Decision Early-Exit: An Efficient Approach to Hasten Offloading in BranchyNets

Mariana S. M. Barbosa[1], Roberto G. Pacheco[1], Rodrigo S. Couto[1],
Dianne S. V. Medeiros[2], and Miguel Elias M. Campista[1]
[1]*GTA/PEE-COPPE/DEL-Poli – Universidade Federal do Rio de Janeiro (UFRJ)* – Rio de Janeiro, Brazil
[2]*MídiaCom/PPGEET – Universidade Federal Fluminense (UFF)* – Rio de Janeiro, Brazil
{maciel, pacheco, rodrigo, miguel}@gta.ufrj.br, diannescherly@id.uff.br

***Abstract*—Many works study partitioning and early exits in Deep Neural Networks (DNNs) to improve the inference time. Early exits allow the inference of samples in advance, based on the fact that some features are learned at DNNs' initial layers. However, usage of early exits can slightly decrease performance. Partitioning enables the shallowest part of the model to reside at the edge while the deeper layers reside in the cloud. Deciding whether samples must be sent to the cloud at each early exit is time consuming, increasing the total inference time. Hence, reducing this time while maintaining the model performance is currently an open challenge. In this paper, we propose a Decision Early Exit (DEEx), implemented at the first early exit, aiming to reduce the total inference time by skipping unnecessary evaluations at early exits, which may not able to improve the model's performance. To this end, the DEEx compares a predefined decision threshold with the prediction confidence level for each sample and decides whether the sample must be offloaded. We assess DEEx through a comparative analysis that investigates the influence of different values for the decision threshold on the inference time. Our results show that there is a cost benefit between the inference time and the threshold. Using DEEx in a simulated BranchyNet, we can reduce the inference time by around 20% while maintaining the same accuracy achieved when the samples are offloaded.**

***Index Terms*—Deep Neural Networks, Early Exit, Early-Exit DNNs, BranchyNet**

I. Introduction

Deep Neural Networks (DNNs) have become increasingly deeper as computing power increases, using multiple layers to improve the extraction of complex data from the inputs. Deeper DNN models can upgrade their accuracy at the cost of increasing the inference runtime, and the computing device energy consumption. The longer the inference runtime, the higher the total latency to provide a response and the higher the energy consumption. Hence, both the latency and the energy consumption are important metrics to take into account when evaluating deep-learning-based solutions for time-critical applications, such as Intelligent Transport System (ITS) [1], or industrial networks [2], which require low response time [3] and energy cost [4].

Early-exit DNNs emerge as a solution to reduce the inference runtime. Such DNNs compose a new model, with a DNN backbone and several branches departing from this backbone. Early-exit DNNs leverage the fact that different inputs have distinct classification difficulties. Therefore, simpler samples can be inferred with good performance at the side branches, used as early exits. At each early exit, the prediction confidence is used to determine whether the inference can stop at the early exit. More complex samples are inferred in the DNN backbone. As a significant number of samples can be classified earlier at the side branches, the inference runtime is reduced.

Even though Early-exit DNNs allow reducing the inference time, it is still an open challenge to determine whether the sample inference must happen at an early exit and, more than that, at which early exit it should happen [5]–[9]. Notwithstanding, some samples will not be inferred at an early exit because they are not simple enough to achieve high prediction confidence. It is common that, if such a sample does not achieve the required prediction confidence to be early inferred at the few first early exits, it will not achieve the required prediction confidence at any early exit. Hence, the ordinary approach of evaluating the confidence threshold at each early exit before sending the sample to the DNN backbone increases the inference time unnecessarily. In this paper, we propose a Decision Early-Exit (DEEx), which is implemented in the first branch of an Early-exit DNN to hasten the decision of sending samples to the DNN backbone. The DEEx compares a predefined decision threshold with the required prediction confidence to decide whether the sample should be inferred at the early exits or offloaded to the DNN backbone. The goal is to reduce the inference time while simultaneously ensuring the model's accuracy. We assess our proposal using the CIFAR-10 dataset. We evaluate the influence of the decision threshold on the inference runtime and the ratios of samples inferred at the early exit and offloaded to the DNN backbone. We also investigate the influence of the decision threshold on inference accuracy. The results show that DEEx can hasten the decision between early exits and DNN backbone whit a classification accuracy greater than 0.75 for thresholds from 0.5.

This paper is organized as follows. Section II reviews relevant papers about Early-exit DNNs. We present our proposal in Section III. Section IV describes the CIFAR-10 dataset and how we train the Early-exit DNN. Section V presents and discusses the results. Finally, Section VI concludes this paper and anticipates research directions.

II. Related Work

DNNs have high learning performance and can extract information from complex data in a hierarchical way [6]. Certain characteristics can be learned in the DNN's initial layers, enabling the shallowest layers of this network to provide an appropriate inference of a fraction of the input data [6]. In this regard, many researchers propose solutions that aim to benefit from this characteristic, resulting in faster inference, while keeping the deeper layers for challenging tasks [4].

The BranchyNets are DNNs with side branches inserted in the intermediate layers, so that simpler samples can be early inferred and, hence, reduce the inference time [6]. Figure 1 illustrates a DNN architecture, which is composed of a standard AlexNet [10] with two early exits added, for this approach. The AlexNet is composed of five convolutional (CONV) layers followed by three Fully-Connected (FC) layers. Each side branch has a FC layer that produces a vector, then calculates the *softmax* layer, in which the elements of the probability vector represent the probability of the sample in every class. The confidence level is based on the probability vector. BranchyNets use entropy as the metric to evaluate the prediction confidence at each early exit, comparing it to a threshold that determines a limit for the confidence level [6]. The results show that BranchyNets present high accuracy and reduce the inference time, compared to using only the traditional DNNs, i.e., DNNs without early exits.

Pacheco and Couto [8] tackle the problem of BranchyNet partitioning. This problem consists of selecting the partitioning layer that splits the BranchyNet model into two parts. The first part is processed at the edge device (i.e., from the input layer to the partitioning layer), while the cloud server runs the remaining layers. To this end, the authors model an Early-exit DNN as a Directed Acyclic Graph (DAG), in which the vertices correspond to the DNN layers and the connections between the layers are the directed links. The link cost is based on the processing time weighted by the probability to reach a given side branch. Therefore, the expected inference time also considers offloading inputs to the cloud, introducing a time to upload to the cloud server. The authors' modeling allows considering the partitioning problem as a shortest path one, which can be solved in polynomial time.

In another work, Pacheco et. al. investigate the performance of Early-exit DNNs varying the number of side branches and threshold values to assess the classification confidence [9]. They use an early-exit DNN architecture based on the pre-trained MobiliNetV2 model, varying the number of side branches from 2 to 5. The initial layers and side branches are processed in the edge and the remaining part is processed in the cloud. For the side branches, the sample confidence level is the maximum value of the probability vector. The sample inference occurs at the early exit if the classification confidence is greater than the threshold. The analysis provides a relationship between edge inference and the classification confidence level. The results suggest that inferences on edge devices improve the inference time, ensuring good classification confidence. All previous works use a decision threshold at each early exit to determine if the sample is inferred at the side branch or sent to the next exit. In contrast, we propose a Decision Early Exit that uses the decision threshold at the first early exit to determine whether the sample is offloaded to the DNN backbone, skipping all early exits.

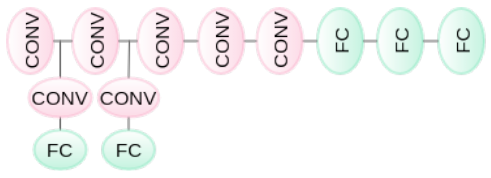


Fig. 1. B-AlexNet architecture with two early exits, the main side branch is composed of five convolutional (CONV) layers followed by three fully-connected layers. The last fully-connected (FC) layer produces a probability vector of output, equal to the possible classes presented in the dataset. Then, the *softmax* function produces the inference results.

III. Decision Early-Exit

In this paper, we propose to hasten the decision of offloading samples to the DNN in the cloud or infer such samples at the early exits in the edge. The idea is to maintain high accuracy while simultaneously reducing the inference time by skipping unnecessary evaluations at each early exit. To this end, The BranchyNet can be composed of N early exits in the edge network. As such, when the sample inference occurs at any early exit, we prevent the addition of time due to offloading. Consequently, the total time spent processing the sample and obtaining the result is automatically reduced, as we avoid offloading and, thus, the high communication delay, which can be significant in this scenario. Figure 2 shows a partitioned Early-exit DNN, highlighting the Decision Early Exit (DEEx) proposed in this work.

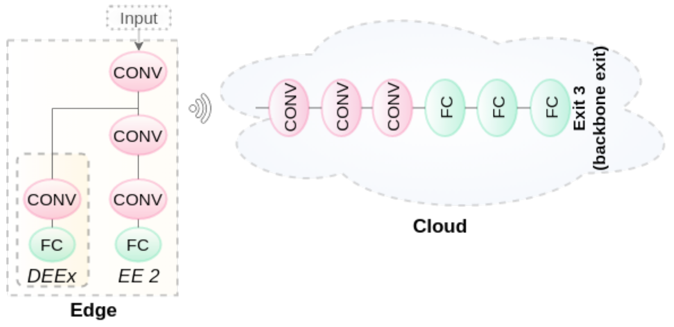


Fig. 2. A partitioned BranchyNet. The early exits reside at the edge, whereas the DNN backbone resides in the cloud. We use the first early exit as Decision Early Exit (DEEx) to evaluate the classification confidence of the sample, comparing it to a threshold which we present as decision threshold p_d.

The DEEx is implemented at the first early exit, following the rationale of skipping the side branches the earliest possible to avoid additional time due to the evaluation of the sample confidence level at each side branch unnecessarily. Such samples are not simple enough to be early inferred and time should not be wasted evaluating their confidence level. Hence, the first exit is responsible for evaluating the confidence level to determine whether the sample should continue to be processed by the DNN backbone in the cloud. If the confidence level is achieved, the sample is sent to the early exit, i.e., the second branch in this case. Otherwise, it is processed by the DNN backbone. It should be noted that there are two processes involved in our proposal. The first one is related to the decision taken at Early Exit 1 (DEEx), and the second is related to the decision taken at the following early exits.

The first process decides whether the sample inference occurs at an Early Exit (EE) or the DNN backbone. Upon deciding in favor of the EE, the second process decides whether the sample inference occurs locally at EE K, with $2 \leq K < N-1$, or at the next EE. When reaching EE $N-1$, the sample inference occurs independently of the confidence level. Note that we do not consider offloading the sample to Exit N, because the first process decided that the inference should happen at an EE. The idea is to highlight the behavior of the proposal when sending the samples in advance to the DNN backbone. Each EE computes the classification confidence through the maximum value of the probability vector. In the first process at the DEEx, if the classification confidence is greater than or equal to the decision threshold p_d, the sample is sent to EE 2. Otherwise, it is sent to Exit N, skipping the side branches. In the second process, at each EE K, if the classification confidence is greater than or equal to the threshold p_{tar}, the sample inference occurs locally. The sample is sent to the next early exit if the classification confidence is lower than the decision threshold until reaching EE $N-1$, where the inference occurs without using the cloud. In this paper, we implement a simple scenario, where we use two early exits to evaluate our proposal.

IV. Dataset and Training

We use the CIFAR-10 dataset [11], which contains $60,000$ samples divided equally into 10 classes (plane, car, bird, cat, deer, dog, frog, horse, ship, truck). We use the dataset in two different evaluation scenarios. In the first scenario, we use a subset of the dataset, considering only 2 classes (cat, dog), which results in $12,000$ samples. We refer to this subset as the binary dataset. We use the hold-out method to split the binary dataset into test, training, and validation. The test dataset contains $2,000$ unseen samples, i.e., 16.67% of samples. The training dataset has $10,000$ samples (83.33%), from which 10% are separated for validation. In the second scenario, we consider the entire CIFAR-10 dataset, using the hold-out method to split it into $10,000$ samples (16.67%) for testing and $50,000$ samples (83.33%) for training, from which 10% are separated for validation. We use the Pythorch[1] library from Python to train and test the Early-exit DNN architecture used in this paper shown in Figure 2. The architecture is composed of a standard AlexNet [10] as the DNN backbone (Exit 3), and two side branches, EE 1 (DEEx) and EE 2. We start the training from the pre-trained AlexNet, to ensure that all exits can jointly work. We use cross-entropy as the loss function and the SGD optimizer at a learning rate of 0.001.

[1]https://pytorch.org/

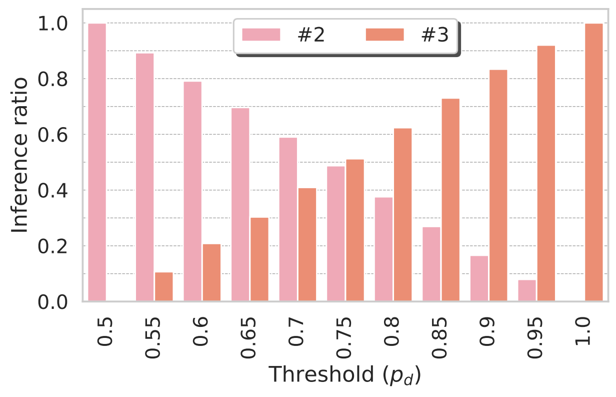
V. Results and Discussion

We assess our proposal through a comparative analysis of the results obtained for the inference performed at EE 2 and the DNN backbone (Exit 3). We investigate how the increase in the decision threshold influences the model accuracy, the inference time, and the ratios of samples inferred locally and offloaded to the DNN backbone. We also evaluate the confusion matrix for the model in three scenarios, when the samples are classified only at EE 2, only at Exit 3, and both exits have the same accuracy. We consider the first threshold value greater than the equal values in the classification probability vector. For instance, the $p = 0.5$ to values in the probability vector to binary dataset means that both classes have equal probability.

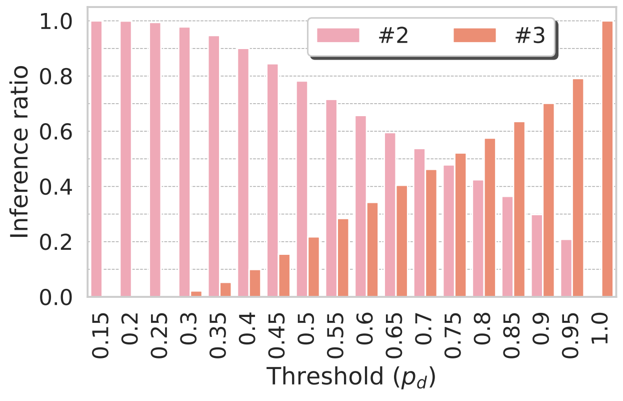
A. Samples and Inference Time

Initially, we analyze the number of samples classified at each exit for different values of p_d, i.e., the ratio of samples inferred at an exit. When one exit classifies all samples, the inference ratio is equal to 1. Figure 3 shows the inference ratio at each exit as a function of p_d. At first, for $p_d = 0.5$ for Figure 3(a), all samples are inferred at EE2 (#2). As p_d increases, the offloading begins and, for $p_d = 1$, all samples are inferred at the DNN backbone (#3). Figure 3(a) shows that, for the binary dataset, when $p_d = 0.55$ some samples are already sent to Exit 3 and the majority of samples are processed at this exit when $p_d = 0.75$. The behavior is similar for the entire CIFAR-10 dataset, as shown in Figure 3(b), but sample offloading begins for $p_d = 0.20$.

Knowing that the samples are gradually sent to the DNN backbone as p_d increases, we evaluate the time spent processing the test dataset as a function of p_d. Figure 4 shows the results. We do not consider the communication delay to offload the samples. To measure our experiments, we use Google Colaboratory, which is a cloud computing service for machine learning equipped with an Intel CPU whit two cores at 2.20GHz, 16 GB VRAM, and GPU Tesla T4. We run the evaluation on the test dataset 10 times and calculate the average inference time with a confidence interval of 95%. The inference time is due to the time spent to infer all samples in the test dataset only. We observe in Figure 4 that the inference time increases as the p_d increases. This happens because more samples are processed by the DNN backbone when p_d increases at DEEx. For the binary dataset, Figure 4(a) shows that processing all samples in the Exit 3 takes up to 7.1 s, while processing all samples at EE 2 takes up to 4.2 s, i.e., 59% of the total time, meaning a reduction of 41%. For the entire dataset, Figure 4(b) shows that the Exit 3 needs 34.0 s to process all samples, while EE 2 needs only 20.9 s, a reduction of 38.5%. The inference time for the inference at the DNN backbone can be even greater if we consider that sending data to the cloud can add a significant communication delay. However, there is an accuracy-time trade-off to each application [12]. For instance, healthcare applications need high accuracy, while applications for industrial maintenance can have a bit lower accuracy but need a quick response to

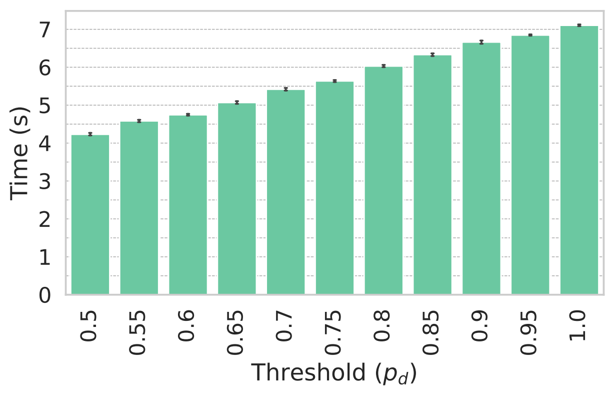


(a) Binary dataset. Samples are sent to Exit #3 from $p_d = 0.55$.

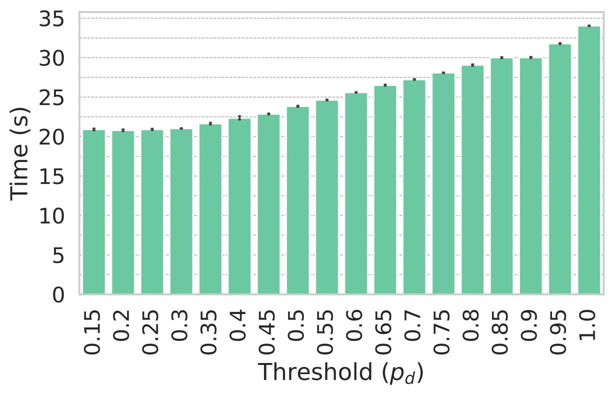


(b) Entire dataset. Samples are sent to Exit #3 from $p_d = 0.20$.

Fig. 3. The proportion of inference at each exit as a function of p_d. The ratio of samples processed by the DNN backbone (Exit #3) is directly proportional to p_d. The majority of samples are processed by the DNN backbone when $p_d = 0.75$.



(a) The inference time can reduce of $7.1s$ to $4.2s$ in binary set when we consider lower p_d values.



(b) For full CIFAR-10 set, the inference time can reduce by up to 38.57% if we reduce p_d.

Fig. 4. The spend time to infer the test set for different p_d values. Initially, the inferences occur only in EE 2, when the p_d increases the samples are sending to (Exit 3) growing the inference time of the architecture.

identify abnormal events. Hence, the latter can benefit from the time reduction provided by early exits [4].

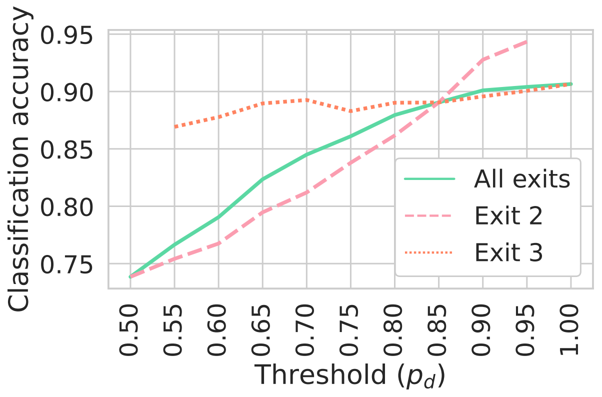
B. Accuracy

The previous results show that the inference time is, indeed, reduced when processing the samples at EE 2. Nevertheless, this is not beneficial if the inference accuracy drops significantly when samples are processed at the early exit. Figure 5 shows the accuracy for each exit as a function of the p_d. At first, Exit 3 does not participate in the inference, because all samples are classified at an early exit. The overall accuracy (All exits) is calculated as the samples correctly classified in both exits by total samples. For the binary dataset, Figure 5(a) shows that the accuracy at this point is approximately equal to 0.75 for EE 2 and, consequently, for the overall architecture, as nothing is sent to the cloud. For the entire dataset, as shown in Figure 5(b), the accuracy at this point is approximately equal to 0.7. In turn, the accuracy is approximately equal to 0.91 for both the binary and the entire datasets when all samples are processed in the cloud.

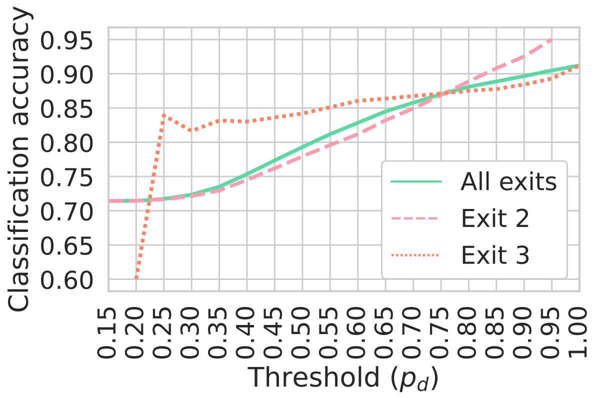
The accuracy gradually increases when p_d increases, for both exits and the overall architecture in both scenarios. For the binary dataset, when $p_{cross} = p_d = 0.85$, the accuracy for EE 2 surpasses the accuracy for Exit 3. For the entire dataset, the crossing point is given by $p_{cross} = p_d = 0.75$. From Figure 3, we find that $26,95\%$ of samples in the binary dataset are classified at EE 2 for $p_d = 0.85$, while 47.84% of samples in the entire dataset are classified at EE 2 for $p_d = 0.75$. The crossing point in both scenarios happens because $p_d \geq p_{cross}$ the higher value of p_d ensures that only samples with high classification confidence are processed at the early exit, i.e., only the simpler samples remain to be inferred in the early exit. As increasing p_d increases the accuracy but also increases the inference time, the optimal p_d value is a relation between the time and the accuracy level that the application requires.

C. Confusion Matrix

We evaluate the confusion matrix when all samples are processed at EE 2, at Exit 3, and when $p_d = p_{cross}$ for both



(a) The cross point for the binary dataset is 0.85. 26.95% of sample are inferred at EE 2.

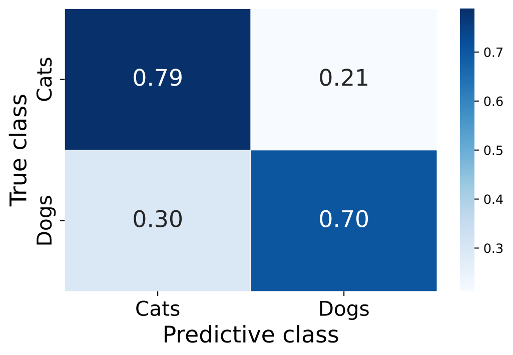


(b) The cross point is equal to 0.75. 47.84% of sample are inferred at EE 2.

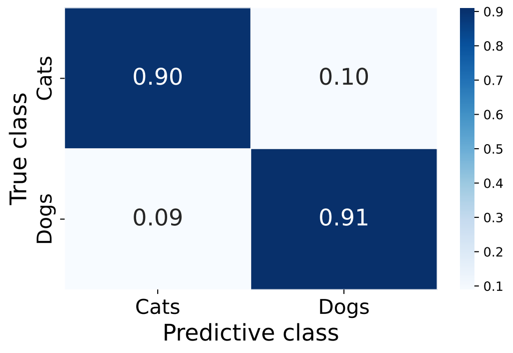
Fig. 5. The classification accuracy for each exit and the full architecture as a function of the p_d. The sending of samples to the Exit 3 improves the classification accuracy. When $p_d \geq p_{cross}$, the accuracy for EE 2 surpass the accuracy for Exit 3.

datasets. Figure 6(a) shows the confusion matrix for the binary dataset when all samples are classified at the EE 2. We observe that 70% of Dogs are correctly classified, while 79% of Cats are correctly classified. This performance is improved when all samples are classified at Exit 3, as shown in Figure 6(b), with 90% of Cats and 91% of Dogs correctly classified. For $p_d = 0.85$, the true predictions are 89% for Cats and 90% for Dogs, which is very similar to the results in Figure 6(c). However, the reduction in the inference time is approximately 10% compared to when $p_d = 0.85$ with $p_d = 1$ (not using the early exit), according to Figure 4(a).

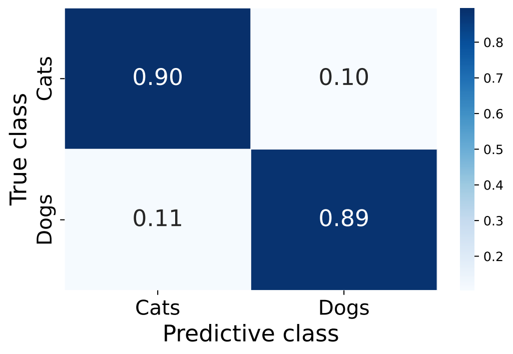
Figure 7 shows the confusion matrix for the entire CIFAR-10 dataset. When the EE 2 classifies all the samples, the true prediction varies from 0.5 to 0.8, depending on the class, as shown in Figure 7(a). This value varies from 0.8 to 0.9 when all samples are processed at Exit 3, as shown in Figure 7(b). For $p_{cross} = 0.75$, Figure 7(c) shows that the true prediction also varies from 0.8 and 0.9. Hence, we can achieve a very similar performance but reduce the inference time by approximately 20% (Figure 4(b)), benefiting time-critical applications.



(a) Early Exit 2.



(b) Exit 3.

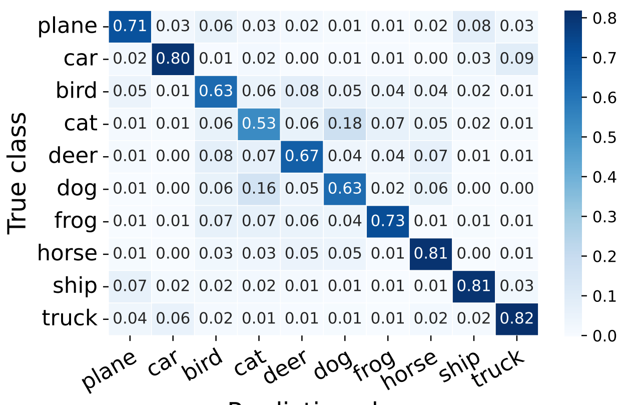


(c) Crossing point, $p_{cross} = 0.85$.

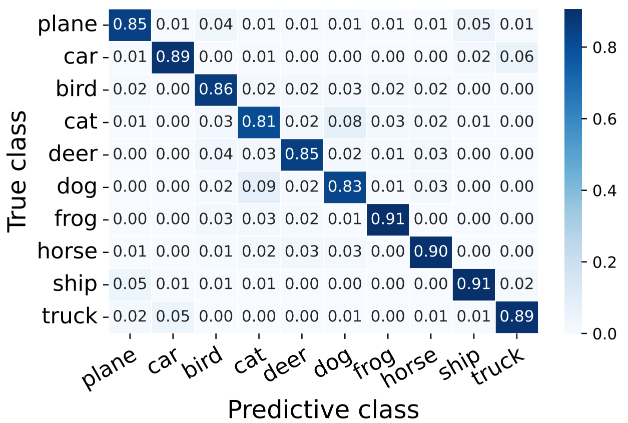
Fig. 6. Confusion matrix for the binary dataset. For p_{cross} the true prediction is similar to the best case, which occurs when all samples are processed in Exit 3. The inference time reduction at this point is approximately 14%.

VI. Conclusion

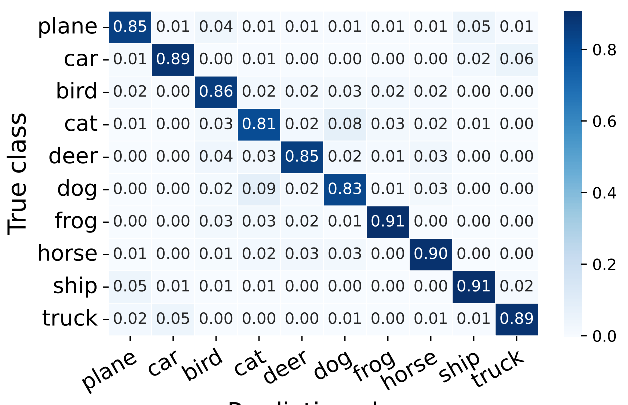
We proposed a Decision Early Exit (DEEx) to anticipate the sending of samples to the backbone exit through a threshold p_d when the classification confidence is less than p_d. we aim to reduce the inference time by avoiding processing the sample in all EE for then sending it to the backbone exit. Our approach uses the BranchyNet architecture with three exits. The first early exit is the DEEx which evaluates the classification



(a) Early Exit #2.



(b) Exit #3.



(c) Crossing point, $p_{cross} = 0.75$

Fig. 7. Confusion matrix for the entire dataset. For p_{cross} the true prediction is similar to the best case, which occurs when all samples are processed in Exit #3. The inference time reduction at this point is approximately 29%.

confidence to decide whether the sample should be processed. We analyze our proposal by increasing the p_d value, using the CIFAR-10 dataset. We assess the relationship between the inference time and the sample ratio at the exit. Moreover, we evaluate the accuracy behavior with the p_d values. We show that the inference time can reduce 41% and 38.5% for the binary and full datasets, respectively. However, the backbone exit receives more samples with p_d values. Consequently, the accuracy can increase to 0.91 for both datasets. Furthermore, we analyze the confusion matrix showing the close behavior between $p_d = p_{cross}$ and the backbone exit.

In future work, we plan to extend the DEEx to deeper network models and larger datasets. Furthermore, we will implement the architecture partitioning between edge and cloud to analyze the p_d effect in the inference time, including more EE to show the benefit of skipping the EE. In addition, we will extend our proposal to send the sample to the cloud when the p_{tar} does not reach exit $N-1$. Finally, we intend to partition the BranchyNet between device, edge, and cloud, each with different resources.

Acknowledgment

This paper was partially supported by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001. It was also supported by CNPq, FAPERJ Grants E-26/211.144/2019 and E-26/202.689/2018, FAPESP Grant 15/24494-8, and RNP.

References

[1] T. Wu, P. Zhou, K. Liu, Y. Yuan, X. Wang, H. Huang, and D. O. Wu, "Multi-agent deep reinforcement learning for urban traffic light control in vehicular networks," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 8, pp. 8243–8256, 2020.

[2] L. Zeng, E. Li, Z. Zhou, and X. Chen, "Boomerang: On-demand cooperative deep neural network inference for edge intelligence on the industrial internet of things," *IEEE Network*, vol. 33, no. 5, pp. 96–103, 2019.

[3] M. Xu, F. Qian, M. Zhu, F. Huang, S. Pushp, and X. Liu, "Deepwear: Adaptive local offloading for on-wearable deep learning," *IEEE Transactions on Mobile Computing*, vol. 19, no. 2, pp. 314–330, 2019.

[4] K. Bochie, M. S. Gilbert, L. Gantert, M. S. Barbosa, D. S. 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.

[5] 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 *International Conference on Mobile Computing and Networking (MobiCom)*, 2020, pp. 1–15.

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

[7] ——, "Distributed deep neural networks over the cloud, the edge and end devices," in *IEEE International Conference on Distributed Computing Sstems (ICDCS)*, 2017, pp. 328–339.

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

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

[10] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Neural Information Processing Systems*, vol. 25, 2012.

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

[12] M. Hosseinzadeh, A. Wachal, H. Khamfroush, and D. E. Lucani, "Optimal accuracy-time trade-off for deep learning services in edge computing systems," in *ICC 2021-IEEE International Conference on Communications*. IEEE, 2021, pp. 1–6.