Motivation and Problem Definitions
• Convolutional Neural Network (CNN) achieves the state-of-art performance in image recognition, natural language processing and bioinformatics.
• High computation complexity in both inference and training, which needs specific hardware to accelerate.
• FPGA plays an important role due to its re-configurability and massive parallelism.
• Automatic Generation Tool: How to design a system that apply to a wide range of CNN architectures remain challenging.

Approach
• Analyze the basic computational structure of CNN and identify fast algorithm for acceleration.
• Build highly-optimized and parameterized hardware accelerator module for specific algorithm including Matrix operations, Fast Fourier Transform, Winograd algorithm, etc.
• Given CNN architecture, utilize the hardware modules from library to build a full system.
• Explore the system level optimization including memory sub-system, resources distribution and scheduling algorithm.
• Prototype on specific FPGA device.

Algorithm and Hardware
• Frequency Domain Acceleration using FFT and Overlap-and-Add.

CPU + FPGA Mapping
• Increase on-chip data reuse to reduce FPGA-memory bandwidth requirement.
• Exploit task parallelism if there are available resources and bandwidth.

System Level Optimization

Experiments and Results

Asymptotic Analysis
• Space convolution: \(O(N^2F^2)\)
• OaA convolution: \(O(N^2\log F)\)

GFLOP Reduction
• CaffeNet (2012): 48.82%
• VGG16 (2014): 54.10%
• GoogLeNet (2014): 39.43%

Experimental Setup
- Intel QuickAssist FPGA Platform (Xeon + Altera Stratix V)
- Shared memory between CPU and FPGA
- 6 GBs FPGA-memory bandwidth + 6.25 MB on-chip BRAM
- Experimented CNN architectures: AlexNet, VGG16, GoogLeNet

Table 7: Performance Comparison with the State-of-Art CNN Implementations on FPGA

Table 8: Execution Time for VGG16 and AlexNet

Discussions and Future Work

Energy Efficiency. A CPU-FPGA based design will consume more power than FPGA-only based design. However, the CPU adds more flexibility to the design. Moreover, since most of the computations are inside convolutional layer, the CPU simply performs adding and data rearrangement and the energy consumption will not scale up quickly if we increase the CNN size.

Automatic Code Generation. Our framework provides complete solution to accelerate CNN on FPGA including inter-layer data rearranging. Modern CNN’s convolutional layers are mainly consist of small kernels. Thus, by zero-padding various kernel sizes to fit a chosen FFT size, and using FPGA to accelerate it by exploiting massive parallelism, we can achieve considerable performance improvement for various CNN models. We can use our framework to develop an automatic code generation tool so high-level users can specify CNN models and generate the design automatically.

Fixed Point vs. Floating Point. Many previous approaches use fixed point instead of floating point for computations. The advantage is less resources consumption and the power efficiency. However, it penalizes the classification accuracy. Some approaches claim that the classification accuracy is tolerable according to experiments. However, it is hard to generalize to an arbitrary CNN model.