Neural Network and Deep Learning

Convolutional Neural Network (CNN)

YANIKA KONGSOROT

Outline

- Introduction to CNNs
- Why Use CNNs?
- CNN Architecture Overview
- Main Components

Introduction to CNNs

Definition:

- A Convolutional Neural Network (CNN) is a type of deep learning model specifically designed for processing structured grid data, such as images.
- It automatically and adaptively learns **spatial** hierarchies of features from input data.
  - **Spatial features** refer to the patterns, structures, and relationships within data that capture how elements are arranged in space.
  - In the context of images, spatial features describe the visual characteristics that reflect the spatial organization of pixels, such as edges, textures, shapes, and object boundaries.

Introduction to CNNs

History of CNNs:

- The foundation of CNNs was laid by Kunihiko Fukushima, who introduced the **Neocognitron** in 1980, which utilized hierarchical layers of neurons to recognize patterns and was among the first to use convolutional layers.
- Yann LeCun and his collaborators developed **LeNet-5**, a pioneering CNN architecture for handwritten digit recognition that demonstrated the effectiveness of CNNs in image processing through its use of convolutional, pooling, and fully connected layers.
- ***The resurgence of deep learning in the early 2010s was fueled by the availability of large datasets and powerful GPUs, making it possible to train deeper CNN architectures effectively.***
- Alex Krizhevsky, along with Ilya Sutskever and Geoffrey Hinton, introduced **AlexNet** in 2012, which won the ImageNet Challenge and significantly improved image classification performance through its deep architecture with multiple convolutional layers, ReLU activation functions, and dropout for regularization.

Why Use CNNs?

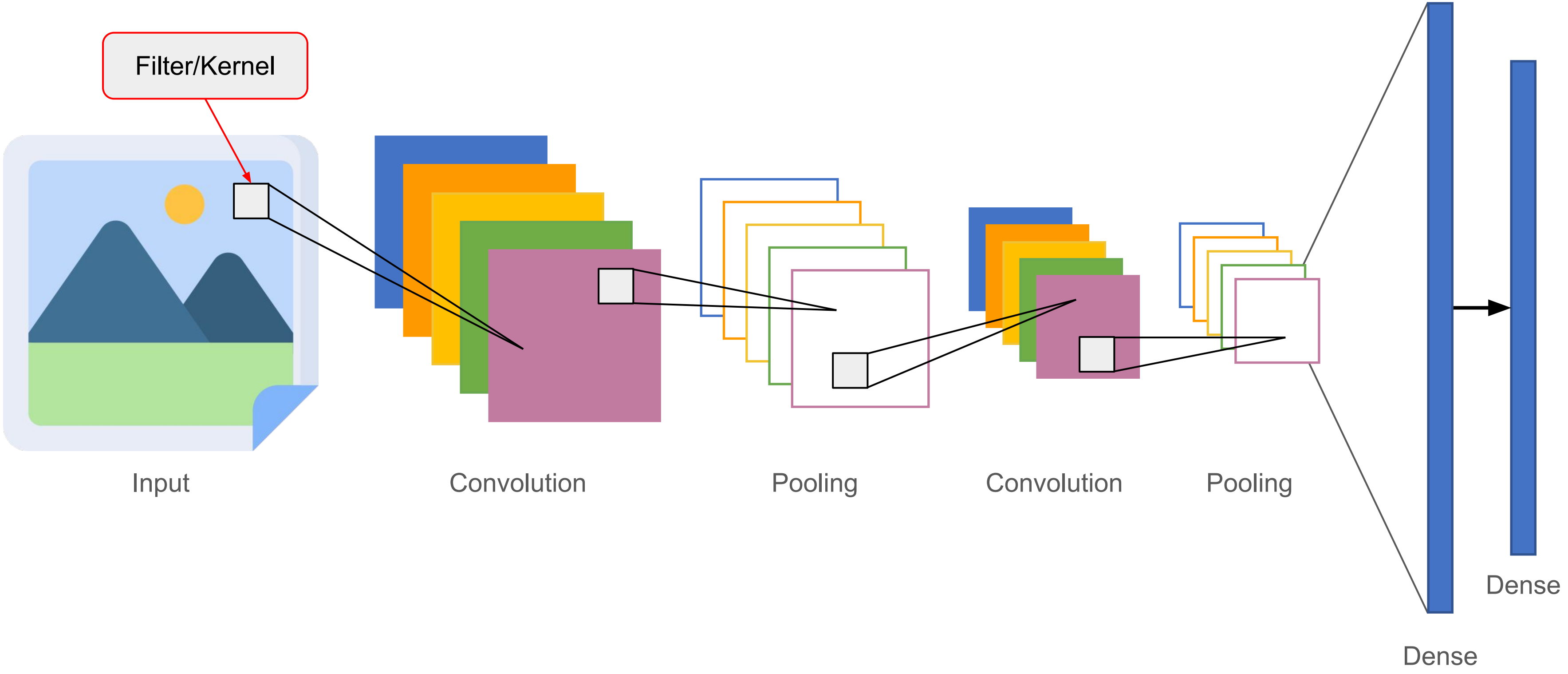
Challenges with Traditional Neural Networks:

- **High Dimensionality:** Images are often high-dimensional, leading to a vast number of parameters in fully connected layers, which increases computational cost and the risk of overfitting.
- **Loss of Spatial Information:** Traditional networks do not leverage the spatial structure of images (e.g., the proximity of pixels).

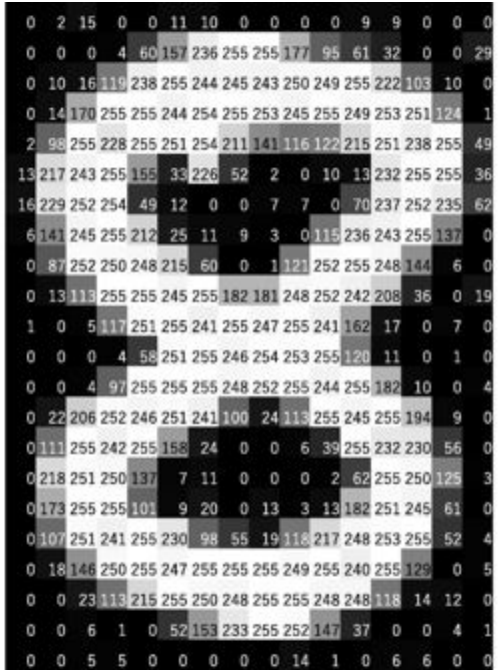
How CNNs Address These Challenges:

- **Parameter Sharing:** CNNs use convolutional layers with shared weights (filters/kernels), drastically reducing the number of parameters compared to fully connected networks.
- **Local Receptive Fields:** CNNs focus on local regions of the input, preserving spatial relationships and enabling the model to learn local patterns like edges, textures, and shapes.
- **Hierarchical Feature Learning:** Through multiple layers, CNNs learn to recognize increasingly complex features, from simple edges to entire objects.

CNN Architecture Overview



Main Components

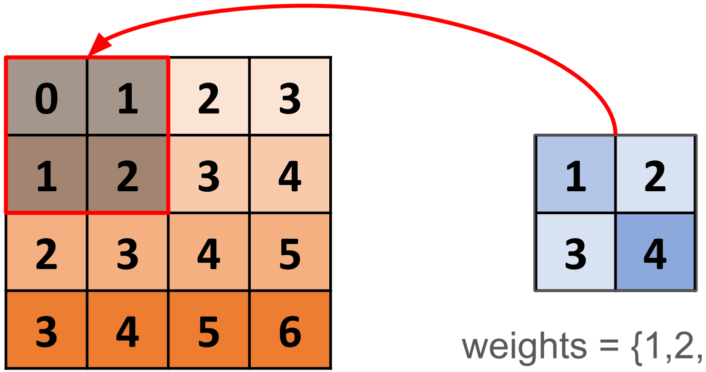


0	2	15	0	0	11	10	0	0	0	0	9	9	0	0	0
0	0	0	4	60	157	236	255	255	177	95	61	32	0	0	29
0	10	16	119	238	255	244	245	243	250	249	255	222	103	10	0
0	14	170	255	255	244	254	255	253	245	255	249	253	251	124	1
2	98	255	228	255	251	254	211	141	116	122	215	251	238	255	49
13	217	243	255	155	33	226	52	2	0	10	13	232	255	255	36
16	229	252	254	49	12	0	0	7	7	0	70	237	252	235	62
6	141	245	255	212	25	11	9	3	0	115	236	243	255	137	0
0	87	252	250	248	215	60	0	1	121	252	255	248	144	6	0
0	13	113	255	255	245	255	182	181	248	252	242	208	36	0	19
1	0	5	117	251	255	241	255	247	255	241	162	17	0	7	0
0	0	0	4	58	251	255	246	254	253	255	120	11	0	1	0
0	0	4	97	255	255	255	248	252	255	244	255	182	10	0	4
0	22	206	252	246	251	241	100	24	113	255	245	255	194	9	0
0	111	255	242	255	158	24	0	0	6	39	255	232	230	56	0
0	218	251	250	137	7	11	0	0	0	2	62	255	250	125	3
0	173	255	255	101	9	20	0	13	3	13	182	251	245	61	0
0	107	251	241	255	230	98	55	19	118	217	248	253	255	52	4
0	18	146	250	255	247	255	255	255	249	255	240	255	129	0	5
0	0	23	113	215	255	250	248	255	255	248	248	118	14	12	0
0	0	6	1	0	52	153	233	255	252	147	37	0	0	4	1
0	0	5	5	0	0	0	0	0	14	1	0	6	6	0	0

Convolutional Layers

Main Components

Convolution



weights = {1,2,3,4}

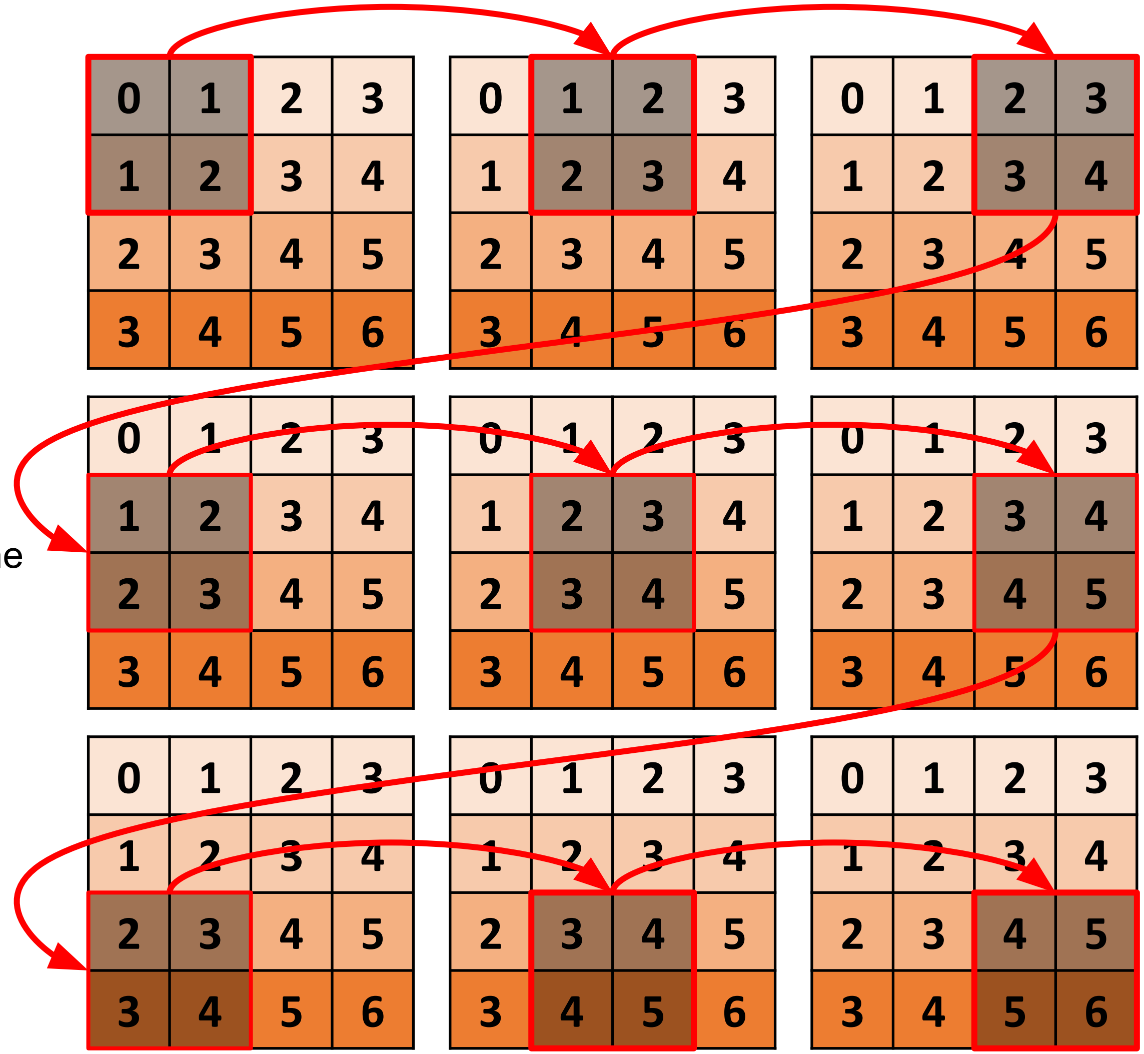
Filter or Kernel with size 2x2

= 0*1 + 1*2 + 1*3 + 2*4
= 0 + 2 + 3 + 8
= 13

Main Components

Convolution (stride = 1)

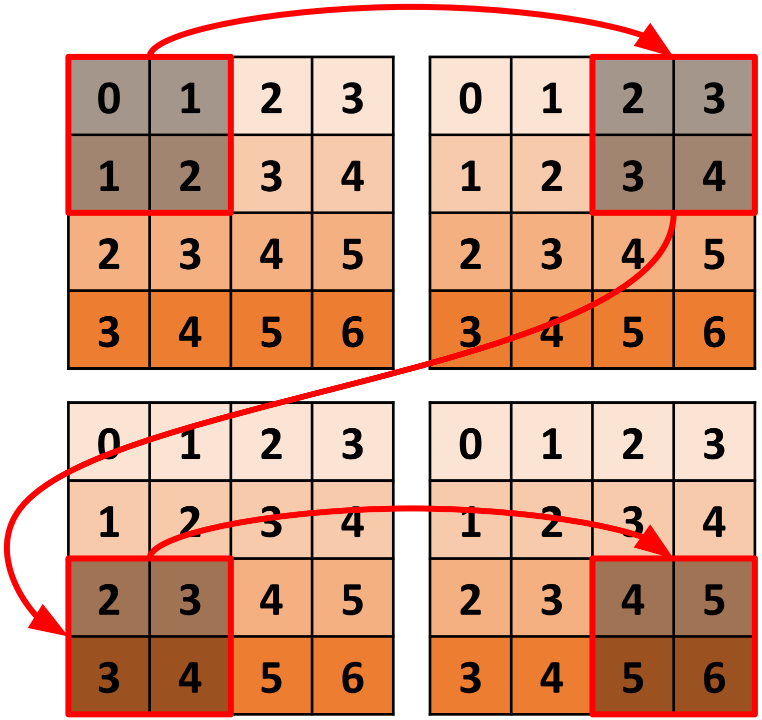
Stride is a parameter in CNNs that defines the number of pixels by which the convolutional filter/kernel moves across the input feature map during the convolution operation.



Main Components

Convolution (stride = 2)

Stride is a parameter in CNNs that defines the number of pixels by which the convolutional filter/kernel moves across the input feature map during the convolution operation.

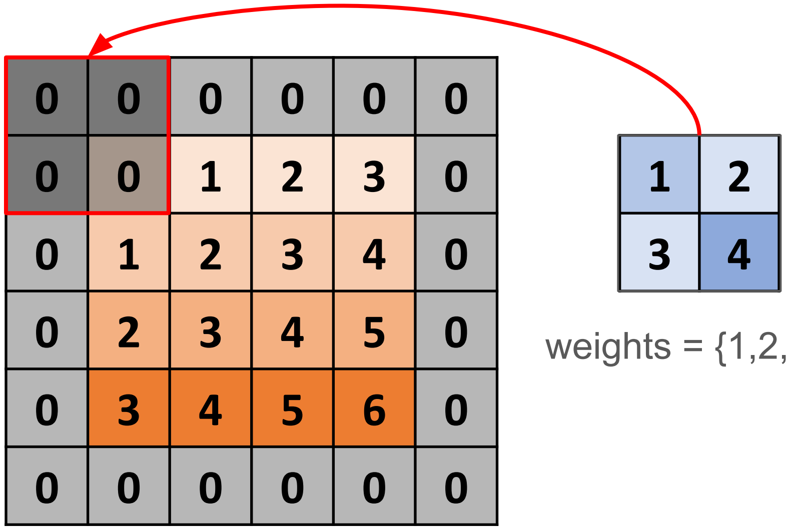


Main Components

- **Padding** is a technique used in CNNs to control the spatial dimensions of the output feature maps produced by convolutional layers.
- It involves adding extra pixels (usually zeros) around the borders of the input feature map before applying the convolution operation.

Convolution (padding)

= 0*1 + 0*2 + 0*3 + 0*4
= 0 + 0 + 0 + 0
= 0



Padding

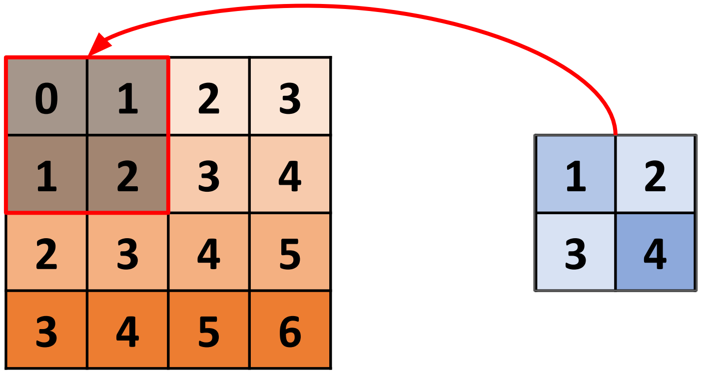
weights = {1,2,3,4}

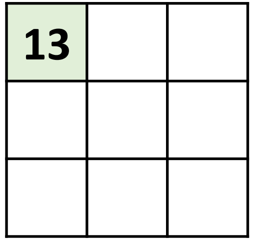
Filter or Kernel with size 2x2

Main Components

Convolution (no padding, stride = 1)

= 0*1 + 1*2 + 1*3 + 2*4
= 0 + 2 + 3 + 8
= 13





Main Components

Convolution (no padding, stride = 1)

= 1*1 + 2*2 + 2*3 + 3*4
= 1 + 4 + 6 + 12
= 23

0	1	2	3
1	2	3	4
2	3	4	5
3	4	5	6

1	2
3	4

13	23	

Main Components

Convolution

0	1	2	3
1	2	3	4
2	3	4	5
3	4	5	6

1	2
3	4

= 4*1 + 5*2 + 5*3 + 6*4
= 4 + 10 + 15 + 24
= 53

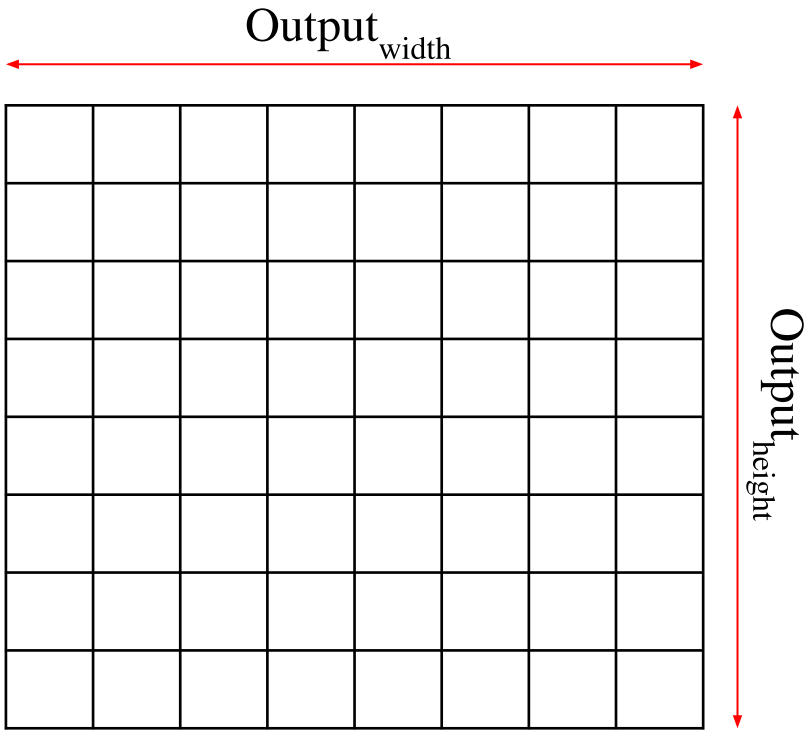
13	23	33
23	33	43
33	43	53

Main Components

Convolution

$$\text{Output}_{\text{width}} = \left\lfloor \frac{I_W + 2P - N_W}{S_x} + 1 \right\rfloor$$

$$\text{Output}_{\text{height}} = \left\lfloor \frac{I_H + 2P - N_H}{S_y} + 1 \right\rfloor$$



Main Components

Convolution

$$\begin{bmatrix} 0 & 1 & 2 & 3 \\ 1 & 2 & 3 & 4 \\ 2 & 3 & 4 & 5 \\ 3 & 4 & 5 & 6 \end{bmatrix} \quad \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \quad \begin{bmatrix} 13 & 23 & 33 \\ 23 & 33 & 43 \\ 33 & 43 & 53 \end{bmatrix}$$

$$\begin{aligned} \text{Output}_{\text{width}} &= \left\lfloor \frac{4 + 2(0) - 2}{1} + 1 \right\rfloor \\ &= 3 \end{aligned}$$

$$\begin{aligned} \text{Output}_{\text{height}} &= \left\lfloor \frac{4 + 2(0) - 2}{1} + 1 \right\rfloor \\ &= 3 \end{aligned}$$

Main Components

Convolution

0	0	0	0	0	0
0	0	1	2	3	0
0	1	2	3	4	0
0	2	3	4	5	0
0	3	4	5	6	0
0	0	0	0	0	0

1	2	3
2	3	4
3	4	5

14	26	38	22
26	43	62	38
38	62	86	54
22	38	54	30

$$\text{Output}_{\text{width}} = \left\lfloor \frac{4 + 2(1) - 3}{1} + 1 \right\rfloor$$
$$= 4$$

$$\text{Output}_{\text{height}} = \left\lfloor \frac{4 + 2(1) - 3}{1} + 1 \right\rfloor$$
$$= 4$$

Main Components

Convolution

0	1	2	3
1	2	3	4
2	3	4	5
3	4	5	6

1	2
3	4

13	33
33	53

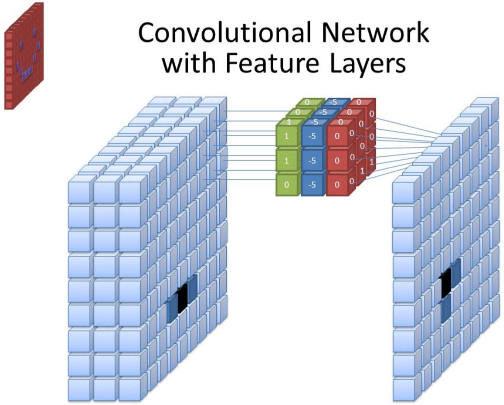
$$\text{Output}_{\text{width}} = \left\lfloor \frac{4 + 2(0) - 2}{2} + 1 \right\rfloor$$
$$= 2$$

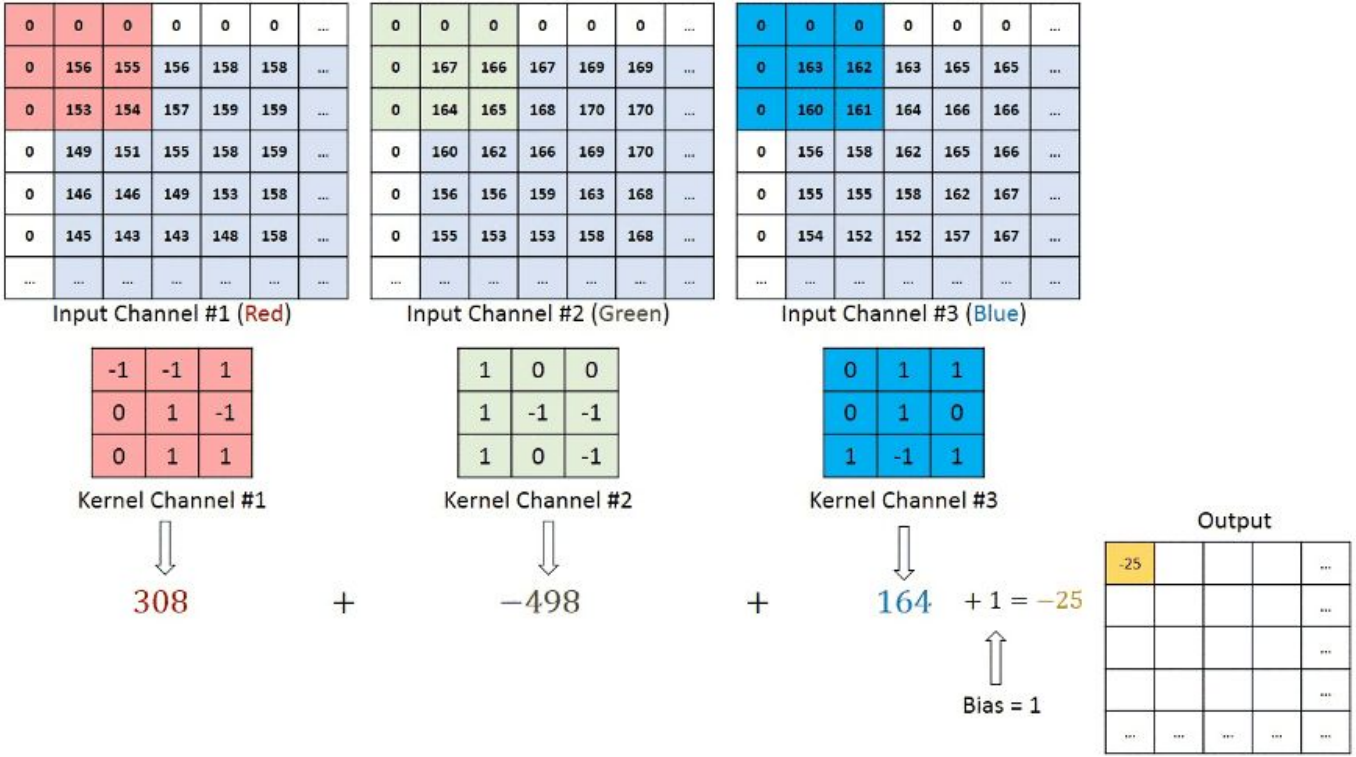
$$\text{Output}_{\text{height}} = \left\lfloor \frac{4 + 2(0) - 2}{2} + 1 \right\rfloor$$
$$= 2$$

Main Components

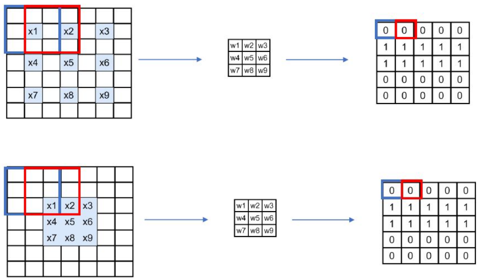
Convolution

(RGB)



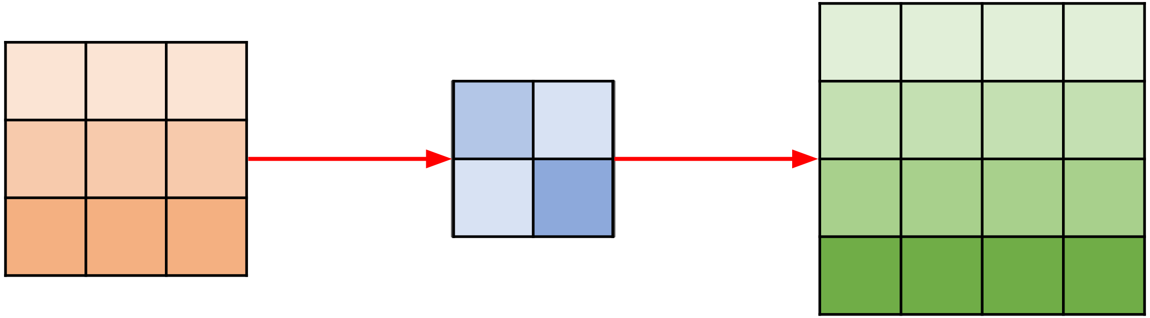


Main Components



Transposed Convolution

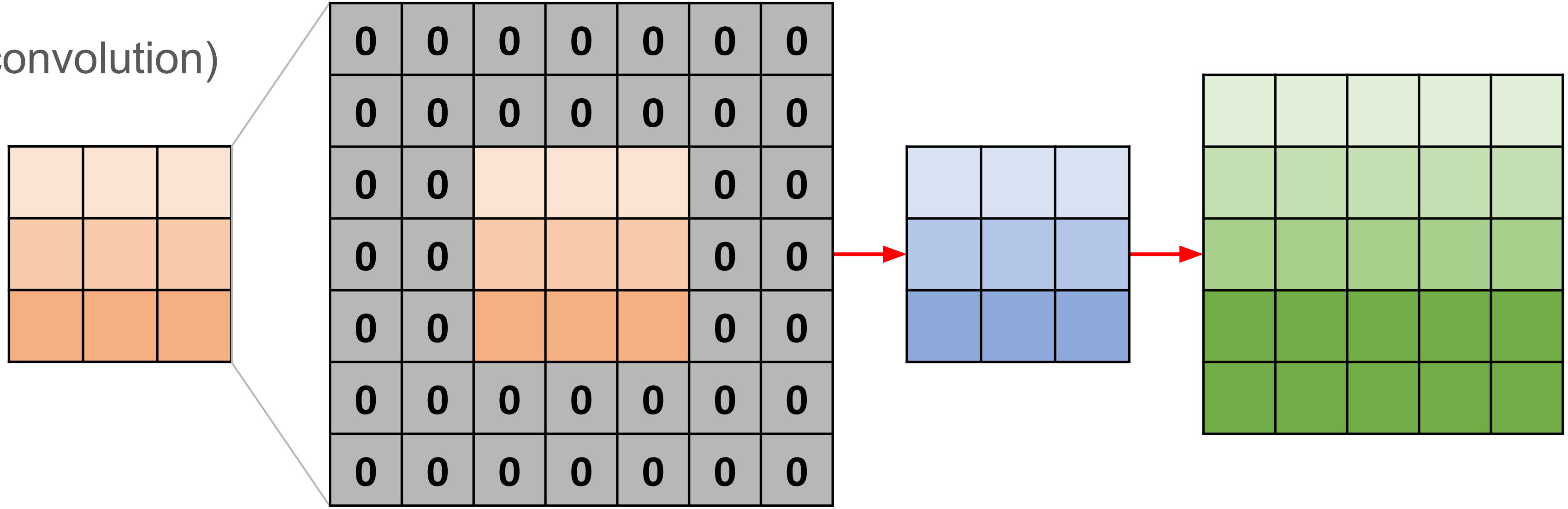
(Deconvolution)



Main Components

Transposed Convolution

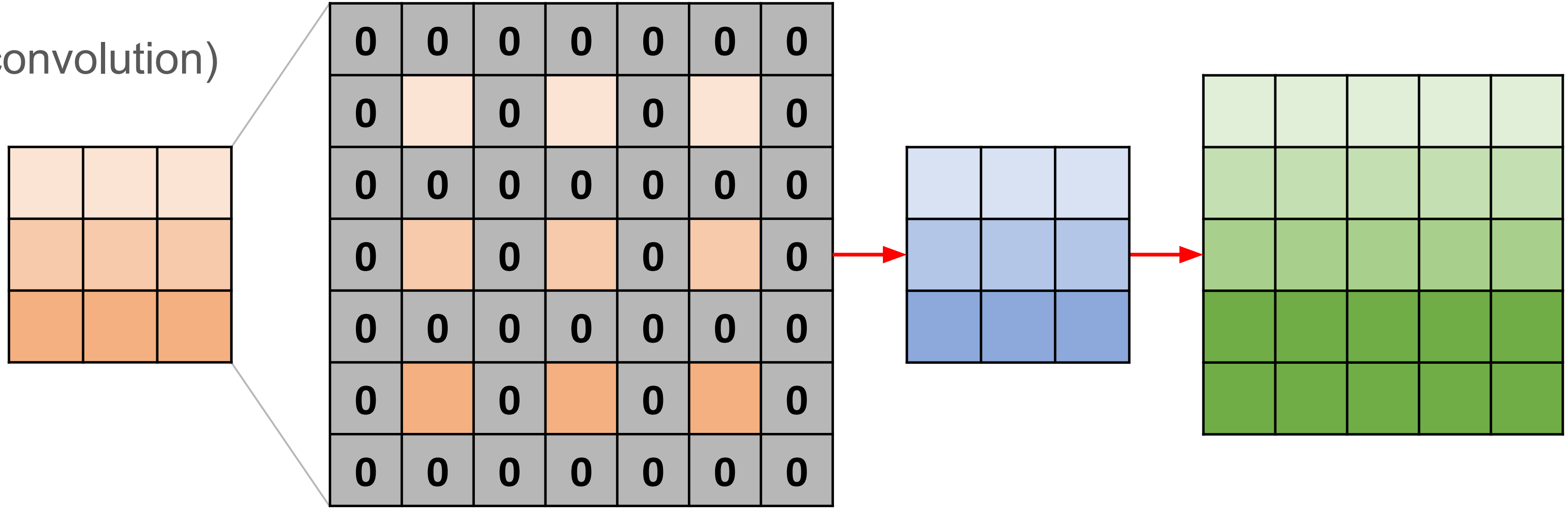
(Deconvolution)



Main Components

Transposed Convolution

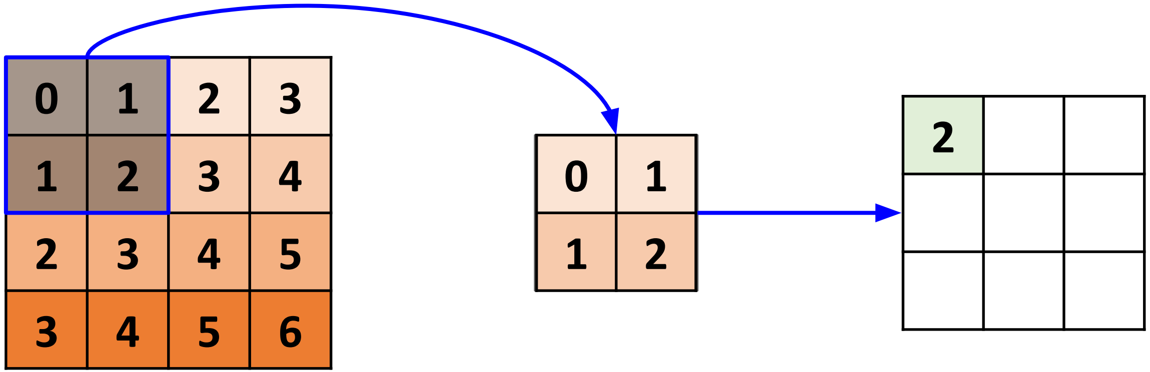
(Deconvolution)



Pooling Layers

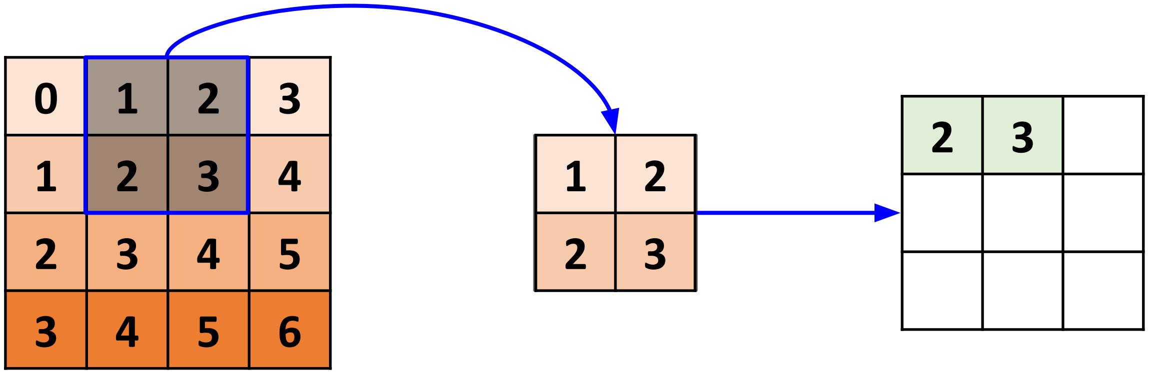
Main Components

Max Pooling



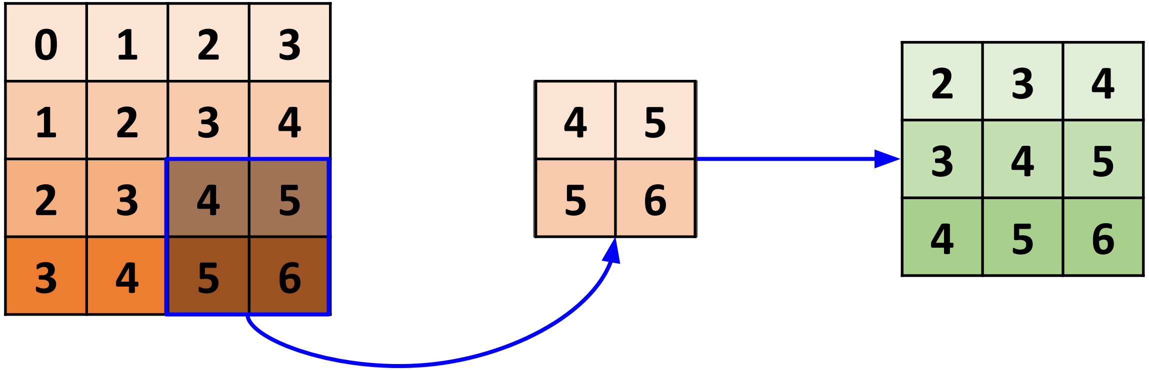
Main Components

Max Pooling



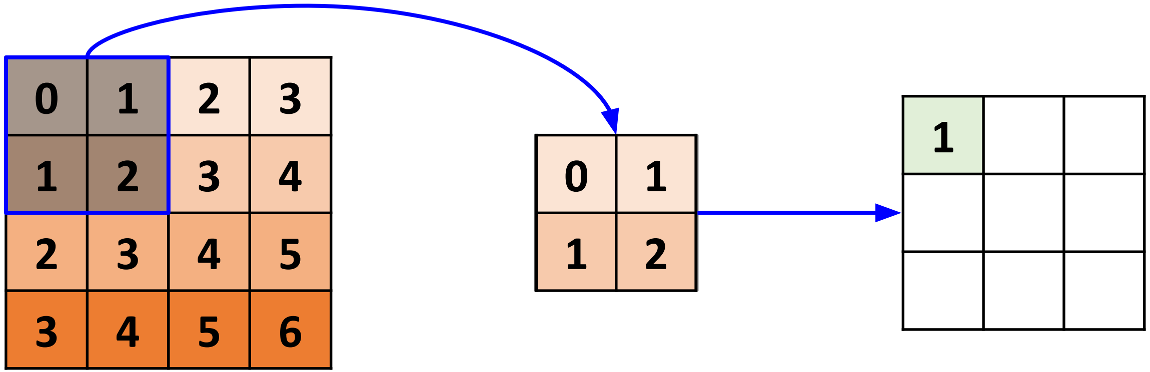
Main Components

Max Pooling



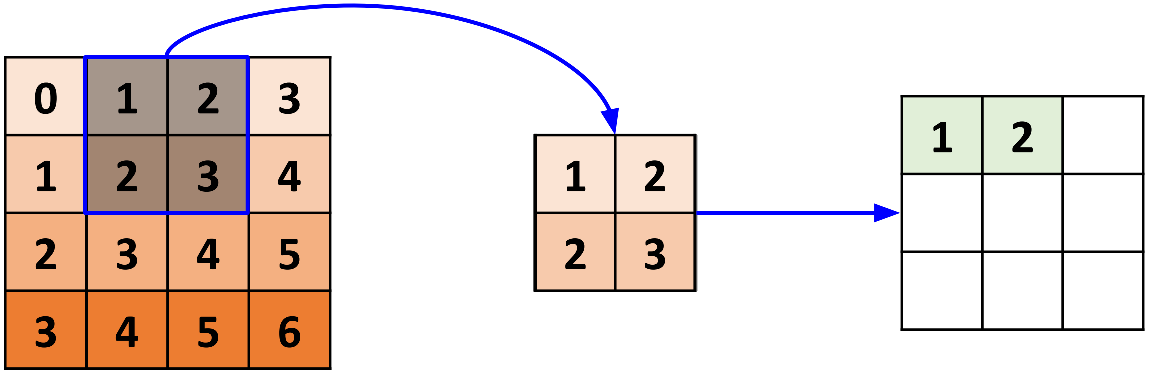
Main Components

Average Pooling



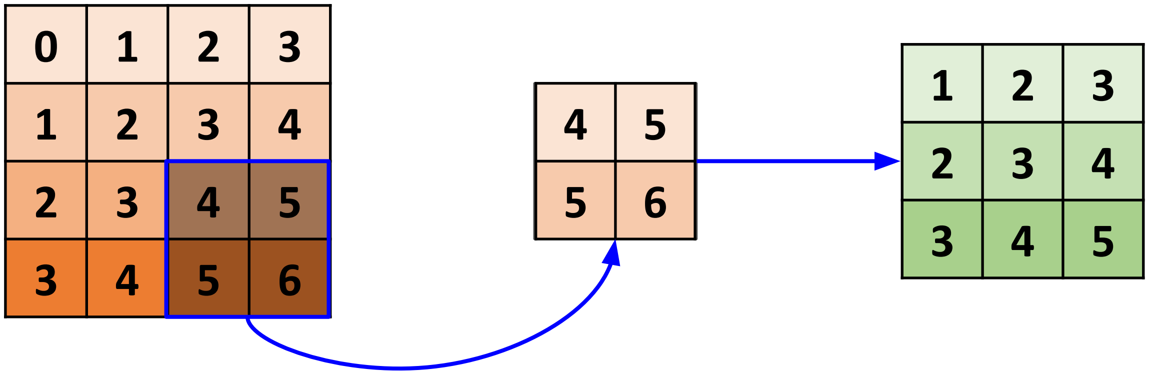
Main Components

Average Pooling



Main Components

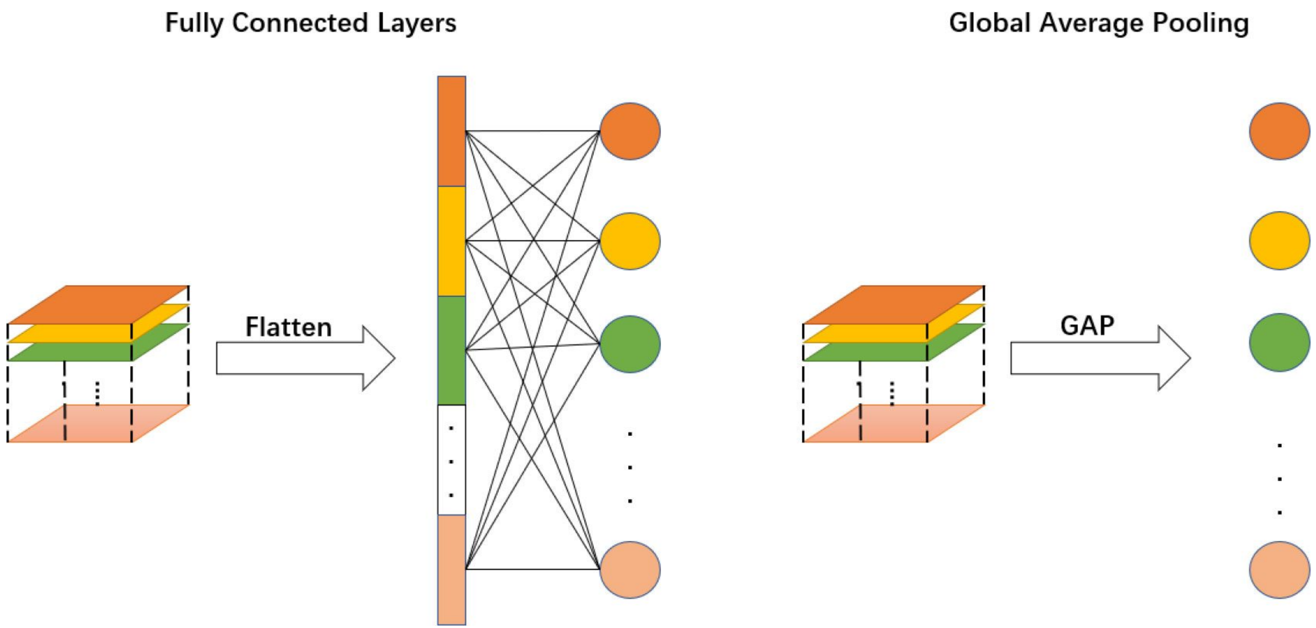
Average Pooling



Fully connected Layers

Main Components

Flatten vs Global Pooling



Summary (จดเอง)

Hands On

จงคำนวณผลลัพธ์ของ convolution layer และ pooling layer จากข้อมูลที่กำหนดให้

พร้อมทั้งคำนวณหาขนาดของ convolution และ pooling ที่เป็นผลลัพธ์ด้วย

Image

1	2	5	8
5	2	1	7
2	1	1	2
1	0	2	2

Kernel

1	0
0	1

stride = 1

Padding = 0

Activation is Linear (F(x) = x)

Pooling is Max pooling with size 2x2

Hands On

จงคำนวณผลลัพธ์ของ convolution layer และ pooling layer จากข้อมูลที่กำหนดให้

พร้อมทั้งคำนวณหาขนาดของ convolution และ pooling ที่เป็นผลลัพธ์ด้วย

Image

1	2	5	8
5	2	1	7
2	1	1	2
1	0	2	2

Kernel

1	0
0	1

stride = 1

Padding = 1

Activation is Linear (F(x) = x)

Pooling is Max pooling with size 2x2