2D Convolution Algorithms Summary

P. Giannakeris, P. Bassia, Prof. Ioannis Pitas
Aristotle University of Thessaloniki
pitas@csd.auth.gr
www.aiia.csd.auth.gr
Version 2.4

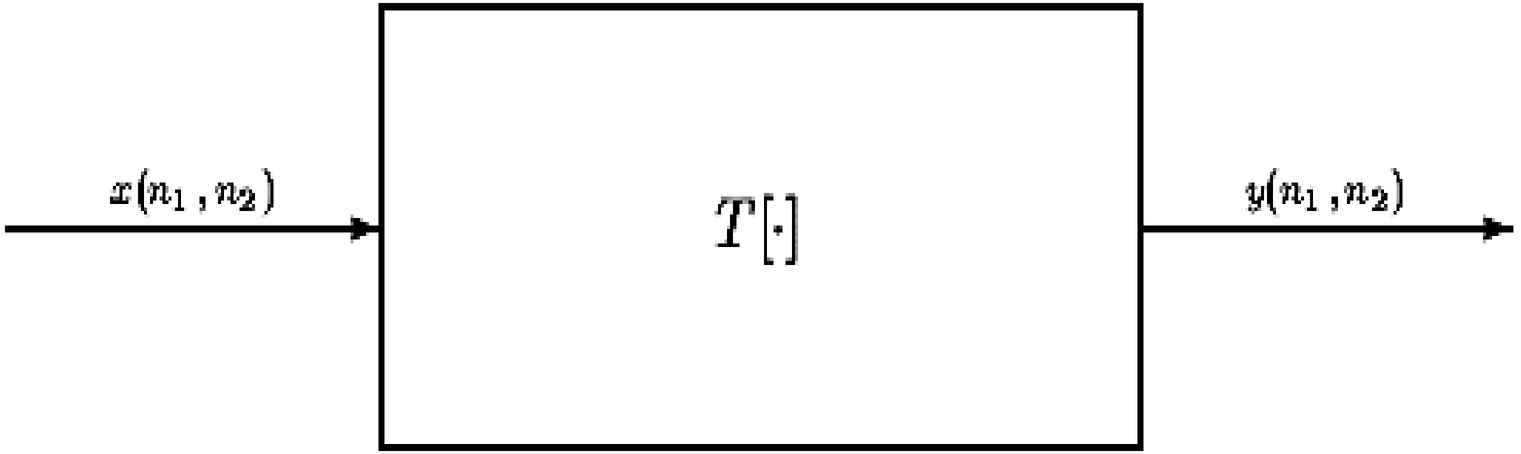
Artificial Intelligence &
Information Analysis Lab

Outline

- 2D linear systems
- 2D convolutions
  - Discrete-time 2D Systems
  - Linear & Cyclic 2D convolutions
  - 2D Discrete Fourier Transform, 2D Fast Fourier Transform
- Other convolution algorithms
  - Winograd algorithm
  - Block methods
- Applications in Machine Learning
  - Convolutional neural networks

2D Discrete-Time Systems

- Transformation of a 2D discrete-time input signal $x(n_1, n_2)$ into a 2D discrete-time output signal $y(n_1, n_2)$.



$$y(n_1, n_2) = T[x(n_1, n_2)]$$

2D Discrete-Time Systems - Properties

- Linearity

$$T[ax_1 + bx_2] = aT[x_1]+bT[x_2]$$

- Shift-Invariant

$$y(n_1, n_2) = T[x(n_1, n_2)] \Rightarrow$$

$$y(n_1 - m_1, n_2 - m_2) = T[x(n_1 - m_1, n_2 - m_2)]$$

2D Discrete-Time Systems - Properties

- A linear shift invariant system is described by a 2D convolution of input x with a convolutional kernel h:

$$y(k_1,k_2) = h(k_1,k_2) ** x(k_1,k_2) = \sum_{i_1}\sum_{i_2} h(i_1,i_2)x(k_1-i_1,k_2-i_2)$$

- Input x has typically limited region of support (size), e.g. it can be an image of $M_1 x M_2$ pixels.
- Convolutional kernel h may have limited or infinite region of support.

Types of 2D Discrete-Time Systems

- Finite impulse response (FIR):

  $h(n_1, n_2)$ is zero outside some filter mask (region) $N_1 \times N_2$, $0 \leq n_1 < N_1$, $0 \leq n_2 < N_2$.

- FIR filters are described by a 2D linear convolution with convolutional kernel h of size $N_1 \times N_2$ is given by:

$$y(k_1, k_2) = h(k_1, k_2) ** x(k_1, k_2) = \sum_{i_1}^{N_1} \sum_{i_2}^{N_2} h(i_1, i_2) x(k_1 - i_1, k_2 - i_2)$$

- Usually digital (discrete-time) systems without feedback are FIR.

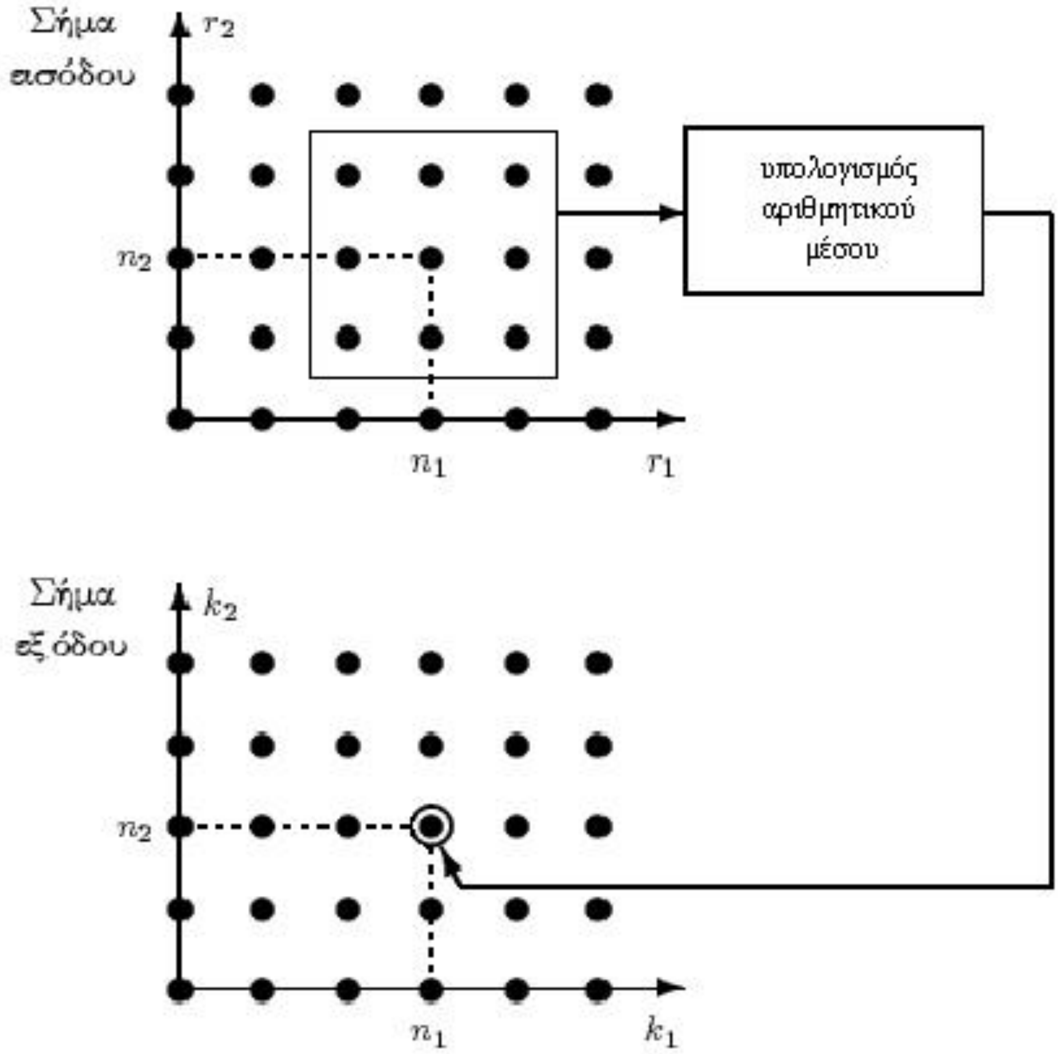
2D Discrete-Time System - Examples

- FIR: The moving average filter $N_1 \times N_2$,

N_i=2v_i+1:

$$y(n_1,n_2) = \frac{1}{N_1 N_2}\sum_{k_1=-v_1}^{v_1}\sum_{k_2=-v_2}^{v_2} x(n_1-k_1,n_2-k_2)$$

- 3x3 moving average filter.



2D Discrete-Time System – Moving Average

Types of 2D Discrete-Time Systems

- Infinite impulse response (IIR) when $h(n_1,n_2)$ has infinite region of support, e.g., if it never becomes zero outside some finite region.

- IIR filters are described by difference equations

$$\sum_{k_1}\sum_{k_2} b(k_1,k_2)y(n_1-k_1,n_2-k_2)=\sum_{r_1}\sum_{r_2} a(r_1,r_2)x(n_1-r_1,n_2-r_2)$$

2D Discrete-Time System - Examples

- IIR Edge Detector output

Artificial Intelligence & Information Analysis Lab

2D linear correlation

- Correlation of template image h and input image x (inner product):

$$r_{hx}(m_1, m_2) = \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} h(n_1, n_2)\, x(n_1 + m_1, n_2 + m_2) = \boldsymbol{h}^T \boldsymbol{x}(n_1, n_2).$$

- $\boldsymbol{h} = [h(0,0), \ldots, h(N_1 - 1, N_2 - 1)]^T$: template image vector.
- $\boldsymbol{x}(k) = [x(n_1, n_2),\ \ldots,\ x(n_1 + N_1 - 1, n_2 + N_2 - 1)]^{\mathrm{T}}$: local image vector.

2D linear correlation

Differences from convolution:

- $x(n_1, n_2)$ is not flipped around $(0,0)$.
- **It is often confused with convolution**: they are identical only if h is centered at and is symmetric about $(0,0)$.
- It is used for 2D template matching and for object tracking in video.

Image autocorrelation:

$$r_{xx}(m_1, m_2) = \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} x(n_1, n_2)\, x(n_1 + m_1, n_2 + m_2).$$

Linear and cyclic 2D convolutions

- Two-dimensional linear convolution with convolutional kernel h of size $N_1 \times N_2$ is given by:

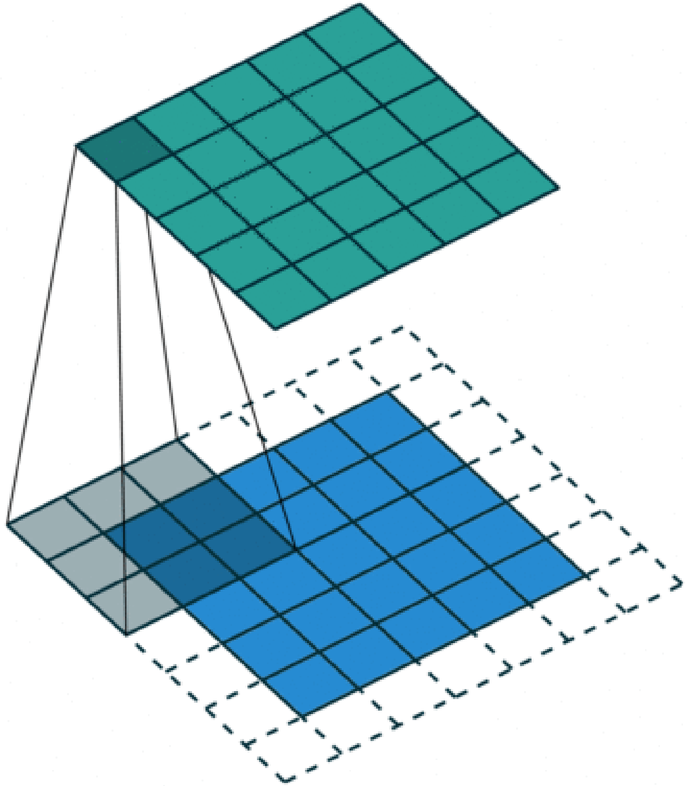
$$y(k_1, k_2) = h(k_1, k_2) ** x(k_1, k_2) = \sum_{i_1}^{N_1} \sum_{i_2}^{N_2} h(i_1, i_2) x(k_1 - i_1, k_2 - i_2)$$

- Its two-dimensional cyclic convolution counterpart of support $N_1 \times N_2$ is defined as:

$$y(k_1, k_2) = h(k_1, k_2) \circledast\circledast x(k_1, k_2) = \sum_{i_1}^{N_1} \sum_{i_2}^{N_2} h(i_1, i_2) x((k_1 - i_1)_{N_1}, (k_2 - i_2)_{N_2})$$

2D Convolution - Example

- With Padding



2D Convolution - Example

- No Padding

$$\mathbf{I} = \begin{array}{|c|c|c|c|c|c|c|}\hline 0 & 1 & 1 & 1_{\times 1} & 0_{\times 0} & 0_{\times 1} & 0 \\ \hline 0 & 0 & 1 & 1_{\times 0} & 1_{\times 1} & 0_{\times 0} & 0 \\ \hline 0 & 0 & 0 & 1_{\times 1} & 1_{\times 0} & 1_{\times 1} & 0 \\ \hline 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ \hline 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ \hline 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ \hline 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ \hline \end{array} \quad * \quad \mathbf{K} = \begin{array}{|c|c|c|}\hline 1 & 0 & 1 \\ \hline 0 & 1 & 0 \\ \hline 1 & 0 & 1 \\ \hline \end{array} \quad = \quad \mathbf{I} * \mathbf{K} = \begin{array}{|c|c|c|c|c|}\hline 1 & 4 & 3 & \boxed{4} & 1 \\ \hline 1 & 2 & 4 & 3 & 3 \\ \hline 1 & 2 & 3 & 4 & 1 \\ \hline 1 & 3 & 3 & 1 & 1 \\ \hline 3 & 3 & 1 & 1 & 0 \\ \hline \end{array}$$

Linear and cyclic 2D convolutions

- A 2D linear convolution of convolutional kernel h of size $M_1 \times M_2$ operating on an image x of size $N_1 \times N_2$ of size produces an output image y:
  - of size $M_1 M_2$using zero padding
    - **Complexity:** $N_1 N_2 M_1 M_2$ multiplications.
  - of size $(N_1 - M_1 + 1)\,(N_2 - M_2 + 1)$, without input image border padding.
    - **Complexity**: $(N_1 - M_1 + 1)\,(N_2 - M_2 + 1)\,M_1 M_2$ multiplications.
- In both cases complexity is $O(N^4)$, if N_1, N_2, M_1, M_2 are of order N.

Artificial Intelligence & Information Analysis Lab

2D Discrete Fourier Transform (DFT) - Notation

$$X(k_1,k_2) = \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} x(n_1,n_2) W_{N_1}^{n_1 k_1} W_{N_2}^{n_2 k_2}$$

$$x(n_1,n_2) = \frac{1}{N_1 N_2} \sum_{k_1=0}^{N_1-1} \sum_{k_2=0}^{N_2-1} X(k_1,k_2)\, W_{N_1}^{-n_1 k_1} W_{N_2}^{-n_2 k_2}$$

$$W_{N_i} = \exp\left(-i\frac{2\pi}{N_i}\right), \qquad i = 1,2$$

2D Cyclic Convolution Calculation with DFT

$$y(k_1,k_2) = h(k_1,k_2) \circledast\circledast x(k_1,k_2) = \sum_{i_1}^{N_1}\sum_{i_2}^{N_2} h(i_1,i_2)x((k_1-i_1)_{N_1},(k_2-i_2)_{N_2})$$

1. Calculate the 2D DFTs of the 2D matrices $\mathbf{h}$ and $\mathbf{x}$:

$$\mathbf{H} = DFT\{\mathbf{h}\}, \mathbf{X} = DFT\{\mathbf{x}\}$$

2. Multiply 2D matrices matrices $\mathbf{H}$ and $\mathbf{X}$ elementwise in the frequency domain:

$$\mathbf{Y} = \mathbf{H} \odot \mathbf{X}$$

3. Calculate the IDFT of $\mathbf{Y}$ to get $\mathbf{y}$:

$$\mathbf{y} = IDFT\{\mathbf{Y}\}$$

Linear Convolution with DFT

- We have:

$x(n_1, n_2)$ in the region $R_{P_1P_2} = [0, P_1) \times [0, P_2)$

$h(n_1, n_2)$ in the region $R_{Q_1Q_2} = [0, Q_1) \times [0, Q_2)$

- The linear convolution is in the region

$$R_{L_1L_2} = [0, L_1) \times [0, L_2) L_i = P_i + Q_i - 1, \qquad i = 1,2$$

1. We need to pad the signals with 0s, choosing $N_1, N_2, Ni \geq Pi + Qi - 1, i = 1,2$

$$x_p(n_1, n_2) = \begin{cases} x(n_1, n_2) & (n_1, n_2) \in R_{P_1P_2} \\ 0 & (n_1, n_2) \in R_{N_1N_2} - R_{P_1P_2} \end{cases}$$

$$h_p(n_1, n_2) = \begin{cases} h(n_1, n_2) & (n_1, n_2) \in R_{Q_1Q_2} \\ 0 & (n_1, n_2) \in R_{N_1N_2} - R_{Q_1Q_2} \end{cases}$$

Linear Convolution with DFT

2. Compute the DFTs of $x_p(n_1, n_2)$ and $h_p(n_1, n_2)$ that have a length of $N_1 \times N_2$

3. Compute $Y_p(k_1, k_2)$ as the product of $X_p(k_1, k_2)$ and $H_p(k_1, k_2)$

4. Compute $y_p(n_1, n_2)$ as the IDFT of $Y_p(k_1, k_2)$

5. The result is the region $[0, L_1) \times [0, L_2)$ of $y_p(n_1, n_2)$.

Row-Column FFT

- Sequential 1D FFTs are computed over rows and columns

$$G(n_1,k_2)=\sum_{n_2=0}^{N_2-1} x(n_1,n_2)W_{N_2}^{n_2k_2}$$

$$X(k_1,k_2)=\sum_{n_1=0}^{N_1-1} G(n_1,k_2)W_{N_1}^{n_1k_1}$$

- The number of complex multiplications are:

$$C=N_1\frac{N_2}{2}\log_2 N_2+N_2\frac{N_1}{2}\log_2 N_1=\frac{N_1N_2}{2}\log_2(N_1N_2)$$

Which is $O(N^2\,log_2N)$

Artificial Intelligence & Information Analysis Lab

Convolutions using FFT

- Memory requirements are x8 for direct computation and x16 using the FFT.
- Direct approach is faster for a small filter $K_1 \times K_2$ when:

$$K_1 K_2 < 6\log_2(N_1 N_2) + 4$$

- For large filters close to the image size:

Direct has $O(N^4)$

Using FFT has $\sim O(N^2 \log_2 N)$

Block-based convolution calculation

- Block-based methods:

The *2D overlap-add* is based on the distributive property of convolution:

- An image $x(i_1, i_2)$ can be divided into $K_1 \times K_2$ non-overlapping subsequences, having dimensions $N_{B1} \times N_{B2}$, each:

$$x_{k_1k_2}(i_1, i_2) = \begin{cases} x(i_1, i_2) & k_1 N_{B1} \le i_1 < (k_1+1)N_{B1},\ k_2 N_{B2} \le i_2 < (k_2+1)N_{B2} \\ 0 & \text{otherwise} \end{cases}$$

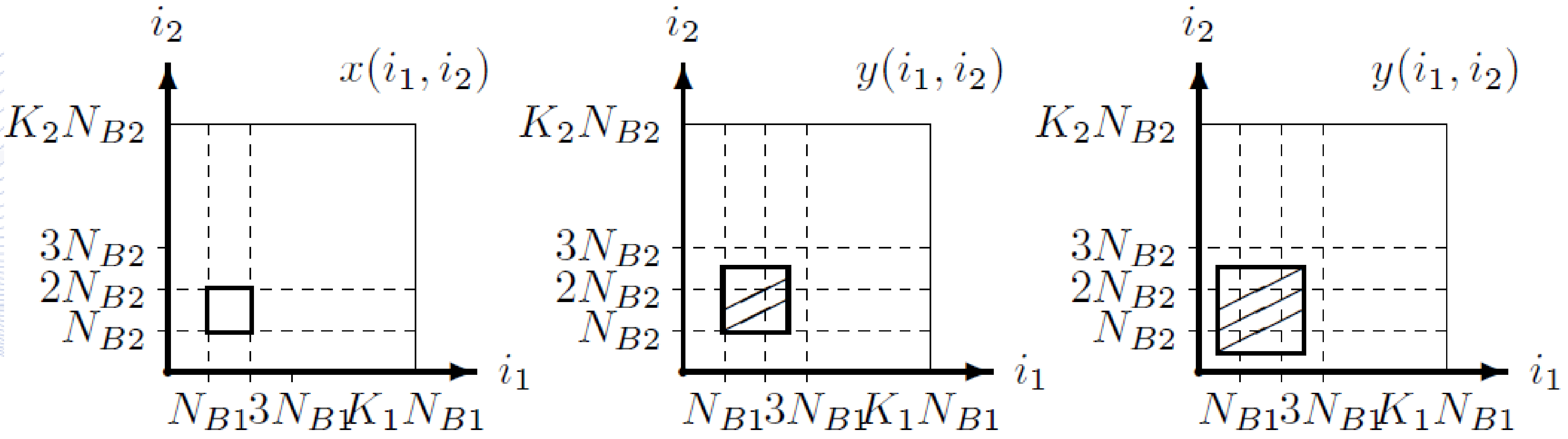
- The linear convolution output $y(n_1, n_2)$ is the sum of the (easily parallelizable) convolution outputs produced by the input sequence blocks:

$$y(i_1, i_2) = x(i_1, i_2) ** h(i_1, i_2) = \sum_{k_1=1}^{K_1} \sum_{k_2=1}^{K_2} \left(x_{k_1k_2}(i_1, i_2)\right) ** h(1_1, 1_2) = \sum_{k_1=1}^{K_1} \sum_{k_2=1}^{K_2} y_{k_1k_2}(i_1, i_2)$$

Overlap-add

The 'partial' convolutions are performed using FFT and then adding the results:

- The blocks and the filter are transformed to the frequency domain.
- Partial output blocks are calculated using the IFFT of the product as usual.
- Then all the <u>overlapping</u> blocks are added to construct the final output image.



Artificial Intelligence & Information Analysis Lab

Overlap-save

- Block-based methods:

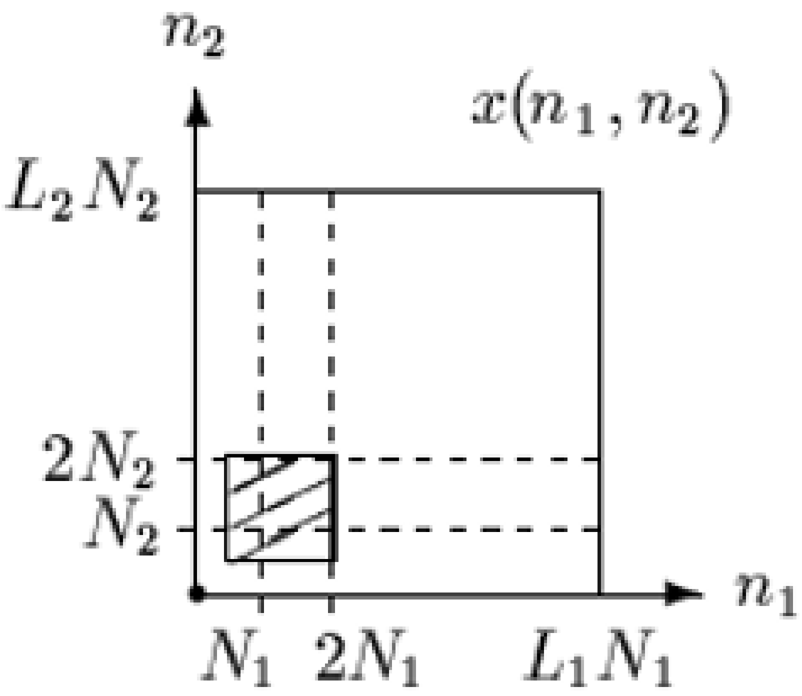
The *2D overlap-save* convolution:

- The output is divided into non overlapping boxes of size $N_1 \times N_2$

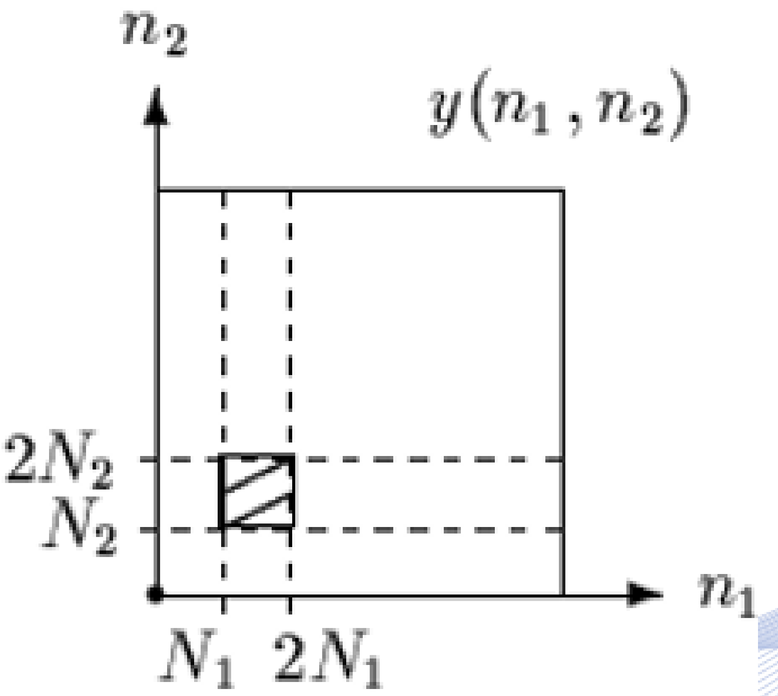
$$y(n_1 n_2) = \sum_{i=1}^{K_1} \sum_{j=1}^{K_2} y_{ij}(n_1 n_2)$$

- To calculate $y_{ij}(n_1, n_2)$:
  - Extension of $x_{ij}(n_1, n_2)$ of size $N_1 \times N_2$ to $x'_{ij}(n_1, n_2)$ of size $(N_1 + M_1 - 1) \times (N_2 + M_2 - 1)$.
  - Calculation of the Cyclic convolution between x'_{ij} and h with a DFT of size $(N_1 + M_1 - 1) \times (N_2 + M_2 - 1)$.
  - Every x_{ij} item is non zero, therefore only the inner $N_1 \times N_2$ is correct.
  - Addition all the 'trimmed' boxes to get the output.

Overlap-save



(β)

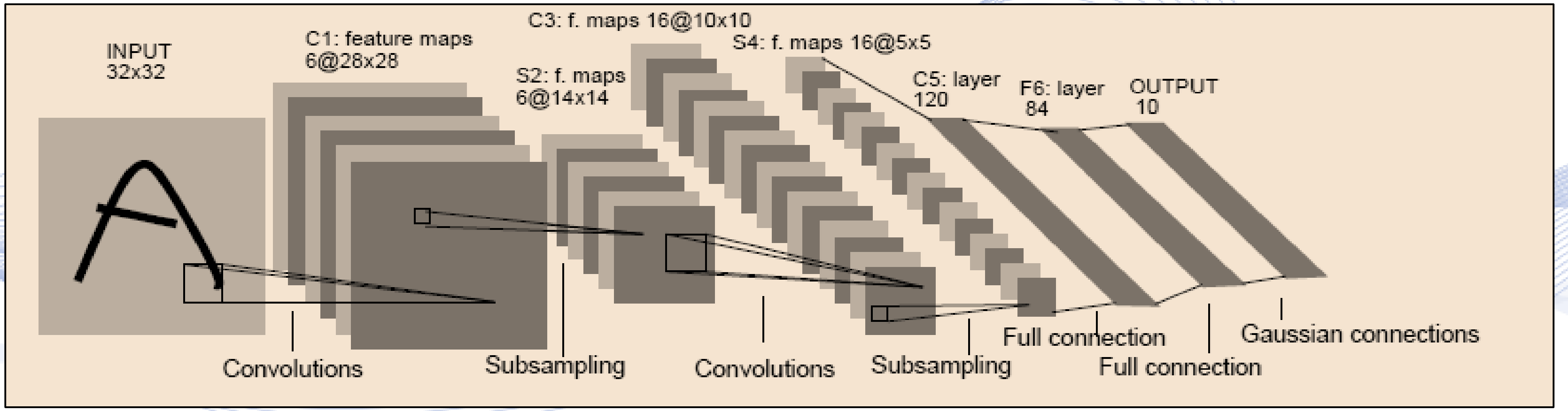


(γ)

Convolutional Neural Networks

Convergence of machine learning and signal processing processing

- Two step architecture:
  - First layers with sparse NN connections: convolutions.
  - Fully connected final layers.
- Need for fast convolution calculations.



Convolutional Neural Networks

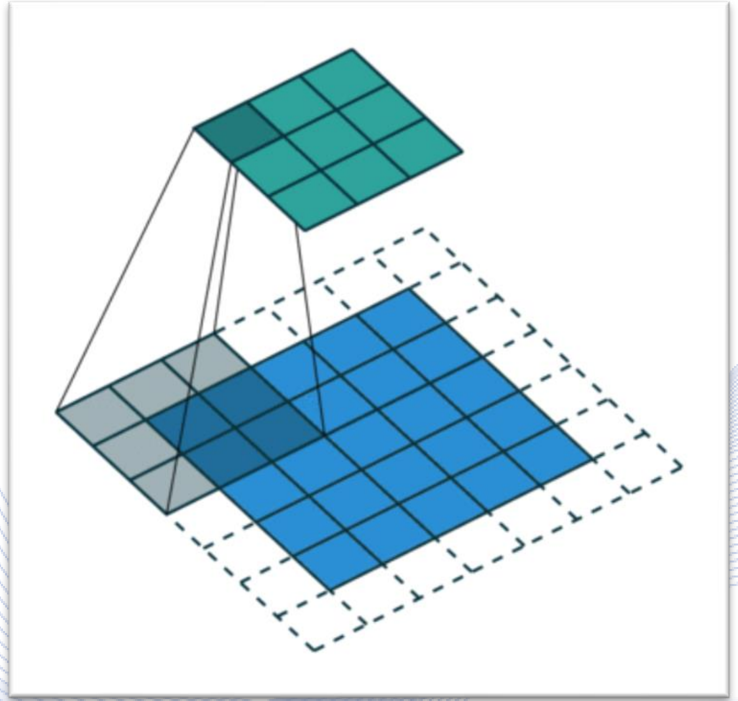
Characteristics: Local Receptive Fields

- The convolution kernel is given by the 4D tensor:

$\boldsymbol{W} = [w_{k_1,k_2,r,o}: \quad k_1 = 1, \ldots, h_1, k_2 = 1, \ldots, h_2,$

$r = 1, \ldots, d_{in}, o = 1, \ldots, d_{out}]$: $\boldsymbol{W} \in \mathbb{R}^{h_1 \times h_2 \times d_{in} \times d_{out}}$.

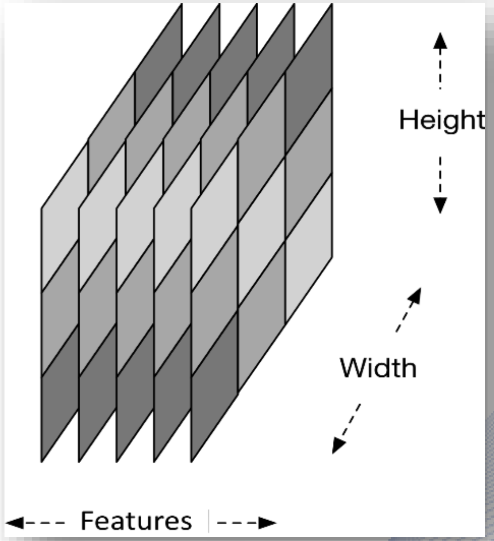
- For specific r, o, the $h_1 \times h_2$ *convolution filters* $\boldsymbol{W}(r, o)$ contain the synaptic weights for the $h_1 \times h_2$ neuron receptive field.



Convolutional Layer

For RGB images

- For a convolutional layer l with an activation function $f_l(\cdot)$, multiple incoming features d_{in} and one single output feature o.



Multiple input features to single feature o transformation

$$y^{(l)}(i,j,o) = f_l\left(b^{(l)} + \sum_{r=1}^{d_{in}} \sum_{k_1=-q_1}^{q_1^{(l)}} \sum_{k_2=-q_2}^{q_2^{(l)}} w^{(l)}(k_1,k_2,r,o)\, x^{(l)}(i-k_1, j-k_2, r)\right)$$

Convolutional Layer Activation Volume (3D tensor)

$$a_{ij}^{(l)}(o) = f_l\left(b^{(l)}(o) + \sum_{r=1}^{d_{in}} \boldsymbol{W}^{(l)}(r,o) * \boldsymbol{X}_{ij}^{(l)}(r)\right) \quad \boldsymbol{A}^{(l)} = [a_{ij}^{(l)}(o)\colon i = 1,..,n^{(l)}, j = 1,..,m^{(l)}, o = 1,\dots,d_{out}]$$

where $\boldsymbol{A}^{(l)}$ is the activation volume for the convolutional layer l, $\boldsymbol{W}^{(l)}(r,o)$ is a 2D slice of the convolutional kernel $\boldsymbol{W}^{(l)} \in \mathbb{R}^{h_1 \times h_2 \times d_{in} \times d_{out}}$ for input feature r and output feature o , $b^{(l)}(o)$ a scalar bias and $\boldsymbol{X}_{ij}^{(l)}(r)$ a region of input feature r centered at $[i,j]^T$, e.g. $\boldsymbol{X}^{(1)}(1)$ the R channel of an image $d_{in} = C = 3$.

Deep Learning Frameworks

Framework	User Interface	Data Parallelism	Model Parallelism
Caffe	protobuf, C++, Python	Yes	Limited
CNTK	BrainScript, C++, C#	Yes	No
TensorFlow	Python, C++	Yes	Yes
Theano	Python	No	No
Torch	LuaJIT	Yes	Yes

Image Source: Heehoon Kim, Hyoungwook Nam, Wookeun Jung, and Jaejin Le - Performance Analysis of CNN Frameworks for GPUs

Deep Learning Frameworks

- All 5 frameworks work with cuDNN as backend.
- cuDNN unfortunately not open source
- cuDNN supports FFT and Winograd

Framework	User Selectable	Heuristic-based	Profile-based	Default
Caffe	No	Yes	No	Heuristic-based
CNTK	No	No	Yes	Profile-based
TensorFlow	No	No	No	Heuristic-based†
Theano	**Yes**	Yes	Yes	GEMM
Torch	**Yes**	Yes	Yes	GEMM

†TensorFlow uses its own heuristic algorithm

Image Source: Heehoon Kim, Hyoungwook Nam, Wookeun Jung, and Jaejin Le - Performance Analysis of CNN Frameworks for GPUs

Q & A

Thank you very much for your attention!

Contact: Prof. I. Pitas
pitas@csd.auth.gr