Machine Learning (CE 40477)

Fall 2024

Ali Sharifi-Zarchi

CE Department
Sharif University of Technology

November 21, 2024

Channels

Previously, we discussed 2D inputs; however, although images are inherently 2D, they need to be represented as **3D matrices** to display color information.

- Pixel values range from 0 to 255.
- It is not possible to represent all the colors in a picture using only a single channel of numbers from 0 to 255.
- Thus, we represent them using **three channels**.

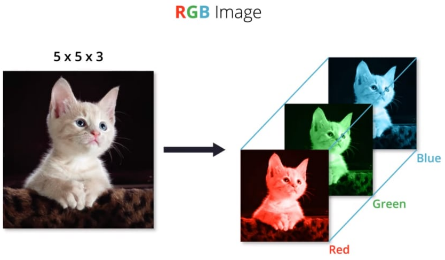


Figure adapted from Source

Channels

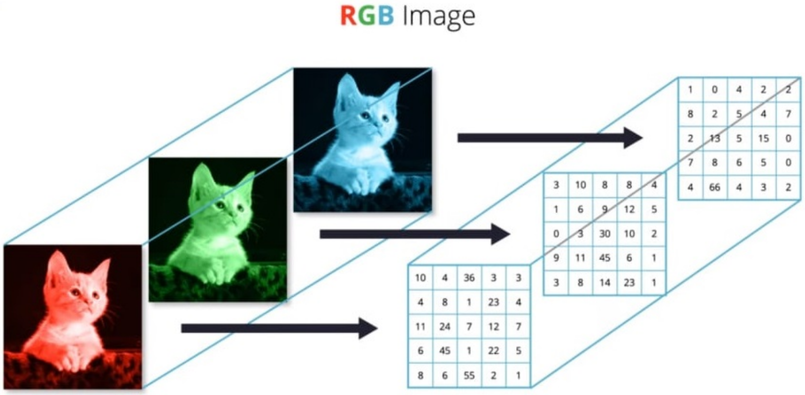


Figure adapted from Source

Channels

- Each filter produces **one output channel**. By applying multiple filters, we can create multiple output channels, allowing each channel to learn **distinct** features.

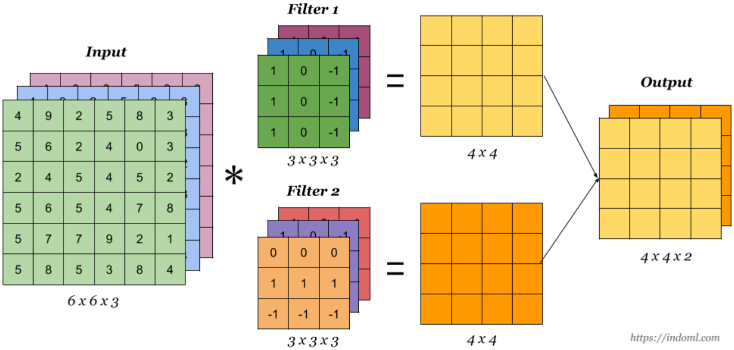


Figure adapted from Source

Let's take a closer look at the calculations

Channels

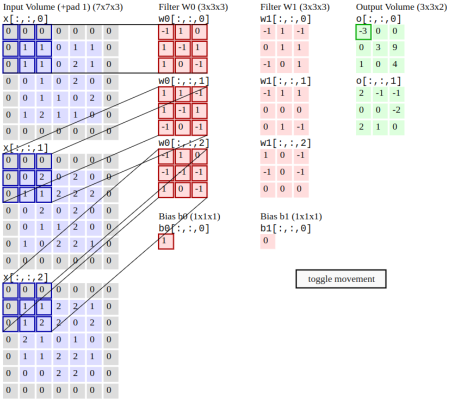


Figure adapted from [2]

Channels

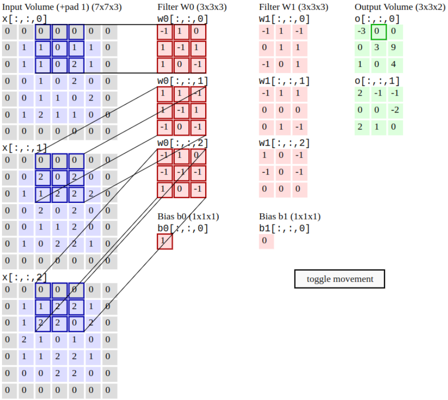


Figure adapted from [2]

Channels

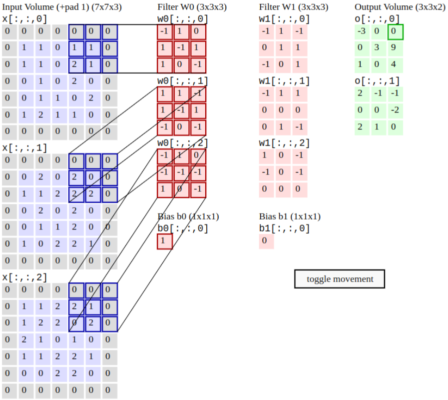


Figure adapted from [2]

Channels

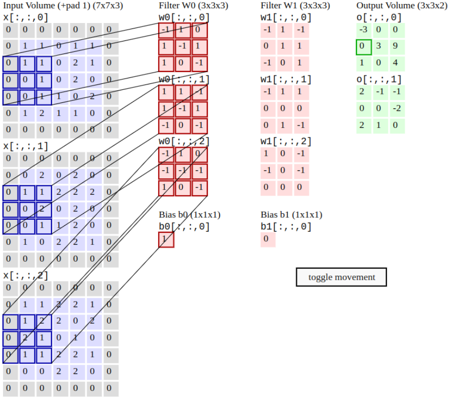


Figure adapted from [2]

Channels

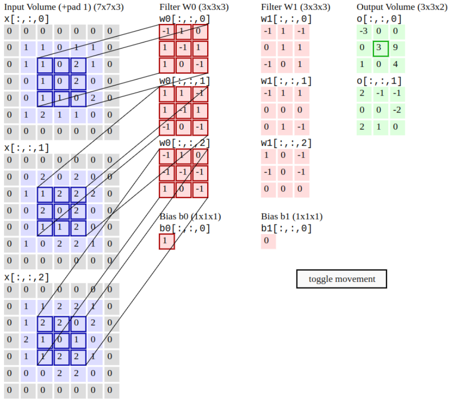


Figure adapted from [2]

Channels

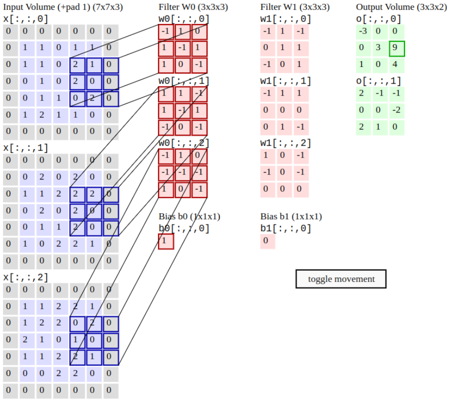


Figure adapted from [2]

Channels

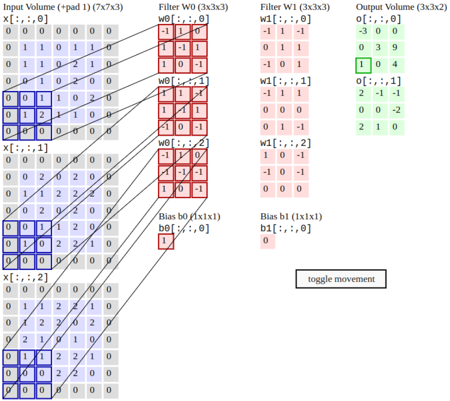


Figure adapted from [2]

Channels

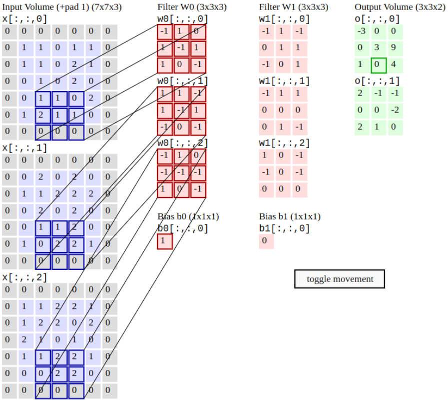


Figure adapted from [2]

Channels

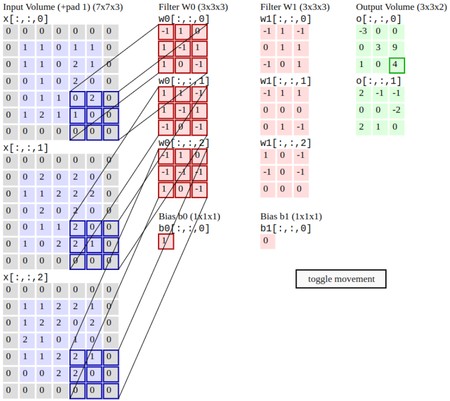


Figure adapted from [2]

Channels

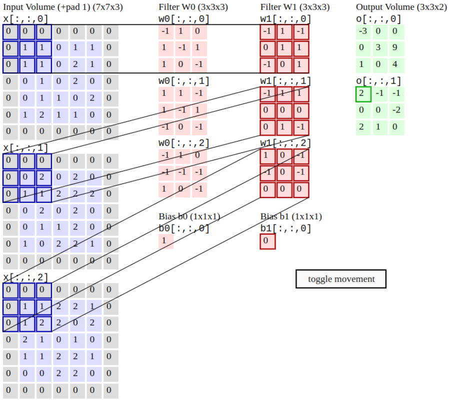


Figure adapted from [2]

Channels

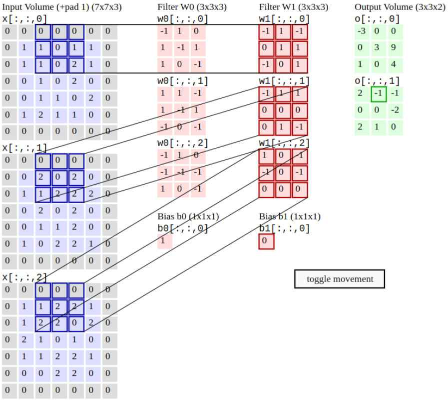


Figure adapted from [2]

Channels

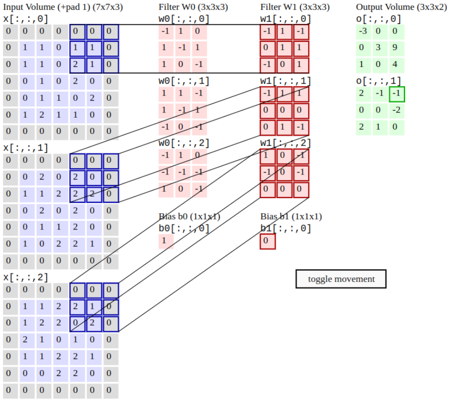


Figure adapted from [2]

Channels

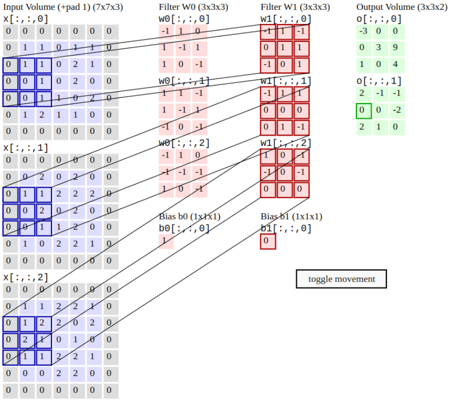


Figure adapted from [2]

Channels

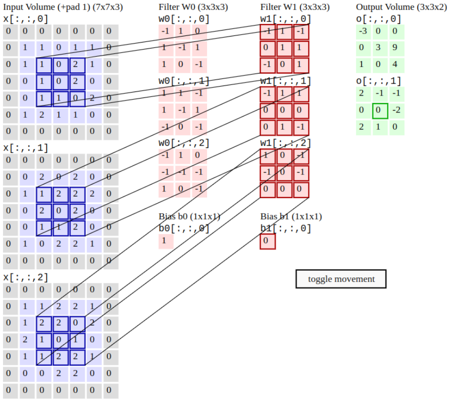


Figure adapted from [2]

Channels

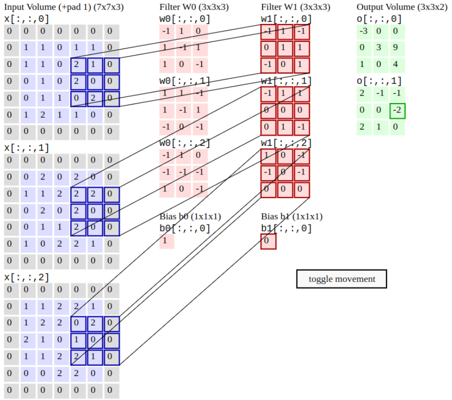


Figure adapted from [2]

Channels

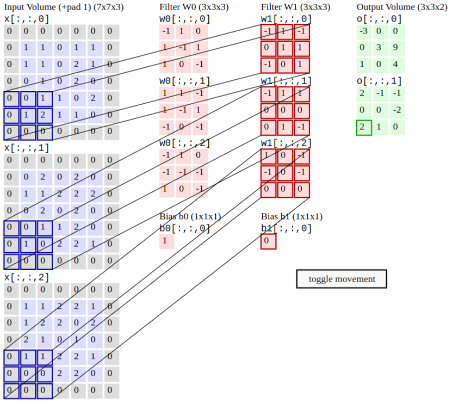


Figure adapted from [2]

Channels

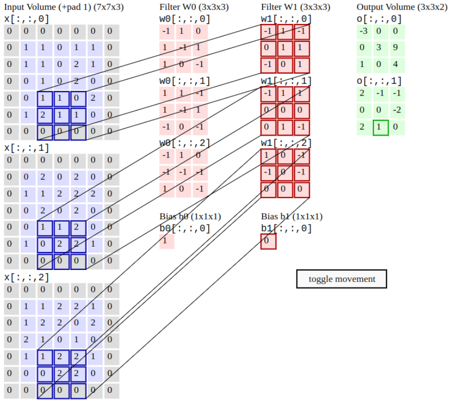


Figure adapted from [2]

Channels

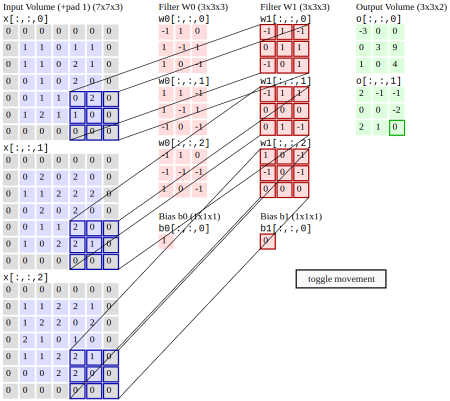


Figure adapted from [2]

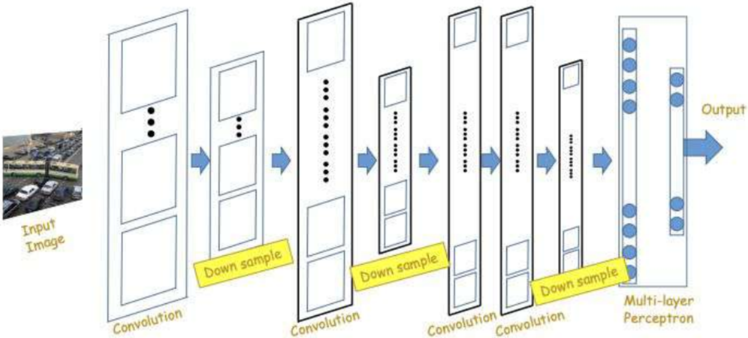
Review

Three Main Types of Layers

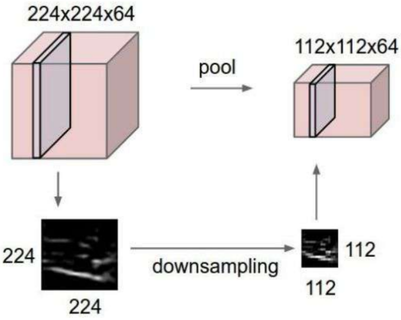
- **Convolutional Layer**
  - Neurons' outputs are connected to local regions in the input.
  - Applying the same filter across the entire image.
  - The parameters of the CONV layer include a set of learnable filters.
- **Pooling Layer**
  - Performs a downsampling operation along the spatial dimensions.
- **Fully-Connected Layer**
  - Typically used in the final stages of the network for combining high-level features to make predictions.

Pooling

- Convolution and activation layers are often followed by pooling layers intermittently.
  - Pooling layers often alternate with convolution layers, although this is not mandatory.



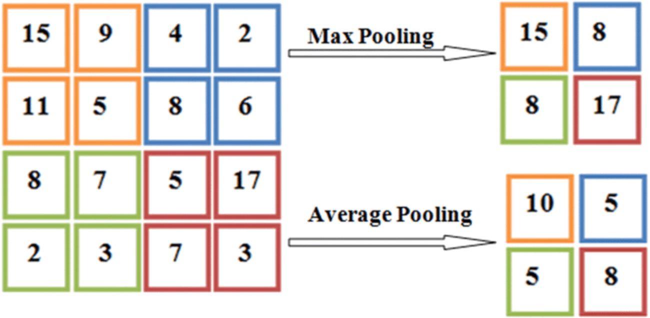
Pooling



- Reduces the **spatial size** of the representation.
  - To reduce the number of parameters and computational demands within the network.
  - To **reduce variability**.
- Enables the network to be invariant to small translations or distortions.

Figure adapted from [2]

Pooling Type

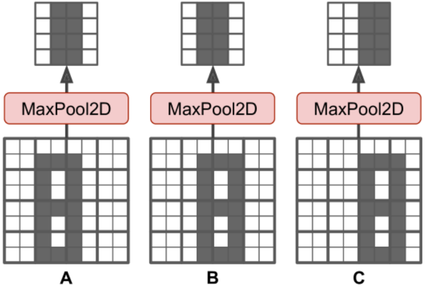


Two Primary Types of Pooling:

- **Max Pooling:** Selects the **maximum** value from each section of the feature map.
- **Min Pooling:** Selects the **minimum** value from each section of the feature map.
- **Average Pooling:** Calculates the **average** value for each section of the feature map.

Figure adapted from source

Max Pooling



- This is the most common type of pooling layer.
- Provides invariance to small translations.

Parameter Setting

Question:

How does a pooling layer change the input image dimensions?

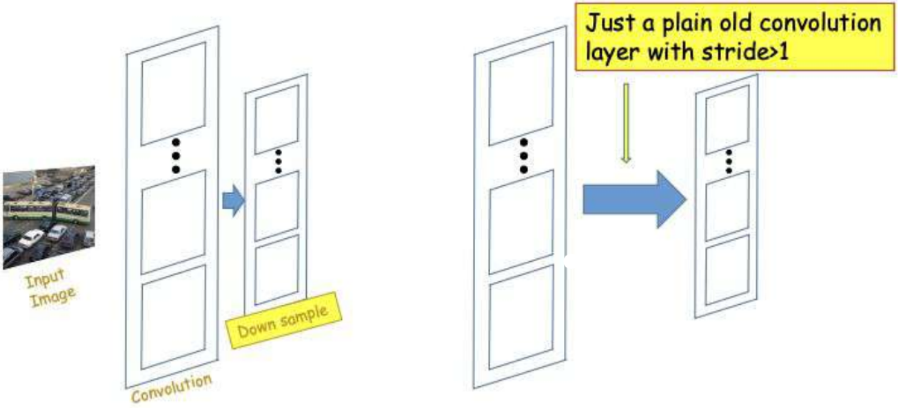
Answer:

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

Parameter Setting

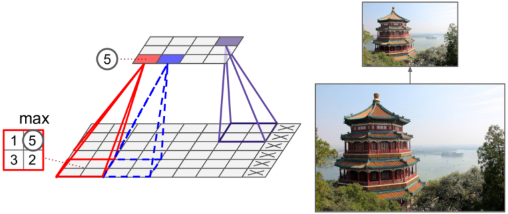
- Pooling takes in a volume of size $W_1 \times H_1 \times D_1$
- Requires two hyperparameters:
  - Their spatial extent P,
  - The stride S.
- Produces an output volume of size $W_2 \times H_2 \times D_2$, where:
  - $W_2 = \frac{(W_1 - P)}{S} + 1$
  - $H_2 = \frac{(H_1 - P)}{S} + 1$
  - $D_2 = D_1$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

Downsampling



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

Pooling Summary



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

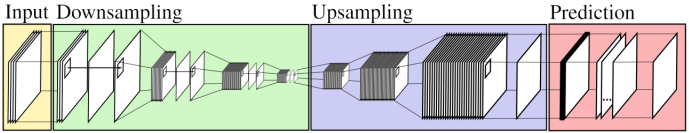
- The goal is to sub-sample the input to reduce:
  - Computational load
  - Memory usage
  - Number of parameters
  - Risk of overfitting

Question

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

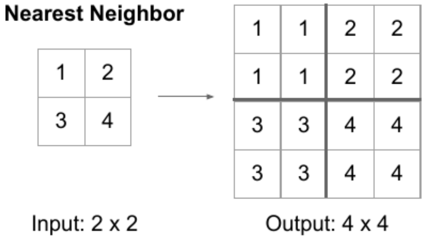
Up-sampling CNN

- Resizing feature maps is a **common** operation in neural networks, especially those used for image segmentation tasks.
- This architecture is often referred to as an **Encoder-Decoder** network.



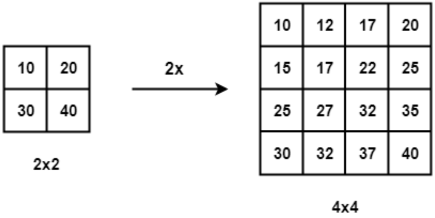
Nearest Neighbors

- **Nearest Neighbors**: Nearest Neighbors involves copying an input pixel value to the K-nearest neighboring pixels, with K based on the expected output.



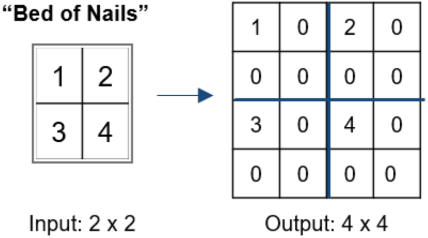
Bilinear Interpolation

- **Bilinear Interpolation**: In Bilinear Interpolation, the four nearest pixel values are used to compute a weighted average based on their distances, resulting in a smoothed output.



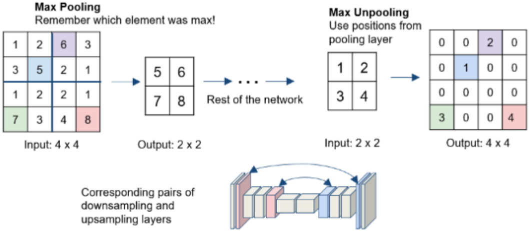
Bed Of Nails

- **Bed Of Nails**: In this method, the input pixel value is copied to the corresponding position in the output image, with zeros filling the remaining positions.



Max-Unpooling

- **Max-Unpooling**: In max-unpooling, the index of the **maximum value** is saved for each max-pooling layer during encoding. During decoding, the saved index is used to map the input pixel to its original position, with zeros filling all other positions.

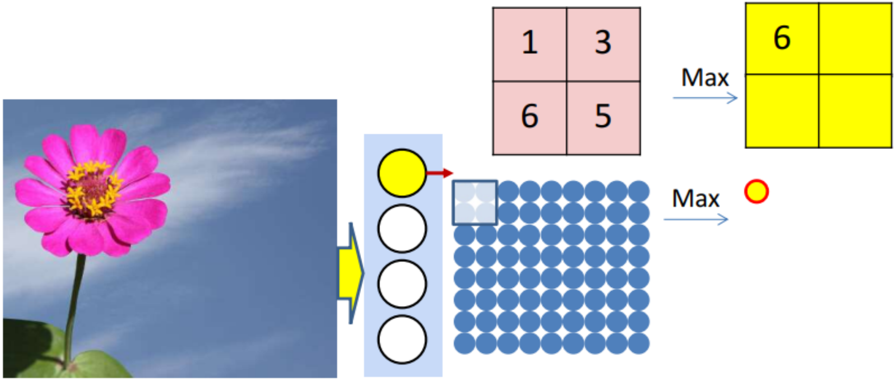


Backpropagation With Pooling Layers

In our previous discussions, we explored the process of backpropagation in CNNs without considering pooling layers. Now, let's discuss how to adapt our algorithm to include pooling layers.

- The primary task is to handle the gradients effectively during backpropagation through pooling layers.
- In both cases, the gradients from the next layer are **passed back** to the previous layer through the pooling operation.

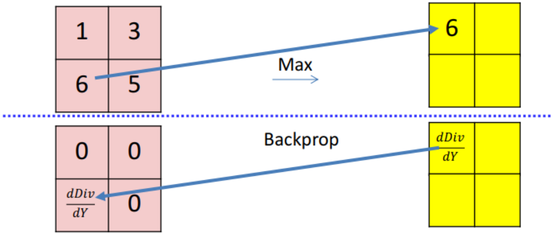
Case 1



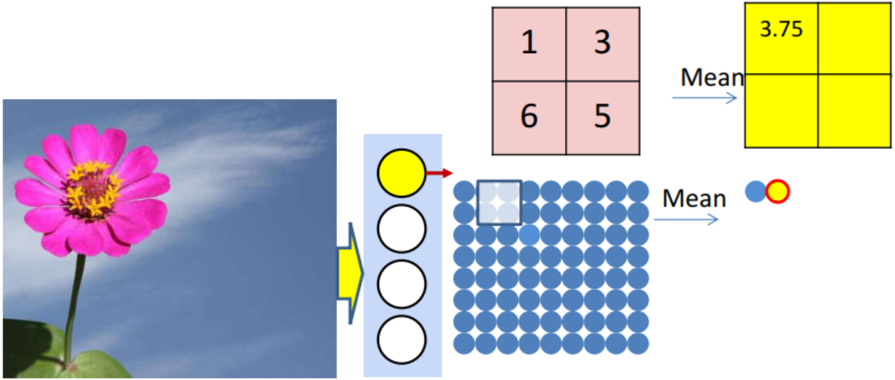
Case 1

For **Max Pooling:**

- The gradient is propagated only through the **indices of the maximum values** identified during the forward pass.
- All other positions receive a gradient of **zero**.



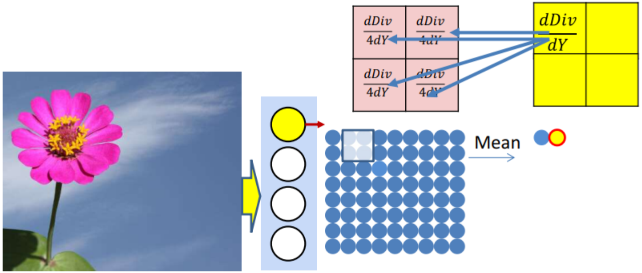
Case 2



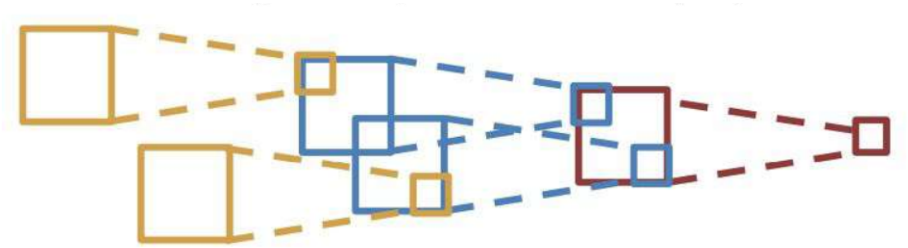
Case 2

For **Average Pooling:**

- The gradients are **uniformly distributed** across the pooled region.



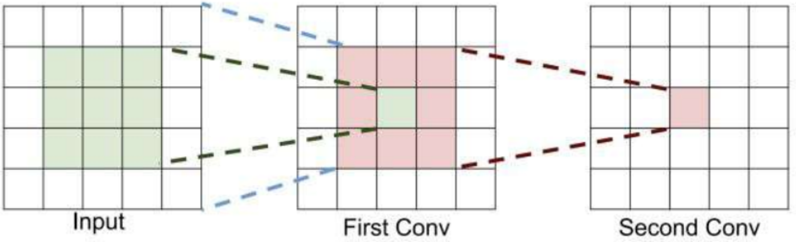
Receptive Field



- **Receptive Field:** How large is the region in the **input or previous layer** seen by a neuron on the n-th convolutional layer?
- In a convolution with kernel size K, each element in the next layer is based on a $K \times K$ **receptive field** from 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 $1+L\cdot(K-1)$.
- **Challenge**: For large images, many layers are required for each output to capture the entire image.
  - **Solution: Downsample** within the network using strides and pooling layers.

Power Of Small Filters

Assuming an input size of $H \times W \times C$, and convolutions are used with C filters to preserve depth (stride 1, with padding to maintain H and W dimensions).

one CONV with 7 x 7 filters

Number of weights
$= C \times (7 \times 7 \times C) = \mathbf{49}C^2$

three CONV with 3 x 3 filters

Number of weights
$= 3 \times C \times (3 \times 3 \times C) = \mathbf{27}C^2$

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 $S = 1$.

- Determine the receptive field size after each layer, beginning with 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: $5+(3-1)=7$
- Consequently, the receptive field size after the third layer is 7.
- A larger receptive field allows neurons in deeper layers to capture more context from the input, crucial for recognizing higher-level patterns.

Inductive Bias In CNNs

Inductive Bias:
The assumptions a model uses to generalize from training data to new, unseen data.

Key Features of Inductive Bias in CNNs:

- **Weight Sharing:**
  A single filter is applied across various regions of the input, significantly decreasing the parameter count.
- **Locality:**
  CNNs utilize small filters (e.g., 3×3) that concentrate on local regions, aligning well with image data where local structures are significant.
- CNNs are more sample-efficient than FCNs due to their inductive biases.

Contributions

- **This slide was prepared with contributions from:**
  - Ali Aghayari
  - Behrooz Azarkhalili
  - Arian Amani
  - Hamidreza Yaghoubi

[1] M. Soleymani Baghshah, “Deep 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.