**Haarcascade Classifier**

**1️⃣ What is a Haar Cascade Classifier?**

* It’s a **machine learning-based** object detection method.
* Mainly used for **face detection** (but can detect eyes, license plates, hands, etc.).
* It works by scanning an image with **many small features** (Haar-like features) and classifying whether a region contains the object of interest.

**2️⃣ The Name**

* **Haar** → From **Haar wavelets** (simple patterns like edges, lines, rectangles).
* **Cascade** → The classifier is a chain of stages, each stage filters out regions that definitely aren’t the object.
  + Early stages are **fast** and remove most negatives.
  + Later stages are **more detailed** and confirm positives.

**3️⃣ How it Works**

1. **Haar-like features**
   * Small patterns of black and white rectangles that detect edges, textures, and shapes.
   * Example: The eye region is darker than cheeks, so a Haar feature can capture that.
2. **Integral image**
   * A fast way to sum pixel values in a rectangle — speeds up detection.
3. **Training** (done beforehand by OpenCV team)
   * A lot of positive and negative images are used.
   * AdaBoost algorithm picks the best features and creates a series of classifiers.
4. **Cascade stages**
   * Stage 1: Quick reject of most regions.
   * Stage 2–N: More complex checks until the object is confirmed.

**4️⃣ Using in OpenCV**

OpenCV provides **pre-trained XML files** for common objects.

**Example: Face Detection**

import cv2

# Load the pre-trained classifier

face\_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + "haarcascade\_frontalface\_default.xml")

# Read image

img = cv2.imread("face.jpg")

gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

# Detect faces

faces = face\_cascade.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=5)

# Draw rectangles

for (x, y, w, h) in faces:

cv2.rectangle(img, (x, y), (x+w, y+h), (255, 0, 0), 2)

cv2.imshow("Faces", img)

cv2.waitKey(0)

cv2.destroyAllWindows()

**5️⃣ Key Parameters in detectMultiScale()**

* scaleFactor: How much the image size is reduced at each scale (e.g., 1.1 = 10% smaller each time).
* minNeighbors: How many detections a region needs to be accepted (higher = fewer false positives).
* minSize: Minimum object size to detect.
* maxSize: Maximum object size to detect.

**6️⃣ Advantages**

* Very fast for its time.
* Works well for **frontal, upright faces**.
* Lightweight — can run in real time on CPU.

**7️⃣ Limitations**

* Sensitive to lighting and orientation changes.
* Not good for non-frontal faces or occlusion.
* Deep learning methods (like DNN face detectors) are now more accurate.

If you want, I can also **draw a diagram showing how Haar features scan across an image and pass through cascade stages** — that makes it much easier to picture what’s happening under the hood.