In this homework, I have achieved a training accuracy of 98.88%, and testing accuracy of 97.34%. A brief introduction of my implementation is given below.

1 Basic Design

Suppose the input image $X \in \mathbf{R}^{d \times d}$, and output $Y \in \{0,1\}^P$ which has been coded with one-hot, i.e. all elements of Y are 0, except only one element as 1. The convolution kernel size is set to be $(k \times k)$, and there are C channels in total,i.e. $K \in \mathbf{R}^{k \times k \times C}$. Hence the weight and bias of the full-connected layer are given as $W \in \mathbf{R}^{P \times (d-k+1) \times (d-k+1) \times C}$ and $b \in \mathbf{R}^P$. The activation function of hidden layer is ReLu, and that of output layer is Softmax.

2 Forward Propagation

Let $*$ represent convolution, and $\cdot$ as direct dot product. The process of forward propagation is given by

$$\begin{aligned} Z &= X * K && \in \mathbf{R}^{(d-k+1) \times (d-k+1) \times C}, \\ H &= \sigma_{ReLu}(Z) && \in \mathbf{R}^{(d-k+1) \times (d-k+1) \times C}, \\ U_p &= W_{p,:,:,:} \cdot Z + b_p && \in \mathbf{R}, \\ \hat{Y} &= \sigma_{Softmax}(U) && \in [0,1]^P. \end{aligned} \tag{1}$$

3 Backward Propagation

Suppose the loss function between Y and $\hat{Y}$ is given by $\rho(Y, \hat{Y}) = -Y^T log(\hat{Y})$, then the process of backward propagation is given by

$$\begin{aligned}
\frac{\partial \rho}{\partial U} &= Y - \hat{Y}, \\
\frac{\partial \rho}{\partial b} &= \frac{\partial \rho}{\partial U}, \\
\frac{\partial \rho}{\partial W_{p,:,:,:}} &= \frac{\partial \rho}{\partial U_p} H, \\
\frac{\partial \rho}{\partial H_{i,j,c}} &= \frac{\partial \rho}{\partial U} \cdot W_{:,i,j,c}, \\
\frac{\partial \rho}{\partial Z} &= \mathbf{1}_{Z>0} \odot \frac{\partial \rho}{\partial H}, \\
\frac{\partial \rho}{\partial K_c} &= X * \frac{\partial \rho}{\partial Z_{:,:,c}}.
\end{aligned} \tag{2}$$

During the implementation, the "numpy.tensordot" function is adopted to avoid "for" loops in the update of $\frac{\partial \rho}{\partial W}$ and $\frac{\partial \rho}{\partial H}$.

4 Update Process

Similarly with the neural network, the update process is presented below.

$$\begin{aligned}
b &= b - \alpha \frac{\partial \rho}{\partial b}, \\
W &= W - \alpha \frac{\partial \rho}{\partial W}, \\
K &= K - \alpha \frac{\partial \rho}{\partial K}.
\end{aligned} \tag{3}$$

5 Hyperparameter Settings and Time Cost

This CNN has set the kernel size to be (3×3), number of channels as 5, number of epochs as 10, and original learning rate as 0.01, which decays to 10% for every 5 epochs.