WIKIPEDIA
The Free Encyclopedia

Viola–Jones object detection framework

The **Viola–Jones object detection framework** is a machine learning object detection framework proposed in 2001 by Paul Viola and Michael Jones.[1][2] It was motivated primarily by the problem of face detection, although it can be adapted to the detection of other object classes.

In short, it consists of a sequence of classifiers. Each classifier is a single perceptron with several binary masks (Haar features). To detect faces in an image, a sliding window is computed over the image. For each image, the classifiers are applied. If at any point, a classifier outputs "no face detected", then the window is considered to contain no face. Otherwise, if all classifiers output "face detected", then the window is considered to contain a face.

The algorithm is efficient for its time, able to detect faces in 384 by 288 pixel images at 15 frames per second on a conventional 700 MHz Intel Pentium III. It is also robust, achieving high precision and recall.

While it has lower accuracy than more modern methods such as convolutional neural network, its efficiency and compact size (only around 50k parameters, compared to millions of parameters for typical CNN like DeepFace) means it is still used in cases with limited computational power. For example, in the original paper,[1] they reported that this face detector could run on the Compaq iPAQ at 2 fps (this device has a low power StrongARM without floating point hardware).

Problem description

Face detection is a binary classification problem combined with a localization problem: given a picture, decide whether it contains faces, and construct bounding boxes for the faces.

To make the task more manageable, the Viola–Jones algorithm only detects full view (no occlusion), frontal (no head-turning), upright (no rotation), well-lit, full-sized (occupying most of the frame) faces in fixed-resolution images.

The restrictions are not as severe as they appear, as one can normalize the picture to bring it closer to the requirements for Viola-Jones.

- any image can be scaled to a fixed resolution
- for a general picture with a face of unknown size and orientation, one can perform blob detection to discover potential faces, then scale and rotate them into the upright, full-sized position.
- the brightness of the image can be corrected by white balancing.
- the bounding boxes can be found by sliding a window across the entire picture, and marking down every window that contains a face.
  - This would generally detect the same face multiple times, for which duplication removal methods, such as non-maximal suppression, can be used.

The "frontal" requirement is non-negotiable, as there is no simple transformation on the image that can turn a face from a side view to a frontal view. However, one can train multiple Viola-Jones classifiers, one for each angle: one for frontal view, one for 3/4 view, one for profile view, a few more for the angles in-between them. Then one can at run time execute all these classifiers in parallel to detect faces at different view angles.

The "full-view" requirement is also non-negotiable, and cannot be simply dealt with by training more Viola-Jones classifiers, since there are too many possible ways to occlude a face.

Components of the framework

A full presentation of the algorithm is in.[3]

Consider an image $I(x, y)$ of fixed resolution (M, N). Our task is to make a binary decision: whether it is a photo of a standardized face (frontal, well-lit, etc) or not.

Viola–Jones is essentially a boosted feature learning algorithm, trained by running a modified AdaBoost algorithm on Haar feature classifiers to find a sequence of classifiers $f_1, f_2, \dots, f_k$. Haar feature classifiers are crude, but allows very fast computation, and the modified AdaBoost constructs a strong classifier out of many weak ones.

At run time, a given image I is tested on $f_1(I), f_2(I), \dots f_k(I)$ sequentially. If at any point, $f_i(I) = 0$, the algorithm immediately returns "no face detected". If all classifiers return 1, then the algorithm returns "face detected". For this reason, the Viola-Jones classifier is also called "Haar cascade classifier".

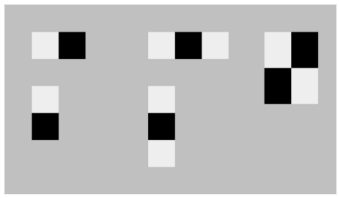
Haar feature classifiers

Consider a perceptron $f_{w,b}$ defined by two variables $w(x, y), b$. It takes in an image $I(x, y)$ of fixed resolution, and returns

$$f_{w,b}(I) = \begin{cases} 1, & \text{if } \sum_{x,y} w(x,y)I(x,y) + b > 0 \\ 0, & \text{else} \end{cases}$$

A Haar feature classifier is a perceptron $f_{w,b}$ with a very special kind of w that makes it extremely cheap to calculate. Namely, if we write out the matrix $w(x, y)$, we find that it takes only three possible values $\{+1, -1, 0\}$, and if we color the matrix with white on $+1$, black on -1, and transparent on 0, the matrix is in one of the 5 possible patterns shown on the right.

Each pattern must also be symmetric to x-reflection and y-reflection (ignoring the color change), so for example, for the horizontal white-black feature, the two rectangles must be of the same width. For the vertical white-black-white feature, the white rectangles must be of the same height, but there is no restriction on the black rectangle's height.



Example rectangle features shown relative to the enclosing detection window

Rationale for Haar features

The Haar features used in the Viola-Jones algorithm are a subset of the more general Haar basis functions, which have been used previously in the realm of image-based object detection.[4]

While crude compared to alternatives such as steerable filters, Haar features are sufficiently complex to match features of typical human faces. For example:

- The eye region is darker than the upper-cheeks.
- The nose bridge region is brighter than the eyes.

Composition of properties forming matchable facial features:

- Location and size: eyes, mouth, bridge of nose
- Value: oriented gradients of pixel intensities

Haar Feature that looks similar to the bridge of the nose is applied onto the face

Further, the design of Haar features allows for efficient computation of $f_{w,b}(I)$ using only constant number of additions and subtractions, regardless of the size of the rectangular features, using the summed-area table.

Haar Feature that looks similar to the eye region which is darker than the upper cheeks is applied onto a face

Learning and using a Viola–Jones classifier

Choose a resolution (M, N) for the images to be classified. In the original paper, they recommended $(M, N) = (24, 24)$.

Learning

Collect a training set, with some containing faces, and others not containing faces. Perform a certain modified AdaBoost training on the set of all Haar feature classifiers of dimension (M, N), until a desired level of precision and recall is reached. The modified AdaBoost algorithm would output a sequence of Haar feature classifiers $f_1, f_2, \ldots, f_k$.

The details of the modified AdaBoost algorithm is detailed below.

Using

To use a Viola-Jones classifier with $f_1, f_2, \ldots, f_k$ on an image I, compute $f_1(I), f_2(I), \ldots f_k(I)$ sequentially. If at any point, $f_i(I) = 0$, the algorithm immediately returns "no face detected". If all classifiers return 1, then the algorithm returns "face detected".

Learning algorithm

The speed with which features may be evaluated does not adequately compensate for their number, however. For example, in a standard 24x24 pixel sub-window, there are a total of $M = 162336$[5] possible features, and it would be prohibitively expensive to evaluate them all when testing an image. Thus, the object detection framework employs a variant of the learning algorithm AdaBoost to both select the best features and to train classifiers that use them. This algorithm constructs a "strong" classifier as a linear combination of weighted simple “weak” classifiers.

$$h(\mathbf{x}) = \operatorname{sgn}\left(\sum_{j=1}^{M} \alpha_j h_j(\mathbf{x})\right)$$

Each weak classifier is a threshold function based on the feature f_j.

$$h_j(\mathbf{x}) = \begin{cases} -s_j & \text{if } f_j < \theta_j \\ s_j & \text{otherwise} \end{cases}$$

The threshold value θ_j and the polarity $s_j \in \pm 1$ are determined in the training, as well as the coefficients α_j.

Here a simplified version of the learning algorithm is reported:[6]

Input: Set of N positive and negative training images with their labels $(\mathbf{x}^i, y^i)$. If image i is a face $y^i = 1$, if not $y^i = -1$.

1. Initialization: assign a weight $w_1^i = \frac{1}{N}$ to each image i.
2. For each feature f_j with $j = 1, \ldots, M$
    1. Renormalize the weights such that they sum to one.
    2. Apply the feature to each image in the training set, then find the optimal threshold and polarity θ_j, s_j that minimizes the weighted classification error. That is $\theta_j, s_j = \arg\min_{\theta,s} \sum_{i=1}^{N} w_j^i \varepsilon_j^i$ where $\varepsilon_j^i = \begin{cases} 0 & \text{if } y^i = h_j(\mathbf{x}^i, \theta_j, s_j) \\ 1 & \text{otherwise} \end{cases}$
    3. Assign a weight α_j to h_j that is inversely proportional to the error rate. In this way best classifiers are considered more.
    4. The weights for the next iteration, i.e. w_{j+1}^i, are reduced for the images i that were correctly classified.

3. Set the final classifier to $h(\mathbf{x}) = \operatorname{sgn}\left(\sum_{j=1}^{M} \alpha_j h_j(\mathbf{x})\right)$

Cascade architecture

- On average only 0.01% of all sub-windows are positive (faces)
- Equal computation time is spent on all sub-windows
- Must spend most time only on potentially positive sub-windows.
- A simple 2-feature classifier can achieve almost 100% detection rate with 50% FP rate.
- That classifier can act as a 1st layer of a series to filter out most negative windows
- 2nd layer with 10 features can tackle “harder” negative-windows which survived the 1st layer, and so on...
- A cascade of gradually more complex classifiers achieves even better detection rates. The evaluation of the strong classifiers generated by the learning process can be done quickly, but it isn't fast enough to run in real-time. For this reason, the strong classifiers are arranged in a cascade in order of complexity, where each successive classifier is trained only on those selected samples which pass through the preceding classifiers. If at any stage in the cascade a classifier rejects the sub-window under inspection, no further processing is performed and continue on searching the next sub-window. The cascade therefore has the form of a degenerate tree. In the case of faces, the first classifier in the cascade – called the attentional operator – uses only two features to achieve a false negative rate of approximately 0% and a false positive rate of 40%.[7] The effect of this single classifier is to reduce by roughly half the number of times the entire cascade is evaluated.

In cascading, each stage consists of a strong classifier. So all the features are grouped into several stages where each stage has certain number of features.

The job of each stage is to determine whether a given sub-window is definitely not a face or may be a face. A given sub-window is immediately discarded as not a face if it fails in any of the stages.

A simple framework for cascade training is given below:

- f = the maximum acceptable false positive rate per layer.
- d = the minimum acceptable detection rate per layer.
- Ftarget = target overall false positive rate.
- P = set of positive examples.
- N = set of negative examples.

```
F(0) = 1.0; D(0) = 1.0; i = 0

while F(i) > Ftarget
   increase i
   n(i) = 0; F(i)= F(i-1)

   while F(i) > f × F(i-1)
      increase n(i)
      use P and N to train a classifier with n(i) features using AdaBoost
      Evaluate current cascaded classifier on validation set to determine F(i) and D(i)
      decrease threshold for the ith classifier (i.e. how many weak classifiers need to accept for strong classifier to accept)
         until the current cascaded classifier has a detection rate of at least d × D(i-1) (this also affects F(i))
   N = ∅
   if F(i) > Ftarget then
      evaluate the current cascaded detector on the set of non-face images
      and put any false detections into the set N.
```

The cascade architecture has interesting implications for the performance of the individual classifiers. Because the activation of each classifier depends entirely on the behavior of its predecessor, the false positive rate for an entire cascade is:

$$F = \prod_{i=1}^{K} f_i.$$

Similarly, the detection rate is:

$$D = \prod_{i=1}^{K} d_i.$$

Thus, to match the false positive rates typically achieved by other detectors, each classifier can get away with having surprisingly poor performance. For example, for a 32-stage cascade to achieve a false positive rate of 10^{-6}, each classifier need only achieve a false positive rate of about 65%. At the same time, however, each classifier needs to be exceptionally capable if it is to achieve adequate detection rates. For example, to achieve a detection rate of about 90%, each classifier in the aforementioned cascade needs to achieve a detection rate of approximately 99.7%.[8]

Using Viola–Jones for object tracking

In videos of moving objects, one need not apply object detection to each frame. Instead, one can use tracking algorithms like the KLT algorithm to detect salient features within the detection bounding boxes and track their movement between frames. Not only does this improve tracking speed by removing the need to re-detect objects in each frame, but it improves the robustness as well, as the salient features are more resilient than the Viola-Jones detection framework to rotation and photometric changes.[9]

References

1. Viola, P.; Jones, M. (2001). "Rapid object detection using a boosted cascade of simple features" (https://dx.doi.org/10.1109/cvpr.2001.990517). *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*. Vol. 1. IEEE Comput. Soc. doi:10.1109/cvpr.2001.990517 (https://doi.org/10.1109%2Fcvpr.2001.990517). ISBN 0-7695-1272-0. S2CID 2715202 (https://api.semanticscholar.org/CorpusID:2715202).
2. Viola, Paul; Jones, Michael J. (May 2004). "Robust Real-Time Face Detection" (https://dx.doi.org/10.1023/b:visi.0000013087.49260.fb). *International Journal of Computer Vision*. **57** (2): 137–154. doi:10.1023/b:visi.0000013087.49260.fb (https://doi.org/10.1023%2Fb%3Avisi.0000013087.49260.fb). ISSN 0920-5691 (https://search.worldcat.org/issn/0920-5691). S2CID 2796017 (https://api.semanticscholar.org/CorpusID:2796017).
3. Wang, Yi-Qing (2014-06-26). "An Analysis of the Viola-Jones Face Detection Algorithm" (https://doi.org/10.5201%2Fipol.2014.104). *Image Processing on Line*. **4**: 128–148. doi:10.5201/ipol.2014.104 (https://doi.org/10.5201%2Fipol.2014.104). ISSN 2105-1232 (https://search.worldcat.org/issn/2105-1232).
4. C. Papageorgiou, M. Oren and T. Poggio. A General Framework for Object Detection. *International Conference on Computer Vision*, 1998
5. "Viola-Jones' face detection claims 180k features" (https://stackoverflow.com/questions/1707620/viola-jones-face-detection-claims-180k-features). *stackoverflow.com*. Retrieved 2017-06-27.
6. R. Szeliski, *Computer Vision, algorithms and applications*, Springer
7. Viola, Jones: Robust Real-time Object Detection, IJCV 2001 (http://research.microsoft.com/~viola/Pubs/Detect/violaJones_IJCV.pdf) See page 11.
8. Torbert, Shane (2016). *Applied Computer Science* (2nd ed.). Springer. pp. 122–131.
9. Face Detection and Tracking using the KLT algorithm (http://in.mathworks.com/help/vision/examples/face-detection-and-tracking-using-the-klt-algorithm.html)

External links

- Slides Presenting the Framework (https://www.slideshare.net/wolf/avihu-efrats-viola-and-jones-face-detection-slides/)
- Information Regarding Haar Basis Functions (http://mathworld.wolfram.com/HaarFunction.html)
- "Extension of Viola–Jones framework using SURF feature" (https://web.archive.org/web/20201018192200/https://sites.google.com/site/leeplus/publications/learningsurfcascadeforfastandaccurateobjectdetection). Archived from the original (https://sites.google.com/site/leeplus/publications/learningsurfcascadeforfastandaccurateobjectdetection) on 2020-10-18.
- "IMMI - Rapidminer Image Mining Extension" (https://web.archive.org/web/20220703171614/http://www.burgsys.com/mumi-image-mining-community.php). Archived from the original (http://www.burgsys.com/mumi-image-mining-community.php) on 2022-07-03. - open-source tool for image mining
- "Robust Real-Time Face Detection" (https://web.archive.org/web/20190202042433/http://www.vision.caltech.edu/html-files/EE148-2005-Spring/pprs/viola04ijcv.pdf) (PDF). Archived from the original (http://www.vision.caltech.edu/html-files/EE148-2005-Spring/pprs/viola04ijcv.pdf) (PDF) on 2019-02-02.
- An improved algorithm on Viola-Jones object detector (https://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6269796)
- Citations of the Viola–Jones algorithm in Google Scholar (https://scholar.google.com/citations?view_op=view_citation&hl=en&user=G2-nFaIAAAAJ&citation_for_view=G2-nFaIAAAAJ:u5HHmVD_uO8C)
- Video lecture on Viola–Jones algorithm (https://www.youtube.com/watch?v=WfdYYNamHZ8) on YouTube - Adaboost Explanation from ppt by Qing Chen, Discovery Labs, University of Ottawa and a video lecture by Ramsri Goutham.

Implementations

- Jensen, Ole Helvig. "Implementing the Viola–Jones Face Detection Algorithm" (https://web.archive.org/web/20181021023153/http://etd.dtu.dk/thesis/223656/ep08_93.pdf) (PDF). Archived from the original (http://etd.dtu.dk/thesis/223656/ep08_93.pdf) (PDF) on 2018-10-21.
- MATLAB: [1] (http://www.mathworks.com/matlabcentral/fileexchange/29437-viola-jones-object-detection), [2] (http://in.mathworks.com/help/vision/ref/vision.cascadeobjectdetector-class.html) Archived (https://web.archive.org/web/20170907140646/http://in.mathworks.com/help/vision/ref/vision.cascadeobjectdetector-class.html) 2017-09-07 at the Wayback Machine
- OpenCV: implemented as `cvHaarDetectObjects()`.
  - Haar Cascade Detection in OpenCV (https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_objdetect/py_face_detection/py_face_detection.html)
  - Cascade Classifier Training in OpenCV (http://docs.opencv.org/doc/user_guide/ug_traincascade.html)