Convolutional Neural Networks

Jerry Peng

1 Convolutional Layer

The convolutional layer is the core building block of a Convolutional Neural Network (CNN). It performs a mathematical operation called convolution, which is fundamentally a weighted sum of the input data. In the context of CNNs, this operation is used to extract features from the input image, like edges, textures, or more complex patterns in deeper layers.

1.1 Key Concepts

Filters/Kernels A filter (or kernel) is a small matrix of weights. Each filter is designed to detect specific features in the input data (like edges, colors, etc.). The size of the filter is usually much smaller than the input data. Common sizes are $3x3$, $5x5$, etc.

Stride Stride is the number of pixels by which we slide the filter over the input matrix. A stride of 1 moves the filter one pixel at a time, while a stride of 2 moves it two pixels, and so on.

Padding Padding involves adding extra pixels around the input image. This is done to control the spatial size of the output volume, allowing for more control over the depth of the network.

Feature Maps The result of one filter applied to the previous layer is a 2D activation map (also called a feature map) that gives the responses of that filter at every spatial position.

1.2 Mathematical Description

Assuming we have an input image as a 2D matrix with size of $H \times W$, and a 2D matrix as the filter with size of $F \times F$. The stride is S, and the padding is P. Then the filter works like the Fig 1 shows.

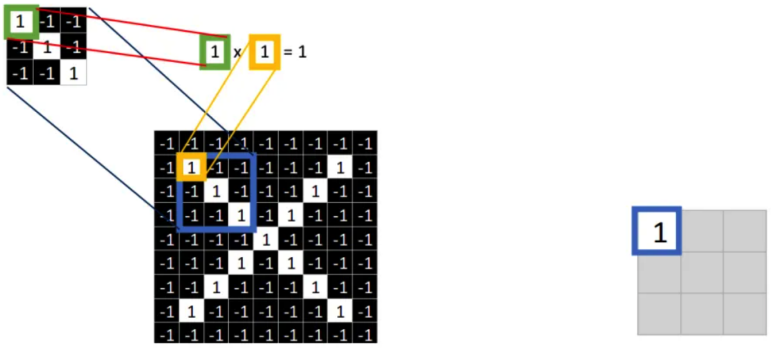

Figure 1: CNN Filter

In this process, the filter is overlayed on the part of image. Then we perform element-wise multiplication between the filter and the part of the image it covers. In Fig 1, we can see the number in the green box from the filter is multiplied by the number in the yellow box from the image, and the result is stored in a new matrix. After all elements are multiplied with the pixel in the image, we will either average or sum the new matrix to get a new number, and store this new number to the feature map, as the Fig 2 shows.

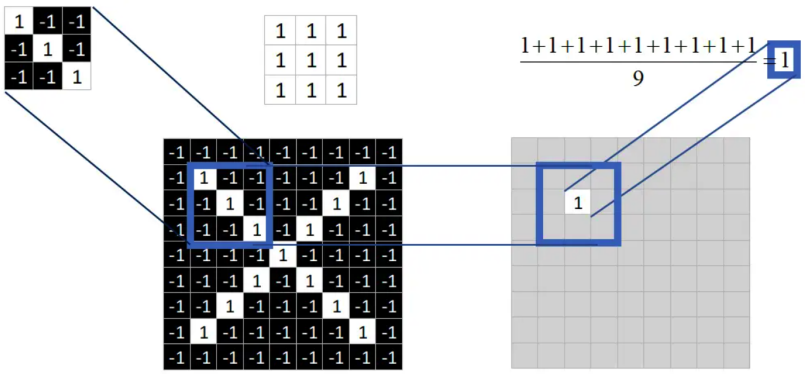

Figure 2: CNN Feature Map

Then the filter will slide by S pixels over the input and repeat the process for the entire image. The mathematical formula for the convolution operation at a given position (i, j) is

$$(I * F)(i, j) = \sum_{m=0}^{F-1} \sum_{n=0}^{F-1} I(i + m, j + n) \times F(m, n)$$

where $I(i+m, j+n)$ is the pixel value of the input at position $(i+m, j+n)$ and $F(m, n)$ is the value of the filter at position (m, n)

The size of the output feature map can be calculated as:

$$Height = \lfloor \frac{H - F + 2P}{S} + 1 \rfloor$$

$$Width = \lfloor \frac{W - F + 2P}{S} + 1 \rfloor$$

where $\lfloor \rfloor$ represents the floor operations

The Fig 3 shows a completed feture map generated by the steps from above

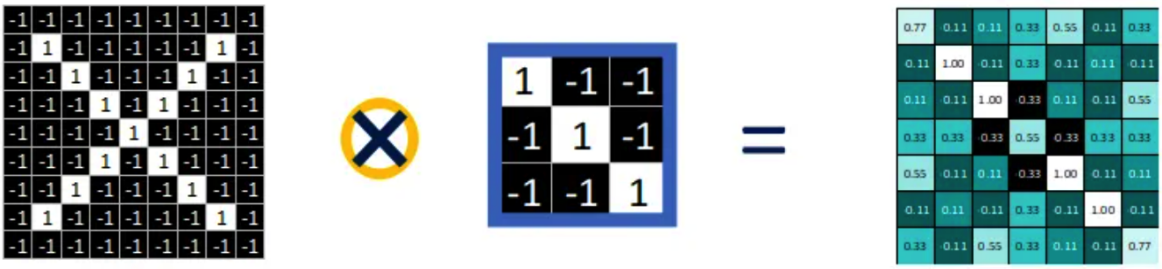

Figure 3: CNN Feature Map with AVG Kernel

2 Pooling Layer

The pooling layer in a Convolutional Neural Network (CNN) is another crucial component that follows the convolutional layers. Its primary function is to progressively reduce the spatial size of the representation, thereby reducing the amount of parameters and computation in the network. This helps in controlling overfitting and also reduces the computational cost.

2.1 Key Concepts

Function of Pooling Pooling layers operate independently on every depth slice of the input and resize it spatially. The most common functions used in pooling are max pooling and average pooling.

Max Pooling Max pooling returns the maximum value from the portion of the image covered by the filter. This is useful for capturing the presence of certain features in the input.

Average Pooling Average pooling, on the other hand, calculates the average of the values in the input portion covered by the filter. This is used less frequently than max pooling.

Stride and Filters Similar to convolutional layers, pooling layers also use filters and strides. However, unlike convolutional layers, pooling layers do not have weights. They use a predefined function (like max or average) over the filter size.

2.2 Mathematical Description

Assuming we have an input image as a 2D matrix with size of $H \times W$, and a 2D matrix as the filter with size of $F \times F$. The stride is S, and the padding is P. The operations for max and average pooling are as follows:

First, for a given position (i, j) in the input,the max pooling operation is defined as:

$$P(i,j) = \max_{0 \le m,n < F} I(i \times S + m, j \times S + n)$$

where $P(i, j)$ is the output of the pooling operation, and $I(i, j)$ is the input value at the position (i, j)

The average pooling operation is defined as:

$$P(i,j) = \frac{1}{F^2} \sum_{m=0}^{F-1} \sum_{n=0}^{F-1} I(i \times S + m, j \times S + n)$$

which calculates the average of the elements in the covered area of the input.

The Fig 4 shows how max pooling layer works. We use the filter to overlay a part of the feature map, and select the max value from it. Store the value to a new matrix.

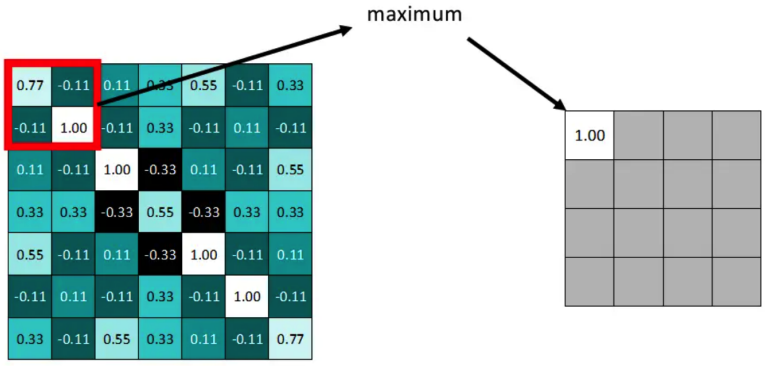

Figure 4: Max Pooling

Then we move the filter to right with S stride, as the Fig 5 shows.

Repeat this process for the entire feature map, then we can get a new feature map after being max pooled, as the Fig 6 shows.

The size of the output feature map can be calculated as:

$$Height = \lfloor \frac{H - F}{S} + 1 \rfloor$$

$$Width = \lfloor \frac{W - F}{S} + 1 \rfloor$$

where $\lfloor \rfloor$ represents the floor operations

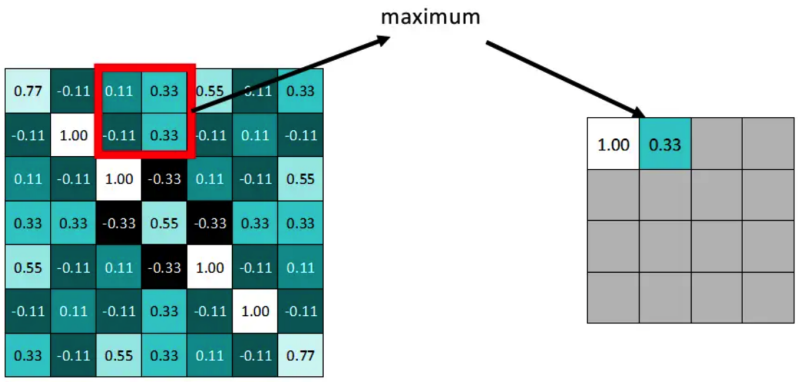

Figure 5: Max Pooling

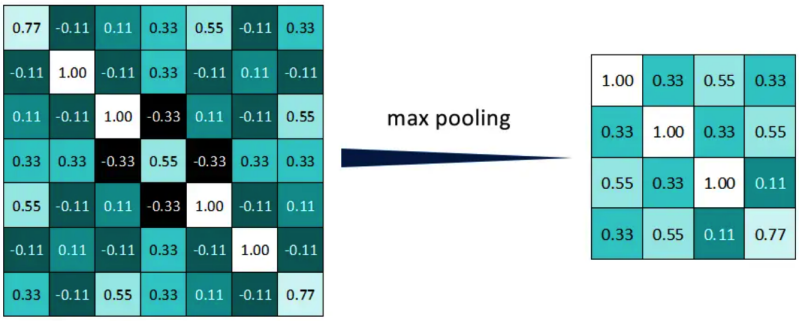

Figure 6: Max Pooling

3 Fully Connected Layer

The Fully Connected (FC) layer is a crucial component of a Convolutional Neural Network (CNN) and many other types of neural networks. It follows the convolutional and pooling layers, and its role is to take the high-level filtered features learned by the previous layers and use them for classifying the input into various classes, depending on the problem.

3.1 Key Concepts

Functions of Fully Connected Layer In a fully connected layer, every input is connected to every output by a weight. This layer combines all the features (learned in previous layers) to classify the image into different classes.

Neurons in Fully Connected Layer Neurons in a fully connected layer have full connections to all activations in the previous layer. These neurons can be visualized as a fully connected graph.

Role in CNN The fully connected layers are often placed at the end of CNN architectures. They are used to flatten the output of convolutional layers to provide the final output size (like the number of classes in classification tasks).

3.2 Mathematical Descriptions

Assuming after the pooling layer and ReLU activation layer, we got a feature map. For the fully connected layer, we transform this map in to a flattened vectors x with size N. For the fully connected layer, we have M neurons.

First of all, each neuron of the layer performs a linear transformation on the input. For the j-neuron, the output o_j is given by:

$$o_j = \sum_{i=1}^{N} w_{ji} x_i + b_j$$

Here, w_{ji} is the weight connecting the i-th input to the j-th neuron, and b_j is the bias term

After the linear transformation, an activation function ϕ is applied to introduce non-linearity:

$$a_j = \phi(o_j)$$

Common activation functions include ReLU, sigmoid, and tanh.

References