MANGALORE INSTITUTE OF TECHNOLOGY & ENGINEERING
(A Unit of Rajalaxmi Education Trust®, Mangalore)
Autonomous Institute affiliated to VTU, Belagavi, Approved by AICTE, New Delhi
Accredited by NAAC with A+ Grade & ISO 9001:2015 Certified Institution

Subject:Mathematics for Artificial Intelligence

Subject Code:23AIPC/CIPC304

Module 4: Convolution Neural Networks and Computer Vision

Convolution and Cross-Correlation, Convolution from a Systems Design Perspective, Convolution and One-Dimensional Discrete Signals, Convolution and Two-Dimensional Discrete Signals: Filtering Images, Feature Maps, Linear Algebra Notation, Pooling, Convolution Neural Network for Image Classification.

Text book 3 : Ch 5

Introduction to Convolutional Neural Networks (CNNs) and Computer Vision

Convolutional Neural Networks (CNNs) are a specialized class of deep learning models primarily designed to process and analyze data with a *grid-like topology*, such as images (2D grids of pixels) or videos (3D sequences of frames). CNNs have revolutionized the field of Computer Vision (CV) the area of artificial intelligence that enables machines to interpret and understand visual information from the world.

Computer Vision aims to simulate the human visual system using mathematical models, enabling tasks such as image classification, object detection, image segmentation, and facial recognition. Unlike traditional machine learning methods that rely on handcrafted features, CNNs automatically learn hierarchical representations of data, capturing edges, textures, patterns, and abstract concepts directly from raw pixel values.

Mathematical Foundations:

CNNs are built on several **mathematical operations** and principles that underpin their effectiveness in visual understanding:

1. Convolution Operation

The core of CNNs is the *convolution*, a linear mathematical operation defined as:

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(i+m, j+n) K(m,n)$$

where:

I= input image,

K= kernel/filter matrix,

S= feature map (result of convolution).

Note: Convolution detects local features (like edges or corners) by sliding the kernel over the image and computing weighted sums enabling *spatial feature extraction*.

2. Activation Functions

Nonlinear activation functions (e.g., ReLU:) introduce *nonlinearity* into the model, allowing CNNs to learn complex visual patterns that cannot be represented by linear functions alone.

3. Pooling (Subsampling)

Pooling reduces the spatial dimensions of feature maps while retaining essential features:

$$S'(i, j) = \max_{(m,n) \in R(i,j)} S(m, n)$$

This step provides translation invariance and reduces computational complexity.

4. Fully Connected Layers

After convolutional and pooling layers, features are flattened and passed through fully connected layers for classification or regression tasks essentially solving:

$$\hat{y} = softmax(Wx + b)$$

where W and b are learned parameters.

5. Backpropagation and Optimization

CNNs are trained using *gradient descent*:

$$\theta \leftarrow \theta - \eta \frac{\partial L}{\partial \theta}$$

where L is the loss function (e.g., cross-entropy), and η is the learning rate. This process adjusts filters to minimize prediction error, guided by calculus (gradients) and linear algebra (matrix operations).

Applications Related to Mathematics and Deep Learning

1. Image Classification and Recognition

CNNs classify images into categories using mathematical feature extraction and optimization. Example: MNIST(**Modified National Institute of Standards and Technology** dataset) digit recognition a classic benchmark in applying linear algebra and gradient-based learning.

2. Object Detection and Localization

Mathematical models like Region-based CNNs (R-CNN) and YOLO (You Only Look Once) use bounding box regression and probability modeling to detect multiple objects within an image.

3. Image Segmentation

Techniques like U-Net and Fully Convolutional Networks (FCNs) perform pixel-level classification, relying on matrix operations, spatial transformations, and convolutional up sampling.

4. Feature Extraction and Dimensionality Reduction

CNNs act as *nonlinear, learnable transformations* to PCA in mathematics capturing essential visual components while discarding redundancy.

5. Medical and Scientific Imaging

CNNs are applied to mathematical image reconstruction problems e.g., MRI denoising, tomography, and pattern detection in astrophysics and materials science.

6. Deep Learning Integration

CNNs serve as foundational components in hybrid architectures:

- CNN + RNN for video analysis (spatiotemporal modeling).
- CNN + GAN for image generation and super-resolution.
- CNN + Attention/Transformers for visual question answering and multimodal AI.

Conclusion

Convolutional Neural Networks bridge **mathematics and deep learning** through rigorous application of linear algebra, calculus, and optimization theory to visual data processing. They demonstrate how mathematical constructs such as convolutions, gradients, and matrix transformations can model complex perceptual systems, enabling machines to "see" and interpret the world intelligently.

CNNs thus represent one of the most elegant examples of **mathematics in action within AI**, where theory and computation converge to solve real-world visual challenges.

Input Image

–1	0	1
–1	0	1
–1	0	1

Kernel

4	0	4
-4	0	4
0	-4	0

Resulting Feature Map

1. Input Image

- The leftmost part shows a **grayscale image** (for example, a face).
- Mathematically, this image can be represented as a **matrix of pixel intensity values**.

$$I = \begin{bmatrix} 120 & 130 & 125 & \dots \\ 110 & 128 & 135 & \dots \\ \vdots & \vdots & \vdots & \end{bmatrix}$$

- Each value (0–255) indicates how bright that pixel is.
- In CNNs, this matrix is the **input** to the convolutional layer.

2. Kernel (Filter)

•The middle part shows a small **3×3 matrix** called a **kernel** or **filter**.

Example kernel:

$$K = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$

•Each kernel is a small matrix of **learnable parameters** that detect specific features like edges or corners.

•This specific kernel is a **Sobel filter**, which detects **vertical edges**.

3. Convolution Operation

•The kernel slides (or "convolves") over the image.

•At each position, the element-wise product between the kernel and the image patch is computed, and the results are summed up.

Mathematically:

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(i+m, j+n) K(m,n)$$

•The result of this operation is a feature map (the third grid in the image).

4. Resulting Feature Map

The resulting feature map highlights areas where the kernel pattern matches features in the image.

In this example:

Bright values (e.g., 4) indicate strong vertical edges.

Dark or negative values indicate opposite edge directions.

The feature map thus represents how strongly a specific feature (edge) is present at each position of the image.

Convolution and Cross-Correlation

Introduction:

Convolutional Neural Networks (CNNs) are a class of deep learning models specifically designed to analyze spatial and visual data, such as images and videos. Their strength lies in a mathematical operation called convolution, which allows the network to extract features like edges, textures, patterns, and shapes.

Mathematics plays a crucial role here it forms the foundation for how CNNs “see” and learn from images:

Linear algebra handles matrix and tensor operations (for images, filters, and feature maps).

Calculus enables gradient computation for learning filter weights.

Statistics helps in normalization, probability estimation, and feature interpretation.

Thus, CNNs are not just computer algorithms they are mathematical systems that learn through convolutional transformations and optimization.

Definition

Convolution is a mathematical operation that combines two functions to produce a third one, expressing how the shape of one is modified by the other.

In CNNs:

•One function = Input image (or feature map)

•Other function = Kernel (or filter)

The convolution computes a weighted sum of pixel values in the local region of the image.

Mathematical Formula (Continuous Form)

For continuous functions $f(t)$and $g(t)$:

$$(f * g)(t) = \int_{-\infty}^{\infty} f(\tau)\, g(t-\tau)\, d\tau$$

This integrates over all shifts τ, multiplying and summing overlapping parts of $f$and g.

Question 1:

Given the image matrix I and kernel K:

$$I = \begin{bmatrix} 2 & 3 & 1 \\ 4 & 6 & 5 \\ 1 & 2 & 3 \end{bmatrix}, K = \begin{bmatrix} 1 & 0 \\ -1 & 2 \end{bmatrix}$$

Compute the 2D convolution output $S = I * K$(using *valid padding*, stride = 1). Then explain what type of feature (edge/contrast) this kernel detects.

1.Flip kernel K both horizontally and vertically.

2.Slide over I, compute element-wise products and sum.

3.Obtain the output matrix S.

4.Interpret whether kernel enhances vertical, horizontal, or diagonal features.

Question 2 :

Given the image matrix I and kernel K:

$$I = \begin{bmatrix} 3 & 1 & 2 \\ 0 & 1 & 3 \\ 2 & 2 & 1 \end{bmatrix}, K = \begin{bmatrix} 1 & -1 \\ 2 & -2 \end{bmatrix}$$

Compute the 2D convolution output $S = I * K$ using valid padding and stride = 1.

1. Flip kernel K both horizontally and vertically before applying.
2. Slide over I, compute the element-wise product and sum at each position.
3. Obtain the output matrix S.
4. Interpret whether this kernel enhances vertical, horizontal, or diagonal edges.

Question 3 :

Consider the input image $I(4\times4)$ and kernel $K(3\times3)$:

$$I = \begin{bmatrix} 1 & 2 & 3 & 0 \\ 0 & 1 & 2 & 3 \\ 3 & 1 & 0 & 2 \\ 1 & 2 & 3 & 1 \end{bmatrix}, K = \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}$$

Compute:

1. The output feature map for stride = 1 and no padding.
2. The output feature map for stride = 2 and padding = 1 (zero padding).
3. Explain how stride and padding affect the spatial size and information preservation.

Question 4 :

Consider the input image $I(4\times4)$ and kernel $K(3\times3)$:

$$I = \begin{bmatrix} 2 & 0 & 1 & 3 \\ 1 & 2 & 2 & 0 \\ 3 & 1 & 0 & 1 \\ 0 & 2 & 3 & 2 \end{bmatrix}, K = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

1.Compute the output feature map for stride = 1 and no padding.

2.Compute the output feature map for stride = 2 and padding = 1 (zero padding).

3. Explain:

a. How stride changes the output spatial size and information density.

b. How padding preserves information near the borders.

Convolution from a Systems Design Perspective

Introduction:

Convolution in Systems Design

In systems design, convolution is a fundamental operation that describes how a system transforms an input signal into an output signal.

It provides a unified mathematical framework to analyze systems in signal processing, control theory, image processing, and deep learning (CNNs).

At its core:

Convolution expresses how the current output depends on weighted combinations of past and present inputs.

It captures the memory, causality, and response characteristics of a system all crucial in designing Linear Time-Invariant (LTI) systems and Convolutional Neural Networks (CNNs).

Definition:

For a **continuous-time system**, the output $y(t)$of an LTI system is defined as:

$$y(t) = (x * h)(t) = \int_{-\infty}^{\infty} x\,(\tau)h(t-\tau)\,d\tau$$

For a **discrete-time system**, the output is:

$$y[n] = (x * h)[n] = \sum_{k=-\infty}^{\infty} x\,[k]\,h[n-k]$$

Where :

Symbol	Meaning
$x(t)$or $x[n]$	Input signal
$h(t)$or $h[n]$	System's impulse response
$y(t)$or $y[n]$	Output signal
$*$	Convolution operator

Question 1 :

Consider a discrete-time system with input signal

$$x[n] = \{1, 2, 1, 0\}$$

and system impulse response

$$h[n] = \{1, -1, 2\}.$$

Compute the convolution $y[n] = x[n] * h[n]$ and interpret the result from a systems design perspective explain how the impulse response modifies the input signal behavior.

Question 2 :

A linear time-invariant (LTI) system has impulse response

$$h[n] = \{2,\ 0, -1\}$$

and the input signal

$$x[n] = \{3, 1, 2, 1\}.$$

i) Compute the convolution output $y[n] = x[n] * h[n]$

ii) Identify whether the system behaves as a smoothing, edge-detecting, or amplifying system based on the output pattern.

Convolution and One-Dimensional Discrete Signals

Introduction

In many systems especially in signal processing, machine learning, and deep learning (CNNs) data is represented as signals that evolve over space or time. A one-dimensional discrete signal is a sequence of numbers that represents how a quantity changes at discrete time steps or spatial intervals.

For example:

$$x[n] = \{x[0], x[1], x[2], \ldots . x[N-1]\}$$

can represent an audio waveform, a time-series sensor reading, or a 1D feature vector.

When such a signal passes through a linear time-invariant (LTI) system, its transformation is described mathematically using convolution.

Definition :

Convolution is a mathematical operation that expresses the relationship between:

- An input signal $x[n]$,
- A system's impulse response $h[n]$,
- And the output signal $y[n]$.

It is denoted by:

$$y[n] = x[n] * h[n]$$

and defined mathematically as:

$$\boxed{y[n] = \sum_{k=-\infty}^{\infty} x\,[k]\,h[n-k]}$$

For finite discrete signals, the sum is taken over valid indices only.

Question 1:

Given

$$x[n] = \{2,\ 3,\ 1,\ \ 1\}, \quad h[n] = \{2, -1,\ 3\}$$

i) Compute the convolution $y[n] = x[n] * h[n]$.

ii) Interpret the system behavior: does $h[n]$amplify, smooth, or differentiate the signal?

Question 2:

Given:

$$x[n] = \{3,\ 0,\ 1,\ 2,\ 1\}, \quad \mathrm{h[n]} = \{1,\ 0, -1\}$$

i) Perform convolution $y[n] = x[n] * h[n]$.

ii) Analyze how the kernel $h[n]$acts as an edge detector or difference operator.

(iii) Comment on whether the system is causal and time-invariant.

Question 3:

An input feature sequence is:

$$x[n] = \{1,\ 3,\ 2,\ 0,\ 1,\ 2\}$$

and a learnable kernel is:

$$w[n] = \{1,\ 0,\ -1\}$$

(i) Compute the valid convolution output (stride = 1, no padding).

(ii) Interpret which type of local feature this filter detects (increasing/decreasing pattern).

(iii) From a mathematical systems design view, relate this operation to the difference between adjacent

Question 4:

An input feature sequence is:

$$x[n] = \{2,\ 4,\ 3,\ 1,\ 0,\ 2\}$$

and a learnable kernel is:

$$w[n] = \{-1,\ 0,\ 1\}$$

i) Compute the valid convolution output with stride = 1 and no padding.

ii) Interpret which type of local pattern this filter detects (e.g., rising edge, falling edge, flat-to-change transition).

iii) From a mathematical systems-design perspective, explain how this convolution corresponds to a first-order difference operator and how it extracts local variations in the input signal.

Introduction to Convolution and Two-Dimensional Discrete Signals

In digital signal processing and image analysis, **two-dimensional (2D) discrete signals** play a fundamental role. A 2D discrete signal is a function defined on a discrete grid of spatial coordinates typically representing an image or spatial data. Just as one-dimensional signals vary with time, 2D signals vary with two spatial dimensions (e.g., x and y).

Convolution is one of the most important operations applied to such signals. It describes how an input signal interacts with a system or filter. In the 2D case, convolution is widely used for image processing tasks such as blurring, sharpening, edge detection, and feature extraction in computer vision and deep learning (e.g., convolutional neural networks).

Definition

Two-Dimensional Discrete Signal

A 2D discrete signal is represented as:

$$x[m, n]$$

were

- m and n are integer spatial indices (rows and columns),
- $x[m, n]$ gives the signal's (or image's) intensity or value at position (m, n).

If the signal represents an image, $x[m, n]$ corresponds to the pixel intensity at that location.

Two-Dimensional Discrete Convolution

The 2D convolution of two discrete signals $x[m, n]$ input signal or image and $h[m, n]$ impulse response or kernel is defined as:

$$y[m, n] = (x * h)[m, n] = \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} x[k, l] h[m-k, n-l]$$

were

- $h[m, n]$ is the filter kernel or mask,
- $y[m, n]$ is the output (filtered image or processed signal).

Key Points

•Convolution combines two signals to produce a third signal that shows how one modifies the other.

•In image processing, $h[m, n]$ often has small dimensions (e.g., 3×3, 5×5) and defines an operation such as blurring, sharpening, or edge enhancement.

•Convolution respects shift invariance the system response depends only on relative position, not absolute coordinates.

Filtering of Images

1. Introduction

In image processing, filtering is a fundamental operation used to modify or enhance an image by emphasizing or suppressing specific features. It is performed by applying a *filter kernel* (or *mask*) over the image using convolution. Filtering enables tasks such as noise reduction, edge detection, blurring, sharpening, and feature extraction all essential for both human visual interpretation and computer vision algorithms.

The operation involves computing a weighted sum of pixel values in a local neighborhood around each pixel. The weights are determined by the filter kernel.

2. Definition

Let an image be represented as a two-dimensional discrete signal $f(x, y)$ and a filter mask (kernel) be $h(x, y)$ .The filtered image $g(x, y)$ is obtained using 2D discrete convolution:

$$g(x, y) = (f * h)(x, y) = \sum_{m=-a}^{a} \sum_{n=-b}^{b} f(x - m,\ y - n) h(m,\ n)$$

were

- $f(x, y)$:input image,
- $h(m, n)$:filter kernel (mask),
- $g(x, y)$:output (filtered) image,
- a, b: half-width and half-height of the mask.

Problem 1:

Given the 3×3 image patch

$$I = \begin{bmatrix} 120 & 140 & 180 \\ 130 & 160 & 210 \\ 190 & 230 & 250 \end{bmatrix}$$

and the same 3×3 averaging filter $H = \frac{1}{9}\mathbf{1}_{3\times3}$.

1. Compute the filtered center pixel (convolution, no padding).
2. Compute the filtered value at the top-left pixel if you apply zero-padding (i.e., treat pixels outside as 0) and use full convolution for that location.
3. If additive Gaussian noise (mean 0, variance 25) is present on each pixel independently, compute the output noise variance after filtering.
4. Compute the reduction in noise power in dB(decibel) produced by the filter.
5. Compute the difference between the original center pixel and the filtered center pixel (quantify the blurring effect numerically).

Problem 2:

Given the 3×3 image patch:

$$I = \begin{bmatrix} 90 & 110 & 130 \\ 100 & 140 & 160 \\ 120 & 150 & 180 \end{bmatrix}$$

and a 3×3 averaging filter

$$H = \frac{1}{9}\mathbf{1}_{3\times3}$$

1. Compute the filtered center pixel using valid convolution.
2. Compute the filtered value at the top-right pixel assuming zero-padding.
3. If Gaussian noise with variance 16 is added independently to each pixel, compute the output noise variance after filtering.
4. Compute the reduction in noise power in dB.
5. Compute the change between the original center pixel and the filtered center pixel and interpret the blurring.

Problem 3:

Given a 3×3 image segment:

$$I = \begin{bmatrix} 10 & 20 & 30 \\ 20 & 30 & 40 \\ 30 & 40 & 50 \end{bmatrix}$$

The Laplacian filter is:

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

a) Compute the filtered value at the center pixel using 2D convolution.
b) Add the Laplacian-filtered result back to the original image to produce a sharpened output.
c) Discuss how this operation enhances the contrast at edges.

Problem 4:

Given a 3×3 image segment:

$$I = \begin{bmatrix} 15 & 25 & 35 \\ 25 & 35 & 45 \\ 35 & 45 & 55 \end{bmatrix}$$

The Laplacian filter is:

$$H = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

a) Compute the filtered (Laplacian) value at the center pixel using 2D convolution.

b) Add the Laplacian-filtered value back to the original center pixel to obtain the sharpened output at the center.

c) Explain how this Laplacian-based sharpening improves edge contrast in the image.

Feature Maps

Introduction :

In image processing and convolutional neural networks (CNNs), a *feature map* represents how different patterns or features (like edges, corners, textures, or shapes) are detected in an input image after applying a filter (kernel) during the convolution operation.

Each convolution layer in a CNN transforms the input image into several feature maps, where each map highlights specific visual characteristics learned by that filter.

As the network goes deeper:

- Early layers detect simple features (edges, colors, orientations).
- Later layers capture complex patterns (faces, objects, textures).

Thus, feature maps act as the internal representation of how the model “sees” and understands an image, extracting essential information while reducing irrelevant details.

Definition

A **Feature Map** is a two-dimensional matrix (or sometimes 3D tensor including depth) that contains the output values obtained after applying a convolution filter followed by a non-linear activation function on an input image or previous layer.

Mathematically,

$$F(x, y) = f((I * W)(x, y) + b)$$

where

I: input image or previous layer's feature map,

W: convolution kernel (filter weights),

b: bias term,

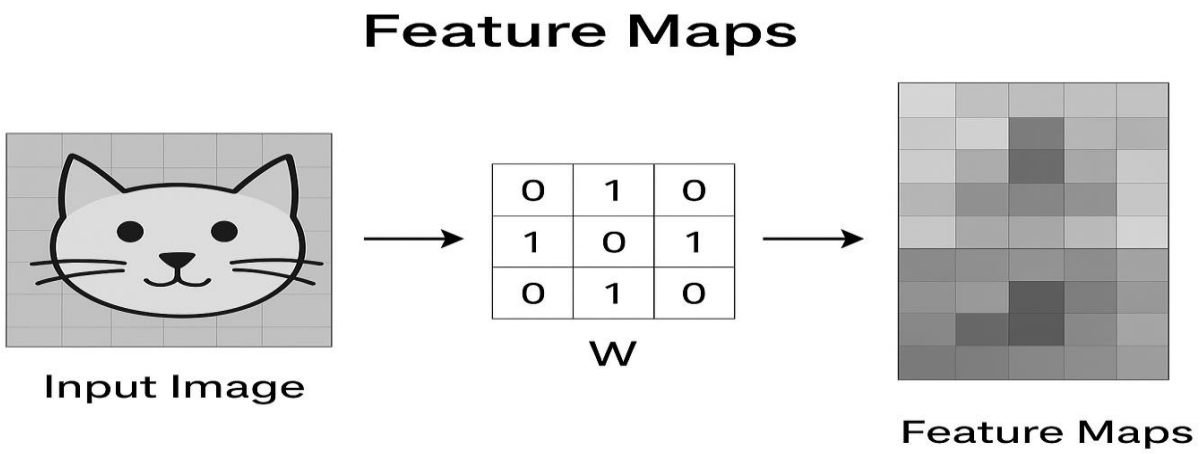
$*$: convolution operation,

f: activation function (e.g., ReLU),

$F(x, y)$:resulting feature map value at position (x, y).

Key Points

- Each filter in a CNN produces one feature map.
- Stacking multiple filters → multiple feature maps per layer.
- Feature maps encode spatial hierarchy local to global visual features.
- They reduce image complexity while retaining important patterns for classification or recognition.



Note: The Sobel operator (or Sobel filter) is a gradient-based edge detection method used to find edges in images, regions where intensity (brightness) changes sharply.

Question-1 :

Consider a 3×3 input image patch:

$$I = \begin{bmatrix} 5 & 6 & 7 \\ 4 & 3 & 2 \\ 1 & 0 & 1 \end{bmatrix}$$

and two convolution kernels:

$$W_1 = \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}, W_2 = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

a) Compute the output (feature map value) for both filters at the center pixel.

b) Identify which filter detects vertical edges and which detects horizontal edges.

c) Explain how these two feature maps would be stacked in a CNN layer.

Question-2 :

Consider a 3×3 input image patch:

$$I = \begin{bmatrix} 8 & 7 & 6 \\ 5 & 4 & 3 \\ 2 & 1 & 0 \end{bmatrix}$$

and two convolution kernels:

$$W_1 = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}, \qquad W_2 = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$$

a) Compute the feature map output at the center pixel for both filters using valid 2D convolution.

b) Identify which kernel detects horizontal edges and which detects vertical edges.

c) Explain how these two feature maps are combined in a multi-channel CNN layer.

Question-3:

Consider a 3×3 image patch:

$$I = \begin{bmatrix} 2 & 3 & 4 \\ 1 & 2 & 3 \\ 0 & 1 & 2 \end{bmatrix}$$

and two convolution kernels (Sobel filters):

$$W_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, \qquad W_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

a) Compute the convolution output (feature map value) at the center pixel for both W_x and W_y.

b) Determine which filter detects horizontal edges and which detects vertical edges.

c) Calculate the magnitude of the gradient at the center pixel using

$$G = \sqrt{\left(G_x\right)^2 + \left(G_y\right)^2}$$

and interpret what it represents in terms of image features.

Question-4:

A feature map has pixel activations:

$$F = \{2, 4, 6, 8, 10\}$$

Batch normalization uses parameters:

$$\mu = 6, \qquad \sigma^2 = 8, \qquad \gamma = 1.5, \qquad \beta = 0.5$$

a) Normalize each feature using

$$F_{norm} = \gamma \cdot \frac{(F-\mu)}{\sqrt{\sigma^2+\epsilon}} + \beta, \text{ where } \epsilon = 10^{-8}.$$

b) Compute the new normalized feature values.

c) Explain the benefit of normalization on feature map stability during CNN training.

Linear Algebra Notation

Introduction:

Linear algebra is the branch of mathematics that studies vectors, matrices, and linear transformations the foundational language of modern data science, machine learning, and deep learning.

To work effectively with high-dimensional data and algorithms, it's crucial to understand the notation system that represents quantities like:

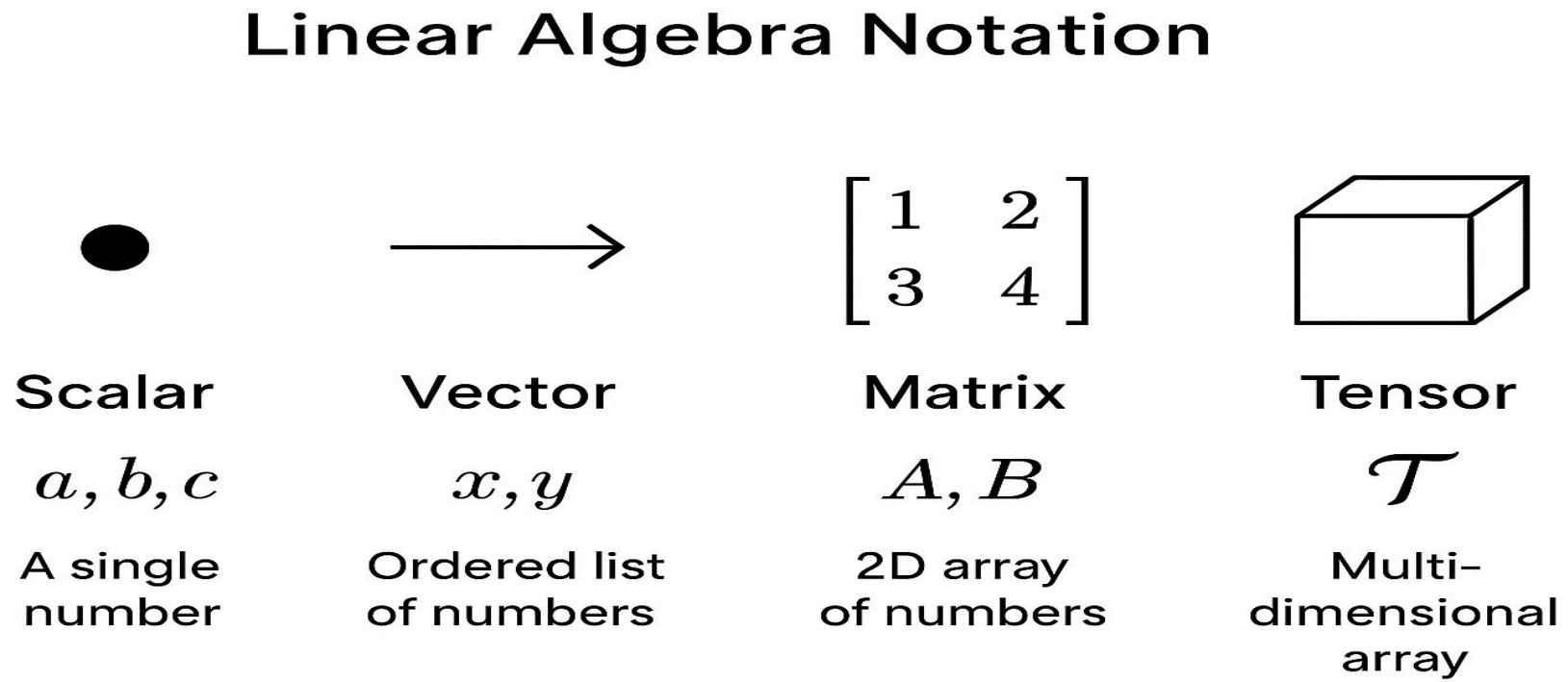
- Scalars (single numbers),
- Vectors (1D arrays of numbers),
- Matrices (2D arrays), and
- Tensors (multi-dimensional generalizations).

These notations allow us to describe operations such as dot products, matrix multiplication, eigen decomposition, and projections concisely and universally.

In AI/ML, linear algebra notation provides the mathematical shorthand for expressing models from linear regression equations to deep neural network transformations.

Defination:

Linear Algebra Notation refers to the symbolic system used to represent and manipulate linear mathematical objects (like vectors and matrices) and operations (like addition, multiplication, and transformation) in a compact and consistent way.



Common Notations in Linear Algebra

Concept	Symbol	Example	Description
Scalar	Lowercase italic letter	a, b, c	A single number (real or complex).
Vector	Bold lowercase letter	$\mathbf{x}, \mathbf{y}$	Ordered list of numbers; element x_i denotes the *i-th component*.
Matrix	Bold uppercase letter	$\mathbf{A}, \mathbf{B}$	2D array of numbers with rows and columns; element A_{ij}.
Tensor	Bold calligraphic letter	$\mathcal{T}$	Multi-dimensional array (generalization of matrix).
Transpose	Superscript T	$\mathbf{A}^T$	Flips rows and columns of a matrix.
Inverse	Superscript -1	$\mathbf{A}^{-1}$	Matrix that satisfies $\mathbf{A}\mathbf{A}^{-1} = \mathbf{I}$.
Identity Matrix	$\mathbf{I}_n$	$n \times n$ matrix with 1's on the diagonal.	
Dot Product	$\mathbf{x} \cdot \mathbf{y}$ or $\mathbf{x}^T\mathbf{y}$	Scalar representing the product of two vectors.	
Matrix Multiplication	$\mathbf{AB}$	Combines rows of **A** with columns of **B**.	

Pooling

Introduction

In Convolutional Neural Networks (CNNs), Pooling is a key operation used after convolutional layers to reduce the spatial dimensions (height and width) of feature maps while retaining the most important information.

Pooling helps to:

- Decrease computational complexity,
- Reduce the number of parameters,
- Prevent overfitting, and
- Make the network more translation invariant, meaning small shifts or distortions in the input image do not drastically change the output.

By summarizing local regions of the feature map, pooling layers preserve essential features (like edges, corners, and textures) while discarding redundant spatial information.

Definition

Pooling is a downsampling operation that partitions a feature map into non-overlapping or overlapping regions (called pooling windows) and applies a statistical function (such as maximum or average) to summarize the values within each region.

Mathematically, for a feature map F and pooling window P,

$$Y(i, j) = pool\big(F(m, n)\big), \qquad were\ (m, n) \in P_{ij}$$

where

- $Y(i, j)$ →output pooled feature value at location (i, j),
- $F(m, n)$ →original feature values in the local pooling window P_{ij},
- pool → function such as max, average, or min.

Key Benefits

- Reduces overfitting by lowering feature map size
- Increases computational efficiency
- Maintains position invariance (robustness to small shifts)
- Simplifies higher-layer learning

Question-1:

Consider a single-channel feature map:

$$F = \begin{bmatrix} 1 & 5 & 2 & 8 \\ 4 & 7 & 3 & 6 \\ 2 & 4 & 8 & 10 \\ 0 & 3 & 6 & 9 \end{bmatrix}$$

Apply 2×2 max pooling and 2×2 average pooling (stride = 2) in parallel, then compute their element-wise average (also called *mixed pooling*).

a) Compute the max-pooled output.

b) Compute the average-pooled output (show numeric division).

c) Combine both by averaging their corresponding elements.

d) Explain how mixed pooling balances sensitivity and stability of feature detection.

Question-2:

A single-channel feature map is given:

$$F = \begin{bmatrix} 3 & 1 & 4 & 2 \\ 6 & 5 & 8 & 7 \\ 2 & 9 & 1 & 3 \\ 4 & 6 & 5 & 8 \end{bmatrix}$$

Apply 2×2 max pooling and 2×2 average pooling (stride = 2) in parallel, then compute their element-wise average (mixed pooling).

a) Compute the max-pooled output.
b) Compute the average-pooled output (show numeric division).
c) Combine both outputs by averaging corresponding elements to obtain mixed pooling.
d) Explain how mixed pooling affects robustness and feature emphasis in CNNs.

Question-3:

Given a 3-channel feature map (3×3×3):

$$F^{(1)} = \begin{bmatrix} 1 & 2 & 1 \\ 3 & 4 & 3 \\ 2 & 1 & 2 \end{bmatrix}, F^{(2)} = \begin{bmatrix} 2 & 3 & 4 \\ 1 & 2 & 3 \\ 0 & 1 & 2 \end{bmatrix}, F^{(3)} = \begin{bmatrix} 4 & 4 & 4 \\ 3 & 3 & 3 \\ 2 & 2 & 2 \end{bmatrix}$$

Apply Global Max Pooling (GMP) to each channel separately.

a) Find the maximum activation from each channel.
b) Form the pooled vector (1×1×3).
c) Explain how this vector is used when connecting to a fully connected layer.
d) Compare Global Max Pooling to Global Average Pooling which is more sensitive to noise, and why?

Question-4

You are given a 3-channel feature map (each 4×4):

$$F^{(1)} = \begin{bmatrix} 1 & 3 & 2 & 4 \\ 0 & 5 & 1 & 2 \\ 3 & 2 & 4 & 1 \\ 1 & 0 & 2 & 3 \end{bmatrix} \quad F^{(2)} = \begin{bmatrix} 2 & 1 & 0 & 3 \\ 4 & 6 & 2 & 1 \\ 3 & 2 & 5 & 4 \\ 0 & 1 & 2 & 3 \end{bmatrix} \quad F^{(3)} = \begin{bmatrix} 7 & 6 & 7 & 5 \\ 4 & 4 & 3 & 3 \\ 2 & 2 & 2 & 2 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

Apply Global Max Pooling (GMP) to each channel.

a) Find the maximum activation from each channel.

b) Form the pooled vector (1×1×3).

c) Explain how GMP reduces parameters before the classifier.

d) Compare GMP and GAP which is better for detecting strong localized features?

Convolution Neural Network for Image Classification

Introduction

Image classification is one of the most fundamental tasks in computer vision, aiming to automatically assign a label or category to a given image based on its visual content. Traditional machine learning methods relied heavily on manual feature extraction, which limited their ability to generalize across diverse datasets.

Convolutional Neural Networks (CNNs) revolutionized image classification by enabling automatic learning of hierarchical visual features directly from raw pixel data. A CNN processes an image through multiple layers each layer extracting increasingly complex features, from edges and corners in the first layers to textures, shapes, and objects in deeper layers.

Through this hierarchical feature learning, CNNs can accurately distinguish between classes, even under variations in lighting, position, or orientation of objects. They have become the core architecture in modern image classification systems used in medical imaging, autonomous driving, facial recognition, and visual search.

Definition

A Convolutional Neural Network (CNN) for Image Classification is a deep learning architecture designed to automatically learn and extract discriminative features from images through convolutional operations and use these learned features to predict the class label of the input image.

Formally, the CNN learns a mapping

$$f: \mathbb{R}^{H \times W \times C} \rightarrow \{1, 2, . , .. K\}$$

where H, W, C represent the height, width, and channels of the input image, and K denotes the number of classes.

Through a sequence of convolution, activation, pooling, and fully connected layers, the network transforms the image into a compact feature vector that is finally passed to a softmax classifier to produce the class probabilities.

Question-1:

An image of size 64 × 64 × 3 (height × width × channels) passes through a CNN layer with:

- 32 filters of size 3 × 3,
- stride = 1,
- padding = 1.

Then it goes through a 2×2 max pooling layer with stride = 2.

a) Calculate the output size after the convolution layer.
b) Calculate the output size after the pooling layer.
c) Determine the total number of trainable parameters in the convolution layer.
d) Explain how this dimensional reduction helps the network generalize in image classification.

Question-2:

An RGB image of size 128 × 128 × 3 is passed through a CNN layer with:

- 64 filters of size 5 × 5,
- stride = 2,
- padding = 2.

The output then passes through a 3×3 max pooling layer with stride = 3.

a) Compute the output size (height × width × depth) after the convolution layer.

By using Output $Size = \frac{(W-F+2P)}{S} + 1$

b) Compute the output size after the pooling layer.

c) Find the total number of trainable parameters in the convolution layer, including bias terms.

d) Explain numerically how stride and padding together influence the feature map size and spatial resolution retention in CNN-based image classification.

Question-3:

The final layer of a CNN outputs the following scores (logits) for three classes (cat, dog, car):

$$z = [2.0, 1.0, 0.1]$$

a) Compute the softmax probabilities for each class (round to 3 decimals).

b) If the true label is “cat,” compute the cross-entropy loss.

c) Explain the significance of softmax in image classification output.

d) Discuss how the network updates weights based on the computed loss.

Question-4:

A CNN trained for image classification contains the following layers:

1. Conv1: 3×3 filter, 32 channels
2. Conv2: 3×3 filter, 64 channels
3. Pooling: 2×2, stride = 2
4. Flatten + Dense: 128 neurons
5. Output: 10-class softmax layer

a) If the input image is 32×32×3, compute the feature map dimensions after Conv1, Conv2, and pooling (assume stride 1 and padding 1).

b) Calculate the number of parameters in Conv1 and Conv2.

c) Explain how convolutional layers act as feature extractors, while dense layers act as classifiers.

d) Suggest one practical application of this CNN in real-world image classification.


Thank You