Building Blocks of CNNs

Saeid Nahavandi
Distinguished Professor
snahavandi@swin.edu.au

Parham Kebria
Senior Research Scientist
parhamkebria@ieee.org

Advanced Study Institute: Artificial Intelligence for Disaster Management
Orlando, November 17 – 25, 2025

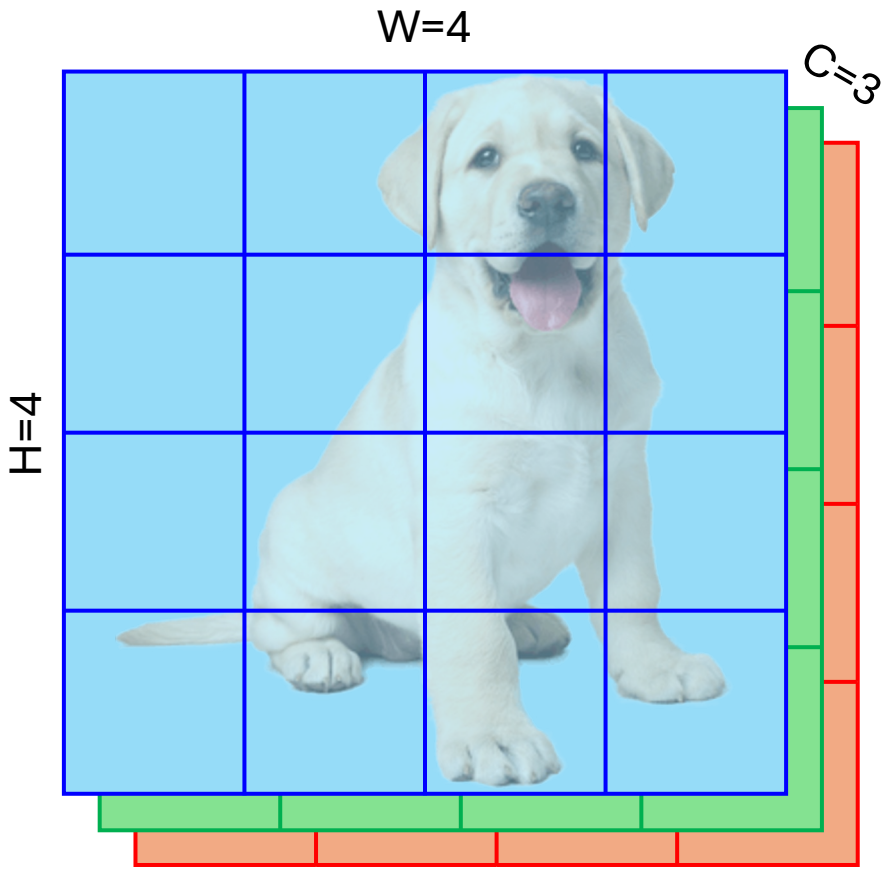
NATO OTAN
This activity is supported by: The NATO Science for Peace and Security Programme

SWINBURNE UNIVERSITY OF TECHNOLOGY

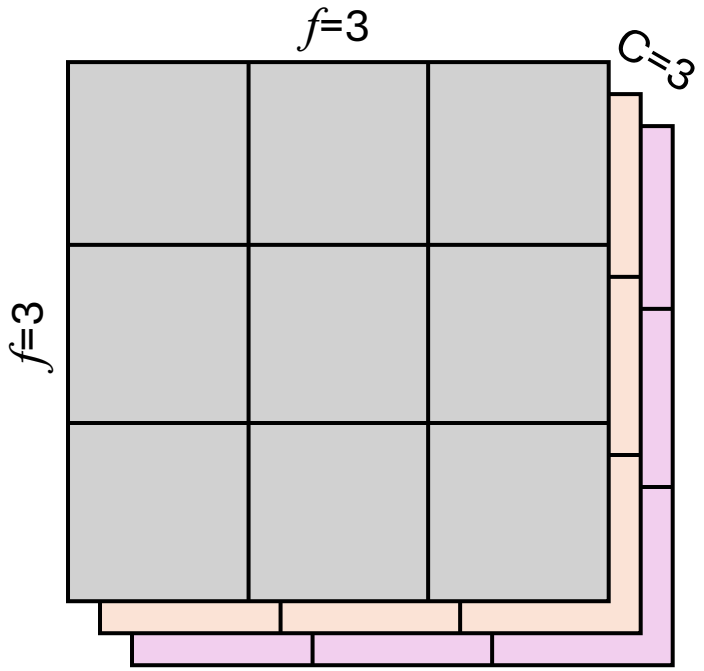
UCF

Convolutional Neural Networks (CNN)

The convolution operation



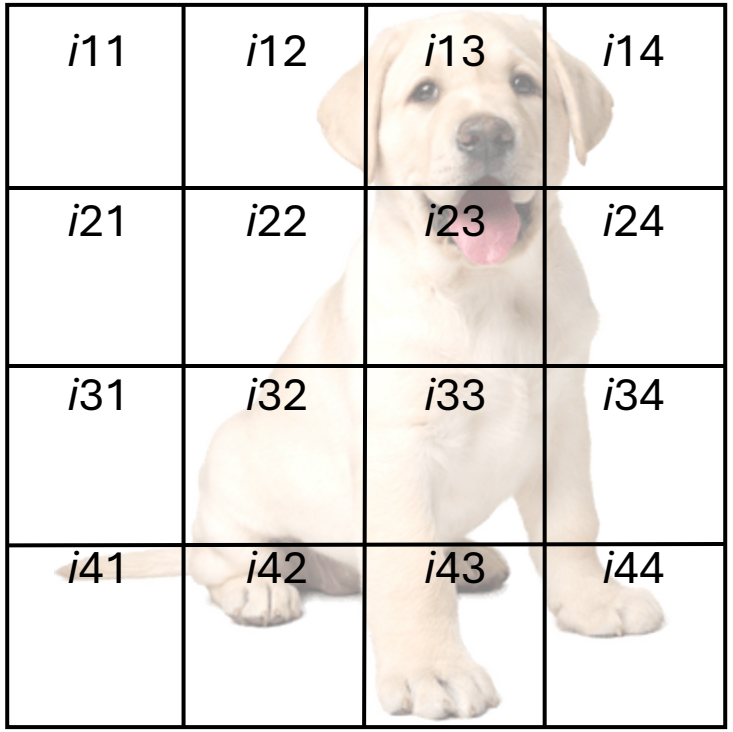
Input Tensor



Convolution filter

Convolutional Neural Networks (CNN)

The convolution operation



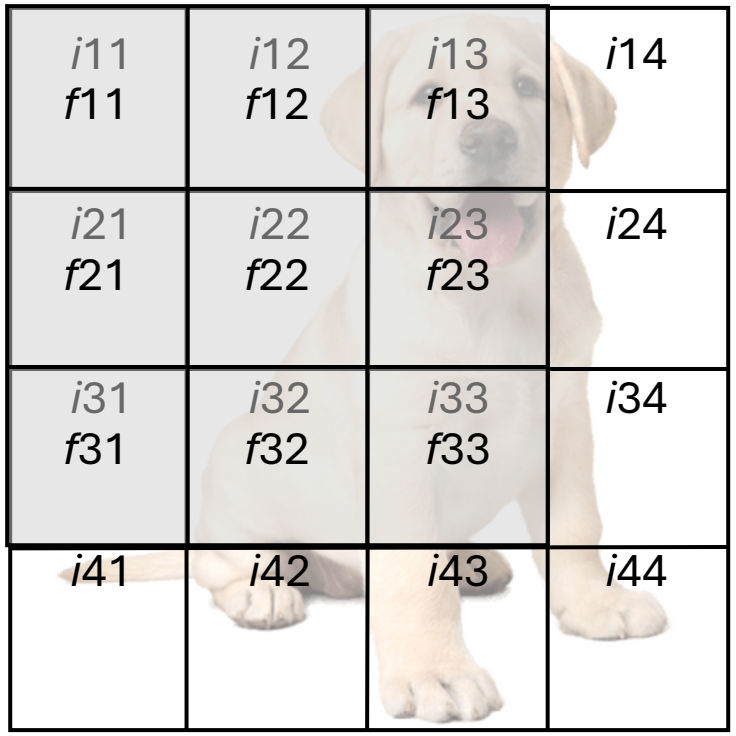
Input Tensor

*f*11	*f*12	*f*13
*f*21	*f*22	*f*23
*f*31	*f*32	*f*33

Convolution filter

Convolutional Neural Networks (CNN)

The convolution operation

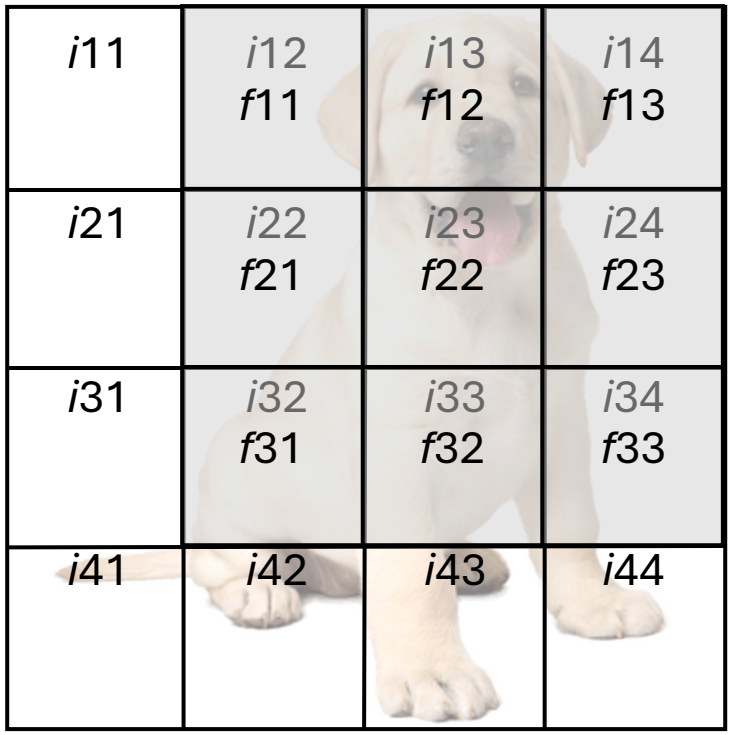


Input Tensor

$$c11 = i11 \times f11 + i12 \times f12 + i13 \times f13 + i21 \times f21 + i22 \times f22 + i23 \times f23 + i31 \times f31 + i32 \times f32 + i33 \times f33$$

Convolutional Neural Networks (CNN)

The convolution operation



Input Tensor

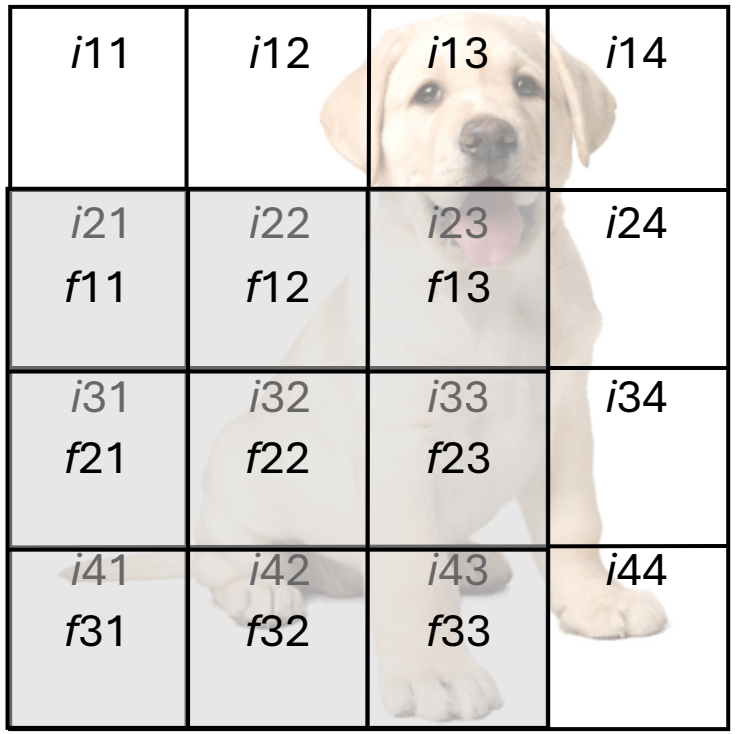
$$c12 = i12xf11 + i13xf12 + i14xf13 + i22xf21 + i23xf22 + i24xf23 + i32xf31 + i33xf32 + i34xf33$$

c11	c12

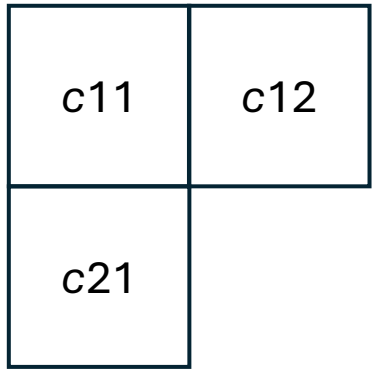
Convolutional Neural Networks (CNN)

The convolution operation

$$c21 = i21 \times f11 + i22 \times f12 + i23 \times f13 + i31 \times f21 + i32 \times f22 + i33 \times f23 + i41 \times f31 + i42 \times f32 + i43 \times f33$$

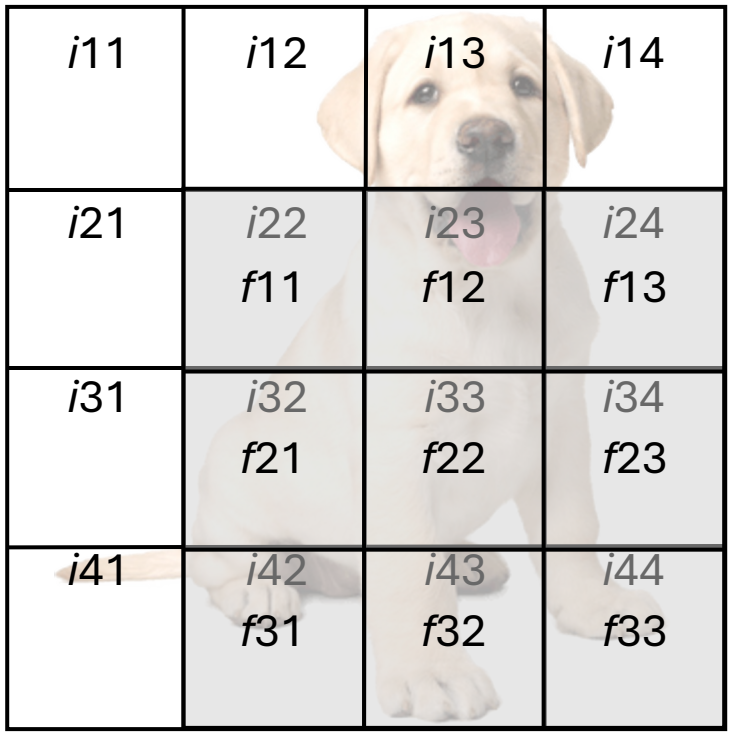


Input Tensor



Convolutional Neural Networks (CNN)

The convolution operation



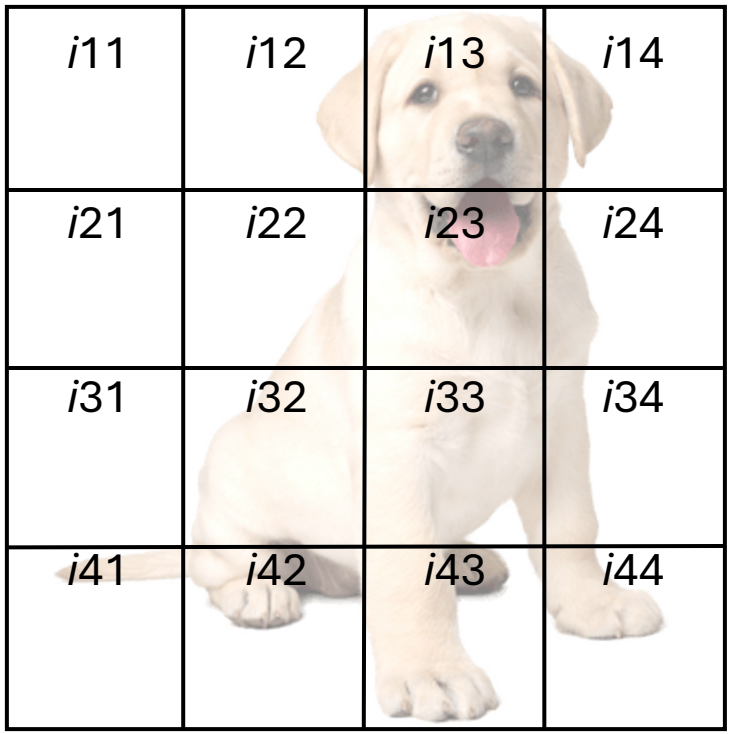
Input Tensor

$$c22 = i22xf11 + i23xf12 + i24xf13 + i32xf21 + i33xf22 + i34xf23 + i42xf31 + i43xf32 + i44xf33$$

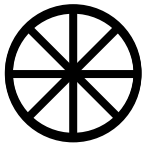
c11	c12
c21	c22

Convolutional Neural Networks (CNN)

The convolution operation



Input Tensor



f11	f12	f13
f21	f22	f23
f31	f32	f33

=>

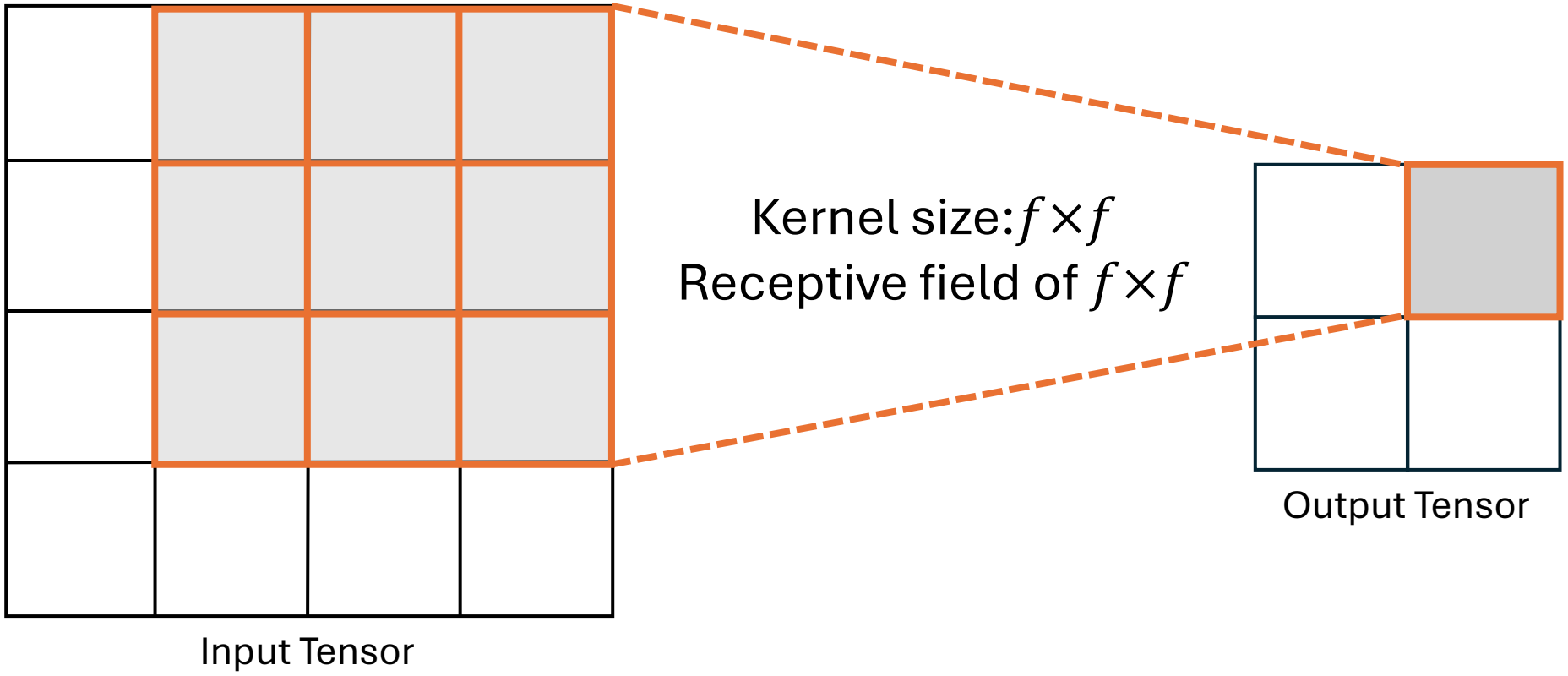
c11	c12
c21	c22

Output Tensor

$$c_{pq} = \sum_{i=1}^{f} \sum_{j=1}^{f} w_{ij} x_{i+p,j+q}$$

Convolutional Neural Networks (CNN)

The convolution operation

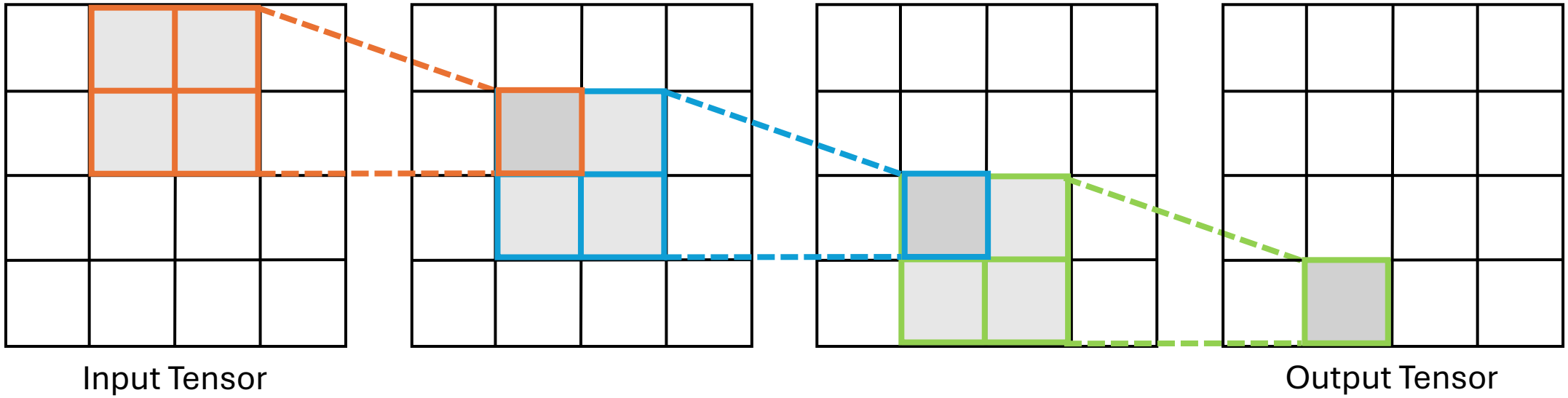


Convolutional Neural Networks (CNN)

The convolution operation

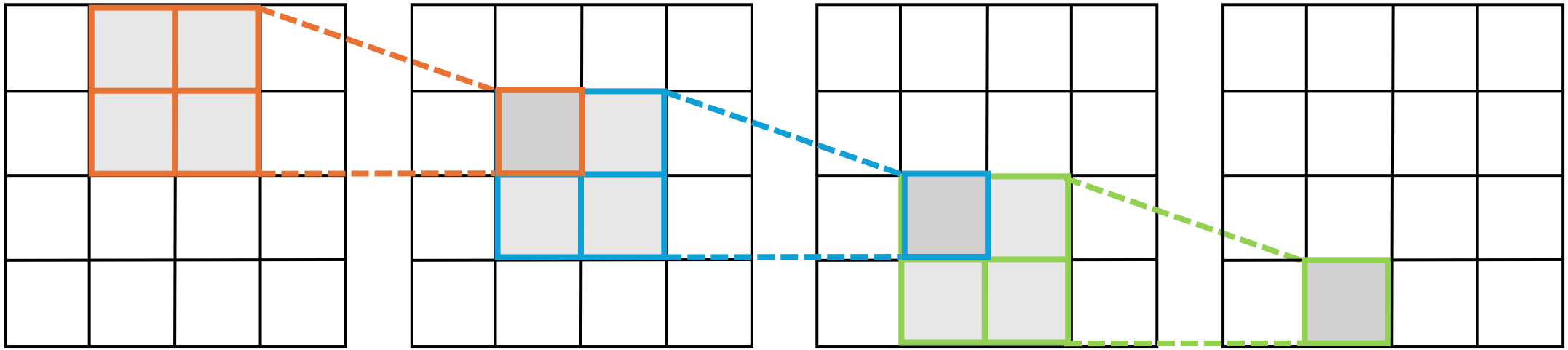
Each conv layer $f \times f$ adds $f - 1$ to the overall receptive field

$\Rightarrow L$ conv layers results in $1 + L \times (f - 1)$ receptive field



Each conv layer $f \times f$ adds $f - 1$ to the overall receptive field

$\Rightarrow L$ conv layers results in $1 + L \times (f - 1)$ receptive field



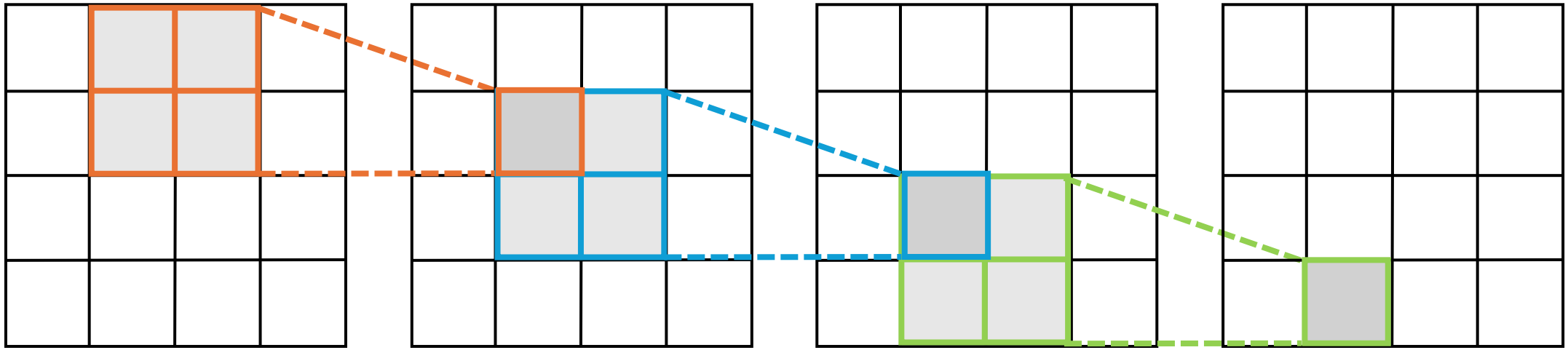
Challenge: For large images we need *MANY* layers
to be able to **see** the whole input image in the output !

Convolutional Neural Networks (CNN)

The convolution operation

Each conv layer $f \times f$ adds $f - 1$ to the overall receptive field

$\Rightarrow L$ conv layers results in $1 + L \times (f - 1)$ receptive field

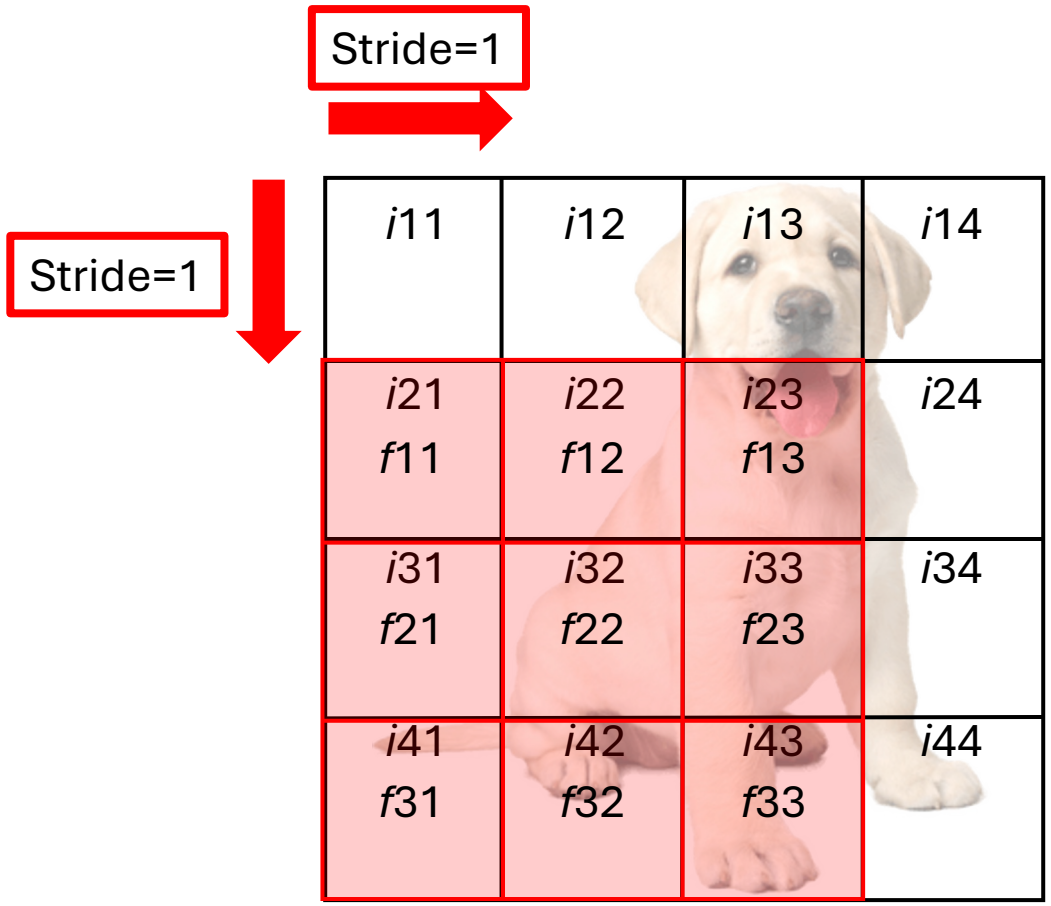


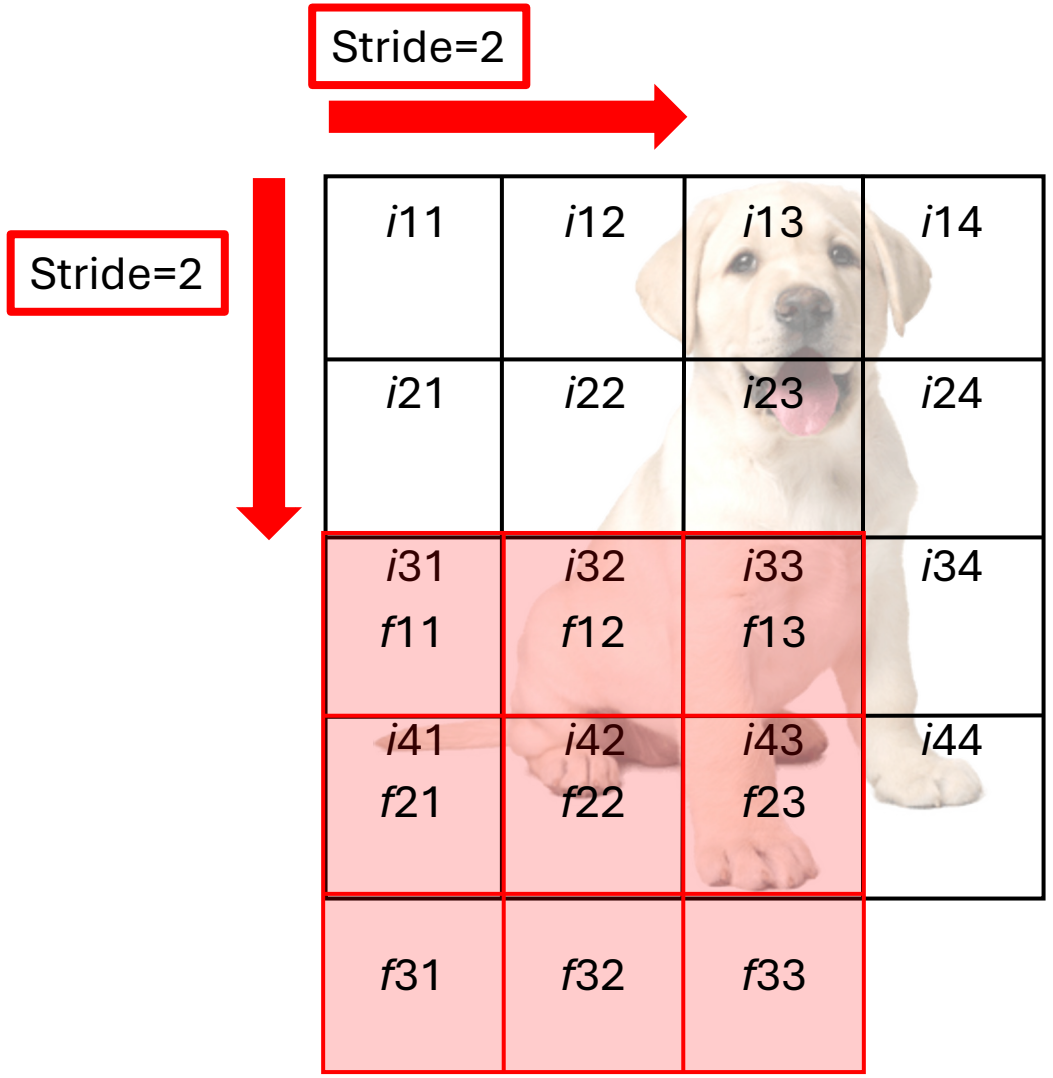
Challenge: For large images we need *MANY* layers to be able to ***see*** the whole input image in the output !

Striding to downsample

Convolutional Neural Networks (CNN)

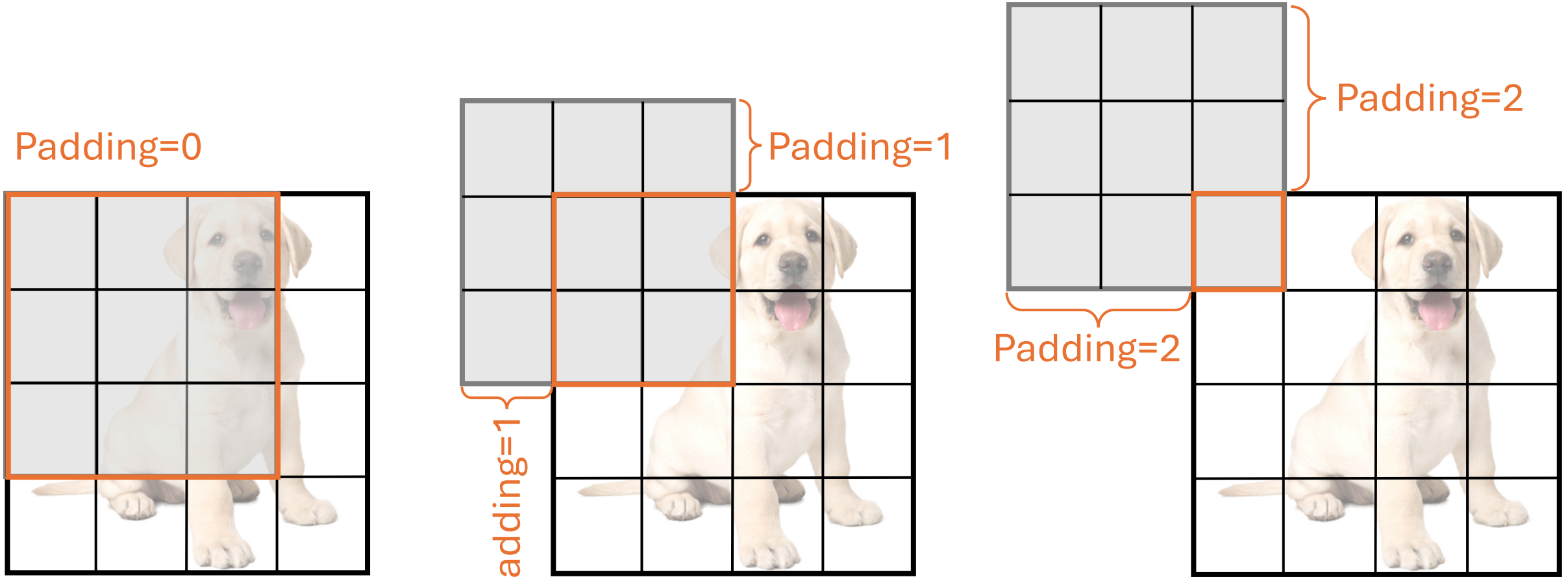
Stride





Convolutional Neural Networks (CNN)

Padding



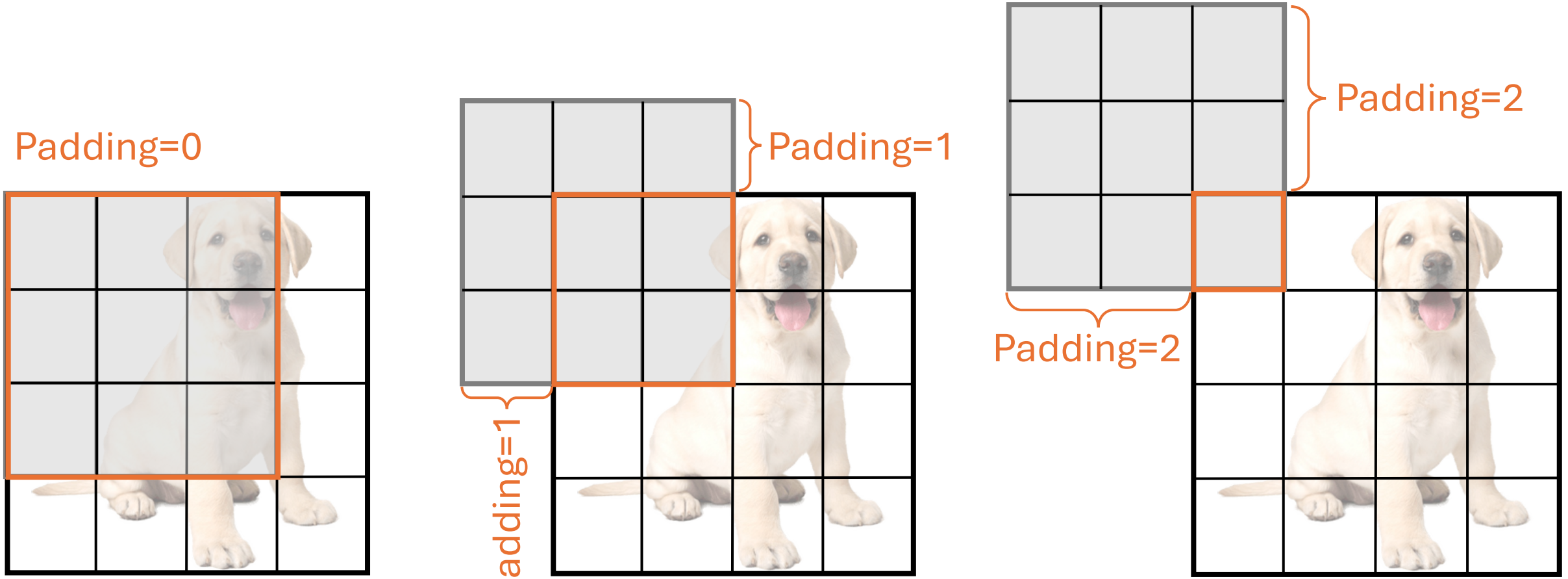
No padding (padding=0)
4×4 input → 2×2 output
(shrinks)

Padding=1
4×4 input → 4×4 output
(same size)

Padding=2
4×4 input → 6×6 output
(grows)

Convolutional Neural Networks (CNN)

Padding



No padding (padding=0)
4×4 input → 2×2 output
(shrinks)

Padding=1
4×4 input → 4×4 output
(same size)

Padding=2
4×4 input → 6×6 output
(grows)

To preserve input size: `padding = (filter_size - 1) // 2`

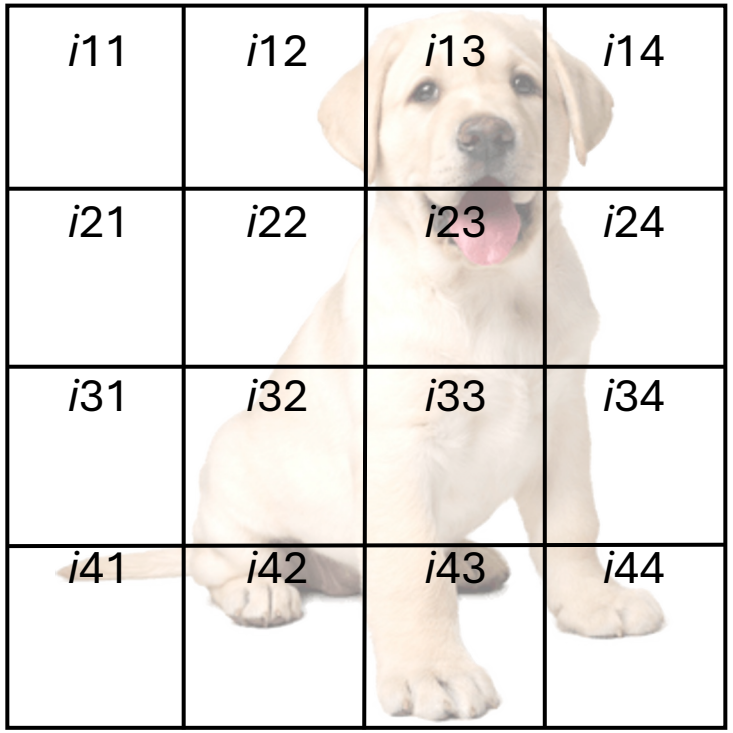
```
output = (input + 2*padding - filter_size) / stride + 1
```

Convolutional Neural Networks (CNN)

Convolution Filters (kernels)

```
tf.keras.layers.Conv2D(
    filters=n,
    kernel_size=(h_f, w_f),
    stride=s, padding=p,
    input_shape=(h_in, w_in, c_in))
```

```
torch.nn.Conv2d(
    kernel_size=(h_f,w_f),
    stride=s, padding=p,
    in_channels=c_in, out_channels=n,
    input_shape=(c_in, h_in, w_in))
```



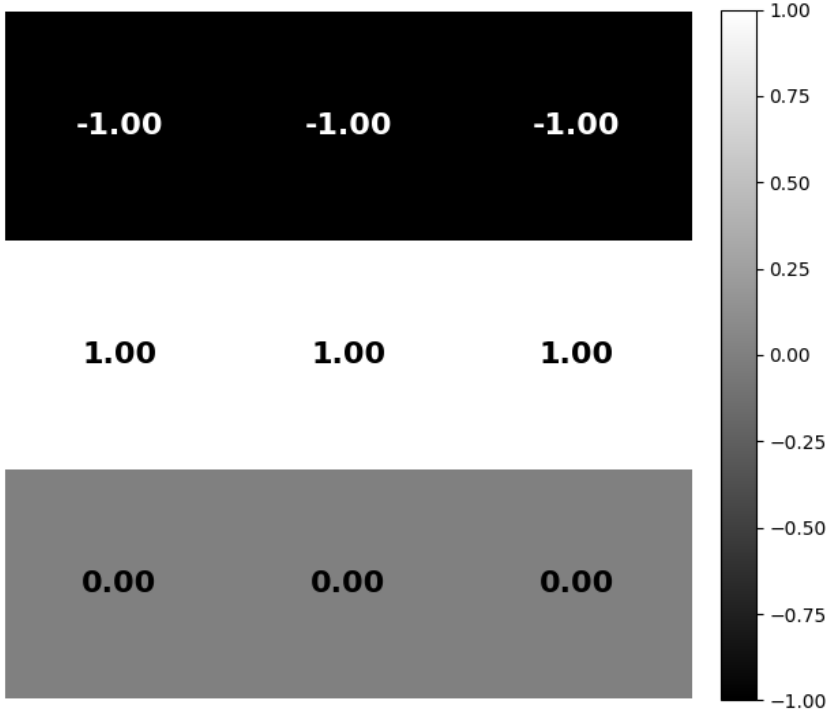
Input tensor

*f*11	*f*12	*f*13
*f*21	*f*22	*f*23
*f*31	*f*32	*f*33

Filter (kernel) tensor

Convolutional Neural Networks (CNN)

Some conventional filters



Convolutional Neural Networks (CNN)

Some conventional filters

Edge Detection (Laplacian)

-1.00	-1.00	-1.00
-1.00	8.00	-1.00
-1.00	-1.00	-1.00

Sharpen Filter

0.00	-1.00	0.00
-1.00	5.00	-1.00
0.00	-1.00	0.00

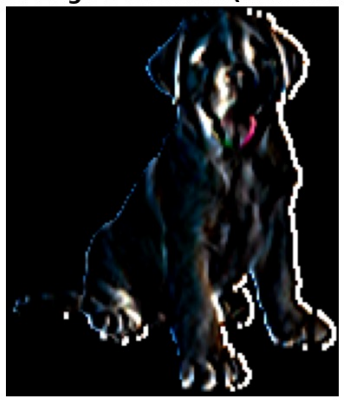
Box Blur (5x5)

0.04	0.04	0.04	0.04	0.04
0.04	0.04	0.04	0.04	0.04
0.04	0.04	0.04	0.04	0.04
0.04	0.04	0.04	0.04	0.04
0.04	0.04	0.04	0.04	0.04

Gaussian Blur (3x3)

0.06	0.12	0.06
0.12	0.25	0.12
0.06	0.12	0.06

Sobel Edge Detection (X-direction)

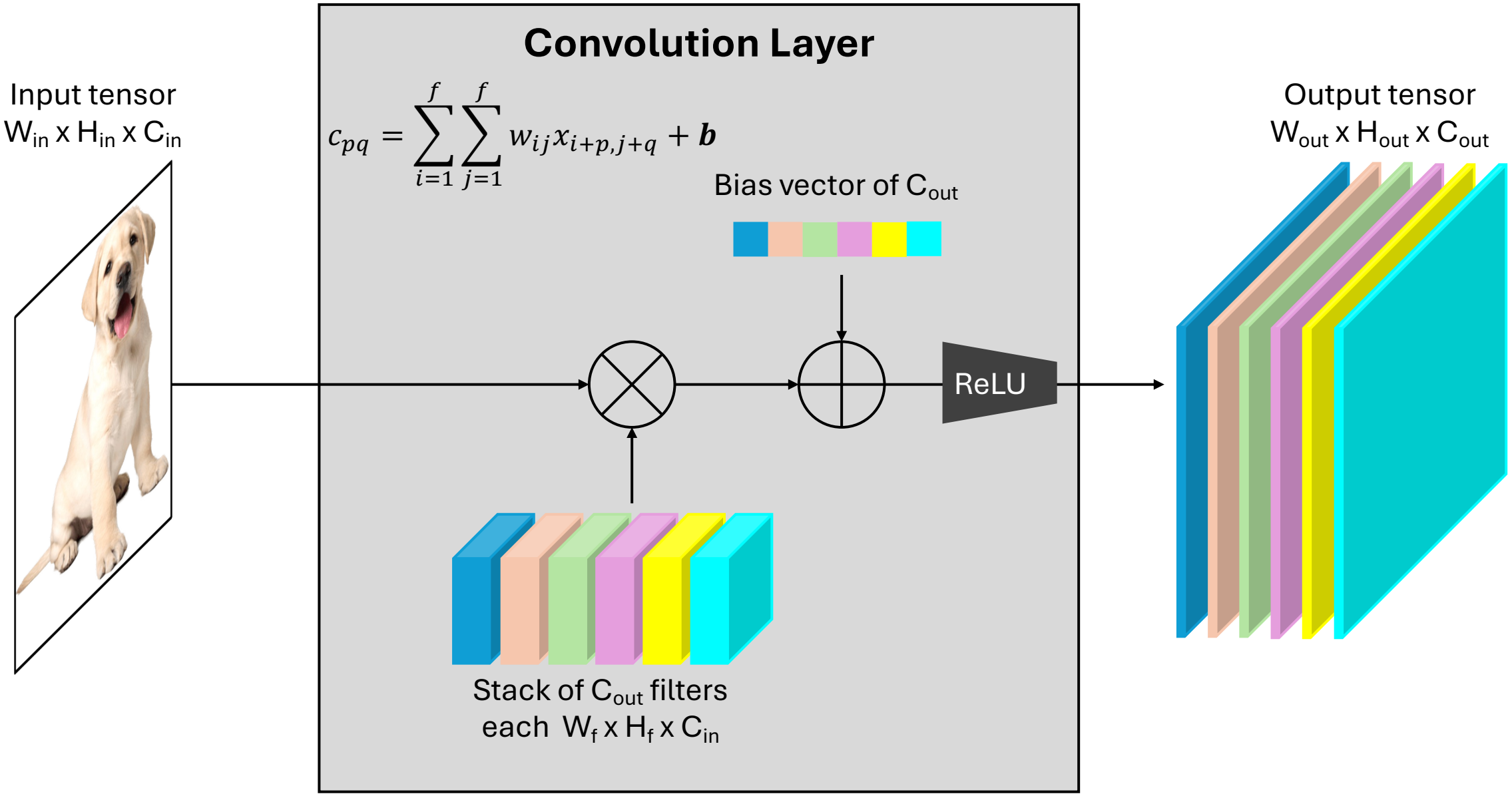


-1.00	0.00	1.00
-2.00	0.00	2.00
-1.00	0.00	1.00

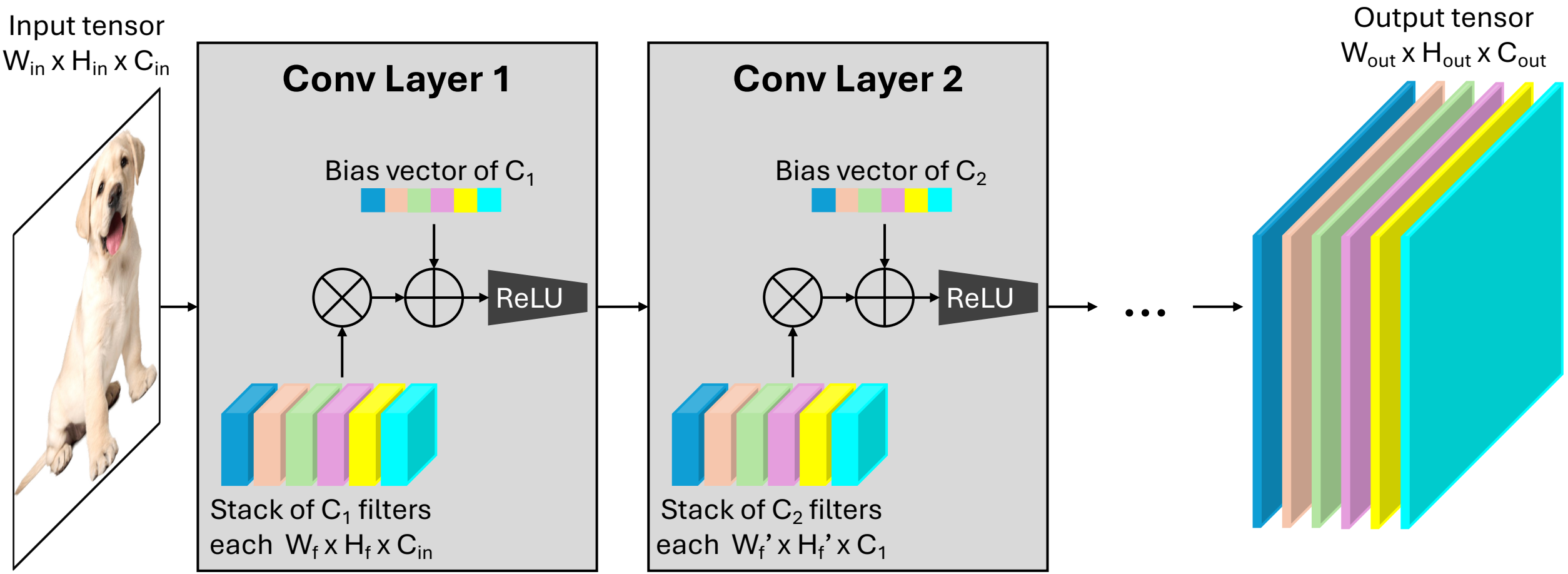
Sobel Edge Detection (Y-direction)

-1.00	-2.00	-1.00
0.00	0.00	0.00
1.00	2.00	1.00

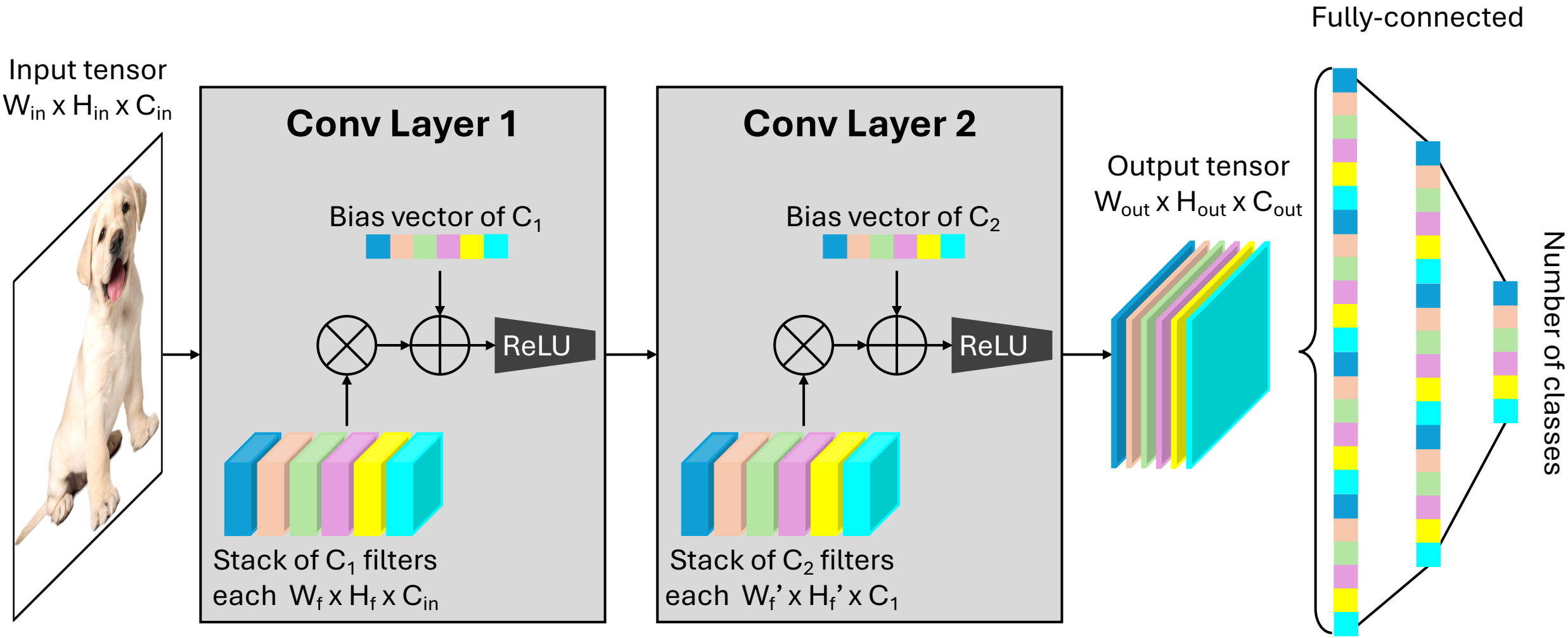
Convolutional Neural Networks (CNN)



Convolutional Neural Networks (CNN)



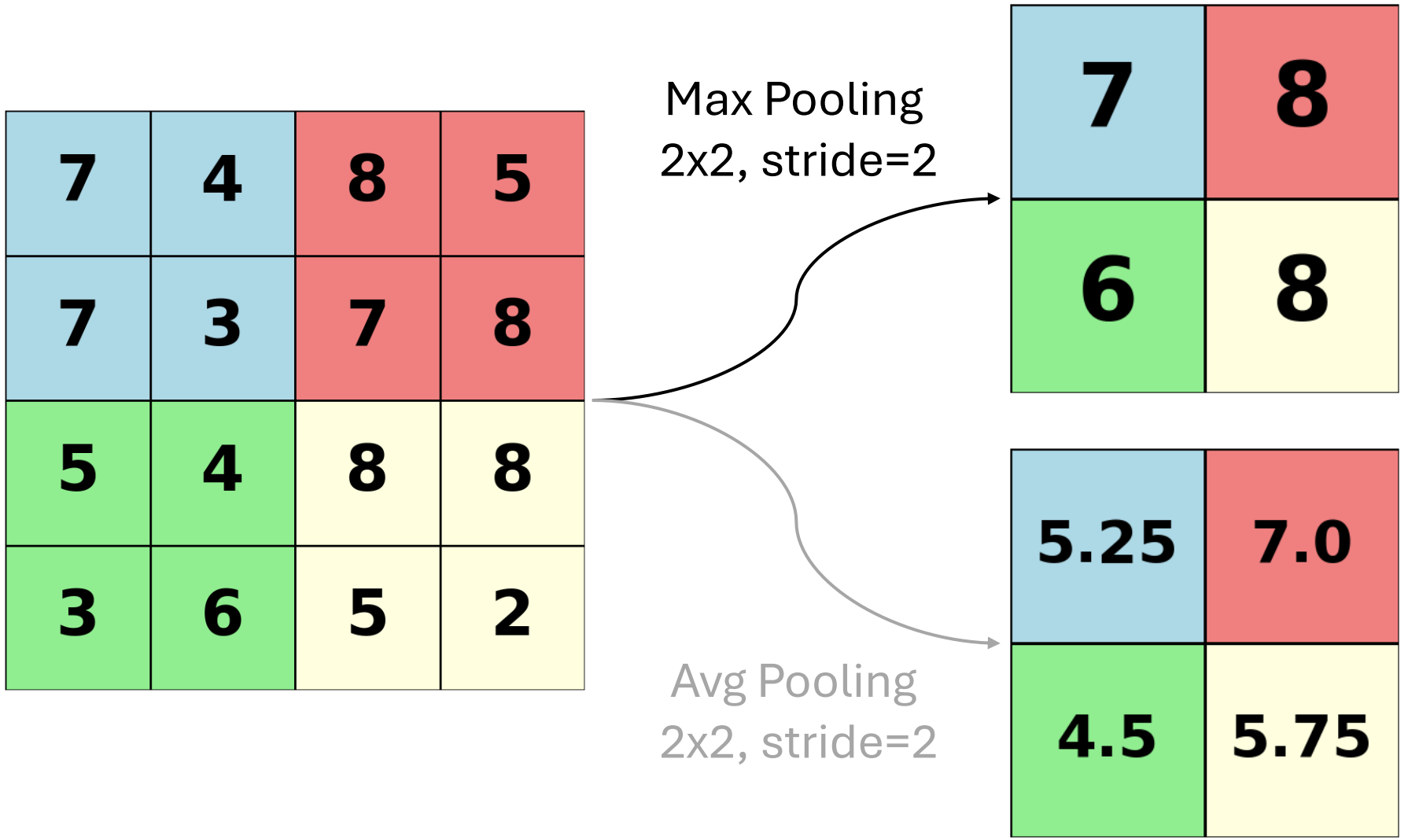
Convolutional Neural Networks (CNN)



Convolutional Neural Networks (CNN)

Pooling

- ✓ Dimension Reduction
- ✓ Spatial invariance

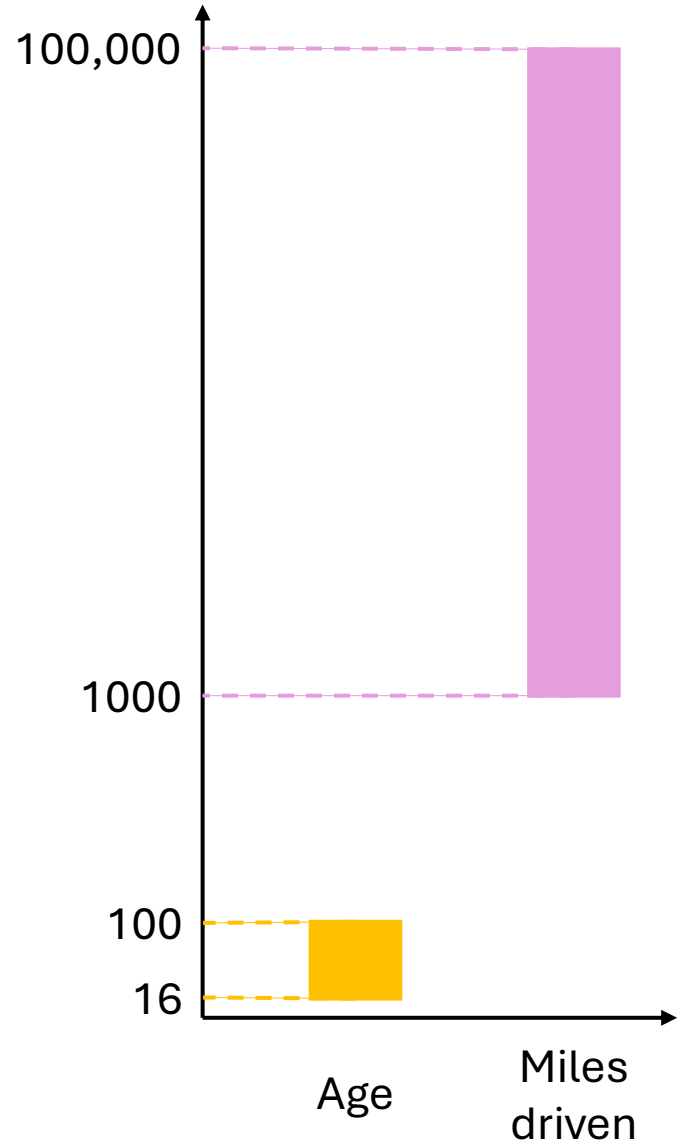



```python
import tensorflow as tf

tf.keras.layers.MaxPooling2D(
    pool_size=(h_f,w_f),
    strides=s)
```

```python
import torch

torch.nn.MaxPool2d(
    kernel_size=(h_f,w_f),
    stride=s)
```

Convolutional Neural Networks (CNN)

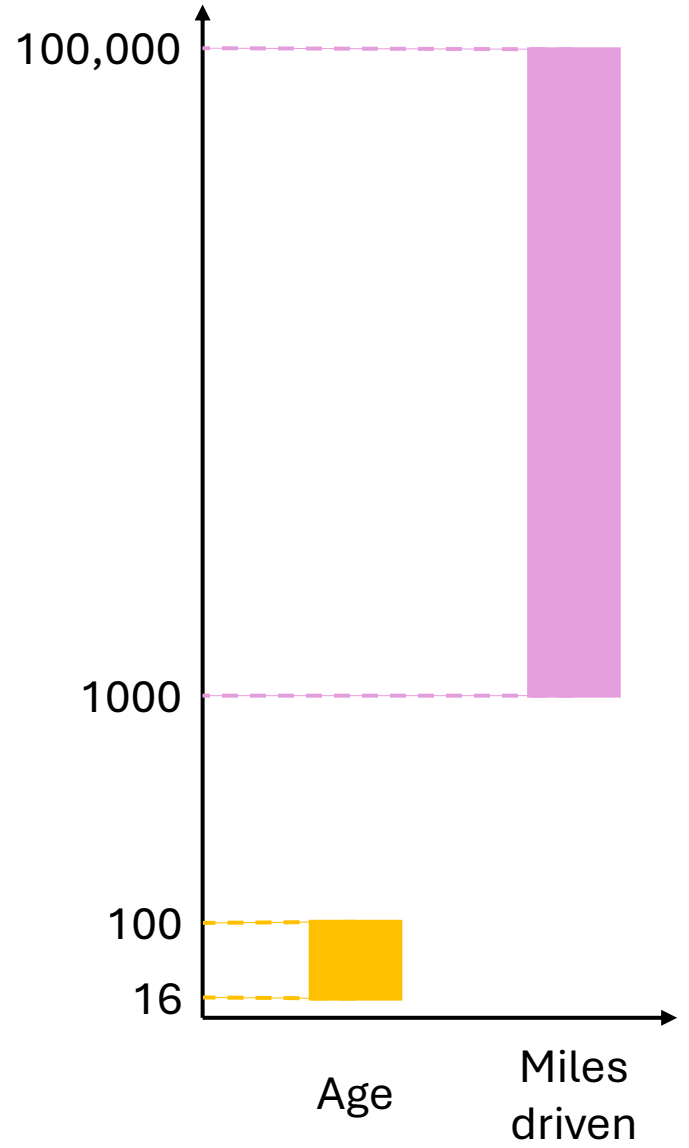
Batch normalization & Dropout



Convolutional Neural Networks (CNN)

Batch normalization & Dropout

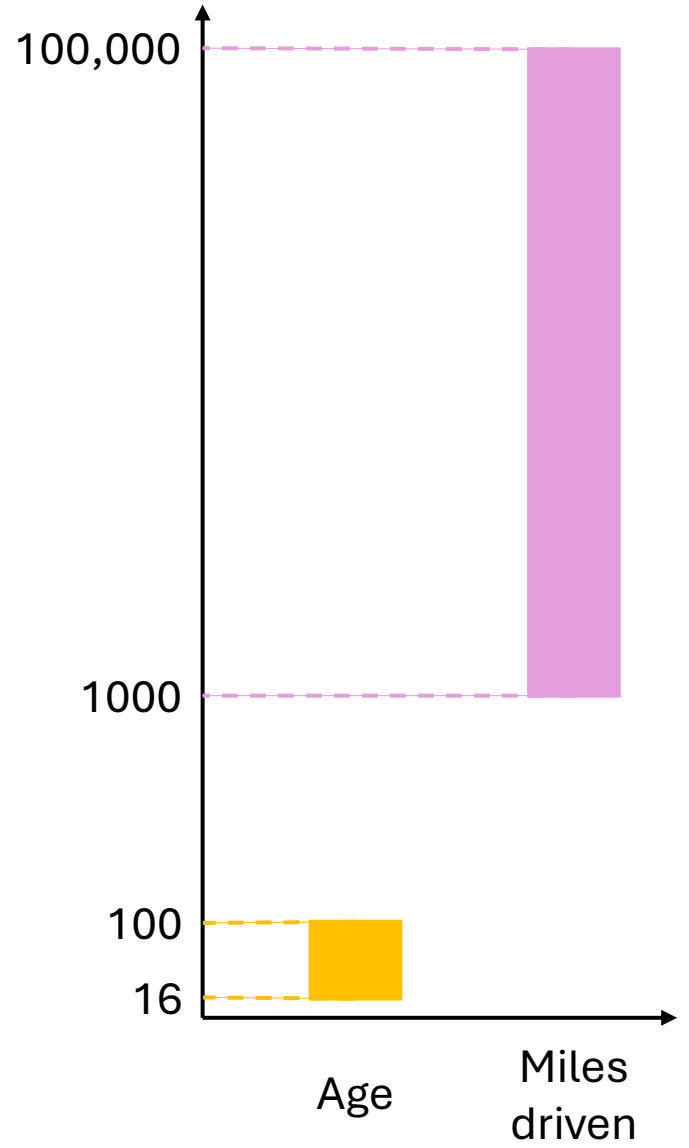
- Biased outputs



Convolutional Neural Networks (CNN)

Batch normalization & Dropout

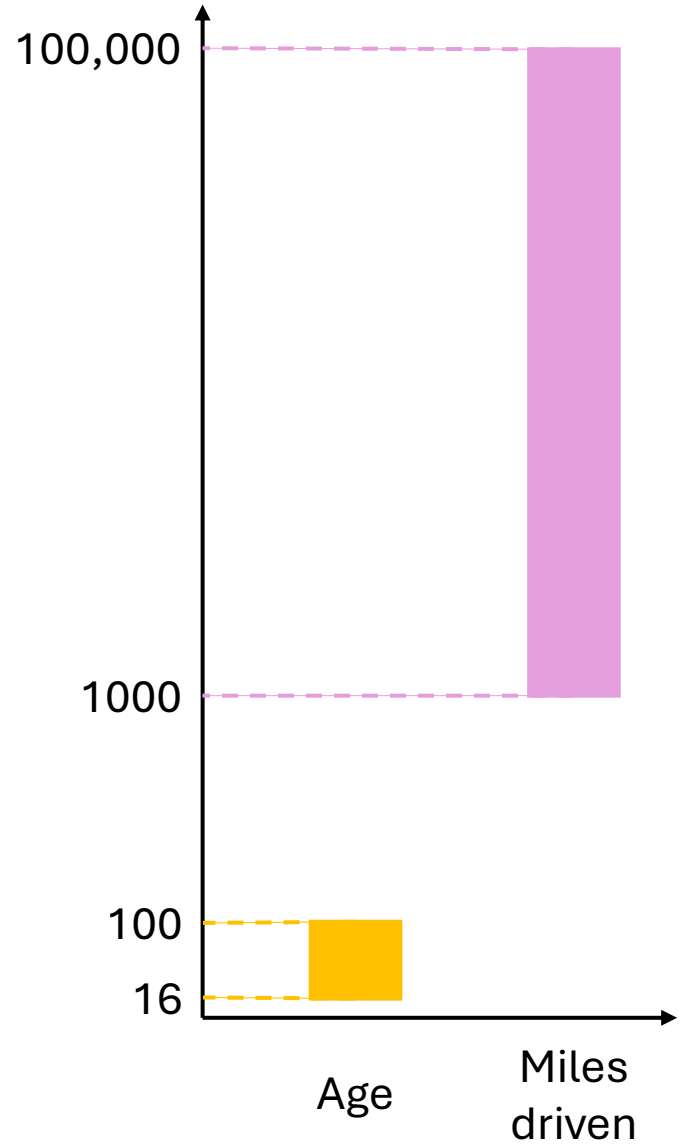
- Biased outputs
- Inefficient training



Convolutional Neural Networks (CNN)

Batch normalization & Dropout

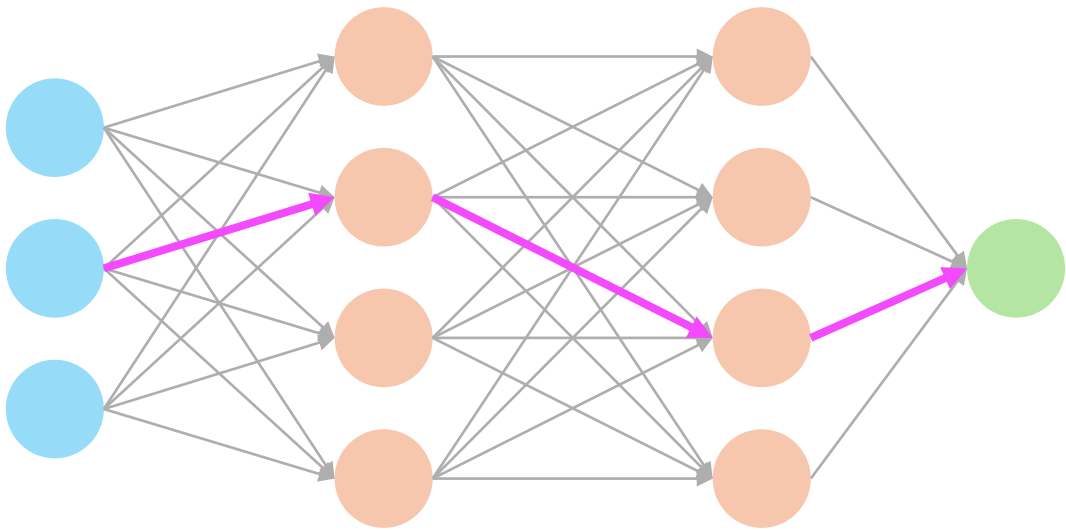
- Biased outputs
- Inefficient training
- Gradient explosion

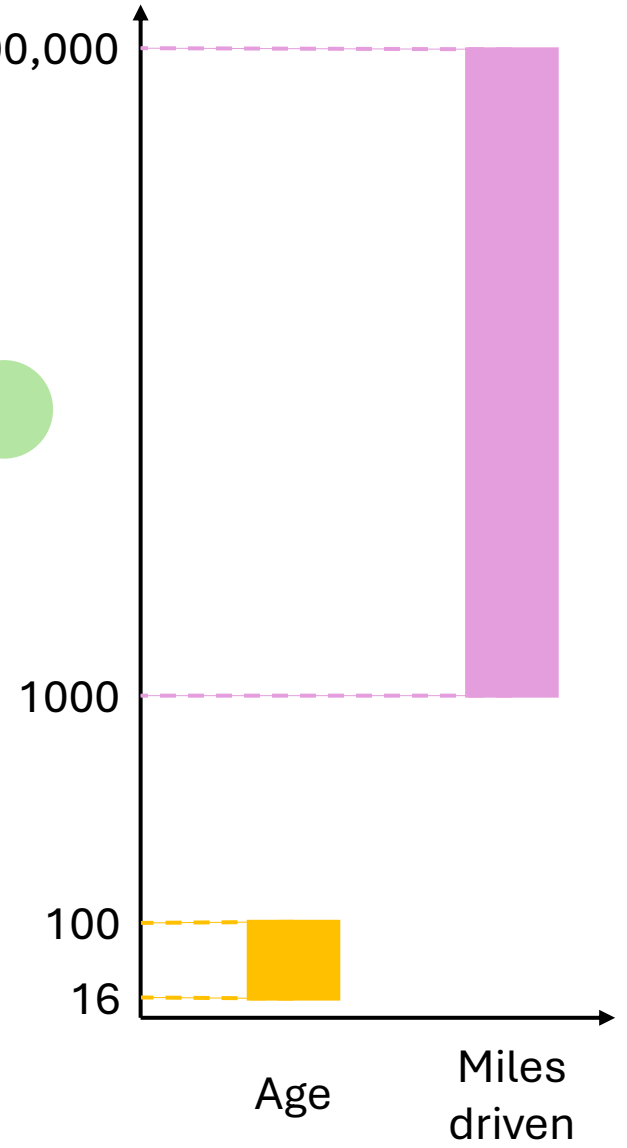


Convolutional Neural Networks (CNN)

Batch normalization & Dropout

- Biased outputs
- Inefficient training
- Gradient explosion
- Imbalanced weights



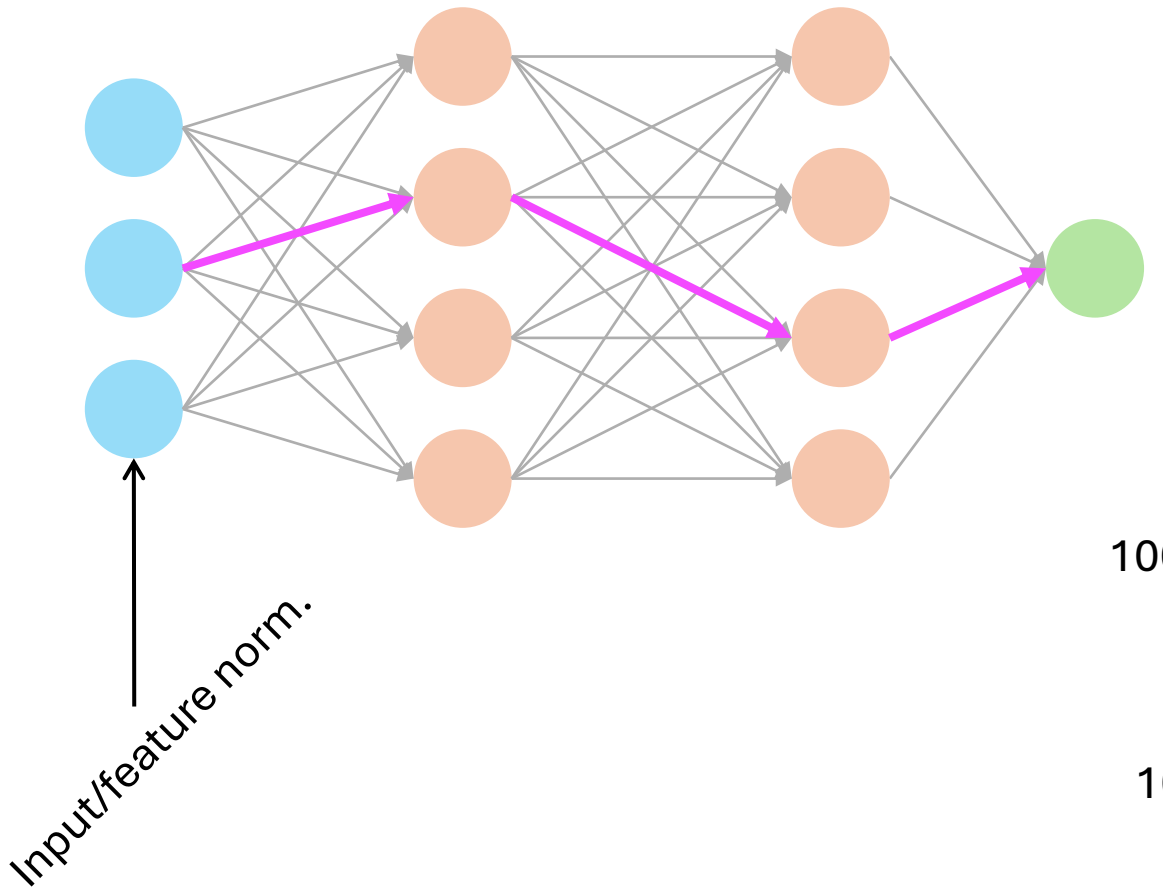


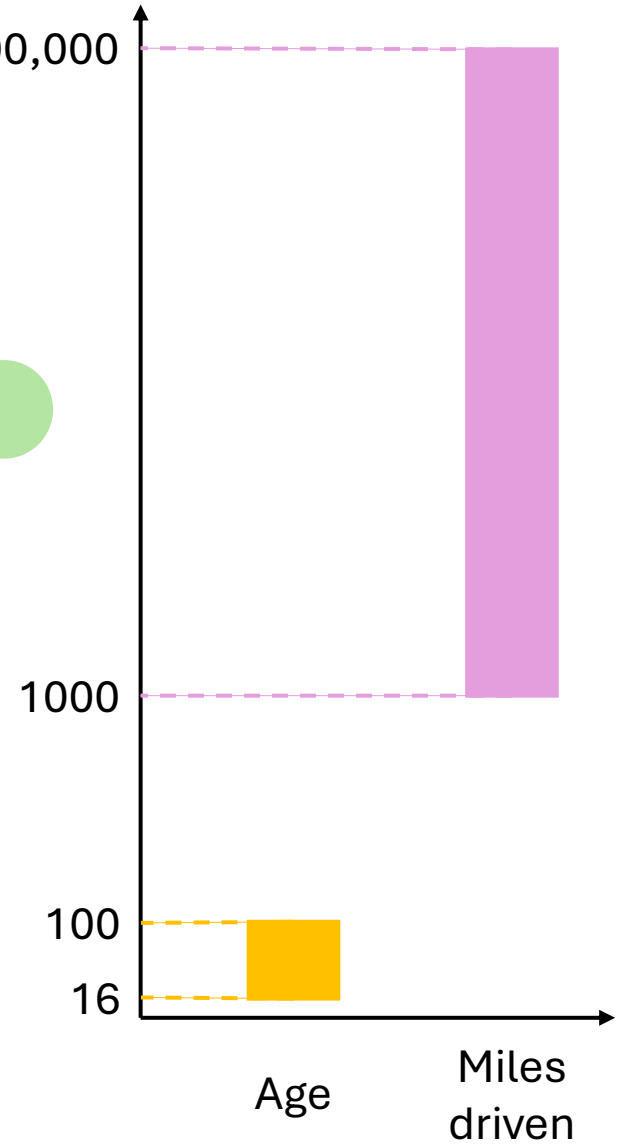
Convolutional Neural Networks (CNN)

Batch normalization & Dropout

- Biased outputs
- Inefficient training
- Gradient explosion
- Imbalanced weights

$$z = \frac{x - \mu}{\sigma}$$



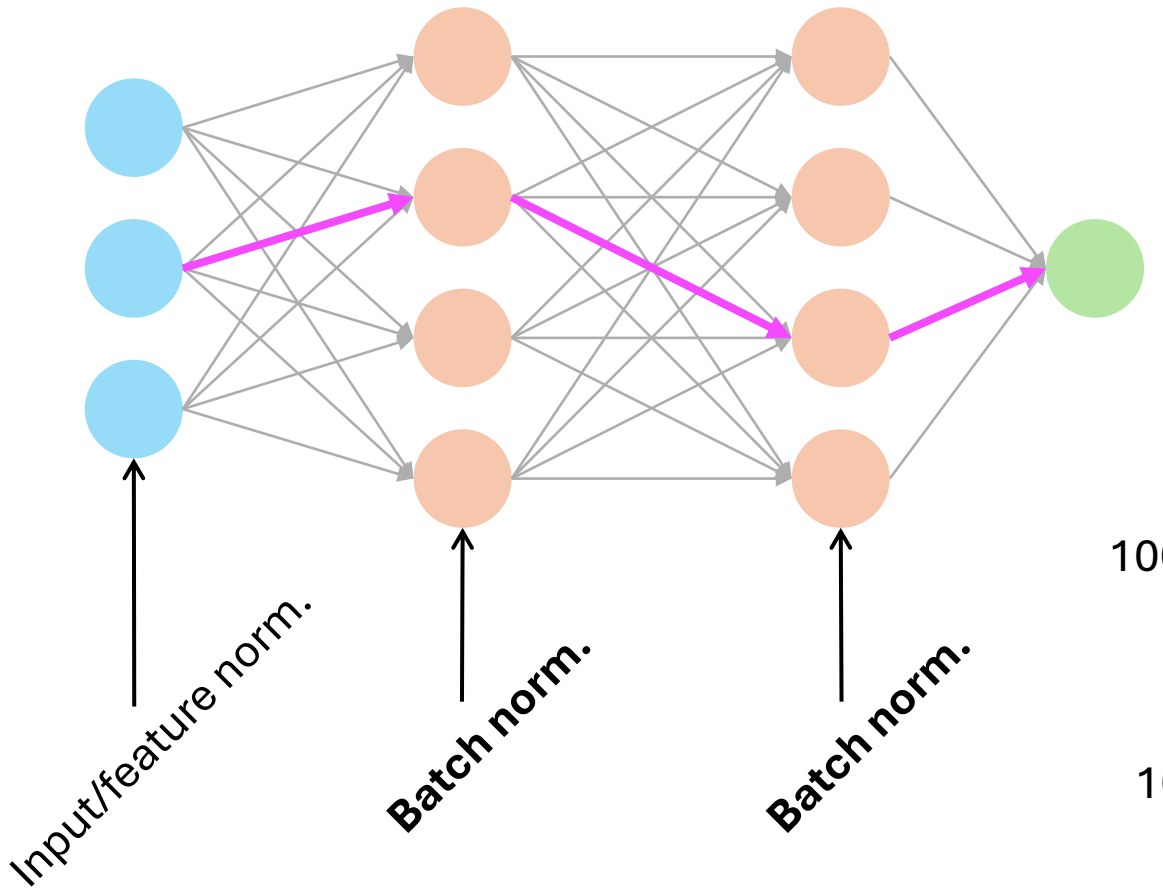


Convolutional Neural Networks (CNN)

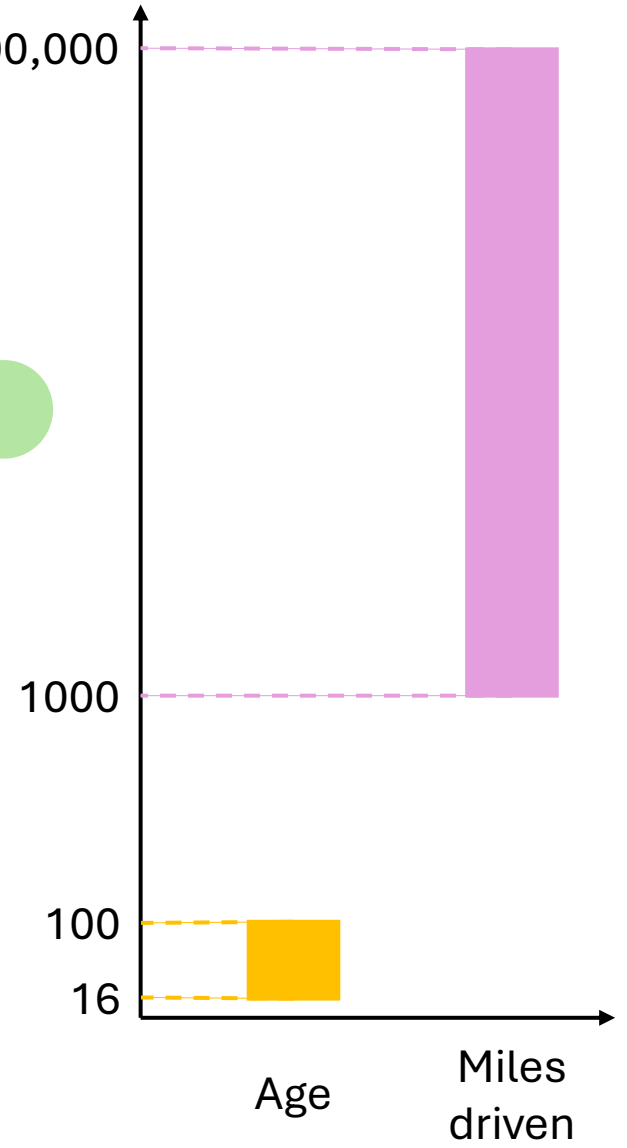
Batch normalization & Dropout

- Biased outputs
- Inefficient training
- Gradient explosion
- Imbalanced weights

$$z = \frac{x - \mu}{\sigma}$$



$$z_{l+1} = g \times z_l + b$$

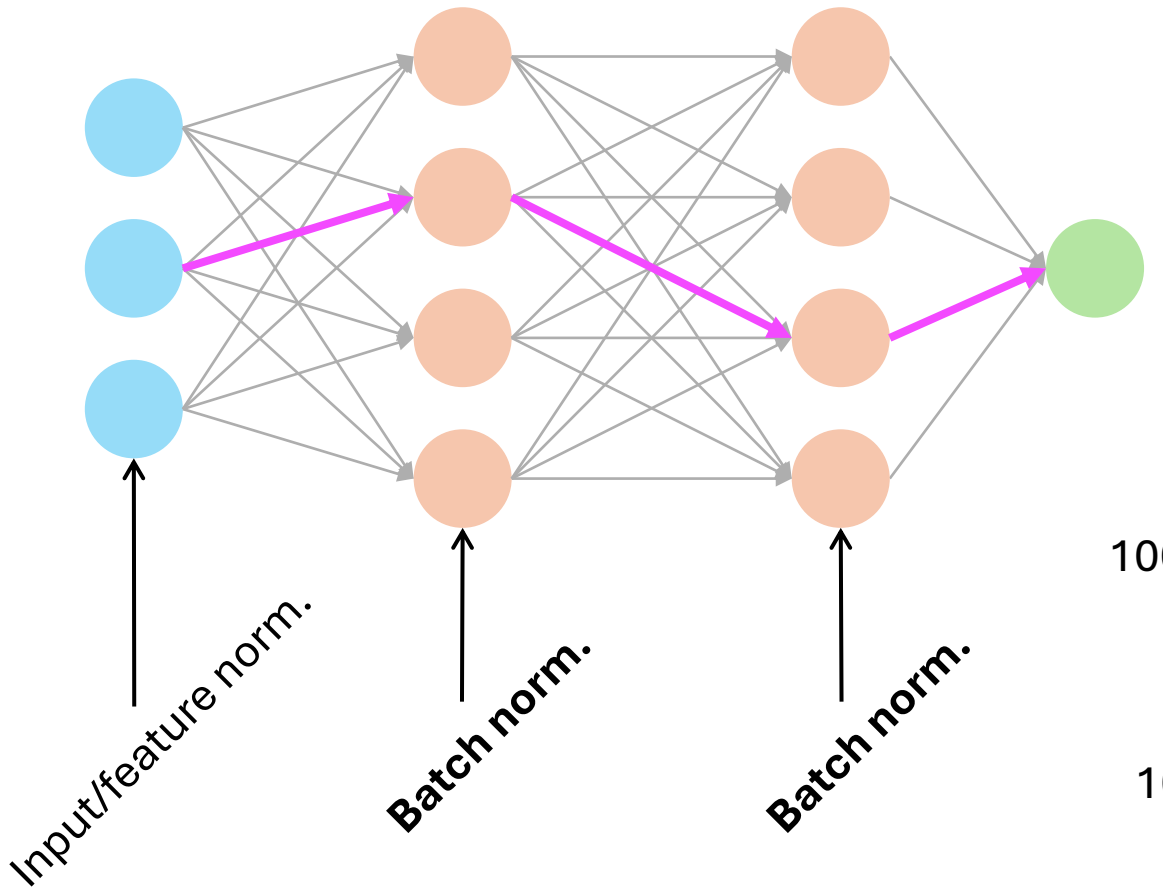


Convolutional Neural Networks (CNN)

Batch normalization & Dropout

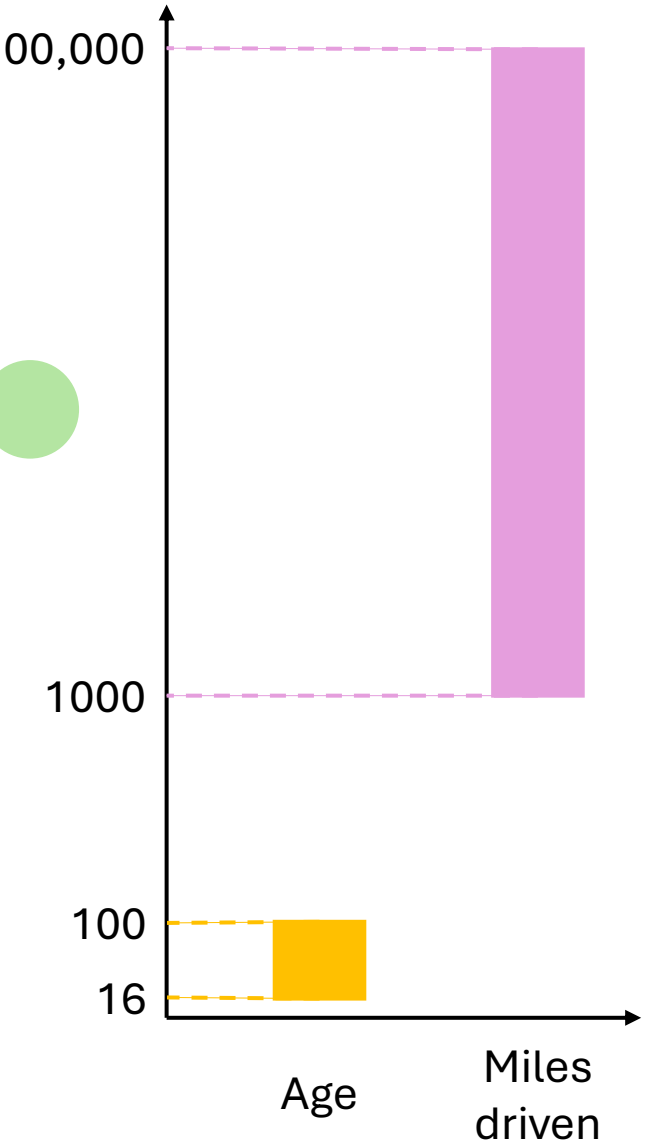
- Biased outputs
- Inefficient training
- Gradient explosion
- Imbalanced weights

$$z = \frac{x - \mu}{\sigma}$$



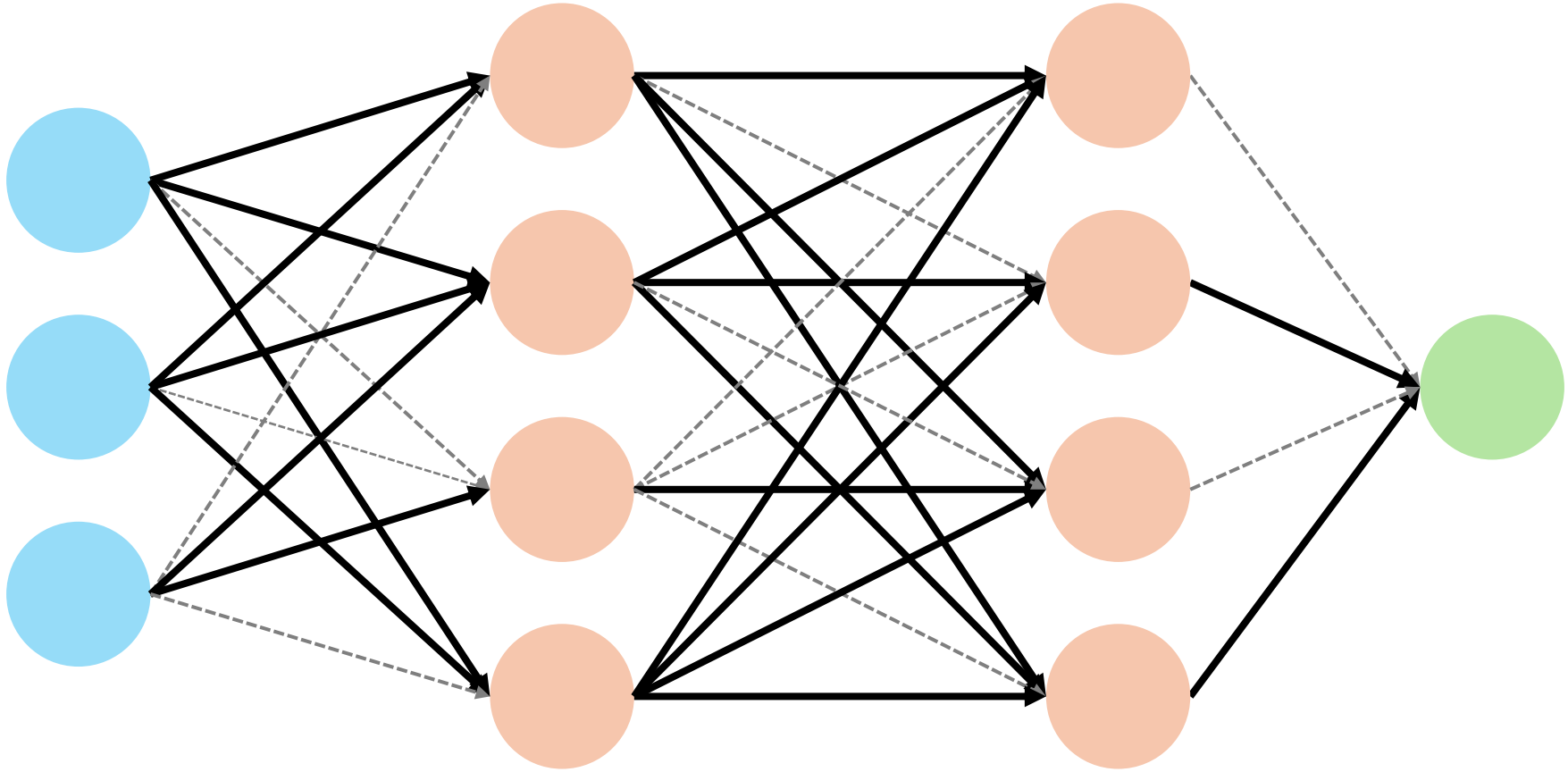
$$z_{l+1} = g \times z_l + b$$

trainable



Convolutional Neural Networks (CNN)

Batch normalization & Dropout



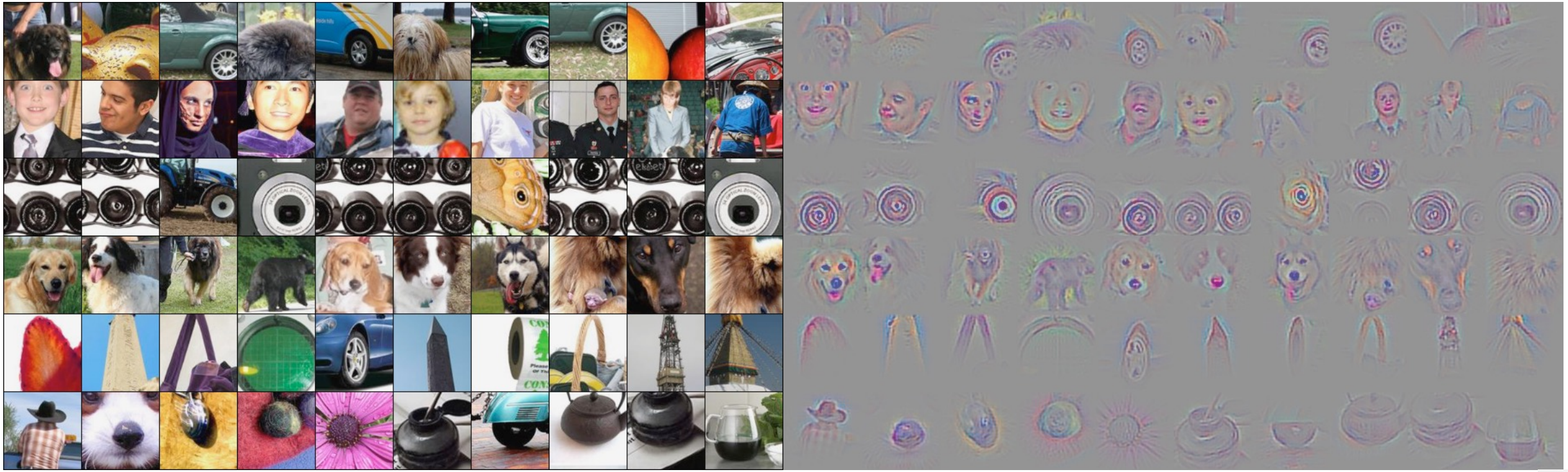
Break

(Q&A)

Convolutional Neural Networks (CNN)

Conv Layer 6

Conv Layer 9



Springenberg, et al, ICLR 2015

Convolutional Neural Networks (CNN)

Overall Implementation

```python
import tensorflow as tf

model = tf.keras.Sequential([
    # first convolutional layer
    tf.keras.layers.Conv2D(32, kernel_size=3, activation='relu'),
    tf.keras.layers.MaxPool2D(pool_size=2, strides=2),

    # second convolutional layer
    tf.keras.layers.Conv2D(64, kernel_size=3, activation='relu'),
    tf.keras.layers.MaxPool2D(pool_size=2, strides=2),

    # fully-connected classifier
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(1024, activation='relu'),
    tf.keras.layers.Dense(10, activation='softmax')
])

model.build(input_shape=(None, 32, 32, 3))
model.summary()
```

Layer (type)	Output Shape	Param #
conv2d_2 (Conv2D)	(None, 30, 30, 32)	896
max_pooling2d_2 (MaxPooling2D)	(None, 15, 15, 32)	0
conv2d_3 (Conv2D)	(None, 13, 13, 64)	18,496
max_pooling2d_3 (MaxPooling2D)	(None, 6, 6, 64)	0
flatten_1 (Flatten)	(None, 2304)	0
dense_2 (Dense)	(None, 1024)	2,360,320
dense_3 (Dense)	(None, 10)	10,250

Total params: 2,389,962 (9.12 MB)

Trainable params: 2,389,962 (9.12 MB)

Non-trainable params: 0 (0.00 B)

Convolutional Neural Networks (CNN)

Overall Implementation

```python
import torch

model = torch.nn.Sequential(
    # first convolutional layer
    torch.nn.Conv2d(in_channels=3, out_channels=32, kernel_size=3),
    torch.nn.ReLU(),
    torch.nn.MaxPool2d(kernel_size=2, stride=2),

    # second convolutional layer
    torch.nn.Conv2d(in_channels=32, out_channels=64, kernel_size=3),
    torch.nn.ReLU(),
    torch.nn.MaxPool2d(kernel_size=2, stride=2),

    # fully-connected classifier
    torch.nn.Flatten(),
    torch.nn.Linear(in_features=64 * 6 * 6, out_features=1024),
    torch.nn.ReLU(),
    torch.nn.Linear(in_features=1024, out_features=10)
    )
input_tensor = torch.randn(1, 3, 32, 32)  # batch size of 1
output = model(input_tensor)
print(model)
print(f"Total parameters: {sum(p.numel() for p in model.parameters()):,}")
print(f"Trainable parameters: {sum(p.numel() for p in model.parameters() if p.requires_grad):,}")
```

```
Sequential(
  (0): Conv2d(3, 32, kernel_size=(3, 3), stride=(1, 1))
  (1): ReLU()
  (2): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (3): Conv2d(32, 64, kernel_size=(3, 3), stride=(1, 1))
  (4): ReLU()
  (5): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (6): Flatten(start_dim=1, end_dim=-1)
  (7): Linear(in_features=2304, out_features=1024, bias=True)
  (8): ReLU()
  (9): Linear(in_features=1024, out_features=10, bias=True)
)
Total parameters: 2,389,962
Trainable parameters: 2,389,962
```

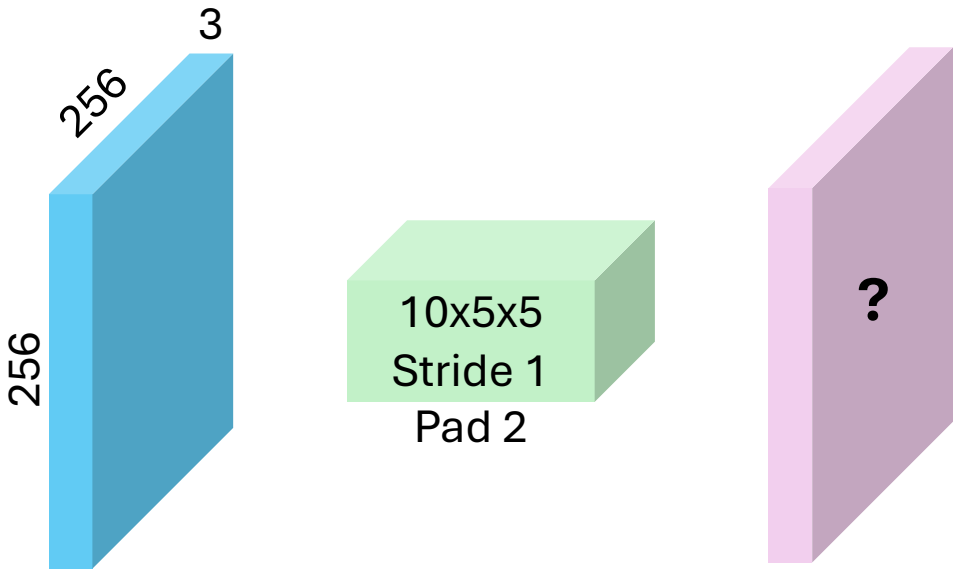
Convolutional Neural Networks (CNN)

Quiz time!

Input tensor: 256 x 256 x 3

Conv kernels: 10 x 5 x 5
Stride 1 and padding 2

What would be the output tensor size?
How many trainable parameters?



Convolutional Neural Networks (CNN)

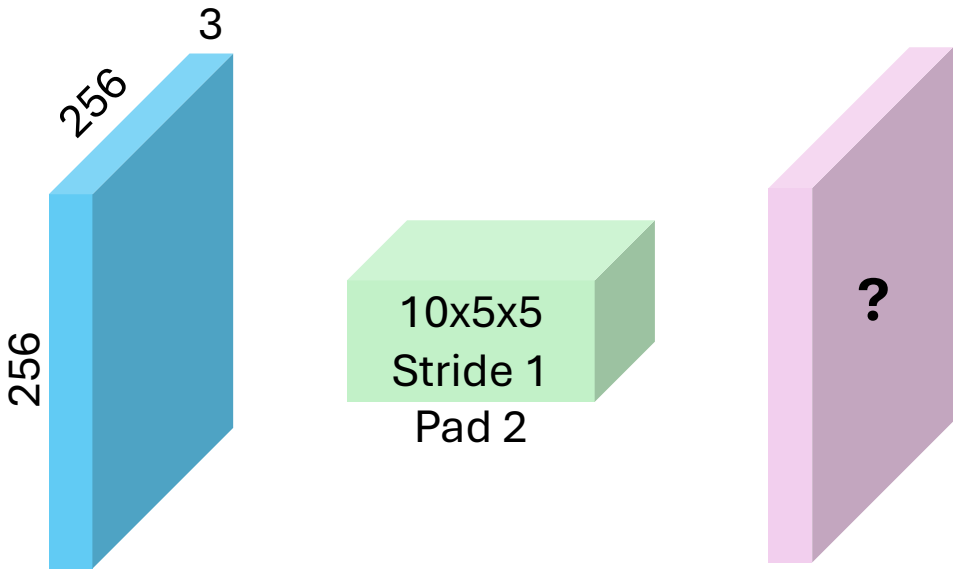
Quiz time!

Input tensor: 256 x 256 x 3

Conv kernels: 10 x 5 x 5
Stride 1 and padding 2

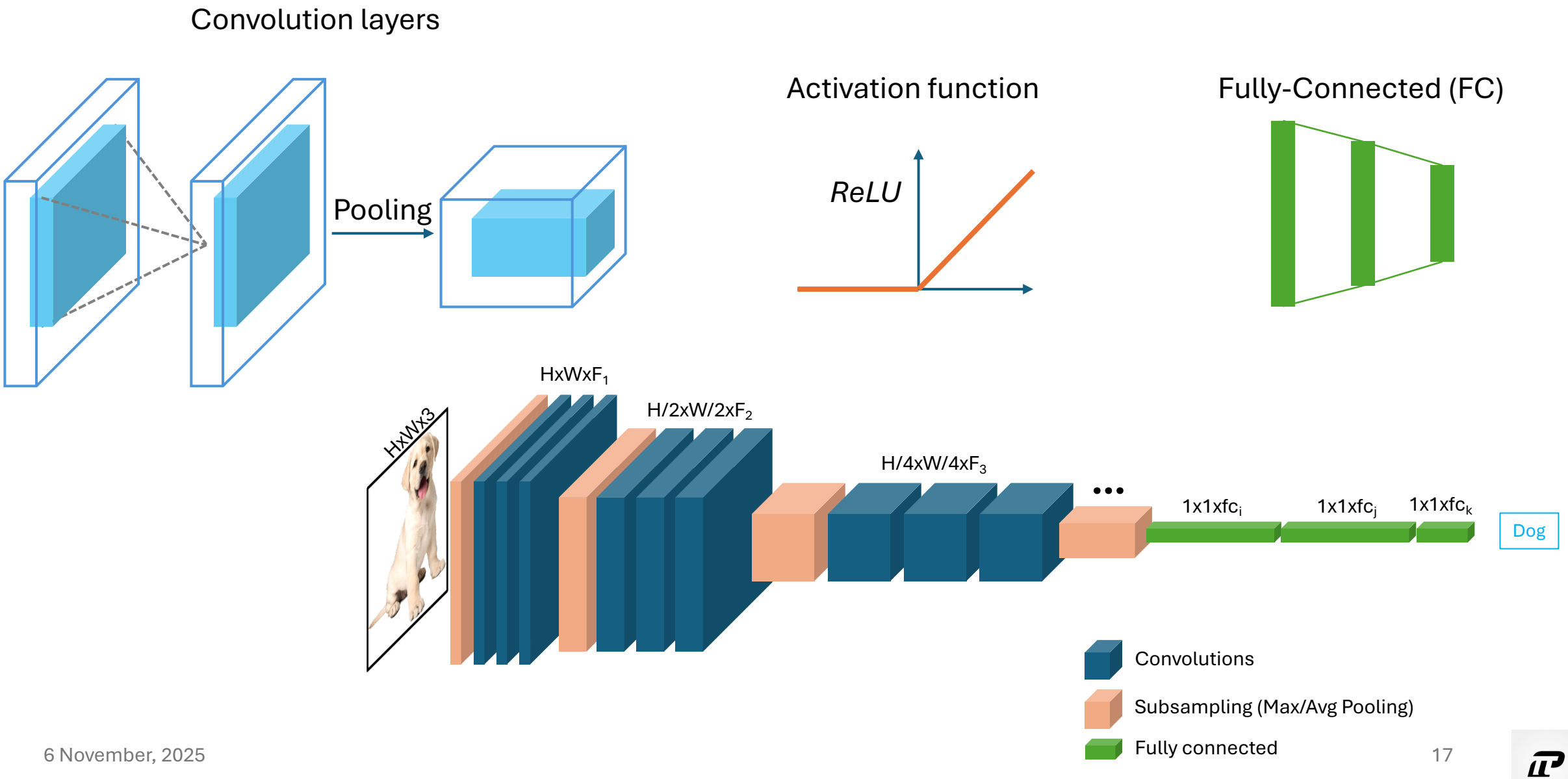
What would be the output tensor size?
(256+2x2-5)/1+1=256 ⇒ 10 x 256 x 256

How many trainable parameters?
Every filter: 3x5x5+1(bias)=76
10x76=760 trainable parameters



Convolutional Neural Networks (CNN)

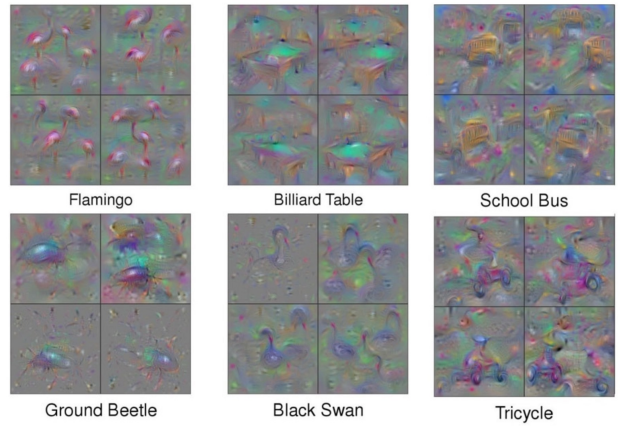
Summary:



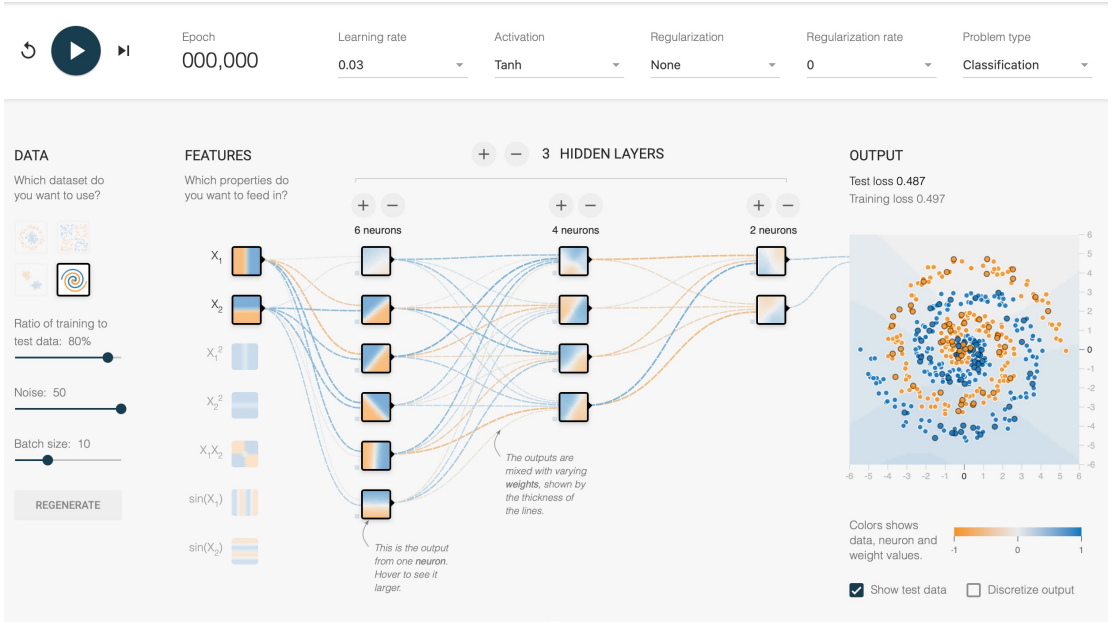
Convolutional Neural Networks (CNN)

Playgrounds:

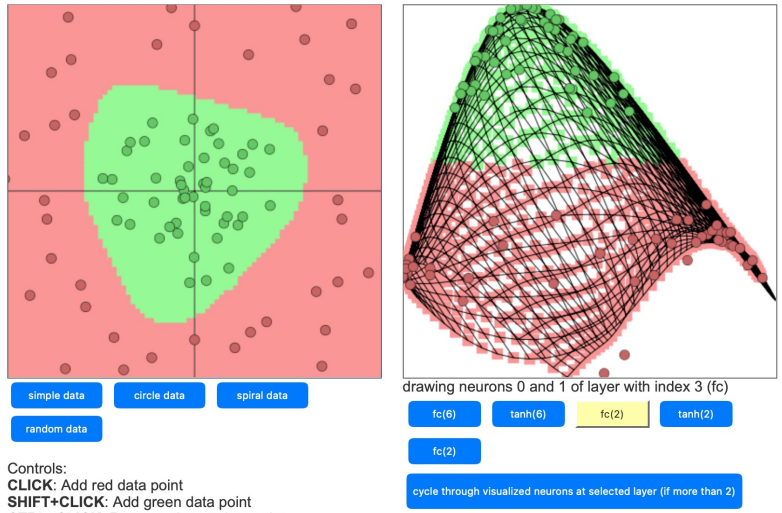
Understanding Neural Networks Through Deep Visualization
Yosinski, et al, ICML 2015 DL Workshop
https://yosinski.com/deepvis



playground.tensorflow.org



ConvNetJS



(Q&A)

Parham Kebria
parhamkebria@ieee.org