Implementing ANNs with TensorFlow

Session 06 - Convolutional Neural Networks

Last Week

- We talked about how to encode certain inputs (e.g. images) and outputs (e.g. classes).

- In the homework you implemented a handwritten digit classifier for the MNIST dataset.

- But you had to reshape the image, because MLP can only process vectors:

$$\vec{x} \in [0,255]^{28\times28} \longrightarrow \vec{x}_{reshaped} \in [0,255]^{784}$$

- This is unfortunate as you loose important information about the spatial structure of the input.

Agenda

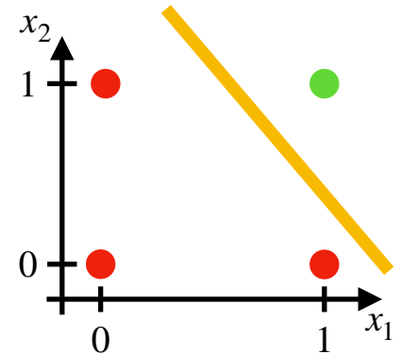
1. Features
2. Cross-correlation
3. Convolutional Neural Network

1. Convolutional Layer
2. Pooling Layer
3. Classification Network

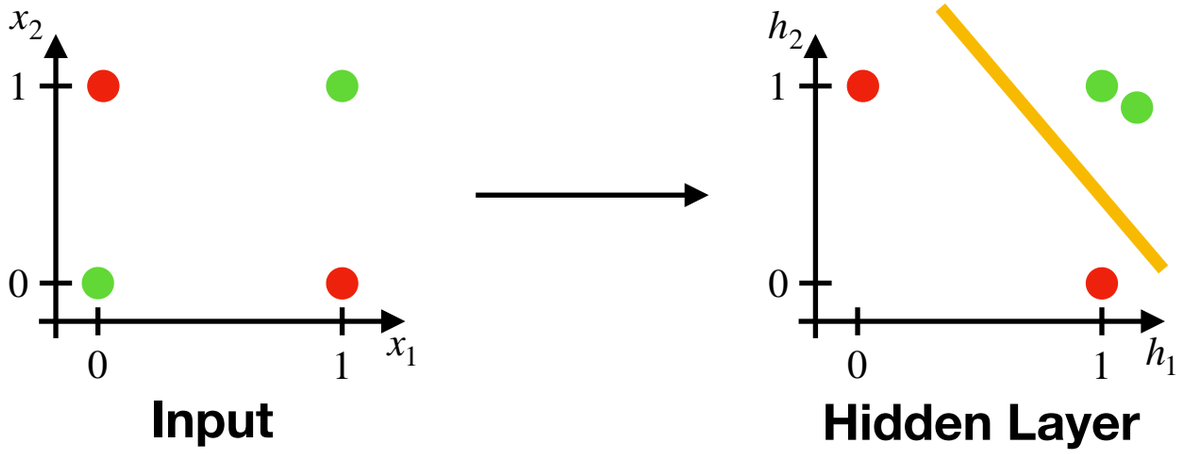
Features

What is an MLP doing?

- We saw that a perceptron learns a linear boundary.



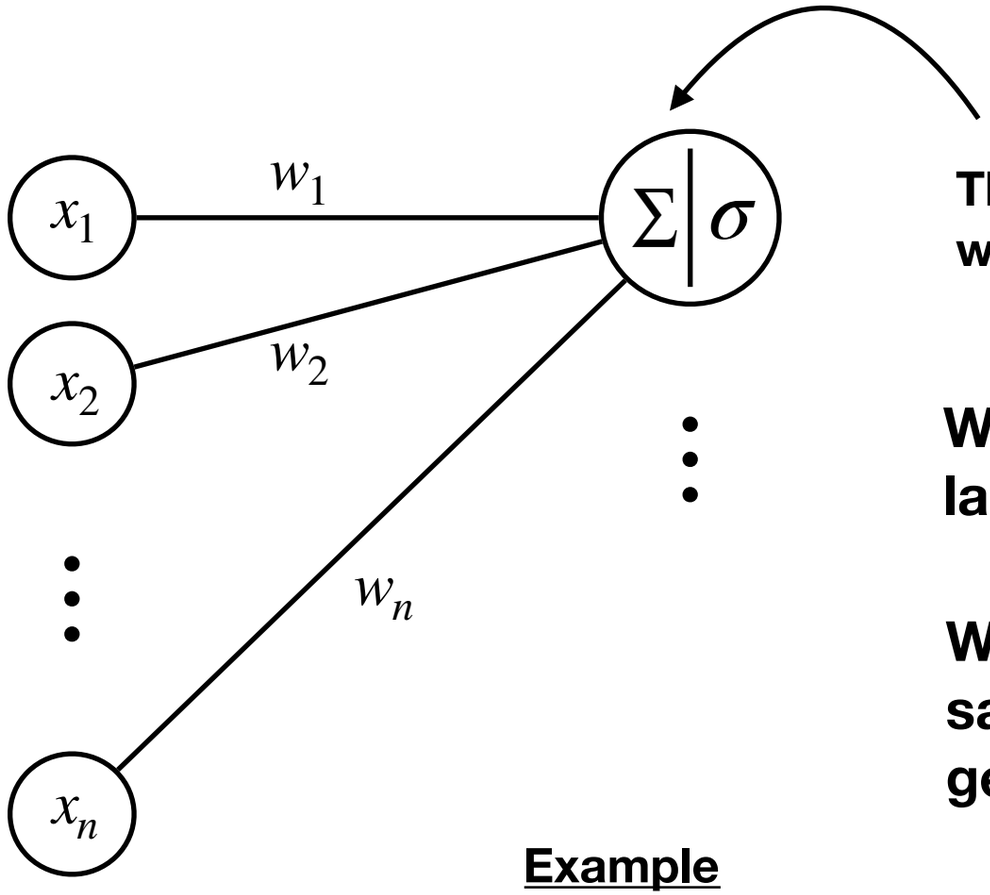
- An MLP can solve non-linear problems, because it can represent the input data, such that it becomes a linear problem



- But what is the network doing while representing the input data?

Feature Extraction

- You can interpret each neuron as a <u>feature extractor</u>.



The neuron gets most excited when the dot product is largest.

When is the dot product of two vectors largest?

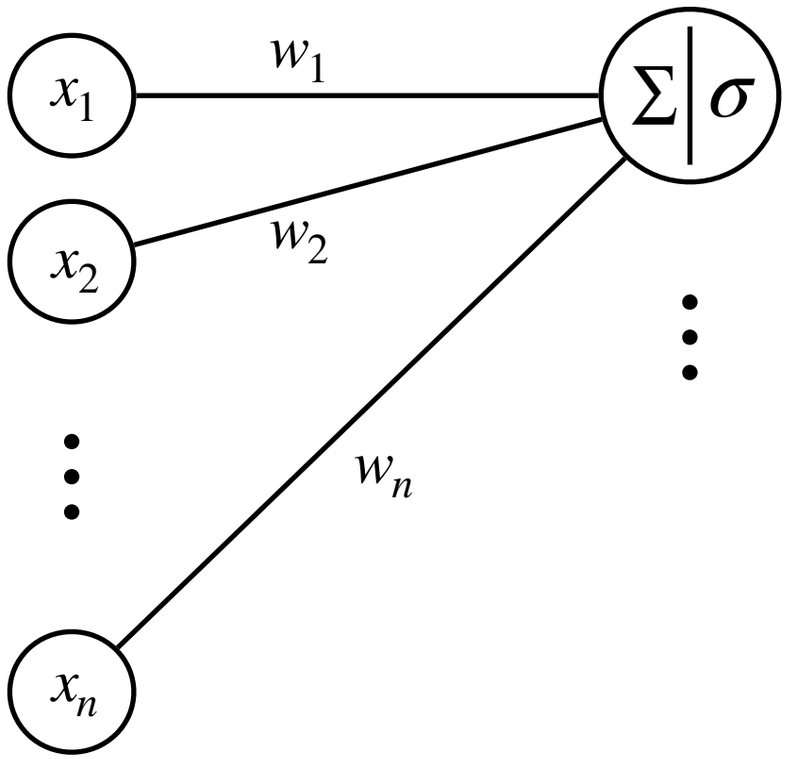
When the two vectors show to the same direction. The less similar they get, the lower is the dot product.

<u>Example</u>

$$\begin{pmatrix}1\\0\end{pmatrix} \cdot \begin{pmatrix}1\\0\end{pmatrix} = 1 \quad \begin{pmatrix}1\\0\end{pmatrix} \cdot \begin{pmatrix}0\\1\end{pmatrix} = 0 \quad \begin{pmatrix}1\\0\end{pmatrix} \cdot \begin{pmatrix}-1\\0\end{pmatrix} = -1$$

Feature Extraction

- You can interpret each neuron as a <u>feature extractor</u>.



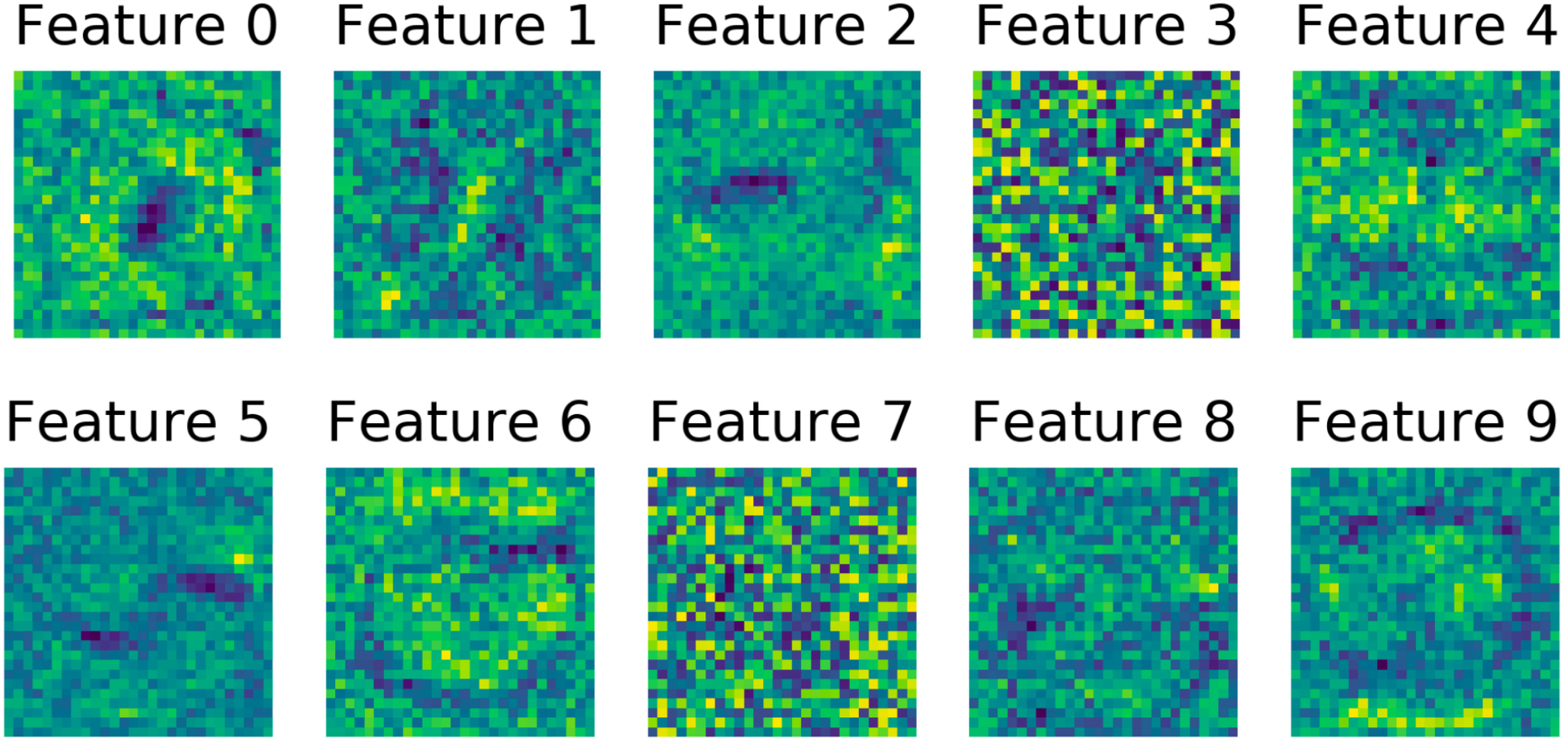
That means the more similar the input is to the weight vector, the more excited the neuron gets.

Thus we can say the neuron is looking for the **feature** $\overrightarrow{w}$ in the **input** $\overrightarrow{x}$.

Global Features

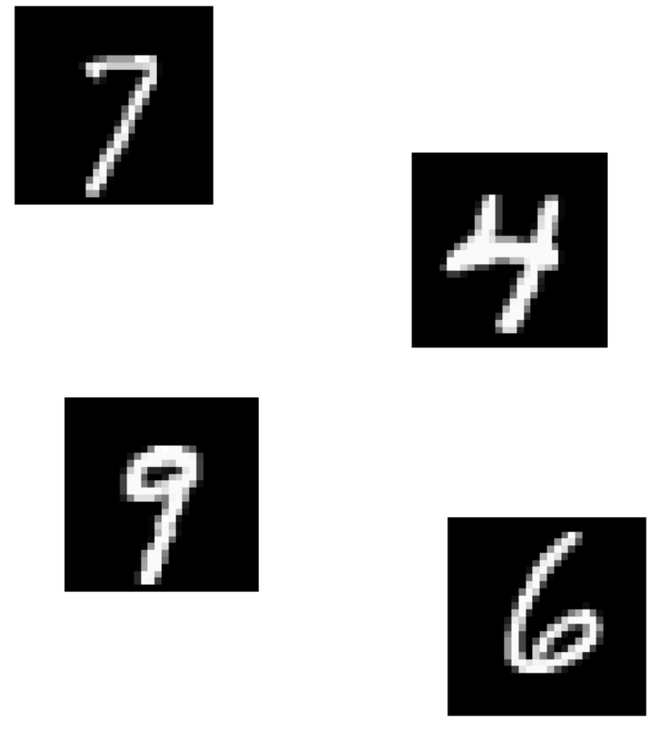
- This way the network extracts global features (features of the whole input).

Example: Visualization of global MNIST features



Local Features

- But what if the data actually has a spatial structure (as images)?
- Shouldn’t we look for local features rather than global ones?
- Example MNIST, look for:
  - Curves
  - Horizontal lines, vertical lines
  - Circles



[ylc]

Cross Correlation

Cross Correlation

How to look for local features?

For that we use a technique called cross correlation or sliding dot product.

Example: Search feature $(2 \quad 0 \quad 2 \quad 1)$ in

$$(1 \quad 3 \quad 0 \quad 1 \quad 2 \quad 2 \quad 0 \quad 2 \quad 1 \quad 0 \quad 2 \quad 1)$$

$$(2 \quad 0 \quad 2 \quad 1) \qquad\qquad 1 \cdot 2 + 3 \cdot 0 + 0 \cdot 2 + 1 \cdot 1 = 3$$

Cross Correlation

How to look for local features?

For that we use a technique called cross correlation or sliding dot product.

Example: Search feature $(2 \quad 0 \quad 2 \quad 1)$ in

$$(1 \quad 3 \quad 0 \quad 1 \quad 2 \quad 2 \quad 0 \quad 2 \quad 1 \quad 0 \quad 2 \quad 1)$$

$(2 \quad 0 \quad 2 \quad 1)$ $\qquad 1 \cdot 2 + 3 \cdot 0 + 0 \cdot 2 + 1 \cdot 1 = 3$

$(2 \quad 0 \quad 2 \quad 1)$ $\qquad 3 \cdot 2 + 0 \cdot 0 + 1 \cdot 2 + 2 \cdot 1 = 10$

Cross Correlation

How to look for local features?

For that we use a technique called cross correlation or sliding dot product.

Example: Search feature $(2 \quad 0 \quad 2 \quad 1)$ in

$$(1 \quad 3 \quad 0 \quad 1 \quad 2 \quad 2 \quad 0 \quad 2 \quad 1 \quad 0 \quad 2 \quad 1)$$

$$(2 \quad 0 \quad 2 \quad 1) \qquad 1 \cdot 2 + 3 \cdot 0 + 0 \cdot 2 + 1 \cdot 1 = 3$$

$$(2 \quad 0 \quad 2 \quad 1) \qquad 3 \cdot 2 + 0 \cdot 0 + 1 \cdot 2 + 2 \cdot 1 = 10$$

$$(2 \quad 0 \quad 2 \quad 1) \qquad 0 \cdot 2 + 1 \cdot 0 + 2 \cdot 2 + 2 \cdot 1 = 6$$

Cross Correlation

How to look for local features?

For that we use a technique called cross correlation or sliding dot product.

Example: Search feature $(2 \quad 0 \quad 2 \quad 1)$ in

$$(1 \quad 3 \quad 0 \quad 1 \quad 2 \quad 2 \quad 0 \quad 2 \quad 1 \quad 0 \quad 2 \quad 1)$$

$$(2 \quad 0 \quad 2 \quad 1) \qquad 1 \cdot 2 + 3 \cdot 0 + 0 \cdot 2 + 1 \cdot 1 = 3$$

$$(2 \quad 0 \quad 2 \quad 1) \qquad 3 \cdot 2 + 0 \cdot 0 + 1 \cdot 2 + 2 \cdot 1 = 10$$

$$(2 \quad 0 \quad 2 \quad 1) \qquad 0 \cdot 2 + 1 \cdot 0 + 2 \cdot 2 + 2 \cdot 1 = 6$$

$$(2 \quad 0 \quad 2 \quad 1) \qquad 6$$

$$(2 \quad 0 \quad 2 \quad 1) \qquad \vdots \qquad 6$$

$$(2 \quad 0 \quad 2 \quad 1) \qquad 9$$

$$(2 \quad 0 \quad 2 \quad 1) \qquad 2$$

$$(2 \quad 0 \quad 2 \quad 1) \qquad 6$$

$$(2 \quad 0 \quad 2 \quad 1) \qquad 7$$

Cross Correlation

The result of the cross correlation of the input

$$(1 \quad 3 \quad 0 \quad 1 \quad 2 \quad 2 \quad 0 \quad 2 \quad 1 \quad 0 \quad 2 \quad 1)$$

with the feature

$$(2 \quad 0 \quad 2 \quad 1)$$

would thus result in

$$(3 \quad 10 \quad 6 \quad 6 \quad 6 \quad 9 \quad 2 \quad 6 \quad 7).$$

Cross Correlation

Note that cross correlation is not looking for the exact feature. It is looking for how strong a feature is in a certain part of the input!

Example: Search feature $(2 \quad 0 \quad 2 \quad 1)$ in

$$(1 \quad 3 \quad 0 \quad 1 \quad 2 \quad 2 \quad 0 \quad 2 \quad 1 \quad 0 \quad 2 \quad 1)$$

$(2 \quad 0 \quad 2 \quad 1)$ $\qquad 3 \cdot 2 + 0 \cdot 0 + 1 \cdot 2 + 2 \cdot 1 = 10$

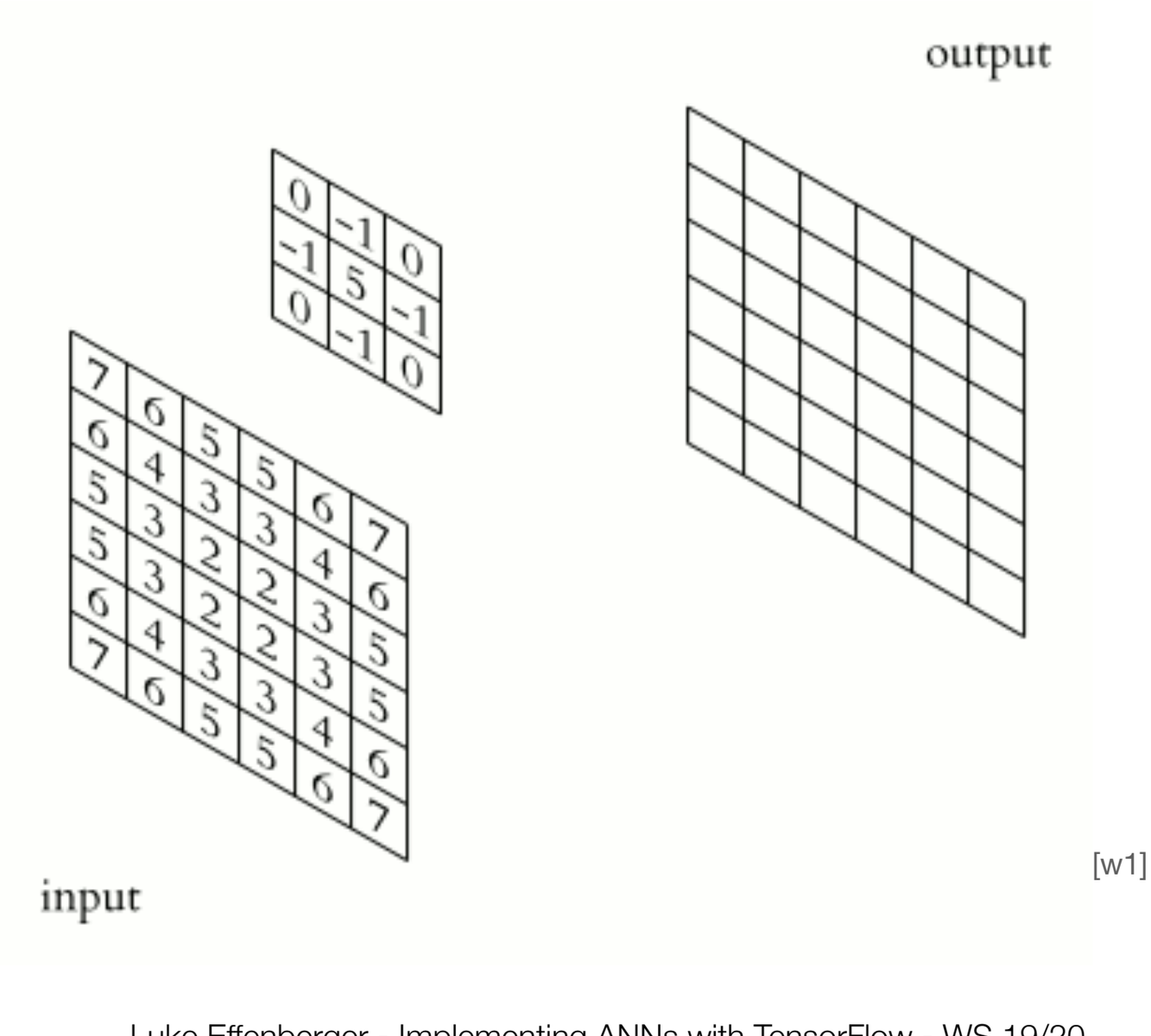
$(2 \quad 0 \quad 2 \quad 1)$ $\qquad 2 \cdot 2 + 0 \cdot 0 + 2 \cdot 2 + 1 \cdot 1 = 9$

Convolution

- In deep learning terms the cross-correlation operation is called a convolution.
- Actually convolutions and cross-correlations differ.
- Convolutions would transpose the feature vector, but that would not matter in DNNs.

2D Convolutional Operation

- Convolution also works in 2D.
- Shifting a 2D kernel over the 2D input produces a 2D output called a feature map.

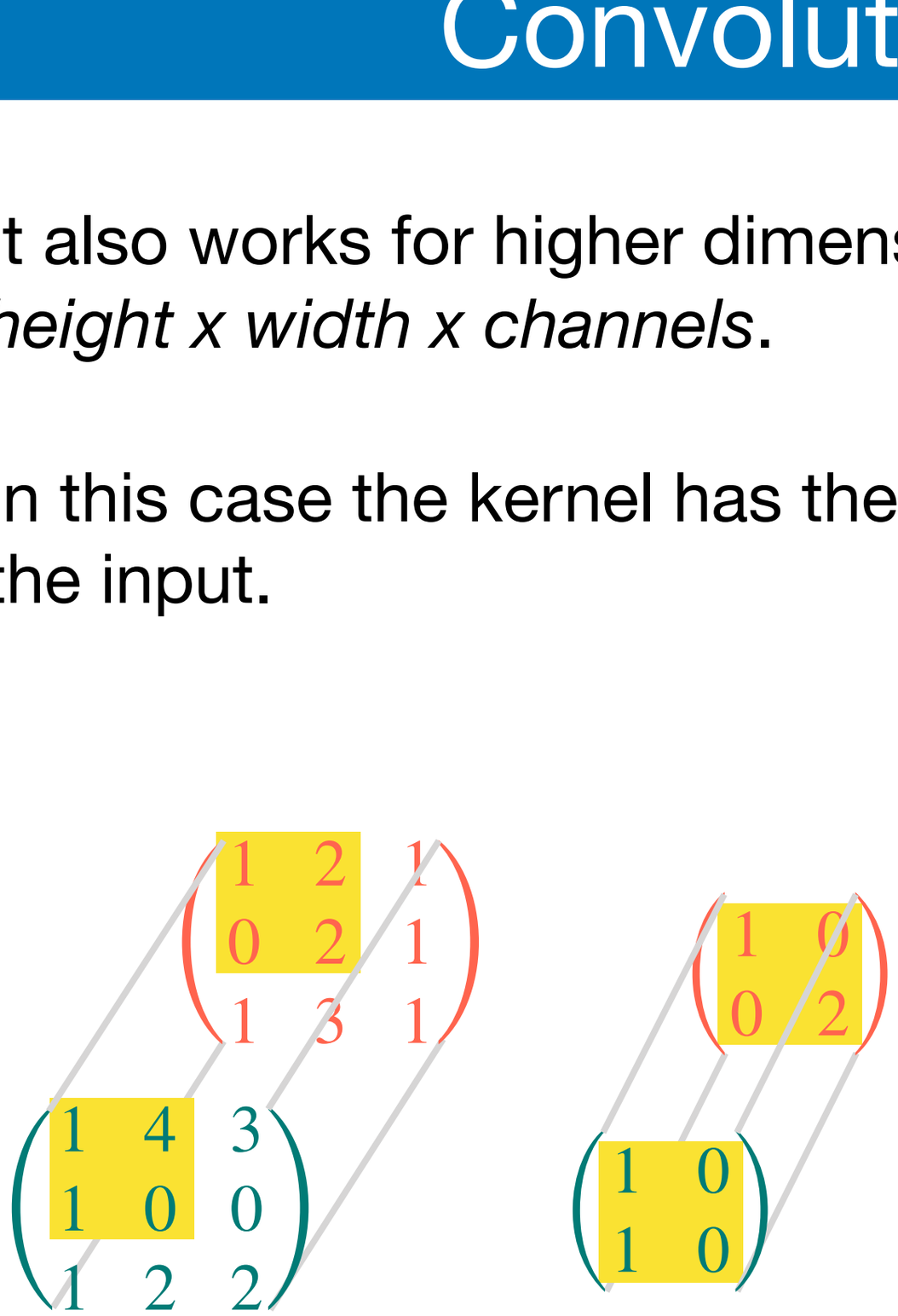


[w1]

Convolution in 3D

- It also works for higher dimensional inputs of shape e.g. *height x width x channels*.

- In this case the kernel has the same depth (=*channels)* as the input.



E.g.

$$
\begin{aligned}
&1 \cdot 1 \\
&+4 \cdot 0 \\
&+1 \cdot 1 \\
&+0 \cdot 0 \\
&+1 \cdot 1 \\
&+2 \cdot 0 \\
&+0 \cdot 0 \\
&+2 \cdot 2 = 7
\end{aligned}
$$

Convolutional Neural Network

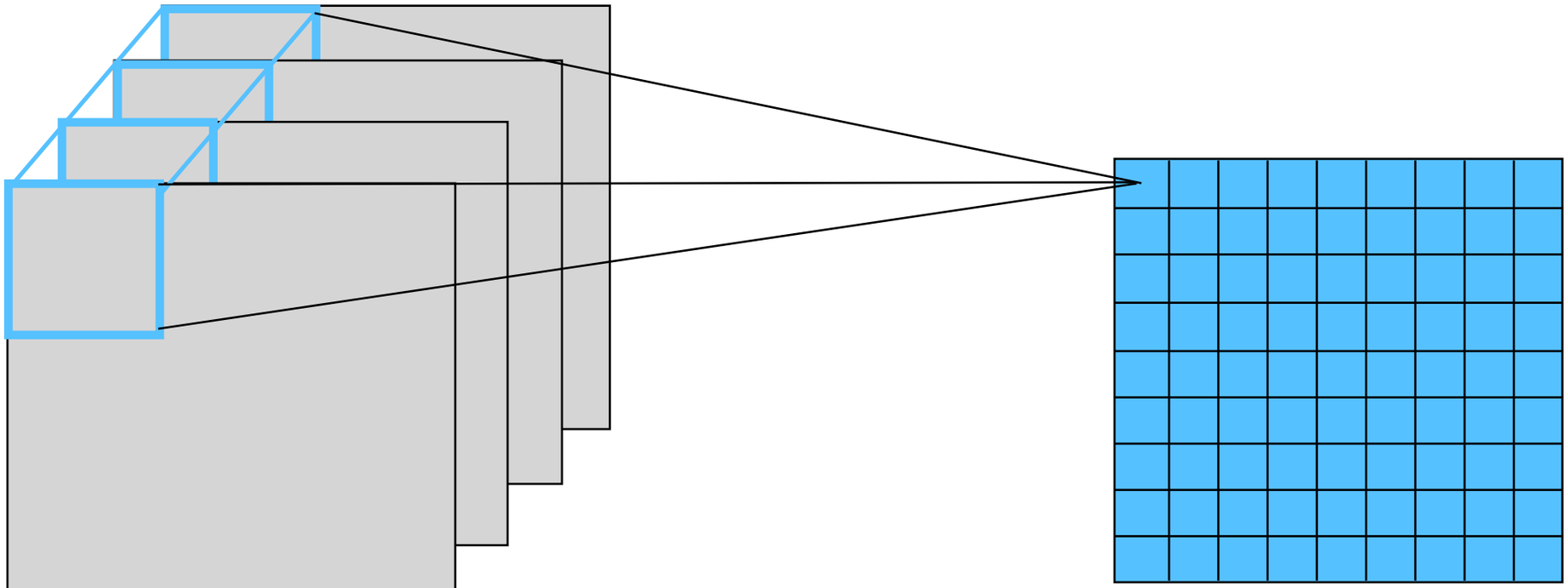
Convolutional Neural Network

- A convolutional neural network (CNN) is a deep neural network, which is mainly built of convolutional layers.

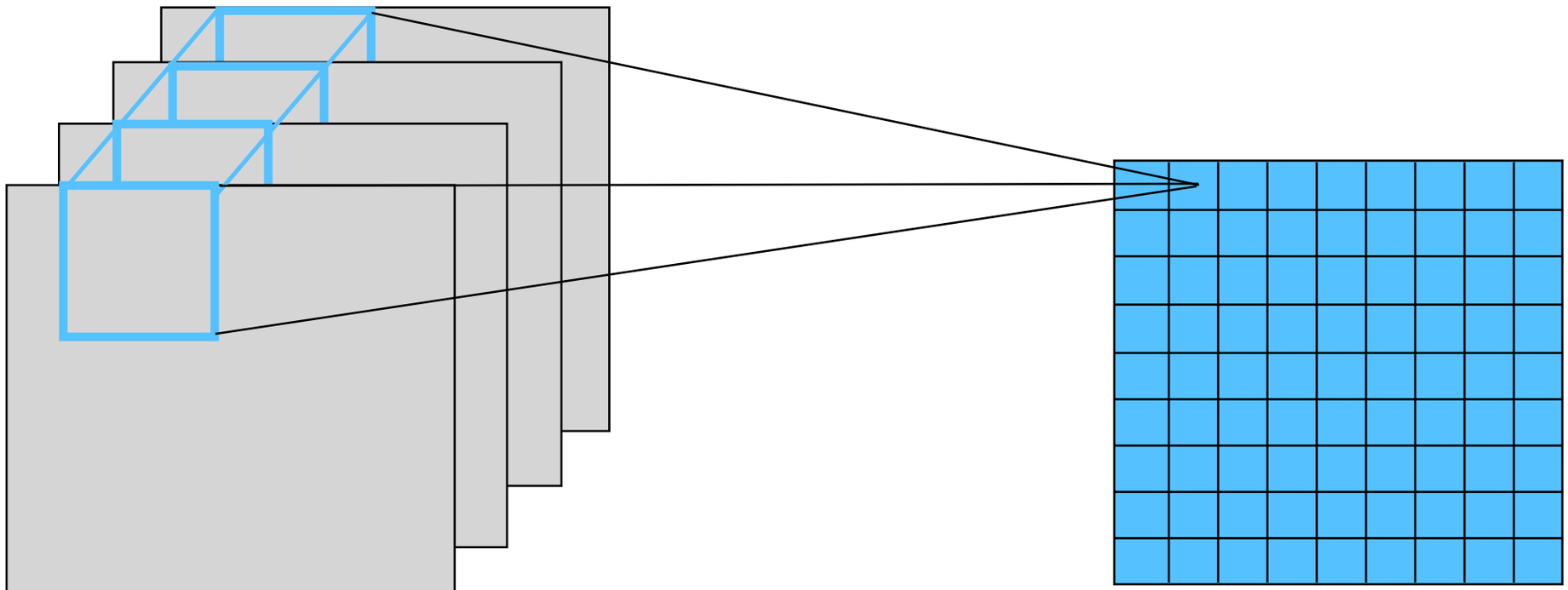
- A convolutional layer implements the previously described convolutional operation.

Convolutional Layer

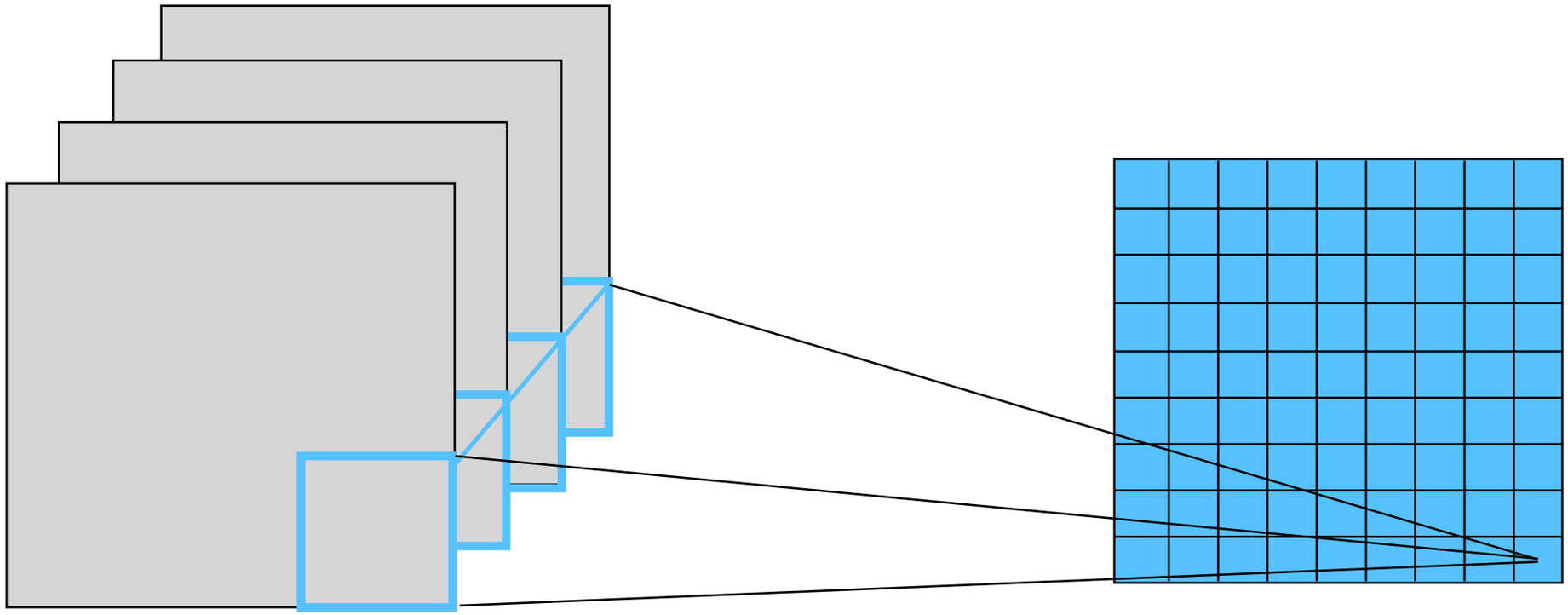
Convolutional Layer



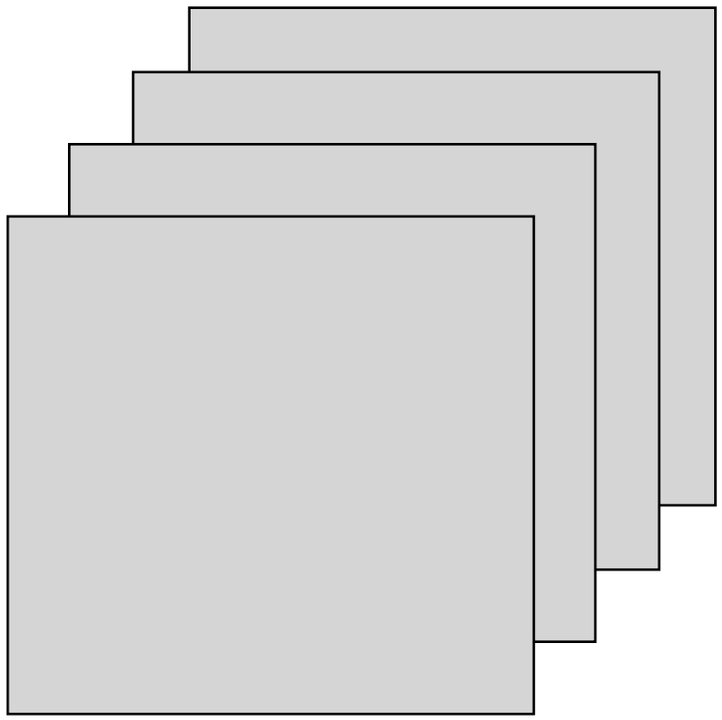
Convolutional Layer



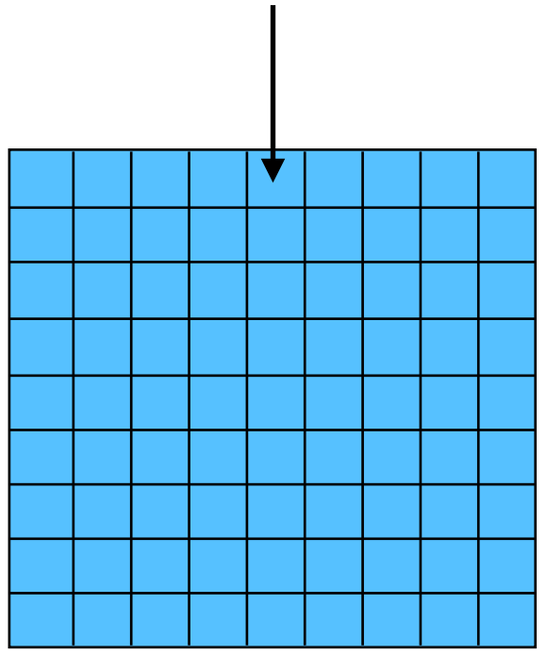
Convolutional Layer



Convolutional Layer

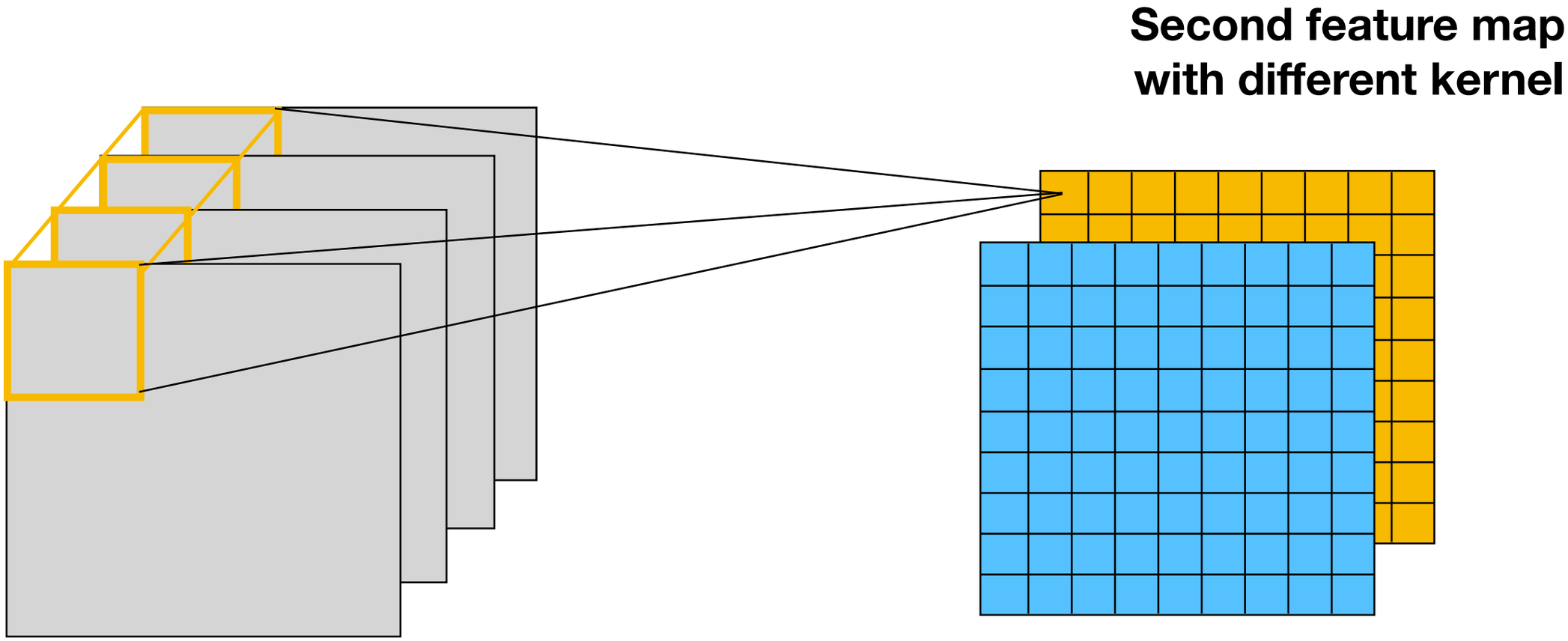


Each of these can be seen as a neuron. Each is using the same kernel (weights) to look for the same feature in different positions of the input.



Feature Map

Convolutional Layer

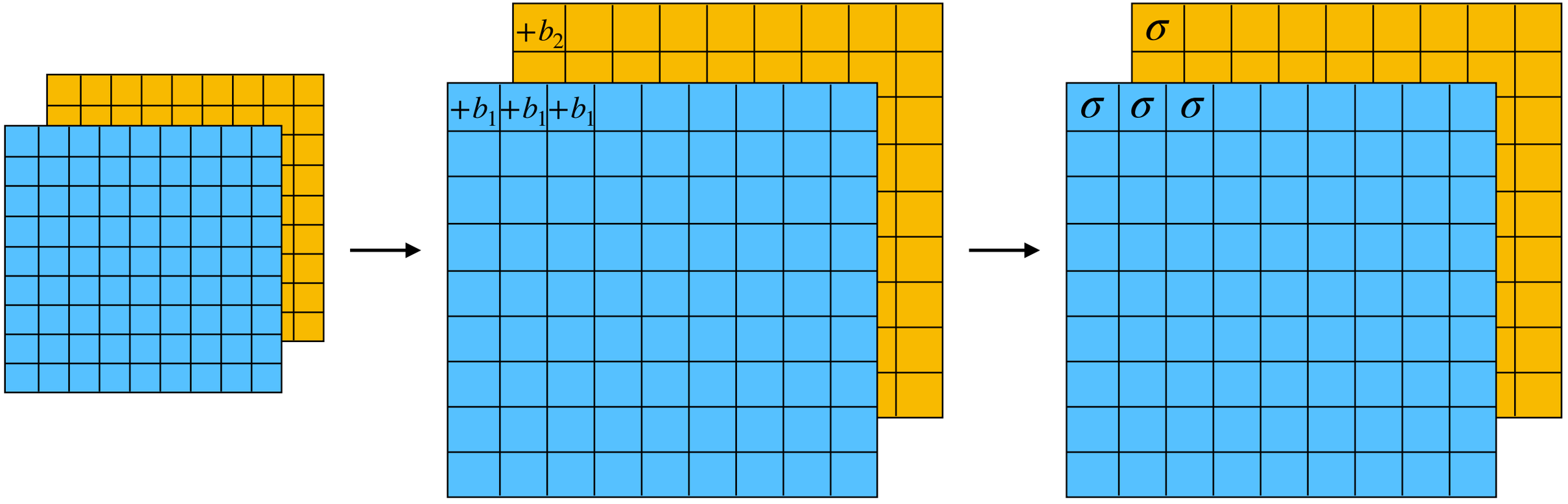


Convolutional Layer

- Convolutional layers differ from fully-connected layers in that they:
    - Only look at a small part of the input.
    - Share the same weights across multiple neurons.
- Besides that they are like a normal layer, including a bias (the same for all neurons in a feature map) and an activation function.

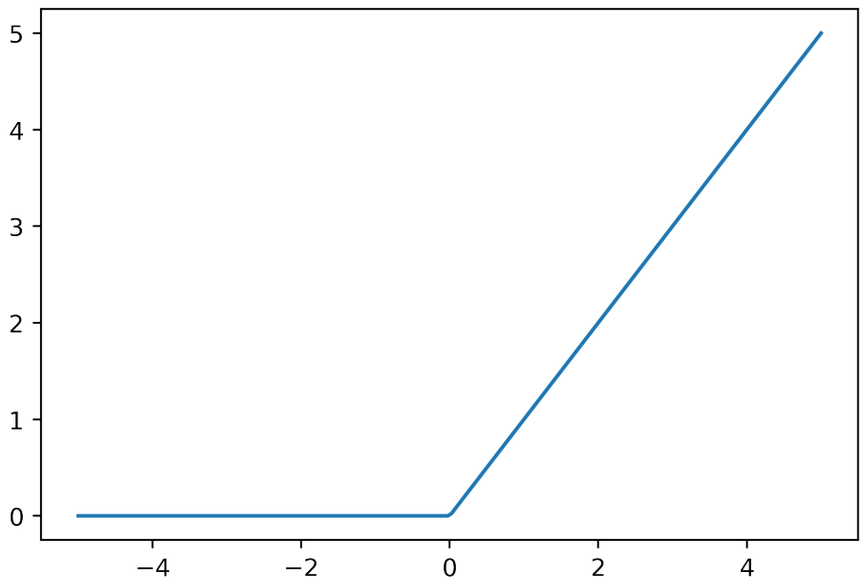
Bias and Activation

- After the convolution, there is the bias added to each neuron
- All neurons in a feature map share the same bias.
- Then the activation function is applied to each neuron.



ReLU Activation

- The most common used activation function for CNNs (but also in general: **Re**ctified **L**inear **U**nit



ReLU

- Intuition: because it has no upper restriction in activation it has more “space” to represent the data.

Convolutional Layer

- Takes an input of shape $height \times width \times channels$.
- The weights are multiple kernels of shape $kernelsize \times kernelsize \times channels$.
- Each kernel produces a feature map of shape $height \times width$ (in default case).
- Given n different kernels the output of the convolutional layer has shape $height \times width \times n$.

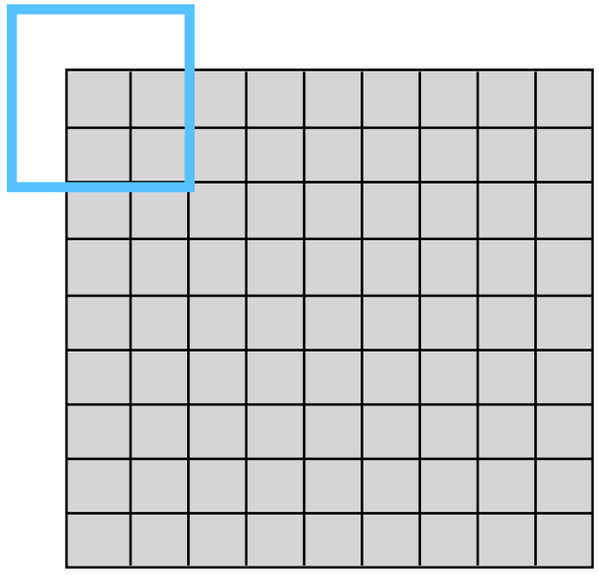
Hyperparameters

- A convolutional layer has a few hyperparameters we can specify:
    - Number of different kernels
    - Kernel size (usually an odd number, e.g. 3×3, 5×5, …)
    - Padding mode
    - Stride size

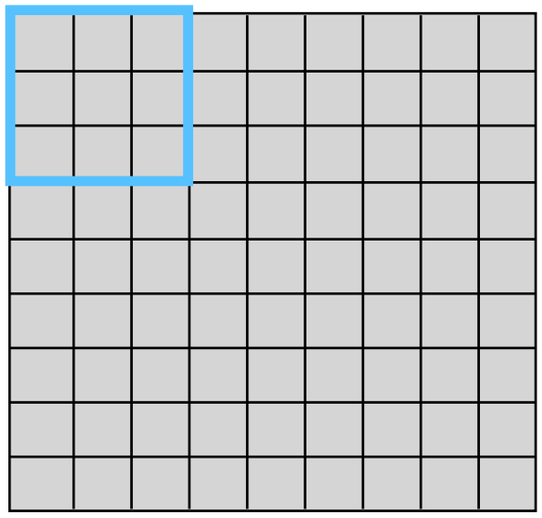
Padding Mode

- What to do with the boundaries?

Include the boundary.

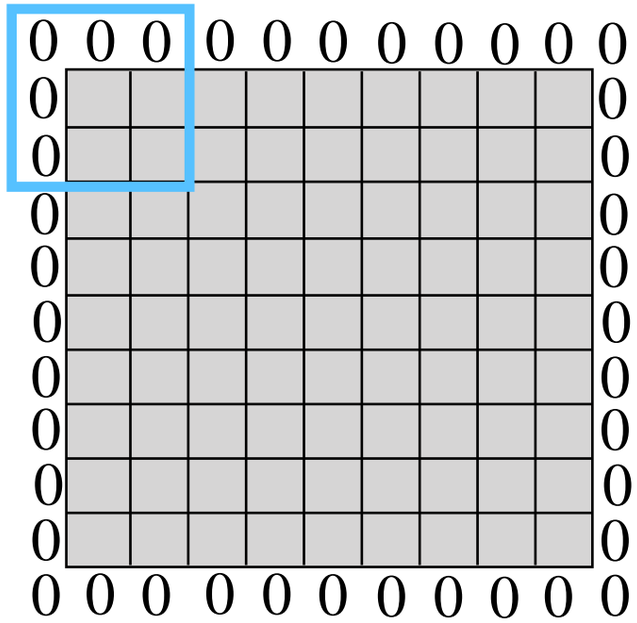


Exclude the boundary.



Zero Padding

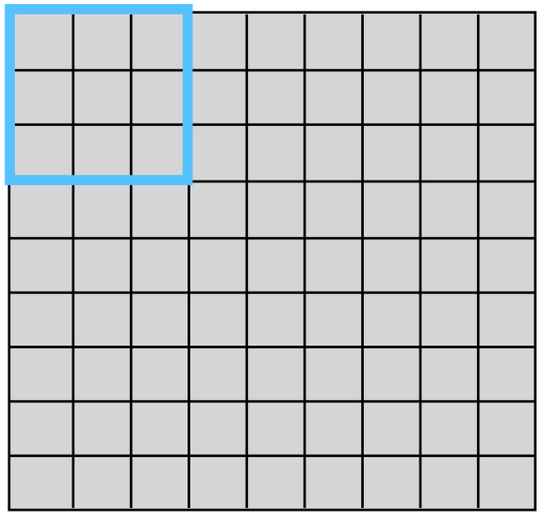
- If you want include the boundary you have to fill in values around it.
- This is called padding.
- Most popular: zero padding.




```
TensorFlow:
padding="same"
```

No Padding

- If you want to exclude the boundary you don’t have to pad.



`TensorFlow: padding=“valid”`

Padding Mode

- The chosen padding mode influences the size of your feature map:

- For an input of shape $n \times n$ and a kernel with size $k \times k$ the feature map has shape

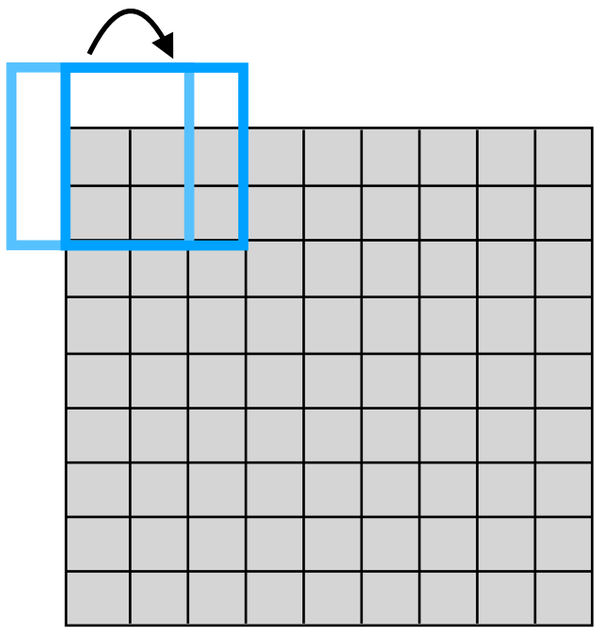
  - $n \times n$ **Zero padding**

  - $n - (k-1) \times n - (k-1)$ **No padding**

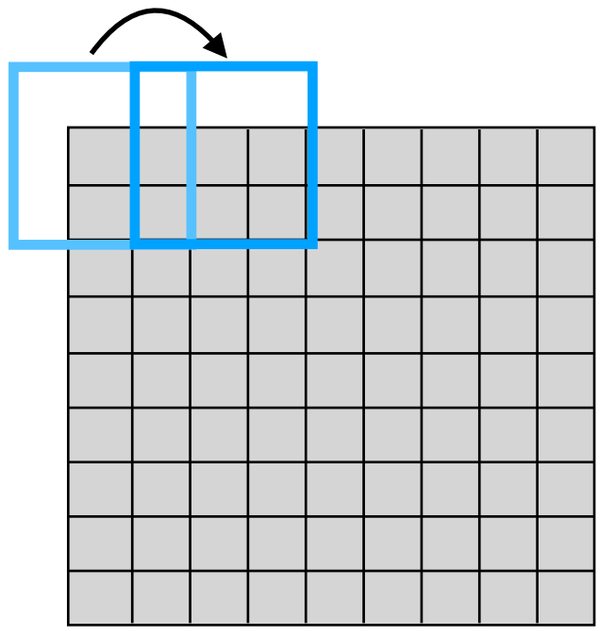
Stride Size

- The stride size says how far you shift your kernel in one step.

Stride size: 1



Stride size: 2

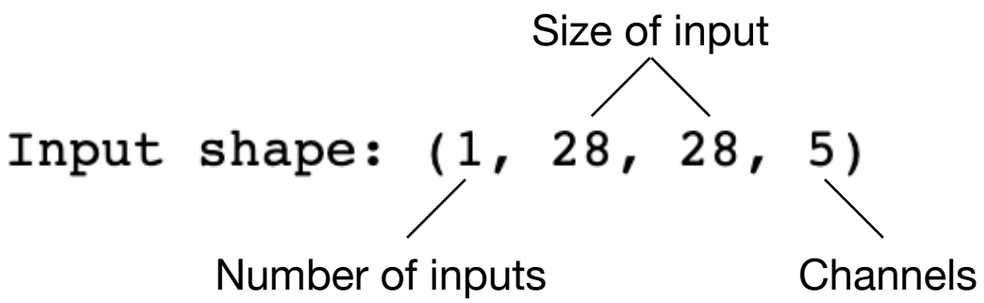


- The stride size also influences the size of the resulting feature map.

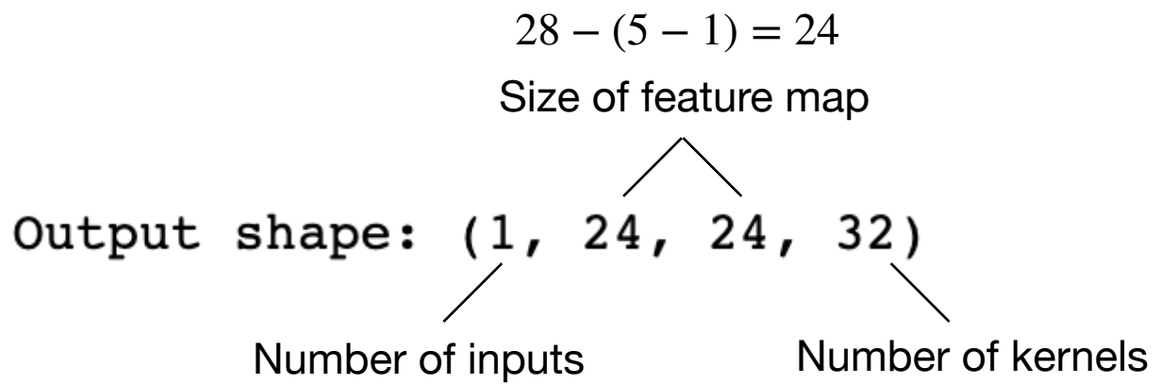
Convolutional Layer - TF

```
conv_layer = tf.keras.layers.Conv2D(
                            filters=32, # number of kernels
                            kernel_size=5, # kernel size (e.g. 5x5)
                            strides=(1,1), # stride size (e.g. 1 in horizontal, 1 in vertical)
                            padding='valid', # padding mode
                            activation=tf.math.sigmoid, # activation function,
                            input_shape=(28,28,5) # specifying the input shape (only need that in first conv. layer)
                            )
```

```
x = np.zeros((1,28,28,5))
feature_maps = conv_layer(x)
print("Input shape: {}".format(x.shape))
print("Output shape: {}".format(feature_maps.shape))
```



$$28 - (5 - 1) = 24$$

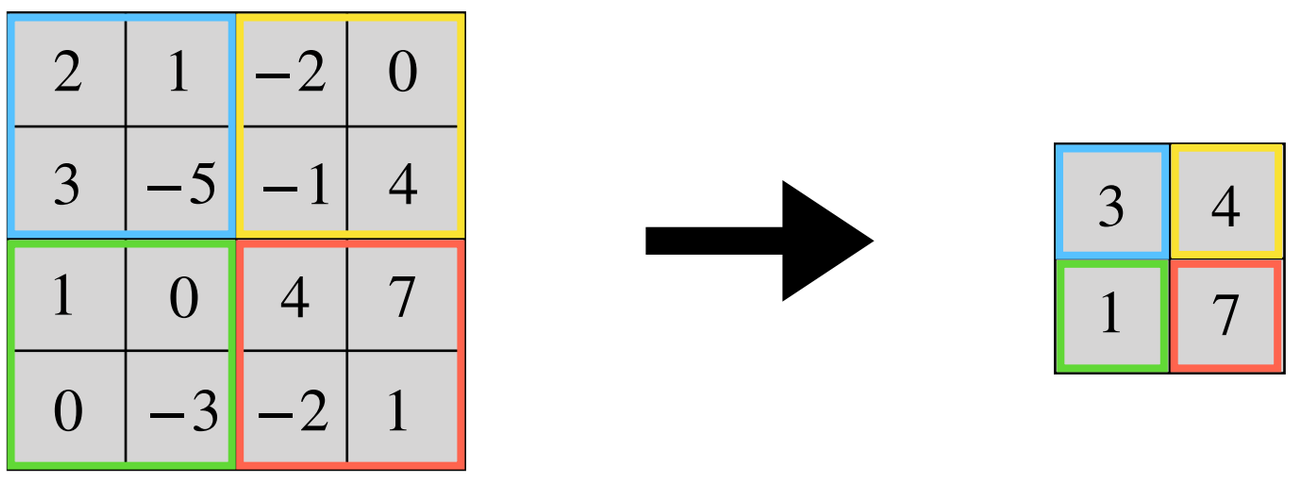


Pooling Layer

Pooling Layer

- The second type of layer that is built into most CNNs is the pooling layer.

- A pooling layer pools together a certain amount of values.

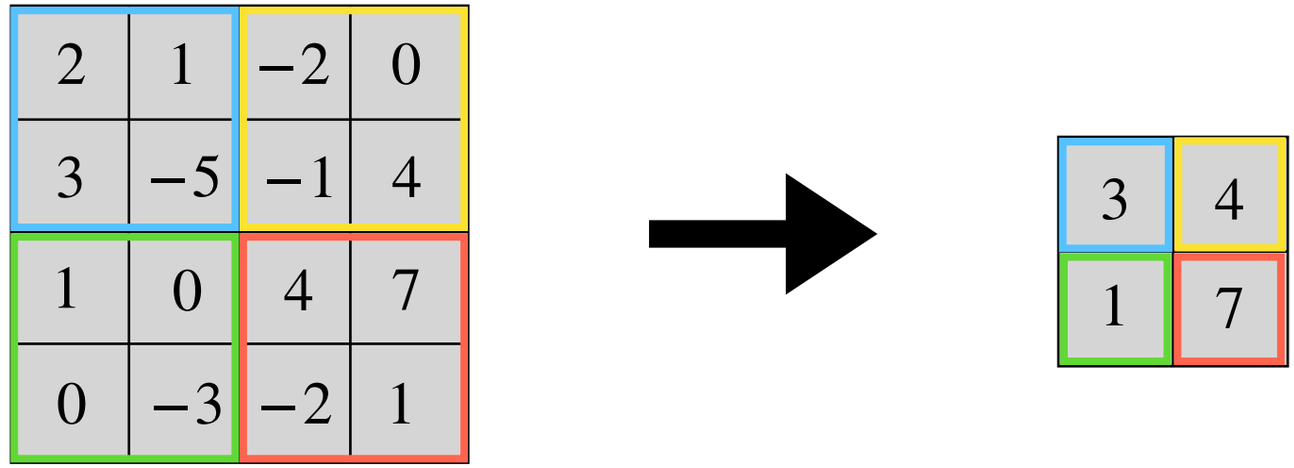


Pooling Modes

- There are two popular pooling modes:

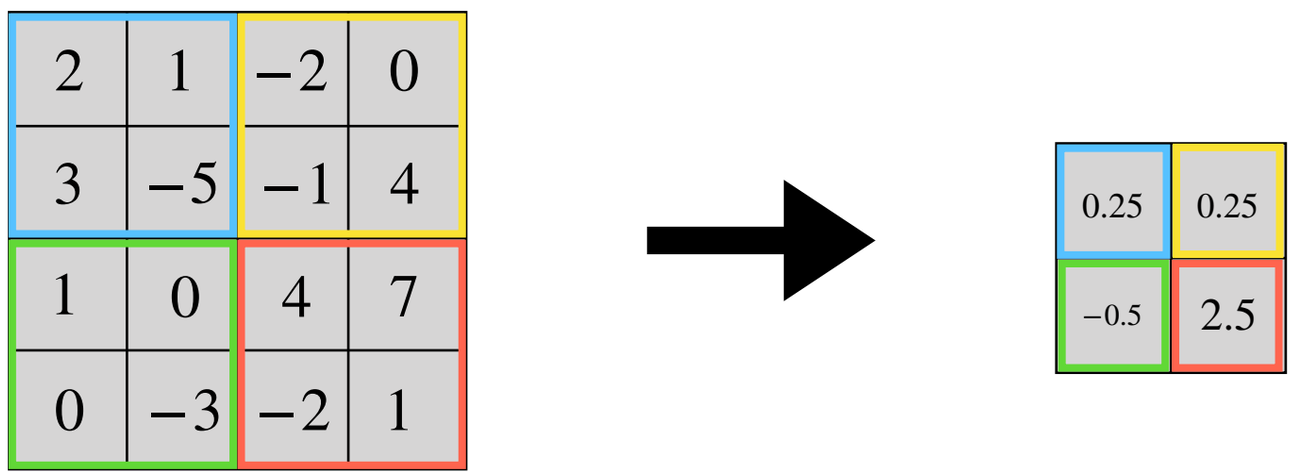
MaxPooling

Take the maximum of pooling region.



AveragePooling

Take the average of pooling region.



Hyperparameters

- A pooling layer also has a few hyper parameters:
    - Pool size (how many values do you want to pool together)
    - Stride size (how many steps is the pooling kernel shifted? Usually stride size = pool size!)

Pooling Layer - TF

```
max_pool_layer = tf.keras.layers.MaxPool2D(
                                pool_size = (2,2), # size of pool
                                strides = (2,2), # how many pixels to shift the pool per step
                                )
```

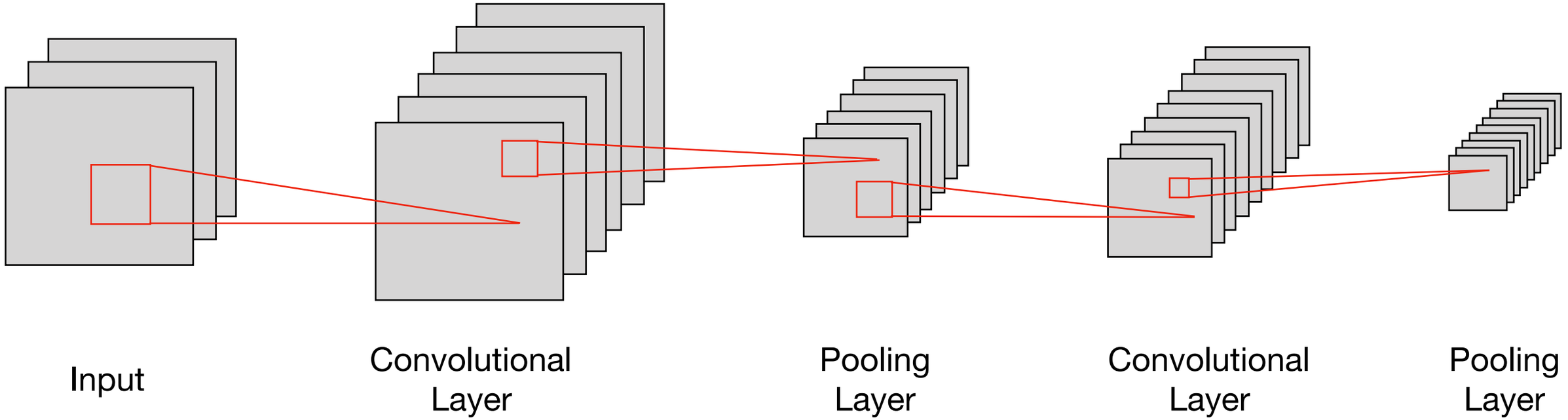
```
pooled_maps = max_pool_layer(feature_maps)
print("Input shape: {}".format(feature_maps.shape))
print("Output shape: {}".format(pooled_maps.shape))
```

```
Input shape: (1, 24, 24, 32)
Output shape: (1, 12, 12, 32)
```

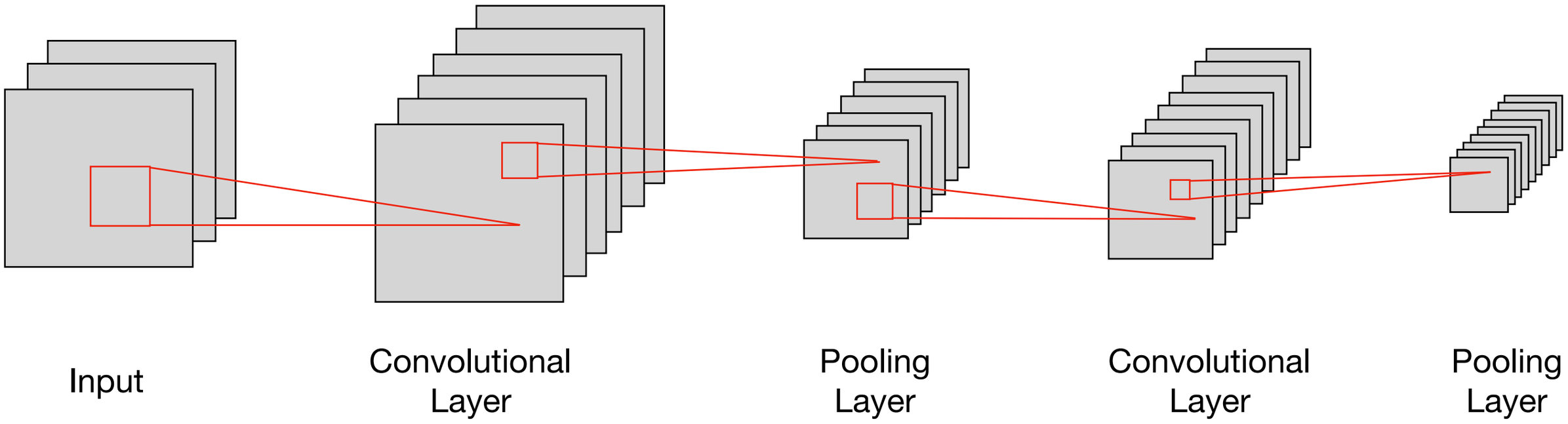
Feature Extractor

Feature Extractor

- The first part of a CNN is the feature extractor.
- It is some sequence of alternating convolutional layers and pooling layers.



Feature Extractor



Convolutional layers usually increase the depth (number of feature maps/channels) of the representation.

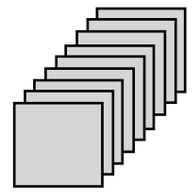
Pooling layers are used to decrease the size of the feature maps.

Feature Extractor

- Note that the sequence does not have to be alternating.
- You can have multiple convolutional layers without a pooling layer in between.

Feature Extractor

- The output of the feature extractor is several small feature maps.

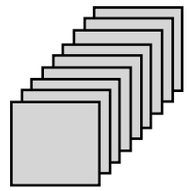


- We will use a fully connected neural network (MLP) to process these extracted features.
- But for that we have to transform it into a vector.

Feature Extractor

There are two ways to do this:

Let's say our last feature map block has shape (h, h, d)



1. Flatten

We simply flatten the block into a long vector of shape $(h \cdot h \cdot d)$.

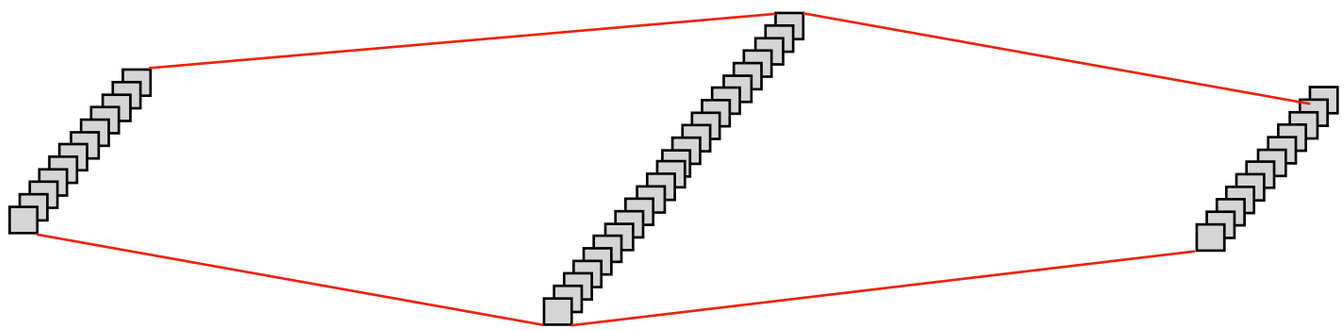
2. Global Average Pooling

We perform one last average pooling over the whole feature map which results in a vector of shape (d).

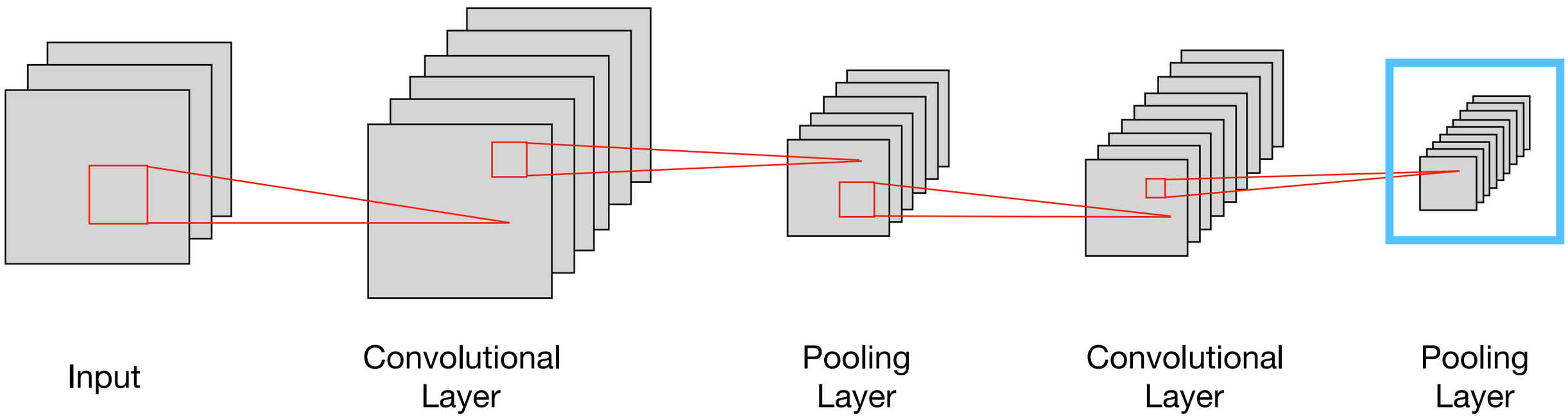
Classifier

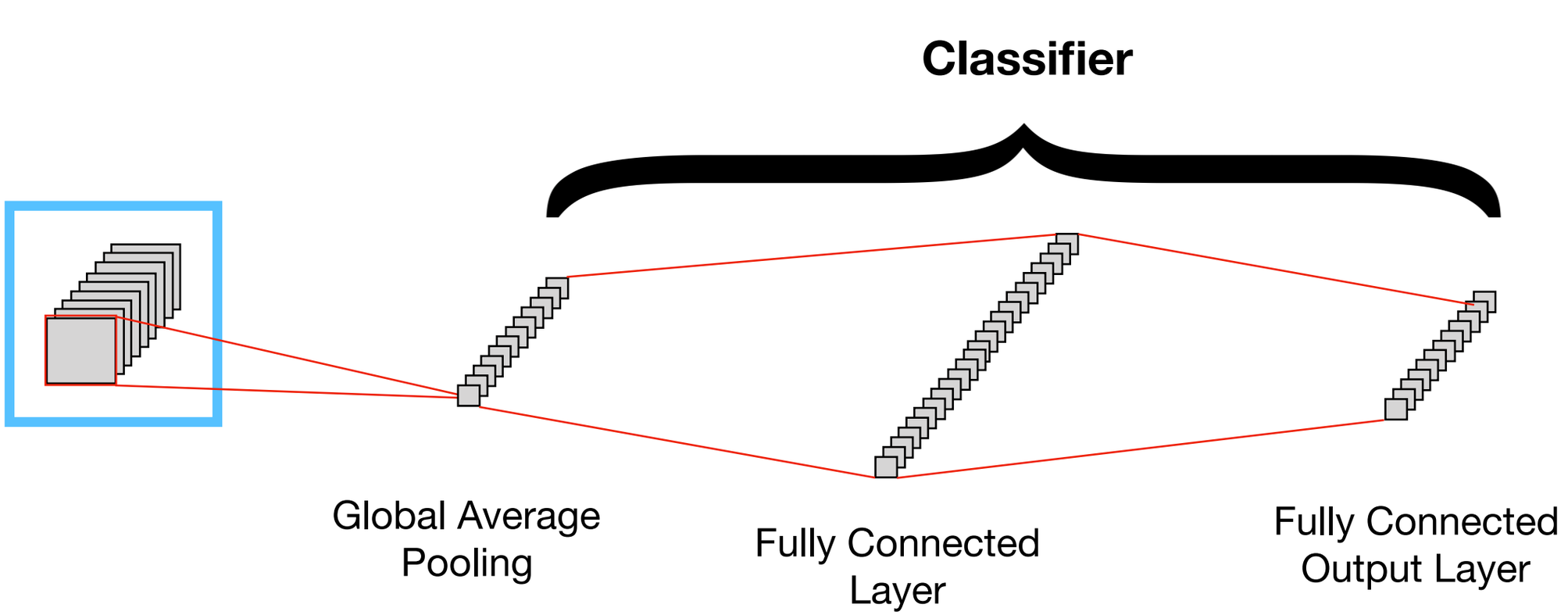
Classifier

- Now the feature extractor represent the input data as a long vector.
- On this vector we perform simple classification with a fully connected neural network.
- This network usually has either no or one hidden layer.



Classifier


Input
Convolutional Layer
Pooling Layer
Convolutional Layer
Pooling Layer


Classifier
Global Average Pooling
Fully Connected Layer
Fully Connected Output Layer

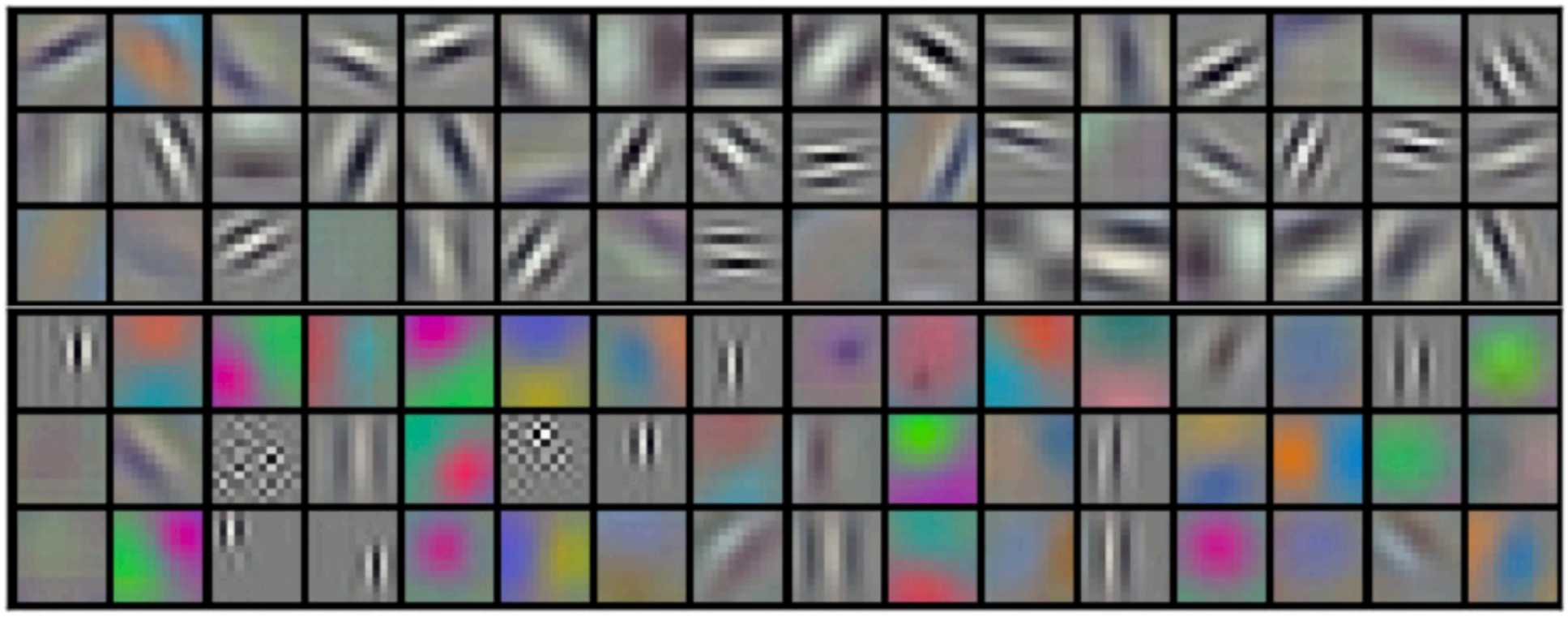
Intuitions

Translational Equivariance

- As all neurons in one feature map share the same kernel the resulting is representation is translational equivariant.
- This means that if you shift the input image, the feature map shifts accordingly.
- Intuitively this allows the network to recognize objects, although they are not at a position that the network observed during training.

Kernel Visualization

We can visualize the kernels of the first layers. It reveals that the learned kernels are similar to what neurons in V1 are detecting.



[ak]

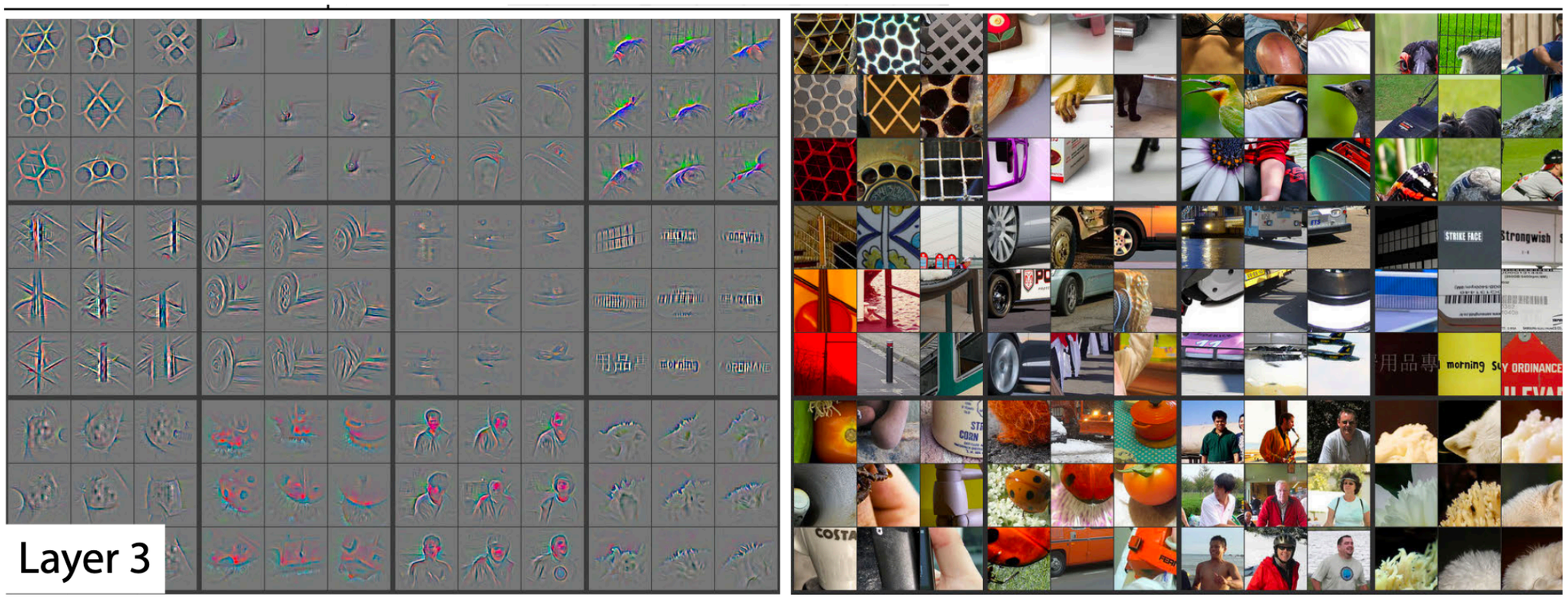
Complex Features in Higher Layers

- While the first layer is looking for simple features, the higher layers are accordingly looking for more and more complex features.

- A theory is that the networks first looks for simple shapes, then components and later for objects (take this theory with a grain of salt.)

Visualizing Higher Layer Features

- There are different possibilities to visualize higher layer features.
- It can give you an intuition about what the network is looking for.



[mz]

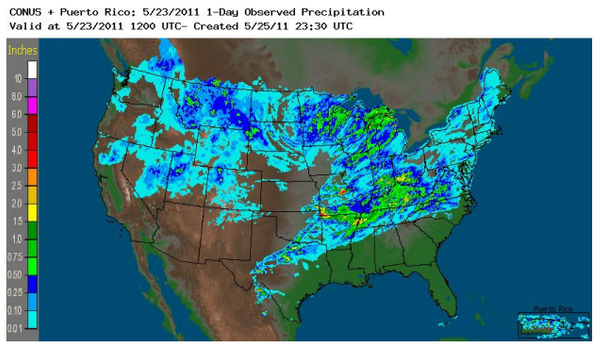
Tasks for CNNs

Tasks for CNNs

- CNNs are suited for all tasks that have some spatial structure:
    - All tasks that includes images (including videos), e.g.:
        - Object recognition
        - Semantic segmentation.
        - Object localization.

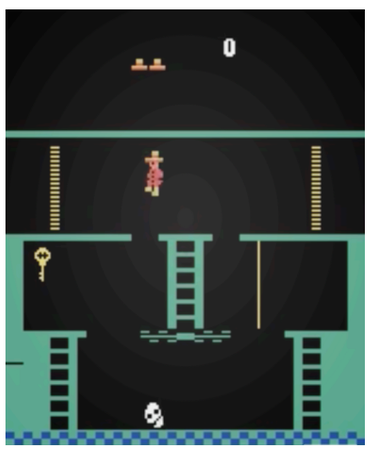
Tasks for CNNs

- CNNs are suited for all tasks that have some spatial structure:
  - Other tasks on map structure inputs, e.g.:
    - Weather prediction.

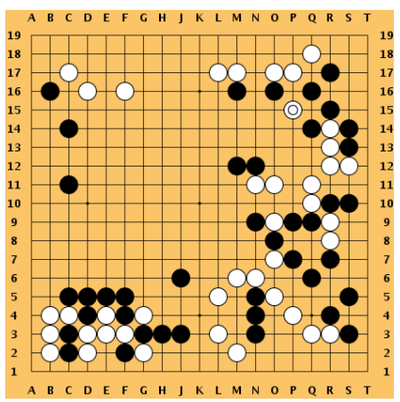


[w3]

    - Playing Games.



[oa]



Any questions left?

Conclusion

- We looked into the features that fully-connected neural networks are extracting, namely global ones.

- To look for local features (as they exist in e.g. images) we can make use of a technique called cross-correlation or convolution.

- A deep neural network, which is based on this principle is called a convolutional neural network (CNN).

- We saw which different parts a CNN has and how a forward processing step is defined.

Outlook

- We have now learned about network architectures that can deal with vector and image data.

- The last important data structure is data, which includes a time component (e.g. speech).

- This will be the first topic after the christmas break.

- Next week we will look into several things that can go wrong when training a deep neural network and what you can do against it.

See you next week!

Resources

[w1] Michael Plotke at Wikipedia (https://commons.wikimedia.org/wiki/File:3D_Convolution_Animation.gif)

[w2] Jérémy Barande at Wikipedia (https://commons.wikimedia.org/wiki/File:Yann_LeCun_-_2018_(cropped).jpg

[ylc] Y. LeCun et al., "Gradient-based learning applied to document recognition." *Proceedings of the IEEE*, 86(11):2278-2324, November 1998.

[ak] A. Krizhevsky et al., "Imagenet classification with deep convolutional neural networks" *Proceedings of the NeurIPS 2012*, 2012.

[mz] M. Zeiler & R. Fergus, “Visualizing and understanding convolutional networks" *Computer Vision - ECCV 2014*, 2014.

[oa] https://gym.openai.com/envs/MontezumaRevenge-v0/

[w3] https://commons.wikimedia.org/wiki/File:Joplin-rain-map-2011.jpg