

# Deep Learning Overview: Models, Architectures, and Optimization

## 0. Notation

Vectors are bold lower-case, and matrices/tensors are bold upper-case.  $\odot$  denotes the Hadamard product, and  $B$  denotes the 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  $\mathbf{x} \in \mathbb{R}^2$  gives

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

Sigmoid function:

$$\sigma(z) = \frac{1}{1 + e^{-z}}.$$

### Binary cross-entropy loss (mini-batch)

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

### Symbolic derivative chain (single sample)

$$\begin{aligned}\frac{\partial L}{\partial z} &= \hat{y} - y, \quad \frac{\partial L}{\partial b} = \frac{\partial L}{\partial z}, \\ \frac{\partial L}{\partial \mathbf{w}} &= \left( \frac{\partial L}{\partial z} \right) (\mathbf{h}^{(1)})^\top, \quad \frac{\partial L}{\partial \mathbf{h}^{(1)}} = \left( \frac{\partial L}{\partial z} \right) \mathbf{w}^\top, \\ \frac{\partial L}{\partial \mathbf{a}^{(1)}} &= \frac{\partial L}{\partial \mathbf{h}^{(1)}} \odot \mathbf{1}_{(\mathbf{a}^{(1)} > 0)}, \quad \frac{\partial L}{\partial \mathbf{b}^{(1)}} = \frac{\partial L}{\partial \mathbf{a}^{(1)}}, \\ \frac{\partial L}{\partial \mathbf{W}^{(1)}} &= \left( \frac{\partial L}{\partial \mathbf{a}^{(1)}} \right) \mathbf{x}^\top.\end{aligned}$$

### Parameter averaging (mini-batch)

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

## 2. Optimizers

- Gradient Descent

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \eta \nabla L(\boldsymbol{\theta}_k).$$

- Stochastic Gradient Descent (SGD)

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

- Momentum

$$\mathbf{v}_k = \beta \mathbf{v}_{k-1} + \eta \nabla L_k, \quad \boldsymbol{\theta}_k = \boldsymbol{\theta}_{k-1} - \mathbf{v}_k.$$

- AdaGrad

$$G_k = G_{k-1} + \nabla L_k \odot \nabla L_k, \quad \boldsymbol{\theta}_k = \boldsymbol{\theta}_{k-1} - \eta \frac{\nabla L_k}{\sqrt{G_k} + \varepsilon}.$$

- RMSProp

$$G_k = \beta G_{k-1} + (1 - \beta) \nabla L_k^2, \quad \boldsymbol{\theta}_k = \boldsymbol{\theta}_{k-1} - \eta \frac{\nabla L_k}{\sqrt{G_k} + \varepsilon}.$$

- Adam

$$\begin{aligned} \mathbf{m}_k &= \beta_1 \mathbf{m}_{k-1} + (1 - \beta_1) \nabla L_k, & \mathbf{v}_k &= \beta_2 \mathbf{v}_{k-1} + (1 - \beta_2) \nabla L_k^2. \\ \hat{\mathbf{m}}_k &= \frac{\mathbf{m}_k}{1 - \beta_1^k}, & \hat{\mathbf{v}}_k &= \frac{\mathbf{v}_k}{1 - \beta_2^k}. \\ \boldsymbol{\theta}_k &= \boldsymbol{\theta}_{k-1} - \eta \frac{\hat{\mathbf{m}}_k}{\sqrt{\hat{\mathbf{v}}_k} + \varepsilon}. \end{aligned}$$

## 3. Convolutional Neural Networks: Detailed Dimensional Analysis

### Single 2-D convolution layer

Input tensor:

$$\mathbf{X} \in \mathbb{R}^{B \times C_{\text{in}} \times H_{\text{in}} \times W_{\text{in}}}.$$

Kernel weights:

$$\mathbf{K} \in \mathbb{R}^{C_{\text{out}} \times C_{\text{in}} \times K_h \times K_w},$$

with stride  $S_h, S_w$  and zero-padding  $P_h, P_w$ .

### Spatial dimensions after convolution

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

Output tensor:

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

### Parameter count

$$C_{\text{out}} C_{\text{in}} K_h K_w + C_{\text{out}}.$$

## Multiply–accumulate operations

$$B C_{\text{out}} H_{\text{out}} W_{\text{out}} C_{\text{in}} K_h K_w.$$

## Receptive field and effective stride (layer $\ell$ )

Initialize  $\text{field}_0 = 1$  and  $\text{stride}_0 = 1$ . For each layer  $\ell$ :

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

## Typical block sequence and shapes

$\mathbf{X}$	shape $(B, C_0, H_0, W_0)$
Conv	$\rightarrow (B, C_1, H_1, W_1)$
BatchNorm+ReLU	$\rightarrow (B, C_1, H_1, W_1)$
MaxPool ( $K = 2, S = 2$ )	$\rightarrow (B, C_1, H_1/2, W_1/2)$
repeat	
Flatten	$\rightarrow (B, D)$ , where $D = C_L H_L W_L$
Fully connected	$\rightarrow (B, n_{\text{classes}})$

## 4. Single-Layer RNN

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

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

For back-propagation through time (BPTT),

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

## 5. Transformer Self-Attention — token-wise view

### Linear projections per token

Let the input sequence length be  $T$  and token embedding dimension  $d_{\text{model}}$ . For each token  $\mathbf{x}_t \in \mathbb{R}^{d_{\text{model}}}$ :

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

where  $\mathbf{W}_Q, \mathbf{W}_K, \mathbf{W}_V \in \mathbb{R}^{d_{\text{model}} \times d_k}$ .

### Similarity and weighting

For a fixed query token  $t$ , the attention weight on key token  $s$  is

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

Thus every query attends to all keys; the matrix  $\mathbf{A} \in \mathbb{R}^{T \times T}$  contains  $T$  rows of query distributions.

## Context construction

$$\mathbf{z}_t = \sum_{s=1}^T \alpha_{t \rightarrow s} \mathbf{v}_s.$$

For a full batch, the matrix form

$$\mathbf{A} = \text{softmax}(\mathbf{Q}\mathbf{K}^\top / \sqrt{d_k}), \quad \mathbf{Z} = \mathbf{AV},$$

where  $\mathbf{Q}, \mathbf{K}, \mathbf{V} \in \mathbb{R}^{B \times T \times d_k}$ .

## Multi-head recap

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

## 6. Principal Component Analysis (expanded)

### Mean-centered 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)

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

### Characteristic polynomial

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

### Eigenvalues

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

### Eigenvector for $\lambda_1$

Solve  $(\mathbf{A} - \lambda_1 \mathbf{I})\mathbf{u}_1 = \mathbf{0}$ . One convenient choice (assuming  $b \neq 0$ ) is

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

### Principal-component scores

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

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

## Explained-variance ratio

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

## PCA rank-1 reconstruction form

$$\mathbf{B} + \mathbf{CC}^T(\mathbf{A} - \mathbf{B}).$$

- A** : input data matrix  $(d \times n)$ ,  
**B** : row means replicated across columns  $(d \times n)$ ,  
**C** : unit eigenvector of largest eigenvalue  $(d \times 1)$ .

## 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, dropout, data augmentation, model capacity reduction, and  $k$ -fold cross-validation.

## 8. Weight initialization schemes

Scheme	Goal	Variance $\sigma^2$
Glorot/Xavier	Balanced forward/backward	$\frac{2}{n_{\text{in}}+n_{\text{out}}}$
He (Kaiming)	ReLU stability	$\frac{2}{n_{\text{in}}}$

## 9. Useful loss functions (mini-batch form)

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

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

## 10. Chain-rule reference

For composite  $L(\mathbf{u})$ ,  $\mathbf{u} = g(\mathbf{v})$ ,  $\mathbf{v} = h(\mathbf{x})$ ,

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

## 11. CNN with Conv–Conv–FC architecture

### Input tensor

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

## Convolution 1

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

Same-padding parameter:

$$p_1 = \frac{K_1 - 1}{2}.$$

$$\begin{aligned} a_k^{(1)} &= \sum_{c=0}^{C_{\text{in}}-1} \text{Conv}(x(\cdot, \cdot, c), w^{(1)}(\cdot, \cdot, c, k), p_1, s_1) + b_k^{(1)}. \\ a^{(1)} &\in \mathbb{R}^{H_1 \times W_1 \times C_1}, \quad 0 \leq k < C_1. \end{aligned}$$

Activation:

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

## Convolution 2

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

Same-padding parameter:

$$p_2 = \frac{K_2 - 1}{2}.$$

$$\begin{aligned} a_k^{(2)} &= \sum_{c=0}^{C_1-1} \text{Conv}(h^{(1)}(\cdot, \cdot, c), w^{(2)}(\cdot, \cdot, c, k), p_2, s_2) + b_k^{(2)}. \\ a^{(2)} &\in \mathbb{R}^{H_2 \times W_2 \times C_2}, \quad 0 \leq k < C_2. \end{aligned}$$

Activation:

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

## Flatten

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

## Dense (hidden layer)

$$\begin{aligned} W^{(3)} &\in \mathbb{R}^{D_h \times D_{\text{flat}}}, \quad b^{(3)} \in \mathbb{R}^{D_h}. \\ a^{(3)} &= W^{(3)} h_{\text{flat}}^{(2)} + b^{(3)} \in \mathbb{R}^{D_h}, \quad h^{(3)} = g(a^{(3)}) \in \mathbb{R}^{D_h}. \end{aligned}$$

## Output layer

$$\begin{aligned} W &\in \mathbb{R}^{D_{\text{out}} \times D_h}, \quad b \in \mathbb{R}^{D_{\text{out}}}. \\ z &= W h^{(3)} + b \in \mathbb{R}^{D_{\text{out}}}. \end{aligned}$$

Prediction: apply softmax( $z$ ) for multi-class or  $\sigma(z)$  for binary classification.

## 12. Model Selection Guide

Plain RNN: sequential data with moderate lengths where local recurrence is enough (e.g., predicting next hourly temperature).

CNN: grid-structured data with strong locality (e.g., classifying handwritten digits in  $28 \times 28$  grayscale images).

Transformer: tasks requiring long-range dependencies and parallel processing (e.g., machine translation).

## 13. Regression vs. Classification

Linear Regression predicts continuous values  $\hat{y} \in \mathbb{R}$  and typically uses MSE loss with a linear output activation.

Binary Classification predicts class probabilities  $\hat{y} \in [0, 1]$  and uses BCE loss with a sigmoid output activation.

Multi-class Classification predicts a probability distribution over  $C$  classes and uses categorical cross-entropy with a softmax output.

## 14. Sequence Models: Task Types and Losses

Sequence-to-Vector (classification):

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

Sequence-to-Sequence (regression):

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

Sequence-to-Vector (many-to-one) examples: spam detection, sentiment classification, speaker identification.

Sequence-to-Sequence (many-to-many) examples: machine translation, speech recognition.

### Unbounded input note

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