Towards Acceleration of Deep Convolutional Neural Networks using Stochastic Computing

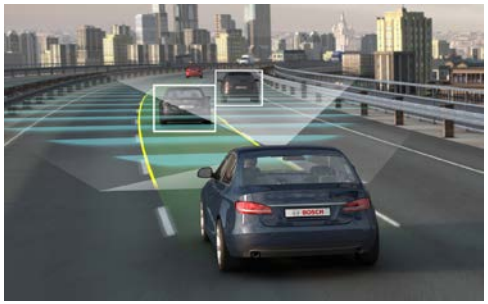
Ji Li	University of Southern California
Ao Ren	Syracuse University
Zhe Li	Syracuse University
Caiwen Ding	Syracuse University
Bo Yuan	City University of New York
Qinru Qiu	Syracuse University
Yanzhi Wang	Syracuse University

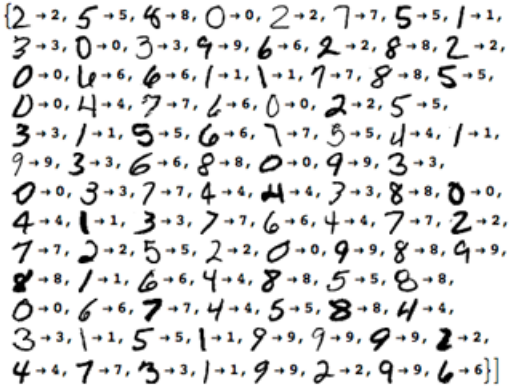
Outline

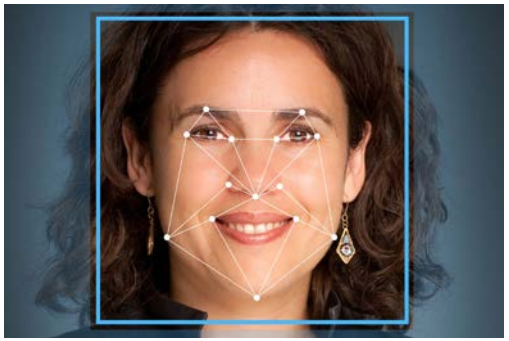
- Introduction
- Background
  - Deep Convolutional Neural Network (DCNN)
  - Stochastic Computing (SC)
- Hardware Design and Optimization
  - Basic Function Blocks
  - Feature Extraction Block Design & Optimization
  - Layer-wise Design
- Reconfiguration and Scalability
- Results
- Conclusion

1. Introduction

- Deep Learning, as an important branch of machine learning and neural network, is playing an increasingly important role in a number of fields like image classification, computer vision, natural language processing, etc.







1. Introduction

- With recent advancing of wearable devices and Internet of Things (IoTs), it becomes very attractive to implement the deep learning systems onto embedded and wearable devices.

1. Introduction

➢ Challenges

- Presently, executing the software-based deep learning systems requires high-performance server clusters in practice, restricting their widespread deployment on the personal and mobile devices.

- In order to overcome this issue, considerable research efforts have been conducted in the context of developing highly-parallel and specific hardware designs, utilizing GPGPUs, FPGAs, and ASICs.

1. Introduction

➢ Stochastic Computing

- It’s a data representation and processing technique, which uses a bit-stream to represent a number within [-1, 1] by counting the number of ones in the bit-stream.
- It has high potential for implementing deep neural network with high scalability and ultra-low hardware footprint.

1. Introduction

➢ Our work:

- Our main focus is on deep convolutional neural network.
- We investigate and explore various implementations of basic functions required by DCNNs.
- We propose the implementations of feature extraction blocks by selectively compose the basic function blocks and jointly optimize them.
- We construct a LeNet5 in stochastic computing.

2. Background

2.1 Deep Convolutional Neural Network (DCNN)

2.2 Stochastic Computing (SC)

2.1 Deep Convolutional Neural Network (DCNN)

2.1.1 DCNN Overview

2.1.2 Basic Operations

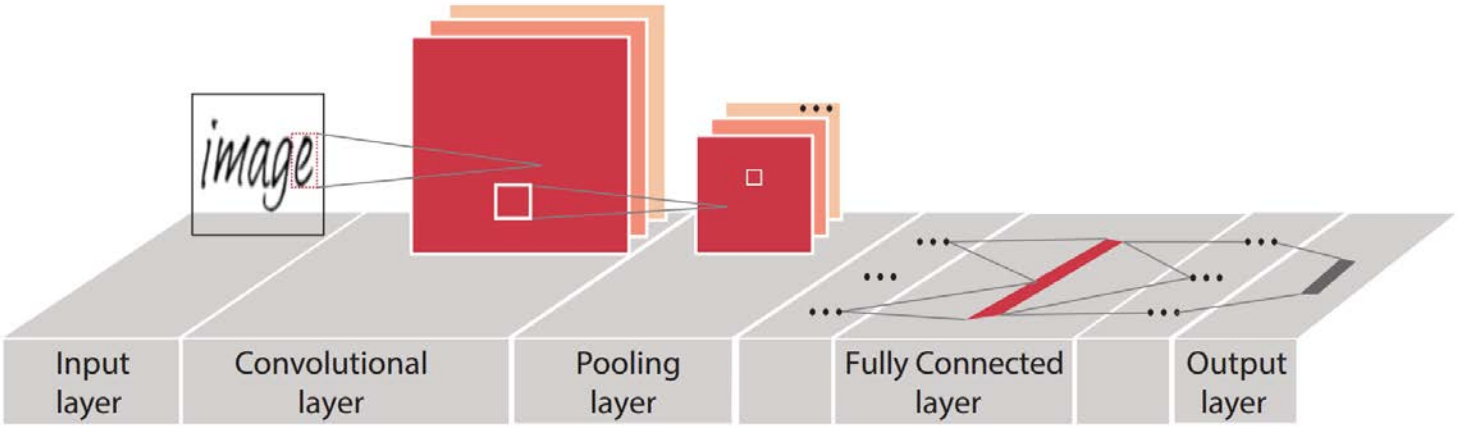
-Convolution

-Pooling

-Activation Function

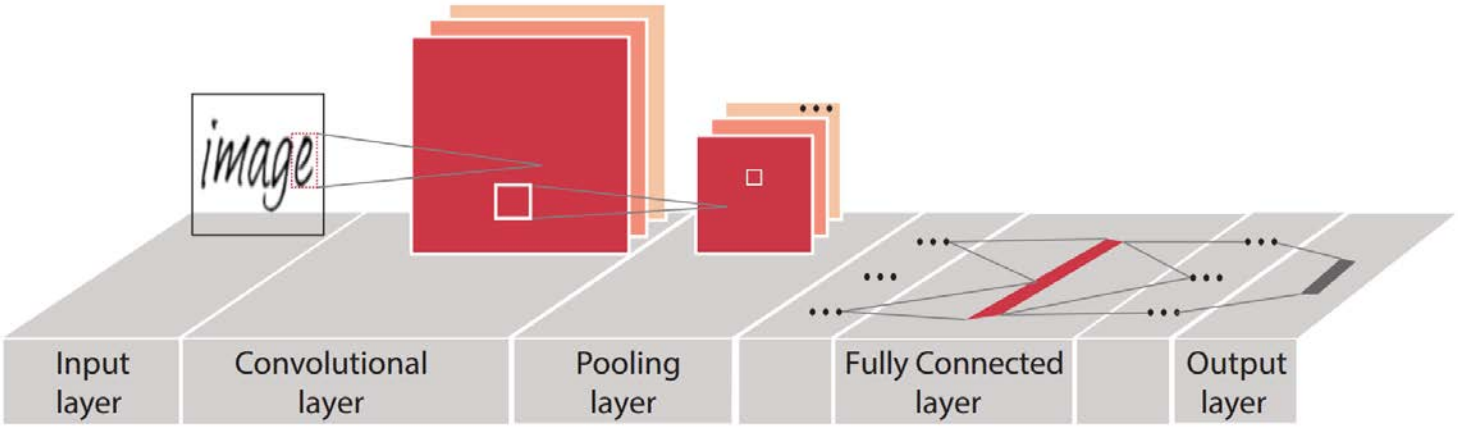
2.1.1 DCNN Overview

- Deep convolutional neural networks are biologically inspired variants of multilayer perceptrons (MLPs) by mimicking the animal visual mechanism.



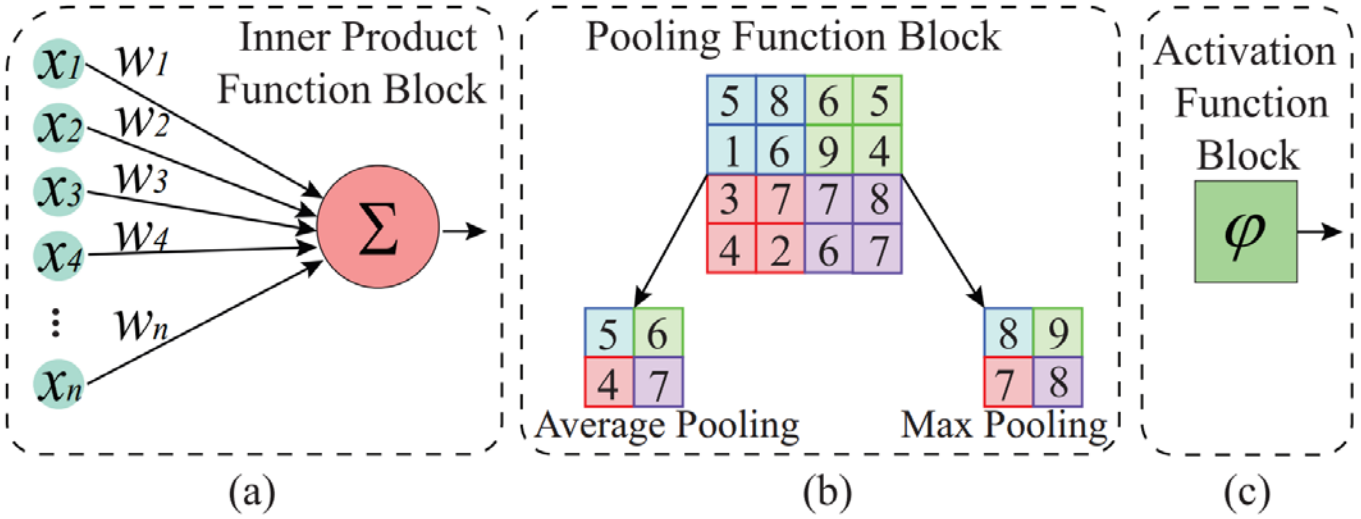
2.1.1 DCNN Overview

- A DCNN is in the simplest case a stack of three types of layers: Convolutional Layer, Pooling Layer, and Fully Connected Layer.



2.1.2 DCNN Basic Operations

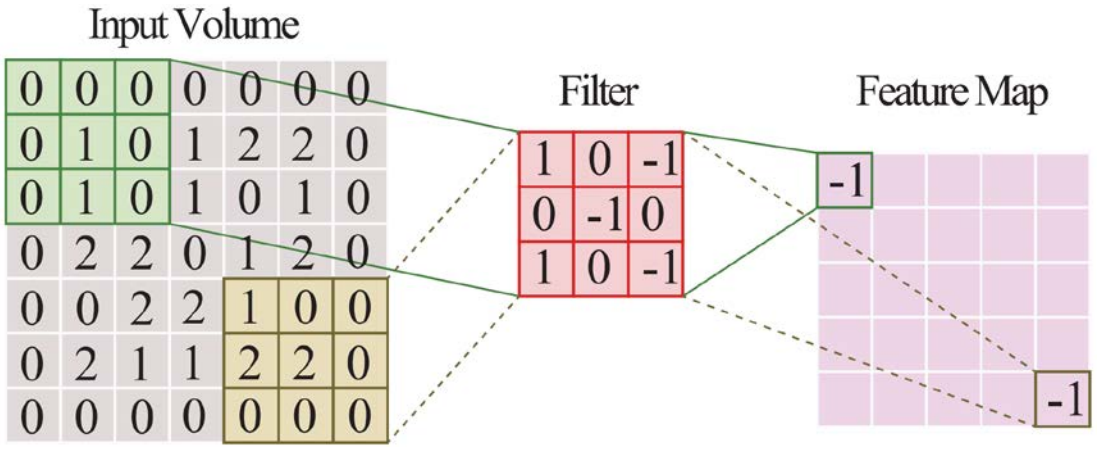
- The main operations in DCNN are: convolution, pooling, and activation.



2.1.2 DCNN Basic Operations

- Convolution

  -The Convolutional layer is the core building block of DCNN, and the main operation is the convolution that calculates the dot-product of receptive fields and a set of learnable filters (or kernels).



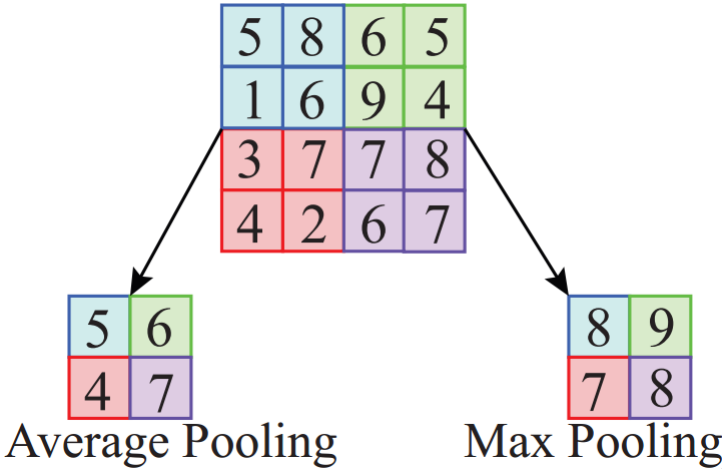
2.1.2 DCNN Basic Operations

- Pooling

  -Nonlinear down samplings.

  -To reduce the dimension of data.

  -Max pooling and average pooling.

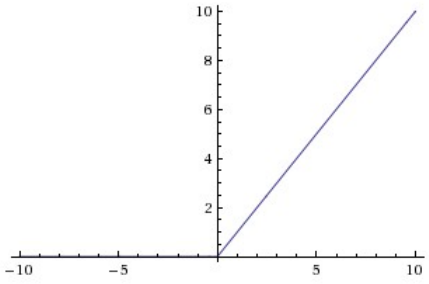


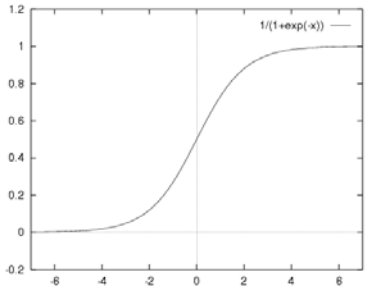
2.1.2 DCNN Basic Operations

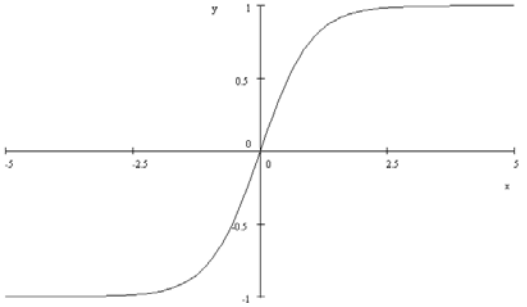
- Activation

  -Nonlinear transformation.

  -Rectified Linear Unit (ReLU) $f(x) = \max(0, x)$; Sigmoid function $f(x) = (1+e^{-x})-1$; and hyperbolic tangent (tanh) function $f(x) = 2/(1+e^{-2x})-1$.







2.2 Stochastic Computing (SC)

2.2.1 Data Representation

2.2.2 Arithmetic Calculations

2.2.1 Data Representation

- Stochastic computing represents a data using a bit-stream by counting the number of ones in the stream.

- Unipolar

  -Represent data in the range of [0, 1].

  -Example: 00100101 represents P(x=1)=3/8=0.375

- Bipolar

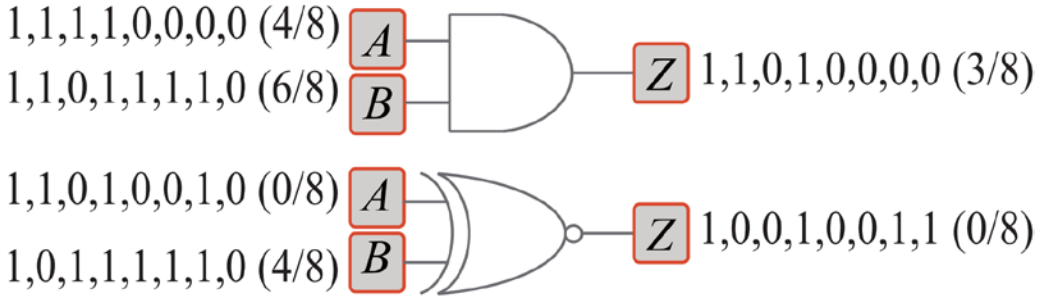
  -Represent data in the range of [-1, 1].

  -P(X=1)=(x+1)/2

  -Example: 00100101 represents x = (3/8) * 2 -1= - 0.25

2.2.2 Arithmetic Calculations

- Multiplication.

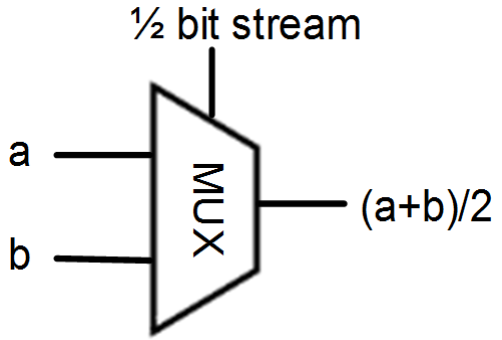


- Addition.

$c=2P(C=1)-1$

$=2(1/2(P(A=1)+1/2P(B=1))-1$

$=1/2(2P(A=1)-1)+(2P(B=1)-1))=1/2(a+b)$



3. Hardware Design and Optimization

3.1 Basic Function Blocks

3.2 Feature Extraction Block Design & Optimization

3.1 Basic Function Blocks

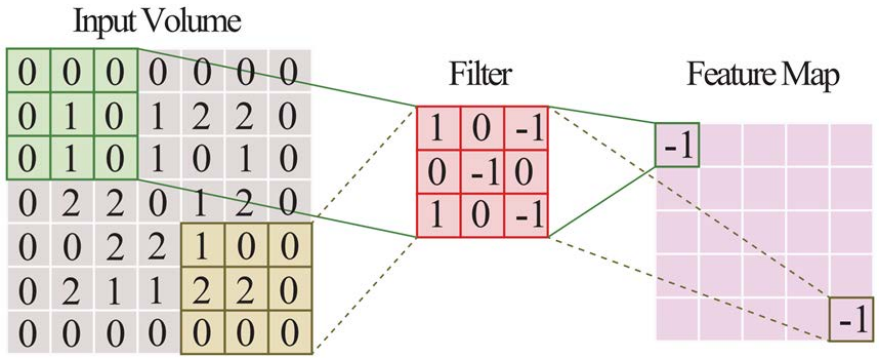
3.1.1 Convolution (Inner Product) Block Design

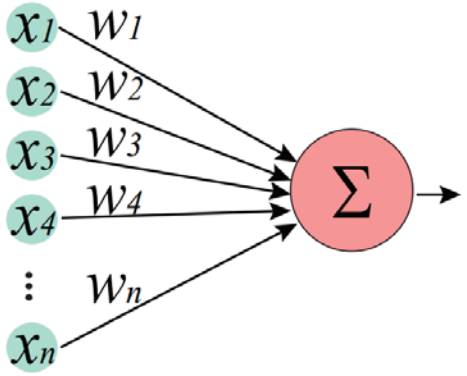
3.1.2 Pooling Block Design

3.1.3 Activation Block Design

3.1.1 Convolution (Inner Product) Block Design

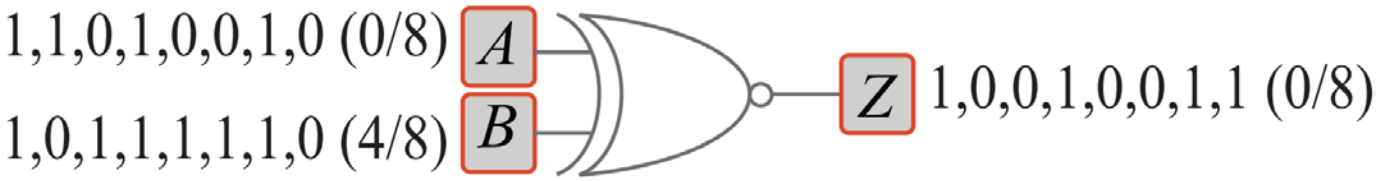
- Convolution (Inner Product) = Multiplications + Addition





3.1.1 Convolution (Inner Product) Block Design

- Multiplication: XNOR



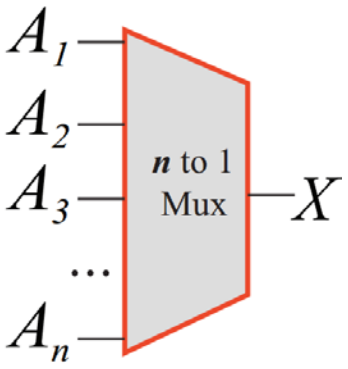
- Addition

-Multiplexer (MUX)-Based

-Approximate Parallel Counter (APC)-Based

3.1.1 Convolution (Inner Product) Block Design

- MUX-Based Adder



- Advantages:

-Low Hardware footprint, low power

-Easy to implement

- Disadvantages:

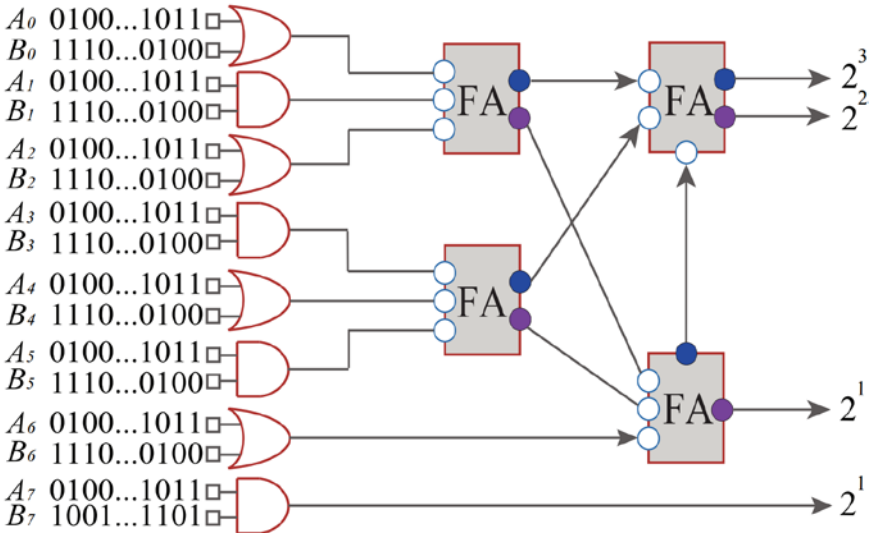
-Relatively low accuracy

-Natural down-scaling

3.1.1 Convolution (Inner Product) Block Design

➢ APC-Based Adder

-Perform an addition by counting the number of ones.



- Advantages:

-Relatively high accuracy

-No down-scaling issue

- Disadvantages:

-Relatively high hardware footprint

Input size	Bit stream length			
	128	256	384	512
16	1.01%	0.87%	0.88%	0.84%
32	0.70%	0.61%	0.58%	0.57%
64	0.49%	0.44%	0.44%	0.42%

3.1.2 Pooling Block Design

➢Average Pooling

-Calculate the average value of bit-stream inputs

- MUX-based inner product block

-Four to one MUX

-Naturally perform average calculation

- APC-based inner product block

-Four to one MUX

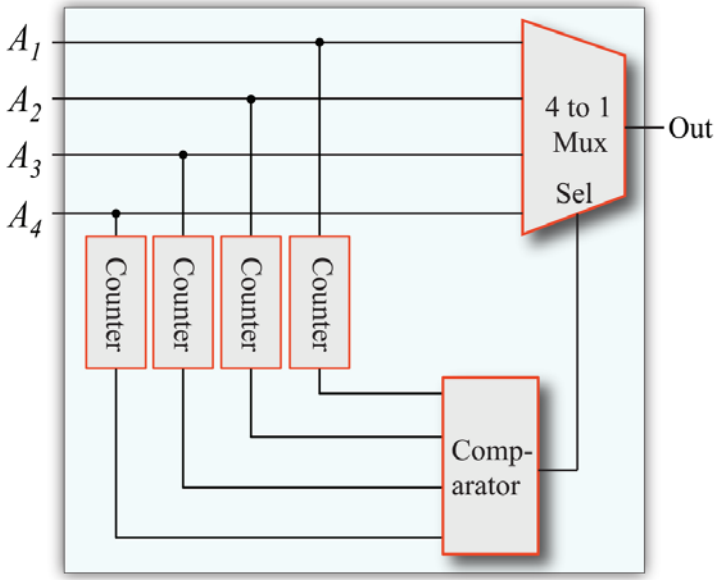
-Or conventional binary component

3.1.2 Pooling Block Design

➢Max Pooling

-Select the bit-stream that represents the maximum value

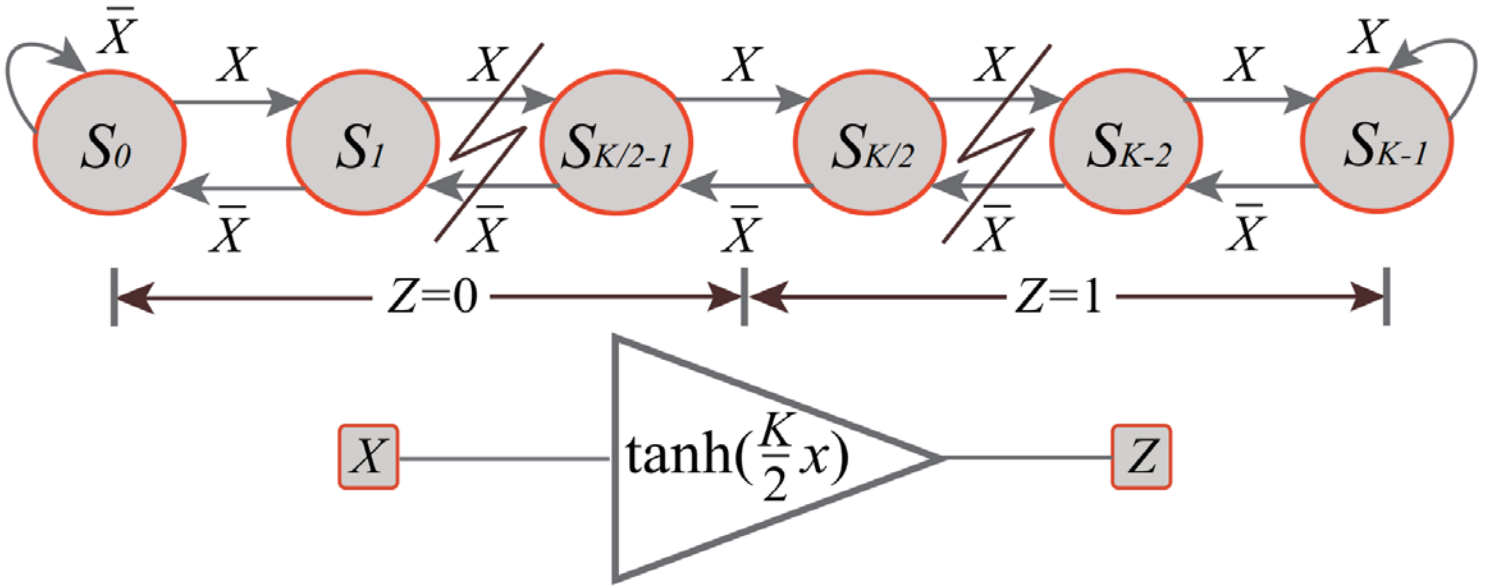
➢ Novel Max Pooling Scheme



3.1.3 Activation Block Design

- Hyperbolic tangent (tanh)

-Easy to implement with finite state machine in SC domain



3.1.3 Activation Block Design

- Binary Hyperbolic Tangent (Btanh)

-Receive binary input and generate stochastic bit-stream.

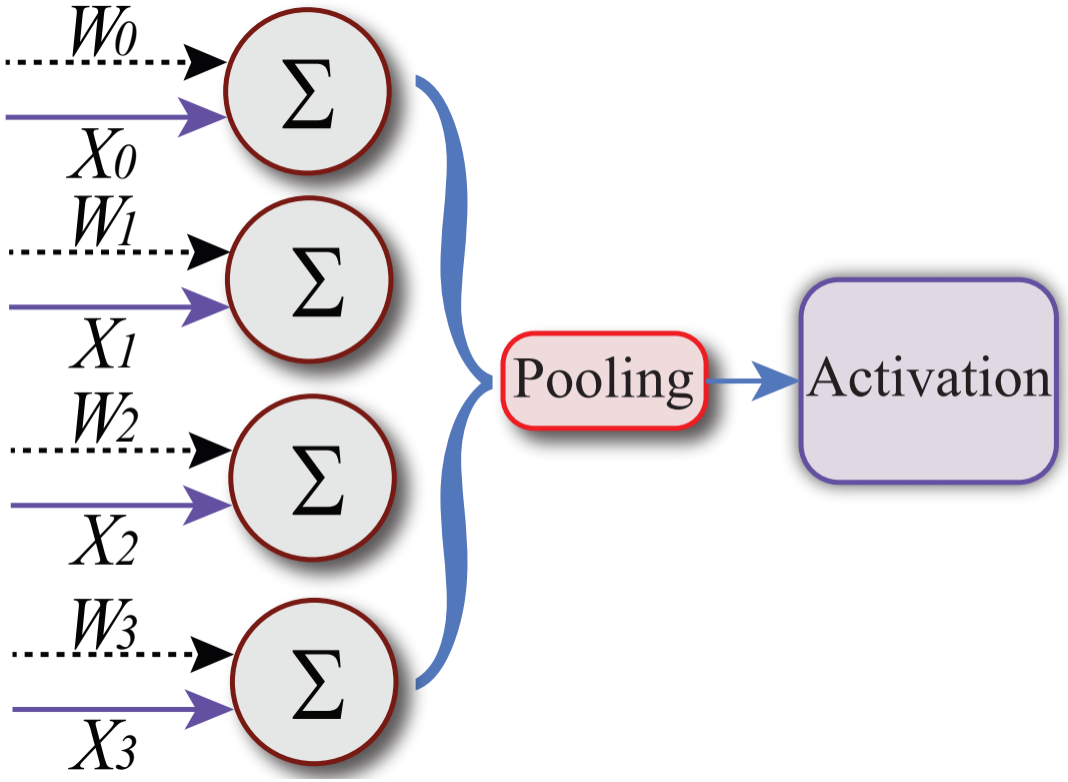
-Connect with APC-based inner product block.

-The same idea with FSM.

-Actually implemented with binary calculation components.

3.2 Feature Extraction Block Design & Optimization

- Inner Product Block-Pooling Block-Activation Block



3.2 Feature Extraction Block Design & Optimization

- Four types of feature extraction blocks (FEBs)

-MUX-Avg-Stanh

-MUX-Max-Stanh

-APC-Avg-Btanh

-APC-Max-Btanh

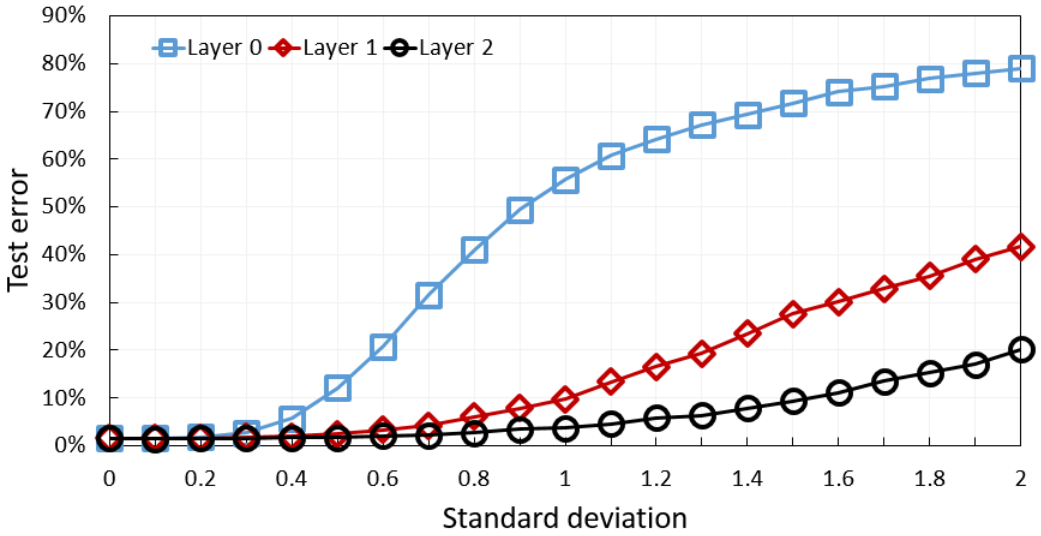
- Jointly Optimization

-Influence factors: input size, bit-stream length, and the inaccuracy introduced by the previous connected block

-A series of joint optimizations are performed on each type of feature extraction block to achieve the optimal performance.

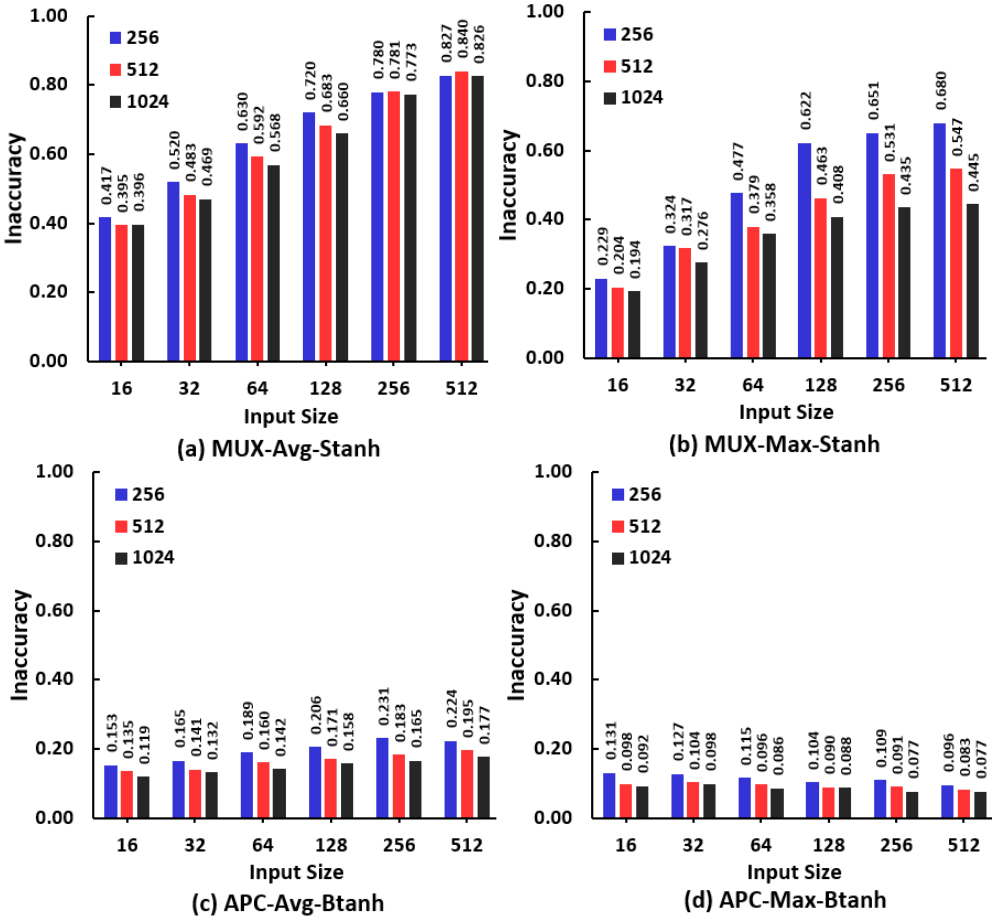
3.3 Layer-wise Design

- Different layer has different sensitivities to errors
- Different layer adopts different feature extraction blocks



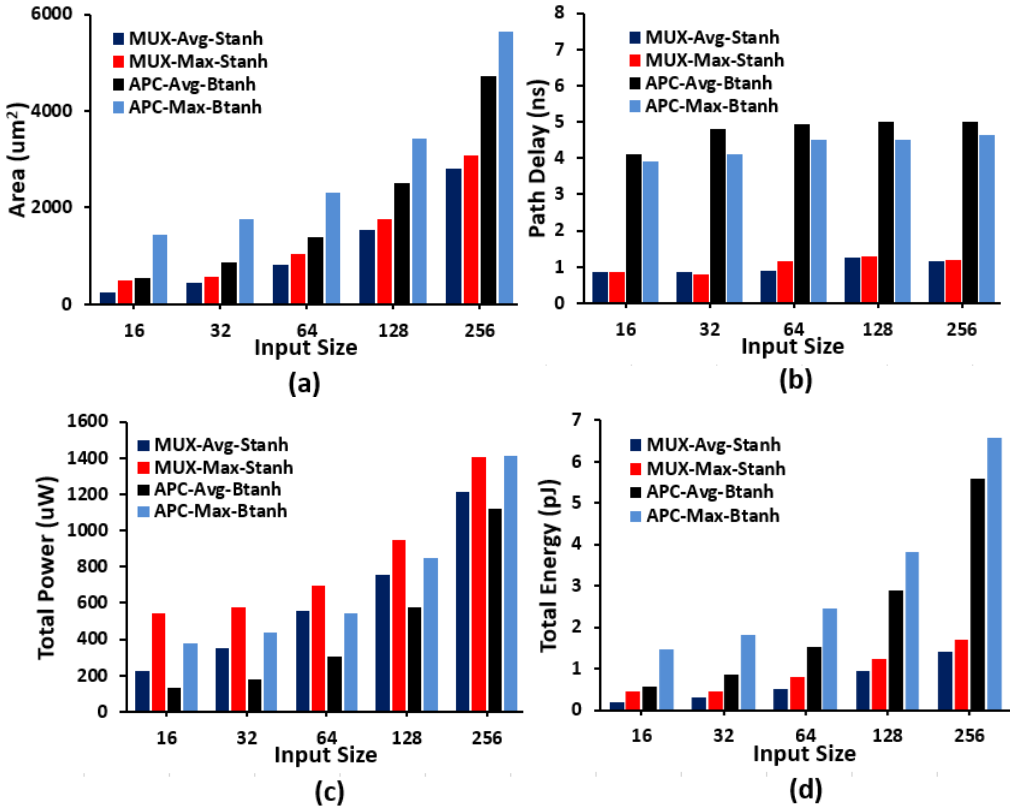
5. Results & Discussions

- Accuracy comparison between different types of FEBs



5. Results & Discussions

- Hardware performance comparison between different types of FEBs



5. Results & Discussions

- Comparison among Various SC-DCNN Designs Implementing LeNet 5

No.	Pooling	Bit Stream	Configuration			Performance				
			Layer 0	Layer 1	Layer 2	Inaccuracy (%)	Area (mm^2)	Power (W)	Delay (ns)	Energy (μJ)
1	Max	1024	MUX	MUX	APC	2.64	19.1	1.74	5120	8.9
2			MUX	APC	APC	2.23	22.9	2.13	5120	10.9
3		512	APC	MUX	APC	1.91	32.7	3.14	2560	8.0
4			APC	APC	APC	1.68	36.4	3.53	2560	9.0
5		256	APC	MUX	APC	2.13	32.7	3.14	1280	4.0
6			APC	APC	APC	1.74	36.4	3.53	1280	4.5
7	Average	1024	MUX	APC	APC	3.06	17.0	1.53	5120	7.8
8			APC	APC	APC	2.58	22.1	2.14	5120	11.0
9		512	MUX	APC	APC	3.16	17.0	1.53	2560	3.9
10			APC	APC	APC	2.65	22.1	2.14	2560	5.5
11		256	MUX	APC	APC	3.36	17.0	1.53	1280	2.0
12			APC	APC	APC	2.76	22.1	2.14	1280	2.7

5. Results & Discussions

- Comparison among with existing hardware platforms

Platform	Dataset	Network Type	Year	Platform Type	Area (mm^2)	Power (W)	Accuracy (%)	Throughput (Images/s)	Area Efficiency (Images/s/mm^2)	Energy Efficiency (Images/J)
SC-DCNN (No.6)	MNIST	CNN	2016	ASIC	36.4	3.53	98.26	781250	21439	221287
SC-DCNN (No.11)			2016	ASIC	17.0	1.53	96.64	781250	45946	510734
2×Intel Xeon W5580			2009	CPU	263	156	98.46	656	2.5	4.2
Nvidia Tesla C2075			2011	GPU	520	202.5	98.46	2333	4.5	3.2
Minitaur		ANN[1]	2014	FPGA	N/A	≤1.5	92.00	4880	N/A	≥3253
SpiNNaker		DBN[2]	2015	ARM	N/A	0.3	95.00	50	N/A	166.7
TrueNorth		SNN[3]	2015	ASIC	430	0.18	99.42	1000	2.3	9259
DaDianNao	ImageNet	CNN	2014	ASIC	67.7	15.97	N/A	147938	2185	9263
EIE-64PE		CNN layer	2016	ASIC	40.8	0.59	N/A	81967	2009	138927

Thank You !