Using Fermat Number Transform to Accelerate Convolutional Neural Network

Weihong Xu[1,2], Xiaohu You[2], Chuan Zhang[1,2]

[1]Lab of Efficient Architectures for Digital-communication and Signal-processing (LEADS)
[2]National Mobile Communications Research Laboratory, Southeast University, Nanjing, China

wh.xu@seu.edu.cn

October 28, 2017

Outline

1. Introduction on CNNs
2. Motivation and Related Work
3. Proposed FNT Acceleration
4. Analysis and Implementation Results
5. Conclusion

Outline

1. Introduction on CNNs
2. Motivation and Related Work
3. Proposed FNT Acceleration
4. Analysis and Implementation Results
5. Conclusion

Convolutional Neural Networks

- Deep neural networks with convolution
- Processing 1D data (time series), 2D data (music) and 3D data (image)
- Strong capacities to extract features

Convolutional Neural Networks

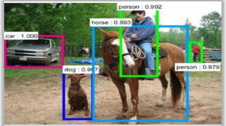
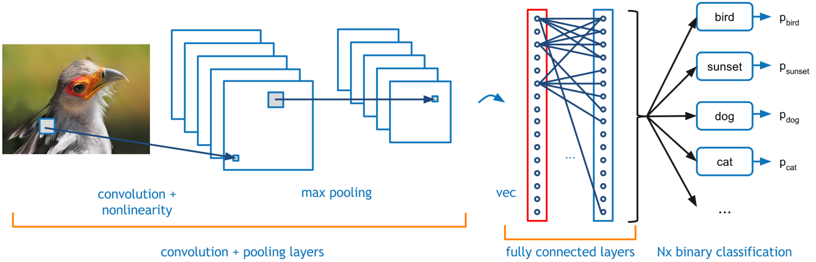


Image Segmentation

Game

....

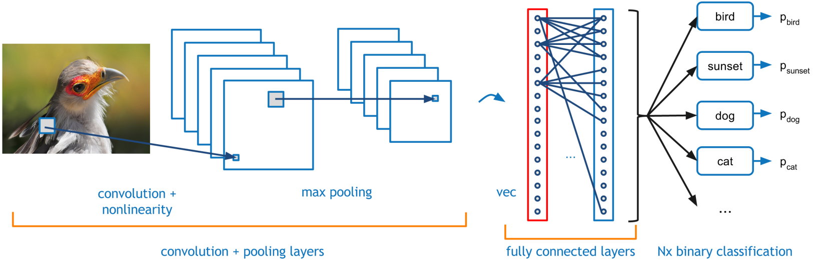
Convolutional Neural Networks



- **Convolution**, **Pooling**, **Non-linear** and **Fully-connected** layers stacked in stages

[1] https://bigsnarf.wordpress.com

Convolutional Neural Networks

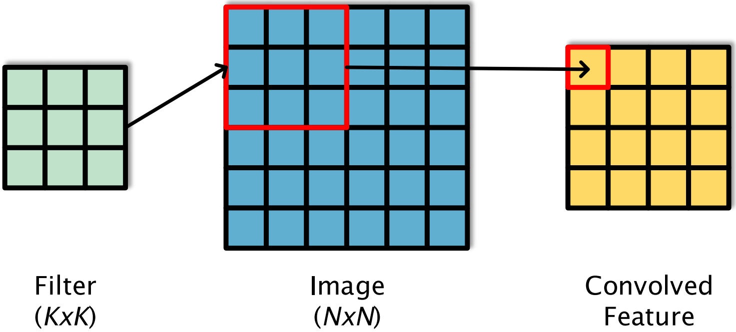


- **Convolution**: feature extraction by convolving various filters over input image
- **Fully-connected**: linear transform over input features
- **Pooling and Non-linear**: perform down sampling and non-linear function

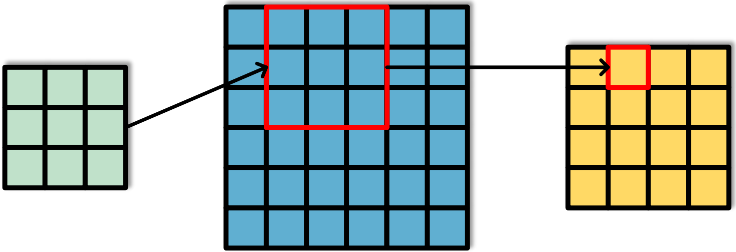
Major Challenges

- **Computation-intensive**: convolution takes up over 95% of ovarall complexity
- **Memory-intensive**: FC layers contribute 90% parameters

Convolution in Single Channel



Convolution in Single Channel



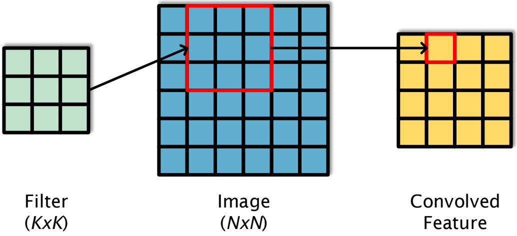
Filter
(KxK)

Image
(NxN)

Convolved
Feature

Outline

Prohibitive Complexity



- Computation Complexity: $\mathcal{O}(N^2K^2)$
- Multiplier is expensive!

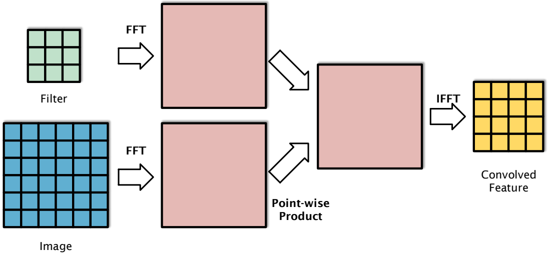
Related Work

1. N. Vasilache et al., "Fast Convolutional Nets With fbfft: A GPU Performance Evaluation," in *ICLR*, 2015
2. C. Zhang and V. Prasanna, "Frequency domain acceleration of convolutional neural networks on CPU-FPGA shared memory system," in *ACM/SIGDA ISFPGA*, 2017
3. T. Abtahi et al., "Accelerating convolutional neural network with fft on tiny cores," in *IEEE ISCAS*, 2017

Convolution based on FFT

Convolution property:

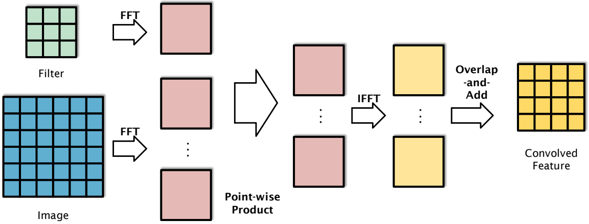
$$y(n) = x * h = \mathcal{F}^{-1}\left\{\mathcal{F}(x(n)). * \mathcal{F}(h(n))\right\}$$



- Computation Complexity: $\mathcal{O}(N^2 \log N)$
- Effective only when $\log N \geq K^2$
- Intensive intermediate memory requirement

Convolution based on Overlap-and-Add FFT

$$y(n) = \sum_k \mathcal{F}^{-1}\left\{\mathcal{F}(x(n-kL)).*\mathcal{F}(h(n))\right\}$$



- Computation Complexity: $\mathcal{O}(N^2 \log K)$
- Effective for small filters
- Negligible intermediate memory build up
- Additional modules for overlap and add

OaA-FFT is the best?

Problems to Solve

- Complex number operation of FFT
- Additional registers for twiddle factors
- Introducing roundoff noise during FFT/IFFT under low-bit fixed point
- Frequency information is not what we need

Outline

1. Introduction on CNNs
2. Motivation and Related Work
3. Proposed FNT Acceleration
4. Analysis and Implementation Results
5. Conclusion

Number Theoretic Transform

- Number Theoretic Transform:

$$X(k) = \mathcal{F}(x(n)) = \sum_{n=0}^{N-1} \left\langle x(n)\alpha^{nk} \right\rangle_m$$

$\langle a \rangle_b$: modular arithmetic

- To Dicrete Fourier Transform:

$$X(k) = \mathcal{F}(x(n)) = \sum_{n=0}^{N-1} x(n) W_N^{nk}$$

Replace α with twiddle factor W_N to obtain DFT

Fermat Number Transform

- Replace α with 2:

$$X(k) = \mathcal{F}(x(n)) = \sum_{n=0}^{N-1} \left\langle x(n) 2^{nk} \right\rangle_{F_t}$$

F_t: the t-th Fermat number

- Constraints:

$$\begin{cases} F_t = 2^b + 1 \\ b = 2^t, \text{input bits} \\ N = 2^{t+1}, \text{sequence length.} \end{cases}$$

- Easy to satisfy with $F_4 = 2^{16} + 1, N = 32, b = 16$.

Overlap-and-Add FNT

- Similar to OaA FFT:

$$y(n) = \sum_k \mathcal{F}^{-1} \left\{ \mathcal{F}(x(n-kL)). * \mathcal{F}(h(n)) \right\},$$

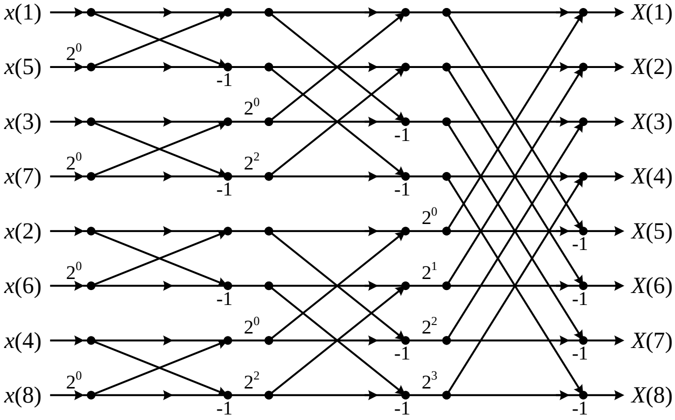
where L is an arbitrary segment length and $L \leq N - K + 1$.

- For maximum efficiency, select $L = N - K + 1$

- Run-time configurable for various input sizes by adjusting L and k

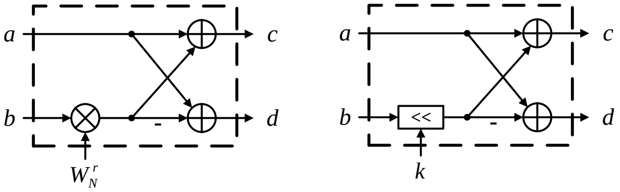
Factor Graph

- $t = 2$, $N = 2^{t+1} = 8$, $b = 2^t = 4$, $F_t = 2^b + 1 = 17$



Hardware Architecture

- No need for extra registers
- Area and power efficiency



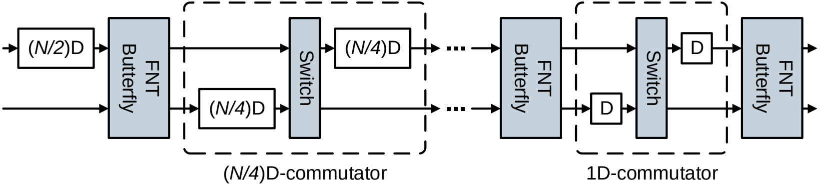
Butterfly modules of FFT and FNT

Complex multiplier $\longrightarrow$ Shift register

Complex adder $\longrightarrow$ Real adder and modular

Pipelined Architecture

- Increased efficiency of basic units
- Reduced area and IO bandwidth

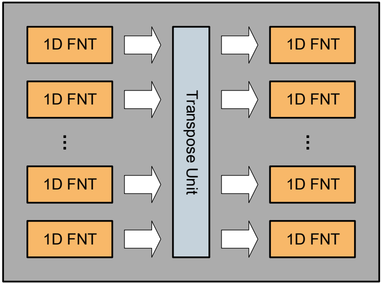


$$\text{Efficiency: } \frac{1}{\log_2 N} \longrightarrow {\color{red}50\%}$$

$$\text{Butterfly : } \frac{N}{2}\log_2 N \longrightarrow {\color{red}\log_2 N}$$

2D FNT based on 1D FNT

- First, perform 1D FNT on each row of the input image
- Then, perform 1D FNT on each column to the intermediate results



2D FNT Kernel

Data Processing for CNNs

- Range of FNT filtering data: $[0, 2^b]$
- Image and weights of CNNs are basically real numbers
- Transform input data with range $[-2^{b-1}, 2^{b-1}]$ into $[0, 2^b]$:

$$x^{'} = \begin{cases} x, & \text{if } 0 \le x \le 2^{b-1}, \\ x + F_t, & \text{if } -2^{b-1} \le x < 0. \end{cases}$$

- **Example:** $F_2 = 2^4 + 1 = 17$ and $2^{b-1} = 8$

$$x = [1, -2, 3, -4] \longrightarrow x^{'} = [1, 15, 3, 13]$$

Data Processing for CNNs

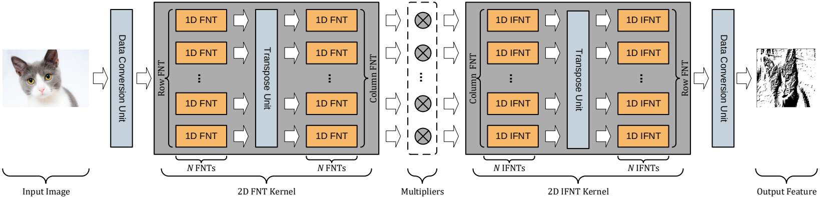
- Transform output data back to real domain:

$$y = \begin{cases} y^{'}, & \text{if } 0 \le y^{'} \le 2^{b-1}, \\ y^{'} - F_t, & \text{if } 2^{b-1} \le y^{'} \le 2^{b}. \end{cases}$$

- **Example:** $h = [1, 1, 1, 1]$, $F_2 = 2^4 + 1 = 17$ and $2^{b-1} = 8$

$$\begin{aligned} y^{'} &= x * h = [1, 16, 2, 15, 14, 16, 13, 0] \\ \longrightarrow y &= [1, -1, 2, -2, -3, -1, -4, 0] \end{aligned}$$

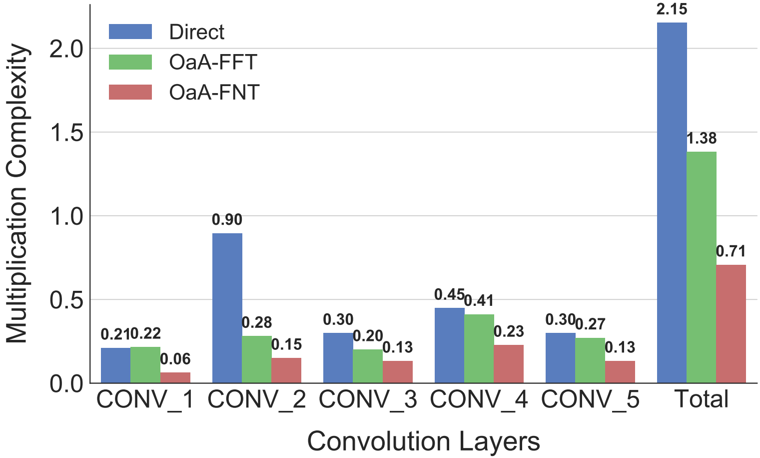
Overall Architecture



- Advantages
  - High data parallelism
  - Further reduction on arithmetic complexity
  - Suitable for various filter and image sizes
  - Round-off noise is zero

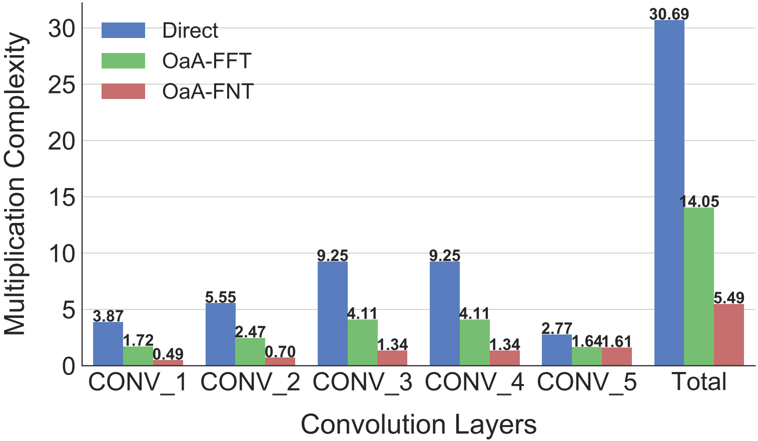
Outline

Complexity Analysis on AlexNet



[1][Zhang and Prasanna, *ISFPGA*, 2017]

Complexity Analysis on VGGNet



[1] [Zhang and Prasanna, *ISFPGA*, 2017]

Experimental Setup

- Xilinx Virtex-7 XC7VX485t FPGA Platform
- Intel i7-7700k CPU
- Quantize pre-trained VGGNet16 from Pytorch into 16 fixed point
- Pre-calculate FNT of weights and load to FPGA
- FNT parameters: $F_4 = 2^{16} + 1, N = 32, b = 16$
- We implement the accelerator for convolutional layer

Implementation Results

Design	Zhang et al. 2015	Qiu et al. 2016	This work
Platform	Virtex-7 VX485t	Zynq XC7Z045	Virtex-7 VX485t
Clock(MHz)	100	150	150
CNN Model	AlexNet	VGG16-SVD	VGG16
Quantization	32-bit float	16-bit fixed	16-bit fixed
LUT	186251	182616	188928
FF	205704	127653	142592
DSP	2240	780	**256**
BRAM	1024	486	512
CONV Throughput (GOP/s)	61.62	187.80	**264.60**
Resource Efficiency (GOP/s/DSP)	0.028	0.241	**1.033**

Outline

1. Introduction on CNNs
2. Motivation and Related Work
3. Proposed FNT Acceleration
4. Analysis and Implementation Results
5. Conclusion

Conclusion

- Efficient OaA-FNT-based 2D convolver to accelerate CNNs
- Data processing of CNNs to fit into FNT
- $\mathbf{1.41}\times$ convolution throughput with $\mathbf{3.05}\times$ less DSPs compared to state-of-the-art design on VGGNet.
- Future Work
  - Use data with lower-bit quantization
  - Build reconfigurable design for variable length

Thanks for Your Attention!

Q & A