



# Convolutional Neural Networks (CNN)

**Prof. Seungchul Lee**  
**Industrial AI Lab.**

# Machine Learning vs. Deep Learning

- Machine learning



- Deep learning

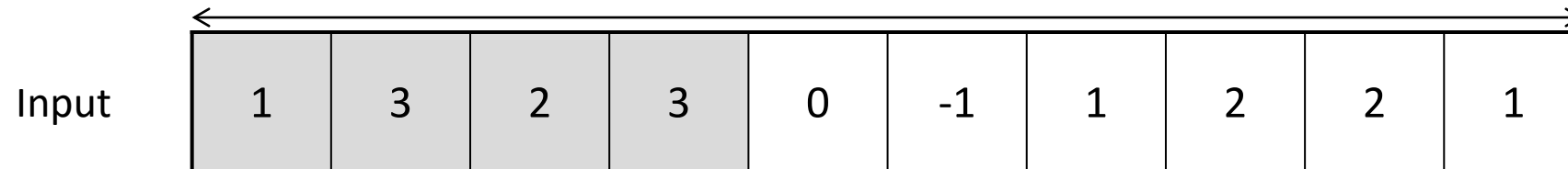


end-to-end learning

# Convolution

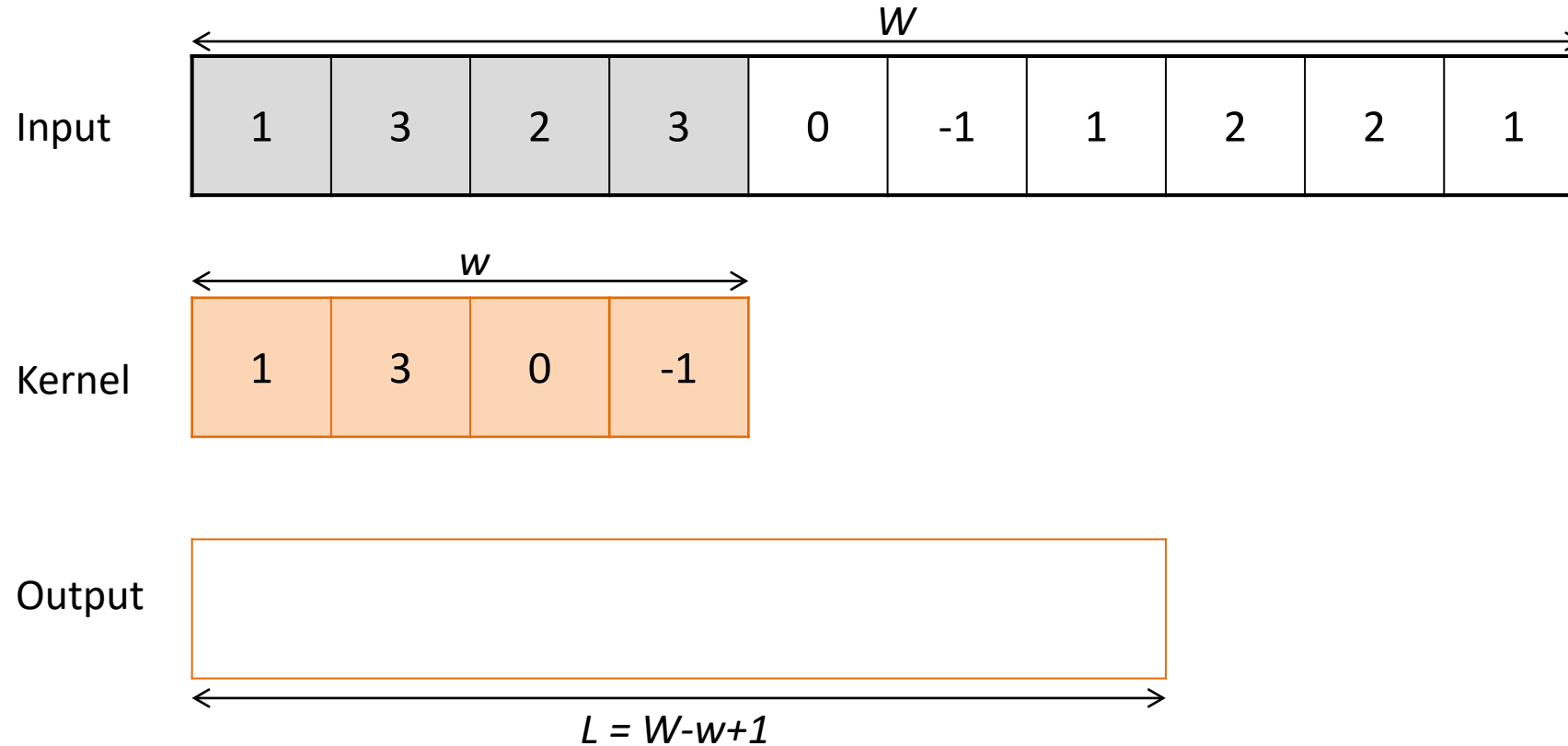
# 1D Convolution

- (actually cross-correlation)



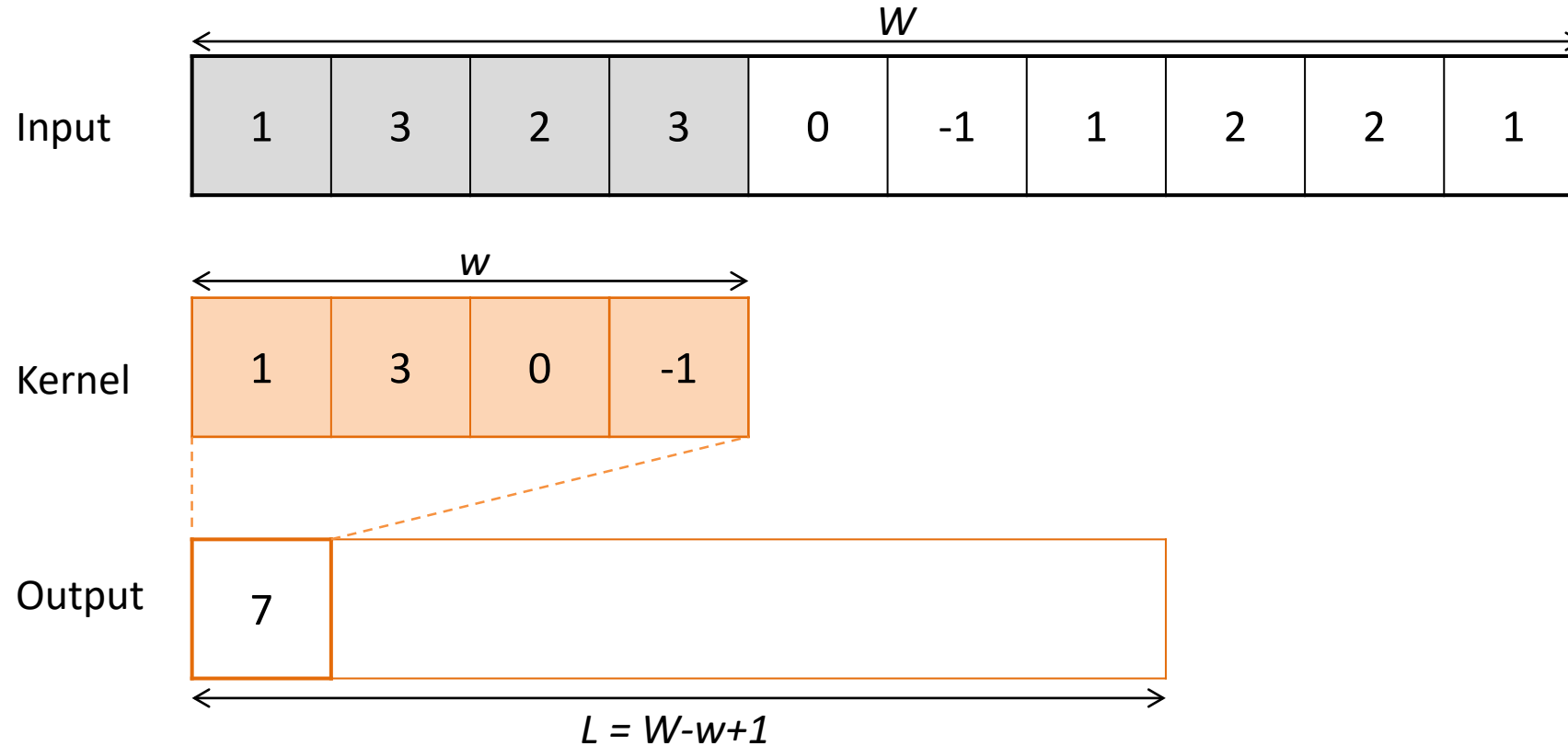
# 1D Convolution

- (actually cross-correlation)



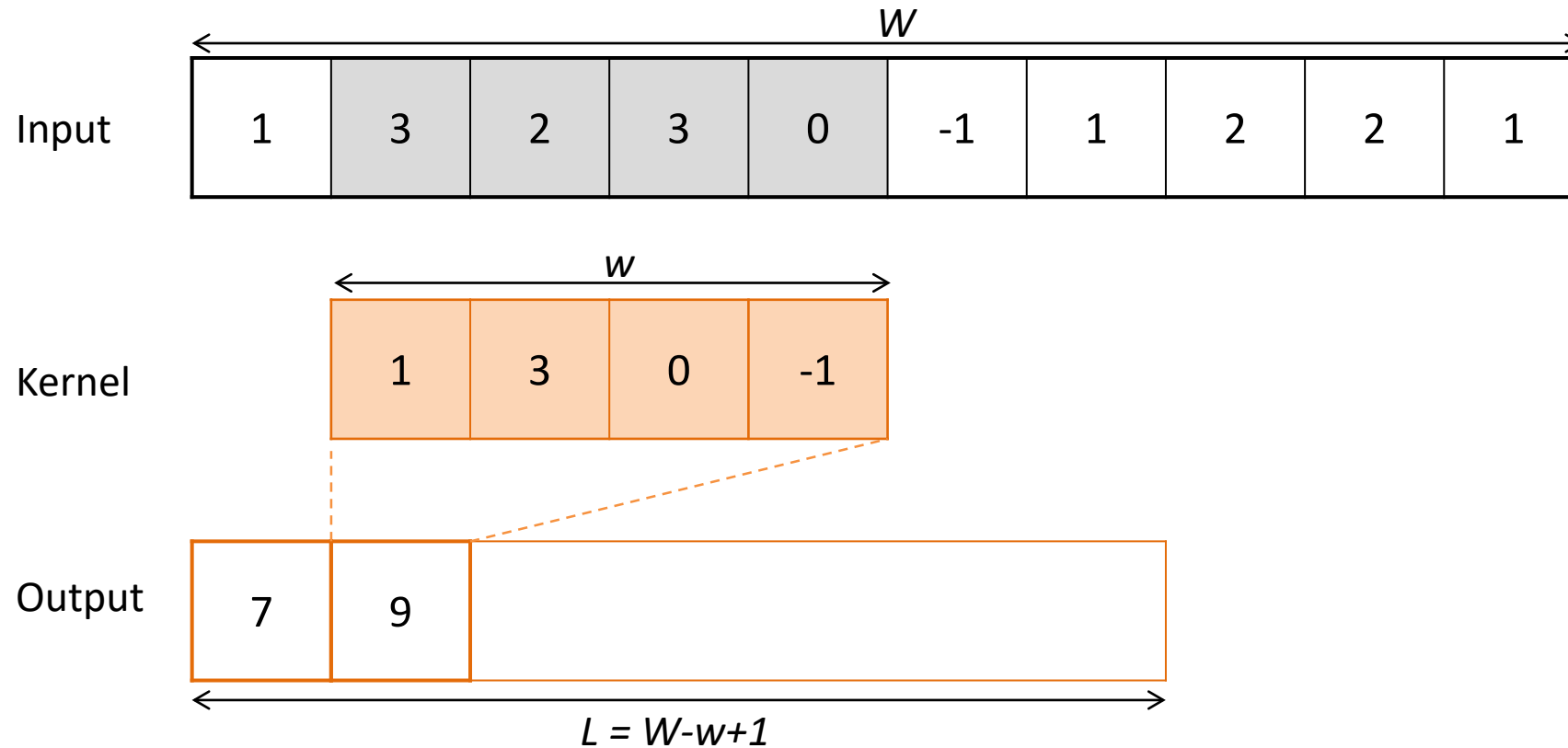
# 1D Convolution

- (actually cross-correlation)



# 1D Convolution

- (actually cross-correlation)



# 2D Convolution



# Convolution on Image (= Convolution in 2D)

- Filter (or Kernel)

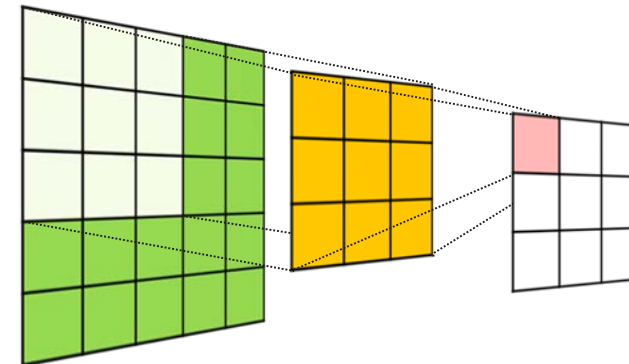
- Discrete convolution can be viewed as element-wise multiplication by a matrix
- Modify or enhance an image by filtering
- Filter images to emphasize certain features or remove other features
- Filtering includes smoothing, sharpening and edge enhancement

1 <sub>x1</sub>	1 <sub>x0</sub>	1 <sub>x1</sub>	0	0
0 <sub>x0</sub>	1 <sub>x1</sub>	1 <sub>x0</sub>	1	0
0 <sub>x1</sub>	0 <sub>x0</sub>	1 <sub>x1</sub>	1	1
0	0	1	1	0
0	1	1	0	0

Image

4		

Convolved  
Feature

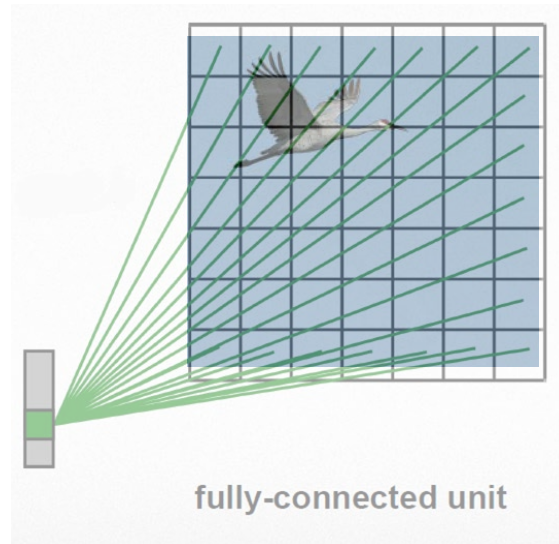


Image

Kernel

Output

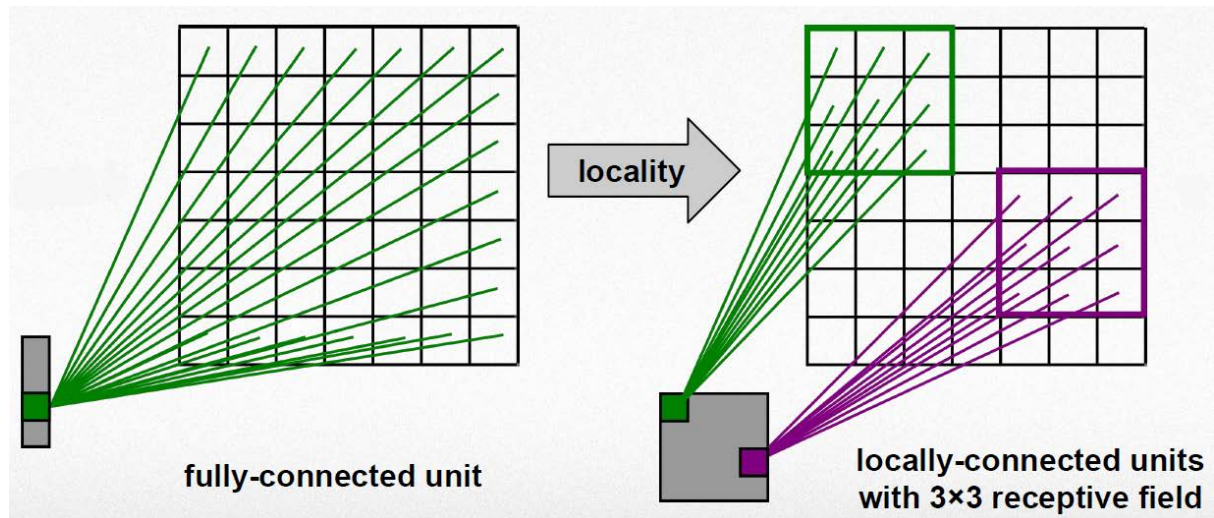
# Convolution Mask + Neural Network



# Locality



- Locality: objects tend to have a local spatial support
  - fully-connected layer  $\rightarrow$  locally-connected layer

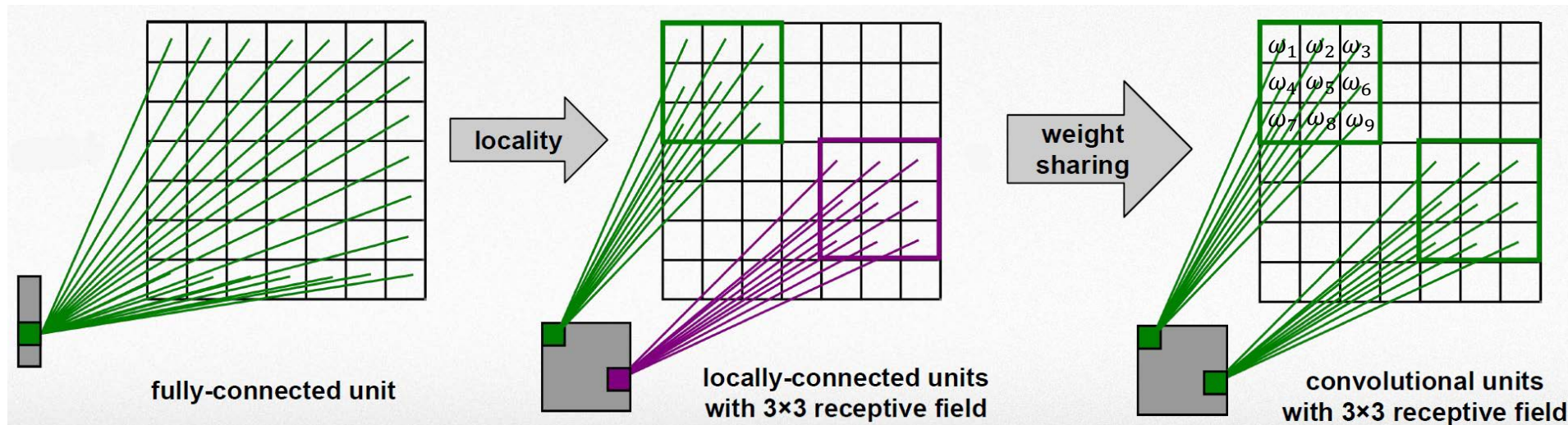


# Locality

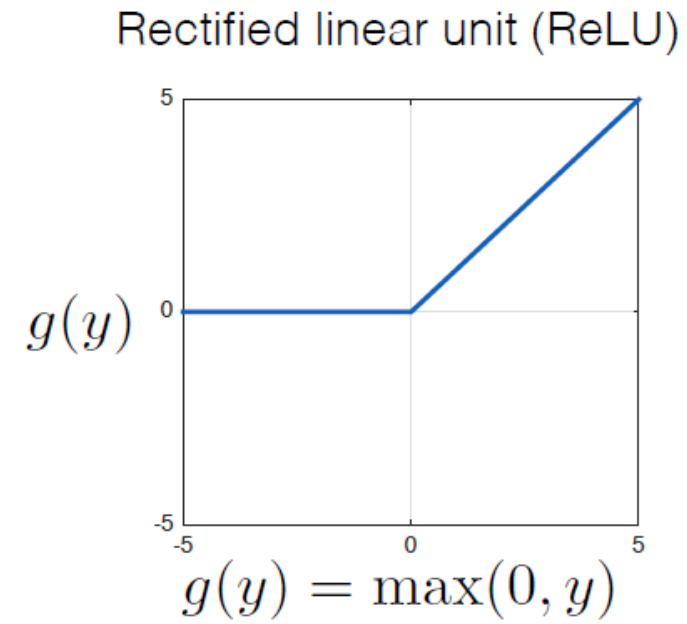
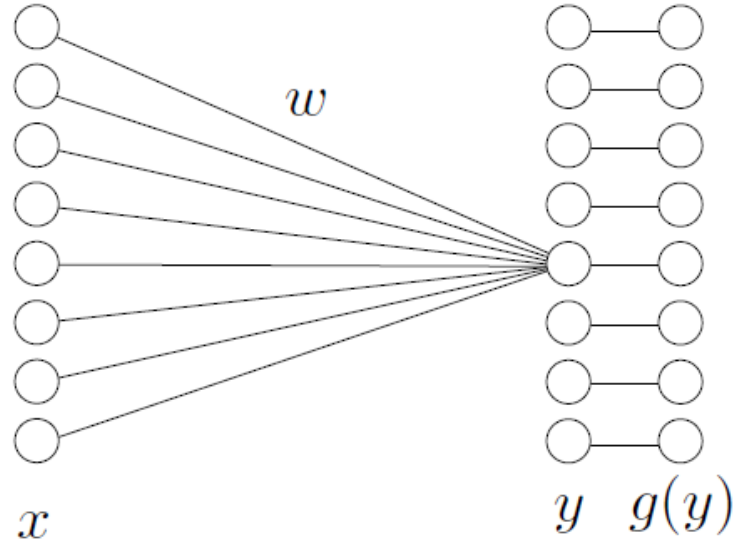


- Locality: objects tend to have a local spatial support
  - fully-connected layer  $\rightarrow$  locally-connected layer

We are not designing the kernel, but are learning the kernel from data  
 $\rightarrow$  Learning feature extractor from data



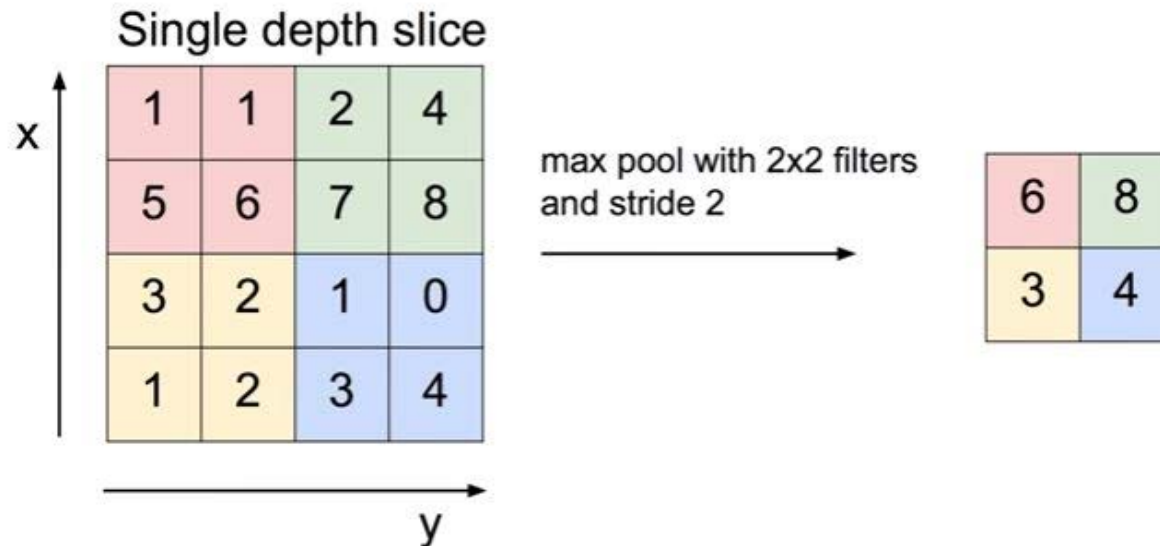
# Nonlinear Activation Function



# Pooling

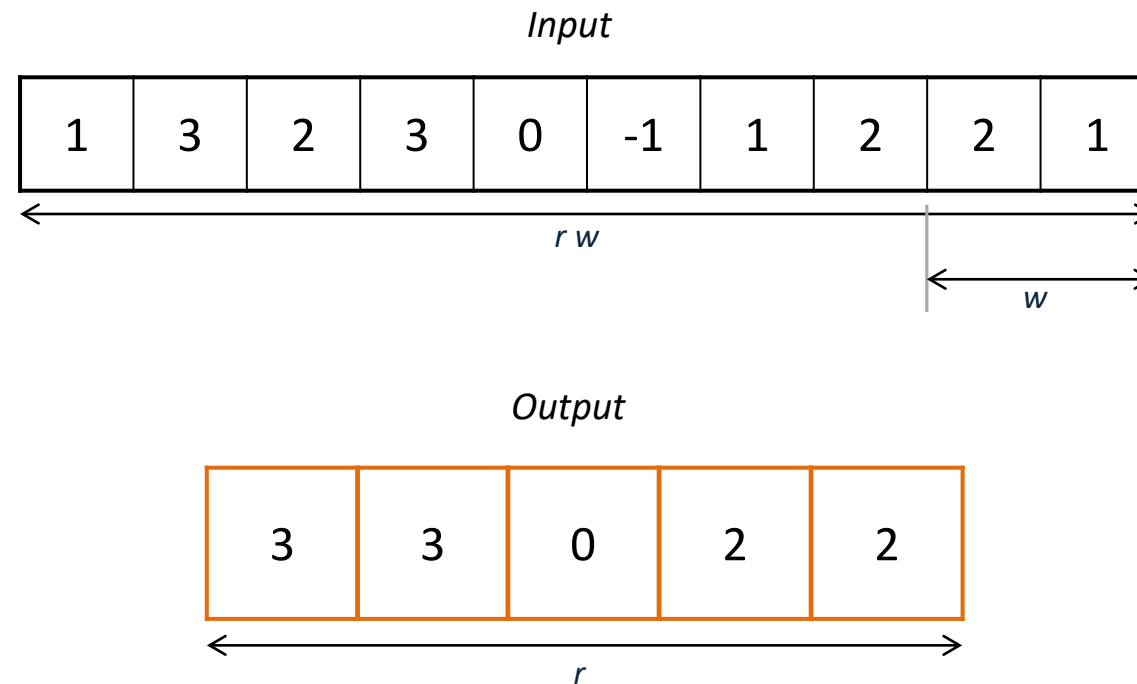
# Pooling

- Compute a maximum value in a sliding window (max pooling)
- Reduce spatial resolution for faster computation
- Achieve invariance to local translation
- Max pooling introduces invariances
  - Pooling size :  $2 \times 2$
  - No parameters: max or average of  $2 \times 2$  units



# Pooling

- Such an operation aims at grouping several activations into a single “more meaningful” one.

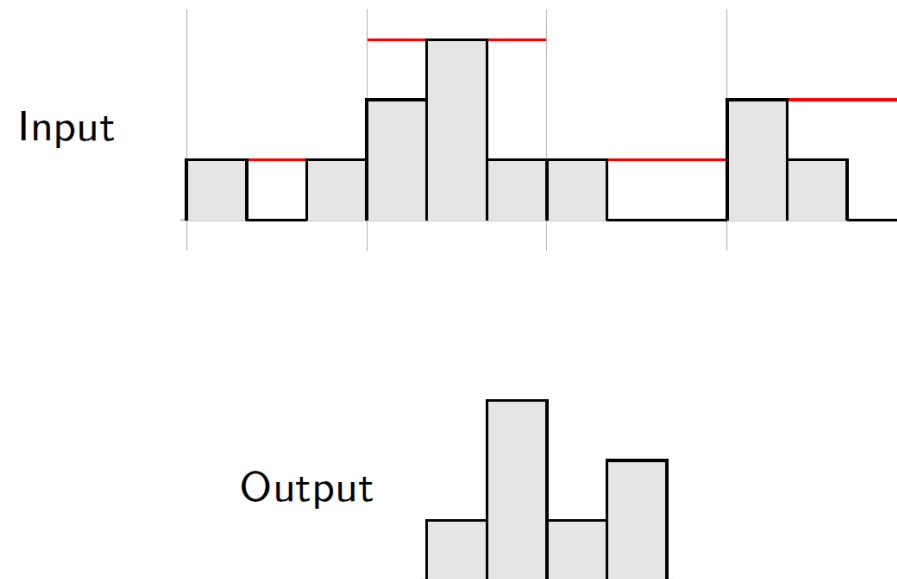


- The average pooling computes average values per block instead of max values



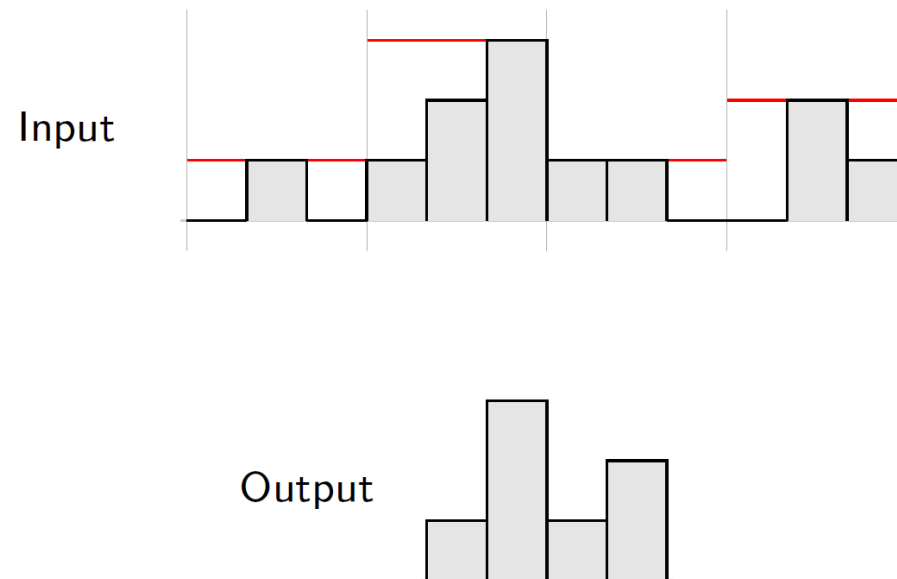
# Pooling: Invariance

- Pooling provides invariance to any permutation inside one of the cell
- More practically, it provides a pseudo-invariance to deformations that result into local translations



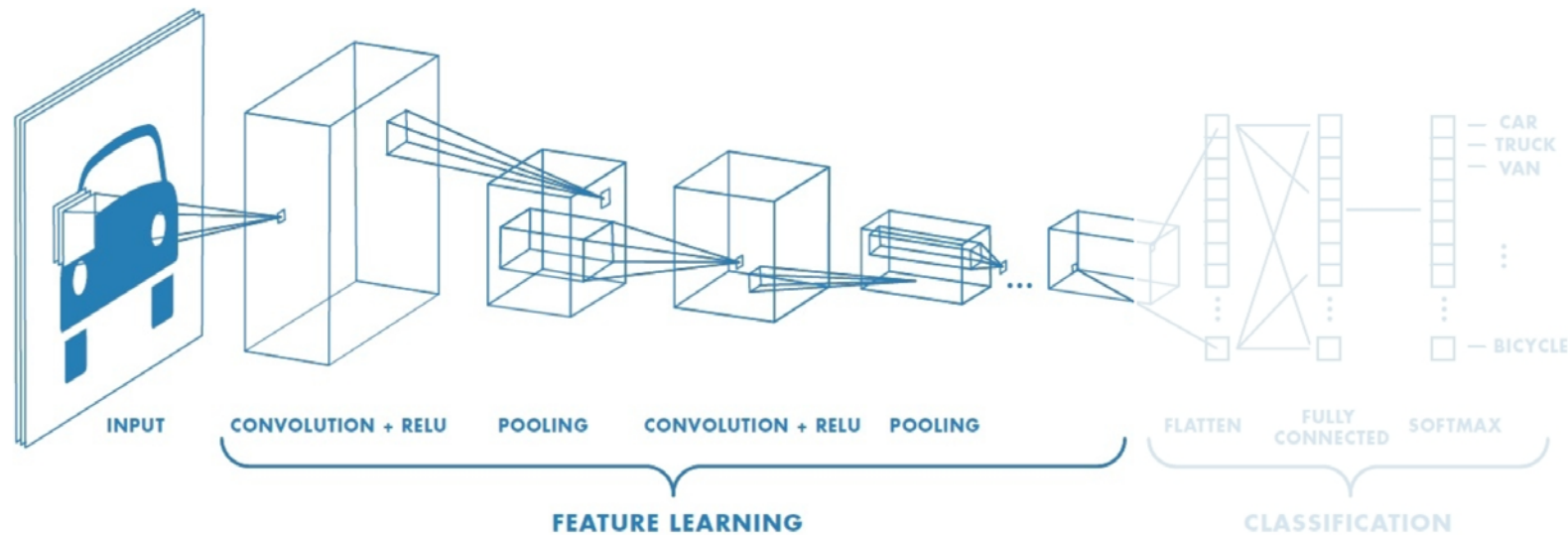
# Pooling: Invariance

- Pooling provides invariance to any permutation inside one of the cell
- More practically, it provides a pseudo-invariance to deformations that result into local translations



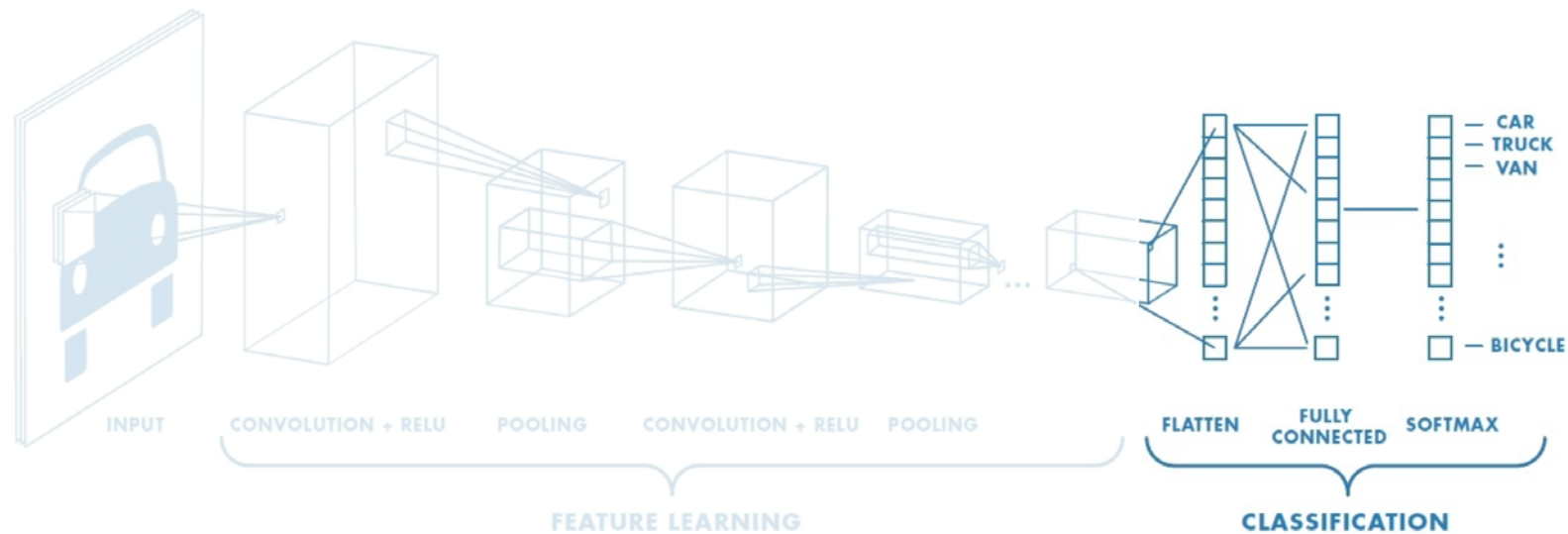
# CNNs for Classification: Feature Learning

- Learn features in input image through convolution
- Introduce non-linearity through activation function (real-world data is non-linear!)
- Reduce dimensionality and preserve spatial invariance with pooling



# CNNs for Classification: Class Probabilities

- CONV and POOL layers output high-level features of input
- Fully connected layer uses these features for classifying input image
- Express output as probability of image belonging to a particular class

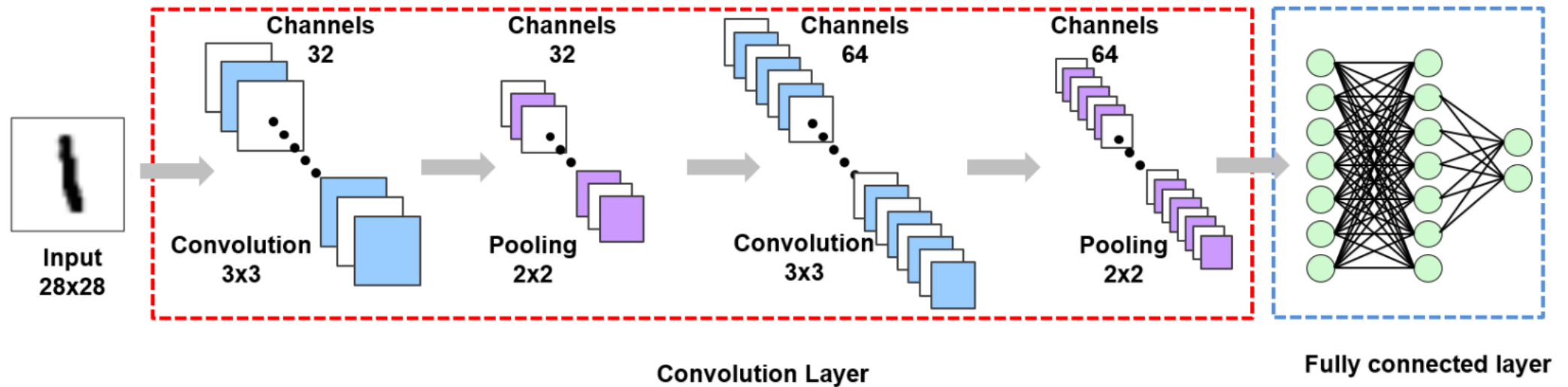


$$\text{softmax}(y_i) = \frac{e^{y_i}}{\sum_j e^{y_j}}$$

# CNN in TensorFlow

# Lab: CNN with TensorFlow

- MNIST example
- To classify handwritten digits



# CNN Structure

```
model = tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(32,
                           (3,3),
                           activation = 'relu',
                           padding = 'SAME',
                           input_shape = (28, 28, 1)),

    tf.keras.layers.MaxPool2D((2,2)),

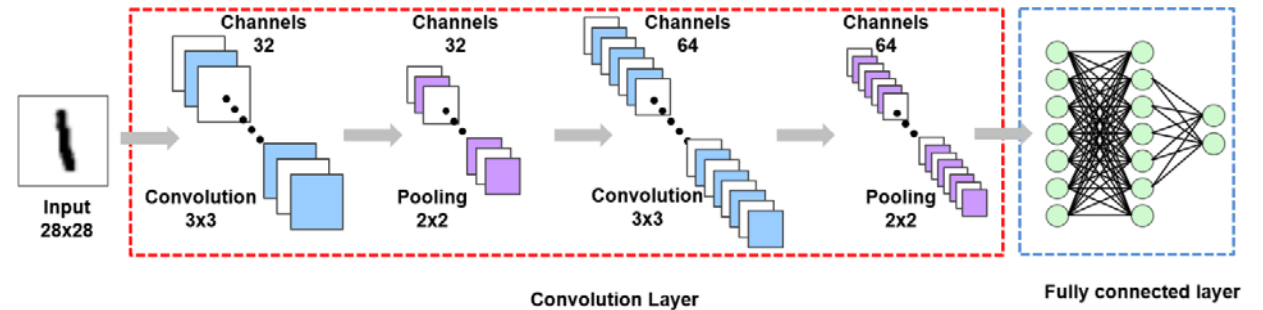
    tf.keras.layers.Conv2D(64,
                           (3,3),
                           activation = 'relu',
                           padding = 'SAME',
                           input_shape = (14, 14, 32)),

    tf.keras.layers.MaxPool2D((2,2)),

    tf.keras.layers.Flatten(),

    tf.keras.layers.Dense(128, activation = 'relu'),

    tf.keras.layers.Dense(10, activation = 'softmax')
])
```



# Loss and Optimizer

- Loss
  - Classification: Cross entropy
  - Equivalent to applying logistic regression
- Optimizer
  - GradientDescentOptimizer
  - AdamOptimizer: the most popular optimizer

```
model.compile(optimizer = 'adam',  
              loss = 'sparse_categorical_crossentropy',  
              metrics = ['accuracy'])
```

```
model.fit(train_x, train_y)
```



# Test or Evaluation

```
test_loss, test_acc = model.evaluate(test_x, test_y)
```

313/313 [=====] - 1s 4ms/step - accuracy: 0.9838 - loss: 0.0466  
loss = 0.05, Accuracy = 98 %



Prediction : 9

