진동, 열, 음향신호 분석을 위한 합성곱 신경망 (CNN)

Prof. Seungchul Lee
Industrial AI Lab.

Machine Learning vs. Deep Learning

- Machine learning

Input → Hand-engineered features → Classifier → output

- Deep learning

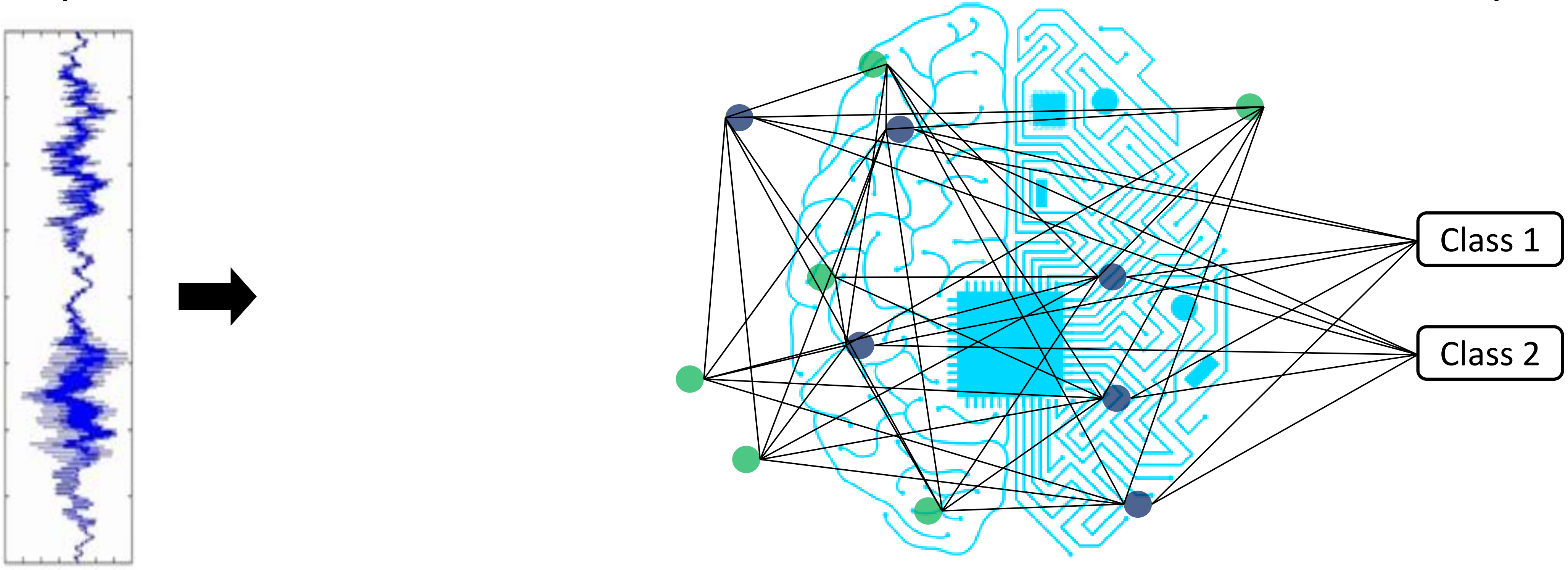
Input → **Deep Learning** → output

end-to-end learning

Artificial Neural Networks

- Complex/Nonlinear function approximator
  - Linearly connected networks
  - Simple nonlinear neurons

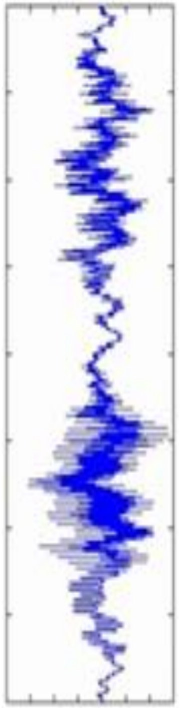
Input **Output**



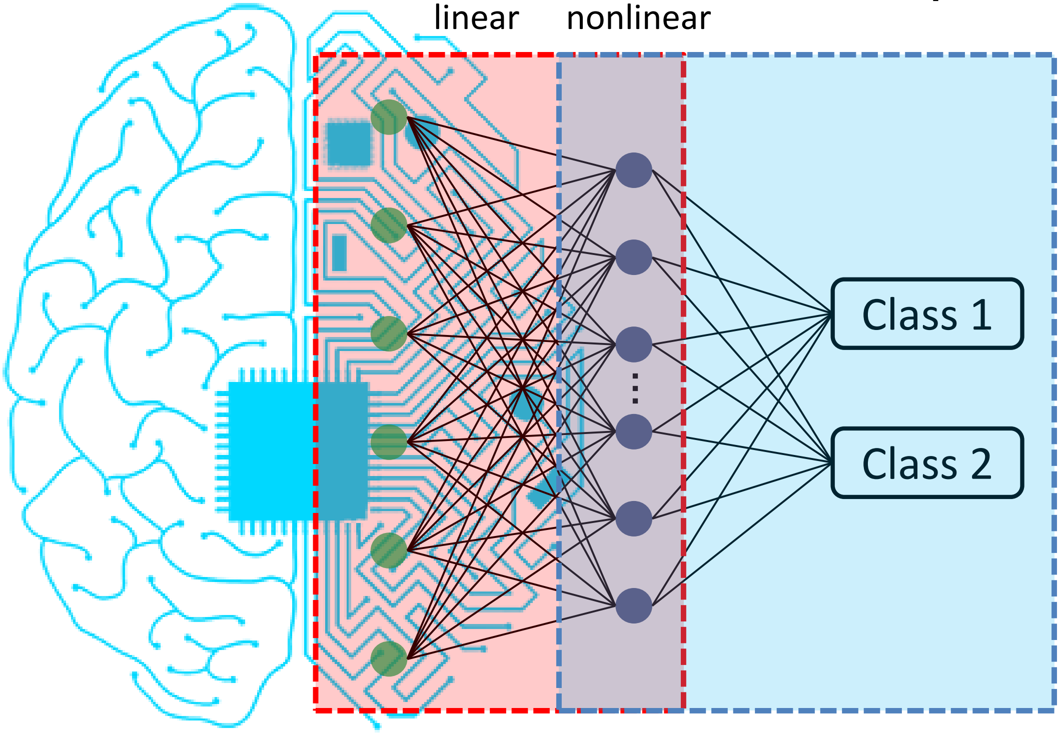
Artificial Neural Networks

- Complex/Nonlinear function approximator
  - Linearly connected networks
  - Simple nonlinear neurons

Input



Output

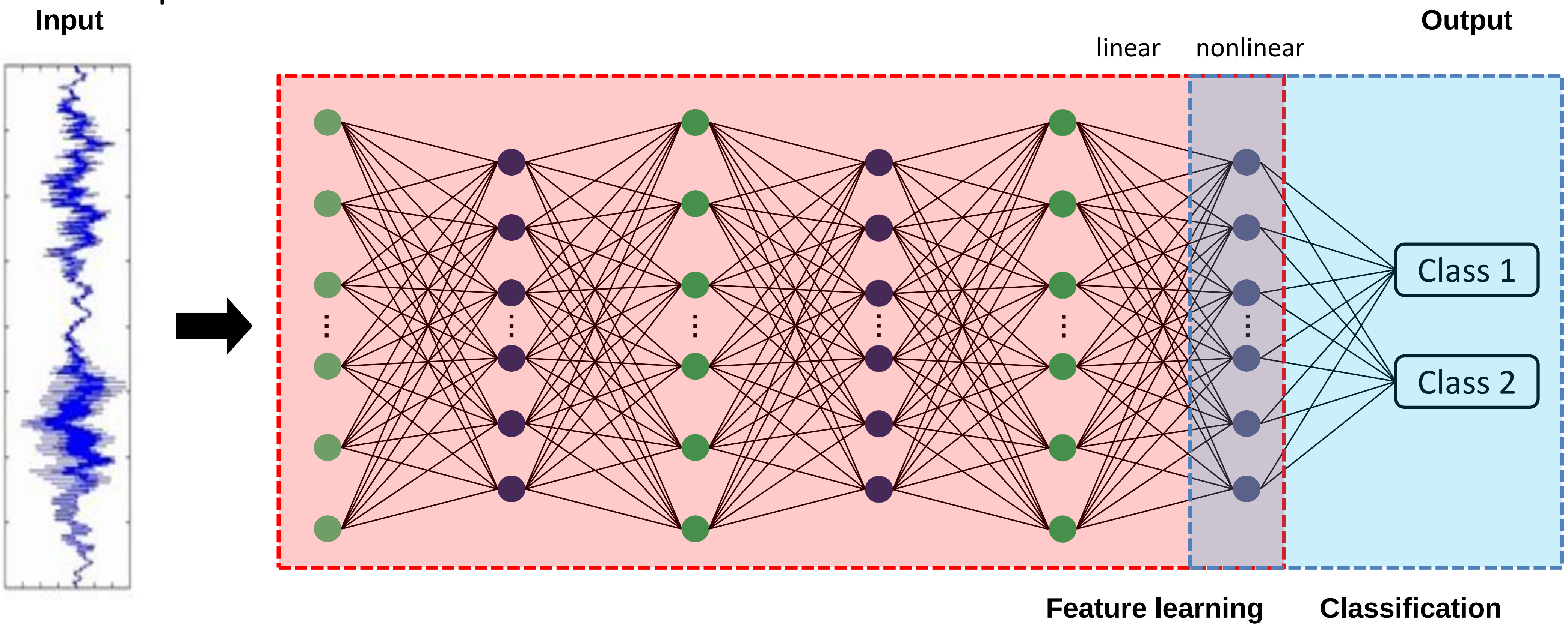


Feature learning

Classification

Deep Artificial Neural Networks

- Complex/Nonlinear function approximator
  - Linearly connected networks
  - Simple nonlinear neurons



Convolution

1D Convolution

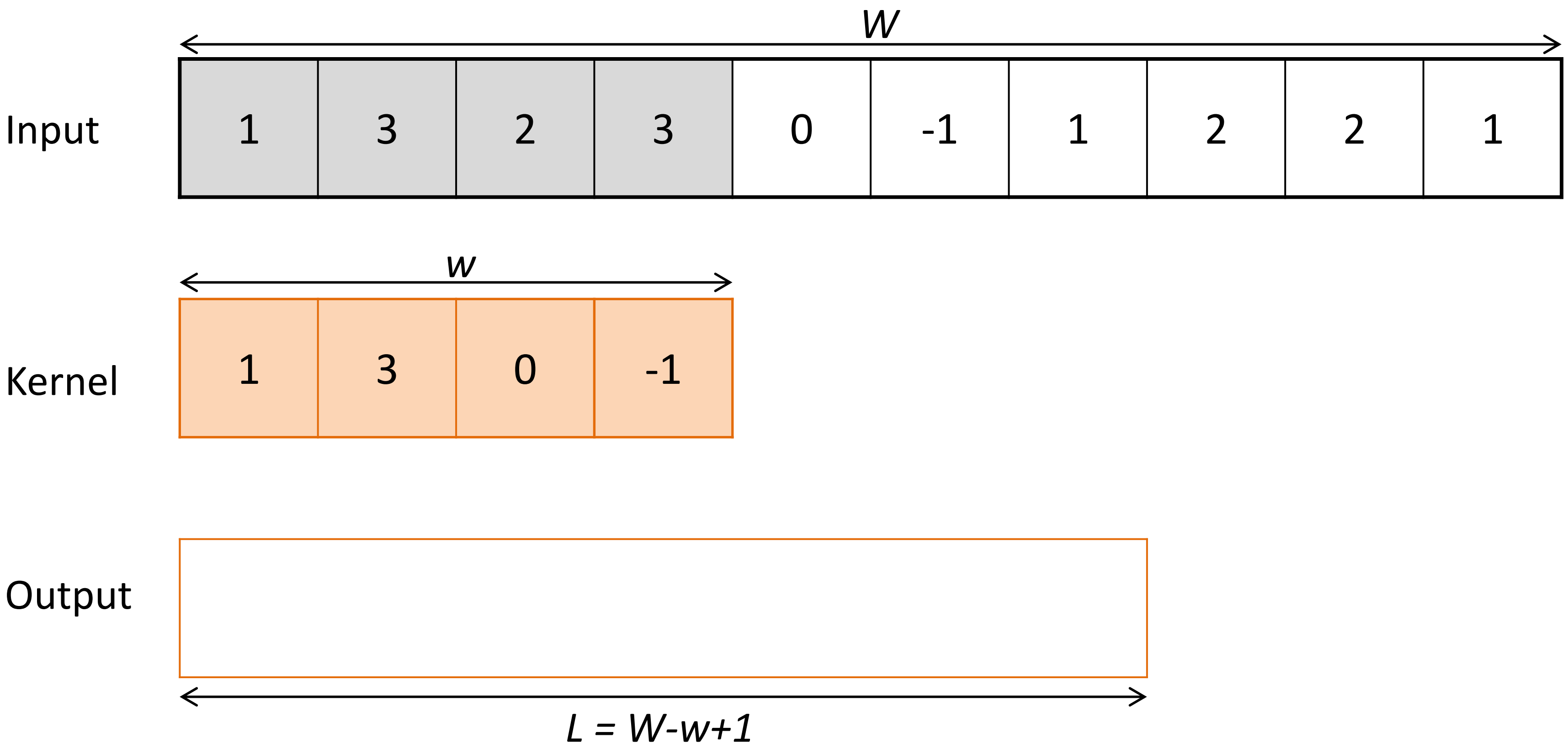
- (actually cross-correlation)

Input	1	3	2	3	0	-1	1	2	2	1

Source: Dr. Francois Fleuret at EPFL

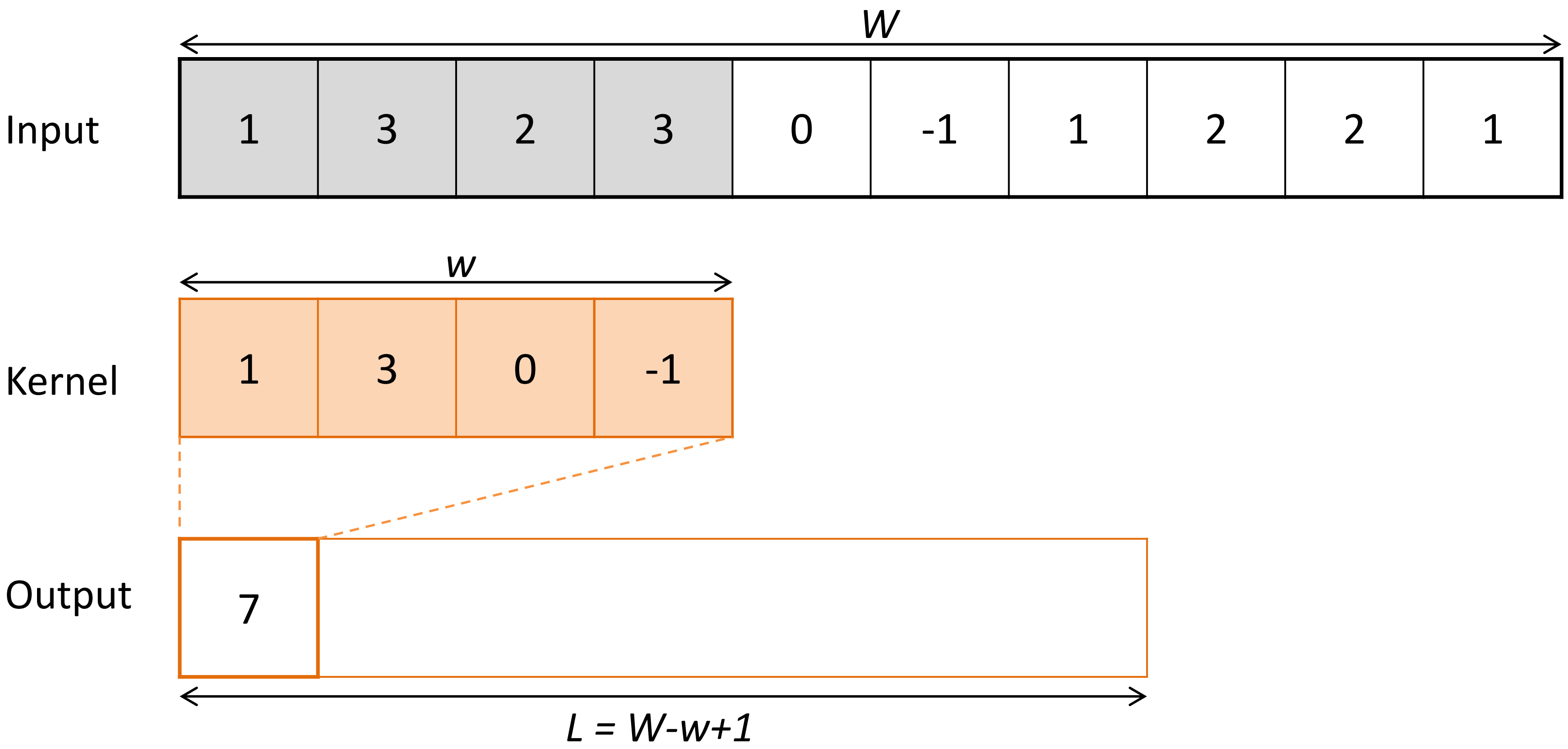
1D Convolution

- (actually cross-correlation)



1D Convolution

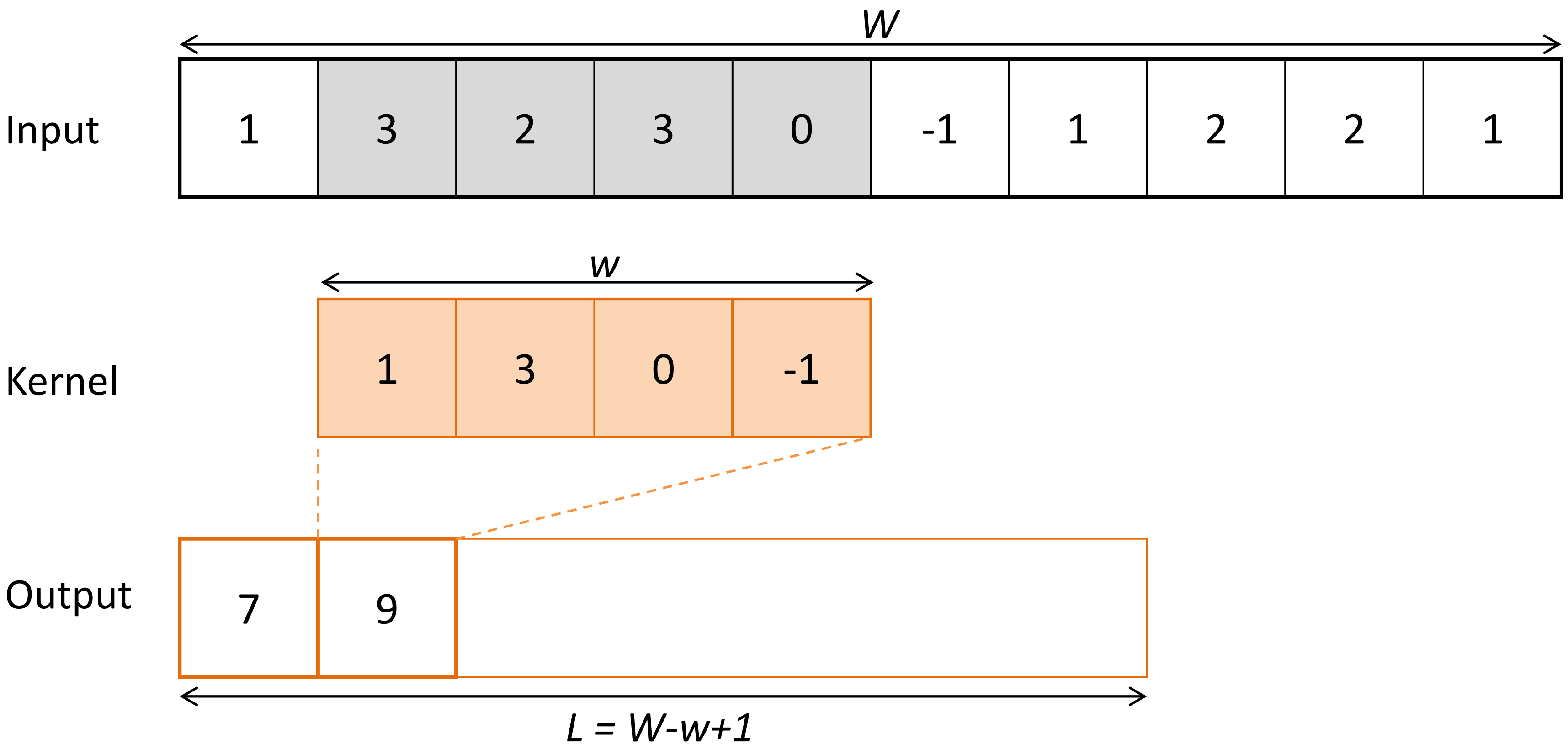
- (actually cross-correlation)



Source: Dr. Francois Fleuret at EPFL

1D Convolution

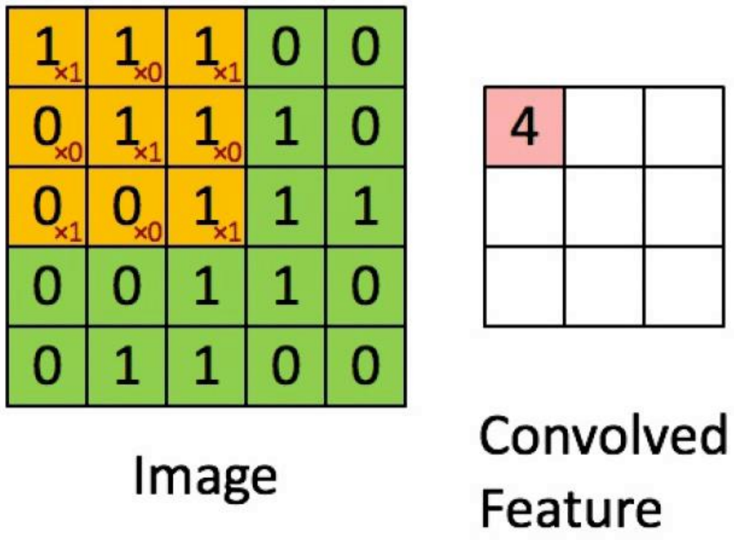
- (actually cross-correlation)

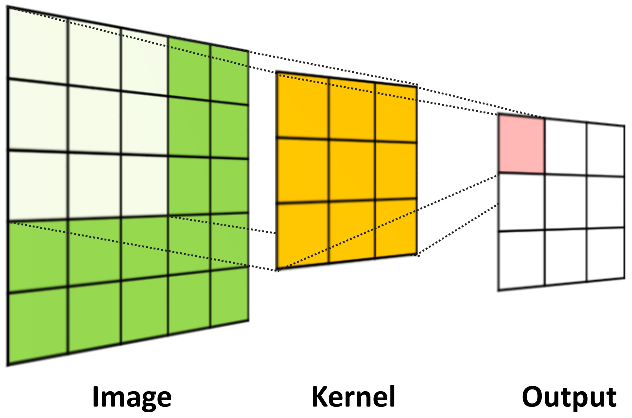


2D Convolution

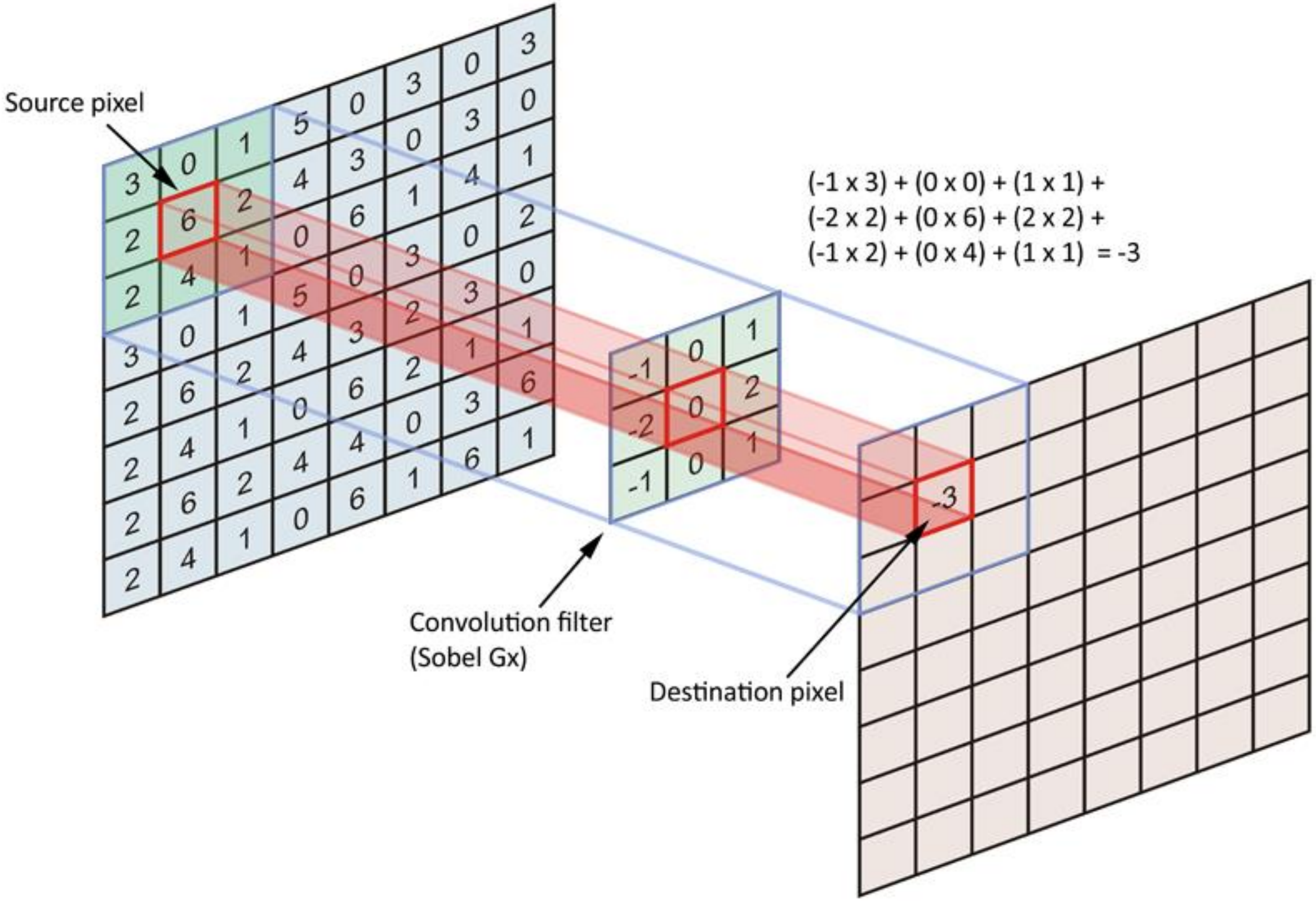
Convolution on Image (= Convolution in 2D)

- 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





Convolution on Image (= Convolution in 2D)

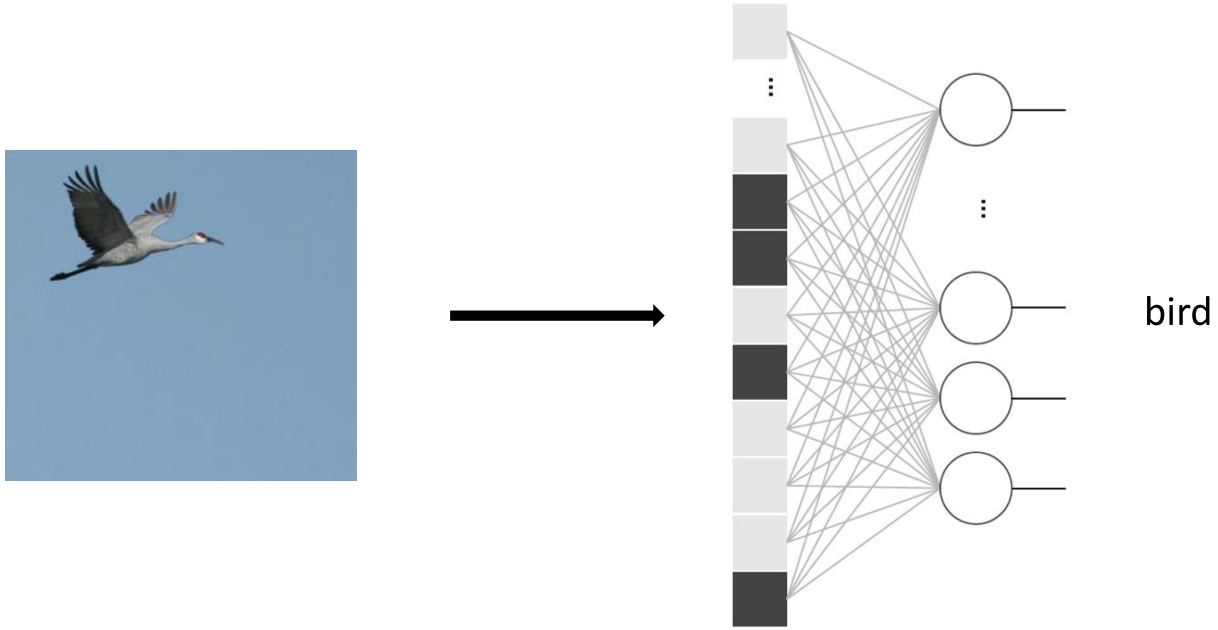


How to Find the Right Kernels

- We learn many different kernels that make specific effect on images
- Let’s apply an opposite approach
- We are not designing the kernel, but are learning the kernel from data
- Can learn feature extractor from data using a deep learning framework

Learning Visual Features

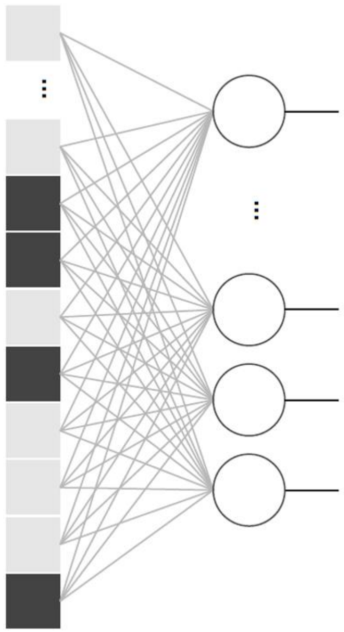
ANN Structure for Object Detection in Image



- Does not seem the best
- Did not make use of the fact that we are dealing with images

Fully Connected Neural Network

- Input
  - 2D image
  - Vector of pixel values

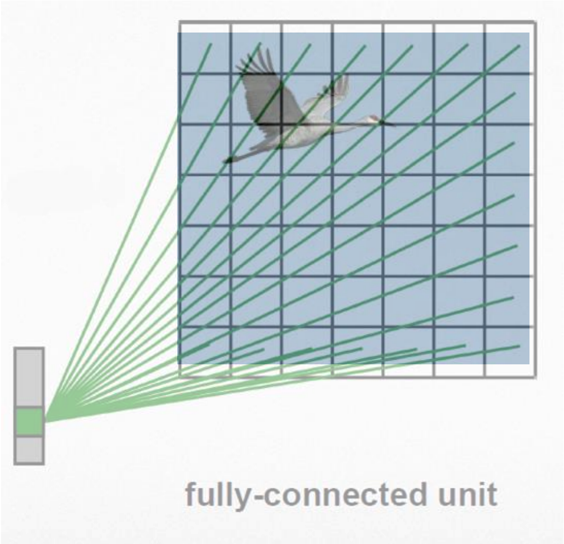


- Fully connected
  - Connect neuron in hidden layer to all neurons in input layer
  - No spatial information
  - Spatial organization of the input is destroyed by flatten
  - And many, many parameters !

- How can we use spatial structure in the input to inform the architecture of the network?

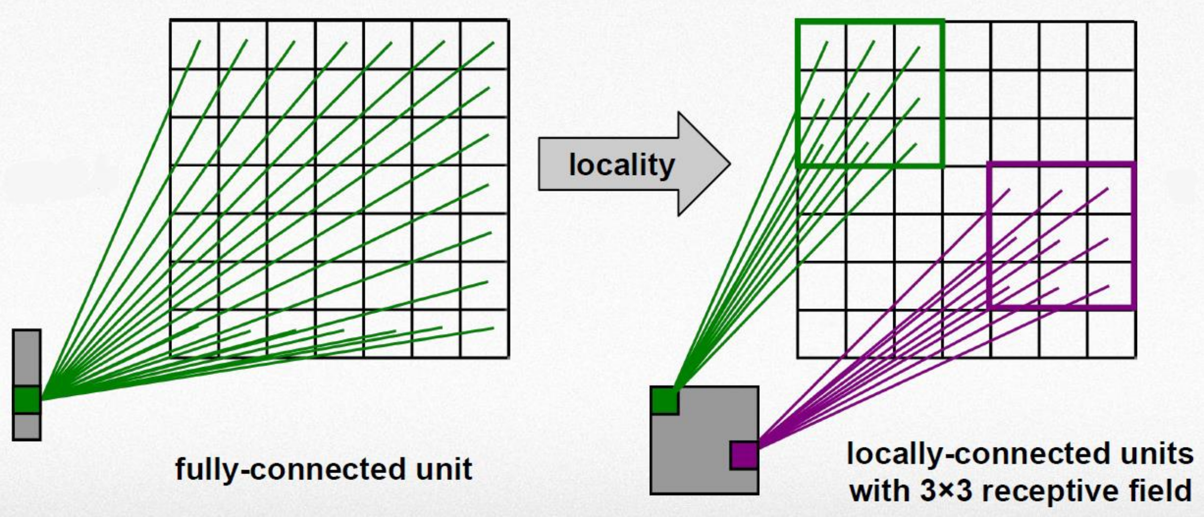
Source: 6.S191 Intro. to Deep Learning at MIT

Convolution Mask + Neural Network



Locality

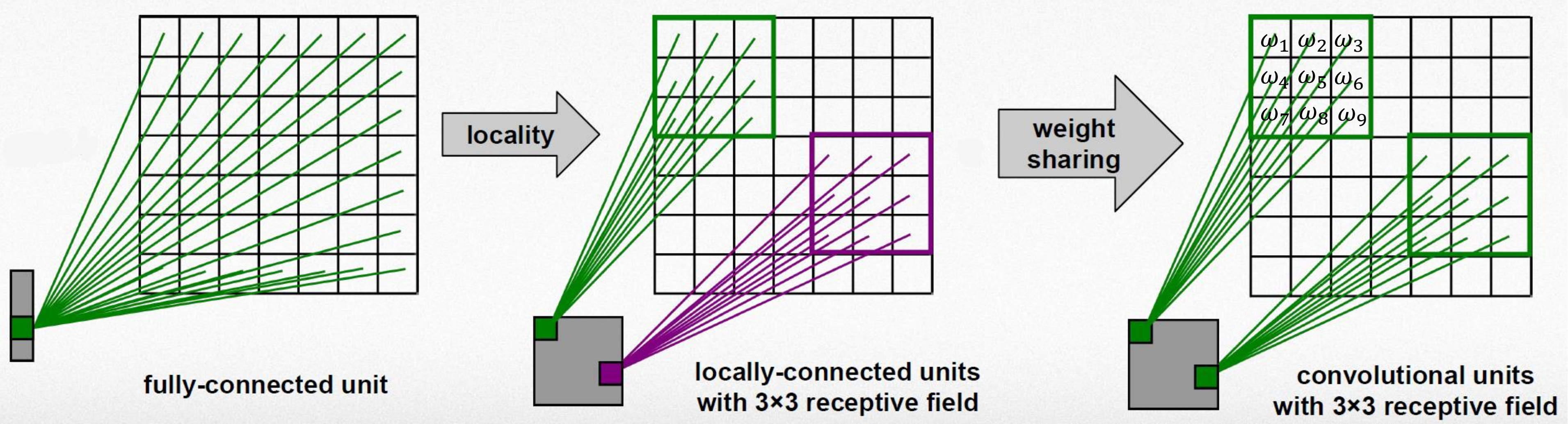
- Locality: objects tend to have a local spatial support
  - fully-connected layer → locally-connected layer



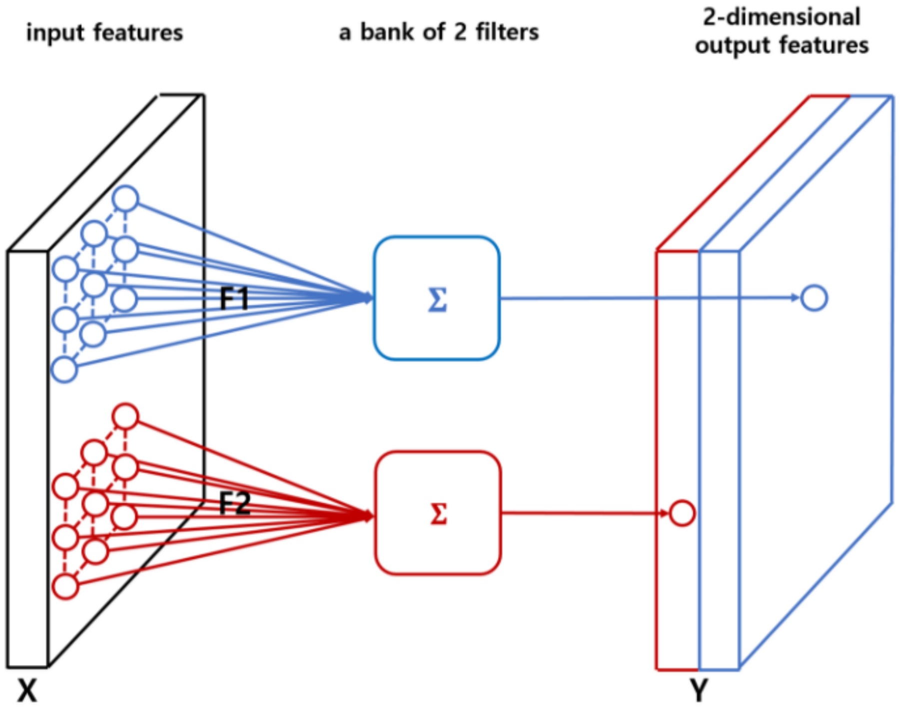
Locality

- Locality: objects tend to have a local spatial support
  - fully-connected layer → locally-connected layer

We are not designing the kernel, but are learning the kernel from data
→ Learning feature extractor from data



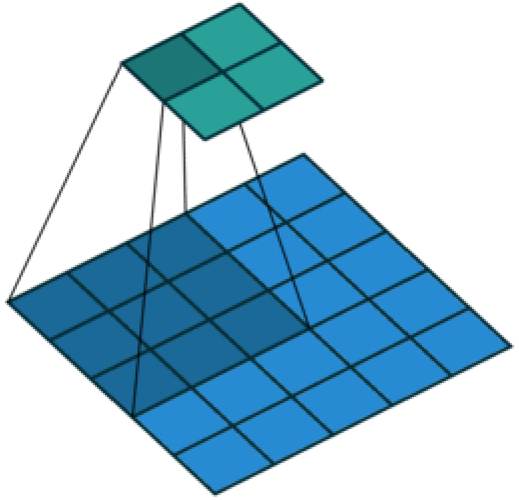
Multiple Filters (or Kernels)



Padding and Stride

Strides

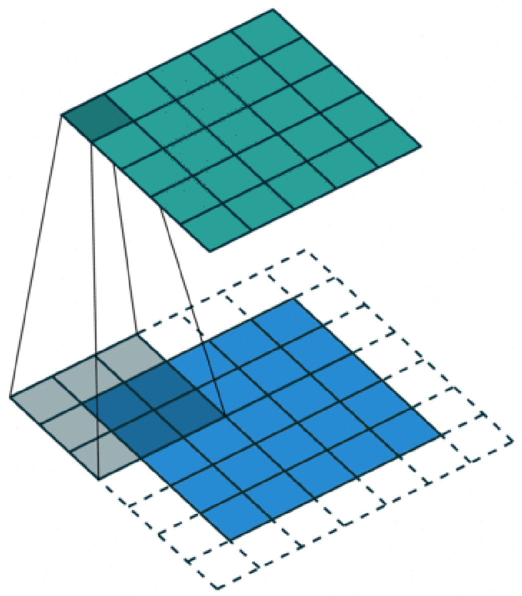
- Strides: increment step size for the convolution operator
- Reduces the size of the output map



Example with kernel size 3×3 and a stride of 2 (image in blue)

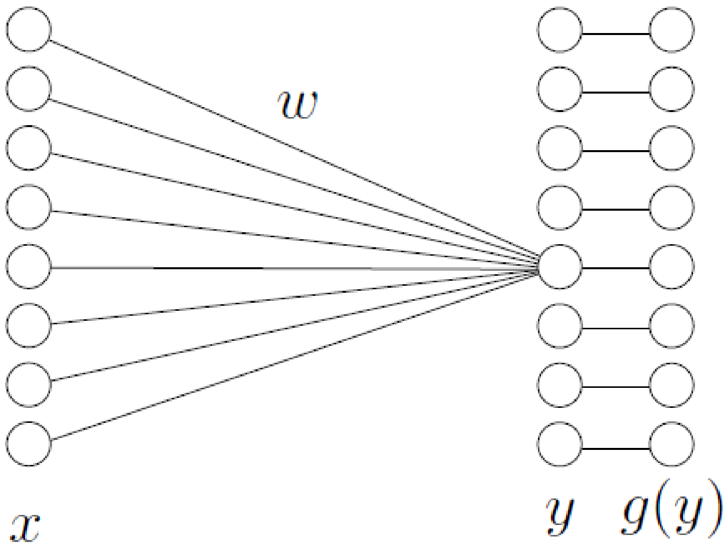
Padding

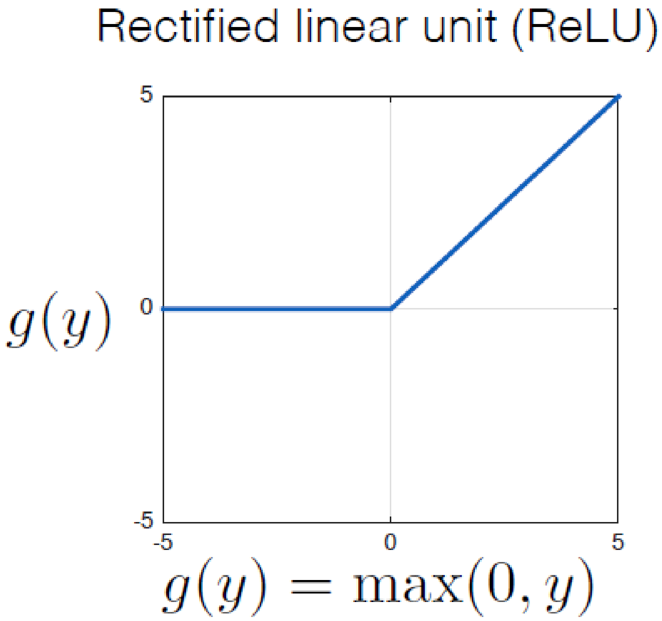
- Padding: artificially fill borders of image
- Useful to keep spatial dimension constant across filters
- Useful with strides and large receptive fields
- Usually fill with 0s



Source: https://github.com/vdumoulin/conv_arithmetic

Nonlinear Activation Function

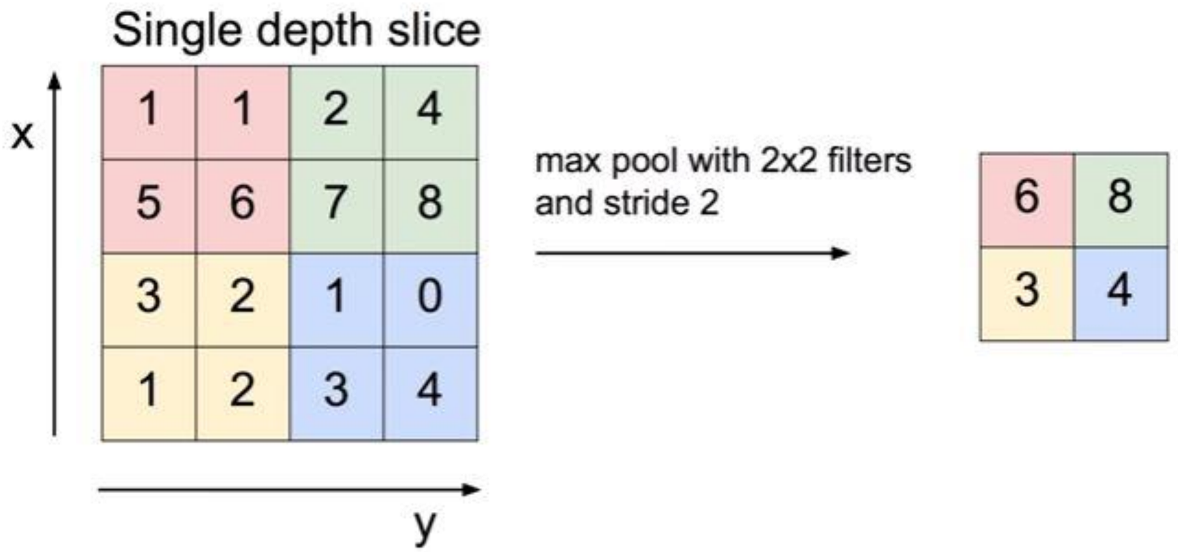




Pooling

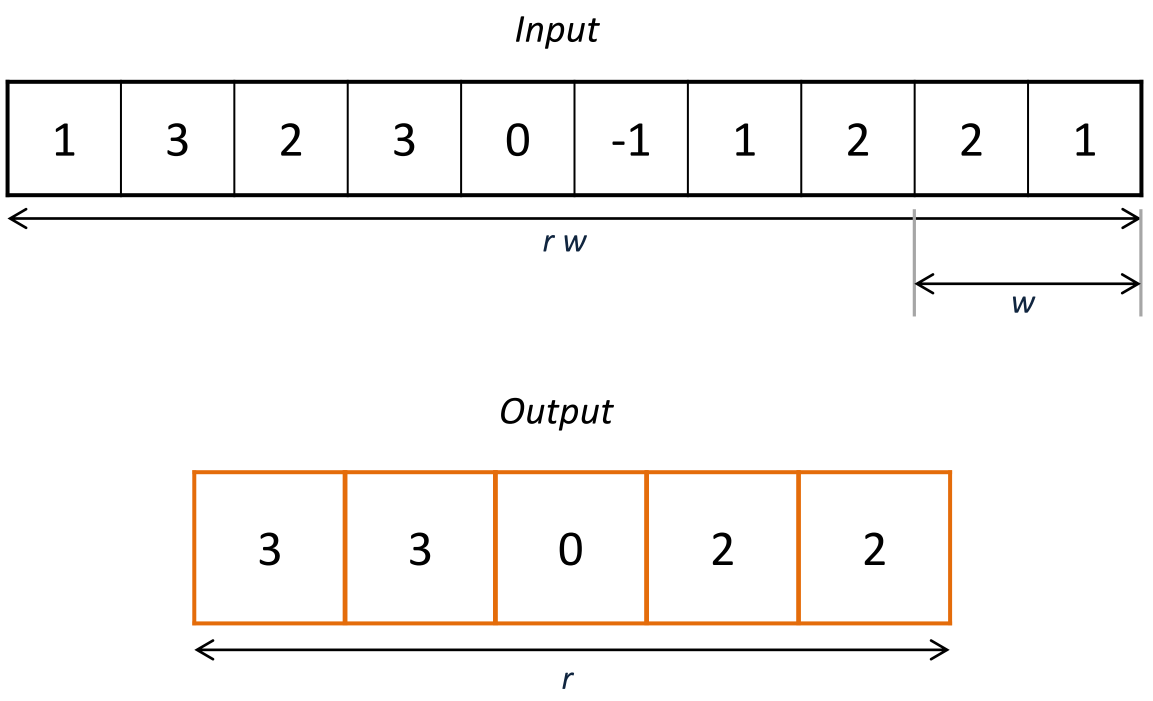
Pooling

- 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 : 2×2
  - No parameters: max or average of 2x2 units



Pooling

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

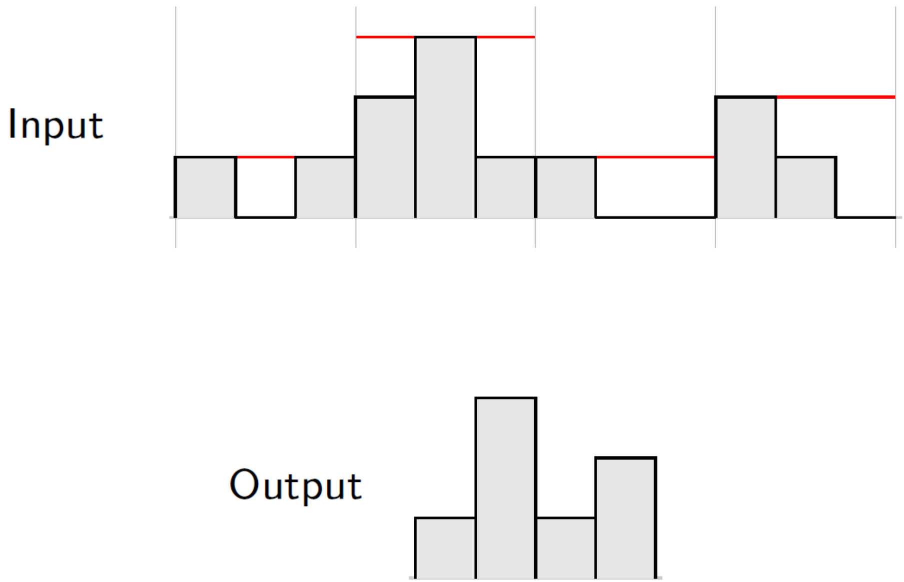


- The average pooling computes average values per block instead of max values

Source: Dr. Francois Fleuret at EPFL

Pooling: Invariance

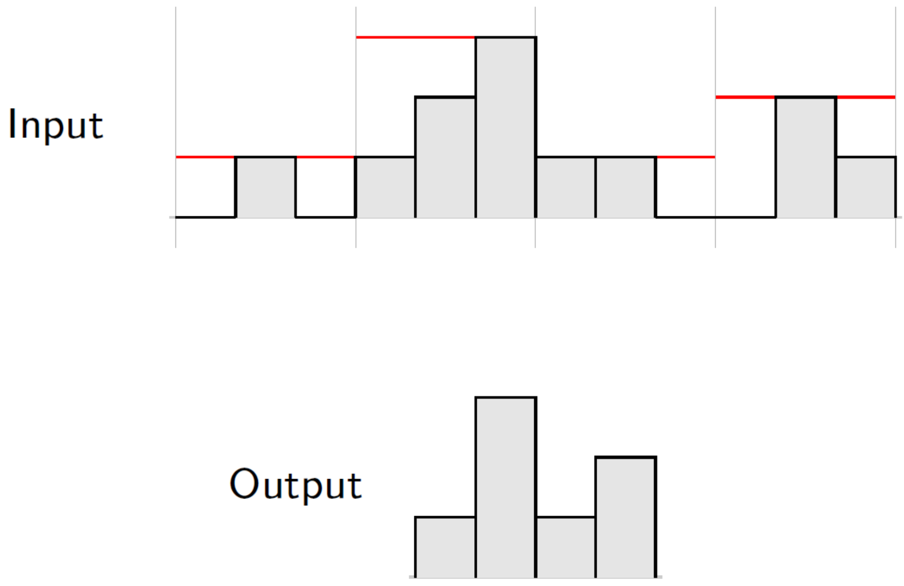
- Pooling provides invariance to any permutation inside one of the cell
- More practically, it provides a pseudo-invariance to deformations that result into local translations



Source: Dr. Francois Fleuret at EPFL

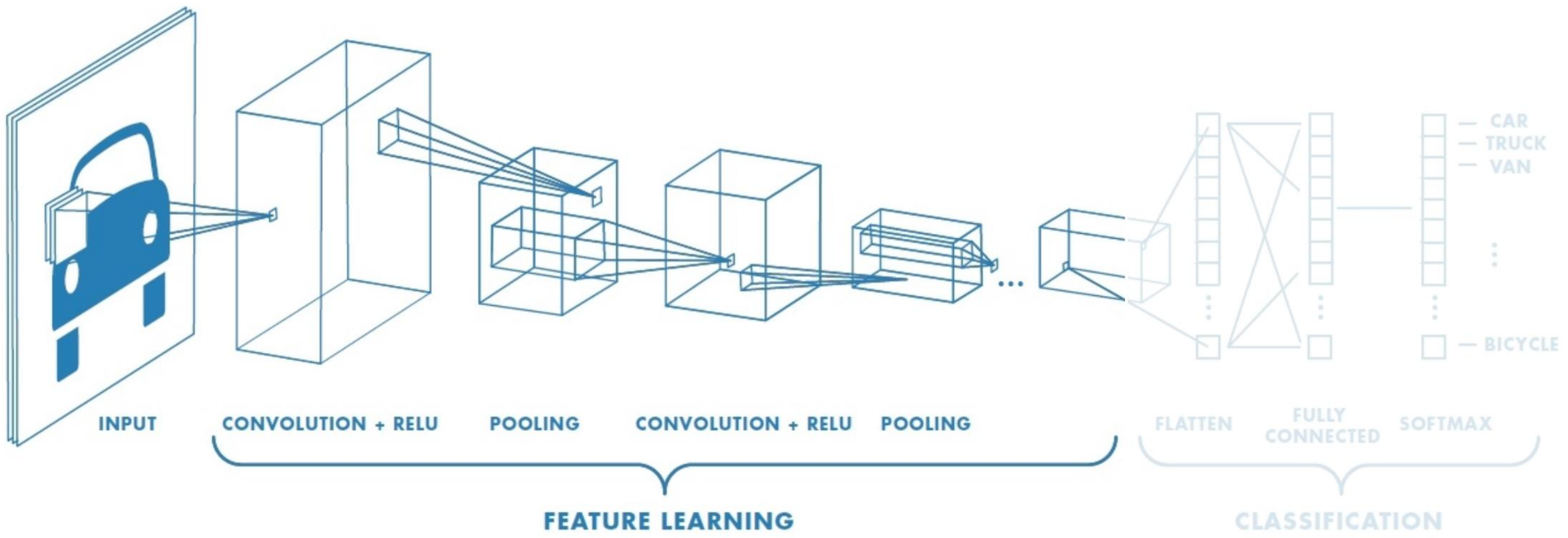
Pooling: Invariance

- 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



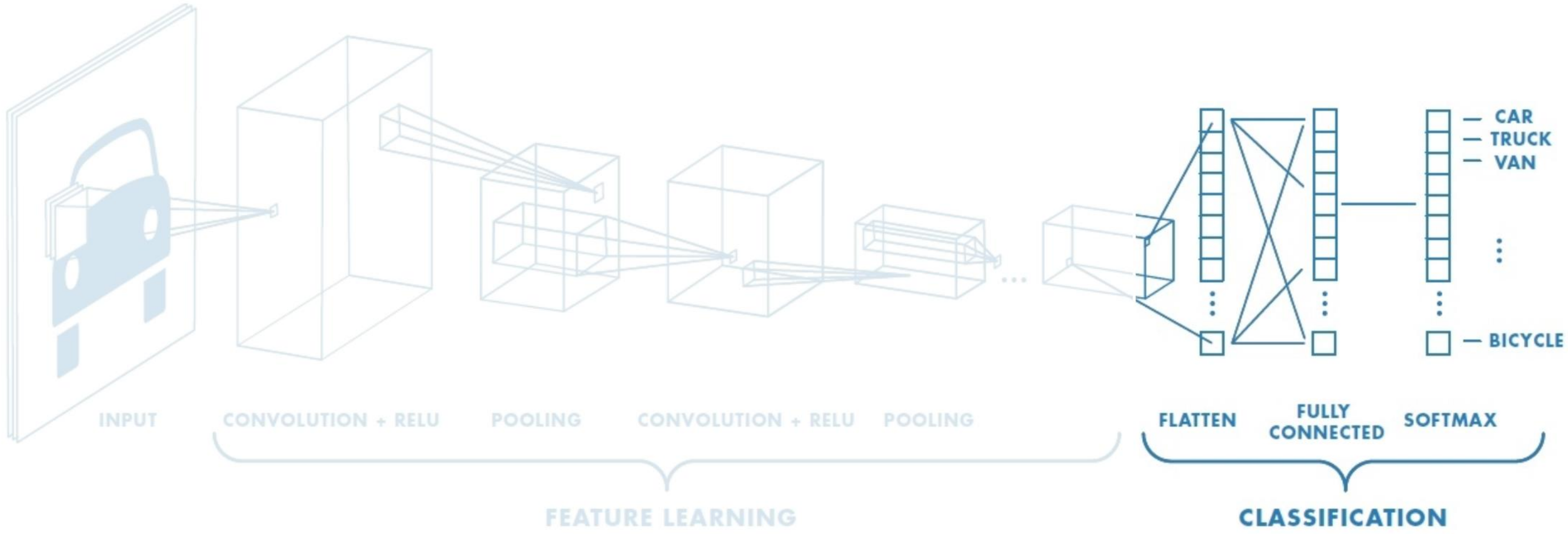
CNNs for Classification: Feature Learning

- 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

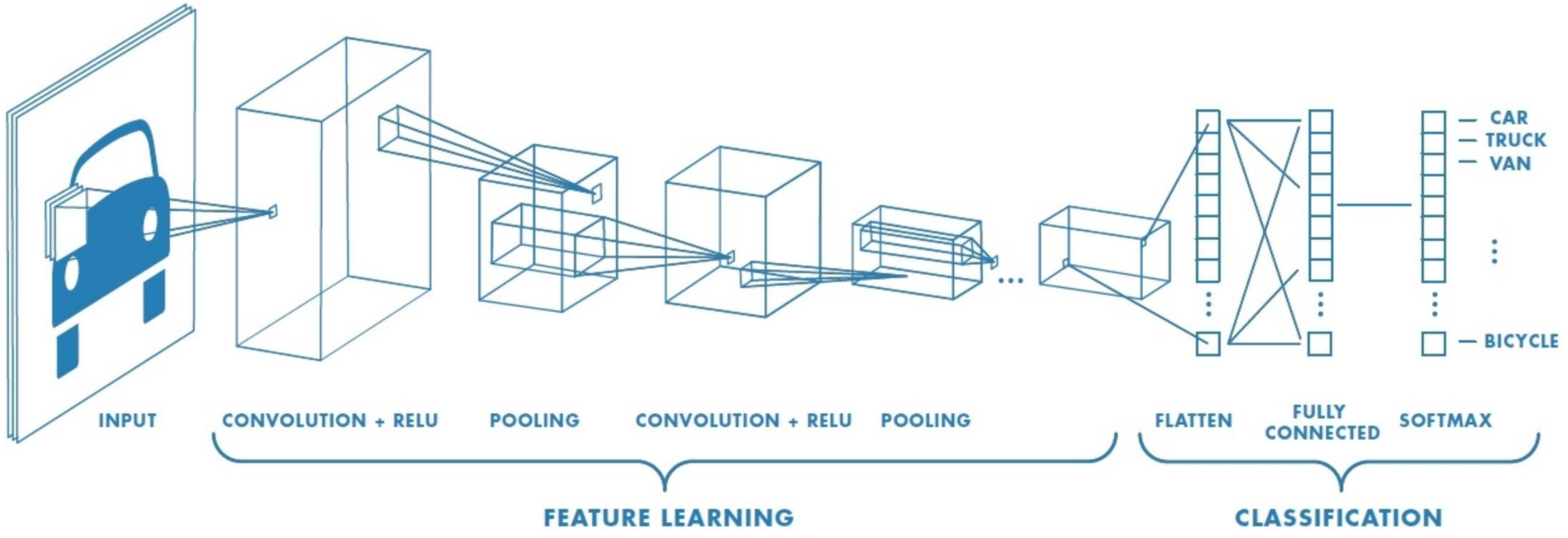


$$\text{softmax}(y_i) = \frac{e^{y_i}}{\sum_j e^{y_j}}$$

Source: 6.S191 Intro. to Deep Learning at MIT

CNNs: Training with Backpropagation

- Learn weights for convolutional filters and fully connected layers
- Backpropagation: cross-entropy loss

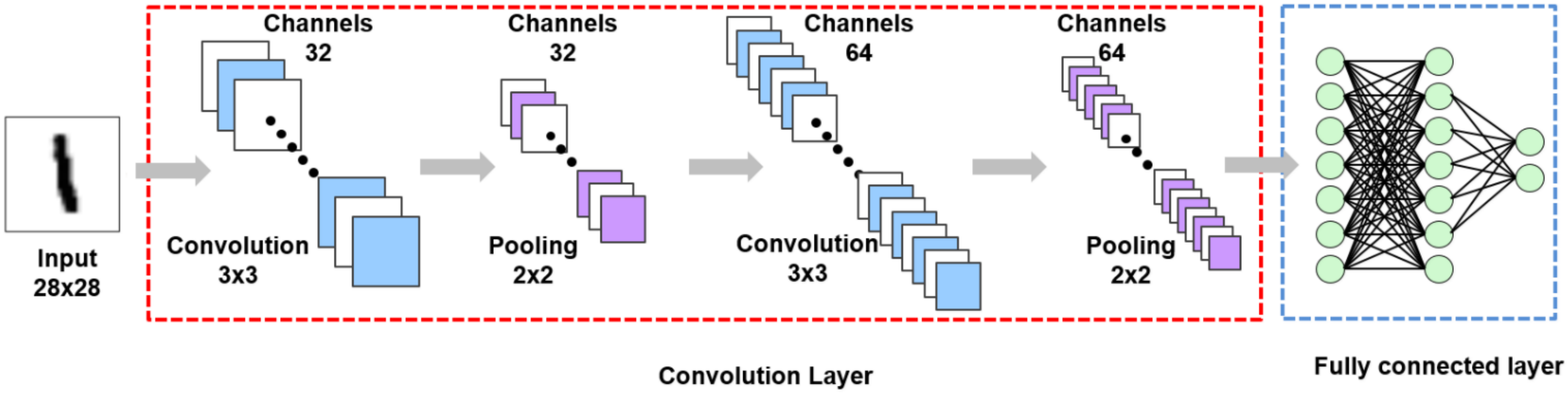


Source: 6.S191 Intro. to Deep Learning at MIT

CNN in TensorFlow

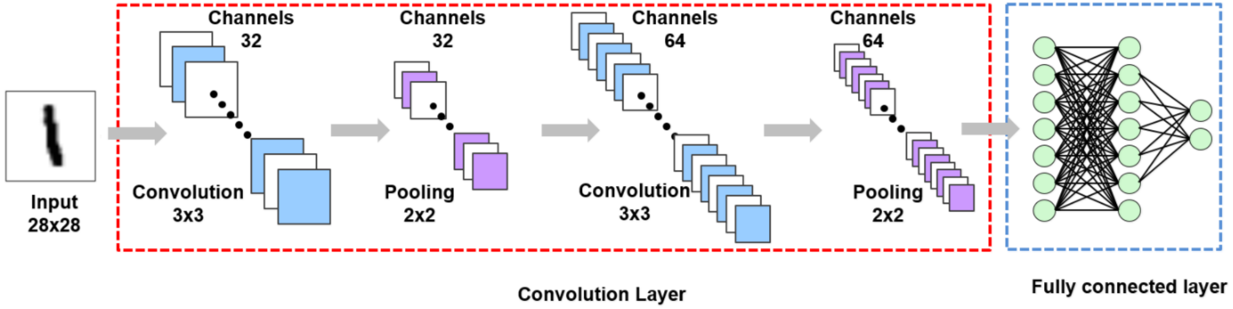
Lab: CNN with TensorFlow

- 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

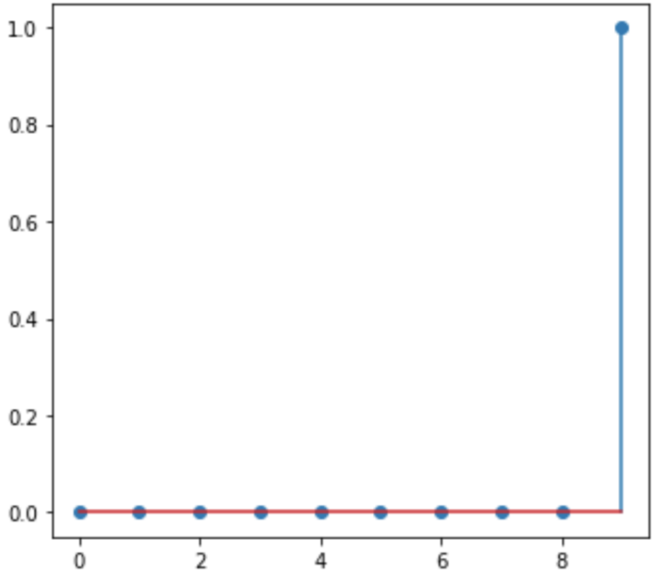
```
model.compile(optimizer = 'adam',
              loss = 'sparse_categorical_crossentropy',
              metrics = ['accuracy'])
```

```
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
```

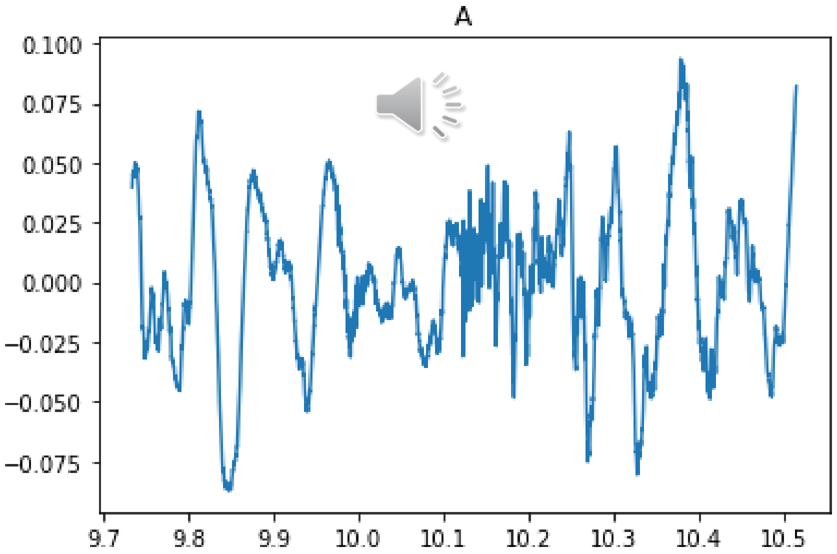
STFT and CNN

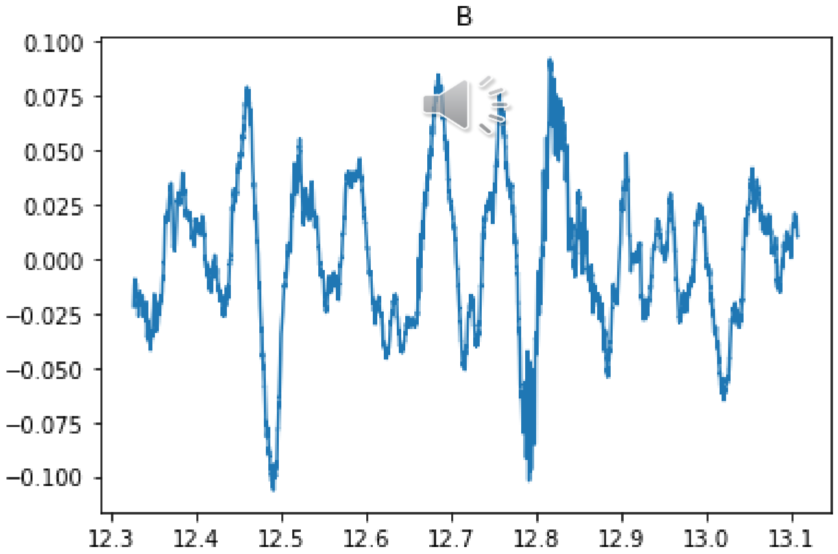
Prof. Seungchul Lee
Industrial AI Lab.

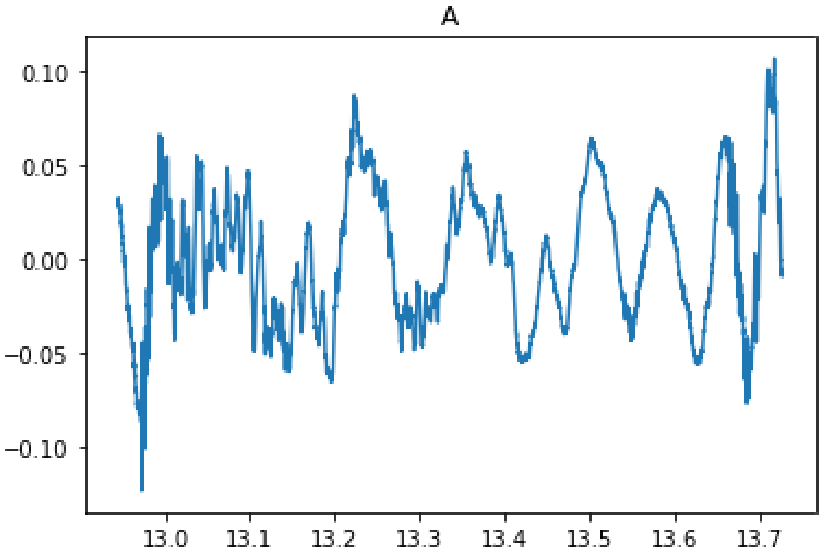
Data

- File format: pkl
- Information: signal, time, label
- Load as dictionary type
- Labels based on one-hot encoding
  - A: [1, 0]
  - B: [0, 1]

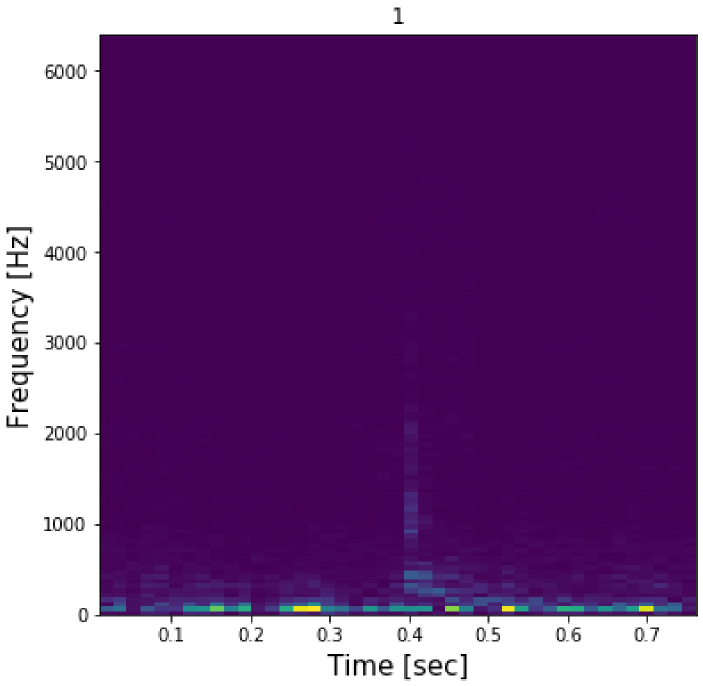
Classification Problem: A or B

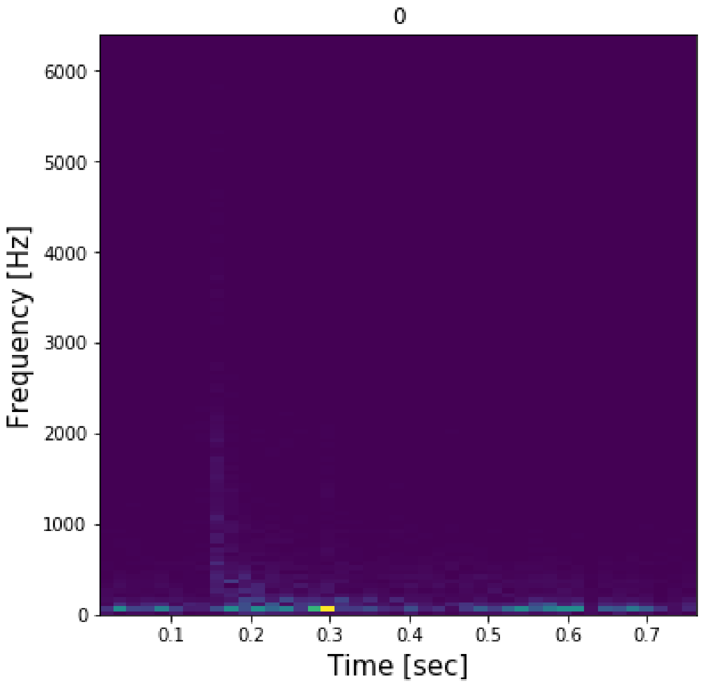




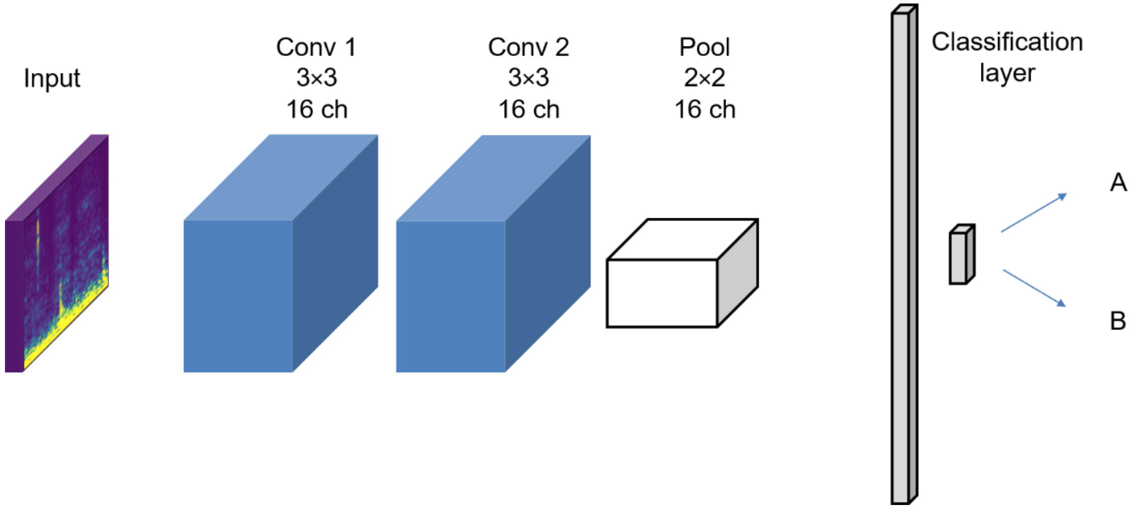


STFT


1
Frequency [Hz]
6000
5000
4000
3000
2000
1000
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
Time [sec]

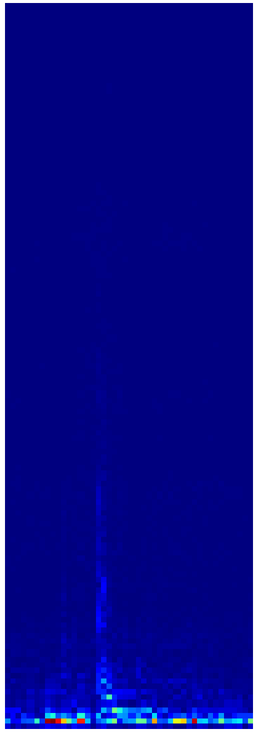

0
Frequency [Hz]
6000
5000
4000
3000
2000
1000
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
Time [sec]

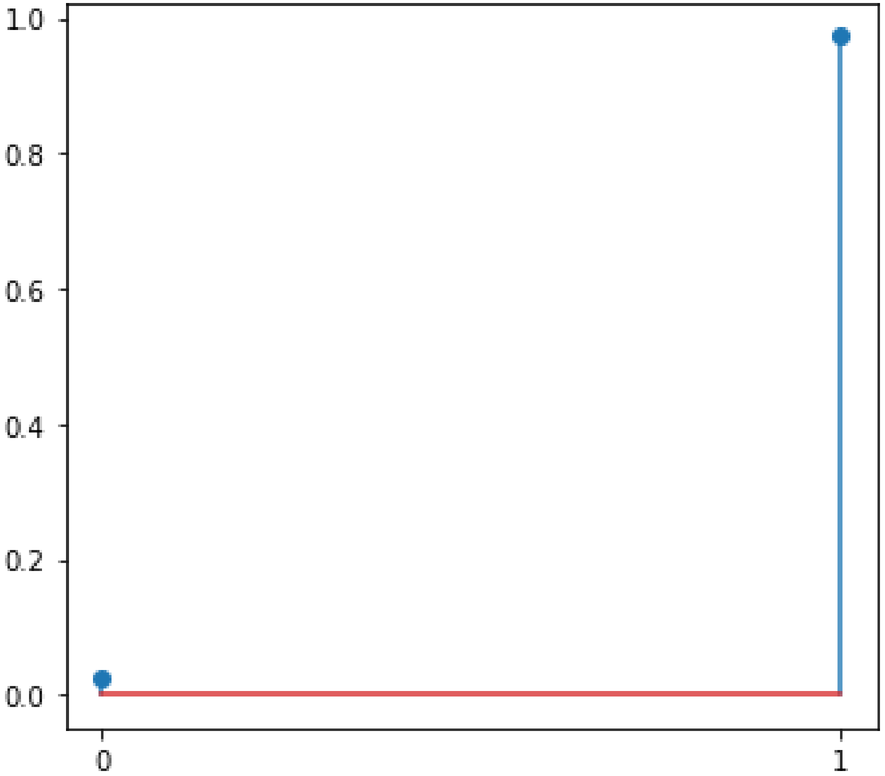
STFT and CNN




```
model = tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(16, (3,3), activation='relu',
                           padding = 'SAME',
                           input_shape = (129, 44, 1)),
    tf.keras.layers.Conv2D(16, (3,3), activation = 'relu',
                           padding = 'SAME',
                           input_shape = (129, 44, 16)),
    tf.keras.layers.MaxPool2D((2,2)),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(128, activation = 'relu'),
    tf.keras.layers.Dense(2, activation = 'softmax')
])
```

Accuracy






```
Prediction : 1
Probability : [0.02604132 0.97395873]
```