Worksheet 4: Convolutional Neural Networks by Hand (Matched to Slides)

Goal. This worksheet mirrors the in-class CNN slides. It contains small, concrete forward-pass calculations you can reproduce on the whiteboard: 1D convolution (with stride/padding), 2D convolution, and max/avg pooling. It closes with a mini end-to-end shape-tracking example (conv $\to$ ReLU $\to$ pool $\to$ dense).

1D Convolution (no padding, stride 1)

Setup. Input signal $x = [2,\ 0,\ 3,\ 1,\ -1]$, filter (kernel) $w = [1,\ 0,\ -1]$, stride $s = 1$, no padding. The valid positions are the length-3 windows:

$$x_{1:3} = [2, 0, 3], \qquad x_{2:4} = [0, 3, 1], \qquad x_{3:5} = [3, 1, -1].$$

Dot products (filter reused at each position):

$$\begin{aligned} y_1 &= \langle [2,0,3], [1,0,-1] \rangle = 2 + 0 - 3 = -1, \\ y_2 &= \langle [0,3,1], [1,0,-1] \rangle = 0 + 0 - 1 = -1, \\ y_3 &= \langle [3,1,-1], [1,0,-1] \rangle = 3 + 0 - (-1) = 4. \end{aligned}$$

Output: $y = [-1,\ -1,\ 4]$ (length $= 5 - 3 + 1 = 3$).

Key ideas to call out. (i) *Locality*—each output only looks at a short window of x. (ii) *Parameter sharing*—the same w is reused at all positions. (iii) *Output length* (1D, valid conv): $L_{\text{out}} = \left\lfloor \frac{L_{\text{in}} - K}{s} \right\rfloor + 1$.

1D Convolution with Padding and Stride

Setup. Same x and w as above. Now use zero-padding of $p = 1$ on both ends and stride $s = 2$.
Padded input: $\tilde{x} = [0,\ 2,\ 0,\ 3,\ 1,\ -1,\ 0]$ (length 7).
Number of outputs: $L_{\text{out}} = \left\lfloor \frac{7-3}{2} \right\rfloor + 1 = 3$.

Windows and dots:

$$\begin{aligned} \tilde{x}_{1:3} = [0,2,0] &\Rightarrow y_1 = 0 \cdot 1 + 2 \cdot 0 + 0 \cdot (-1) = 0, \\ \tilde{x}_{3:5} = [0,3,1] &\Rightarrow y_2 = 0 \cdot 1 + 3 \cdot 0 + 1 \cdot (-1) = -1, \\ \tilde{x}_{5:7} = [1,-1,0] &\Rightarrow y_3 = 1 \cdot 1 + (-1) \cdot 0 + 0 \cdot (-1) = 1. \end{aligned}$$

Output: $y = [0,\ -1,\ 1]$.

Formula to remember. With padding p on each side and stride s, $L_{\text{out}} = \left\lfloor \frac{L_{\text{in}} + 2p - K}{s} \right\rfloor + 1$.

2D Convolution (single-channel image)

Setup. Grayscale input (height=width= 3), valid convolution ($p = 0$), stride $s = 1$:

$$\boldsymbol{X} = \begin{bmatrix} 1 & 2 & 1 \\ 0 & -1 & 0 \\ 2 & 1 & 1 \end{bmatrix}, \qquad \boldsymbol{K} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}.$$

Output has size $(3 - 2 + 1) \times (3 - 2 + 1) = 2 \times 2$.

$$Y_{1,1} = \langle \begin{bmatrix} 1 & 2 \\ 0 & -1 \end{bmatrix}, \boldsymbol{K} \rangle = 1 \cdot 1 + 2 \cdot 0 + 0 \cdot 0 + (-1) \cdot (-1) = 2,$$

$$Y_{1,2} = \langle \begin{bmatrix} 2 & 1 \\ -1 & 0 \end{bmatrix}, \boldsymbol{K} \rangle = 2 \cdot 1 + 1 \cdot 0 + (-1) \cdot 0 + 0 \cdot (-1) = 2,$$

$$Y_{2,1} = \langle \begin{bmatrix} 0 & -1 \\ 2 & 1 \end{bmatrix}, \boldsymbol{K} \rangle = 0 \cdot 1 + (-1) \cdot 0 + 2 \cdot 0 + 1 \cdot (-1) = -1,$$

$$Y_{2,2} = \langle \begin{bmatrix} -1 & 0 \\ 1 & 1 \end{bmatrix}, \boldsymbol{K} \rangle = (-1) \cdot 1 + 0 \cdot 0 + 1 \cdot 0 + 1 \cdot (-1) = -2.$$

Output: $\boldsymbol{Y} = \begin{bmatrix} 2 & 2 \\ -1 & -2 \end{bmatrix}$.

Channels and parameter count. For RGB input (3 channels), a 3×3 filter has $3 \cdot 3 \cdot 3 = 27$ weights (plus an optional bias). Each filter produces one feature map; F filters $\Rightarrow$ F output channels.

Pooling Layers (downsampling features)

Setup. Input feature map (4×4). Apply 2×2 pooling with stride 2.

$$\boldsymbol{A} = \begin{bmatrix} 2 & 1 & 3 & 2 \\ 0 & 4 & 5 & 1 \\ 1 & 2 & 0 & 3 \\ 2 & 1 & 2 & 0 \end{bmatrix}.$$

Max pooling takes the maximum in each non-overlapping 2×2 block: $\begin{bmatrix} \max\{2,1,0,4\} & \max\{3,2,5,1\} \\ \max\{1,2,2,1\} & \max\{0,3,2,0\} \end{bmatrix} = \begin{bmatrix} 4 & 5 \\ 2 & 3 \end{bmatrix}$.

Average pooling takes the mean of each block: $\begin{bmatrix} (2+1+0+4)/4 & (3+2+5+1)/4 \\ (1+2+2+1)/4 & (0+3+2+0)/4 \end{bmatrix} = \begin{bmatrix} 1.75 & 2.75 \\ 1.50 & 1.25 \end{bmatrix}$.

When to emphasise. Pooling reduces spatial size (here $4 \times 4 \to 2 \times 2$) and introduces small translation invariance by summarising patches.

Mini CNN: Shape & Parameter Tracking

CNNs as matrix multiplications

$$\textbf{Input } X = \begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \end{bmatrix}, \qquad \textbf{Filter } K = \begin{bmatrix} k_{11} & k_{12} \\ k_{21} & k_{22} \end{bmatrix}, \qquad (\text{stride } 1,\ \text{valid})$$

$$\underbrace{\begin{bmatrix} y_{11} \\ y_{12} \\ y_{21} \\ y_{22} \end{bmatrix}}_{\text{vec}(Y)} = \underbrace{\begin{bmatrix} k_{11} & k_{12} & 0 & k_{21} & k_{22} & 0 & 0 & 0 & 0 \\ 0 & k_{11} & k_{12} & 0 & k_{21} & k_{22} & 0 & 0 & 0 \\ 0 & 0 & 0 & k_{11} & k_{12} & 0 & k_{21} & k_{22} & 0 \\ 0 & 0 & 0 & 0 & k_{11} & k_{12} & 0 & k_{21} & k_{22} \end{bmatrix}}_{\text{Toeplitz } (4\times 9)} \underbrace{\begin{bmatrix} x_{11} \\ x_{12} \\ x_{13} \\ x_{21} \\ x_{22} \\ x_{23} \\ x_{31} \\ x_{32} \\ x_{33} \end{bmatrix}}_{\text{vec}(X)}$$

$$\underbrace{\begin{bmatrix} y_{11} \\ y_{12} \\ y_{21} \\ y_{22} \end{bmatrix}}_{\text{vec}(Y)} = \underbrace{\begin{bmatrix} x_{11} & x_{12} & x_{21} & x_{22} \\ x_{12} & x_{13} & x_{22} & x_{23} \\ x_{21} & x_{22} & x_{31} & x_{32} \\ x_{22} & x_{23} & x_{32} & x_{33} \end{bmatrix}}_{\text{im2col}(4\times 4)} \underbrace{\begin{bmatrix} k_{11} \\ k_{12} \\ k_{21} \\ k_{22} \end{bmatrix}}_{\text{vec}(K)}$$

$$\text{where} \quad \begin{aligned} y_{11} &= k_{11}x_{11} + k_{12}x_{12} + k_{21}x_{21} + k_{22}x_{22}, \\ y_{12} &= k_{11}x_{12} + k_{12}x_{13} + k_{21}x_{22} + k_{22}x_{23}, \\ y_{21} &= k_{11}x_{21} + k_{12}x_{22} + k_{21}x_{31} + k_{22}x_{32}, \\ y_{22} &= k_{11}x_{22} + k_{12}x_{23} + k_{21}x_{32} + k_{22}x_{33}. \end{aligned}$$

Layer and batch norms

We use a tiny feed-forward network with two inputs, a hidden layer with two units, and a single sigmoid output (a 2–2–1 network).

$$\begin{aligned} \boldsymbol{z}^{(1)} &= W^{(1)}\boldsymbol{x} + \boldsymbol{b}^{(1)}, \\ \boldsymbol{h} &= \phi\Big(\text{Norm}\big(\boldsymbol{z}^{(1)}\big)\Big), \quad \phi = \text{ReLU (unless noted)}, \\ z^{(2)} &= W^{(2)}\boldsymbol{h} + b^{(2)}, \qquad \hat{y} = \sigma(z^{(2)}) = \frac{1}{1+e^{-z^{(2)}}}. \end{aligned}$$

When no normalisation is used, Norm is the identity. Vectors are column vectors.

Common parameters (unless stated otherwise):

$$W^{(1)} = \begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix}, \quad \boldsymbol{b}^{(1)} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \qquad W^{(2)} = \begin{bmatrix} 1 & 2 \end{bmatrix}, \quad b^{(2)} = -1.$$

1 Baseline: No Normalisation

Input: $\boldsymbol{x} = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$.

$$\begin{aligned}
\boldsymbol{z}^{(1)} &= \begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix} \begin{bmatrix} 1 \\ -1 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 3 \\ -2 \end{bmatrix}, \\
\boldsymbol{h} &= \mathrm{ReLU}\left(\begin{bmatrix} 3 \\ -2 \end{bmatrix} \right) = \begin{bmatrix} 3 \\ 0 \end{bmatrix}, \\
z^{(2)} &= \begin{bmatrix} 1 & 2 \end{bmatrix} \begin{bmatrix} 3 \\ 0 \end{bmatrix} - 1 = 2, \qquad \hat{y} = \sigma(2) \approx 0.8808.
\end{aligned}$$

Student task. Repeat for $\boldsymbol{x} = \begin{bmatrix} 2 \\ 3 \end{bmatrix}$ (answer check: $\hat{y} \approx 0.9999992$).

2 Batch Normalisation (BN)

BN normalises *per feature across the mini-batch.* Place BN between the linear and ReLU layers. For batch $\{\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}\}$ with $\boldsymbol{x}^{(1)} = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$ and $\boldsymbol{x}^{(2)} = \begin{bmatrix} 2 \\ 3 \end{bmatrix}$, compute

$$Z^{(1)} = \begin{bmatrix} 3 & -2 \\ 1 & 7 \end{bmatrix} \quad \text{(rows = samples, cols = hidden units).}$$

Per hidden unit j, with batch size $m = 2$,

$$\begin{aligned}
\mu_j &= \tfrac{1}{m} \sum_{i=1}^{m} z_{ij}^{(1)}, \quad \sigma_j^2 = \tfrac{1}{m} \sum_{i=1}^{m} \big(z_{ij}^{(1)} - \mu_j\big)^2, \\
\hat{z}_{ij}^{(1)} &= \frac{z_{ij}^{(1)} - \mu_j}{\sqrt{\sigma_j^2 + \varepsilon}}, \quad y_{ij}^{(1)} = \gamma_j \hat{z}_{ij}^{(1)} + \beta_j.
\end{aligned}$$

Use $\gamma = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$, $\beta = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$, $\varepsilon = 10^{-5}$. Means and variances: $\mu_1 = 2$, $\sigma_1^2 = 1$; $\mu_2 = 2.5$, $\sigma_2^2 = 20.25$. Thus, for the two samples

$$\boldsymbol{y}_{(1)}^{(1)} = [1, -1], \qquad \boldsymbol{y}_{(2)}^{(1)} = [-1, 1].$$

After ReLU:

$$\boldsymbol{h}_{(1)} = [1, 0]^T, \qquad \boldsymbol{h}_{(2)} = [0, 1]^T.$$

Outputs:

$$\begin{aligned}
z_{(1)}^{(2)} &= [1\ 2][1, 0]^T - 1 = 0 \Rightarrow \hat{y}_{(1)} = 0.5, \\
z_{(2)}^{(2)} &= [1\ 2][0, 1]^T - 1 = 1 \Rightarrow \hat{y}_{(2)} \approx 0.7311.
\end{aligned}$$

Notes. BN uses batch statistics and can change which units are active. Try other γ, β to see the effect.

3 Layer Normalisation (LN)

LN normalises *across features within each sample* (independent of batch size). For a given sample with $\boldsymbol{z}^{(1)} = [z_1, z_2]^T$,

$$\mu = \tfrac{1}{2}(z_1 + z_2), \qquad \sigma^2 = \tfrac{1}{2}\big[(z_1 - \mu)^2 + (z_2 - \mu)^2\big],$$
$$\hat{z}_j = \frac{z_j - \mu}{\sqrt{\sigma^2 + \varepsilon}}, \qquad y_j = \gamma_j \hat{z}_j + \beta_j.$$

With $\gamma = [1, 1]^T$, $\beta = [0, 0]^T$, $\varepsilon = 10^{-5}$:
Sample 1: $\boldsymbol{z}^{(1)} = [3, -2]^T \Rightarrow \mu = 0.5,\ \sigma^2 = 6.25,\ \hat{\boldsymbol{z}} = [1, -1]^T,\ \boldsymbol{y}^{(1)} = [1, -1]^T$.
Sample 2: $\boldsymbol{z}^{(1)} = [1, 7]^T \Rightarrow \mu = 4,\ \sigma^2 = 9,\ \hat{\boldsymbol{z}} = [-1, 1]^T,\ \boldsymbol{y}^{(1)} = [-1, 1]^T$.
After ReLU and output, the predictions match the BN example above: $\hat{y}_{(1)} = 0.5$, $\hat{y}_{(2)} \approx 0.7311$.

Notes. LN does not depend on batch statistics, so it works with batch size 1.

4 Group Normalisation (GN)

GN splits the features of a *single sample* into G groups and normalises within each group. It interpolates between LN (G=1) and InstanceNorm (G=C). To make groups non-trivial, we use a 2–**4**–1 network.

Setup for GN example

Hidden layer (4 units):

$$W^{(1)} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ -1 & 2 \\ 2 & -1 \end{bmatrix}, \quad \boldsymbol{b}^{(1)} = \mathbf{0}_4, \qquad W^{(2)} = \begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix}, \ b^{(2)} = 0,$$
$$\boldsymbol{x} = \begin{bmatrix} 2 \\ 1 \end{bmatrix}, \qquad G = 2 \text{ groups of size } 2,\ \gamma = \mathbf{1}_4,\ \beta = \mathbf{0}_4,\ \varepsilon = 10^{-5}.$$

Forward pass

Hidden pre-activations:

$$\boldsymbol{z}^{(1)} = W^{(1)}\boldsymbol{x} = \begin{bmatrix} 2 \\ 1 \\ 0 \\ 3 \end{bmatrix}.$$

Group 1: $(z_1, z_2) = (2, 1)$, $\mu_1 = 1.5$, $\sigma_1^2 = 0.25 \Rightarrow \hat{z}_1 = 1,\ \hat{z}_2 = -1$.
Group 2: $(z_3, z_4) = (0, 3)$, $\mu_2 = 1.5$, $\sigma_2^2 = 2.25 \Rightarrow \hat{z}_3 = -1,\ \hat{z}_4 = 1$.
Thus $\hat{\boldsymbol{z}}^{(1)} = [1, -1, -1, 1]^T$, $\boldsymbol{h} = \mathrm{ReLU}(\hat{\boldsymbol{z}}^{(1)}) = [1, 0, 0, 1]^T$. Finally,

$$z^{(2)} = [1\ 1\ 1\ 1]\,[1, 0, 0, 1]^T = 2, \qquad \hat{y} = \sigma(2) \approx 0.8808.$$

Notes. GN is independent of batch size (like LN) but controls the degree of feature coupling via the number of groups G.

Summary Table

Method	Normalises over	Batch-size dependent?	Placement
BatchNorm	Features across samples (per feature)	Yes	Linear $\to$ BN $\to$ ReLU
LayerNorm	Features within each sample	No	Linear $\to$ LN $\to$ ReLU
GroupNorm	Feature groups within each sample	No	Linear $\to$ GN $\to$ ReLU

Extra Exercises

1. BN: change $\gamma = [0.5, 2]^T$, $\beta = [0.1, -0.2]^T$ and recompute outputs.
2. LN: compute with batch size 1 (only $\boldsymbol{x}^{(1)}$). What changes vs BN?
3. GN: set $G = 1$ and show it reduces to LN; set $G = 4$ (instance norm) and compare.