0. Notation

Vectors: **bold lower-case** ($\boldsymbol{x}$), Matrices/Tensors: **bold upper-case** ($\boldsymbol{X}$)
$\odot$ = Hadamard product, B = mini-batch size
Indicator $\mathbf{1}_{(\cdot)} = 1$ if condition is true

1. Fully Connected Network (2–2–1)

Forward Mapping
Input $\boldsymbol{x} \in \mathbb{R}^2$:

$$\begin{aligned} \boldsymbol{a}^{(1)} &= \boldsymbol{W}^{(1)}\boldsymbol{x} + \boldsymbol{b}^{(1)} \in \mathbb{R}^2 \\ \boldsymbol{h}^{(1)} &= \mathrm{ReLU}(\boldsymbol{a}^{(1)}) \\ z &= \boldsymbol{w}^\top \boldsymbol{h}^{(1)} + b, \quad \hat{y} = \sigma(z) \end{aligned}$$

Sigmoid: $\sigma(z) = \frac{1}{1+e^{-z}}$
Binary Cross-Entropy (BCE):

$$\boxed{\mathcal{L} = -\frac{1}{B}\sum_{i \in \mathcal{B}} \big[y_i \ln \hat{y}_i + (1-y_i)\ln(1-\hat{y}_i)\big]}$$

Backpropagation Chain:

$$\begin{aligned} \frac{\partial L}{\partial z} &= \hat{y} - y \\ \frac{\partial L}{\partial b} &= \frac{\partial L}{\partial z} \\ \frac{\partial L}{\partial \boldsymbol{w}} &= \frac{\partial L}{\partial z}\,(\boldsymbol{h}^{(1)})^\top \\ \frac{\partial L}{\partial \boldsymbol{h}^{(1)}} &= \frac{\partial L}{\partial z}\,\boldsymbol{w} \\ \frac{\partial L}{\partial \boldsymbol{a}^{(1)}} &= \frac{\partial L}{\partial \boldsymbol{h}^{(1)}} \odot g'(\boldsymbol{a}^{(1)}) \\ \frac{\partial L}{\partial \boldsymbol{b}^{(1)}} &= \frac{\partial L}{\partial \boldsymbol{a}^{(1)}} \\ \frac{\partial L}{\partial \boldsymbol{W}^{(1)}} &= (\frac{\partial L}{\partial \boldsymbol{a}^{(1)}})\,\boldsymbol{x}^\top \end{aligned}$$

Mini-batch: Average gradients over B samples

2. Optimizers

$$\begin{aligned} \textbf{GD:}\quad & \boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \eta\,\nabla L(\boldsymbol{\theta}_k) \\ \textbf{SGD:}\quad & \boldsymbol{\theta}_k = \boldsymbol{\theta}_{k-1} - \eta\,\frac{1}{B}\sum_{i\in\mathcal{B}} \nabla L_i \\ \textbf{Momentum:}\quad & \boldsymbol{v}_k = \beta \boldsymbol{v}_{k-1} + \eta\,\nabla L_k \\ & \boldsymbol{\theta}_k = \boldsymbol{\theta}_{k-1} - \boldsymbol{v}_k \\ \textbf{AdaGrad:}\quad & G_k = G_{k-1} + \nabla L_k \odot \nabla L_k \\ & \boldsymbol{\theta}_k = \boldsymbol{\theta}_{k-1} - \frac{\eta\,\nabla L_k}{\sqrt{G_k} + \varepsilon} \\ \textbf{RMSProp:}\quad & G_k = \beta G_{k-1} + (1-\beta)\nabla L_k^2 \\ & \text{then same update as AdaGrad} \\ \textbf{Adam:}\quad & \boldsymbol{m}_k = \beta_1 \boldsymbol{m}_{k-1} + (1-\beta_1)\nabla L_k \\ & \boldsymbol{v}_k = \beta_2 \boldsymbol{v}_{k-1} + (1-\beta_2)\nabla L_k^2 \\ & \hat{\boldsymbol{m}}_k = \boldsymbol{m}_k/(1-\beta_1^k) \\ & \hat{\boldsymbol{v}}_k = \boldsymbol{v}_k/(1-\beta_2^k) \\ & \boldsymbol{\theta}_k = \boldsymbol{\theta}_{k-1} - \frac{\eta\,\hat{\boldsymbol{m}}_k}{\sqrt{\hat{\boldsymbol{v}}_k} + \varepsilon} \end{aligned}$$

3. CNNs: Dimensional Analysis

Single 2-D Convolution Layer
Input: $\boldsymbol{X} \in \mathbb{R}^{B \times C_{\text{in}} \times H_{\text{in}} \times W_{\text{in}}}$
Kernel: $\boldsymbol{K} \in \mathbb{R}^{C_{\text{out}} \times C_{\text{in}} \times K_h \times K_w}$
with stride (S_h, S_w), padding (P_h, P_w)
Output Spatial Dimensions:

$$\begin{aligned} H_{\text{out}} &= \left\lfloor \frac{H_{\text{in}} + 2P_h - K_h}{S_h} \right\rfloor + 1 \\ W_{\text{out}} &= \left\lfloor \frac{W_{\text{in}} + 2P_w - K_w}{S_w} \right\rfloor + 1 \end{aligned}$$

Output: $\boldsymbol{Z} \in \mathbb{R}^{B \times C_{\text{out}} \times H_{\text{out}} \times W_{\text{out}}}$

Parameters: $C_{\text{out}} C_{\text{in}} K_h K_w + C_{\text{out}}$
MACs: $B\, C_{\text{out}}\, H_{\text{out}}\, W_{\text{out}}\, C_{\text{in}}\, K_h K_w$
Receptive Field & Stride (layer ℓ):

$$\begin{aligned} \text{stride}_\ell &= \text{stride}_{\ell-1}\, S_\ell \\ \text{field}_\ell &= \text{field}_{\ell-1} + (K_\ell - 1)\,\text{stride}_{\ell-1} \end{aligned}$$

Init: $\text{field}_0 = 1,\ \text{stride}_0 = 1$
Typical CNN Block:

$$\begin{aligned} \boldsymbol{X} &\to (B, C_0, H_0, W_0) \\ \text{Conv} &\to (B, C_1, H_1, W_1) \\ \text{BN+ReLU} &\to (B, C_1, H_1, W_1) \\ \text{MaxPool}(2,2) &\to (B, C_1, H_1/2, W_1/2) \\ \text{Flatten} &\to (B, D) \text{ where } D = C_L H_L W_L \\ \text{FC} &\to (B, n_{\text{classes}}) \end{aligned}$$

4. Single-Layer RNN

State $\boldsymbol{h}_{t-1} \in \mathbb{R}^H$, input $\boldsymbol{x}_t \in \mathbb{R}^D$:

$$\begin{aligned} \boldsymbol{a}_t &= \boldsymbol{W}\boldsymbol{h}_{t-1} + \boldsymbol{U}\boldsymbol{x}_t + \boldsymbol{b} \\ \boldsymbol{h}_t &= f(\boldsymbol{a}_t) \\ \boldsymbol{z}_t &= \boldsymbol{V}\boldsymbol{h}_t + \boldsymbol{c} \\ \hat{\boldsymbol{y}}_t &= g(\boldsymbol{z}_t) \end{aligned}$$

BPTT gradient:

$$\delta_t = \big(\nabla_{\boldsymbol{h}_t} L + \boldsymbol{W}^\top \delta_{t+1}\big) \odot f'(\boldsymbol{a}_t)$$

5. Transformer Self-Attention

Per-Token Projections
For each token $\boldsymbol{x}_t \in \mathbb{R}^{d_{\text{model}}}$:

$$\begin{aligned} \boldsymbol{q}_t &= \boldsymbol{W}_Q \boldsymbol{x}_t \\ \boldsymbol{k}_t &= \boldsymbol{W}_K \boldsymbol{x}_t \quad \boldsymbol{W}_Q, \boldsymbol{W}_K, \boldsymbol{W}_V \in \mathbb{R}^{d_{\text{model}} \times d_k} \\ \boldsymbol{v}_t &= \boldsymbol{W}_V \boldsymbol{x}_t \end{aligned}$$

Attention Weights:
Query token t attends to key token s:

$$\alpha_{t \to s} = \frac{\exp(\boldsymbol{q}_t^\top \boldsymbol{k}_s/\sqrt{d_k})}{\sum_{u=1}^T \exp(\boldsymbol{q}_t^\top \boldsymbol{k}_u/\sqrt{d_k})}$$

Context Vector:

$$\boldsymbol{z}_t = \sum_{s=1}^T \alpha_{t \to s}\, \boldsymbol{v}_s$$

Matrix Form:
$\boldsymbol{A} = \mathrm{softmax}(\boldsymbol{Q}\boldsymbol{K}^\top/\sqrt{d_k})$, $\boldsymbol{Z} = \boldsymbol{A}\boldsymbol{V}$
where $\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V} \in \mathbb{R}^{B \times T \times d_k}$
Multi-Head: Split into h heads with
$d_k = d_{\text{model}}/h$, concatenate, apply dense layer

6. PCA (2-Feature Case)

Mean-Centered Data:

$$\mathbf{X} = \begin{bmatrix} x_{11} & \cdots & x_{1n} \\ x_{21} & \cdots & x_{2n} \end{bmatrix} \in \mathbb{R}^{2 \times n}$$

Covariance:

$$\mathbf{A} = \mathbf{X}\mathbf{X}^\top = \begin{bmatrix} a & b \\ b & c \end{bmatrix}$$

Eigenvalues:

$$\lambda_{1,2} = \frac{(a+c) \pm \sqrt{(a-c)^2 + 4b^2}}{2}$$

where $\lambda_1 \geq \lambda_2$
Eigenvector for λ_1 (if $b \neq 0$):

$$\mathbf{u}_1 = \begin{bmatrix} \lambda_1 - c \\ b \end{bmatrix}, \quad \mathbf{v}_1 = \frac{\mathbf{u}_1}{\|\mathbf{u}_1\|}$$

PC Scores:

$$\mathbf{h}_1 = \mathbf{v}_1^\top \mathbf{X} \in \mathbb{R}^{1 \times n}$$

Explained Variance:

$$\mathrm{EVR}_1 = \frac{\lambda_1}{\lambda_1 + \lambda_2}$$

Rank-1 Reconstruction:

$$\boxed{\mathbf{B} + \mathbf{C}\,\mathbf{C}^\top(\mathbf{A} - \mathbf{B})}$$

where $\mathbf{A}$ = data ($d \times n$), $\mathbf{B}$ = row means,
$\mathbf{C}$ = top eigenvector ($d \times 1$)

7. Overfitting: Diagnostics & Remedies

Symptoms:

- Training loss ↓, validation loss ↑
- Widening train–val accuracy gap

Remedies:

- Early stopping with patience
- L_2 weight decay: $\mathcal{L} + \lambda\|\boldsymbol{\theta}\|^2$
- Dropout: randomly zero activations
- Data augmentation
- Reduce model capacity
- k-fold cross-validation

8. Weight Initialization

Method	**Use**	$\boldsymbol{\sigma^2}$
Glorot/Xavier	sigmoid, tanh	$\frac{2}{n_{\text{in}}+n_{\text{out}}}$
He (Kaiming)	ReLU	$\frac{2}{n_{\text{in}}}$

9. Loss Functions

Binary Cross-Entropy:

$$\mathcal{L}_{\text{BCE}} = -\frac{1}{B}\sum_{i\in\mathcal{B}}\big[y_i \ln \hat{y}_i + (1-y_i)\ln(1-\hat{y}_i)\big]$$

Mean Squared Error:

$$\mathcal{L}_{\text{MSE}} = \frac{1}{B}\sum_{i\in\mathcal{B}}\|\hat{\boldsymbol{y}}_i - \boldsymbol{y}_i\|_2^2$$

Categorical Cross-Entropy:

$$\mathcal{L}_{\text{CCE}} = -\frac{1}{B}\sum_{i\in\mathcal{B}}\sum_{c=1}^{C} y_{i,c}\ln \hat{y}_{i,c}$$

10. Chain Rule Reference

For $L(\boldsymbol{u})$, $\boldsymbol{u} = g(\boldsymbol{v})$, $\boldsymbol{v} = h(\boldsymbol{x})$:

$$\nabla_{\boldsymbol{x}} L = (\nabla_{\boldsymbol{u}} L)\frac{\partial \boldsymbol{u}}{\partial \boldsymbol{v}}\frac{\partial \boldsymbol{v}}{\partial \boldsymbol{x}}$$

11. CNN Architecture Example

Conv-Conv-FC Network
Input: $x \in \mathbb{R}^{H_{\text{in}}\times W_{\text{in}}\times C_{\text{in}}}$
Conv 1:
Kernel $w^{(1)} \in \mathbb{R}^{K_1\times K_1\times C_{\text{in}}\times C_1}$
Bias $b^{(1)} \in \mathbb{R}^{C_1}$, padding $p_1 = (K_1-1)/2$

$$a_k^{(1)} = \sum_{c=0}^{C_{\text{in}}-1}\text{Conv}(x(\cdot,\cdot,c), w^{(1)}(\cdot,\cdot,c,k)) + b_k^{(1)}$$

$$h^{(1)} = g(a^{(1)}) \in \mathbb{R}^{H_1\times W_1\times C_1}$$

Conv 2:
Kernel $w^{(2)} \in \mathbb{R}^{K_2\times K_2\times C_1\times C_2}$

$$a_k^{(2)} = \sum_{c=0}^{C_1-1}\text{Conv}(h^{(1)}(\cdot,\cdot,c), w^{(2)}(\cdot,\cdot,c,k)) + b_k^{(2)}$$

$$h^{(2)} = g(a^{(2)}) \in \mathbb{R}^{H_2\times W_2\times C_2}$$

Flatten:

$$h_{\text{flat}}^{(2)} \in \mathbb{R}^{D_{\text{flat}}}, \quad D_{\text{flat}} = H_2 W_2 C_2$$

Dense:
$W^{(3)} \in \mathbb{R}^{D_h\times D_{\text{flat}}}$, $b^{(3)} \in \mathbb{R}^{D_h}$

$$a^{(3)} = W^{(3)}h_{\text{flat}}^{(2)} + b^{(3)}, \quad h^{(3)} = g(a^{(3)})$$

Output:
$W \in \mathbb{R}^{D_{\text{out}}\times D_h}$, $b \in \mathbb{R}^{D_{\text{out}}}$

$$z = Wh^{(3)} + b \in \mathbb{R}^{D_{\text{out}}}$$

Apply $\text{softmax}(z)$ or $\sigma(z)$ for prediction

12. Model Selection Guide

RNN: Sequential data, moderate lengths
Ex: time series prediction
CNN: Grid data with spatial locality
Ex: image classification (MNIST)
Transformer: Long-range dependencies
Ex: machine translation, NLP tasks

13. Regression vs. Classification

Linear Regression: Predicts continuous values $\hat{y} \in \mathbb{R}$. Uses MSE loss. Output layer has linear activation.
Binary Classification: Predicts class probabilities $\hat{y} \in [0,1]$. Uses BCE loss. Output layer has sigmoid activation.
Multi-class Classification: Predicts probability distribution over C classes. Uses categorical cross-entropy. Output has softmax.

14. Sequence Model Tasks

Seq→Vec (many→one):
Sentiment analysis, spam detection
Seq→Seq (many→many):
Machine translation, speech recognition
Seq→Vec Classification Loss:

$$\mathcal{L} = \frac{1}{N}\sum_{i=1}^{N}\left[-\sum_{c=1}^{C} y_{i,c}\ln p_{i,c}\right]$$

Seq→Seq Regression Loss:

$$\mathcal{L} = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{T_i}\sum_{t=1}^{T_i} m_{i,t}\|y_{i,t} - \hat{y}_{i,t}\|_2^2$$

Note: RNN update $\boldsymbol{h}_t = f(\boldsymbol{h}_{t-1}, \boldsymbol{x}_t)$ is recursive ⇒ handles arbitrary length sequences