Machine Learning (CE 40717)

Fall 2024

Ali Sharifi-Zarchi

CE Department
Sharif University of Technology

November 5, 2024

Channels

So far, we have talked about 2D inputs, but although images are 2D, we need to represent them as **3D matrices** to show their colors:

- Pixel values range from 0 to 255.
- We **cannot** represent all the colors in a picture with only one channel of numbers from 0 to 255.
- So we represent them in **3 channels**.

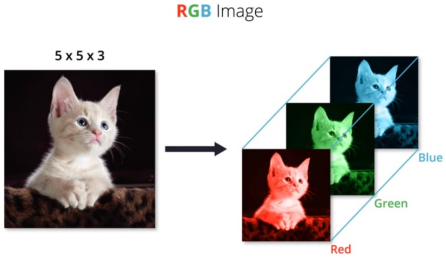


Figure adapted from Source

Channels

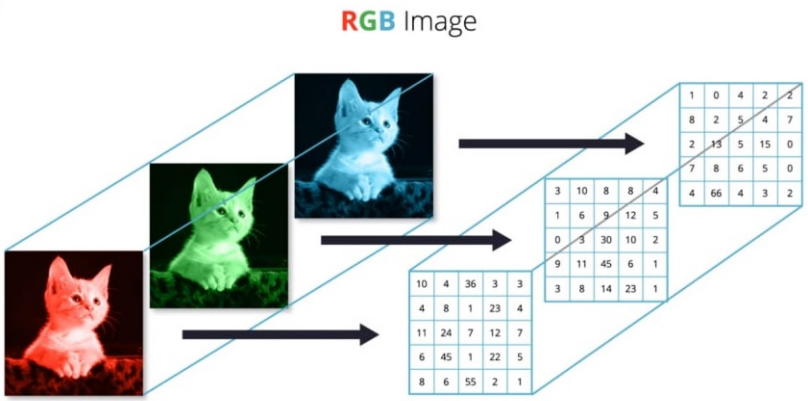


Figure adapted from Source

Let's have a closer look at the calculations

Channels

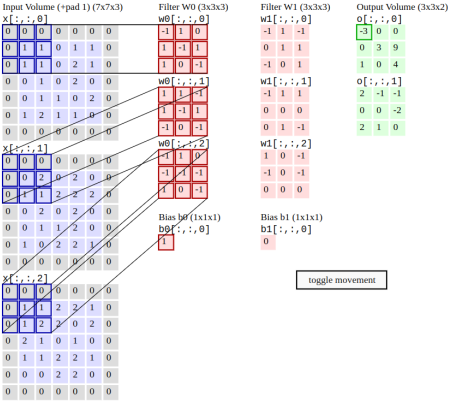


Figure adapted from [2]

Channels

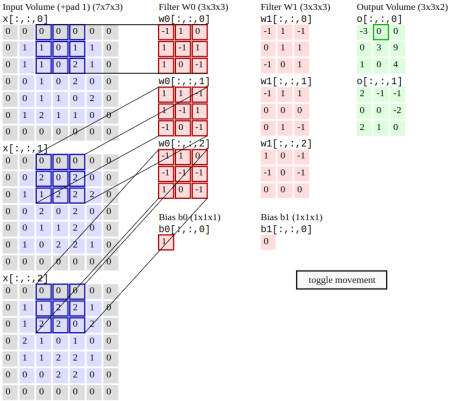


Figure adapted from [2]

Channels

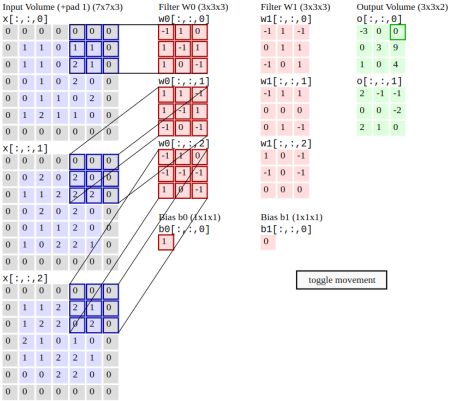


Figure adapted from [2]

Channels

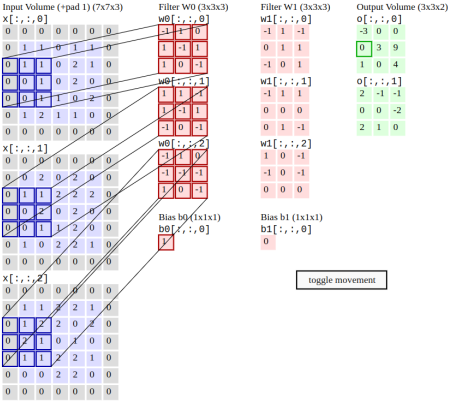


Figure adapted from [2]

Channels

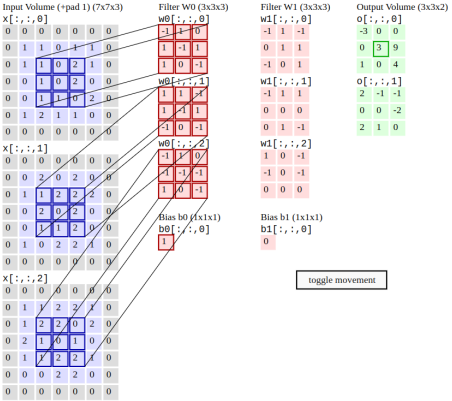


Figure adapted from [2]

Channels

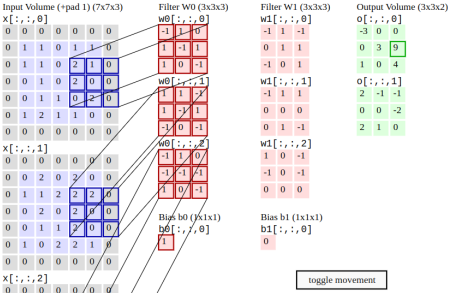


Figure adapted from [2]

Channels

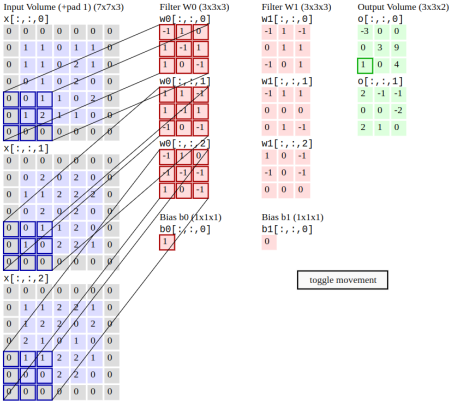


Figure adapted from [2]

Channels

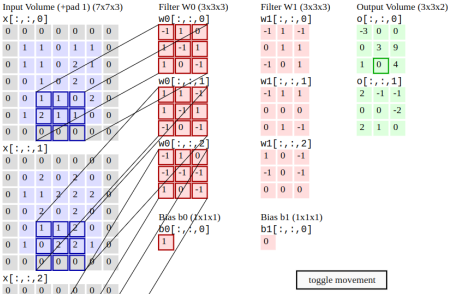


Figure adapted from [2]

Channels

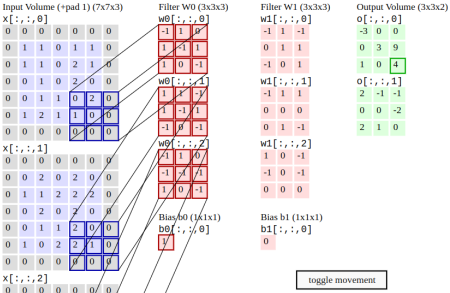


Figure adapted from [2]

Channels

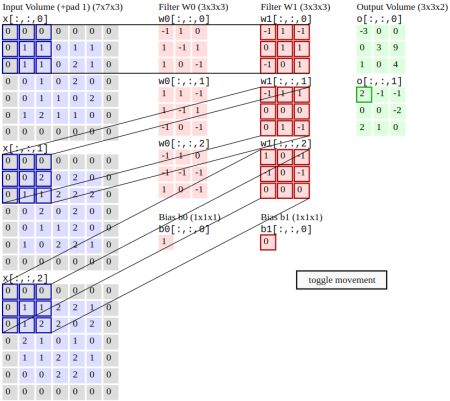


Figure adapted from [2]

Channels

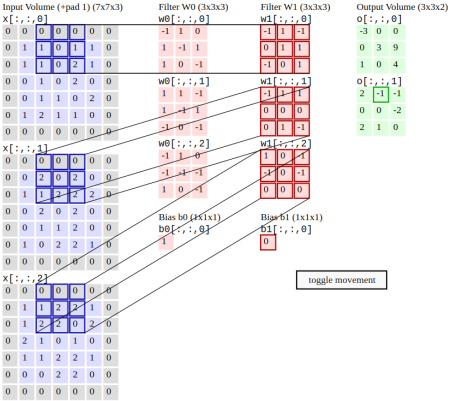


Figure adapted from [2]

Channels

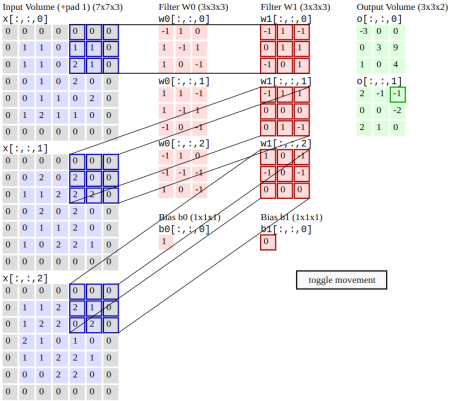


Figure adapted from [2]

Channels

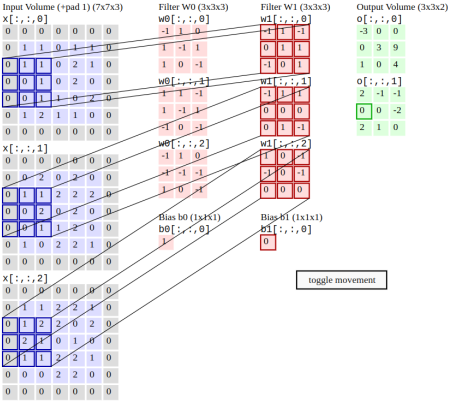


Figure adapted from [2]

Channels

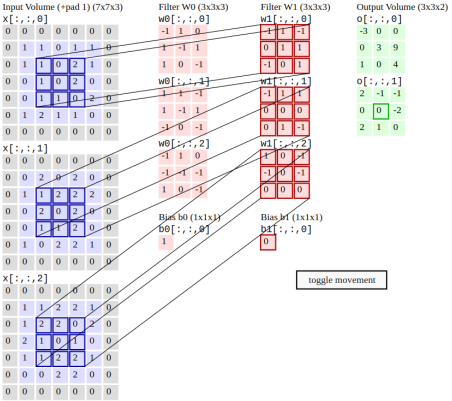


Figure adapted from [2]

Channels

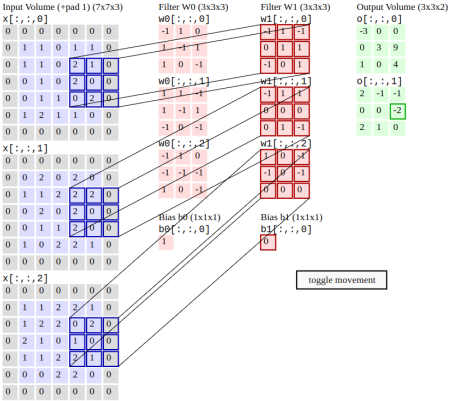


Figure adapted from [2]

Channels

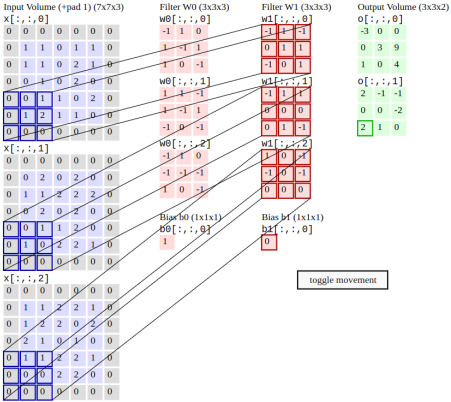


Figure adapted from [2]

Channels

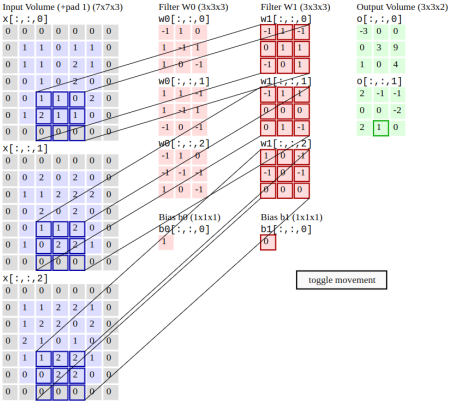


Figure adapted from [2]

Channels

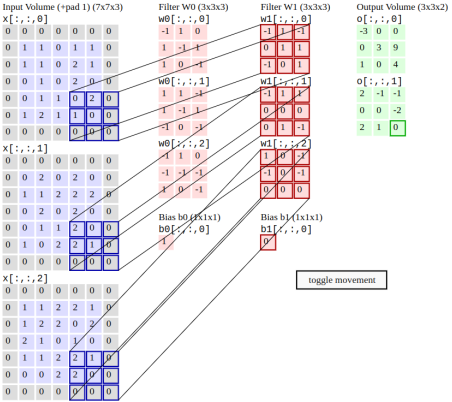


Figure adapted from [2]

Channels

- Each filter results in **one output channel**. We can apply multiple different filters to obtain multiple output channels. Each channel can learn something **distinct**.

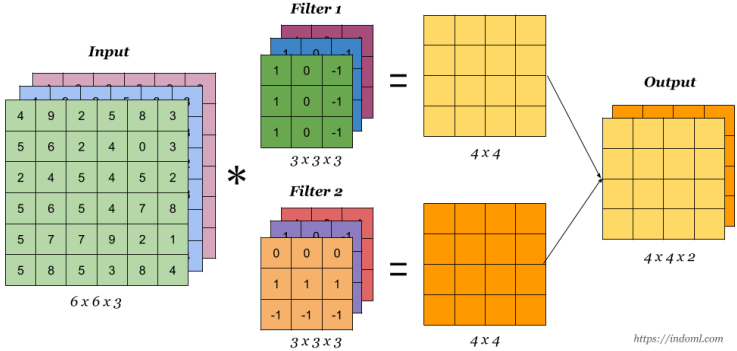


Figure adapted from Source

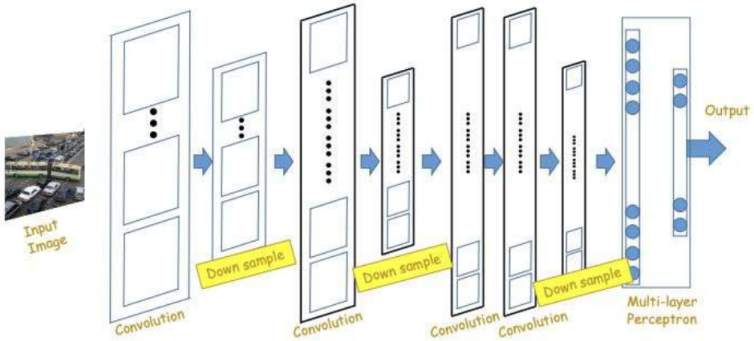
Review

Three Main Types of Layers

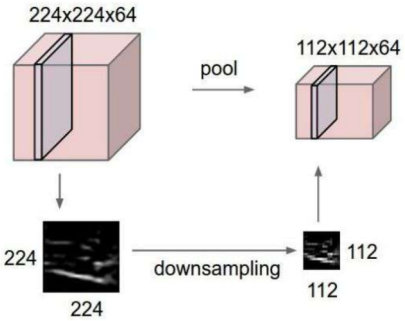
- **Convolutional Layer**
  - Output of neurons are connected to local regions in the input.
  - Applying the same filter across the entire image.
  - CONV layer's parameters consist of a set of learnable filters.
- **Pooling Layer**
  - Performs a downsampling operation along the spatial dimensions.
- **Fully-Connected Layer**
  - Typically used in the final stages of the network to combine high-level features and make predictions.

Pooling

- Convolution & activation layers are **followed intermittently by pooling layers**.
  - Often, they alternate with convolution, though this is not necessary.



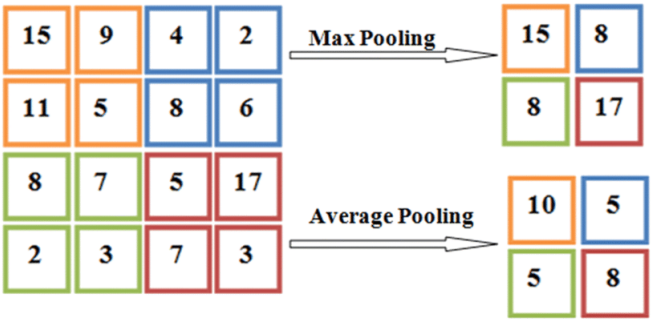
Pooling



- **Reduce the spatial size** of the representation.
  - To reduce the number of parameters and computational demands within the network.
  - To **reduce variance**.
- Helps the network become invariant to small translations or distortions.
- Operates over each activation map independently.

Figure adapted from [2]

Pooling Type



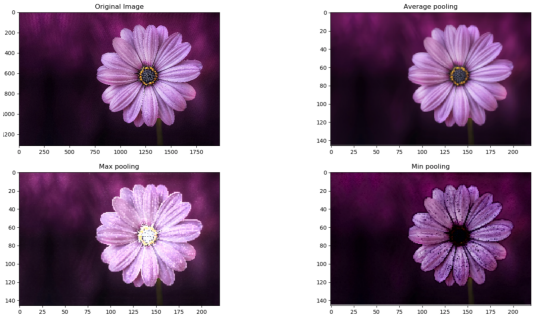
Two Primary Types of Pooling:

- **Max Pooling:** Selects the **maximum** value from each patch of the feature map.
- **Min Pooling:** Selects the **minimum** value from each patch of the feature map.
- **Average Pooling:** Computes the **average** of the values in each patch of the feature map.

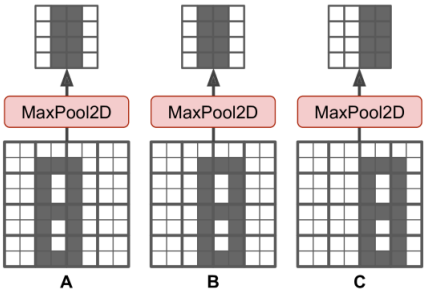
Figure adapted from source

Pooling Type

- Max Pooling: performs well **when the background is light**, and the object is dark.
- Min Pooling: performs well **when the background is dark**, and the object is light.
- Average Pooling: **smooths** the image.



Pooling Advantage



- Most common type of pooling layer.
- Invariance to small translations.

Parameter Setting

Question:
How does a pooling layer affect the dimensions of the input image?

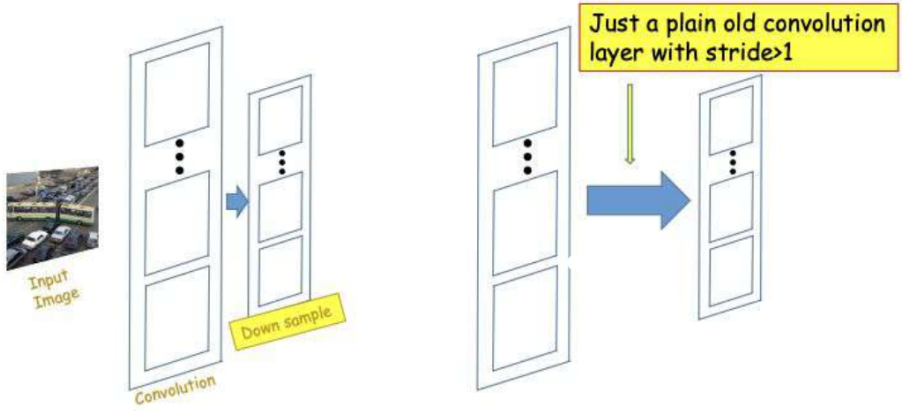
Answer:

- An $N \times N$ picture compressed by a $P \times P$ pooling filter with stride S results in an output map of side $\left\lceil \frac{(N-P)}{S} \right\rceil + 1$.

Parameter Setting

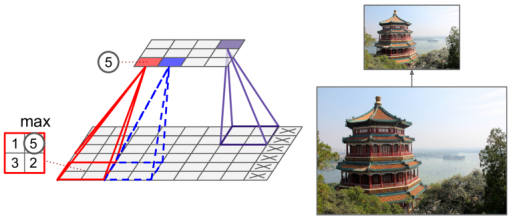
- Pooling accepts a volume of size $W_1 \times H_1 \times D_1$
- Requires two hyperparameters:
  - Their spatial extent F,
  - The stride S.
- Produces a volume of size $W_2 \times H_2 \times D_2$ where:
  - $W_2 = \frac{(W_1 - F)}{S} + 1$
  - $H_2 = \frac{(H_1 - F)}{S} + 1$
  - $D_2 = D_1$
- Introduces **zero parameters** since it computes a fixed function of the input.
- It is **uncommon** to use zero-padding for pooling layers.

Alternative To Pooling



- Downsampling can be done by a simple convolution layer with stride larger than 1, Replacing the max pooling layer with a convolution layer.

Pooling Summary



Max pooling layer (2 × 2 pooling kernel, stride 2, no padding)

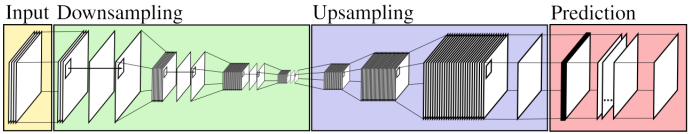
- Goal is to sub-sample the input to reduce:
  - Computational load
  - Memory usage
  - Number of parameters
  - Risk of overfitting

Question

- **What if we want to increase the size?**

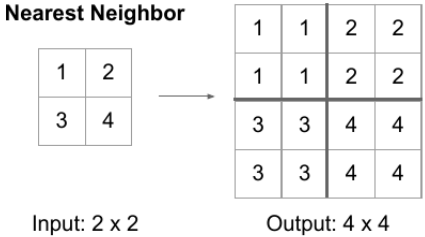
Up-sampling CNN

- **Resizing feature maps** is a **common** operation in many neural networks, especially those that perform some kind of image segmentation task.
- This kind of architecture is famously known as the **Encoder-Decoder** network.



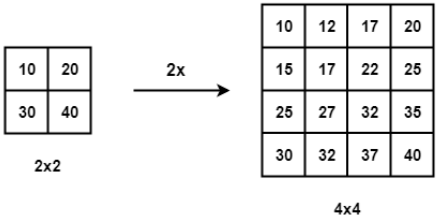
Nearest Neighbors

- **Nearest Neighbors**: In Nearest Neighbors, as the name suggests, we take an input pixel value and copy it to the K-Nearest Neighbors, where K depends on the expected output.



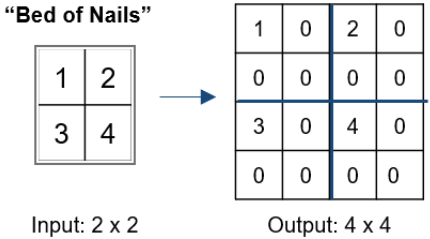
Bi-Linear Interpolation

- **Bi-Linear Interpolation**: In Bi-Linear Interpolation, we take the **4 nearest pixel** values of the input pixel and perform a **weighted average** based on the distance of the four nearest cells, smoothing the output.



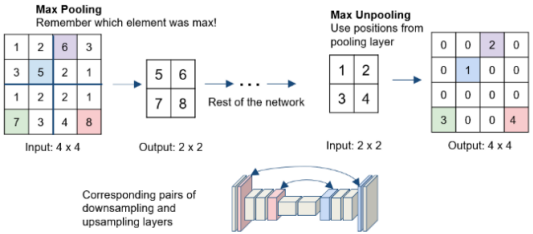
Bed Of Nails

- **Bed Of Nails**: In Bed of Nails, we copy the value of the input pixel at the corresponding position in the output image and **fill zeros** in the remaining positions.

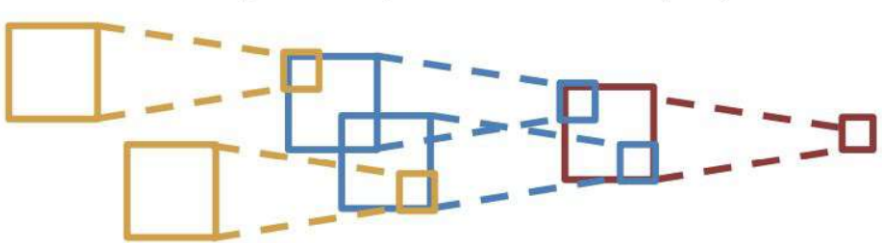


Max-Unpooling

- **Max-Unpooling**: To perform max-unpooling, first, the index of the **maximum value is saved** for every max-pooling layer during the encoding step. The saved index is then used during the decoding step where the input pixel is mapped to the saved index, **filling zeros everywhere else**.



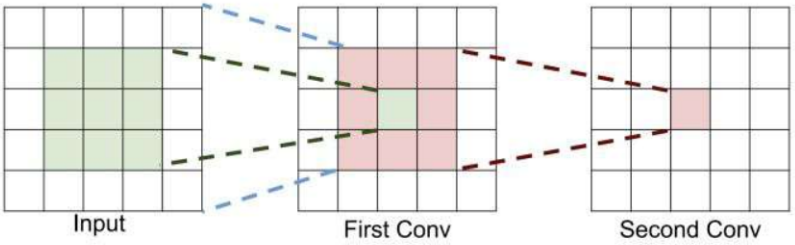
Receptive Field



- **Receptive Field :** How big of a region in the **input or previous** layer does a neuron on the n-th conv-layer see?
- For convolution with kernel size K, each element in the next layer depends on a $K \times K$ **receptive field** in the previous layer.

Figure adapted from [3]

Receptive Field



- Units in the deeper layers can be **indirectly** connected to most of the input image.
- Each successive convolution adds $K-1$ to the receptive field size. With L layers, the receptive field size is $1+L\cdot(K-1)$.
- **Problem:** For large images, we need many layers for each output to **see** the whole image.
  - **Solution: Downsample** inside the network using strides and pooling.

Power Of Small Filters

Suppose the input is $H \times W \times C$ and we use convolutions with C filters to preserve depth (stride 1, padding to preserve H, W).

one CONV with 7 x 7 filters
Number of weights
$= C \times (7 \times 7 \times C) =$ **49**C^2

three CONV with 3 x 3 filters
Number of weights
$= 3 \times C \times (3 \times 3 \times C) =$ **27**C^2

Both options achieve a receptive field of 7; however, using **multiple smaller filters** reduces the number of **parameters**, introduces more **nonlinearity**, and generally leads to a **more efficient**, expressive model.

Question

Question: In a convolutional neural network, each layer increases the receptive field size. Suppose a network has 3 convolutional layers, each with a kernel size of $K = 3$ and a stride of $S = 1$.

- Calculate the receptive field size after each layer, starting from an initial receptive field size of 1.
- How large is the receptive field after the third layer?
- Why is the growing receptive field important in deeper layers?

Answer

Answer:

- The receptive field size increases by $K-1$ with each layer.
  - After the 1st layer: $1+(3-1)=3$
  - After the 2nd layer: $3+(3-1)=5$
  - After the 3rd layer: $5+(3-1)=7$
- Thus, the receptive field after the third layer is 7.
- A larger receptive field allows neurons in deeper layers to capture more context from the input image, which is essential for understanding higher-level patterns.

Inductive Bias In CNNs

Inductive Bias:
Refers to the assumptions a model incorporates to generalize from training data to unseen data.

Key Features of Inductive Bias in CNNs:

- **Weight Sharing:**
  A single filter is applied across different regions of the input, significantly reducing the number of parameters.
- **Locality:**
  CNNs use small filters (e.g., 3×3) that focus on local regions, aligning well with image data where local structures are important.
- CNNs are more sample-efficient than FCNs due to their inductive biases.
- CNNS reduce sample complexity from $O(d^2)$ in FCNs to $O(\log^2 d)$.

Contributions

- **This slide has been prepared thanks to:**
  - Ali Aghayari

[1] M. Soleymani Baghshah, "Machine learning." Lecture slides.

[2] F.-F. Li, Y. Li, and R. Gao, "Cs231n: Deep learning for computer vision." Lecture slides.

[3] B. Raj, R. Singh, and B. Dhingra, "11-785: Introduction to deep learning." Lecture slides.

[4] M. Kellis, "6.874 computational systems biology: Deep learning in the life sciences." Lecture slides.

[5] H. Li, "Eleg 5491: Introduction to deep learning." Lecture slides.

[6] M. Elgendy, *Deep Learning for Vision Systems*. Manning Publications, 2020.

[7] DeepMind, "Deep learning for ai with geoffrey hinton, yoshua bengio, and yann lecun." YouTube video.