Convolutional Neural Network Basics

Applied Deep Learning — Chapter 4

Yin Yang

Foundation Books Project

Roadmap

From MNIST MLP to Image Structure

Locally Connected Layers

Convolution: Local Connections + Shared Weights

Classical Image Kernels

ConvNets on MNIST

Channels and 1×1 Convolutions

CIFAR-10 Baseline

Practical Checklist

Summary

From MNIST MLP to Image Structure

Why Flattening Hides Information

The MNIST MLP treated a 28×28 image as a vector $x \in \mathbb{R}^{784}$.

- Pixel values preserved.
- Spatial structure lost as explicit input: rows, columns, neighborhoods are no longer presented to the model.
- Two pixels in the same stroke are likely related; flattening makes the dense layer learn that from coordinates alone.

Convolutional networks keep the 2D layout and read *local patches* first.

Inductive Bias of CNNs

Spatial structure
Pixel values organized by row, column, and channel.

Inductive bias
Architectural assumption that makes some functions easier to learn.

CNN bias
Local image patterns matter; similar local patterns can be useful at many positions.

An inductive bias is a tradeoff, not a guarantee.

Locally Connected Layers

1D Locally Connected Layer

Input $x = (x_0, \ldots, x_7) \in \mathbb{R}^8$, window size $k = 3$, stride $s = 1$.

Each output unit reads only k neighbors:

$$y_j = \phi(w_{j,0}x_j + w_{j,1}x_{j+1} + w_{j,2}x_{j+2} + b_j).$$

Subscript j on weights: not shared between positions.

Number of valid output positions:

$$\left\lfloor \frac{n-k}{s} \right\rfloor + 1 = \left\lfloor \frac{8-3}{1} \right\rfloor + 1 = 6.$$

Parameters: $6 \times (3+1) = 24$.

Receptive Field, Stride, Padding

Receptive field Set of input positions that can affect one output unit.

Kernel size k Width of the local window.

Stride s Step between adjacent window positions.

Padding p Artificial border values (often zeros) added so windows can sit near boundaries.

Output length in one dimension:

$$\left\lfloor \frac{n + 2p - k}{s} \right\rfloor + 1.$$

Verify shapes before training. Wrong stride/padding is a common bug.

2D Locally Connected Layer

Input $X \in \mathbb{R}^{28\times 28}$. A 3×3 patch around (i, j):

$$Y_{i,j} = \phi\left(b_{i,j} + \sum_{a=0}^{k_h-1} \sum_{c=0}^{k_w-1} W_{i,j,a,c} X_{i+a,j+c} \right).$$

Weights depend on (i, j) and on patch position (a, c).

Same-padded 3×3 on MNIST, 16 output channels:

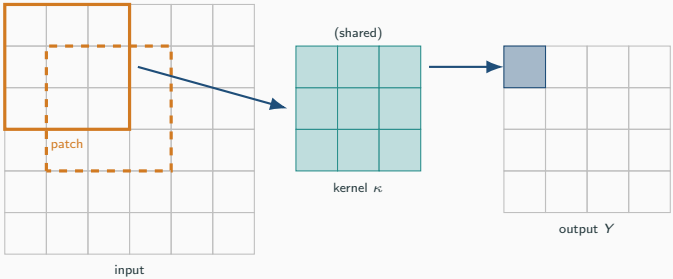
$$28 \times 28 \times 16 \times (3 \times 3 + 1) = 125{,}440 \text{ parameters.}$$

Locality reduces fan-in, but *location-specific* weights still pile up.

Convolution: Local Connections + Shared Weights

The Sharing Constraint

Convolution adds one rule: the same kernel slides across positions.



- Same kernel κ applied at every location $\Rightarrow$ one output *feature map*.
- Library convolution = cross-correlation (no flip). Naming is convention.

Translation Equivariance

Definition
A layer is *translation equivariant* when shifting the input pattern shifts the corresponding output response, rather than changing its meaning.

- Stride-1 convolution has this property away from boundaries.
- Stride > 1: only equivariant for shifts aligned with the stride.
- Same kernel applied at every spatial location encodes this directly.

Parameter Count Drops Dramatically

3×3 conv on MNIST, 1 input channel, 16 output channels:

$$(3 \times 3 \times 1 + 1) \times 16 = 160 \text{ parameters.}$$

Compare:

- Locally connected (16 ch, same shape): 125,440.
- Dense first hidden layer (256 units): 200,960.

Smaller, not because the layer ignores the image, but because it assumes the same local detector can be useful everywhere.

When Sharing Is Imperfect

- Tightly aligned document: a horizontal mark at the top vs. at the bottom may have different semantic meaning.
- Medical or satellite image: top/bottom/center regions can carry distinct physical meaning.
- Tasks where *absolute position* matters violate the assumption.

Mitigations: positional features, padding choices, or different architectures.

Classical Image Kernels

Hand-Designed Kernels Predate CNNs

Classical image processing already shared weights:

- Identity: leaves the center unchanged.
- Box blur: averages the 3×3 neighborhood, weight $1/9$.
- Gaussian blur: more weight to center than corners.
- Sharpen: subtracts neighbors from center.
- Sobel / vertical-edge kernels: detect intensity changes.

ConvNets learn kernels whose responses help the loss.

A Vertical-Edge Calculation by Hand

Patch and kernel:

$$P = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix}, \quad K = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}.$$

Cross-correlation response (sum of entrywise products):

$$0 + 0 + 0 + 0 + 0 + 1 + 0 + 0 + 1 = 2.$$

Bright on the right, dark on the left $\Rightarrow$ positive response. Reverse the patch: negative response. Constant patch: near zero.

ConvNets on MNIST

Adding the Channel Axis

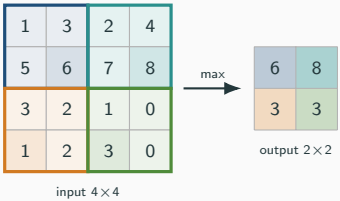
Raw MNIST: 28×28. For `Conv2D`, add a channel axis:

- One example: `(28, 28, 1)` (channels-last).
- Batch: `(batch, 28, 28, 1)`.
- Input channel = measured layer (here, grayscale).
- Output channel = one learned feature map.

CIFAR-10 already has 3 RGB channels; no axis to add.

Max Pooling

$$Y_{i,j,c} = \max_{0 \le a < k_h,\ 0 \le b < k_w} X_{si+a,sj+b,c}.$$



Pooling trades location detail for smaller tensors. No trainable weights; independent per channel.

A Small MNIST ConvNet

Stride 1, same padding for convs; default 2×2 pooling:

$$
\begin{aligned}
(28, 28, 1) &\to \mathrm{Conv2D}(16, 3{\times}3) \\
&\to \mathsf{ReLU} \to \mathrm{MaxPool}(2{\times}2) \\
&\to \mathrm{Conv2D}(32, 3{\times}3) \\
&\to \mathsf{ReLU} \to \mathrm{MaxPool}(2{\times}2) \\
&\to \mathrm{Flatten} \to \mathrm{Dense}(10)
\end{aligned}
$$

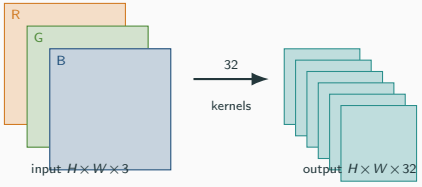
Final dense layer outputs 10 logits; cross-entropy is configured `from_logits=True`.

Channels and 1×1 Convolutions

Multi-Channel Convolution

An output channel reads *all* input channels inside the patch:

$$Y_{i,j,c_{\text{out}}} = b_{c_{\text{out}}} + \sum_{a,b'} \sum_{c_{\text{in}}} \kappa_{a,b',c_{\text{in}},c_{\text{out}}} X_{i+a,j+b',c_{\text{in}}}.$$



- RGB: a 3×3 conv reads 3×3 in R, G, B.
- CIFAR-10 first layer (32 out): $(3 \cdot 3 \cdot 3 + 1) \times 32 = 896$ params.
- C_{in} can be any positive integer (hyperspectral, medical volumes).

The 1×1 Convolution

$$Y_{i,j,c_{\text{out}}} = b_{c_{\text{out}}} + \sum_{c_{\text{in}}} \kappa_{0,0,c_{\text{in}},c_{\text{out}}} X_{i,j,c_{\text{in}}}.$$

- Spatial receptive field of size 1.
- Mixes channels, not neighbors. Same dense transform applied per pixel.
- Cannot detect a horizontal edge or a corner alone.
- $64 \to 16$ channels: $(64 + 1) \times 16 = 1{,}040$ parameters.

Use case: change channel count without changing height/width.

CIFAR-10 Baseline

CIFAR-10: A Harder Image Dataset

- $32 \times 32 \times 3$ RGB images.
- 10 classes: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck.
- 50,000 train + 10,000 test.
- Classroom split: 45,000 train, 5,000 validation, official 10,000 test.

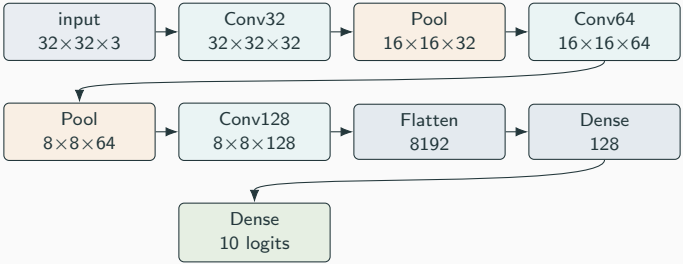
Harder than MNIST: cluttered backgrounds, variable position, color matters, visually similar classes (cat/dog, automobile/truck).

Keras 3 ConvNet Baseline (sketch)

$$
\begin{aligned}
\text{Input}(32, 32, 3) &\to \text{Conv2D}(32, 3, \text{same}, \text{ReLU}) \\
&\to \text{MaxPool} \\
&\to \text{Conv2D}(64, 3, \text{same}, \text{ReLU}) \\
&\to \text{MaxPool} \\
&\to \text{Conv2D}(128, 3, \text{same}, \text{ReLU}) \\
&\to \text{Flatten} \to \text{Dense}(128, \text{ReLU}) \\
&\to \text{Dense}(10) \text{ logits}
\end{aligned}
$$

Loss: `SparseCategoricalCrossentropy(from_logits=True)`. Optimizer: `Adam(1e-3)`.

Shape Path Through the ConvNet



Where the Parameters Live

Layer	Calculation	Trainable
`Conv2D(32)`	$(3 \cdot 3 \cdot 3 + 1) \times 32$	896
`Conv2D(64)`	$(3 \cdot 3 \cdot 32 + 1) \times 64$	18,496
`Conv2D(128)`	$(3 \cdot 3 \cdot 64 + 1) \times 128$	73,856
`Dense(128)`	$(8{,}192 + 1) \times 128$	1,048,704
`Dense(10)`	$(128 + 1) \times 10$	1,290

Dense head dominates. If the ConvNet overfits, audit the head before growing the convolution stack. Global pooling is a common alternative.

Receptive Field Recursion

Stacking layers grows the region of the original image that influences one later cell:

$$r_\ell = r_{\ell-1} + (k_\ell - 1) j_{\ell-1}, \quad j_\ell = j_{\ell-1}\, s_\ell, \quad r_0 = j_0 = 1.$$

- r_ℓ: receptive-field size after layer ℓ.
- j_ℓ: effective stride.

Small 3×3 kernels can still support large image evidence after several stages of convolution and pooling.

Practical Checklist

Before Changing the Model

- Confirm input batch shape `(batch, 32, 32, 3)` after normalization.
- Confirm labels are 1D and in CIFAR-10 class order.
- Read `model.summary()`; trace each height/width/channel change.
- Check the final layer outputs 10 logits; set `from_logits=True`.
- Treat val. accuracy and elapsed time as run-record fields, not final claims.

When the Curve Flattens

Validation accuracy rises early then stalls while training accuracy keeps rising:

- First suspect: overfitting or validation-distribution mismatch — not "ConvNet too small."
- Verify split, evaluation mode, augmentation policy.
- Try one controlled change: less capacity, defined augmentation, early stopping.

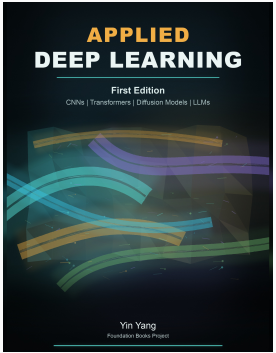
Avoid changing optimizer, regularization, schedule, and test policy all at once.

Summary

Summary

- Flattening hides spatial structure; CNNs preserve the 2D grid.
- Locally connected layers introduce receptive fields, stride, padding.
- Convolution = local connections + shared weights.
- Classical kernels (blur, sharpen, edge) show why sharing is natural.
- Channels stack measurements at each pixel; 1×1 convs mix channels without spatial reach.
- Pooling reduces spatial size; receptive field still grows by depth.
- CIFAR-10 baseline: shapes, parameter audit, and dense-head check come before architecture tuning.

Next: training neural networks — losses, optimizers, schedules, initialization, regularization, and CIFAR-10 practice.



Applied Deep Learning

Chapter overview slides for the book by Yin Yang.

Book page	`foundation-books.github.io/applied-deep-learning`
Code	`github.com/foundation-books/applied-deep-learning-code`
Print	`amazon.com/dp/B0GZF7QRSH`
Kindle	`amazon.com/dp/B0GZF9221X`
License	CC BY-NC-ND 4.0

These free overview slides may be shared non-commercially with attribution and without modifications. Full explanations, exercises, and companion-code context are in the book and linked repository.