Basics of CNN

Kaiser Hamid

December 16, 2024

Contents

1 Introduction **2**

2 Initial Setup **2**

3 Formulas for Output Size Calculation **2**

4 The Input Image **3**

5 First Convolutional Layer (Conv1) **3**

6 Convolution Operation Example **4**

7 Second Convolutional Layer (Conv2) **4**

8 Max Pooling Layer **5**

9 Fully Connected Layer **5**

10 Summary of Parameter Counts **5**

11 How Parameters Affect the Network **6**

12 Conclusion **6**

1 Introduction

This article provides a simple step-by-step example of how a Convolutional Neural Network (CNN) processes an input image. We will begin with a single convolutional layer and a max pooling operation, and then connect the result to a fully connected layer. After that, we will briefly introduce a second convolutional layer. The goal is to give a clear and understandable explanation, suitable for someone learning these concepts for the first time.

We will also mention the formulas used to calculate the output sizes after convolution and pooling, considering stride and padding.

2 Initial Setup

- Input image: $8 \times 8 \times 1$ (grayscale).
- First convolutional layer (Conv1):
  - Number of filters: $F_1 = 8$
  - Kernel size: $K = 3 \times 3$
  - Padding: $P = 1$
  - Stride: $S = 1$
- Max pooling layer:
  - Pool size: 2×2
  - Stride: 2
- Fully connected layer at the end with 10 output neurons.

3 Formulas for Output Size Calculation

For a convolutional layer, if the input size is $H \times W \times D$, and we use:

- Kernel size: $K \times K$
- Padding: P
- Stride: S
- Number of filters: F

The output height and width are given by:

$$H_{out} = \frac{H - K + 2P}{S} + 1, \quad W_{out} = \frac{W - K + 2P}{S} + 1$$

The output depth is F.

For a pooling layer with pool size $R \times R$ and stride S_p (no padding):

$$H_{out}^{pool} = \frac{H}{S_p}, \quad W_{out}^{pool} = \frac{W}{S_p}$$

The depth remains the same after pooling.

4 The Input Image

$$I = \begin{bmatrix} 2 & 5 & 1 & 0 & 3 & 8 & 7 & 9 \\ 4 & 1 & 9 & 2 & 5 & 6 & 2 & 0 \\ 3 & 8 & 8 & 5 & 0 & 0 & 1 & 4 \\ 6 & 2 & 1 & 9 & 9 & 3 & 5 & 2 \\ 1 & 0 & 7 & 5 & 6 & 4 & 8 & 8 \\ 9 & 1 & 0 & 2 & 3 & 7 & 1 & 1 \\ 5 & 6 & 3 & 8 & 8 & 8 & 0 & 2 \\ 4 & 4 & 4 & 1 & 1 & 0 & 9 & 3 \end{bmatrix}$$

5 First Convolutional Layer (Conv1)

Using the formula for convolution output:

$$H_{out} = \frac{8 - 3 + 2 \times 1}{1} + 1 = 8$$

$$W_{out} = \frac{8 - 3 + 2 \times 1}{1} + 1 = 8$$

The output depth is equal to the number of filters:

$$D_{out} = F_1 = 8$$

So, after Conv1:

$$8 \times 8 \times 8$$

Each filter has $3 \times 3 \times 1 = 9$ weights plus 1 bias, total 10 parameters. With 8 filters:

$$\text{Conv1 parameters} = 8 \times 10 = 80$$

6 Convolution Operation Example

Padding with $P = 1$ adds a border of zeros, making the padded input 10×10. Consider one filter:

$$K = \begin{bmatrix} 1 & 0 & -1 \\ 0 & 1 & 0 \\ -1 & 0 & 1 \end{bmatrix}, \quad b = 1.$$

At the top-left position, the 3×3 patch from the padded input is:

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 5 \\ 0 & 4 & 1 \end{bmatrix}.$$

Multiply element-wise, sum them up, and add the bias:

$$(0 \cdot 1) + (0 \cdot 0) + (0 \cdot (-1)) + (0 \cdot 0) + (2 \cdot 1) + (5 \cdot 0) + (0 \cdot (-1)) + (4 \cdot 0) + (1 \cdot 1) + b$$

$$= 2 + 1 + 1 = 4.$$

This is the output at the top-left corner for this filter. Sliding over the entire image produces an 8×8 feature map. Doing this for all 8 filters results in $8 \times 8 \times 8$ output.

7 Second Convolutional Layer (Conv2)

Now we apply another convolutional layer to the $8 \times 8 \times 8$ output of Conv1. Suppose:

- $F_2 = 16$ filters
- Kernel size $= 3 \times 3$
- Padding $= 1$
- Stride $= 1$

Using the formula:

$$H_{out} = \frac{8 - 3 + 2 \times 1}{1} + 1 = 8, \quad W_{out} = 8$$

Depth $= F_2 = 16$.

After Conv2:

$$8 \times 8 \times 16.$$

8 Max Pooling Layer

Apply 2×2 max pooling with stride 2:

$$H_{out}^{pool} = \frac{8}{2} = 4, \quad W_{out}^{pool} = \frac{8}{2} = 4$$

Depth remains 16, so after pooling:

$$4 \times 4 \times 16.$$

9 Fully Connected Layer

Flatten the $4 \times 4 \times 16$ output into 256 values.

Connect to 10 output neurons. Each neuron has 256 weights plus 1 bias:

$$257 \text{ parameters per neuron}, \quad 10 \text{ neurons} \implies 2570 \text{ parameters}.$$

10 Summary of Parameter Counts

- Conv1: 80 parameters.
- Conv2: More parameters (not calculated in detail here, but formula is the same as Conv1 with adjusted input depth).
- Pooling: No parameters.
- Fully Connected Layer: 2570 parameters.

11 How Parameters Affect the Network

- Increasing the number of filters increases representational power but also increases parameters.
- Larger kernel sizes capture larger patterns, increasing the number of weights.
- Changing stride affects the output size. A larger stride reduces spatial dimensions more quickly.
- Padding can preserve spatial dimensions, allowing more layers without rapid size reduction.

12 Conclusion

Starting from an $8 \times 8 \times 1$ input, we applied a convolutional layer that preserved the spatial size (due to padding and stride settings), added another convolutional layer to increase depth, then used a pooling layer to reduce spatial dimensions, and finally connected to a fully connected layer with 10 outputs.

By following these steps and formulas, we understand how each layer transforms the data and how changing parameters like kernel size, padding, stride, and number of filters affects the network's complexity and feature extraction capabilities.