Exploration of Decision Sub-Network Architectures for FPGA-based Dynamic DNNs

Anstasios Dimitriou[1], Mingyu Hu[2], Jonathon Hare[3], Geoff V. Merrett[4]
School of Electronics and Computer Science, University of Southampton, UK
Email: {[1]ad1r20, [2]mh1u20}@soton.ac.uk, { [3]jsh2, [4]gvm}@ecs.soton.ac.uk

***Abstract*—Dynamic Deep Neural Networks (DNNs) can achieve faster execution and less computationally intensive inference by spending fewer resources on easy to recognise or less informative parts of an input. They make data-dependent decisions, which strategically deactivate a model's components, e.g. layers, channels or sub-networks. However, dynamic DNNs have only been explored and applied on conventional computing systems (CPU+GPU) and programmed with libraries designed for static networks, limiting their effects. In this paper, we propose and explore two approaches for efficiently realising the sub-networks that make these decisions on FPGAs. A *pipeline* approach targets the use of the existing hardware to execute the sub-network, while a *parallel* approach uses dedicated circuitry for it. We explore the performance of each using the *BranchyNet* early exit approach on LeNet-5, and evaluate on a Xilinx ZCU106. The *pipeline* approach is 36% faster than a desktop CPU. It consumes 0.51 mJ per inference, 16x lower than a non-dynamic network on the same platform and 8x lower than an Nvidia Jetson Xavier NX. The *parallel* approach executes 17% faster than the *pipeline* approach when on dynamic inference no early exits are taken, but incurs an increase in energy consumption of 28%.**

***Index Terms*—dynamic DNNs, hardware architecture for machine learning, FPGA**

I. Introduction

Deep Neural Networks (DNNs) have become popular due to their effectiveness and accuracy in solving many real-life machine learning problems including computer vision [1] and voice recognition [2]. Their success, however, comes at the cost of increased depth, complexity and computational burden, leading to very high energy demands. Dynamic inference is an emerging approach that utilises information from the input data to selectively execute only important subsets of the DNN, e.g. layers [3], channels [4] or sub-networks [5].

Key components of a dynamic DNN are the decision sub-networks. They are typically placed between layers, with a purpose of deciding the parts of the DNN to execute. While this adds computation to the network, with careful design and training the overall network performance is accelerated substantially [6], [7], improving inference time and efficiency, and enabling inference on more constrained computing platforms. However, we argue that there is a gap between theoretical results and practical implementations [8]. Most dynamic DNN approaches are developed using libraries fine-tuned for static DNN models leading to inconsistency between expected and actual efficiency. In addition, they are applied and tested on conventional CPU-GPU systems.

To address this, we explore decision extraction architectures for FPGA-based dynamic DNNs. FPGAs have been shown to offer enhanced configurability and parallelization capabilities to efficiently accommodate the computational needs of DNNs, while achieving very low energy consumption. The novel contributions of this paper are:

- Consideration for realising and implementing dynamic DNNs on FPGAs, exploring *pipeline* and *parallel* approaches for decision sub-networks.
- An experimental implementation and evaluation on a ZCU106 FPGA, comparing accuracy, execution time, power and energy consumption against a desktop CPU, CPU+GPU and Nvidia Jetson Xavier.
- Experimental results show faster execution time and lower energy consumption, and that a parallel approach to decision sub-network implementation can decrease latency at the cost of increased energy consumption.

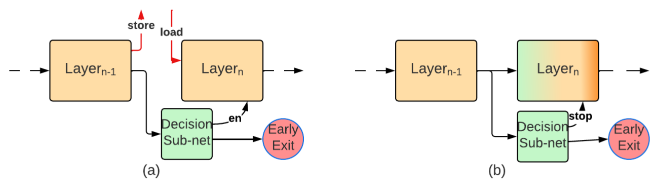

Fig. 1. Explored designs for (a) Pipeline and (b) Parallel approaches to Decision Sub-Networks.

II. Decision Sub-Network Architectures

Decision sub-networks are small neural networks, containing convolutional, fully-connected (FC), or pooling layers [7], [9]. Since these components are the same as those in the backbone network, there is an opportunity for reusing existing IP, minimising the area and power overhead. This, however, leads to some under-utilisation of existing components (the backbone is stalled while the sub-network is computed), but can be useful in cases where the target devices have very limited resources. This *pipeline* (fig. 1(a)) approach executes inference as normal up until the layer before the decision sub-network. Then it is stalled and the layer's output is stored while the sub-network executes. Following this decision, inference either stops and restarts with the next input sample (if the early exit is taken), or else the stored output is loaded and continues with the next layer (if the early exit is not taken).

Alternatively a *parallel* (fig. 1(b)) approach executes both the backbone and sub-network at the same time. In this case, the output of the previous layer is fed to both the next layer and the decision sub-network, and they are both activated. This reduces the latency overhead while computing the decision, as if the early exit is not taken, the backbone network has already started executing. If the decision is to exit, the layer stops prematurely and the network proceeds with the the next input sample. Additionally, in this design there is no need to store the intermediate output as it's already fed to the next layer, so the memory footprint is also reduced. However simultaneous activation of both paths requires the design of separate IPs for the sub-network, increasing the area and power/energy demands.

III. Hardware Implementation

As mentioned above the decision sub-networks and DNNs share the same components, so their realisation on an FPGA require the implementation of the different network layers. An array of process elements is used for convolutional layers. Input data is fed into a sliding window with a same size as the kernel's. Input-kernel multiplication is calculated by the PEs, and outputs then moved to an adder tree. This adds the biases and produces the final convolution result. The activation function (ReLU), which is just a conditional branch, is immediately applied and the results are stored in buffers that feed the next layers.

The architecture of FC layers is similar to that of the convolutional layers, but instead of convolution they require vector-matrix multiplication. Input vectors contain a large number of values, as they are generated by flattening the previous layer's output, which leads to a lot of multiplication operations. To perform it, the inputs and weights are split into equal parts, and calculated separately. Finally, pooling layers are implemented using a sliding window and set to calculate the maximum of the selected value. Throughout the designs, 8-bit fixed point representation is used with three bits for the integer part and five bits for the fractional part.

IV. Experimental Results and Analysis

To verify, test and compare performance, the BranchyNet [6] early exit LeNet-5 network is used. It consists of three convolutional and two FC layers, while the decision sub-network contains one convolutional and one fully connected layer. Additionally, at all exit points, a softmax function is applied to classify a 28x28 image of a handwritten digit (MNIST data-set). It is executed on a desktop computer (Intel Core i9 + Nvidia RTX 2080 TI) and on an embedded platform (Nvidia Jetson Xavier NX).

Figure 2(a) shows the per sample execution time of the early-exit network on different platforms. We observe that the dynamic DNN is effective on all of them, as the average time is always lower than the no early exits (orange bars). Furthermore the *pipeline* approach is **1.5x** faster than on a desktop CPU and **1.8x** faster than the Jetson. The *parallel* approach, which executes the backbone and decision sub-network simultaneously, is **1.2x** faster than the *pipeline* approach in cases where an early exit is not decided.

The *pipeline* and the *parallel* approaches showcase comparatively low energy consumption (fig.2(b)), requiring 0.52 mJ

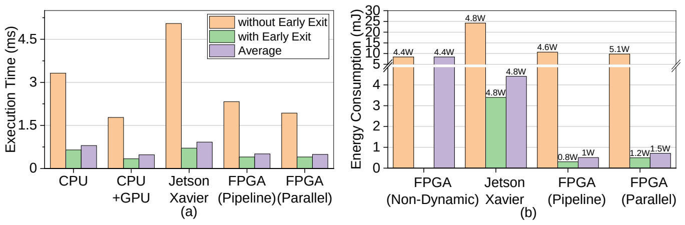

Fig. 2. Experimental results comparing (a) Execution time and (b) energy consumption per sample. Average values are calculated based on the frequency of the decision to early exit, which was calculated to be 94.37% yes and 5.63% no.

and 0.71 mJ persample respectively. This equates to **8.5x** and **6x** less energy than on the Jetson, and **16x** and **12x** than a non-dynamic FPGA LeNet-5 design. The effects of Dynamic DNN on energy consumption are also highlighted, as even on Jetson the average energy per inference was almost **2x** less than than the non-dynamic FPGA design.

Finally the *parallel* approach integrating the extra IPs for the decision sub-network had higher energy consumption, but avoided a 2.88kB data transfer to the memory and back. This has minimal effect on a network like LeNet, but on deeper networks, with more exit points and larger feature maps, can have a higher impact.

V. Conclusion

In this paper we explored dynamic DNNs on FPGAS, considering two decision sub-network architectures, and compared them to a desktop (CPU+GPU) system and an embedded device. All implementations achieved the same inference accuracy, with FPGA implementations achieving fastest execution and lowest energy consumption. Our future work, is to consider and evaluate networks containing more layers and exit points.

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/D2514.

References

[1] A. Krizhevsky, I. Sutskever, G.Hinton. "ImageNet classification with deep convolutional neural networks". in *Conf. NeurIPS*, 2012, pp.84–90.
[2] C. Shan, J. Zhang, Y. Wang, L. Xie. "Attention-based end-to-end speech recognition on voice search". in *Conf. ICASSP*, 2018.
[3] G. Huang, D. Chen, T. Li, F. Wu, L. van der Maaten, K. Q. Weinberger. "Multi-Scale Dense Convolutional Networks for Efficient Prediction". CoRR, 2017.
[4] Z. Chen, Y. Li, S. Bengio, S. Si. "GaterNet: Dynamic Filter Selection in Convolutional Neural Network via a Dedicated Global Gating Network." in *Conf CVPR*, 2019.
[5] L. Yang, Z. He, Y. Cao, D. Fan. "Non-uniform DNN Structured Subnets Sampling for Dynamic Inference.", in *Conf. DAC*, 2020, pp.1–6.
[6] S. Teerapittayanon, B. McDanel, H. T. Kung. "BranchyNet: Fast Inference via Early Exiting from Deep Neural Networks.", in *Conf. ICPR*, 2016.
[7] M. Wang, J. Mo, J. Lin, Z. Wang, L. Du. "DynExit: A Dynamic Early-Exit Strategy for Deep Residual Networks.", in *Conf. SiPS*, 2019, pp.178–183.
[8] Y. Han, G. Huang, S. Song, L. Yang, Honghui Wang, and Yulin Wang. "Dynamic neural networks: A survey.", CoRR, 2021.
[9] X. Dai, X. Kong, T. Guo. "EPNet: Learning to Exit with Flexible Multi-Branch Network.", in *Int. Conf. on Information & Knowledge Management*, 2020, pp235–244.