AI VIETNAM
All-in-One Course

# Introduction to CNN

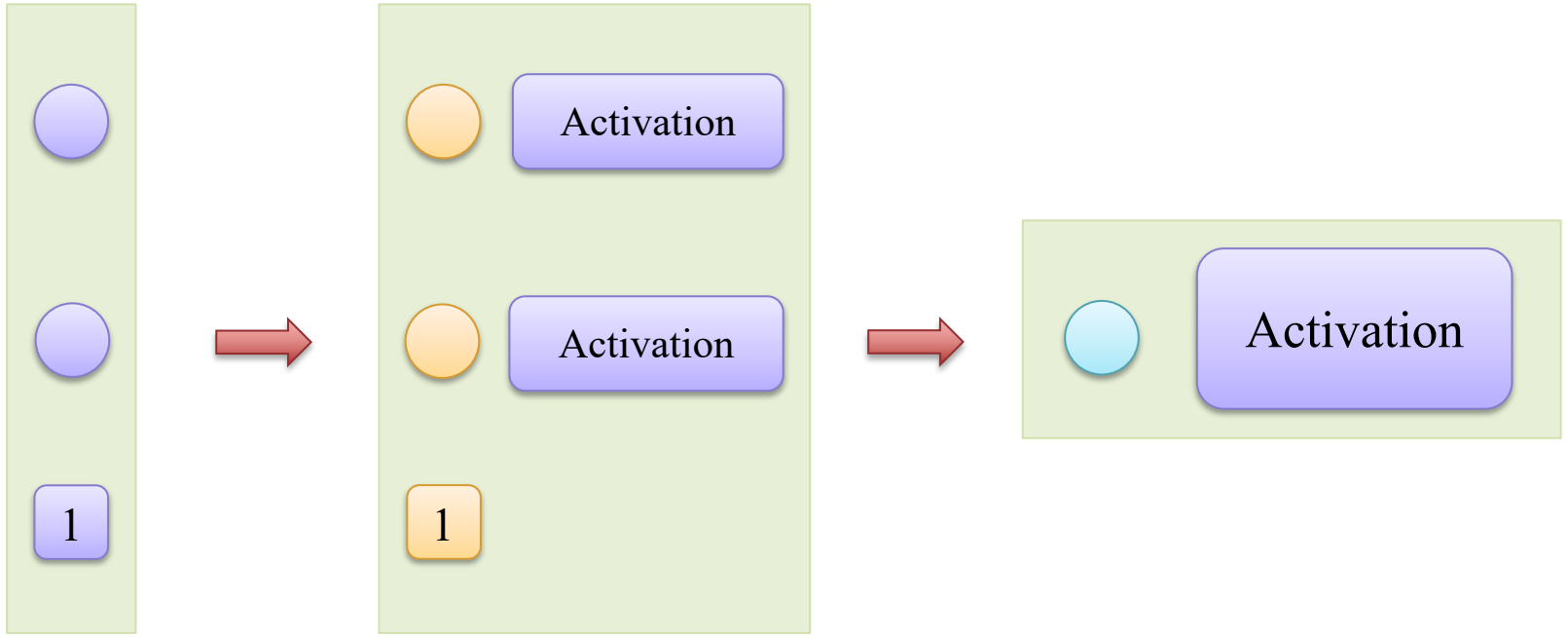
Nguyen Quoc Thai

Year 2023

# CONTENT

# 1 – Neural Network

## Neural Network




**Loss: CrossEntropyLoss**

$$L(\boldsymbol{\theta}) = -\sum_i y_i \log(\hat{y}_i)$$

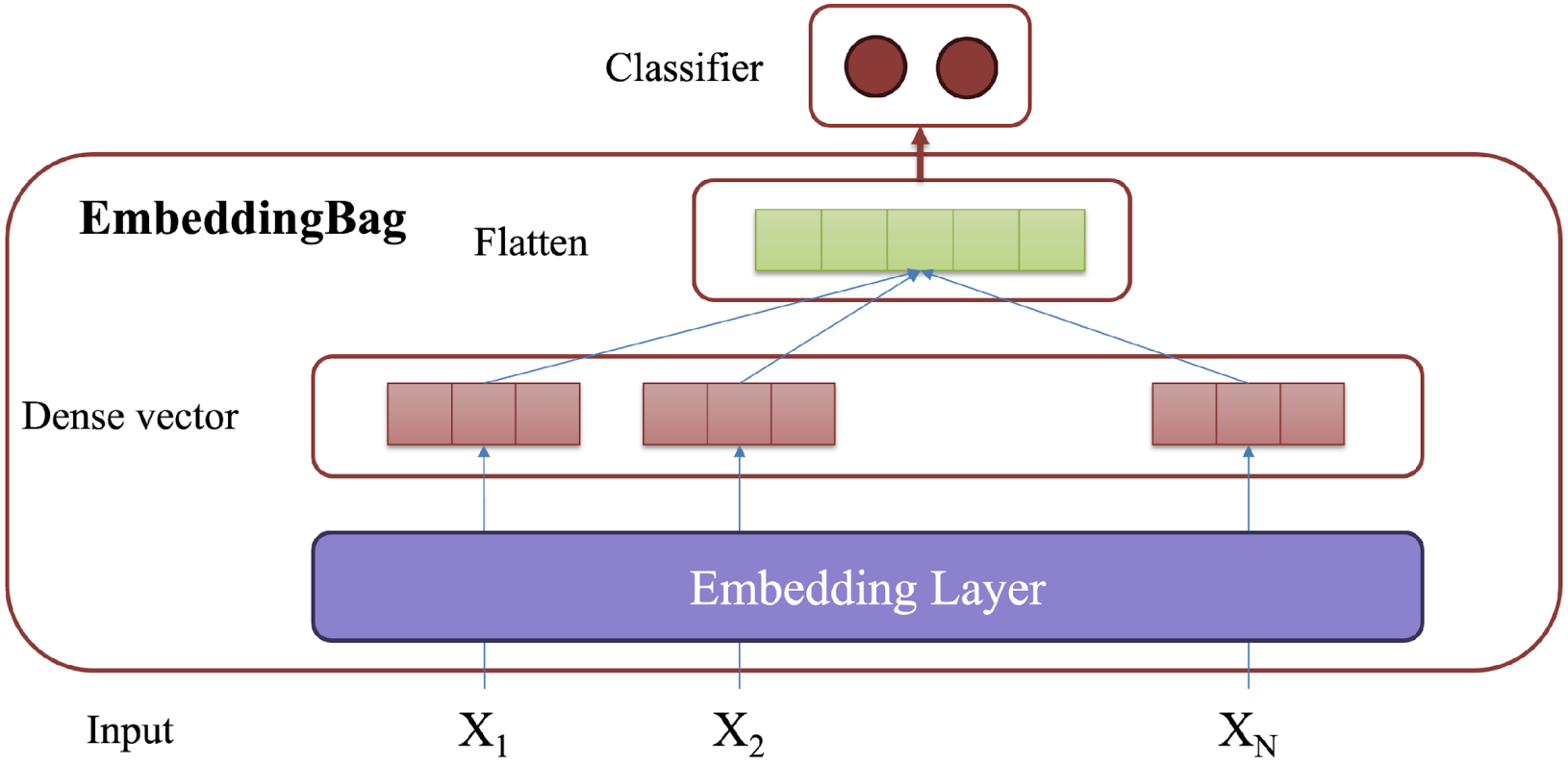
**Optimizer: SGD**

$$x = x - \eta * f'(x)$$

# 1 – Neural Network

## Neural Network for Text (Time Series)

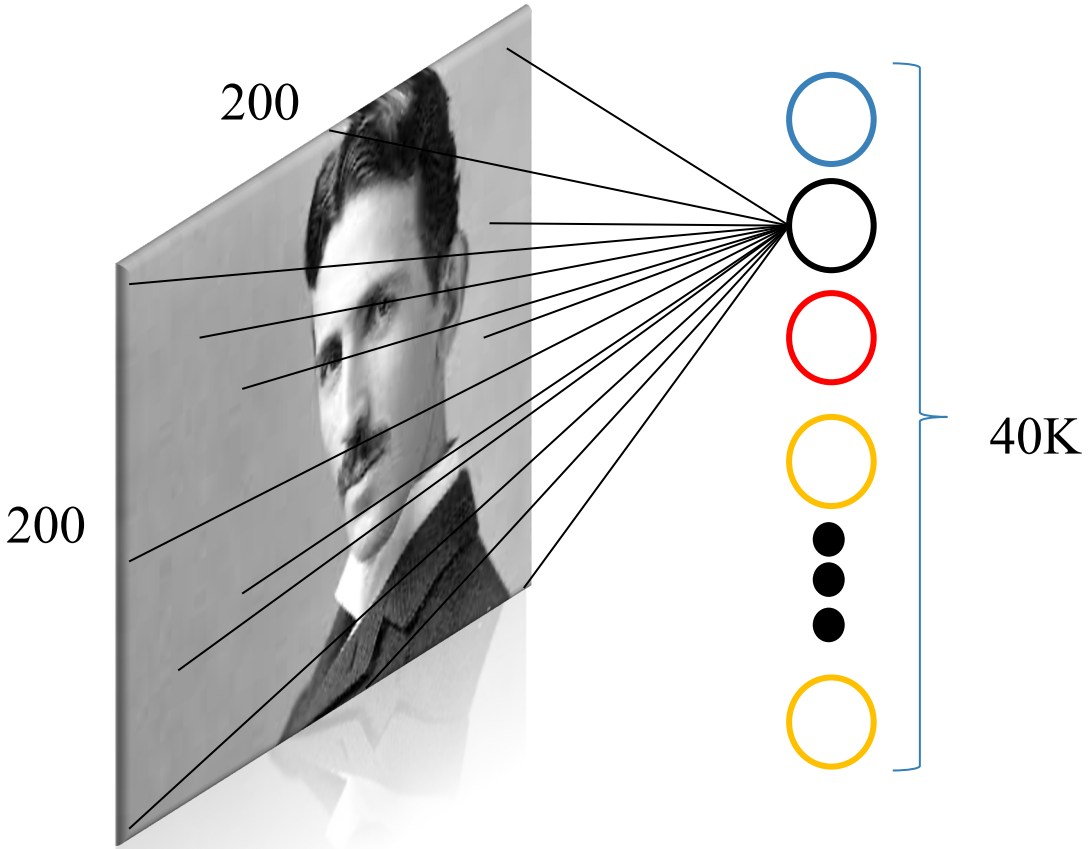
- No capture the order and importance of words in a sentence

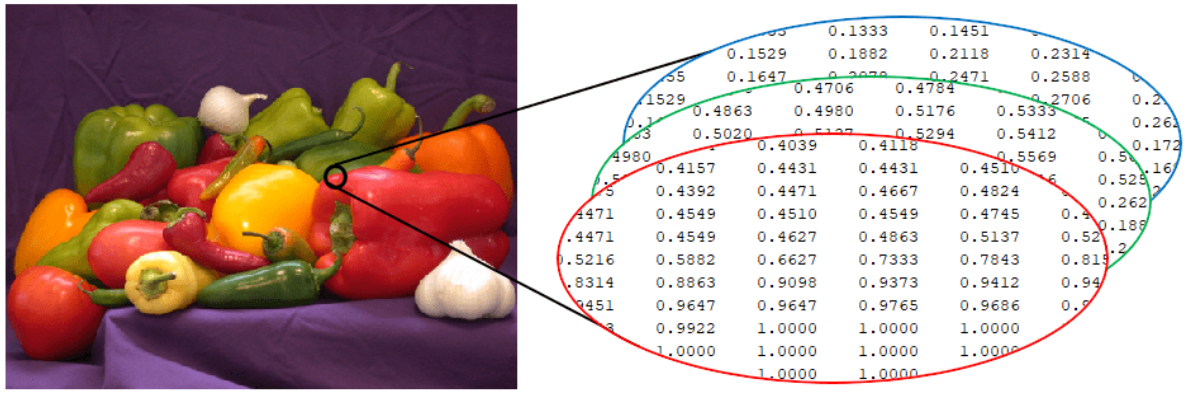


# 1 – Neural Network

## Neural Network for Image

- Each hidden node connects to all the other nodes



# 1 – Neural Network

## Neural Network

- **Need better network architectures…**

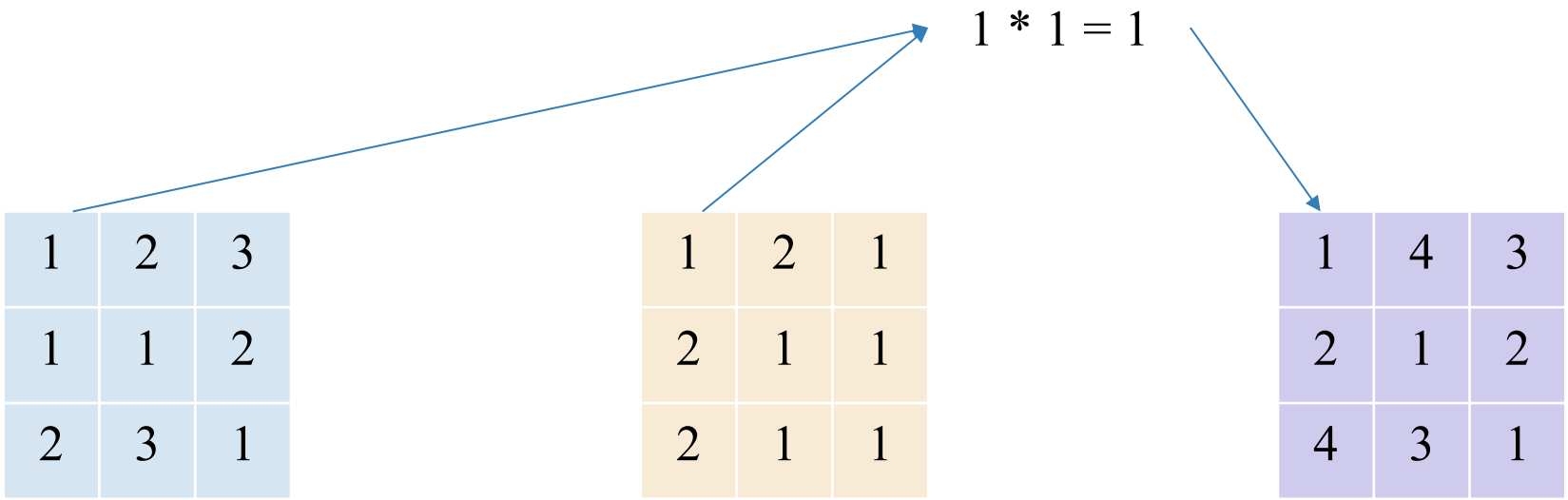
RNNs for Sequence

CNNs for Image

# 2 – Convolutional Layer

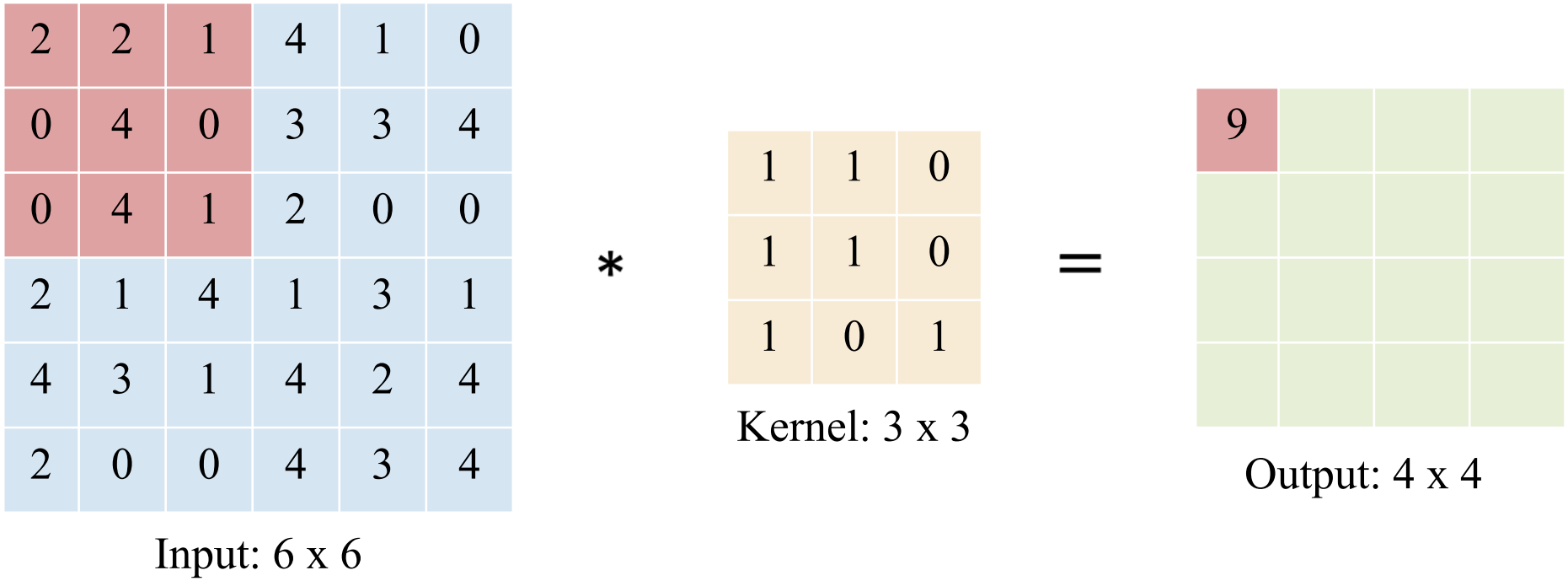
## Convolutional Operation

- **Element-wise Multiplication Matrix**
  - A (MxN) B (MxN) => C (MxN)



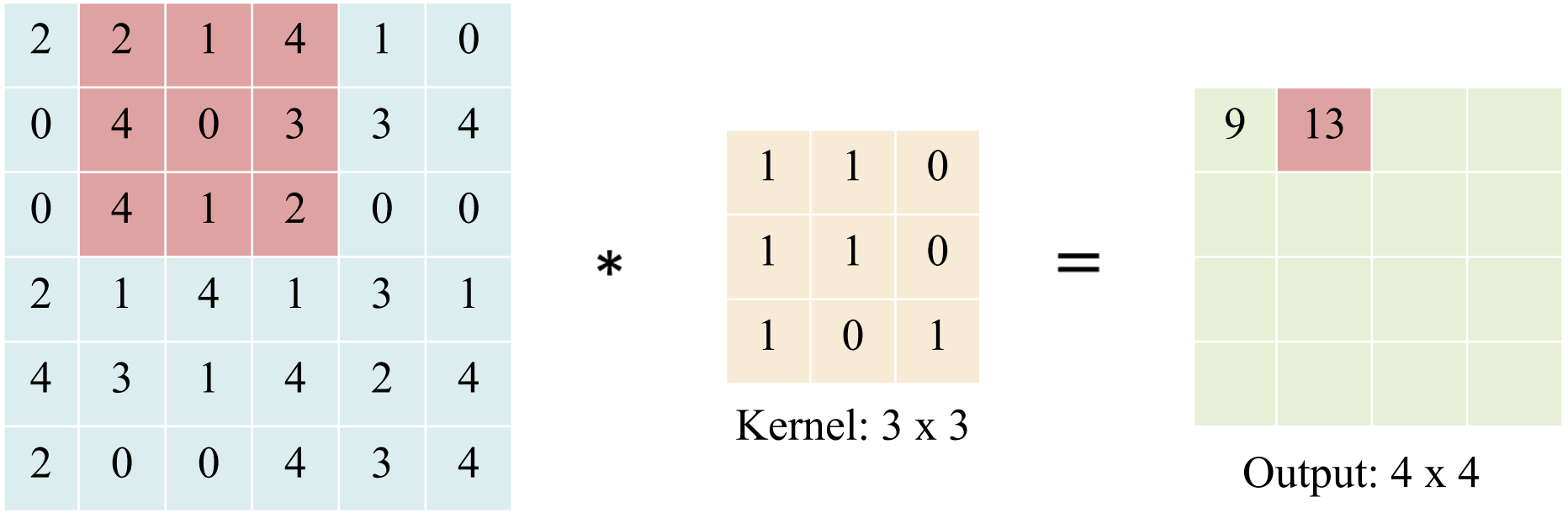
# 2 – Convolutional Layer

## Convolutional Operation



# 2 – Convolutional Layer

## Convolutional Operation



# 2 – Convolutional Layer

## Convolutional Operation

| | | | | | |
|---|---|---|---|---|---|
| 2 | 2 | 1 | 4 | 1 | 0 |
| 0 | 4 | 0 | 3 | 3 | 4 |
| 0 | 4 | 1 | 2 | 0 | 0 |
| 2 | 1 | 4 | 1 | 3 | 1 |
| 4 | 3 | 1 | 4 | 2 | 4 |
| 2 | 0 | 0 | 4 | 3 | 4 |

Input: 6 x 6

*

| | | |
|---|---|---|
| 1 | 1 | 0 |
| 1 | 1 | 0 |
| 1 | 0 | 1 |

Kernel: 3 x 3

=

| | | | |
|---|---|---|---|
| 9 | 13 | 9 | 13 |
| 14 | 11 | 13 | 10 |
| 12 | 17 | 11 | 14 |
| 12 | 13 | 13 | 18 |

Output: 4 x 4

# 2 – Convolutional Layer

## Convolutional Operation

### Pytorch

```
input = torch.randint(5, (1, 6, 6), dtype=torch.float32)
input
```

```
tensor([[[2., 2., 1., 4., 1., 0.],
         [0., 4., 0., 3., 3., 4.],
         [0., 4., 1., 2., 0., 0.],
         [2., 1., 4., 1., 3., 1.],
         [4., 3., 1., 4., 2., 4.],
         [2., 0., 0., 4., 3., 4.]]])
```

```
# define convolutional layer
conv_layer = nn.Conv2d(
    in_channels=1,
    out_channels=1,
    kernel_size=3, # create a kernel: 3 x 3
    bias=False
)
```

```
conv_layer.weight
```

```
Parameter containing:
tensor([[[[ 0.0520,  0.2693,  0.0364],
          [-0.1051,  0.0896, -0.0904],
          [ 0.1403,  0.2976,  0.1927]]]], requires_grad=True)
```

```
init_kernel_weight = torch.randint(
    high=2,
    size=(conv_layer.weight.data.shape),
    dtype=torch.float32
)
init_kernel_weight
```

```
tensor([[[[1., 1., 0.],
          [1., 1., 0.],
          [1., 0., 1.]]]])
```

```
# init weight
conv_layer.weight.data = init_kernel_weight
```

```
conv_layer.weight
```

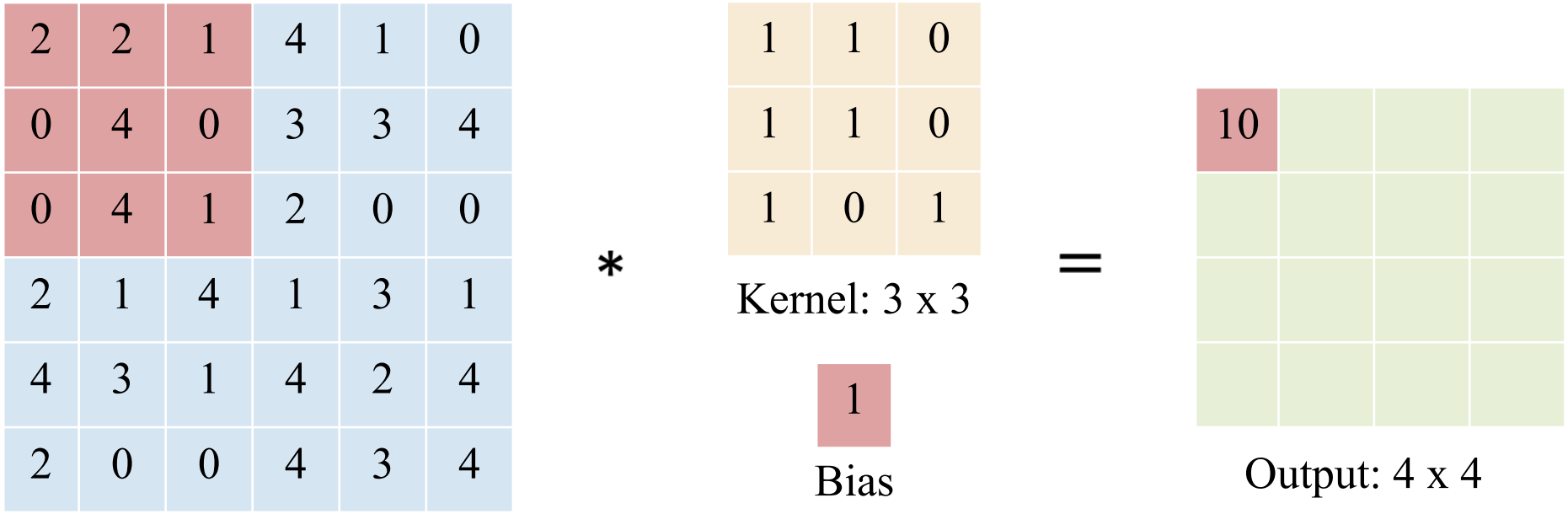
```
Parameter containing:
tensor([[[[1., 1., 0.],
          [1., 1., 0.],
          [1., 0., 1.]]]], requires_grad=True)
```

```
output = conv_layer(input)
output
```

```
tensor([[[ 9., 13.,  9., 13.],
         [14., 11., 13., 10.],
         [12., 17., 11., 14.],
         [12., 13., 13., 18.]]], grad_fn=<SqueezeBackward1>)
```

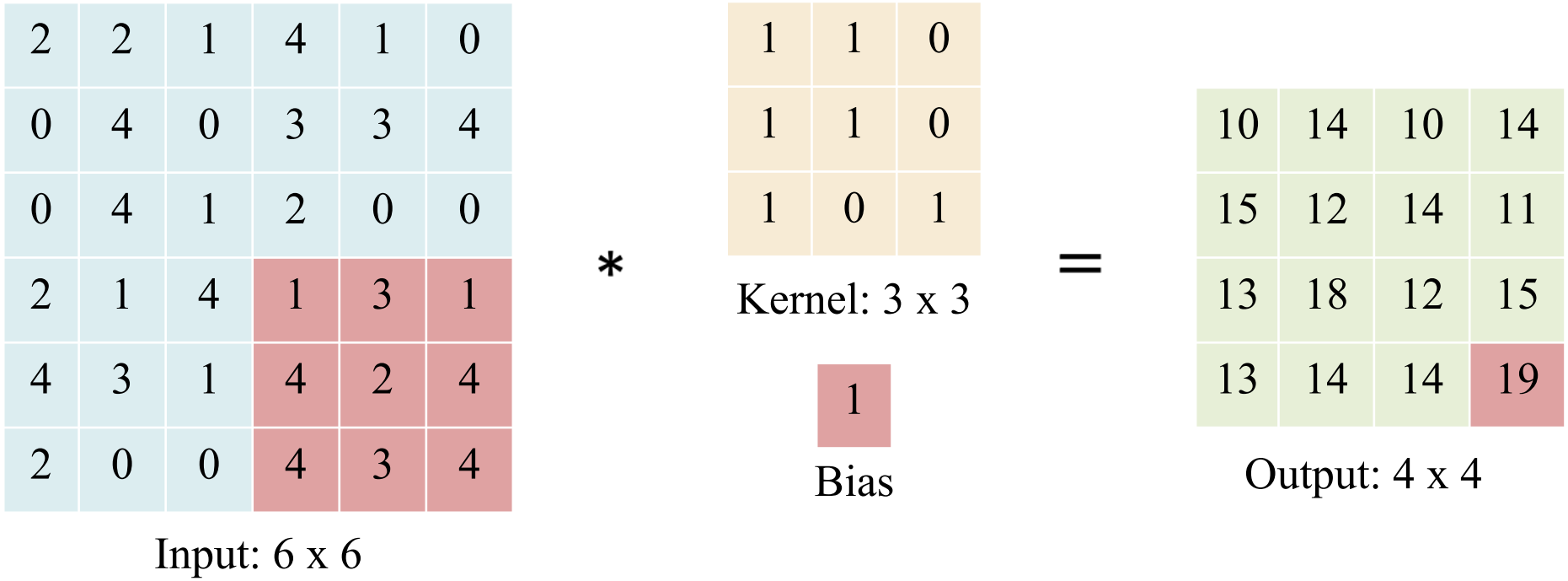
# 2 – Convolutional Layer

## Convolutional Operation



# 2 – Convolutional Layer

## Convolutional Operation



# 2 – Convolutional Layer

## Convolutional Operation

- **Pytorch**

```
input
```

```
tensor([[[2., 2., 1., 4., 1., 0.],
         [0., 4., 0., 3., 3., 4.],
         [0., 4., 1., 2., 0., 0.],
         [2., 1., 4., 1., 3., 1.],
         [4., 3., 1., 4., 2., 4.],
         [2., 0., 0., 4., 3., 4.]]])
```

```
# define convolutional layer
conv_layer = nn.Conv2d(
    in_channels=1,
    out_channels=1,
    kernel_size=3, # create a kernel: 3 x 3
)
```

```
init_kernel_weight
```

```
tensor([[[[1., 1., 0.],
          [1., 1., 0.],
          [1., 0., 1.]]]])
```

```
# init weight
conv_layer.weight.data = init_kernel_weight
```

```
conv_layer.weight
```

```
Parameter containing:
tensor([[[[1., 1., 0.],
          [1., 1., 0.],
          [1., 0., 1.]]]], requires_grad=True)
```

```
conv_layer.bias
```

```
Parameter containing:
tensor([-0.1148], requires_grad=True)
```

```
# init bias
conv_layer.bias =  nn.Parameter(
    torch.tensor([1], dtype=torch.float32)
)
```

```
conv_layer.bias
```

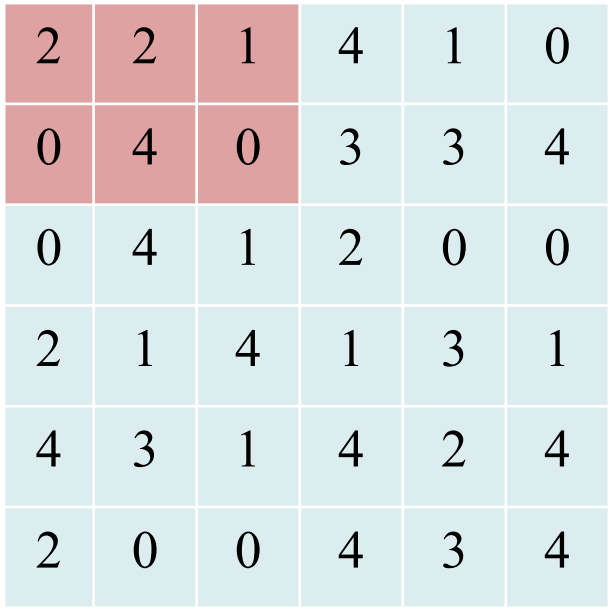
```
Parameter containing:
tensor([1.], requires_grad=True)
```

```
output = conv_layer(input)
output
```

```
tensor([[[10., 14., 10., 14.],
         [15., 12., 14., 11.],
         [13., 18., 12., 15.],
         [13., 14., 14., 19.]]], grad_fn=<SqueezeBackward1>)
```

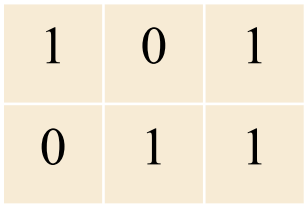
# 2 – Convolutional Layer

## Convolutional Operation




Input: 6 x 6

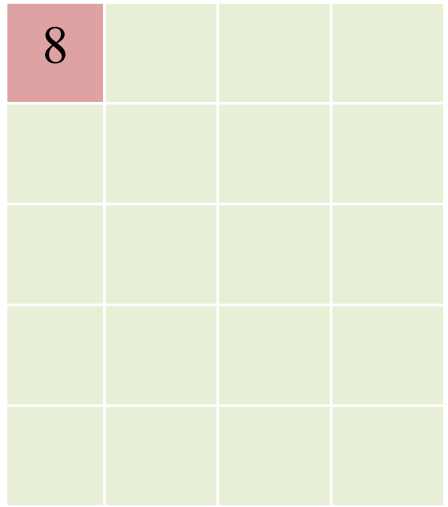
*




Kernel: 2 x 3


Bias

=




Output: 5 x 4

# 2 – Convolutional Layer

## Convolutional Operation

| | | | | | |
|---|---|---|---|---|---|
| 2 | 2 | 1 | 4 | 1 | 0 |
| 0 | 4 | 0 | 3 | 3 | 4 |
| 0 | 4 | 1 | 2 | 0 | 0 |
| 2 | 1 | 4 | 1 | 3 | 1 |
| 4 | 3 | 1 | 4 | 2 | 4 |
| 2 | 0 | 0 | 4 | 3 | 4 |

Input: 6 x 6

*

| | | |
|---|---|---|
| 1 | 0 | 1 |
| 0 | 1 | 1 |

Kernel: 2 x 3

| |
|---|
| 1 |

Bias

=

| | | | |
|---|---|---|---|
| 8 | 10 | 9 | 12 |
| 6 | 11 | 6 | 8 |
| 7 | 12 | 6 | 7 |
| 11 | 8 | 14 | 9 |
| 6 | 12 | 11 | 16 |

Output: 5 x 4

# 2 – Convolutional Layer

## Convolutional Operation

### Pytorch

```
input
```

```
tensor([[[2., 2., 1., 4., 1., 0.],
         [0., 4., 0., 3., 3., 4.],
         [0., 4., 1., 2., 0., 0.],
         [2., 1., 4., 1., 3., 1.],
         [4., 3., 1., 4., 2., 4.],
         [2., 0., 0., 4., 3., 4.]]])
```

```
init_kernel_weight = torch.randint(
    high=2,
    size=(conv_layer.weight.data.shape),
    dtype=torch.float32
)
init_kernel_weight
```

```
tensor([[[[1., 0., 1.],
          [0., 1., 1.]]]])
```

```
# define convolutional layer
conv_layer = nn.Conv2d(
    in_channels=1,
    out_channels=1,
    kernel_size=(2, 3), # create a kernel: 2 x 3
)
```

```
# init weight & bias
conv_layer.weight.data = init_kernel_weight
conv_layer.weight
```

```
Parameter containing:
tensor([[[[1., 0., 1.],
          [0., 1., 1.]]]], requires_grad=True)
```

```
conv_layer.bias
```

```
Parameter containing:
tensor([0.3672], requires_grad=True)
```

```
# init bias
conv_layer.bias =  nn.Parameter(
    torch.tensor([1], dtype=torch.float32)
)
conv_layer.bias
```

```
Parameter containing:
tensor([1.], requires_grad=True)
```

```
output = conv_layer(input)
output
```

```
tensor([[[ 8., 10.,  9., 12.],
         [ 6., 11.,  6.,  8.],
         [ 7., 12.,  6.,  7.],
         [11.,  8., 14.,  9.],
         [ 6., 12., 11., 16.]]], grad_fn=<SqueezeBackward1>)
```

# 2 – Convolutional Layer

## Convolutional Operation

| | | | | | |
|---|---|---|---|---|---|
| 2 | 2 | 1 | 4 | 1 | 0 |
| 0 | 4 | 0 | 3 | 3 | 4 |
| 0 | 4 | 1 | 2 | 0 | 0 |
| 2 | 1 | 4 | 1 | 3 | 1 |
| 4 | 3 | 1 | 4 | 2 | 4 |
| 2 | 0 | 0 | 4 | 3 | 4 |

Input: M x N

*

| | | |
|---|---|---|
| 1 | 0 | 1 |
| 0 | 1 | 1 |

Kernel: K x O

| |
|---|
| 1 |

Bias

=

| | | | |
|---|---|---|---|
| 8 | 10 | 9 | 12 |
| 6 | 11 | 6 | 8 |
| 7 | 12 | 6 | 7 |
| 11 | 8 | 14 | 9 |
| 6 | 12 | 11 | 16 |

Output:
M – (K – 1) x N – (O – 1)

# 2 – Convolutional Layer

## Padding

| | | | |
|---|---|---|---|
| 2 | 3 | 1 | 4 |
| 1 | 1 | 3 | 2 |
| 0 | 4 | 3 | 0 |
| 3 | 2 | 2 | 0 |

Input: 4 x 4

Padding: 1 x 1

| | | | | | |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 2 | 3 | 1 | 4 | 0 |
| 0 | 1 | 1 | 3 | 2 | 0 |
| 0 | 0 | 4 | 3 | 0 | 0 |
| 0 | 3 | 2 | 2 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 |

Shape: 6 x 6

*

| | | |
|---|---|---|
| 1 | 1 | 1 |
| 1 | 1 | 1 |
| 0 | 1 | 0 |

Kernel: 3 x 3

| |
|---|
| 1 |

Bias

=

| | | | |
|---|---|---|---|
| 7 | 8 | 12 | 8 |
| 8 | 16 | 18 | 11 |
| 10 | 15 | 16 | 9 |
| 10 | 15 | 12 | 6 |

Output: 4 x 4

# 2 – Convolutional Layer

## Padding

```
input = torch.randint(5, (1, 4, 4), dtype=torch.float32)
input
```

```
tensor([[[2., 3., 1., 4.],
         [1., 1., 3., 2.],
         [0., 4., 3., 0.],
         [3., 2., 2., 0.]]])
```

```
init_kernel_weight = torch.randint(
    high=2,
    size=(conv_layer.weight.data.shape),
    dtype=torch.float32
)
init_kernel_weight
```

```
tensor([[[[1., 1., 1.],
          [1., 1., 1.],
          [0., 1., 0.]]]])
```

```
# define convolutional layer
conv_layer = nn.Conv2d(
    in_channels=1,
    out_channels=1,
    kernel_size=3, # create a kernel: 3 x 3
    padding='same'
)
```

{"valid", "same"}

```
conv_layer.weight.data = init_kernel_weight
conv_layer.weight
```

```
Parameter containing:
tensor([[[[1., 1., 1.],
          [1., 1., 1.],
          [0., 1., 0.]]]], requires_grad=True)
```

```
# init bias
conv_layer.bias =  nn.Parameter(
    torch.tensor([1], dtype=torch.float32)
)
conv_layer.bias
```

```
Parameter containing:
tensor([1.], requires_grad=True)
```

```
output = conv_layer(input)
output
```

```
tensor([[[ 7.,  8., 12.,  8.],
         [ 8., 16., 18., 11.],
         [10., 15., 16.,  9.],
         [10., 15., 12.,  6.]]], grad_fn=<SqueezeBackward1>)
```

# 2 – Convolutional Layer

## Padding

| | | | |
|---|---|---|---|
| 2 | 3 | 1 | 4 |
| 1 | 1 | 3 | 2 |
| 0 | 4 | 3 | 0 |
| 3 | 2 | 2 | 0 |

Input: 4 x 4

Padding: 2 x 1

| | | | | | |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 2 | 3 | 1 | 4 | 0 |
| 0 | 1 | 1 | 3 | 2 | 0 |
| 0 | 0 | 4 | 3 | 0 | 0 |
| 0 | 3 | 2 | 2 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 |

Shape: 8 x 6

*

| | | |
|---|---|---|
| 1 | 1 | 1 |
| 1 | 1 | 1 |
| 0 | 1 | 0 |

Kernel: 3 x 3

1

Bias

=

| | | | |
|---|---|---|---|
| 3 | 4 | 2 | 5 |
| 7 | 8 | 12 | 8 |
| 8 | 16 | 18 | 11 |
| 10 | 15 | 16 | 9 |
| 10 | 15 | 12 | 6 |
| 6 | 8 | 5 | 3 |

Output: 6 x 4

# 2 – Convolutional Layer

## Padding

```
input
```

```
tensor([[[2., 3., 1., 4.],
         [1., 1., 3., 2.],
         [0., 4., 3., 0.],
         [3., 2., 2., 0.]]])
```

```
# define convolutional layer
conv_layer = nn.Conv2d(
    in_channels=1,
    out_channels=1,
    kernel_size=3, # create a kernel: 3 x 3
    padding=(2, 1)
)
```

An int / a tuple of ints

```
conv_layer.weight.data = init_kernel_weight
conv_layer.weight
```

```
Parameter containing:
tensor([[[[1., 1., 1.],
          [1., 1., 1.],
          [0., 1., 0.]]]], requires_grad=True)
```

```
# init bias
conv_layer.bias =  nn.Parameter(
    torch.tensor([1], dtype=torch.float32)
)
conv_layer.bias
```

```
Parameter containing:
tensor([1.], requires_grad=True)
```

```
output = conv_layer(input)
output
```

```
tensor([[[ 3.,  4.,  2.,  5.],
         [ 7.,  8., 12.,  8.],
         [ 8., 16., 18., 11.],
         [10., 15., 16.,  9.],
         [10., 15., 12.,  6.],
         [ 6.,  8.,  5.,  3.]]], grad_fn=<SqueezeBackward1>)
```

# 2 – Convolutional Layer

## Padding

| | | | |
|---|---|---|---|
| 2 | 3 | 1 | 4 |
| 1 | 1 | 3 | 2 |
| 0 | 4 | 3 | 0 |
| 3 | 2 | 2 | 0 |

Input: M x N

Padding: P x Q

| | | | | | |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 2 | 3 | 1 | 4 | 0 |
| 0 | 1 | 1 | 3 | 2 | 0 |
| 0 | 0 | 4 | 3 | 0 | 0 |
| 0 | 3 | 2 | 2 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 |

Shape: (M+2P) x (N+2Q)

*

| | | |
|---|---|---|
| 1 | 1 | 1 |
| 1 | 1 | 1 |
| 0 | 1 | 0 |

Kernel: K x O

1

Bias

=

| | | | |
|---|---|---|---|
| 3 | 4 | 2 | 5 |
| 7 | 8 | 12 | 8 |
| 8 | 16 | 18 | 11 |
| 10 | 15 | 16 | 9 |
| 10 | 15 | 12 | 6 |
| 6 | 8 | 5 | 3 |

Output:
(M+2P-K+1) x (N+2Q-O+1)

# 2 – Convolutional Layer

## Stride

Stride: 1 (1x1)

| | | | | | |
|---|---|---|---|---|---|
| 1 | 0 | 1 | 3 | 1 | 3 |
| 0 | 1 | 4 | 0 | 0 | 4 |
| 0 | 2 | 0 | 3 | 3 | 2 |
| 2 | 2 | 1 | 3 | 2 | 2 |
| 1 | 3 | 0 | 3 | 1 | 0 |
| 3 | 2 | 3 | 3 | 4 | 3 |

Input: 6 x 6

*

| | | |
|---|---|---|
| 1 | 1 | 1 |
| 1 | 1 | 1 |
| 0 | 1 | 0 |

Kernel: 3 x 3

| |
|---|
| 1 |

Bias

=

| | | | |
|---|---|---|---|
| 10 | 10 | 13 | 15 |
| 10 | 12 | 14 | 15 |
| 11 | 12 | 16 | 17 |
| 12 | 16 | 14 | 16 |

Output: 4 x 4

# 2 – Convolutional Layer

## Stride

Stride: 2 (2x2)

| | | | | | |
|---|---|---|---|---|---|
| 1 | 0 | 1 | 3 | 1 | 3 |
| 0 | 1 | 4 | 0 | 0 | 4 |
| 0 | 2 | 0 | 3 | 3 | 2 |
| 2 | 2 | 1 | 3 | 2 | 2 |
| 1 | 3 | 0 | 3 | 1 | 0 |
| 3 | 2 | 3 | 3 | 4 | 3 |

Input: 6 x 6

*

| | | |
|---|---|---|
| 1 | 1 | 1 |
| 1 | 1 | 1 |
| 0 | 1 | 0 |

Kernel: 3 x 3

| |
|---|
| 1 |

Bias

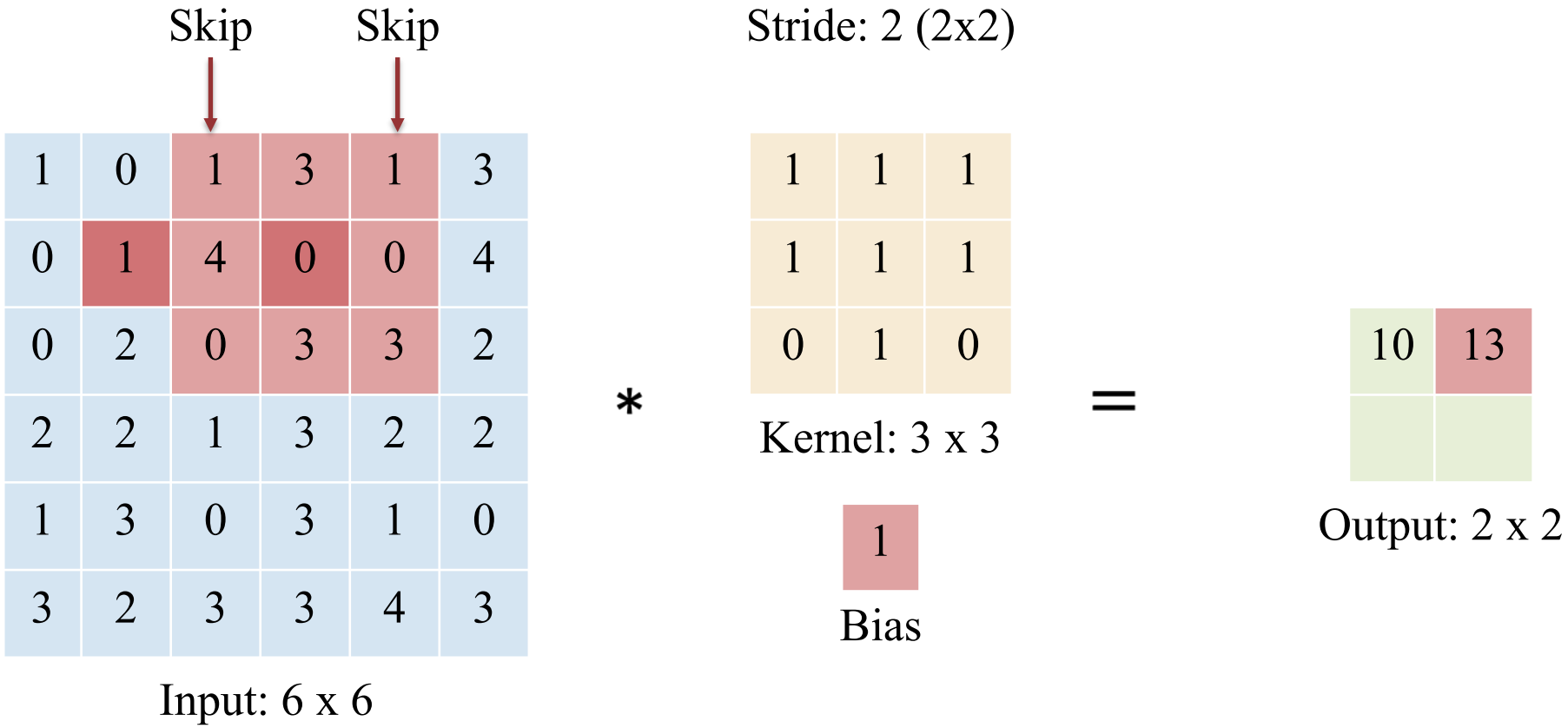
=

| | |
|---|---|
| 10 | |
| | |

Output: 2 x 2

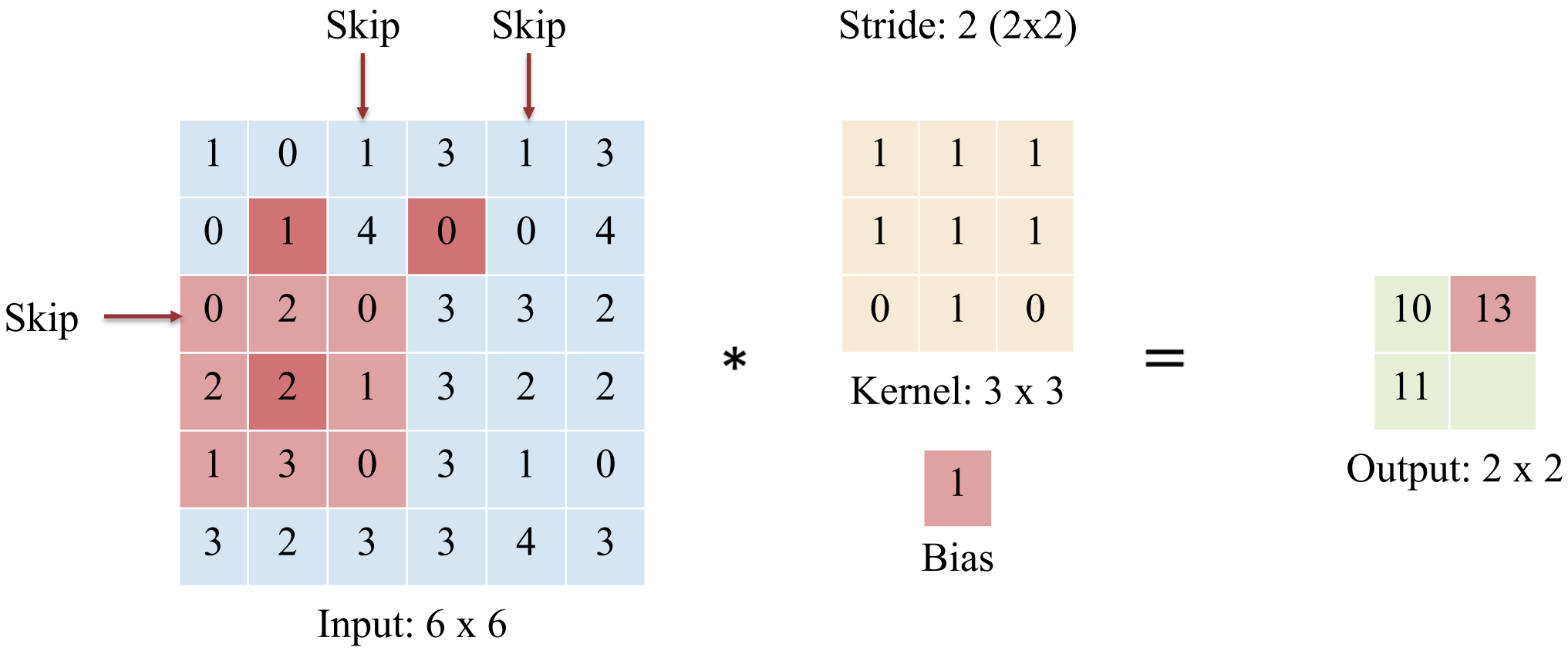
# 2 – Convolutional Layer

## Stride



# 2 – Convolutional Layer

## Stride



# 2 – Convolutional Layer

## Stride

Stride: 2 (2x2)

| | | | | | |
|---|---|---|---|---|---|
| 1 | 0 | 1 | 3 | 1 | 3 |
| 0 | 1 | 4 | 0 | 0 | 4 |
| 0 | 2 | 0 | 3 | 3 | 2 |
| 2 | 2 | 1 | 3 | 2 | 2 |
| 1 | 3 | 0 | 3 | 1 | 0 |
| 3 | 2 | 3 | 3 | 4 | 3 |

Input: 6 x 6

*

| | | |
|---|---|---|
| 1 | 1 | 1 |
| 1 | 1 | 1 |
| 0 | 1 | 0 |

Kernel: 3 x 3

| |
|---|
| 1 |

Bias

=

| | |
|---|---|
| 10 | 13 |
| 11 | 16 |

Output: 2 x 2

# 2 – Convolutional Layer

## Stride

```
input = torch.randint(5, (1, 6, 6), dtype=torch.float32)
input
```

```
tensor([[[1., 0., 1., 3., 1., 3.],
         [0., 1., 4., 0., 0., 4.],
         [0., 2., 0., 3., 3., 2.],
         [2., 2., 1., 3., 2., 2.],
         [1., 3., 0., 3., 1., 0.],
         [3., 2., 3., 3., 4., 3.]]])
```

```
# define convolutional layer
conv_layer = nn.Conv2d(
    in_channels=1,
    out_channels=1,
    kernel_size=3, # create a kernel: 3 x 3
    stride=2
)
```

```
conv_layer.weight.data = init_kernel_weight
conv_layer.weight
```

```
Parameter containing:
tensor([[[[1., 1., 1.],
          [1., 1., 1.],
          [0., 1., 0.]]]], requires_grad=True)
```

```
# init bias
conv_layer.bias =  nn.Parameter(
    torch.tensor([1], dtype=torch.float32)
)
conv_layer.bias
```

```
Parameter containing:
tensor([1.], requires_grad=True)
```

```
output = conv_layer(input)
output
```

```
tensor([[[10., 13.],
         [11., 16.]]], grad_fn=<SqueezeBackward1>)
```

# 2 – Convolutional Layer

## Stride

Stride: 2 (2x2)

| | | | |
|---|---|---|---|
| 0 | 3 | 1 | 1 |
| 3 | 1 | 2 | 0 |
| 3 | 4 | 2 | 3 |
| 3 | 0 | 0 | 2 |

Input: 4 x 4

Padding: 1 x 1

| | | | | | |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 3 | 1 | 1 | 0 |
| 0 | 3 | 1 | 2 | 0 | 0 |
| 0 | 3 | 4 | 2 | 3 | 0 |
| 0 | 3 | 0 | 0 | 2 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 |

Shape: 6 x 6

*

| | | |
|---|---|---|
| 1 | 1 | 1 |
| 1 | 1 | 1 |
| 0 | 1 | 0 |

Kernel: 3 x 3

| |
|---|
| 1 |

Bias

=

| | |
|---|---|
| 7 | 8 |
| 15 | 13 |

Output: 2 x 2

# 2 – Convolutional Layer

## Stride

```
input = torch.randint(5, (1, 4, 4), dtype=torch.float32)
input
```

```
tensor([[[0., 3., 1., 1.],
         [3., 1., 2., 0.],
         [3., 4., 2., 3.],
         [3., 0., 0., 2.]]])
```

```
# define convolutional layer
conv_layer = nn.Conv2d(
    in_channels=1,
    out_channels=1,
    kernel_size=3, # create a kernel: 3 x 3
    padding=1,
    stride=(2, 2)
)
```

```
conv_layer.weight.data = init_kernel_weight
conv_layer.weight
```

```
Parameter containing:
tensor([[[[1., 1., 1.],
          [1., 1., 1.],
          [0., 1., 0.]]]], requires_grad=True)
```

```
# init bias
conv_layer.bias =  nn.Parameter(
    torch.tensor([1], dtype=torch.float32)
)
conv_layer.bias
```

```
Parameter containing:
tensor([1.], requires_grad=True)
```

```
output = conv_layer(input)
output
```

```
tensor([[[ 7.,  8.],
         [15., 13.]]], grad_fn=<SqueezeBackward1>)
```

# 2 – Convolutional Layer

## Stride

Stride: (S, T)

| | | | |
|---|---|---|---|
| 0 | 3 | 1 | 1 |
| 3 | 1 | 2 | 0 |
| 3 | 4 | 2 | 3 |
| 3 | 0 | 0 | 2 |

Input: M x N

Padding: (P, Q)

| | | | | | |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 3 | 1 | 1 | 0 |
| 0 | 3 | 1 | 2 | 0 | 0 |
| 0 | 3 | 4 | 2 | 3 | 0 |
| 0 | 3 | 0 | 0 | 2 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 |

Shape: (M+2P) x (N+2Q)

*

| | | |
|---|---|---|
| 1 | 1 | 1 |
| 1 | 1 | 1 |
| 0 | 1 | 0 |

Kernel: K x O

| |
|---|
| 1 |

Bias

=

| | |
|---|---|
| 7 | 8 |
| 15 | 13 |

$$\left\lfloor \frac{M + 2P - K}{S} + 1 \right\rfloor x \left\lfloor \frac{N + 2Q - K}{T} + 1 \right\rfloor$$

# 3 – Pooling Layer

## Max Pooling

Kernel Size: 2
Stride: 2

| | | | | | |
|---|---|---|---|---|---|
| 3 | 2 | 1 | 0 | 0 | 3 |
| 0 | 3 | 3 | 1 | 1 | 0 |
| 3 | 1 | 4 | 1 | 1 | 0 |
| 2 | 4 | 1 | 1 | 0 | 4 |
| 1 | 0 | 3 | 0 | 3 | 0 |
| 3 | 4 | 4 | 3 | 3 | 4 |

Input: 6 x 6

| | |
|---|---|
| 3 | 2 |
| 0 | 3 |

Max values

=

| | | |
|---|---|---|
| 3 | | |
| | | |
| | | |

Output: 3 x 3

# 3 – Pooling Layer

## Max Pooling

Kernel Size: 2
Stride: 2

| | | | | | |
|---|---|---|---|---|---|
| 3 | 2 | 1 | 0 | 0 | 3 |
| 0 | 3 | 3 | 1 | 1 | 0 |
| 3 | 1 | 4 | 1 | 1 | 0 |
| 2 | 4 | 1 | 1 | 0 | 4 |
| 1 | 0 | 3 | 0 | 3 | 0 |
| 3 | 4 | 4 | 3 | 3 | 4 |

Input: 6 x 6

| | |
|---|---|
| 1 | 0 |
| 3 | 1 |

Max values

=

| | | |
|---|---|---|
| 3 | 3 | |
| | | |
| | | |

Output: 3 x 3

# 3 – Pooling Layer

## Max Pooling

Kernel Size: 2
Stride: 2

| | | | | | |
|---|---|---|---|---|---|
| 3 | 2 | 1 | 0 | 0 | 3 |
| 0 | 3 | 3 | 1 | 1 | 0 |
| 3 | 1 | 4 | 1 | 1 | 0 |
| 2 | 4 | 1 | 1 | 0 | 4 |
| 1 | 0 | 3 | 0 | 3 | 0 |
| 3 | 4 | 4 | 3 | 3 | 4 |

Input: 6 x 6

| | |
|---|---|
| 1 | 0 |
| 3 | 1 |

Max values

=

| | | |
|---|---|---|
| 3 | 3 | 3 |
| 4 | 4 | 4 |
| 4 | 4 | 4 |

Output: 3 x 3

# 3 – Pooling Layer

## Max Pooling

| | | | | | |
|---|---|---|---|---|---|
| 3 | 2 | 1 | 0 | 0 | 3 |
| 0 | 3 | 3 | 1 | 1 | 0 |
| 3 | 1 | 4 | 1 | 1 | 0 |
| 2 | 4 | 1 | 1 | 0 | 4 |
| 1 | 0 | 3 | 0 | 3 | 0 |
| 3 | 4 | 4 | 3 | 3 | 4 |

Input: 6 x 6

Kernel Size: 2
Stride: 2

| | | |
|---|---|---|
| 3 | 3 | 3 |
| 4 | 4 | 4 |
| 4 | 4 | 4 |

Output: 3 x 3

```
input = torch.randint(5, (1, 6, 6), dtype=torch.float32)
input
```

```
tensor([[[3., 2., 1., 0., 0., 3.],
         [0., 3., 3., 1., 1., 0.],
         [3., 1., 4., 1., 1., 0.],
         [2., 4., 1., 1., 0., 4.],
         [1., 0., 3., 0., 3., 0.],
         [3., 4., 4., 3., 3., 4.]]])
```

```
max_pool_layer = nn.MaxPool2d(kernel_size=2)
```

Default: Stride = 2

```
output = max_pool_layer(input)
output
```

```
tensor([[[3., 3., 3.],
         [4., 4., 4.],
         [4., 4., 4.]]])
```

# 3 – Pooling Layer

## Max Pooling

| | | | | | |
|---|---|---|---|---|---|
| 3 | 2 | 1 | 0 | 0 | 3 |
| 0 | 3 | 3 | 1 | 1 | 0 |
| 3 | 1 | 4 | 1 | 1 | 0 |
| 2 | 4 | 1 | 1 | 0 | 4 |
| 1 | 0 | 3 | 0 | 3 | 0 |
| 3 | 4 | 4 | 3 | 3 | 4 |

Input: 6 x 6

Kernel Size: 2
Stride: (1, 2)

| | | |
|---|---|---|
| 3 | 3 | 4 |
| 3 | 4 | 1 |
| 4 | 4 | 4 |
| 4 | 3 | 4 |
| 4 | 4 | 4 |

Output: 5 x 3

```
input
```

```
tensor([[[3., 2., 1., 0., 0., 3.],
         [0., 3., 3., 1., 1., 0.],
         [3., 1., 4., 1., 1., 0.],
         [2., 4., 1., 1., 0., 4.],
         [1., 0., 3., 0., 3., 0.],
         [3., 4., 4., 3., 3., 4.]]])
```

```
max_pool_layer = nn.MaxPool2d(
    kernel_size=2,
    stride=(1, 2)
)
```

```
output = max_pool_layer(input)
output
```

```
tensor([[[3., 3., 3.],
         [3., 4., 1.],
         [4., 4., 4.],
         [4., 3., 4.],
         [4., 4., 4.]]])
```

# 3 – Pooling Layer

## Max Pooling

**MaxPool1d**
Kernel Size: 3
Stride: 3

| | | | | | |
|---|---|---|---|---|---|
| 3 | 2 | 1 | 0 | 0 | 3 |
| 0 | 3 | 3 | 1 | 1 | 0 |
| 3 | 1 | 4 | 1 | 1 | 0 |
| 2 | 4 | 1 | 1 | 0 | 4 |
| 1 | 0 | 3 | 0 | 3 | 0 |
| 3 | 4 | 4 | 3 | 3 | 4 |

Input: 6 x 6

| | |
|---|---|
| 3 | 3 |
| 3 | 1 |
| 4 | 1 |
| 4 | 4 |
| 3 | 3 |
| 4 | 4 |

Output: 6 x 2

```
input
```

```
tensor([[[3., 2., 1., 0., 0., 3.],
         [0., 3., 3., 1., 1., 0.],
         [3., 1., 4., 1., 1., 0.],
         [2., 4., 1., 1., 0., 4.],
         [1., 0., 3., 0., 3., 0.],
         [3., 4., 4., 3., 3., 4.]]])
```

```
max_pool_layer = nn.MaxPool1d(
    kernel_size=3,
    stride=3
)
```

```
max_pool_layer(input)
```

```
tensor([[[3., 3.],
         [3., 1.],
         [4., 1.],
         [4., 4.],
         [3., 3.],
         [4., 4.]]])
```

# 3 – Pooling Layer

## Average Pooling

Kernel Size: (3, 2)
Stride: 2

| | | | | | |
|---|---|---|---|---|---|
| 3 | 2 | 1 | 0 | 0 | 3 |
| 0 | 3 | 3 | 1 | 1 | 0 |
| 3 | 1 | 4 | 1 | 1 | 0 |
| 2 | 4 | 1 | 1 | 0 | 4 |
| 1 | 0 | 3 | 0 | 3 | 0 |
| 3 | 4 | 4 | 3 | 3 | 4 |

Input: 6 x 6

| | |
|---|---|
| 3 | 2 |
| 0 | 3 |
| 3 | 1 |

Average values

=

| | | |
|---|---|---|
| 2.0 | | |
| | | |

Output: 2 x 3

# 3 – Pooling Layer

## Average Pooling

Kernel Size: (3, 2)
Stride: 2

| | | | | | |
|---|---|---|---|---|---|
| 3 | 2 | 1 | 0 | 0 | 3 |
| 0 | 3 | 3 | 1 | 1 | 0 |
| 3 | 1 | 4 | 1 | 1 | 0 |
| 2 | 4 | 1 | 1 | 0 | 4 |
| 1 | 0 | 3 | 0 | 3 | 0 |
| 3 | 4 | 4 | 3 | 3 | 4 |

Input: 6 x 6

| | |
|---|---|
| 1 | 0 |
| 3 | 1 |
| 4 | 1 |

Average values

=

| | | |
|---|---|---|
| 2 | 1.7 | |
| | | |

Output: 2 x 3

# 3 – Pooling Layer

## Average Pooling

Kernel Size: (3, 2)
Stride: 2

| | | | | | |
|---|---|---|---|---|---|
| 3 | 2 | 1 | 0 | 0 | 3 |
| 0 | 3 | 3 | 1 | 1 | 0 |
| 3 | 1 | 4 | 1 | 1 | 0 |
| 2 | 4 | 1 | 1 | 0 | 4 |
| 1 | 0 | 3 | 0 | 3 | 0 |
| 3 | 4 | 4 | 3 | 3 | 4 |

Input: 6 x 6

| | |
|---|---|
| 3 | 1 |
| 2 | 4 |
| 1 | 0 |

Average values

=

| | | |
|---|---|---|
| 2 | 1.7 | 0.8 |
| 1.8 | | |

Output: 2 x 3

# 3 – Pooling Layer

## Average Pooling

| | | | | | |
|---|---|---|---|---|---|
| 3 | 2 | 1 | 0 | 0 | 3 |
| 0 | 3 | 3 | 1 | 1 | 0 |
| 3 | 1 | 4 | 1 | 1 | 0 |
| 2 | 4 | 1 | 1 | 0 | 4 |
| 1 | 0 | 3 | 0 | 3 | 0 |
| 3 | 4 | 4 | 3 | 3 | 4 |

Input: 6 x 6

Kernel Size: (3, 2)
Stride: 2

| | | |
|---|---|---|
| 2 | 1.7 | 0.8 |
| 1.8 | 1.6 | 1.3 |

Output: 2 x 3

```
input
```

```
tensor([[[3., 2., 1., 0., 0., 3.],
         [0., 3., 3., 1., 1., 0.],
         [3., 1., 4., 1., 1., 0.],
         [2., 4., 1., 1., 0., 4.],
         [1., 0., 3., 0., 3., 0.],
         [3., 4., 4., 3., 3., 4.]]])
```

```
avg_pool_layer = nn.AvgPool2d(
    kernel_size=(3, 2),
    stride=(2, 2)
)
```

```
output = avg_pool_layer(input)
output
```

```
tensor([[[2.0000, 1.6667, 0.8333],
         [1.8333, 1.6667, 1.3333]]])
```

# 3 – Pooling Layer

## Average Pooling

**AvgPool1d**
Kernel Size: 3
Stride: 3

| | | | | | |
|---|---|---|---|---|---|
| 3 | 2 | 1 | 0 | 0 | 3 |
| 0 | 3 | 3 | 1 | 1 | 0 |
| 3 | 1 | 4 | 1 | 1 | 0 |
| 2 | 4 | 1 | 1 | 0 | 4 |
| 1 | 0 | 3 | 0 | 3 | 0 |
| 3 | 4 | 4 | 3 | 3 | 4 |

Input: 6 x 6

| | |
|---|---|
| 2.0 | 1.0 |
| 2.0 | 0.7 |
| 2.7 | 0.7 |
| 2.3 | 1.7 |
| 1.3 | 1.0 |
| 3.7 | 3.3 |

Output: 6 x 2

```
input
```

```
tensor([[[3., 2., 1., 0., 0., 3.],
         [0., 3., 3., 1., 1., 0.],
         [3., 1., 4., 1., 1., 0.],
         [2., 4., 1., 1., 0., 4.],
         [1., 0., 3., 0., 3., 0.],
         [3., 4., 4., 3., 3., 4.]]])
```

```
avg_pool_layer = nn.AvgPool1d(
    kernel_size=3,
    stride=3
)
```

```
output = avg_pool_layer(input)
output
```

```
tensor([[[2.0000, 1.0000],
         [2.0000, 0.6667],
         [2.6667, 0.6667],
         [2.3333, 1.6667],
         [1.3333, 1.0000],
         [3.6667, 3.3333]]])
```

# 4 - Flatten

**Flattens a contiguous range of dims into a tensor**

| | |
|---|---|
| 2 | 4 |
| 3 | 1 |
| 3 | 4 |

Input: 3 x 2

| | | | | | |
|---|---|---|---|---|---|
| 2 | 4 | 3 | 1 | 3 | 4 |

Output: 1 x 6

```
input = torch.randint(5, (1, 3, 2), dtype=torch.float32)
input
```

```
tensor([[[2., 4.],
         [3., 1.],
         [3., 4.]]])
```

```
flatten_layer = nn.Flatten()
```

```
output = flatten_layer(input)
output
```

```
tensor([[2., 4., 3., 1., 3., 4.]])
```

# 5 - Practice

## 1 Exercise – Convolutional Layer

| | | | | | |
|---|---|---|---|---|---|
| 2 | 2 | 1 | 4 | 1 | 0 |
| 0 | 4 | 0 | 3 | 3 | 4 |
| 0 | 4 | 1 | 2 | 0 | 0 |
| 2 | 1 | 4 | 1 | 3 | 1 |

Input: 4 x 6

$*$

| | | |
|---|---|---|
| 1 | 1 | 0 |
| 1 | 0 | 0 |
| 0 | 0 | 0 |

Kernel: 3 x 3

$=$

# 5 - Practice

## 1 Exercise – Convolutional Layer

| | | | | | |
|---|---|---|---|---|---|
| 2 | 2 | 1 | 4 | 1 | 0 |
| 0 | 4 | 0 | 3 | 3 | 4 |
| 0 | 4 | 1 | 2 | 0 | 0 |
| 2 | 1 | 4 | 1 | 3 | 1 |

Input: 4 x 6

*

| | | |
|---|---|---|
| 1 | 1 | 0 |
| 1 | 0 | 0 |
| 0 | 0 | 0 |

Kernel: 3 x 3

=

| | | | |
|---|---|---|---|
| 4 | | | |
| | | | |

# 5 - Practice

## 1 Exercise – Convolutional Layer

| | | | | | |
|---|---|---|---|---|---|
| 2 | 2 | 1 | 4 | 1 | 0 |
| 0 | 4 | 0 | 3 | 3 | 4 |
| 0 | 4 | 1 | 2 | 0 | 0 |
| 2 | 1 | 4 | 1 | 3 | 1 |

Input: 4 x 6

*

| | | |
|---|---|---|
| 1 | 1 | 0 |
| 1 | 0 | 0 |
| 0 | 0 | 0 |

Kernel: 3 x 3

=

| | | | |
|---|---|---|---|
| 4 | 7 | | |
| | | | |

# 5 - Practice

## 1 Exercise – Convolutional Layer

| | | | | | |
|---|---|---|---|---|---|
| 2 | 2 | 1 | 4 | 1 | 0 |
| 0 | 4 | 0 | 3 | 3 | 4 |
| 0 | 4 | 1 | 2 | 0 | 0 |
| 2 | 1 | 4 | 1 | 3 | 1 |

Input: 4 x 6

*

| | | |
|---|---|---|
| 1 | 1 | 0 |
| 1 | 0 | 0 |
| 0 | 0 | 0 |

Kernel: 3 x 3

=

| | | | |
|---|---|---|---|
| 4 | 7 | 5 | |
| | | | |

# 5 - Practice

## 1 Exercise – Convolutional Layer

| | | | | | |
|---|---|---|---|---|---|
| 2 | 2 | 1 | 4 | 1 | 0 |
| 0 | 4 | 0 | 3 | 3 | 4 |
| 0 | 4 | 1 | 2 | 0 | 0 |
| 2 | 1 | 4 | 1 | 3 | 1 |

Input: 4 x 6

*

| | | |
|---|---|---|
| 1 | 1 | 0 |
| 1 | 0 | 0 |
| 0 | 0 | 0 |

Kernel: 3 x 3

=

| | | | |
|---|---|---|---|
| 4 | 7 | 5 | 8 |
| | | | |

# 5 - Practice

## 1 Exercise – Convolutional Layer

| | | | | | |
|---|---|---|---|---|---|
| 2 | 2 | 1 | 4 | 1 | 0 |
| 0 | 4 | 0 | 3 | 3 | 4 |
| 0 | 4 | 1 | 2 | 0 | 0 |
| 2 | 1 | 4 | 1 | 3 | 1 |

Input: 4 x 6

*

| | | |
|---|---|---|
| 1 | 1 | 0 |
| 1 | 0 | 0 |
| 0 | 0 | 0 |

Kernel: 3 x 3

=

| | | | |
|---|---|---|---|
| 4 | 7 | 5 | 8 |
| 4 | 8 | 4 | 8 |

# 5 - Practice

## 1 Exercise – Convolutional Layer

| | | | | | |
|---|---|---|---|---|---|
| 2 | 2 | 1 | 4 | 1 | 0 |
| 0 | 4 | 0 | 3 | 3 | 4 |
| 0 | 4 | 1 | 2 | 0 | 0 |
| 2 | 1 | 4 | 1 | 3 | 1 |

Input: 4 x 6

$*$

| | | |
|---|---|---|
| 1 | 1 | 0 |
| 1 | 0 | 0 |
| 0 | 0 | 0 |

Kernel: 3 x 3

| |
|---|
| 2 |

Bias

$=$

# 5 - Practice

## 1 Exercise – Convolutional Layer

Stride: 1 (1x1)

| | | | | | |
|---|---|---|---|---|---|
| 2 | 2 | 1 | 4 | 1 | 0 |
| 0 | 4 | 0 | 3 | 3 | 4 |
| 0 | 4 | 1 | 2 | 0 | 0 |
| 2 | 1 | 4 | 1 | 3 | 1 |

Input: 4 x 6

$*$

| | | |
|---|---|---|
| 1 | 1 | 0 |
| 1 | 0 | 0 |
| 0 | 0 | 0 |

Kernel: 3 x 3

| |
|---|
| 2 |

Bias

$=$

| | | | |
|---|---|---|---|
| 6 | 9 | 7 | 10 |
| 6 | 10 | 6 | 10 |

# 5 - Practice

## 2 Exercise – Padding

| | | |
|---|---|---|
| 2 | 4 | 2 |
| 3 | 3 | 4 |
| 3 | 2 | 0 |
| 4 | 0 | 4 |
| 1 | 4 | 0 |

Input: 4 x 6

Padding: 1 x 1

| | | | | |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 2 | 4 | 2 | 0 |
| 0 | 3 | 3 | 4 | 0 |
| 0 | 3 | 2 | 0 | 0 |
| 0 | 4 | 0 | 4 | 0 |
| 0 | 1 | 4 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 |

Input: 6 x 8

Stride: 1 (1x1)

*

| | | |
|---|---|---|
| 1 | 0 | 1 |
| 1 | 1 | 1 |
| 0 | 1 | 0 |

Kernel: 3 x 3

2

Bias

=

# 5 - Practice

## 2 Exercise – Padding

| | | |
|---|---|---|
| 2 | 4 | 2 |
| 3 | 3 | 4 |
| 3 | 2 | 0 |
| 4 | 0 | 4 |
| 1 | 4 | 0 |

Input: 5 x 3

Padding: 1 x 1

| | | | | |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 2 | 4 | 2 | 0 |
| 0 | 3 | 3 | 4 | 0 |
| 0 | 3 | 2 | 0 | 0 |
| 0 | 4 | 0 | 4 | 0 |
| 0 | 1 | 4 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 |

Input: 7 x 5

*

Stride: 1 (1x1)

| | | |
|---|---|---|
| 1 | 0 | 1 |
| 1 | 1 | 1 |
| 0 | 1 | 0 |

Kernel: 3 x 3

| |
|---|
| 2 |

Bias

=

| | | |
|---|---|---|
| 11 | 13 | 12 |
| | | |
| | | |
| | | |

# 5 - Practice

## 2 Exercise – Padding

| | | |
|---|---|---|
| 2 | 4 | 2 |
| 3 | 3 | 4 |
| 3 | 2 | 0 |
| 4 | 0 | 4 |
| 1 | 4 | 0 |

Input: 5 x 3

Padding: 1 x 1

| | | | | |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 2 | 4 | 2 | 0 |
| 0 | 3 | 3 | 4 | 0 |
| 0 | 3 | 2 | 0 | 0 |
| 0 | 4 | 0 | 4 | 0 |
| 0 | 1 | 4 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 |

Input: 7 x 5

*

Stride: 1 (1x1)

| | | |
|---|---|---|
| 1 | 0 | 1 |
| 1 | 1 | 1 |
| 0 | 1 | 0 |

Kernel: 3 x 3

| |
|---|
| 2 |

Bias

=

| | | |
|---|---|---|
| 11 | 13 | 12 |
| 15 | 18 | 13 |
| | | |
| | | |
| | | |

# 5 - Practice

## 2 Exercise – Padding

| | | |
|---|---|---|
| 2 | 4 | 2 |
| 3 | 3 | 4 |
| 3 | 2 | 0 |
| 4 | 0 | 4 |
| 1 | 4 | 0 |

Input: 5 x 3

Padding: 1 x 1

| | | | | |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 2 | 4 | 2 | 0 |
| 0 | 3 | 3 | 4 | 0 |
| 0 | 3 | 2 | 0 | 0 |
| 0 | 4 | 0 | 4 | 0 |
| 0 | 1 | 4 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 |

Input: 7 x 5

*

Stride: 1 (1x1)

| | | |
|---|---|---|
| 1 | 0 | 1 |
| 1 | 1 | 1 |
| 0 | 1 | 0 |

Kernel: 3 x 3

| |
|---|
| 2 |

Bias

=

| | | |
|---|---|---|
| 11 | 13 | 12 |
| 15 | 18 | 13 |
| 14 | 14 | 11 |
| | | |
| | | |

# 5 - Practice

## 2 Exercise – Padding

| | | |
|---|---|---|
| 2 | 4 | 2 |
| 3 | 3 | 4 |
| 3 | 2 | 0 |
| 4 | 0 | 4 |
| 1 | 4 | 0 |

Input: 5 x 3

Padding: 1 x 1

| | | | | |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 2 | 4 | 2 | 0 |
| 0 | 3 | 3 | 4 | 0 |
| 0 | 3 | 2 | 0 | 0 |
| 0 | 4 | 0 | 4 | 0 |
| 0 | 1 | 4 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 |

Input: 7 x 5

*

Stride: 1 (1x1)

| | | |
|---|---|---|
| 1 | 0 | 1 |
| 1 | 1 | 1 |
| 0 | 1 | 0 |

Kernel: 3 x 3

| |
|---|
| 2 |

Bias

=

| | | |
|---|---|---|
| 11 | 13 | 12 |
| 15 | 18 | 13 |
| 14 | 14 | 11 |
| 9 | 17 | 8 |
| | | |

# 5 - Practice

## 2 Exercise – Padding

| | | |
|---|---|---|
| 2 | 4 | 2 |
| 3 | 3 | 4 |
| 3 | 2 | 0 |
| 4 | 0 | 4 |
| 1 | 4 | 0 |

Input: 5 x 3

Padding: 1 x 1

| | | | | |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 2 | 4 | 2 | 0 |
| 0 | 3 | 3 | 4 | 0 |
| 0 | 3 | 2 | 0 | 0 |
| 0 | 4 | 0 | 4 | 0 |
| 0 | 1 | 4 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 |

Input: 7 x 5

*

Stride: 1 (1x1)

| | | |
|---|---|---|
| 1 | 0 | 1 |
| 1 | 1 | 1 |
| 0 | 1 | 0 |

Kernel: 3 x 3

| |
|---|
| 2 |

Bias

=

| | | |
|---|---|---|
| 11 | 13 | 12 |
| 15 | 18 | 13 |
| 14 | 14 | 11 |
| 9 | 17 | 8 |
| 7 | 15 | 6 |

# 5 - Practice

## 2 Exercise – Padding

| | | |
|---|---|---|
| 2 | 4 | 2 |
| 3 | 3 | 4 |
| 3 | 2 | 0 |
| 4 | 0 | 4 |
| 1 | 4 | 0 |

Input: 5 x 3

Padding: 1 x 1

| | | | | |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 2 | 4 | 2 | 0 |
| 0 | 3 | 3 | 4 | 0 |
| 0 | 3 | 2 | 0 | 0 |
| 0 | 4 | 0 | 4 | 0 |
| 0 | 1 | 4 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 |

Input: 7 x 5

Stride: 2 (2x2)

*

| | | |
|---|---|---|
| 1 | 0 | 1 |
| 1 | 1 | 1 |
| 0 | 1 | 0 |

Kernel: 3 x 3

=



| |
|---|
| 2 |

Bias

# 5 - Practice

## 2 Exercise – Padding

| | | |
|---|---|---|
| 2 | 4 | 2 |
| 3 | 3 | 4 |
| 3 | 2 | 0 |
| 4 | 0 | 4 |
| 1 | 4 | 0 |

Input: 5 x 3

Padding: 1 x 1

| | | | | |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 2 | 4 | 2 | 0 |
| 0 | 3 | 3 | 4 | 0 |
| 0 | 3 | 2 | 0 | 0 |
| 0 | 4 | 0 | 4 | 0 |
| 0 | 1 | 4 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 |

Input: 7 x 5

Stride: 2 (2x2)

*

| | | |
|---|---|---|
| 1 | 0 | 1 |
| 1 | 1 | 1 |
| 0 | 1 | 0 |

Kernel: 3 x 3

2

Bias

=

| | |
|---|---|
| 11 | 12 |
| | |
| | |

# 5 - Practice

## 2 Exercise – Padding

| | | |
|---|---|---|
| 2 | 4 | 2 |
| 3 | 3 | 4 |
| 3 | 2 | 0 |
| 4 | 0 | 4 |
| 1 | 4 | 0 |

Input: 5 x 3

Padding: 1 x 1

| | | | | |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 2 | 4 | 2 | 0 |
| 0 | 3 | 3 | 4 | 0 |
| 0 | 3 | 2 | 0 | 0 |
| 0 | 4 | 0 | 4 | 0 |
| 0 | 1 | 4 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 |

Input: 7 x 5

Stride: 2 (2x2)

*

| | | |
|---|---|---|
| 1 | 0 | 1 |
| 1 | 1 | 1 |
| 0 | 1 | 0 |

Kernel: 3 x 3

2

Bias

=

| | |
|---|---|
| 11 | 12 |
| 14 | 11 |
| | |

# 5 - Practice

## 2 Exercise – Padding

| | | |
|---|---|---|
| 2 | 4 | 2 |
| 3 | 3 | 4 |
| 3 | 2 | 0 |
| 4 | 0 | 4 |
| 1 | 4 | 0 |

Input: 5 x 3

Padding: 1 x 1

| | | | | |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 2 | 4 | 2 | 0 |
| 0 | 3 | 3 | 4 | 0 |
| 0 | 3 | 2 | 0 | 0 |
| 0 | 4 | 0 | 4 | 0 |
| 0 | 1 | 4 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 |

Input: 7 x 5

Stride: 2 (2x2)

*

| | | |
|---|---|---|
| 1 | 0 | 1 |
| 1 | 1 | 1 |
| 0 | 1 | 0 |

Kernel: 3 x 3

2

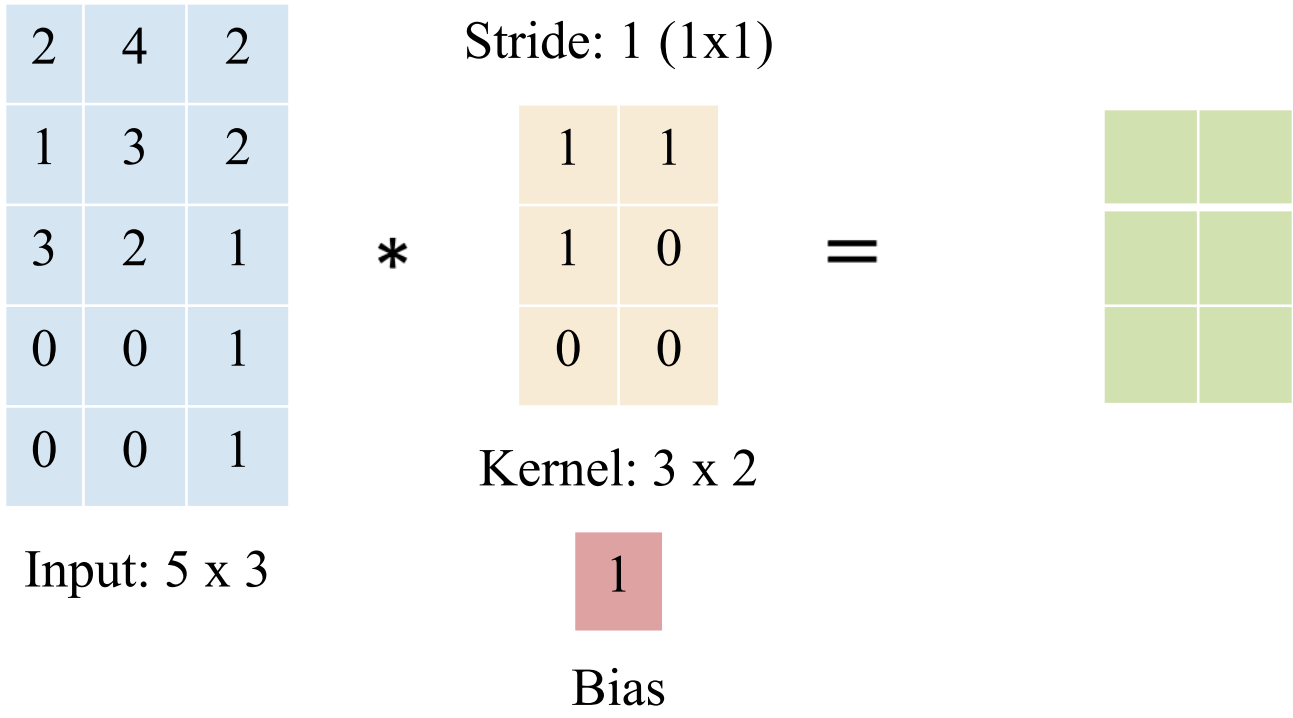
Bias

=

| | |
|---|---|
| 11 | 12 |
| 14 | 11 |
| 7 | 6 |

# 5 - Practice

## 3 Exercise – Convolutional Layer + Pooling



# 5 - Practice

## 3 Exercise – Convolutional Layer + Pooling

| | | |
|---|---|---|
| 2 | 4 | 2 |
| 1 | 3 | 2 |
| 3 | 2 | 1 |
| 0 | 0 | 1 |
| 0 | 0 | 1 |

Input: 5 x 3

*

Stride: 1 (1x1)

| | |
|---|---|
| 1 | 1 |
| 1 | 0 |
| 0 | 0 |

Kernel: 3 x 2

| |
|---|
| 1 |

Bias

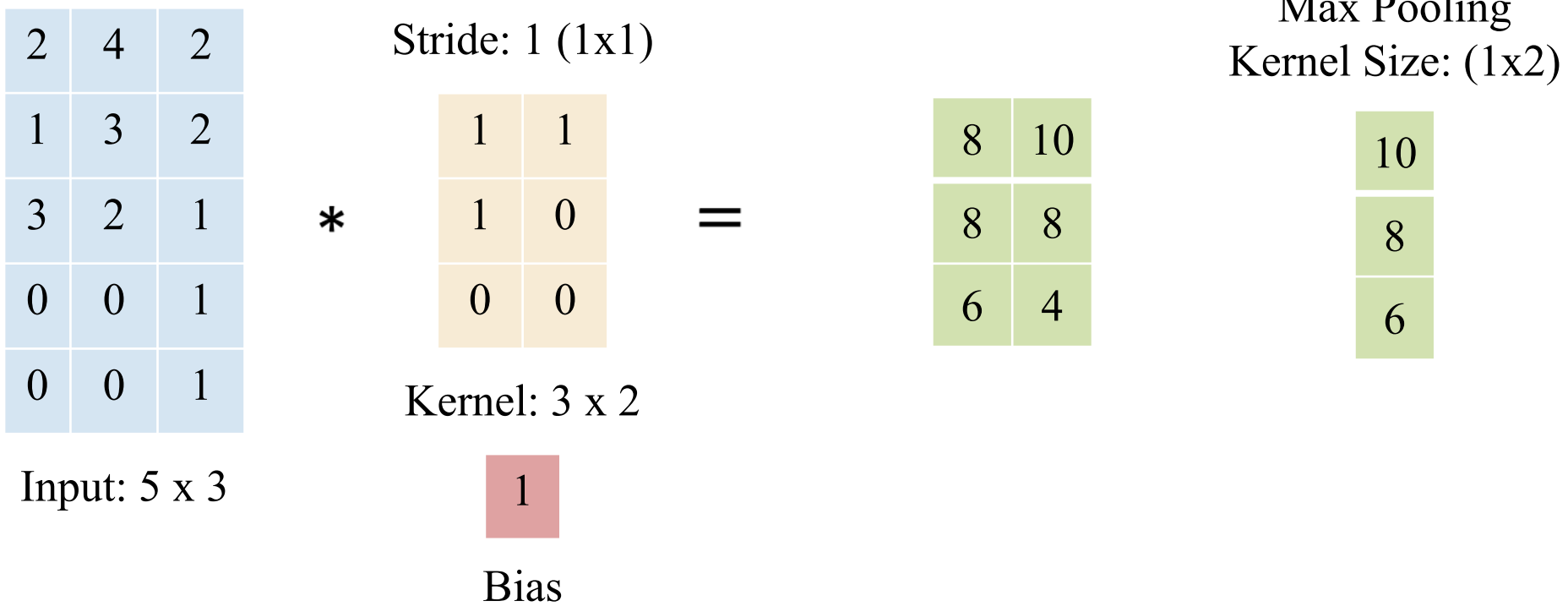
=

| | |
|---|---|
| 8 | 10 |
| 8 | 8 |
| 6 | 4 |

Max Pooling
Kernel Size: (1x2)

# 5 - Practice

## 3 Exercise – Convolutional Layer + Pooling



# 5 - Practice

## 4 Exercise – Pooling For Grayscale Image

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 43 | 43 | 0 | 0 |
| 0 | 30 | 250 | 230 | 125 | 251 | 0 |
| 0 | 191 | 38 | 0 | 0 | 81 | 0 |
| 0 | 241 | 0 | 35 | 119 | 250 | 0 |
| 0 | 49 | 193 | 198 | 83 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Input: 7 x 7

MaxPooling
2x2

=

| | | |
|---|---|---|
| | | |
| | | |
| | | |

Output: 3 x 3

# 5 - Practice

## 4 Exercise – Pooling For Grayscale Image

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 43 | 43 | 0 | 0 |
| 0 | 30 | 250 | 230 | 125 | 251 | 0 |
| 0 | 191 | 38 | 0 | 0 | 81 | 0 |
| 0 | 241 | 0 | 35 | 119 | 250 | 0 |
| 0 | 49 | 193 | 198 | 83 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Input: 7 x 7

MaxPooling 2x2 =

| | | |
|---|---|---|
| 0 | 43 | 43 |
| 191 | 250 | 251 |
| 241 | 198 | 250 |

Output: 3 x 3

# 5 - Practice

## 4 Exercise – Convolutional For Grayscale Image

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 43 | 43 | 0 | 0 |
| 0 | 30 | 250 | 230 | 125 | 251 | 0 |
| 0 | 191 | 38 | 0 | 0 | 81 | 0 |
| 0 | 241 | 0 | 35 | 119 | 250 | 0 |
| 0 | 49 | 193 | 198 | 83 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Input: 7 x 7

*

| | | |
|---|---|---|
| 1 | 0 | -1 |
| 1 | 0 | -1 |
| 1 | 0 | -1 |

Kernel: 3 x 3

=

| | | | | |
|---|---|---|---|---|
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

Output: 5 x 5

# 5 - Practice

## 4 Exercise – Convolutional For Grayscale Image

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 43 | 43 | 0 | 0 |
| 0 | 30 | 250 | 230 | 125 | 251 | 0 |
| 0 | 191 | 38 | 0 | 0 | 81 | 0 |
| 0 | 241 | 0 | 35 | 119 | 250 | 0 |
| 0 | 49 | 193 | 198 | 83 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Input: 7 x 7

*

| | | |
|---|---|---|
| 1 | 0 | -1 |
| 1 | 0 | -1 |
| 1 | 0 | -1 |

Kernel: 3 x 3

=

| | | | | |
|---|---|---|---|---|
| -250 | -243 | 82 | 22 | 168 |
| -288 | 34 | 206 | -59 | 168 |
| 212 | 657 | 294 | 185 | 244 |
| -155 | 248 | 29 | 64 | 202 |
| -193 | 127 | 229 | 486 | 202 |

Output: 5 x 5

# 5 - Practice

## 4 Exercise – Convolutional For Grayscale Image

| | | | | |
|---|---|---|---|---|
| -250 | -243 | 82 | 22 | 168 |
| -288 | 34 | 206 | -59 | 168 |
| 212 | 657 | 294 | 185 | 244 |
| -155 | 248 | 29 | 64 | 202 |
| -193 | 127 | 229 | 486 | 202 |

Input: 5 x 5

MaxPooling
Kernel: 2

| | |
|---|---|
| 34 | 206 |
| 657 | 297 |

AI VIET NAM

@aivietnam.edu.vn

# Thanks!

## Any questions?