# A PYNQ-based Framework for Rapid CNN Prototyping

Erwei Wang, James J. Davis and Peter Y. K. Cheung
*Department of Electrical and Electronic Engineering*
*Imperial College London*
*Email:* {erwei.wang13, james.davis, p.cheung}@imperial.ac.uk

***Abstract*—This work presents a self-contained and modifiable framework for fast and easy convolutional neural network prototyping on the Xilinx PYNQ platform. With a Python-based programming interface, the framework combines the convenience of high-level abstraction with the speed of optimised FPGA implementation. Our work is freely available on GitHub[1] for the community to use and build upon.**

## Introduction

Convolutional neural networks (CNNs) have achieved significant success in solving a wide range of classification problems. The inherent parallelism of two-dimensional convolutions matches well with FPGAs' strong parallel-processing capability, leading to higher data processing throughput and/or lower power consumption than reliance on more traditional embedded platforms.

## Framework Highlights

**Modularity:** The framework constructs CNNs using the synchronous dataflow paradigm [1]. A library of CNN layers is provided, which can be used to construct various models.

**Ease of use:** RTL programming is not required.

**Parametrisability:** Users can apply the provided library to different CNN specifications by modifying generic parameters before synthesis with Vivado HLS.

**Python bindings:** The framework follows a "software-down" development flow, where a Python-based application programming interface controls the execution of the hardware and communicates with the rest of the program. The combination of Python software and FPGA performance helps our framework appeal to a broader audience.

**Pre-trained CNN models:** The framework provides tutorials on directly loading and executing pre-trained CNN models in Caffe or Theano.

**Weight reloading:** To achieve higher throughput, parameters are quantised and stored in on-chip memory. Parameters can be reloaded at runtime for flexible deployment.

**Open source:** The framework is BSD licensed. Anyone can freely obtain or contribute to the framework, making it particularly suitable for educational purposes.

1. https://github.com/awai54st/PYNQ-Classification

TABLE 1. EMBEDDED PLATFORM LENET-5 INFERENCING RESULTS.

| | Platform | Quant. | Gop/s | Gop/s/W |
|---|---|---|---|---|
| CPU (PYNQ) | Cortex-A9 | float32 | 0.133 | 0.0479 |
| CPU (TK1) [2] | Cortex-A15 | float32 | 2.73 | 0.546 |
| GPU (TK1) [2] | Kepler | float32 | 7.31 | 0.731 |
| fpgaConvNet [1] | XC7Z020 | fixed16 | 0.480 | 0.274 |
| This work | XC7Z020 | fixed8 | 2.56 | 1.35 |

## Performance Benchmark

We compared our work against other embedded platforms using the publicly available and popular benchmark, LeNet-5 (classifying MNIST handwriting dataset). Table 1 shows that our framework achieves around $20\times$ throughput and $28\times$ power efficiency improvements compared to PYNQ's embedded ARM CPU. It also achieves a $1.8\times$ efficiency increase over a TK1 embedded GPU implementation.

## Conclusion

Although our framework can achieve very good inferencing performance with small CNNs, its scalability is limited since all layers and their weights must be on chip simultaneously. We are currently working on a new framework with a single processing unit which will exploit resource reuse and memory tiling to compute on a layer-by-layer basis instead. With this, we aim to implement large-scale CNN models on embedded FPGA platforms.

## Acknowledgements

This work was supported by EPSRC grants EP/K034448/1, EP/P010040/1 and EP/N031768/1. The authors would also like thank Xilinx's PYNQ team for their support and development board donation.

## References

[1] S. I. Venieris *et al.*, "fpgaConvNet: A Framework for Mapping Convolutional Neural Networks on FPGAs," in *FCCM*, 2016.

[2] K. Guo *et al.*, "Angel-eye: A Complete Design Flow for Mapping CNN onto Customized Hardware," in *ISVLSI*, 2016.