Convolutional Neural Networks (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

Convolution

1D Convolution

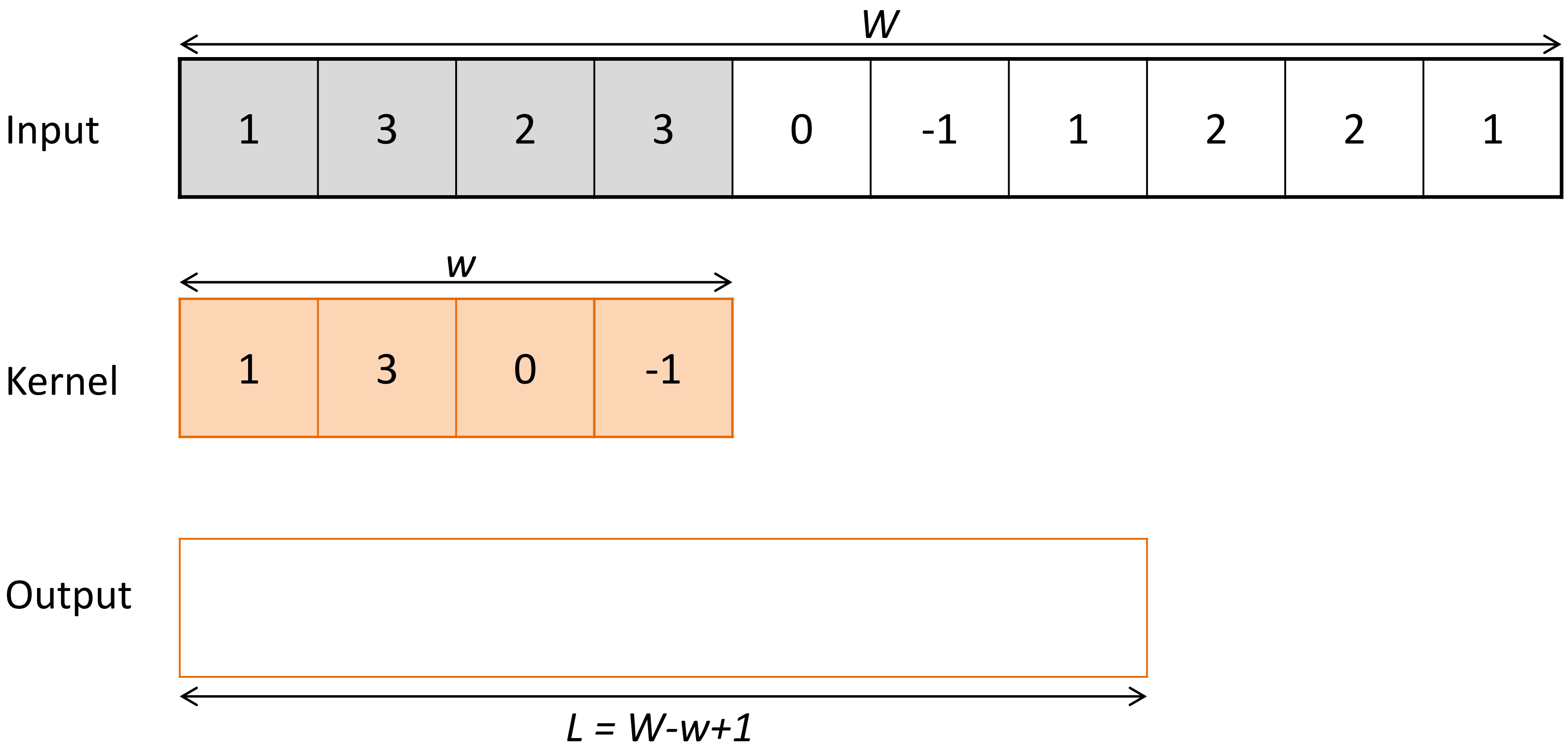
- (actually cross-correlation)

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

Source: Dr. Francois Fleuret at EPFL

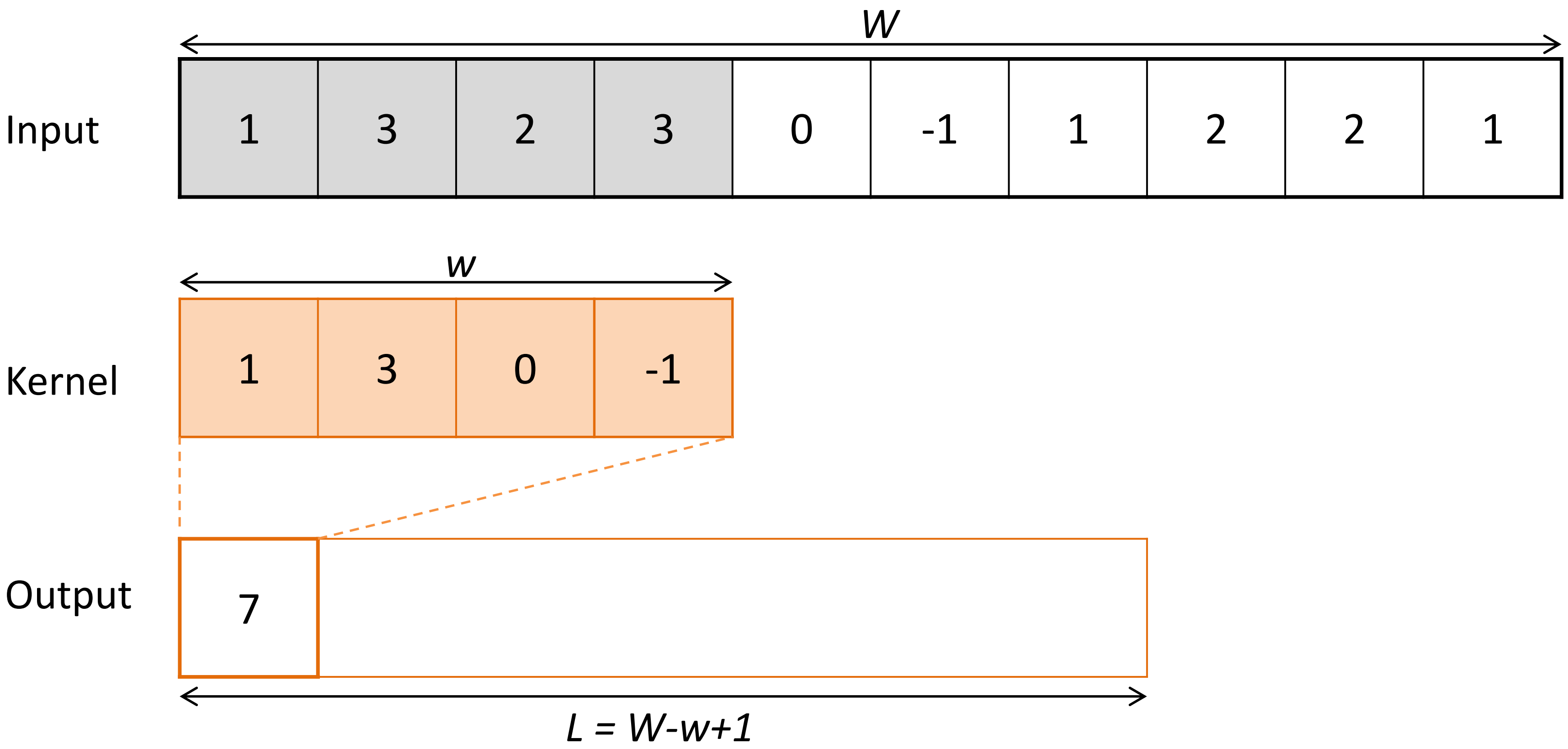
1D Convolution

- (actually cross-correlation)



1D Convolution

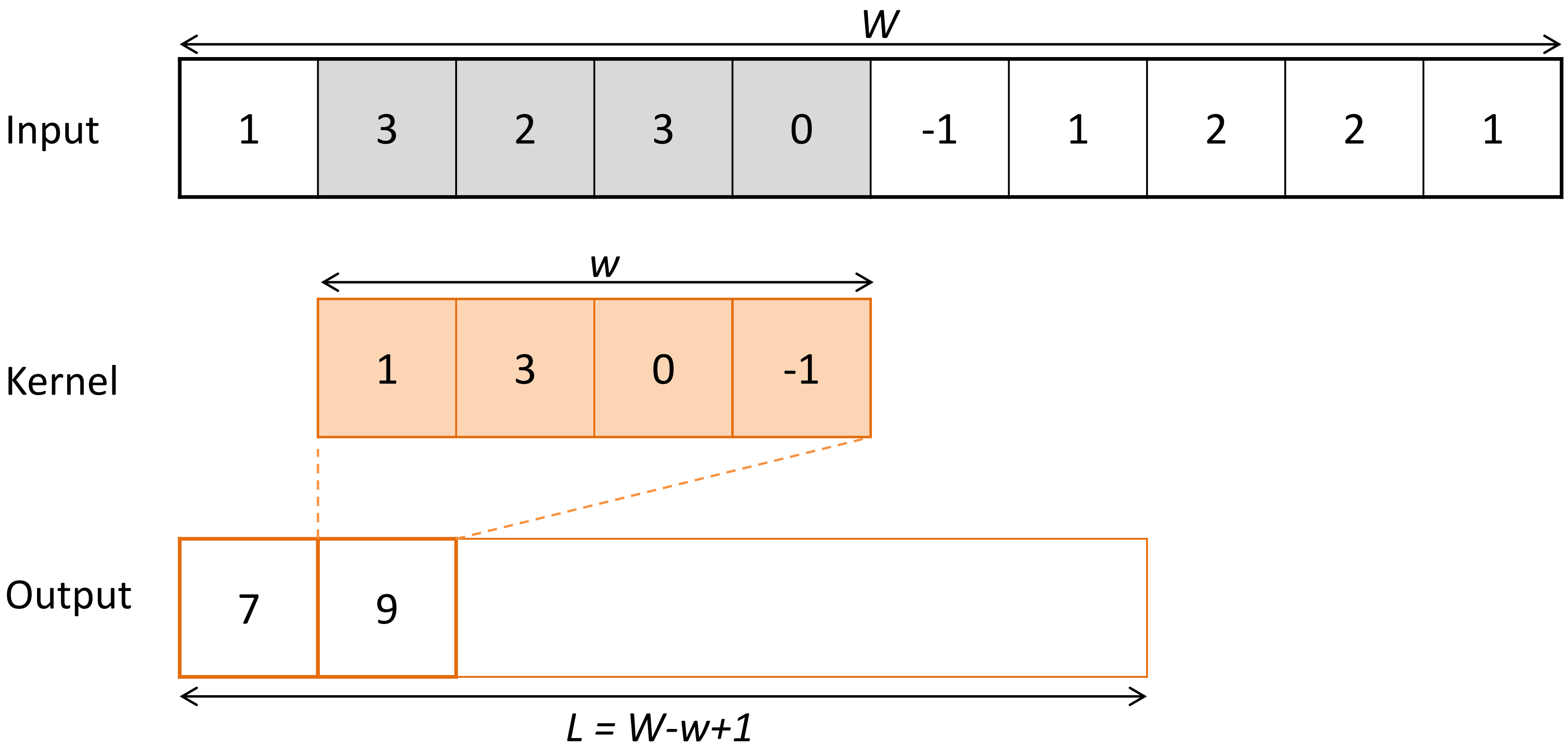
- (actually cross-correlation)



Source: Dr. Francois Fleuret at EPFL

1D Convolution

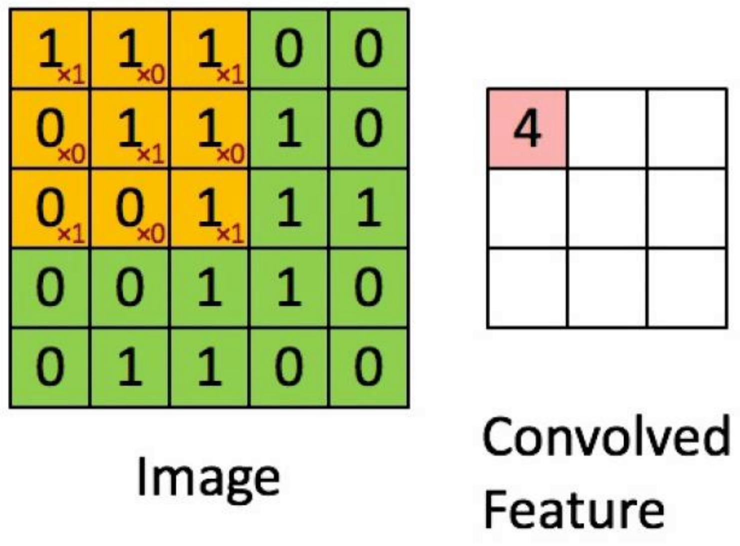
- (actually cross-correlation)

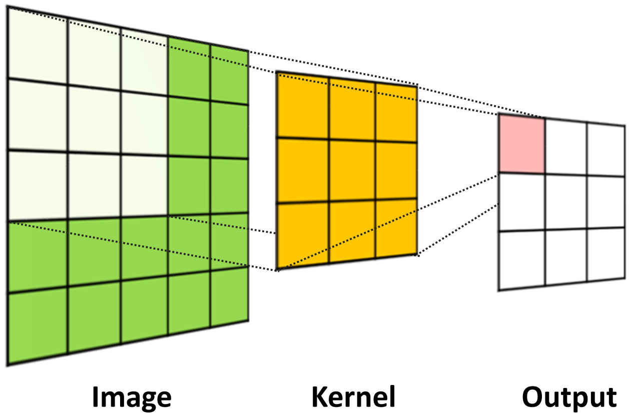


2D Convolution

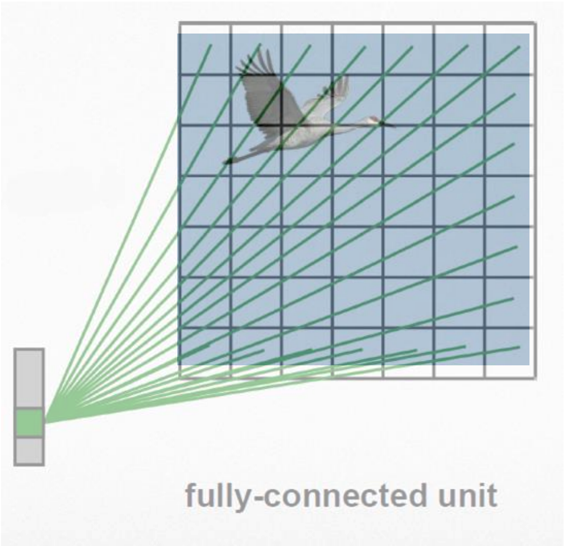
Convolution on Image (= Convolution in 2D)

- 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



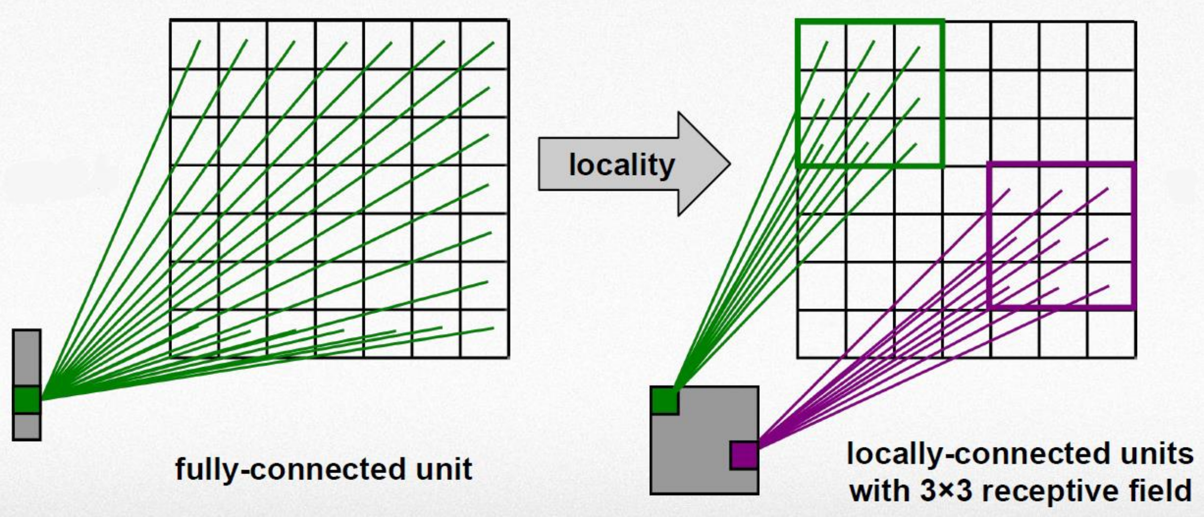


Convolution Mask + Neural Network



Locality

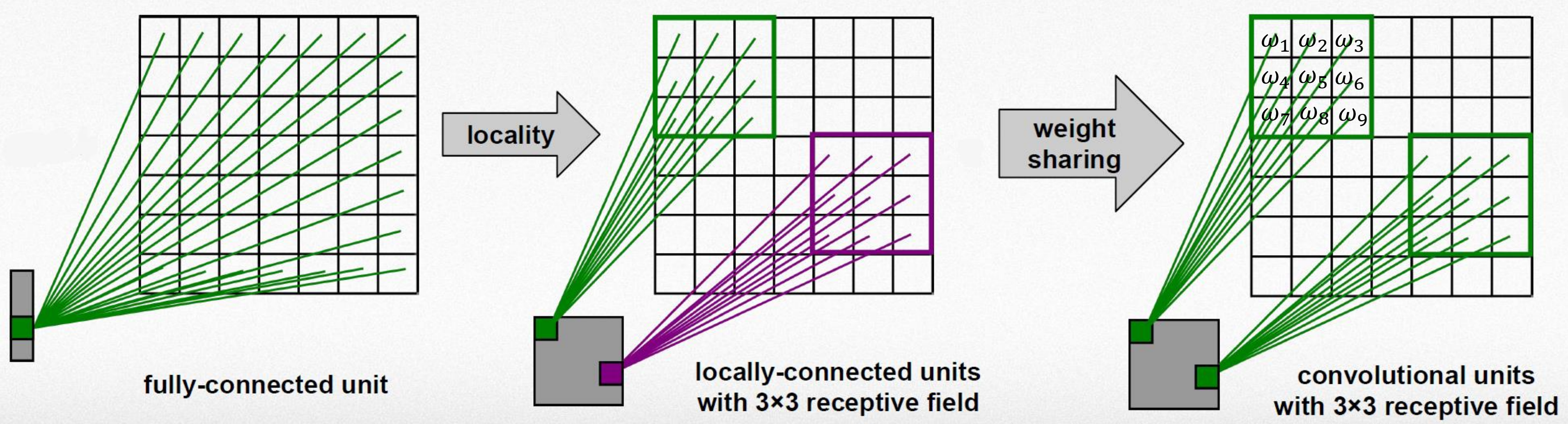
- Locality: objects tend to have a local spatial support
  - fully-connected layer → locally-connected layer



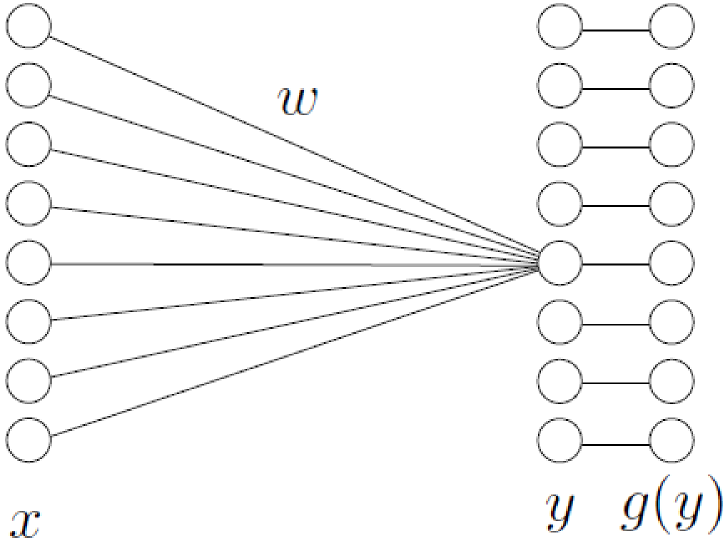
Locality

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

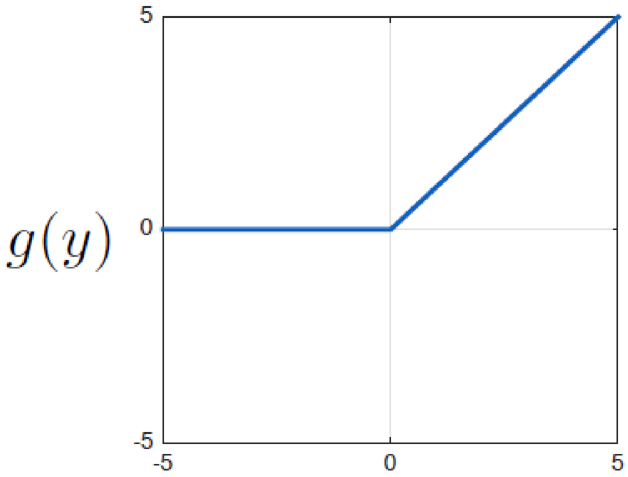
We are not designing the kernel, but are learning the kernel from data
→ Learning feature extractor from data



Nonlinear Activation Function



Rectified linear unit (ReLU)

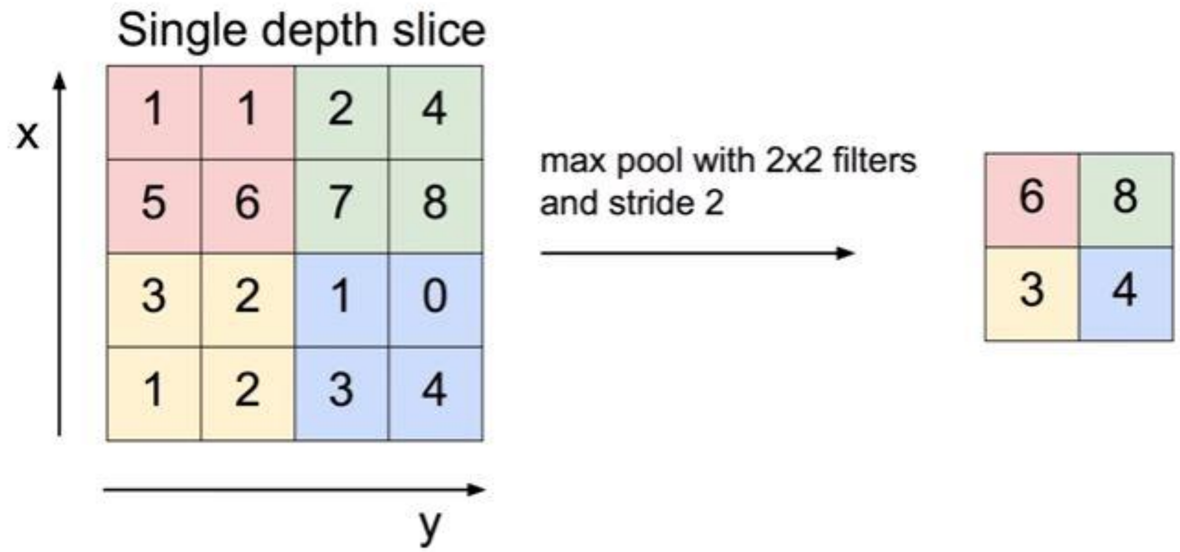


$$g(y) = \max(0, y)$$

Pooling

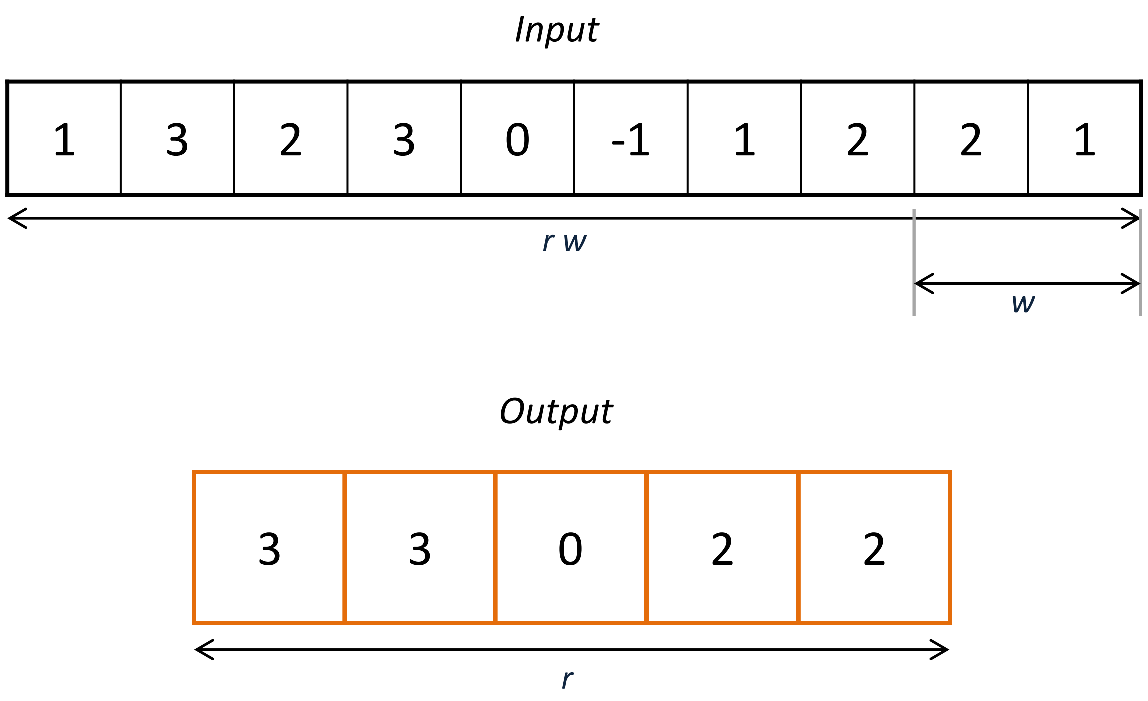
Pooling

- 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

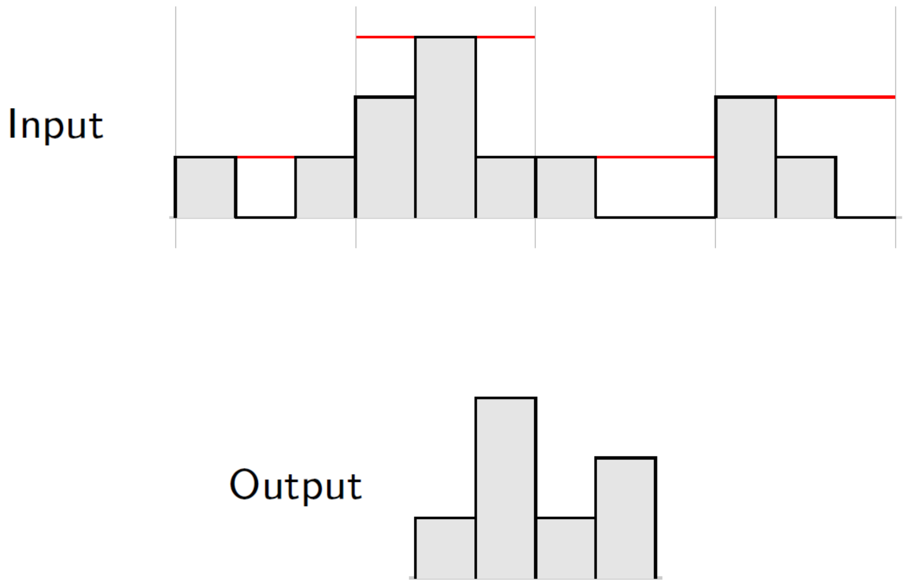
- Such an operation aims at grouping several activations into a single "more meaningful" one.



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

Pooling: Invariance

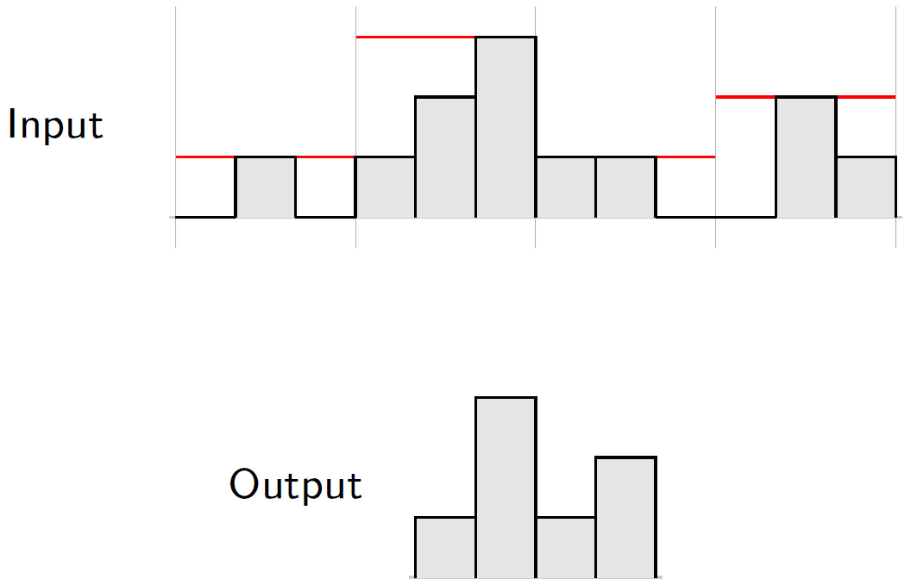
- Pooling provides invariance to any permutation inside one of the cell
- More practically, it provides a pseudo-invariance to deformations that result into local translations



Source: Dr. Francois Fleuret at EPFL

Pooling: Invariance

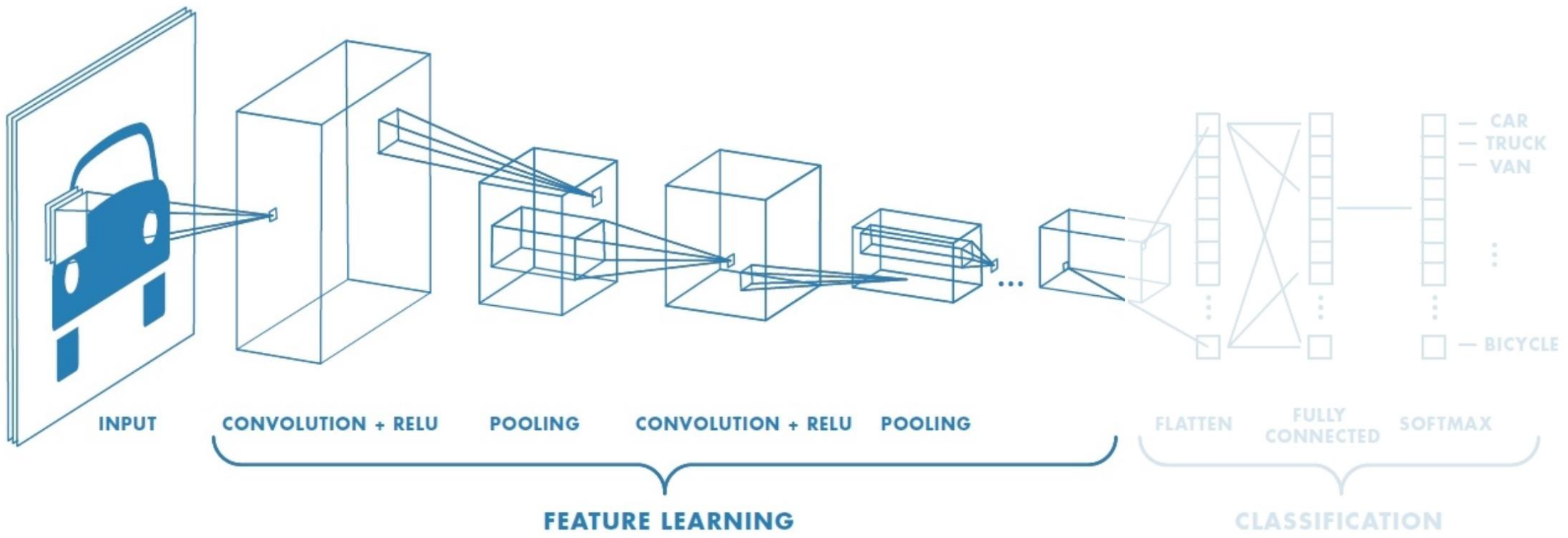
- Pooling provides invariance to any permutation inside one of the cell
- More practically, it provides a pseudo-invariance to deformations that result into local translations



Source: Dr. Francois Fleuret at EPFL

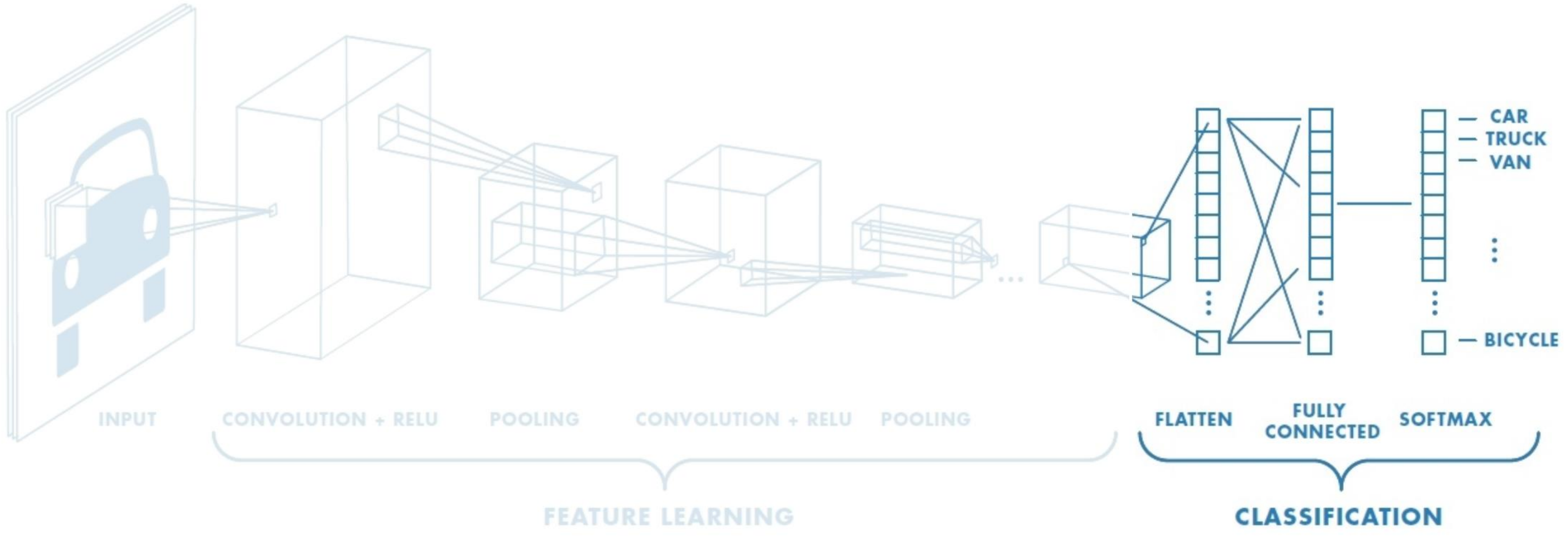
CNNs for Classification: Feature Learning

- 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

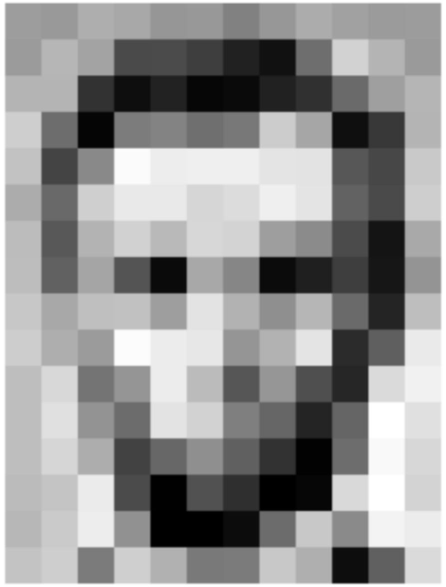


$$\text{softmax}(y_i) = \frac{e^{y_i}}{\sum_j e^{y_j}}$$

실시간 강의자료

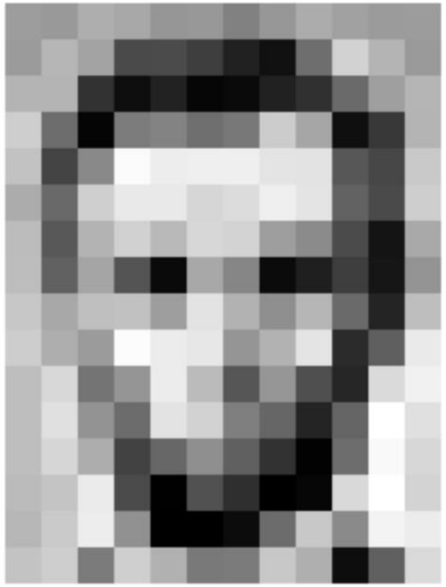
Images

Images Are Numbers



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

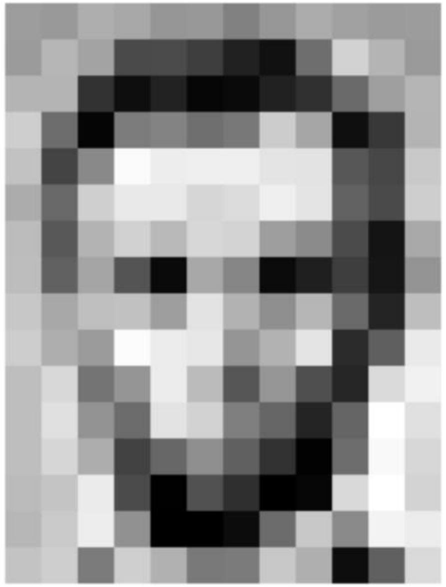
Images Are Numbers

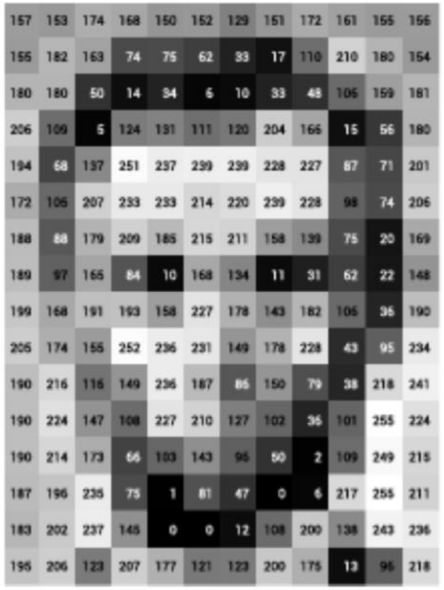


157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	106	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	85	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	95	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	96	218

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

Images Are Numbers





What the computer sees

157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	86	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	96	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	96	218

An image is just a matrix of numbers [0,255]!
i.e., 1080x1080x3 for an RGB image

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

Images

Original image

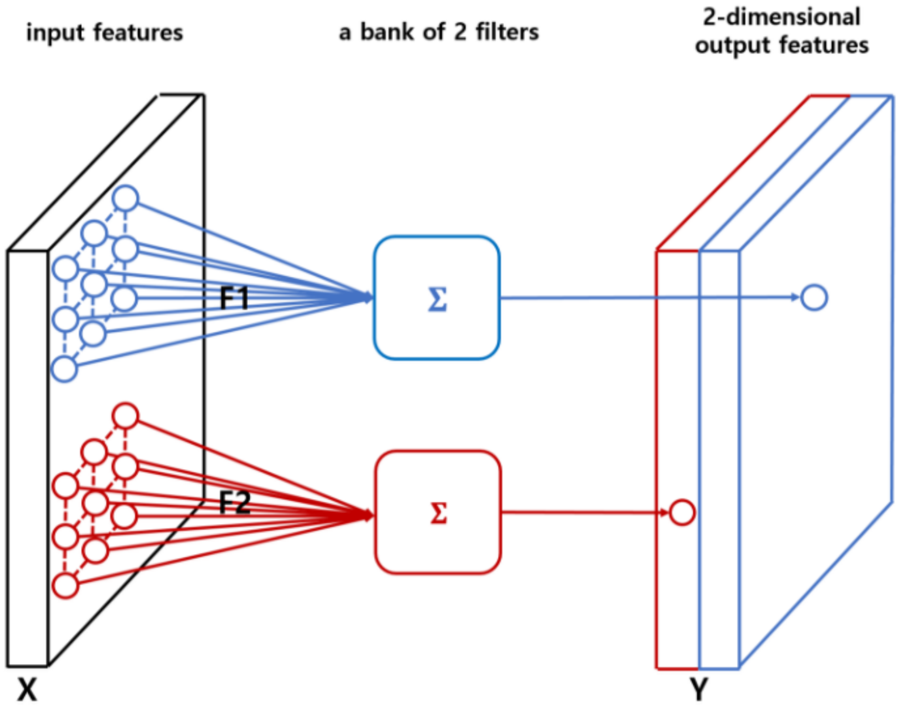
R

G

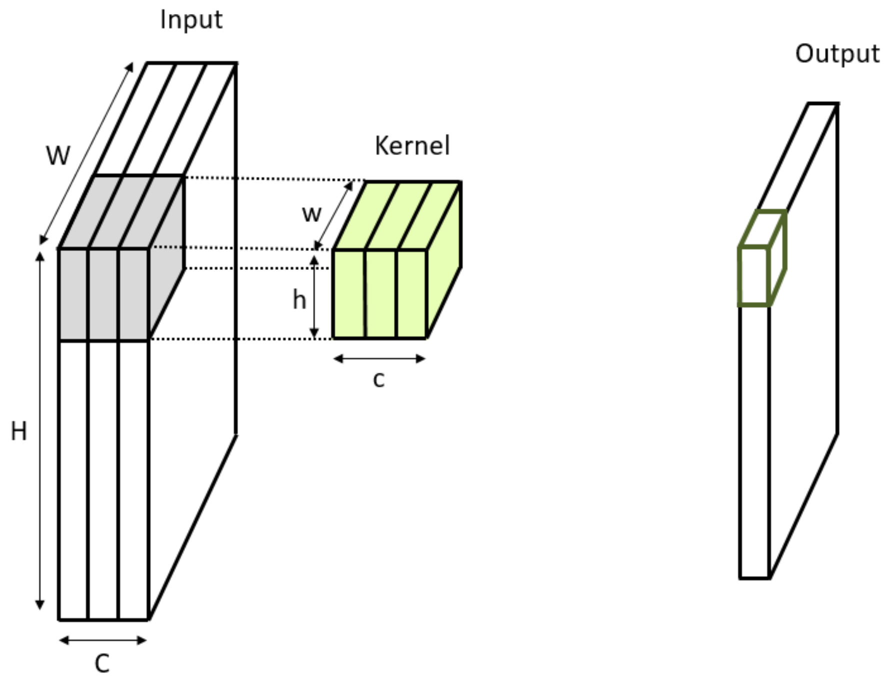
B

Gray image

Multiple Filters (or Kernels)

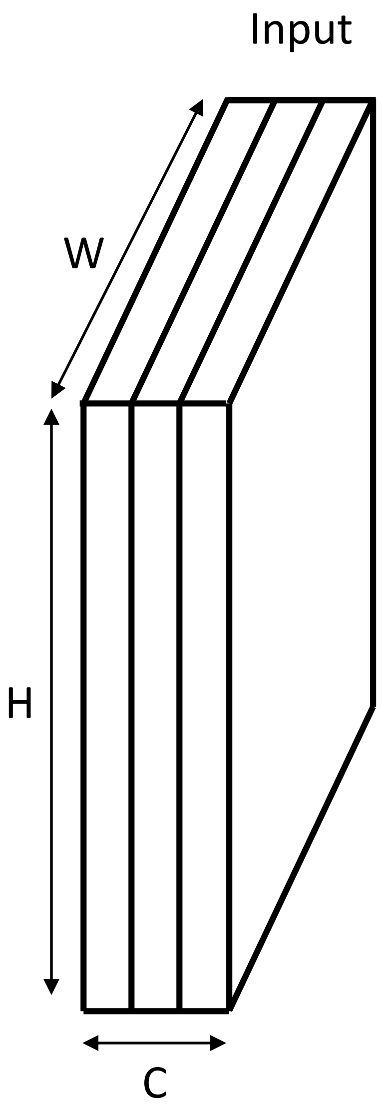


Channels



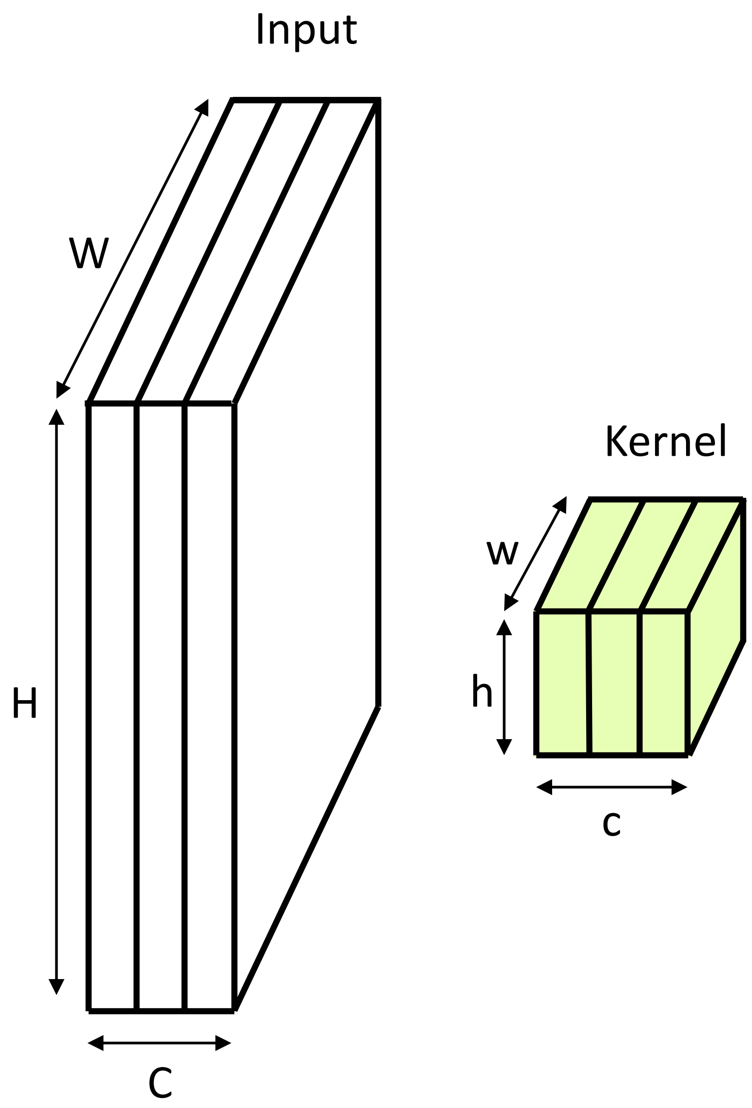
- Colored image = tensor of shape (height, width, channels)
- Convolutions are usually computed for each channel and summed:
- Kernel size aka receptive field (usually 1, 3, 5, 7, 11)

Multi-channel 2D Convolution



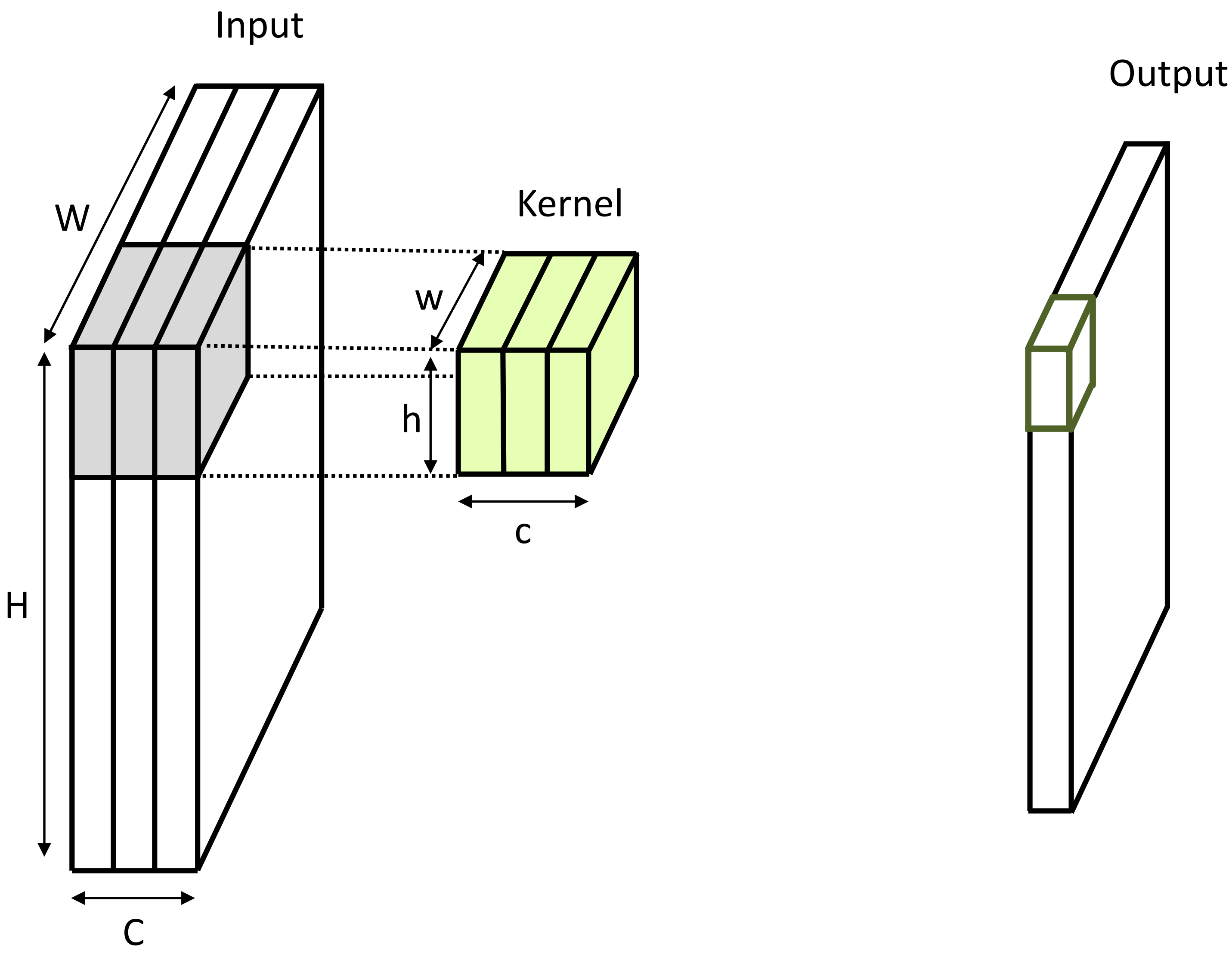
Source: Dr. Francois Fleuret at EPFL

Multi-channel 2D Convolution

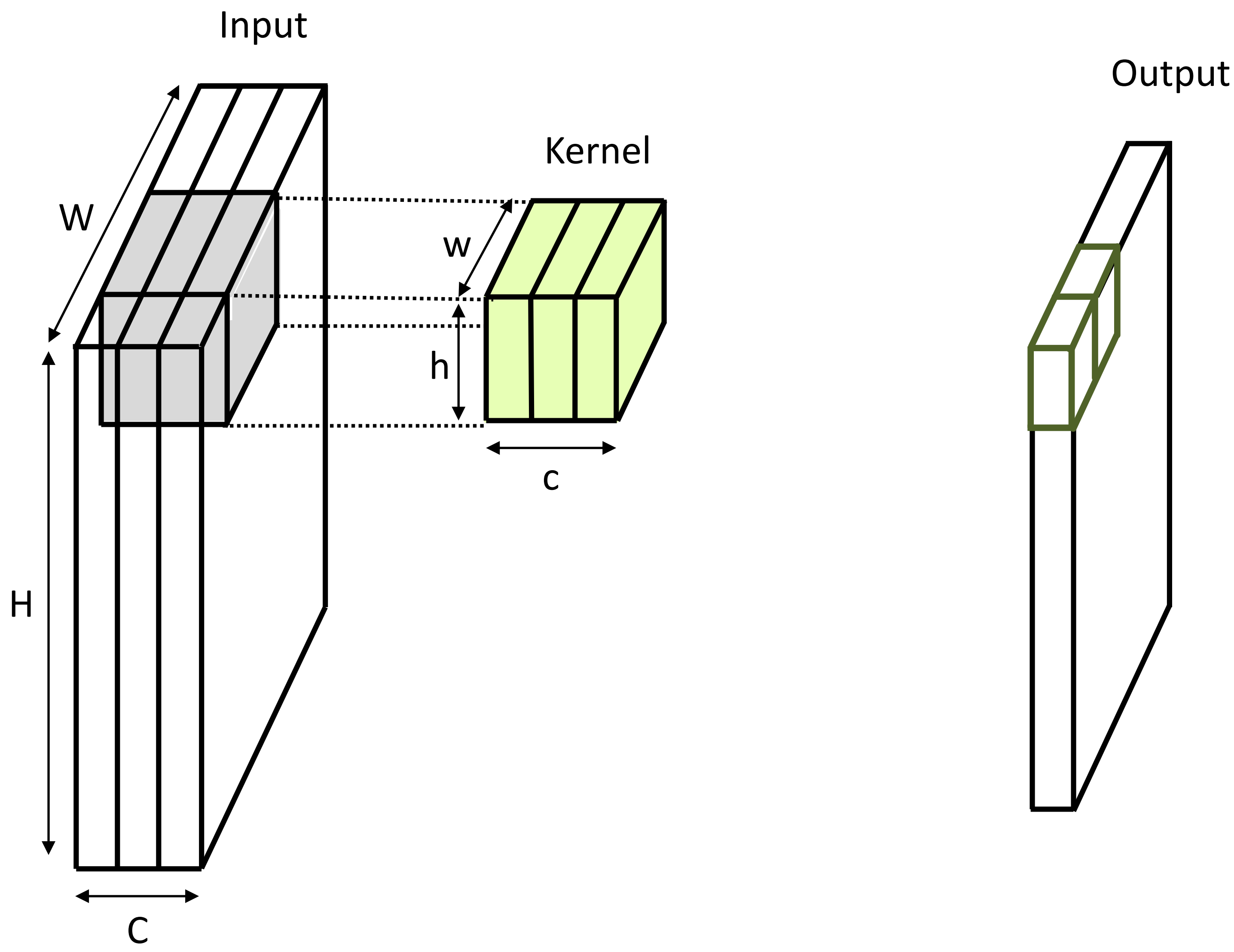


Source: Dr. Francois Fleuret at EPFL

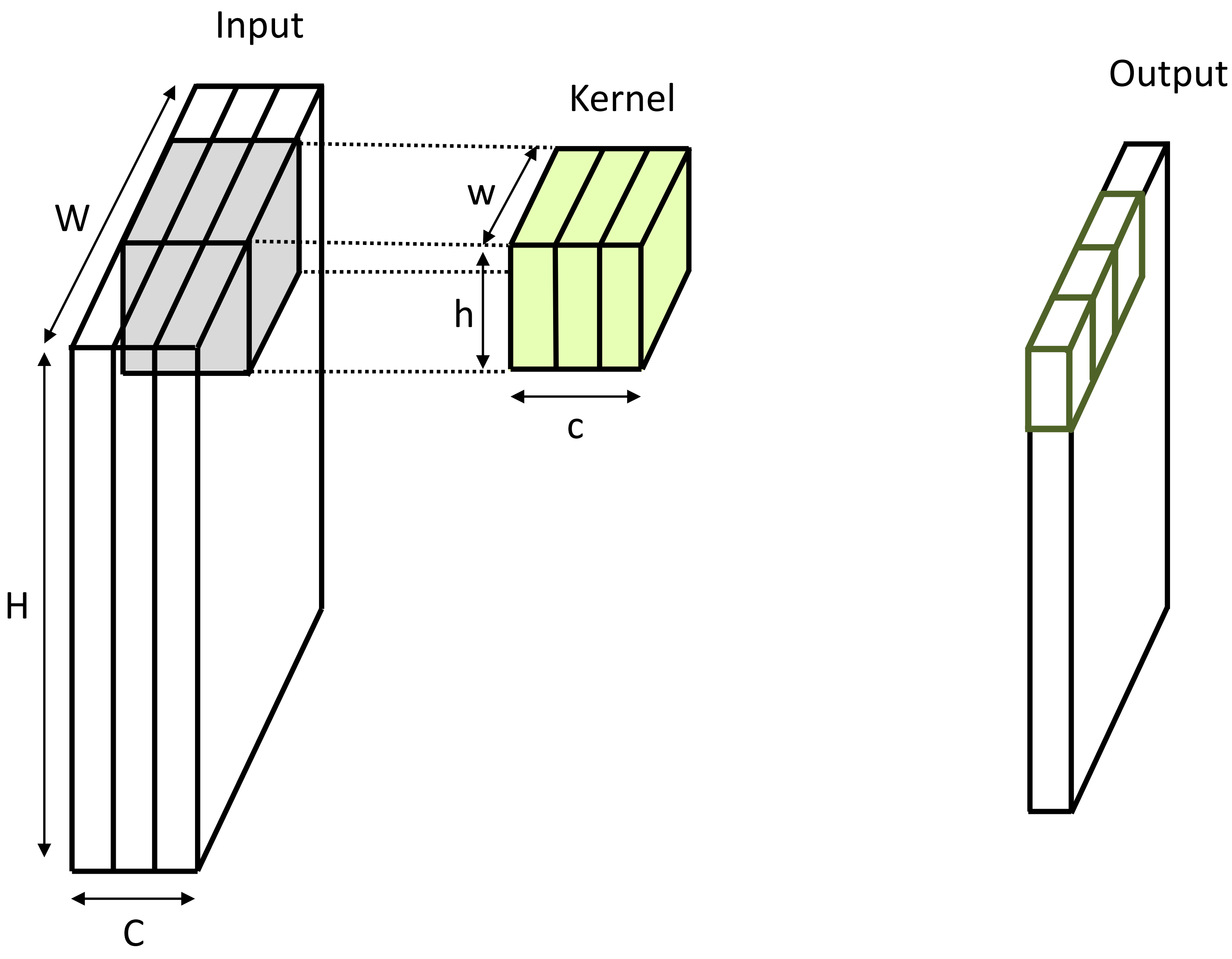
Multi-channel 2D Convolution



Multi-channel 2D Convolution

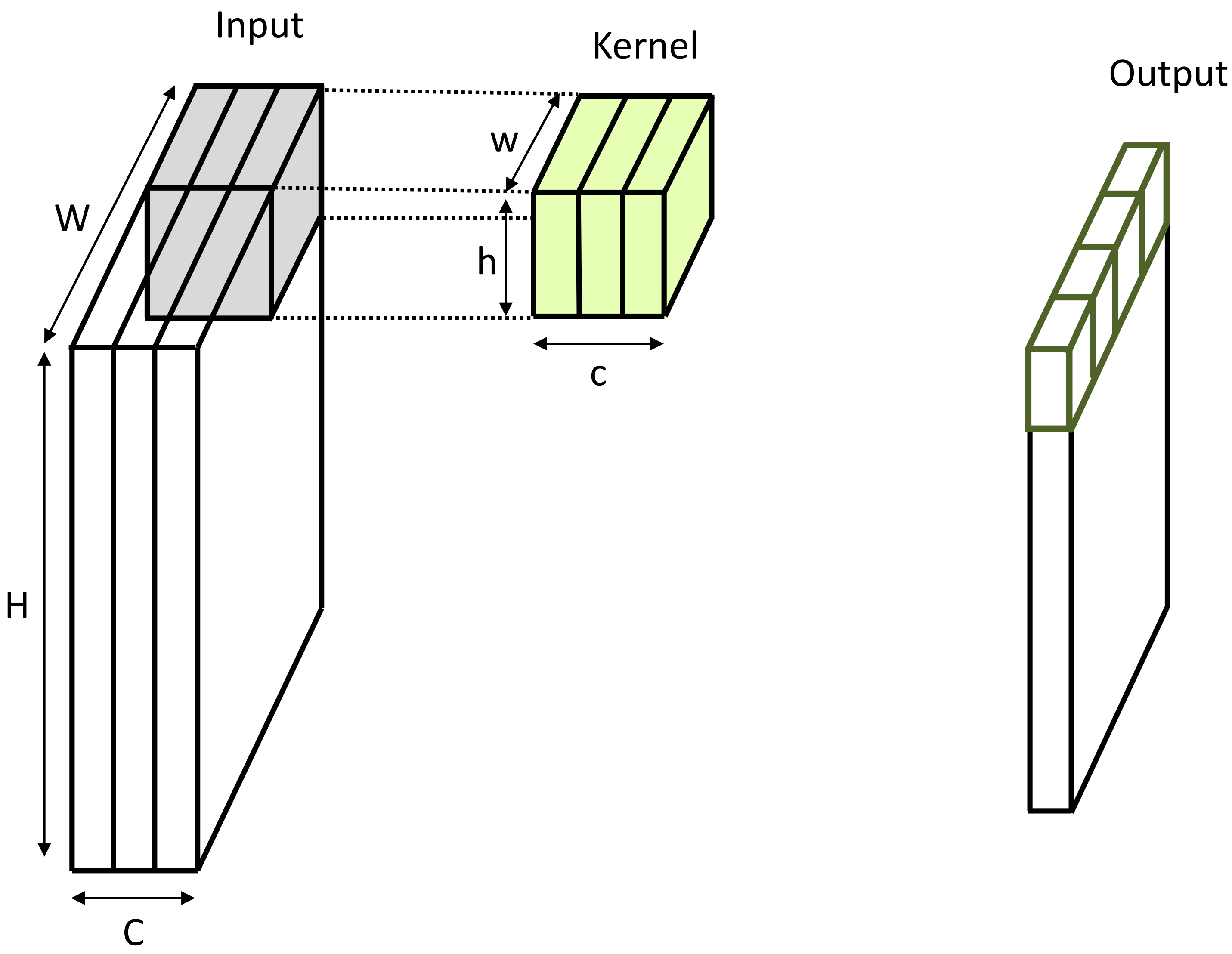


Multi-channel 2D Convolution



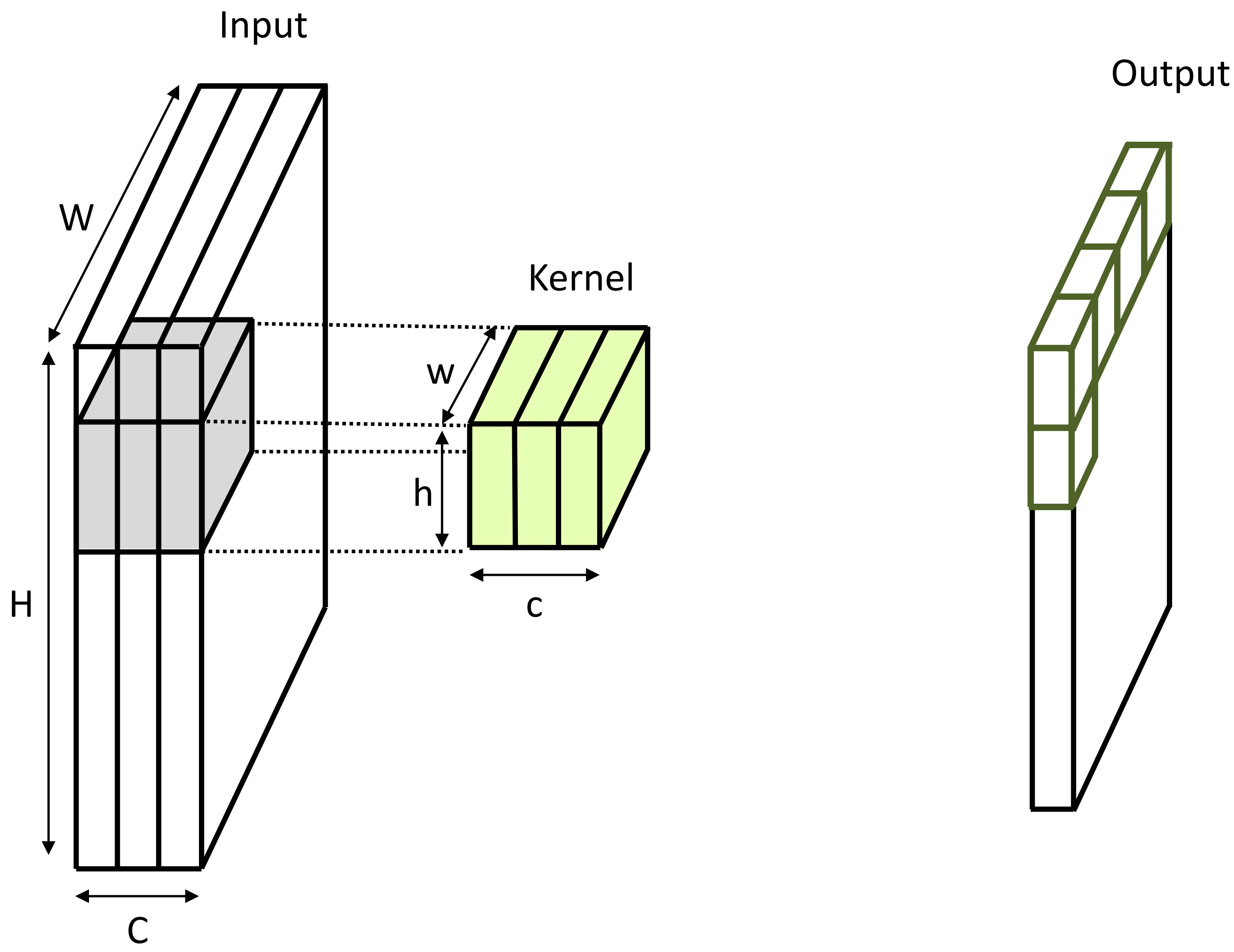
Source: Dr. Francois Fleuret at EPFL

Multi-channel 2D Convolution



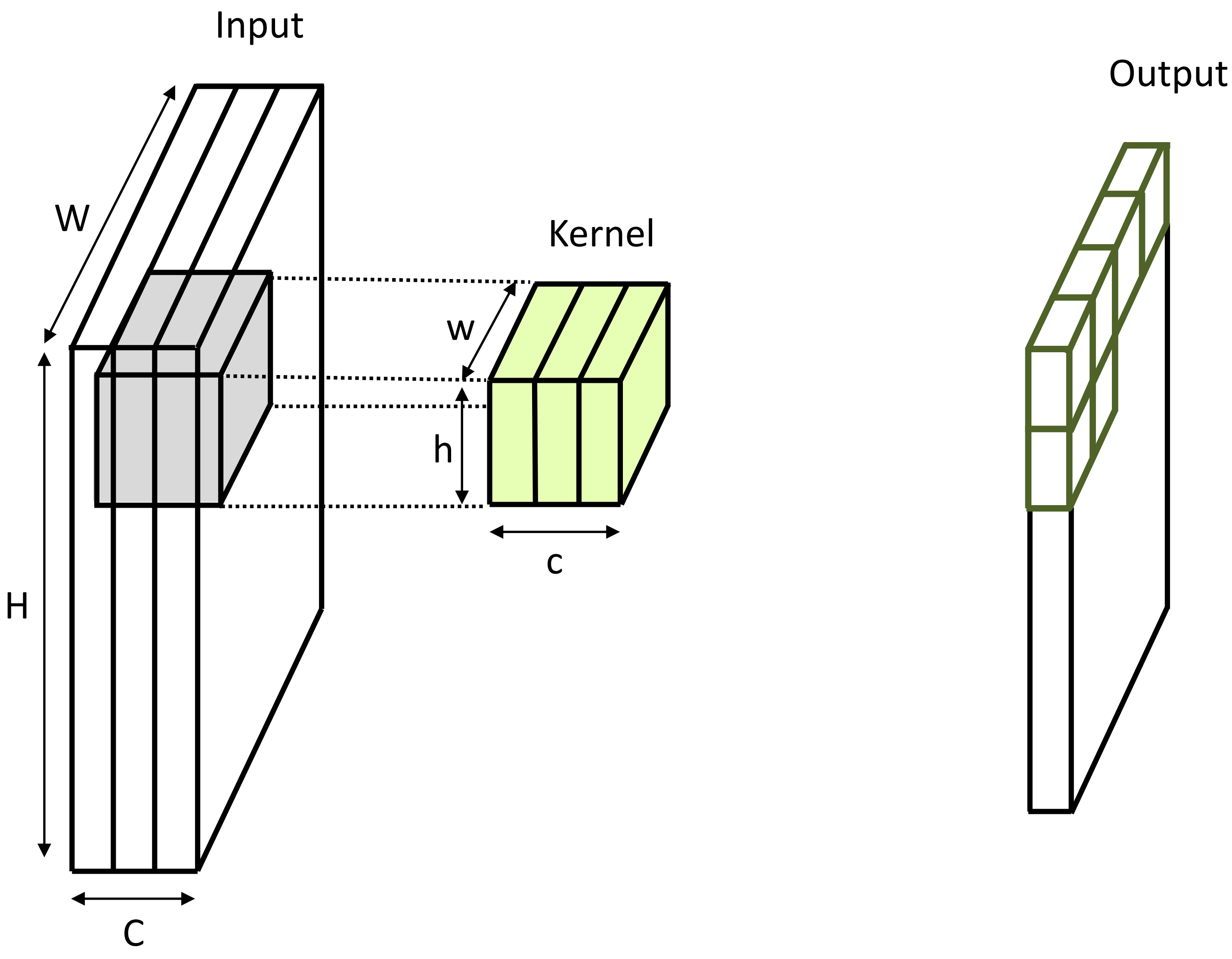
Source: Dr. Francois Fleuret at EPFL

Multi-channel 2D Convolution



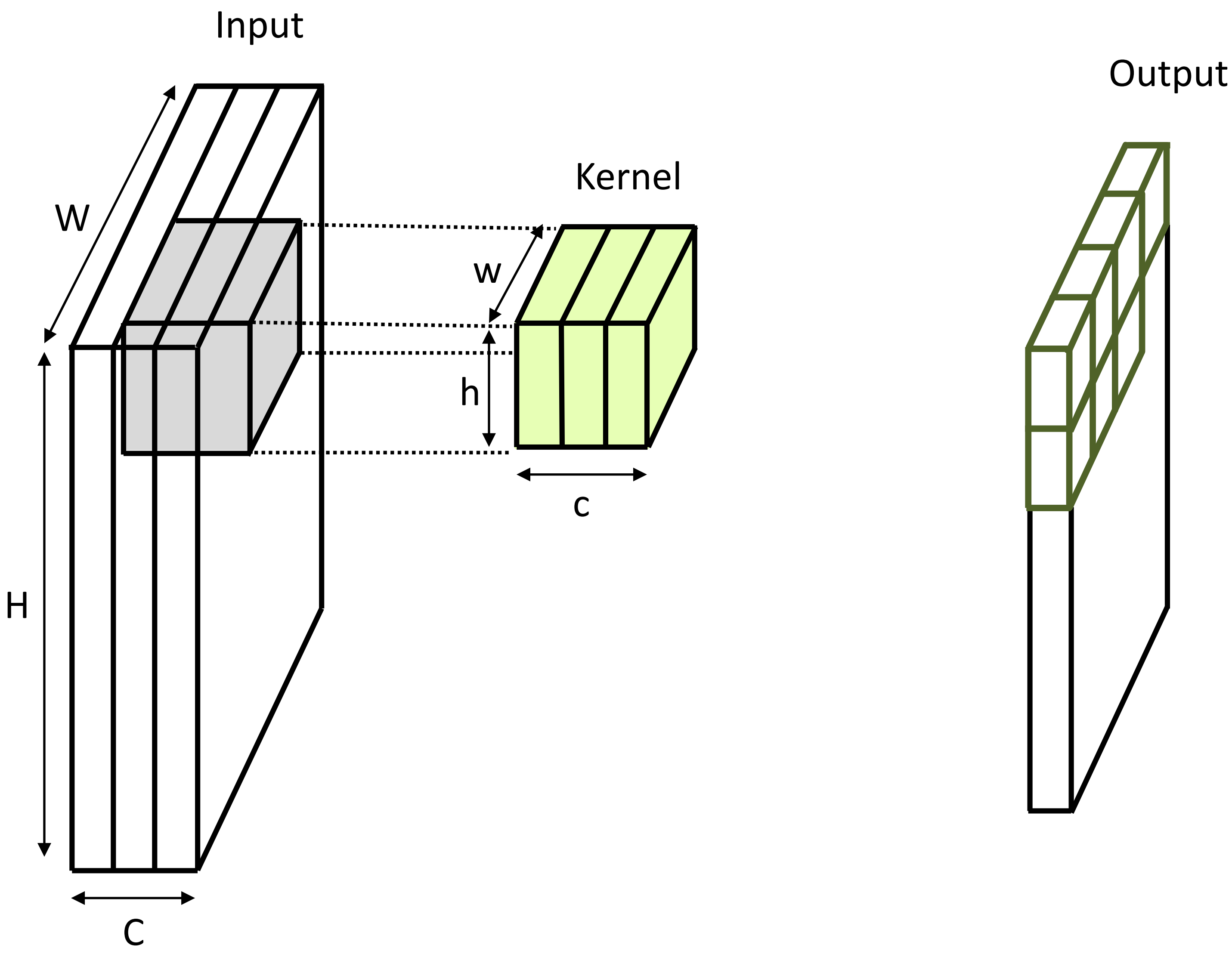
Source: Dr. Francois Fleuret at EPFL

Multi-channel 2D Convolution



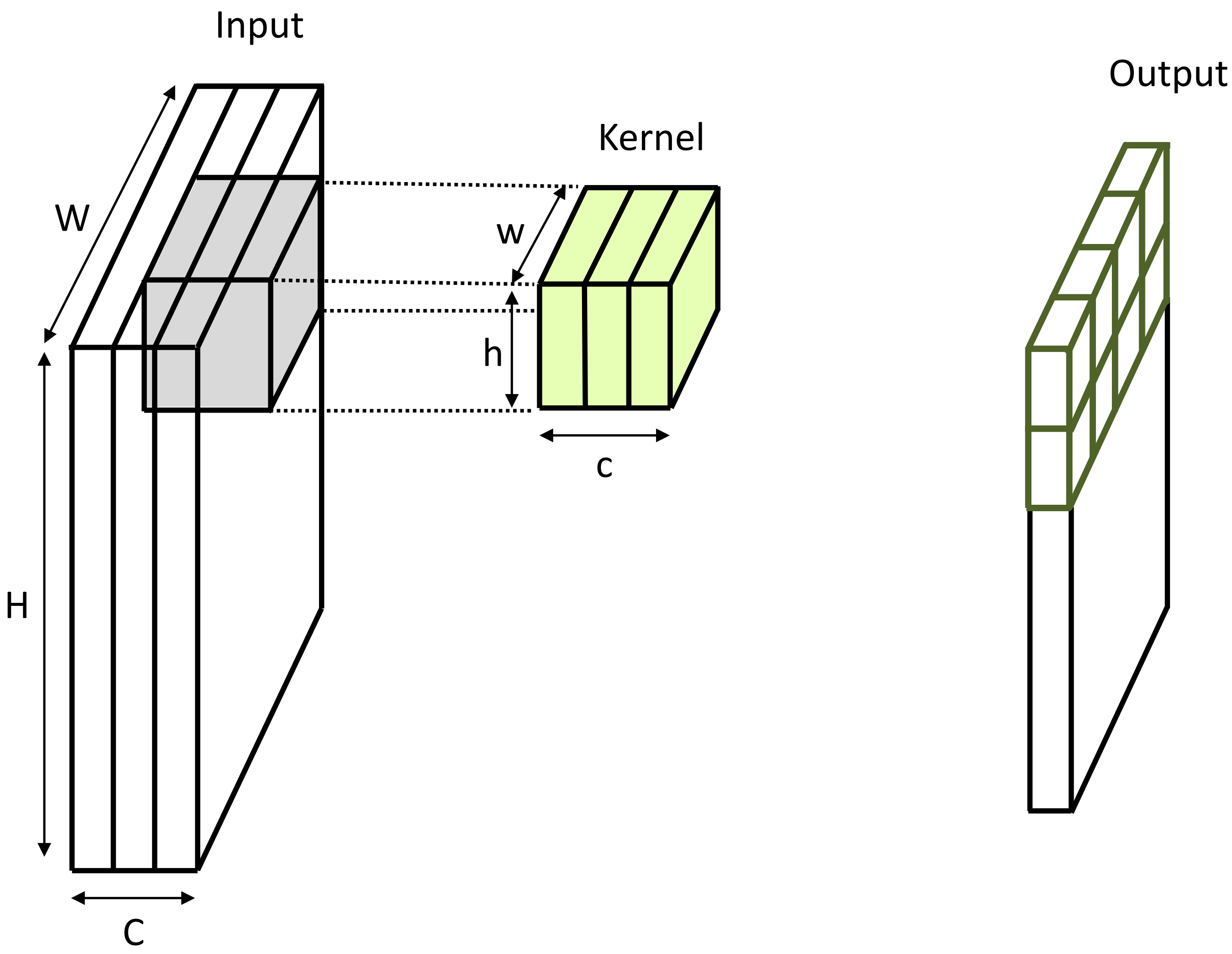
Source: Dr. Francois Fleuret at EPFL

Multi-channel 2D Convolution



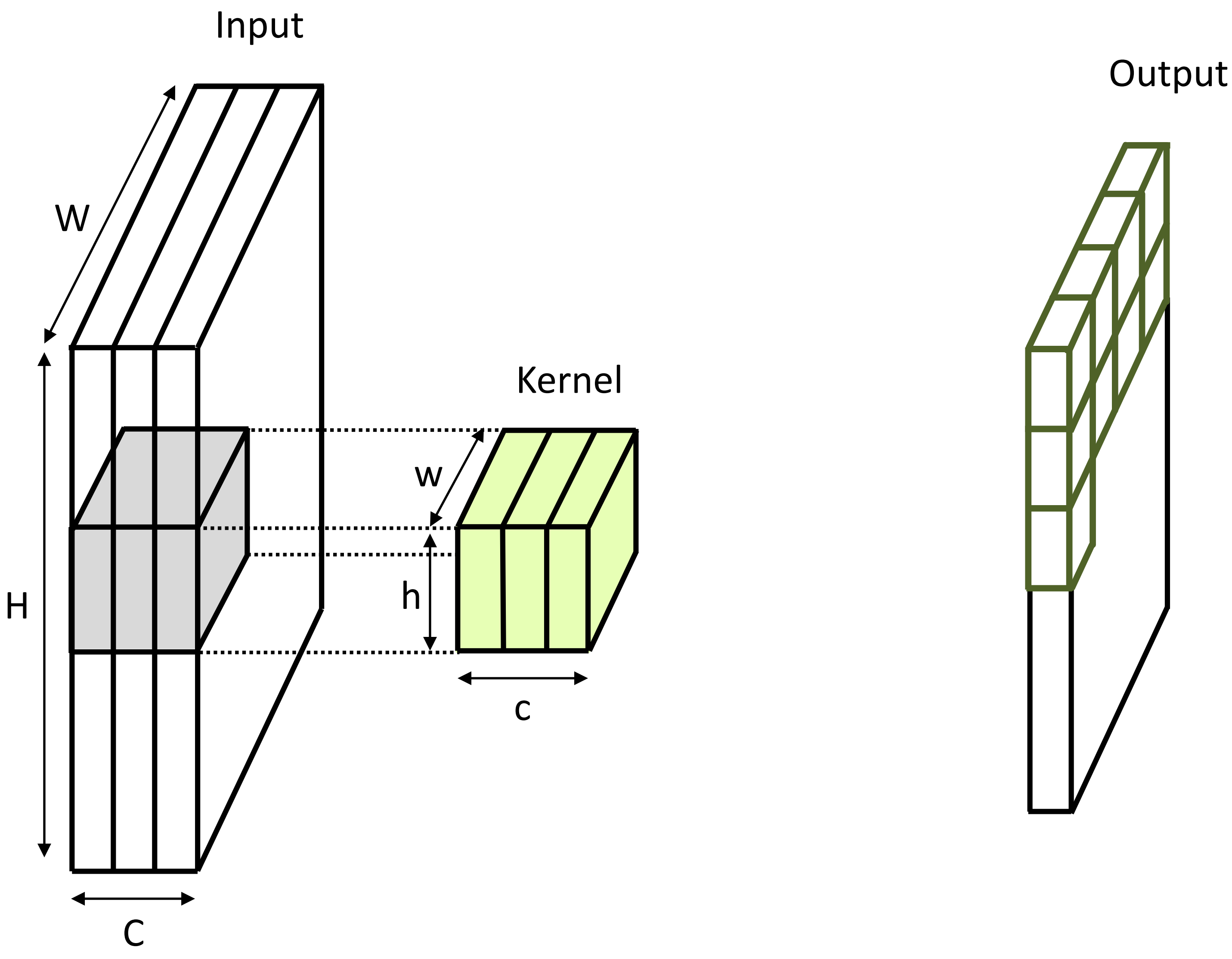
Source: Dr. Francois Fleuret at EPFL

Multi-channel 2D Convolution



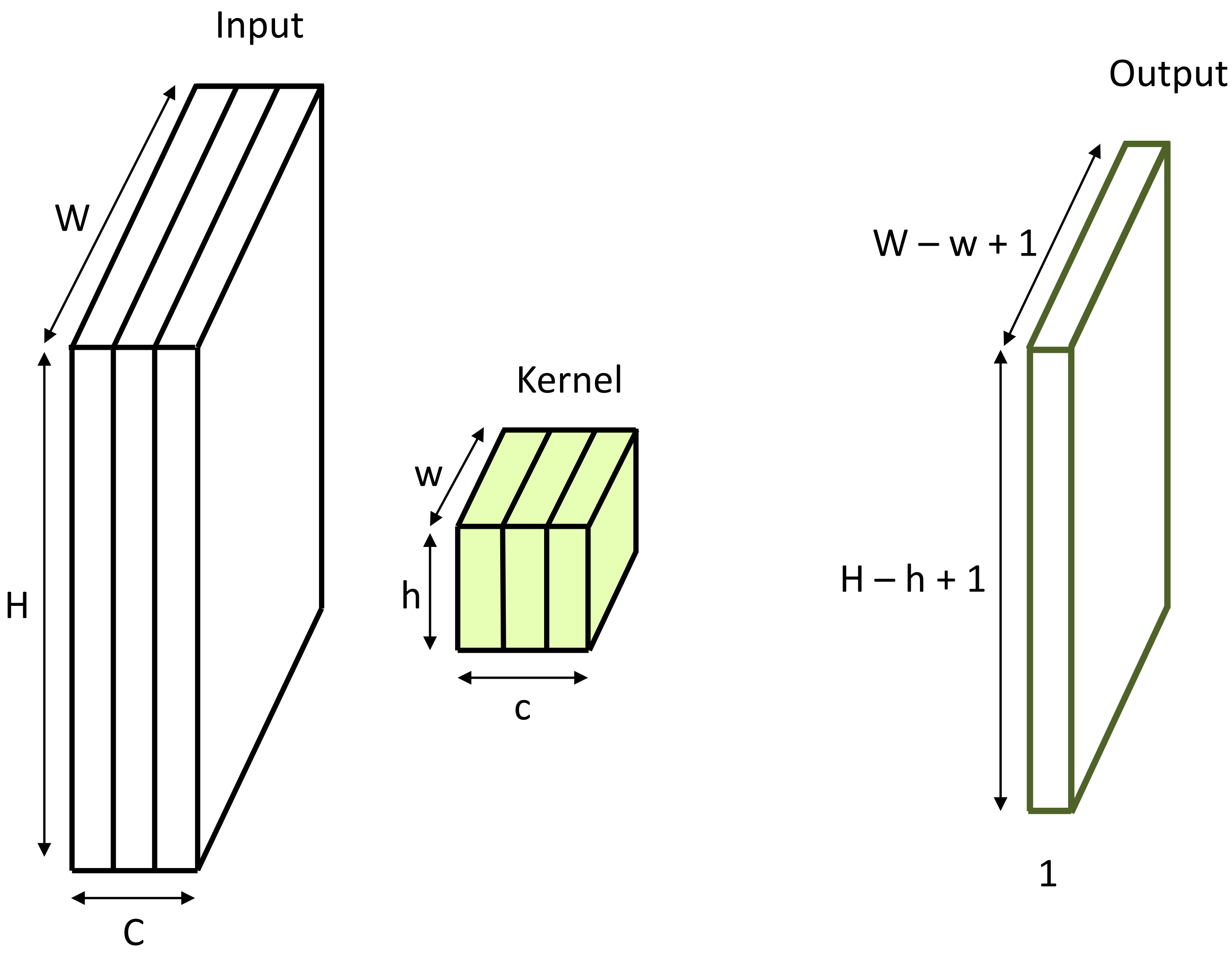
Source: Dr. Francois Fleuret at EPFL

Multi-channel 2D Convolution



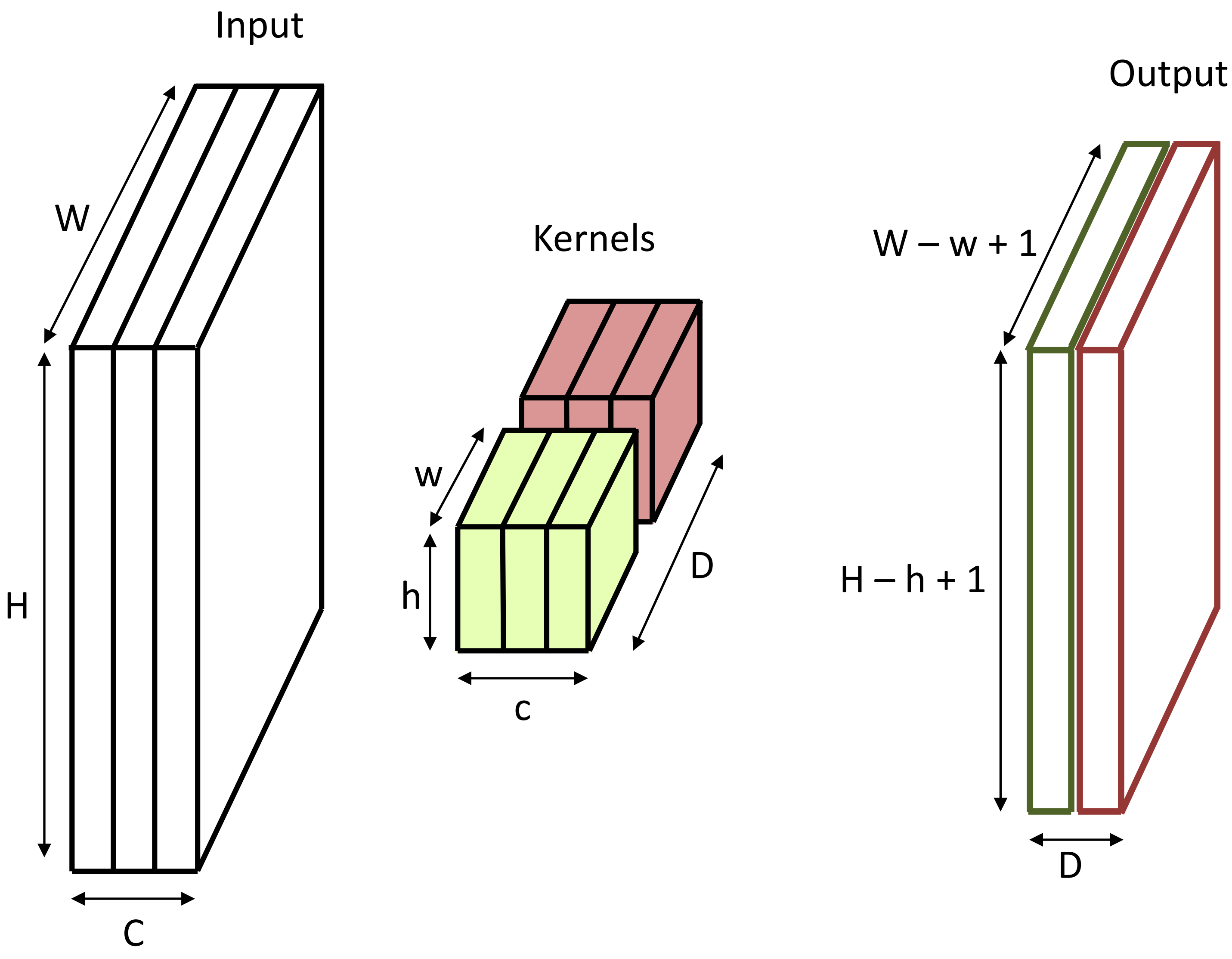
Source: Dr. Francois Fleuret at EPFL

Multi-channel 2D Convolution



Source: Dr. Francois Fleuret at EPFL

Multi-channel and Multi-kernel 2D Convolution



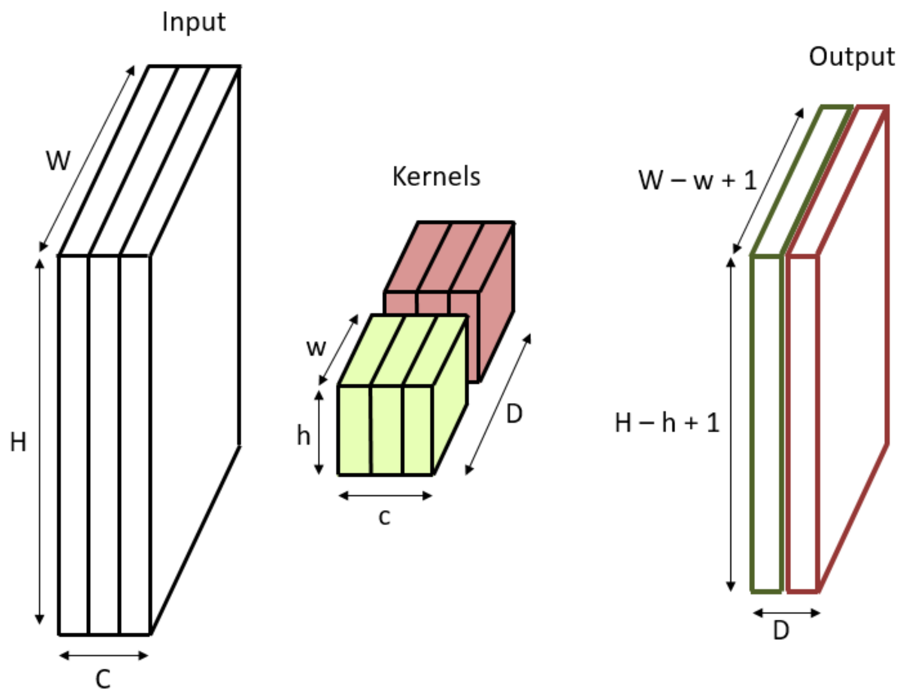
Source: Dr. Francois Fleuret at EPFL

Dealing with Shapes

- Activations or feature maps shape
  - Input (W^i, H^i, C)
  - Output (W^o, H^o, D)

- Kernel of Filter shape (w, h, C, D)
  - $w \times h$ Kernel size
  - C Input channels
  - D Output channels

- Numbers of parameters: $(w \times h \times C + 1) \times D$
  - bias

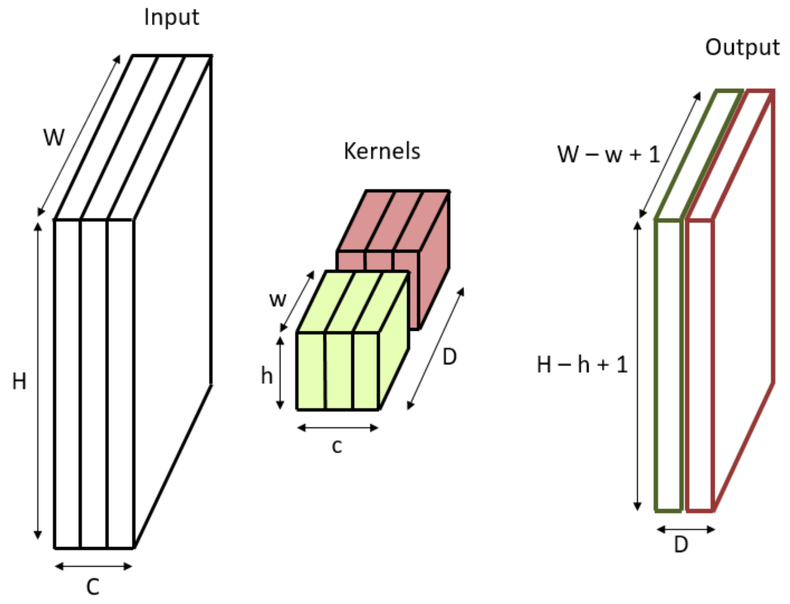


Source: Dr. Francois Fleuret at EPFL

Multi-channel 2D Convolution

- The kernel is not swiped across channels, just across rows and columns.
- Note that a convolution preserves the signal support structure.
  - A 1D signal is converted into a 1D signal, a 2D signal into a 2D, and neighboring parts of the input signal influence neighboring parts of the output signal.

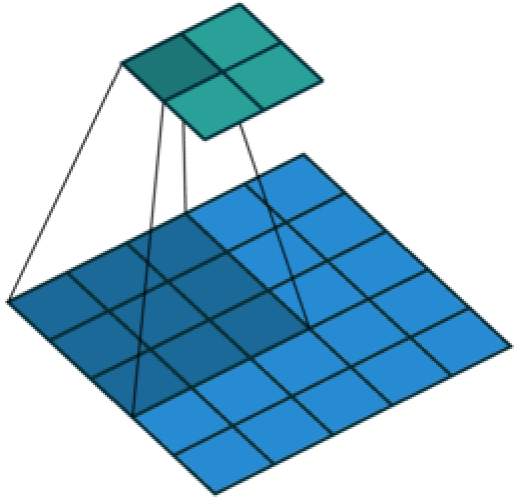
- We usually refer to one of the channels generated by a convolution layer as an activation map.
- The sub-area of an input map that influences a component of the output as the receptive field of the latter.



Padding and Stride

Strides

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

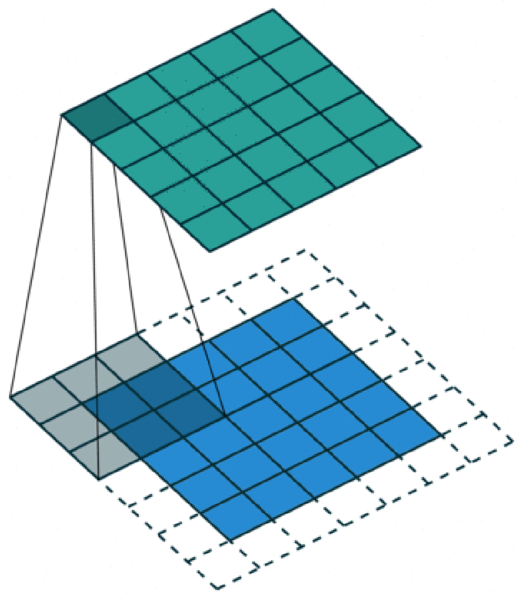


Example with kernel size 3×3 and a stride of 2 (image in blue)

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

Padding

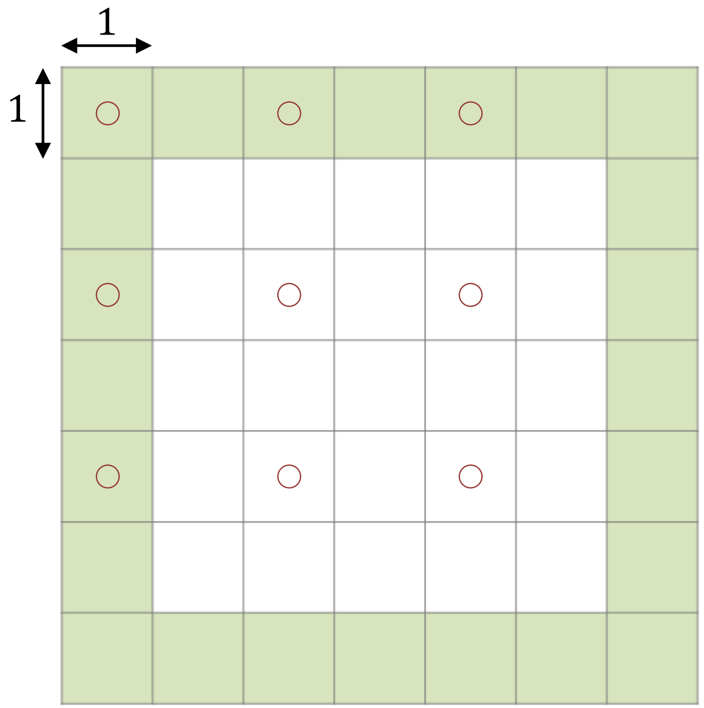
- Padding: artificially fill borders of image
- Useful to keep spatial dimension constant across filters
- Useful with strides and large receptive fields
- Usually fill with 0s



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

Padding and Stride

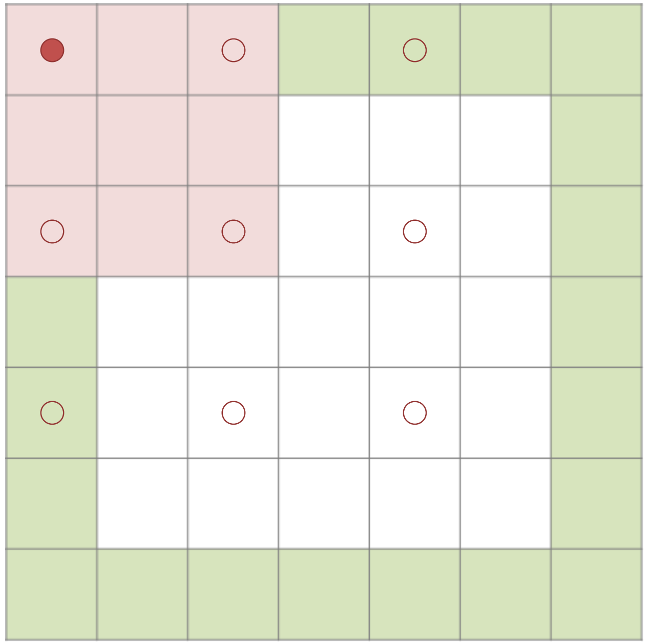
- Here with $5 \times 5 \times C$ as input, a padding of $(1\,,1)$, a stride of $(2\,,2)$



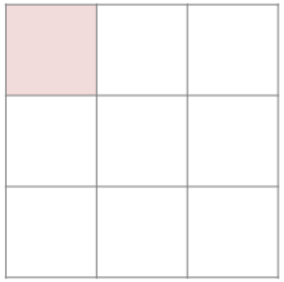
Input

Padding and Stride

- Here with $5 \times 5 \times C$ as input, a padding of $(1,1)$, a stride of $(2,2)$, and a kernel of size $3 \times 3 \times C$



Input

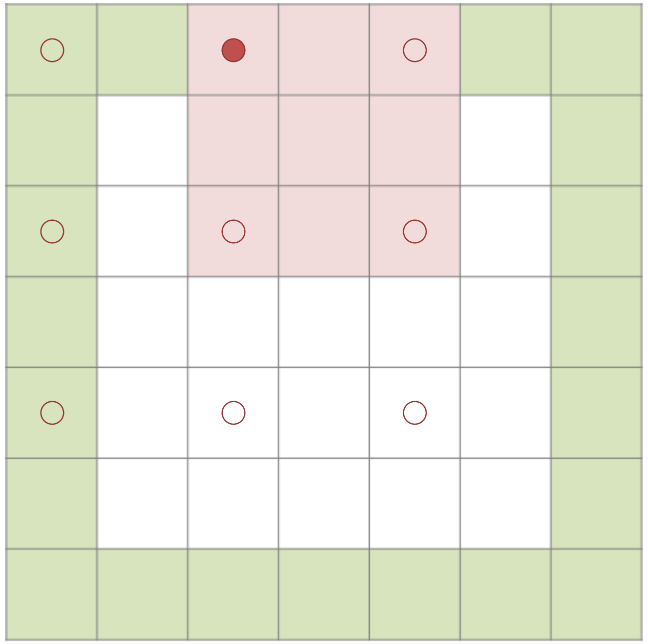


Output

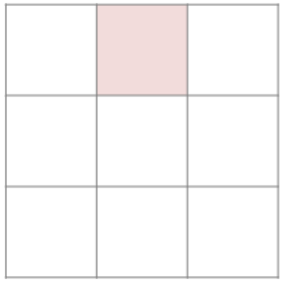
Source: Dr. Francois Fleuret at EPFL

Padding and Stride

- Here with $5 \times 5 \times C$ as input, a padding of $(1,1)$, a stride of $(2,2)$, and a kernel of size $3 \times 3 \times C$



Input



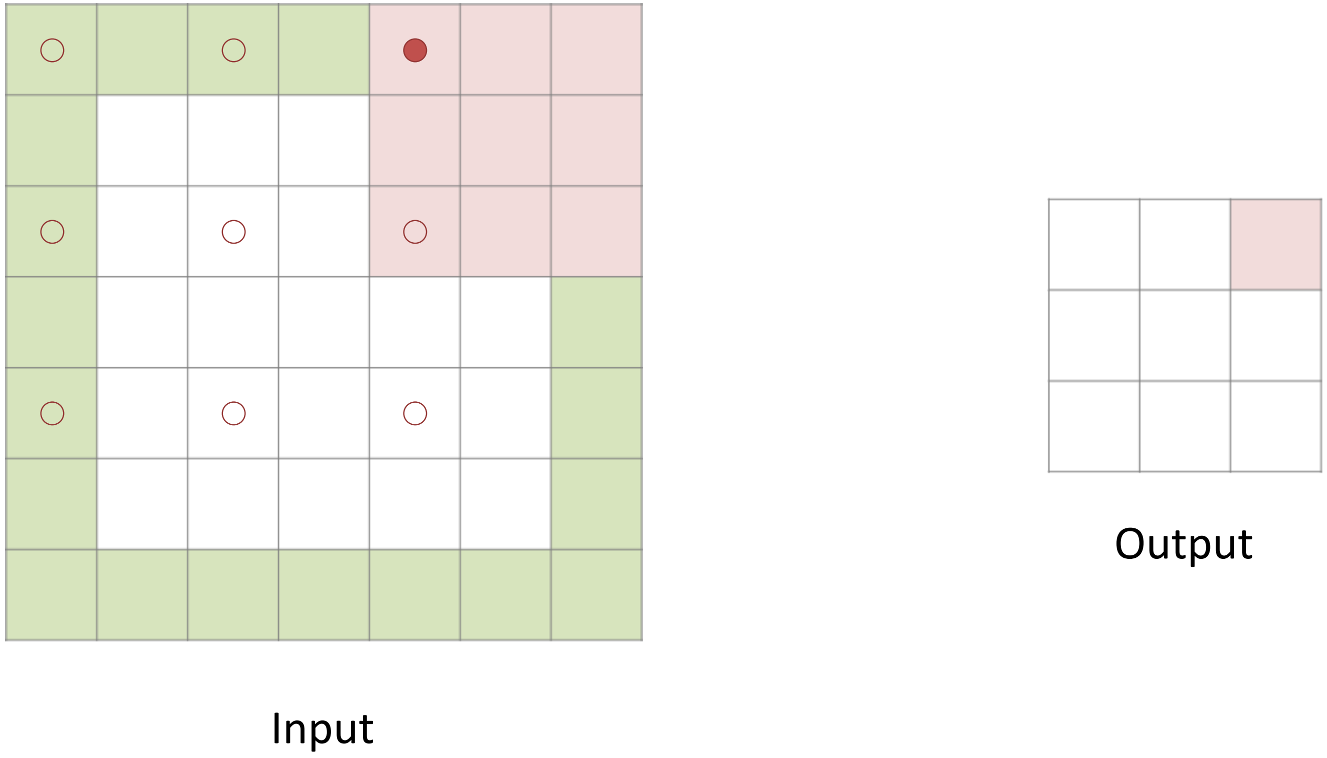
Output

POSTECH

Source: Dr. Francois Fleuret at EPFL

Padding and Stride

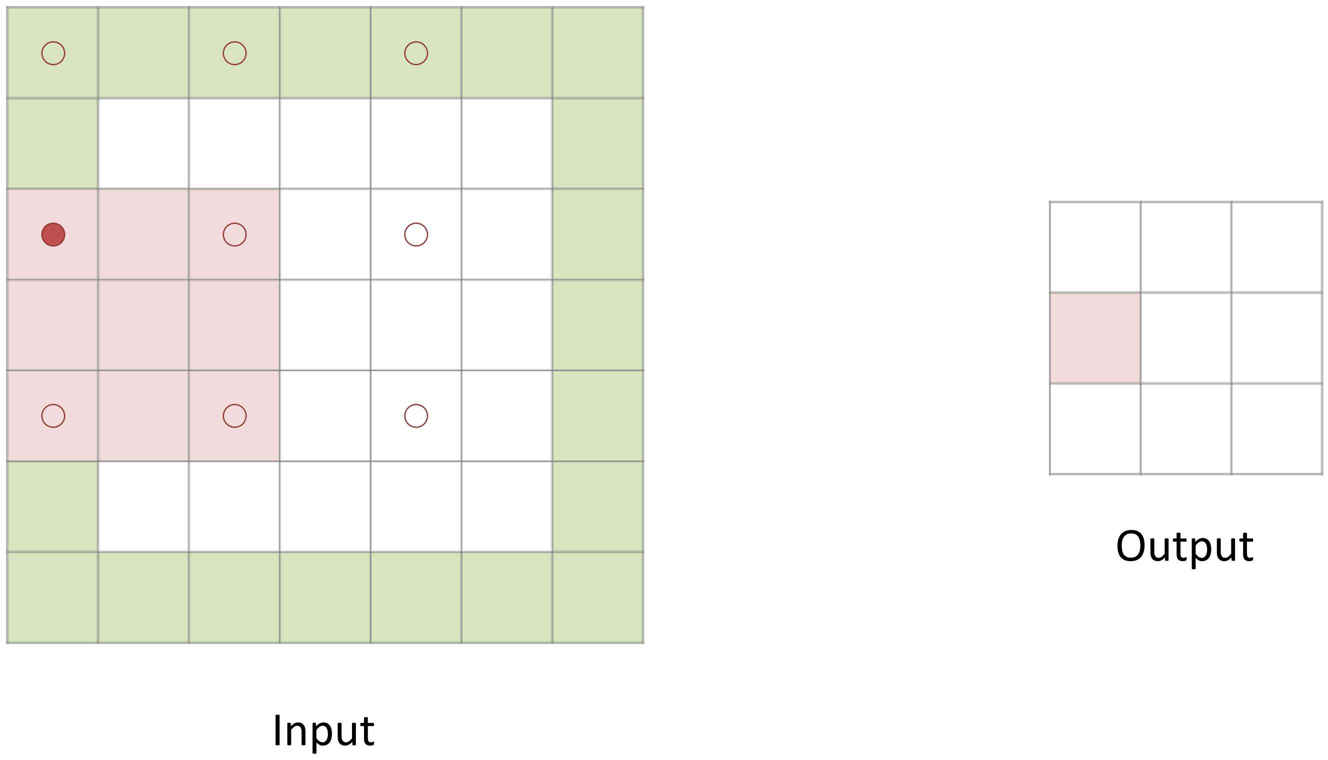
- Here with $5 \times 5 \times C$ as input, a padding of $(1,1)$, a stride of $(2,2)$, and a kernel of size $3 \times 3 \times C$



Source: Dr. Francois Fleuret at EPFL

Padding and Stride

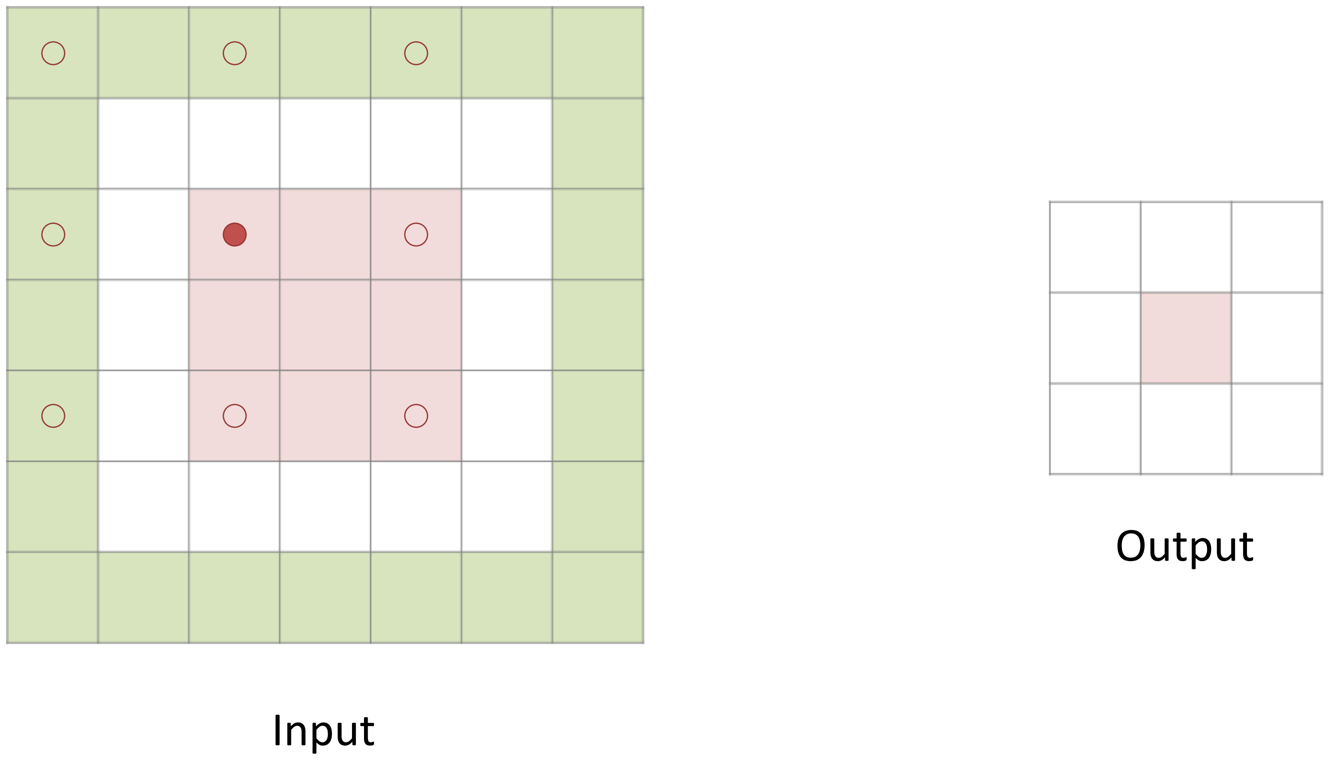
- Here with $5 \times 5 \times C$ as input, a padding of $(1,1)$, a stride of $(2,2)$, and a kernel of size $3 \times 3 \times C$



Source: Dr. Francois Fleuret at EPFL

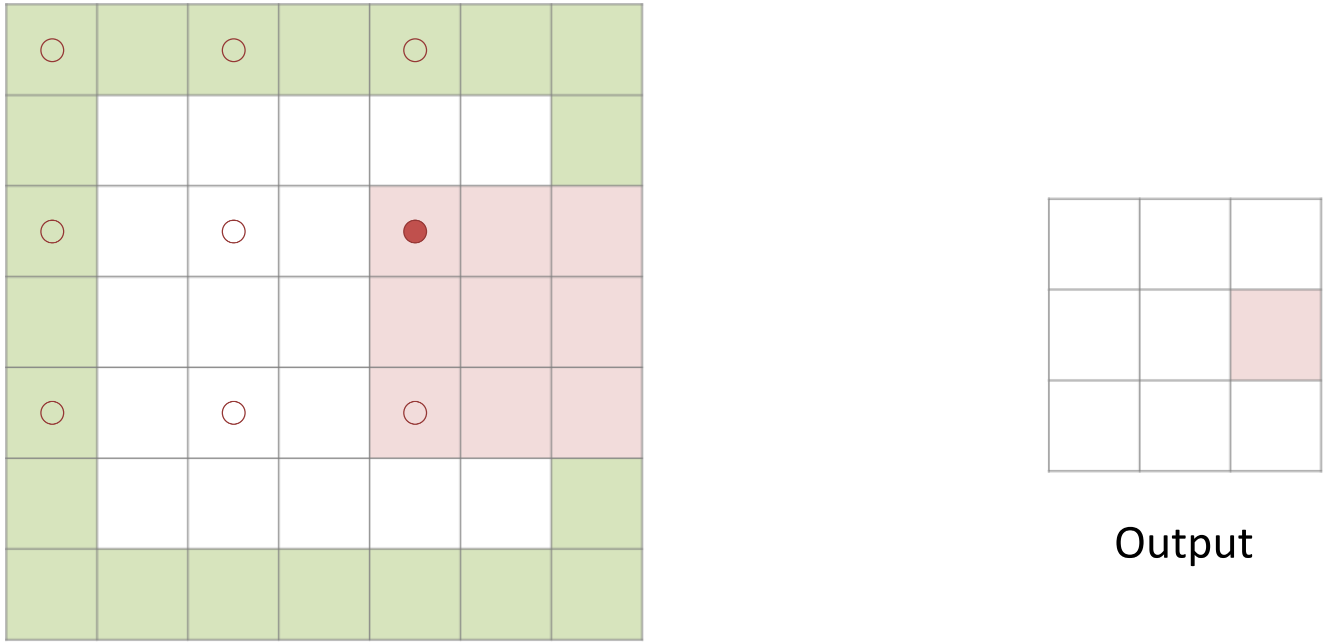
Padding and Stride

- Here with $5 \times 5 \times C$ as input, a padding of $(1{,}1)$, a stride of $(2{,}2)$, and a kernel of size $3 \times 3 \times C$



Padding and Stride

- Here with $5 \times 5 \times C$ as input, a padding of $(1{,}1)$, a stride of $(2{,}2)$, and a kernel of size $3 \times 3 \times C$

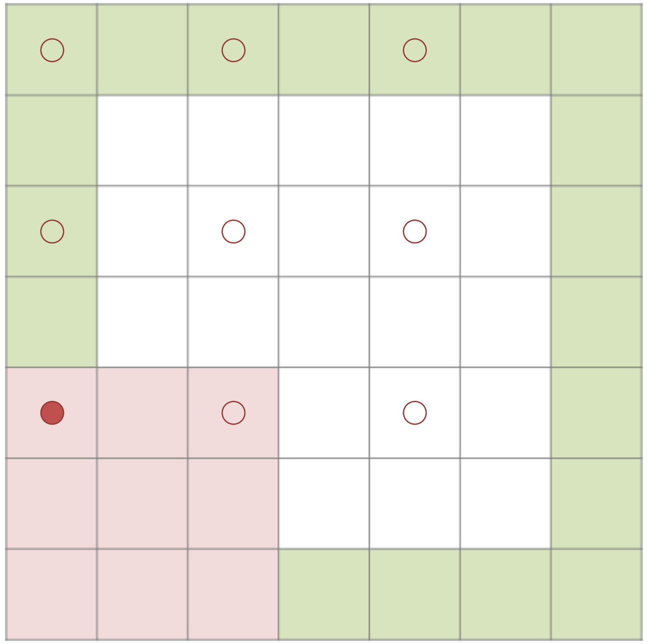


Input

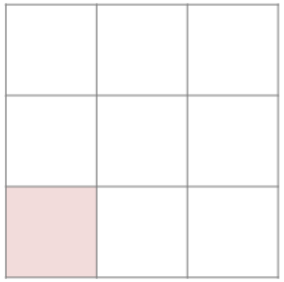
Output

Padding and Stride

- Here with $5 \times 5 \times C$ as input, a padding of $(1,1)$, a stride of $(2,2)$, and a kernel of size $3 \times 3 \times C$



Input

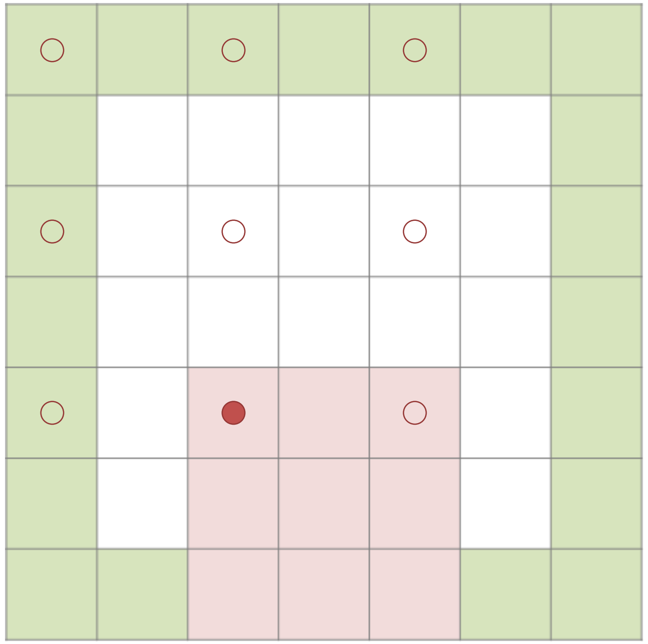


Output

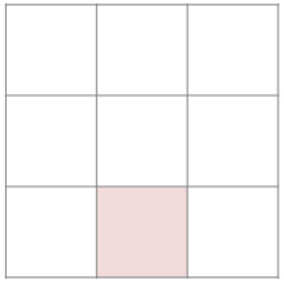
Source: Dr. Francois Fleuret at EPFL

Padding and Stride

- Here with $5 \times 5 \times C$ as input, a padding of $(1,1)$, a stride of $(2,2)$, and a kernel of size $3 \times 3 \times C$



Input

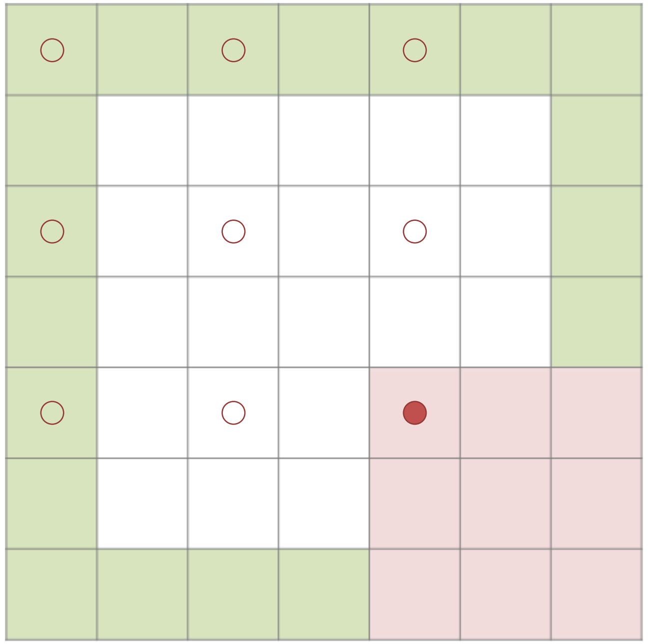


Output

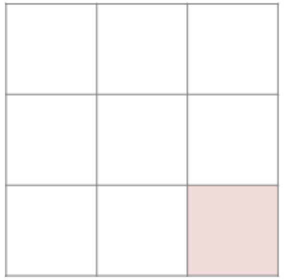
Source: Dr. Francois Fleuret at EPFL

Padding and Stride

- Here with $5 \times 5 \times C$ as input, a padding of $(1,1)$, a stride of $(2,2)$, and a kernel of size $3 \times 3 \times C$

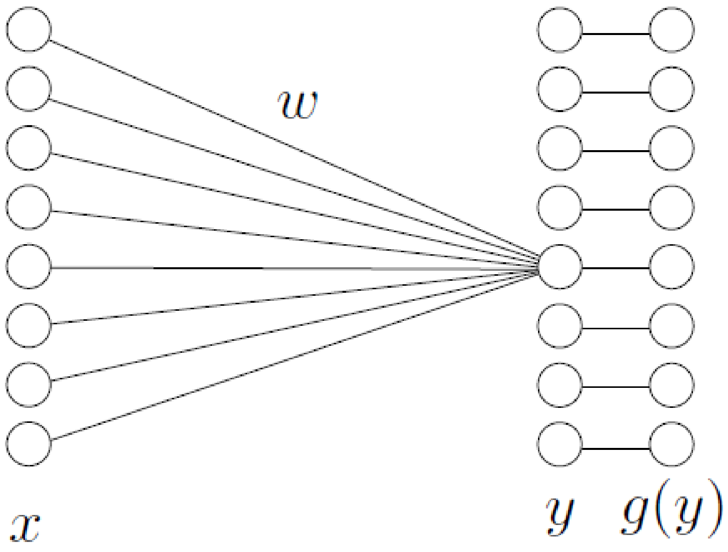


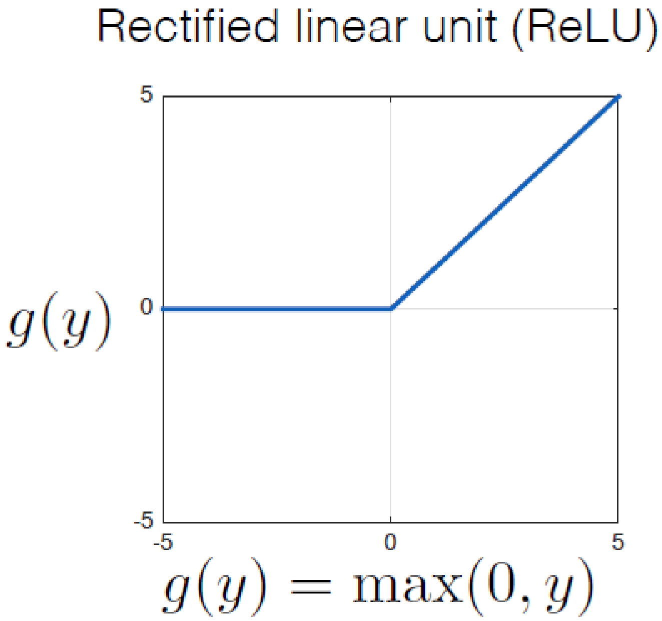
Input



Output

Nonlinear Activation Function



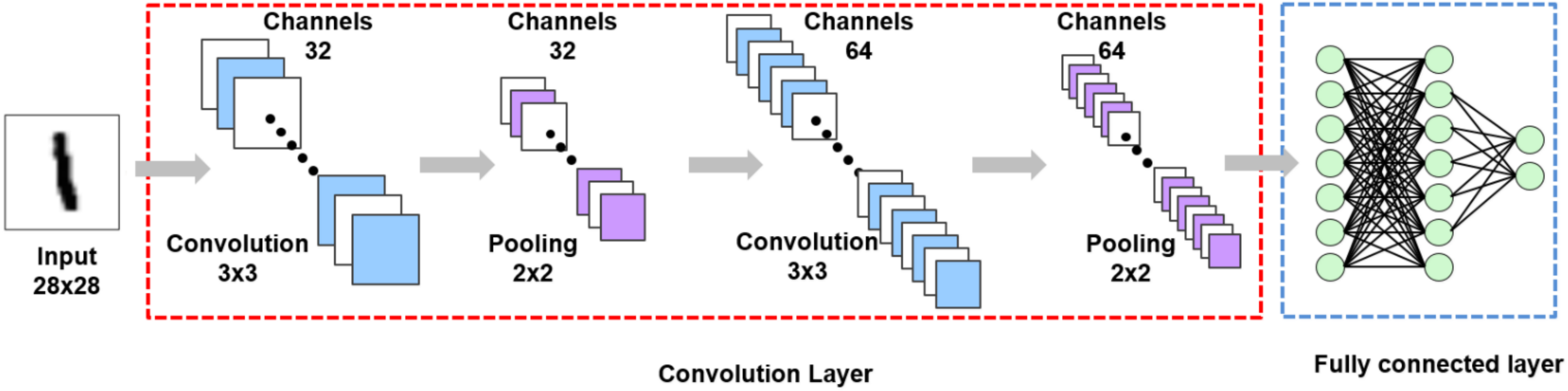


POSTECH

CNN in TensorFlow

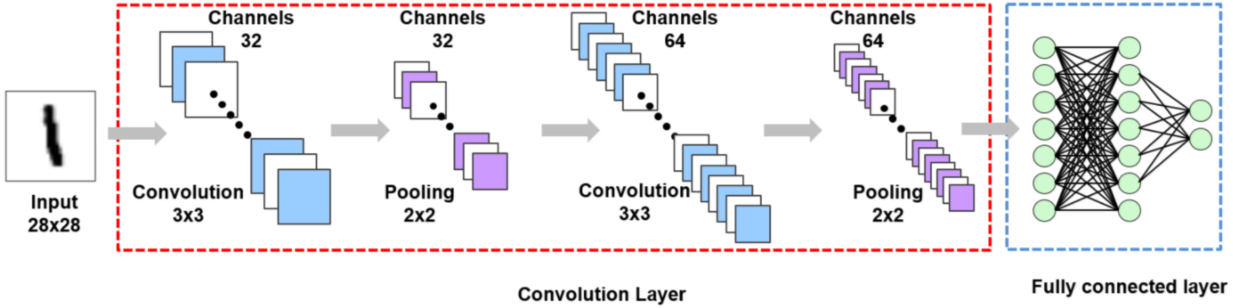
Lab: CNN with TensorFlow

- 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

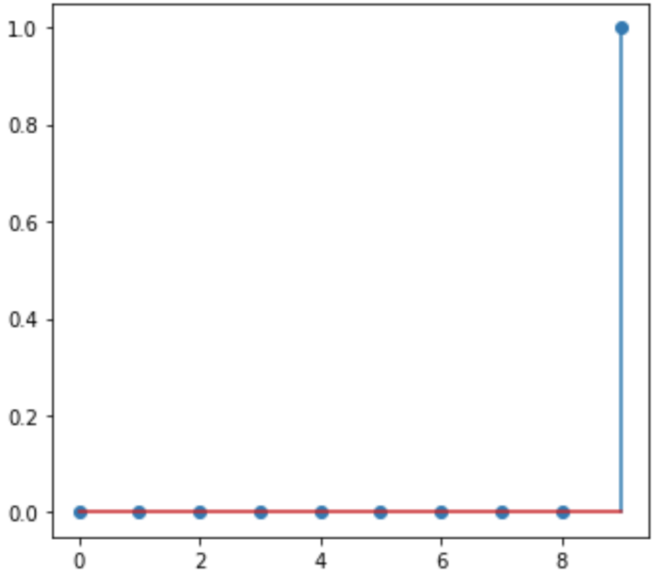
```
model.compile(optimizer = 'adam',
              loss = 'sparse_categorical_crossentropy',
              metrics = ['accuracy'])
```

```
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
```