



Навчання з підкріпленням

Лекція 2: Згорткові мережі

Кочура Юрій Петрович
iuriy.kochura@gmail.com
[@y_kochura](https://twitter.com/y_kochura)

Сьогодні

Розуміння згорткових нейронних мереж ([convnets](#) або [CNNs](#))

- Повнозв'язна vs згорткова мережа
- Операція згортки
- Крок згортки
- Ефект доповнення (padding)
- Операція агрегації (pooling)

Повнозв'язна мережа

MNIST: приклади



Примітка! У машинному навчанні для задачі класифікації категорія даних називається **класом**. Кожна одиниця даних називається **прикладом**. Клас, який пов'язаний із певним прикладом називається **міткою (label)**.

Імпортування набору даних MNIST у Keras

```
from tensorflow.keras.datasets import mnist
(train_images, train_labels), (test_images, test_labels) = mnist.load_data()
```

- `train_images` та `train_labels` – навчальний набір даних (дані на яких модель буде навчатись).
- `test_images` та `test_labels` – тестовий набір (дані на яких буде оцінено продуктивність моделі)

Навчальний набір

```
>>> train_images.shape  
(60000, 28, 28)  
>>> len(train_labels)  
60000  
>>> train_labels  
array([5, 0, 4, ..., 5, 6, 8], dtype=uint8)
```

Тестовий набір

```
>>> test_images.shape  
(10000, 28, 28)  
>>> len(test_labels)  
10000  
>>> test_labels  
array([7, 2, 1, ..., 4, 5, 6], dtype=uint8)
```

Архітектура мережі

```
from tensorflow import keras
from tensorflow.keras import layers
model = keras.Sequential([
    layers.Dense(512, activation="relu"),
    layers.Dense(10, activation="softmax")
])
```

Робочий процес буде таким: спочатку ми передамо нейронній мережі навчальні дані `train_images` та `train_labels`. Таким чином мережа навчиться пов'язувати зображення з мітки. Потім ми попросимо мережу створити прогнози для `test_images` та ми перевіримо, чи відповідають ці прогнози міткам з `test_labels`.

Готуємо мережу до навчання

Щоб підготувати мережу до навчання, нам потрібно визначити на етапі компіляції:

- Оптимізатор – алгоритм за допомогою якого модель оновлюватиметься на основі навчальних даних, які надаються моделі для покращення свої продуктивності.
- Функція втрат – спосіб виміру втрат моделі. Оптимізатор намагається мінімізувати втрати моделі.
- Метрики для моніторингу під час навчання та тестування – у цій задачі ми слідкуватимо за точністю (відсоток зображень, які були правильно класифіковані).

Компіляція моделі

```
model.compile(optimizer="rmsprop",
              loss="sparse_categorical_crossentropy",
              metrics=["accuracy"])
```

Підготовка даних

```
train_images = train_images.reshape((60000, 28 * 28))
train_images = train_images.astype("float32") / 255
test_images = test_images.reshape((10000, 28 * 28))
test_images = test_images.astype("float32") / 255
```

Раніше наші навчальні зображення зберігалися в масиві форми **(60000, 28, 28)** типу uint8 зі значеннями в інтервалі [0, 255]. Ми перетворимо його на масив float32 форми **(60000, 28 * 28)** зі значеннями від 0 до 1.

Навчання моделі

```
>>> model.fit(train_images, train_labels, epochs=5, batch_size=128)
Epoch 1/5
60000/60000 [=====] - 5s - loss: 0.2524 - acc: 0.9273
Epoch 2/5
51328/60000 [=====>.....] - ETA: 1s - loss: 0.1035 - acc: 0.9692
```

Виконання прогнозу

```
>>> test_digits = test_images[0:10]
>>> predictions = model.predict(test_digits)
>>> predictions[0]
array([1.0726176e-10, 1.6918376e-10, 6.1314843e-08, 8.4106023e-06,
       2.9967067e-11, 3.0331331e-09, 8.3651971e-14, 9.9999106e-01,
       2.6657624e-08, 3.8127661e-07], dtype=float32)
```

Кожне індекс *i* в цьому масиві відповідає ймовірності того, що наше тестове зображення `test_digits[0]` належить до класу *i*.

Виконання прогнозу

Перше тестове зображення має найбільше ймовірність (0.99999106, майже 1) для індекса масива 7, тому відповідно до цього прогнозу моделі це має бути 7:

```
>>> predictions[0].argmax()  
7  
>>> predictions[0][7]  
0.99999106
```

Перевіримо на відповідність тестовій мітці:

```
>>> test_labels[0]  
7
```

Оцінка моделі на нових даних

```
>>> test_loss, test_acc = model.evaluate(test_images, test_labels)
>>> print(f"test_acc: {test_acc}")
test_acc: 0.9785
```

Згорткові мережі

Створюємо згорткову мережу

```
from tensorflow import keras
from tensorflow.keras import layers
inputs = keras.Input(shape=(28, 28, 1))
x = layers.Conv2D(filters=32, kernel_size=3, activation="relu")(inputs)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(filters=64, kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(filters=128, kernel_size=3, activation="relu")(x)
x = layers.Flatten()(x)
outputs = layers.Dense(10, activation="softmax")(x)
model = keras.Model(inputs=inputs, outputs=outputs)
```

Важливо, що convnet приймає на вхід тензори форми (image_height, image_width, image_channels), не включаючи розмірність пакету. У цьому випадку ми налаштуємо convnet для обробки вхідних даних розміром (28, 28, 1), що відповідає формату зображень MNIST.

Summary

```
>>> model.summary()  
Model: "model"
```

Layer (type)	Output Shape	Param #
<hr/>		
input_1 (InputLayer)	[None, 28, 28, 1]	0
conv2d (Conv2D)	(None, 26, 26, 32)	320
max_pooling2d (MaxPooling2D)	(None, 13, 13, 32)	0
conv2d_1 (Conv2D)	(None, 11, 11, 64)	18496
max_pooling2d_1 (MaxPooling2D)	(None, 5, 5, 64)	0
conv2d_2 (Conv2D)	(None, 3, 3, 128)	73856
flatten (Flatten)	(None, 1152)	0
dense (Dense)	(None, 10)	11530
<hr/>		
Total params:	104,202	
Trainable params:	104,202	
Non-trainable params:	0	

Навчання

```
from tensorflow.keras.datasets import mnist

(train_images, train_labels), (test_images, test_labels) = mnist.load_data()
train_images = train_images.reshape((60000, 28, 28, 1))
train_images = train_images.astype("float32") / 255
test_images = test_images.reshape((10000, 28, 28, 1))
test_images = test_images.astype("float32") / 255
model.compile(optimizer="rmsprop",
              loss="sparse_categorical_crossentropy",
              metrics=["accuracy"])
model.fit(train_images, train_labels, epochs=5, batch_size=64)
```

Оцінка моделі на нових даних

```
>>> test_loss, test_acc = model.evaluate(test_images, test_labels)
>>> print(f"Test accuracy: {test_acc:.3f}")
Test accuracy: 0.991
```

Будівельні блоки

Тензор (**tensor**)

масив чисел, розташованих у сітці зі змінною кількістю осей





0D tensor

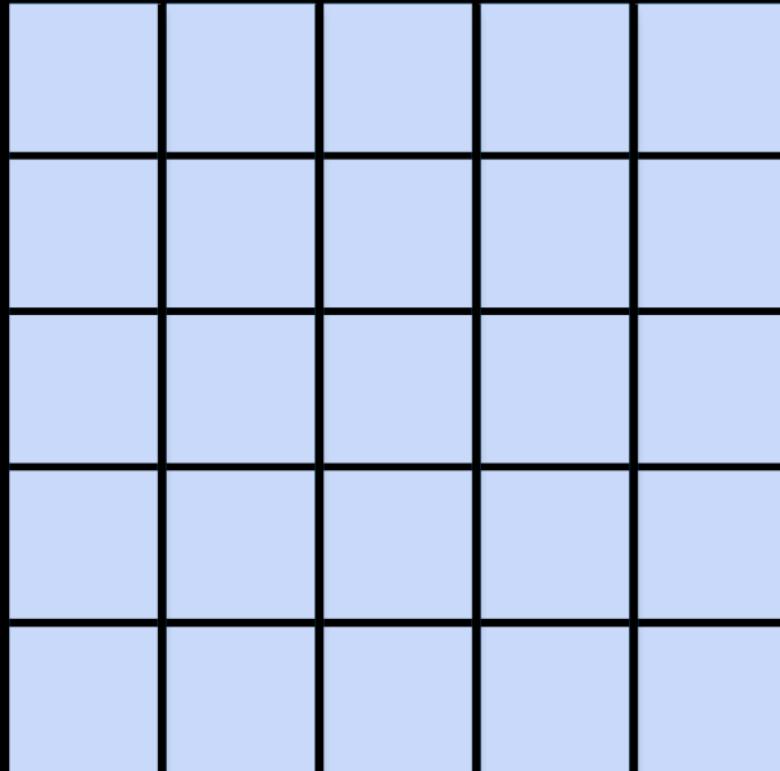


$\text{rows} \times 1$

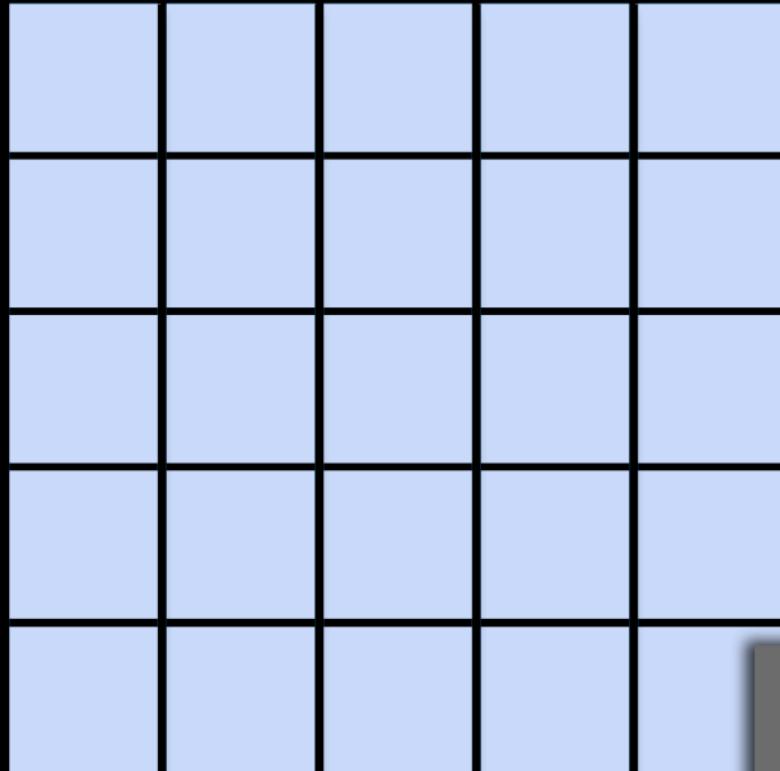


1D tensor

$\text{rows} \times 1$

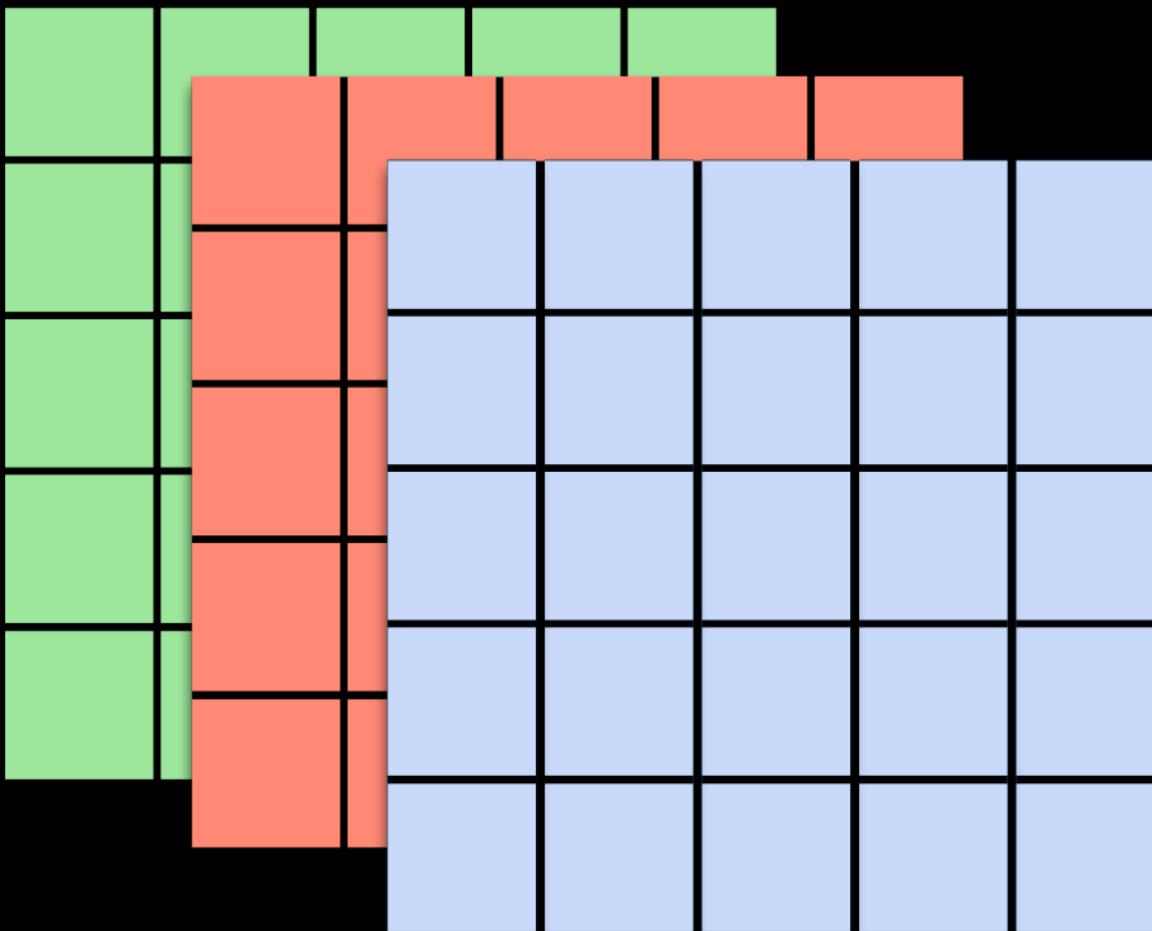


$\text{rows} \times \text{cols}$

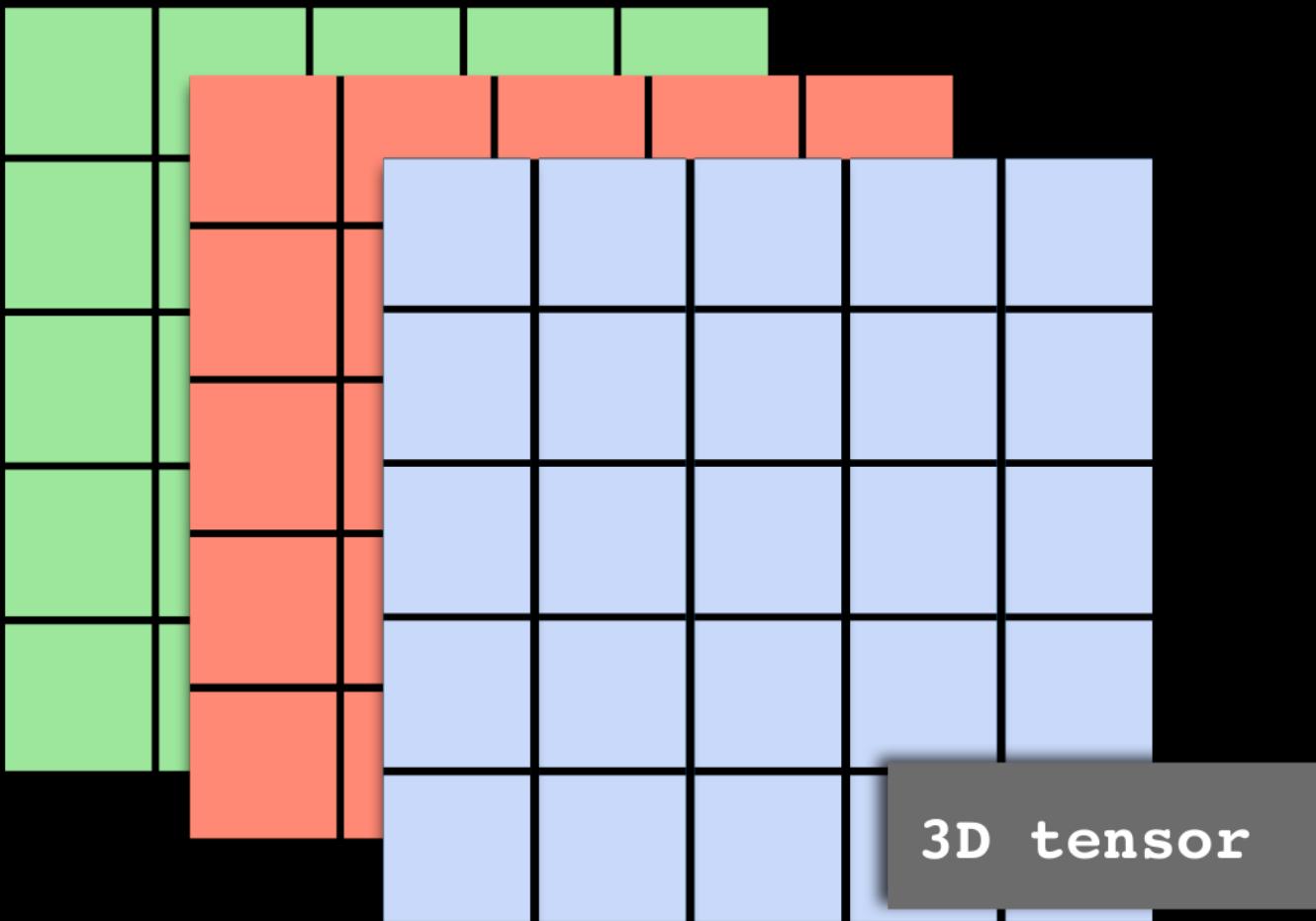


2D tensor

rows \times cols

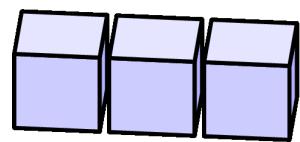


rows × cols × channels

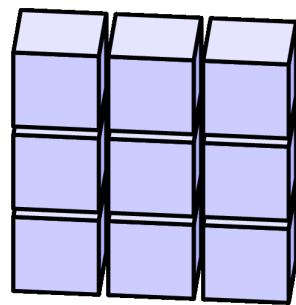


$\text{rows} \times \text{cols} \times \text{channels}$

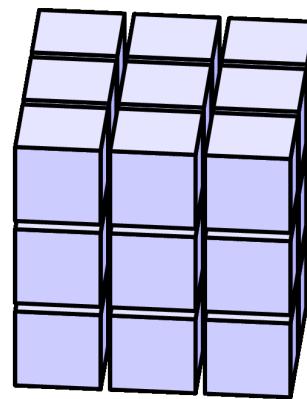
vector



matrix

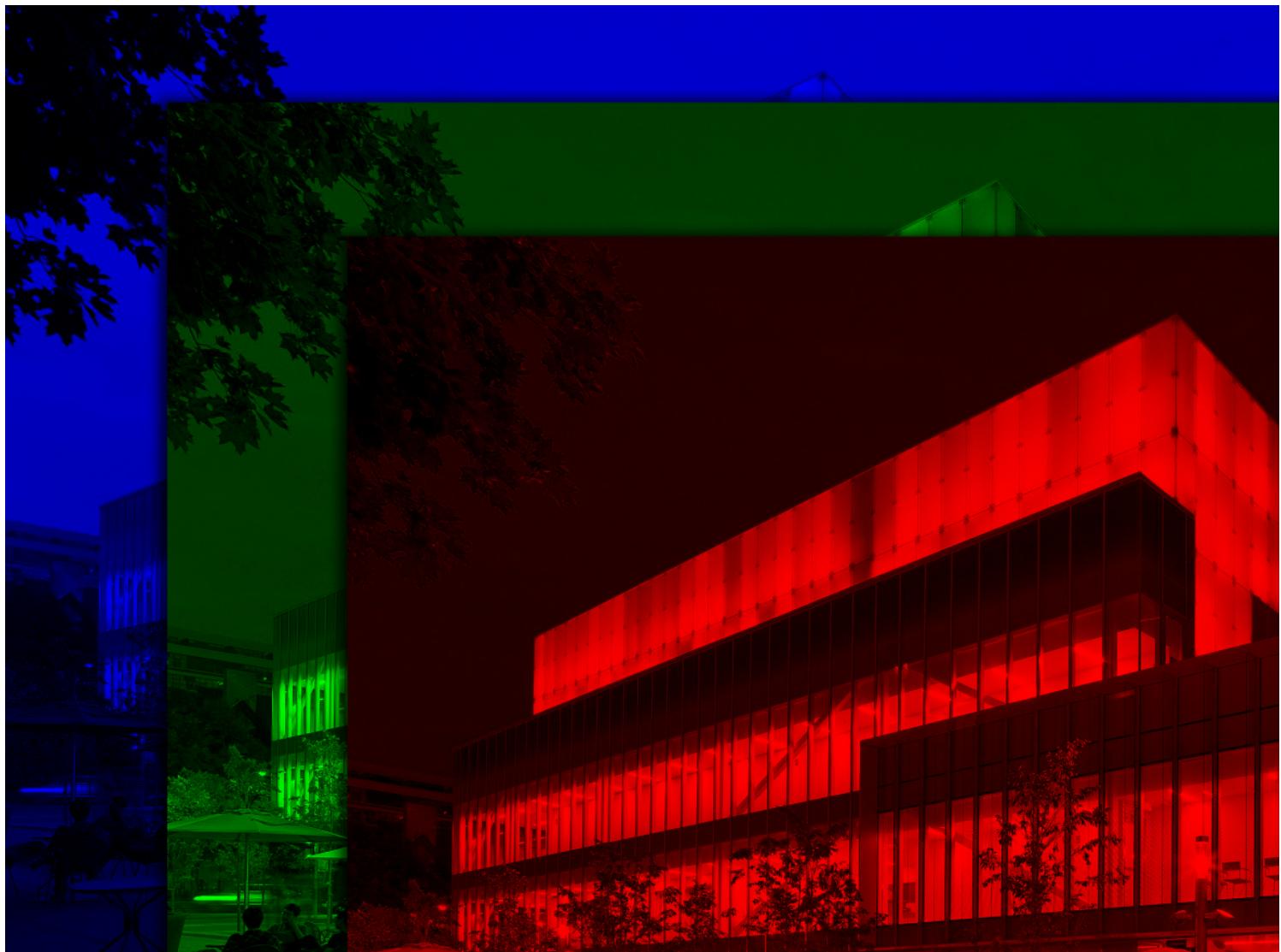


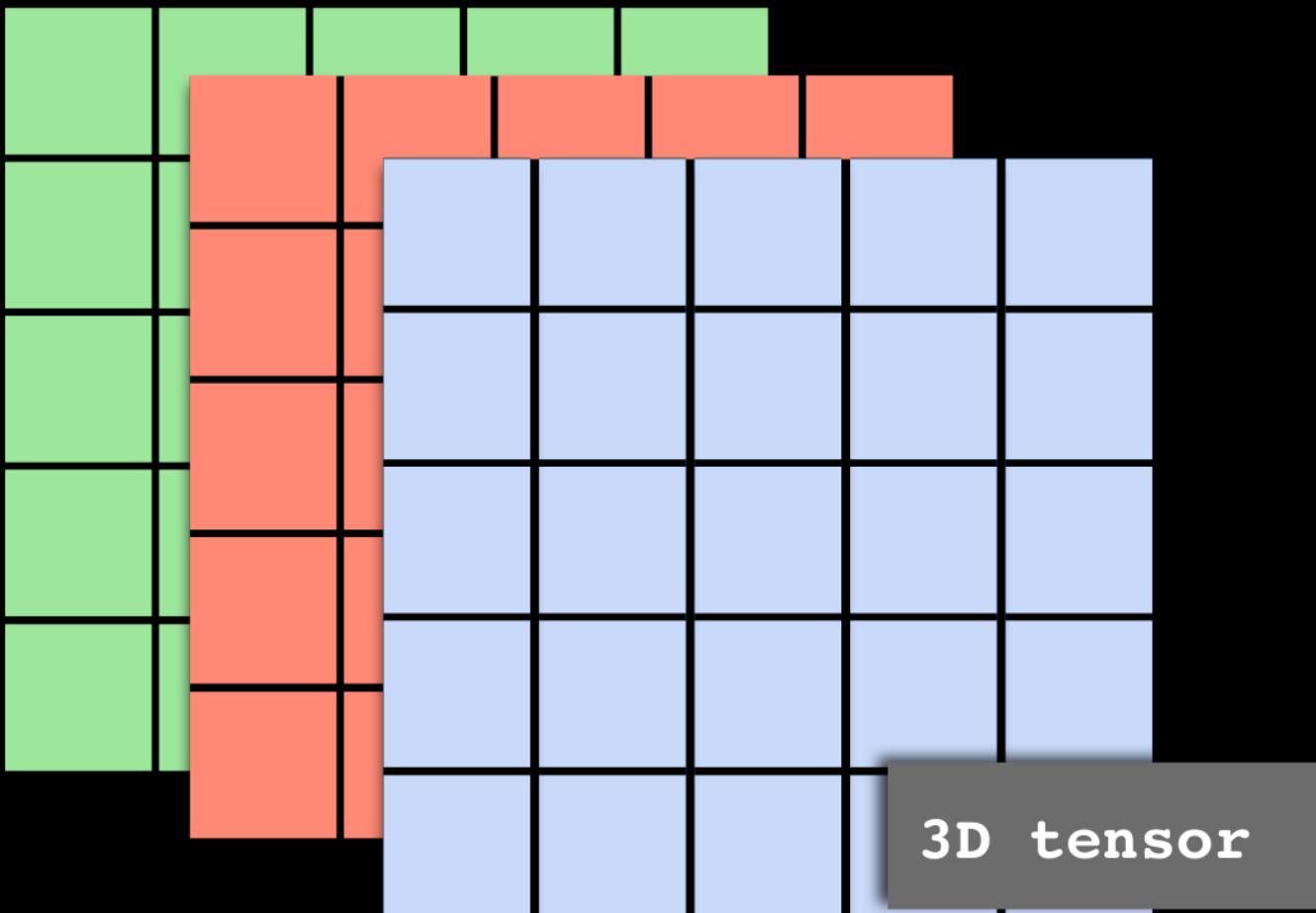
array



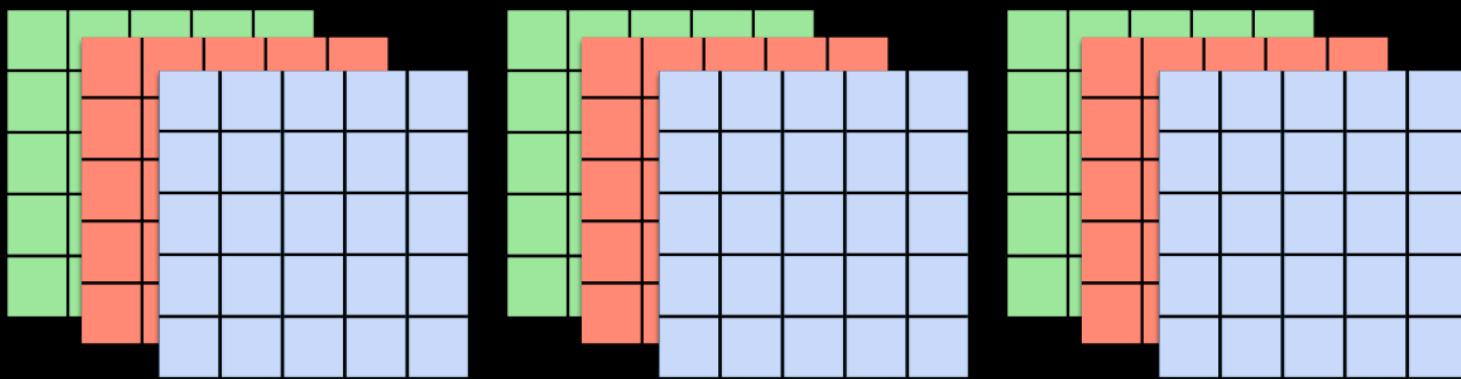


768 × 1024 × 3

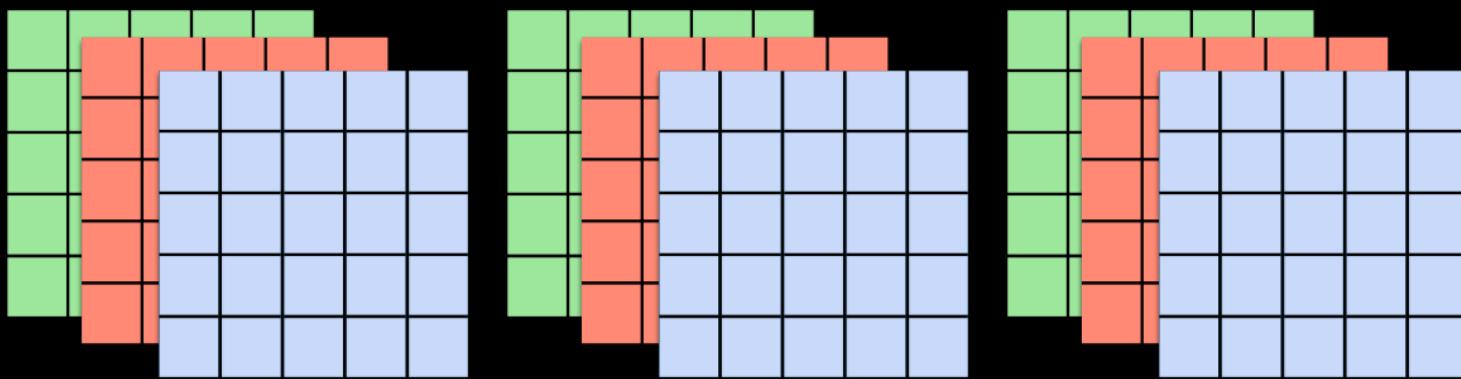




$\text{rows} \times \text{cols} \times \text{channels}$



rows \times cols \times channels \times t



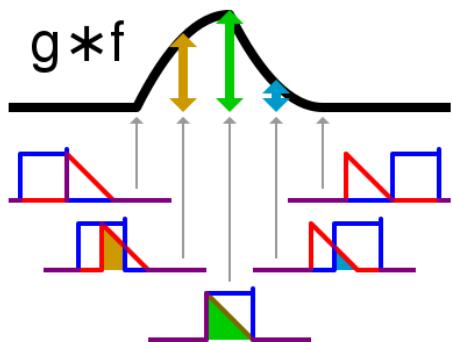
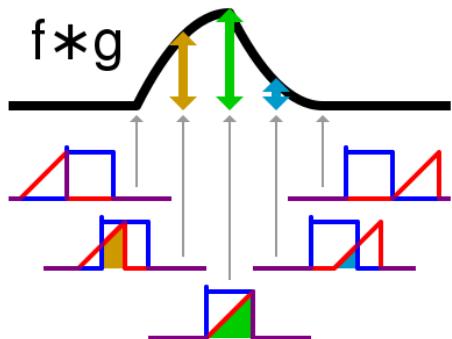
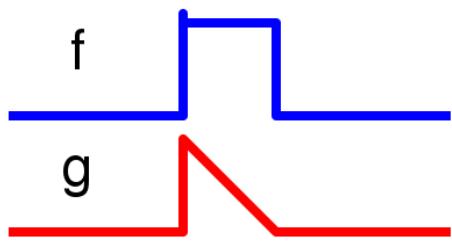
rows \times cols \times channels \times t

4D tensor

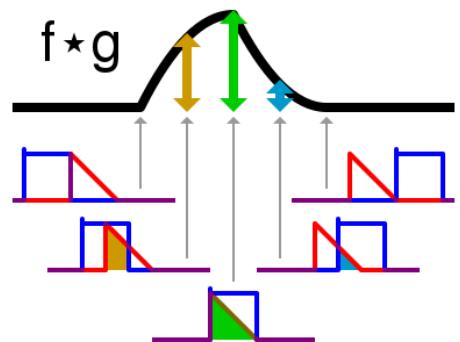
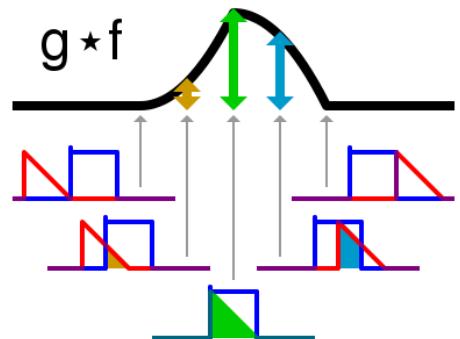
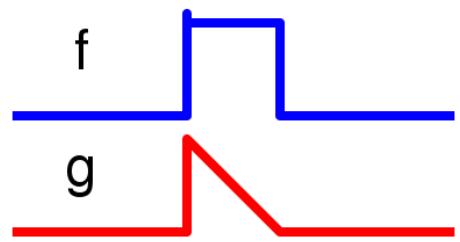
Згортка (convolution)

$$(f * g)(x) = \int f(z)g(x - z) dz$$

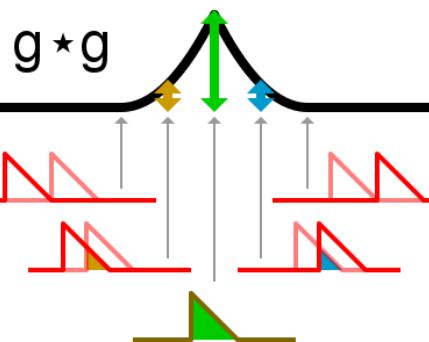
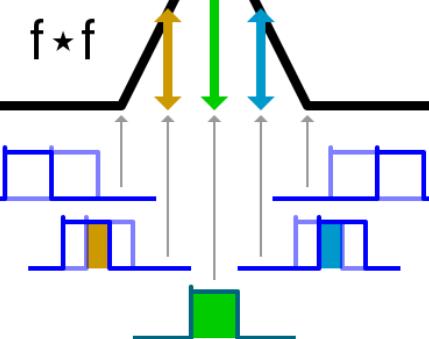
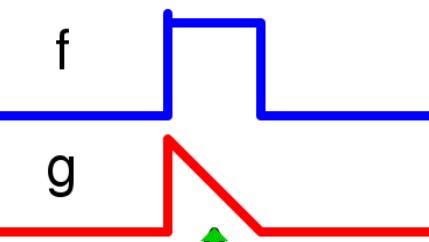
Convolution



Cross-correlation



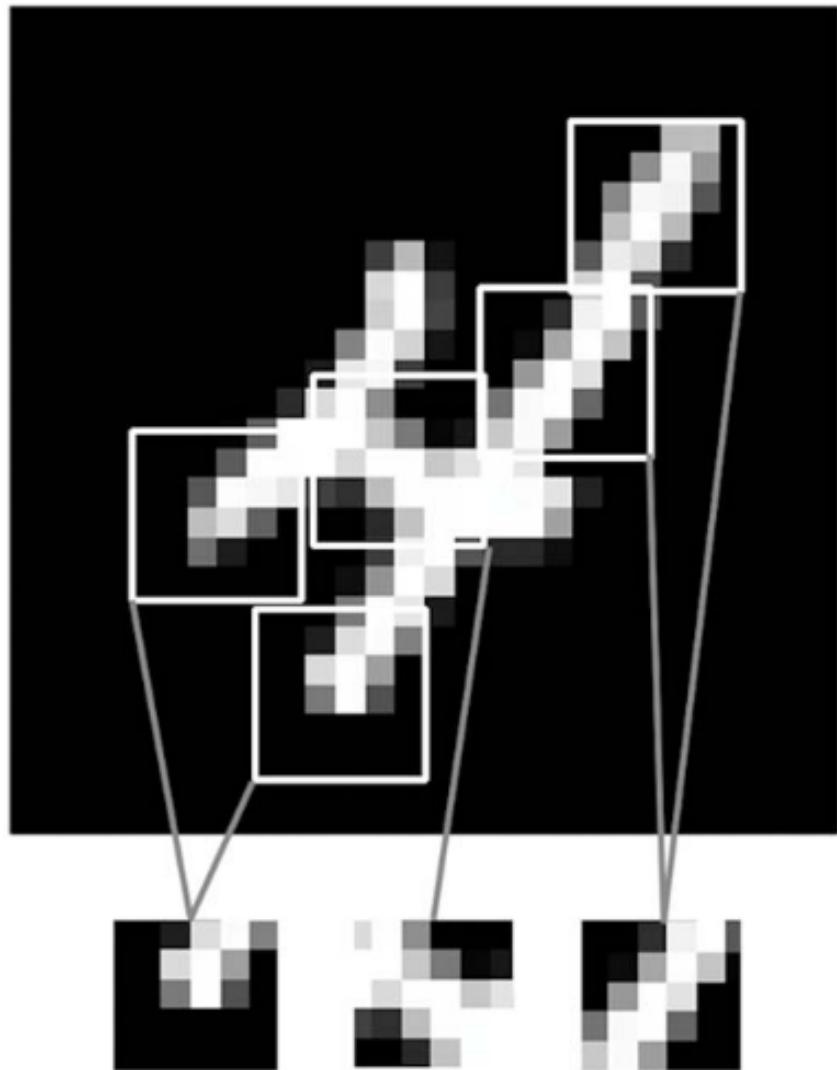
Autocorrelation



convolution

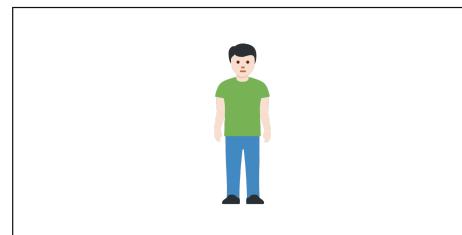


correlation



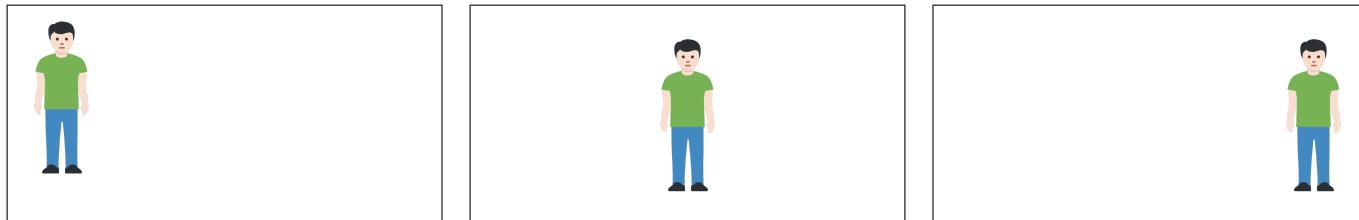
Ця особливість надає згортковим нейронним мережам наступні дві важливі властивості:

1. Шаблони, які вивчають згорткові мережі, є інваріантами відносно зміщень об'єктів.

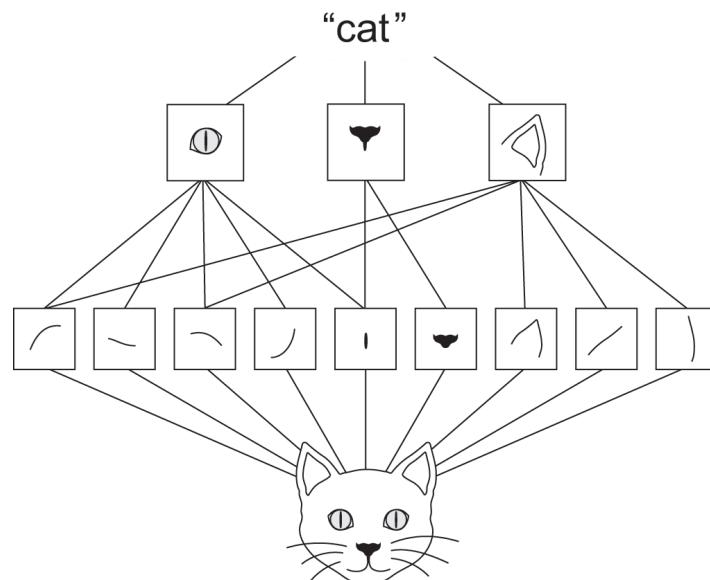


Ця особливість надає згортковим нейронним мережам наступні дві важливі властивості:

1. Шаблони, які вивчають згорткові мережі, є інваріантами відносно зміщень об'єктів.



2. CNNs можуть вивчати просторові ієрархії шаблонів.



Операція згортки

3	3	2	1	0
0	0	1	3	1
3	1	2	2	3
2	0	0	2	2
2	0	0	0	1

*

0	1	2
2	2	0
0	1	2

convolution

$$\begin{array}{c|ccc|cc} & & & & & \\ \hline & 0 & 1 & 2 & & \\ \hline 0 & & & & & \\ \hline 2 & 3 & 2 & 3 & 0 & 2 & 1 & 0 \\ \hline 0 & 0 & 1 & 0 & 2 & 1 & 3 & 1 \\ \hline & 3 & 1 & 2 & 2 & 3 & & \\ \hline & 2 & 0 & 0 & 2 & 2 & & \\ \hline & 2 & 0 & 0 & 0 & 1 & & \\ \hline \end{array} = \begin{array}{c|ccc|cc} & & & & & \\ \hline \end{array}$$

3 0	3 1	2 2	1	0
0 2	0 2	1 0	3	1
3 0	1 1	2 2	2	3
2	0	0	2	2
2	0	0	0	1

=

3 0	3 1	2 2	1	0		
0 2	0 2	1 0	3	1		
3 0	1 1	2 2	2	3		
=						
=						
2	0	0	0	1		

pointwise multiplication and sum

3 0	3 1	2 2	1	0	
0 2	0 2	1 0	3	1	
3 0	1 1	2 2	2	3	
=					12
=					
2	0	0	0	1	

pointwise multiplication and sum

3	3 0	2 1	1 2	0
0	0 2	1 2	3 0	1
3	1 0	2 1	2 2	3
2	0	0	2	2
2	0	0	0	1

=

12		

$$\begin{array}{|c|c|c|c|c|} \hline
 3 & 3_0 & 2_1 & 1_2 & 0 \\ \hline
 0 & 0_2 & 1_2 & 3_0 & 1 \\ \hline
 3 & 1_0 & 2_1 & 2_2 & 3 \\ \hline
 2 & 0 & 0 & 2 & 2 \\ \hline
 2 & 0 & 0 & 0 & 1 \\ \hline
 \end{array}
 =
 \begin{array}{|c|c|c|} \hline
 12 & 12 & \\ \hline
 & & \\ \hline
 & & \\ \hline
 & & \\ \hline
 & & \\ \hline
 \end{array}$$

3	3	2 0	1 1	0 2		
0	0	1 2	3 2	1 0		
3	1	2 0	2 1	3 2		
2	0	0	2	2		
2	0	0	0	1		

=

12	12	

3	3	2 0	1 1	0 2			
0	0	1 2	3 2	1 0			
3	1	2 0	2 1	3 2			
2	0	0	2	2			
2	0	0	0	1			

=

12	12	17

$$\begin{array}{ccccc}
 3 & 3 & 2 & 1 & 0 \\
 \hline
 0 & 0 & 0 & 1 & 1 \\
 3 & 2 & 1 & 2 & 2 \\
 2 & 0 & 0 & 1 & 2 \\
 \hline
 2 & 0 & 0 & 0 & 1
 \end{array}
 =
 \begin{array}{ccc}
 12 & 12 & 17 \\
 \hline
 \end{array}$$

$$\begin{array}{ccccc}
 3 & 3 & 2 & 1 & 0 \\
 \hline
 0 & 0 & 0 & 1 & 1 \\
 3 & 2 & 1 & 2 & 2 \\
 2 & 0 & 0 & 1 & 2 \\
 \hline
 2 & 0 & 0 & 0 & 1
 \end{array}
 =
 \begin{array}{ccccc}
 & 12 & 12 & 17 & \\
 \hline
 & 10 & & & \\
 & & & & \\
 & & & &
 \end{array}$$

$$\begin{array}{ccccc}
 3 & 3 & 2 & 1 & 0 \\
 \hline
 0 & 0_0 & 1_1 & 3_2 & 1 \\
 \hline
 3 & 1_2 & 2_2 & 2_0 & 3 \\
 \hline
 2 & 0_0 & 0_1 & 2_2 & 2 \\
 \hline
 2 & 0 & 0 & 0 & 1
 \end{array}
 =
 \begin{array}{ccc}
 12 & 12 & 17 \\
 \hline
 10 & &
 \end{array}$$

$$\begin{array}{ccccc}
 3 & 3 & 2 & 1 & 0 \\
 \hline
 0 & 0_0 & 1_1 & 3_2 & 1 \\
 \hline
 3 & 1_2 & 2_2 & 2_0 & 3 \\
 \hline
 2 & 0_0 & 0_1 & 2_2 & 2 \\
 \hline
 2 & 0 & 0 & 0 & 1
 \end{array}
 =
 \begin{array}{ccc}
 12 & 12 & 17 \\
 \hline
 10 & 17 &
 \end{array}$$

$$\begin{array}{ccccc}
 3 & 3 & 2 & 1 & 0 \\
 \hline
 0 & 0 & 1_0 & 3_1 & 1_2 \\
 3 & 1 & 2_2 & 2_2 & 3_0 \\
 \hline
 2 & 0 & 0_0 & 2_1 & 2_2 \\
 \hline
 2 & 0 & 0 & 0 & 1
 \end{array} =
 \begin{array}{ccc}
 12 & 12 & 17 \\
 10 & 17 &
 \end{array}$$

3	3	2	1	0			
0	0	1 0	3 1	1 2	12	12	17
3	1	2 2	2 2	3 0	10	17	19
2	0	0 0	2 1	2 2			
2	0	0	0	1			

$$\begin{array}{ccccc}
 3 & 3 & 2 & 1 & 0 \\
 0 & 0 & 1 & 3 & 1 \\
 3_0 & 1_1 & 2_2 & 2 & 3 \\
 2_2 & 0_2 & 0_0 & 2 & 2 \\
 2_0 & 0_1 & 0_2 & 0 & 1
 \end{array}
 =
 \begin{array}{ccc}
 12 & 12 & 17 \\
 10 & 17 & 19
 \end{array}$$

$$\begin{array}{ccccc}
 3 & 3 & 2 & 1 & 0 \\
 0 & 0 & 1 & 3 & 1 \\
 3_0 & 1_1 & 2_2 & 2 & 3 \\
 2_2 & 0_2 & 0_0 & 2 & 2 \\
 2_0 & 0_1 & 0_2 & 0 & 1
 \end{array}
 =
 \begin{array}{ccc}
 12 & 12 & 17 \\
 10 & 17 & 19 \\
 9 & &
 \end{array}$$

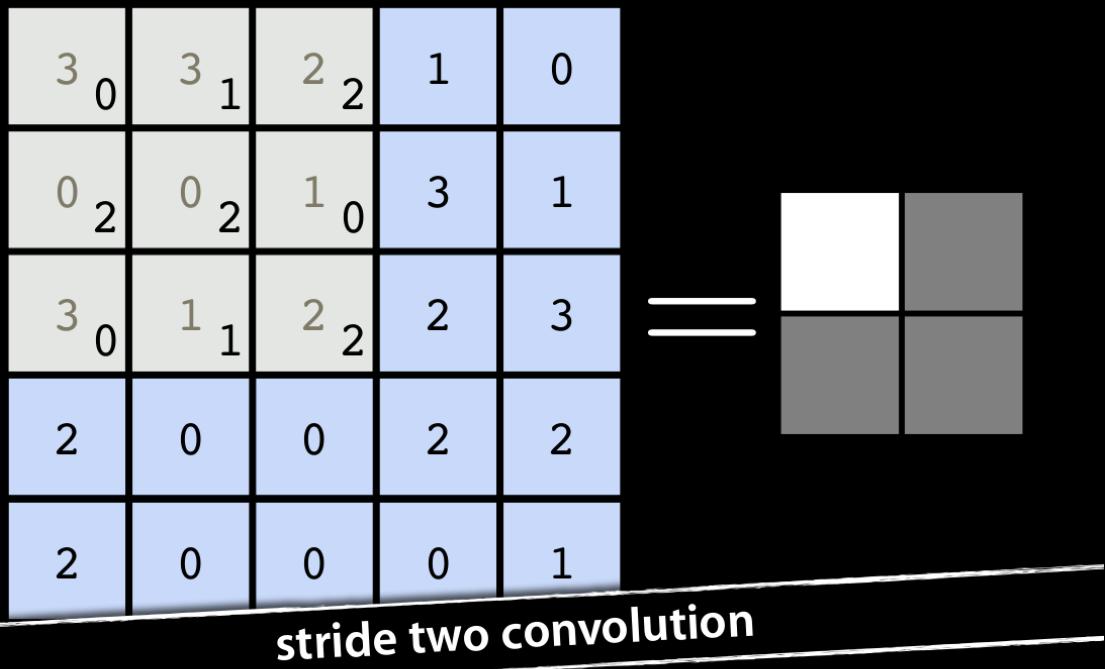
$$\begin{array}{ccccc}
 3 & 3 & 2 & 1 & 0 \\
 0 & 0 & 1 & 3 & 1 \\
 3 & 1_0 & 2_1 & 2_2 & 3 \\
 2 & 0_2 & 0_2 & 2_0 & 2 \\
 2 & 0_0 & 0_1 & 0_2 & 1
 \end{array}
 =
 \begin{array}{ccc}
 12 & 12 & 17 \\
 10 & 17 & 19 \\
 9 & &
 \end{array}$$

$$\begin{array}{ccccc}
 3 & 3 & 2 & 1 & 0 \\
 0 & 0 & 1 & 3 & 1 \\
 3 & 1_0 & 2_1 & 2_2 & 3 \\
 2 & 0_2 & 0_2 & 2_0 & 2 \\
 2 & 0_0 & 0_1 & 0_2 & 1
 \end{array}
 =
 \begin{array}{ccc}
 12 & 12 & 17 \\
 10 & 17 & 19 \\
 9 & 6 &
 \end{array}$$

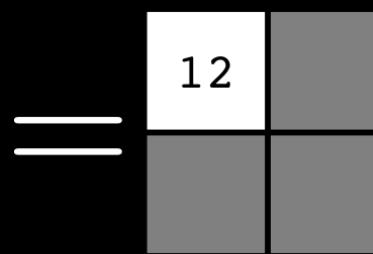
$$\begin{array}{ccccc}
 3 & 3 & 2 & 1 & 0 \\
 0 & 0 & 1 & 3 & 1 \\
 3 & 1 & 2_0 & 2_1 & 3_2 \\
 2 & 0 & 0_2 & 2_2 & 2_0 \\
 2 & 0 & 0_0 & 0_1 & 1_2
 \end{array}
 \quad = \quad
 \begin{array}{ccc}
 12 & 12 & 17 \\
 10 & 17 & 19 \\
 9 & 6 &
 \end{array}$$

$$\begin{array}{ccccc}
 3 & 3 & 2 & 1 & 0 \\
 0 & 0 & 1 & 3 & 1 \\
 3 & 1 & 2_0 & 2_1 & 3_2 \\
 2 & 0 & 0_2 & 2_2 & 2_0 \\
 2 & 0 & 0_0 & 0_1 & 1_2
 \end{array}
 \quad = \quad
 \begin{array}{ccc}
 12 & 12 & 17 \\
 10 & 17 & 19 \\
 9 & 6 & 14
 \end{array}$$

$$\begin{array}{ccccc} 3 & 0 & 3 & 1 & 2 & 2 \\ \hline 0 & 2 & 0 & 2 & 1 & 0 \\ \hline 3 & 0 & 1 & 1 & 2 & 2 \\ \hline 2 & 0 & 0 & 0 & 2 & 2 \\ \hline 2 & 0 & 0 & 0 & 0 & 1 \end{array} = \begin{array}{c|c} & \\ \hline & \\ \hline \end{array}$$



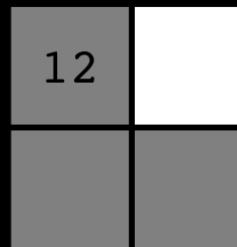
3	0	3	1	2	2	1	0
0	2	0	2	1	0	3	1
3	0	1	1	2	2	2	3
2	0	0	0	2	2	2	2
2	0	0	0	0	0	0	1



stride two convolution

3	3	2 0	1 1	0 2
0	0	1 2	3 2	1 0
3	1	2 0	2 1	3 2
2	0	0	2	2
2	0	0	0	1

=



3	3	2 0	1 1	0 2
0	0	1 2	3 2	1 0
3	1	2 0	2 1	3 2
2	0	0	2	2
2	0	0	0	1

=

12	17

3	3	2	1	0
0	0	1	3	1
3 0	1 1	2 2	2	3
2 2	0 2	0 0	2	2
2 0	0 1	0 2	0	1

=

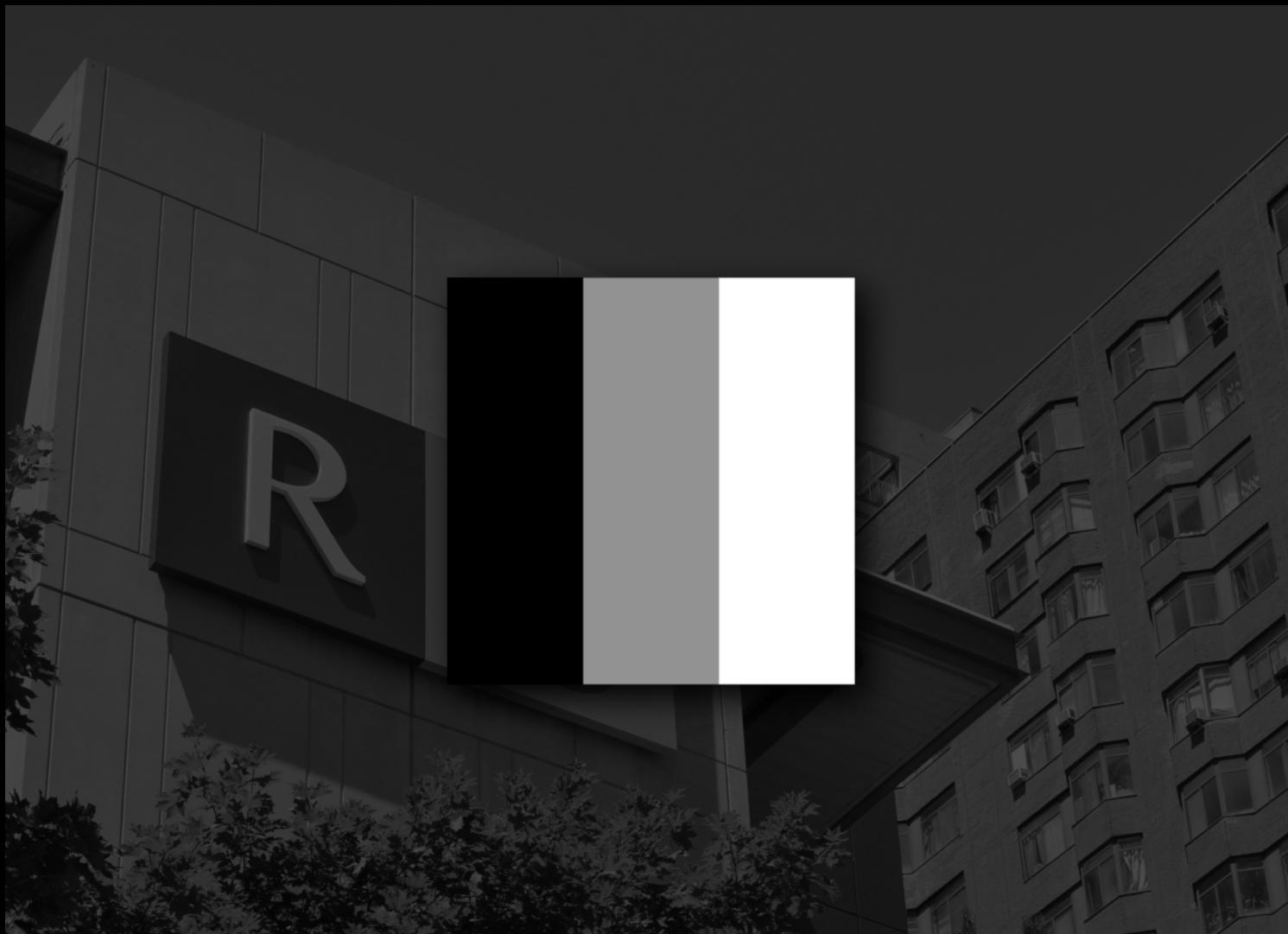
12	17

$$\begin{array}{ccccc}
 3 & 3 & 2 & 1 & 0 \\
 0 & 0 & 1 & 3 & 1 \\
 3 & 1 & 2_0 & 2_1 & 3_2 \\
 2 & 0 & 0_2 & 2_2 & 2_0 \\
 2 & 0 & 0_0 & 0_1 & 1_2
 \end{array}
 =
 \begin{array}{cc}
 12 & 17 \\
 9 &
 \end{array}$$

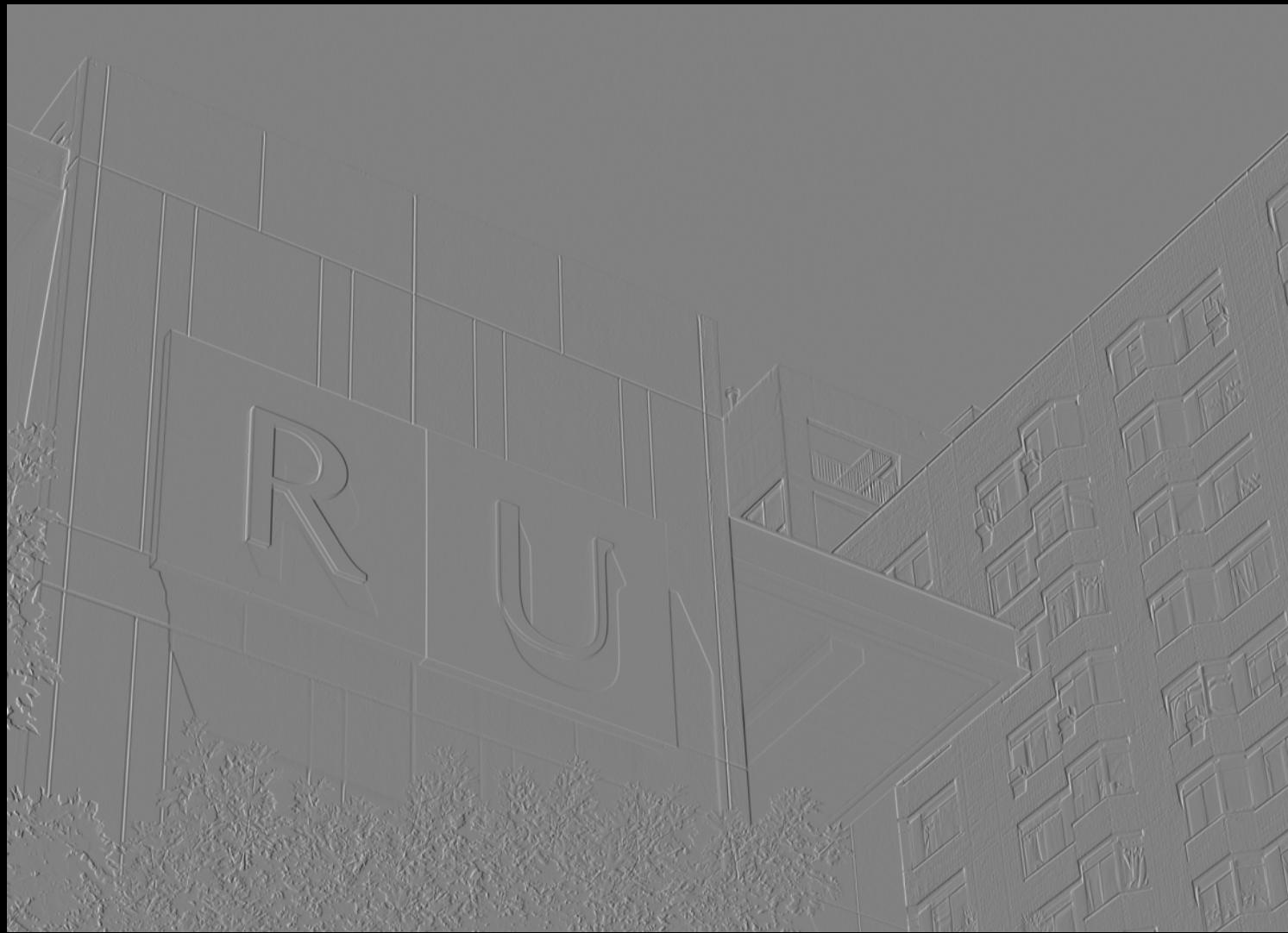
$$\begin{array}{ccccc}
 3 & 3 & 2 & 1 & 0 \\
 0 & 0 & 1 & 3 & 1 \\
 3 & 1 & 2_0 & 2_1 & 3_2 \\
 2 & 0 & 0_2 & 2_2 & 2_0 \\
 2 & 0 & 0_0 & 0_1 & 1_2
 \end{array}
 =
 \begin{array}{cc}
 12 & 17 \\
 9 & 14
 \end{array}$$



$$\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$





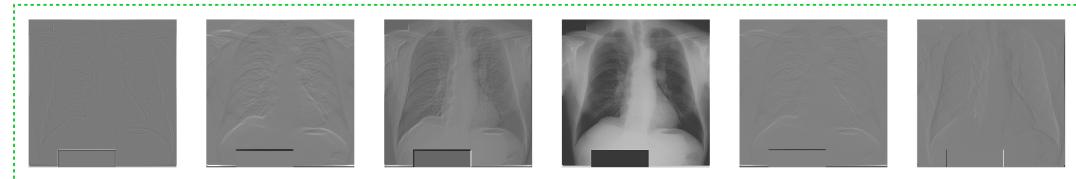




Оригінальний вхід

Виявлення меж	Фільтр Собеля	Ембосування	Різкість	Горизонтальний	Вертикальний
$\begin{matrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{matrix}$	$\begin{matrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{matrix}$	$\begin{matrix} -2 & -1 & 0 \\ -1 & 1 & 1 \\ 0 & 1 & 2 \end{matrix}$	$\begin{matrix} 0 & -1 & 0 \\ -1 & 8 & -1 \\ 0 & -1 & 0 \end{matrix}$	$\begin{matrix} -1 & -1 \\ 1 & 1 \end{matrix}$	$\begin{matrix} -1 & 1 \\ -1 & 1 \end{matrix}$

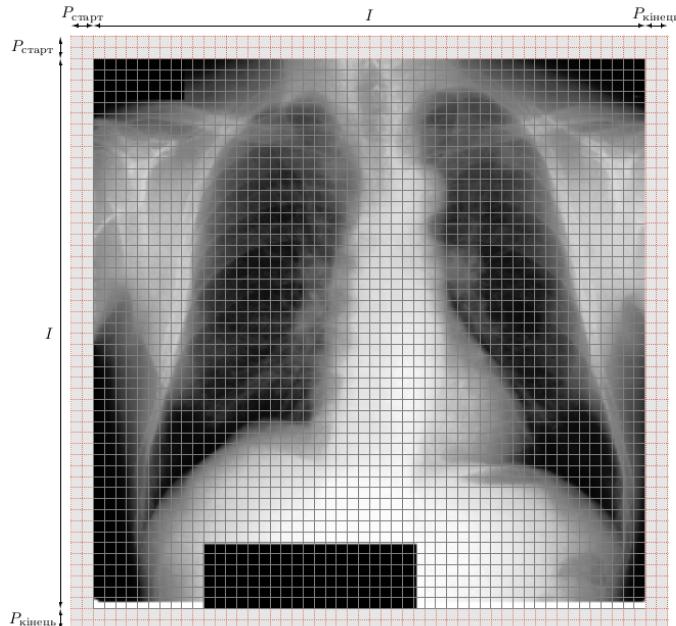
Фільтри



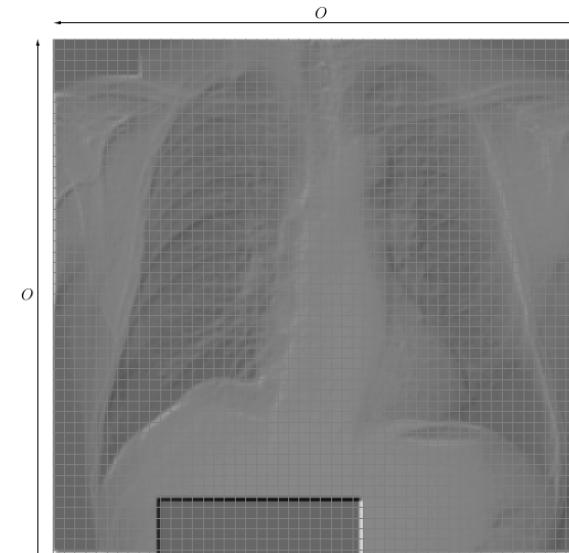
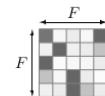
Карти ознак

Розмір карти ознак на виході згортки

$$O = \frac{I - F + P_{\text{start}} + P_{\text{end}}}{S} + 1$$



Вхід



Фільтр

Вихід

Агрегування (pooling)

max
pooling

1	1	4	2
7	3	9	5
1	1	7	3
4	2	8	6

Агрегування (pooling)

1	1	4	2
7	3	9	5
1	1	7	3
4	2	8	6

2-by-2 max pool with stride 2

Агрегування (pooling)

1	1	4	2
7	3	9	5
1	1	7	3
4	2	8	6

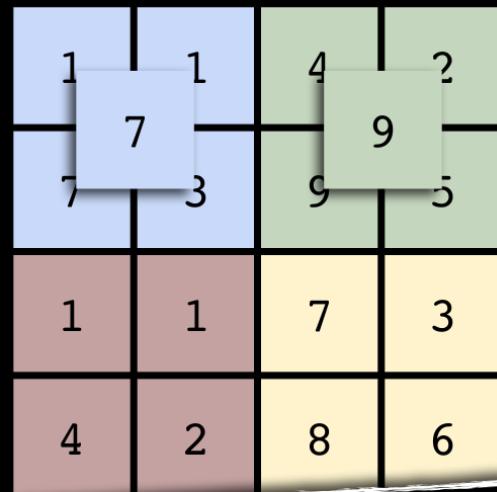
2-by-2 max pool with stride 2

Агрегування (pooling)

1	1	4	2
7	3	9	5
1	1	7	3
4	2	8	6

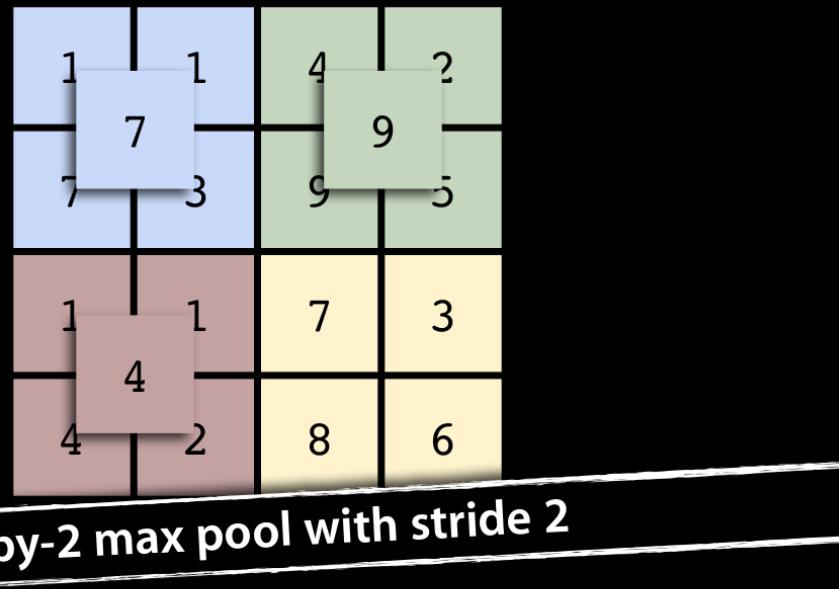
2-by-2 max pool with stride 2

Агрегування (pooling)

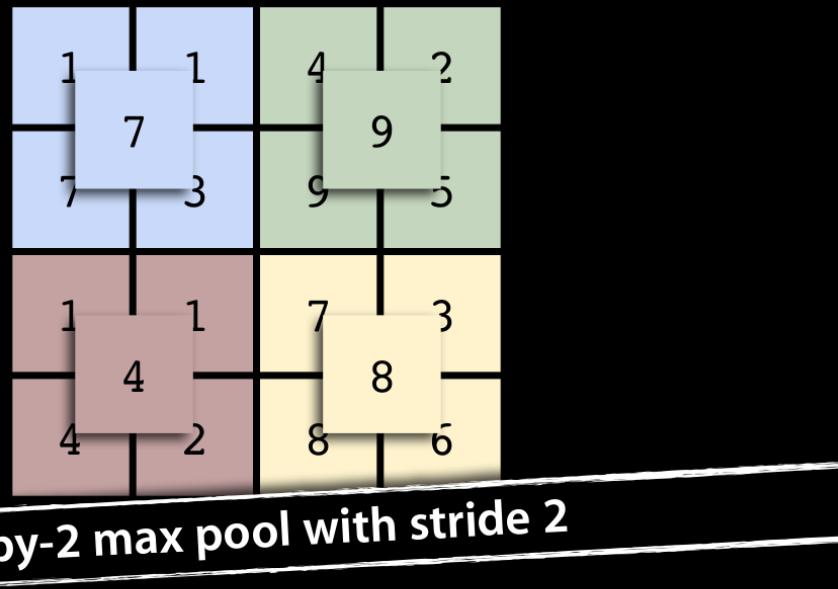


2-by-2 max pool with stride 2

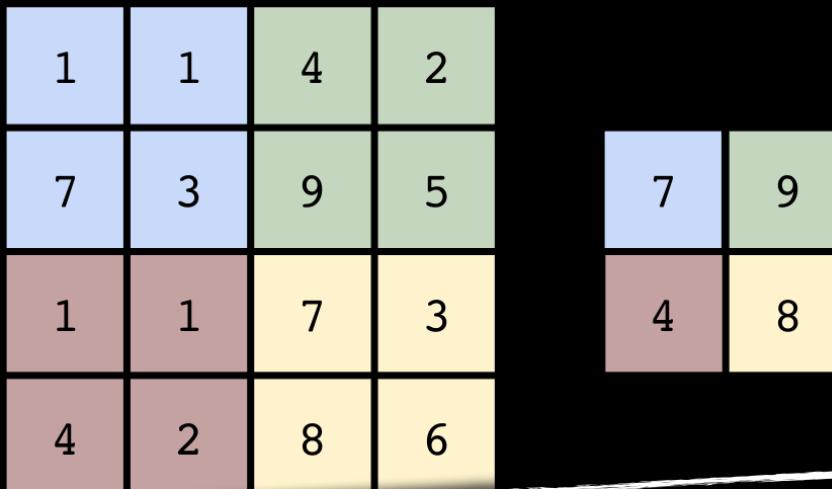
Агрегування (pooling)



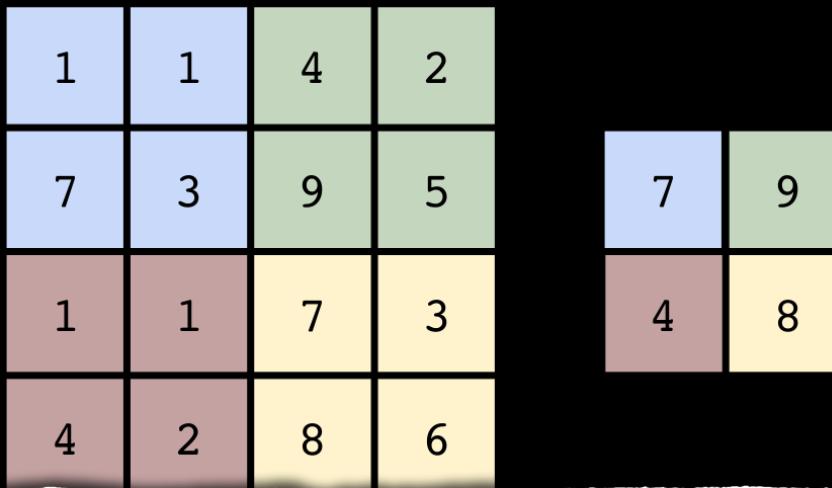
Агрегування (pooling)



Агрегування (pooling)



Агрегування (pooling)



$$h_k(x, y, c) = \max_{(i,j) \in \mathcal{N}(x,y)} h_{k-1}(i, j, c)$$

Демо

Як працює згортка?

Транспонована згортка

0	1
2	3

Вхід

Транспонована
згортка

0	1
2	3

Фільтр

=

0	0	
0	0	

+

	0	1
	2	3

+

0	2	
4	6	

+

	0	3
	6	9

0	0	1
0	4	6
4	12	9

Вихід

Транспонована згортка

0	1
2	3

Вхід

Транспонована
згортка
 $S = 2, P = 1$

0	1
2	3

Фільтр

=

0	0		
0	0		

+

		0	1
		2	3

+

0	2		
4	6		

+

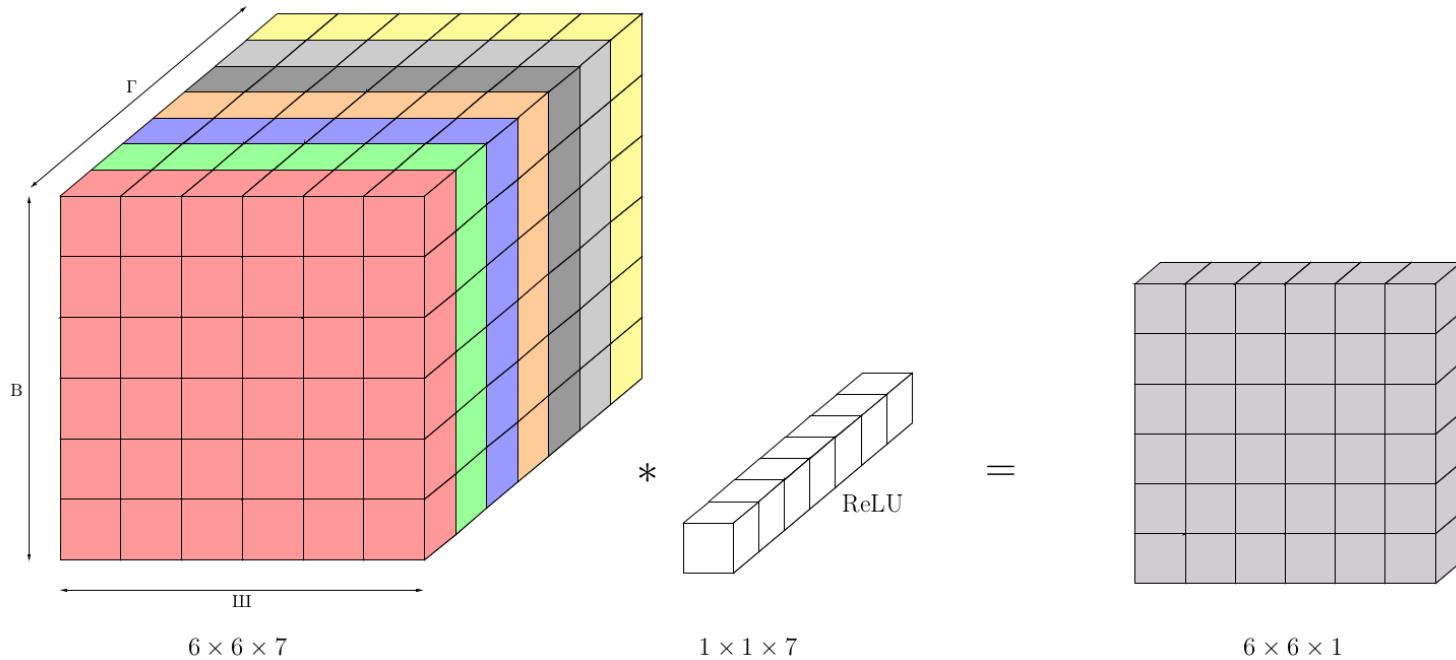
		0	3
		6	9

=

0	0	0	1
0	0	2	3
0	2	0	3
4	6	6	9

Вихід

1×1 згортка



Роздільна згортка

Кінець

Література

- LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *nature*, 521(7553), 436-444.
- Zhang, Aston and Lipton, Zachary C. and Li, Mu and Smola, Alexander J. (2021). [Dive into Deep Learning](#).