CNN Filter Learning and Practices

Sure! Let's go through the numerical problem step by step **without code**.

Problem Statement

A **3×3 input image** is convolved with a **2×2 filter (kernel)** to produce a **2×2 output feature map**. The goal is to update the filter using **gradient descent** to minimize the difference between the computed output and a **target output (ground truth feature map).**

We will perform one step of backpropagation using Mean Squared Error (MSE) loss.

Given Data

1. Input Image (3×3)

$$I = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 1 & 2 \\ 2 & 1 & 0 \end{bmatrix}$$

2. Initial Random Filter (2×2)

$$W = \begin{bmatrix} 0.5 & 0.2 \\ 0.3 & 0.7 \end{bmatrix}$$

3. Target Output (2×2)

$$Y = \begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix}$$

Step 1: Compute Convolution Output

We slide the **2×2 filter** over the **3×3 image** to compute the **2×2 feature map**:

Top-Left (i=0, j=0)

$$(1 \cdot 0.5) + (2 \cdot 0.2) + (0 \cdot 0.3) + (1 \cdot 0.7) = 0.5 + 0.4 + 0 + 0.7 = 1.6$$

Top-Right (i=0, j=1)

$$(2 \cdot 0.5) + (1 \cdot 0.2) + (1 \cdot 0.3) + (2 \cdot 0.7) = 1 + 0.2 + 0.3 + 1.4 = 2.9$$

Bottom-Left (i=1, j=0)

$$(0 \cdot 0.5) + (1 \cdot 0.2) + (2 \cdot 0.3) + (1 \cdot 0.7) = 0 + 0.2 + 0.6 + 0.7 = 1.5$$

Bottom-Right (i=1, j=1)

$$(1 \cdot 0.5) + (2 \cdot 0.2) + (1 \cdot 0.3) + (0 \cdot 0.7) = 0.5 + 0.4 + 0.3 + 0 = 1.2$$

Computed Feature Map

$$\hat{Y} = \begin{bmatrix} 1.6 & 2.9 \\ 1.5 & 1.2 \end{bmatrix}$$

Step 2: Compute Loss (Mean Squared Error)

$$L = \frac{1}{4} \sum (Y - \hat{Y})^2$$

$$L = \frac{1}{4} [(1 - 1.6)^2 + (2 - 2.9)^2 + (2 - 1.5)^2 + (1 - 1.2)^2]$$

$$L = \frac{1}{4} [(0.6)^2 + (0.9)^2 + (-0.5)^2 + (-0.2)^2]$$

$$L = \frac{1}{4} [0.36 + 0.81 + 0.25 + 0.04] = \frac{1.46}{4} = 0.365$$

Step 3: Compute Gradients (dL/dW)

We update each filter weight using the **gradient of the loss** with respect to the filter. The derivative follows:

$$\frac{\partial L}{\partial W} = \frac{2}{4} \sum (Y - \hat{Y}) \times \text{corresponding input region}$$

Computing for each weight:

Gradient for W_{00} (top-left weight = 0.5)

$$\frac{\partial L}{\partial W_{00}} = \frac{2}{4}[(1.6-1)\cdot 1 + (1.5-2)\cdot 0]$$

$$= \frac{2}{4}[0.6\cdot 1 + (-0.5)\cdot 0] = \frac{2}{4}[0.6] = 0.3$$

Gradient for W_{01} (top-right weight = 0.2)

$$\frac{\partial L}{\partial W_{01}} = \frac{2}{4}[(1.6-1)\cdot 2 + (1.5-2)\cdot 1]$$

$$= \frac{2}{4}[(0.6\cdot 2) + (-0.5\cdot 1)] = \frac{2}{4}[1.2-0.5] = 0.35$$

Gradient for W_{10} (bottom-left weight = 0.3)

$$\frac{\partial L}{\partial W_{10}} = \frac{2}{4}[(1.6-1)\cdot 0 + (1.5-2)\cdot 2]$$

$$= \frac{2}{4}[0 + (-0.5\cdot 2)] = \frac{2}{4}[-1] = -0.5$$

Gradient for W_{11} (bottom-right weight = 0.7)

$$\frac{\partial L}{\partial W_{11}} = \frac{2}{4}[(1.6-1)\cdot 1 + (1.5-2)\cdot 1]$$

$$= \frac{2}{4}[0.6\cdot 1 + (-0.5)\cdot 1] = \frac{2}{4}[0.6-0.5] = 0.05$$

Step 4: Update the Filter Using Gradient Descent

Using the update rule:

$$W' = W - \eta \cdot \frac{\partial L}{\partial W}$$

(With learning rate $\eta = 0.1$)

$$W' = \begin{bmatrix} 0.5 - 0.1(0.3) & 0.2 - 0.1(0.35) \\ 0.3 - 0.1(-0.5) & 0.7 - 0.1(0.05) \end{bmatrix}$$

$$W' = \begin{bmatrix} 0.47 & 0.165 \\ 0.35 & 0.695 \end{bmatrix}$$

Final Results

- **Initial Loss:** 0.365
- **Updated Filter:**

$$\begin{bmatrix} 0.47 & 0.165 \\ 0.35 & 0.695 \end{bmatrix}$$

- **Loss will decrease in the next iteration** as the filter is adjusted.

Conclusion

This process mimics **how CNNs learn filters**—iteratively adjusting weights via backpropagation to minimize the loss. Would you like me to extend this with more iterations? 🚀