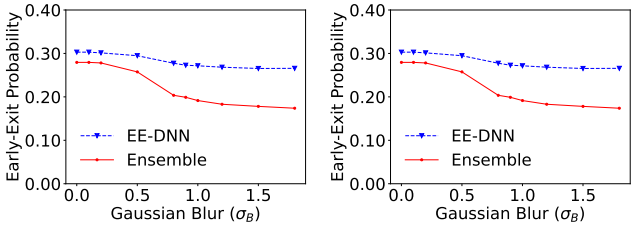
Figure 3 – Overall accuracy of EE-DNN, ensemble of side branches and conventional DNN, considering various confidence thresholds α and different number of side branches at the edge device. This image is sourced from our previous research [14].

4.4 Early exit probability

This section assesses the impact of EE-DNNs and the adoption of the ensemble strategy in terms of the probability of classifying an image on edge device. To compute this probability, the experimental procedure described in Section 4.2 is executed. Subsequently, this probability is computed as the number of samples classified early at the edge device divided by the total number of samples in the

test set. This computation is performed for each level of Gaussian blur.

Fig. 4 presents the early exit probability results across different levels of Gaussian blur using three side branches on the edge device. The blue curve corresponds to the results obtained by an EE-DNN, while the red curve represents the results obtained by the ensemble of side branches. In particular, there are no results for the conventional DNN, as the early exit probability is consistently zero. Fig. 4 illustrates that a substantial portion of the test set images can be classified early at the edge device. For instance, Fig. 4(a) demonstrates that at least 20% of images with Gaussian blur of $\sigma_{GB} = 1$ can be classified early at the edge, regardless of the use of an ensemble. These results show the implementation of solutions based on EE-DNNs can be effective to reduce inference time considering distorted images, as a significant portion can be classified at the edge. Fig. 4 reveals that the ensemble reduces the number of images classified early at the edge compared to the EE-DNN for various levels of Gaussian blur. Therefore, the results in figures 3 and 4 show a clear tradeoff between accuracy and the early exit probability, since the ensemble needs to rely more on the cloud to achieve a slight accuracy gain over the EE-DNN. Consequently, when compared to an EE-DNN, the ensemble may increase the inference time due to the need to send more images to the cloud. Additionally, the ensemble requires more processing, as all branches are utilized in inference. The forthcoming experiments aim to analyze this processing and inference time penalty in the ensemble strategy, while also comparing the outcomes to the conventional DNN.



(a) $\alpha = 0.8$. (b) $\alpha = 0.9$.

Figure 4 – Early exit probability at the edge device of EE-DNNs and ensemble of side branches for various thresholds. The image is extracted from our paper [14].

4.5 Number of floating-point operations

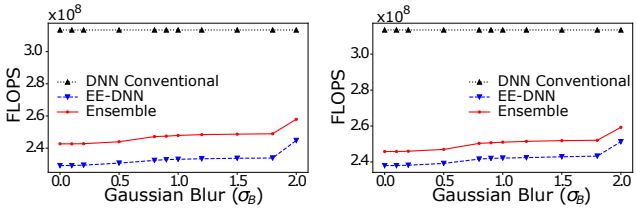
The DNNs examined in this study consist of convolutional layers that perform multiple convolution operations to classify a given image. The convolution operation involves numerous addition and multiplication operations, thus representing floating-point operations (FLOP). This section evaluates the number of FLOPs required for image classification. To measure the FLOPs count for each evaluated model, the inference procedure described in Section 3 is executed, and the number of FLOPs needed until the completion of inference is measured. For instance, if a classification is completed in the second side branch, only the number of FLOPs up to that branch is counted. If classification at the edge is not possible, the additional FLOPs needed to process the output layer of the main branch are also measured. For each distortion level, the average number of FLOPs is computed for each evaluated model. The library utilized for measuring FLOPs is `pthflops 0.4.2`[5], available in Python.

Fig. 5 shows the average number of FLOPs required for image classification across different levels of Gaussian blur. It is noteworthy that the curve corresponding to the conventional DNN remains constant, regardless of the level of Gaussian blur σ_{GB} present in the image. This is because an image is processed through all its neural layers in the conventional DNN, regardless of the blur level. On the other hand, the curves for EE-DNN and the ensemble of side branches exhibit a slight increase in FLOPs, which varies based on the image's blur level. This behavior arises from images with higher levels of blur being less frequently classified prematurely at the edge. Consequently, such images need to traverse more neural layers, resulting in a higher count of FLOPs.

Fig. 5 primarily demonstrates that the utilization of solutions based on early exits DNNs, such as EE-DNN and the ensemble of side branches, can significantly reduce the number of FLOPs compared to the conventional DNN, where all neural layers are always processed. Desislavov *et al.* experimentally demonstrate a strong relationship between the number of FLOPs and energy consumption, where a higher count of FLOPs indicates a greater number of neural layers, leading to increased energy consumption by the device that executes the DNN [18].

[5] `https://pypi.org/project/pthflops/`

Consequently, EE-DNNs and the ensemble of side branches indirectly contribute to energy savings by reducing the number of processed neural layers. When comparing an EE-DNN and the ensemble of side branches, it becomes evident that the ensemble performs significantly more operations than the EE-DNN, mainly because it classifies fewer samples prematurely at the edge. Hence, this outcome suggests that the ensemble of side branches might consume more energy than the EE-DNN and require a longer time to classify samples, as demonstrated in the following experiments.

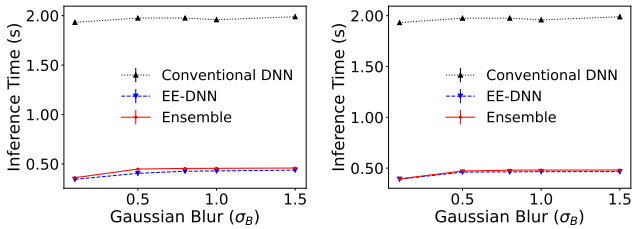


(a) $\alpha = 0.8$. (b) $\alpha = 0.9$.

Figure 5 – Number of FLOPs for EE-DNNs and ensemble of side branches given different confidence thresholds. This image is derived from our previous work [14].

4.6 Inference time

This section analyzes the inference time of the DNN approaches for different Gaussian blur levels. This experiment utilizes the implementation of the adaptive offloading scenario, as described in Section 4.2, considering three side branches at the edge device. For each blur level value, the average inference time and its corresponding 95% confidence interval are calculated for the three evaluated approaches.



(a) $\alpha = 0.8$. (b) $\alpha = 0.9$.

Figure 6 – Average inference time, in seconds, of EE-DNN and ensemble of side branches in an adaptive offloading scenario and conventional DNN. This image is extracted from our paper [14].

Fig. 6 illustrates the average inference time and its corresponding confidence interval[6] for the evaluated approaches as a function of the Gaussian blur level. This figure reveals that solutions based on EE-DNNs have significantly reduced the inference time for image classification compared to the conventional DNN. This behavior arises because EE-DNN-based solutions, including both EE-DNN itself and the ensemble of side branches, can classify a substantial number of images at the edge, thus preventing the introduction of communication delays. In contrast, the conventional DNN always requires sending the raw image to the cloud, introducing communication delay. As mentioned in Section 4.2, the cloud instance is located in São Paulo, which is close to the edge device located in Rio de Janeiro. Nevertheless, the results demonstrate a drastic reduction in the inference time, which could be even more significant if considering a cloud instance in more distant regions. Figures 3 and 6 indicate that EE-DNN and the ensemble-based solution yield similar results, especially with low blur levels. This is expected, due to the small difference between the EE-DNN and ensemble, contrasting with higher blur levels.

5. APPLICATION-LEVEL METRICS

The metrics evaluated in Section 4 are commonly used for evaluating the performance of DNN models during inference. However, these metrics often fall short in capturing a crucial aspect of DNN model reliability necessary for meeting the demands of an application. To this end, we evaluate DNN performance under application-level metrics that directly assess the DNN model's ability to meet application requirements.The first metric, *inference outage probability*, quantifies the model's capability to achieve an accuracy target. The second metric, known as *missed deadline probability*, accounts for the probability of meeting both accuracy and inference deadline requirements.

5.1 Inference outage probability

In practical applications, the primary goal for an early exit DNN model is to attain a specific target accuracy level. In this context, we can adjust the confidence threshold α as the desired target accuracy level. Hence, the threshold α plays the role of the desired target accuracy. Therefore, a suitable accuracy-related performance metric is the fraction of inputs that are classified with an accuracy level

[6]In Fig. 6, the confidence interval is not visually noticeable on the curve

greater than α. In our previous article [15], this metric is referred to as *inference outage probability*. To calculate this metric, we segment test input into batches and assess the average accuracy. In our specific scenario, we must evaluate two distinct types of accuracy. The first type, accuracy at the edge device, denoted as Acc_{edge}, exclusively accounts for inputs classified at the edge device. Consequently, for a given batch denoted as B_b, $Acc_{\text{edge}}(B_b)$ is computed as the number of inputs correctly classified at the edge divided by the total number of inputs classified at the edge device. Note that only inputs classified at the edge are included in the average. The edge inference outage probability is then computed as the fraction of batches that the edge inference outage event occurs [15].
Fig. 7 presents the edge inference outage probability as a function of the confidence threshold for varying blur levels. This plot serves as a comparative analysis of the performance between the EE-DNN and ensemble approaches, specifically in terms of edge inference outage probability. As previously explained, the conventional DNN considers a cloud-based scenario, devoid of any edge devices. Consequently, there exists no concept of edge inference outage probability in a conventional DNN. Fig. 7 shows that, for pristine images ($\sigma_B = 0$), the EE-DNN and ensemble present similar behavior in terms of edge inference outage probability considering different blur levels and distinct thresholds.

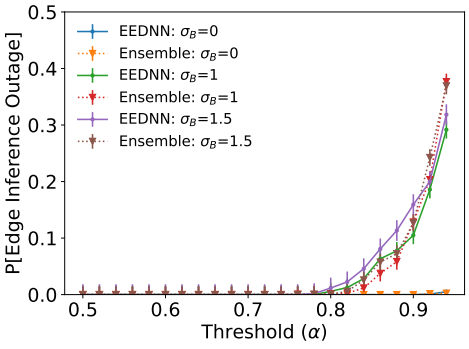


Figure 7 – Edge inference outage probability of EE-DNNs, ensemble and conventional DNN for different blur levels.

Another type of accuracy to be evaluated is the overall accuracy, which is computed as the fraction of inputs correctly classified, considering images classified at the edge device and on the cloud server. Fig. 8 illustrates the overall inference outage probability as a function of confidence thresholds across various blur levels. According to this metric, both the EE-DNN and ensemble demonstrate similar behavior. Consequently, there is no significant difference between these two approaches. In contrast, the conventional DNN demonstrates a significant reduction in overall inference outage probability when compared to early exit-based solutions, such as EE-DNN and ensemble. At first glance, the conventional DNN appears to be the most suitable choice for reducing the overall inference outage probability and is thus likely to satisfy the desired accuracy requirement. However, this enhancement, in terms of this metric, comes at the cost of introducing communication delays since the conventional DNN uses a cloud-based scenario. As a result, conventional DNN may face challenges in satisfying both accuracy and inference time requirements, which we explore in the subsequent section.

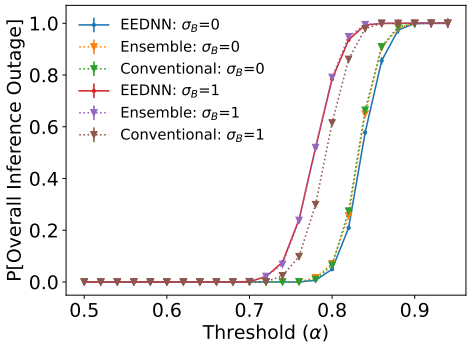


Figure 8 – Overall inference outage probability of EE-DNNs, ensemble and conventional DNN for different blur levels.

5.2 Missed deadline probability

In contrast to the previous metric under consideration, a successful application must simultaneously meet both accuracy and inference time requirements. In pursuit of this objective, our prior article [15] introduces a novel metric named *missed deadline probability*, which assesses a model's ability in attaining these crucial requirements.
To describe this metric, we must define missed deadline. A missed deadline is an event that happens when either the average inference time T_{inf} is larger than an application-defined latency deadline t_{tar} (i.e., $T_{\text{inf}} > t_{\text{tar}}$) *or* when the average overall accuracy $\text{Acc}_{\text{total}}$ is smaller than an accuracy requirement α (i.e., $\text{Acc}_{\text{total}} < \alpha$). Thus, the missed deadline probability quantifies the model's ability in meeting inference time and accuracy requirements. The missed deadline probability is an end-to-end application performance metric, since this metric depends on overall accuracy, and also infer-

ence time which includes the communication delay to offloading data to the cloud.

To calculate the missed deadline probability, we follow a similar process of grouping the test inputs $\boldsymbol{x}$ into batches, denoted as $B_1, \cdots, B_{N_B}$. For each batch B_b, we compute two key metrics: the average inference time, denoted as $T_{\text{inf}}(B_b)$, and the average overall accuracy, referred to as $\text{Acc}_{\text{total}}(B_b)$. The missed deadline probability P_{md} is computed as the fraction of the batches in which the missed deadline event occurs [15].

Fig. 9 presents the performance of the evaluated approaches, EE-DNNs, ensemble, and conventional DNN, in terms of missed deadline probability P_{md} according to the deadline requirement t_{tar}. Specifically, Fig. 9(a) and Fig. 9(b) show the results considering the confidence threshold $\alpha = 0.8$ and $\alpha = 0.83$, respectively. We choose the threshold $\alpha = 0.83$ because it represents a a higher accuracy requirement compared to $\alpha = 0.8$, yet it is not set so high that it would consistently result in $P_{md} = 1$ for all batches. Therefore, the choice of $\alpha = 0.83$ strikes a balance, ensuring that the batches can still meet the accuracy requirement without consistently resulting in $P_{md} = 1$. In these figures, each curve corresponds to the results obtained using images with a different blur level σ_B.

Fig. 9 shows that, for a tight deadline requirement, the EE-DNN, ensemble and conventional DNN present the same result, by obtaining $P_{md} = 1$. This happens because the evaluated approaches cannot meet the deadline requirement t_{tar}, incurring a missed deadline event. For example, in Fig. 9(a), considering a $t_{\text{tar}} \leq 250\,\text{ms}$ and images with blur level $\sigma_B = 1.0$, the EE-DNN, the ensemble and conventional DNN obtains the same results of $P_{md} = 1$. As t_{tar} increases, the early exit-based approaches, EE-DNN and ensemble, start to reduce the missed deadline probability, while the conventional DNN remains obtaining $P_{md} = 1$. This result occurs because the EE-DNN and ensemble approaches start to meet t_{tar}, due to the use of edge-cloud co-inference, which allows us to classify a significant fraction at the edge, reducing the average inference time $T_{\text{inf}}(B_b)$ for each batch. On the other hand, the conventional approach cannot satisfy t_{tar}, due to the introduction of communication delay in a cloud-only scenario. Moreover, we can notice that the EE-DNN and ensemble approaches obtains similar behaviors.

In Fig. 9(b), we can notice that, when $\sigma_B = 1.5$, the missed deadline probability P_{md} is always equal to 1, due to the low accuracy observed in highly blurred images. Furthermore, except when $\sigma_B = 1.5$, the ensemble exhibits a slight advantage over the EE-DNN, due to its capacity to meet demanding accuracy requirements.

As a conclusion, the results experimentally demonstrate that the use of early exit-based approaches, such as EE-DNN and ensemble, improve the model's ability to meet accuracy and deadline requirements compared to the conventional approach for several blur levels, which is crucial for computer vision applications.

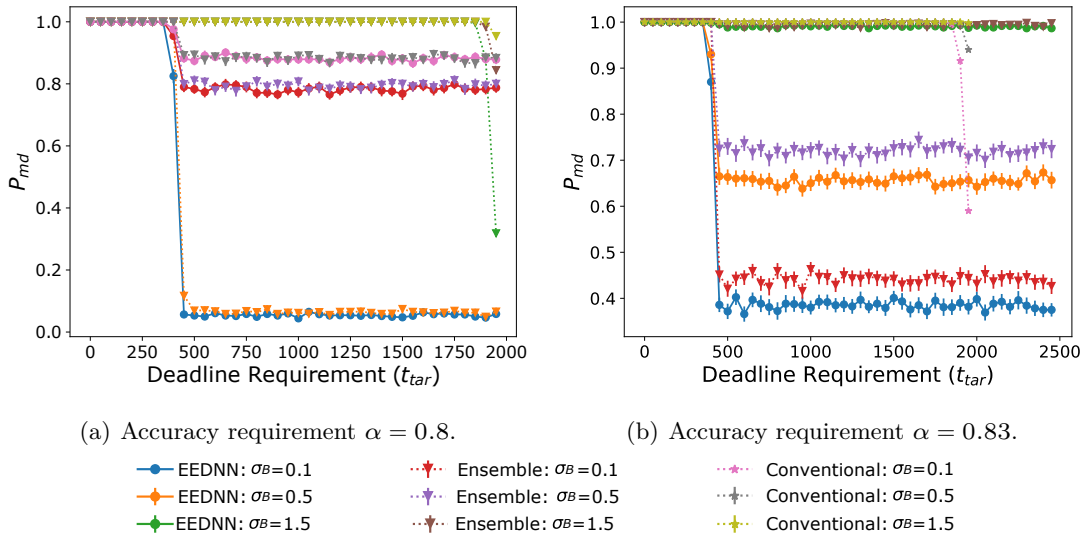
6. CONCLUSIONS AND FUTURE WORK

This study shows that the EE-DNN and the ensemble of side branches improve accuracy slightly compared to conventional DNNs in images with minimal Gaussian blur. Thus, adopting an EE-DNN-based solutions is recommended for undistorted or lightly distorted images. However, as Gaussian blur increases, their accuracy aligns with or slightly lags behind conventional DNNs. In adaptive offloading contexts, early exit strategies offer advantages. The EE-DNN and the ensemble of side branches classify a significant portion of inputs at the edge early, reducing inference time and FLOPs, potentially lowering energy consumption. Despite minor accuracy reductions in specific cases, their early classification ability at the edge proves advantageous.

Future work will focus on enhancing EE-DNN robustness against various distortions like Gaussian noise and image compression. Novel ensemble strategies will also be explored to improve DNN inference in distorted images while meeting deadline requirements. Additionally, analyzing these approaches across various image distortions will be a focus of future investigations.

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.



(a) Accuracy requirement $\alpha = 0.8$. (b) Accuracy requirement $\alpha = 0.83$.

Figure 9 – Missed deadline probability of EE-DNNs, ensemble and conventional DNN for different blur levels.

REFERENCES

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

[2] Roberto G. Pacheco, Fernanda D.V.R. Oliveira, and Rodrigo S. Couto. "Early-exit deep neural networks for distorted images: providing an efficient edge offloading". In: *IEEE Global Communications Conference (GLOBECOM)*. 2021, pp. 1–6.

[3] Samuel F Dodge and Lina J Karam. "Quality robust mixtures of deep neural networks". In: *IEEE Transactions on Image Processing* 27.11 (2018), pp. 5553–5562.

[4] Zida Liu, Guohao Lan, Jovan Stojkovic, Yunfan Zhang, Carlee Joe-Wong, and Maria Gorlatova. "Collabar: Edge-assisted collaborative image recognition for mobile augmented reality". In: *ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*. 2020, pp. 301–312.

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

[6] Stefanos Laskaridis, Stylianos I Venieris, Mario Almeida, Ilias Leontiadis, and Nicholas D Lane. "SPINN: synergistic progressive inference of neural networks over device and cloud". In: *International Conference on Mobile Computing and Networking (MobiCom)*. 2020, pp. 1–15.

[7] Lorena Qendro, Alexander Campbell, Pietro Lio, and Cecilia Mascolo. "Early Exit Ensembles for Uncertainty Quantification". In: *Machine Learning for Health*. Vol. 158. 2021, pp. 181–195.

[8] Fang Dong, Huitian Wang, Dian Shen, Zhaowu Huang, Qiang He, Jinghui Zhang, Liangsheng Wen, and Tingting Zhang. "Multi-exit DNN inference acceleration based on multi-dimensional optimization for edge intelligence". In: *IEEE Transactions on Mobile Computing* (2022).

[9] Weiyu Ju, Wei Bao, Dong Yuan, Liming Ge, and Bing Bing Zhou. "Learning early exit for deep neural network inference on mobile devices through multi-armed bandits". In: *2021 IEEE/ACM 21st International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*. 2021, pp. 11–20.

[10] Hongxiang Fan, Mark Chen, Liam Castelli, Zhiqiang Que, He Li, Kenneth Long, and Wayne Luk. "When Monte-Carlo Dropout Meets Multi-Exit: Optimizing Bayesian Neural Networks on FPGA". In: *2023 60th ACM/IEEE Design Automation Conference (DAC)*. 2023, pp. 1–6.

[11] Bing Yu, Hua Qi, Qing Guo, Felix Juefei-Xu, Xiaofei Xie, Lei Ma, and Jianjun Zhao. "Deeprepair: Style-guided repairing for deep neural networks in the real-world operational environment". In: *IEEE Transactions on Reliability* 71.4 (2021), pp. 1401–1416.

[12] Roberto G Pacheco, Mark Shifrin, Rodrigo S Couto, Daniel S Menasché, Manjesh K Hanawal, and Miguel Elias M Campista. "AdaEE: Adaptive Early-Exit DNN Inference Through Multi-Armed Bandits". In: *ICC 2023-IEEE International Conference on Communications*. 2023, pp. 3726–3731.

[13] Divya J Bajpai, Vivek K Trivedi, Sohan L Yadav, and Manjesh K Hanawal. "SplitEE: Early Exit in Deep Neural Networks with Split Computing". In: *arXiv preprint arXiv:2309.09195* (2023).

[14] Roberto G Pacheco, Fernanda DVR Oliveira, and Rodrigo S Couto. "Redes Neurais Profundas com Saídas Antecipadas para Imagens com Distorção em Ambientes de Nuvem". In: *Anais do XLI Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*. SBC. 2023, pp. 532–545.

[15] Roberto G Pacheco, Rodrigo S Couto, and Osvaldo Simeone. "On the impact of deep neural network calibration on adaptive edge offloading for image classification". In: *Journal of Network and Computer Applications* (2023), p. 103679.

[16] Gregory Griffin, Alex Holub, and Pietro Perona. *Caltech-256 object category dataset*. Tech. rep. California Institute of Technology, 2007.

[17] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. "Mobilenets: Efficient convolutional neural networks for mobile vision applications". In: *arXiv preprint arXiv:1704.04861* (2017).

[18] Radosvet Desislavov, Fernando Martínez-Plumed, and José Hernández-Orallo. "Compute and energy consumption trends in deep learning inference". In: *arXiv preprint arXiv:2109.05472* (2021).

AUTHORS

Roberto G. Pacheco received a B.Sc. (with Hons.) degree in electronic engineering from Universidade do Estado do Rio de Janeiro (UERJ). He obtained an M.Sc. degree in 2020 and a D.Sc. degree in 2023 in electrical engineering, both from Universidade Federal do Rio de Janeiro (UFRJ). Currently, he is associate professor at Universidade Federal Fluminense (UFF) since 2023. His primary research interests include edge computing, and early exit deep neural networks.

Fernanda D. V. R Oliveira (IEEE member) graduated as an electronic engineer in 2012 and by the end of 2013 received her M.S. degree in electrical engineering, both from Universidade Federal do Rio de Janeiro (UFRJ), Brazil. In 2015, during her second year as a doctoral student, she did a one year internship in the Microelectronics Institute of Seville. In 2018 she received her D.Sc degree from the Electrical Engineering Program of COPPE/UFRJ. CMOS image sensors and focal plane processing have been her field of study since she was an undergraduate student. Currently, she is a professor at the Federal University of Rio de Janeiro. Her research fields are image sensors, image processing, focal plane processing and artificial intelligence.

Rodrigo S. Couto received a cum laude B.Sc. degree in electronics and computing engineering from Universidade Federal do Rio de Janeiro, Rio de Janeiro, in 2011, and a D.Sc. degree in electrical engineering also from Universidade Federal do Rio de Janeiro in 2015. He has been an associate professor at Universidade Federal do Rio de Janeiro, Rio de Janeiro, since 2018. His primary research interests include Internet of Things, cloud/edge computing, network optimization, and deep learning. Prof. Couto is an associate editor of IEEE Access.