Deep Learning

Use mnemonic devices, visualization, and association to remember information. The Method of Loci (associating information with locations) can be particularly effective.

Try to learn one new word daily. Use flashcards or apps like Anki to make the process interactive.

Visual language such as charts and plots is ubiquitous in the human world.

Intersection over Union (IoU), Dense, Affine, Kernel, Gaussian blur, Bilateral filter, Image pyramid, Downsampling, Kernel, Alpha, sigma,

dense → closely compacted in substance. **dense layer** (also known as a fully connected layer)

- **input**: The data fed into the layer.
- **kernel**: A matrix of weights that determines how much influence each input has on the output.
- **dot**: Represents the dot product operation between the input and the kernel.
- **bias**: A vector added to the output to optimize the model.
- **activation**: An activation function applied to introduce non-linearity into the mode

So multiplication varies the most, so we apply addition as a bias

affine → allowing for or preserving parallel relationships.

Viso Suite - End-to-End Computer Vision Infrastructure - viso.ai → Use it for creating projects

An **alpha channel** is a component in digital images and video files that defines the transparency of each pixel. It is typically represented as an 8-bit grayscale channel, where:

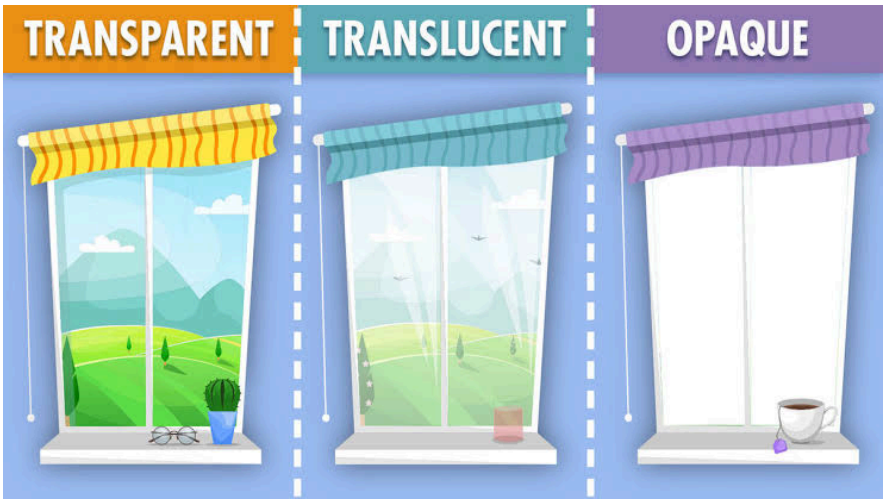
- **White** indicates full opacity (100% visible).
- **Black** signifies full transparency (0% visible).
- **Gray** represents varying levels of partial transparency.

This channel works alongside the standard RGB (Red, Green, Blue) color channels to create a complete image.

The **alpha channel** is a component of an image that represents the level of transparency or opacity for each pixel.

https://plantcv.readthedocs.io/en/stable/transform_affine_color_correction/

- **RGBA**: In the RGBA color model, which stands for **Red**, **Green**, **Blue**, and **Alpha**, the alpha channel is the fourth channel. While the RGB channels define the color, the **A** (alpha) defines how transparent or opaque a pixel is:
  - An alpha value of **0** means completely transparent.
  - An alpha value of **255** (or 100%) means fully opaque.



This allows you to blend images with different backgrounds, creating effects like soft edges, shadows, or overlaying images without solid backgrounds.

Reading the images

- IMREAD_COLOR loads the image in the BGR 8-bit format. This is the **default** that is used here.
- IMREAD_UNCHANGED loads the image as is (including the alpha channel if present)
- IMREAD_GRAYSCALE loads the image as an intensity one

Mouse event can be anything related to mouse like left-button down, left-button up, left-button double-click etc. It gives us the coordinates (x,y) for every mouse event. With this event and location, we can do whatever we like.

Bilateral Filter: A bilateral filter is a ***non-linear, edge-preserving, and noise-reducing smoothing filter*** for images. It is used for blurring it is same as gaussian blur.

When to use:
Use it when you want to smooth the image without blurring edges. It's often used in denoising, especially for images with sharp edges or details that should remain intact.

Here's an analogy: Imagine smoothing a paint job. A Gaussian blur is like sanding the entire surface uniformly, softening edges. A Bilateral filter is like carefully sanding only the flat areas, while protecting the crisp edges of any trim

The basic idea underlying bilateral filtering is to do in the range of an image what traditional filters do in its domain. Two pixels can be *close* to one another, that is, occupy nearby spatial location, or they can be *similar* to one another, that is, have nearby values, possibly in a perceptually meaningful fashion.

A higher sigma value causes the filter to have a larger spread, meaning the image will be blurred more since more pixels are averaged together. Conversely, a **lower sigma value** results in less blurring.

A
gradient refers to the rate of change or slope of a quantity with respect to another.

> **Gradient Descent** is an optimization algorithm used in machine learning and deep learning to minimize a function, typically the **loss function** (which measures how far off a model's predictions are from the actual outcomes). The goal of gradient descent is to adjust the model's parameters (like weights in a neural network) in such a way that the loss function is minimized, which means the model gets better at making predictions.

If d=9, the filter will consider a 9×9 square area around each pixel.

sigmaColor (float): This parameter controls how much influence the color difference between the central pixel and its neighbors has on the smoothing.

- **High values of** `sigmaColor` allow a greater difference in color intensity between the central pixel and its neighbors to still have a smoothing effect.
- **Low values of** `sigmaColor` make the filter more sensitive to color differences, meaning that pixels with a large difference in color will be less affected (i.e., edges are preserved better).
- In this case, `sigmaColor=75` means that the filter will consider color differences up to a certain threshold when smoothing, which results in moderate edge preservation and color smoothing.

- **Low sigmaSpace** = More localized smoothing, better edge preservation.
- **High sigmaSpace** = Broader smoothing, less edge definition, smoother image.

`**` is used to unpack dictionaries into keyword arguments, allowing the function to receive individual key-value pairs.

convolve:- combine (one function or series) with another by forming their convolution

kernel refers to a small matrix or filter that is applied to an image to perform various operations like blurring, sharpening, edge detection, and more. The kernel is a grid (usually 3×3, 5×5, etc.) of numbers that is moved (convolved) over an image to perform a transformation.

A kernel is usually a
small square or rectangular matrix.

The
Sobel kernel is commonly used for edge detection, calculating the gradient of the image in the horizontal or vertical direction.

Steps to Calculate Standard Deviation:

1. **Find the mean** (average) of the dataset.
2. **Subtract the mean** from each data point and **square** the result.
3. **Find the average** of those squared differences (this is called the variance).
4. **Take the square root** of the variance to get the standard deviation.

Example:

Consider the following dataset:

5,7,3,9,65, 7, 3, 9, 65,7,3,9,6

1. **Calculate the mean**:μ=55+7+3+9+6=530=6

   μ=5+7+3+9+65=305=6\mu = \frac{5 + 7 + 3 + 9 + 6}{5} = \frac{30}{5} = 6

2. **Subtract the mean from each data point** and square the result:
    - (5−6)2=1(5 - 6)^2 = 1(5−6)2=1
    - (7−6)2=1(7 - 6)^2 = 1(7−6)2=1
    - (3−6)2=9(3 - 6)^2 = 9(3−6)2=9
    - (9−6)2=9(9 - 6)^2 = 9(9−6)2=9
    - (6−6)2=0(6 - 6)^2 = 0(6−6)2=0
3. **Calculate the variance** (average of squared differences):Variance=51+1+9+9+0=520=4

    Variance=1+1+9+9+05=205=4\text{Variance} = \frac{1 + 1 + 9 + 9 + 0}{5} = \frac{20}{5} = 4
4. **Take the square root** of the variance to get the standard deviation:σ=4=2

    σ=4=2\sigma = \sqrt{4} = 2

Interpreting Standard Deviation:

- **Small Standard Deviation**: If the standard deviation is small, it means the values in the dataset are **close to the mean** and there is **less variability**.
- **Large Standard Deviation**: If the standard deviation is large, it means the values in the dataset are **spread out** over a wider range, indicating **greater variability**.
- **Variance** is the squared measure of data spread.
- **Standard deviation** is the square root of variance, making it more directly interpretable in real-world terms.

In statistics: Standard deviation is used to quantify the amount of variation or dispersion in a dataset.

In image processing: Standard deviation is used to measure the **contrast** of an image or the degree of variation in pixel values.

Intensity similarity: Unlike Gaussian blur, bilateral filtering also considers **how similar the pixel intensities are**. Pixels that are **more similar** in intensity to the center pixel will have a greater influence.

Key Differences:

1. **Gaussian Blur**:
    - **All pixels in the kernel influence each pixel**. The kernel is applied uniformly, and spatial distance is the main factor.
    - It smooths **based purely on distance** from the center pixel (spatial consideration only).
2. **Bilateral Filter**:
    - Only **nearby pixels with similar color/intensity** will influence the center pixel, regardless of spatial distance.
    - It **considers both spatial proximity** and **intensity similarity** to preserve edges and details better than Gaussian blur.
    - The filter **preserves edges** better since pixels with significantly different intensities won't blur together.

Gaussian blur is better suited for **backgrounds** or **landscapes** where edge preservation isn't as important.

1. **Uniform smoothing**

Bilateral filter

is particularly effective for images of faces (or portraits) because:

1. **Preserves edges**

- **Bilateral Filter**: Best for images where you need to **preserve edges** (like faces) while smoothing areas of similar color (like skin), making it ideal for portrait photography or skin smoothing.

- **Gaussian Blur**: Best for **blurring uniform regions** (like backgrounds or places) where you want a smooth transition and don't need to preserve fine details, making it great for backgrounds or softening elements in landscapes.

In
image processing, a **pyramid** is a technique used to represent an image at multiple resolutions. It is often used in applications where images are scaled, down-sampled, or processed at different levels of detail.

In pyramid image processing,
downsampling refers to the process of reducing the resolution of an image by decreasing the number of pixels.

This is typically done to create different levels or layers in an image pyramid, where each layer represents the image at progressively lower resolutions

How Downsampling Works:

- **Resolution Reduction**: In downsampling, you reduce the size of the image by removing pixels or averaging groups of pixels. For example, you might take every second pixel from an image or average a group of pixels into a single pixel.
- **Purpose**: **The goal is to make the image simpler and faster to process by reducing the data size**, while still maintaining important structural features. Downsampling helps in multi-scale image processing, where you need to analyze the image at different resolutions (for tasks like object detection, image matching, or texture analysis).

Downsampling allows you to efficiently analyze the image at a lower resolution to capture broader features and avoid processing the entire high-resolution image every time.

An **image pyramid** is essentially a series of images (or layers) where each successive image is a **downsampled version** of the original image, typically reduced in size by a factor of 2. It allows processing at multiple resolutions, which can be useful in tasks such as **object detection**, **image matching**, and **image blending**.

Object Detection:

- In applications like **face detection** or **object recognition**, **pyramids** allow the system to look at the object at different scales, helping detect the object at multiple sizes (small or large).
- Example: In **Haar cascades**, an image pyramid is used to detect faces at different scales by processing the image at multiple resolutions.

Interpolation:- the insertion of something of a different nature into something else, Add something between that.

Cons of Downsampling:

1. **Loss of Detail**:
2. **Artifacts Introduction**:

`cv2.buildPyramid()` is useful for creating multi-resolution representations of an image, primarily a **Gaussian pyramid**.

Deep Learning-Based Approaches: Modern **CNNs** (Convolutional Neural Networks) can handle multiple scales of features through their layers, meaning you might not need to explicitly use pyramids.

Backpropagation is a technique to improve the performance of the network. It backpropagates the error and updates the weights to reduce the error.

Erosion:
Erosion is a morphological operation in image processing, often used to remove noise or shrink the boundaries of objects in a binary image.

> It works by "eroding" or "shrinking" the white regions (foreground) in a binary image, making them smaller. Essentially, erosion removes pixels from the boundary of objects, and the amount of erosion depends on the shape of the structuring element used.

Erosion makes objects smaller, and removes small noise by shrinking the foreground.

Dilation is the opposite of erosion: it makes the white regions (foreground) expand and grow, and it adds pixels to the boundaries of objects.

https://sites.google.com/site/cvonlinewiki/home/generic-computer-vision-methods

Transformations

OpenCV provides two transformation functions, **cv.warpAffine** and **cv.warpPerspective**, with which you can perform all kinds of transformations. **cv.warpAffine** takes a 2×3 transformation matrix while **cv.warpPerspective** takes a 3×3 transformation matrix as input.

Scaling is just resizing of the image. OpenCV comes with a function **cv.resize()** for this purpose. The size of the image can be specified manually, or you can specify the scaling factor. Different interpolation methods are used. Preferable interpolation methods are **cv.INTER_AREA** for shrinking

and **cv.INTER_CUBIC** (slow) & **cv.INTER_LINEAR** for zooming. By default, the interpolation method **cv.INTER_LINEAR** is used for all resizing purposes.

How is it different from other interpolation methods?

1. `INTER_NEAREST`: This is the simplest interpolation method, where the nearest neighbor pixel is selected. It can result in a jagged or pixelated look, especially when enlarging the image.
2. `INTER_LINEAR`: This method uses linear interpolation (a weighted average of the nearest 2 pixels in each direction) and works well for most images. It's faster than cubic interpolation but doesn't produce as smooth results when enlarging images.
3. `INTER_CUBIC`: Uses cubic interpolation, which considers the 4 nearest pixels in each direction (along both the x and y axes) and fits a cubic polynomial. This results in smoother and more visually appealing results when scaling up an image.

Mathematical Concept:

The general equation for cubic interpolation is based on a cubic function (third-degree polynomial):

f(x)=ax3+bx2+cx+df(x) = ax^3 + bx^2 + cx + d

Where
`a`, `b`, `c`, and `d` are coefficients determined by the surrounding pixels. The goal is to find the coefficients of the cubic polynomial that best fit the values of the 16 neighboring pixels, ensuring the pixel values in the resized image are smooth and continuous.

Transformation

Affine refers to a type of transformation that preserves certain geometric

properties such as straight lines and parallelism, but not necessarily distances or angles.