**SUMMARY:**

>When we look at any image, most of the time we identify a person using a face. An image might contain multiple faces, also the face can be obstructed and not clear. The first step in our pre-processing pipeline is to detect faces from an image. Once face is detected, we will detect eyes, if two eyes are detected then only we keep that image otherwise discard it.

Haar Cascade is a **machine learning-based approach** used to detect objects (like faces, eyes, etc.) in images. It was proposed by **Paul Viola and Michael Jones** in 2001 and is widely used in OpenCV.

> Preprocessing: Use wavelet transform as a feature for traning our model

In wavelet transformed image, you can see edges clearly and that can give us clues on various facial features such as eyes, nose, lips etc

wavelet transformed image that gives clues on facial features such as eyes, nose, lips etc. This along with raw pixel image can be used as an input for our classifier

Preprocessing: Load image, detect face. If eyes >=2, then save and crop the face region

>Images in cropped folder can be used for model training. We will use these raw images along with wavelet transformed images to train our classifier. Let's prepare X and y now.

>code prepares training data for a machine learning model by combining **raw image features** and **wavelet-transformed features** into a single feature vector for each image.

>How X and y Are Computed

* **X (Feature Vectors):**
  + For each image:
  + Read and resize the raw image to 32×32 pixels → shape becomes (32×32×3,).
  + Apply wavelet transform (w2d) to extract texture features → resized to (32×32,).
  + Flatten both arrays and vertically stack them → final shape: (4096 + 1024, 1) = (5120, 1).
  + Append this combined vector to X.
* **y (Labels):**
* For each image:
* Use the person name to look up its class label from class\_dict.
* Append the label to y.