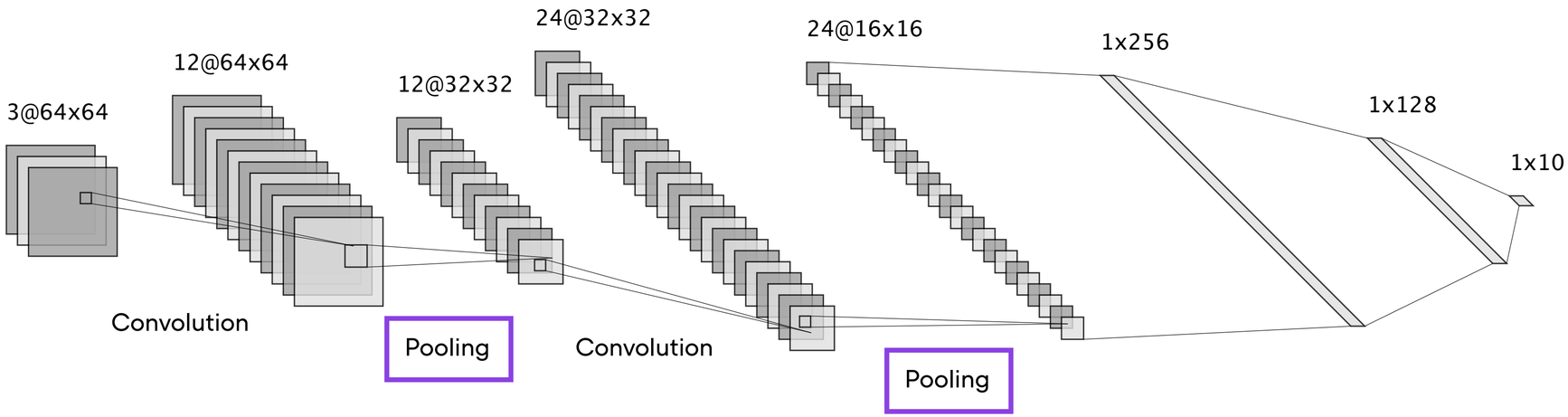
7.2

How Convolutional Neural Networks Work

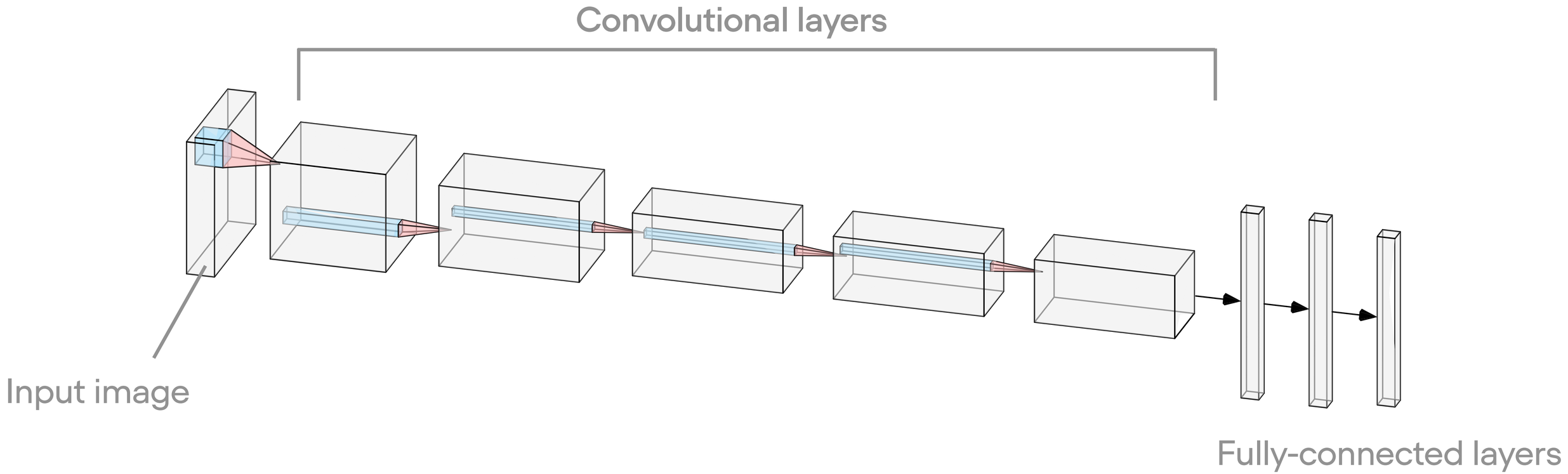
Part 4: What Are Pooling Layers?

Sebastian Raschka and the Lightning AI Team

Let's talk about pooling layers



CNN architectures are like a funnel



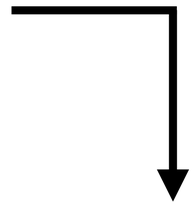
CNN architectures are like a funnel

Squeezing out information,
learning representations

CNN architectures are like a funnel

We decrease height and width
and increase the number of channels

We decrease height and width
and increase the number of channels



using convolutional layers

using pooling

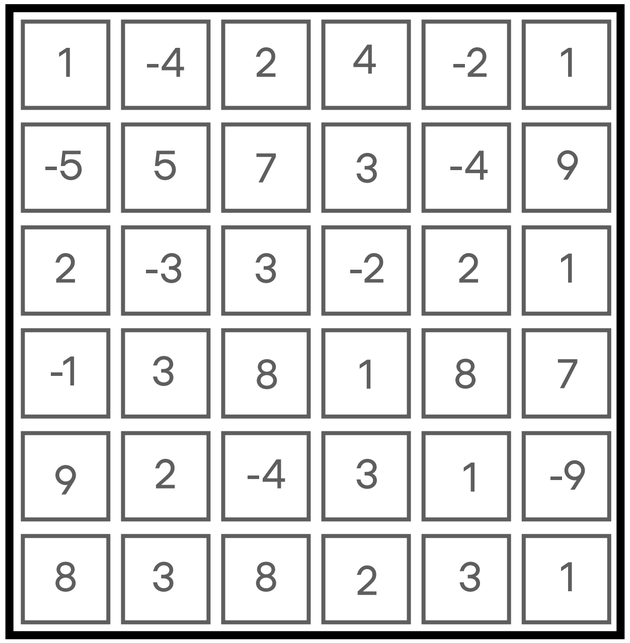
We decrease height and width

and increase the number of channels

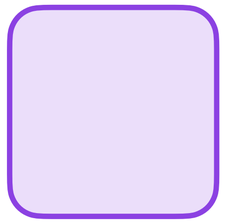
using convolutional layers

Max Pooling

with the following input:



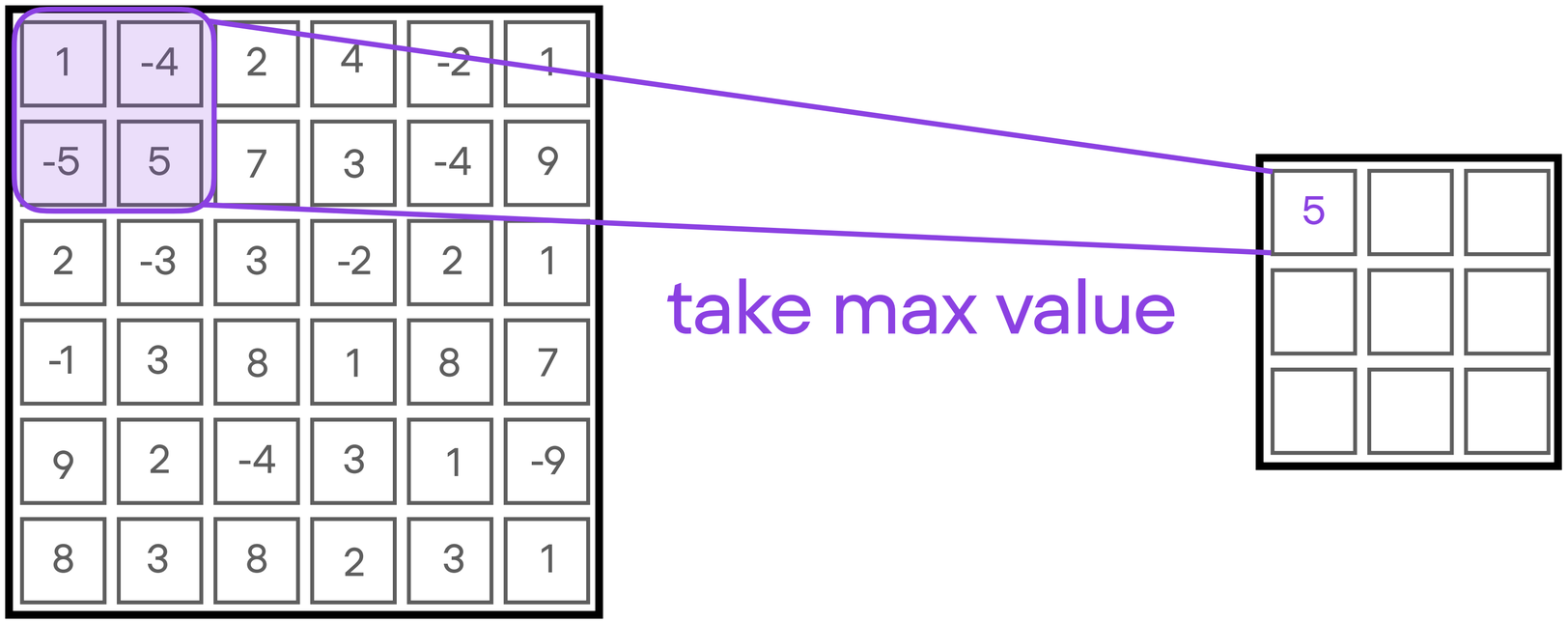
Max Pooling



and kernel size 2x2

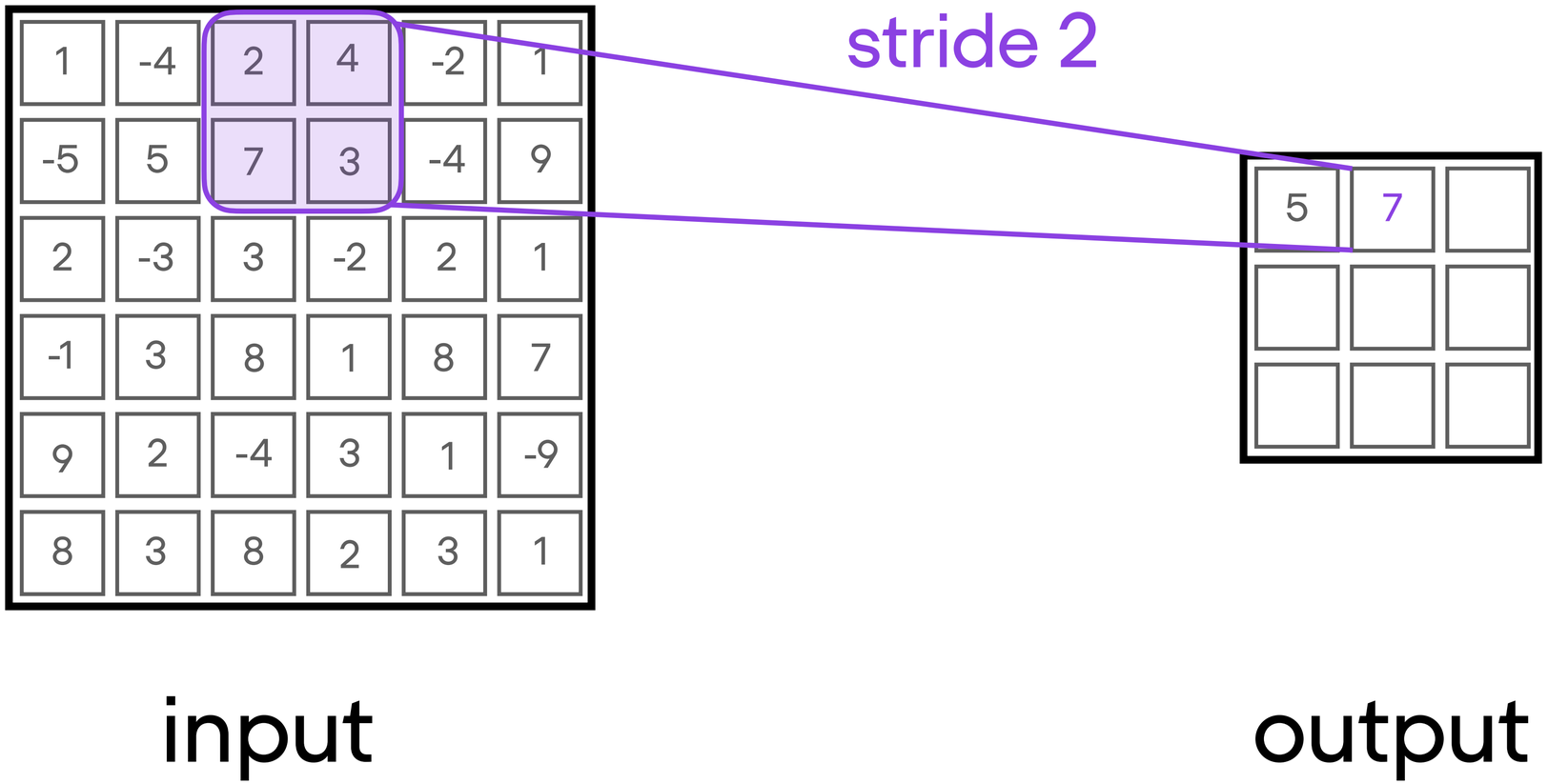
1	-4	2	4	-2	1
-5	5	7	3	-4	9
2	-3	3	-2	2	1
-1	3	8	1	8	7
9	2	-4	3	1	-9
8	3	8	2	3	1

Max Pooling

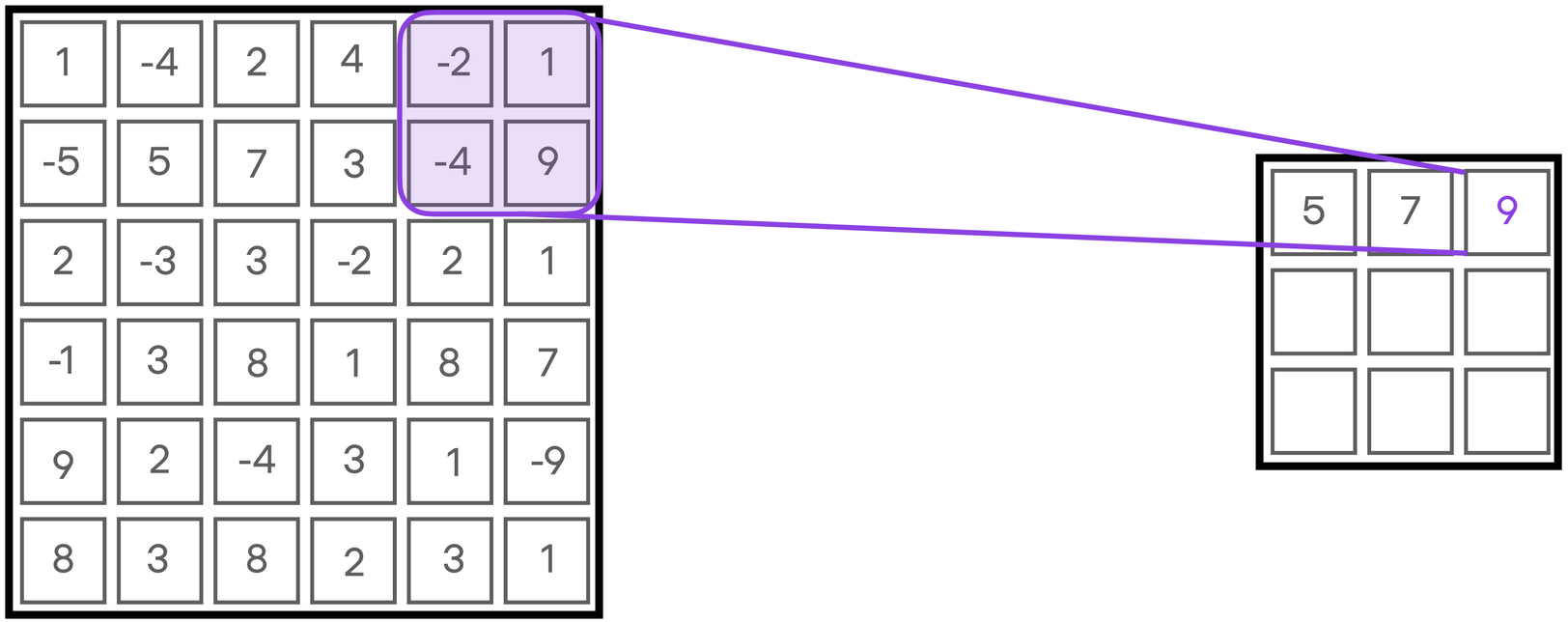


input output

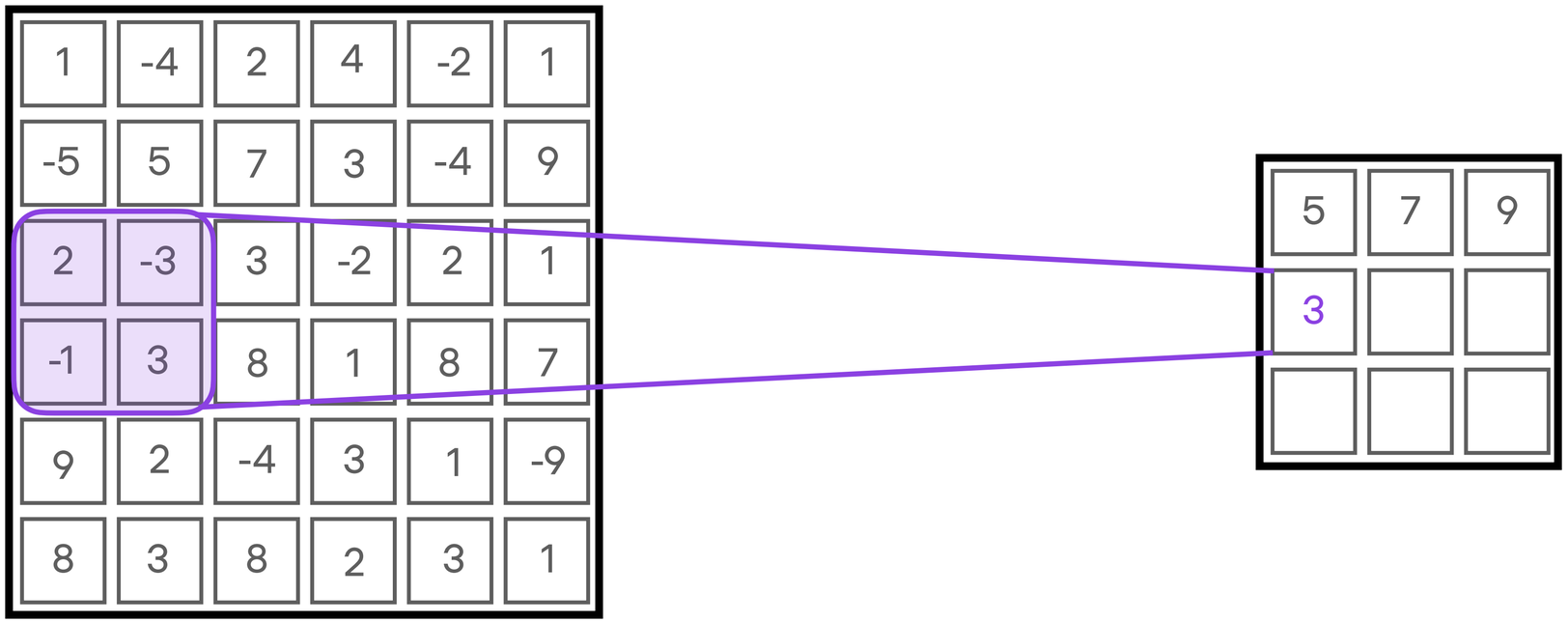
Max Pooling


1 -4 2 4 -2 1
-5 5 7 3 -4 9
2 -3 3 -2 2 1
-1 3 8 1 8 7
9 2 -4 3 1 -9
8 3 8 2 3 1
stride 2
5 7
input
output

Max Pooling

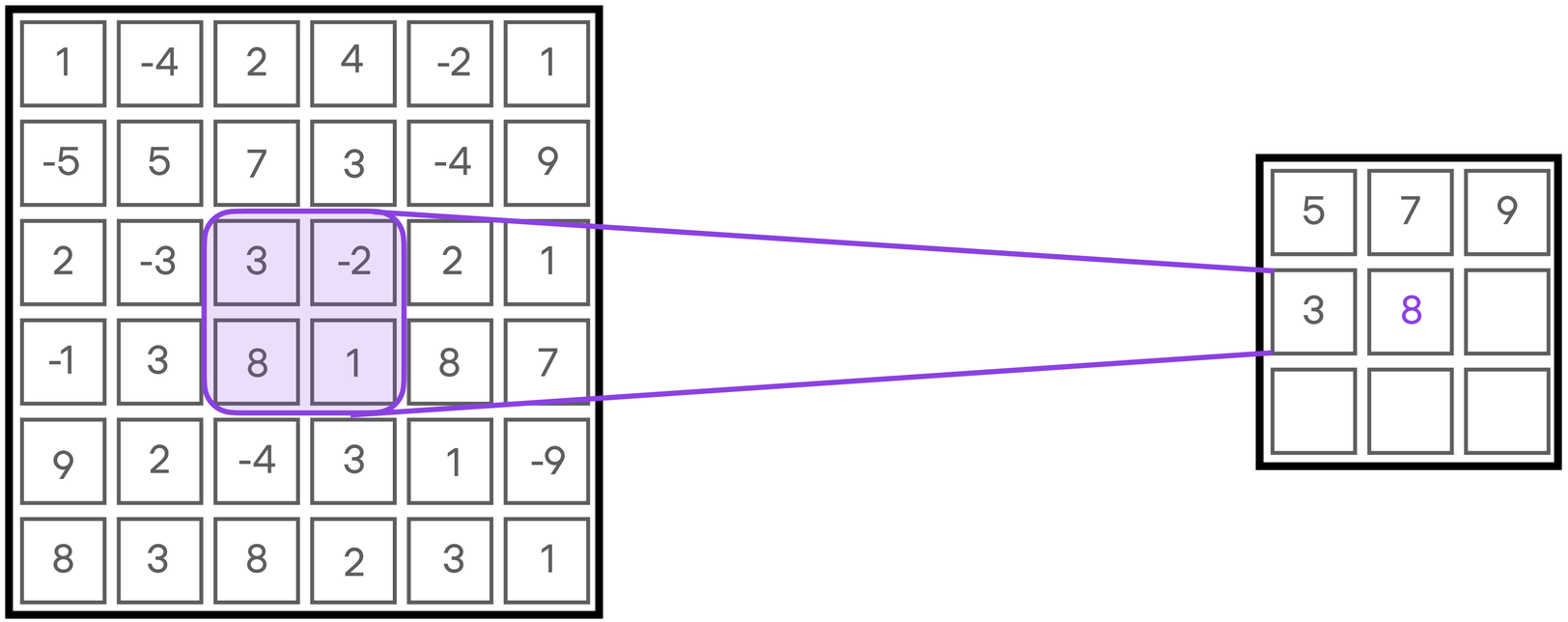

1 -4 2 4 -2 1
-5 5 7 3 -4 9
2 -3 3 -2 2 1
-1 3 8 1 8 7
9 2 -4 3 1 -9
8 3 8 2 3 1
5 7 9
input
output

Max Pooling



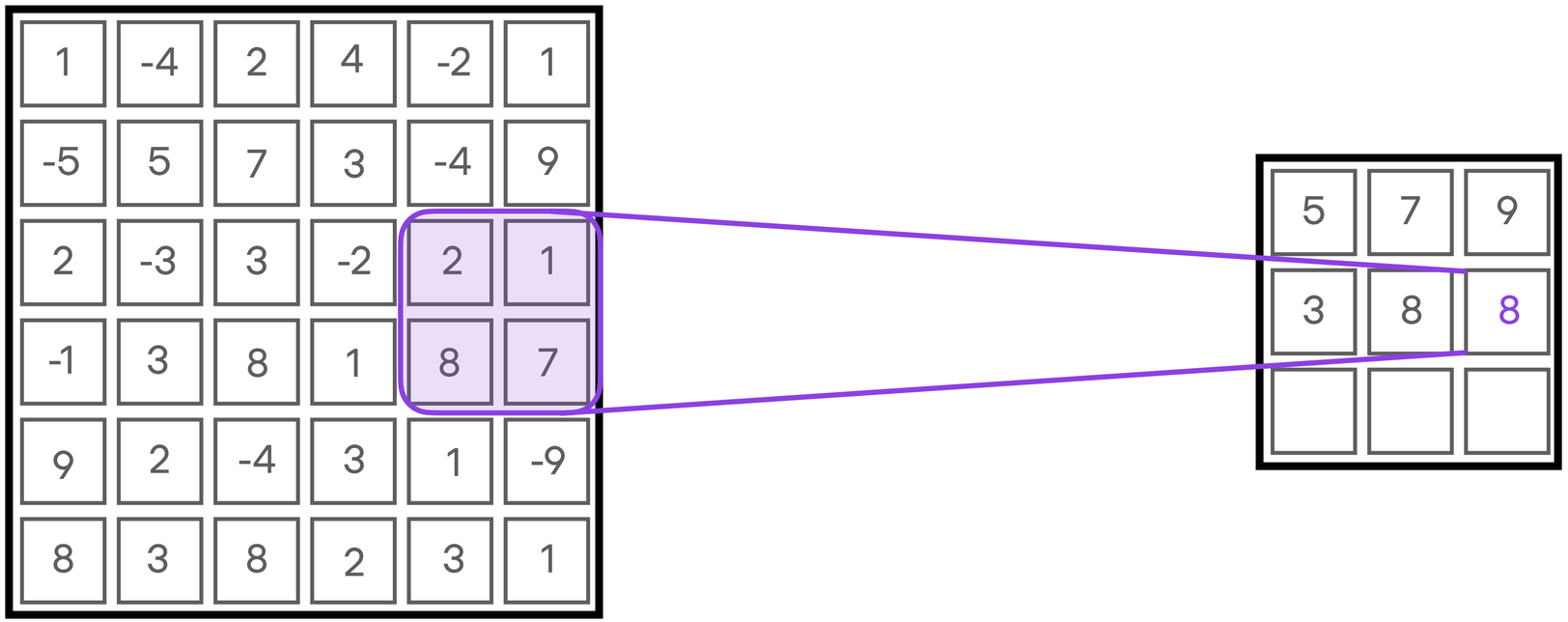
input output

Max Pooling



input output

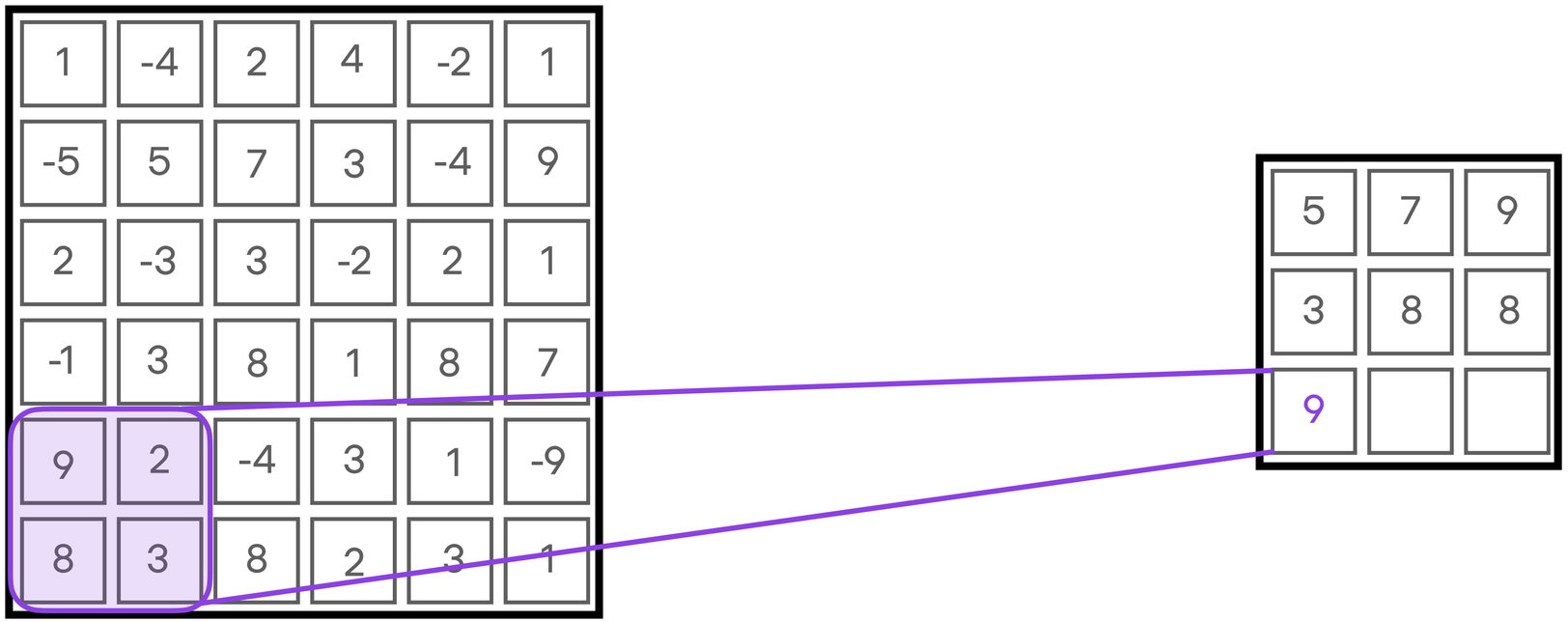
Max Pooling



input

output

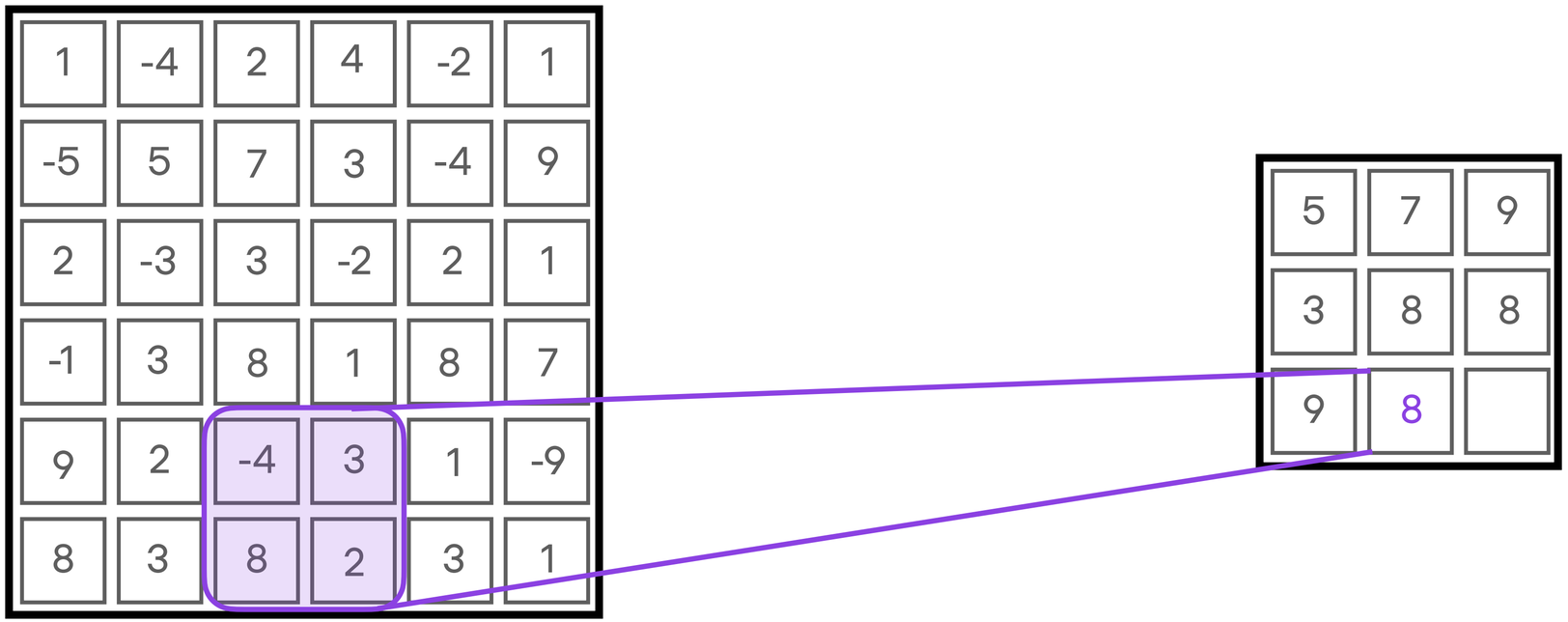
Max Pooling



input

output

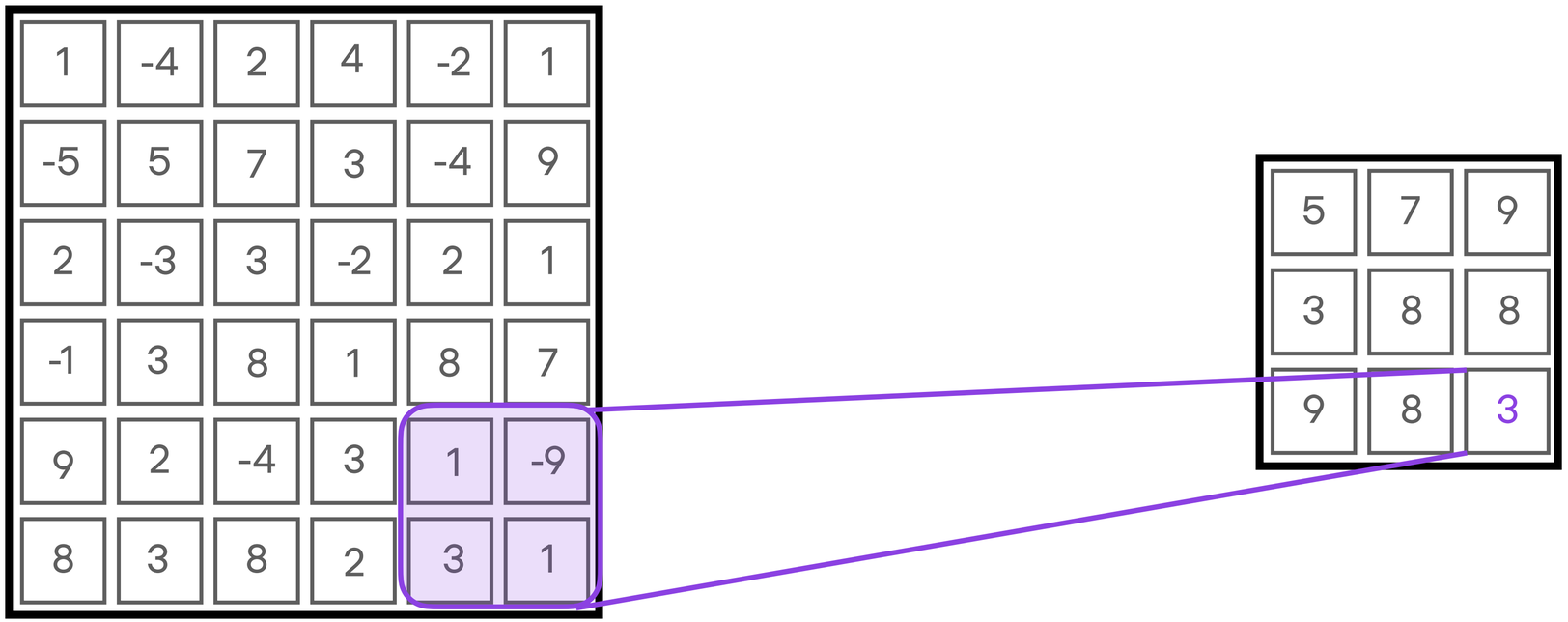
Max Pooling



input

output

Max Pooling

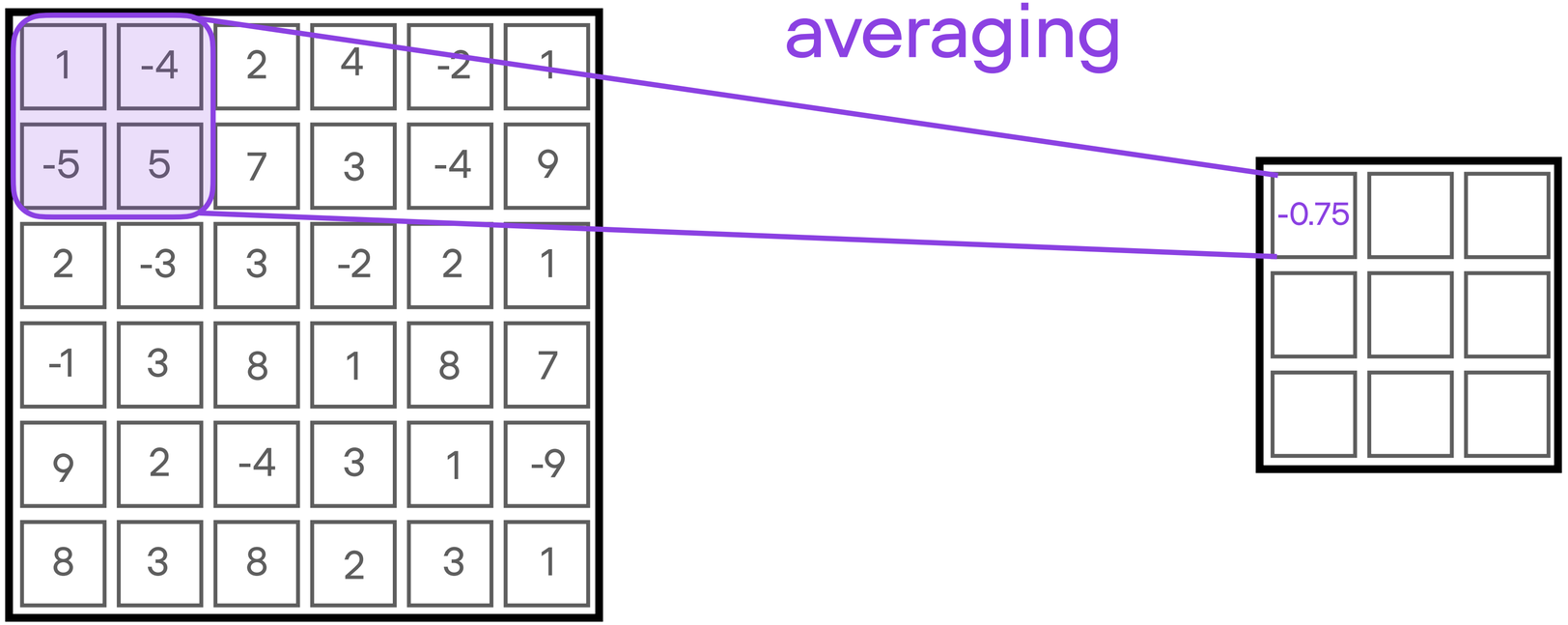


input

output

Average Pooling

Average Pooling



input output

Typical pooling layers don't have any learnable parameters

Pooling layers can help with local invariance

... but some information will be lost

Some architectures skip pooling layers altogether

Springenberg, Dosovitskiy, Brox, Riedmiller (2014). Striving for Simplicity: The All Convolutional Net. https://arxiv.org/abs/1412.6806

We can reduce the feature map size with more "stride"

A typical setup

```
import torch
import torch.nn as nn

layers_with_pooling = nn.Sequential(
    nn.Conv2d(3, 8, kernel_size=3),
    nn.MaxPool2d(kernel_size=2, stride=2),
    nn.Conv2d(8, 16, kernel_size=3),
    nn.MaxPool2d(kernel_size=2, stride=2),
)

example = torch.rand(3, 110, 110)
layers_with_pooling(example).shape
```

```
torch.Size([16, 26, 26])
```

A typical setup

```
import torch
import torch.nn as nn

layers_with_pooling = nn.Sequential(
    nn.Conv2d(3, 8, kernel_size=3),
    nn.MaxPool2d(kernel_size=2, stride=2),
    nn.Conv2d(8, 16, kernel_size=3),
    nn.MaxPool2d(kernel_size=2, stride=2),
)

example = torch.rand(3, 110, 110)
layers_with_pooling(example).shape
```

```
torch.Size([16, 26, 26])
```

without pooling layers

```
layers_no_pooling = nn.Sequential(
    nn.Conv2d(3, 8, kernel_size=3),
    #nn.MaxPool2d(kernel_size=2, stride=2),
    nn.Conv2d(8, 16, kernel_size=3),
    #nn.MaxPool2d(kernel_size=2, stride=2),
)

example = torch.rand(3, 110, 110)
layers_no_pooling(example).shape
```

```
torch.Size([16, 106, 106])
```

A typical setup

```
import torch
import torch.nn as nn

layers_with_pooling = nn.Sequential(
    nn.Conv2d(3, 8, kernel_size=3),
    nn.MaxPool2d(kernel_size=2, stride=2),
    nn.Conv2d(8, 16, kernel_size=3),
    nn.MaxPool2d(kernel_size=2, stride=2),
)

example = torch.rand(3, 110, 110)
layers_with_pooling(example).shape
```

```
torch.Size([16, 26, 26])
```

without pooling layers but with stride=2

```
import torch
import torch.nn as nn

layers_no_pooling_2 = nn.Sequential(
    nn.Conv2d(3, 8, kernel_size=3, stride=2),
    #nn.MaxPool2d(kernel_size=2, stride=2),
    nn.Conv2d(8, 16, kernel_size=3, stride=2),
    #nn.MaxPool2d(kernel_size=2, stride=2),
)

example = torch.rand(3, 110, 110)
layers_no_pooling_2(example).shape
```

```
torch.Size([16, 26, 26])
```

Next: More control over the output size via padding