0. Notation

Vectors are bold lower-case, and matrices/tensors are bold upper-case. $\odot$ = Hadamard product, B = mini-batch size. The indicator $\mathbf{1}_{(\cdot)}$ equals 1 if the condition is true.

1. Fully Connected Network (2–2–1 example)

Forward mapping Input $\boldsymbol{x} \in \mathbb{R}^2$ gives

$$\boldsymbol{a}^{(1)} = \boldsymbol{W}^{(1)}\boldsymbol{x} + \boldsymbol{b}^{(1)} \in \mathbb{R}^2,$$
$$\boldsymbol{h}^{(1)} = \mathrm{ReLU}(\boldsymbol{a}^{(1)}),$$
$$z = \boldsymbol{w}\boldsymbol{h}^{(1)} + b, \qquad \hat{y} = \sigma(z).$$

Sigmoid function $\sigma(z) = \frac{1}{1+e^{-z}}$
Binary cross-entropy loss (mini-batch)

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

Symbolic derivative chain

$$\frac{\partial L}{\partial z} = \hat{y} - y, \qquad \frac{\partial L}{\partial b} = \frac{\partial L}{\partial z}$$
$$\frac{\partial L}{\partial \boldsymbol{W}} = \left(\tfrac{\partial L}{\partial z}\right)^\top (\boldsymbol{h}^{(1)})^\top$$
$$\frac{\partial L}{\partial \boldsymbol{h}^{(1)}} = \frac{\partial L}{\partial z}\boldsymbol{W}, \qquad \frac{\partial L}{\partial \boldsymbol{a}^{(1)}} = \frac{\partial L}{\partial \boldsymbol{h}^{(1)}} \odot g'(\boldsymbol{a}^{(1)})^\top$$
$$\frac{\partial L}{\partial \boldsymbol{b}^{(1)}} = \frac{\partial L}{\partial \boldsymbol{a}^{(1)}}, \qquad \frac{\partial L}{\partial \boldsymbol{W}^{(1)}} = \left(\tfrac{\partial L}{\partial \boldsymbol{a}^{(1)}}\right)^\top \boldsymbol{x}^\top$$

Parameter averaging (mini-batch)
For a mini-batch, average the gradients for $\boldsymbol{b}$, $\boldsymbol{W}$, $\boldsymbol{b}^{(1)}$, and $\boldsymbol{W}^{(1)}$ over all B elements in the batch.

2. Optimizers

Gradient Descent: $\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \eta\,\nabla L(\boldsymbol{\theta}_k)$

SGD: $\boldsymbol{\theta}_k = \boldsymbol{\theta}_{k-1} - \eta\,\frac{1}{B}\sum_{i\in\mathcal{B}}\nabla L_i$

Momentum: $\boldsymbol{v}_k = \beta\boldsymbol{v}_{k-1} + \eta\,\nabla L_k,\ \boldsymbol{\theta}_k = \boldsymbol{\theta}_{k-1} - \boldsymbol{v}_k$

AdaGrad: $G_k = G_{k-1} + \nabla L_k \odot \nabla L_k,\ \boldsymbol{\theta}_k = \boldsymbol{\theta}_{k-1} - \eta\,\nabla L_k/(\sqrt{G_k}+\varepsilon)$

RMSProp: $G_k = \beta G_{k-1} + (1-\beta)\nabla L_k^2$, then same update as AdaGrad

Adam:
$$\boldsymbol{m}_k = \beta_1\boldsymbol{m}_{k-1} + (1-\beta_1)\nabla L_k$$
$$\boldsymbol{v}_k = \beta_2\boldsymbol{v}_{k-1} + (1-\beta_2)\nabla L_k^2$$
$$\hat{\boldsymbol{m}}_k = \boldsymbol{m}_k/(1-\beta_1^k),\ \hat{\boldsymbol{v}}_k = \boldsymbol{v}_k/(1-\beta_2^k)$$
$$\boldsymbol{\theta}_k = \boldsymbol{\theta}_{k-1} - \eta\,\hat{\boldsymbol{m}}_k/(\sqrt{\hat{\boldsymbol{v}}_k}+\varepsilon)$$

3. Convolutional Neural Networks: Detailed dimensional analysis

Single 2-D convolution layer
Input tensor $\boldsymbol{X} \in \mathbb{R}^{B\times C_{\text{in}}\times H_{\text{in}}\times W_{\text{in}}}$
Kernel weights $\boldsymbol{K} \in \mathbb{R}^{C_{\text{out}}\times C_{\text{in}}\times K_h\times K_w}$ with stride S_h, S_w and zero-padding P_h, P_w.

Spatial dimensions after convolution

$$H_{\text{out}} = \lfloor (H_{\text{in}} + 2P_h - K_h)/S_h \rfloor + 1,$$
$$W_{\text{out}} = \lfloor (W_{\text{in}} + 2P_w - K_w)/S_w \rfloor + 1.$$

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

Parameter count $C_{\text{out}}C_{\text{in}}K_hK_w + C_{\text{out}}$ (biases).

Multiply–accumulate operations $B\,C_{\text{out}}\,H_{\text{out}}\,W_{\text{out}}\,C_{\text{in}}\,K_hK_w$.

Receptive field and effective stride (layer ℓ)

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

Initialization $\text{field}_0 = 1,\ \text{stride}_0 = 1$.

Typical block sequence and shapes

$\boldsymbol{X}$	shape (B, C_0, H_0, W_0)
Conv	$\to (B, C_1, H_1, W_1)$
BatchNorm+ReLU	$\to (B, C_1, H_1, W_1)$
MaxPool ($K=2, S=2$)	$\to (B, C_1, H_1/2, W_1/2)$
repeat . . .	
Flatten	$\to (B, D)$ where $D = C_LH_LW_L$
Fully connected	$\to (B, n_{\text{classes}})$

4. Single-Layer RNN

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

$$\boldsymbol{a}_t = \boldsymbol{W}\boldsymbol{h}_{t-1} + \boldsymbol{U}\boldsymbol{x}_t + \boldsymbol{b} \in \mathbb{R}^H, \quad \boldsymbol{h}_t = f(\boldsymbol{a}_t)$$
$$\boldsymbol{z}_t = \boldsymbol{V}\boldsymbol{h}_t + \boldsymbol{c}, \quad \hat{\boldsymbol{y}}_t = g(\boldsymbol{z}_t).$$

For back-propagation through time (BPTT) $\delta_t = (\nabla_{\boldsymbol{h}_t}L + \boldsymbol{W}^\top\delta_{t+1}) \odot f'(\boldsymbol{a}_t)$.

5. Transformer Self-Attention — token-wise view

Linear projections per token
Let the input sequence length be T and token embedding dimension d_{model}. For *each* token $\boldsymbol{x}_t \in \mathbb{R}^{d_{\text{model}}}$ we form

$$\boldsymbol{q}_t = \boldsymbol{W}_Q\boldsymbol{x}_t, \quad \boldsymbol{k}_t = \boldsymbol{W}_K\boldsymbol{x}_t, \quad \boldsymbol{v}_t = \boldsymbol{W}_V\boldsymbol{x}_t,$$

where $\boldsymbol{W}_Q, \boldsymbol{W}_K, \boldsymbol{W}_V \in \mathbb{R}^{d_{\text{model}}\times d_k}$. *There is exactly one query, one key, and one value per token.*

Similarity and weighting
For a fixed *query token* t the attention weight placed on *key token* s is

$$\alpha_{t\mapsto s} = \frac{\exp(\boldsymbol{q}_t^\top\boldsymbol{k}_s/\sqrt{d_k})}{\sum_{u=1}^T \exp(\boldsymbol{q}_t^\top\boldsymbol{k}_u/\sqrt{d_k})} \quad \text{(soft-max over all } T \text{ keys).}$$

Thus every query attends to **all** keys; the matrix $\boldsymbol{A} \in \mathbb{R}^{T\times T}$ contains T rows of query distributions, one per token.

Context construction
The context (output) vector for token t is a weighted sum of all values:

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

For a full batch the familiar matrix form $\boldsymbol{A} = \mathrm{softmax}(\boldsymbol{Q}\boldsymbol{K}^\top/\sqrt{d_k})$ and $\boldsymbol{Z} = \boldsymbol{A}\boldsymbol{V}$ is recovered, where $\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V} \in \mathbb{R}^{B\times T\times d_k}$.

Multi-head recap
Split $\boldsymbol{q}, \boldsymbol{k}, \boldsymbol{v}$ into h sub-spaces $d_k = d_{\text{model}}/h$, run attention independently, concatenate the h contexts, then apply a final dense layer.

6. Principal Component Analysis (expanded)

Mean-centred data (two features)

$$\mathbf{X} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \end{bmatrix} \in \mathbb{R}^{2\times n}$$

Covariance (outer-product form)

$$\mathbf{A} = \mathbf{X}\mathbf{X}^\top = \begin{bmatrix} a & b \\ b & c \end{bmatrix}, \quad a, b, c \in \mathbb{R},\ a, c > 0.$$

Characteristic polynomial

$$\det(\mathbf{A} - \lambda\mathbf{I}) = \lambda^2 - (a+c)\lambda + (ac - b^2) = 0.$$

Eigenvalues

$$\lambda_{1,2} = \frac{(a+c) \pm \sqrt{(a-c)^2 + 4b^2}}{2}, \quad \lambda_1 \geq \lambda_2.$$

Eigen-vector for λ_1
Solve $(\mathbf{A} - \lambda_1\mathbf{I})\mathbf{u}_1 = \mathbf{0}$. One convenient choice (assuming $b \neq 0$):

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

Principal-component scores

For each centred sample $\mathbf{x}_i$,

$$h_{1i} = \mathbf{v}_1^\top \mathbf{x}_i \quad \Rightarrow \quad \mathbf{h}_1 = \mathbf{v}_1^\top \mathbf{X} \in \mathbb{R}^{1\times n}.$$

Explained-variance ratio

$$\mathrm{EVR}_1 = \frac{\lambda_1}{\lambda_1 + \lambda_2}, \quad \mathrm{EVR}_2 = 1 - \mathrm{EVR}_1.$$

PCA rank-1 reconstruction form

$$\boxed{\mathbf{B} + \mathbf{C}\mathbf{C}^\mathsf{T}(\mathbf{A} - \mathbf{B})}$$

$\mathbf{A}$:	input data matrix	$(d \times n)$
$\mathbf{B}$:	row means replicated across columns	$(d \times n)$
$\mathbf{C}$:	unit eigen-vector of largest eigen-value	$(d \times 1)$

7. Over-fitting diagnostics and remedies

Symptoms : training loss decreases while validation loss increases; widening train–validation accuracy gap.
Counter-measures : early stopping with patience, L_2 weight decay to the loss, discouraging large weights. Dropout

CNN with Conv-Conv-FC Architecture

Input tensor

$$x \in \mathbb{R}^{H_{\text{in}} \times W_{\text{in}} \times C_{\text{in}}}$$

where H_{in} = input height, W_{in} = input width, C_{in} = input channels.

Convolution 1

$$w^{(1)} \in \mathbb{R}^{K_1 \times K_1 \times C_{\text{in}} \times C_1}, \quad b^{(1)} \in \mathbb{R}^{C_1},$$

$$p_1 = \frac{K_1 - 1}{2} \quad (\text{“same” padding, stride } s_1).$$

$$a_k^{(1)} = \sum_{c=0}^{C_{\text{in}}-1} \mathrm{Conv}\big(x(\cdot,\cdot,c),\ w^{(1)}(\cdot,\cdot,c,k),\ p_1,\ s_1\big) + b_k^{(1)}$$

$$a^{(1)} \in \mathbb{R}^{H_1 \times W_1 \times C_1}, \quad \text{where } 0 \le k < C_1$$

where K_1 = kernel size, C_1 = number of output channels, $H_1 = H_{\text{in}}$ for same padding, $W_1 = W_{\text{in}}$ for same padding.
Activation: $h^{(1)} = g\big(a^{(1)}\big) \in \mathbb{R}^{H_1 \times W_1 \times C_1}$.

Convolution 2

$$w^{(2)} \in \mathbb{R}^{K_2 \times K_2 \times C_1 \times C_2}, \quad b^{(2)} \in \mathbb{R}^{C_2},$$

$$p_2 = \frac{K_2 - 1}{2} \quad (\text{“same” padding, stride } s_2).$$

$$a_k^{(2)} = \sum_{c=0}^{C_1-1} \mathrm{Conv}\big(h^{(1)}(\cdot,\cdot,c),\ w^{(2)}(\cdot,\cdot,c,k),\ p_2,\ s_2\big) + b_k^{(2)}$$

$$a^{(2)} \in \mathbb{R}^{H_2 \times W_2 \times C_2}, \quad \text{where } 0 \le k < C_2$$

where K_2 = kernel size, C_2 = number of output channels, $H_2 = H_1$ for same padding, $W_2 = W_1$ for same padding.
Activation: $h^{(2)} = g\big(a^{(2)}\big) \in \mathbb{R}^{H_2 \times W_2 \times C_2}$.

Flatten

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

Dense (hidden layer)

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

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

where D_h = number of hidden units in the dense layer.

– randomly zeros hidden activations during training , data augmentation, model capacity reduction, k-fold cross-validation for hyper-tuning.

Scheme	**Goal (var.)**	σ^2 formula
Glorot/Xavier SIGmoid,tanh	fwd & bwd = 1	$\frac{2}{n_{\text{in}}+n_{\text{out}}}$
He (Kaiming)	ReLU keep-prob 0.5	$\frac{2}{n_{\text{in}}}$

8. Useful loss functions (mini-batch form)

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

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

9. Chain-Rule reference

For composite $L(\boldsymbol{u}),\ \boldsymbol{u} = g(\boldsymbol{v}),\ \boldsymbol{v} = h(\boldsymbol{x})$ gradient obeys $\nabla_{\boldsymbol{x}} L = \big(\nabla_{\boldsymbol{v}} L\big) \dfrac{\partial \boldsymbol{v}}{\partial \boldsymbol{x}} = \big(\nabla_{\boldsymbol{u}} L\big) \dfrac{\partial \boldsymbol{u}}{\partial \boldsymbol{v}} \dfrac{\partial \boldsymbol{v}}{\partial \boldsymbol{x}}$.

Output layer

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

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

where D_{out} = number of output units (e.g., number of classes for classification).
Prediction: apply softmax(z) for multi–class or $\sigma(z)$ for binary.

10. Model Selection Guide

(a) Plain RNN – sequential data with moderate lengths where local recurrence is enough, e.g. predicting next hourly temperature.
(b) CNN – grid-structured data with strong locality, e.g. classifying hand-written digits in 28 × 28 grayscale images.
(c) Transformer – tasks requiring long-range dependencies and parallel processing, e.g. machine-translation.

11. 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.

12. Sequence Models: Task Types Losses

$\mathrm{L}_{\text{seq}\to\text{Vec, cls}} = \frac{1}{N} \sum_{i=1}^{N} \Big[-\sum_{c=1}^{C} y_{i,c} \ln p_{i,c} \Big]$ (cls)$\mathcal{L}_{\text{seq}\to\text{Seq, reg}} = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{T_i} \sum_{t=1}^{T_i} m_{i,t} \|y_{i,t} - \hat{y}_{i,t}\|_2^2$**Seq,→,Vec**$(many, ß, one) : spamdetection, sentiment, speakerID.$**Seq,→,Seq**$(many, ß, many) : machinetranslation, speechrecognition.$

Unbounded input note
A recurrent update $\boldsymbol{h}t = f(\boldsymbol{h}t - 1, \boldsymbol{x}_t)$ is defined *recursively*, so an RNN can process sequences of arbitrary length.