Efficient Deployment of Early-Exit DNN Architectures on FPGA Platforms

DATE PhD Forum 2024

Anastasios Dimitriou
School of Electronics and Computer Science
University of Southampton, United Kingdom
ad1r20@soton.ac.uk

Advisors: Geoff V. Merrett & Jonathon Hare
School of Electronics and Computer Science
University of Southampton, United Kingdom
{gvm, jsh2}@ecs.soton.ac.uk

I. Introduction

Deep Neural Networks (DNNs) have demonstrated remarkable success in various cognitive applications, benefiting from innovative network architectures such as AlexNet [7], ResNet [6], VGG [11] and Transformers [13]. However, their success is immediately connected with an increase in depth and parameter count resulting in higher computational burdens and greater power consumption. This limits their deployability and arise challenges in adapting to resource-restricted devices and varying computational budgets.

Dynamic neural networks [5] are approaches that strategically allocate computations by their ability to adapt network structures or parameters during inference. The core idea lies in recognizing that different inputs have different computational requirements. Developing techniques to abstract this information significantly reduces computations and enhances the efficiency, capacity, and compactness of the networks.

The focus of this research lays on Early Exiting [8]. It is a structure-focused dynamic approach, enabling the inference process to stop at shallow exits, when the network is confident in recognizing the input. In order to enable an early exit, intermediate classifiers are placed in-between each or sets of layers. These are very small decision sub-networks containing convolutional, fully-connected (FC), or pooling layers. Based on this architecture early-exit can be implemented with confidence-based criteria [12], [9], usually using a softmax's output, or learned decision functions [1], [3].

Despite their effectiveness though, we argue that there is a gap between theoretical findings and practical applications. The majority of early exit networks are designed using libraries optimized for static models, resulting in inconsistency between anticipated and actual results. Furthermore, these networks are typically accelerated and evaluated on conventional CPU-GPU systems, platforms that cannot meet the real-time processing requirements in embedded devices [2].

In this research we target FPGAs, a platform that has already been proven to be very effective in accelerating CNNs [10]. They are programmable devices that allow users to implement digital circuits custom-designed for a specific task. Their parallelization capabilities can accelerate very effectively a large amount of neural network operations, while maintaining very low levels of power consumption. In this work we highlight some of the difficulties on realising early-exit networks on FPGAs and propose two different designs *pipeline* and *parallel*. They are experimental implemented and evaluated on a ZCU106 FPGA, comparing accuracy, execution time, power and energy consumption against a desktop CPU, CPU+GPU and Nvidia Jetson Xavier. Finally, we analyse the placement of the exit points on a ResNet32 and taking advantage of the *parallel* architecture, further accelerate the dynamic inference.

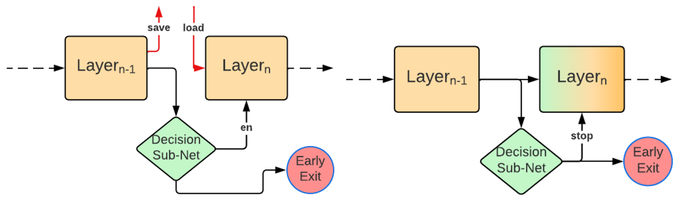

Fig. 1. Explored designs for (a) Pipeline and (b) Parallel approaches.

II. Limitations in Designing Dynamic Networks Architectures on FPGAs

Programming and testing FPGAs targeting complex applications like neural networks can be difficult and time consuming. Hence many FPGA manufacturers and researchers put a lot of effort on creating mapping tool-flows for DNNs (e.g. Xilinx's Vitis AI, Nvidia's NVDLA, fpgaConvNet [14], Angel-Eye[4] etc.). They are connected with popular static network libraries (caffe, tensorflow, torch) and produce complete and inaccessible designs. However, the structure of dynamic networks demands the use of custom layer types and interfere on intermediate stages of the backbone network, which renders the tools incapable to produce efficient designs. Therefore we used an architecture for the backbone network similar to ones produced from the tools, and designed from scratch the decision sub-networks.

In the targeted approaches, these sub-networks are responsible to make a decision on stopping inference. To achieve that they use the intermediate layer outputs, which generates an issue as these data is of vital importance when an early exit does not happen. In order to maintain them, while the sub-networks are executing, we can use buffers or the external memory. However the first option is not always feasible, as these outputs can be very large and the second has severe effects on the latency and the memory footprint of inference. A way to efficiently eliminate this issue is presented below.

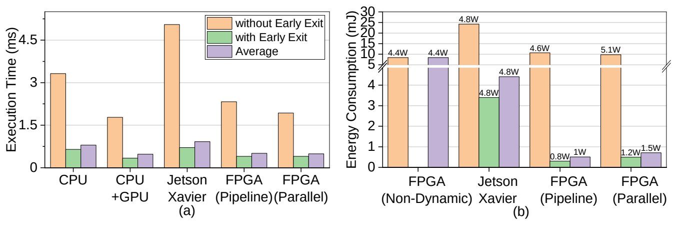

Fig. 2. Experimental results comparing (a) Execution time and (b) energy consumption per sample. Average values are calculated based on the frequency of the decision to early exit, 94.37% yes and 5.63% no.

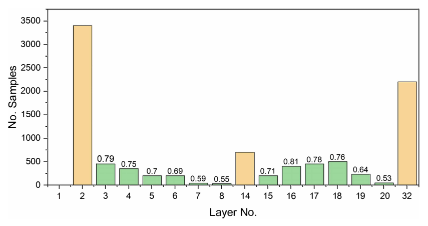

Fig. 3. Number of samples to be firstly correctly predicted in the next 6 layers after being rejected from the predetermined exit points on ResNet-32. On top of bars is the average confidence they had on the original exit points.

III. FPGA Realisation of Early-Exit Dynamic Networks

We explore two different FPGA architecture for early-exit networks, focusing on the implementation of the decision sub-networks. The ***pipeline*** approach (fig.1 (a)) follows the traditional early exiting architecture. Inference is paused until a decision is made, based on which the network either stops, generates a prediction and proceeds with the next input, or continues with the deeper layers. As the sub-networks contain the same type of layers as the backbone, this architecture enables reusing the already designed IPs, leading to very compact designs. However for the reason mentioned above, it requires the usage of external memory.

The ***parallel*** approach (fig.1 (b)) takes advantage of the FPGA parallelization capabilities and contains a separate module that is designed specifically to execute the decision sub-network. In this architecture the intermediate layer output is fed simultaneously to the next layer of the main network and to the decision branch. Thus, it removes the need of pausing inference as well as storing the intermediate outputs to memory. It achieves lower latency and reduces the memory footprint, however it utilises more of the platform's resources, increasing the energy demands.

The core of both designs is an Array of Processing Elements (PEs). They are hardware configurations that replace a pipeline structure with an array of homogeneous PEs, capable of performing a common mathematical operation. These elements are locally interconnected, and able to synchronously communicate with each other. It accelerates the feature-weight multiplications that dominate Convolution layers. Concerning the FC layers a module of multipliers connected to an adder tree is implemented. Inputs and weights are split into equal parts, and calculated separately. Finally max pooling layers are implemented using a sliding window while ReLu activation function is immediately applied to the results.

To verify, test and compare the performance of the designs we follow the BranchyNet's [12] approach on confidence-based early exiting. Figure 2 (a) shows the per sample execution time of the early-exit LeNet-5 using the MNIST data-set. We observe that the dynamic DNN is always faster, highlighting the effect of the approach. Furthermore focusing on our designs, the ***pipeline*** approach is **1.5x** faster than a desktop CPU and **1.8x** than the Jetson. The ***parallel*** approach further accelerates inference by **1.2x** over the ***pipeline***, in cases where an early exit is not decided, while avoiding a 2.88kB data transfer to the memory and back.

Concerning energy consumption the dynamic DNN is again consistently less energy demanding than the non-dynamic. The *pipeline* and the *parallel* approaches also showcase comparatively low energy consumption (Fig.2 (b)), requiring 12.68 mJ and 15.69 mJ per-sample respectively. This equates to **6.9x** and **5.6x** less energy than on the Jetson, and **2.7x** and **2.2x** than the inference of the backbone on the FPGA. For both energy and time we achieved similar results while evaluating the designs using AlexNet, VGG19 and ResNet32.

IV. Early-Exit Placement Optimisation

Early-exit architecture dictates that the placement of the exit points is decided before the deployment and cannot change during execution. However, we found that on a ResNet32 (Fig.3) network with an early-exit branch after every layer, 31% of the rejected outputs from the original exit points (after 2nd and 14th layer) could be correctly recognised by the next 6 layer, rather than continuing to the next original exit points (after 14th and 32nd layer). Furthermore the average confidence of these samples was very close to the original points' confidence threshold (Fig.3).

Taking advantage of the versatility that the ***parallel*** approach provides, as the dedicated hardware can be used in between every layer, we explore a confidence-controlled placement of the early-exit points. When the difference between the confidence level of a rejected sample and the threshold of the original exit point is small, we try to early exit before the next predetermined point. In such manner we reduced the computations of a ResNet32 dynamic network by 24% while maintaning the same levels of accuracy.

V. Conclusion

In this paper we explored the benefits and limitations of realising dynamic DNN approaches on FPGAs. We propose two efficient early-exit FPGA architectures, the ***pipeline*** and the ***parallel***, focusing on the better implementation of the decision sub-network. Finally using the ***parallel*** approach's versatility we explore a dynamic placement of the early-exit branch further improving the efficiency of the dynamic network and the FPGA platform.

Acknowledgments

This work was supported by the Engineering and Physical Sciences Research Council (EPSRC) under EP/S030069/1. Data associated with this paper is available on https://doi.org/10.5258/SOTON/D2885.

References

[1] T. Bolukbasi, J. Wang, O. Dekel, and V. Saligrama. “Adaptive Neural Networks for Efficient Inference”. In: *International Conference on Machine Learning*. 2017.

[2] J. Cong and B.-Y. Xiao. “Minimizing Computation in Convolutional Neural Networks”. In: *International Conference on Artificial Neural Networks*. 2014.

[3] X. Dai, X. Kong, and T. Guo. “EPNet: Learning to Exit with Flexible Multi-Branch Network”. In: *Proceedings of the 29th ACM International Conference on Information & Knowledge Management* (2020).

[4] K. Guo, L. Sui, J. Qiu, J. Yu, J. Wang, S. Yao, et al. “Angel-Eye: A Complete Design Flow for Mapping CNN Onto Embedded FPGA”. In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 37 (2018), pp. 35–47.

[5] Y. Han, G. Huang, S. Song, L. Yang, H. Wang, and Y. Wang. “Dynamic Neural Networks: A Survey”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44 (2021), pp. 7436–7456.

[6] K. He, X. Zhang, S. Ren, and J. Sun. “Deep Residual Learning for Image Recognition”. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 770–778.

[7] A. Krizhevsky, I. Sutskever, and G. E. Hinton. “ImageNet Classification with Deep Convolutional Neural Networks”. In: *Advances in Neural Information Processing Systems*. Ed. by F. Pereira, C. Burges, L. Bottou, and K. Weinberger. Vol. 25. Curran Associates, Inc., 2012.

[8] S. Laskaridis, A. Kouris, and N. D. Lane. “Adaptive Inference through Early-Exit Networks: Design, Challenges and Directions”. In: *Proceedings of the 5th International Workshop on Embedded and Mobile Deep Learning* (2021).

[9] Leroux, Sam and Bohez, Steven and De Coninck, Elias and Verbelen, Tim and Vankeirsbilck, Bert and Simoens, Pieter and Dhoedt, Bart. “The cascading neural network : building the Internet of Smart Things”. eng. In: *KNOWLEDGE AND INFORMATION SYSTEMS* 52.3 (2017), 791–814.

[10] S. Mittal. “A survey of FPGA-based accelerators for convolutional neural networks”. In: *Neural Computing and Applications* 32 (2018), pp. 1109–1139.

[11] K. Simonyan and A. Zisserman. “Very Deep Convolutional Networks for Large-Scale Image Recognition”. In: *International Conference on Learning Representations*. 2015.

[12] S. Teerapittayanon, B. McDanel, and H. T. Kung. “BranchyNet: Fast inference via early exiting from deep neural networks”. In: *2016 23rd International Conference on Pattern Recognition (ICPR)* (2016), pp. 2464–2469.

[13] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, et al. “Attention is All you Need”. In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, et al. Vol. 30. Curran Associates, Inc., 2017.

[14] S. I. Venieris and C.-S. Bouganis. “fpgaConvNet: Mapping Regular and Irregular Convolutional Neural Networks on FPGAs”. In: *IEEE Transactions on Neural Networks and Learning Systems* 30 (2019), pp. 326–342.