On the Automation of High Level Synthesis of Convolutional Neural Networks

Chicago
RAW @ IPDPS
05/24/2016

E. Del Sozzo, A. Solazzo, A. Miele, M. D. Santambrogio
emanuele.delsozzo@polimi.it
Politecnico di Milano

Rationale

- Convolutional Neural Networks (CNNs) represent the state of the art in image recognition and classification
- CNNs are applied in different fields like Big Data analysis, video surveillance and robot vision
- *However…*
  - *due to the **huge amount of data** to be processed, it is crucial to find techniques to **speed up** the computation*
  - *In particular, the **dataflow pattern** of CNN classification algorithm results to be suitable for **hardware acceleration***

Proposed Solution

- A framework to automatically generate a hardware implementation of CNNs on FPGAs, based on the HLS of configurable offline-trained networks

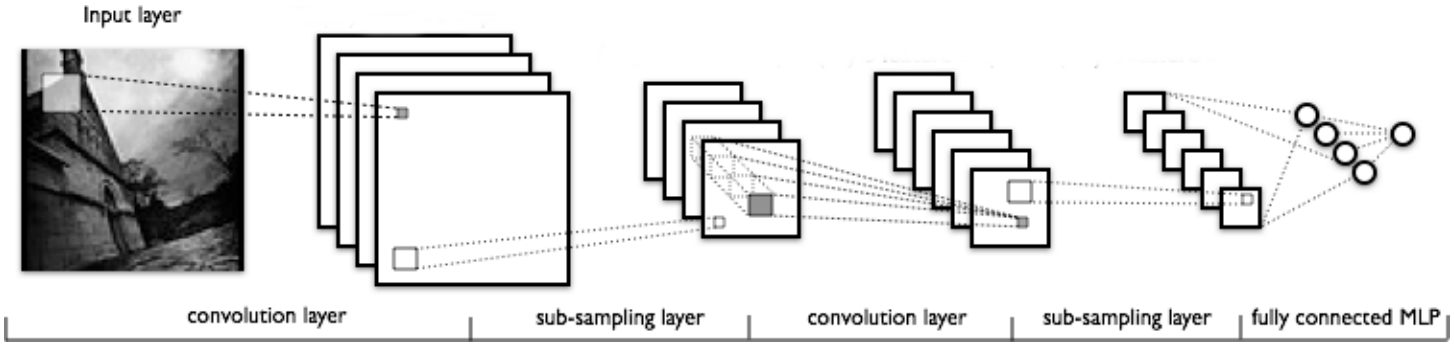
- Main features of the framework:
  - generation of a synthesizable C++ code starting from the weights of a CNN
  - generation of scripts for Xilinx Vivado and Vivado HLS toolchains
  - CNN design customization and support for Zedboard and Zybo platforms

Motivation

- In order to generate the weights of a CNN, a software version of the CNN itself has to be built

- So, why should one use this framework?
  - HLS tools deal only with a small set of programming languages (C, C++, etc.), while machine learning frameworks use many different languages
  - Even though the CNN is implemented in C/C++, it may not be synthesizable

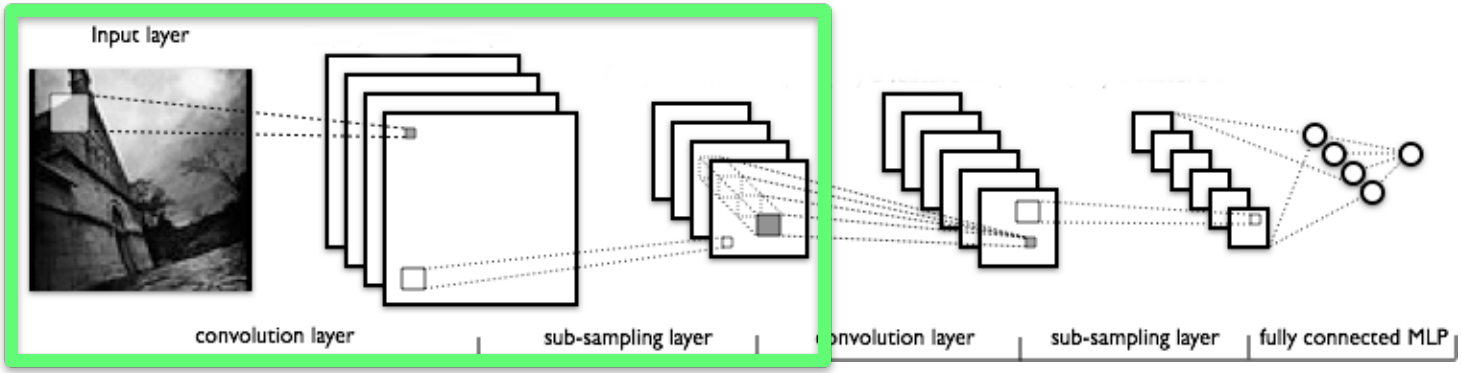
Convolutional Neural Networks



- A CNN is a particular type of Artificial Neural Network inspired by cells in the primary visual cortex of animals [1]
- A CNN is composed of one or more **convolutional** and **linear** layers
- In this example, the CNN is of 2 convolutional layers and 1 linear layer

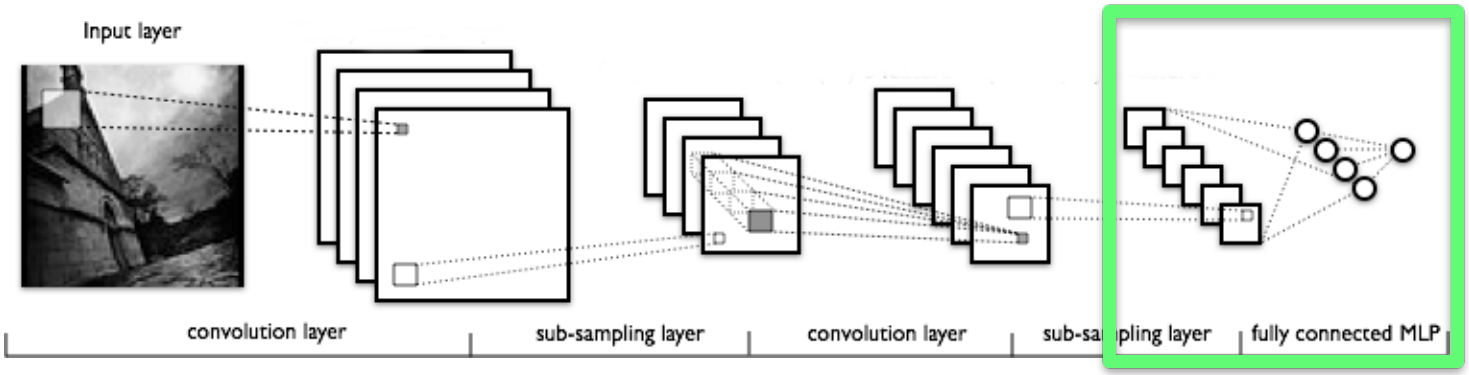
[1] Y. LeCun et al., "Gradient-based learning applied to document recognition," *Proceedings of the IEEE,* 1998

Convolutional Layers



- extract features from images by applying different filters (kernels)
- the more layers are used, the more complex features are extracted
- may be alternated with sub-sampling layers to reduce stored data

Linear Layers



- Implemented as a fully connected *Multi-Layer Perceptron*
- group information collected by convolutional part
- predict the class of the initial input image

State of the Art

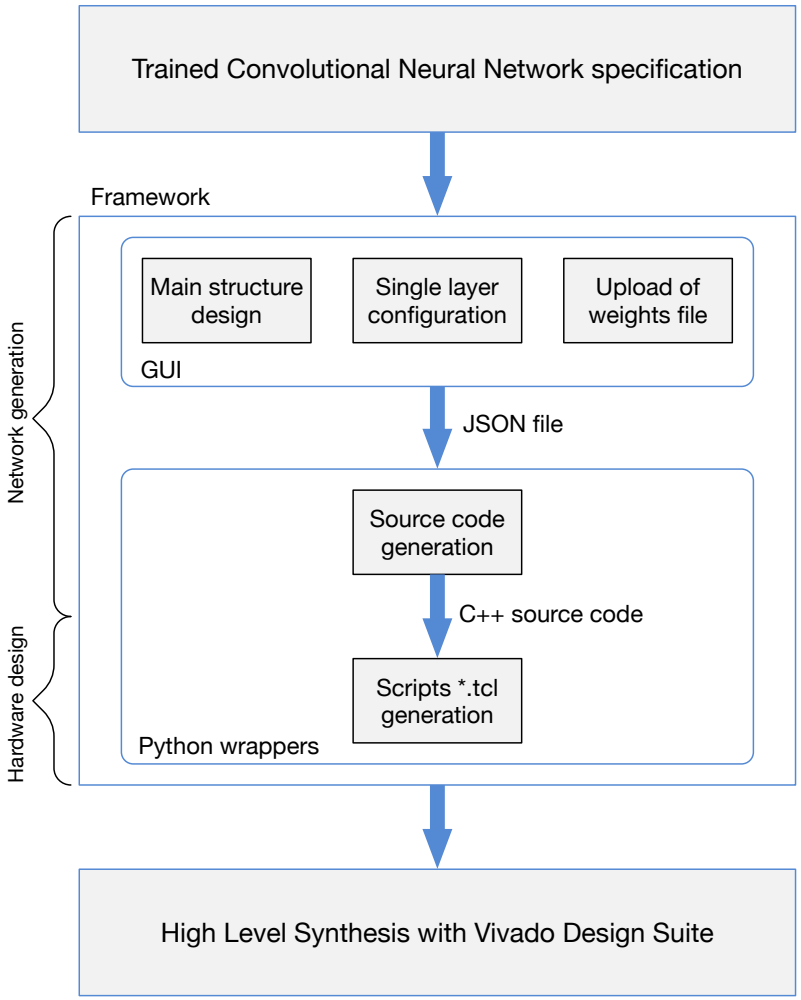
- Nowadays CNNs are employed in different fields:
  - Human action recognition [2]
  - Image classification [3]
  - Natural language processing [4]

- The dataflow pattern of classification phase well suits hardware acceleration on both GPUs [5] and FPGAs [6]

- To the best of our knowledge, there are no available frameworks that ease the synthesis of CNNs on FPGAs

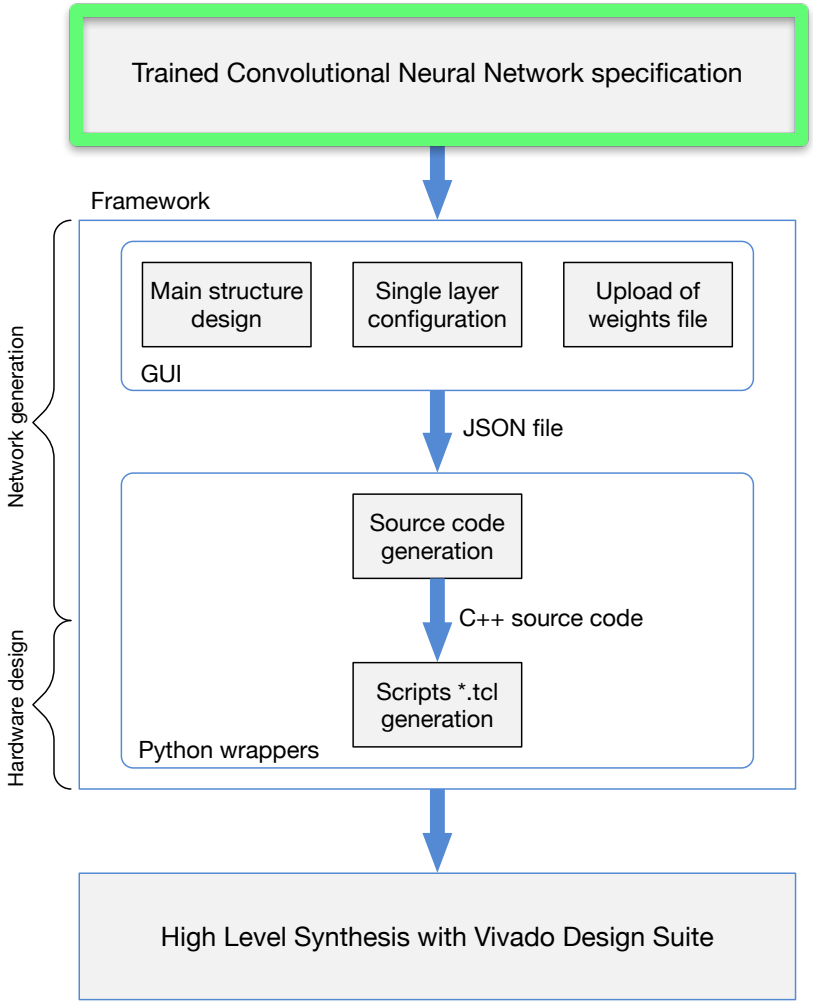
[2] S. Ji et al. "3D convolutional neural networks for human action recognition," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 2013.
[3] A. Krizhevsky et al., "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*, 2012.
[4] R. Collobert et al., "A unified architecture for natural language processing: Deep neural networks with multitask learning," in *Proceedings of the 25th International Conference on Machine Learning*, 2008.
[5] D. C. Ciresan et al., "Flexible, high performance convolutional neural networks for image classification," in *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*, 2011.
[6] C. Zhang et al., "Optimizing fpga-based accelerator design for deep convolutional neural networks," in *Proceedings of the 2015 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, 2015.

The Proposed Framework



- We propose a easy-to-use framework that allows to design and configure a CNN
- The front-end is designed as a web application
- The back-end is designed in Python

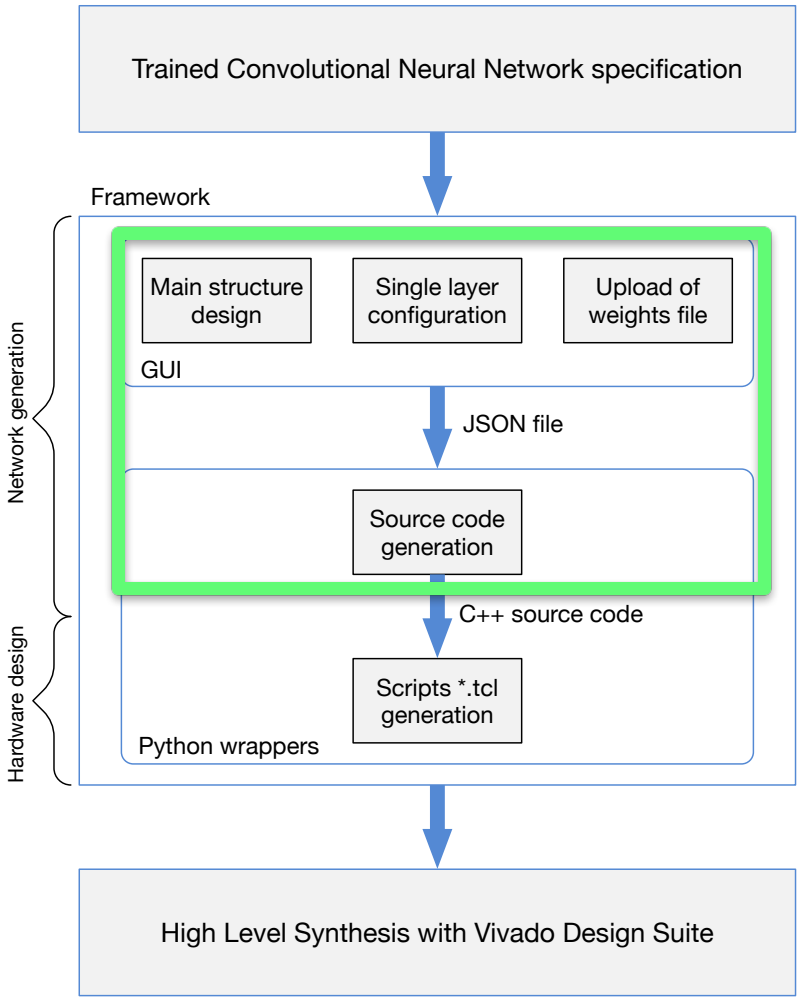
Input



- The input are the weights of a trained CNN
- The weights may be generated by means of machine learning framework like *Torch [7]* and *TensorFlow [8]*

[7] "Torch Framework." [Online]. Available: http://torch.ch
[8] "TensorFlow." [Online]. Avaliable: https://www.tensorflow.org

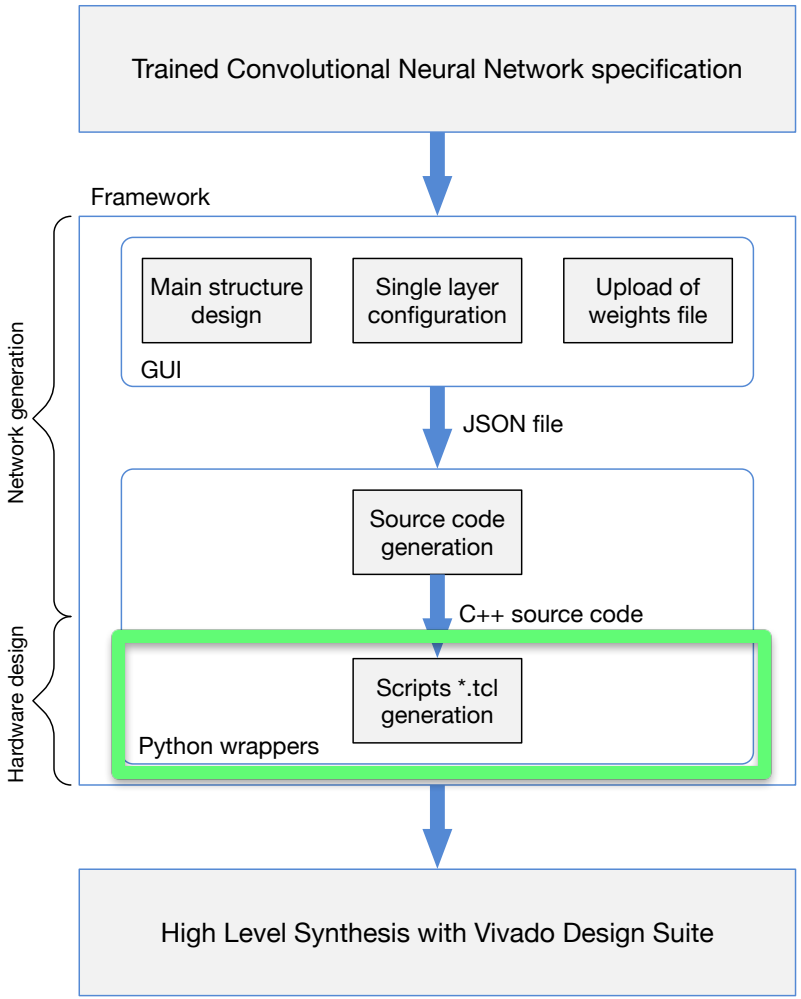
Network Generation



Customization of:

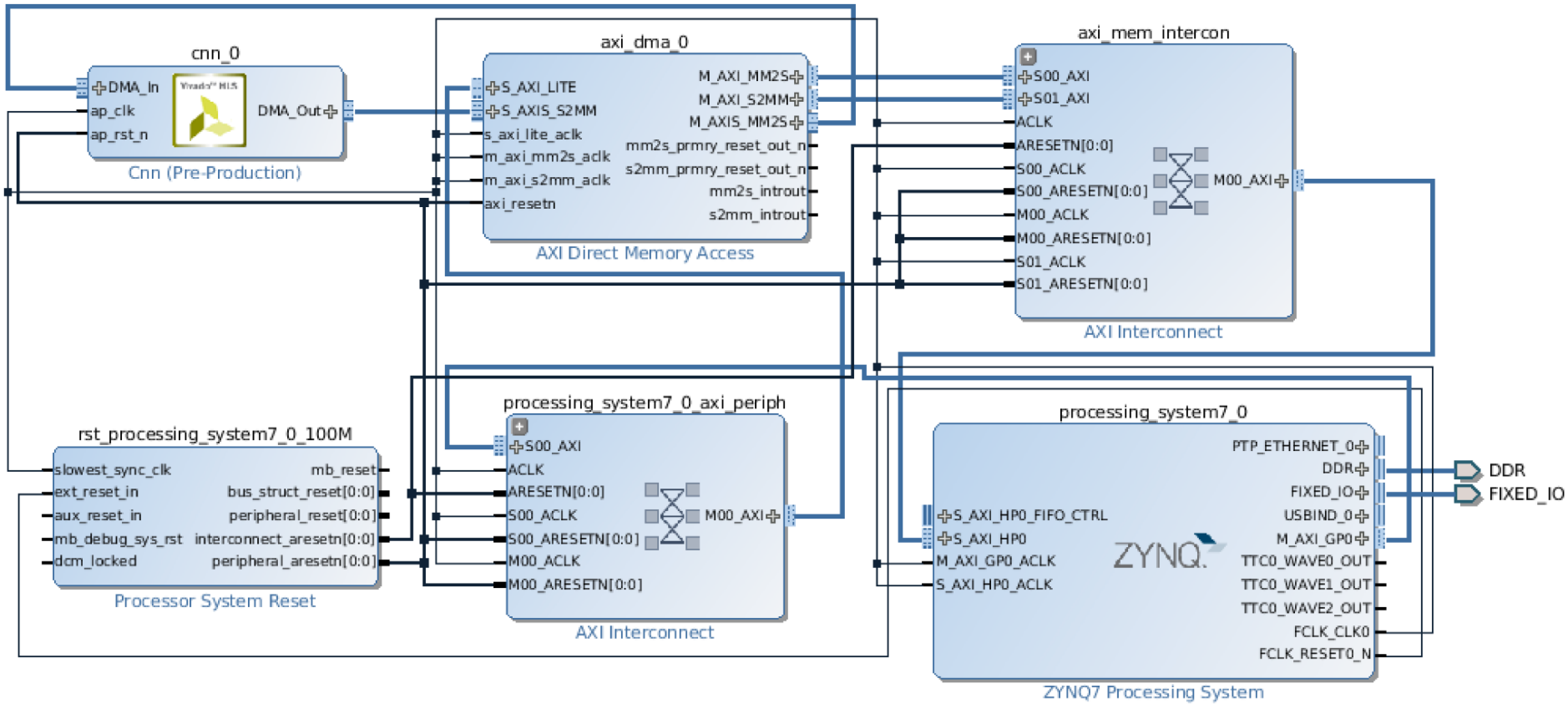
- Convolutional part
  - Number of layers
  - Size and number of kernels
  - Presence of sub-sampling
  - Kernel size of sub-sampling
- Linear part
  - Number of layers
  - Number of neurons

Hardware Design

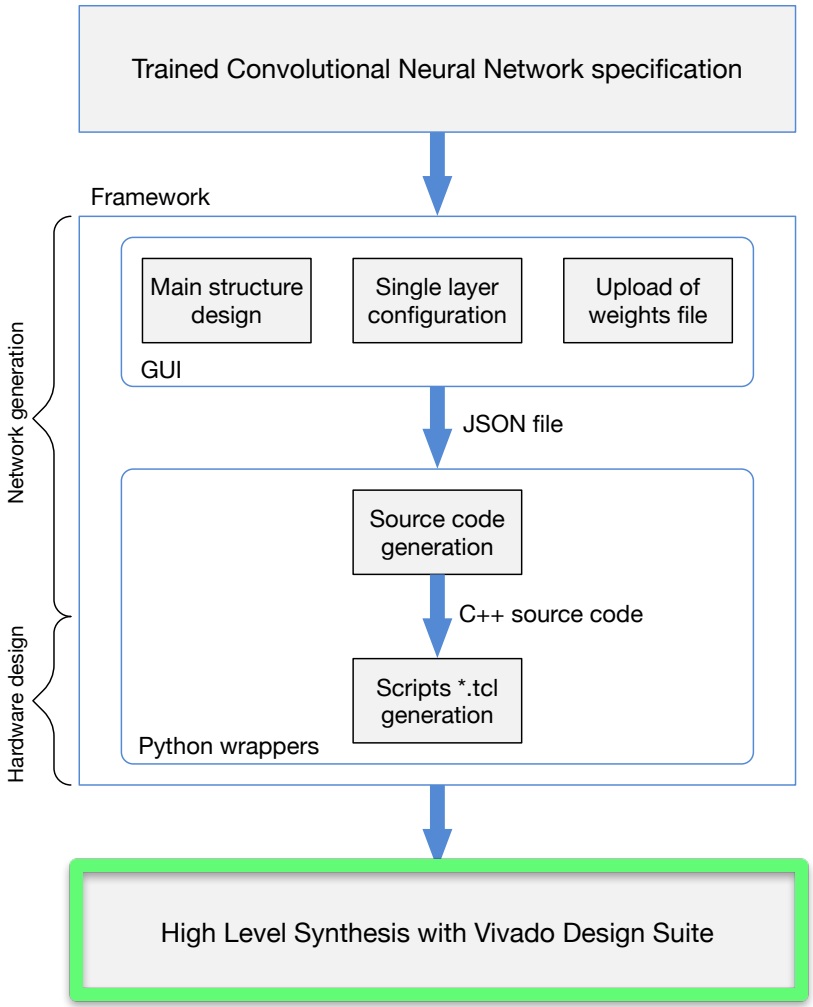


- Choice of target platform (Zybo or Zedboard)
- Hardware design composed of:
  - ZYNQ7 Processing System
  - AXI DMA
  - 2 AXI Interconnect
  - Processor System Reset
  - CNN IP Core
- The CNN IP Core uses *AXI4-Stream Connection* for data streaming

Block Design



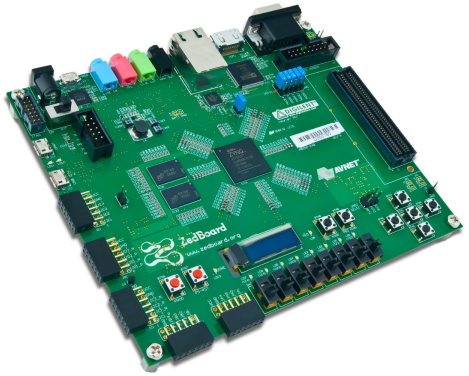
Output



- Generation of
  - CNN C++ source code
  - tcl scripts for Xilinx Vivado and Vivado HLS toolchains (2015.2 version)
- HLS and bitstream generation is (at the moment) up to the user

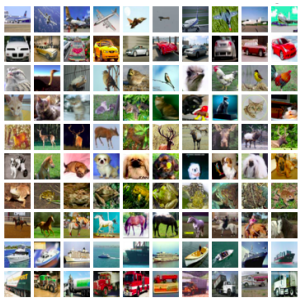
Experimental Results

- We synthesized different types of CNNs for Zedboard platform



- FPGA performance were compared with ARM A9 processor in terms of:
  - Prediction error
  - Execution time
  - Power/energy consumption

- We employed USPS and CIFAR-10 [9] datasets

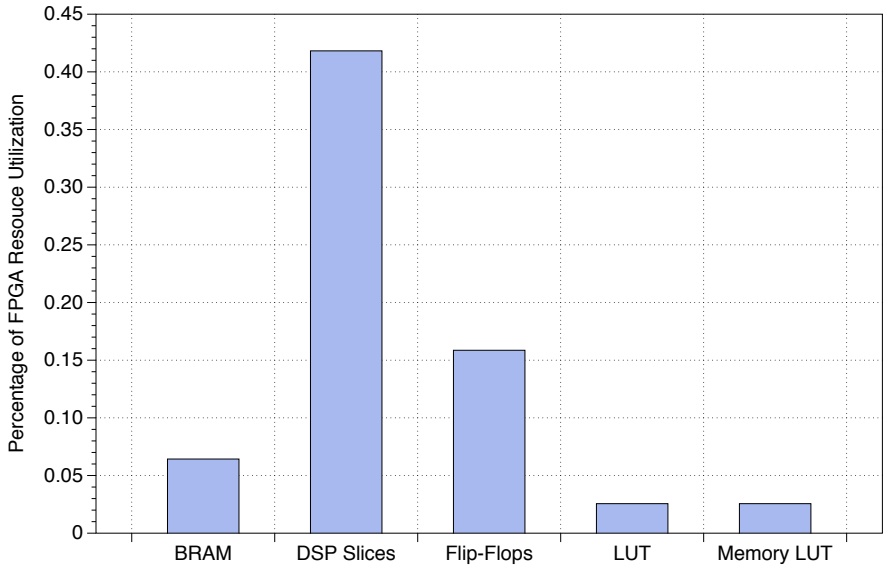


[9] “CIFAR-10.” [Online]. Available: http://www.cs.toronto.edu/~kriz/cifar.html

Test 1

Setup

- 16x16 grayscale USPS Dataset
- one convolutional layer:
  - six 5x5 kernels and sub-sampling
- one linear layer:
  - 10 neurons

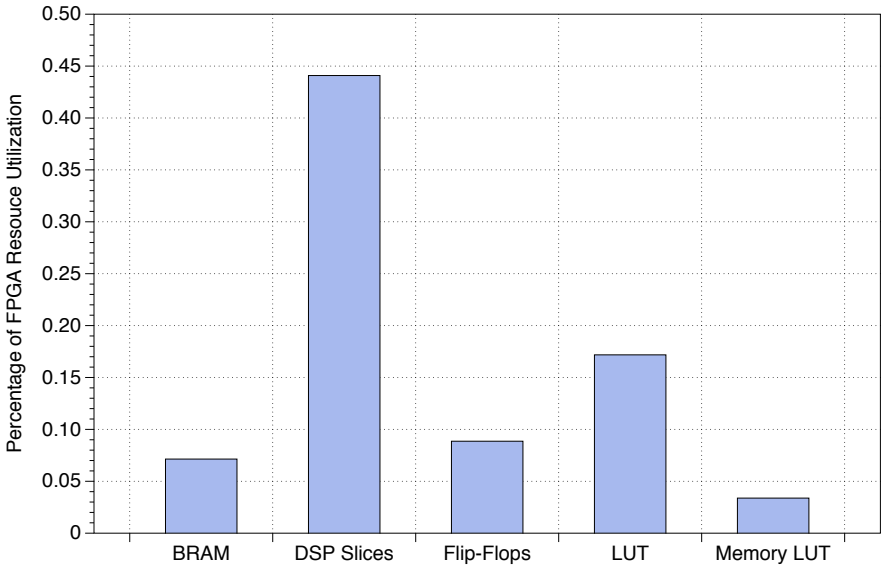


Prediction Error		Execution Time		Speedup	Power		Energy	
Software	Hardware	Software	Hardware		CPU	CPU+FPGA	Software	Hardware
3.9%	3.9%	3.3s	2.8s	1.18X	2.2W	4.19W	7.26J	11.73J

Test 2

Setup

- 16x16 grayscale USPS Dataset
- one convolutional layer:
  - six 5x5 kernels and sub-sampling
- one linear layer:
  - 10 neurons
- Directives:
  - DATAFLOW
  - PIPELINE

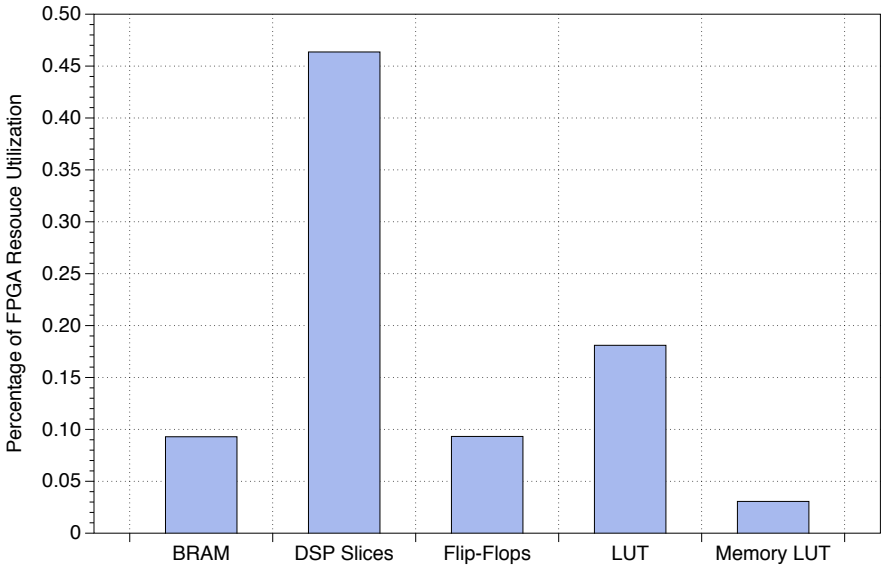


Prediction Error		Execution Time		Speedup	Power		Energy	
Software	Hardware	Software	Hardware		CPU	CPU+FPGA	Software	Hardware
3.9%	3.9%	3.3s	0.53s	6.23X	2.2W	4.21W	7.26J	2.23J

Test 3

Setup

- 16x16 grayscale USPS Dataset
- 1° convolutional layer:
  - six 5x5 kernels and sub-sampling
- 2° convolutional layer:
  - six 5x5 kernels and sub-sampling
- one linear layer:
  - 10 neurons
- Directives:
  - DATAFLOW
  - PIPELINE

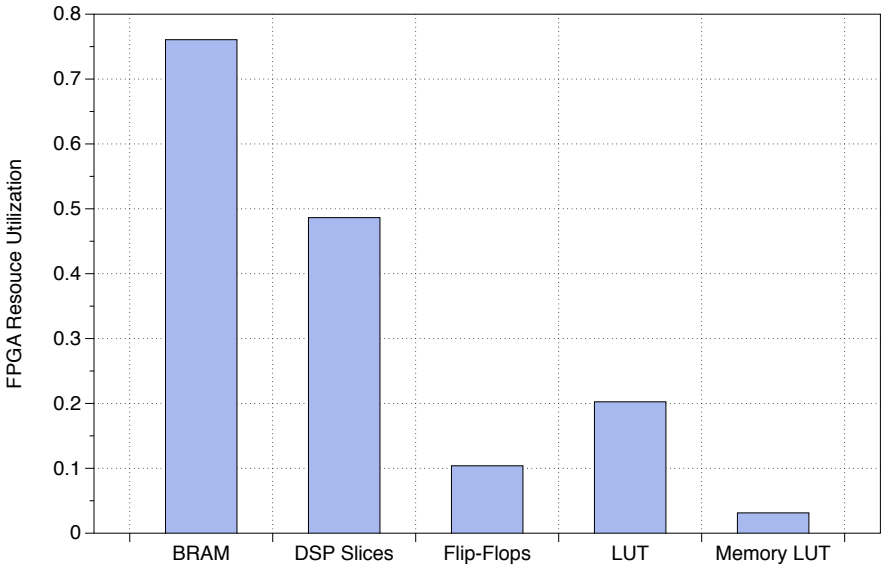


Prediction Error		Execution Time		Speedup	Power		Energy	
Software	Hardware	Software	Hardware		CPU	CPU+FPGA	Software	Hardware
7.1%	7.1%	4.3s	0.48s	9.0X	2.2W	4.24W	9.46J	2.04J

Test 4

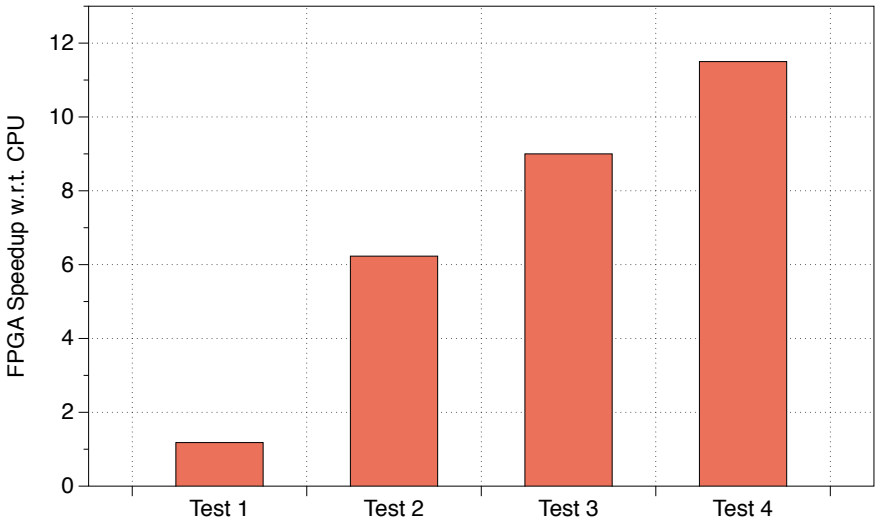
Setup

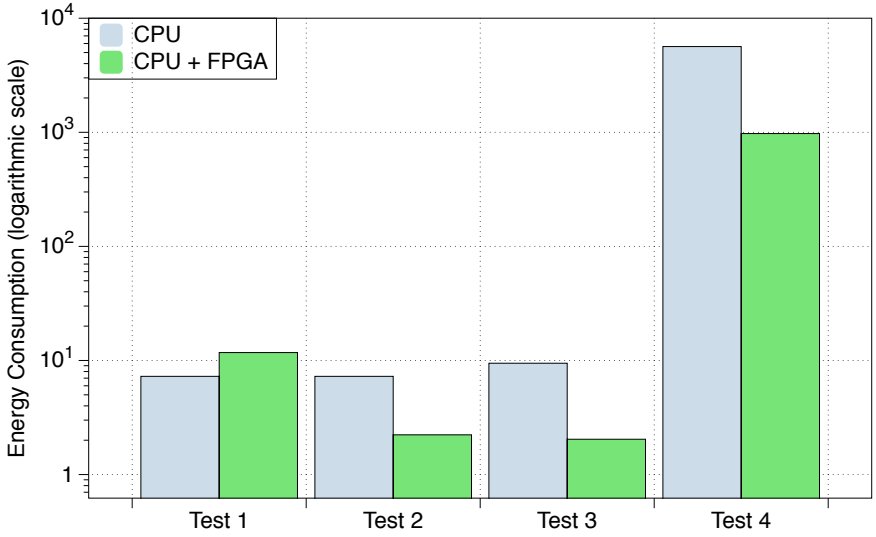
- 32x32 RGB CIFAR-10 Dataset
- 1° convolutional layer:
  - Twelve 5x5 kernels and sub-sampling
- 2° convolutional layer:
  - Thirty-six 5x5 kernels and sub-sampling
- 1° linear layer:
  - 36 neurons
- 2° linear layer:
  - 10 neurons
- Directives:
  - DATAFLOW
  - PIPELINE



Prediction Error		Execution Time		Speedup	Power		Energy	
Software	Hardware	Software	Hardware		CPU	CPU+FPGA	Software	Hardware
89.4%	89.4%	2565s	223s	11.5X	2.2W	4.37W	5643J	975J

Experimental Results Summary





Conclusions & Future Works

- We presented a preliminary framework for the automation of HLS of CNNs

- We plan to:
  - Reduce FPGA resource consumption
  - Expand the framework to support other platforms
  - Add more CNN configuration options

- The **new** version of the framework will be online at:
  http://cnn2fpga.hosting.necst.it

- Follow us on Facebook:
  CNNECST-Convolutional Neural Network

Thank You for the Attention