Cascade$^{\text{CNN}}$: Pushing the performance limits of quantisation

Alexandros Kouris
Dept. of Electrical and Electronic Eng.
Imperial College London
a.kouris16@ic.ac.uk

Stylianos I. Venieris
Dept. of Electrical and Electronic Eng.
Imperial College London
stylianos.venieris10@ic.ac.uk

Christos-Savvas Bouganis
Dept. of Electrical and Electronic Eng.
Imperial College London
christos-savvas.bouganis@ic.ac.uk

ABSTRACT

This work presents *CascadeCNN*, an automated toolflow that pushes the quantisation limits of any given CNN model, to perform high-throughput inference by exploiting the computation time-accuracy trade-off. Without the need for retraining, a two-stage architecture tailored for any given FPGA device is generated, consisting of a low- and a high-precision unit. A confidence evaluation unit is employed between them to identify misclassified cases at run time and forward them to the high-precision unit or terminate computation. Experiments demonstrate that *CascadeCNN* achieves a performance boost of up to 55% for VGG-16 and 48% for AlexNet over the baseline design for the same resource budget and accuracy.

1 INTRODUCTION

While Convolutional Neural Networks are becoming the state-of-the-art algorithm in various Machine Vision tasks [1][2][3], they are challenged to deal with problems of continuously increasing complexity. The significant advances of CNNs came with increased number of layers [4], increased number of kernels [5] and more complex architectures [6][7], which introduce substantial costs in terms of computational and memory resources. To deploy CNNs in real-world tasks which deal with vast amounts of data, it is necessary that the high computation and memory requirements of such models are alleviated. To this end, numerous compression and precision quantisation techniques [8][9][10][11] have been proposed which exploit the redundancy in CNN models to enable the efficient deployment of CNNs on processing platforms.

In this context, FPGAs constitute a promising platform for CNN inference due to their customisability which enables the use of optimised low-precision arithmetic units to achieve high performance at a low power envelope [12]. Existing FPGA-based CNN accelerators have produced hardware designs that span from uniform 16-bit activations and weights [13][14] with minimal effect on accuracy, down to very high-performance binarised networks [15] but with a significant accuracy loss. In this setting, given a fixed resource budget, the attainable performance for a given error tolerance is limited by the shortest wordlength that meets the error bound.

In this paper, we propose *CascadeCNN*, a novel automated approach of pushing the performance of precision-quantised CNN models under the same resource budget, with negligible accuracy loss. *CascadeCNN* employs a low-precision processing unit to obtain rapid classification predictions together with a parametrised mechanism for identifying misclassified cases based on prediction confidence. Such detected cases are recomputed on a high-precision unit to restore application-level accuracy and meet user-specified limits. *CascadeCNN* considers the error tolerance and the target CNN-device pair to select quantisation scheme, configure the confidence evaluation mechanism and generate the cascaded low- and high-precision processing units.

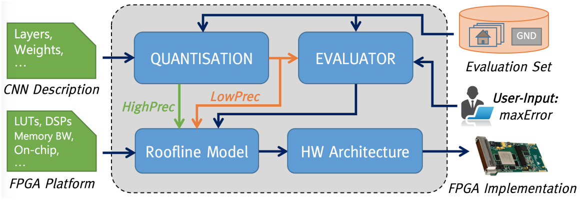

Figure 1: High-level *CascadeCNN* toolflow

2 CASCADE CNN

2.1 Overview

Fig. 1 shows the processing flow of *CascadeCNN*. The framework is supplied with a high-level description of a trained CNN model (i.e. Caffe model), the available computational and memory resources of the target platform and an application-level error tolerance in a user-defined metric (e.g. top-1/top-5 classification error), along with a small evaluation set. *CascadeCNN* searches the architectural design space and generates a two-stage hardware architecture, optimised for the particular CNN model and target device. The generated system (Fig. 2) consists of:

- A low-precision unit (LPU) which employs low-precision arithmetic to trade lower accuracy with high-throughput CNN inference.
- A high-precision unit (HPU) which guarantees the same accuracy level as the reference model.
- A tunable Confidence Evaluation Unit (CEU) that detects samples that were wrongly classified by the LPU and redirects them to HPU for re-processing.

The key idea behind the proposed approach is that during the execution of the system, the LPU will process the whole workload, while the HPU will only process a fraction of it, based on the CEU's evaluation of classification confidence on LPU's predictions, reducing its memory and compute requirements. Moreover, the accuracy loss that is induced due to the extreme model quantisation of the LPU is restored to meet the user-specified error threshold.

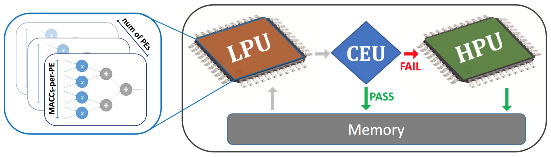

Figure 2: *CascadeCNN* architecture

2.2 Quantisation

Arithmetic precision reduction is a widely studied technique which exploits the inherent redundancy of CNNs to considerably reduce the memory bandwidth and footprint requirements, minimise power consumption and achieve higher performance. *CascadeCNN* employs a fine-grained search space across possible precision quantisation schemes, that allows determining the number of integer and fractional bits of weight and activation values by introducing a different scaling factor for each layer. In this dynamic fixed-point approach, the wordlength is kept uniform across layers with a different scaling factor for each layer. For each explored wordlength, statistics regarding the quantisation effect of each layer on the application-level accuracy are extracted using the user-provided evaluation set. The per-layer statistics are used to guide the exploration to the combination of scaling factors that achieve the highest accuracy for each explored wordlength. In contrast to other frameworks, *CascadeCNN* selects for the LPU a precision that achieves intermediate application-level accuracy, but with significantly higher performance when mapped on its custom precision-optimised hardware units. All input samples are processed by the LPU to obtain a rapid classification decision, which is then fed to the Confidence Evaluation Unit. A wordlength that achieves an accuracy that complies with the user-specified error margins is selected for the HPU.

Since the reduced-precision model employed by the LPU is derived by straight quantisation (without retraining), its parameters are extracted at run time in hardware from the HPU's higher precision model. As a result of this weight-sharing approach, the memory footprint of the proposed cascade system remains the same as in the case of a single-stage architecture employing the HPU's model.

2.3 Confidence Evaluation

The *CascadeCNN* tool allows the exploration of extreme quantisation schemes for the LPU, by aiming to identify potentially misclassified inputs based on the confidence of the LPU classification prediction. To estimate this confidence, we build on the work of [16] by generalising the proposed Best-vs-Second-Best (BvSB) metric, which was previously examining solely binary classification problems. Our generalised BvSB (gBvSB) metric is described as:

$$\text{gBvSB}_{<M,N>}(\boldsymbol{p}) = \sum_{i=1}^{M} p_i - \sum_{j=M+1}^{N} p_j \qquad (1)$$

where p_i denotes the i-th element of the sorted probability vector $\boldsymbol{p}$ of the prediction and M and N are tunable parameters of gBvSB. In this context, a prediction is considered confident, and thus the processing ends on the low-precision unit, when $\text{gBvSB}_{<M,N>}(\boldsymbol{p}) \geq th$ where M, N and threshold th form tunable parameters whose values are automatically determined using the evaluation set data and the user-specified error tolerance. In this manner, the degree of uncertainty on the classification decision is based on how spiky the sorted probability distribution of the CNN's prediction is.

2.4 Architecture

A scalable, fine-grained hardware architecture is designed that is able to execute CNN inference, scale its performance with the resources of a target FPGA and exploit higher degrees of parallelism as the wordlength of activation and weight representation decreases. The core of the architecture is a matrix multiplication (*MM*) unit, parametrised with respect to the tiling of each matrix dimension and the arithmetic precision of both activations and weights. The *MM* unit comprises Multiply-Accumulate (MACC) units, grouped into Processing Elements (PEs) that perform dot-product operations (shown in Fig. 2). By casting convolution operations as matrix multiplications and using batch processing for fully-connected (FC) layers, both CONV and FC layers are mapped on the *MM* unit.

Given a CNN-FPGA pair and a particular wordlength, *CascadeCNN* searches the architectural design space by means of a roofline-based performance model [17] in order to determine the highest performing configuration of the architecture. The configurable parameters comprise the matrix tile sizes, that correspond to different levels of parallelism in terms of number of PEs and MACCs-per-PE. In this manner, *CascadeCNN* generates two architectures, the LPU and the HPU, which are optimised for different wordlengths.

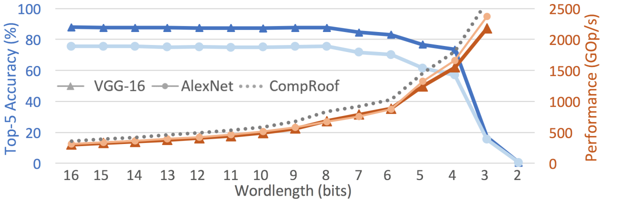

Figure 3: Top-5 classification accuracy on ImageNet and performance as a function of wordlength on Zynq ZC706.

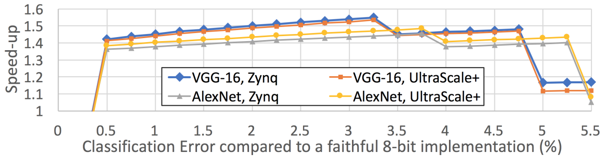

Figure 4: *CascadeCNN* speed-up

3 EVALUATION

To evaluate the proposed toolflow, we target image classification using pretrained models on the ImageNet [18] dataset. *CascadeCNN* is provided with models of VGG-16 [4] and AlexNet [1], along with a small subset of the ImageNet validation set as an evaluation set (200 labelled samples), targeting two different FPGA platforms, Xilinx Zynq ZC706 and UltraScale+ ZCU102.

For both VGG-16 and AlexNet, *CascadeCNN* yields a wordlength of 4 bits for the LPU. The selected 4-bit quantisation scheme introduces a 14.38% and 18.65% degradation in classification accuracy compared to an 8-bit precision respectively (Fig. 3). The CEU parameters are tuned on the evaluation dataset to generate systems that introduce a wide range of classification errors, compared to a faithful 8-bit implementation. To evaluate the performance gains of *CascadeCNN*, we compare the generated two-stage system for each error tolerance with a baseline single-stage architecture that is optimised with a quantisation scheme that achieves the same or better accuracy (ranging from 5 to 7 bit wordlengths). The achieved speed-up on throughput is illustrated in Fig. 4 across a wide range of error thresholds. In the case of high error tolerance, the speed-up becomes less significant as the difference in wordlength between the LPU and the baseline design decreases. On both target platforms the performance has been improved by up to 55% for VGG-16 and up to 48% for AlexNet over the baseline design for the same resource budget and error tolerance. The proposed methodology can also be applied to other existing CNN accelerator architectures, with variable performance gains.

4 CONCLUSION

This work presents *CascadeCNN*, an automated toolflow for CNN inference acceleration exploiting the computation time-accuracy trade-off. The cascaded two-stage architecture generated by the toolflow demonstrates a performance boost of up to 55% for VGG-16 and 48% for AlexNet compared to a single-stage baseline architecture for the same resource budget and error tolerance.

ACKNOWLEDGMENT

The support of the EPSRC Centre for Doctoral Training in High Performance Embedded and Distributed Systems (HiPEDS, Grant Reference EP/L016796/1) is gratefully acknowledged. This work is also supported by EPSRC grant 1507723.

REFERENCES

[1] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[2] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You Only Look Once: Unified, Real-Time Object Detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 779–788, 2016.

[3] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(12):2481–2495, 2017.

[4] Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. *International Conference on Learning Representations*, 2015.

[5] Matthew D Zeiler and Rob Fergus. Visualizing and Understanding Convolutional Networks. In *European Conference on Computer Vision*, pages 818–833. Springer, 2014.

[6] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going Deeper with Convolutions. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9, 2015.

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[8] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Quantized Neural Networks: Training Neural Networks with Low Precision Weights and Activations. *arXiv preprint arXiv:1609.07061*, 2016.

[9] Song Han, Huizi Mao, and William J Dally. Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding. *International Conference on Learning Representations (ICLR)*, 2016.

[10] Darryl Lin, Sachin Talathi, and Sreekanth Annapureddy. Fixed Point Quantization of Deep Convolutional Networks. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, pages 2849–2858, 2016.

[11] Jiaxiang Wu, Cong Leng, Yuhang Wang, Qinghao Hu, and Jian Cheng. Quantized Convolutional Neural Networks for Mobile Devices. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4820–4828, 2016.

[12] Stylianos I. Venieris and Christos-Savvas Bouganis. fpgaConvNet: A Toolflow for Mapping Diverse Convolutional Neural Networks on Embedded FPGAs. In *Workshop on Machine Learning on the Phone and other Consumer Devices (MLPCD), NIPS*, 2017.

[13] Stylianos I. Venieris and Christos-Savvas 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)*, pages 40–47, 2016.

[14] Yufei Ma, Yu Cao, and Jae sun Seo. An Automatic RTL Compiler for High-Throughput FPGA Implementation of Diverse Convolutional Neural Networks. In *2017 27th International Conference on Field Programmable Logic and Applications (FPL)*, 2017.

[15] Yaman Umuroglu, Nicholas J. Fraser, Giulio Gambardella, Michaela Blott, Philip Leong, Magnus Jahre, and Kees Vissers. FINN: A Framework for Fast, Scalable Binarized Neural Network Inference. In *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, FPGA '17, 2017.

[16] Ajay J Joshi, Fatih Porikli, and Nikolaos Papanikolopoulos. Multi-Class Active Learning for Image Classification. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2372–2379, 2009.

[17] Samuel Williams, Andrew Waterman, and David Patterson. Roofline: an insightful visual performance model for multicore architectures. *Communications of the ACM*, 52(4):65–76, 2009.

[18] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.