

Computer Vision, and Digital Image Processing

Prepared by Dr. Mohamed Mostafa

AI 612

Lecture 10



Feature Extraction, and Object Tracking

Lecture 10

AI 612

Prepared by
Dr. Mohamed Mostafa

Chapters

11

Resources for this
lecture

Feature Extraction, and Object Tracking



Chapter 11:

Gonzalez, Rafael C. *Digital image processing*.
Pearson education, India, 2009.

Image Features

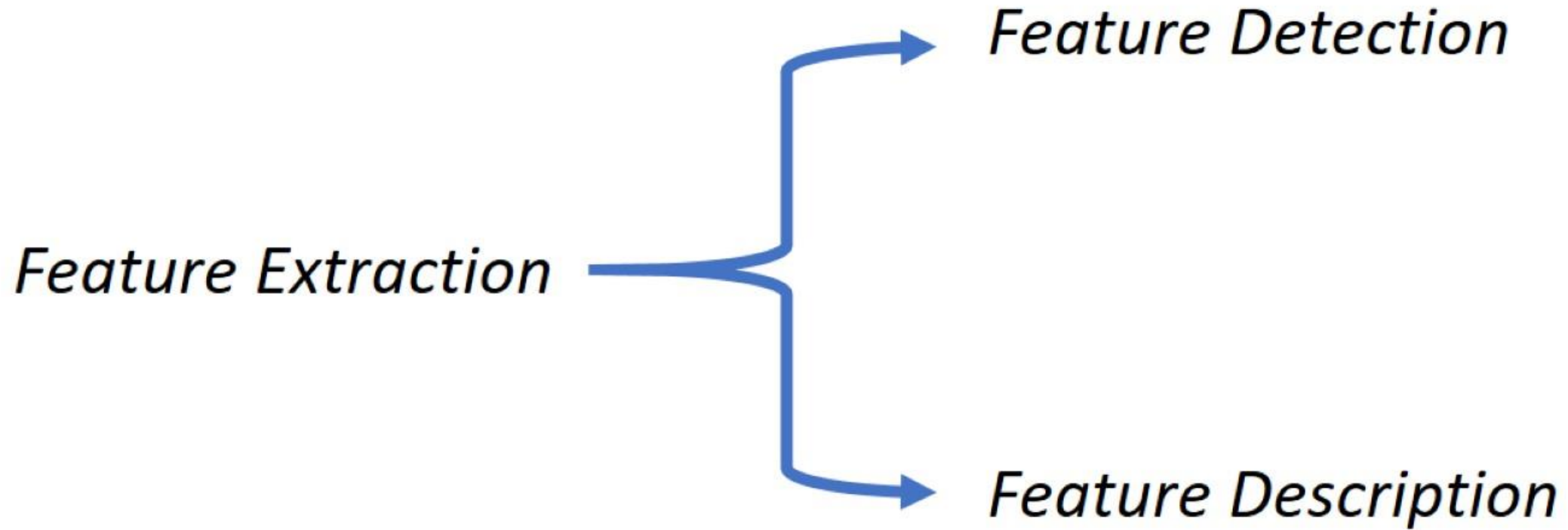
Feature

- A feature is a piece of information which is relevant for solving the computational task related to a certain application.
- A feature must be relevant to the problem. For example:
 - For **object recognition**, key features might include edges, corners, or color patterns.
 - For **face detection**, features could be eyes, nose, or mouth positions.

Image Features

- **Two types of image features:**
 - **Local Features** (e.g., corners, blobs, edges) – Useful for matching different views of an object.
 - **Global Features** (e.g., color histograms, texture patterns) – Describe the entire image or large regions.
 - **Structural Features** (e.g., lines, contours) – Help in shape analysis.

Feature Extraction



Feature Extraction

Feature Detection:

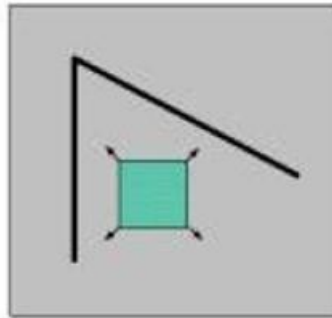
- *The process of identifying key points or regions in an image that are unique and relevant for analysis.*
- *These features are often edges, corners, blobs, or other patterns that stand out from their surroundings.*

Example: Detecting corners in an image using the **Harris Corner Detector**, which identifies points where there is a significant change in intensity in multiple directions.

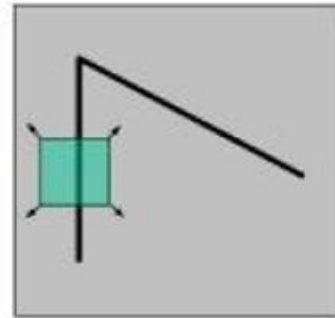
Feature Extraction

Corner:

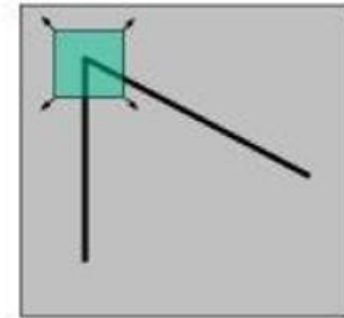
- Corners are the important features in the image, and they are generally termed as interest points which are invariant to translation, rotation, and illumination.*



"flat" region:
no change in all
directions



"edge":
no change along the
edge direction



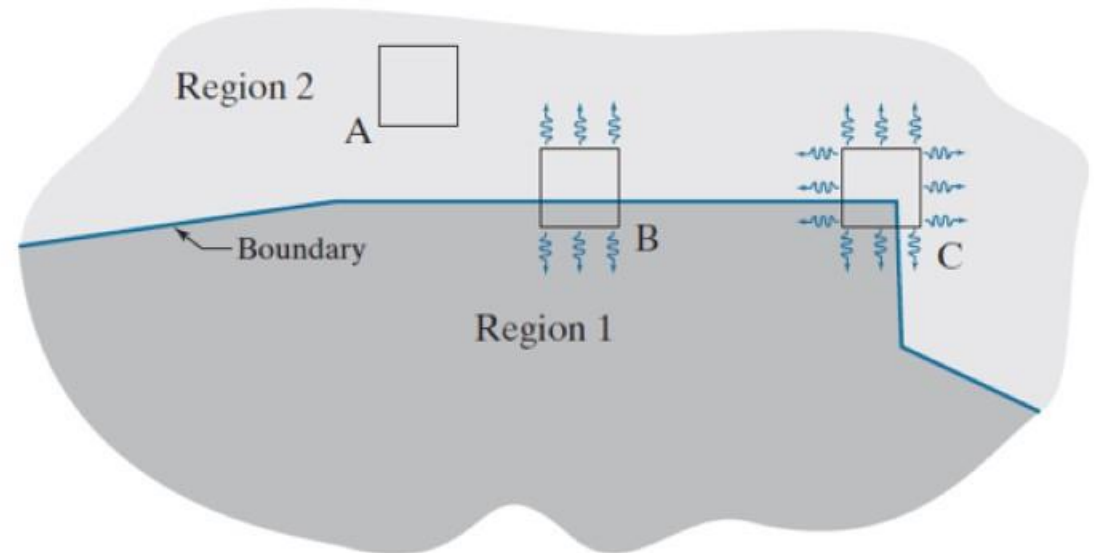
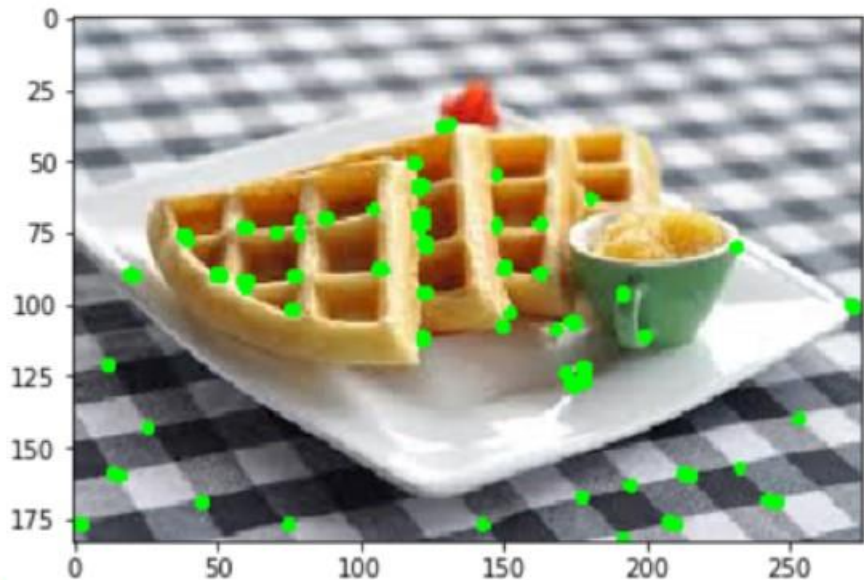
"corner":
significant change in
all directions

Feature Extraction

Harris Corner Detector:

Corners are detected by running a small window over an image, as we did in for spatial filtering.

Corners are areas of significant changes in all directions.



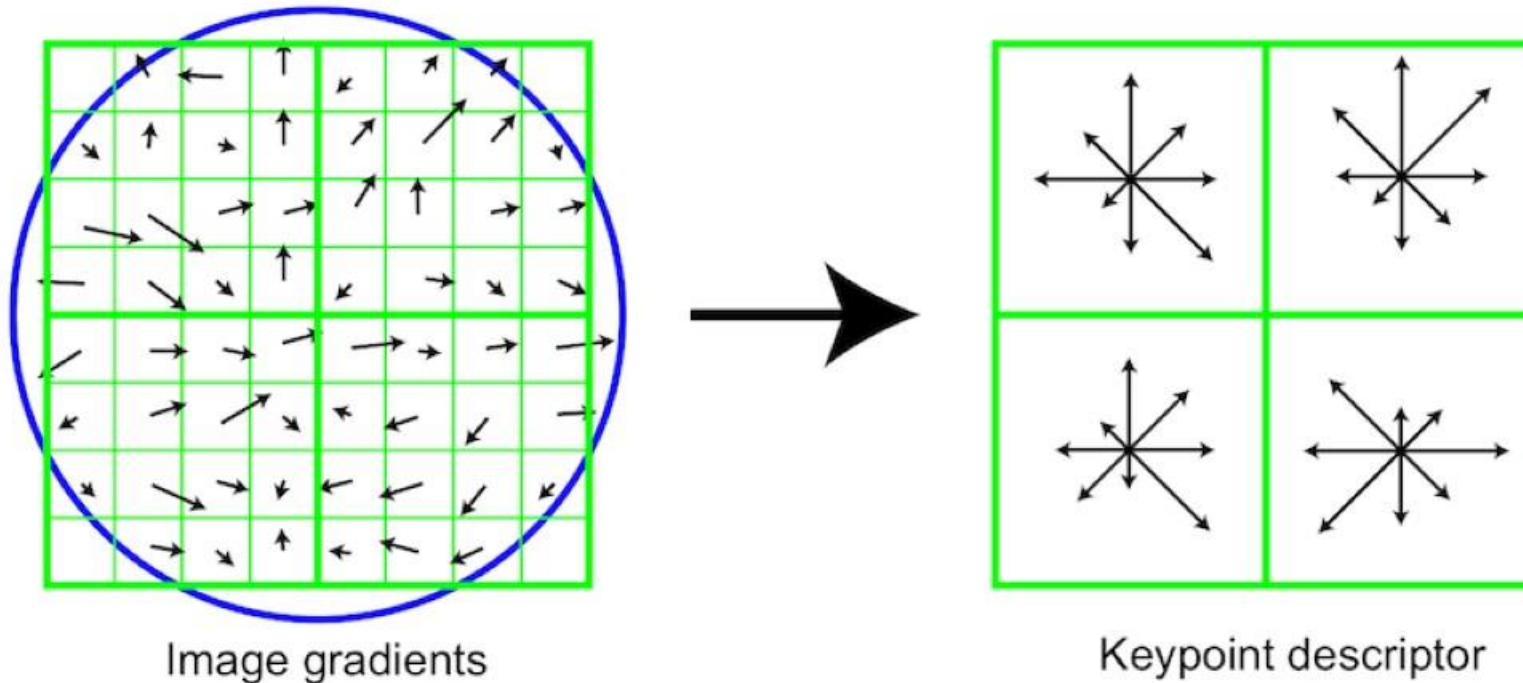
Feature Extraction

Feature Description:

- After detecting features, the next step is to describe them in a way that allows for matching and comparison between images.
- Feature descriptors capture information about the local neighborhood of a detected feature, typically in a vector format.

Example: Using **SIFT (Scale-Invariant Feature Transform)** to generate a descriptor for each detected keypoint, ensuring that the features are robust to changes in scale, rotation, and illumination.

Feature Descriptor

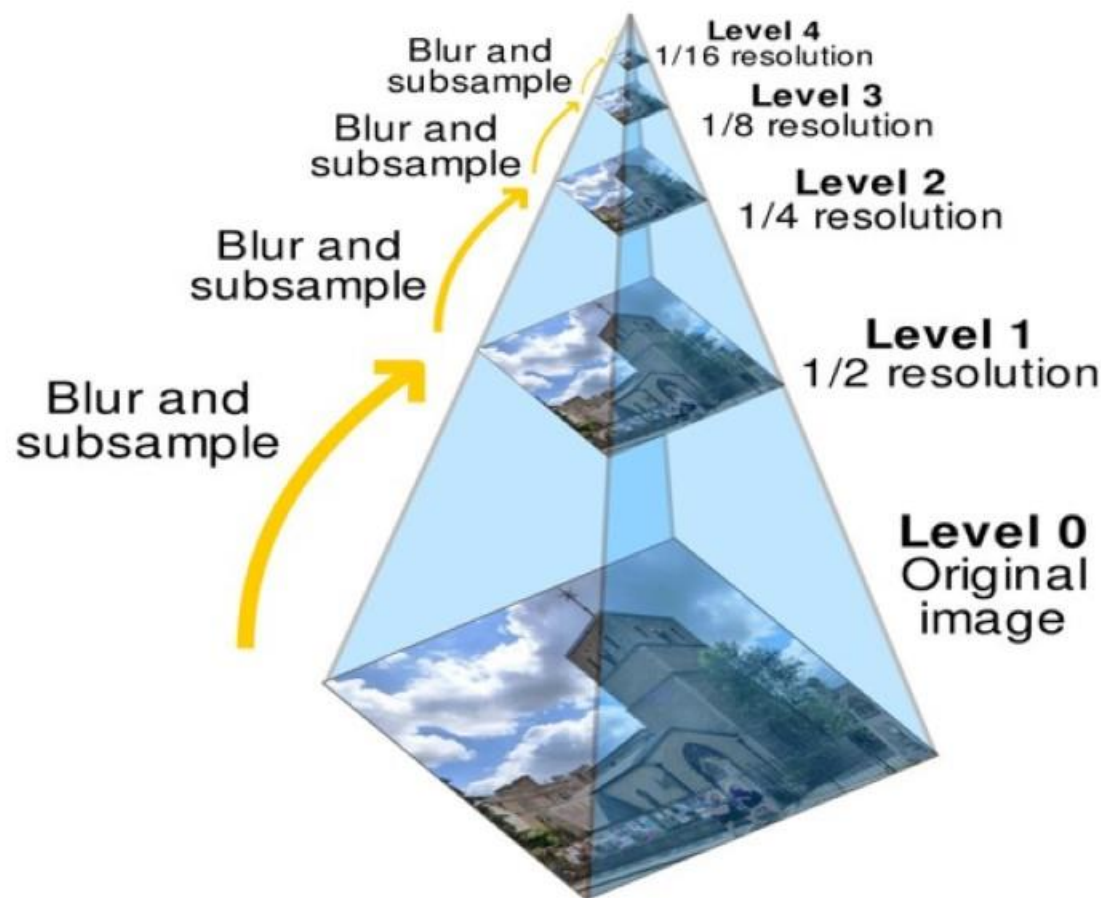


Feature description typically is used as a preprocessing step for higher-level tasks, such as image registration, object recognition for automated inspection, searching for patterns in image databases, and autonomous applications, such as robot and vehicle navigation.

- SIFT (Scale-Invariant Feature Transform) is a powerful algorithm used in computer vision for feature extraction.
- It was developed by David G. Lowe [2004] and is widely used for tasks such as object recognition, image stitching, and augmented reality.

1- Scale Space Construction:

- The first stage of the SIFT algorithm is to find image locations that are invariant to scale change.
- It starts by constructing a scale space, which is a set of images at different scales. This is achieved by convolving the original image with Gaussian filters (kernels) of varying sizes.



Gaussian Image Pyramid

Source: [Eric C. Stewart](#)

1- Scale Space Construction:

Based on this, the scale space, $L(x, y, \sigma)$, of a grayscale image, $f(x, y)$, is produced by convolving f with a variable-scale Gaussian kernel, $G(x, y, \sigma)$

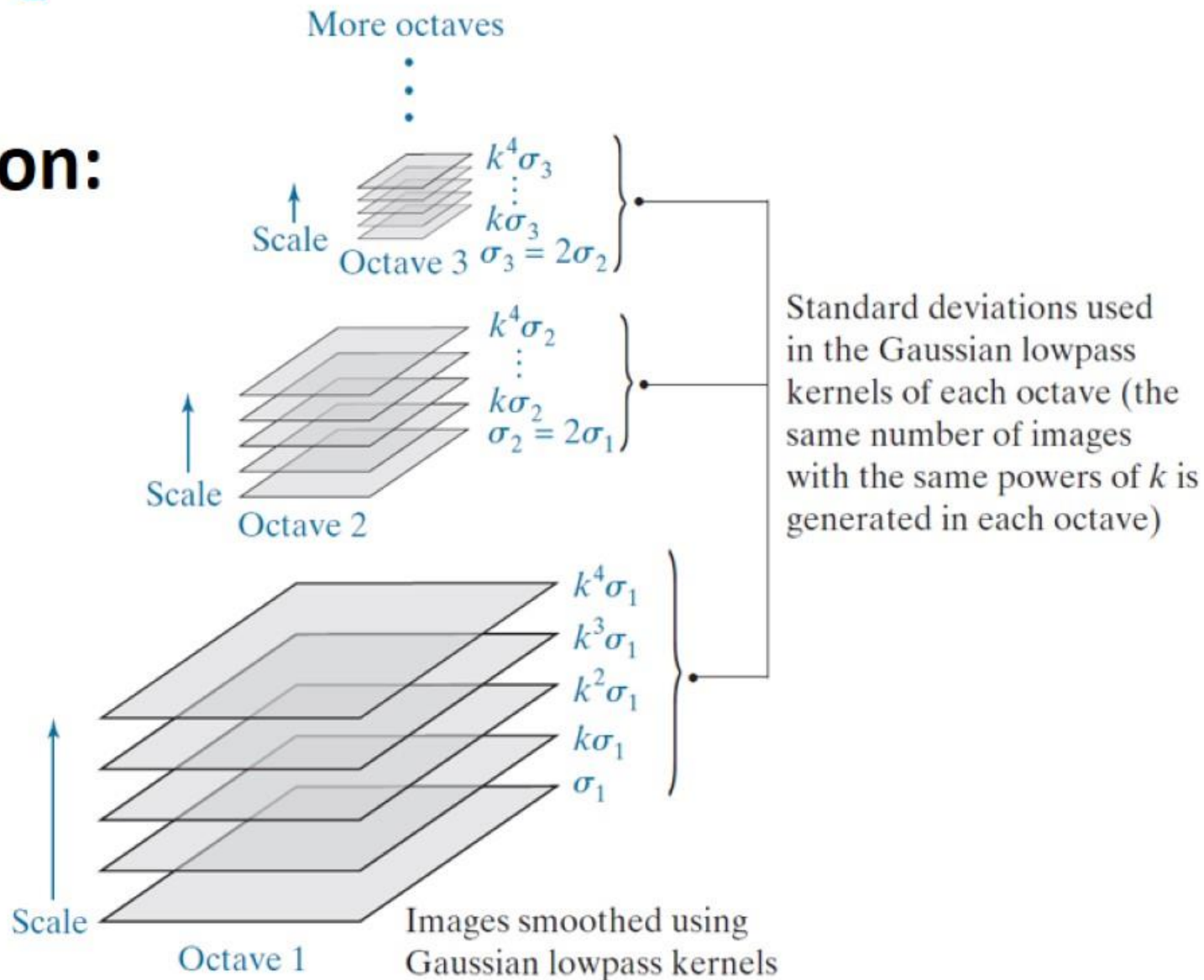
$$L(x, y, \sigma) = G(x, y, \sigma) \star f(x, y)$$

where the scale is controlled by parameter σ (representing blur level), and G is of the form

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2 + y^2)/2\sigma^2}$$

Gaussian kernels having standard deviations $\sigma, k\sigma, k^2\sigma, k^3\sigma,$

1- Scale Space Construction:

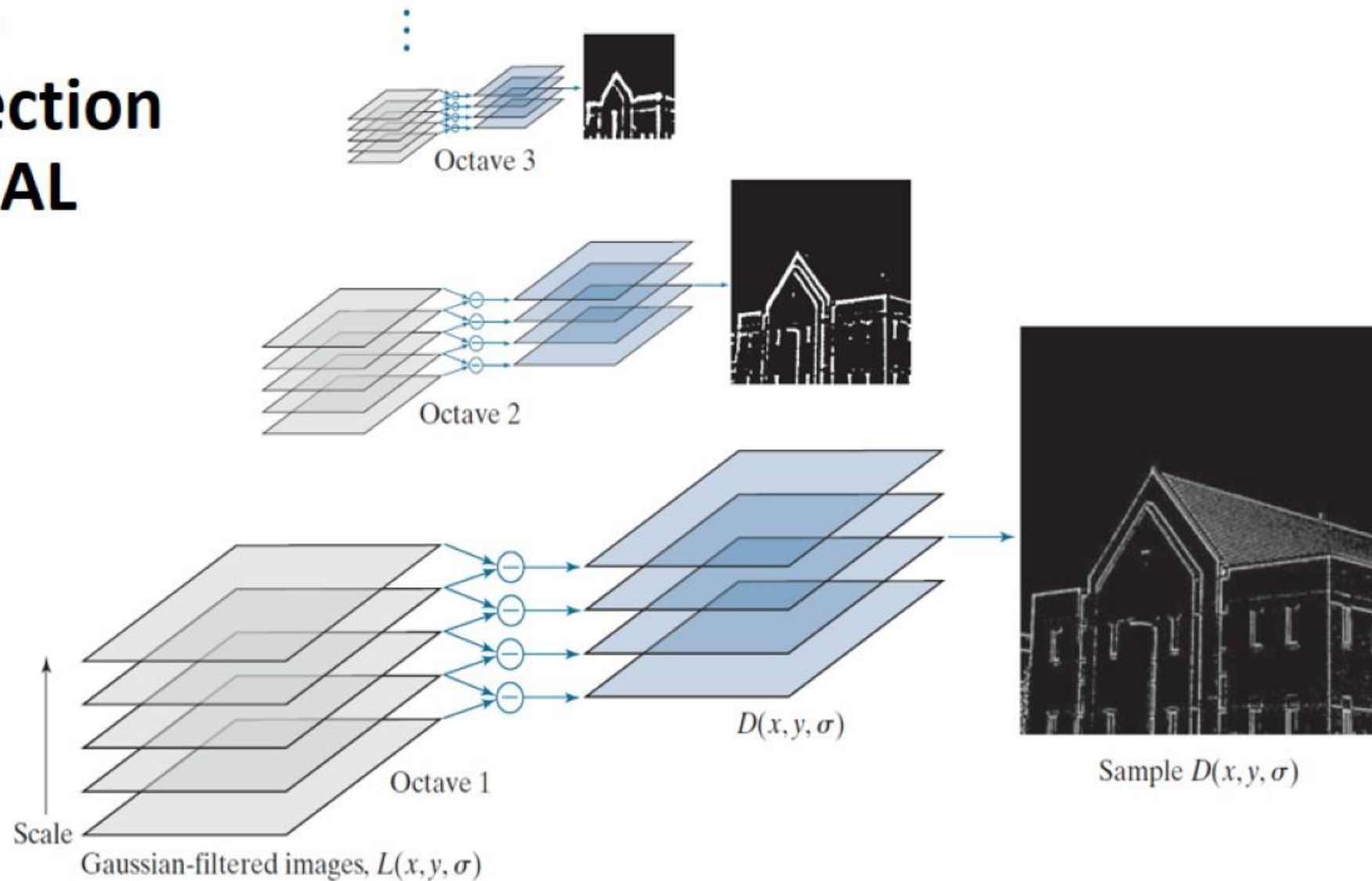


2- Keypoint Detection (DETECTING LOCAL EXTREMA):

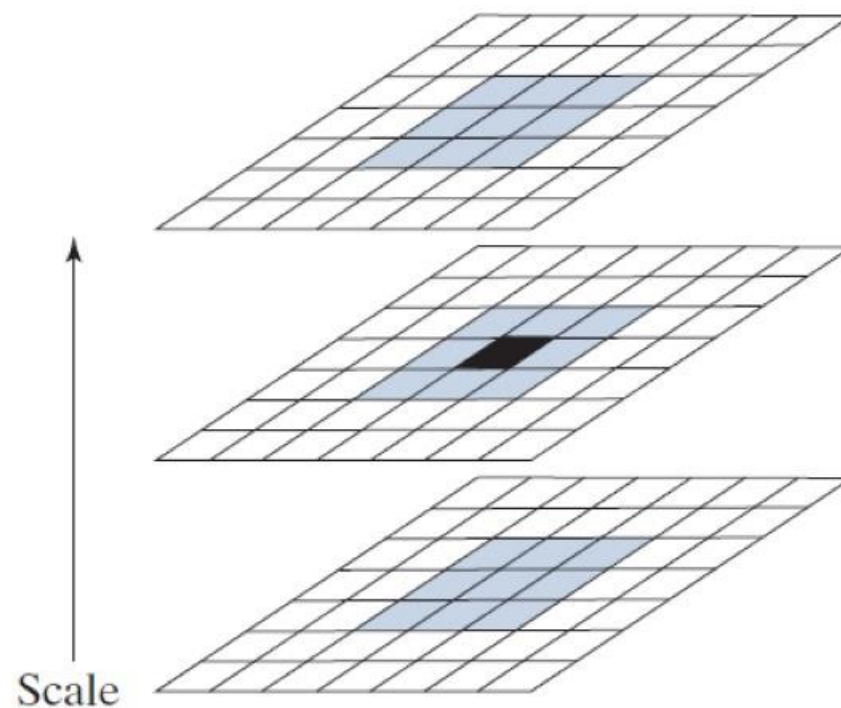
- SIFT detects keypoints by identifying extrema in the Difference of Gaussian (DoG) images.
- These keypoints are locations where the gradient magnitude is significant and stable across different scales.

$$D(x, y, \sigma) = [G(x, y, k\sigma) - G(x, y, \sigma)] \star f(x, y)$$

2- Keypoint Detection (DETECTING LOCAL EXTREMA):



- **Keypoint Detection (DETECTING LOCAL EXTREMA):**



Corresponding sections of three contiguous $D(x, y, \sigma)$ images

3- Improving the Accuracy of Keypoint Locations:

- After detecting candidate keypoints in the **Difference-of-Gaussians (DoG)** scale space, SIFT refines their positions to **sub-pixel accuracy** using a 3D quadratic fit. This step eliminates low-contrast points and improves localization precision.
- The refinement is based on a **second-order Taylor series expansion** of the DoG function $D(x,y,\sigma)$ around the detected keypoint:

$$\begin{aligned} D(\mathbf{x}) &= D + \left(\frac{\partial D}{\partial \mathbf{x}} \right)^T \mathbf{x} + \frac{1}{2} \mathbf{x}^T \frac{\partial}{\partial \mathbf{x}} \left(\frac{\partial D}{\partial \mathbf{x}} \right) \mathbf{x} \\ &= D + (\nabla D)^T \mathbf{x} + \frac{1}{2} \mathbf{x}^T \mathbf{H} \mathbf{x} \end{aligned}$$

$$\mathbf{H} = \begin{bmatrix} \partial^2 D / \partial x^2 & \partial^2 D / \partial x \partial y \\ \partial^2 D / \partial y \partial x & \partial^2 D / \partial y^2 \end{bmatrix} = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{yx} & D_{yy} \end{bmatrix}$$

Hessian Matrix

4- Keypoint Orientation:

- SIFT assigns an orientation to each keypoint based on the dominant gradient direction in its neighborhood.
- This allows us to represent a keypoint relative to its orientation and thus achieve *invariance to image rotation*.
- For **each pixel** in an image sample $L(x,y)$ (at a given scale), SIFT computes two key properties to build descriptors:
 - **Gradient magnitude** $M(x,y) \rightarrow$ How strong the intensity change is.
 - **Gradient orientation** $\theta(x,y) \rightarrow$ Direction of the intensity change.

$$M(x,y) = \left[(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2 \right]^{\frac{1}{2}}$$

$$\theta(x,y) = \tan^{-1} \left[(L(x,y+1) - L(x,y-1)) / (L(x+1,y) - L(x-1,y)) \right]$$

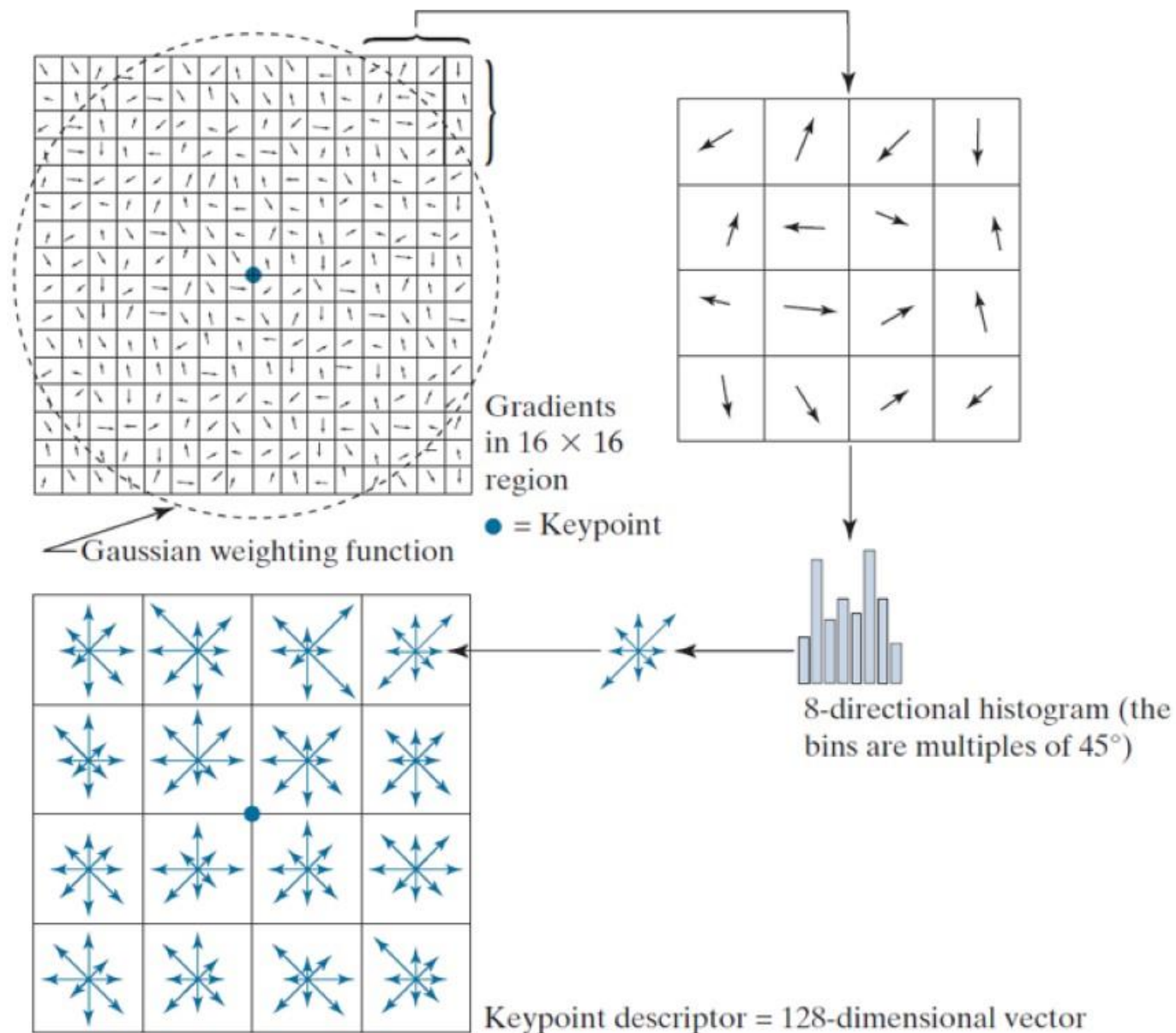
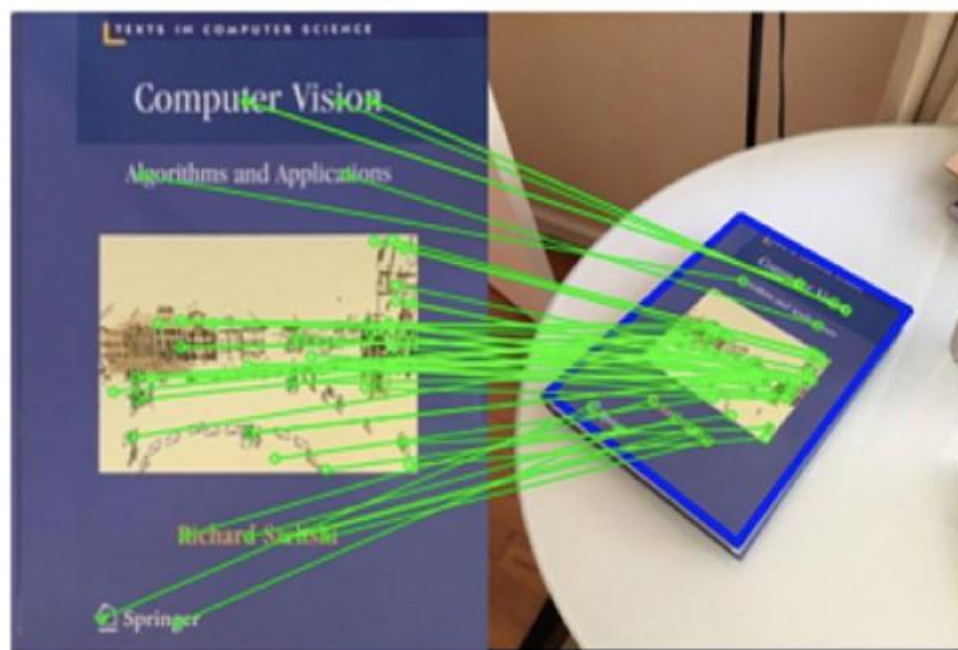
4- Keypoint Orientation:



4- Descriptor Generation:

- For each keypoint, a feature descriptor is generated by computing the gradient magnitude and direction in a neighborhood around the keypoint. This results in a 128-dimensional vector that describes the local appearance of the keypoint.
- The idea is to be able to use these descriptors to identify matches (similarities) between local regions in two or more images.

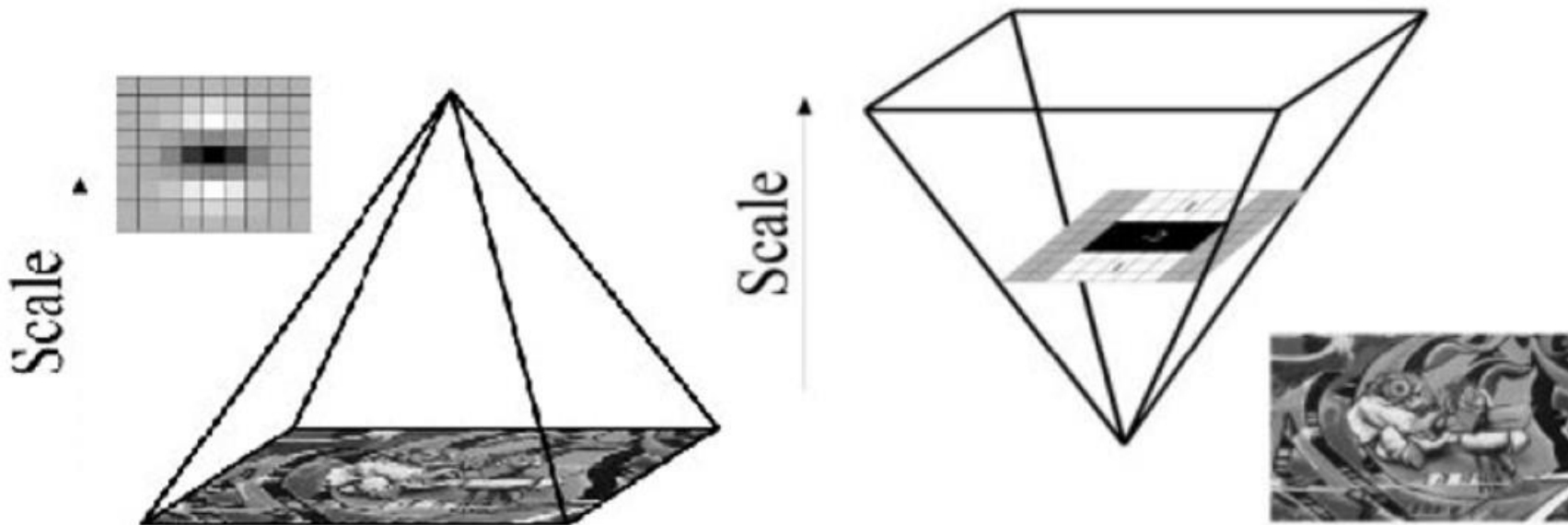
4- Descriptor Generation:



SURF (Speeded Up Robust Features) Feature Extraction

- **SURF** is a high-performance algorithm designed for quickly detecting and describing distinctive local features in images.
- Similarly to the SIFT method, the detection of features in SURF relies on a scale-space representation and rapid computation of feature descriptors using **box filters**. It is significantly speeding up the process.
- This efficiency makes SURF particularly suitable for real-time applications **like object tracking and recognition.**"

- While the Gaussian scale space is obtained by convolution of the initial images with Gaussian kernels, the discrete box-space is also obtained by convolving the original image with box filters at various scales.



Source:
[Dr Darshana Mistry](#)

- **The key innovation of using the SURF is the use of the Integral Images,** which allow a fast computation of any size box filter is used.
- **Integral Image (Summed-Area Table):**
 - Each pixel in the integral image stores the sum of all pixels from the top-left corner (origin) to that pixel.
 - This means each pixel contains information about the total intensity up to that point.
 - Once the integral image is built, you can find the sum of any rectangular region **using just four additions**, no matter how large the rectangle is.
 - Instead of summing individual pixels one by one (which is slow), we use precomputed values to get the sum instantly.

- **Integral Image (Summed-Area Table):**

The **integral image** at each position (x,y) stores the sum of all pixels from the top-left corner $(0,0)$ to (x,y) . It is calculated as:

$$I_{\Sigma}(x, y) = I(x, y) + I_{\Sigma}(x - 1, y) + I_{\Sigma}(x, y - 1) - I_{\Sigma}(x - 1, y - 1)$$

I=

1	2	3
4	5	6
7	8	9

- **Integral Image (Summed-Area Table):**

The **integral image** at each position (x,y) stores the sum of all pixels from the top-left corner $(0,0)$ to (x,y) . It is calculated as:

$$I_{\Sigma}(x, y) = I(x, y) + I_{\Sigma}(x - 1, y) + I_{\Sigma}(x, y - 1) - I_{\Sigma}(x - 1, y - 1)$$

I =

1	2	3
4	5	6
7	8	9

- **Integral Image (Summed-Area Table):**

The **integral image** at each position (x,y) stores the sum of all pixels from the top-left corner $(0,0)$ to (x,y) . It is calculated as:

$$I_{\Sigma}(x, y) = I(x, y) + I_{\Sigma}(x - 1, y) + I_{\Sigma}(x, y - 1) - I_{\Sigma}(x - 1, y - 1)$$

I=

1	2	3
4	5	6
7	8	9

1		

- **Integral Image (Summed-Area Table):**

The **integral image** at each position (x,y) stores the sum of all pixels from the top-left corner $(0,0)$ to (x,y) . It is calculated as:

$$I_{\Sigma}(x, y) = I(x, y) + I_{\Sigma}(x - 1, y) + I_{\Sigma}(x, y - 1) - I_{\Sigma}(x - 1, y - 1)$$

I =

1	2	3
4	5	6
7	8	9

1		

- **Integral Image (Summed-Area Table):**

The **integral image** at each position (x,y) stores the sum of all pixels from the top-left corner $(0,0)$ to (x,y) . It is calculated as:

$$I_{\Sigma}(x, y) = I(x, y) + I_{\Sigma}(x - 1, y) + I_{\Sigma}(x, y - 1) - I_{\Sigma}(x - 1, y - 1)$$

I =

1	2	3
4	5	6
7	8	9

1	3	

- **Integral Image (Summed-Area Table):**

The **integral image** at each position (x,y) stores the sum of all pixels from the top-left corner $(0,0)$ to (x,y) . It is calculated as:

$$I_{\Sigma}(x, y) = I(x, y) + I_{\Sigma}(x - 1, y) + I_{\Sigma}(x, y - 1) - I_{\Sigma}(x - 1, y - 1)$$

I=

1	2	3
4	5	6
7	8	9

1	3	

- **Integral Image (Summed-Area Table):**

The **integral image** at each position (x,y) stores the sum of all pixels from the top-left corner $(0,0)$ to (x,y) . It is calculated as:

$$I_{\Sigma}(x, y) = I(x, y) + I_{\Sigma}(x - 1, y) + I_{\Sigma}(x, y - 1) - I_{\Sigma}(x - 1, y - 1)$$

I=

1	2	3
4	5	6
7	8	9

1	3	6

- **Integral Image (Summed-Area Table):**

The **integral image** at each position (x,y) stores the sum of all pixels from the top-left corner $(0,0)$ to (x,y) . It is calculated as:

$$I_{\Sigma}(x, y) = I(x, y) + I_{\Sigma}(x - 1, y) + I_{\Sigma}(x, y - 1) - I_{\Sigma}(x - 1, y - 1)$$

I =

1	2	3
4	5	6
7	8	9

1	3	6

- **Integral Image (Summed-Area Table):**

The **integral image** at each position (x,y) stores the sum of all pixels from the top-left corner $(0,0)$ to (x,y) . It is calculated as:

$$I_{\Sigma}(x, y) = I(x, y) + I_{\Sigma}(x - 1, y) + I_{\Sigma}(x, y - 1) - I_{\Sigma}(x - 1, y - 1)$$

I =

1	2	3
4	5	6
7	8	9

1	3	6
5		

- **Integral Image (Summed-Area Table):**

The **integral image** at each position (x,y) stores the sum of all pixels from the top-left corner $(0,0)$ to (x,y) . It is calculated as:

$$I_{\Sigma}(x, y) = I(x, y) + I_{\Sigma}(x - 1, y) + I_{\Sigma}(x, y - 1) - I_{\Sigma}(x - 1, y - 1)$$

I =

1	2	3
4	5	6
7	8	9

1	3	6
5		

- **Integral Image (Summed-Area Table):**

The **integral image** at each position (x,y) stores the sum of all pixels from the top-left corner $(0,0)$ to (x,y) . It is calculated as:

$$I_{\Sigma}(x, y) = I(x, y) + I_{\Sigma}(x - 1, y) + I_{\Sigma}(x, y - 1) - I_{\Sigma}(x - 1, y - 1)$$

I =

1	2	3
4	5	6
7	8	9

1	3	6
5	12	

- **Integral Image (Summed-Area Table):**

The **integral image** at each position (x,y) stores the sum of all pixels from the top-left corner $(0,0)$ to (x,y) . It is calculated as:

$$I_{\Sigma}(x, y) = I(x, y) + I_{\Sigma}(x - 1, y) + I_{\Sigma}(x, y - 1) - I_{\Sigma}(x - 1, y - 1)$$

I =

1	2	3
4	5	6
7	8	9

1	3	6
5	12	

- **Integral Image (Summed-Area Table):**

The **integral image** at each position (x,y) stores the sum of all pixels from the top-left corner $(0,0)$ to (x,y) . It is calculated as:

$$I_{\Sigma}(x, y) = I(x, y) + I_{\Sigma}(x - 1, y) + I_{\Sigma}(x, y - 1) - I_{\Sigma}(x - 1, y - 1)$$

I =

1	2	3
4	5	6
7	8	9

1	3	6
5	12	21

- **Integral Image (Summed-Area Table):**

The **integral image** at each position (x,y) stores the sum of all pixels from the top-left corner $(0,0)$ to (x,y) . It is calculated as:

$$I_{\Sigma}(x, y) = I(x, y) + I_{\Sigma}(x - 1, y) + I_{\Sigma}(x, y - 1) - I_{\Sigma}(x - 1, y - 1)$$

I =

1	2	3
4	5	6
7	8	9

1	3	6
5	12	21

- **Integral Image (Summed-Area Table):**

The **integral image** at each position (x,y) stores the sum of all pixels from the top-left corner $(0,0)$ to (x,y) . It is calculated as:

$$I_{\Sigma}(x, y) = I(x, y) + I_{\Sigma}(x - 1, y) + I_{\Sigma}(x, y - 1) - I_{\Sigma}(x - 1, y - 1)$$

I =

1	2	3
4	5	6
7	8	9

1	3	6
5	12	21
12		

- **Integral Image (Summed-Area Table):**

The **integral image** at each position (x,y) stores the sum of all pixels from the top-left corner $(0,0)$ to (x,y) . It is calculated as:

$$I_{\Sigma}(x, y) = I(x, y) + I_{\Sigma}(x - 1, y) + I_{\Sigma}(x, y - 1) - I_{\Sigma}(x - 1, y - 1)$$

I =

1	2	3
4	5	6
7	8	9

1	3	6
5	12	21
12		

- **Integral Image (Summed-Area Table):**

The **integral image** at each position (x,y) stores the sum of all pixels from the top-left corner $(0,0)$ to (x,y) . It is calculated as:

$$I_{\Sigma}(x, y) = I(x, y) + I_{\Sigma}(x - 1, y) + I_{\Sigma}(x, y - 1) - I_{\Sigma}(x - 1, y - 1)$$

I =

1	2	3
4	5	6
7	8	9

1	3	6
5	12	21
12	27	

- **Integral Image (Summed-Area Table):**

The **integral image** at each position (x,y) stores the sum of all pixels from the top-left corner $(0,0)$ to (x,y) . It is calculated as:

$$I_{\Sigma}(x, y) = I(x, y) + I_{\Sigma}(x - 1, y) + I_{\Sigma}(x, y - 1) - I_{\Sigma}(x - 1, y - 1)$$

I =

1	2	3
4	5	6
7	8	9

1	3	6
5	12	21
12	27	

- **Integral Image (Summed-Area Table):**

The **integral image** at each position (x,y) stores the sum of all pixels from the top-left corner $(0,0)$ to (x,y) . It is calculated as:

$$I_{\Sigma}(x, y) = I(x, y) + I_{\Sigma}(x - 1, y) + I_{\Sigma}(x, y - 1) - I_{\Sigma}(x - 1, y - 1)$$

I =

1	2	3
4	5	6
7	8	9

1	3	6
5	12	21
12	27	45

- **Integral Image (Summed-Area Table):**

The **integral image** at each position (x,y) stores the sum of all pixels from the top-left corner $(0,0)$ to (x,y) . It is calculated as:

$$I_{\Sigma}(x, y) = I(x, y) + I_{\Sigma}(x - 1, y) + I_{\Sigma}(x, y - 1) - I_{\Sigma}(x - 1, y - 1)$$

I =

1	2	3
4	5	6
7	8	9

1	3	6
5	12	21
12	27	45

Integral Image of image "I"

- **Integral Image (Summed-Area Table):**
Applying a 2×2 Box Filter at position (1,1) to (2,2).

I =

1	2	3
4	5	6
7	8	9

1	3	6
5	12	21
12	27	45

Integral Image of image "I"

- **Integral Image (Summed-Area Table):**

Applying a 2×2 Box Filter at position (1,1) to (2,2).

$$\text{Sum} = 5 + 6 + 8 + 9 = 28$$

I =

1	2	3
4	5	6
7	8	9

1	3	6
5	12	21
12	27	45

Integral Image of image "I"

- **Integral Image (Summed-Area Table):**

Applying a 2×2 Box Filter at position $(1,1)$ to $(2,2)$.

The fast summation formula using the integral image $\rightarrow$ Sum =

$$I_{\Sigma}(x_2, y_2) - I_{\Sigma}(x_1 - 1, y_2) - I_{\Sigma}(x_2, y_1 - 1) + I_{\Sigma}(x_1 - 1, y_1 - 1)$$

I =

1	2	3
4	5	6
7	8	9

1	3	6
5	12	21
12	27	45

Integral Image of image "I"

- **Integral Image (Summed-Area Table):**

Applying a 2×2 Box Filter at position (1,1) to (2,2).

The fast summation formula using the integral image $\rightarrow$ Sum =

$$I_{\Sigma}(x_2, y_2) - I_{\Sigma}(x_1 - 1, y_2) - I_{\Sigma}(x_2, y_1 - 1) + I_{\Sigma}(x_1 - 1, y_1 - 1)$$

I =

1	2	3
4	5	6
7	8	9

1	3	6
5	12	21
12	27	45

Integral Image of image "I"

- **Integral Image (Summed-Area Table):**

Applying a 2×2 Box Filter at position (1,1) to (2,2).

The fast summation formula using the integral image $\rightarrow$ Sum =

$$I_{\Sigma}(x_2, y_2) - I_{\Sigma}(x_1 - 1, y_2) - I_{\Sigma}(x_2, y_1 - 1) + I_{\Sigma}(x_1 - 1, y_1 - 1)$$

I =

1	2	3
4	5	6
7	8	9

1	3	6
5	12	21
12	27	45

Integral Image of image "I"

- **Integral Image (Summed-Area Table):**

Applying a 2×2 Box Filter at position (1,1) to (2,2).

The fast summation formula using the integral image $\rightarrow$ Sum =

$$I_{\Sigma}(x_2, y_2) - I_{\Sigma}(x_1 - 1, y_2) - I_{\Sigma}(x_2, y_1 - 1) + I_{\Sigma}(x_1 - 1, y_1 - 1)$$

I =

1	2	3
4	5	6
7	8	9

1	3	6
5	12	21
12	27	45

Integral Image of image "I"

- **Integral Image (Summed-Area Table):**

Applying a 2×2 Box Filter at position (1,1) to (2,2).

The fast summation formula using the integral image $\rightarrow$ Sum =

$$I_{\Sigma}(x_2, y_2) - I_{\Sigma}(x_1 - 1, y_2) - I_{\Sigma}(x_2, y_1 - 1) + I_{\Sigma}(x_1 - 1, y_1 - 1)$$

I =

1	2	3
4	5	6
7	8	9

1	3	6
5	12	21
12	27	45

Integral Image of image "I"

- **Integral Image (Summed-Area Table):**

Applying a 2×2 Box Filter at position (1,1) to (2,2).

The fast summation formula using the integral image $\rightarrow$ Sum =

$$I_{\Sigma}(x_2, y_2) - I_{\Sigma}(x_1 - 1, y_2) - I_{\Sigma}(x_2, y_1 - 1) + I_{\Sigma}(x_1 - 1, y_1 - 1)$$

$$\text{Sum} = 45 - 6 - 12 + 1 = 28$$

Only 4 operation instead of
summing each pixel individually

1	3	6
5	12	21
12	27	45

Integral Image of image "I"

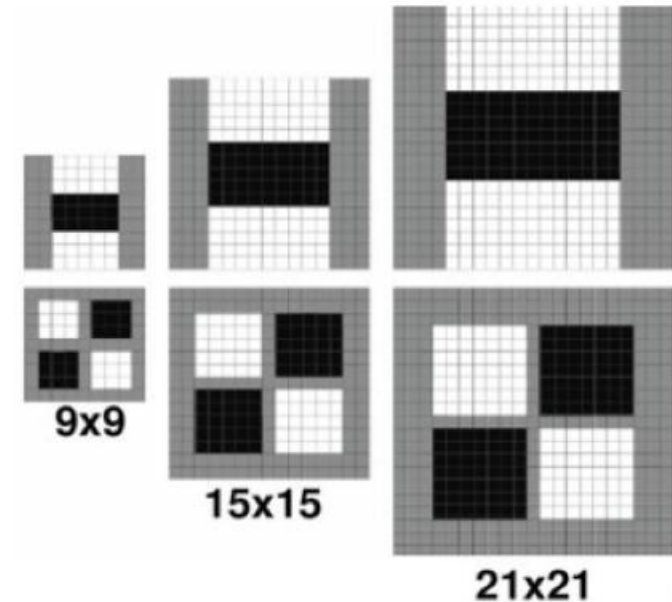
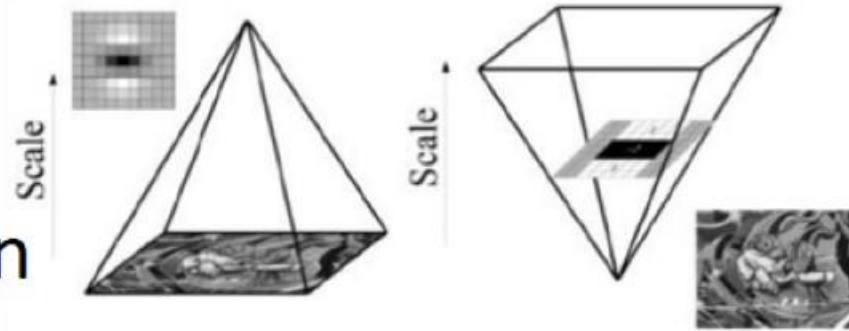
Key Steps in SURF

- Interest Point detection using the Hessian matrix for each pixel at location (x,y) and scale σ .
- Interest points are selected where the determinant of H is maximized

$$H(x, \sigma) = \begin{bmatrix} L_{xx}(x, \sigma) & L_{xy}(x, \sigma) \\ L_{xy}(x, \sigma) & L_{yy}(x, \sigma) \end{bmatrix}$$

Scale Space Construction

Scales are grouped into octaves



Key Steps in SURF

- **Steps for Orientation Assignment in SURF**

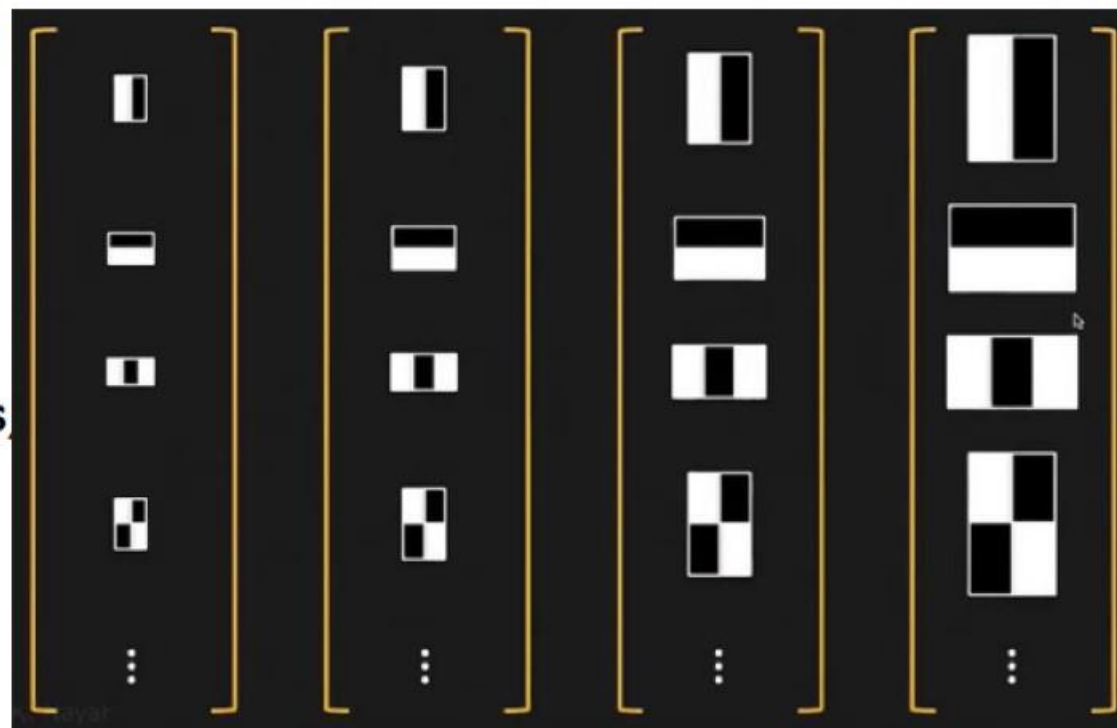
- Calculate Haar-wavelet responses in **x** and **y** directions within a circular neighborhood of **radius $6s$** around the keypoint.
- The scale **s** is the detection scale of the keypoint.

- **Scale-Dependent Sampling**

- The **sampling step** is proportional to the scale s .
- Wavelet responses are computed at the **current scale s** meaning larger wavelets for higher scales.

- **Use Integral Images for Fast Filtering**

- Since large wavelets are computationally expensive, **integral images** are used to speed up filtering.

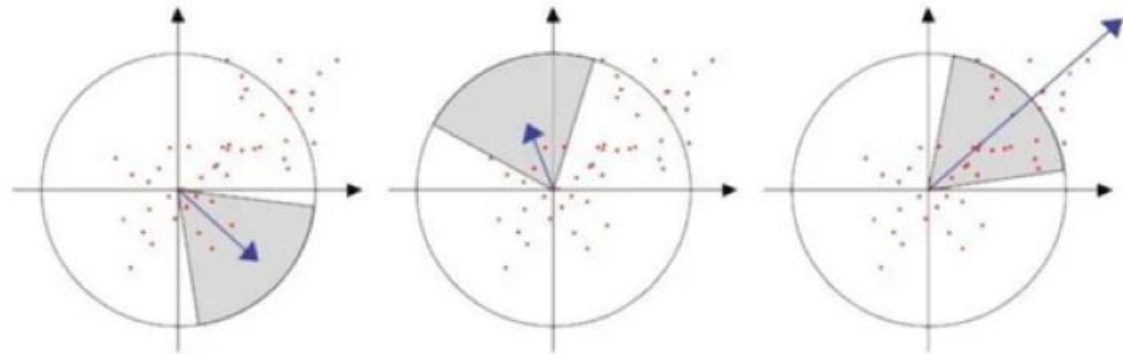


Source:

[First Principles of Computer Vision](#)

Key Steps in SURF

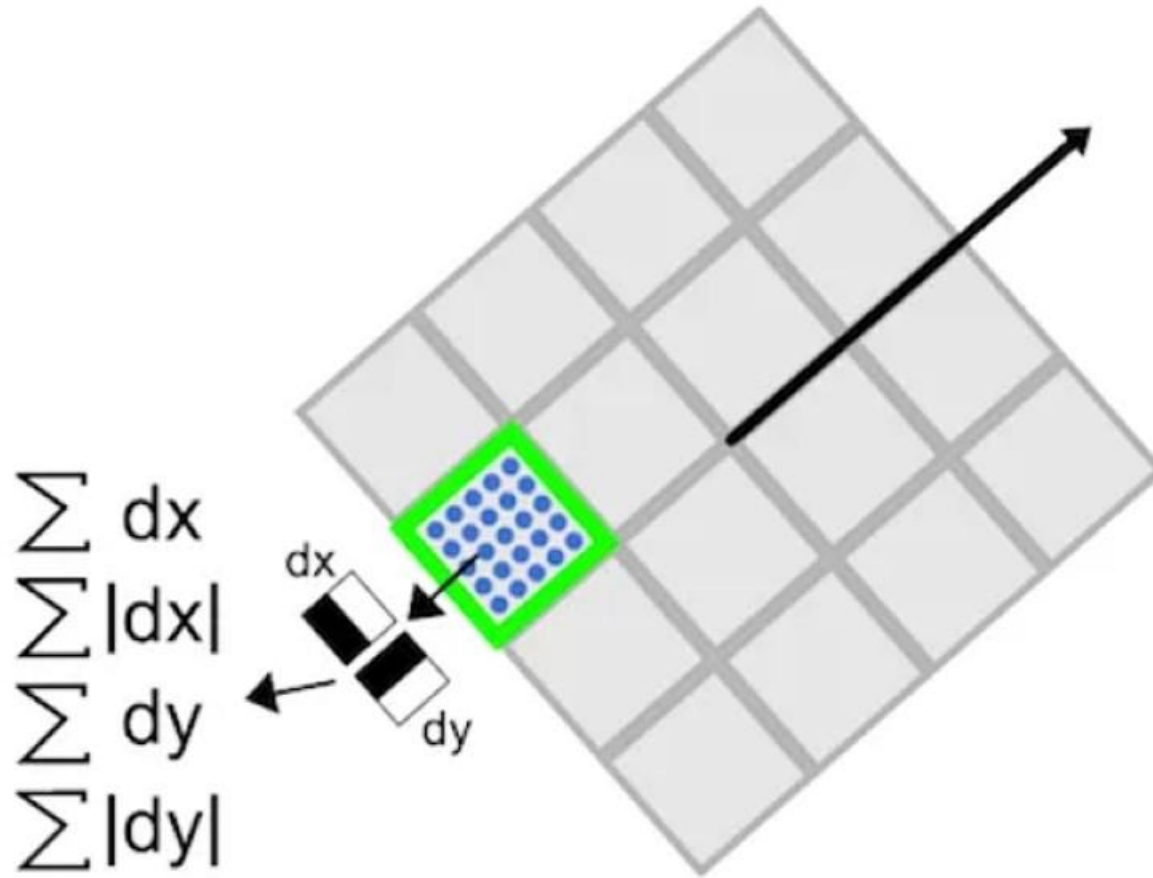
- **Sum Wavelet Responses in a Scanning Area**
 - Compute the **sum of vertical and horizontal** Haar-wavelet responses over a scanning region.
- **Rotate the Scanning Orientation**
 - Change the scanning orientation by $\pi/3$ (**60 degrees**) and re-calculate the response sum.
- **Find the Dominant Orientation**
 - Repeat scanning in different orientations until the **largest sum value** is found.
 - The orientation with the **maximum sum value** is assigned as the **main feature orientation**.



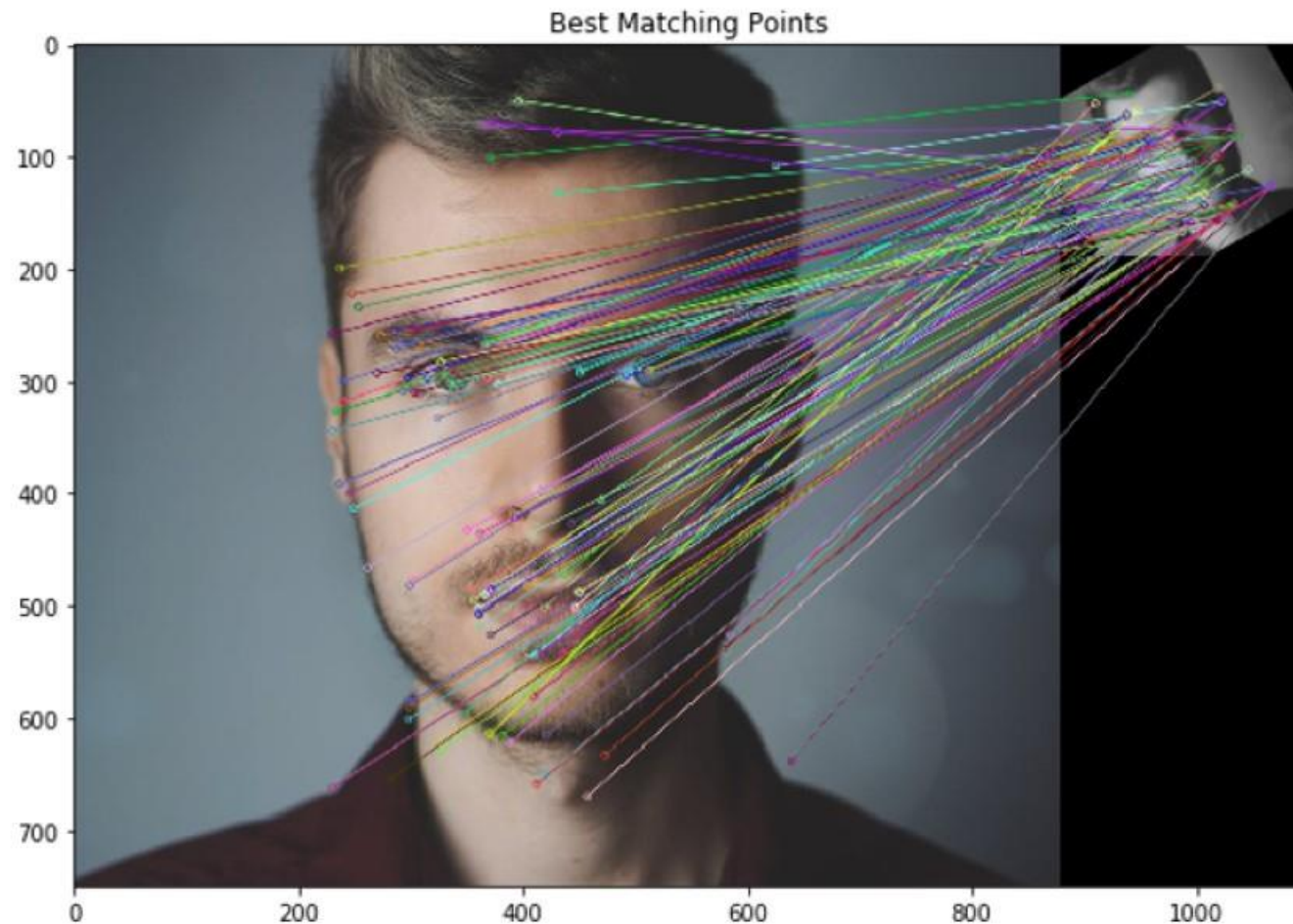
Key Steps in SURF

- **Descriptor Extraction in SURF**
- Centered around the **keypoint** and aligned with its **orientation**.
- The window size is **$20s$** , where s is the scale of the keypoint.
- The region is split into **16 smaller sub-regions** (4×4 grid).
- At each **5×5 regularly spaced** sample point, compute Haar-wavelet responses:
 - dx : Response in the **horizontal** direction.
 - dy : Response in the **vertical** direction.
- The filter size used for wavelets is **$2s$** .
- **Apply Gaussian Weighting** To improve robustness to noise and distortions, **Gaussian weighting ($\sigma = 3.3s$)** for dx and dy .

Key Steps in SURF



Face Detection Using SURF



The background features a series of vertical panels with rounded ends. The first panel shows hands typing on a laptop. The second panel shows a graduation cap icon. The third panel shows a play button icon. The fourth panel shows a target icon. The fifth panel shows an open book icon. The sixth panel shows a magnifying glass icon. The seventh panel shows a group of people icon. The eighth panel shows a bar chart icon. The ninth panel shows a line graph icon. The tenth panel shows a pie chart icon. The eleventh panel shows a scatter plot icon. The twelfth panel shows a line graph icon. The thirteenth panel shows a bar chart icon. The fourteenth panel shows a line graph icon. The fifteenth panel shows a bar chart icon. The sixteenth panel shows a line graph icon. The seventeenth panel shows a bar chart icon. The eighteenth panel shows a line graph icon. The nineteenth panel shows a bar chart icon. The twentieth panel shows a line graph icon. The twenty-first panel shows a bar chart icon. The twenty-second panel shows a line graph icon. The twenty-third panel shows a bar chart icon. The twenty-four panel shows a line graph icon. The twenty-fifth panel shows a bar chart icon. The twenty-sixth panel shows a line graph icon. The twenty-seventh panel shows a bar chart icon. The twenty-eighth panel shows a line graph icon. The twenty-ninth panel shows a bar chart icon. The thirtieth panel shows a line graph icon. The thirty-first panel shows a bar chart icon. The thirty-second panel shows a line graph icon. The thirty-third panel shows a bar chart icon. The thirty-fourth panel shows a line graph icon. The thirty-fifth panel shows a bar chart icon. The thirty-sixth panel shows a line graph icon. The thirty-seventh panel shows a bar chart icon. The thirty-eighth panel shows a line graph icon. The thirty-ninth panel shows a bar chart icon. The fortieth panel shows a line graph icon. The forty-first panel shows a bar chart icon. The forty-second panel shows a line graph icon. The forty-third panel shows a bar chart icon. The forty-fourth panel shows a line graph icon. The forty-fifth panel shows a bar chart icon. The forty-sixth panel shows a line graph icon. The forty-seventh panel shows a bar chart icon. The forty-eighth panel shows a line graph icon. The forty-ninth panel shows a bar chart icon. The fiftieth panel shows a line graph icon. The fifty-first panel shows a bar chart icon. The fifty-second panel shows a line graph icon. The fifty-third panel shows a bar chart icon. The fifty-fourth panel shows a line graph icon. The fifty-fifth panel shows a bar chart icon. The fifty-sixth panel shows a line graph icon. The fifty-seventh panel shows a bar chart icon. The fifty-eighth panel shows a line graph icon. The fifty-ninth panel shows a bar chart icon. The sixtieth panel shows a line graph icon. The sixty-first panel shows a bar chart icon. The sixty-second panel shows a line graph icon. The sixty-third panel shows a bar chart icon. The sixty-fourth panel shows a line graph icon. The sixty-fifth panel shows a bar chart icon. The sixty-sixth panel shows a line graph icon. The sixty-seventh panel shows a bar chart icon. The sixty-eighth panel shows a line graph icon. The sixty-ninth panel shows a bar chart icon. The seventieth panel shows a line graph icon. The seventy-first panel shows a bar chart icon. The seventy-second panel shows a line graph icon. The seventy-third panel shows a bar chart icon. The seventy-fourth panel shows a line graph icon. The seventy-fifth panel shows a bar chart icon. The seventy-sixth panel shows a line graph icon. The seventy-seventh panel shows a bar chart icon. The seventy-eighth panel shows a line graph icon. The seventy-ninth panel shows a bar chart icon. The eightieth panel shows a line graph icon. The eighty-first panel shows a bar chart icon. The eighty-second panel shows a line graph icon. The eighty-third panel shows a bar chart icon. The eighty-fourth panel shows a line graph icon. The eighty-fifth panel shows a bar chart icon. The eighty-sixth panel shows a line graph icon. The eighty-seventh panel shows a bar chart icon. The eighty-eighth panel shows a line graph icon. The eighty-ninth panel shows a bar chart icon. The ninetieth panel shows a line graph icon. The ninety-first panel shows a bar chart icon. The ninety-second panel shows a line graph icon. The ninety-third panel shows a bar chart icon. The ninety-fourth panel shows a line graph icon. The ninety-fifth panel shows a bar chart icon. The ninety-sixth panel shows a line graph icon. The ninety-seventh panel shows a bar chart icon. The ninety-eighth panel shows a line graph icon. The ninety-ninth panel shows a bar chart icon. The hundredth panel shows a line graph icon.

Object Detection

Bag of Features
(Bag of Virtual Words)

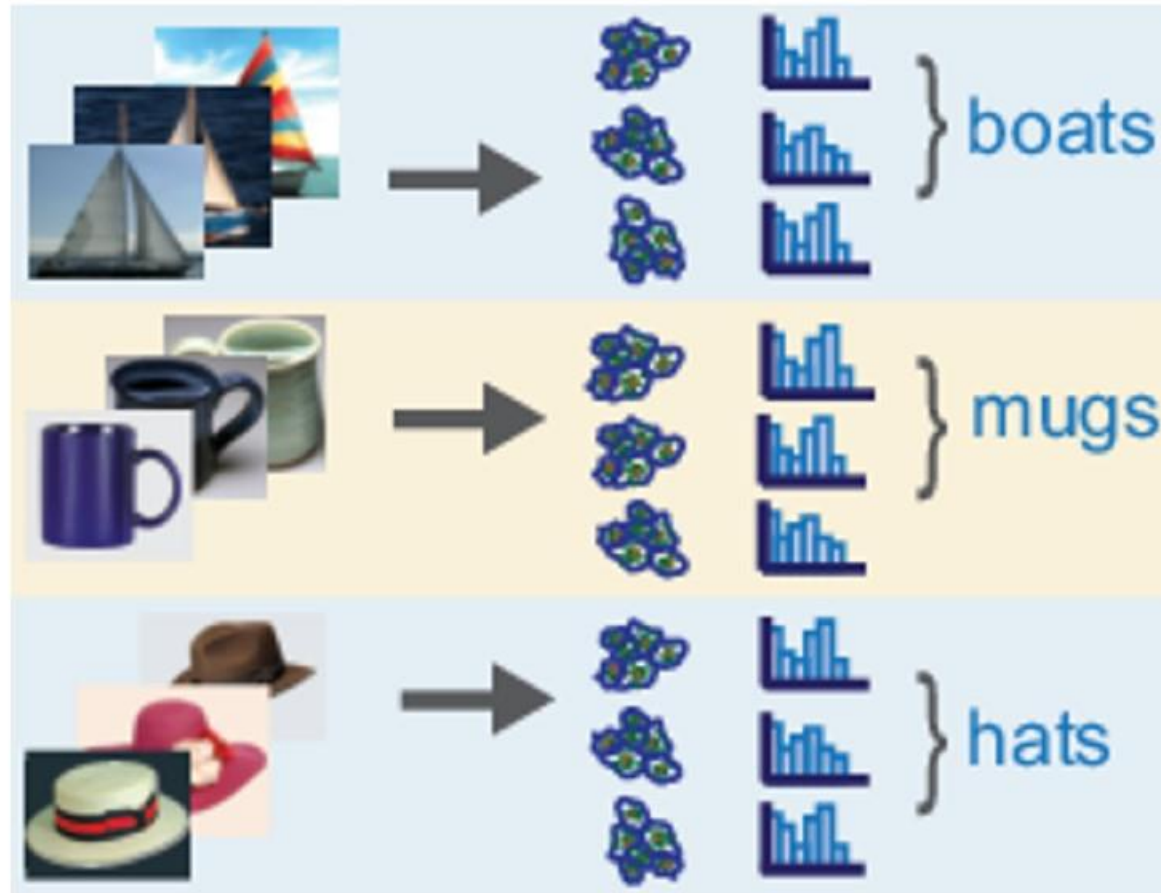
Bag of Words

I love this movie! It's sweet, but with satirical humor. The dialogue is great and the adventure scenes are fun... It manages to be whimsical and romantic while laughing at the conventions of the fairy tale genre. I would recommend it to just about anyone. I've seen it several times, and I'm always happy to see it again whenever I have a friend who hasn't seen it yet!



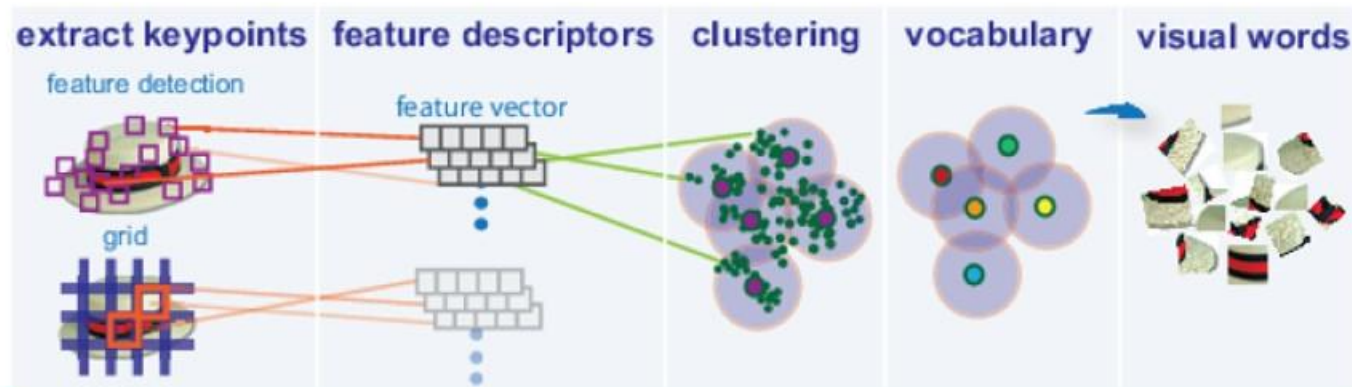
it	6
I	5
the	4
to	3
and	3
seen	2
yet	1
would	1
whimsical	1
times	1
sweet	1
satirical	1
adventure	1
genre	1
fairy	1
humor	1
have	1
great	1
...	...

Bag of Visual Words (BoVW)

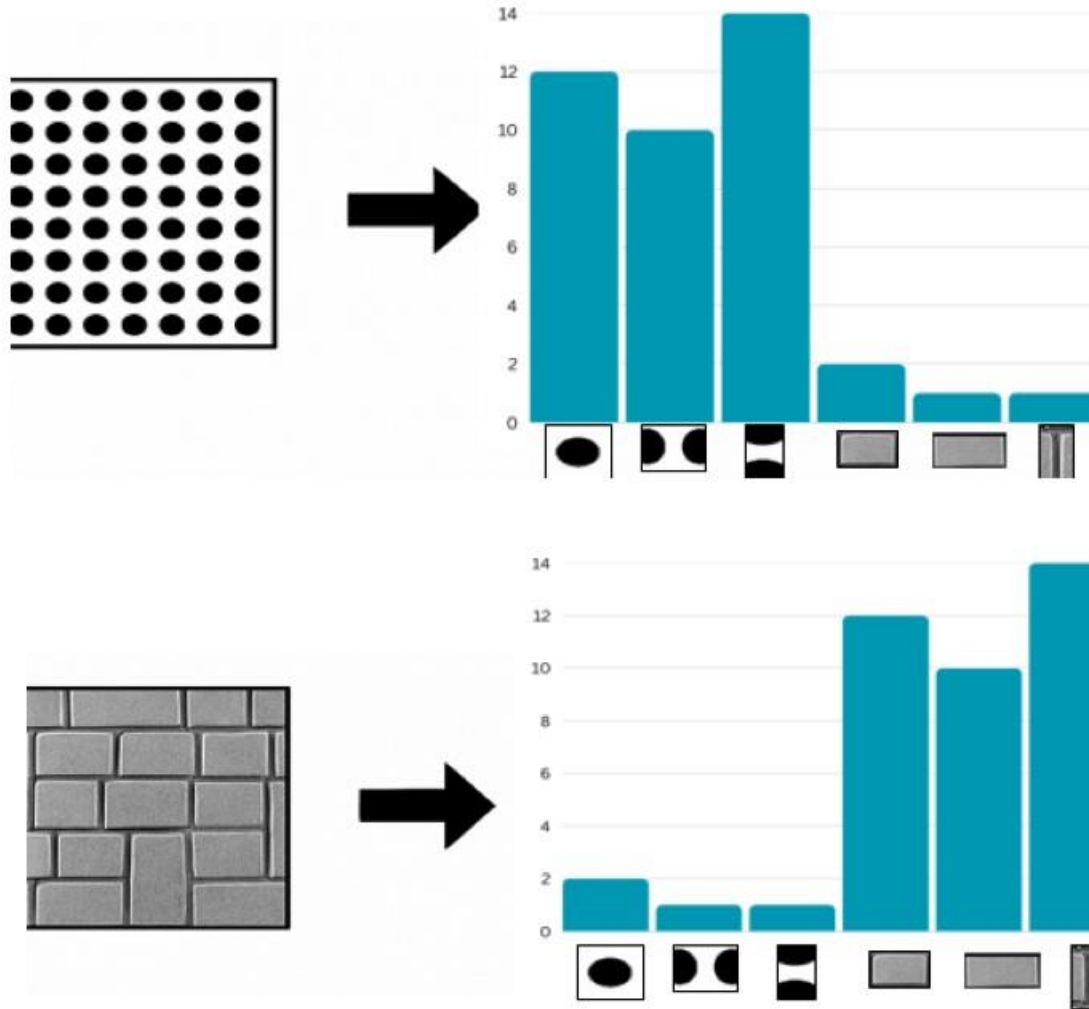


Bag of Visual Words

- **Bag of Features** (or Bag of Visual Words) is a framework to learn descriptors based on patches of data.
- It has the following general steps:
 - extract features locally
 - quantize or cluster the local features
 - learn a vocabulary of visual words: dictionary
 - obtain descriptors as the frequency of visual words in an image, i.e. histogram



Bag of Visual Words



Bag of Visual Words

Step 1 – Feature Extraction

- Detect keypoints in images and extract descriptors (e.g., SIFT, SURF).



Face



Cat



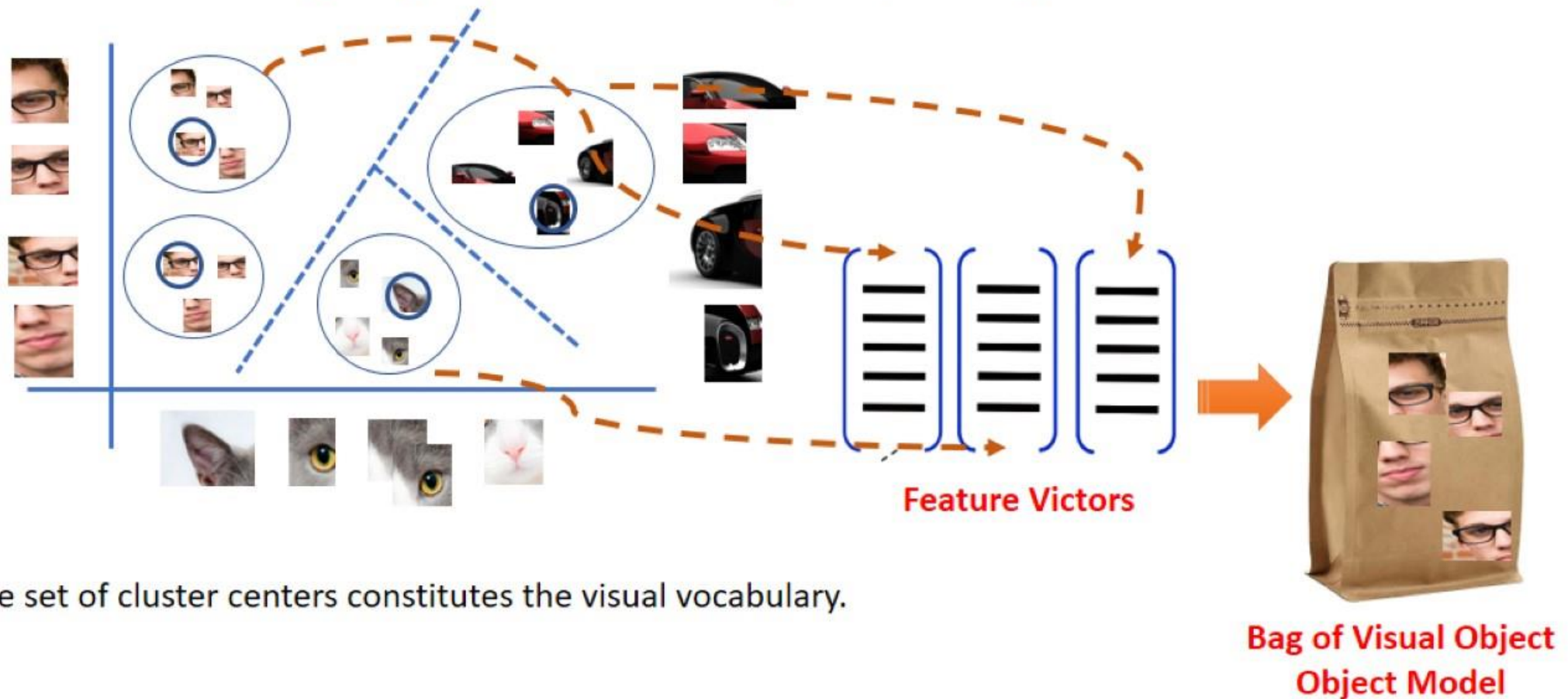
Car



Bag of Visual Words

Step 2 – Building the Visual Vocabulary

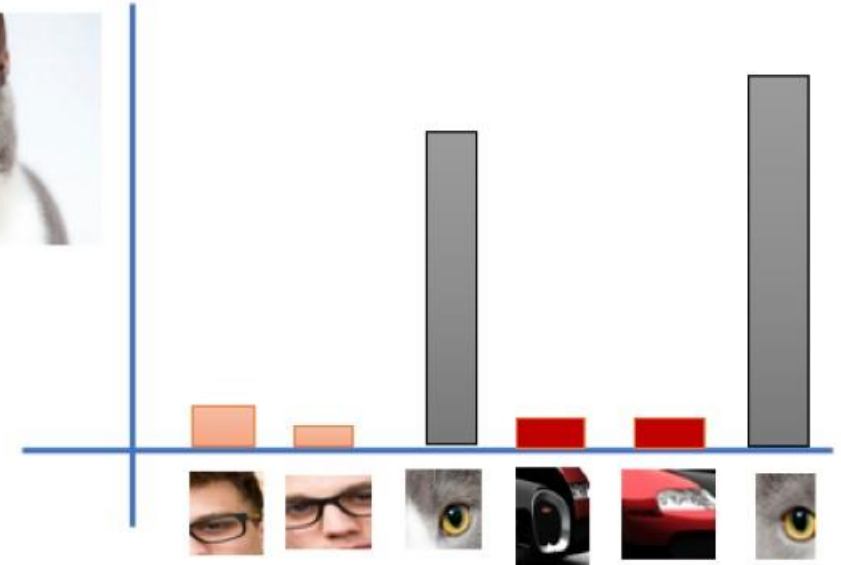
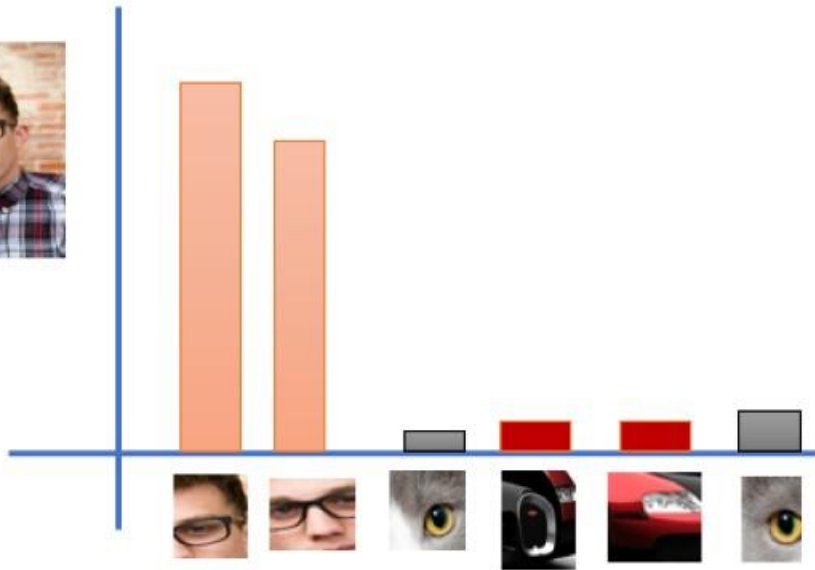
- Apply k-means clustering to group similar descriptors, forming 'visual words'.



Bag of Visual Words

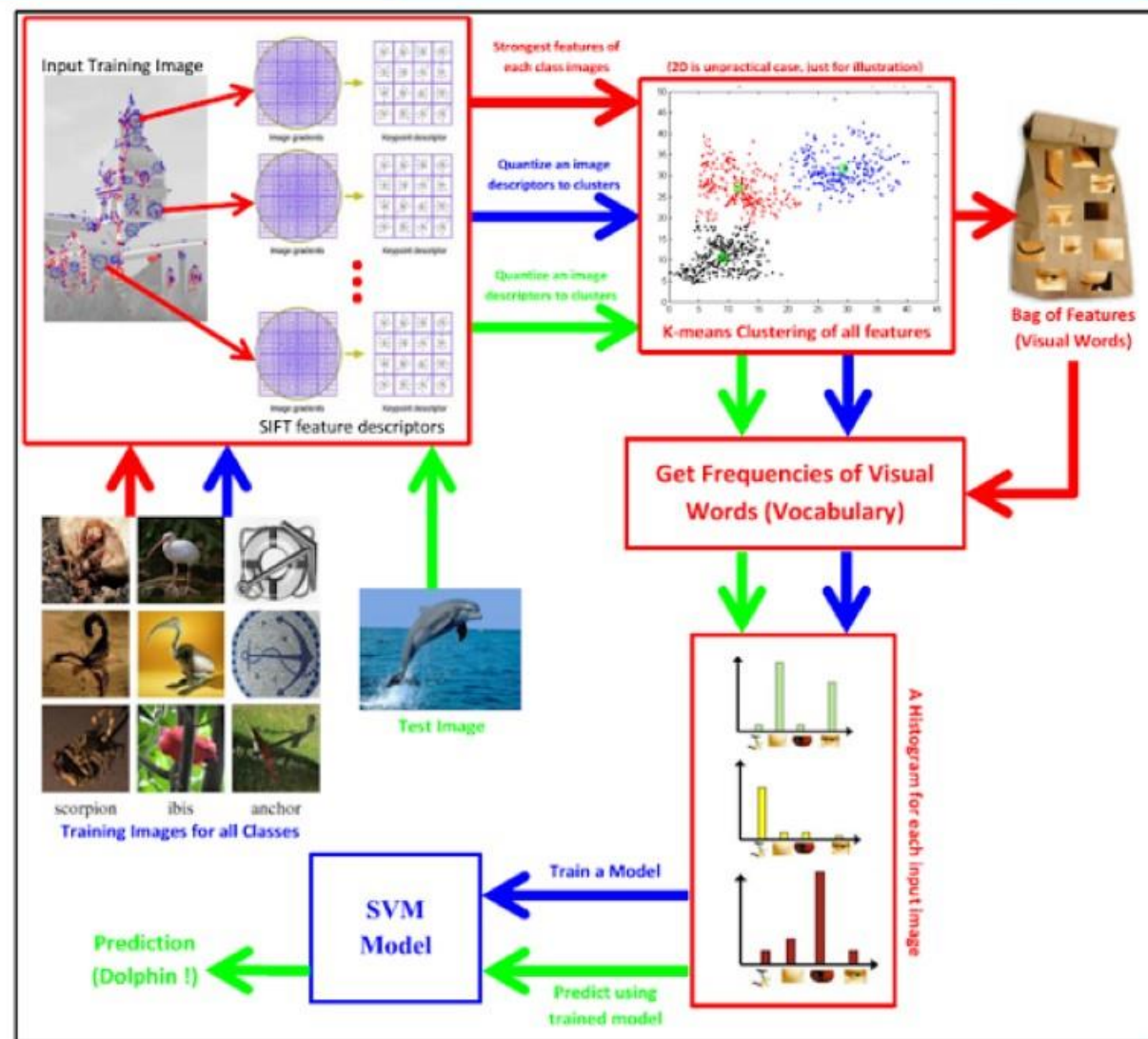
Step 3 – Image Representation as Histograms

- For each image, **count the frequency** of each visual word to create a histogram.



Application – Image Classification

- Use the histograms as feature vectors to train classifiers (e.g., SVM).



K-means Clustering Algorithm

Clustering means grouping.

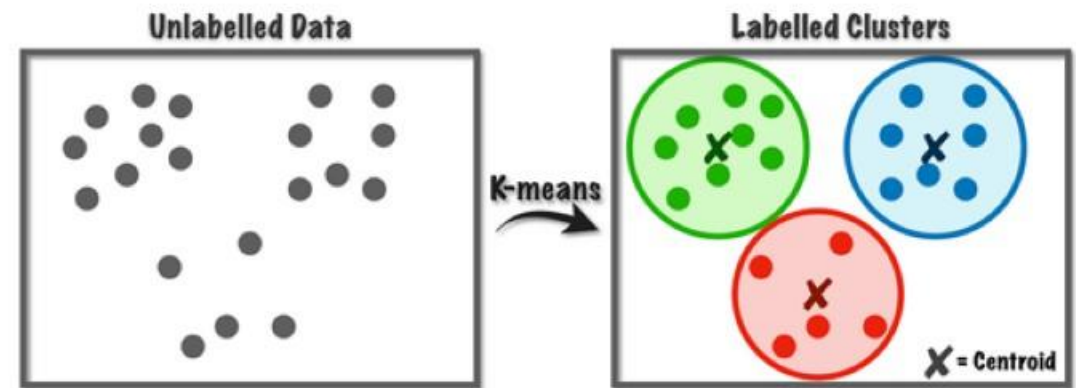
- K-means
 - One of the most popular clustering methods
 - An unsupervised learning method for clustering data points.
 - Ease of understanding
 - Easy implementation
 - Group data into K number of groups

K-means Clustering Algorithm

Clustering means grouping.

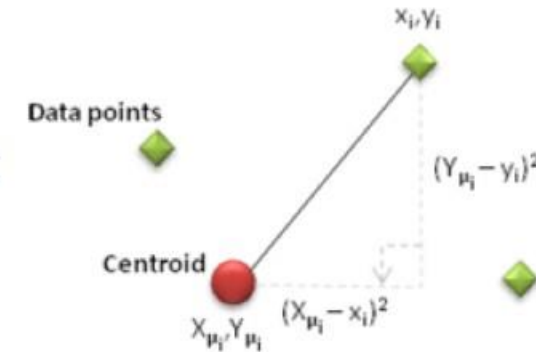
- K-means

- Determine a number of centroids we want to create to start creating clusters around those centroids
- A centroid is the center of a cluster
- Every data point is going to be assigned to the nearest centroid
- Reduce the within group sum of squares, or make the data within the groups as similar as possible
- Maximize the between groups sum of squares, or make the different clusters as different as possible



K-means Clustering Process

- Iterative process
- Randomly place the amount of centroids we have determined among the data Assign each data point to the nearest cluster
- Recalculate the centroids of clusters using some math
- Repeat the previous two steps until the model does not improve



x_i : data point
 μ_j : centroid of cluster j
 d : number of features
(dimensions)

$$\text{Distance}(x_i, \mu_j) = \sqrt{\sum_{d=1}^D (x_{id} - \mu_{jd})^2}$$

Used to assign each point to the nearest centroid:

K-means Clustering Process

- Update the centroid of each cluster to be the **mean of all points** in that cluster:

μ_j : updated centroid of cluster j

C_j : set of points assigned to cluster j

$|C_j|$: number of points in cluster j

$$\mu_j = \frac{1}{|C_j|} \sum_{x_i \in C_j} x_i$$

K-means Clustering Process

- K-Means minimizes the **Within-Cluster Sum of Squares (WCSS)**

$$J = \sum_{j=1}^K \sum_{x_i \in C_j} \|x_i - \mu_j\|^2$$

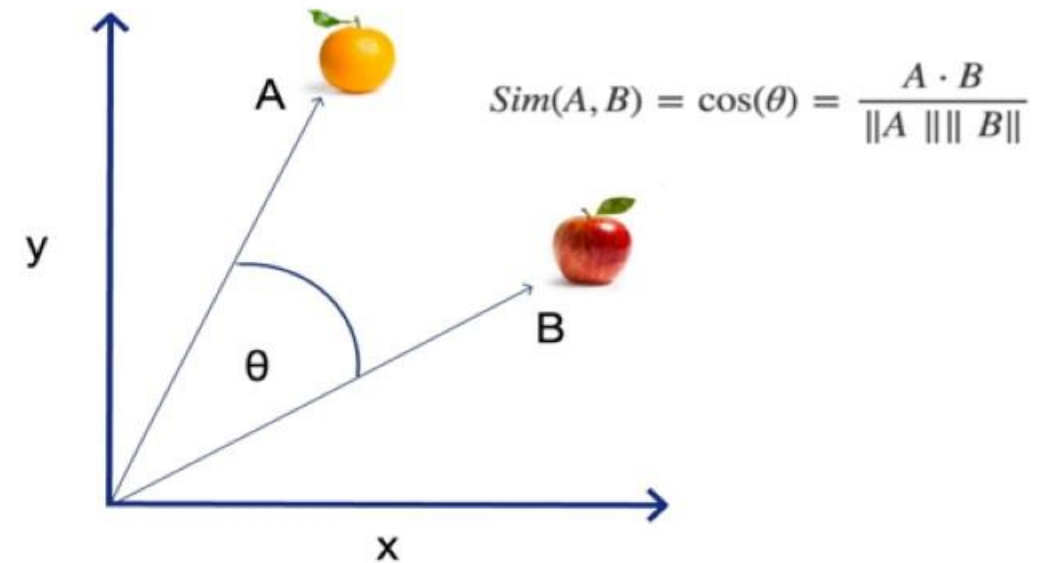
This is the total variance within all clusters.

Comparing Between Vectors

- Using **Cosine Similarity**

- It is a metric used to determine how similar two vectors are by calculating the cosine of the angle between them.
- It ranges from -1 (completely dissimilar) to 1 (identical), with 0 indicating orthogonality (no similarity).
- This measure is particularly useful in high-dimensional spaces, where Euclidean distance can be misleading.

Cosine Similarity



- $\vec{A} \cdot \vec{B}$ is the dot product of vectors A and B.
- $\|\vec{A}\|$ and $\|\vec{B}\|$ are the magnitudes (lengths) of vectors A and B.

Object Tracking

Tracking by Feature Detection

Object Tracking

• Object Tracking

- It is the task of taking an initial set of object detections.
- Create a unique ID for each of the initial detections.
- Then tracking each of the objects as they move around frames in a video, maintaining the ID assignment.



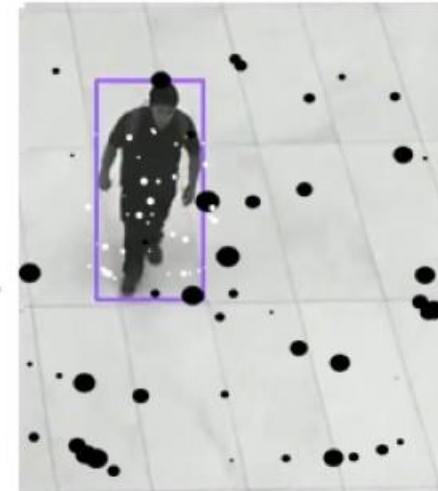
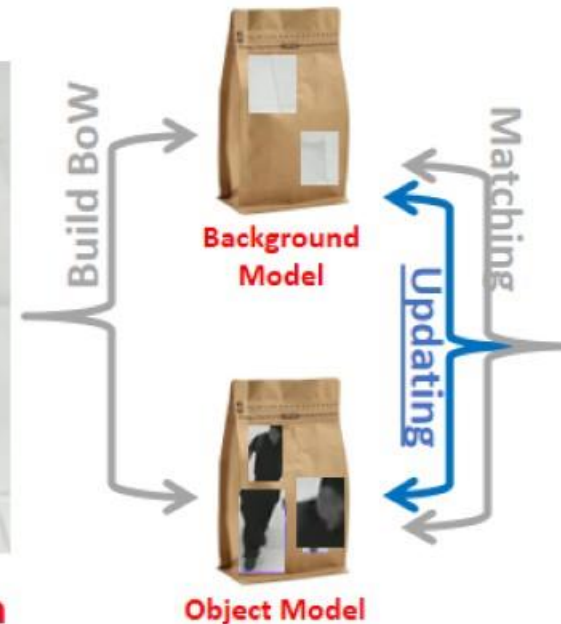
Fram1



Fram1
Bounding Box



Apply SIFT Algorithm

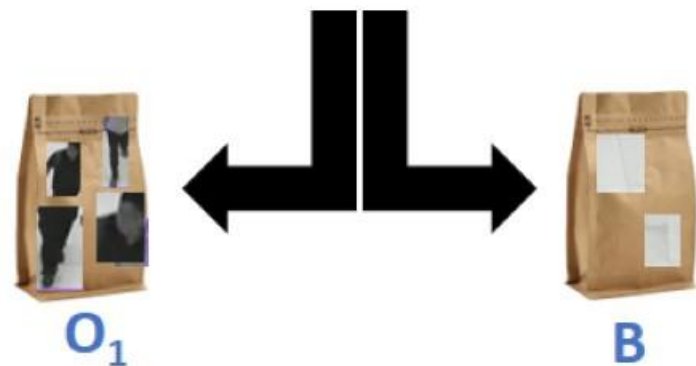


Frame 2

Object Tracking Initialization

At frame 1:

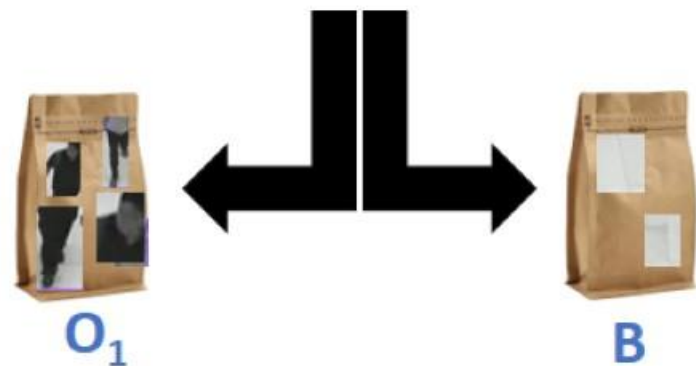
- User selects a bounding box W_1 as object/target.
- Compute SIFT (or similar) features for the frame.
- Classify features within the box as object and assign them to set O_1 .
- Classify remaining features as background and assign them to set B .



Object Tracking Initialization

At frame 1:

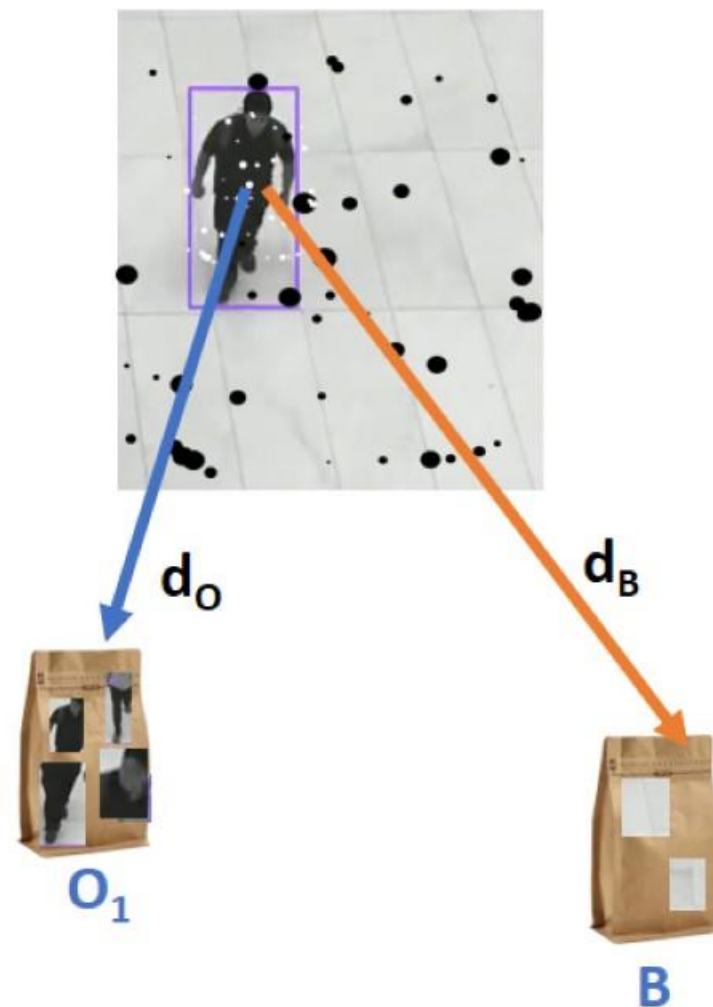
- User selects a bounding box W_1 as object/target.
- Compute SIFT (or similar) features for the frame.
- Classify features within the box as object and assign them to set O_1 .
- Classify remaining features as background and assign them to set B .



At frame t :

- Compute SIFT features and SIFT descriptors $\{v_1, \dots, v_K\}$ for frame I_t .
- For each feature and corresponding descriptor v_i :
 - a. Compute distance d_O between v_i and the best match in object set O_{t-1} .
 - b. Compute distance d_B between v_i and the best match in background set B .

$$C(v_i) = \begin{cases} +1 & \text{if } d_O/d_B < 0.5 \quad (v_i \text{ may belong to object}) \\ -1 & \text{otherwise } (v_i \text{ does not belong to object}) \end{cases}$$



For each Search Window W :

- a.** Compute $\phi(W) = \sum C(v_i)$ for all features v_i inside W .
- b.** Compute a heuristic $\tau(W, W_{t-1})$ that penalizes large deviations from previous location, size and shape W_{t-1} .

c. Compute **Match Score**

$$\mu(W) = \phi(W) - \tau(W, W_{t-1})$$

- Select window W_t with the best match score as new object location.
- Update object appearance model:
 $O_t = O_{t-1} \cup \{v_i\} \forall v_i \text{ inside } W_t \text{ such that } C(v_i) = +1$



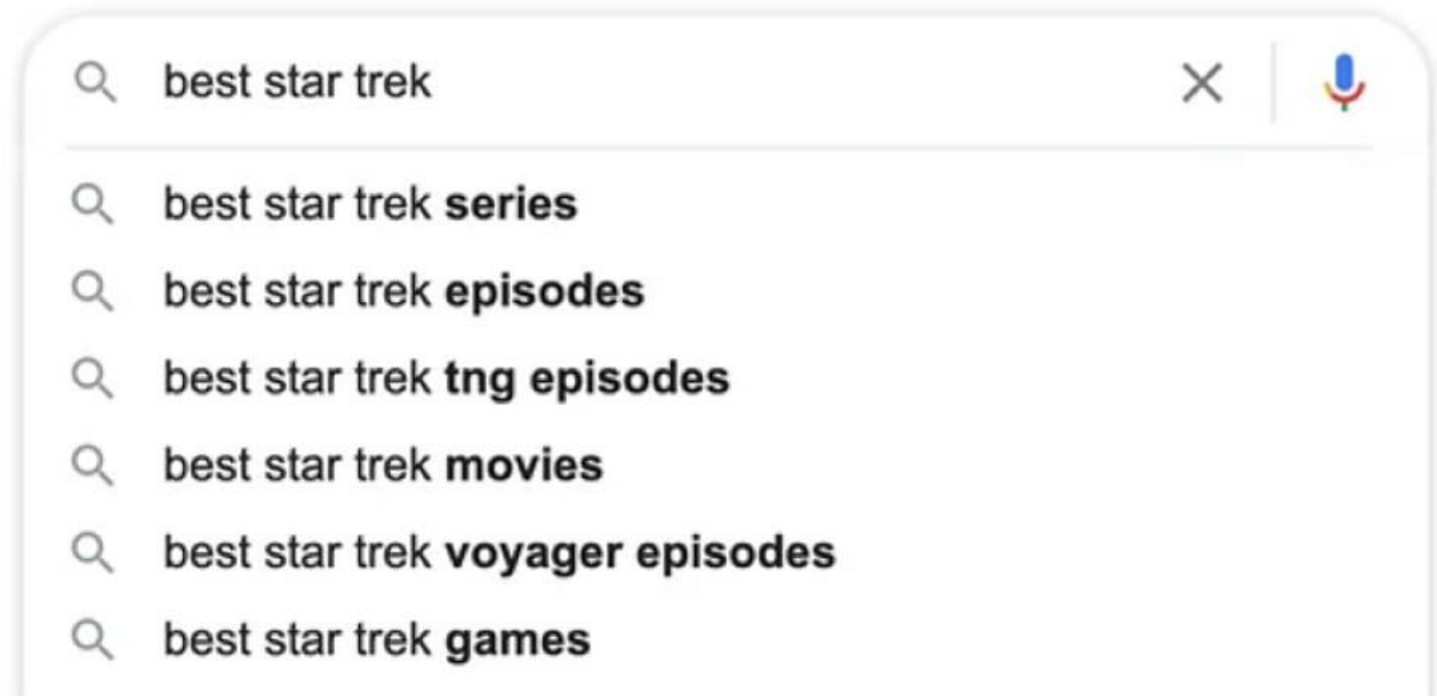
Object Tracking Example



What is Large Language Models?

What is a Language Models?

- The ability to predict the next word in a sentence.



What is **Large Language Models**?

A type of artificial intelligence designed to understand, generate, and manipulate human-like text.



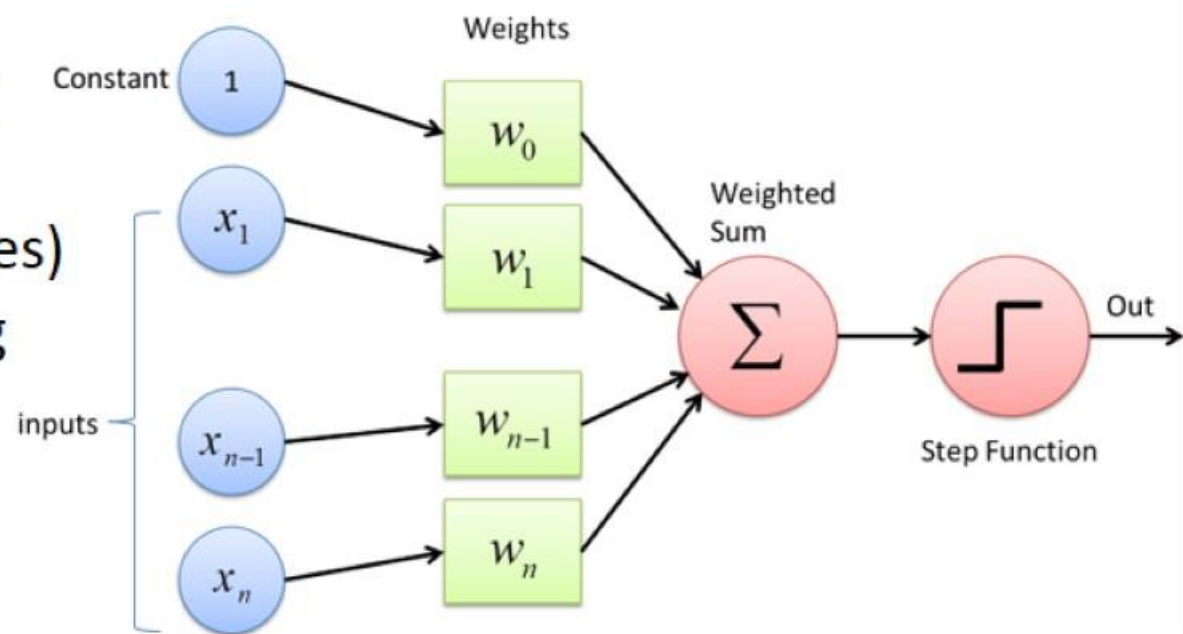
**Large Language
Model (LLM)**

What is Large Language Models?

- **Large**

Refers to the **massive size** of the model in terms of:

- **Parameters** (often billions or even trillions)
- **Training data** (typically sourced from huge amounts of text like books, articles, websites)
- **Computational power** required for training and inference



What is Large Language Models?

- **Language**

Indicates that the model is designed to work with **natural language**:

- Understanding text
- Generating human-like sentences
- Translating languages
- Answering questions, summarizing, and more

What is **Large Language Models**?

- **Model**

A **mathematical representation** trained using machine learning techniques (especially deep learning) to **learn patterns in data** and make predictions based on it.

In brief an LLM is:

A **deep learning model** trained on **large-scale language data** to perform **tasks** like text generation, understanding, translation, summarization, and more—mimicking human-like language abilities.

How LLMs Work?

Attention Is All You Need

Attention Is All You Need

Ashish Vaswani*
Google Brain
avaswani@google.com

Noam Shazeer*
Google Brain
noam@google.com

Niki Parmar*
Google Research
nikip@google.com

Jakob Uszkoreit*
Google Research
usz@google.com

Llion Jones*
Google Research
llion@google.com

Aidan N. Gomez* †
University of Toronto
aidan@cs.toronto.edu

Łukasz Kaiser*
Google Brain
lukaszkaiser@google.com

Illia Polosukhin* †
illia.polosukhin@gmail.com

How LLMs Work?

Attention Is All You Need

- Main Concepts:
 - Word Embedding
 - Attention Mechanism
 - Transformer

- Word embedding represent words as numbers (vectors)

love $\rightarrow$ [0.8, -0.1, 0.9, 0.3]

hate $\rightarrow$ [-0.8, 0.1, -0.7, -0.2]

banana $\rightarrow$ [0.1, 0.9, 0.2, 0.7]

carrot $\rightarrow$ [0.0, 0.8, 0.3, 0.6]

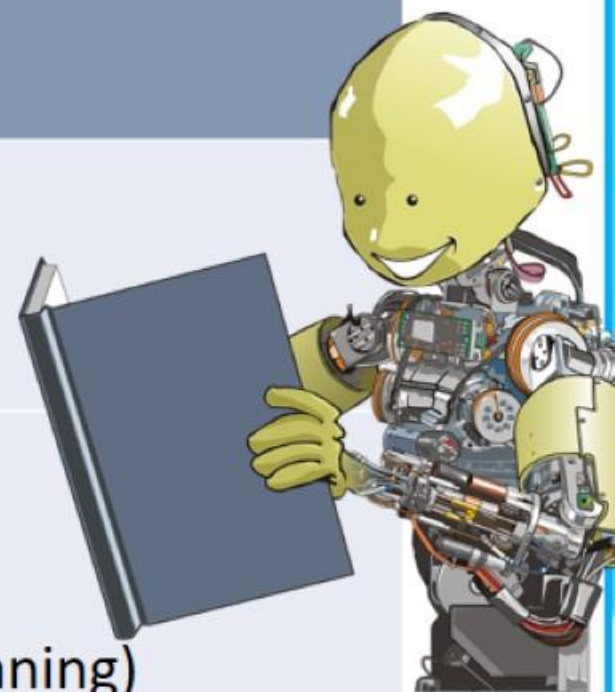
- Methods for the word embedding:
 - Frequency-Based Word Embedding
 - Prediction-Based Embedding.
 - **Contextual-Based Embedding**
 - A word's vector changes depending on the sentence it's in — based on **context**.

Word in Sentence	Static Embedding (Word2Vec/GloVe)
" bank " in: "I sat by the bank of the river."	
" bank " in: "He deposited money in the bank ."	

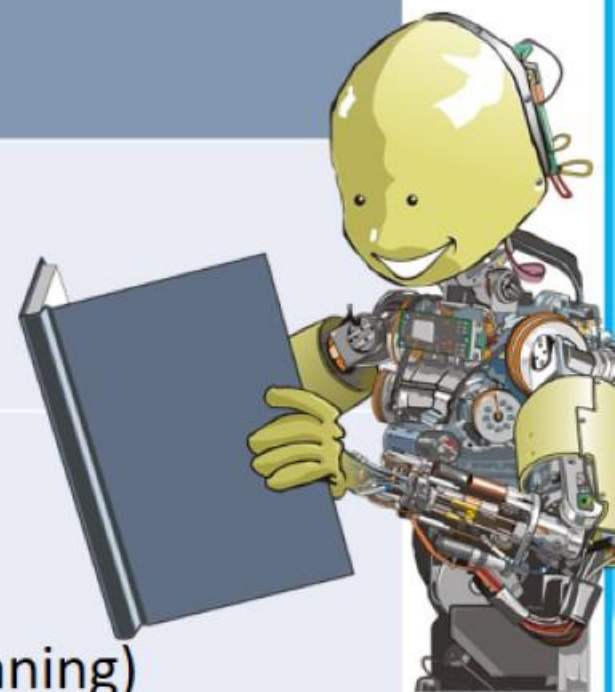
Word in Sentence	Static Embedding (Word2Vec/GloVe)
" bank " in: "I sat by the bank of the river."	[0.8, -0.5, 0.2, ...]
" bank " in: "He deposited money in the bank ."	[0.8, -0.5, 0.2, ...]

Word in Sentence	Static Embedding (Word2Vec/GloVe)
" bank " in: "I sat by the bank of the river."	[0.8, -0.5, 0.2, ...] <u>(Same vector)</u>
" bank " in: "He deposited money in the bank ."	[0.8, -0.5, 0.2, ...] <u>(Same vector)</u>

Word in Sentence	Contextual Embedding
" bank " in: "I sat by the bank of the river."	[0.3, -0.9, 0.4, ...] (Riverbank meaning)
" bank " in: "He deposited money in the bank ."	[-0.2, 0.7, 0.1, ...] (Financial institution meaning)



Word in Sentence	Contextual Embedding
" bank " in: "I sat by the bank of the river ."	[0.3, -0.9, 0.4, ...] (Riverbank meaning)
" bank " in: "He deposited money in the bank ."	[-0.2, 0.7, 0.1, ...] (Financial institution meaning)



Thanks to new technology, going to school these days can be a pleasant experience. No longer having to rely on 'chalk and talk', teachers can give pupils the opportunity to play a more active role in their own learning. This makes learning a more individual experience and students with different learning styles can take in knowledge more easily. This can motivate students [53].

Nowadays students can learn individually or in groups to improve their skills. Using technology is said to develop students' critical thinking, cooperative learning [54]. Students can also learn at their own pace with the use of computers.

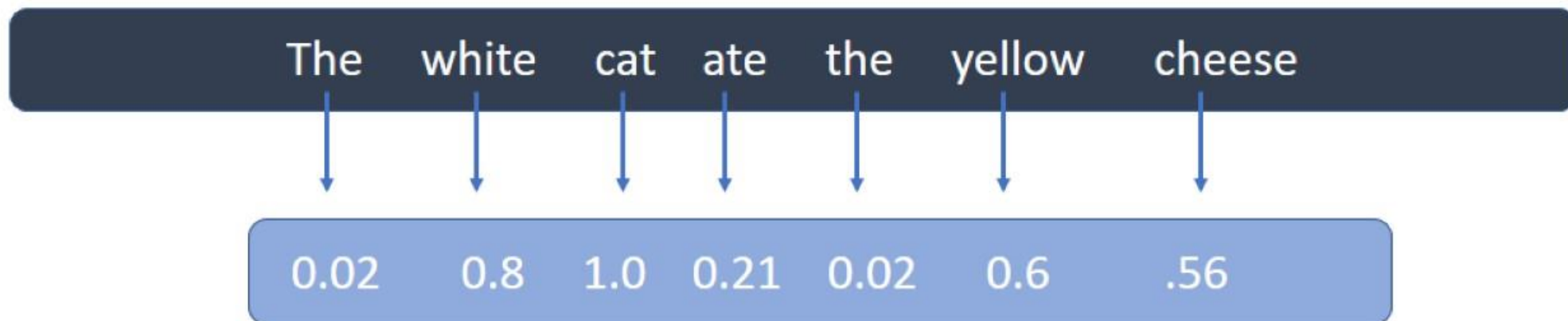
However, not all schools are taking full advantage of these opportunities. Sometimes schools cannot afford to buy the equipment [55].

In other cases, schools have the resources [56]. Schools need to decide which types of technology can improve teaching to make learning [57]. In addition, teachers need to become familiar with new products, software and on-line resources [58]. Only in this way will schools be able to function effectively in the future.



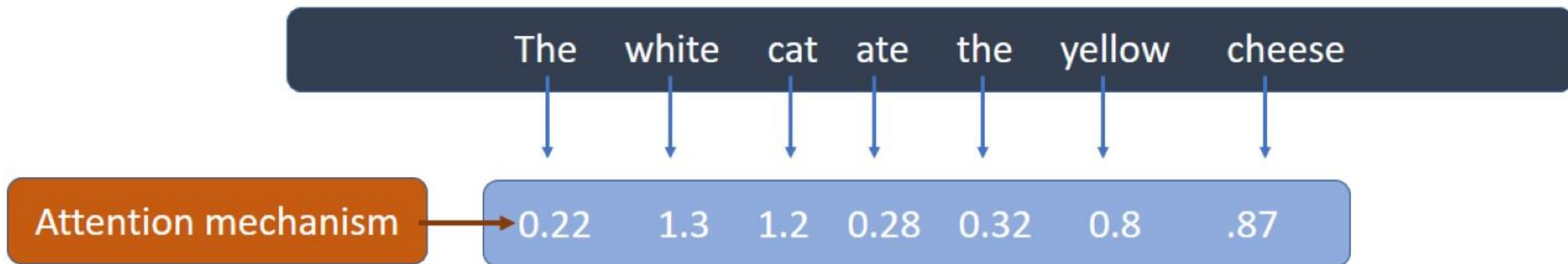
Without Attention Mechanism

- Assign weight to the inputs



Attention Mechanism

- Assign weight to the inputs



- To improve the Text Translation

The white cat ate the yellow cheese

Encoder - Decoder

القطّة البيضاء أكلت الجبن الأصفر

Attention Mechanism

- To improve the Text Translation

Computes **Query (Q)**, **Key (K)**, and **Value (V)** vectors for each word. **Attention(Q,K,V)**

Attention mechanism

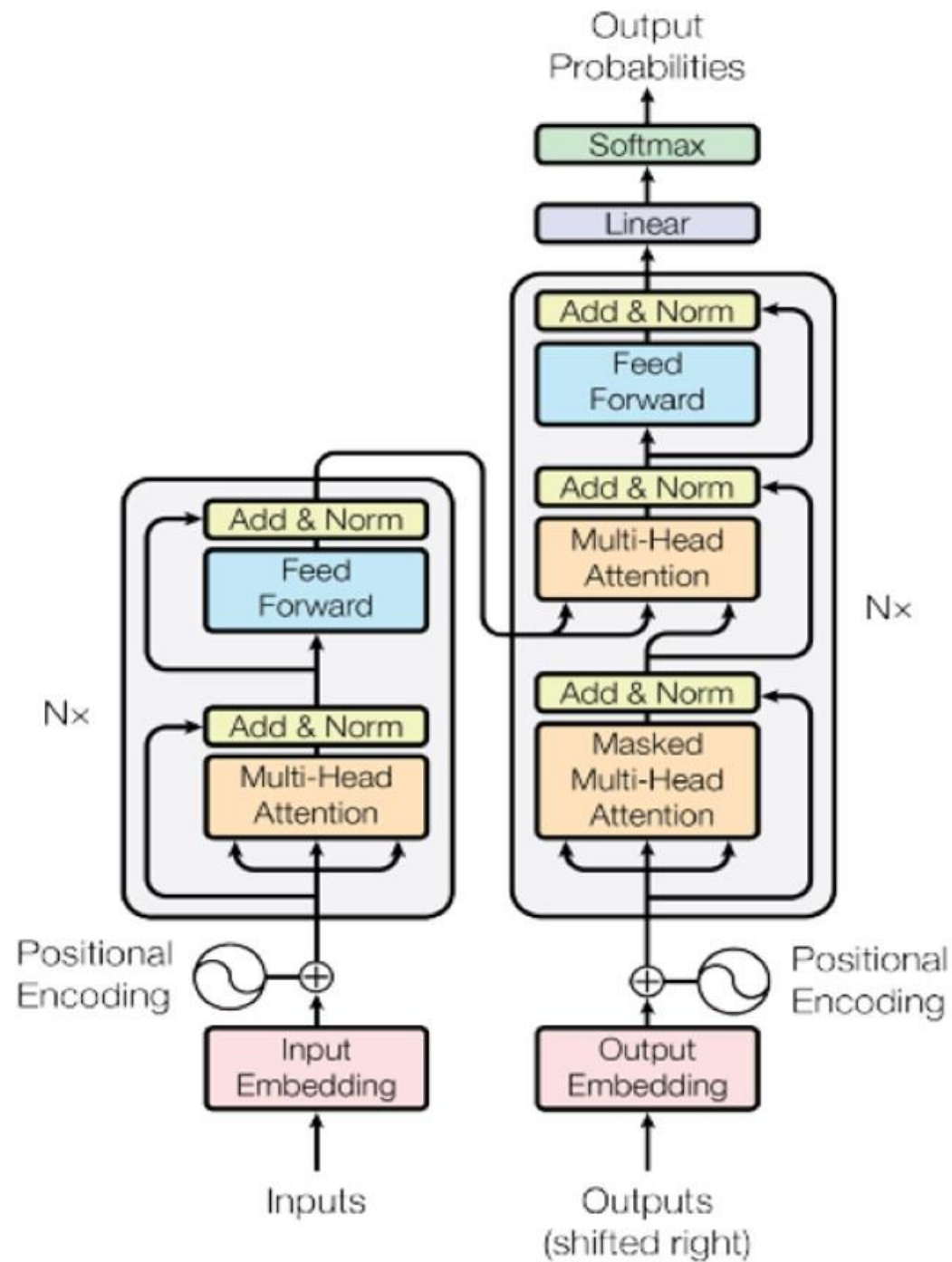
The white cat ate the yellow cheese

Encoder - Decoder

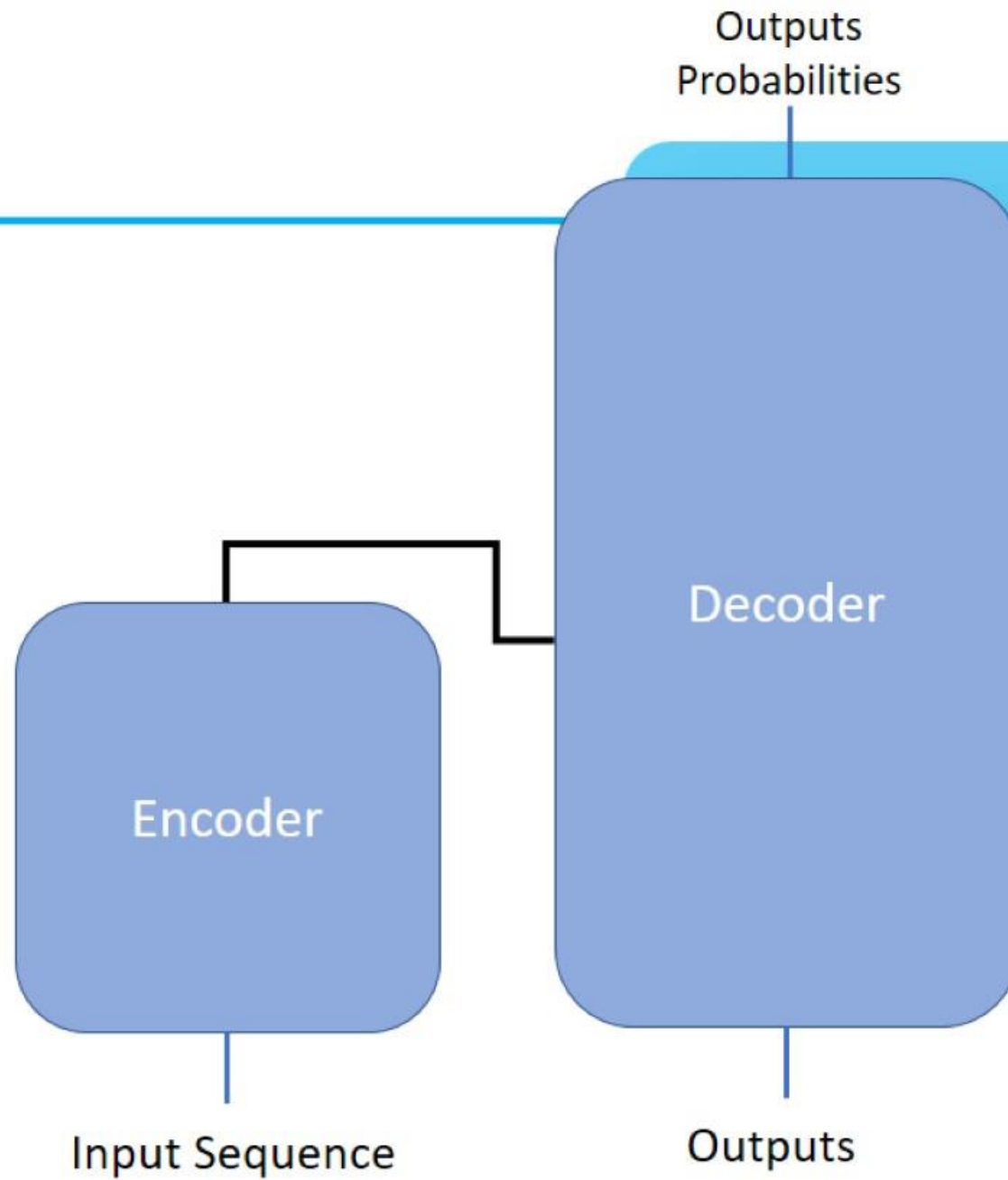
Multi-head Attention

القطّة البيضاء أكلت الجبن الصفراء

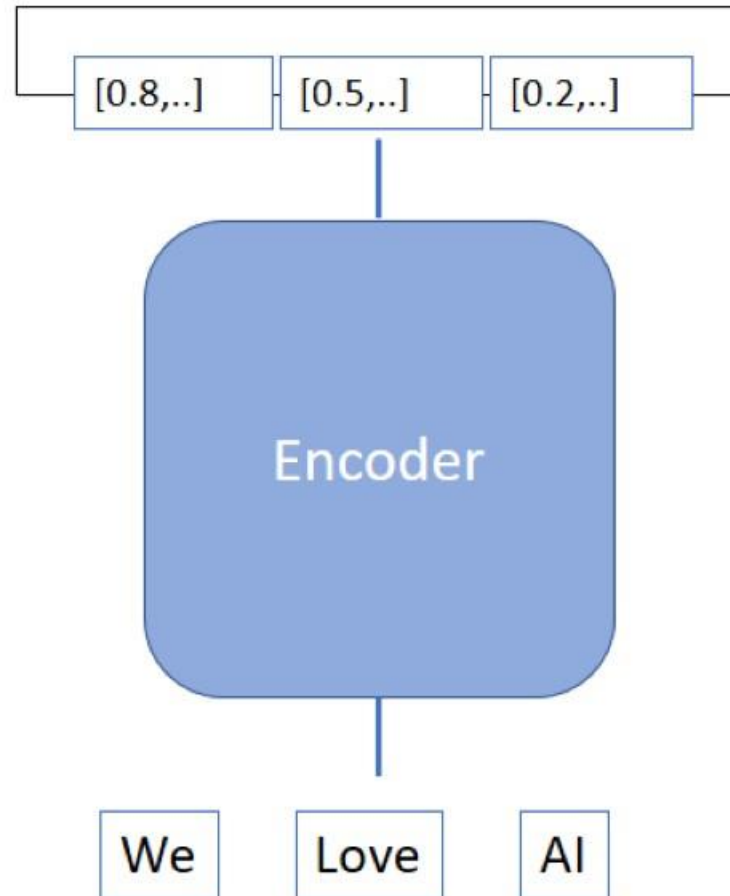
Transformer



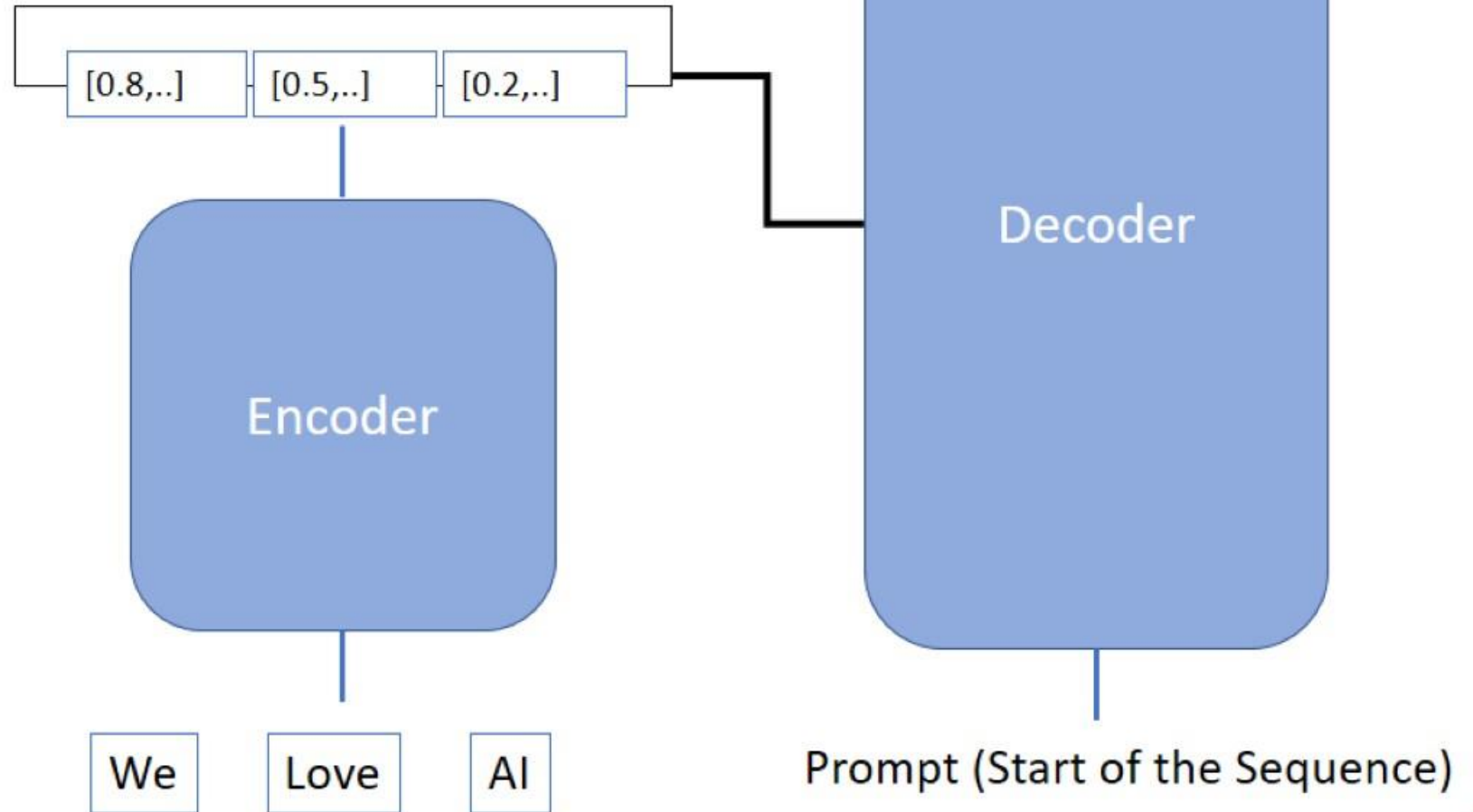
Transformer



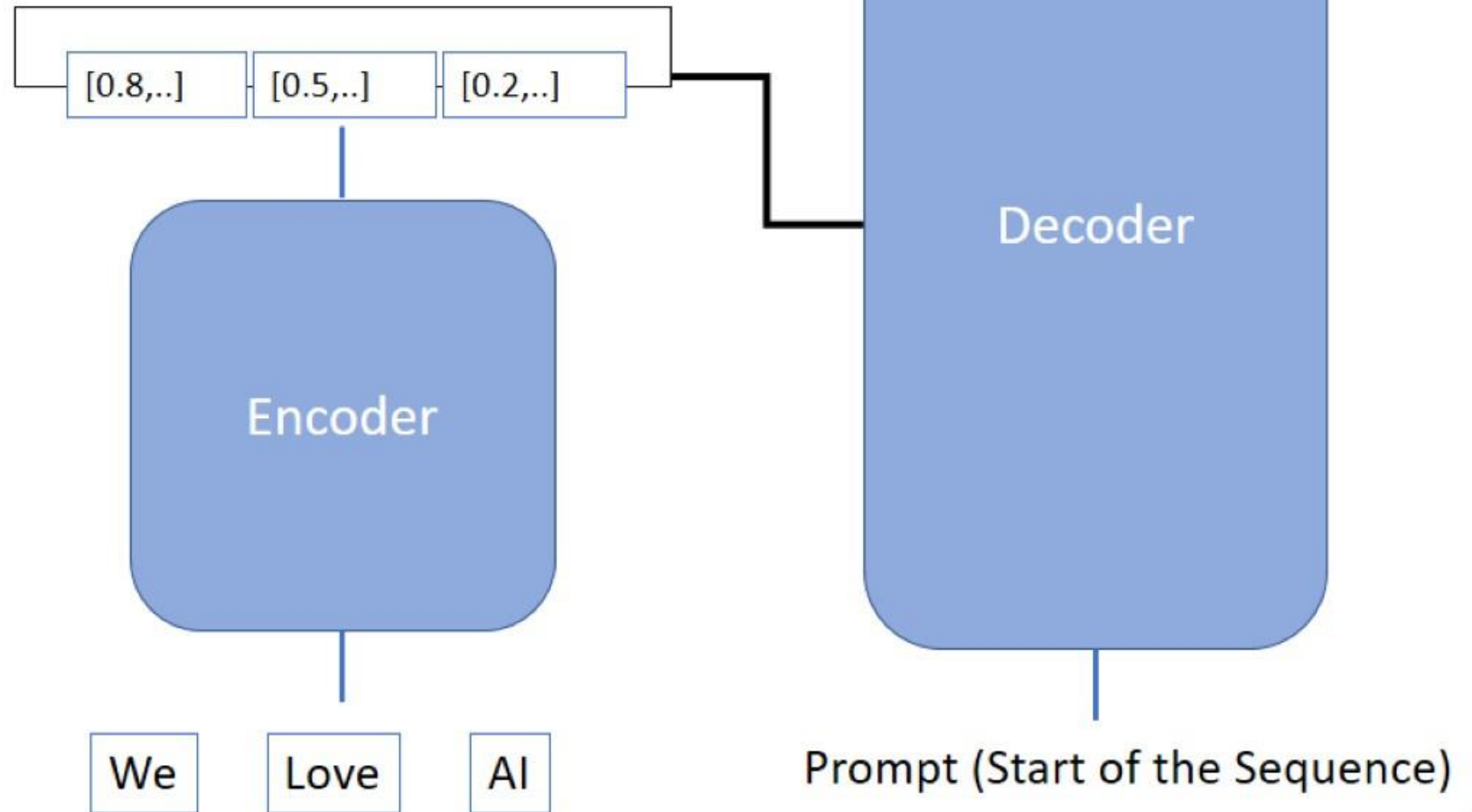
Transformer



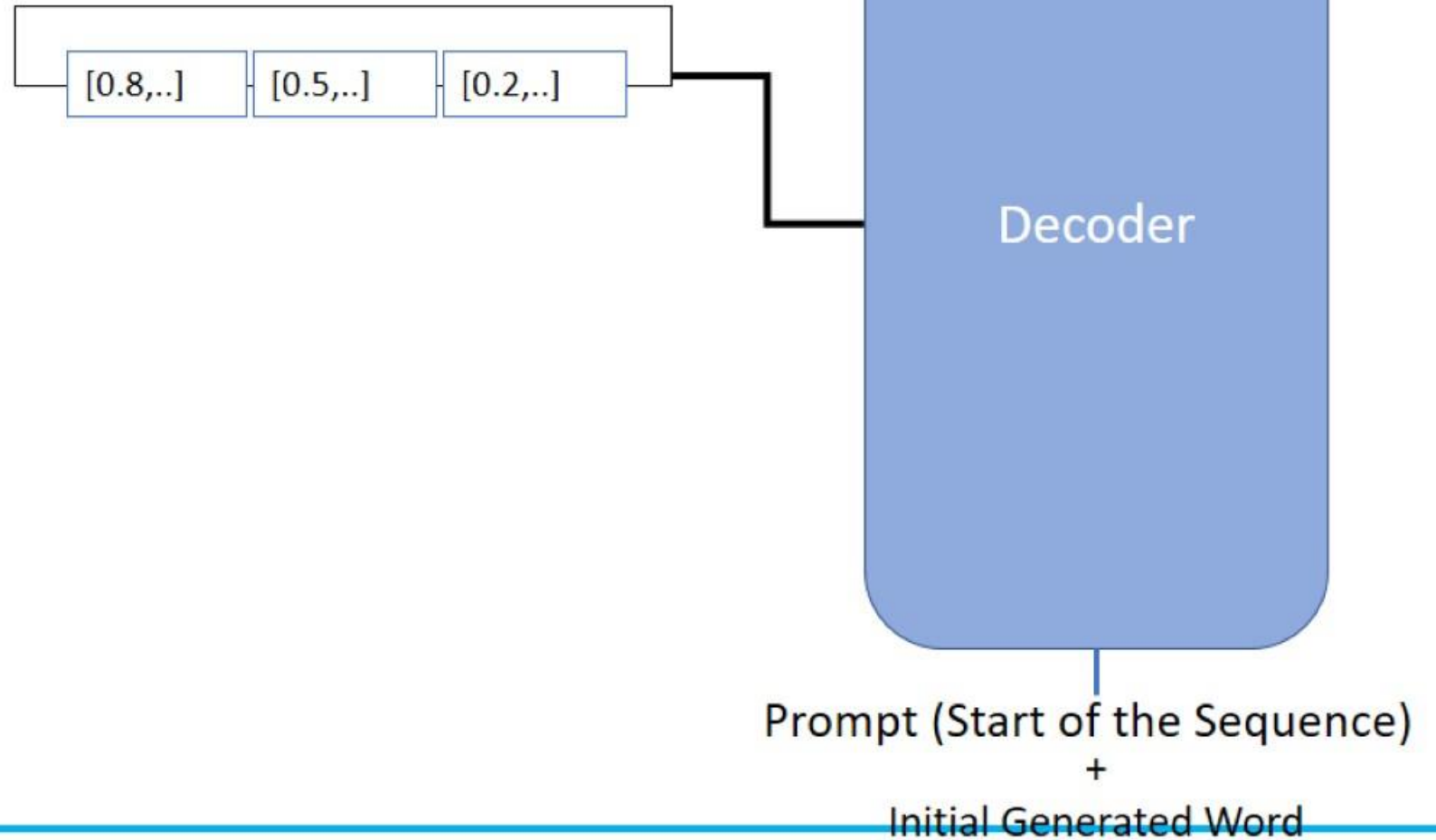
Transformer



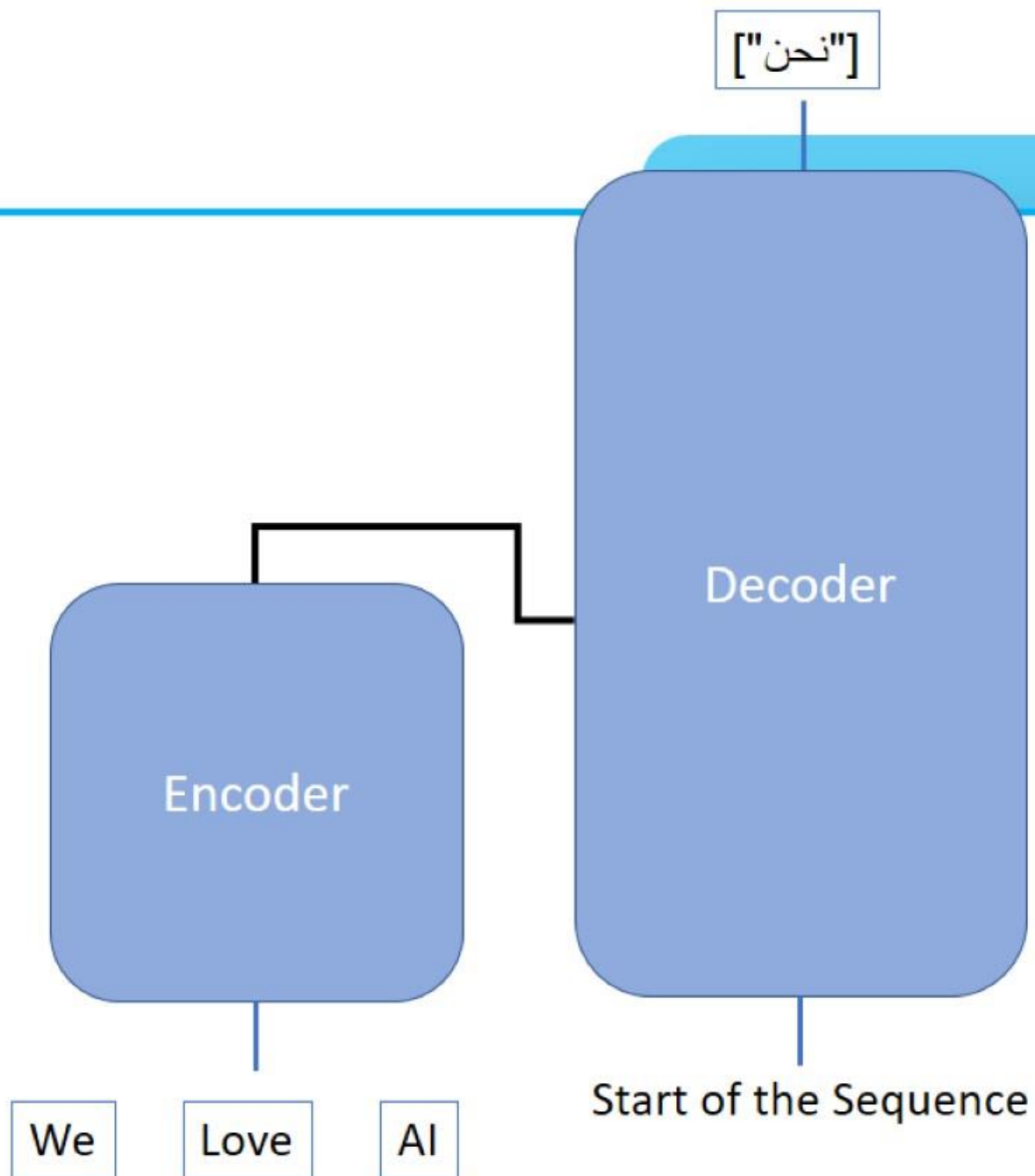
Transformer



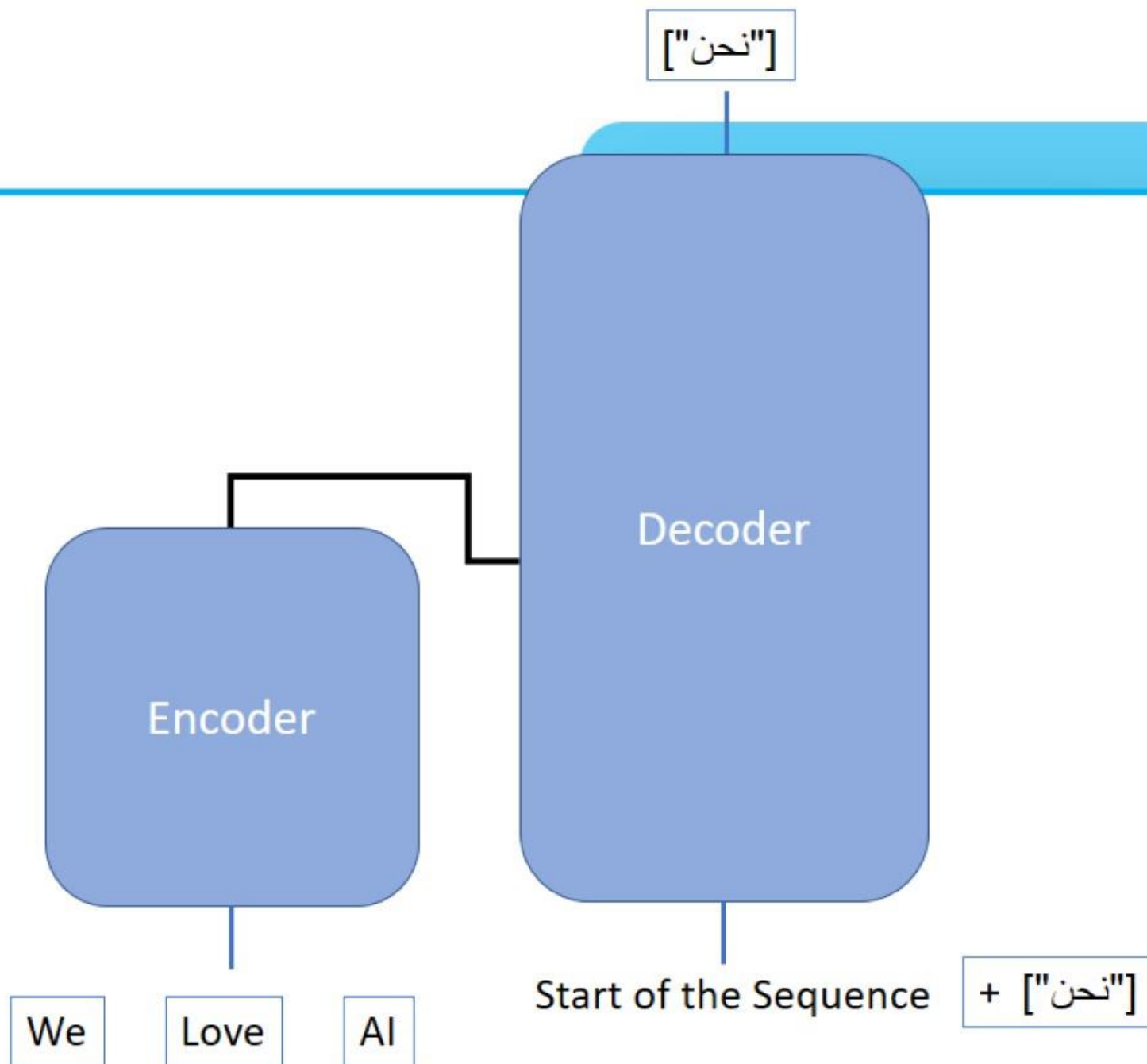
Transformer



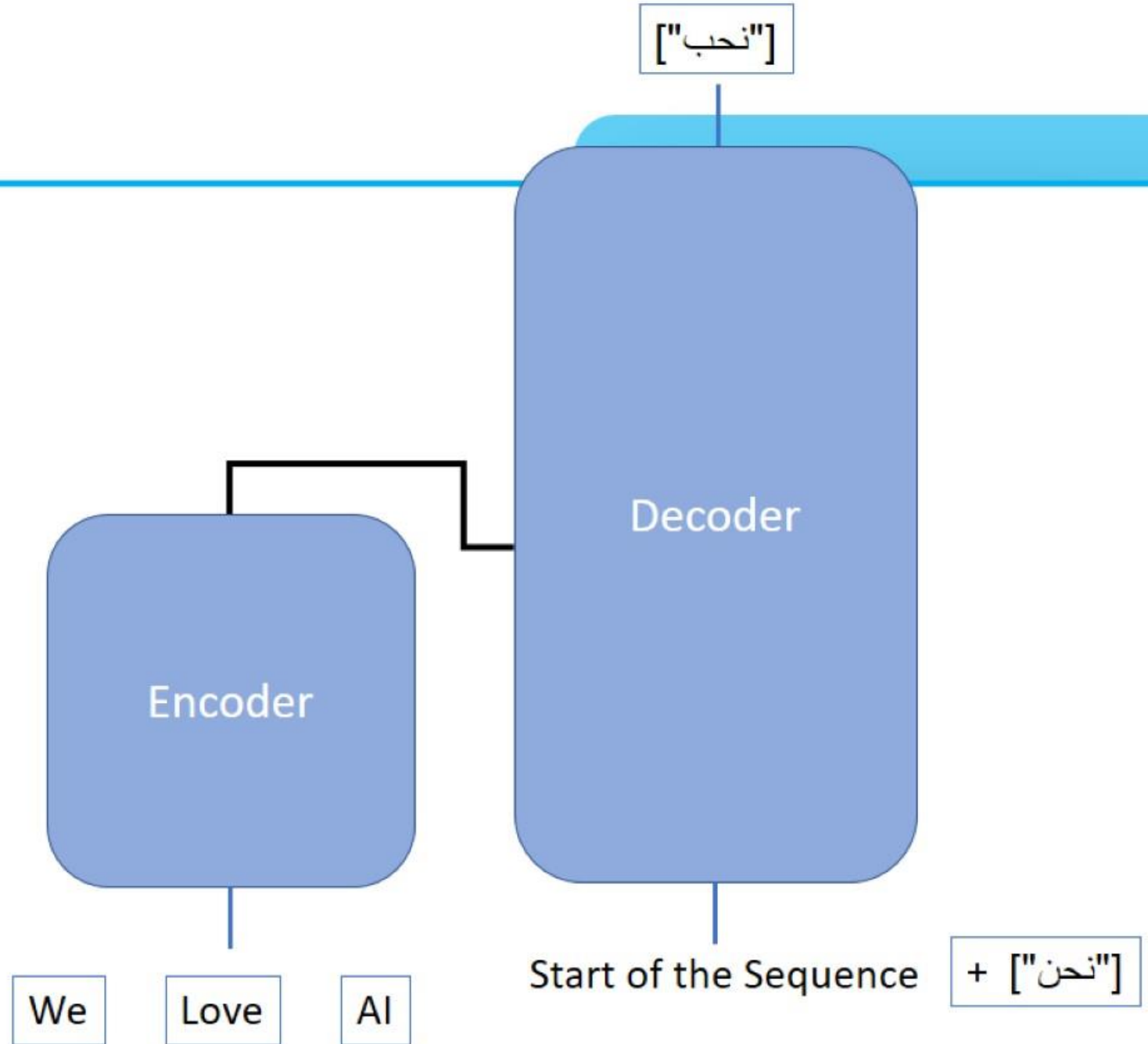
Transformer



Transformer

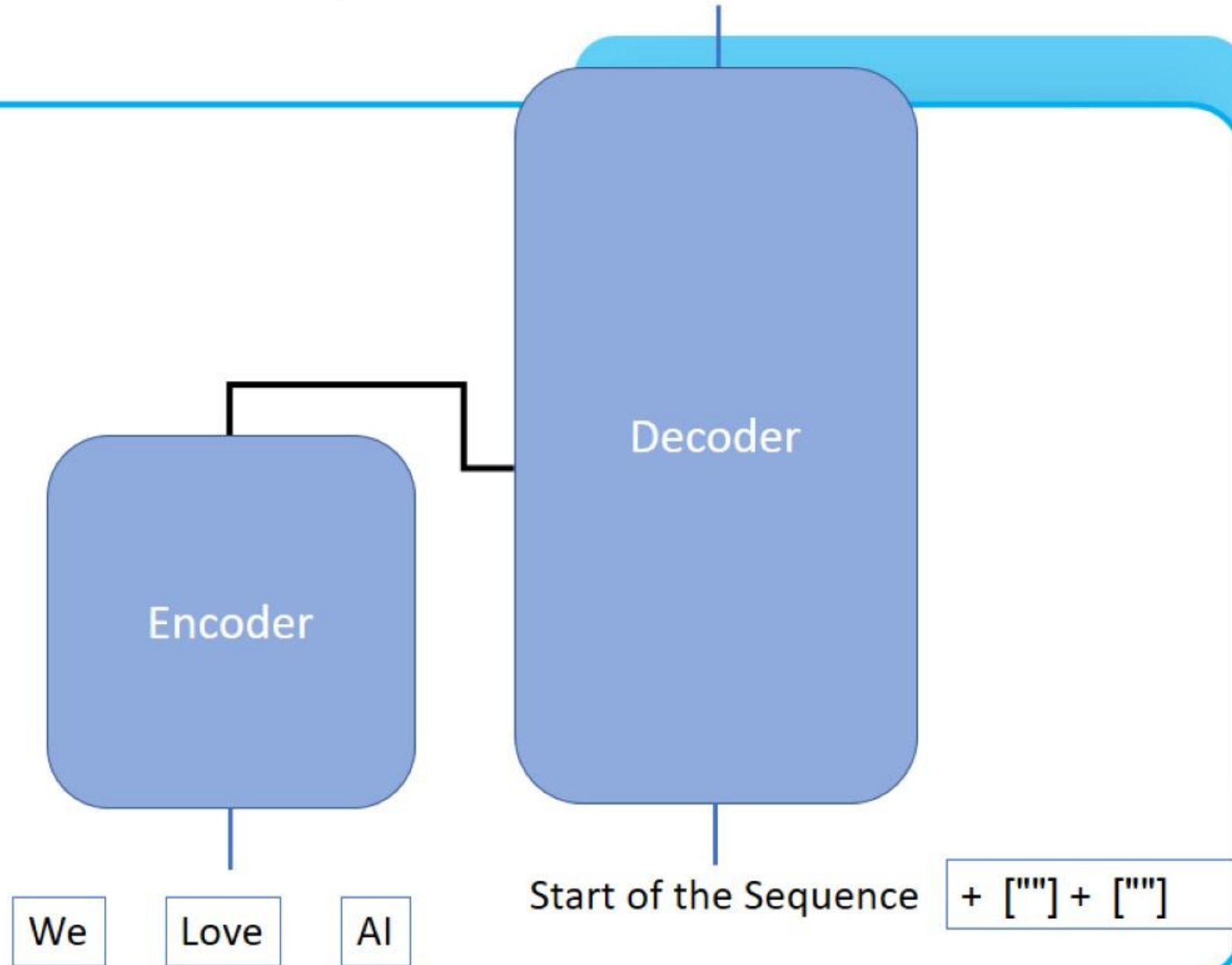


Transformer

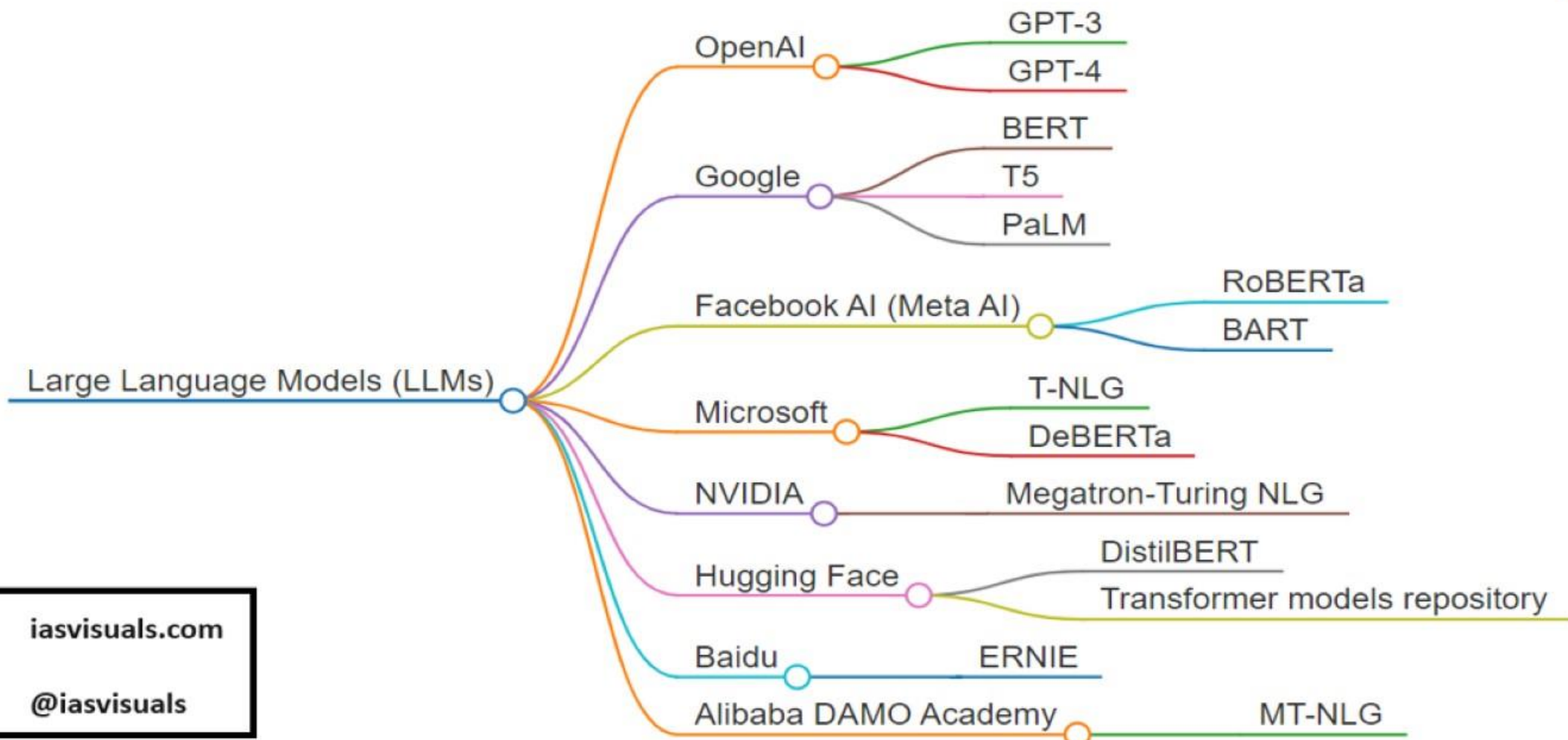


[نحن, "نحب", "ال", "ذكاء", "ال", "اصطناعي"]

Transformer



List of important Large Language Models



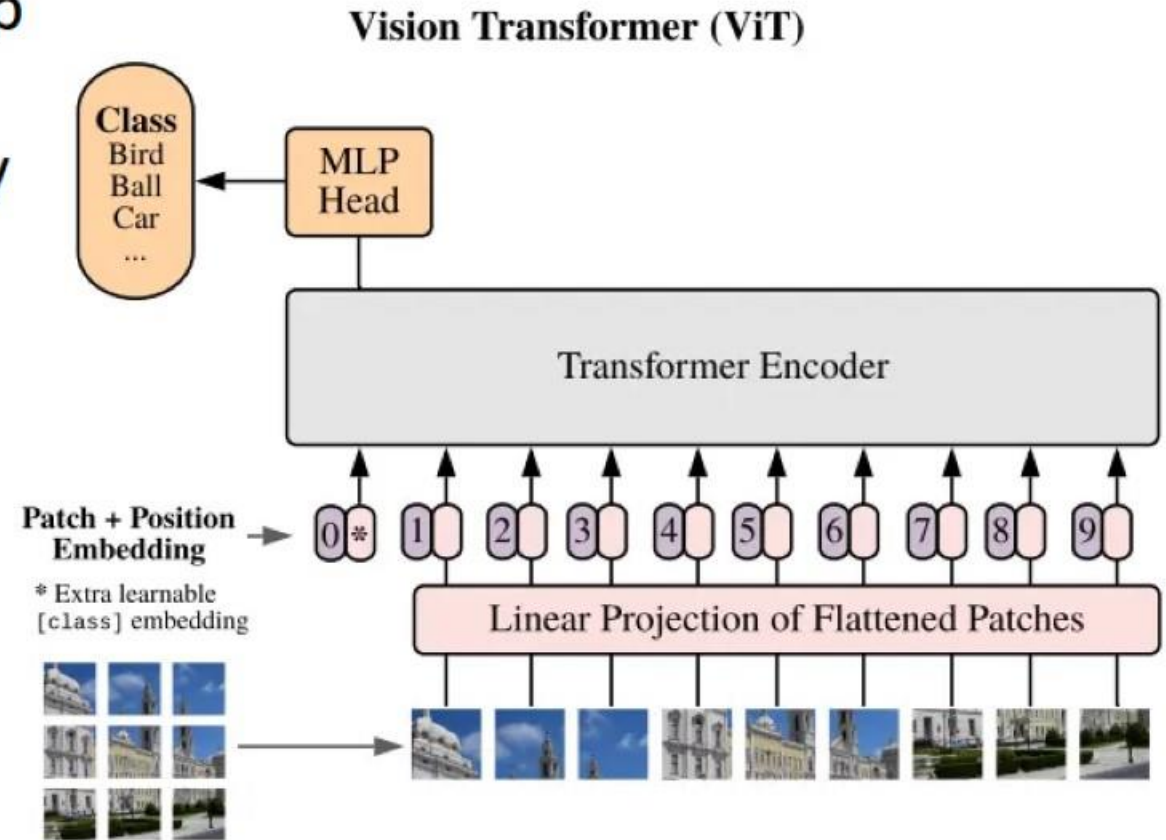
iasvisuals.com

@iasvisuals

A Vision Transformer (ViT): A Transformer for computer vision tasks

A Vision Transformer (ViT)

- A Vision Transformer (ViT) is a deep learning model that applies the transformer architecture, originally used for natural language processing, to computer vision tasks.





Thank You