D70447E Lab 0

Adesijibomi Aderinto   Oghenero Jeremiah Opia

2025-02-11

Task 1

- Convolutions are mathematical operations where a filter or kernel slides over an input (like an image), performing element-wise multiplication and summing the results to create a transformed output. In deep learning, convolutions help neural networks detect patterns and features by processing data through multiple layers, gradually recognizing more complex structures.

Given the image I and the kernel k:

$$I = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 2 & 1 \\ 1 & -3 & -4 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} \quad \text{and} \quad k = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 2 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

Step 1: Zero Padding the Image

We pad the image with zeros to maintain the same output size. The padded image I_{pad} becomes:

$$I_{\text{pad}} = \begin{bmatrix} 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 2 & 1 \\ 0 & 1 & -3 & -4 & 1 \\ 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \end{bmatrix}$$

Step 2: Convolution Calculation

Now, we apply the kernel $ k $ to the padded image $ I_{pad} $. Let's calculate the convolution step by step.

Following the same process for all positions, the final output matrix is:

$$I * k = \begin{bmatrix} 4 & 4 & 4 \\ 4 & 7 & 4 \\ 4 & 4 & 4 \end{bmatrix}$$

Task 0.3 Max pooling

Let's apply **Max Pooling** with a **filter size of (2, 2)** on the result of the previous convolution operation. We will use **valid pooling**, which means no padding, so the output size will be smaller than the input.

Step-by-Step Process:

$$I * k = \begin{bmatrix} 4 & 4 & 4 \\ 4 & 7 & 4 \\ 4 & 4 & 4 \end{bmatrix}$$

We will apply *valid Max Pooling* with a *2x2 filter* and a *stride of 2*.

Max Pooling Calculation:

- The *filter size* is 2×2, meaning each pool will cover a 2x2 block of values.
- *Valid pooling* means the filter won't go outside the bounds of the input matrix, so the output matrix will be smaller.
- *Stride of 2* means the filter moves 2 steps at a time.

We divide the matrix into 2×2 sections and select the *maximum value* in each section.

First Pooling Block:

$$\begin{bmatrix} 4 & 4 \\ 4 & 7 \end{bmatrix}$$

The maximum value in this block is **7**.

Second Pooling Block:

The second 2×2 block is:

$$\begin{bmatrix} 4 & 4 \\ 4 & 4 \end{bmatrix}$$

The maximum value in this block is **4**.

Result of Max Pooling:

After applying Max Pooling, the output matrix is:

$$\text{Max Pooling Output} = \begin{bmatrix} 7 & 4 \end{bmatrix}$$

Task 0.4 Flattening

To **flatten** the result of the previous **Max Pooling** operation, we reshape the matrix into a one-dimensional vector by placing all the elements of the matrix in a single row.

Max Pooling Output from the Previous Task:

After applying Max Pooling, the output matrix is:

$$\begin{bmatrix} 7 & 4 \end{bmatrix}$$

Flattening the Matrix:

Flattening the matrix means converting it into a one-dimensional vector. In this case, the matrix $\begin{bmatrix} 7 & 4 \end{bmatrix}$ will be flattened as:

$$[7, 4]$$

Task 0.5 Fully Connected Laye

Matrix multiplication for a fully connected layer involves multiplying the input vector $ f{x} $ (1x2) with the weight matrix $ f{W} $ (2x2). The result will be a 1x2 vector (if we don't add a bias term).

The formula for matrix multiplication is:

$$fy = fx \times fW$$

Where:

$$fx = \begin{bmatrix} 7 & 4 \end{bmatrix}, \quad fW = \begin{bmatrix} 0.5 & -0.2 \\ 0.1 & 0.8 \end{bmatrix}$$

Performing the multiplication:

$$fy = \begin{bmatrix} 7 & 4 \end{bmatrix} \times \begin{bmatrix} 0.5 & -0.2 \\ 0.1 & 0.8 \end{bmatrix}$$

This results in:

$$fy = \begin{bmatrix} (7 \times 0.5 + 4 \times 0.1) & (7 \times -0.2 + 4 \times 0.8) \end{bmatrix}$$

$$fy = \begin{bmatrix} (3.5 + 0.4) & (-1.4 + 3.2) \end{bmatrix}$$

$$fy = \begin{bmatrix} 3.9 & 1.8 \end{bmatrix}$$

Task 0.6 SoftMax

3. Now, calculate Softmax for each element:

$$\text{Softmax}(3.9) = \frac{e^{3.9}}{e^{3.9} + e^{1.8}} = \frac{49.402}{55.451} \approx 0.890$$

$$\text{Softmax}(1.8) = \frac{e^{1.8}}{e^{3.9} + e^{1.8}} = \frac{6.049}{55.451} \approx 0.110$$

$$\text{Softmax}(fy) = \begin{bmatrix} 0.890 & 0.110 \end{bmatrix}$$

Since the highest probability is $ 0.890 $, the output class corresponds to the first element of the vector.

Task 0.7 Loss Functions

Results: - **Cross-Entropy Loss**: 0.5108 - **Mean Squared Error Loss**: 0.08167 - **Hinge Loss**: 1.2

Evaluation Metrics