Mathematical Foundations of Data Sciences

Gabriel Peyré
CNRS & DMA, École Normale Supérieure

CHAPTER 17

Deep Learning

September 9, 2026

Deep networks learn representations through successive transformations of the input. Their architecture controls which structures they can exploit and how efficiently they can be trained, making it essential to understand the operations within each layer. We develop fully connected and convolutional models and their gradients, then introduce residual connections, normalization, attention, and wavelet scattering.

17.1 Deep Architectures

17.1.1 Deep Network Structure

Deep learning constructs estimators $f(x, \beta)$ by composing parameterized maps. The hidden layers compute a feature representation of the input, and the output layer applies a linear or generalized linear predictor to those features. Unlike a fixed kernel feature map, this representation is learned from the training data.

Feedforward architectures. A feedforward chain has the structure introduced in (15.20), with the loss $\mathcal{L}$ omitted. Starting from $x_0 = x$, compute

$$x_{\ell+1} = f_\ell(x_\ell, \beta_\ell),$$

so that $x_L = f(x, \beta)$ with

$$f(\cdot, \beta) = f_{L-1}(\cdot, \beta_{L-1}) \circ f_{L-2}(\cdot, \beta_{L-2}) \circ \ldots \circ f_0(\cdot, \beta_0)$$

where $\beta = (\beta_0, \ldots, \beta_{L-1})$ collects the parameters, and each layer has the dimensions

$$f_\ell(\cdot, \beta_\ell) : \mathbb{R}^{n_\ell} \to \mathbb{R}^{n_{\ell+1}}.$$

We focus on feedforward chains. More general architectures use branched computational graphs or share parameters across recurrent steps.

The parameters β are fitted by empirical risk minimization (13.11), typically using the stochastic methods of Section 15.2. The resulting objective is generally nonconvex, so Theorem 15.5 does not apply. Under suitable smoothness, noise, and step-size assumptions, one can instead bound measures of stationarity. Initialization and optimization dynamics can also provide implicit regularization.

For chain architectures, reverse mode computes the gradient of the ERM loss (15.13) as described in Section 15.3.5. For convolutional layers, it gives the backpropagation equations (17.3) below. More general computational graphs require the reverse pass to accumulate contributions from every branch and every use of a shared parameter; Section 15.3.4 explains this general procedure.

Deep MLP. A common computational block $f_\ell(\cdot, \beta_\ell)$ consists of
- an affine map, $B_\ell \cdot + b_\ell$ with a matrix $B_\ell \in \mathbb{R}^{\tilde{n}_\ell \times n_\ell}$ and a vector $b_\ell \in \mathbb{R}^{\tilde{n}_\ell}$ parametrized (in most cases linearly) by β_ℓ,
- a nonlinearity independent of β_ℓ, given by $\rho_\ell : \mathbb{R}^{\tilde{n}_\ell} \to \mathbb{R}^{n_{\ell+1}}$

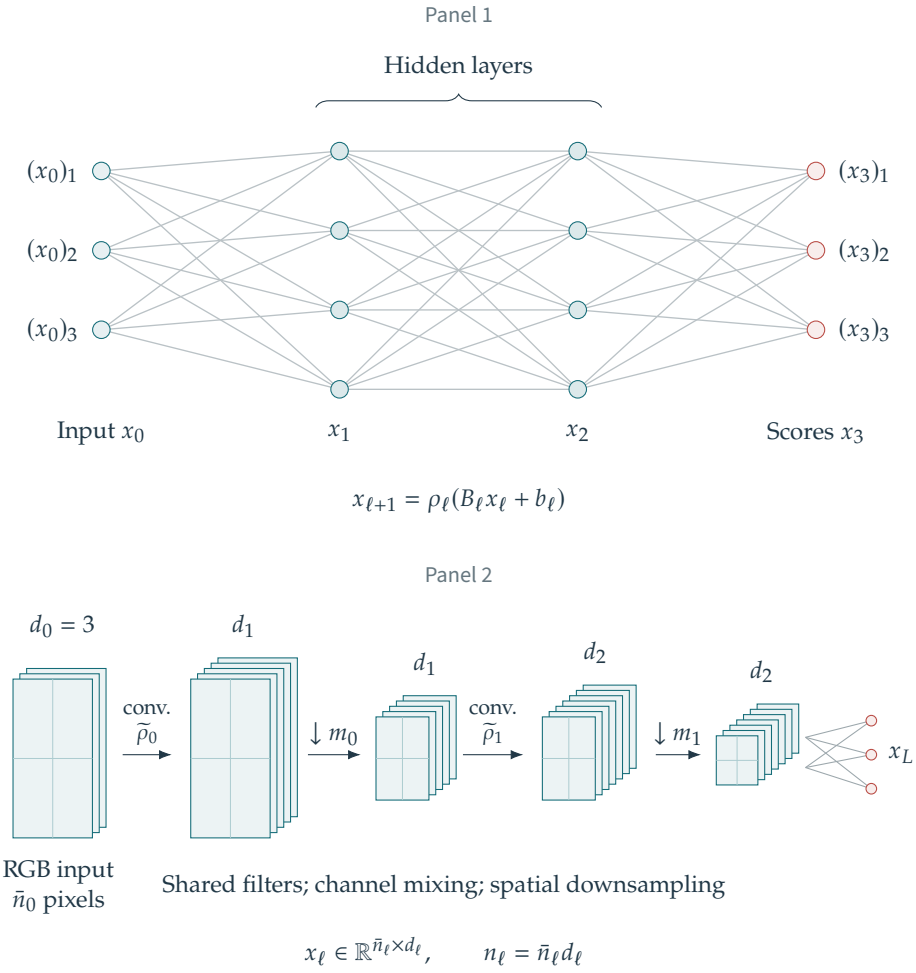

Figure 17.1. Panel 1: fully connected network. Panel 2: convolutional neural network.

which we write as

$$\forall\, x_\ell \in \mathbb{R}^{n_\ell}, \quad f_\ell(x_\ell, \beta_\ell) = \rho_\ell(B_\ell x_\ell + b_\ell) \in \mathbb{R}^{n_{\ell+1}}. \tag{17.1}$$

In a fully connected layer, every entry of B_ℓ and b_ℓ is trainable, so $(B_\ell, b_\ell) = \beta_\ell$. A common choice is a pointwise activation ρ_ℓ, written as $\rho_\ell(z) = (\tilde{\rho}_\ell(z_k))_k$ with $\tilde{\rho}_\ell : \mathbb{R} \to \mathbb{R}$ nonlinear. Such an activation preserves dimension, so $n_{\ell+1} = \tilde{n}_\ell$. Standard examples are the rectified linear unit (ReLU), $\tilde{\rho}_\ell(s) = \max(s, 0)$, and the sigmoid, $\tilde{\rho}_\ell(s) = \theta(s) = (1 + e^{-s})^{-1}$.

Interleaving affine maps with nonlinear activations produces increasingly expressive functions $f(\cdot, \beta)$.

Training the parameters $\beta = (B_\ell, b_\ell)_\ell$ means minimizing (13.11), for example by stochastic gradient descent. Automatic differentiation computes the gradient at a cost proportional to a forward evaluation, namely $O(\sum_\ell n_\ell \tilde{n}_\ell)$ for dense layers with pointwise activations. The chain structure gives the backpropagation formula (15.23).

For regression tasks, one usually uses an affine output layer with a squared ℓ^2 loss L. An output nonlinearity is appropriate when the target is constrained, for example to nonnegative values. For classification, a multiclass logistic map (13.24) converts the final scores into class probabilities.

For high-dimensional inputs such as images, fully connected layers can require prohibitively many parameters. They also ignore spatial structure and symmetries. Convolutional architectures exploit this structure, as detailed in Section 17.1.3.

17.1.2 Perceptron and Shallow Models

The logistic classifiers from Sections 13.4.2 and 13.4.3 are simple instances of this network formulation.

The binary logistic model (13.21) is a one-layer ($L = 1$) instance of (17.1). Omitting the bias, its components are

$$B_0 x = \langle x, \beta \rangle \quad \text{and} \quad \tilde{\rho}_0(u) = \theta(u).$$

The network $f(x, \beta) = \theta(\langle x, \beta \rangle)$ can include a bias by appending a constant coordinate to x. For labels $y \in \{0, 1\}$, it is trained with binary cross-entropy:

$$L(t, y) = -\log(t^y (1-t)^{1-y}) = -y \log(t) - (1-y) \log(1-t).$$

The resulting empirical-risk objective is convex in the parameters.

For K classes, compute the scores $B_0 x = (\langle x, \beta_k \rangle)_{k=1}^K$ and apply the normalized exponential map

$$f(x, \beta) = \mathcal{N}((\exp(\langle x, \beta_k \rangle))_k) \quad \text{where} \quad \mathcal{N}(u) = \frac{u}{\sum_k u_k}$$

For target vectors y on the probability simplex, use the cross-entropy loss

$$L(t, y) = -\sum_{k=1}^K y_k \log(t_k).$$

17.1.3 Convolutional Neural Networks

Signals, images, and videos carry spatial structure that can guide the architecture. Convolutional layers respect translations of the spatial grid, subject to boundary conditions, and reuse the same filters at every location.

At depth ℓ, a convolutional network arranges the activation vector $x_\ell \in \mathbb{R}^{n_\ell}$ as an array in $\mathbb{R}^{\bar{n}_\ell \times d_\ell}$, with $\bar{n}_\ell$ spatial positions and d_ℓ channels, so $n_\ell = \bar{n}_\ell d_\ell$. The spatial positions typically form a one-, two-, or three-dimensional grid. For an RGB image, $\bar{n}_0$ is the number of pixels and $d_0 = 3$. Write $x_\ell = ((x_\ell)_r[i])_{r,i}$, where $i \in \{1, \dots, \bar{n}_\ell\}$ indexes the spatial position and $r \in \{1, \dots, d_\ell\}$ the channel.

We require the linear map $B_\ell : \mathbb{R}^{\bar{n}_\ell \times d_\ell} \to \mathbb{R}^{\bar{n}_\ell \times d_{\ell+1}}$ to be translation equivariant. With periodic boundary conditions, this is equivalent to a sum of convolutions across the input channels. The number of output channels may differ from the number of input channels.

Proposition 17.1. *Let $B : \mathbb{R}^{n \times d} \to \mathbb{R}^{n \times d'}$ be linear, with n spatial positions and periodic boundary conditions. The map $x = (x_r[i])_{r,i} \mapsto Bx$ is translation equivariant, meaning*

$$\forall \tau, \quad B(x[\cdot - \tau]) = (Bx)[\cdot - \tau]$$

(the same shift is applied to every channel) if and only if there exist filters $\psi_{r,s} \in \mathbb{R}^n$, for $1 \leqslant r \leqslant d'$ and $1 \leqslant s \leqslant d$, such that

$$\forall r = 1, \dots, d', \quad (Bx)_r = \sum_{s=1}^d \psi_{r,s} \star x_s \tag{17.2}$$

$$\text{i.e.} \quad (Bx)_r[i] = \sum_{s=1}^d \sum_j \psi_{r,s}[i-j] x_s[j]$$

where $\star$ denotes periodic discrete convolution on the spatial grid, with all spatial indices interpreted periodically.

Proof. Periodic convolution commutes with spatial shifts, so (17.2) defines an equivariant operator. Conversely, suppose B is equivariant. Let δ^s be the impulse at spatial position $i = 0$ in channel s: $(\delta^s)_{s'}[i] = 1_{s=s'} 1_{i=0}$. Define its response by $\psi_{r,s} \overset{\text{def.}}{=} (B\delta^s)_r$. Every input decomposes into shifted impulses:

$$x = \sum_{j,s} x_s[j] \delta^s[\cdot - j],$$

so that by linearity and translation equivariance

$$(Bx)_r = \sum_{j,s} x_s[j] (B\delta^s[\cdot - j])_r = \sum_{j,s} x_s[j] (B\delta^s)_r[\cdot - j] = \sum_{j,s} x_s[j] \psi_{r,s}[\cdot - j] = \sum_s \psi_{r,s} \star x_s$$

Thus each filter is the corresponding impulse response. The adjoint convolution uses the reversed filter, as in (17.3) below. □

The proposition parameterizes an equivariant map B_ℓ by filters $(\psi_\ell)_{r,s}$ indexed by output channel r and input channel s. Writing $x_\ell = ((x_\ell)_s)_{s=1}^{d_\ell}$ for the input channels, the map becomes

$$\forall\, r \in \{1, \dots, d_{\ell+1}\}, \quad (B_\ell x_\ell)_r = \sum_{s=1}^{d_\ell} (\psi_\ell)_{r,s} \star (x_\ell)_s,$$

and the bias is spatially constant to preserve translation equivariance: $(b_\ell)_r[i] = (\tilde{b}_\ell)_r$ for every spatial position i, with one trainable scalar $(\tilde{b}_\ell)_r$ per output channel. This is weight sharing: the same filters ψ_ℓ act at every spatial position.

A pointwise nonlinearity is followed, when desired, by downsampling to reduce the spatial dimension and computational cost. The filtering/downsampling structure resembles the fast wavelet transform. Let m_ℓ be the downsampling factor, commonly $m_\ell = 1$ (no reduction) or $m_\ell = 2$ (a factor of two in each spatial direction). Then

$$\rho_\ell(u) = (\tilde{\rho}_\ell(u_s[m_\ell \cdot]))_{s=1\dots,d_{\ell+1}}.$$

Max-pooling replaces subsampling by local maxima, for example over groups of m_ℓ successive values in one dimension. Strided convolutions provide another way to reduce spatial resolution. These operations preserve equivariance only to shifts compatible with the sampling stride.

Composing local layers enlarges a unit's receptive field: the set of input locations that can affect it. Its actual sensitivity within that set depends on the parameters, nonlinearities, and input. Multiple channels allow different features to be represented, from local edges to more elaborate patterns.

After spatial downsampling, a classifier can use fully connected output layers or aggregate the spatial features before the final prediction.

The trainable parameters are the filters and biases. To describe backpropagation, consider a single channel per layer, omit biases and downsampling, and use periodic convolution. The forward pass computes

$$z_\ell = \psi_\ell \star x_\ell, \qquad x_{\ell+1} = \rho_\ell(z_\ell), \qquad \ell = 0, \dots, L-1.$$

For the loss $\mathcal{E}(\beta) = \mathcal{L}(x_L, y)$, let $a_\ell = \nabla_{x_\ell}\mathcal{E}$ denote the activation gradient. Initialize $a_L = \nabla_1 \mathcal{L}(x_L, y)$ and, for $\ell = L-1, \dots, 0$, compute

$$q_\ell = \rho_\ell'(z_\ell) \odot a_{\ell+1}, \qquad a_\ell = \bar{\psi}_\ell \star q_\ell, \qquad \nabla_{\psi_\ell}\mathcal{E} = \bar{x}_\ell \star q_\ell. \tag{17.3}$$

Here $\bar{v}[i] = v[-i]$ denotes reversal, and $\odot$ denotes coordinatewise multiplication. The activation and filter gradients are distinct. For a filter with restricted support, retain only the trainable coefficients of the filter gradient. A downsampling layer contributes its adjoint upsampling operator to the reverse pass.

These equations are an instance of reverse-mode automatic differentiation. For nonsmooth activations such as ReLU, implementations choose a derivative convention at the kink; away from such points, the ordinary chain rule applies. More general computational graphs, including shared weights and recurrent connections, require accumulation of all contributions as described in Section 15.3.4.

17.1.4 Advanced Architectures

Residual Networks Residual networks add skip connections at selected layers ℓ:

$$f_\ell(x_\ell, \beta_\ell) = x_\ell + \tilde{f}_\ell(x_\ell, \beta_\ell)$$

The residual map $\tilde{f}_\ell(\cdot, \beta_\ell) : \mathbb{R}^{n_\ell} \to \mathbb{R}^{n_\ell}$ preserves the dimension. Such connections help stabilize the training of very deep networks and admit an interpretation as explicit Euler steps for an ordinary differential equation. A common residual block uses a bottleneck:

$$\tilde{f}_\ell(x_\ell, \beta_\ell) = C_\ell^\top \rho_\ell(B_\ell x_\ell)$$

(ignoring biases), where $\beta_\ell = (B_\ell, C_\ell) \in \mathbb{R}^{m_\ell \times n_\ell} \times \mathbb{R}^{m_\ell \times n_\ell}$. Choosing $m_\ell \ll n_\ell$ reduces the parameter count and forces the residual map to pass through a lower-dimensional representation.

Batch normalization Batch normalization [1] standardizes activations during training. For one feature with values $z_1, \dots, z_m$ in a mini-batch, define

$$\bar{z} = \frac{1}{m}\sum_{i=1}^{m} z_i, \qquad s^2 = \frac{1}{m}\sum_{i=1}^{m}(z_i - \bar{z})^2, \qquad \widetilde{z}_i = \gamma \frac{z_i - \bar{z}}{\sqrt{s^2 + \varepsilon}} + \beta.$$

The stabilizer $\varepsilon > 0$ is fixed, while γ and β are learned. During training, the output depends on the whole mini-batch. At inference, stored estimates of the mean and variance are used. For convolutional layers, statistics are usually shared across spatial positions within each channel.

Transformer Networks A self-attention layer [2] acts on a sequence $x = (x_i)_{i=1}^{n}$ with $x_i \in \mathbb{R}^d$. A single scaled dot-product attention head maps

$$x_k \longmapsto \sum_{i=1}^{n} a_{ki} V x_i, \qquad a_{ki} = \frac{\exp(\langle Q x_k,\, K x_i\rangle / \sqrt{d_k})}{\sum_{j=1}^{n} \exp(\langle Q x_k,\, K x_j\rangle / \sqrt{d_k})}.$$

Here $Q, K \in \mathbb{R}^{d_k \times d}$ and $V \in \mathbb{R}^{d_v \times d}$ are the learned query, key, and value matrices. For each query index k, the weights a_{ki} are nonnegative and sum to one over i. Dense attention computes all pairs of scores, so its cost is quadratic in the sequence length for fixed feature dimensions. Transformer blocks combine several attention heads with residual connections, normalization, and positionwise feedforward maps. Positional information is required to distinguish sequence order; without it, unmasked self-attention is permutation equivariant.

17.1.5 Scattering Transform

The scattering transform, introduced by Mallat and collaborators, is a convolutional architecture with fixed wavelet filters, modulus nonlinearities, and local averaging. It is a nonlinear extension of the wavelet transform. Its filters are chosen from the geometry of signals and images rather than learned from data. Under appropriate assumptions on the wavelets and deformations, scattering has stability guarantees with respect to small diffeomorphisms. Scattering coefficients can also be used as fixed features before a learned predictor.

References

[1] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37, pages 448–456. PMLR, 2015.

[2] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, 2017.