# Convolutional Neural Network - Derivations & Proofs

Paul F. Roysdon, Ph.D.

## Contents

**1 Mathematical Derivations & Proofs** 1
1.1 Introduction 1
1.2 Data and Notation 1
1.3 Model Formulation: Discrete Convolution (Cross-Correlation) 2
1.4 Backpropagation Through Convolution 2
1.5 Pooling Layers: Forward and Backward 3
1.6 Classification Head and Loss 4
1.7 Proofs & Identities 4
1.8 Computational Aspects 4
1.9 Algorithm (Supervised CNN Training) 5
1.10 Summary of Variables and Their Dimensions 5
1.11 Summary 5

# 1 Mathematical Derivations & Proofs

## 1.1 Introduction

A Convolutional Neural Network (CNN) is a deep architecture for signals on regular grids (e.g., images) built from convolutional layers, elementwise nonlinearities, and pooling/normalization. A convolutional layer applies a bank of learned local, translation-equivariant linear filters to the input, followed by a pointwise nonlinearity. We derive discrete convolution precisely (with padding/stride/dilation), prove translation equivariance, show matrix (Toeplitz/`im2col`) equivalence, and give full backpropagation formulas for weights, biases, and inputs. We also derive pooling forward/backward and a standard classification head with cross-entropy loss. All variables and dimensions are explicit.

## 1.2 Data and Notation

Let a mini-batch of inputs be

$$\mathbf{X} \in \mathbb{R}^{N\times H\times W\times C_{\text{in}}} \quad \text{(batch, height, width, channels)},$$

with $N$ samples, spatial size $(H, W)$, and $C_{\text{in}}$ channels. A filter bank (kernel) is

$$\mathbf{K} \in \mathbb{R}^{k_h\times k_w\times C_{\text{in}}\times C_{\text{out}}}, \qquad \mathbf{b} \in \mathbb{R}^{C_{\text{out}}},$$

and hyperparameters are *padding* $(p_h, p_w) \in \mathbb{Z}^2_{\geq 0}$, *stride* $(s_h, s_w) \in \mathbb{Z}^2_{\geq 1}$, and *dilation* $(d_h, d_w) \in \mathbb{Z}^2_{\geq 1}$. Define the *effective receptive field* per axis

$$r_h = d_h\,(k_h - 1) + 1, \qquad r_w = d_w\,(k_w - 1) + 1.$$

Then the output spatial size is

$$H_{\text{out}} = \left\lfloor \frac{H + 2p_h - r_h}{s_h} \right\rfloor + 1, \qquad W_{\text{out}} = \left\lfloor \frac{W + 2p_w - r_w}{s_w} \right\rfloor + 1,$$

and the layer output tensor is

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

Throughout, bold capitals denote tensors/matrices; indices are zero- or one-based as convenient (we will write integer equalities explicitly).

## 1.3 Model Formulation: Discrete Convolution (Cross-Correlation)

Deep-learning libraries implement *cross-correlation* (no kernel flip) by default. Define the padded input $\mathbf{X}^{(\text{pad})} \in \mathbb{R}^{N \times (H+2p_h) \times (W+2p_w) \times C_{\text{in}}}$ by zero-padding $\mathbf{X}$. For $n \in [N]$, $(i, j) \in [H_{\text{out}}] \times [W_{\text{out}}]$, $c \in [C_{\text{out}}]$,

$$\mathbf{Y}[n, i, j, c] \;=\; \mathbf{b}[c] \;+\; \sum_{u=0}^{k_h-1} \sum_{v=0}^{k_w-1} \sum_{c'=0}^{C_{\text{in}}-1} \mathbf{K}[u, v, c', c]\, \mathbf{X}^{(\text{pad})}\Big[n,\; is_h + ud_h,\; js_w + vd_w,\; c'\Big]. \tag{1}$$

When $d_h = d_w = s_h = s_w = 1$ and $p_h = p_w = 0$, Eqn. (1) reduces to the classical unit-stride valid correlation.

**Matrix (im2col/Toeplitz) equivalence.** Unfold each output location $(i, j)$ into a row that collects the corresponding receptive-field entries of $\mathbf{X}^{(\text{pad})}$ across channels. This defines a linear operator

$$\mathcal{T}(\mathbf{X}) \in \mathbb{R}^{(NH_{\text{out}}W_{\text{out}}) \;\times\; (k_h k_w C_{\text{in}})},$$

and stack kernels column-wise $\text{vec}(\mathbf{K}) \in \mathbb{R}^{(k_h k_w C_{\text{in}})\, C_{\text{out}}}$. Then

$$\text{mat}(\mathbf{Y}) \;=\; \mathcal{T}(\mathbf{X})\, \text{vec}(\mathbf{K}) \;+\; \mathbf{1}\, \mathbf{b}^{\top}, \tag{2}$$

where $\text{mat}(\mathbf{Y}) \in \mathbb{R}^{(NH_{\text{out}}W_{\text{out}}) \times C_{\text{out}}}$ flattens $(n, i, j)$ into rows and $\mathbf{1}$ is the all-ones column. Thus convolution is a *linear map* w.r.t. both $\mathbf{X}$ and $\mathbf{K}$.

**Translation equivariance (stride = 1, no padding-edge effects).** Let $\tau_\Delta$ shift $\mathbf{X}$ by integer offset $\Delta = (\Delta_h, \Delta_w)$, i.e., $(\tau_\Delta \mathbf{X})[n, a, b, c'] = \mathbf{X}[n, a - \Delta_h, b - \Delta_w, c']$ where defined, else 0. Then for $p$ large enough to avoid border clipping,

$$\text{Conv}(\tau_\Delta \mathbf{X}; \mathbf{K}, \mathbf{b})[n, i, j, c] \;=\; \tau_\Delta\big(\text{Conv}(\mathbf{X}; \mathbf{K}, \mathbf{b})\big)[n, i, j, c], \tag{3}$$

i.e., shifts commute with convolution: the layer is translation *equivariant*.

*Proof.* Substitute $\tau_\Delta \mathbf{X}$ into Eqn. (1) and re-index the sums: the output indices shift by the same $\Delta$ because the kernel support is finite and shared over all locations. ∎

**Nonlinearity and block.** A standard block is

$$\mathbf{Z} \;=\; \sigma(\text{Conv}(\mathbf{X}; \mathbf{K}, \mathbf{b}))\,, \quad \text{e.g.,} \;\; \sigma = \text{ReLU} : \; \sigma(t) = \max\{t, 0\} \; \text{(elementwise)}.$$

## 1.4 Backpropagation Through Convolution

Let $\mathcal{L}$ be the scalar loss. Define the *output sensitivity*

$$\mathbf{G}_Y \;\triangleq\; \frac{\partial \mathcal{L}}{\partial \mathbf{Y}} \in \mathbb{R}^{N \times H_{\text{out}} \times W_{\text{out}} \times C_{\text{out}}}.$$

All gradients follow from linearity (plus the chain rule for nonlinearities).

**Bias gradient.** By Eqn. (1), $\partial\mathbf{Y}[n,i,j,c]/\partial\mathbf{b}[c'] = \mathbf{1}\{c = c'\}$, hence

$$\frac{\partial\mathcal{L}}{\partial\mathbf{b}[c]} = \sum_{n,i,j}\mathbf{G}_Y[n,i,j,c]. \tag{4}$$

**Kernel (weight) gradient.** From Eqn. (1), $\partial\mathbf{Y}[n,i,j,c]/\partial\mathbf{K}[u,v,c',c] = \mathbf{X}^{(\text{pad})}[n, is_h{+}ud_h, js_w{+}vd_w, c']$. Therefore

$$\frac{\partial\mathcal{L}}{\partial\mathbf{K}[u,v,c',c]} = \sum_{n=0}^{N-1}\sum_{i=0}^{H_{\text{out}}-1}\sum_{j=0}^{W_{\text{out}}-1}\mathbf{G}_Y[n,i,j,c]\,\mathbf{X}^{(\text{pad})}[n, is_h+ud_h, js_w+vd_w, c']. \tag{5}$$

**Interpretation.** $\partial\mathcal{L}/\partial\mathbf{K}$ is a (multi-channel) *cross-correlation* between $\mathbf{X}^{(\text{pad})}$ and $\mathbf{G}_Y$, accumulated over batch and output sites.

**Input gradient (transposed convolution).** For a fixed input position $(a,b,c')$ (in the *padded* tensor),

$$\frac{\partial\mathbf{Y}[n,i,j,c]}{\partial\mathbf{X}^{(\text{pad})}[n,a,b,c']} = \sum_{u,v}\mathbf{K}[u,v,c',c]\;\mathbf{1}\{a = is_h + ud_h,\; b = js_w + vd_w\}.$$

Thus

$$\frac{\partial\mathcal{L}}{\partial\mathbf{X}^{(\text{pad})}[n,a,b,c']} = \sum_{c=0}^{C_{\text{out}}-1}\sum_{u=0}^{k_h-1}\sum_{v=0}^{k_w-1}\mathbf{K}[u,v,c',c]\;\mathbf{G}_Y\Big[n,\; \tfrac{a-ud_h}{s_h},\; \tfrac{b-vd_w}{s_w},\; c\Big], \tag{6}$$

where terms with non-integer indices are zero (Kronecker constraints). After computing $\partial\mathcal{L}/\partial\mathbf{X}^{(\text{pad})}$, crop the padding to obtain $\partial\mathcal{L}/\partial\mathbf{X}$.

**Compact forms.** *Proof.* Equations Eqn. (5) and Eqn. (6) can be implemented as standard convolutions:

$$\frac{\partial\mathcal{L}}{\partial\mathbf{K}} = \mathrm{XCorr}_{n,i,j}\big(\mathbf{X}^{(\text{pad})},\,\mathbf{G}_Y\big), \tag{7}$$

$$\frac{\partial\mathcal{L}}{\partial\mathbf{X}} = \mathrm{ConvTransp}\big(\mathbf{G}_Y,\,\mathbf{K};\; s,p,d\big),\;\text{(transpose of forward w.r.t. }\mathbf{X}\text{)}. \tag{8}$$

Alternatively, via Eqn. (2):

$$\mathrm{vec}\Big(\frac{\partial\mathcal{L}}{\partial\mathbf{K}}\Big) = \mathcal{T}(\mathbf{X})^\top\,\mathrm{vec}(\mathbf{G}_Y), \qquad \frac{\partial\mathcal{L}}{\partial\mathbf{b}} = \big(\mathrm{mat}(\mathbf{G}_Y)^\top\mathbf{1}\big), \tag{9}$$

$$\mathrm{mat}\Big(\frac{\partial\mathcal{L}}{\partial\mathbf{X}}\Big) = \mathcal{T}^\top(\mathrm{vec}(\mathbf{K}))\;\mathrm{mat}(\mathbf{G}_Y), \tag{10}$$

where $\mathcal{T}^\top$ denotes the adjoint (col2im) operator. These equalities follow from standard matrix calculus for $\mathcal{L} = \phi(\mathcal{T}(\mathbf{X})\,\theta + \mathbf{1}\mathbf{b}^\top)$. ∎

**Nonlinearity backprop.** For $\mathbf{Z} = \sigma(\mathbf{Y})$ and upstream $\mathbf{G}_Z = \partial\mathcal{L}/\partial\mathbf{Z}$,

$$\mathbf{G}_Y = \mathbf{G}_Z \odot \sigma'(\mathbf{Y}).$$

For ReLU, $\sigma'(t) = \mathbf{1}\{t > 0\}$.

## 1.5 Pooling Layers: Forward and Backward

Let $\mathrm{Pool}_{k_h\times k_w}^{(s_h,s_w)}$ operate per-channel on non-overlapping or strided windows.

**Average pooling.** Forward:

$$\mathbf{Y}[n,i,j,c] \;=\; \frac{1}{k_h k_w}\sum_{u=0}^{k_h-1}\sum_{v=0}^{k_w-1}\mathbf{X}[n,\, is_h+u,\, js_w+v,\, c].$$

Backward distributes gradients uniformly:

$$\frac{\partial\mathcal{L}}{\partial\mathbf{X}[n,a,b,c]} \;=\; \sum_{i,j}\frac{1}{k_h k_w}\,\mathbf{G}_Y[n,i,j,c]\,\mathbf{1}\{a\in[is_h, is_h{+}k_h{-}1],\; b\in[js_w, js_w{+}k_w{-}1]\}.$$

**Max pooling.** Forward selects maxima per window; backward routes gradients to argmax locations:

$$\frac{\partial\mathcal{L}}{\partial\mathbf{X}[n,a,b,c]} \;=\; \sum_{i,j}\mathbf{G}_Y[n,i,j,c]\;\mathbf{1}\Big\{\mathbf{X}[n,a,b,c]=\max_{u,v}\mathbf{X}[n, is_h{+}u, js_w{+}v, c]\Big\}.$$

## 1.6 Classification Head and Loss

Let the final convolutional block output $\mathbf{H}\in\mathbb{R}^{N\times H'\times W'\times C'}$. **Global average pooling** (GAP) aggregates spatially:

$$\mathbf{g}[n,c] \;=\; \frac{1}{H'W'}\sum_{i=0}^{H'-1}\sum_{j=0}^{W'-1}\mathbf{H}[n,i,j,c], \quad \mathbf{g}\in\mathbb{R}^{N\times C'}.$$

A linear readout to $K$ classes:

$$\mathbf{r}=\mathbf{g}\,\mathbf{W}_{\text{cls}}+\mathbf{b}_{\text{cls}}, \qquad \mathbf{W}_{\text{cls}}\in\mathbb{R}^{C'\times K},\; \mathbf{b}_{\text{cls}}\in\mathbb{R}^{K}.$$

With one-hot labels $\mathbf{y}\in\{0,1\}^{N\times K}$ and softmax $p_{nk}=\exp(r_{nk})/\sum_t\exp(r_{nt})$, the cross-entropy loss is

$$\mathcal{L} \;=\; -\sum_{n=1}^{N}\sum_{k=1}^{K}\mathbf{y}_{nk}\,\log p_{nk}, \quad \Rightarrow \quad \frac{\partial\mathcal{L}}{\partial\mathbf{r}} \;=\; \mathbf{p}-\mathbf{y}.$$

Gradients propagate through GAP by uniform redistribution over spatial sites.

## 1.7 Proofs & Identities

**Convolution is a linear map.** *Proof.* For fixed $(\mathbf{K},\mathbf{b})$, Eqn. (1) is affine in $\mathbf{X}$: $\mathrm{Conv}(\alpha\mathbf{X}_1+\beta\mathbf{X}_2)=\alpha\,\mathrm{Conv}(\mathbf{X}_1)+\beta\,\mathrm{Conv}(\mathbf{X}_2)$ plus $\mathbf{b}$. Similarly, for fixed $\mathbf{X}$, it is linear in $\mathbf{K}$. ■

**Translation equivariance (formal).** *Proof.* Let $s_h=s_w=1$ and padding large enough to avoid boundary truncation. For any $\Delta$, by substituting $\tau_\Delta\mathbf{X}$ into Eqn. (1) and re-indexing $i'=i+\Delta_h$, $j'=j+\Delta_w$, $\mathbf{Y}_{\tau_\Delta\mathbf{X}}[i,j,c]=\mathbf{Y}_{\mathbf{X}}[i-\Delta_h, j-\Delta_w, c]$, which is the statement of Eqn. (3). ■

**Backprop correctness (matrix view).** *Proof.* Let $\mathbf{Y}=\mathcal{T}(\mathbf{X})\,\Theta+\mathbf{1}\mathbf{b}^\top$ with $\Theta=\mathrm{vec}(\mathbf{K})$. For any scalar loss $\mathcal{L}(\mathbf{Y})$, matrix calculus gives $\partial\mathcal{L}/\partial\Theta=\mathcal{T}(\mathbf{X})^\top\,\mathrm{vec}(\partial\mathcal{L}/\partial\mathbf{Y})$ and $\partial\mathcal{L}/\partial\mathbf{X}=\mathcal{T}^*\big(\Theta\,(\partial\mathcal{L}/\partial\mathbf{Y})^\top\big)$, where $\mathcal{T}^*$ is the adjoint operator (col2im). These match Eqn. (9). ■

## 1.8 Computational Aspects

A direct convolution costs

$$\mathcal{O}(N\,H_{\text{out}}W_{\text{out}}\,k_hk_w\,C_{\text{in}}C_{\text{out}})\,.$$

Implementations use im2col+GEMM, Winograd (for small 3×3), or FFT for large kernels. Memory/computation scale linearly with $C_{\text{in}}C_{\text{out}}$ and kernel area.

### 1.9 Algorithm (Supervised CNN Training)

1. **Input:** batch $(\mathbf{X}, \mathbf{y})$, parameters $\{\mathbf{K}^{(\ell)}, \mathbf{b}^{(\ell)}\}_{\ell=1}^{L}$ and head $(\mathbf{W}_{\text{cls}}, \mathbf{b}_{\text{cls}})$; hyperparameters $(p, s, d)$, learning rate $\eta$.

2. **Forward:** For layers $\ell = 1, \ldots, L$,

$$\mathbf{Y}^{(\ell)} \leftarrow \mathrm{Conv}\big(\mathbf{X}^{(\ell-1)}; \mathbf{K}^{(\ell)}, \mathbf{b}^{(\ell)}\big), \quad \mathbf{X}^{(\ell)} \leftarrow \sigma\big(\mathbf{Y}^{(\ell)}\big),$$

   optionally with pooling/normalization between layers. Apply GAP and linear head, then softmax and cross-entropy.

3. **Backward:** Compute $\partial\mathcal{L}/\partial\mathbf{r} = \mathbf{p} - \mathbf{y}$, backprop through head and GAP, then per layer

$$\mathbf{G}_Y^{(\ell)} \leftarrow \mathbf{G}_X^{(\ell)} \odot \sigma'\big(\mathbf{Y}^{(\ell)}\big), \quad \frac{\partial\mathcal{L}}{\partial\mathbf{K}^{(\ell)}} \leftarrow \text{Eqn. (5)}, \quad \frac{\partial\mathcal{L}}{\partial\mathbf{b}^{(\ell)}} \leftarrow \text{Eqn. (4)}, \quad \mathbf{G}_X^{(\ell-1)} \leftarrow \text{Eqn. (6) (cropped)}.$$

4. **Update:** Apply an optimizer (SGD/Adam/AdamW) to all parameters.

### 1.10 Summary of Variables and Their Dimensions

- $\mathbf{X} \in \mathbb{R}^{N\times H\times W\times C_{\text{in}}}$: input batch.
- $\mathbf{K} \in \mathbb{R}^{k_h\times k_w\times C_{\text{in}}\times C_{\text{out}}}$: convolution kernels.
- $\mathbf{b} \in \mathbb{R}^{C_{\text{out}}}$: per-output-channel bias.
- $(p_h, p_w)$, $(s_h, s_w)$, $(d_h, d_w)$: padding, stride, dilation (scalars per axis).
- $\mathbf{Y} \in \mathbb{R}^{N\times H_{\text{out}}\times W_{\text{out}}\times C_{\text{out}}}$: pre-activation outputs.
- $\mathbf{Z} = \sigma(\mathbf{Y})$: post-activation outputs (same shape as $\mathbf{Y}$).
- $\mathcal{T}(\mathbf{X}) \in \mathbb{R}^{(NH_{\text{out}}W_{\text{out}})\times(k_hk_wC_{\text{in}})}$: im2col operator.
- $\mathbf{G}_Y = \partial\mathcal{L}/\partial\mathbf{Y}$, $\mathbf{G}_X = \partial\mathcal{L}/\partial\mathbf{X}$: sensitivities.
- Head: $\mathbf{W}_{\text{cls}} \in \mathbb{R}^{C'\times K}$, $\mathbf{b}_{\text{cls}} \in \mathbb{R}^{K}$; logits $\mathbf{r}$, probabilities $\mathbf{p}$.

### 1.11 Summary

From first principles, a convolutional layer is a translation-equivariant, local linear operator with shared weights, expressed by Eqn. (1) (or equivalently the Toeplitz product Eqn. (2)). Backpropagation yields closed-form gradients: the bias sum Eqn. (4), kernel cross-correlation Eqn. (5), and input gradient as a transposed convolution Eqn. (6)/Eqn. (7). Pooling layers are simple linear (average) or selection (max) operators with corresponding adjoints. Stacking conv–nonlinearity–pooling blocks with a classification head and training via cross-entropy completes a CNN, with all dimensions and derivations explicitly specified.