# 7.2

# How Convolutional Neural Networks Work

## Part 5: Controlling The Output Size With Padding

Sebastian Raschka and the Lightning AI Team

# How do we calculate the feature map size?

input width

kernel width

padding

$$O = \frac{W - K + 2P}{S} + 1$$

output width

stride

# How do we calculate the feature map size?

input width

kernel width

padding

$$O = \frac{W - K + 2P}{S} + 1$$

output width

stride

(The same formula works for "height")

# Example 1

input width

kernel width

padding

$$O = \frac{100 - 3 + 0}{1} + 1 \quad = 98$$

output width

stride

# Example 1

input width

kernel width

padding

$$O = \frac{100 - 3 + 0}{1} + 1 \quad = 98$$

output width

stride

```python
import torch
import torch.nn as nn

layer = nn.Conv2d(1, 1, kernel_size=3, padding=0, stride=1)

example = torch.rand(1, 100, 100)
layer(example).shape
```

```
torch.Size([1, 98, 98])
```

# Example 2

input width

kernel width

padding

$$O = \frac{100 - 5 + 0}{2} + 1 \quad = 48.5$$

output width

stride

# Example 2

input width

kernel width

padding

$$O = \frac{100 - 5 + 0}{2} + 1 \quad = 48.5$$

output width

stride

```python
import torch
import torch.nn as nn

layer = nn.Conv2d(1, 1, kernel_size=5, padding=0, stride=2)

example = torch.rand(1, 100, 100)
layer(example).shape
```

```
torch.Size([1, 48, 48])
```

# Example 2

input width

kernel width

padding

$$O = \left\lfloor \frac{100 - 5 + 0}{2} \right\rfloor + 1 \quad = 48.5$$

output width

stride

floor

```python
import torch
import torch.nn as nn

layer = nn.Conv2d(1, 1, kernel_size=5, padding=0, stride=2)

example = torch.rand(1, 100, 100)
layer(example).shape
```

```
torch.Size([1, 48, 48])
```

# Padding = 1

Will add a row/column of zeros to each size

| | | | | |
|---|---|---|---|---|
| 1 | -4 | 2 | 4 | -2 |
| -5 | 5 | 7 | 3 | -4 |
| 2 | -3 | 3 | -2 | 2 |
| -1 | 3 | 8 | 1 | 8 |
| 9 | 2 | -4 | 3 | 1 |

image with no padding

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | -4 | 2 | 4 | -2 | 0 |
| 0 | -5 | 5 | 7 | 3 | -4 | 0 |
| 0 | 2 | -3 | 3 | -2 | 2 | 0 |
| 0 | -1 | 3 | 8 | 1 | 8 | 0 |
| 0 | 9 | 2 | -4 | 3 | 1 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |

padding = 1
padding = (1, 1)

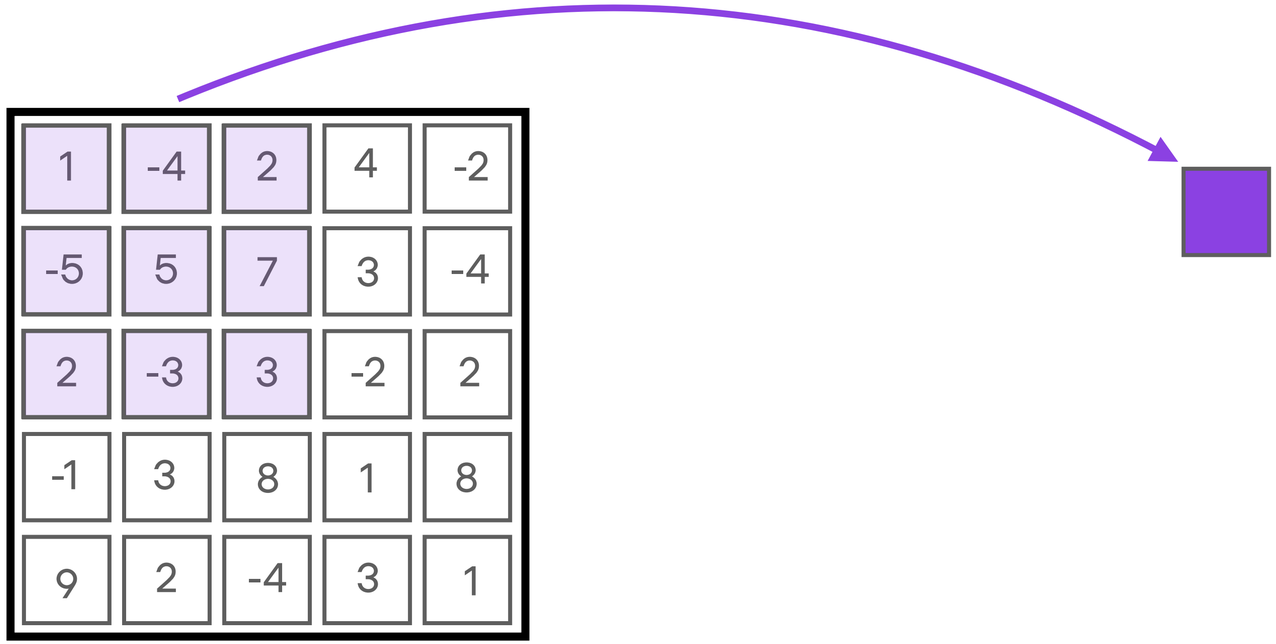



image with no padding

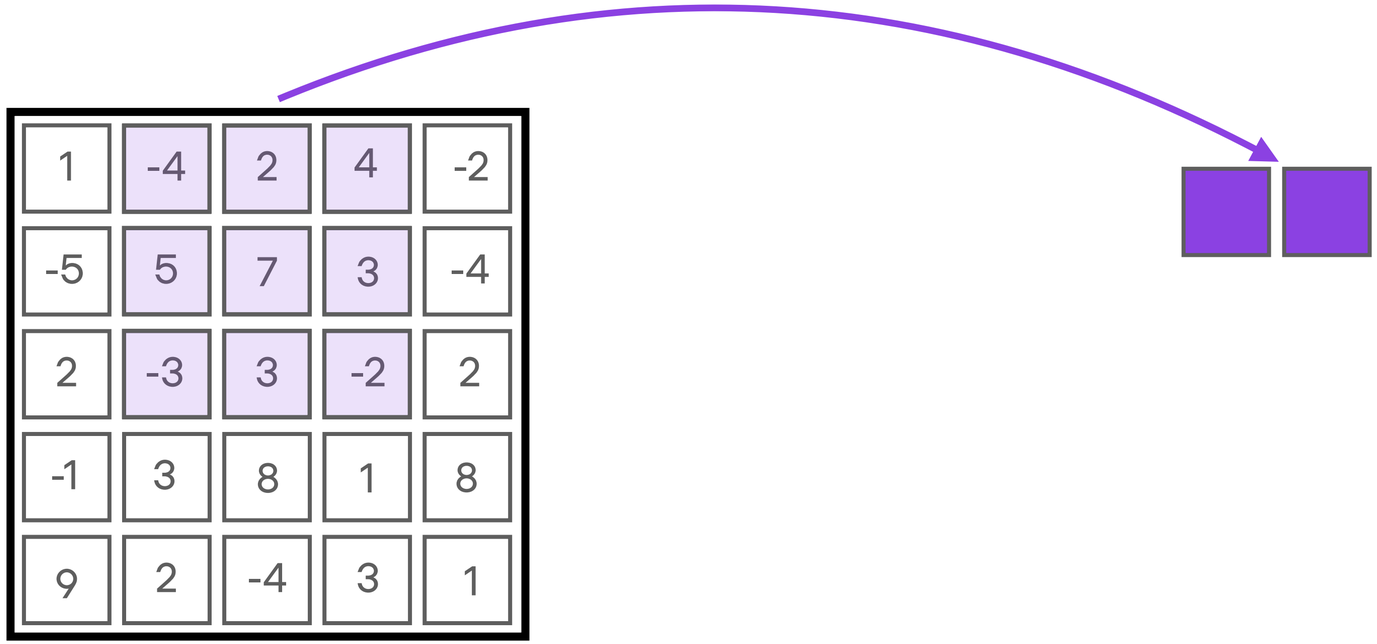



image with no padding

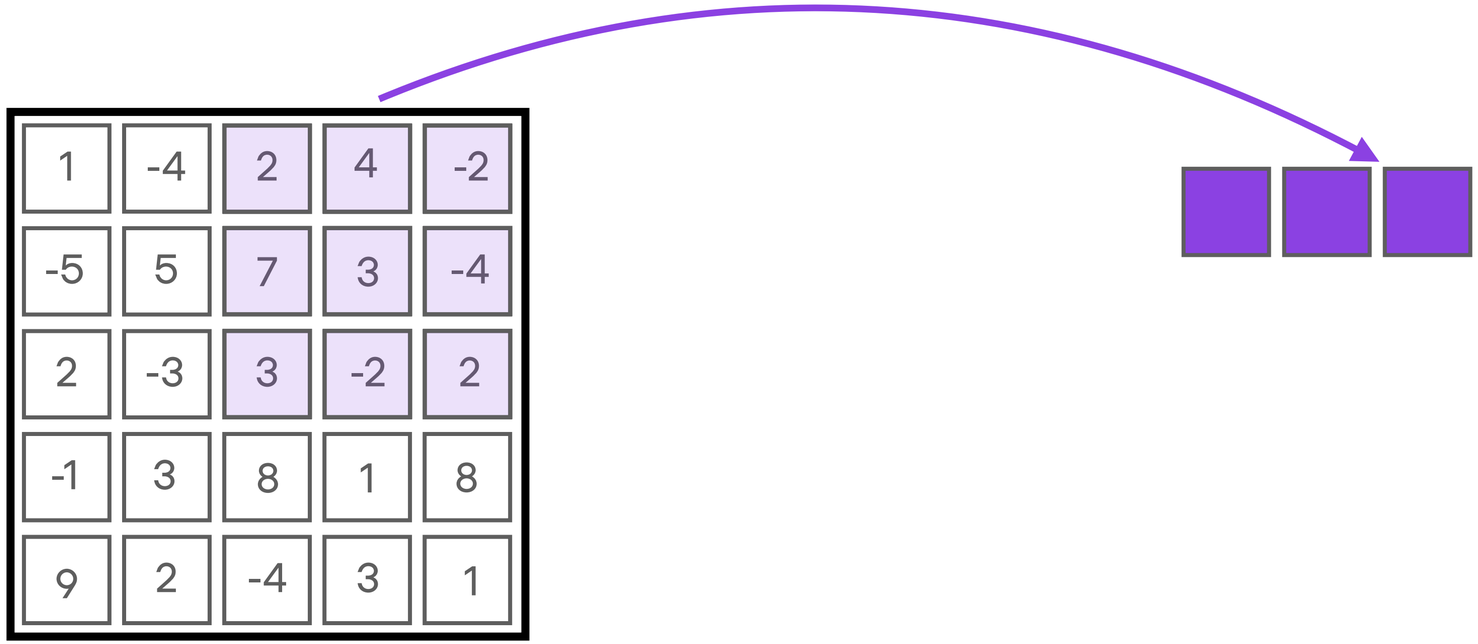
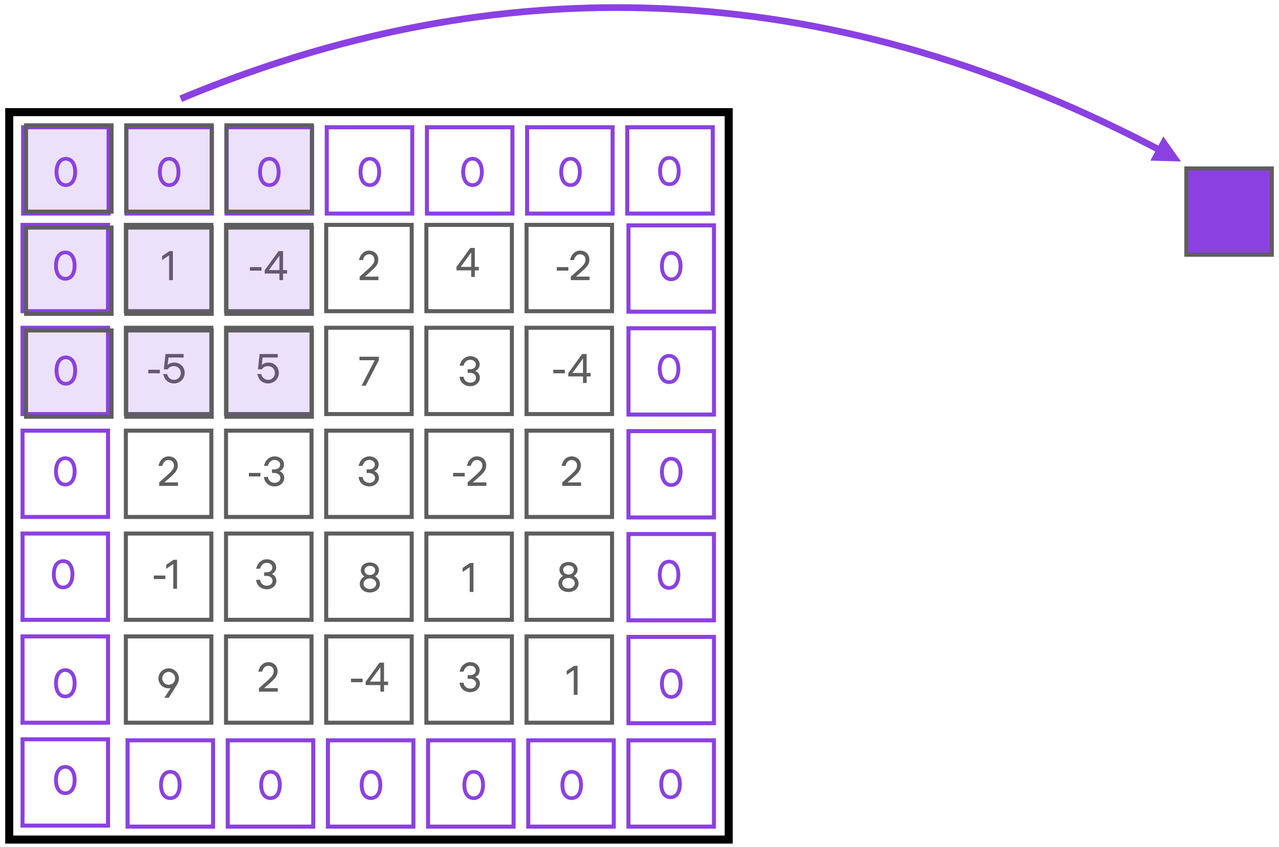


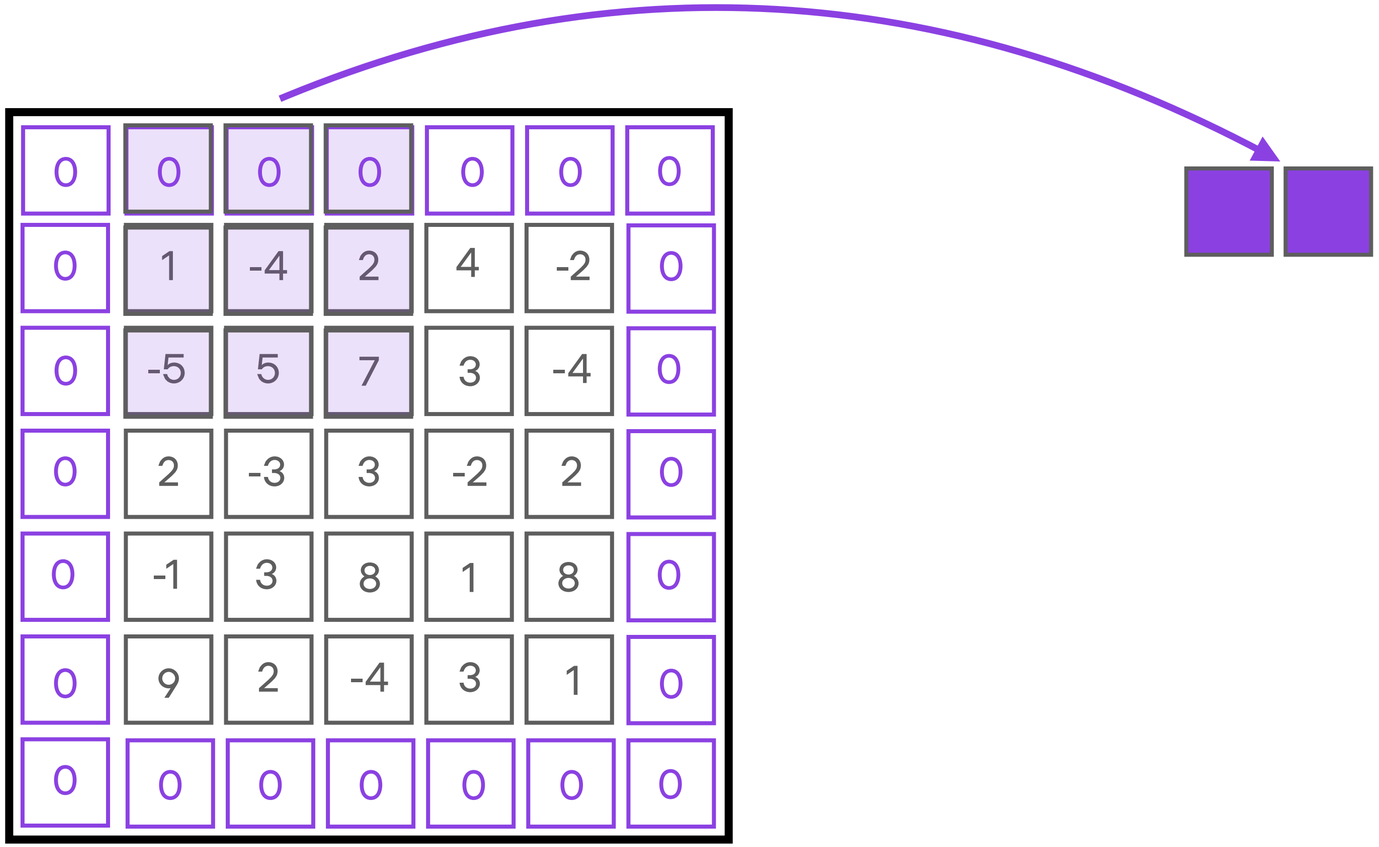

image with no padding

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | -4 | 2 | 4 | -2 | 0 |
| 0 | -5 | 5 | 7 | 3 | -4 | 0 |
| 0 | 2 | -3 | 3 | -2 | 2 | 0 |
| 0 | -1 | 3 | 8 | 1 | 8 | 0 |
| 0 | 9 | 2 | -4 | 3 | 1 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |

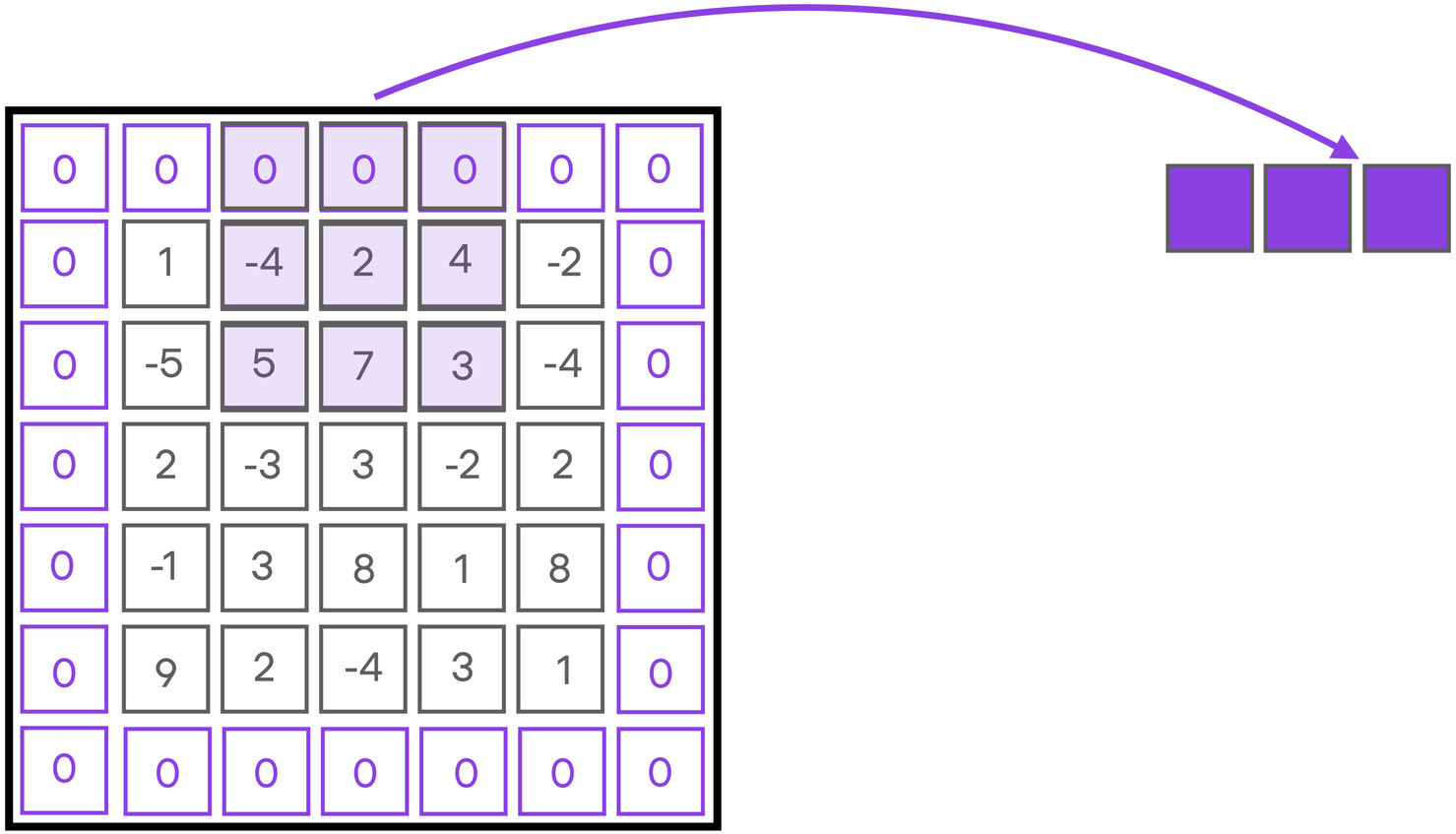
padding = 1
padding = (1, 1)



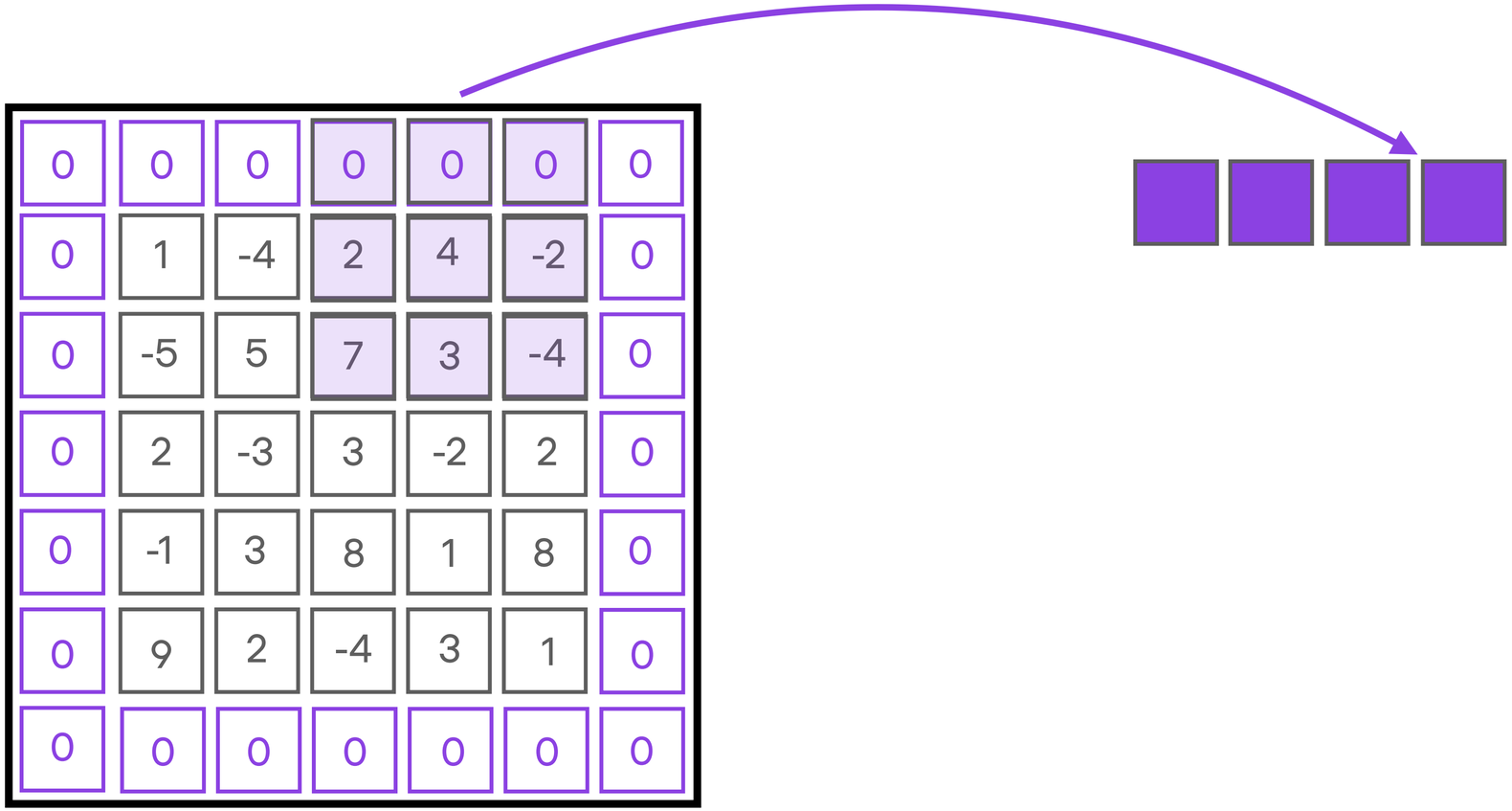

padding = 1
padding = (1, 1)



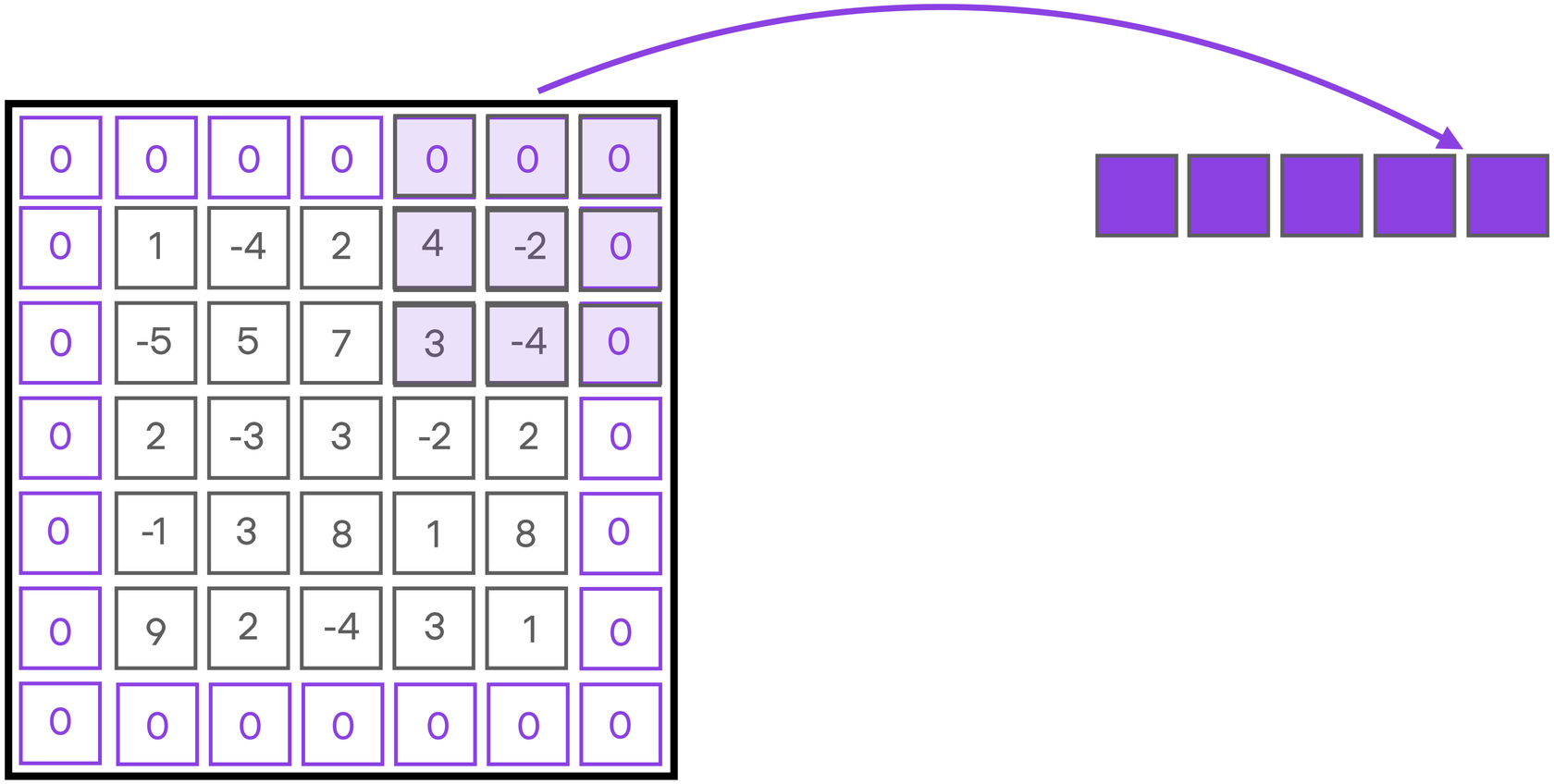

padding = 1
padding = (1, 1)



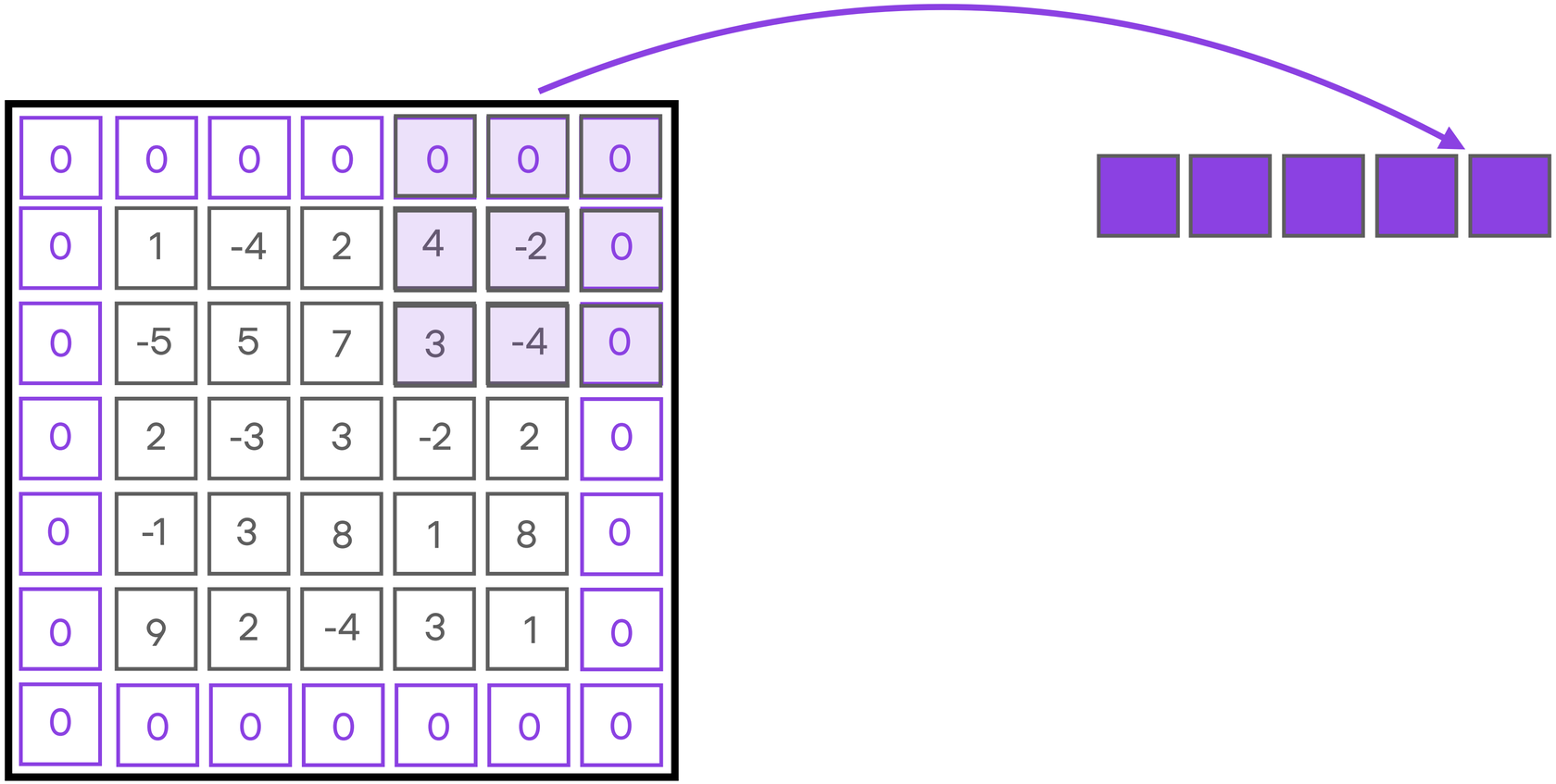

padding = 1
padding = (1, 1)




padding = 1
padding = (1, 1)




padding = 1
padding = (1, 1)




padding = 1
padding = (1, 1)

Or, instead of doing the math, we can use padding = "same"

Next: Let's take a look at some common CNN architectures