UNIVERSITY OF WASHINGTON
ELECTRICAL ENGINEERING

EEP 590 Spring 2022

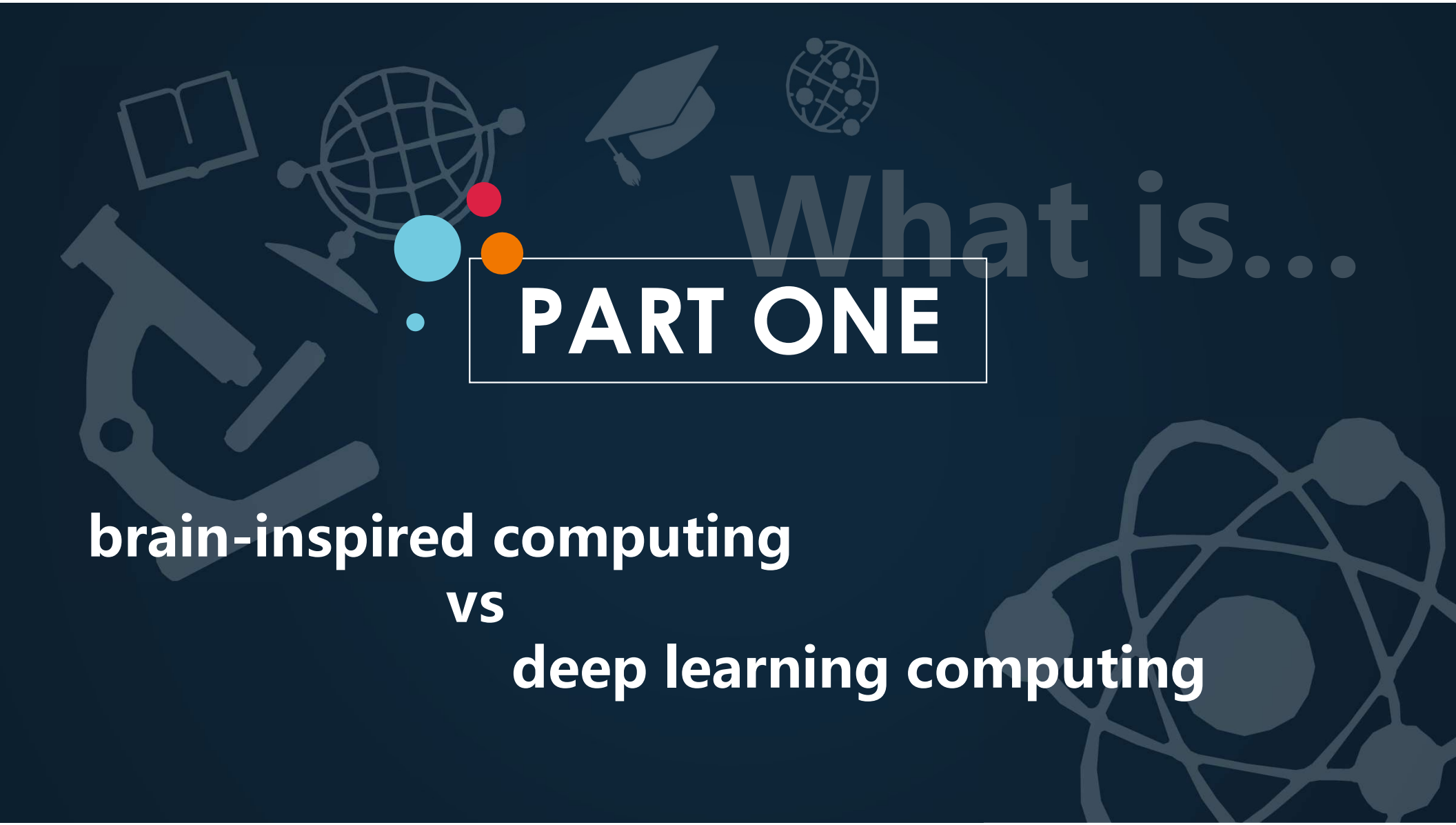
Topic 1: From Deep Learning Algorithms to Embedded Device Implementation: An Overview

Richard Shi cjshi@uw.edu

UNIVERSITY of WASHINGTON

Organization

1. Deep learning algorithms & implementations
2. Applications of deep learning
3. Issues of deep learning accelerator design
4. Key aspects of deep learning hardware implementation


What is...
PART ONE
brain-inspired computing
vs
deep learning computing

PART ONE "What is"

Neuromorphic

Neuromorphology Brain-inspired Computing

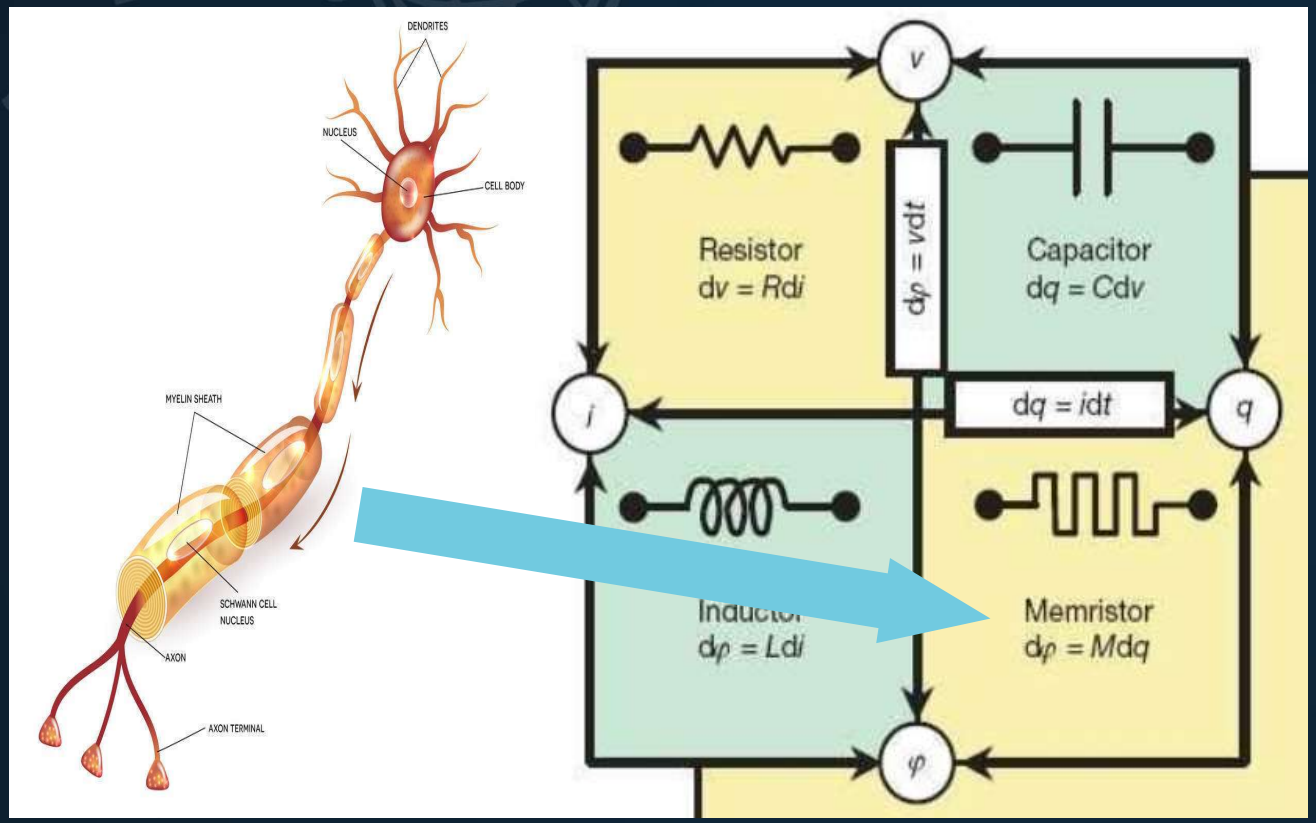
Use spiking neural networks
memristor as the core device
Event-driven
Not perfect training methods
Representative works:
IBM TrueNorth

Deep learning

Deep Learning Brain-inspired Computing

Use deep neural networks
Two classes: CNN & RNN
Systematic Training methods
Widely use in various areas
Representative works:
DeepMind AlphaGo

PART ONE "What is"



Neuromorphology Brain-inspired Computing

- Brain-inspired computing by estimating neurons with integrated circuits
- Implemented CMOS Circuits
  > Low efficiency
- The memristor can well simulate the action of the neuron
  > High cost

PART ONE "What is"

Neuromorphic Computing

Source: P. Merolla *et al*. Science, 2014

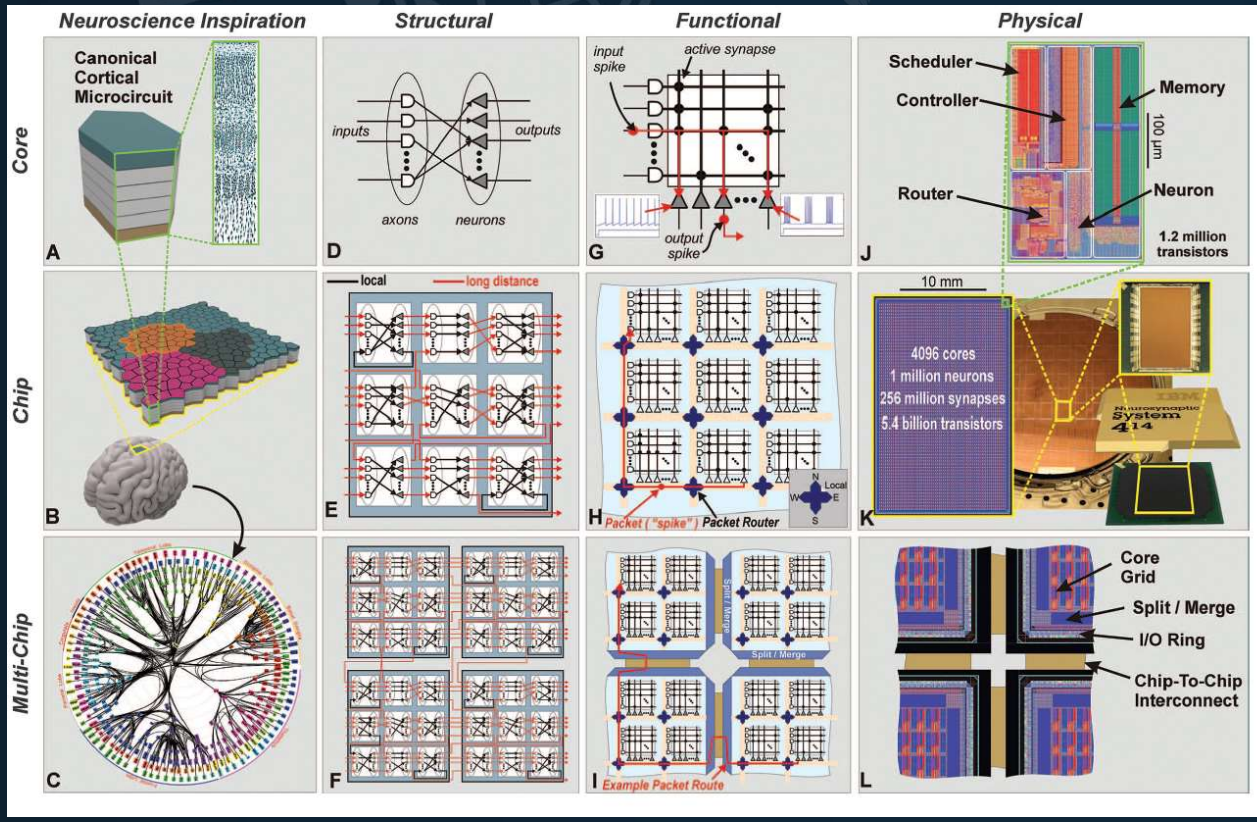
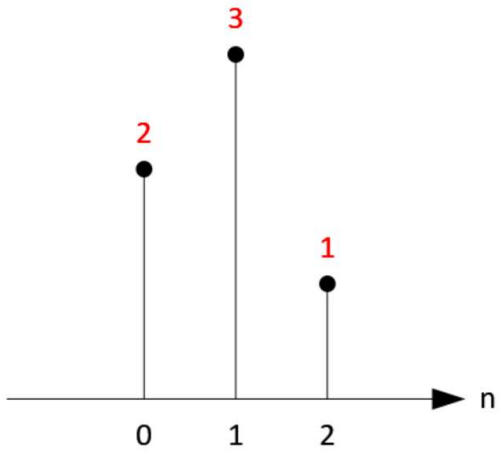


Fig. 3. Real-time multi-object recognition on TrueNorth. (**A**) The Neovision2 Tower data set is a video from a fixed camera, where the objective is to identify the labels and locations of objects among five classes. We show an example frame along with the selected region that is input to the chip. (**B**) The region is transduced from pixels into spike events to create two parallel channels: a high-resolution channel (left) that represents the what pathway for labeling objects and a low-resolution channel (right) that represents the where pathway for locating salient objects. These pathways are inspired by dorsal and ventral streams in visual cortex (*4*). (**C**) What and where pathways are combined to form a what-where map. In the what network, colors represent the spiking activity for a grid of neurons, where different neurons were trained (offline) to recognize different object types. By overlaying the responses, brighter colors indicate more-confident labels. In the where network, neurons were trained (offline) to detect salient regions, and darker responses indicate more-salient regions. (**D**) Object bounding boxes reported by the chip.

TrueNorth architecture: 1million spiking-neuron

TrueNorth application: Object recognition

1-D Signal Representation: from EE233/241



Time series data: time points n=,…,-2,-1,0,1,2,..

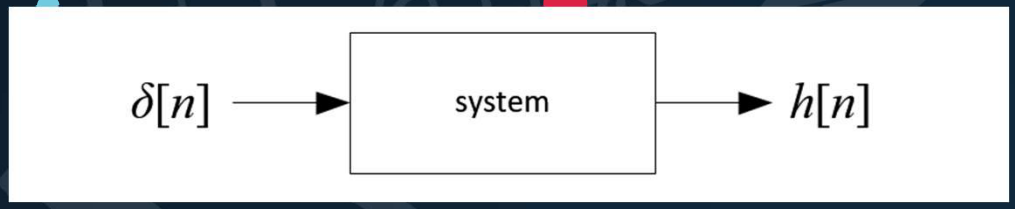
$$x[0] = x[0] \cdot \delta[n] \quad = 2 \cdot \delta[n-0]$$
$$x[1] = x[1] \cdot \delta[n-1] = 3 \cdot \delta[n-1]$$
$$x[2] = x[2] \cdot \delta[n-2] = 1 \cdot \delta[n-2]$$

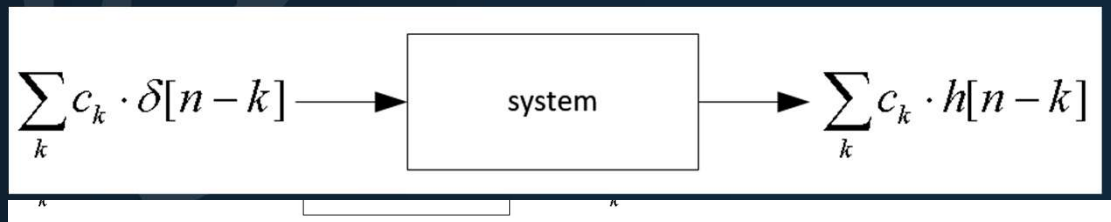
$$x[n] = x[0] \cdot \delta[n-0] + x[1] \cdot \delta[n-1] + x[2] \cdot \delta[n-2]$$

In general, a signal can be written as sum of scaled and shifted delta functions;

$$x[n] = \sum_{k=-\infty}^{\infty} x[k] \cdot \delta[n-k]$$

Transfer Function of a Linear System: EE233





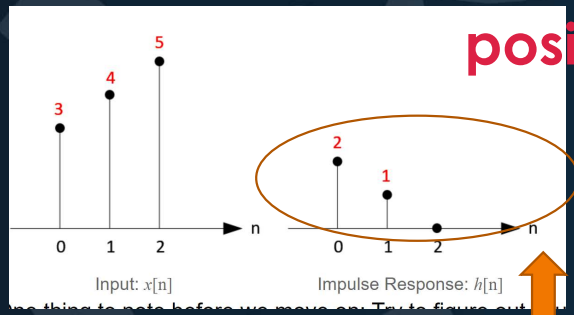
For example, if input signal is $x[n] = 2\cdot\delta[n] + 3\cdot\delta[n-1] + 1\cdot\delta[n-2]$, then the output is simply $y[n] = 2\cdot h[n] + 3\cdot h[n-1] + 1\cdot h[n-2]$.

Therefore, we now clearly see that if the input signal is $x[n] = \sum x[k] \cdot \delta[n-k]$, then the output will be $y[n] = \sum x[k] \cdot h[n-k]$. Note one condition; convolution works on the linear and time invariant system.

To summarize, a signal is decomposed into a set of impulses and the output signal can be computed by adding the scaled and shifted impulse responses.

1-D Convolution Example

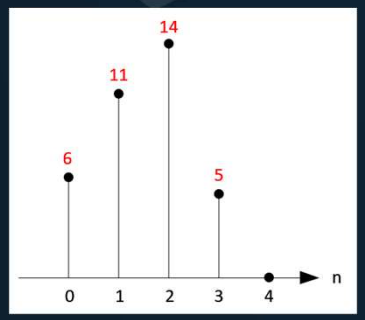
$x[n] = \{ 3, 4, 5 \}$
$h[n] = \{ 2, 1 \}$



Shifted by n positions

flipped

$$y[n] = \sum x[k] \cdot h[n-k]$$



$$y[0] = \sum_{k=-\infty}^{\infty} x[k] \cdot h[0-k]$$
$$= x[0] \cdot h[0] + x[1] \cdot h[0-1] + x[2] \cdot h[0-2] + \dots$$
$$= x[0] \cdot h[0]$$
$$= 3 \cdot 2$$
$$= 6$$

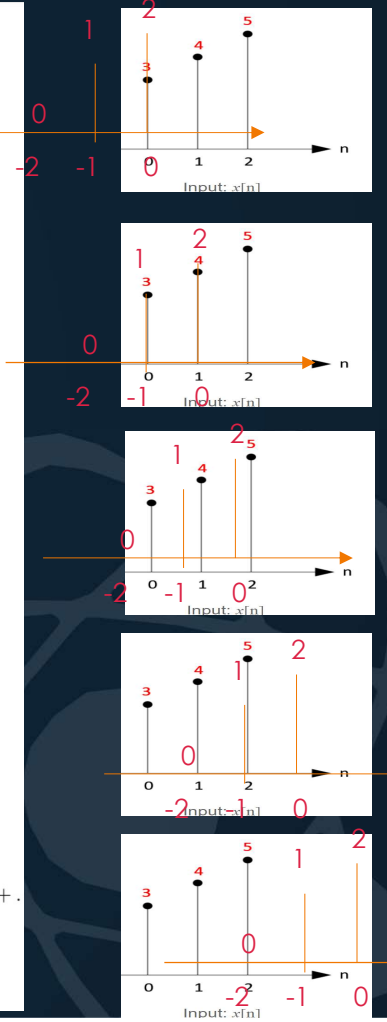
$$y[1] = \sum_{k=-\infty}^{\infty} x[k] \cdot h[1-k]$$
$$= x[0] \cdot h[1-0] + x[1] \cdot h[1-1] + x[2] \cdot h[1-2] + \dots$$
$$= x[0] \cdot h[1] + x[1] \cdot h[0]$$
$$= 3 \cdot 1 + 4 \cdot 2$$
$$= 11$$

$$y[2] = \sum_{k=-\infty}^{\infty} x[k] \cdot h[2-k]$$
$$= x[0] \cdot h[2-0] + x[1] \cdot h[2-1] + x[2] \cdot h[2-2] + \dots$$
$$= x[0] \cdot h[2] + x[1] \cdot h[1] + x[2] \cdot h[0]$$
$$= 3 \cdot 0 + 4 \cdot 1 + 5 \cdot 2$$
$$= 14$$

$$y[3] = \sum_{k=-\infty}^{\infty} x[k] \cdot h[3-k]$$
$$= x[0] \cdot h[3-0] + x[1] \cdot h[3-1] + x[2] \cdot h[3-2] + x[3] \cdot h[3-3] + \dots$$
$$= x[0] \cdot h[3] + x[1] \cdot h[2] + x[2] \cdot h[1] + x[3] \cdot h[0]$$
$$= 3 \cdot 0 + 4 \cdot 0 + 5 \cdot 1 + 0 \cdot 2$$
$$= 5$$

$$y[4] = \sum_{k=-\infty}^{\infty} x[k] \cdot h[4-k]$$
$$= x[0] \cdot h[4-0] + x[1] \cdot h[4-1] + x[2] \cdot h[4-2] + x[3] \cdot h[4-3] + x[4] \cdot h[4-4] + \dots$$
$$= x[0] \cdot h[4] + x[1] \cdot h[3] + x[2] \cdot h[2] + x[3] \cdot h[1] + x[4] \cdot h[0]$$
$$= 3 \cdot 0 + 4 \cdot 0 + 5 \cdot 0 + 0 \cdot 2 + 0 \cdot 1 + 0 \cdot 0$$
$$= 0$$

Shifted Vector Dot product



1-D Convolution in C++

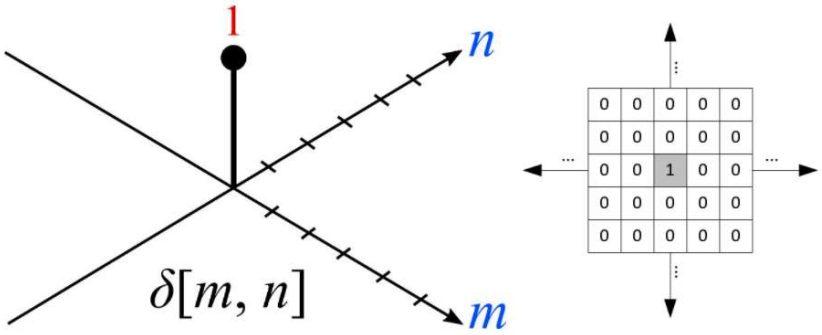
C++ Implementation for Convolution 1D

Implementing the convolution algorithm is quite simple. The code snippet is following;

```
for ( i = 0; i < sampleCount; i++ )
{
    y[i] = 0;                          // set to zero before sum
    for ( j = 0; j < kernelCount; j++ )
    {
        y[i] += x[i - j] * h[j];       // convolve: multiply and accumulate
    }
}
```

Shifted Vector Dot product

2-D Signal and 2-D Filter



Delta function (Impulse function) in 2D space

The second image is 2D matrix representation of impulse function. The shaded center point is the origin where m=n=0.

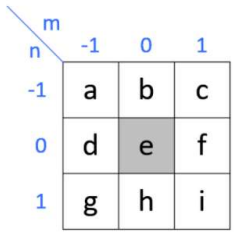
Once again, a signal can be decomposed into a sum of scaled and shifted impulse (delta) functions;

$$x[m,n] = \sum_{j=-\infty}^{\infty} \sum_{i=-\infty}^{\infty} x[i,j] \cdot \delta[m-i, n-j]$$

For example, $x[0, 0]$ is $x[0, 0]\cdot\delta[m, n]$, $x[1, 2]$ is $x[1, 2]\cdot\delta[m-1, n-2]$, and so on. Note that the matrices are referenced here as [column, row], not [row, column]. m is horizontal (column) direction and n is vertical (row) direction.

And, the output of linear and time invariant system can be written by convolution of input signal $x[m, n]$, and impulse response, $h[m, n]$;

$$y[m,n] = x[m,n] * h[m,n] = \sum_{j=-\infty}^{\infty} \sum_{i=-\infty}^{\infty} x[i,j] \cdot h[m-i, n-j]$$



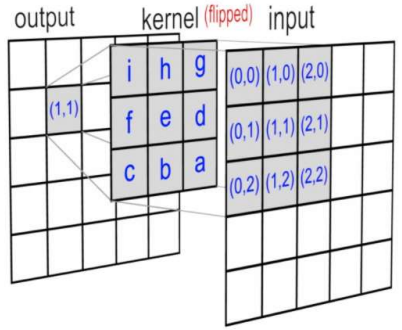
Examine an example to clarify how to convolve in 2D space.
Let's say that the size of impulse response (kernel) is 3x3, and it's values are a, b, c, d,...

Notice the origin (0,0) is located in the center of the kernel.

Let's pick a simplest sample and compute convolution, for instance, the output at (1, 1) will be;

$$\begin{aligned} y[1,1] = \sum_{j=-\infty}^{\infty} \sum_{i=-\infty}^{\infty} x[i,j] \cdot h[1-i, 1-j] \\ = \; x[0,0]\cdot h[1,1] \;+ x[1,0]\cdot h[0,1] \;+ x[2,0]\cdot h[-1,1] \\ + x[0,1]\cdot h[1,0] \;+ x[1,1]\cdot h[0,0] \;+ x[2,1]\cdot h[-1,0] \\ + x[0,2]\cdot h[1,-1] + x[1,2]\cdot h[0,-1] + x[2,2]\cdot h[-1,-1] \end{aligned}$$

It results in sum of 9 elements of scaled and shifted impulse responses. The following image shows the graphical representation of 2D convolution.

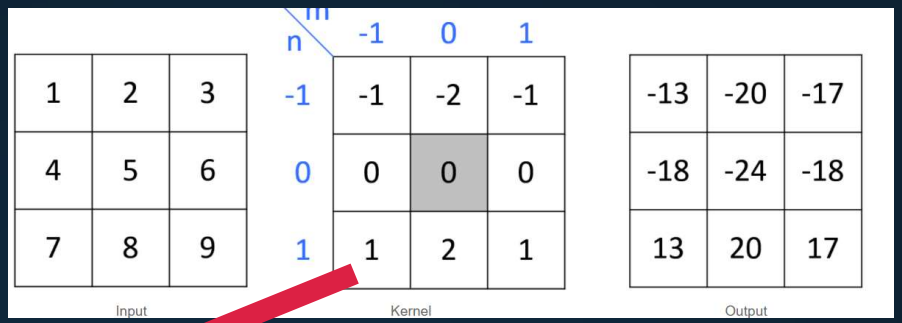


2D Convolution

Notice that the kernel matrix is flipped both horizontal and vertical direction before multiplying the overlapped input data, because $x[0,0]$ is multiplied by the last sample of impulse response, $h[1,1]$. And $x[2,2]$ is multiplied by the first sample, $h[-1,-1]$.

2-D Convolution Breakdown

Shifted Matrix Dot product



flipped

$$y[0,0] = \sum_j \sum_i x[i,j] \cdot h[0-i, 0-j]$$
$$= x[-1,-1] \cdot h[1,1] + x[0,-1] \cdot h[0,1] + x[1,-1] \cdot h[-1,1] + x[-1,0] \cdot h[1,0] + x[0,0] \cdot h[0,0] + x[1,0] \cdot h[-1,0] + x[-1,1] \cdot h[1,-1] + x[0,1] \cdot h[0,-1] + x[1,1] \cdot h[-1,-1]$$
$$= 0 \cdot 1 + 0 \cdot 2 + 0 \cdot 1 + 0 \cdot 0 + 1 \cdot 0 + 2 \cdot 0 + 0 \cdot (-1) + 4 \cdot (-2) + 5 \cdot (-1)$$
$$= -13$$

$$y[1,0] = \sum_j \sum_i x[i,j] \cdot h[1-i, 0-j]$$
$$= x[0,-1] \cdot h[1,1] + x[1,-1] \cdot h[0,1] + x[2,-1] \cdot h[-1,1] + x[0,0] \cdot h[1,0] + x[1,0] \cdot h[0,0] + x[2,0] \cdot h[-1,0] + x[0,1] \cdot h[1,-1] + x[1,1] \cdot h[0,-1] + x[2,1] \cdot h[-1,-1]$$
$$= 0 \cdot 1 + 0 \cdot 2 + 0 \cdot 1 + 1 \cdot 0 + 2 \cdot 0 + 3 \cdot 0 + 4 \cdot (-1) + 5 \cdot (-2) + 6 \cdot (-1)$$
$$= -20$$

$$y[2,0] = \sum_j \sum_i x[i,j] \cdot h[2-i, 0-j]$$
$$= x[1,-1] \cdot h[1,1] + x[2,-1] \cdot h[0,1] + x[3,-1] \cdot h[-1,1] + x[1,0] \cdot h[1,0] + x[2,0] \cdot h[0,0] + x[3,0] \cdot h[-1,0] + x[1,1] \cdot h[1,-1] + x[2,1] \cdot h[0,-1] + x[3,1] \cdot h[-1,-1]$$
$$= 0 \cdot 1 + 0 \cdot 2 + 0 \cdot 1 + 2 \cdot 0 + 3 \cdot 0 + 0 \cdot 0 + 5 \cdot (-1) + 6 \cdot (-2) + 0 \cdot (-1)$$
$$= -17$$

$$y[0,1] = \sum_j \sum_i x[i,j] \cdot h[0-i, 1-j]$$
$$= x[-1,0] \cdot h[1,1] + x[0,0] \cdot h[0,1] + x[1,0] \cdot h[-1,1] + x[-1,1] \cdot h[1,0] + x[0,1] \cdot h[0,0] + x[1,1] \cdot h[-1,0] + x[-1,2] \cdot h[1,-1] + x[0,2] \cdot h[0,-1] + x[1,2] \cdot h[-1,-1]$$
$$= 0 \cdot 1 + 1 \cdot 2 + 2 \cdot 1 + 0 \cdot 0 + 4 \cdot 0 + 5 \cdot 0 + 0 \cdot (-1) + 7 \cdot (-2) + 8 \cdot (-1)$$
$$= -18$$

$$y[1,1] = \sum_j \sum_i x[i,j] \cdot h[1-i, 1-j]$$
$$= x[0,0] \cdot h[1,1] + x[1,0] \cdot h[0,1] + x[2,0] \cdot h[-1,1] + x[0,1] \cdot h[1,0] + x[1,1] \cdot h[0,0] + x[2,1] \cdot h[-1,0] + x[0,2] \cdot h[1,-1] + x[1,2] \cdot h[0,-1] + x[2,2] \cdot h[-1,-1]$$
$$= 1 \cdot 1 + 2 \cdot 2 + 3 \cdot 1 + 4 \cdot 0 + 5 \cdot 0 + 6 \cdot 0 + 7 \cdot (-1) + 8 \cdot (-2) + 9 \cdot (-1)$$
$$= -24$$

$$y[2,1] = \sum_j \sum_i x[i,j] \cdot h[2-i, 1-j]$$
$$= x[1,0] \cdot h[1,1] + x[2,0] \cdot h[0,1] + x[3,0] \cdot h[-1,1] + x[1,1] \cdot h[1,0] + x[2,1] \cdot h[0,0] + x[3,1] \cdot h[-1,0] + x[1,2] \cdot h[1,-1] + x[2,2] \cdot h[0,-1] + x[3,2] \cdot h[-1,-1]$$
$$= 2 \cdot 1 + 3 \cdot 2 + 0 \cdot 1 + 5 \cdot 0 + 6 \cdot 0 + 0 \cdot 0 + 8 \cdot (-1) + 9 \cdot (-2) + 0 \cdot (-1)$$
$$= -18$$

$$y[0,2] = \sum_j \sum_i x[i,j] \cdot h[0-i, 2-j]$$
$$= x[-1,1] \cdot h[1,1] + x[0,1] \cdot h[0,1] + x[1,1] \cdot h[-1,1] + x[-1,2] \cdot h[1,0] + x[0,2] \cdot h[0,0] + x[1,2] \cdot h[-1,0] + x[-1,3] \cdot h[1,-1] + x[0,3] \cdot h[0,-1] + x[1,3] \cdot h[-1,-1]$$
$$= 0 \cdot 1 + 4 \cdot 2 + 5 \cdot 1 + 0 \cdot 0 + 7 \cdot 0 + 8 \cdot 0 + 0 \cdot (-1) + 0 \cdot (-2) + 0 \cdot (-1)$$
$$= 13$$

$$y[1,2] = \sum_j \sum_i x[i,j] \cdot h[1-i, 2-j]$$
$$= x[0,1] \cdot h[1,1] + x[1,1] \cdot h[0,1] + x[2,1] \cdot h[-1,1] + x[0,2] \cdot h[1,0] + x[1,2] \cdot h[0,0] + x[2,2] \cdot h[-1,0] + x[0,3] \cdot h[1,-1] + x[1,3] \cdot h[0,-1] + x[2,3] \cdot h[-1,-1]$$
$$= 4 \cdot 1 + 5 \cdot 2 + 6 \cdot 1 + 7 \cdot 0 + 8 \cdot 0 + 9 \cdot 0 + 0 \cdot (-1) + 0 \cdot (-2) + 0 \cdot (-1)$$
$$= 20$$

$$y[2,2] = \sum_j \sum_i x[i,j] \cdot h[2-i, 2-j]$$
$$= x[1,1] \cdot h[1,1] + x[2,1] \cdot h[0,1] + x[3,1] \cdot h[-1,1] + x[1,2] \cdot h[1,0] + x[2,2] \cdot h[0,0] + x[3,2] \cdot h[-1,0] + x[1,3] \cdot h[1,-1] + x[2,3] \cdot h[0,-1] + x[3,3] \cdot h[-1,-1]$$
$$= 5 \cdot 1 + 6 \cdot 2 + 0 \cdot 1 + 8 \cdot 0 + 9 \cdot 0 + 0 \cdot 0 + 0 \cdot (-1) + 0 \cdot (-2) + 0 \cdot (-1)$$
$$= 17$$

2-D Convoluion in C++

C++ Algorithm for Convolution 2D

We need 4 nested loops for 2D convolution instead of 2 loops in 1D convolution.

```
// find center position of kernel (half of kernel size)
kCenterX = kCols / 2;
kCenterY = kRows / 2;

for(i=0; i < rows; ++i)              // rows
{
    for(j=0; j < cols; ++j)          // columns
    {
        for(m=0; m < kRows; ++m)     // kernel rows
        {
            mm = kRows - 1 - m;      // row index of flipped kernel

            for(n=0; n < kCols; ++n) // kernel columns
            {
                nn = kCols - 1 - n;  // column index of flipped kernel

                // index of input signal, used for checking boundary
                ii = i + (kCenterY - mm);
                jj = j + (kCenterX - nn);

                // ignore input samples which are out of bound
                if( ii >= 0 && ii < rows && jj >= 0 && jj < cols )
                    out[i][j] += in[ii][jj] * kernel[mm][nn];
            }
        }
    }
}
```

Data Flow

MAC
Multiply-And-Accumulate

FC-Full Connection

The general formula for a matrix-vector product is

$$\mathbf{A}\mathbf{x} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n \\ \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n \end{bmatrix},$$

$$\begin{aligned} \mathbf{A}\mathbf{x} &= \begin{bmatrix} 1 & -1 & 2 \\ 0 & -3 & 1 \end{bmatrix} \begin{bmatrix} 2 \\ 1 \\ 0 \end{bmatrix} \\ &= \begin{bmatrix} 2 \cdot 1 - 1 \cdot 1 + 0 \cdot 2 \\ 2 \cdot 0 - 1 \cdot 3 + 0 \cdot 1 \end{bmatrix} \\ &= \begin{bmatrix} 1 \\ -3 \end{bmatrix}. \end{aligned}$$

The Number of Multiplications Sparsity

C++ Matrix-Vector Product

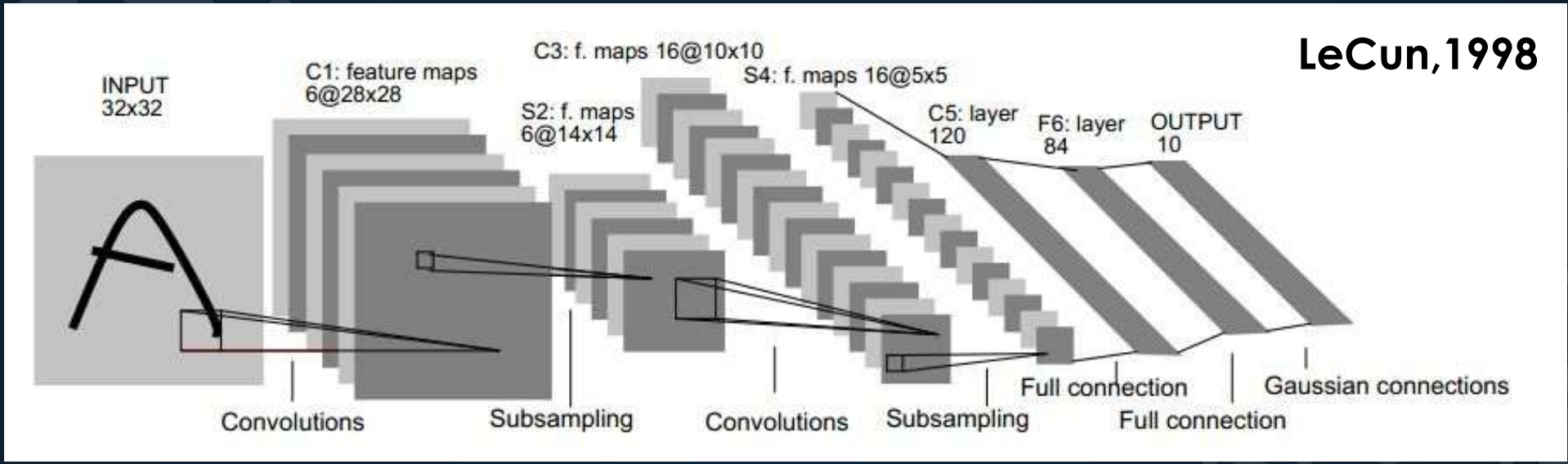
```
#include<stdio.h>
#include <iostream>
#include<omp.h>
#define m 3
#define n 3
using namespace std;

int main() {
  int a[m]={2,2,2},i,j,b[m][n]{{1,1,1},{2,2,2},{1,1,1}},c[m]={0,0,0};

  for(i=0;i<m;i++){
    for(j=0;j<n;j++){
      c[i]=c[i]+(a[i]* b[i][j]);
    }
  }
}
```

PART ONE "What is"

DEEP LEARNING

Deep Learning Algorithms

Early Neural Networks LeNet-5



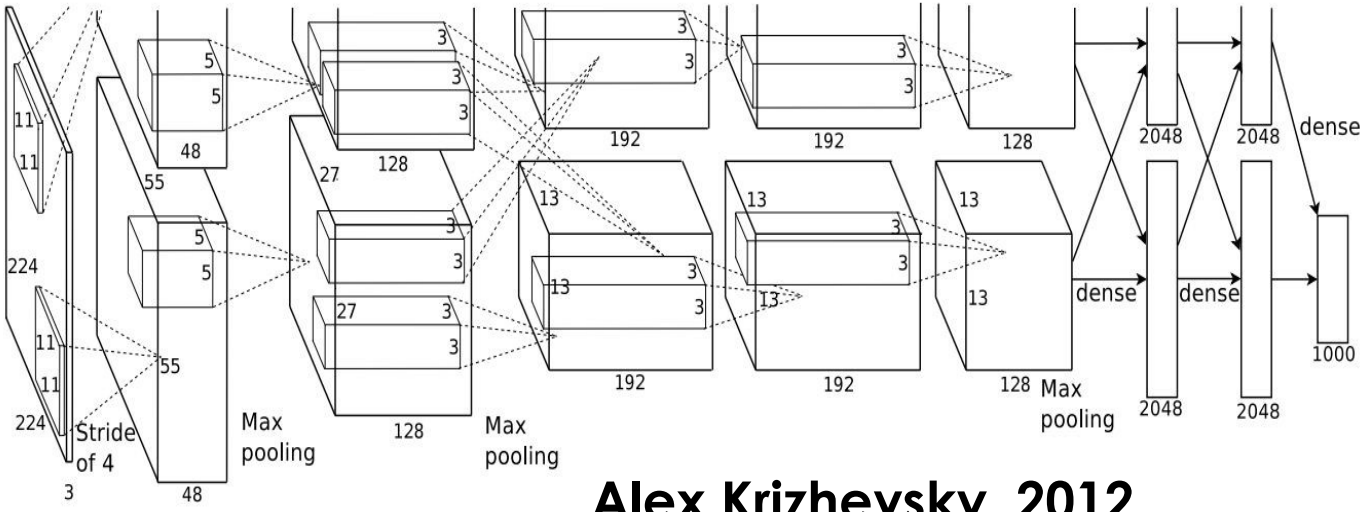
PART ONE "What is"

DEEP LEARNING

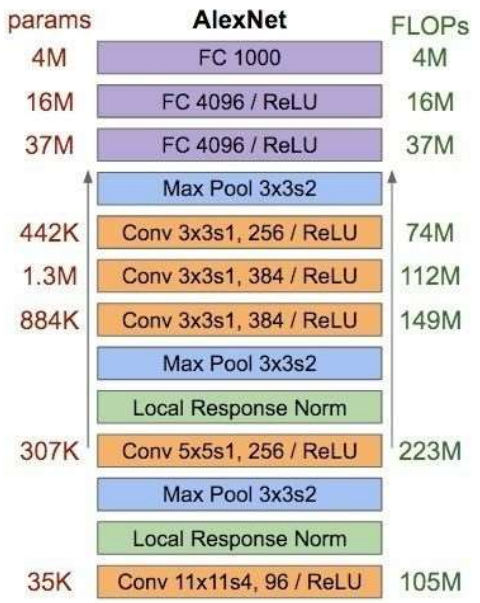
Deep Learning Algorithms

Classic CNN

AlexNet



Alex Krizhevsky, 2012



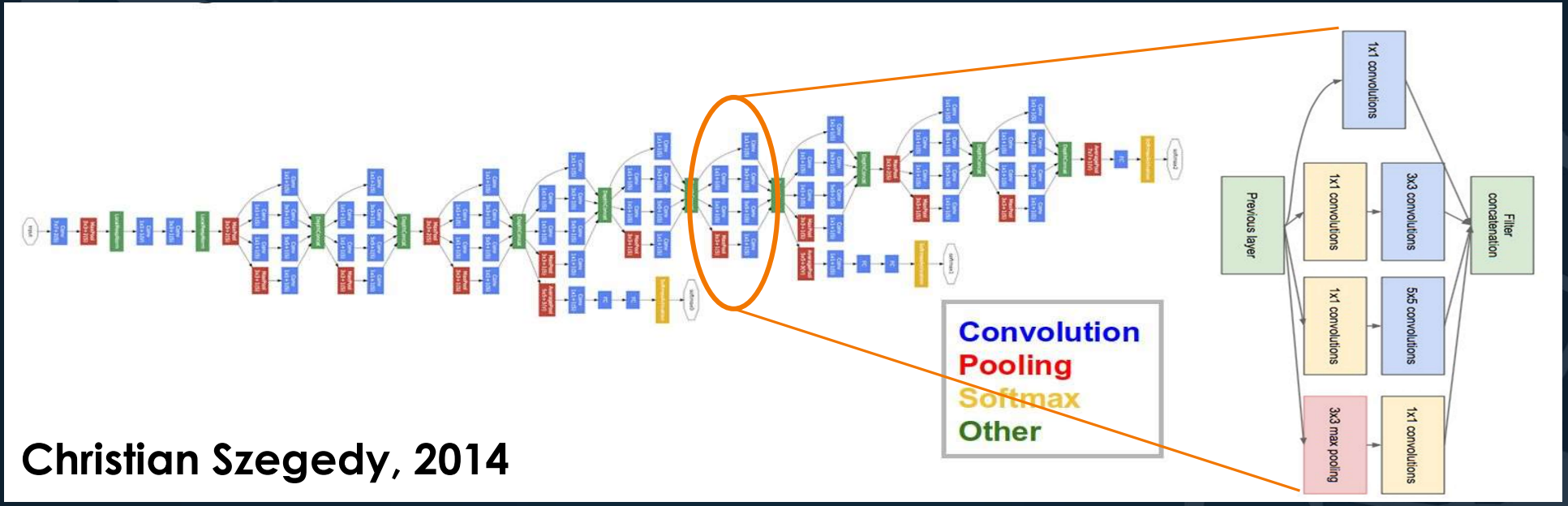
PART ONE "What is"

DEEP LEARNING

Deep Learning Algorithms

Classic CNN

GoogLeNet

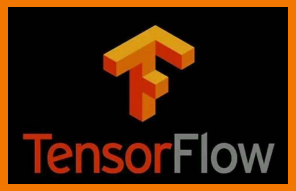


Christian Szegedy, 2014

PART ONE "What is"

DEEP LEARNING

Deep Learning Frameworks



Frame: TensorFlow
Holder: Google
Language: Python/C++/...

Frame: CNTK
Holder: Microsoft
Language: C++

Frame : Caffe
Holder: BVLC
Language: Python/C++

Frame : Torch
Holder: Facebook
Language: Lua

PART ONE "What is"

DEEP LEARNING

Deep Learning Advantages

Realizable 01	Trainable 02	Program -mable 03	Practical 04
CMOS	**Back Propagation**	**CPU, GPU...**	**Applications**
The implementation of the algorithm does not require special processes and devices. Standard CMOS Technology	Mature theory and training methosd (Back Propagation)	Multiple programming languages and frameworks support CPU & GPU support	Widely used in computer vision, speech recognition and self-driving cars

Mains tream


What for...
PART TWO
Applications of Deep Learning:
Throughput and Power

PART TWO "Use"

AI Applications

Security monitoring

- Crossing monitoring
- Vehicle recognition
- Video structured description
- Face recognition
- Video detection

Smart medical

- Hierarchical diagnosis and treatment system
- Medical imaging
- Medical large data system
- Medical Assistant
- Gene detection & analysis
- Drug development

Smart home

- Human-computer interaction terminal
- Entrance guard system
- Remote control
- Building monitoring
- Home service robot

Consumer electronics

- Smart phone
- Wearable devices
- Virtual / augmented reality
- UAV

PART TWO "Use"

Security monitoring

Crossing monitoring

By placing camera and auxiliary lighting at crowded places such as road intersections, entrances and exits of specific places, corridors, etc., intelligent and automated multi-target real-time video monitoring can be realized after applying the deep learning method of artificial intelligence

Vehicle Recognition

Application of artificial intelligence, visual computing combined with image processing technology, can be on an all-weather environment of the vehicle license plate, model, the driver's identity or even automatically identify the status

Face Recognition

automatically detected and tracked in the images to match the facial features of detected human faces. Using the deep learning convolutional neural network, Achieve 99% recognition rate By collecting images or video streams containing human faces

Video structured description

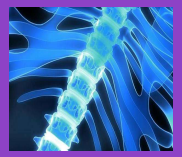
Detect the moving target as a foreground object from a relatively constant background, and mark and identify related faces, gender, age, dress features, related items and accessories

PART TWO “Use”

Smart medical

Hierarchical diagnosis and treatment system

The promotion of artificial intelligence has made it possible to transfer a large number of patients with common diseases, frequently-occurring diseases and chronic diseases from third-level medical institutions from experts to primary-level medical institutions through the use of the machine-aided medical system



Medical imaging

Artificial intelligence relies on powerful image recognition and deep learning techniques to reduce the pressure on doctors compared with human beings in medical imaging analysis compared with human beings with objective and highly repetitive, quantitative analysis, which can accumulate knowledge experience and low cost advantages Improve the efficiency and accuracy of diagnosis and treatment

Medical Assistant

Provide medical big data collection, preliminary medical diagnosis, medication regimen, etc. through virtual or physical machine medical assistants, use artificial intelligence natural language processing or voice interaction technology, and network home medical devices and sensors

Drug development

Developing virtual screening technology in combination with machine learning and artificial intelligence during drug development to replace or enhance traditional high-throughput screening processes can increase the probability of success of drugs reaching late-stage testing and reduce the cost of drug development

PART TWO "Use"

Smart home

Human-computer interaction terminal

By having human-like ability in artificial intelligence such as voice recognition and speech synthesis, domestic consumers can use voice mode to network and interact with household appliances and electronic devices

Entrance guard system

Through artificial intelligence in face recognition breakthrough, through the associated camera, to achieve offline localization of face certification, so as to achieve attendance and access control functions, replacing the traditional credit card and fingerprint attendance

Building monitoring

Realize intelligent and automated multi-target real-time video surveillance by deploying cameras and auxiliary lighting at key locations in residential communities and commercial real estate, applying visual image processing of artificial intelligence

Home service robot

Home robots, who can replace home-based home-service work, deepen their learning in voice and visual computing, lead to significant breakthroughs in perception and human-computer interaction, and greatly push home-service robots to practical use

PART TWO “Use”

Consumer electronics

Smart phone

Smartphones and computers, as computing and networking terminals, will bring new entertainment features, security certifications, human-computer interaction and work productivity gains when artificial intelligence is introduced.

Wearable devices

Wearable smart device will serve as a sensor carrier, through the recognition of people's instructions and respond to the needs of people, machines and even the cloud to achieve seamless experience and interaction, and through situational awareness to achieve intelligent sleep, wake-up and event detection to reduce work Consumption

Virtual / augmented reality

Application of deep learning and visual computing technology, UAVs can achieve better obstacle avoidance and path planning through accurate target detection and location, tracking and recognition, and visual interaction control through gesture and gesture recognition

UAV

Artificial Intelligence will provide an effective and natural interface for interaction between virtual and augmented reality, and the combination of the two will achieve a very realistic user experience

PART TWO "Use"

Deep Learning Vision Algorithms

Image Classification

Classical image processing and machine learning problems, given the number of image categories, through the depth of learning training image classification model to determine the type of object in the test image

Object Detection

Based on the deep learning convolution neural network technology, whether there is an interested object in the image is detected, if the object exists, the object is located and extracted while the image is classified. The object detection includes both detection and location and image classification

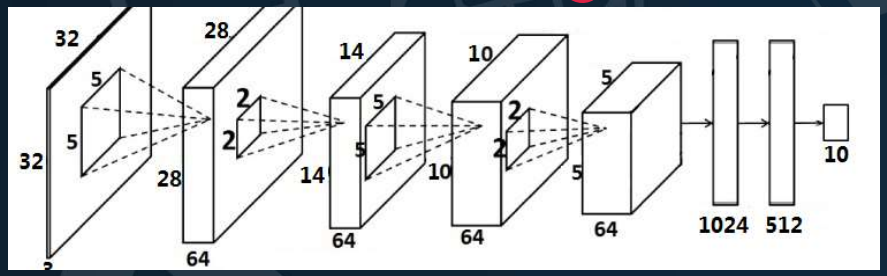
Feature Identification

The classic pattern recognition problem, the biggest breakthrough in depth learning, lies in the ability to generalize and distinguish the eigenvalues of recognition models through the powerful feature representation of deep neural networks

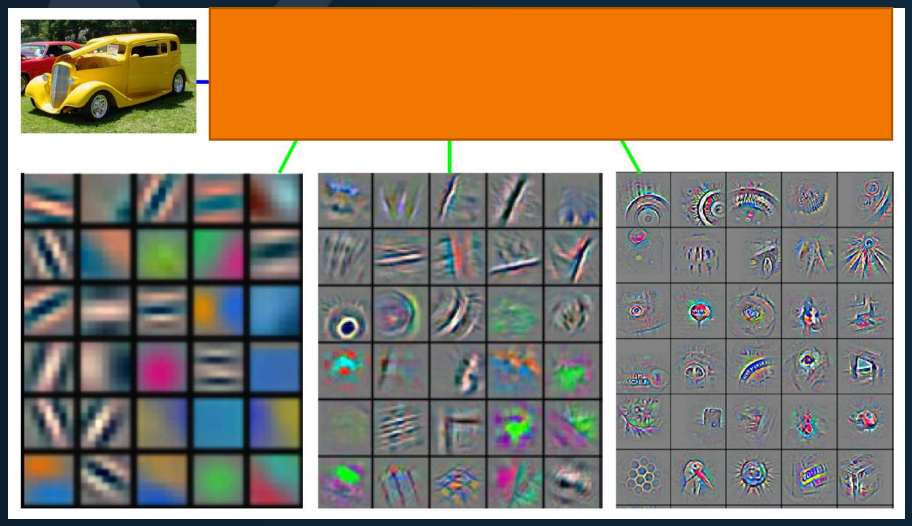
Object Tracking

Based on the target detection and recognition, this algorithm is used to achieve the continuous positioning of the target object in the video stream data, the association of the same target of continuous image data, and reduce the complexity of the target detection to achieve real-time video processing

PART TWO "Use"



CNN



CNN Features

Image Classification

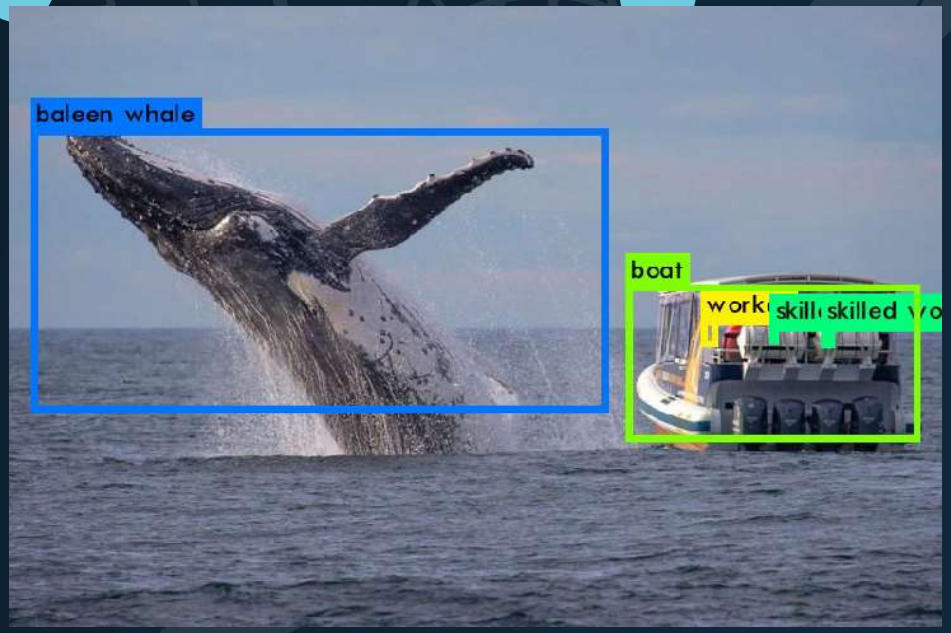
ImageNet ILSVRC

Feature Extractor: CNN

Classifier: Soft Max

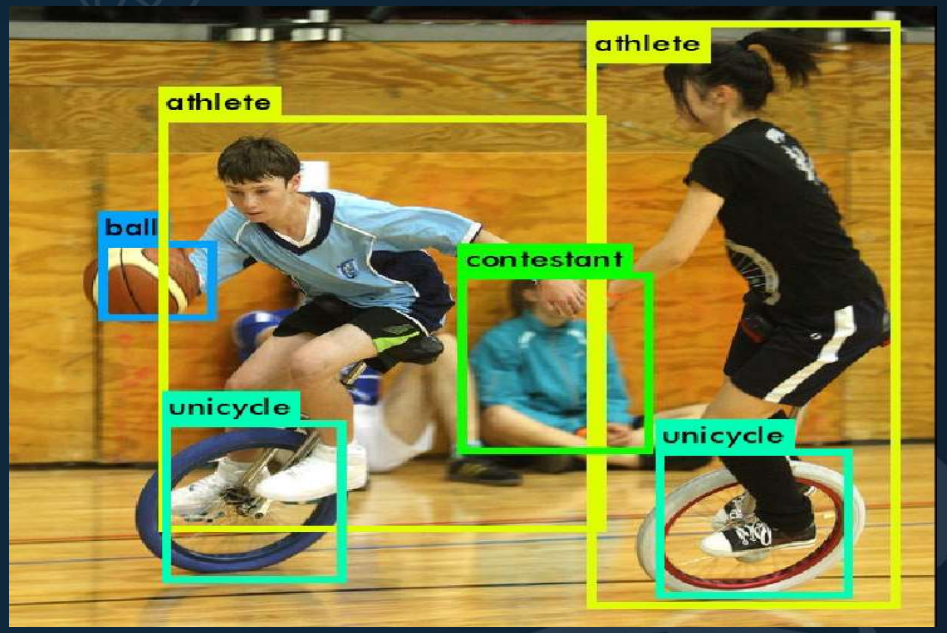
PART TWO "Use"

Object Detection



- End-to-end full convolution neural network method, while achieving the target detection and positioning and target identification classification
- Low complexity, good real-time, lack of positioning accuracy

Representative: **YOLO (2015), SSD (2015)**



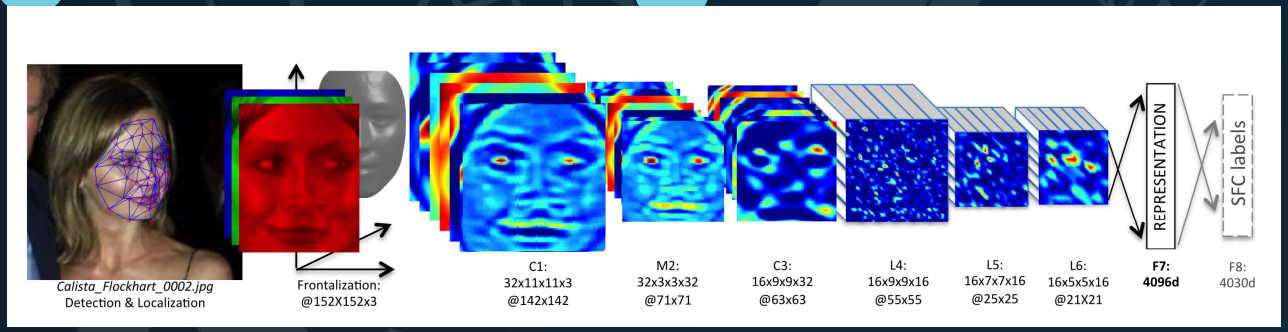
- Based on Region Proposal's deep learning method, the target localization and recognition classification are step by step
- With higher detection accuracy, high complexity

Representative :

Fast R-CNN (2015), Faster R-CNN (2015)

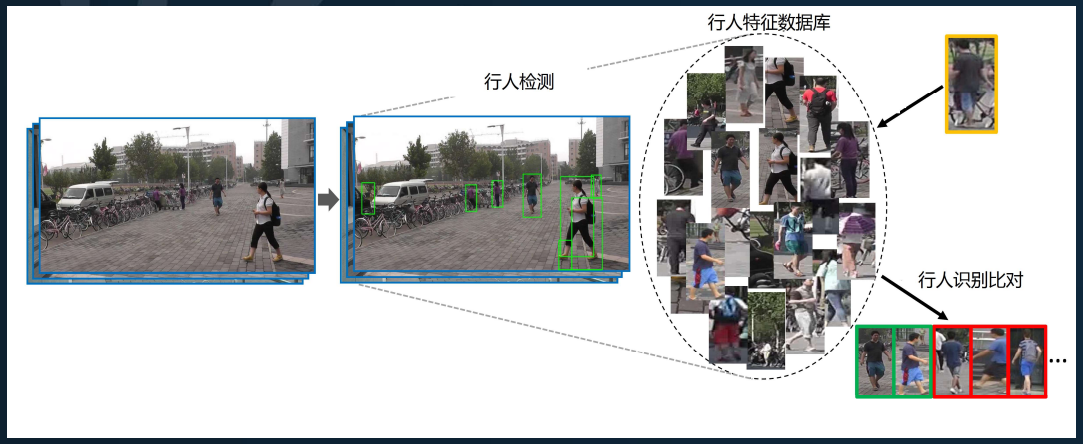
PART TWO “Use”

Eigenvalue matching



Face Comparison

Source：Facebook DeepFace, Closing the Gap to Human-Level Performance in Face Verification



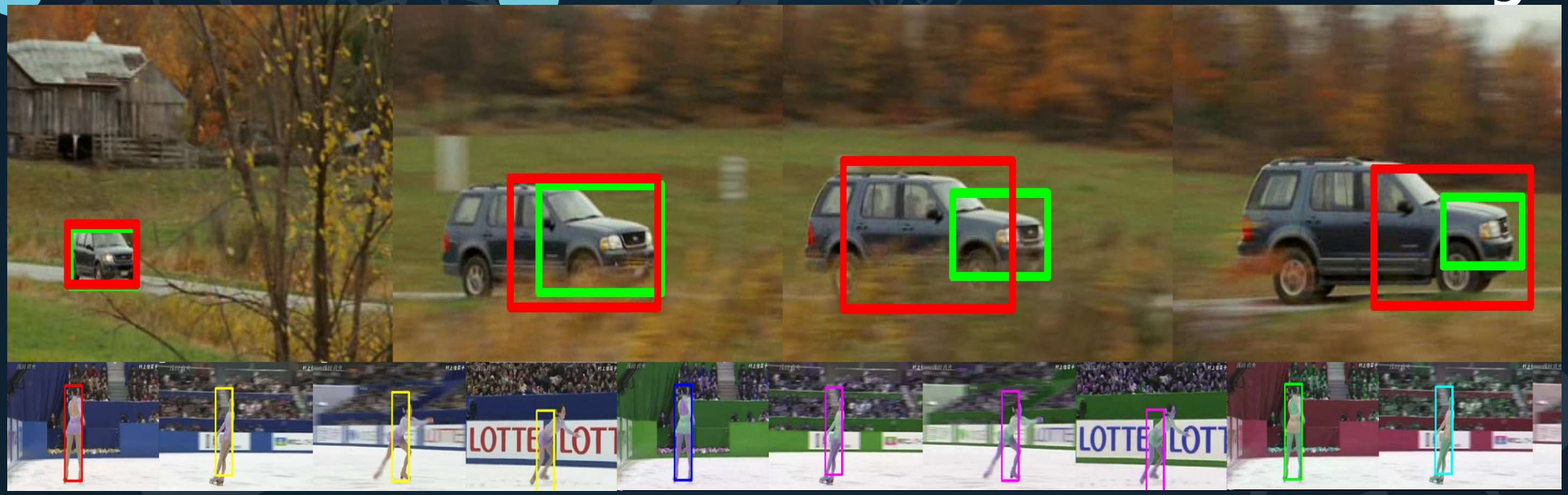
Pedestrian detection and comparison

Source：Person Re-identification: Past, Present and Future

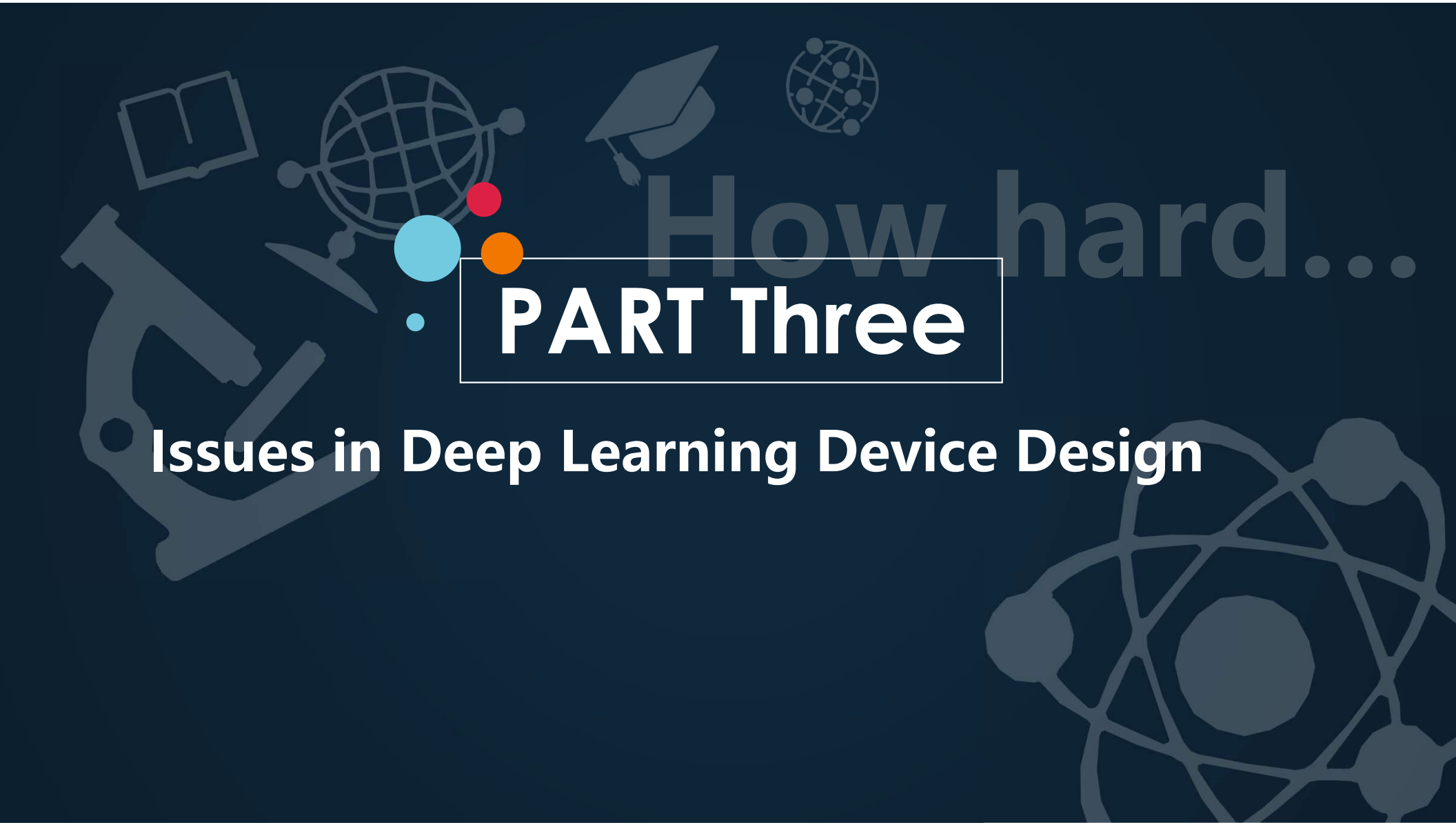
Compared with the traditional eigenvalue extraction methods, such as LBP, SURF, SIFT, HOG and other operators, the deep learning model has stronger feature-expressive ability and can use hierarchical expression and deeper model of big data and convolutional network model Better to adapt to a truly diverse environment, such as lighting, posture, angle rotation, shape changes

PART TWO "Use"

Target Tracking



- Compared with the traditional method of generating class (Kalman filter, particle filter and Mean-shift), discriminant class based depth learning has a better tracking effect
- The actual system of video target tracking and target detection combined with the convolutional network algorithm can share target computing and computing resources to achieve real-time video processing
- Representatives: **MDNet (2015), RNN tracker (2016), Fully Convolutional Siamese Network**


How hard...
PART Three
Issues in Deep Learning Device Design

PART THREE "Difficulty"

Deep Learning

Three elements

—— AlphaGo as example

Hardware as core

Algorithms

Rollout policy

Fast rollout

Value network

MCTS

…

Hardware

1202 CPU

176 GPU

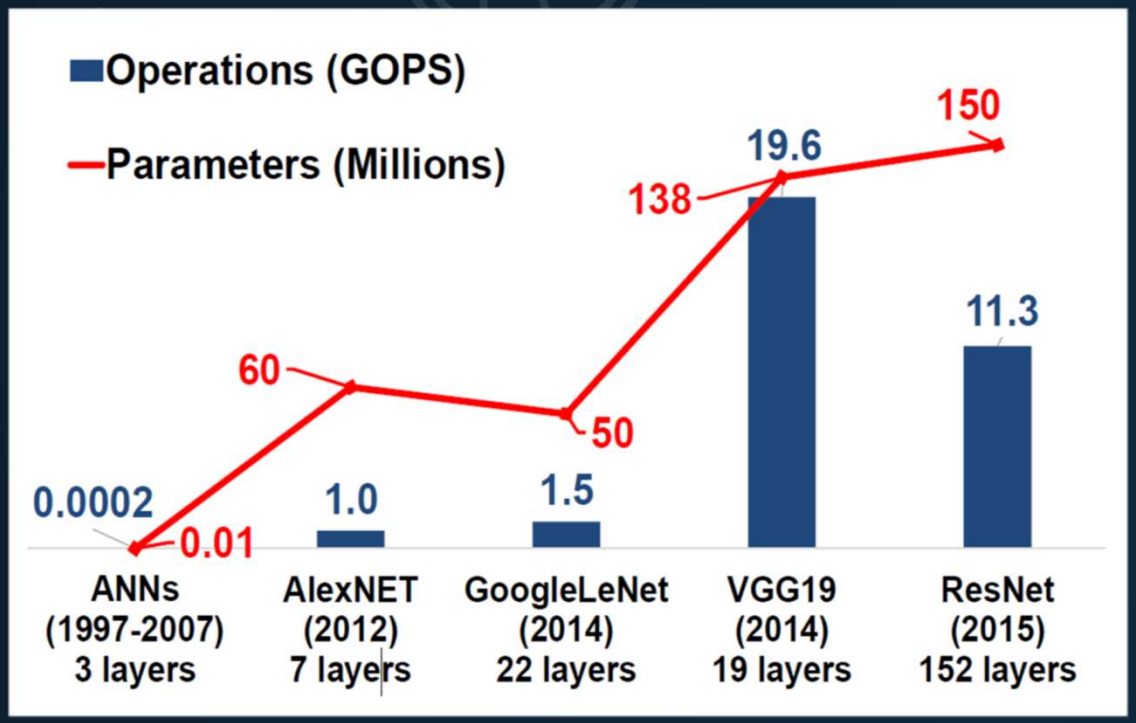
Data

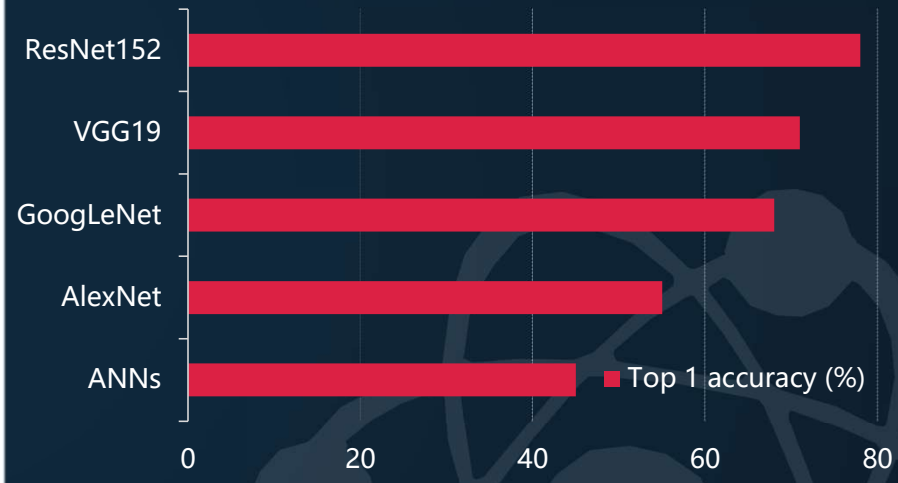
3M Human go map

As well as on maps the generated by massive self-play

PART THREE "Difficulty"

DNN evolution





PART THREE "Difficulty"

DNN Algorithms Evolution

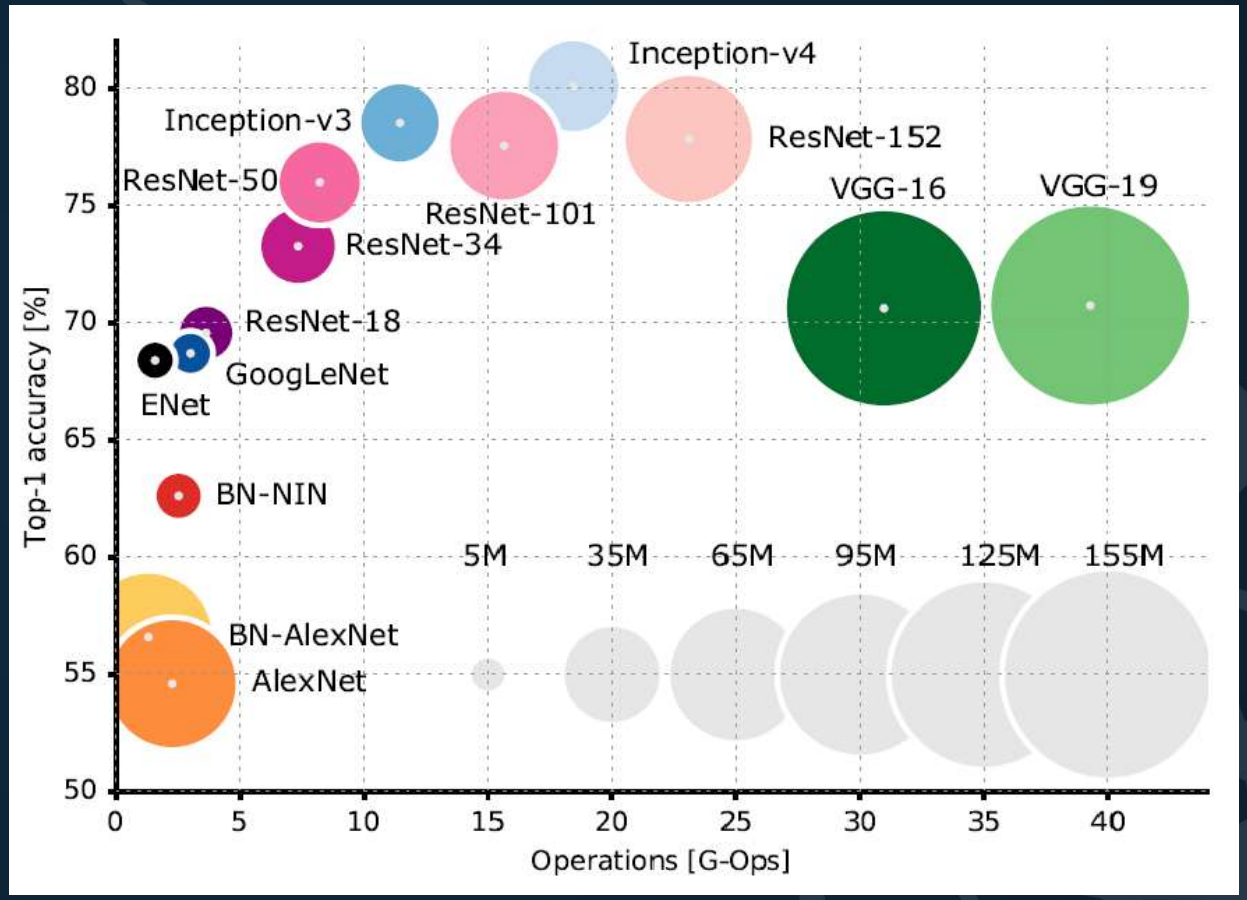
Mainstream NN

Performance

VS.

complexity

- The evolution direction of the algorithm evolves from low performance (lower left), multiple operations (upper right) and multiple parameters (big circle) toward reducing computation and parameters while improving performance (upper left and large circles), easing hardware requirements.



PART THREE "Difficulty"

DL Hardware Difficulties

1. Intense Computation

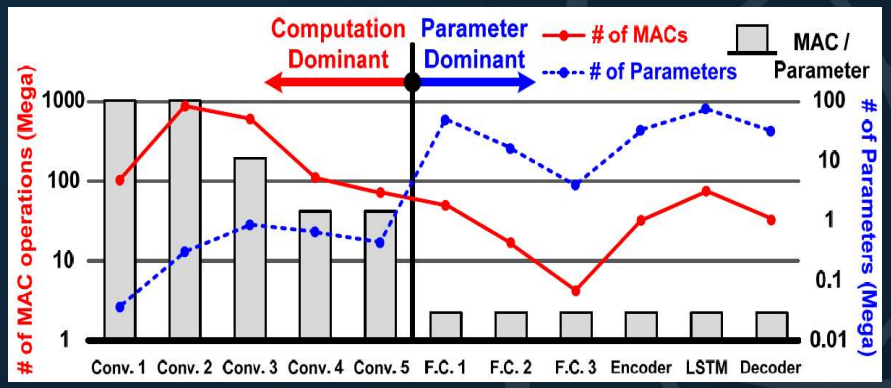
Mainly by the convolution operation-led,
With the image size, the number of features and the number of convolutional layers increase dramatically
In the mainstream network, the amount of computation per convolutional layer is on the order of 100-1000M

2. Considerable Parameter

Mainly by the full connection network-led,
Related to the dimensions and levels of the fully connected network
In the mainstream network, the parameters of each frame of image processing capacity of 15 ~ 150M order of magnitude

Look into AlexNet

Processing 227x227 pixels per frame Image:
832M operation +60 M data transmission



PART THREE "Difficulty"

DL Hardware Requirements

—— AlexNet as example

1. Computation Requirement

Use Alexnet to accomplish simple categorization tasks
Choose 227x227 pixel core region for each frame of picture
The calculation of each frame of picture 832MOPS
Standard frame rate 25fps
Using Intel i7 processor, clocked at 3.6GHz

6 cores parallel computing

The calculation of the core area has not yet taken into account the (usually also use a neural network), data transfer, etc. caused by additional computing. Therefore, to complete the video detection tasks in real time dozens of CPUs parallel processing

PART THREE "Difficulty"

DL Hardware Requirements

—— AlexNet as example

2. Bandwidth

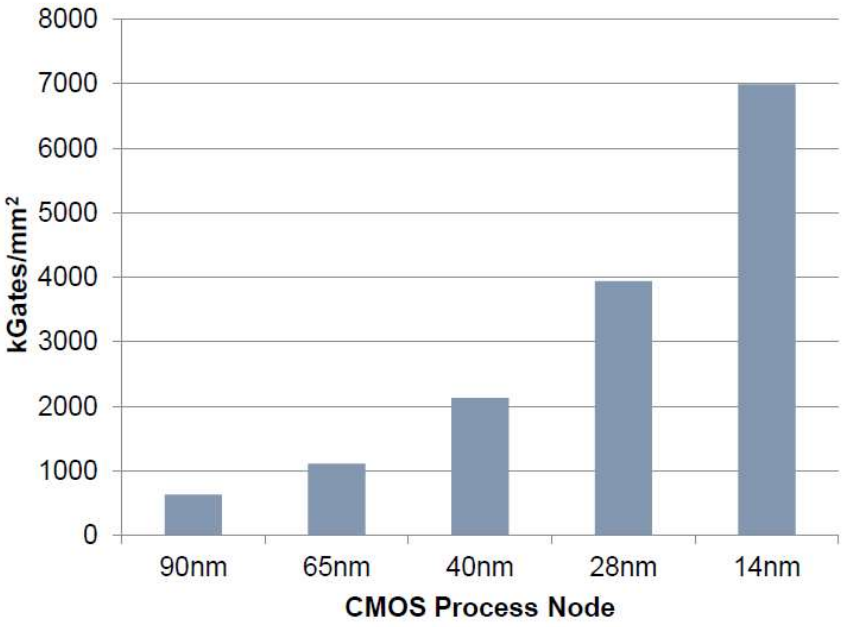
Data interface between processor and memory speed requirements:24G bps

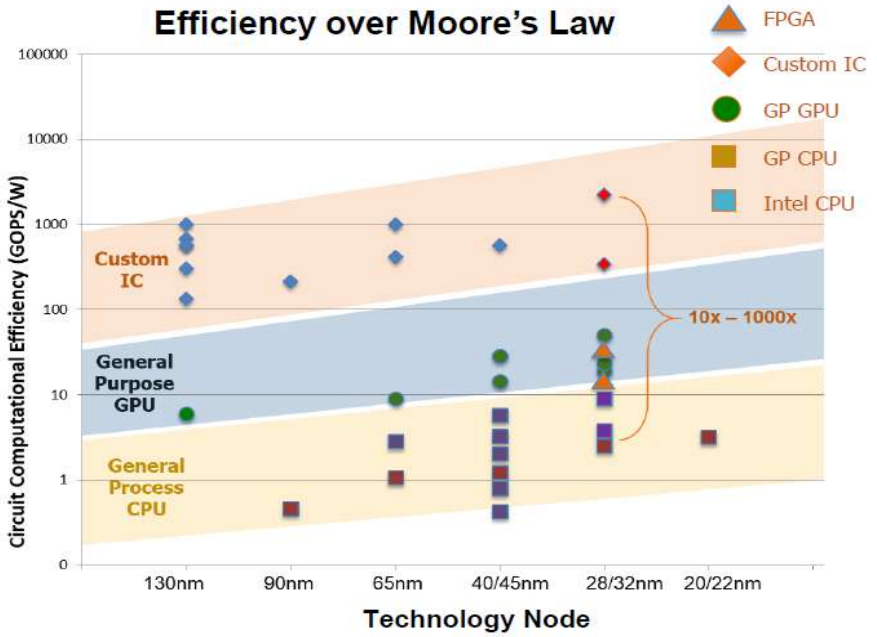
AlexNet has 60M parameters
Every frame of image processing uses all the parameters
Assume that each parameter bit width is 16b
Standard frame rate 25fps

Only parameters are considered in the calculation process, and the transmission of images and intermediate results has not been considered.

PART THREE "Difficulty"

Logic Circuits under Moore's Law



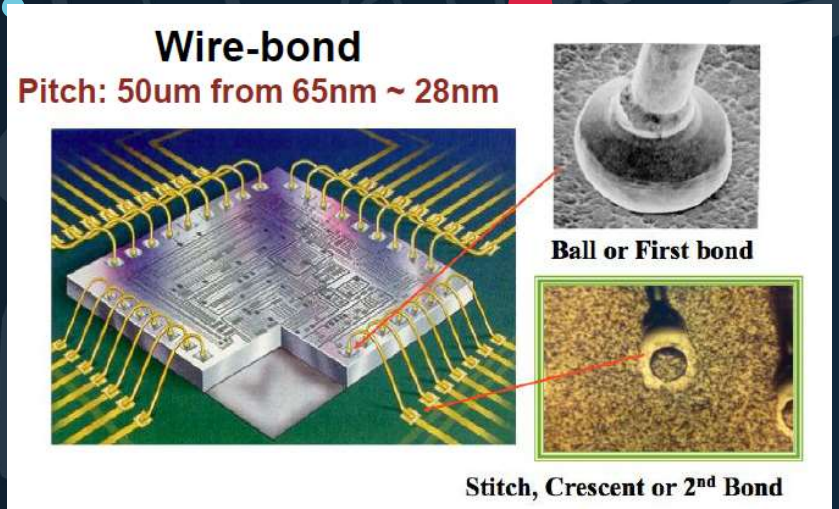


Increasing logic gate density
Faster and faster, load smaller and smaller
Energy efficiency according to Moore's Law evolution

Higher performance computing circuit, register design!

PART THREE "Difficulty"

Interface Package Under Moore's Law



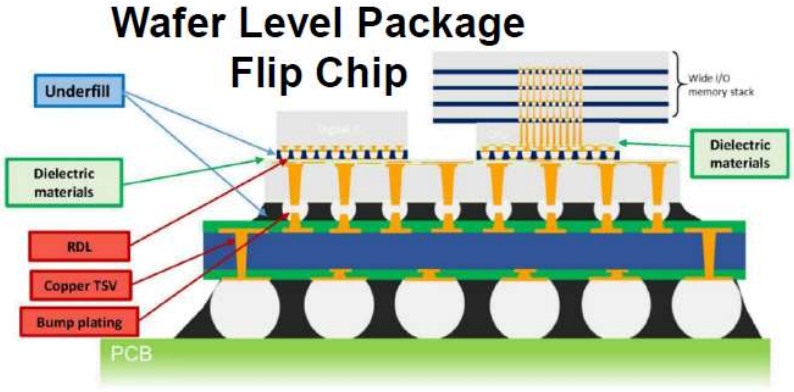
Low-Cost Wire-Bond Technology PAD Size Does Not Decrease with Moore's Law

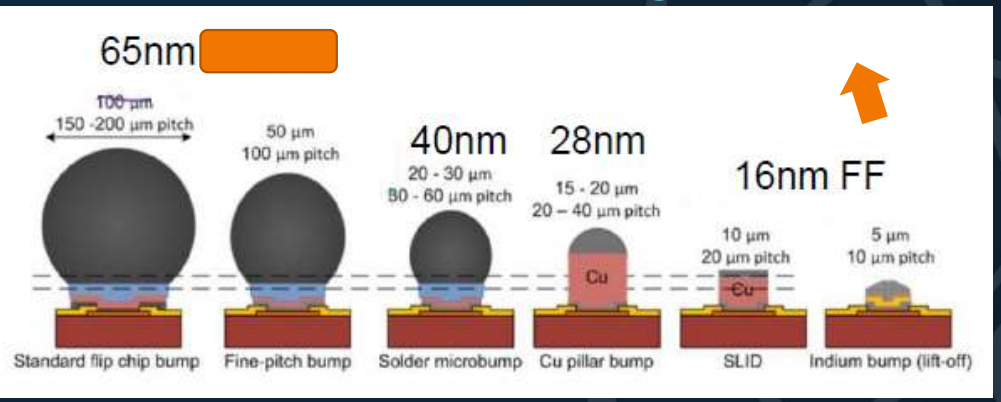
Inter-chip communication bandwidth and memory access speed

Can not develop, keep up with the computing power of upgrading

High-Cost Flip Chip Technology PAD size decreases with Moore's Law

In 2.5D / 3D packaging technology with the next Bandwidth and access speed continue to rise, but the cost is high





PART THREE "Difficulty" Computing power and interface bandwidth, which is the real challenge

Logic vs. Interface

Assumptions: 1 mm x 1mm area, assuming a 16-bit multiply-add unit (MAC) and register to be about 4K Gates

Question: In order to achieve a 100% multiplication efficiency of a 16 × 16 matrix multiplication unit under a 65-nm / 28-nm process at a 100-MHz operating frequency, what is the baud rate of the interface data rate?

answer:

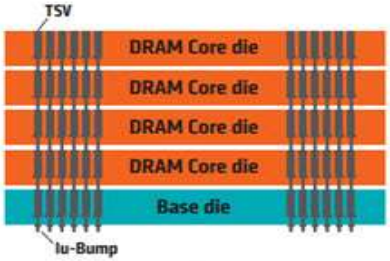
65nm

Can achieve 256 MAC / FF
> Requires about 50Gbps data input baud rate

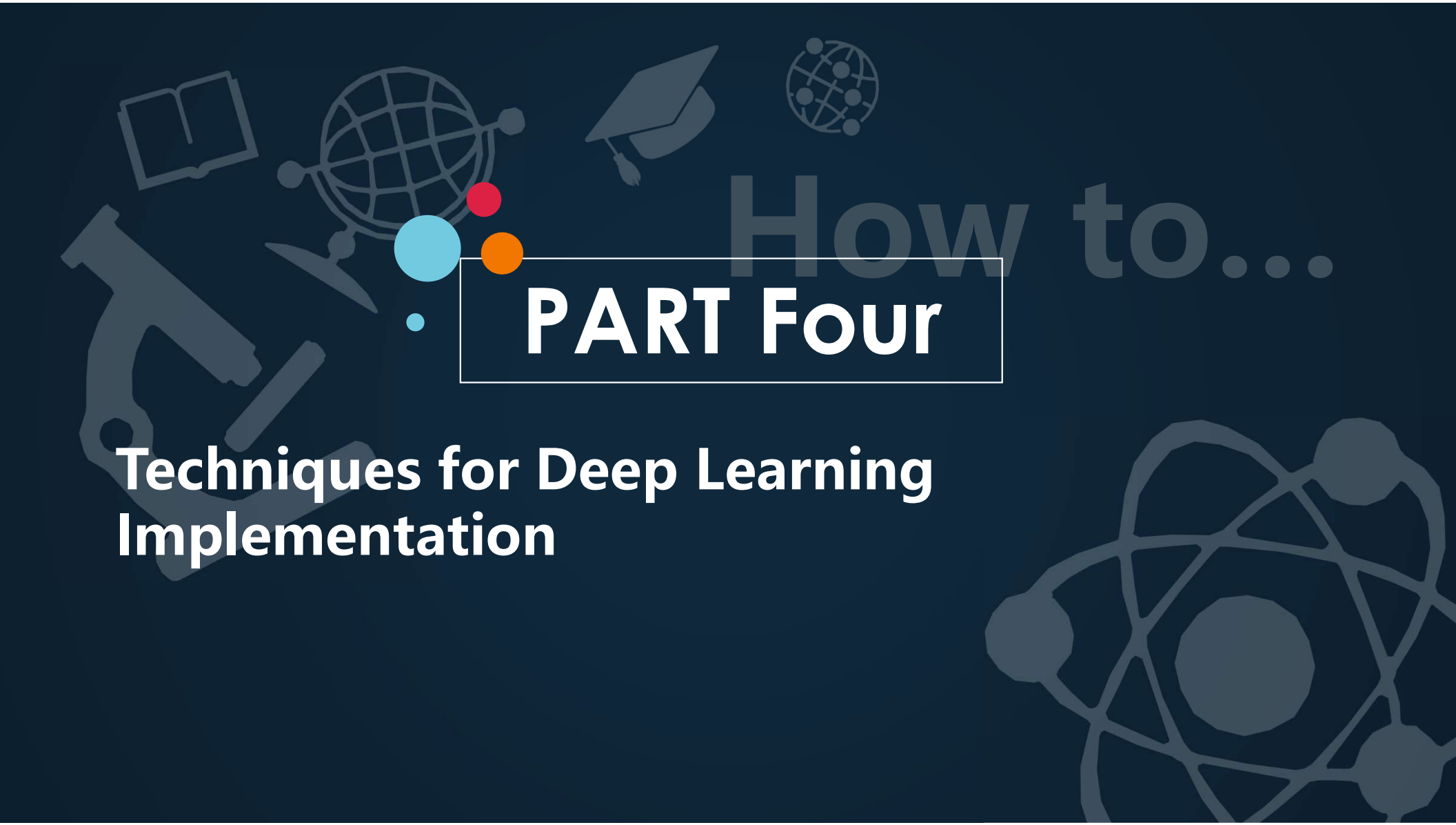
28nm

Can achieve 1024 MAC / FF
> Requires about 200 Gbps data input baud rate

The 1024-bit wide GDDR5 in the 3D-TSV stack package barely meets the demand!

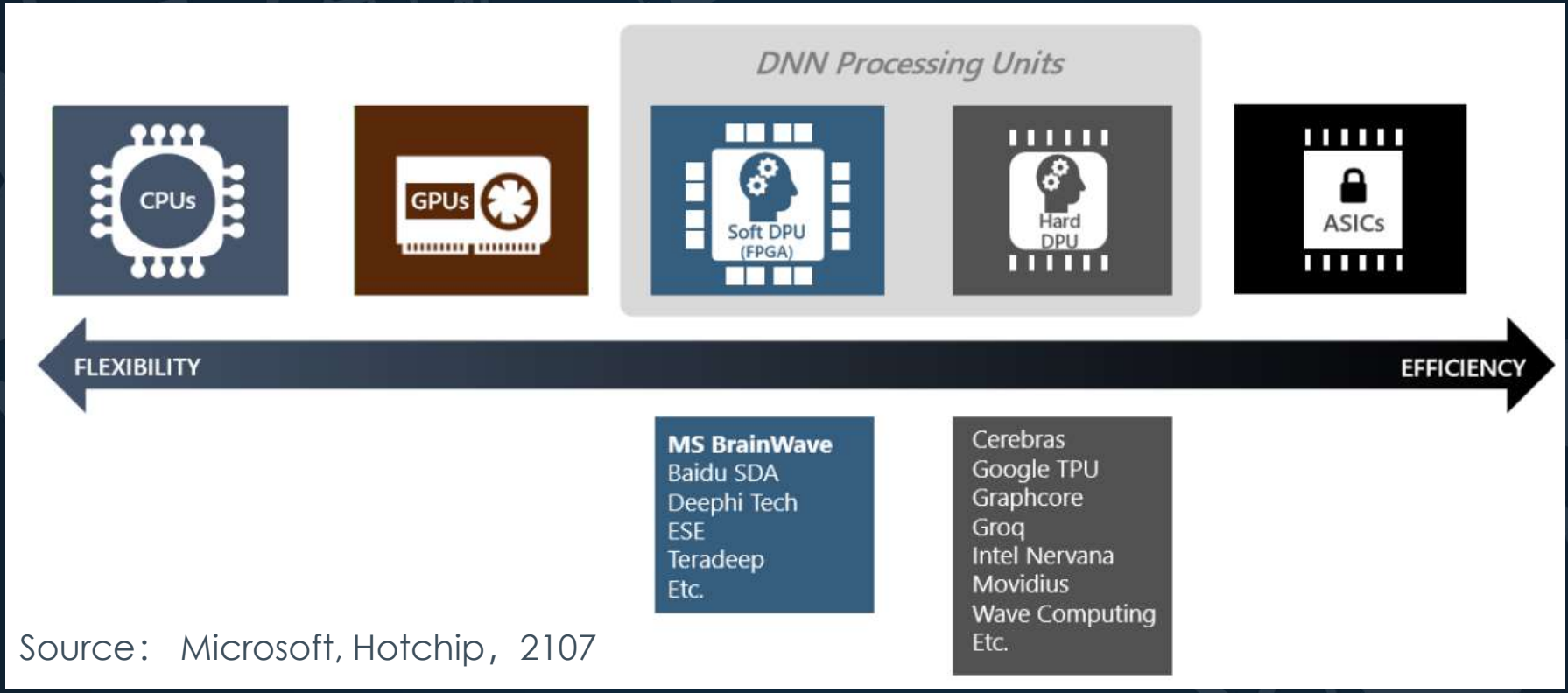


GDDR5	Per Package	HBM
32-bit	Bus Width	1024-bit
Up to 1750MHz (7GBps)	Clock Speed	Up to 500MHz (1GBps)
Up to 28GB/s per chip	Bandwidth	>100GB/s per stack
1.5V	Voltage	1.3V


How to…
PART Four
Techniques for Deep Learning Implementation

PART FOUR "How"

DL Hardware Solution

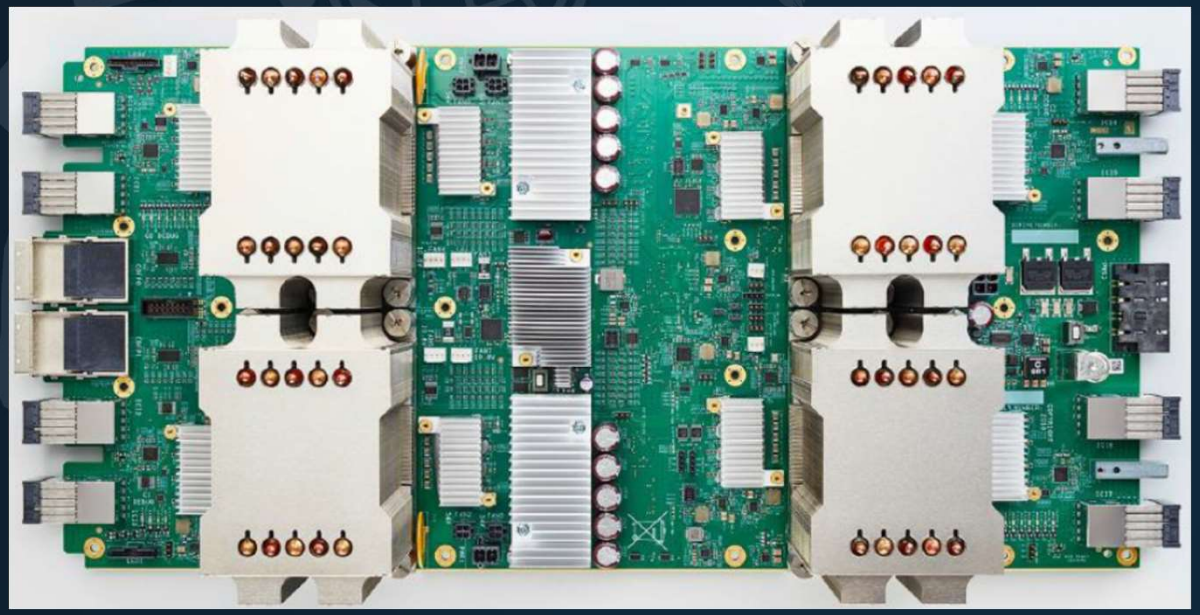


Source: Microsoft, Hotchip, 2107

DL Processor/Accelerator Option with flexibility and efficiency

PART FOUR “How”

DL processor Representation：TPU



1202 CPU

+176 GPU

VS.

4 TPU!

Google TPU2

AlphaGo Zero, trained on 40 TPU2 for 40 days beats AlphaGo which was trained with 1206 CPUs and 176 GPU over a few months and beated Lee Seung-shi, Ke-kie 。

PART FOUR "How"

DL Accleration

Reuse

Efficient Dataflow
Maximize reuse

Memory

Improve memory architecture
Reduce memory access

Precision

Less bits
Similar performance

Sparsity

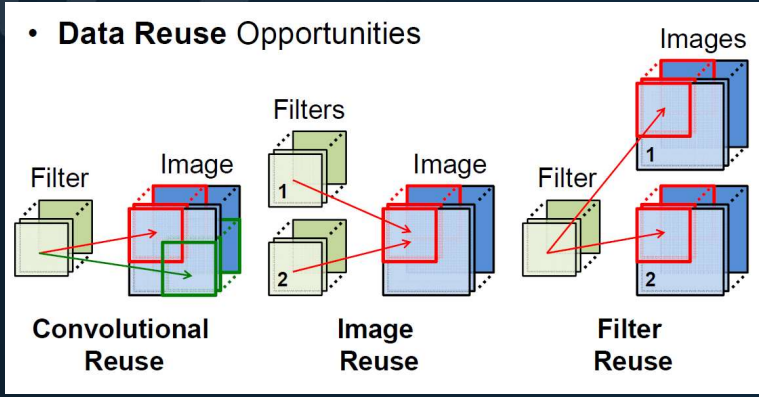
Utilize sparsity
Reduce unnecessary ops

PART FOUR "How"

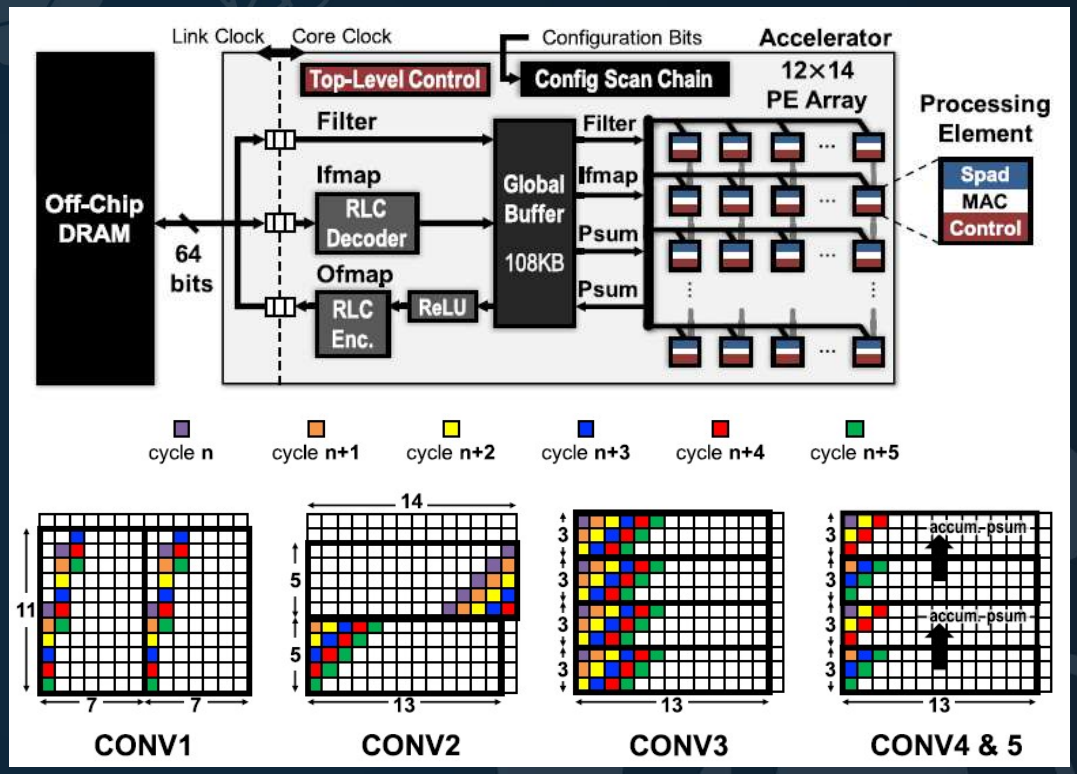
Aspect One: Reuse

MIT: Eyeriss

- Convolution reuse
- Image reuse and filter reuse



Three reuses in conv



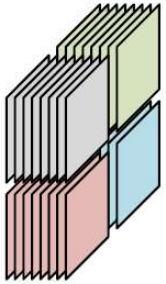
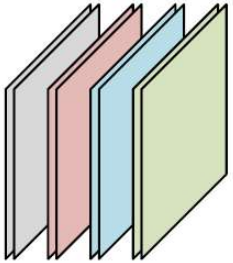
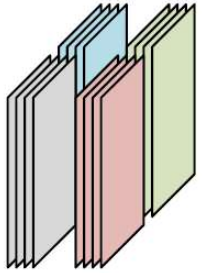
Efficient architecture and dataflow

Reduce off-chip DRAM access

PART FOUR "How"

Aspect One: Reuse

KAIST: DNPU

Input Layer Division Method	Image Division	Channel Division	Mixed Division
			
Off-chip Access (W/O Compression Scheme)			
Input Image	$W_i \times H_i \times C_i$	$W_i \times H_i \times C_i$	$W_i \times H_i \times C_i$
Weight	$W_f \times H_f \times C_i \times C_o$ x Img. Div. #	$W_f \times H_f \times C_i \times C_o$	$W_f \times H_f \times C_i \times C_o$ x Img. Div. #
Output Image	$\frac{W_o \times H_o \times C_o}{\text{Pooling Size}}$	$W_o \times H_o \times C_o$ x Ch. Div. # x 2	$W_o \times H_o \times C_o$ x Ch. Div. #

Maximize Division

Different division methods according to image size and filter size

> Image Division

> Channel Division

> Mixed Division

Maximize data reuse, reduce memory access

PART FOUR "How"

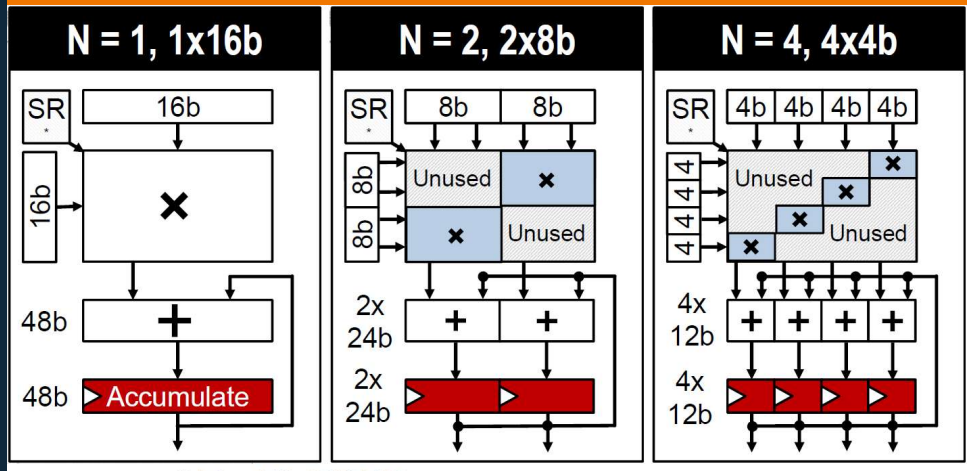
Aspect Two: Precision

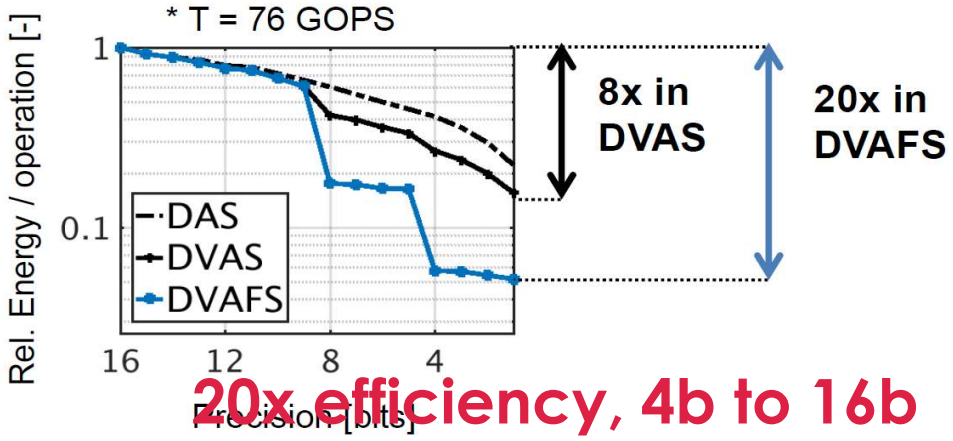
CONV-1	CONV-2	CONV-3	CONV-4
6 MMACs 22 kB 5-44%=0	12 MMACs 42 kB 8-45%=0	500 MMACs 742 kB 8-47%=0	15 GMACs 15 MB 5-82%=0
2-4b Ops	**3-4b Ops**	**4-6b Ops**	**4-6b Ops**
94 % acc.	96 % acc.	94 % acc.	92.5 % acc.
Always-on	~1% on	~0.1% on	~0.01% on

Increasing # Classes / Network Size / **FP precision** / Energy per frame

The effect of using a smaller bit width on computational accuracy is almost negligible, with 4-8b gaining most of the network performance

Leuven: Envision



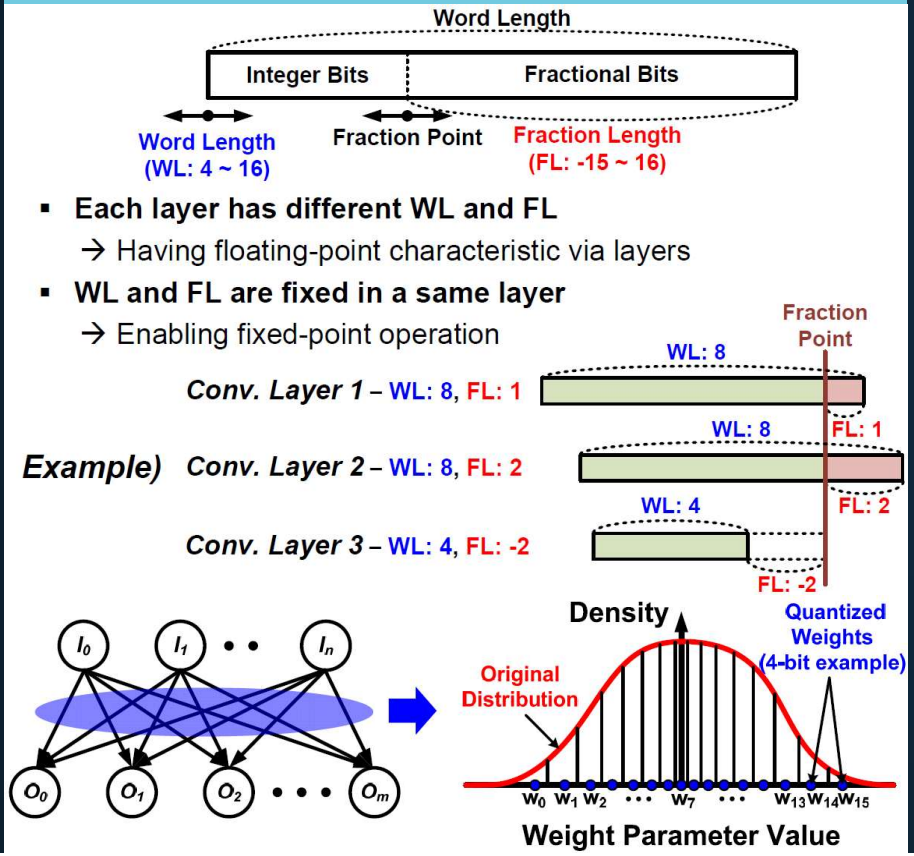


20x efficiency, 4b to 16b

PART FOUR "How"

Aspect Two: Precision

KAIST: DNPU



Dynamic point

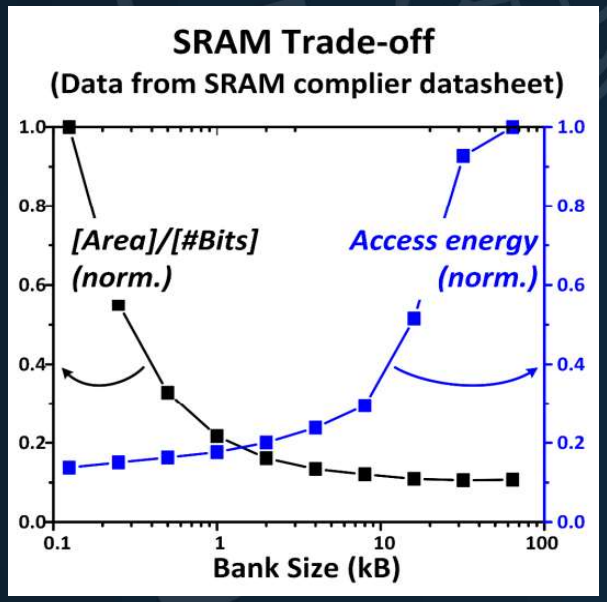
Layers and layers using different data bit width and decimal point position, with 4-8b finite bit width to obtain approximate accuracy of floating-point operations

Nonlinear Mapping

Using non-linear mapping method for parameter encoding, reducing the read bit width parameters

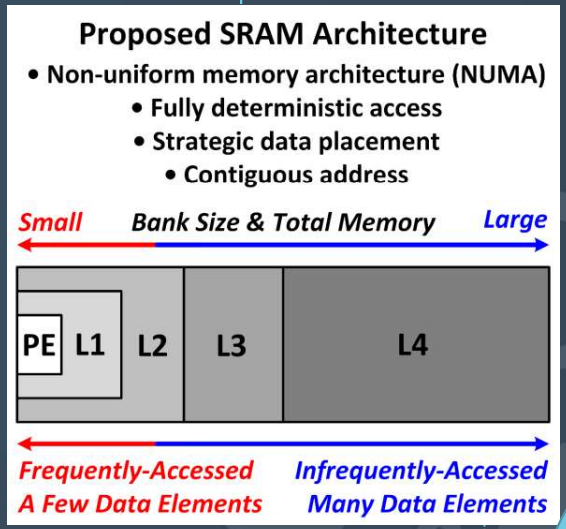
PART FOUR "How"

Aspect Three: Memory



Deep Learning Coprocessor Multilayer Caching Mechanism

Reduce memory access latency



UMICH: DLA

Improve memory architecture and efficiency

The data location is defined by the importance of the data

PART FOUR "How"

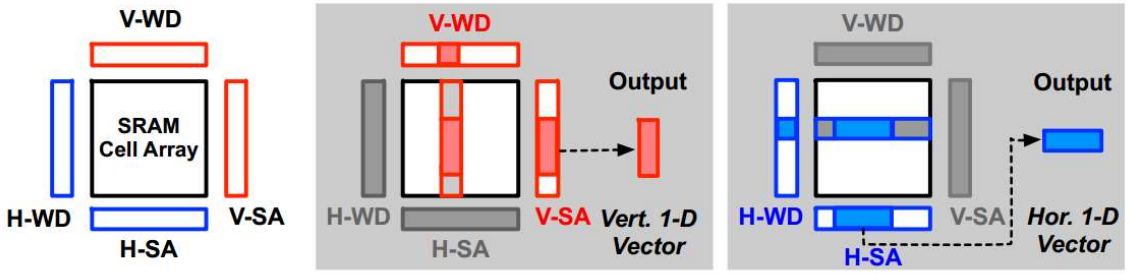
Aspect Three: Memory

KAIST:
Face Recognition

- Data can be read and written vertically or horizontally
- Eliminate the need for additional data in convolutional networks
- Facilitate the deconstruction of large convolution

Transpose-Read SRAM (T-SRAM)

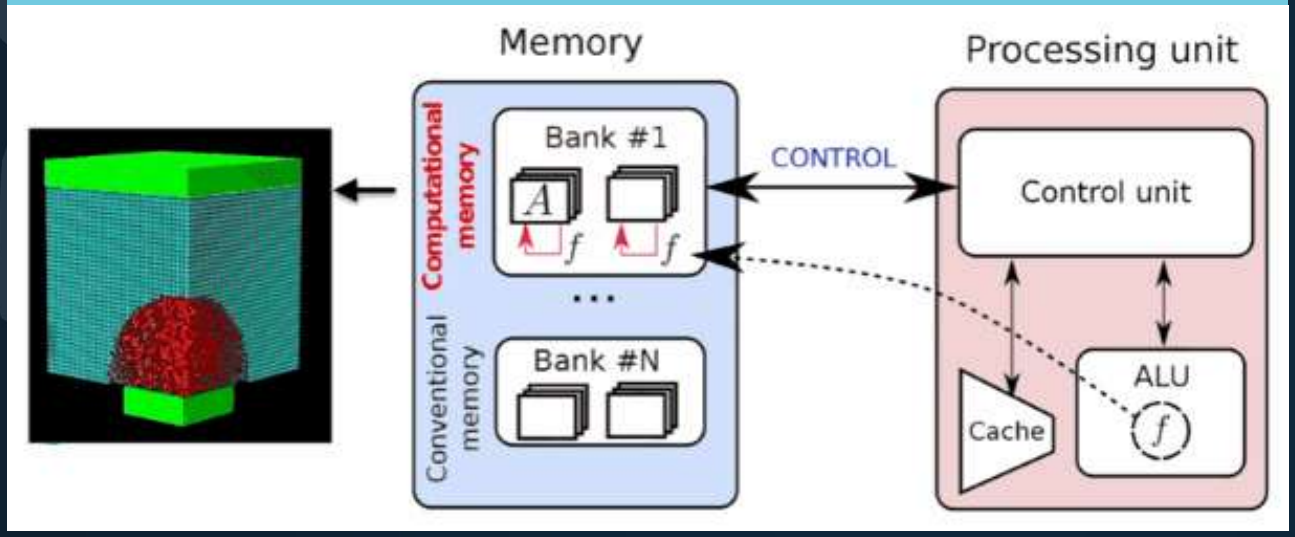
- ❑ V-WD & V-SA: *transposed* *read/write*
  - – Vertical wordline-driver & Sense amplifier
  - – Output vertical 1-D vector
- ❑ H-WD & H-SA: *conventional* *read/write*
  - – Output horizontal 1-D vector



PART FOUR "How"

Aspect Three: Memory

IBM: IMC (In memory computing)



Phase change storage and calculatoin

Adopting phase change memory as storage and calculation unit, a special computing hardware structure can be formed by integrating phase change memory directly with CPU, and the calculation and storage can occur on the same structure at the same time without additional transmission. Write action.

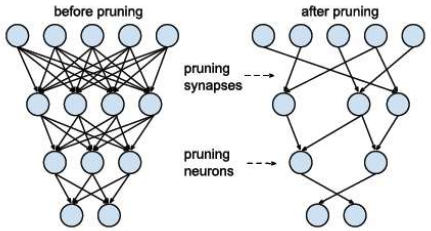
PART FOUR "How"

Aspect Four: Sparsity

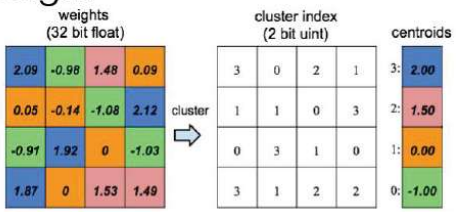
Stanford: EIE

Parameter and input of NN have high level sparsity

- **Network Pruning[1]**: 10x fewer weights



- **Weight Sharing[2]**: only 4-bits per remaining weight



[1]. Han et al. NIPS 2015
[2]. Han et al. ICLR 2016, best paper award

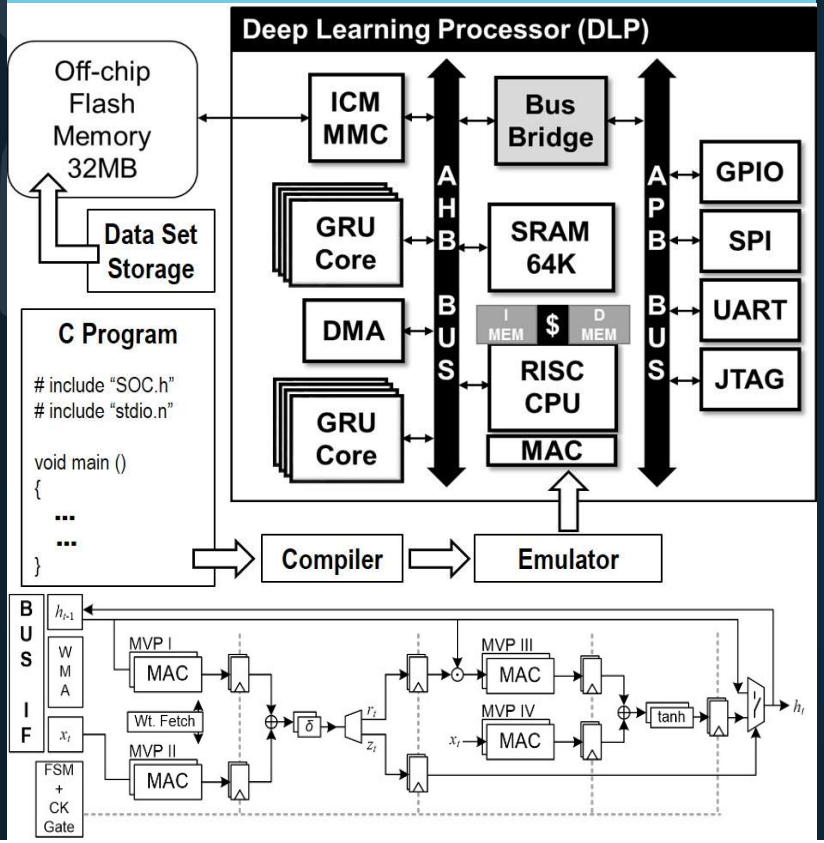
Layer	Size	Weight Density	Activation Density	FLOP %	Description
AlexNet-6	4096 × 9216	9%	35.1%	3%	AlexNet for image classification
AlexNet-7	4096 × 4096	9%	35.3%	3%	
AlexNet-8	1000 × 4096	25%	37.5%	10%	
VGG-6	4096 × 25088	4%	18.3%	1%	VGG-16 for image classification
VGG-7	4096 × 4096	4%	37.5%	2%	
VGG-8	1000 × 4096	23%	41.1%	9%	
NeuralTalk-We	600 × 4096	10%	100%	10%	RNN and LSTM for image caption
NeuralTalk-Wd	8791 × 600	11%	100%	11%	
NeuralTalk-LSTM	2400 × 1201	10%	100%	11%	

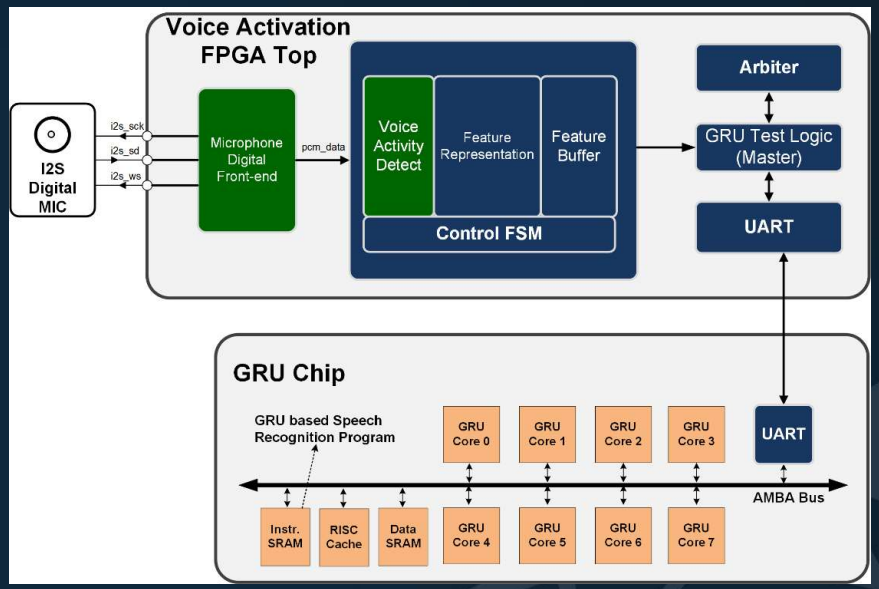
- Using the sparseness of neural networks to optimize methods such as pruning, the network size is reduced by 10x ~ 49x

PART FOUR "How"

Our OwnDesign One

OCEAN



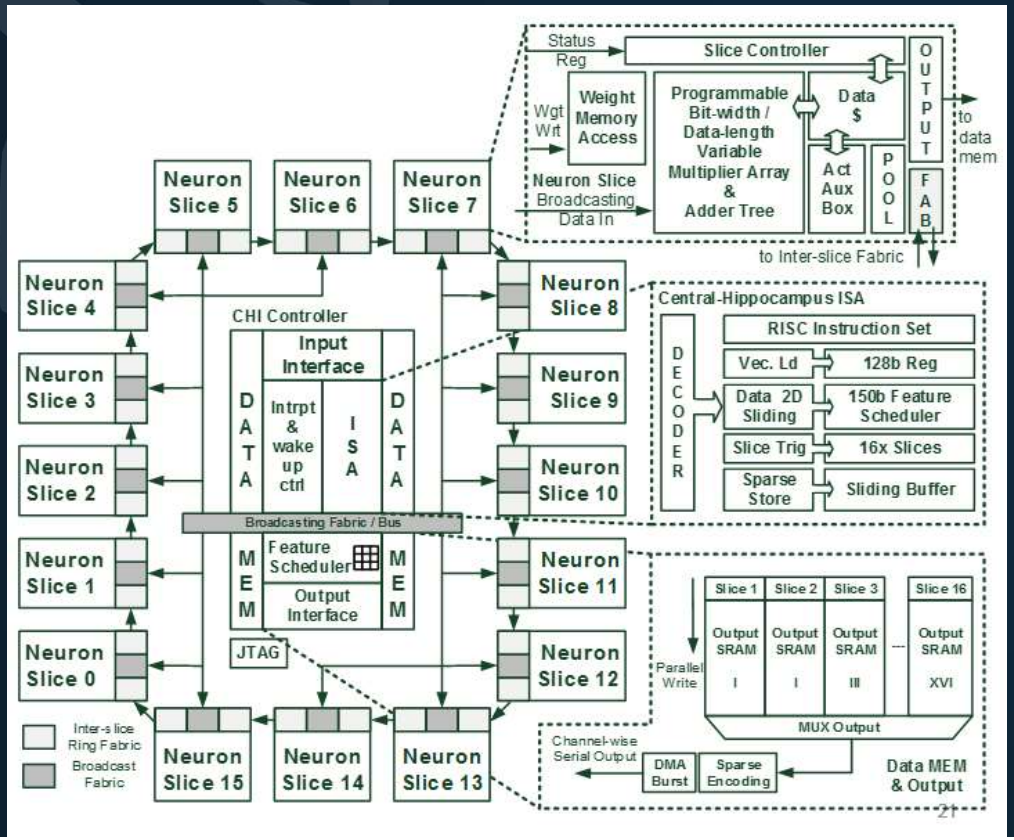


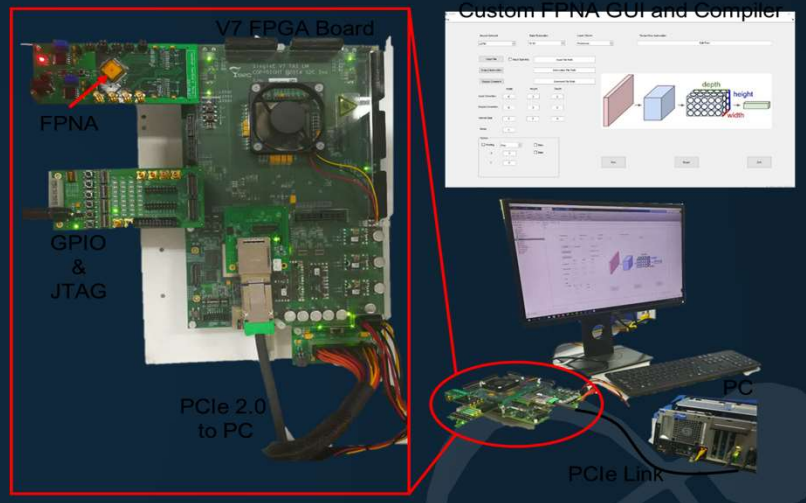
- Based on LSTM
- Successful taped out and tested
- 2017 ESSCIRC

PART FOUR "How"

Our Own Design Two

iFPNA





- Programmable Neuron Arrays
- Successful taped out and tested
- CNN/RNN embedded applications
- Power < 10mW

Summary

➔Use Deep Learning AcceleratorDesign As An Example to Show the need, the issues and techniques of embedded deep learning: Key Operation: Multiplication+Addition+Memory-Access

- ➔Smaller silicon + cheap modules (cost)
- ➔Lower power (5 orders of magnitude)
- ➔High throughputs (3 orders of magnitude)

➔AI-infused society is driving the need of embedded deep learning

- ➔Automotive; Consumer; Industry; Medical; Defense
- ➔Metaverse…

➔How to convert an algorithm to a chip

- ➔Describe the algorithm in a hardware description language (!!!): data flow
- ➔Quantization
- ➔Sparsity