

CS181: Computer Vision

CNN Components

Building/Programming a CNN using PyTorch

Thursday, September 10th, 2026



Md Alimoor Reza

Associate Professor of Computer Science

What type of recognition problem?



Object Recognition

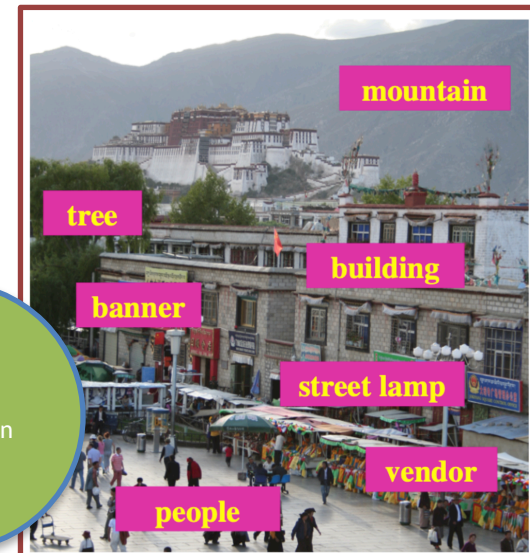


What type of recognition problem?



Single label classification

What type of recognition problem?



Multi-label classification

What type of recognition problem?



Scene and context classification

Recap: Current focus is on object classification using deep learning models

- Various types of **deep learning model based** — more specifically neural networks — can be employed to categorize an image into distinct classes
 - **Multilayer Perceptrons (MLP):** is the simplest type of neural network. It consists of perceptrons (aka nodes, neurons) arranged in layers
 - **Convolution Neural Network (CNN):** good for computer vision (CV) tasks
 - **Transformers:** rising star DL model; it had its inception in Natural Language Processing domain but is now gradually taking over all other AI domains such as Computer Vision, Audio/Speech, Robotics
 - **Very Recent additions (Early 2024):**
 - Mamba Network
 - Kolmogorov Arnold Networks (KAN)

Recap: In-class Activity#2

How to calculate the output volume size for a convolution?

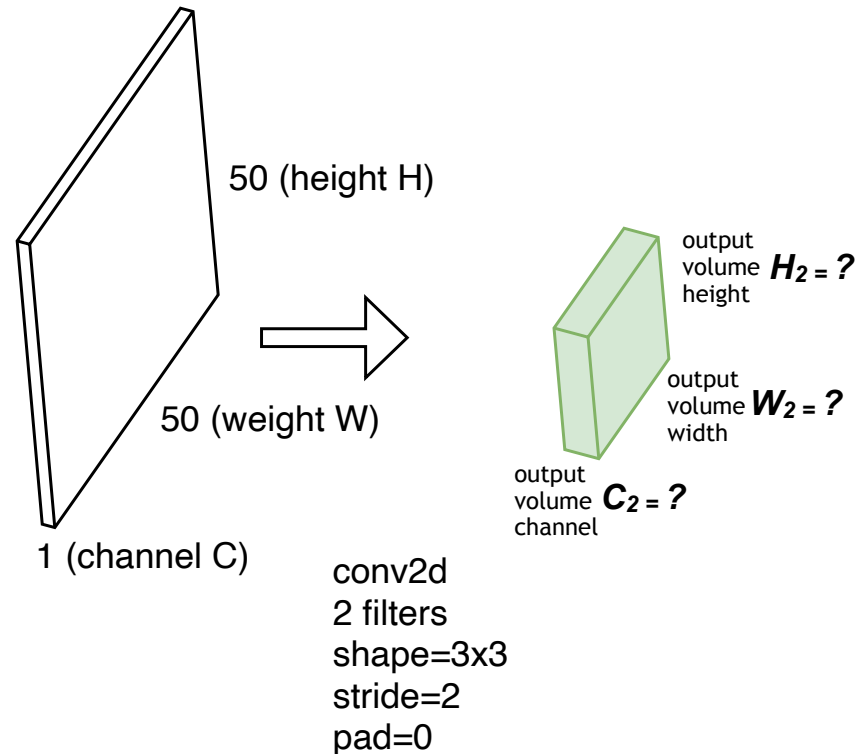
- Input volume size ($C_1 \times H_1 \times W_1$) = (1 x 50 x 50)
- Filter size/receptive field is ($F \times F$) = (3x3)
- Spatial stride $S=2$
- Padding size $P=0$
- Number of filters $K=2$

Output volume ($C_2 \times H_2 \times W_2$)

$$W_2 = \frac{(W_1 - F + 2P)}{S} + 1$$

$$H_2 = \frac{(H_1 - F + 2P)}{S} + 1$$

$$C_2 = K$$



Recap: Answer In-class Activity#2

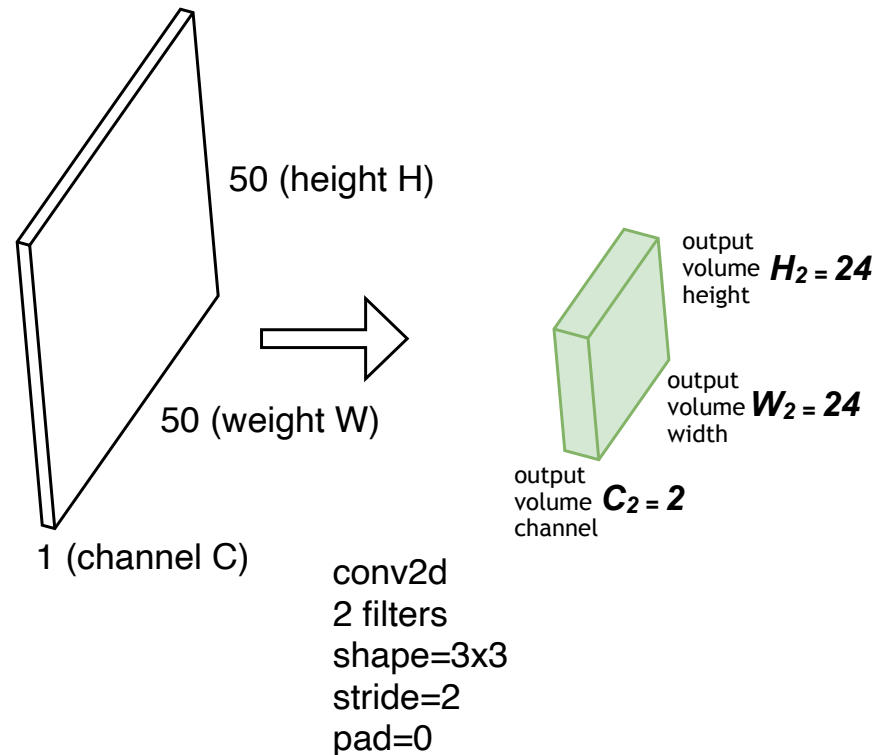
- Input volume size ($C_1 \times H_1 \times W_1$) = (1 x 50 x 50)
- Filter size/receptive field is ($F \times F$) = (3x3)
- Spatial stride $S=2$
- Padding size $P=0$
- Number of filters $K=2$

Output volume ($C_2 \times H_2 \times W_2$)

$$W_2 = \frac{(W_1 - F + 2P)}{S} + 1$$

$$H_2 = \frac{(H_1 - F + 2P)}{S} + 1$$

$$C_2 = K$$



Recap: In-class Activity#2

How to calculate the output volume size for a convolution?

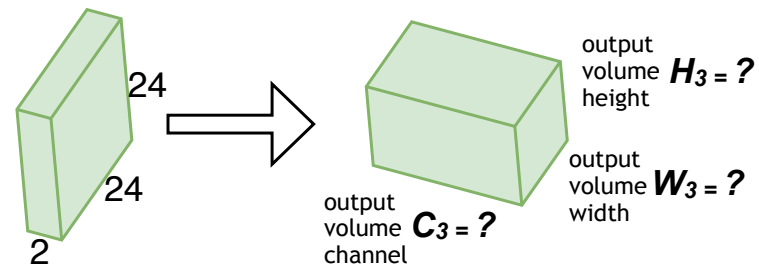
- Input volume size ($C_2 \times H_2 \times W_2$) = (2 x 24 x 24)
- Filter size/receptive field is ($F \times F$) = (3x3)
- Spatial stride $S=2$
- Padding size $P=0$
- Number of filters $K=4$

Output volume ($C_3 \times H_3 \times W_3$)

$$W_3 = \frac{(W_2 - F + 2P)}{S} + 1$$

$$H_3 = \frac{(H_2 - F + 2P)}{S} + 1$$

$$C_3 = K$$



conv2d
4 filters
shape=3x3
stride=2
pad=0

Recap: Answer In-class Activity#2

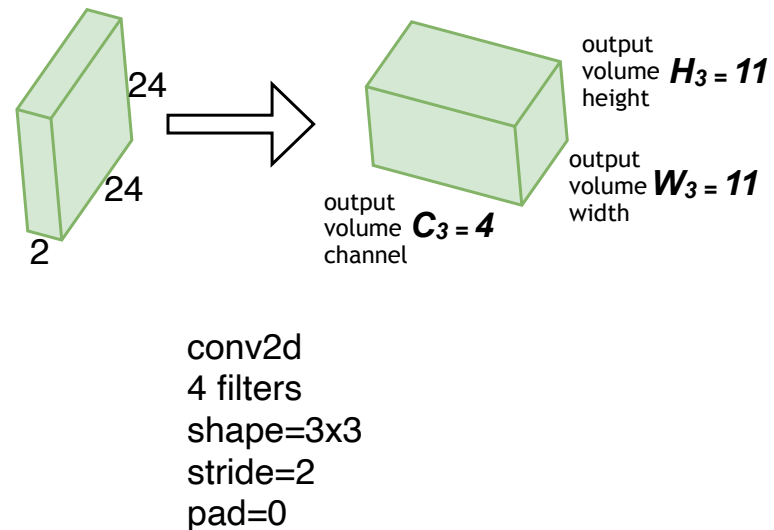
- Input volume size ($C_2 \times H_2 \times W_2$) = (2 x 24 x 24)
- Filter size/receptive field is ($F \times F$) = (3x3)
- Spatial stride $S=2$
- Padding size $P=0$
- Number of filters $K=4$

Output volume ($C_3 \times H_3 \times W_3$)

$$W_3 = \frac{(W_2 - F + 2P)}{S} + 1$$

$$H_3 = \frac{(H_2 - F + 2P)}{S} + 1$$

$$C_3 = K$$



Recap: In-class Activity#2

How to calculate the output volume size for a convolution?

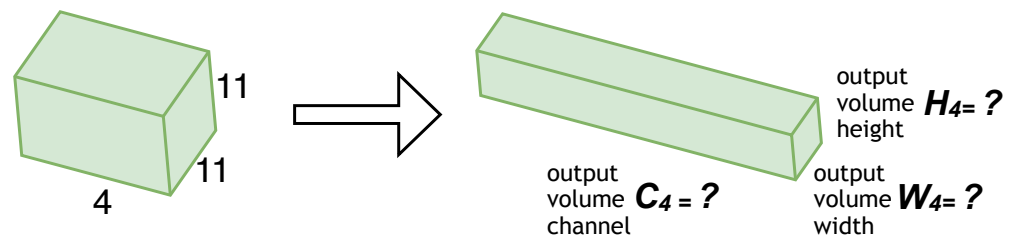
- Input volume size ($C_3 \times H_3 \times W_3$) = (4 x 11 x 11)
- Filter size/receptive field is ($F \times F$) = (3x3)
- Spatial stride $S=2$
- Padding size $P=0$
- Number of filters $K=8$

Output volume ($C_4 \times H_4 \times W_4$)

$$W_4 = \frac{(W_3 - F + 2P)}{S} + 1$$

$$H_4 = \frac{(H_3 - F + 2P)}{S} + 1$$

$$C_4 = K$$



conv2d
8 filters
shape=3x3
stride=2
pad=0

Recap: Answer In-class Activity#2

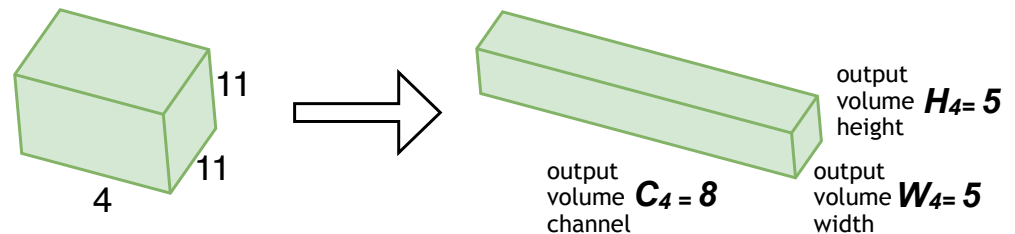
- Input volume size ($C_3 \times H_3 \times W_3$) = (4 x 11 x 11)
- Filter size/receptive field is ($F \times F$) = (3x3)
- Spatial stride $S=2$
- Padding size $P=0$
- Number of filters $K=8$

Output volume ($C_4 \times H_4 \times W_4$)

$$W_4 = \frac{(W_3 - F + 2P)}{S} + 1$$

$$H_4 = \frac{(H_3 - F + 2P)}{S} + 1$$

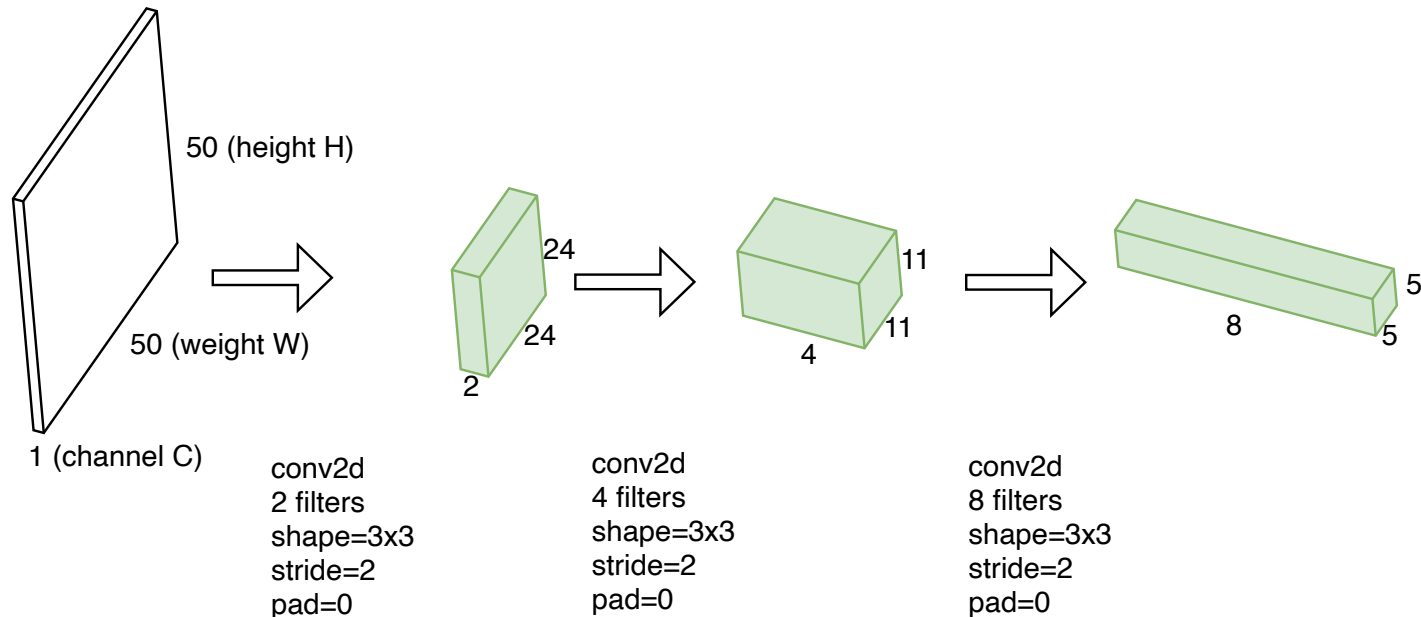
$$C_4 = K$$



conv2d
8 filters
shape=3x3
stride=2
pad=0

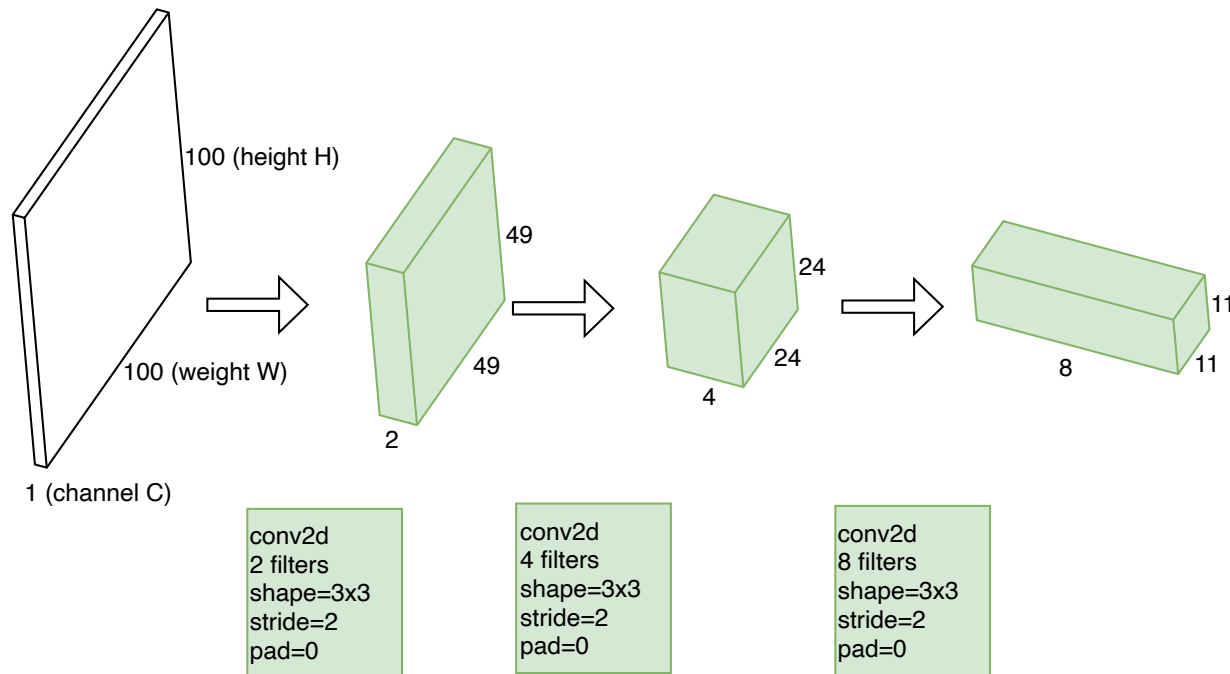
A convolutional neural network (CNN) with 3 Layers

- Now, if we want to combine the three convolution layers in a sequential manner, and apply a (50x50) input image we get a 3-layer CNN as follows:



A convolutional neural network (CNN) with 3 Layers

- Now, if we want to combine the three convolution layers in a sequential manner, and apply a (100x100) input image we get a 3-layer CNN as follows:



Recap: CNN implementation

- Go to Blackboard and let's work on the notebook titled “CNN for recognition using PyTorch”
- Link shown below:

https://github.com/alimoorreza/CS181-fall26-notes/blob/main/day06_cnn_for_recognition.ipynb

Topics

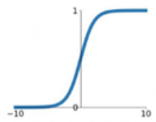
- **Convolutional Neural Network (CNN):** good for computer vision (CV) tasks
 - Convolution operation
 - **Nonlinearity** (may or may not include this after a convolution)
 - Pooling operation (may or may not include this after a convolution)
 - CNN: convolutional layer + nonlinearity + pooling layer
- Popular CNNs
 - LeNet
 - AlexNet
 - VGG
 - ResNet

Nonlinear Function

- Just like an MLP, each convolutional output goes through a non-linear function such as **Sigmoid**, **Tanh**, or Rectified Linear Unit (**ReLU**)

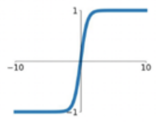
Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



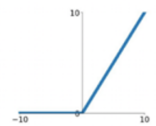
tanh

$$\tanh(x)$$



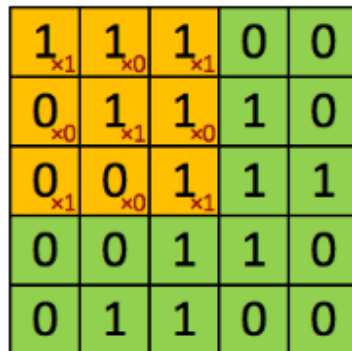
ReLU

$$\max(0, x)$$

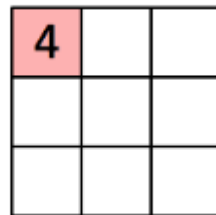


$$\text{convolution} = 1 * 1 + 1 * 0 + 1 * 1 + 0 * 0 + 1 * 1 + 1 * 0 + 0 * 1 + 0 * 0 + 1 * 1 = 4$$

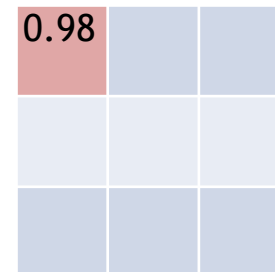
$$\text{Sigmoid}(4) = \frac{1}{1 + \exp(-4)} = 0.98$$



Image



Convolved
Feature



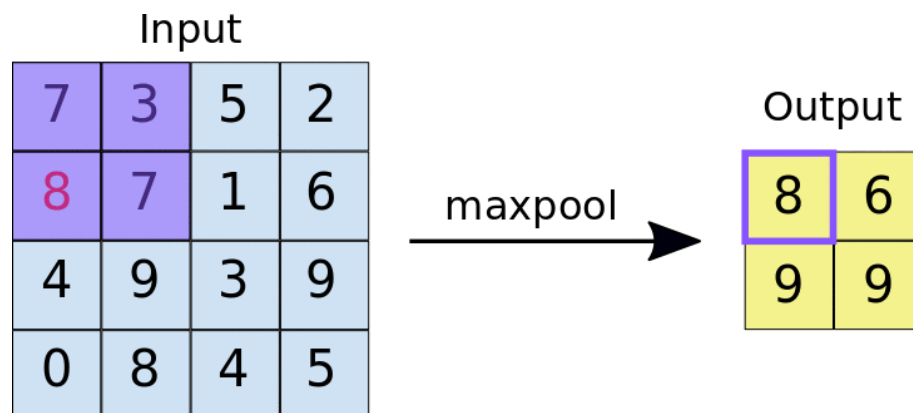
Sigmoid

Topics

- **Convolutional Neural Network (CNN):** good for computer vision (CV) tasks
 - Convolution operation
 - Nonlinearity (may or may not include this after a convolution)
 - **Pooling operation** (may or may not include this after a convolution)
 - CNN: convolutional layer + nonlinearity + pooling layer
- Popular CNNs
 - LeNet
 - AlexNet
 - VGG
 - ResNet

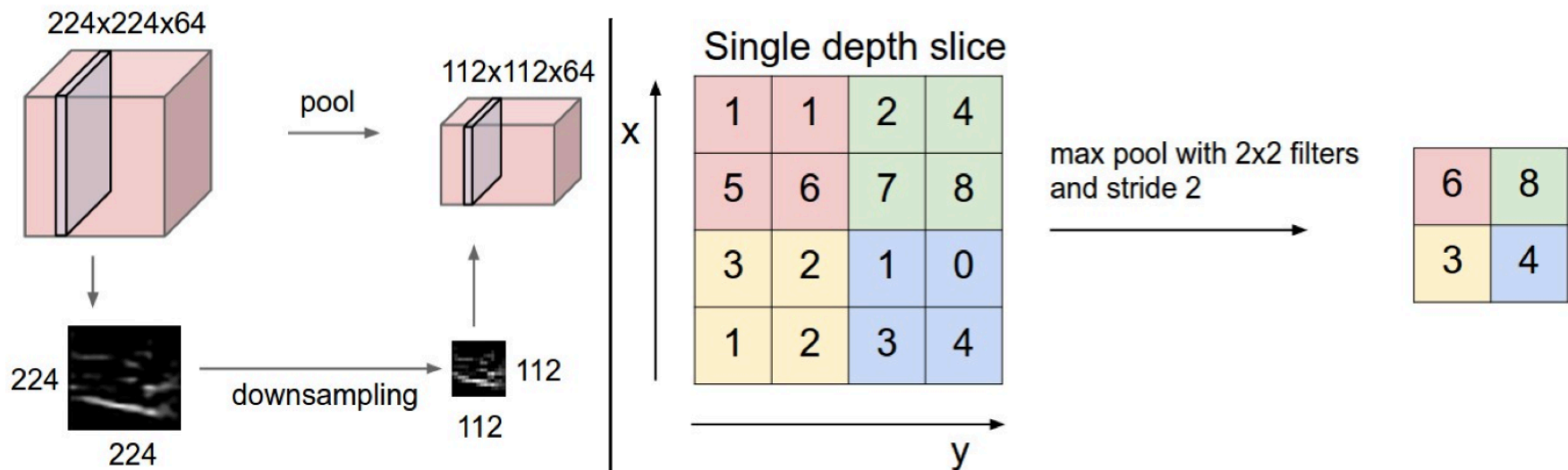
Pooling Operation

- Image data can get computationally inefficient, really quickly. To avoid this, we often toss in a layer that helps us to **summarize** and **downsample** the data
- In classical CNN, we find another useful operation called **pooling operation**
- A common pooling operation is **max pooling**, and its goal is to locally summarize the convolution. It performs something like a convolution, but rather than taking the dot product, it **takes the maximum element in the filter area**



Pooling Operation

- Pooling operation downsamples the volume spatially, **independently in each depth slice of the input volume**
- Besides **max pooling**, other pooling operations include: **sum pooling**, **average pooling**



How to calculate the output volume size for a pooling operation?

- Input volume size = $(C_1 \times H_1 \times W_1)$
 - Filter size/receptive field is $(F \times F)$
 - Spatial stride size S
 - Padding size P
- Spatial sizes of the output volume $(C_2 \times H_2 \times W_2)$

$$W_2 = \frac{(W_1 - F + 2P)}{S} + 1$$

$$H_2 = \frac{(H_1 - F + 2P)}{S} + 1$$

$$C_2 = C_1$$

- Please note: number of filter weight parameters = **zero parameter** since it computes a fixed function eg *max()* or *average()*

Group Activity

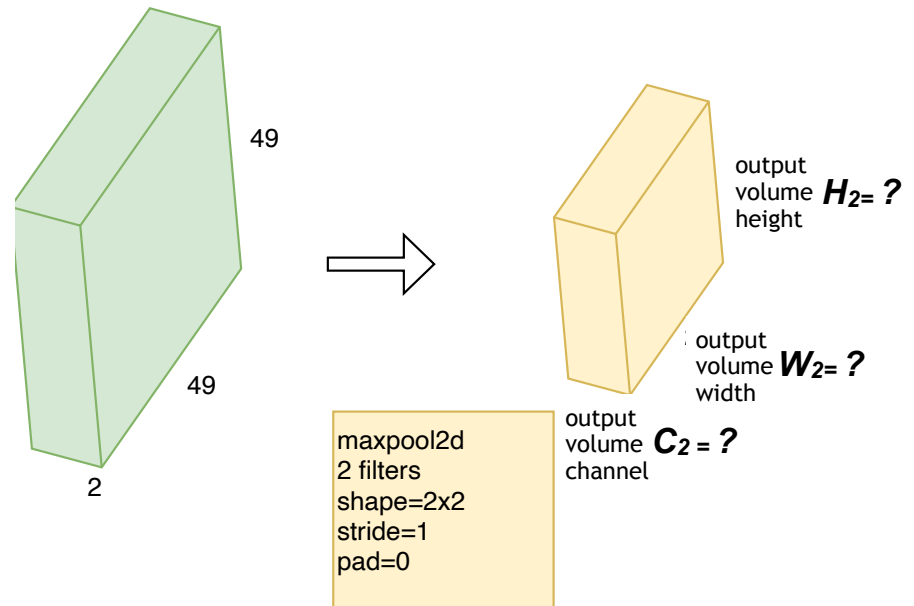
How to calculate the output volume size for a pooling operation?

- Input volume size $(C_1 \times H_1 \times W_1) = (2 \times 49 \times 49)$
- Filter size/receptive field is $(F \times F) = (2 \times 2)$
- Spatial stride $S=2$
- Padding size $P=0$
- Number of filters $K=2$

$$W_2 = \frac{(W_1 - F + 2P)}{S} + 1$$

$$H_2 = \frac{(H_1 - F + 2P)}{S} + 1$$

$$C_2 = C_1$$



Please note: number of filter weight parameters = **zero parameter** since it computes a fixed function eg `max()` or `average()`

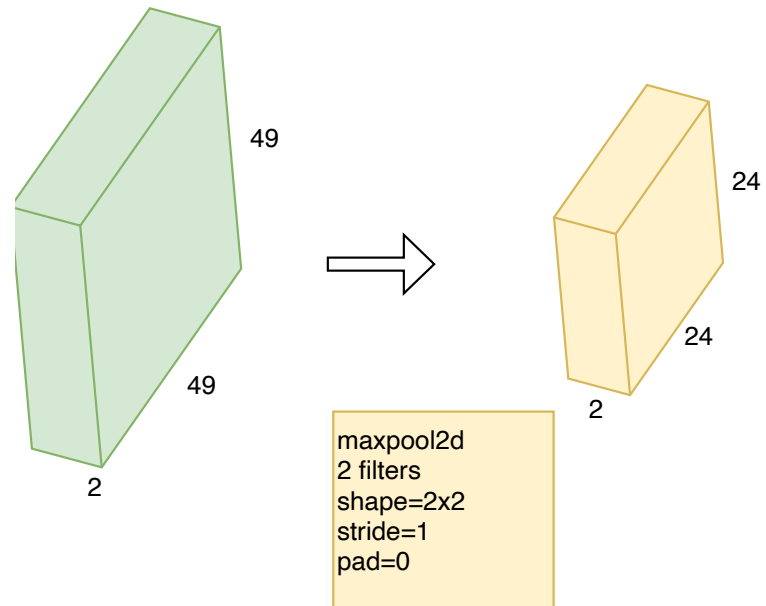
Answer: Group Activity

- Input volume size $(C_1 \times H_1 \times W_1) = (2 \times 49 \times 49)$
- Filter size/receptive field is $(F \times F) = (2 \times 2)$
- Spatial stride $S=2$
- Padding size $P=0$
- Number of filters $K=2$

$$W_2 = \frac{(W_1 - F + 2P)}{S} + 1$$

$$H_2 = \frac{(H_1 - F + 2P)}{S} + 1$$

$$C_2 = C_1$$



Please note: number of filter weight parameters = **zero parameter** since it computes a fixed function eg max() or average()

Group Activity

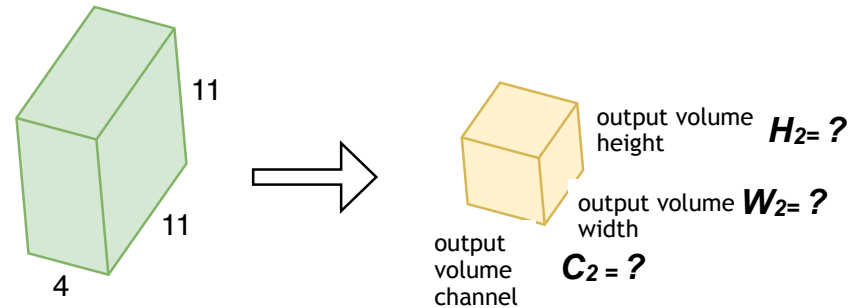
How to calculate the output volume size for a pooling operation?

- Input volume size ($C_1 \times H_1 \times W_1$) = (4 x 11 x 11)
- Filter size/receptive field is ($F \times F$) = (2x2)
- Spatial stride $S=2$
- Padding size $P=0$
- Number of filters $K=4$

$$W_2 = \frac{(W_1 - F + 2P)}{S} + 1$$

$$H_2 = \frac{(H_1 - F + 2P)}{S} + 1$$

$$C_2 = C_1$$



maxpool2d
4 filters
shape=2x2
stride=1
pad=0

Please note: number of filter weight parameters = **zero parameter** since it computes a fixed function eg `max()` or `average()`

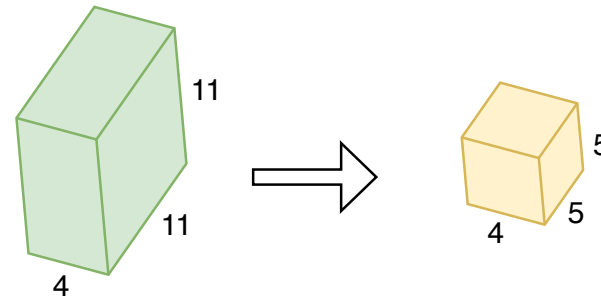
Answer: Group Activity

- Input volume size $(C_1 \times H_1 \times W_1) = (4 \times 11 \times 11)$
- Filter size/receptive field is $(F \times F) = (2 \times 2)$
- Spatial stride $S=2$
- Padding size $P=0$
- Number of filters $K=4$

$$W_2 = \frac{(W_1 - F + 2P)}{S} + 1$$

$$H_2 = \frac{(H_1 - F + 2P)}{S} + 1$$

$$C_2 = C_1$$



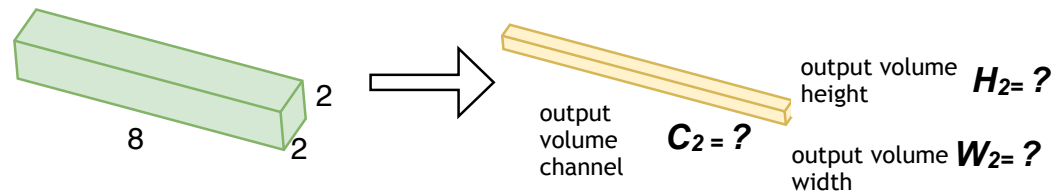
maxpool2d
4 filters
shape=2x2
stride=1
pad=0

Please note: number of filter weight parameters = **zero parameter** since it computes a fixed function eg `max()` or `average()`

Group Activity

How to calculate the output volume size for a pooling operation?

- Input volume size ($C_1 \times H_1 \times W_1$) = (8 x 2 x 2)
- Filter size/receptive field is ($F \times F$) = (2x2)
- Spatial stride $S=2$
- Padding size $P=0$
- Number of filters $K=8$



$$W_2 = \frac{(W_1 - F + 2P)}{S} + 1$$

$$H_2 = \frac{(H_1 - F + 2P)}{S} + 1$$

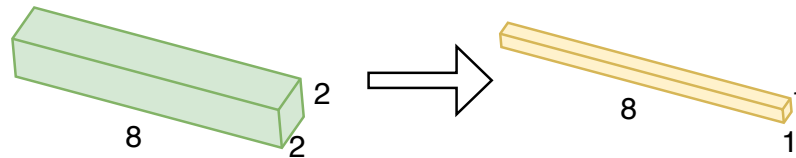
$$C_2 = C_1$$

maxpool2d
8 filters
shape=2x2
stride=1
pad=0

Please note: number of filter weight parameters = **zero parameter** since it computes a fixed function eg max() or average()

Answer: Group Activity

- Input volume size ($C_1 \times H_1 \times W_1$) = (8 x 2 x 2)
- Filter size/receptive field is ($F \times F$) = (2x2)
- Spatial stride $S=2$
- Padding size $P=0$
- Number of filters $K=8$



$$W_2 = \frac{(W_1 - F + 2P)}{S} + 1$$

$$H_2 = \frac{(H_1 - F + 2P)}{S} + 1$$

$$C_2 = C_1$$

maxpool2d
8 filters
shape=2x2
stride=1
pad=0

Please note: number of filter weight parameters = **zero parameter** since it computes a fixed function eg max() or average()

A CNN with ReLU activations + max-pooling layers

- Let's modify our last CNN as shown below by inserting a *ReLU* activation layer and a *max-pooling* layer in between each pair of adjacent *conv2d* layers and connect them sequentially, and then apply a (100x100) input image through the CNN.

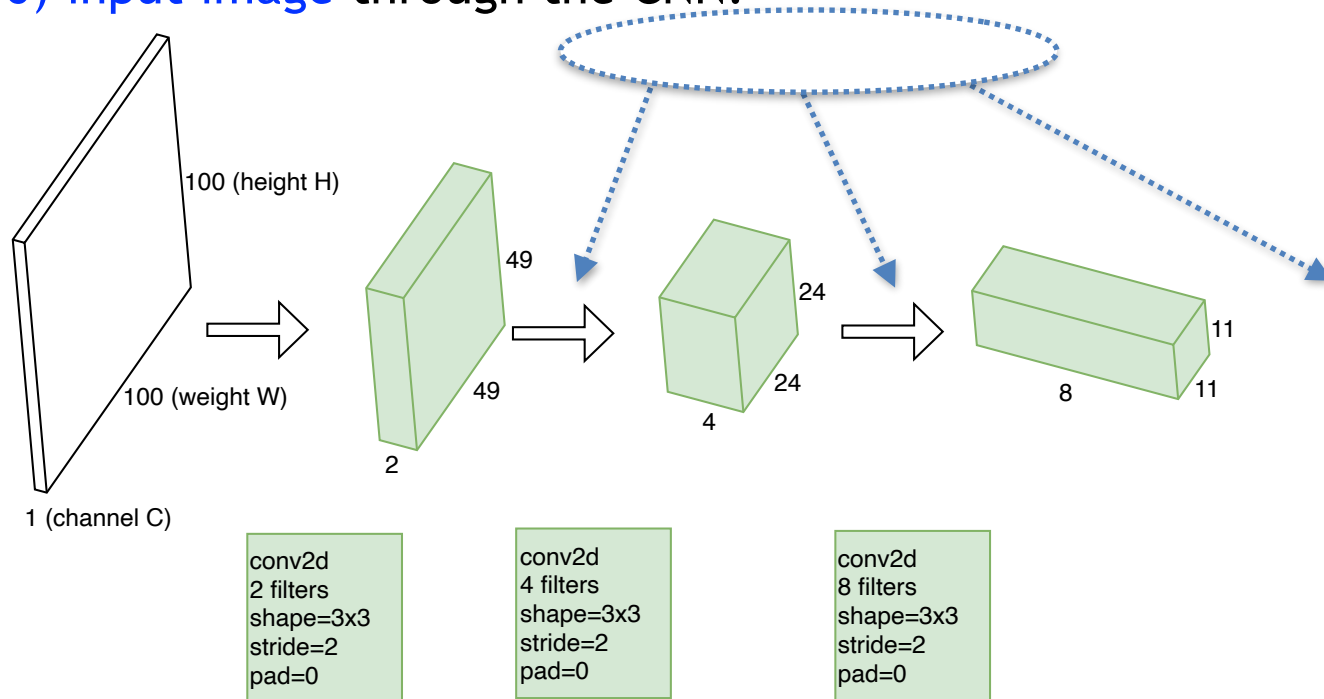


Figure: CNN **before** inserting ReLU + max-pooling layers

CNN with ReLU activations + max-pooling layers

- Let's modify our last CNN as shown below by inserting a *ReLU* activation layer and a *max-pooling* layer in between each pair of adjacent *conv2d* layers and connect them sequentially, and apply a (100x100) input image like before.

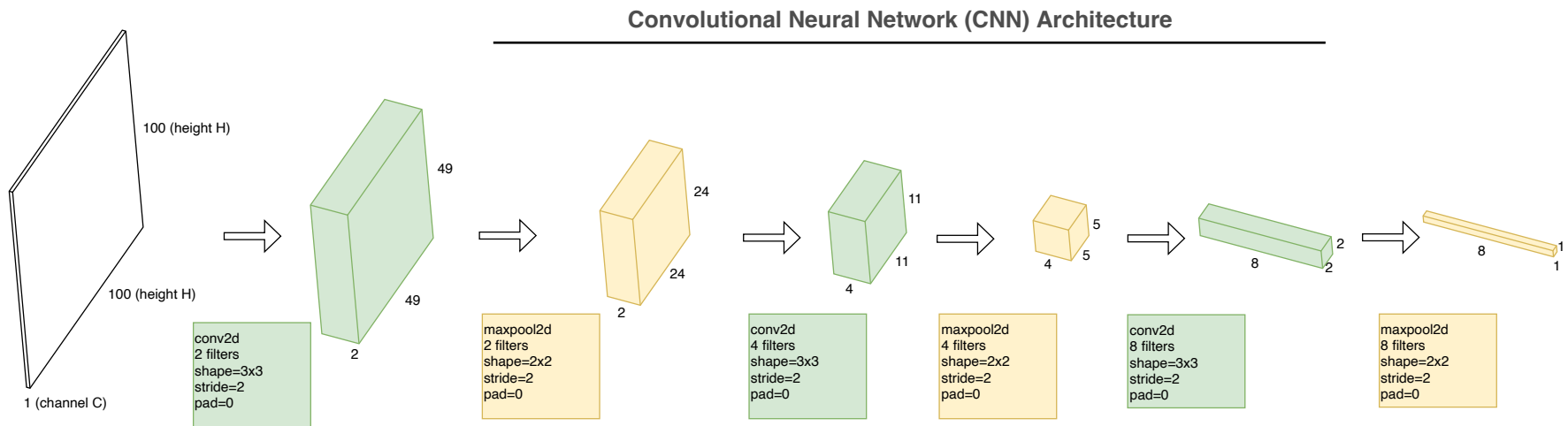


Figure: CNN **after** inserting ReLU + max-pooling layers

Coding demo: CNN with ReLU activations + max-pooling layers

- Go to Blackboard and let's work on the notebook titled “CNN with activation and pooling layer using PyTorch”
- Link shown below:

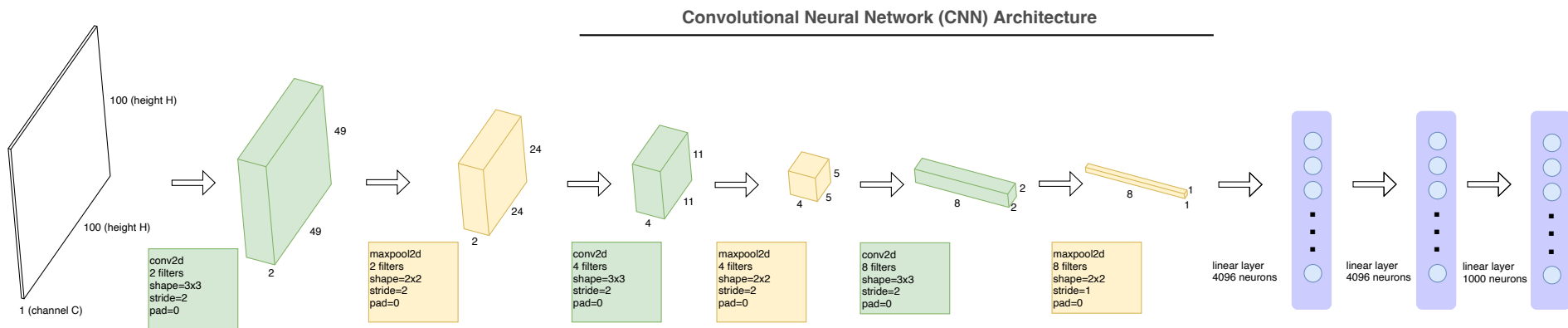
https://github.com/alimoorreza/CS181-fall26-notes/blob/main/day07_cnn_with_activation_and_pooling_layers.ipynb

Topics

- **Convolutional Neural Network (CNN):** good for computer vision (CV) tasks
 - Convolution operation
 - Nonlinearity (may or may not include this after a convolution)
 - Pooling operation (may or may not include this after a convolution)
 - **CNN: convolutional layer + nonlinearity + pooling layer**
- Popular CNNs
 - LeNet
 - AlexNet
 - VGG
 - ResNet

CNN: A composition of convolutional (+non-linear activation + pooling) layers

- We've talked about **image data**, **convolutions**, **nonlinearity**, **max pooling**, and how they are related to some computer vision tasks. Let's connect the dots
 - input is an image (in this case a color image, so 3 channels—red, green, and blue)
 - there are several filters, not just one.
 - **conv2d** layers with **ReLU** are often followed by **maxpool2d**
 - towards the end of the model, we switch to fully connected layer (or **linear** layer)
 - we have as many output nodes as we have classes to predict



CNN: A composition of convolutional (+non-linear activation + pooling) layers

- **Big idea:** different filters can be used to extract specific information from the original image
- **Bigger idea:** Instead of using manually made filters for feature extraction, through **deep CNNs we can learn these kernel values** (just like the weights of a traditional NN). These kernels can extract latent features
 - In CNNs, the way we learn is by changing the values in the **filters/kernels**
 - **only for the conv2d layers and no weight learning for ReLU or max-pooling**
 - We have an MLP (at the end) for which we learn values of their connections by changing the **weights**
 - We learn these **filters** + **weights** using back propagation algorithm (PyTorch handles this for us)

