Weight Sharing in Convolutional Neural Networks

Sayan CHAKI

April 9, 2025

Outline

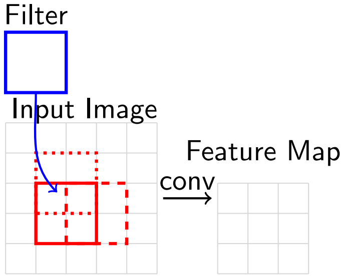
What is Weight Sharing?

- A fundamental concept in convolutional neural networks (CNNs)
- In traditional fully-connected networks: each input-output connection has **unique weights**
- In CNNs: the same weights are reused across different spatial locations

The same filter weights are applied at every position in the input

The Convolutional Layer

- A **filter/kernel** (e.g., 3×3) slides across the input image
- At each position, performs the same computation
- Produces an output feature map
- The **same filter weights** are used at **every spatial location**



Why Share Weights?

Key Advantages

- **Translation Equivariance**
  - Objects are detected regardless of their position in the image
  - If we shift the input, the output shifts accordingly
- **Parameter Efficiency**
  - Drastically reduces the number of learnable parameters
  - Makes training feasible for large images
- **Better Generalization**
  - Captures spatial patterns regardless of location
  - Strong inductive bias for visual data

Parameter Efficiency: A Comparison

Consider processing a 32×32 image:

Fully Connected Layer:

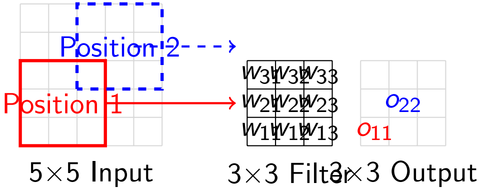
- Each output neuron connected to all 1,024 inputs
- For 64 output neurons: $1,024 \times 64 = 65,536$ parameters

Convolutional Layer:

- 8 filters of size 3×3
- Only $3 \times 3 \times 8 = 72$ parameters
- **910× fewer parameters!**

Weight sharing enables deep architectures that can process high-resolution images with reasonable computational resources

Visual Explanation: Weight Sharing



The same filter weights w_{ij} are used at every position!

Mathematical Definition

For a filter W of size $k \times k$ and an input image I:

$$\text{Output}(i,j) = \sum_{m=0}^{k-1} \sum_{n=0}^{k-1} W(m,n) \cdot I(i+m, j+n) \tag{1}$$
$$= (W * I)(i,j) \tag{2}$$

Where:

- $W(m,n) =$ Weight at position (m,n) in the filter
- $I(i+m, j+n) =$ Input value at offset position
- $* =$ Convolution operation

Critical point: The weights $W(m,n)$ remain the **same** for all spatial positions (i,j) in the input image.

Intuitive Analogies

The Stencil Analogy

- A stencil pressed across different areas of a page
- The **same pattern** is applied everywhere
- Different input regions produce different outputs
- But the **stencil itself** doesn't change

The Cookie Cutter Analogy

- One cookie cutter shapes multiple cookies
- **Reuses the same tool** across the dough
- Different dough regions (inputs) produce different cookies (outputs)
- But the **cutter itself** is identical

In both cases, we use the **same tool** across different locations to detect patterns or features.

Food for Thought

What if we didn't share weights?

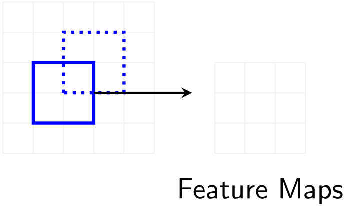
- For a 224×224 image with 3×3 filters:
  - We'd need $3 \times 3 \times 222 \times 222$ parameters per filter!
  - $\approx$ 148,000 parameters vs. just 9 with weight sharing
- Training would be extremely inefficient
- Models would overfit drastically

When might weight sharing not be ideal?

- When processing data with strong positional dependencies
- Example: Face recognition where features like eyes/nose have fixed positions
- Some architectures use partially locally-connected layers without full weight sharing

Summary: Benefits of Weight Sharing

- **Drastically Reduces Parameters**
  - Makes deep networks feasible
  - Enables faster training
- **Provides Translation Equivariance**
  - Detects patterns regardless of position
- **Improves Generalization**
  - Strong inductive bias for spatial data
- **Enables Processing Large Images**
  - Parameters independent of input size



Weight sharing is what makes convolutional networks so powerful and efficient for visual tasks!