A Throughput-Latency Co-Optimised Cascade of Convolutional Neural Network Classifiers

Alexandros Kouris
Electrical & Electronic Engineering Dept.
Imperial College London, UK
a.kouris16@imperial.ac.uk

Stylianos I. Venieris
Samsung AI Center
Cambridge, UK
s.venieris@samsung.com

Christos-Savvas Bouganis
Electrical & Electronic Engineering Dept.
Imperial College London, UK
christos-savvas.bouganis@imperial.ac.uk

***Abstract*—Convolutional Neural Networks constitute a prominent AI model for classification tasks, serving a broad span of diverse application domains. To enable their efficient deployment in real-world tasks, the inherent redundancy of CNNs is frequently exploited to eliminate unnecessary computational costs. Driven by the fact that not all inputs require the same amount of computation to drive a confident prediction, multi-precision cascade classifiers have been recently introduced. FPGAs comprise a promising platform for the deployment of such input-dependent computation models, due to their enhanced customisation capabilities. Current literature, however, is limited to throughput-optimised cascade implementations, employing large batching at the expense of a substantial latency aggravation prohibiting their deployment on real-time scenarios. In this work, we introduce a novel methodology for throughput-latency co-optimised cascaded CNN classification, deployed on a custom FPGA architecture tailored to the target application and deployment platform, with respect to a set of user-specified requirements on accuracy and performance. Our experiments indicate that the proposed approach achieves comparable throughput gains with related state-of-the-art works, under substantially reduced overhead in latency, enabling its deployment on latency-sensitive applications.**

I. Introduction

In recent years, the rapid evolution of Convolutional Neural Networks (CNNs) brought their adoption on a wide spectrum of machine vision applications. Nevertheless, the unprecedented ability to learn robust representations and the high predictive accuracy of CNNs is also marked by heavy computational workloads and memory requirements. These challenges are further aggravated in scenarios that require the embedded deployment of Artificial Intelligence algorithms. In the case of autonomous systems, for example, computationally expensive models are lined up with imperative performance requirements imposed within the resource-constrained embedded landscape.

From a software perspective, to address these challenges recent research has focused on exploiting the inherent redundancy evident in most state-of-the-art deep learning (DL) models. The aim of these efforts is to alleviate the computational cost with no or minimum compromise in the model's predictive accuracy. Representative examples of such works include model quantisation [1] [2], parameter pruning [3], compression [4] [5] and low-rank factorisation [6]. In these cases, the level of approximation is tightly bounded by the model's accuracy resilience, as captured on the target dataset.

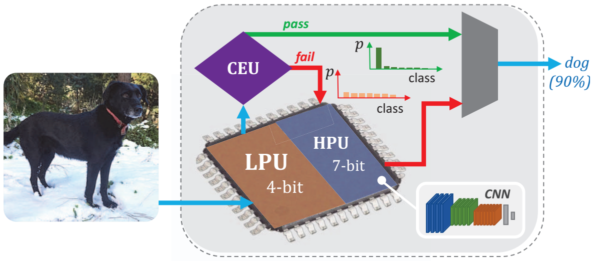

Fig. 1: Multi-precision Cascade for image classification, consisting of a Low-Precision and a High-Precision Unit mapped concurrently on the same FPGA device. A Confidence Evaluation Unit acts on the LPU outputs, determining which samples need to be forwarded to the HPU for re-processing.

Driven by the fact that not all inputs require the same amount of computation to drive a confident prediction, a recent stream of works, such as cascade [7] and early-stop [8] classifiers, further utilise the above techniques in an *input-dependent computation* setting. This way, a greater performance boost can be achieved on more resilient samples by pushing the approximation limits, whereas the overall accuracy is not degraded by using a more faithful variant of the original model for handling more challenging inputs (Fig. 1).

Capitalising on such optimisations in order to maximise the attainable performance and efficiency, the employed computational platforms have diverted from conventional general-purpose processors, towards many-core accelerators (e.g. GPUs) and custom hardware solutions tailored to the needs of the targeted application (e.g. ASICs and FPGAs).

Field-Programmable Gate Arrays (FPGAs) form a cardinal platform for the implementation of approximate inference approaches [9] due to their enhanced flexibility and customisation capabilities, also empowered by the small development cycles enabled by the emerging High-Level Synthesis (HLS) tools and automated toolflows developed by the community [10]. In recent literature, the reconfigurability of FPGAs has enabled the expansion to a new design dimension, allowing to switch between highly-optimised tailored hardware architectures at run time [11]. Such works, being commonly optimised for high-throughput inference, alleviate the prohibitive device reconfiguration time by the employment of large batch sizes.

However, adding to the reconfiguration time overhead, batching also introduces a substantial latency penalty making

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.

these systems unsuitable for latency-sensitive applications, involving real-time mission critical decision making, such as robot navigation, autonomous driving, etc. To address this issue, in this work, we introduce a methodology for efficient FPGA deployment of high-throughput cascades of CNN classifiers, preserving the prediction latency below a user-specified requirement. The main contributions of our work are:

- The development of a configurable multi-precision (2-stage) cascade hardware architecture for FPGAs, exploiting the performance-accuracy trade-off to accelerate inference in the embedded space of real-time applications, without the need for *batching* and *device reconfiguration*.
- The introduction of a Design Space Exploration methodology for multi-objective optimisation of the proposed cascade system's architectural configuration for a target FPGA, considering the *throughput*, *avg. latency* and *accuracy* requirements of the underlying application.

The proposed approach is particularly useful for visual tasks related to autonomous systems, which frequently impose competing high-throughput and low-latency requirements. In Simultaneous Localisation and Mapping (SLAM) for example, where DL models are emergently incorporated to enhance the constructed map with semantic information [12], high frame-rate is necessary to sustain robust localisation (tracking of the mobile agent's position and orientation) [13], while low response time is required for making safe real-time navigational decisions based on the continuously updated map [14].

II. Related Work

Typical neural network approximation approaches consist of computational shortcuts such as quantisation [15], pruning [1] and compression [4], applied uniformly across inputs to exploit the performance-accuracy trade-off. Particularly, low-precision quantisation is becoming increasingly popular, having demonstrated the ability to provide significant performance gains, while reducing the memory footprint and bandwidth requirements (and consequently energy) [16], taking advantage of the inherent redundancy of state-of-the art DL models.

However, when pushed to the limits, as in the case of Binarised Neural Networks (BNNs) [17] [18], these approaches suffer an accuracy degradation due to the existence of challenging samples/regions, being less resilient to approximation. Driven by the fact that not all inputs require the same amount of computation, recent literature has studied *conditional-computation* approaches, targeting to avoid computationally expensive workloads on inference, except where needed.

Karpathy et al. utilised an additional CNN model focusing on the centre of each frame (usually demonstrating higher concentration of information), alongside the main model acting on the entire image, to unevenly increase the model's capacity across the input for the task of video classification [19].

Dynamic Capacity Networks (DCN) [20] divert from the uniform/predetermined workload distribution paradigm by using a attention-based mechanism to resort to variable-capacity sub-networks for different spatial locations of the input image, allocating higher workload to the regions identified as task-relevant at runtime. Applied on the task of Recurrent Neural Network (RNN)-based speech recognition, DeltaRNN [21] forms a temporal variant of this methodology, updating the output of each neuron only when a significant difference is evident compared to its previous activation.

Focusing on the task of image super-resolution, MobiSR [22] evaluates the upscaling difficulty of low-resolution input images, in order to schedule their inference on heterogeneous compute platforms, incorporating models with variable workload-accuracy characteristics.

In recent years, cascade models comprising multiple connected classification units of increasing complexity, are gaining attention. The output of each unit is evaluated to determine whether the computation of the current input will be terminated or forwarded to the next stage. In [23] a general CNN-based classifier is responsible to route the more challenging input samples through cascaded "expert" classifiers. These classifiers are trained on subgroups of classes, formed based on the model's confusion matrix, targeting to boost prediction accuracy at an extra computation cost as required.

In the landscape of systems, CascadeCNN [7] exploits the performance-accuracy trade-off provided by CNN quantisation to build a two-stage FPGA-based classifier. As depicted in Fig. 2a, initially, an excessively quantised low-precision unit (LPU; e.g. 4 bits) is employed to obtain a rapid classification for every sample. Based on the confidence of each prediction (estimated by the spikiness of the probability distribution (Fig. 1)), a confidence evaluation unit (CEU) either terminates the computation feeding-forward this prediction to the output, or directs the sample to a more computationally demanding high-precision unit (HPU; e.g. 8 bits) for re-processing. Towards the edge of this philosophy, [24] introduces a heterogeneous cascade featuring a fully binarised FPGA-based (high-throughput) CNN implementation alongside a CPU-mapped floating-point (high-accuracy) network, with a trainable Perceptron acting as an evaluator on the BNN's predictions.

III. Background: CascadeCNN

Our approach builds on our previous work, CascadeCNN [7], in which an automated toolflow generates a custom hardware architecture of cascaded variable-precision units, tailored to the target CNN-FPGA pair. CascadeCNN automatically selects the precision for both the LPU and the HPU, after a guided search across the quantisation space, utilising a user-provided validation set. Dissimilar to related works automatically traversing the quantisation space [15] [25], in CascadeCNN the precision of the LPU is excessively reduced to provide performance gains enabled by the enhanced flexibility of FPGAs, at the expense of a considerable accuracy loss on certain inputs, whereas the HPU precision is tuned to achieve the desired accuracy. The rigidity of the confidence evaluator is tuned with respect to a user-specified error tolerance, to identify the LPU's missclassified samples and forward them to the HPU for reprocessing, restoring the overall accuracy.

Subsequently, Design Space Exploration (DSE) is performed independently for each unit, through an analytical

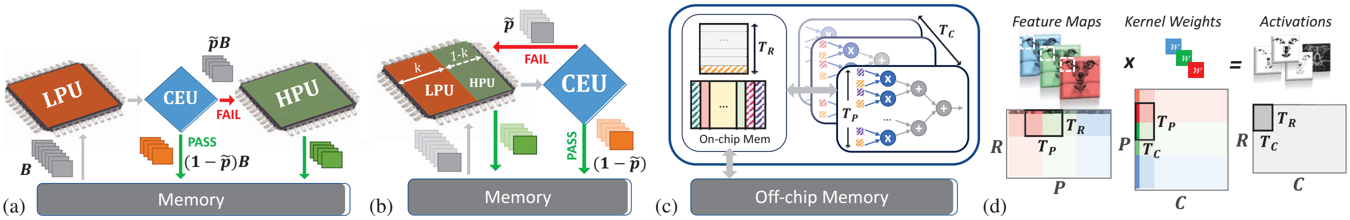

Fig. 2: (a) CascadeCNN [7]. (b) Proposed resource-sharing design. (c) Configurable hardware architecture. (d) Convolution as tiled matrix multiplication.

performance model of the developed architecture to configure its tunable parameters, corresponding to various parallelism dimensions. Convolutional and Fully Connected Layers of the model are uniformly mapped to Matrix Multiplication (MM) operations, similar to [26]. The feature-map ($R\times P$) and weight ($P\times C$) matrices of each layer (Fig. 2d), being stored in the off-chip memory, are accessed in a tiled manner ($T_R\times T_P$ and $T_P\times T_C$) using double buffering. The hardware architecture consists of Multiply-and-Accumulate (MACC) Processing Elements (PEs), organised in a configurable structure that offers control over inter- and intra-PE parallelism through tile sizes T_P and T_C respectively (Fig. 2c). Furthermore, T_R provides control over the pipeline depth of scheduled dot-product operations on the PEs.

Upon deployment, the device is regularly reconfigured between the LPU- and the HPU-optimised implementations (Fig. 2a). To alleviate the performance cost of reconfiguration, a large batch size B is employed (bounded by the available off-chip memory of the target platform). Although this strategy is effectively sustaining high throughput, the inference latency for the portion of the inputs forwarded to the HPU for re-classification is severely penalised, prohibiting its deployment in latency-sensitive applications.

IV. Methodology

To remedy the latency burden of the existing design flow, in this work we adopt CascadeCNN's quantisation and confidence evaluation methodology, but employ a resource-sharing approach to replace the previously compulsory device reconfiguration between the LPU and HPU deployment stages and its consequential need for input-batching, enabling its deployment to latency-sensitive scenarios. The proposed method, introduces a configurable custom hardware architecture in which the LPU and HPU are deployed concurrently on the same FPGA device (Fig. 2b), forming a two-stage cascade. Alongside, we propose a novel multi-stage Design Space Exploration methodology that repeatedly expands and prunes the design space to arrive at a set of platform-supported {LPU,HPU} pairs, out of which a resource-efficient and highly-optimised architectural configuration is selected, maximising inference throughput while satisfying user-specified latency and accuracy requirements. This approach is effectively reducing the complexity of an exhaustive configuration search, maintaining the ability to identify the best cascade design point. In more detail, our approach consists of the following stages:

(a) Independent LPU and HPU Performance Analysis. Starting from the single-stage hardware architecture of Fig. 2c, initially, an exhaustive performance analysis of all possible architectural configurations is conducted for the selected LPU and HPU precision (wordlength; WL) independently, using a developed roofline-based model [27]. Iterating through values for the architecture's tunable parameters, the attainable performance for each layer ℓ of the target CNN is estimated as:

$$Perf_\ell = \frac{wkld_\ell}{II_\ell} = \frac{2R_\ell P_\ell C_\ell}{R_\ell \lceil \frac{P_\ell}{T_P} \rceil \lceil \frac{C_\ell}{T_C} \rceil} f_{WL} \ \ (ops/s) \qquad (1)$$

where f_{WL} denotes the achieved clock frequency of the hardware architecture when adopting a given precision, $wkld_\ell$ the workload and II_ℓ the Initiation Interval of layer ℓ. Accordingly, the performance model considers the Computation-to-(off-chip memory) Communication (CTC) ratio of each design point, employing the computation of an output tile ($T_R \times T_C$) as a unit to capture both write and read transactions:

$$CTC_\ell = \frac{tileOps}{tileMemAcc} = \frac{2T_R P_\ell T_C}{(T_R P_\ell + P_\ell T_C + T_R T_C)WL} \ \ (ops/bit) \qquad (2)$$

Finally, the on-chip memory requirements of each architectural configuration, across the CNN, are calculated as:

$$onChipMem = 2(T_R T_P + T_P T_C + T_R T_C)WL \ \ (bits) \qquad (3)$$

In this manner, each design point's computational and memory demands are fully characterised for both units.

(b) LPU and HPU Design Space Pruning. Having expanded to all possible architectural configurations of each unit, the subspace of platform-supported design points concerning the target FPGA device is identified by establishing a theoretical design-specific computational (CR) and memory (MR) roof. The main limiting factors of the platform's peak performance consist of its available computational (DSPs, LUTs) and memory (on-chip storage, off-chip bandwidth) resources, as well as the implementation-specific resource utilisation per computational element (obtained through benchmarking). Design points overshooting the CR ($\exists \ell \in [1, NL] : Perf_\ell >$ CR) are discarded, whereas design points met above the MR are projected to the MR (Fig. 3d). After this step, the overall performance of each design point (across all NL layers) is estimated as a weighted average w.r.t each layer's workload:

$$Perf = \sum_{\ell=1}^{NL} wkld_\ell Perf_\ell / \sum_{\ell=1}^{NL} wkld_\ell \qquad (4)$$

(c) Exploration of Cascaded LPU-HPU Combinations. Different from relevant literature employing single-stage hardware architectures, that would select a design point optimising the performance objective at this point of the analysis (e.g. [28]), we employ a resource sharing approach targeting to fit an {LPU,HPU} pair concurrently on the target device. Therefore, having identified the sets of platform-supported design points DP_{LPU} and DP_{HPU} for the LPU and HPU accordingly, the combinatorial pair space of {LPU, HPU} = $DP_{\text{LPU}} \times DP_{\text{HPU}}$ is explored. To reduce this search space, individual design points demonstrating excessively low performance or approaching the platform's CR are discarded.

Providing that the LUT utilisation for a single MACC operation continuously scales with wordlength, whereas DSPs can only take advantage of excessively low-precision arithmetic through discrete packing of MACCs [29], the proposed framework evaluates the LPU and HPU wordlengths, in accordance to resource utilisation models obtained through benchmarking ($LUTperMACC = \mathcal{F}(WL) \in \mathbb{N}^*$ and $DSPperMACC = \mathcal{G}(WL) \in \{1/4, 1/2, 1, 2, 4\}$), in order to perform an efficient allocation of FPGA resources between the two units.

(d) Cascade Design Space Pruning. In the proposed approach, the LPU and HPU operate concurrently on the same device, being allocated dedicated computational and on-chip memory resources but competing for the off-chip memory bandwidth (BW). Consequently, all {LPU, HPU} pairs whose cumulative computational resource requirements are exceeding the availability of the target platform are discarded. Furthermore, each unit processes a different sample and may perform calculations corresponding to different layers of the CNN at each instant. Therefore, we model the joint memory needs of both units, considering the BW requirement ($bits/s$) of each design point as the ratio between CTC_ℓ and $Perf_\ell$ (equivalent to the tangent of ϕ, as illustrated in Fig. 3d):

$$BW_\ell = \tan\left(\measuredangle(CTC_\ell, 0)(0,0)(CTC_\ell, Perf_\ell)\right) = \tan(\phi_\ell) = \tfrac{CTC_\ell}{Perf_\ell} \quad (5)$$

Two modelling approaches are investigated: (i) *worst-case*: considers the maximum BW requirement of each model:

$$BW = \max_{\ell=1}^{NL}\left(\tan(\phi_\ell^{(LPU)})\right) + \max_{\ell=1}^{NL}\left(\tan(\phi_\ell^{(HPU)})\right) \quad (6)$$

while (ii) *avg-case* considers the average BW between layers, weighted w.r.t. each layer's workload:

$$BW = \frac{\sum_{\ell=1}^{NL} wkld_\ell tan(\phi_\ell^{(LPU)})}{\sum_{\ell=1}^{NL} wkld_\ell} + \frac{\sum_{\ell=1}^{NL} wkld_\ell tan(\phi_\ell^{(HPU)})}{\sum_{\ell=1}^{NL} wkld_\ell} \quad (7)$$

In either approach, design point pairs that overpass the target platform's available memory BW are discarded.

(e) Multi-objective Optimisation Search. All the remaining {LPU,HPU} design pairs comprise platform-supported cascade design-points. The proposed methodology formulates the selection of the best architectural configuration as a multi-objective optimisation problem, in view of the attainable throughput ($Thpt$) and avg. latency (t_{avg}) of each cascade, to accommodate the application-specific requirements.

Considering the total workload $wrkld = \sum_{\ell=1}^{NL} wkld_\ell$ of the base model, the response latency of each individual design point for unit U can be estimated as: $\hat{t}_U = wkld/Perf^{(U)}$. Therefore, given the sample re-processing ratio $\tilde{p}$ characterising the employed confidence evaluator (configured to meet the user-specified error tolerance), the overall prediction latency of the cascade classifier is formulated as:

$$t_{avg} = \hat{t}_L + \tilde{p} \cdot \left(\hat{t}_H + \sum_{i=1}^{\lceil \hat{t}_H/\hat{t}_L \rceil} \tilde{p} \cdot (\hat{t}_H - i\hat{t}_L)\right) \quad (8)$$

where the last term accounts for the HPU "waiting time", when consecutive missclassifications of the LPU are queuing for re-processing to the slower HPU (Fig. 3c). However, since in the proposed system the LPU and HPU operate concurrently, the LPU can receive new inputs at a rate depending solely to its response latency: $Thpt = wkld/\hat{t}_L \ \ (ops/s) \quad | \quad \hat{t}_L \geq \tilde{p}\hat{t}_H$ (9)

In order to effectively project the LPU's input rate to the output of the overall cascade system, the condition in the second part of Eq. (9), ensuring that the HPU is "fast enough" to accommodate the LPU's missclassification rate and avoid stacking of samples, should be satisfied. Therefore, cascade design points violating this condition are abandoned.

V. Evaluation

A. Experimental Setup

In our experiments, following the practise of related works, we evaluate the proposed cascade classifiers across the validation set of ImageNet dataset [30], mapping AlexNet [31] and VGG-16 [32] models to the platform. Our hardware designs have been synthesised and placed-and-routed with Xilinx Vivado HLS and Design Suite (v17.2), targeting a Xilinx Zynq ZC706 board. Performance measurements are conducted using the ARM CPU of the embedded FPGA platform, which is also used to schedule computations on the architecture's Processing Elements and set-up the off-chip memory transactions. MATLAB 2017b has been employed to implement and run the proposed DSE methodology.

Two CNN cascades are developed for AlexNet and VGG-16, following the methodology introduced in [7]. By setting the error tolerance to be below 3.5 percentage points (p.p.), the toolflow selects 4-bit precision for the weights and computations of the LPU and 7-bit representation for the HPU of both models. Alongside, the CEU is configured to satisfy the predetermined accuracy requirements, yielding a sample re-processing ratio $\tilde{p}$ of 46.3% and 36.5% for AlexNet and VGG-16 respectively. Given the above cascaded CNN models, the proposed methodology (described in Sec. IV) is used to develop optimised cascade hardware architectures for each model, embodying the introduced resource-sharing approach.

B. Baselines

We compare the proposed two-stage cascade classifier with state-of-the-art single-stage approaches from the literature [2], [33]–[35], as well as a strong HPU baseline implemented based on the proposed hardware architecture. These approaches follow the typical uniform compute paradigm, in contrast to our proposed input-dependent computation scheme. Due to the simplicity of single-stage classifiers (Fig. 3a), the typical performance metrics are adopted, with throughput ($Thpt$) being inversely proportional to latency (t_H):

$$t_{avg} = t_H, \quad Thpt = wkld/t_H \quad (10)$$

We also conduct a comparison with CascadeCNN [7], which is most closely related to our work. Being optimised for throughput[1], CascadeCNN embodies device reconfiguration and batch processing, shaping its performance metrics as:

$$\begin{aligned} t_{avg} = t_L + \tilde{p} \cdot \Big((B\text{-}1)/2\ \ t_L + t_{Rcfg} + \tilde{p} \cdot (B\text{-}1)/2\ \ t_H + t_H\Big) \\ Thpt = B\ \ wkld/(Bt_L + \tilde{p}Bt_H + t_{Rcfg}) \end{aligned} \quad (11)$$

where t_L and t_H denote the response latency of LPU and HPU respectively, when deployed across the resources of

[1] Featuring a favorable Batch Size B.

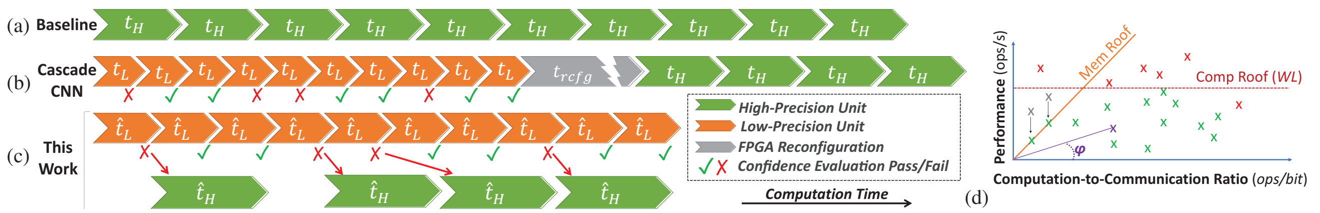

Fig. 3: Processing timeline of a 10-sample input instance processed by: (a) Baseline HPU, (b) CascadeCNN [7] and (c) this work. (d) Roofline model.

the target FPGA and t_{Rcfg} the board-specific reconfiguration time. As depicted in Fig. 3b and described in Eq. (11), a proportion $\tilde{p}$ samples of the batch B is forwarded by the CEU to the HPU for re-processing. These samples suffer a substantial latency penalty, including the time for processing the remainder of the batch on the LPU (consisting of $(B\text{-}1)\tilde{p}/2$ samples on avg. case), device reconfiguration, "queuing time" while other ($(B\text{-}1)\tilde{p}^2/2$ on avg.) samples of the batch requiring re-processing are served, and HPU processing time. In contrast, the proposed resource-sharing approach sacrifices part of the LPU speed-up to eliminate the need for batching and reconfiguration time. In this manner, misclassified samples can be forwarded directly to the concurrent HPU unit (Fig. 3c), being prone to significantly reduced latency overhead (Eq. (8)).

C. Performance Comparison

The results of these comparisons are listed in Table I. All the examined methods are implemented on the same FPGA board. For both benchmark models, the achieved clock frequency and the numerical precision of the underlying implementation are reported, along with its throughput and avg. latency. For cascaded and excessively quantised models, the accuracy degradation with respect to the original CNN is also reported.

Overall, it can be seen that the proposed methodology is providing significant speed-ups both in terms of throughput and latency, compared to single-stage approaches from the literature, targeting applications that can tolerate a controlled accuracy sacrifice. Furthermore, our two-stage methodology introduces a trade-off between throughput and latency in contrast to the throughput-optimised CascadeCNN and latency-optimised HPU baseline, being able to achieve comparable throughput improvement against the HPU with CascadeCNN, with substantially smaller impact on latency, under the same error tolerance. This enables its adoption by a wider span of applications related to real-time systems, such as Augmented Reality (AR), wearable devices, mobile robots and Unmanned Aerial Vehicles, demanding the consumption of high frame-rate inputs while imposing stringent latency constraints.

In more detail, for AlexNet [31], the proposed throughput-latency co-optimised cascade reaches 87% of the throughput-optimised CascadeCNN implementation's throughput, demonstrating however 677× less latency (on average) under the same error tolerance of 3.5p.p.. Compared to the latency-optimised single-stage HPU baseline built based on the proposed hardware architecture under the same error tolerance, the proposed approach achieves a 1.52× speed-up in throughput, being able to maintain real-time response latency (5.6ms).

Similarly, in the case of VGG-16 [32], the proposed approach sustains up to 99% of the throughput-optimised CascadeCNN operation rate, while demonstrating 341× less response time on average compared to CascadeCNN. This translates to a performance boost of 61% in throughput compared to the HPU single-stage approach, at the expense of a much smaller latency overhead than CascadeCNN.

TABLE I. Comparison with Related Works (Z-7045)

Method	WL	f_{WL}	Latency	Throughput	Error
AlexNet [31]					
[33]	16-bit	125MHz	8.22ms	161.98GOp/s	-
[34]	16-bit	100MHz	12.30ms	108.25GOp/s	-
[35]	16-bit	250MHz	9.95ms	120.30GOp/s	-
HPU	16-bit	131MHz	7.1ms	304.04GOp/s	-
HPU	7-bit	150MHz	2.9ms	747.46GOp/s	≤3.5p.p.
[7]	[4,7]bit†	150MHz	3793.9ms	1310.7GOp/s	≤3.5p.p.
This work	[4,7]bit†	150MHz	5.6ms	1139.9GOp/s	≤3.5p.p.
VGG-16 [32]					
[33]	16-bit	125MHz	249.50ms	123.12GOp/s	-
[2]	16-bit	150MHz	163.42ms	187.80GOp/s	-
HPU	16-bit	131MHz	104.4ms	293.9GOp/s	-
HPU	7-bit	150MHz	40.8ms	751.9GOp/s	≤3.5p.p.
[7]	[4,7]bit†	150MHz	25991.5ms	1222.9GOp/s	≤3.5p.p.
This work	[4,7]bit†	150MHz	76.1ms	1213.1GOp/s	≤3.5p.p.

† Denotes precision for [LPU,HPU] cascade pair.

Additionally, the comparison with 16-bit works from the literature indicates that there exists significant space for adoption of input-dependent computation methodologies, such as this work, for applications that can tolerate a controlled accuracy degradation (e.g. ≤3.5p.p for the examined instance), to gain substantial performance improvement both in throughput and latency. Indicatively, the proposed approach provides gains of 3.74×-10.5× in throughput and 1.46×-2.2× in avg. latency, compared to its 16-bit single-stage counterparts for AlexNet; and 6.45×-9.85× in throughput and 2.14×-3.27× in avg. latency for VGG-16 accordingly.

D. Performance Model Accuracy

Finally, the accuracy of the developed performance models is evaluated across multiple points of the design space, by comparing the predicted and the measured performance of the hardware architecture. This is in order to capture the confidence in generating the best designs from the DSE. The results are summarised in Table II.

In the case of the precision-aware roofline model focusing on each unit independently (Sec. IVa,b), an error of 6.8% (geo. mean) is revealed (row 1). I/O delay variations and software overhead are considered as driving factors of the model error.

Considering the whole cascade system (Sec. IVc-e), when adopting the *avg-case* memory model of Eq. (7) the error (row 2) expands up to 9.7% (geo. mean), accounted to the non-uniform distribution of LPU missclassifications across the inputs, and the unpredictable off-chip memory read/write

TABLE II. PERFORMANCE MODEL ACCURACY

Performance Model	Error: AlexNet	Error: VGG-16
Individual LPU/HPU	6.8%	6.8%
Combined - *avg-case* Mem.	9.2%	9.7%
Combined - *worst-case* Mem.	8.4%	-

transaction scheduling, when the available bandwidth is surpassed. When the worst-case memory model (Eq. (6)) is used, the model error is reduced to 8.4% in the case of AlexNet (row 3), due to the stringent barrier established on the memory bandwidth requirements of both compute units. However, this model fails to provide any valid combination of design points for VGG-16, conceivably due to its challenging (and unevenly distributed between layers) memory requirements [10].

VI. CONCLUSION

In this work we introduce a novel FPGA-based configurable architecture for cascaded CNN classifiers, along with an analytical performance model and design space exploration methodology to optimise the architectural configuration with respect to a target CNN-FPGA pair. Eliminating the need for device reconfiguration and input batching, the proposed approach introduces a new design point providing a significantly improved trade-off between throughput and avg. latency compared to related works, enabling its deployment on latency-critical applications in the embedded landscape.

REFERENCES

[1] A. Zhou, A. Yao, Y. Guo, L. Xu, and Y. Chen, “Incremental Network Quantization: Towards Lossless CNNs with Low-Precision Weights,” in *Int. Conf. on Learning Representations (ICLR)*, 2017.

[2] J. Qiu, J. Wang, S. Yao, K. Guo, B. Li, E. Zhou, J. Yu, T. Tang, N. Xu, S. Song *et al.*, “Going Deeper with Embedded FPGA Platform for Convolutional Neural Network,” in *ACM/SIGDA Int. Symp. on Field-Programmable Gate Arrays (FPGA)*, 2016.

[3] S. Han, J. Pool, J. Tran, and W. Dally, “Learning Both Weights and Connections for Efficient Neural Network,” in *Advances in Neural Information Processing Systems (NIPS)*, 2015.

[4] S. Han, H. Mao, and W. J. Dally, “Deep Compression: Compressing Deep Neural Network with Pruning, Trained Quantization and Huffman Coding,” *Int. Conf. on Learning Representations (ICLR)*, 2016.

[5] S. Han, J. Kang, H. Mao, Y. Hu, X. Li, Y. Li, D. Xie, H. Luo, S. Yao, Y. Wang *et al.*, “ESE: Efficient Speech Recognition Engine with Sparse LSTM on FPGA,” in *ACM/SIGDA Int. Symp. on Field-Programmable Gate Arrays (FPGA)*, 2017.

[6] M. Jaderberg, A. Vedaldi, and A. Zisserman, “Speeding up Convolutional Neural Networks with Low Rank Expansions,” in *British Machine Vision Conference (BMVC)*, 2014.

[7] A. Kouris, S. I. Venieris, and C. Bouganis, “CascadeCNN: Pushing the Performance Limits of Quantisation in Convolutional Neural Networks,” in *Int. Conf. on Field Programmable Logic & Applications (FPL)*, 2018.

[8] S. Teerapittayanon, B. McDanel, and H.-T. Kung, “BranchyNet: Fast Inference via Early Exiting from Deep Neural Networks,” in *Int. Conf. on Pattern Recognition (ICPR)*, 2016.

[9] E. Wang, J. J. Davis, R. Zhao, H.-C. Ng, X. Niu, W. Luk, P. Y. Cheung, and G. A. Constantinides, “Deep Neural Network Approximation for Custom Hardware: Where We’ve Been, Where We’re Going,” *ACM Computing Surveys (CSUR)*, vol. 52, no. 2, 2019.

[10] S. I. Venieris, A. Kouris, and C.-S. Bouganis, “Toolflows for Mapping Convolutional Neural Networks on FPGAs: A Survey and Future Directions,” *ACM Computing Surveys (CSUR)*, vol. 51, no. 3, 2018.

[11] S. I. Venieris and C.-S. Bouganis, “fpgaConvNet: A Framework for Mapping Convolutional Neural Networks on FPGAs,” in *Int. Symp. on Field-Programmable Custom Computing Machines (FCCM)*, 2016.

[12] J. McCormac, A. Handa, A. Davison, and S. Leutenegger, “Semanticfusion: Dense 3d Semantic Mapping with Convolutional Neural Networks,” in *Int. Conf. on Robotics and Automation (ICRA)*, 2017.

[13] A. Handa, R. A. Newcombe, A. Angeli, and A. J. Davison, “Real-time Camera Tracking: When is High Frame-rate Best?” in *European Conference on Computer Vision (ECCV)*, 2012.

[14] K. Boikos and C. Bouganis, “A High-Performance System-on-Chip Architecture for Direct Tracking for SLAM,” in *Int. Conf. on Field Programmable Logic and Applications (FPL)*, 2017.

[15] P. Gysel, J. Pimentel, M. Motamedi, and S. Ghiasi, “Ristretto: A Framework for Empirical Study of Resource-Efficient Inference in Convolutional Neural Networks,” *IEEE Transactions on Neural Networks and Learning Systems (TNNLS)*, vol. 29, no. 11, 2018.

[16] S. Hashemi, N. Anthony, H. Tann, R. I. Bahar, and S. Reda, “Understanding the Impact of Precision Quantization on the Accuracy and Energy of Neural Networks,” in *Design, Automation & Test in Europe Conference (DATE)*, 2017.

[17] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, “Binarized Neural Networks,” in *Advances in Neural Information Processing Systems (NIPS)*, 2016.

[18] Y. Umuroglu, N. J. Fraser, G. Gambardella, M. Blott, P. Leong, M. Jahre, and K. Vissers, “FINN: A Framework for Fast, Scalable Binarized Neural Network Inference,” in *ACM/SIGDA Int. Symp. on Field-Programmable Gate Arrays (FPGA)*, 2017.

[19] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, “Large-scale Video Classification with Convolutional Neural Networks,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[20] A. Almahairi *et al.*, “Dynamic Capacity Networks,” in *Int. Conf. on Machine Learning (ICML)*, 2016.

[21] C. Gao, D. Neil, E. Ceolini, S.-C. Liu, and T. Delbruck, “DeltaRNN: A Power-efficient Recurrent Neural Network Accelerator,” in *ACM/SIGDA Int. Symp. on Field-Programmable Gate Arrays (FPGA)*, 2018.

[22] R. Lee, S. Venieris, Ł. Dudziak, S. Bhattacharya, and N. Lane, “MobiSR: Efficient On-Device Super-Resolution through Heterogeneous Mobile Processors,” in *Mobile Computing and Networking (MobiCom)*, 2019.

[23] Z. Wei *et al.*, “A Self-Adaptive Cascade ConvNets Model Based on Label Relation Mining,” *Neurocomputing*, vol. 328, 2019.

[24] S. Amiri, M. Hosseinabady, S. McIntosh-Smith, and J. Nunez-Yanez, “Multi-Precision Convolutional Neural Networks on Heterogeneous Hardware,” in *Design, Automation & Test in Europe Conf. (DATE)*, 2018.

[25] K. Guo, L. Sui, J. Qiu, J. Yu, J. Wang, S. Yao, S. Han, Y. Wang, and H. Yang, “Angel-Eye: A Complete Design Flow for Mapping CNN onto Embedded FPGA,” *Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 1, 2017.

[26] N. Suda, V. Chandra, G. Dasika, A. Mohanty, Y. Ma, S. Vrudhula, J. Seo, and Y. Cao, “Throughput-optimized OpenCL-based FPGA Accelerator for Large-scale Convolutional Neural Networks,” in *ACM/SIGDA Int. Symp. on Field-Programmable Gate Arrays (FPGA)*, 2016.

[27] S. Williams, A. Waterman, and D. Patterson, “Roofline: an Insightful Visual Performance Model for Multicore Architectures,” *Communications of the ACM*, vol. 52, no. 4, 2009.

[28] C. Zhang *et al.*, “Optimizing FPGA-based Accelerator Design for Deep Convolutional Neural Networks,” in *Int. Symp. on Field-Programmable Gate Arrays (FPGA)*, 2015.

[29] D. Nguyen, D. Kim, and J. Lee, “Double MAC: Doubling the Performance of Convolutional Neural Networks on Modern FPGAs,” in *Design, Automation & Test in Europe Conference (DATE)*, 2017.

[30] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “ImageNet: A Large-Scale Hierarchical Image Database,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[31] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet Classification with Deep Convolutional Neural Networks,” in *Advances in Neural Information Processing Systems (NIPS)*, 2012.

[32] K. Simonyan and A. Zisserman, “Very Deep Convolutional Networks for Large-Scale Image Recognition,” in *Int. Conf. on Learning Representations (ICLR)*, 2015.

[33] S. I. Venieris and C.-S. Bouganis, “Latency-driven Design for FPGA-based Convolutional Neural Networks,” in *Int. Conf. on Field Programmable Logic and Applications (FPL)*, 2017.

[34] 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 *Annual Design Automation Conference (DAC)*, 2016.

[35] V. Gokhale, A. Zaidy, A. X. M. Chang, and E. Culurciello, “Snowflake: An Efficient Hardware Accelerator for Convolutional Neural Networks,” in *Int. Symp. on Circuits and Systems (ISCAS)*, 2017.