Hardware Deep Neural Network Acceleration by combining Early Exit Neural Network with Dynamic Partial Reconfigurable FPGA

Zeyu Yang
Electrical & Electronic Engineering
Imperial College London
London, UK
zeyu.yang18@imperial.ac.uk

I. Introduction

Deep learning has shown its capability in doing machine learning tasks, enabling advances in computer vision, natural language processing, autonomous driving, and robotics. However, such amazing capabilities come at the cost of computationally intensive algorithms, which will require high computational power and have high execution latency. This impedes its deployment in some of the latency-sensitive tasks, such as autonomous driving.

Research has been done on deep learning acceleration in recent years, from both algorithms and hardware perspectives. Research also focused on the inference stage, as the training stage is only a one-time job before the deployment. The energy for reading and writing to DRAM are orders of magnitudes higher than arithmetic operations, thus researchers have been trying to minimize memory footprints of the neural network. Researchers found that while it is important to train the neural network with high precision number representations, simplifying a trained neural network can still achieve high accuracy at inference. Methods include reducing the accuracy of number representation, from FP32 to INT8 [1] and even binary [2], Pruning, which eliminates neurons with small weights, and Weight Sharing, groups and quantize clustered weights [3]. Some methods from traditional fields like signal processing and communications, such as low-rank approximation, singular value decomposition [4], and Huffman coding [3] are applied to the weights, also reducing the size of memory.

The adaptive inference is another family of techniques that modifies the architecture/structure of the neural network according to different inputs, mainly based on the idea that different inputs have various sorts of different requirements. Model Selection is an approach that trains a family of models with different latency-accuracy goals, and the most appropriate model is selected based on the input at inference [5]. Dynamic Networks is a similar technique but dynamically selects the internals of a model, including branches [6], filters [7], or skipping layers [8]. Early-exiting is another idea that allows simper inputs to exit the network earlier to reduce the inference latency. The details, advantages, and drawbacks will be discussed in more detail later.

On the hardware side, GPUs have been the main facilitation for deep learning tasks thanks to their parallel floating-point arithmetic capabilities and high memory bandwidth. GPU vendors are also adding deep learning acceleration into their products in recent years, notably the Nvidia tensor cores. However, the high power consumption is one of the major drawbacks, causing difficulties at both ends: data centers and battery-powered embedded devices.

Researchers from academia and industry have developed several ASIC-based accelerators, such as DaDiannao [9], EIE [10], Eyeriss [11], Google Coral Edge TPU, and Intel Movidius. Compared to a GPU, they use a less accurate number representation, having additional on-chip memory and buffers for efficient data reuse, and special architectures such as systolic array and tiling to exploit parallelizability of neural networks. However, these accelerators lack the flexibility for handling different DL tasks and cannot be efficiently adapted to different applications. Additionally, they have all the drawbacks of being an ASIC: long development cycle and expensive production to name a few.

FPGA is the current research hotspot for deep learning acceleration. The DSP blocks allow efficient computation, while the on-chip memory is highly configurable for custom structures allowing data reuse. The overall reconfigurability can also adapt to different neural networks and tasks. However, these advantages come at the cost of the need to program in hardware description languages like Verilog. Recent advancement in High-Level Synthesis allows the FPGA to be programmed using C, significantly reducing the development time of FPGAs.

In addition to all the techniques used in those ASIC accelerators, researchers also focus on optimizations unique for FPGAs. To maximize resource utilization, work has been done on using polyhedral-base data dependence analysis to find the optimal unrolling factor for the convolutional layers [12]. Researchers also tried to group less computationally intensive layers in batches and process them together [13]. To maximize throughput, techniques using multiple convolutional layer processors instead of one were able to achieve layer-level parallelism [14]. Automation tools have also been developed for building networks on FPGA automatically, either using RTL modules [15] or HLS [16].

Modern FPGA has the feature of dynamic partial reconfiguration, where part of the FPGA can be reconfigured while other parts are still online doing calculations. However, not much research has been done on utilizing this characteristic on neural network acceleration where the FPGA can be reconfigured to match to different layers, improving the acceleration at the cost of reconfiguration delay. This will be discussed more in detail later.

The goal of this research is trying to accelerate the inference latency of a Deep Neural Network by combining the Early-Exit network structure with Partial Dynamic Reconfigurable FPGA.

The rest of this research proposal is organized as follows. In section II, I review the literature in Early-Exit Neural Network and Partial Dynamic reconfigurable FPGA. The research aims are presented in section III and methodologies for tackling the problem are discussed in section IV. Finally, we present the timescale and management/organization in section V.

II. Literature Review

A. Early-Exit Neural Network

First proposed in BranchyNet [17], Early-Exit networks are based on the idea that different inputs have a different level of difficulty to classify. While a difficult input needs to propagate through all the layers in a deep neural network, a simple input may already have a clear result after only a few layers. Early-Exits are additional exits of the network that can be placed in the middle throughout a neural network. When an input propagates to an Early-Exit, it can exit the network if a certain confidence level is met. The confidence is determined by an exit policy, which is usually another convolutional layer.

This technique reduces the inference latency by letting easier inputs propagate only a portion of the neural network, omitting unnecessary computations that do not contribute to accuracy. Say an Early-Exit is placed at the middle of a neural network, and half of the input is easy and able to exit early, then the average latency will be reduced by 25%.

The Early-Exit technique has been applied to all kinds of tasks including image classification [17], segmentation [18], enhancement [19], Speech Recognition [20], and Natural Language processing [21], all showing significant improvements in inference latency.

There are still several open questions, for example, what's the best way to train a network with multiple exits, and what's the best method for determining the exit policy. One particular question that is interesting to us is how many exits show we have.

Although it is awesome if an easy input exits the network early, for a difficult input that needs to continue propagating through, the confidence check becomes an additional latency. Lots of Early-Exits would minimize unnecessary computations for easy inputs but add latency onto the difficult inputs. It is challenging to balance the granularity of Early-Exits and their overheads.

B. Dynamic Partial Reconfigurable FPGA

Dynamic Partial Reconfiguration (DPR) allows runtime modification of an operating FPGA. Partial bit-streams can be loaded into the FPGA to reconfigure selected regions without affecting the functionality of other regions. Internal Configuration Access Port (ICAP) is offered by Xilinx FPGAs and Processor Configuration Access Port (PCAP) is also provided to reconfigure FPGA from the processor side in Zynq FPGAs. The reconfiguration takes time and is proportional to the size of the region to be reconfigured. This introduces a time overhead that needs to be considered compared to fully static design.

A number of research exploited this feature in designing deep learning accelerators. Meloni et al. dynamically changed the kernel size of the convolutional layers [22], Yang et al. implemented dynamic adaptive data truncation [23], Li et al. used DPR to switch between FP32 and INT8 calculations [24], and Seyoum et al. and Irmak et al. used DPR to schedule different accelerator cores [25] [26]. Kastner et al. packed each separate layer as a partial bit-stream file and dynamically load and process them [27]. Farhadi et al. took a step further and implemented the Model Selection technique (one of the adaptive inference techniques mentioned in the introduction) using DPR, where a shallow and a deep network can be dynamically switched based on the input [28].

Generally, in these implementations, the researchers were able to achieve better throughput or latency by dynamically changing to a more efficient architecture depending on the input, at the cost of some reconfiguration delays.

III. Research Aims

As one might have noticed, the overhead in the Early-Exit network and DPR shares some similarities: they are both additional sources of latency, and they are at the same location – between the layers. What if we overlap the two latencies together? For example, if we need to wait for the FPGA to reconfigure for the next layer, we could do the computations for confidence check of the Early exits. This would allow us to effectively add Early-exit at the end of every single layer at no extra cost, and maximize the Early-Exit technique, allowing the inputs to exit as soon as possible.

The aim of the research is to implement an FPGA-based deep learning accelerator using Dynamic Partial Reconfiguration and Early-Exit technique and exploit the reconfiguration delay to perform exit policy computation, to reduce the inference latency of a deep neural network.

IV. Methodology

As the research aim is trying to combine the advantages of two relatively new techniques, a thorough literature review and investigation for unanswered questions needs to be done of each of them.

For the Earl-Exit technique, we are interested in how many Early-Exits should we have. In theory, the performance vs the number of early-exits graph would be a parabola shape, where adding a few early-exits would help, but adding too many the overheads will be too significant. However, at the moment no research has been done on this question. Without this information, we can't really show the advantage of the final goal of adding lots of Early-exits with no additional latency. This could be a starting point of the research. We could take for example a VGG 16 and add from 1 to 15 early-exits to the network, and test its performance on the ImageNet, and recoding statistics like percentage of the inputs exited early and latency difference of the inputs that propagated the whole network. This may possibly lead to a publication. This investigation could be done in high-level tools such as TensorFlow. Possible challenges could be how to keep track of which Early-exit did the output exit.

For the DPR side, it is infeasible to develop an FPGA accelerator from scratch, as it will probably take up the whole Ph.D. Luckily most of the FPGA accelerator designs are open-sourced, allowing us to stand on the shoulder of the giants. Time would need to spend on selecting which implementation we should choose to build upon and realize it on our own hardware. Possible factors to consider are hardware requirements, scalability, flexibility, and how easy can it be modified for our purpose. The uncertainty of this task would be quite high, as currently, we don't know some of the exact details and compatibilities of others' work. If there are no suitable open-source implementations, we might need to find collaborations with other researchers. However, given the amount of FPGA accelerator research out there, I don't believe there will be no suitable implementations at all, and we are forced to develop from the ground up.

The next step is to bring the two together and implement the Early-exits on the FPGA. The exit policy is convolutional layers, so there already exist lots of efficient designs. The challenge will be finding the optimal design that uses minimal area on the FPGA, but still being able to finish the computation in the reconfiguration time. There is the possibility that there will be no more available FPGA resources for the early-exit implementation, as the FPGA accelerator design we picked could be able to fully utilize the FPGA already. In this case, we could either shrink the original design by for example reduce the loop unrolling factor, we could also try to offload the early-exit computation to the CPU or some other co-processors.

In terms of equipment, we will need an FPGA that supports DPR, such as the Xilinx's Zynq series, a reasonably powerful PC for programming the FPGA, and maybe a modern GPU to allow us to make performance comparisons between our FPGA accelerator and the GPU.

V. Timescale

The first task of investigating the number of Early-exits is planned to finish within 1-2 months. If we find it to be suitable for a publication, it may need up to an extra month to write and polish the results to a publication level.

The second task, as mentioned above, has quite a high uncertainty. The initial search will take a couple of weeks to have a more detailed idea of all the available implementations and pick the most suitable one. Fully understanding and realizing the one we picked will take another several weeks, and there is quite a high possibility that in the middle of the process we find it unsuitable or impossible to realize, we will need to go back to the previous step. This might even happen several times in the worst case. I suspect this task will take 3 months to account for some failures, and maybe an extra month or two if extremely unlucky.

I don't have a clear idea of how the last task will take, as this not only depends on how successful the progress is but also depends on the FPGA accelerator implementation selected in the previous step. There is the possibility that a lot of debugging and modifications need to be made before we could start trying to implement the Early-exit. As this is the core part of the research aim, I believe this will take at least 6 months.

Nonetheless, the first two tasks must be finished in the first year of the Ph.D. study.

References

[1] V. Vanhoucke, A. Senior, and M. Z. Mao, "Improving the speed of neural networks on cpus," 2011.

[2] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1," *arXiv preprint arXiv:1602.02830*, 2016.

[3] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," *arXiv preprint arXiv:1510.00149*, 2015.

[4] E. L. Denton, W. Zaremba, J. Bruna, Y. LeCun, and R. Fergus, "Exploiting linear structure within convolutional networks for efficient evaluation," in *Advances in neural information processing systems*, 2014, pp. 1269–1277.

[5] B. Taylor, V. S. Marco, W. Wolff, Y. Elkhatib, and Z. Wang, "Adaptive deep learning model selection on embedded systems," *ACM SIGPLAN Notices*, vol. 53, no. 6, pp. 31–43, 2018.

[6] R. T. Mullapudi, W. R. Mark, N. Shazeer, and K. Fatahalian, "Hydranets: Specialized dynamic architectures for efficient inference," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8080–8089.

[7] Z. Chen, Y. Li, S. Bengio, and S. Si, "You look twice: Gaternet for dynamic filter selection in cnns," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 9172–9180.

[8] Z. Wu, T. Nagarajan, A. Kumar, S. Rennie, L. S. Davis, K. Grauman, and R. Feris, "Blockdrop: Dynamic inference paths in residual networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8817–8826.

[9] Y. Chen, T. Luo, S. Liu, S. Zhang, L. He, J. Wang, L. Li, T. Chen, Z. Xu, N. Sun *et al.*, "Dadiannao: A machine-learning supercomputer," in *2014 47th Annual IEEE/ACM International Symposium on Microarchitecture*. IEEE, 2014, pp. 609–622.

[10] S. Han, X. Liu, H. Mao, J. Pu, A. Pedram, M. A. Horowitz, and W. J. Dally, "Eie: Efficient inference engine on compressed deep neural network," *ACM SIGARCH Computer Architecture News*, vol. 44, no. 3, pp. 243–254, 2016.

[11] Y.-H. Chen, T. Krishna, J. S. Emer, and V. Sze, "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," *IEEE Journal of Solid-State Circuits*, vol. 52, no. 1, pp. 127–138, 2017.

[12] S. Williams, A. Waterman, and D. Patterson, "Roofline: an insightful visual performance model for multicore architectures," *Communications of the ACM*, vol. 52, no. 4, pp. 65–76, 2009.

[13] C. Zhang, G. Sun, Z. Fang, P. Zhou, P. Pan, and J. Cong, "Caffeine: Toward uniformed representation and acceleration for deep convolutional neural networks," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 38, no. 11, pp. 2072–2085, 2018.

[14] S. I. Venieris and C.-S. Bouganis, "fpgaconvnet: A framework for mapping convolutional neural networks on fpgas," in *2016 IEEE 24th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*. IEEE, 2016, pp. 40–47.

[15] Y. Ma, N. Suda, Y. Cao, J.-s. Seo, and S. Vrudhula, "Scalable and modularized rtl compilation of convolutional neural networks onto fpga," in *2016 26th International Conference on Field Programmable Logic and Applications (FPL)*. IEEE, 2016, pp. 1–8.

[16] Y. Wang, J. Xu, Y. Han, H. Li, and X. Li, "Deepburning: Automatic generation of fpga-based learning accelerators for the neural network family," in *2016 53nd ACM/EDAC/IEEE Design Automation Conference (DAC)*. IEEE, 2016, pp. 1–6.

[17] 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)*. IEEE, 2016, pp. 2464–2469.

[18] X. Li, Z. Liu, P. Luo, C. Change Loy, and X. Tang, "Not all pixels are equal: Difficulty-aware semantic segmentation via deep layer cascade," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 3193–3202.

[19] Q. Xing, M. Xu, T. Li, and Z. Guan, "Early exit or not: Resource-efficient blind quality enhancement for compressed images," in *European Conference on Computer Vision*. Springer, 2020, pp. 275–292.

[20] S. Chen, Y. Wu, Z. Chen, T. Yoshioka, S. Liu, J. Li, and X. Yu, "Don't shoot butterfly with rifles: Multi-channel continuous speech separation with early exit transformer," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 6139–6143.

[21] J. Xin, R. Tang, J. Lee, Y. Yu, and J. Lin, "Deebert: Dynamic early exiting for accelerating bert inference," *arXiv preprint arXiv:2004.12993*, 2020.

[22] P. Meloni, G. Deriu, F. Conti, I. Loi, L. Raffo, and L. Benini, "A high-efficiency runtime reconfigurable ip for cnn acceleration on a mid-range all-programmable soc," in *2016 International Conference on ReConFigurable Computing and FPGAs (ReConFig)*. IEEE, 2016, pp. 1–8.

[23] C. Yang, Y. Wang, H. Zhang, X. Wang, and L. Geng, "A reconfigurable cnn accelerator using tile-by-tile computing and dynamic adaptive data truncation," in *2019 IEEE International Conference on Integrated Circuits, Technologies and Applications (ICTA)*. IEEE, 2019, pp. 73–74.

[24] Z. Li, L. Wang, S. Guo, Y. Deng, Q. Dou, H. Zhou, and W. Lu, "Laius: An 8-bit fixed-point cnn hardware inference engine," in *2017 IEEE International Symposium on Parallel and Distributed Processing with Applications and 2017 IEEE International Conference on Ubiquitous Computing and Communications (ISPA/IUCC)*. IEEE, 2017, pp. 143–150.

[25] B. B. Seyoum, M. Pagani, A. Biondi, S. Balleri, and G. Buttazzo, "Spatio-temporal optimization of deep neural networks for reconfigurable fpga socs," *IEEE Transactions on Computers*, 2020.

[26] H. Irmak, D. Ziener, and N. Alachiotis, "Increasing flexibility of fpga-based cnn accelerators with dynamic partial reconfiguration," in *2021 31st International Conference on Field-Programmable Logic and Applications (FPL)*. IEEE, 2021, pp. 306–311.

[27] F. Kästner, B. Janßen, F. Kautz, M. Hübner, and G. Corradi, "Hardware/software codesign for convolutional neural networks exploiting dynamic partial reconfiguration on pynq," in *2018 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*. IEEE, 2018, pp. 154–161.

[28] M. Farhadi, M. Ghasemi, and Y. Yang, "A novel design of adaptive and hierarchical convolutional neural networks using partial reconfiguration on fpga," in *2019 IEEE High Performance Extreme Computing Conference (HPEC)*. IEEE, 2019, pp. 1–7.