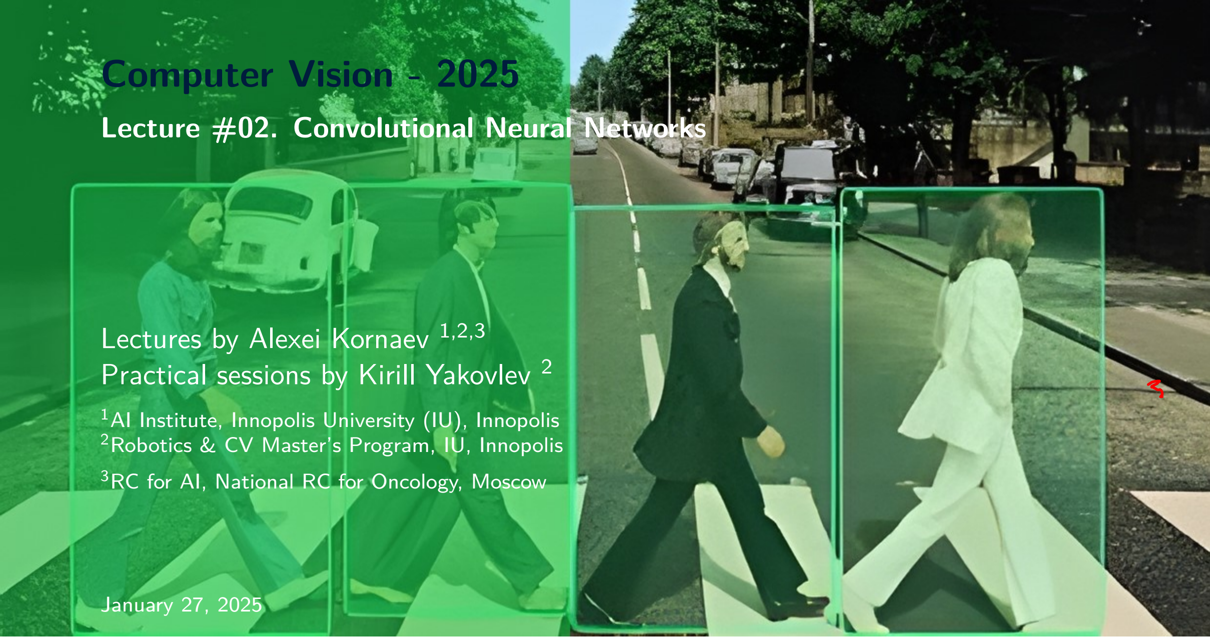


# Computer Vision - 2025

## Lecture #02. Convolutional Neural Networks


Lectures by Alexei Kornaev [1,2,3]

Practical sessions by Kirill Yakovlev [2]

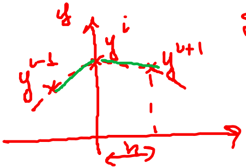
[1]AI Institute, Innopolis University (IU), Innopolis
[2]Robotics & CV Master's Program, IU, Innopolis

[3]RC for AI, National RC for Oncology, Moscow


January 27, 2025

# Agenda

$$\delta = \frac{y^{i+1} - y^{i-1}}{2h}$$




Дано: $y^i, y^{i-1}, y^{i+1}, h$

$$\left.\frac{dy}{dx}\right|_i = 2ax + b = \delta$$

$$y = ax^2 + bx + c$$

Найти в т. $i$ $\frac{dy}{dx} = ?$

$$\boxed{\frac{dy}{dx} \approx \frac{y^{i+1} - y^{i-1}}{2h}} \approx \frac{y^{i} - y^{i-1}}{h} \approx$$

$$\approx \frac{y^{i+1} - y^{i}}{h}$$

# Section 1. Outcomes

# Outcomes



This week's lecture and practice on CNNs is designed to provide a deep understanding of the theoretical concepts and practical skills necessary for working with CNN-based architectures. By the end of this lecture, students will be able to:

1. Understand core concepts: convolutions, filters, feature maps, padding, pooling, and key CNN architectures (e.g., ResNet, DenseNet, ConvNeXt).
2. Apply transfer learning and fine-tuning to adapt pre-trained models for new tasks using tools like PyTorch.
3. Implement CNNs on datasets like MNIST and CIFAR-10, bridging the gap from educational tasks to real-world applications.

Key Takeaway: Transfer learning accelerates solving practical problems by leveraging pre-trained models.

# Section 2. Key Ideas in CNNs

# Physiology of cats



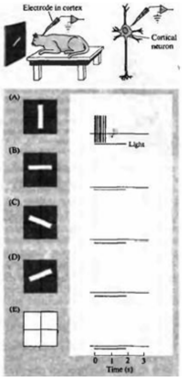



Figure: Responses of the cat's visual cortex cell.

## Implementation in math.

The horizontal derivative kernel approximates $\frac{\partial I}{\partial x}$ using **finite differences**. For images, this translates to computing intensity changes along the $x$-axis. The Sobel kernel for horizontal derivative is:

$\longrightarrow x$

$$\begin{bmatrix}0 & 1 & 1\\ 0 & 1 & 1\\ 0 & 1 & 1\end{bmatrix} \cdot \begin{bmatrix}-1 & 0 & 1\\ -2 & 0 & 2\\ -1 & 0 & 1\end{bmatrix} = \underbrace{+0(-1)+0\cdot 1+1\cdot 1+}_{} = \ldots$$

This kernel computes:

$$\frac{\partial I}{\partial x} \approx (I(x+1, y) - I(x-1, y))$$

and incorporates smoothing to reduce noise. It detects horizontal edges by highlighting intensity changes from left to right.

INNOPOLIS UNIVERSITY

# Hands-on coding: Kernels [CV-2025]



| # | Filter | Description |
|---|---|---|
| 1 | Median Filtering | Replaces each pixel with the median value of its neighborhood. |
| 2 | Gaussian Blur | A Gaussian kernel. Reduces noise but blurs edges. |
| 3 | Bilateral Filtering | Smooths the image while preserving edges by considering both spatial and intensity differences. |
| 4 | Non-Local Means Denoising | Removes noise by averaging similar patches across the image. Computationally intensive. |
| 5 | Histogram Equalization | Redistributes pixel intensities to improve contrast. Simple and effective for enhancing details. |
| 6 | Edge Detection (Canny) | Detects edges using gradient-based methods and hysteresis thresholding. Preserves strong edges. |
| 7 | Thresholding | Converts a grayscale image to binary based on a threshold value. Simple pixel-wise operation. |
| 8 | Morphological Operations (Closing) | Closes small defects in binary images using a structuring element. |
| 9 | Inpainting | Fills missing or corrupted regions in the image using surrounding information. Solves partial differential equations. |

Table: Filters.

# Padding, Pooling, and Striding



p = [1, 1]

s = [2, 2]

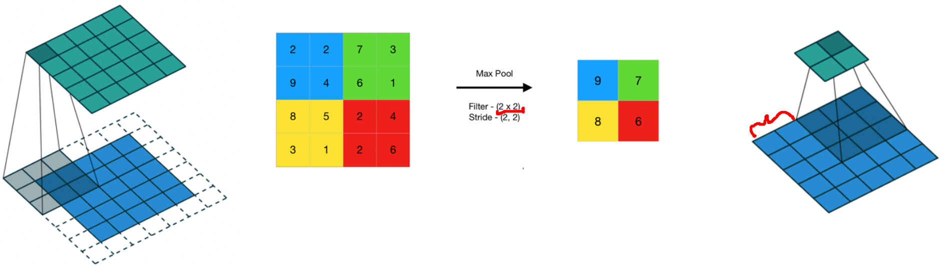



Figure: Padding (left), Pooling (middle), and Striding (right). CNNs by Neurohive.

# 2D Convolution Applied to an Image



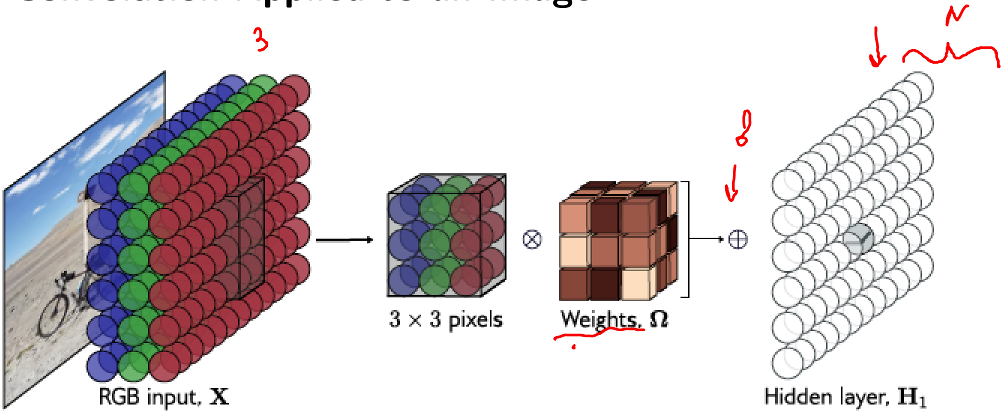



Figure: The image is treated as a 2D input with three channels corresponding to the red, green, and blue components. With a $3 \times 3$ kernel, each pre-activation in the first hidden layer is computed by pointwise multiplying the $3 \times 3 \times 3$ kernel weights with the $3 \times 3$ RGB image patch centered at the same position, summing, and adding the bias. To calculate all the pre-activations in the hidden layer, we âslideâ the kernel over the image in both horizontal and vertical directions. The output is a 2D layer of hidden units. To create multiple output channels, we would repeat this process with multiple kernels, resulting in a 3D tensor of hidden units at hidden layer H1 [Prince, 2023]

# Result of a convolution



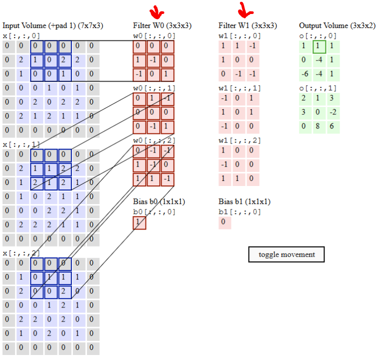



Figure: CS231n: CNNs.

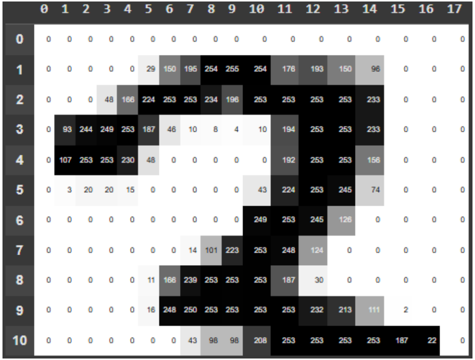
The result of convolving an image

of size $[q, q]$ with a kernel of size $[k, k]$, with padding $[p, p]$ and stride $[s, s]$, is a matrix of size $[r, r]$:

$$r = \frac{q - k + 2p}{s} + 1 = \frac{5 - 3 + 2}{2} + 1 = 3.$$

# Hands-on the book by Howard and Gugger [2020]

CNN utilizes the features of the visual cortex, where simple cells are activated by simple features (such as lines), and complex cells by combinations of activations of simple cells. The CNN is associated with the mathematical operation of convolution for reducing matrix sizes.



| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 29 | 150 | 195 | 254 | 255 | 254 | 176 | 193 | 150 | 96 | 0 | 0 | 0 |
| 2 | 0 | 0 | 0 | 48 | 166 | 224 | 253 | 253 | 234 | 196 | 253 | 253 | 253 | 253 | 233 | 0 | 0 | 0 |
| 3 | 0 | 93 | 244 | 249 | 253 | 187 | 46 | 10 | 8 | 4 | 10 | 194 | 253 | 253 | 233 | 0 | 0 | 0 |
| 4 | 0 | 107 | 253 | 253 | 230 | 48 | 0 | 0 | 0 | 0 | 0 | 192 | 253 | 253 | 156 | 0 | 0 | 0 |
| 5 | 0 | 3 | 20 | 20 | 15 | 0 | 0 | 0 | 0 | 0 | 43 | 224 | 253 | 245 | 74 | 0 | 0 | 0 |
| 6 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 249 | 253 | 245 | 126 | 0 | 0 | 0 | 0 |
| 7 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 14 | 101 | 223 | 253 | 248 | 124 | 0 | 0 | 0 | 0 | 0 |
| 8 | 0 | 0 | 0 | 0 | 0 | 11 | 166 | 239 | 253 | 253 | 253 | 187 | 30 | 0 | 0 | 0 | 0 | 0 |
| 9 | 0 | 0 | 0 | 0 | 0 | 16 | 248 | 250 | 253 | 253 | 253 | 253 | 232 | 213 | 111 | 2 | 0 | 0 |
| 10 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 43 | 98 | 98 | 208 | 253 | 253 | 253 | 253 | 187 | 22 | 0 |

Figure: Check the code: 13 convolutions.ipynb, and 14 resnet.ipynb

# Section 3. CNNs: the Current State and Prospects

# Paper Reading



The following review papers on CNNs [Kiranyaz et al., 2021; Li et al., 2021] are the most cited in 2025:

1. 1D Convolutional Neural Networks and Applications - A Survey
2. A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects

# Section 4. Conclusion

# Conclusion



## Summary:

- Convolutions reduce parameters and improve efficiency by enforcing structured connectivity, enabling deeper models with less overfitting.
- Thoughtful architecture design (e.g., convolutions) outperforms theoretical but impractical fully connected networks in real-world applications.
- Scheduling the learning rate, batch normalization, and activation analysis are key tools for stabilizing training and monitoring progress, paving the way for advanced architectures like residual networks.

# Bibliography



CV-2025. The course repository. https://github.com/Mechanics-Mechatronics-and-Robotics/CV-2025, 2025. Accessed: January 27, 2025.

J. Howard and S. Gugger. **Deep Learning for Coders with Fastai and Pytorch: AI Applications Without a PhD**. O'Reilly Media, Incorporated, 2020. ISBN 9781492045526. URL https://books.google.no/books?id=xd6LxgEACAAJ.

Serkan Kiranyaz, Onur Avci, Osama Abdeljaber, Turker Ince, Moncef Gabbouj, and Daniel J Inman. 1d convolutional neural networks and applications: A survey. **Mechanical systems and signal processing**, 151:107398, 2021.

Zewen Li, Fan Liu, Wenjie Yang, Shouheng Peng, and Jun Zhou. A survey of convolutional neural networks: analysis, applications, and prospects. **IEEE transactions on neural networks and learning systems**, 33(12):6999–7019, 2021.

Simon J.D. Prince. **Understanding Deep Learning**. The MIT Press, 2023. URL http://udlbook.com.