Total time: 90 mins Total marks: 18 | Name and Roll Number: ______________________

Questions

1. (4 points) Implementing 2D Convolution Using `nn.Linear`.

   PyTorch's `nn.Conv2d` uses a weight tensor $W \in \mathbb{R}^{C_{\text{out}} \times C_{\text{in}} \times k \times k}$ and bias $b \in \mathbb{R}^{C_{\text{out}}}$. Given an input $x \in \mathbb{R}^{1 \times C_{\text{in}} \times H_{\text{in}} \times W_{\text{in}}}$, implement a *stride-1, no-padding* 2D convolution using only slicing, reshaping, and a single shared `nn.Linear` layer.

   Define:

```
def conv2d_linear(x, W, b):
    ...
```

   that reproduces the output of `nn.Conv2d`.

Solution:

```
import torch
import torch.nn as nn

def conv2d_linear(x, W, b):
    # x:  (1, C_in, H_in, W_in)
    # W:  (C_out, C_in, k, k)
    # b:  (C_out,)

    C_out, C_in, k, _ = W.shape
    _, _, H_in, W_in = x.shape

    H_out = H_in - k + 1
    W_out = W_in - k + 1

    # ---------------------- RUBRIC (4 marks) ----------------------
    # (1 mark) Correct construction of Linear layer and reshaping:
    #          linear.weight = W.reshape(C_out, -1)
    #          linear.bias   = b
    #
    # (1 mark) Correct extraction of each (C_in × k × k) patch:
    #          patch = x[0, :, i:i+k, j:j+k]
    #          Explanation: convolution uses all channels in patch.
    #
    # (1 mark) Correct flattening into a 1D vector:
    #          flat = patch.reshape(-1)
    #          Explanation: Linear expects a 1D input of size C_in*k*k.
    #
    # (1 mark) Correct loop structure and placement of output:
    #          out[0, :, i, j] = linear(flat)
    #          Explanation: identical to applying conv filter at (i,j).
    # ---------------------------------------------------------------

    linear = nn.Linear(C_in * k * k, C_out, bias=True)
    linear.weight.data = W.reshape(C_out, -1).clone()
    linear.bias.data   = b.clone()

    out = torch.zeros(1, C_out, H_out, W_out)

    for i in range(H_out):
        for j in range(W_out):
            patch = x[0, :, i:i+k, j:j+k]      # (C_in, k, k)
            flat  = patch.reshape(-1)          # (C_in*k*k,)
            out[0, :, i, j] = linear(flat)     # shared linear map

    return out
```

2. (4 points) Parameter counting for a modified LeNet-style CNN with two parallel paths.

   The input is a grayscale image of shape $(1 \times 32 \times 32)$. The network is defined as follows:

- **Conv1:** 6 output channels, kernel 5×5, stride 1, *no padding.*
- **Pool1:** 2×2 max-pooling with stride 2.
- The resulting feature map is then sent into two parallel branches:
  - **Path A:**
    - ConvA1: 10 output channels, kernel 3×3, stride 1, padding 1.
    - ConvA2: 12 output channels, kernel 1×1, stride 1.
  - **Path B:**
    - ConvB1: 8 output channels, kernel 1×1, stride 1.
    - ConvB2: 16 output channels, kernel 5×5, stride 1, padding 2.
- The outputs of Path A and Path B are concatenated along the channel dimension (i.e., if Path A produces (C_A, H, W) and Path B produces (C_B, H, W), the concatenated output is $(C_A + C_B,\ H,\ W)$).
- **Pool2:** 2×2 max-pooling with stride 2 is applied to the concatenated output.
- **FC1:** Fully connected layer with 50 outputs. Its input is the flattened output of Pool2.
- **FC2:** Fully connected layer with 10 outputs.

Assume all convolution layers include a bias term.

(a) For each of the following stages, compute the output shape (channels, height, width):

$$\text{Conv1},\ \text{Pool1},\ \text{ConvA1},\ \text{ConvA2},\ \text{ConvB1},\ \text{ConvB2},\ \text{Concat},\ \text{Pool2}.$$

(b) Compute the number of trainable parameters in: Conv1, ConvA1, ConvA2, ConvB1, ConvB2, FC1, FC2.

(c) Compute the total number of trainable parameters in the entire network.

Solution:

Useful formula:

$$\#\text{params(conv)} = \text{outC} \cdot (\text{inC} \cdot k_h \cdot k_w) + \text{outC}, \qquad \#\text{params(FC)} = D_{\text{out}} D_{\text{in}} + D_{\text{out}}.$$

(a) Shapes:

$$(1, 32, 32) \xrightarrow{\text{Conv1 (5×5, s=1, p=0)}} (6, 28, 28) \xrightarrow{\text{Pool1 (2×2, s=2)}} (6, 14, 14),$$
$$(6, 14, 14) \xrightarrow{\text{ConvA1 (3×3, p=1)}} (10, 14, 14) \xrightarrow{\text{ConvA2 (1×1)}} (12, 14, 14),$$
$$(6, 14, 14) \xrightarrow{\text{ConvB1 (1×1)}} (8, 14, 14) \xrightarrow{\text{ConvB2 (5×5, p=2)}} (16, 14, 14),$$
$$\text{Concat: } (12 + 16, 14, 14) = (28, 14, 14) \xrightarrow{\text{Pool2 (2×2, s=2)}} (28, 7, 7).$$

(b) Parameter counts:

$$\begin{aligned}
\text{Conv1: } & 6(1 \cdot 5 \cdot 5) + 6 = 156. \\
\text{ConvA1: } & 10(6 \cdot 3 \cdot 3) + 10 = 550. \\
\text{ConvA2: } & 12(10 \cdot 1 \cdot 1) + 12 = 132. \\
\text{ConvB1: } & 8(6 \cdot 1 \cdot 1) + 8 = 56. \\
\text{ConvB2: } & 16(8 \cdot 5 \cdot 5) + 16 = 3216.
\end{aligned}$$

$$\begin{aligned}
\text{FC1: } & D_{\text{in}} = 28 \cdot 7 \cdot 7 = 1372, \\
& 50 \cdot 1372 + 50 = 68650. \\
\text{FC2: } & 10 \cdot 50 + 10 = 510.
\end{aligned}$$

(c) Total parameters:

$$156 + 550 + 132 + 56 + 3216 + 68650 + 510 = \boxed{73270}.$$

TA Rubric (4 marks):

1 Correct shapes for all listed stages (minor off-by-one: -0.5).

2 Correct parameter counts for all conv and FC layers (small arithmetic mistakes: -0.5).

1 Correct total parameter count with consistent working.

3. (2 points) Feature space dimension for a polynomial kernel.

Consider the polynomial kernel

$$K(\mathbf{x}, \mathbf{z}) = (\mathbf{x}^\top \mathbf{z} + 1)^2,$$

where $\mathbf{x}, \mathbf{z} \in \mathbb{R}^2$ with coordinates $\mathbf{x} = (x_1, x_2)$ and $\mathbf{z} = (z_1, z_2)$.

(a) Expand $K(\mathbf{x}, \mathbf{z})$ explicitly as a polynomial in x_1, x_2, z_1, z_2.

(b) Find an explicit feature map $\phi : \mathbb{R}^2 \to \mathbb{R}^m$ such that

$$K(\mathbf{x}, \mathbf{z}) = \langle \phi(\mathbf{x}), \phi(\mathbf{z}) \rangle,$$

and determine m.

Solution:

(a)

$$\begin{aligned} K(\mathbf{x}, \mathbf{z}) &= (\mathbf{x}^\top \mathbf{z} + 1)^2 = (x_1 z_1 + x_2 z_2 + 1)^2 \\ &= (x_1 z_1)^2 + (x_2 z_2)^2 + 2x_1 z_1 x_2 z_2 + 2x_1 z_1 + 2x_2 z_2 + 1. \end{aligned}$$

(b) We want to write $K(\mathbf{x}, \mathbf{z}) = \langle \phi(\mathbf{x}), \phi(\mathbf{z}) \rangle$.

One valid choice (not unique) is:

$$\phi(\mathbf{x}) = \begin{bmatrix} x_1^2 \\ x_2^2 \\ \sqrt{2}\, x_1 x_2 \\ \sqrt{2}\, x_1 \\ \sqrt{2}\, x_2 \\ 1 \end{bmatrix} \in \mathbb{R}^6.$$

Then

$$\langle \phi(\mathbf{x}), \phi(\mathbf{z}) \rangle = x_1^2 z_1^2 + x_2^2 z_2^2 + 2x_1 x_2 z_1 z_2 + 2x_1 z_1 + 2x_2 z_2 + 1 = K(\mathbf{x}, \mathbf{z}).$$

So the feature space dimension is $m = 6$.

TA Rubric (2 marks):

1 Correct expansion in (a).

1 Valid explicit $\phi(\mathbf{x})$ with inner product matching K, and $m = 6$.

4. (3 points) Hard-margin SVM: primal, dual, and reason for using the dual.

We are given a linearly separable binary classification dataset $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$ with $\mathbf{x}_i \in \mathbb{R}^d$ and $y_i \in \{-1, +1\}$.

(a) Write the **primal** optimization problem for the hard-margin SVM.

(b) Write the corresponding **dual** optimization problem (directly give the final form).

(c) Give one clear reason why the **dual** formulation is preferred in practice for nonlinear SVMs.

Solution:

(a) Primal (hard-margin SVM):

$$\begin{aligned} \min_{\mathbf{w}, b} \quad & \frac{1}{2}\|\mathbf{w}\|^2 \\ \text{s.t.} \quad & y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1, \quad i = 1, \ldots, N. \end{aligned}$$

(b) Dual:

$$\begin{aligned} \max_{\boldsymbol{\alpha}} \quad & \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j y_i y_j (\mathbf{x}_i^\top \mathbf{x}_j) \\ \text{s.t.} \quad & \alpha_i \geq 0, \qquad \sum_{i=1}^N \alpha_i y_i = 0. \end{aligned}$$

(c) The dual depends on the data only through inner products $\mathbf{x}_i^\top \mathbf{x}_j$, so we can replace them by a kernel $K(\mathbf{x}_i, \mathbf{x}_j)$ and implicitly work in a high- or infinite-dimensional feature space. This "kernel trick" is not possible in the primal.

TA Rubric (3 marks):

1 Correct primal formulation.

1 Correct dual formulation with objective and constraints.

1 Correct explanation of kernel trick as the reason for preferring the dual.

5. (2 points) Backpropagation through a 2×2 max-pooling layer.

Consider a single-channel feature map (activation) before pooling:

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

We apply a 2×2 max-pooling operation with stride 2 and no padding. This produces a 2×2 output:

$$P = \text{MaxPool}_{2\times 2,\, s=2}(A).$$

(a) Compute the pooled output P explicitly.

(b) Suppose the gradient of the loss w.r.t. the pooled output is

$$\frac{\partial L}{\partial P} = G = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}.$$

Compute the gradient of the loss w.r.t. the input activations, $\frac{\partial L}{\partial A}$, as a 4×4 matrix. Briefly explain in one sentence how max-pooling routes gradient back to the input.

Solution:

(a) We have four pooling windows (stride 2):

$$W_{11} : A[1{:}2,\, 1{:}2] = \begin{bmatrix} 1 & -2 \\ 4 & 5 \end{bmatrix} \Rightarrow \max = 5,$$

$$W_{12} : A[1{:}2,\, 3{:}4] = \begin{bmatrix} 3 & 0 \\ -1 & 2 \end{bmatrix} \Rightarrow \max = 3,$$

$$W_{21} : A[3{:}4,\, 1{:}2] = \begin{bmatrix} 0 & 7 \\ 8 & -4 \end{bmatrix} \Rightarrow \max = 8,$$

$$W_{22} : A[3{:}4,\, 3{:}4] = \begin{bmatrix} 6 & -3 \\ 1 & 9 \end{bmatrix} \Rightarrow \max = 9.$$

Thus

$$P = \begin{bmatrix} 5 & 3 \\ 8 & 9 \end{bmatrix}.$$

(b) Max-pool gradient: each G_{ij} is sent back only to the position of the max in window W_{ij}; all other positions get zero.

Max positions in A:

- W_{11}: value 5 at $(2, 2)$ in A.

- W_{12}: value 3 at $(1, 3)$ in A.

- W_{21}: value 8 at $(4, 1)$ in A.

- W_{22}: value 9 at $(4, 4)$ in A.

So

$$\frac{\partial L}{\partial A} = \begin{bmatrix} 0 & 0 & 2 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 3 & 0 & 0 & 4 \end{bmatrix},$$

where each nonzero entry is copied from the corresponding element of G.

Max-pooling routes the gradient only to the input element(s) that achieved the maximum in each pooling window; all other elements in that window receive zero gradient.

TA Rubric (2 marks):

1 Correct pooled output $P = \begin{bmatrix} 5 & 3 \\ 8 & 9 \end{bmatrix}$.

1 Correct locations of max elements and nonzero entries in $\frac{\partial L}{\partial A}$. Correct value assignments for the gradient (matching G) and brief explanation that only argmax positions receive gradient.

6. (1 point) Can cross-validation be used to choose the number of clusters K in K-Means? Briefly justify.

Solution: No: K-Means has no labels, so there is no validation error to compute. Also, its objective (within-cluster sum of squares) always decreases as K increases, making cross-validation meaningless for selecting K.

7. (2 points) Q-learning update.

(a) Write the **Bellman optimality Q-learning update equation** and briefly explain what each term represents (reward, discount, bootstrap target, learning rate, and current Q-value).

(b) You are given the Q-table:

$$Q = \begin{array}{c|cc} & L & R \\ \hline A & 1.0 & 0.5 \\ B & 0.2 & 0.4 \end{array}$$

The agent is in state A, takes action R, receives reward $r = 2$, and transitions to state B. Use learning rate $\alpha = 0.5$ and discount $\gamma = 0.9$.

Compute the updated value of $Q(A, R)$.

Solution:

(a)

$$Q(s,a) \leftarrow Q(s,a) + \alpha\Big(r + \gamma \max_{a'} Q(s', a') - Q(s,a)\Big).$$

- r: immediate reward
- γ: discount factor
- $\max_{a'} Q(s', a')$: bootstrap estimate of optimal future value
- $Q(s,a)$: current estimate
- α: learning rate controlling how much new information overrides old

(b)

$$\max_{a'} Q(B, a') = 0.4,$$

$$Q(A,R) \leftarrow 0.5 + 0.5\big(2 + 0.9 \cdot 0.4 - 0.5\big) = 0.5 + 0.93 = 1.43.$$

TA Rubric (2 marks):

1 Correct Bellman update equation and each term identified clearly.

1 Correct numerical update: $Q(A, R) = 1.43$.