Deploying Deep Neural Networks in the Embedded Space

Stylianos I. Venieris, Alexandros Kouris and Christos-Savvas Bouganis
Dept. of Electrical and Electronic Enigineering, Imperial College London
{stylianos.venieris10,a.kouris16,christos-savvas.bouganis}@imperial.ac.uk

ABSTRACT

Recently, Deep Neural Networks (DNNs) have emerged as the dominant model across various AI applications. In the era of IoT and mobile systems, the efficient deployment of DNNs on embedded platforms is vital to enable the development of intelligent applications. This paper summarises our recent work on the optimised mapping of DNNs on embedded settings. By covering such diverse topics as DNN-to-accelerator toolflows, high-throughput cascaded classifiers and domain-specific model design, the presented set of works aim to enable the deployment of sophisticated deep learning models on cutting-edge mobile and embedded systems.

1 INTRODUCTION

The effective mapping of DNN inference on embedded platforms can offer multiple benefits for mobile AI applications. These include: 1) enabling the processing of high resolution inputs without compromising the user experience due to high-latency access of cloud services, 2) enabling the processing of multiple data sources in high-throughput applications, 3) reducing response time and 4) complying with the power constraints of embedded platforms.

In this context, custom hardware accelerators offer unique opportunities for tailor-made solutions that meet the system-level constraints while providing higher-performance execution and lower power consumption than conventional programmable architectures. However, the inherent complexity of mapping applications on such specialised platforms hinders their adoption from deep learning practitioners. In our recent work, we have tackled a number of critical problems to enhance the accessibility of such platforms. The scope of our work ranges from software infrastructure for the automated generation of DNN accelerators to DNN model design with application-specific optimisations. The key categories involve: 1) CNN-to-accelerator toolflows for the automated generation of CNN accelerators, targeting both high-throughput and low-latency settings [5–7]; 2) the exploitation of the resilience of CNNs to low-precision arithmetic to achieve substantial performance gains [1, 2]; 3) automated synthesis of optimised architectures for emerging multi-CNN applications that employ multiple models for different tasks [9]; 4) LSTM accelerators that enable deployment in latency-critical applications with limited computation time budget [4]; and 5) design of domain-specific DNN models tailored to both the target system's accuracy requirements and compute capabilities [3]. The rest of the paper presents a high-level view of our work.

2 CNN-TO-ACCELERATOR AUTOMATION

The success of CNNs has come with an increase in compute and memory requirements. In this context, FPGAs stand as a promising platform that can meet both the compute requirements and the power constraints of emerging CNN applications. Currently, several obstacles stand as a barrier between deep learning practitioners and FPGAs. From a development perspective, FPGA system development requires expertise in hardware design and familiarity with FPGA toolchains, two skills that typically do not fall within the skillset of deep learning scientists. Moreover, due to the design flexibility of FPGAs, the possible mappings of a CNN on an FPGA lie on a high-dimensional design space that cannot be explored manually. From an application perspective, the diversity of CNN application domains results in a wide spectrum of performance needs. Spanning from the high-throughput needs of multi-sensor systems to latency-critical self-driving cars, the underlying hardware has to be optimised for the particular performance metric of interest. In this context, there is a need for frameworks that abstract the low-level details of FPGAs and automate the generation of FPGA-based CNN accelerators, optimised for the needs of the target application [10].

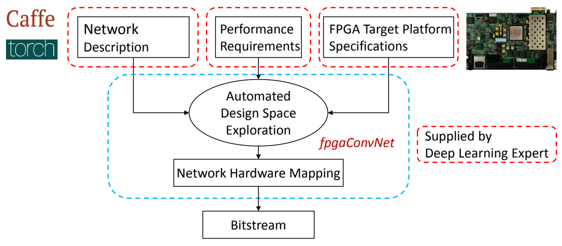
Caffe
torch
Network Description
Performance Requirements
FPGA Target Platform Specifications
Automated Design Space Exploration
fpgaConvNet
Supplied by Deep Learning Expert
Network Hardware Mapping
Bitstream

Figure 1: Overview of fpgaConvNet's flow.

2.1 The fpgaConvNet Toolflow

fpgaConvNet is a toolflow whose goal is to automate the mapping of CNNs on FPGAs [5–7]. Starting from a high-level description of a CNN model, fpgaConvNet (Fig. 1) considers both the supplied model's workload and the application-level performance needs, including the required throughput and latency, and generates an optimised accelerator for the target FPGA device. At the hardware level, fpgaConvNet employs a highly customisable streaming architectural template which exploits the parallelism both within and across layers and supports large networks by posing no constraints on the model size. To tailor the generated hardware to the CNN-FPGA pair, fpgaConvNet employs an analytical Synchronous Dataflow model for capturing both CNN workloads and hardware mappings. This formulation enables the fast exploration of the design space by means of a set of algebraic operations that correspond to a wide range of optimisations and modify the performance-resource cost space of the implementation. To yield the final hardware design, fpgaConvNet casts design space exploration as a mathematical optimisation problem with an objective function that captures the throughput and latency requirements of the target application and automatically generates the resulting hardware design by means of code generation. Overall, in low-power embedded and mobile settings, fgpaConvNet's designs demonstrate performance gains of up to 6.65× over highly optimised embedded GPU implementations when operating under the same power budget.

2nd International Workshop on Embedded and Mobile Deep Learning, Munich, Germany, June 2018.

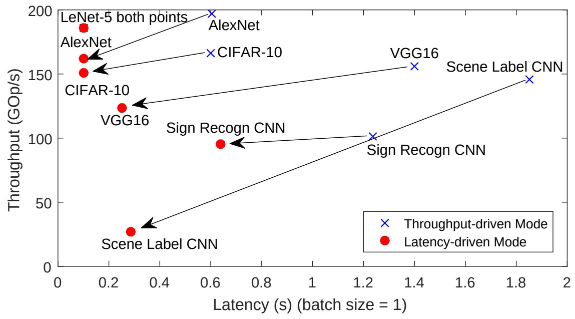

Figure 2: Throughput-driven vs. latency-driven mode.

2.2 A Latency-Driven Methodology

The majority of existing CNN implementations on CPUs, GPUs and FPGAs are optimised for high throughput. Emerging new AI systems, from autonomous drones and cars to low response-time mobile applications, require the very low-latency execution of CNN inference without the overhead of batch processing. To enable this type of applications, fpgaConvNet can place latency at the centre of optimisation and generate latency-optimised hardware designs for the target CNN-FPGA pair [8]. The latency-driven methodology comprises a run-time configurable architecture that enables the high-performance execution of CNNs without the latency penalty imposed by batching, together with a latency-centric optimiser that guides the design space exploration to previously unreachable low-latency regions (Fig. 2). fpgaConvNet's latency-driven flow delivers up to 73.45× and 5.61× improvements in latency over throughput-optimised designs of AlexNet and VGG16 respectively.

3 PRECISION

A common strategy to alleviate the large compute and memory requirements of state-of-the-art DNNs is the use of reduced precision. The majority of approaches assume the availability of the training set and employ a retraining step to restore the quantised model's accuracy. However, in privacy-aware scenarios, such data are not available and hence methods for reducing the precision without retraining are required. Moreover, in applications with low error tolerance, the accuracy loss due to quantisation often prohibits the use of low-precision arithmetic. In this context, there is a need for efficient methods of executing DNNs that combine the gains of reduced precision with negligible accuracy drop.

3.1 CascadeCNN's Approach

CacsadeCNN [1, 2] introduces an automated toolflow for generating a high-throughput cascade of CNN classifiers that pushes the performance of precision-quantised CNNs. Our key observation is that not all inputs of a CNN require the same level of precision in the computation to yield a confident prediction. In this respect, *CascadeCNN* exploits this property and generates a two-stage architecture of precision-quantised models (Fig. 3). The first stage consists of an excessively low-precision processing unit that enables rapid classification prediction. The outputs from the first stage are fed to a confidence evaluation unit which estimates the prediction confidence. The samples that are detected as misclassified are recomputed on a high-precision unit to restore the application-level accuracy and comply with the user-specified error tolerance. Overall, *CascadeCNN* considers the error tolerance and the input CNN-FPGA pair to select quantisation scheme, configure the confidence evaluation mechanism and generate the cascaded low- and high-precision processing units. The *CascadeCNN*'s designs demonstrate a performance boost of up to 55% for VGG16 and 48% for AlexNet over baseline designs achieving the same accuracy.

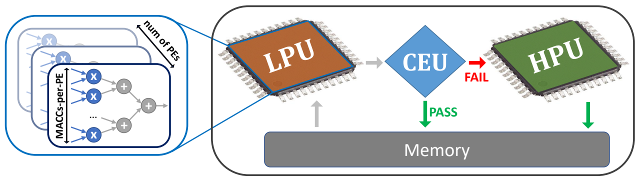

Figure 3: *CascadeCNN*'s high-level architecture.

4 MULTI-DNN SYSTEMS

In the construction of complex AI systems, DNN models are used as building blocks of a larger system. In this respect, multi-DNN systems have emerged, employing several models, each one trained for a different subtask. In particular, in the emerging field of intelligent autonomous systems, such as drones and self-driving cars, the system's perception is largely based on computer vision tasks, such as object detection and semantic segmentation. Such systems require the concurrent execution of these subtasks and hence the parallel and continuous execution of multiple models.

Nevertheless, deploying multiple models on a target platform poses a number of challenges. From a resource allocation perspective, with each model targeting a different task, the performance constraints, such as required throughput and latency, vary accordingly. Instead of being model-agnostic, this property requires the design of an architecture that captures and reflects the performance requirements of each model. Moreover, in resource-constrained setups, multiple DNNs compete for the same pool of resources and hence resource allocation between models becomes a critical factor. In this respect, the mapping of multiple DNNs is a multidimensional design problem that encompasses both the performance needs of each model and the resource constraints of the target platform.

4.1 f-CNN^x: Deploying Multiple CNNs

f-CNN^x [9] is a toolflow which addresses the challenge of mapping multiple CNNs on a target FPGA platform while meeting the required performance for each model. f-CNN^x exploits the structure of CNN workloads and the fine-grained control over resource allocation of FPGAs to yield latency-optimised designs. From a hardware perspective, the developed toolflow introduces a highly parametrised multi-CNN architecture (Fig. 4) that allows the fine-grained allocation of resources among CNNs and the deterministic scheduling of competing memory transfers. f-CNN^x explores a wide range of resource and bandwidth allocations and incorporates the application-level importance of each model by means of multiobjective cost functions to guide the design space exploration to the optimum hardware design. Overall, f-CNN^x overcomes the limitations of other parallel platforms by yielding up to 6.8× gains in performance-per-Watt over highly optimised embedded GPU designs in multi-CNN settings. To the best of our knowledge, this work addresses for the first time in the literature the latency-optimised mapping of multiple CNNs.

5 COMPUTING UNDER TIME CONSTRAINTS

Modern intelligent systems such as mobile robots and UAVs that employ DNNs to perceive and interact with their surroundings often operate under time-constrained, latency-critical settings. In such scenarios, the output of a DNN would typically yield an action,

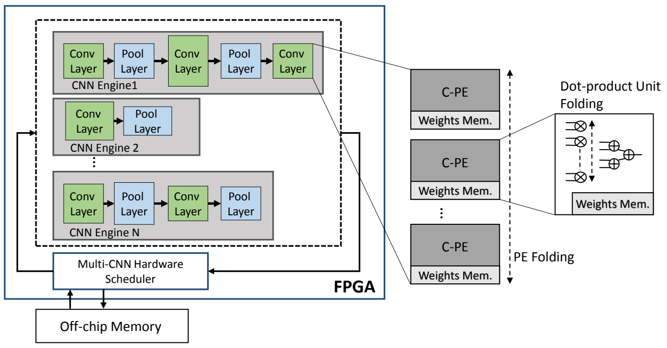

Figure 4: Parallel architecture for multiple CNNs.

such as a manoeuvre to avoid an obstacle. For such decision-making to happen both in real-time and with the best possible outcome, obtaining the most informative output from a DNN given a constraint in computation time is vital. In this respect, the design of computing systems that exploit the runtime-accuracy trade-off of DNNs is necessary to enable the timely operation of such systems.

5.1 Approximate FPGA-based LSTMs

With a focus on the high-performance deployment of LSTMs under time-constrained settings, [4] presents a framework that comprises an approximate computing scheme together with a novel FPGA-based hardware architecture for LSTMs. The proposed framework (Fig. 5) employs an iterative approximation method to compress and prune the target LSTM and explore the computation time-accuracy trade-off. Internally, the framework co-optimises the LSTM approximation and the hardware design in order to meet the computation time constraints. By targeting a real-life image captioning application, the designs generated by the developed framework demonstrate 6.5× less time to achieve the same application-level accuracy over a baseline accelerator, while reaching an average of 25× higher accuracy under the same computation time constraints.

6 DOMAIN-SPECIFIC DNN DESIGN

Conventionally, the deep learning community's design methodology for DNN models focuses on maximising the accuracy on the target task, while largely neglecting the implications on the inference-time computational cost. By considering domain-specific properties in order to guide the design of DNNs, more efficient models can be constructed which both meet the required accuracy and lie within the compute capabilities of the target platform.

6.1 The DroNet Vehicle Detector

Drones are emerging as a promising technology for a broad range of applications from domains such as agriculture, security and infrastructure monitoring. A prominent application includes the detection of vehicles for emergency response and traffic monitoring. In this scenario, drones operate autonomously and employ CNN models for the detection of vehicles. [3] presents an end-to-end investigation of different single-shot CNN detectors for drone-based vehicle detection. Starting from the dataset collection and model design down to the deployment on the Odroid-XU4 and Rasperry Pi 3 embedded hardware platforms, this work presents an exploration over the structure of the CNN. To find the CNN that optimises both accuracy and computation cost, a custom metric is employed that, given a model instance, captures both the detection accuracy and the achieved runtime on the target hardware platform. By following this methodology, the resulting detection CNN, named

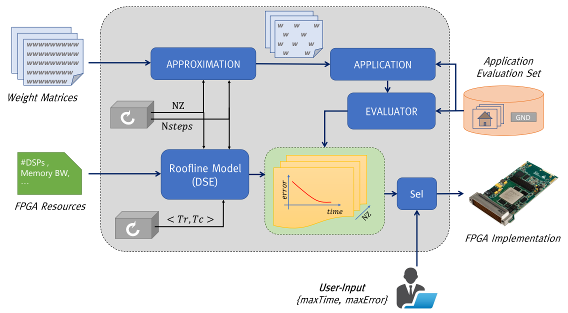

Figure 5: Overview of approximate LSTMs design flow.

DroNet, yields the highest performing balance between detection accuracy and fast execution on the two target embedded platforms.

7 CONCLUSION

The presented set of works focuses on bridging the gap between the deep learning community and the deployment of models in the embedded space. The fpgaConvNet toolflow automates the optimised generation of FPGA-based CNN accelerators targeting both high-throughput and latency-driven applications. In the context of quantised CNNs, *CascadeCNN* alleviates the need for training data availability and enables in this way the use of high-throughput CNN accelerators that employ extremely low-precision arithmetic in privacy-critical applications. By targeting multi-CNN systems, f-CNN^x paves the way in executing multiple CNNs under latency constraints on FPGAs. Moreover, an approximate computing methodology for the deployment of LSTMs in time-constrained applications is presented, enabling latency-critical systems to make informed decisions in real-time. Finally, by considering domain-specific optimisations, a model design methodology has been developed to construct DNNs that are optimised for both the task-level accuracy and compute capabilities of the target platform.

REFERENCES

[1] Alexandros Kouris, Stylianos I. Venieris, and Christos-Savvas Bouganis. 2018. CascadeCNN: Pushing the performance limits of quantisation. In *SysML*.

[2] Alexandros Kouris, Stylianos I. Venieris, and Christos-Savvas Bouganis. 2018. CascadeCNN: Pushing the Performance Limits of Quantisation in Convolutional Neural Networks. In *2018 28th International Conference on Field Programmable Logic and Applications (FPL)*. 1–8.

[3] C. Kyrkou, G. Plastiras, T. Theocharides, S. I. Venieris, and C. S. Bouganis. 2018. DroNet: Efficient Convolutional Neural Network Detector for Real-Time UAV Applications. In *2018 Design, Automation Test in Europe Conference Exhibition (DATE)*. 967–972. https://doi.org/10.23919/DATE.2018.8342149

[4] Michalis Rizakis, Stylianos I. Venieris, Alexandros Kouris, and Christos-Savvas Bouganis. 2018. Approximate FPGA-based LSTMs under Computation Time Constraints. In *Applied Reconfigurable Computing - 14th International Symposium, ARC 2018, Santorini, Greece, May 2 - 4, 2018, Proceedings*. 3–15.

[5] Stylianos I. Venieris and Christos-Savvas Bouganis. 2016. fpgaConvNet: A Framework for Mapping Convolutional Neural Networks on FPGAs. In *2016 IEEE 24th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*. Institute of Electrical and Electronics Engineers (IEEE), 40–47.

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

[7] Stylianos I. Venieris and Christos-Savvas Bouganis. 2017. fpgaConvNet: Automated Mapping of Convolutional Neural Networks on FPGAs (Abstract Only). In *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*. ACM, 291–292.

[8] S. I. Venieris and C. S. Bouganis. 2017. Latency-Driven Design for FPGA-based Convolutional Neural Networks. In *2017 27th International Conference on Field Programmable Logic and Applications (FPL)*. 1–8.

[9] S. I. Venieris and C. S. Bouganis. 2018. f-CNN^x: A Toolflow for Mapping Multiple Convolutional Neural Networks on FPGAs. In *2018 28th International Conference on Field Programmable Logic and Applications (FPL)*. 1–8.

[10] Stylianos I. Venieris, Alexandros Kouris, and Christos-Savvas Bouganis. 2018. Toolflows for Mapping Convolutional Neural Networks on FPGAs: A Survey and Future Directions. *ACM Comput. Surv.* (2018).