MA477: Data Science
Lesson 26-27 Outline — 19 / 24 March 2026
United States Military Academy, West Point
Instructor: MAJ Patrick Kuiper

---

1 Administrative

- Calendar review
- Student review
- CNN Discussion
- Coding Exercise

2 CNN Lesson Objectives

- Understand the architecture of a typical convolutional neural network to include convolution and pooling layers.
- Understand how data augmentation improves model performance in image classification.
- Use functions in Python's keras module to fit convolutional neural networks for image classification.

3 Convolutional Neural Networks: Guided Lesson via Questions

Learning Objective

Understand convolution using **Géron-style terminology**: feature maps, kernels, stride, padding, and how output feature maps are computed mathematically.

—

Question 1: What problem do CNNs solve?

In a fully connected network, the input is flattened into a vector, losing spatial structure.

In Géron's terminology, an image is treated as a:

- **Input feature map** of size $n_H \times n_W \times n_C$

where:

- n_H: height
- n_W: width
- n_C: number of channels

CNNs address this by using:

- **Local receptive fields** (each neuron sees only a small region)
- **Shared weights** (same kernel applied everywhere)

Key Idea: Instead of learning independent weights for each pixel, CNNs learn small filters that detect patterns across the entire feature map.

—

Question 2: How does Géron formulate convolution mathematically?

In Géron, we define:

- Input feature map: X
- Kernel (filter): K of size $f \times f$
- Bias: b
- Stride: s
- Padding: p

The output is called a **feature map** Z.
The convolution operation at position (i, j) is:

$$Z_{i,j} = \sum_{u=0}^{f-1} \sum_{v=0}^{f-1} K_{u,v} \cdot X_{i \cdot s+u,\ j \cdot s+v} + b$$

Then apply activation:

$$A_{i,j} = g(Z_{i,j})$$

—

Output Size Formula (Géron):

$$n_{\text{out}} = \left\lfloor \frac{n_{\text{in}} - f + 2p}{s} \right\rfloor + 1$$

This determines the spatial dimensions of the output feature map.

—

Interpretation (Géron language):

- Extract a **receptive field** (local patch)
- Apply the kernel (shared weights)
- Compute a weighted sum + bias
- Store result in the output feature map

—

Question 3: What is weight sharing and feature maps?

Each kernel produces one **feature map**.

- One kernel $\rightarrow$ one feature map
- Multiple kernels $\rightarrow$ multiple feature maps

Weight sharing means:

- The same kernel values K are used at every spatial location

- The kernel detects the same pattern across the entire image

Result:

- Far fewer parameters than fully connected layers
- Translation-invariant pattern detection

—

Question 4: What additional operations are used in Géron?

After convolution, Géron describes two key operations:

1. **Activation Function (Nonlinearity)**
   Typically ReLU:

   $$g(z) = \max(0, z)$$

   This produces the **activation map**.
   —

2. **Pooling Layer**
   Pooling reduces spatial resolution.
   For max pooling:

   - Define pool size (e.g., 2×2)
   - Define stride s
   - Take the maximum value in each region

   Mathematically:

   $$A'_{i,j} = \max_{\text{region}} A_{i,j}$$

   —

—

Key Géron Terminology Summary

- **Input feature map**: original image or previous layer output
- **Kernel / Filter**: small matrix of weights
- **Receptive field**: local region of input
- **Stride**: step size of kernel movement
- **Padding**: border added around input
- **Feature map**: output of convolution
- **Activation map**: after applying nonlinearity

—

Key Takeaway

A convolutional layer in Géron's framework:

- Applies kernels over local receptive fields
- Uses shared weights to detect patterns
- Produces feature maps that encode learned features

—

—

Instructor Example: Convolution Step-by-Step (Géron Notation)

We illustrate a simple convolution using the notation and terminology from Géron.

Given:

$$X = \begin{bmatrix} 1 & 2 & 0 \\ 0 & 1 & 3 \\ 2 & 1 & 0 \end{bmatrix} \quad K = \begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix} \quad b = 0$$

Parameters:

- Kernel size: $f = 2$
- Stride: $s = 1$
- Padding: $p = 0$ (valid)

Output size:

$$n_{\text{out}} = \frac{3 - 2 + 0}{1} + 1 = 2$$

So the output will be 2×2.

—

Step 1: Compute $Z_{1,1}$

Extract the top-left patch:

$$X_{1:2,1:2} = \begin{bmatrix} 1 & 2 \\ 0 & 1 \end{bmatrix}$$

Apply the convolution formula:

$$Z_{1,1} = (1)(1) + (2)(0) + (0)(-1) + (1)(1) = 2$$

Interpretation:

- Multiply corresponding entries
- Sum the results
- Add bias (here zero)

—

Step 2: Move the filter (stride = 1)

We slide the filter one step to the right and repeat the same process.

This continues across all valid positions.

—

Final Result (computed similarly):

$$Z = \begin{bmatrix} 2 & 4 \\ -1 & 0 \end{bmatrix}$$

—

Class Exercise

Given:

$$X = \begin{bmatrix} 2 & 1 & 0 \\ 1 & 3 & 2 \\ 0 & 1 & 2 \end{bmatrix} \quad K = \begin{bmatrix} 1 & -1 \\ 0 & 1 \end{bmatrix} \quad b = 0$$

Parameters:

- Kernel size: $f = 2$
- Stride: $s = 1$
- Padding: $p = 0$

—

Tasks:

1. Compute the output dimension
2. Compute all values:
$$Z_{1,1},\ Z_{1,2},\ Z_{2,1},\ Z_{2,2}$$
3. Show each step:
    - Extracted patch
    - Elementwise multiplication
    - Final sum

—

Student Solution

Output size:

$$n_{\text{out}} = \frac{3-2}{1} + 1 = 2$$

—

Compute $Z_{1,1}$

$$\begin{bmatrix} 2 & 1 \\ 1 & 3 \end{bmatrix}$$

$$(2)(1) + (1)(-1) + (1)(0) + (3)(1) = 4$$

—

Compute $Z_{1,2}$

$$\begin{bmatrix} 1 & 0 \\ 3 & 2 \end{bmatrix}$$

$$(1)(1) + (0)(-1) + (3)(0) + (2)(1) = 3$$

—

Compute $Z_{2,1}$

$$\begin{bmatrix} 1 & 3 \\ 0 & 1 \end{bmatrix}$$

$$(1)(1) + (3)(-1) + (0)(0) + (1)(1) = -1$$

—

Compute $Z_{2,2}$

$$\begin{bmatrix} 3 & 2 \\ 1 & 2 \end{bmatrix}$$

$$(3)(1) + (2)(-1) + (1)(0) + (2)(1) = 3$$

—

Final Output:

$$Z = \begin{bmatrix} 4 & 3 \\ -1 & 3 \end{bmatrix}$$

—

Key Takeaway

- Convolution applies the same kernel across all positions
- Each output value is a weighted sum of a local patch
- Stride controls movement, padding controls borders

—