**FPL'19 Demo-Night Submission – Cover Page**

# Capella: Customizing Perception for Edge Devices by Efficiently Allocating FPGAs to DNNs


**Authors:**

Younmin Bae, Georgia Institute of Technology, Atlanta, GA, USA
Ramyad Hadidi, Georgia Institute of Technology, Atlanta, GA, USA
Bahar Asgari, Georgia Institute of Technology, Atlanta, GA, USA
Jiashen Cao, Georgia Institute of Technology, Atlanta, GA, USA
Hyesoon Kim, Georgia Institute of Technology, Atlanta, GA, USA


## Main Objectives and Relevance to FPL community:

This work seeks to achieve the following two objectives. First, we show the feasibility of implementing large-scale DNN models on the limited resources of edge devices by using cost-efficient computation engines. Second, we use novel edge-tailored models for distributing the DNNs across more-than-one devices without paying for extra communication overhead. The open-source PYNQ™ and TVM project assists us to implement Capella on Xilinx Zynq® systems on chips (SoCs). We have released our implementation from DNN model definition and its training to FPGA code in our Github[a]. We hope our effort in suggesting the implementation of systolic arrays on low-cost FPGAs for Edge application would be appreciated by FPL community.

[a]https://github.com/parallel-ml/Capella-FPL19-SplitNetworksOnFPGA

## The short biography of the presenter:

**Hyesoon Kim** is an associate professor in the School of Computer Science at the Georgia Institute of Technology. Her research interests include high-performance, energy-efficient heterogeneous architectures; interaction between programmers, compilers, and microarchitectures; and developing tools to help parallel programming. Kim has a PhD in computer engineering from the University of Texas at Austin. She is a member of IEEE and the ACM.

## Logistical Requirements:

- Access to power outlet.
- Access to wired Ethernet connection (required for code demonstration on Github).
- Presenter brings:
  - Two PYNQ boards.
  - Network Switch

# Capella: Customizing Perception for Edge Devices by Efficiently Allocating FPGAs to DNNs

Younmin Bae
Georgia Tech

Ramyad Hadidi*
Georgia Tech

Bahar Asgari
Georgia Tech

Jiashen Cao
Georgia Tech

Hyesoon Kim
Georgia Tech

***Abstract*— Deep neural networks (DNNs) have seen resurgent attraction to be implemented in edge applications. However, such implementations are not easy to achieve because execution of DNNs often require more resources than those provided by individual edge devices. On the other hand, relying on model-level distribution methods to implement a DNN on connected edge devices leads to costly communication overheads. To utilize available in-the-edge resources with less communication overhead, we propose using edge-tailored models comprised of nearly-independent narrow DNNs, the inference of which are accelerated using small cost-efficient RISC-based engines. We implement these engines on PYNQ™ boards as a platform that mimics the limited resources of edge devices. We create the narrow DNNs based on the available resources of PYNQ boards, and allocate each narrow DNN to one engine, implemented in an FPGA. We compare the communication overhead of our implantation against the state-of-the-art model-level distribution methods.**

***Index Terms*—Edge Computing, FPGA, DNN.**

## I. Customizing Perception for Edge Devices

To utilize the limited compute resources of edge devices with very slight communication between them, this paper customizes perception for edge devices [1], [2] by efficiently allocating FPGAs to DNNs (Capella). To do so, we divide all layers of the original model into $n$ ($n$: number of devices) equal parts but the pre-final layer. Then, nearly-independent narrow DNNs are created by removing the connections between the branches. The new model needs to be retrained to sustain accuracy. Since the extra connections are removed, each branch has less than $1/n$ of original parameters. Therefore, the memory and computation resources required by each branch are less than those required by partitions created by data- and model-parallelism methods. As an example, Figure 1a illustrates splitting VGG-S into two branches. As Figure 1b illustrates, Capella assigns the computations of a branch to the FPGA of a PYNQ™ board. The computations of the final layer are assigned to the ARM processor of one of the PYNQ boards to reduce unnecessary data movements. In this example, the two branches only need to communicate to exchange their final partial results. Thus, the amount of communication is comparable to the communication required by partitions created by data parallelism and is much less than that required by model parallelism. The ARM processors handle the communication operations.

*All correspondence to rhadidi@gatech.edu.

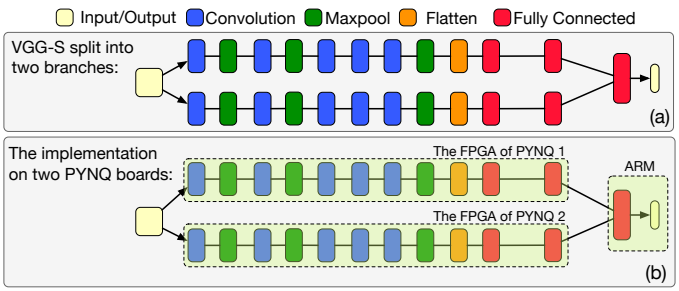


Fig. 1. Splitting VGG-S into two narrow DNNs.

## II. RISC-Based Engine

The computations of each branch are implemented using a RISC-based engine from TVM stack [3] on the FPGA of a PYNQ board. After training the DNN model in MxNet, we use TVM compiler to create RISC-based instructions to be executed on the engine. The RISC-based instructions perform the operations on tensor registers. Instructions get executed in a general matrix-matrix multiplication (GEMM) core on the processing engine for matrix operations by concurrently processing data through buffers and queues. Data is partitioned and tiled to increase the data-reuse rate and minimize accesses to memory. Finally, the outputs are transferred to the main memory to be processed by the ARM processor.

## III. Experimental Studies

We explore the performance and energy efficiency of implementing Capella on PYNQ boards by examining metrics of inference-per-second and energy-per-inference. Our demo is on ResNet-18 model trained on ImageNet dataset and split in two. The outcome narrow and original models are then launched to the PYNQ boards for evaluation. Our result for ResNet-18 shows 2.4× higher throughput for our edge-tailored implementation. We also customize and train VGG16 and AlexNet models on ImageNet dataset as further studies for our edge-tailored models. Generally, we observe around 5% accuracy loss for the split in two models with half of the memory and computation footprint. We released our PYNQ implementation with accompanying MxNet model on Github[1].

## References

[1] R. Hadidi *et al.*, "Distributed perception by collaborative robots," *IEEE RA-L and IROS'18*, vol. 3, Oct 2018.
[2] R. Hadidi *et al.*, "An edge-centric scalable intelligent framework to collaboratively execute dnn," *SysML'19 Demo, Palo Alto, CA*, 2019.
[3] T. Chen *et al.*, "Tvm: end-to-end optimization stack for deep learning," *arXiv preprint arXiv:1802.04799*, 2018.

[1] https://github.com/parallel-ml/Capella-FPL19-SplitNetworksOnFPGA