Object Detection

Viola-Jones Detector

Lecture presented by

Amirhossein Dadashzadeh

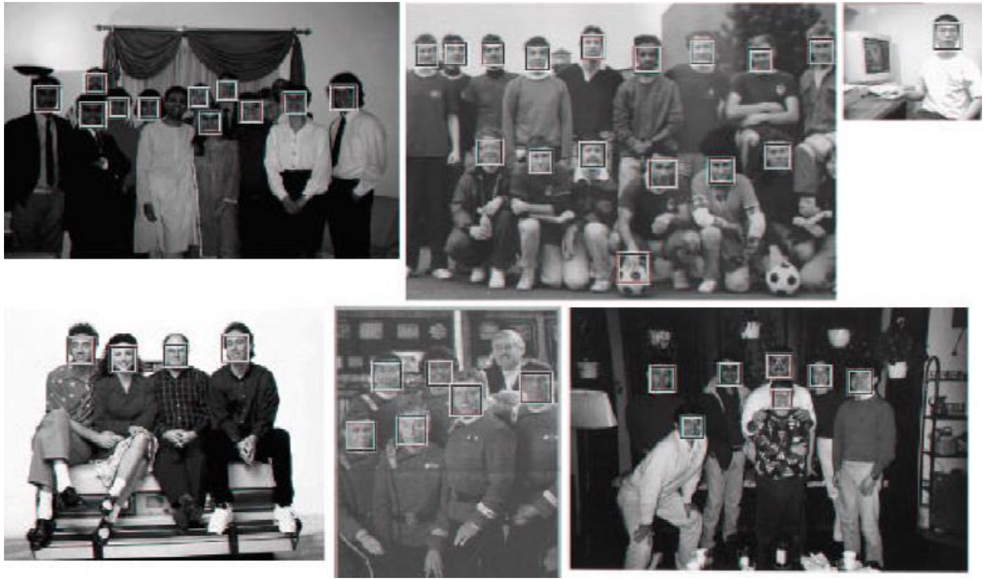
Object Detection Techniques

- ✓ **Line and circle detection**: Techniques like the Hough Transform can be used to detect lines and circles in an image, which can indirectly help locate objects with specific geometric shapes.
- ✓ **Colour-based detection**: In some cases, objects can be detected based on their colour properties. This is especially useful when objects have distinct and consistent colors.
- ✓ **Template matching**: Using sliding a template over the input image and finding regions where the template best matches the local image content.
- ✓ **Classifiers with sliding window detectors**: Applying image classification on overlapped patches in the image.
- ✗ **Deep learning-based object detectors**: Object detector automatically learns image features required for detection tasks, and instance segmentation.

(out of scope in this unit)

Classifiers with sliding window detectors

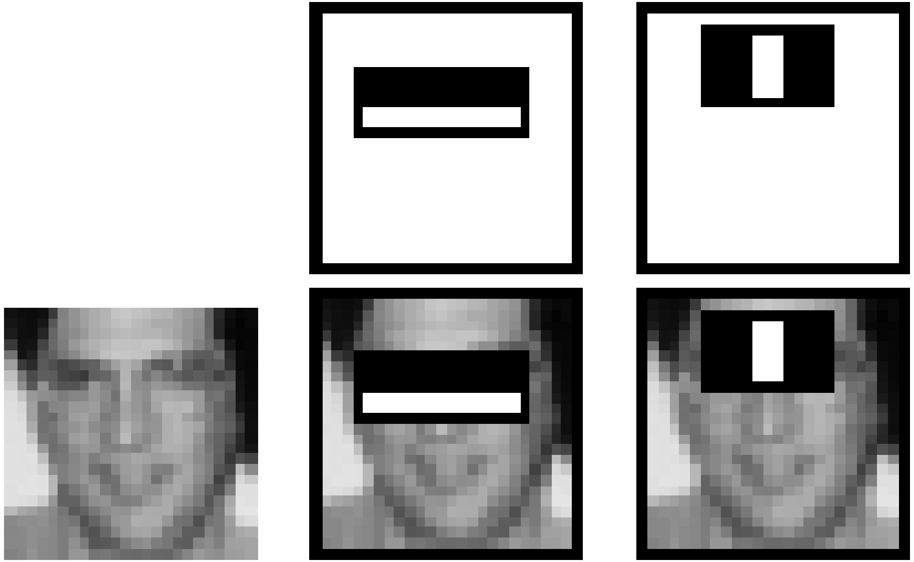
- Example Algorithm: **Viola & Jones' Real-time Method**
  - Sliding Window Detectors
  - Haar-like Features
  - Feature Extraction and Integral Images
  - Weak Classifiers
  - Boosting and Classifier Evaluation
  - Cascades of Boosted Classifiers



Best description of full details available in consolidated paper by Viola and Jones, International Journal of Computer Vision, 2004

Haar-like Features

Viola & Jones’ (2001)



feature = sum of white pixels – sum of black pixels

filter 1

-1	-1	1	1
-1	-1	1	1
-1	-1	1	1
-1	-1	1	1

hard edge
feature1 = 2040

0	0	255	255
0	0	255	255
0	0	255	255
0	0	255	255

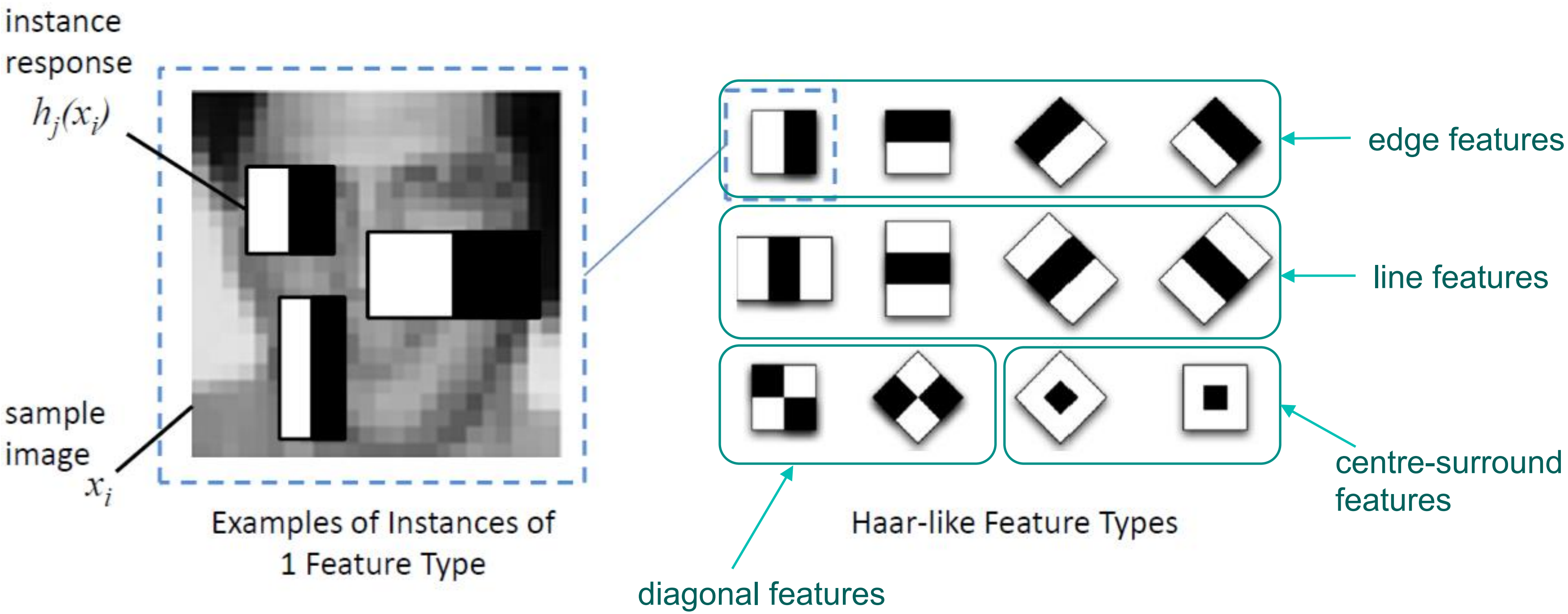
soft edge
feature1 = 1245

20	120	220	230
25	125	205	225
25	105	220	234
24	110	215	250

no edge
feature1 = 17

218	230	220	230
200	230	205	225
220	234	220	244
210	250	215	250

Haar-like Features



Examples of Instances of 1 Feature Type

Haar-like Feature Types

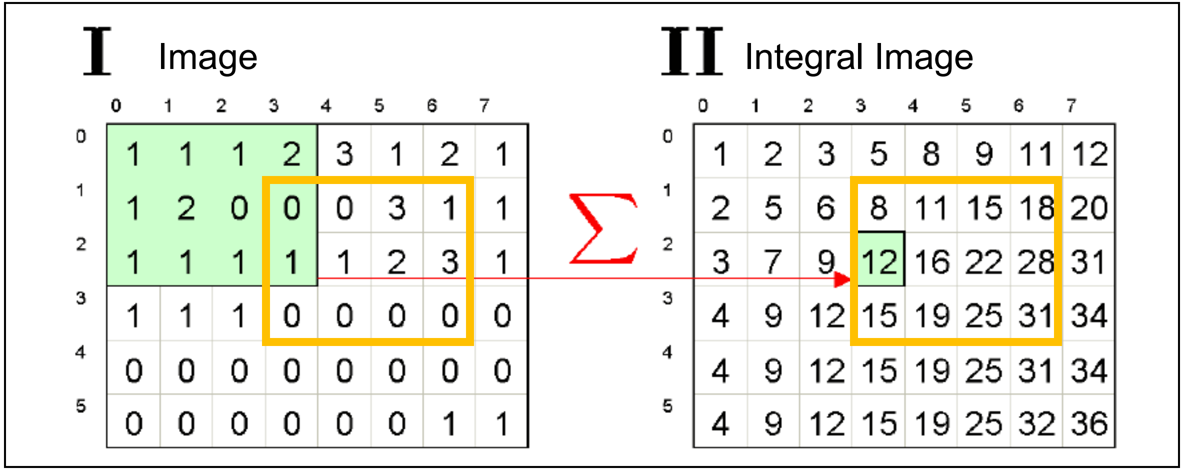
edge features

line features

centre-surround features

diagonal features

Integral Images



I Image

	0	1	2	3	4	5	6	7
0	1	1	1	2	3	1	2	1
1	1	2	0	0	0	3	1	1
2	1	1	1	1	1	2	3	1
3	1	1	1	0	0	0	0	0
4	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	1	1

II Integral Image

	0	1	2	3	4	5	6	7
0	1	2	3	5	8	9	11	12
1	2	5	6	8	11	15	18	20
2	3	7	9	12	16	22	28	31
3	4	9	12	15	19	25	31	34
4	4	9	12	15	19	25	31	34
5	4	9	12	15	19	25	32	36

(IMAGE INTEGRATION)

$$\mathbf{II}(-1,y)=0; \qquad \mathbf{II}(x,y)=\mathbf{II}(x-1,y)+A(x,y);$$

$$A(x,-1)=0; \qquad A(x,y)=A(x,y-1)+\mathbf{I}(x,y).$$

x=0, y=0: A(0,0) = A(0,-1) + I(0,0) = 0 + 1 = 1
II(0,0) = II(-1,0) + A(0,0) = 0 + 1 = 1

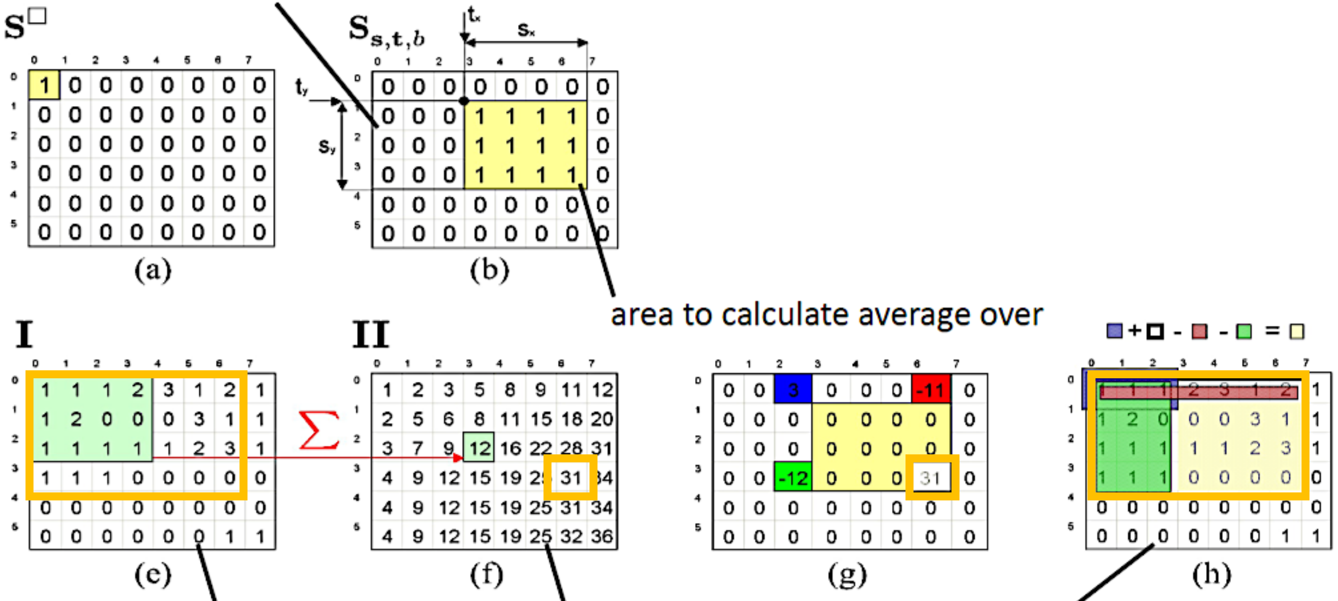
x=0, y=1: A(0,1) = A(0,0) + I(0,1) = 1 + 1 = 2
II(0,1) = II(-1,1) + A(0,1) = 0 + 2 = 2

x=1, y=0: A(1,0) = A(1,-1) + I(1,0) = 0 + 1 = 1
II(1,0) = II(0,0) + A(1,0) = 1 + 1 = 2

x=1, y=1: A(1,1) = A(1,0) + I(1,1) = 1 + 2 = 3
II(1,1) = II(0,1) + A(1,1) = 2 + 3 = 5

…

Calculating the Avg Pixel Value of Large Block Fast



block image encoding area

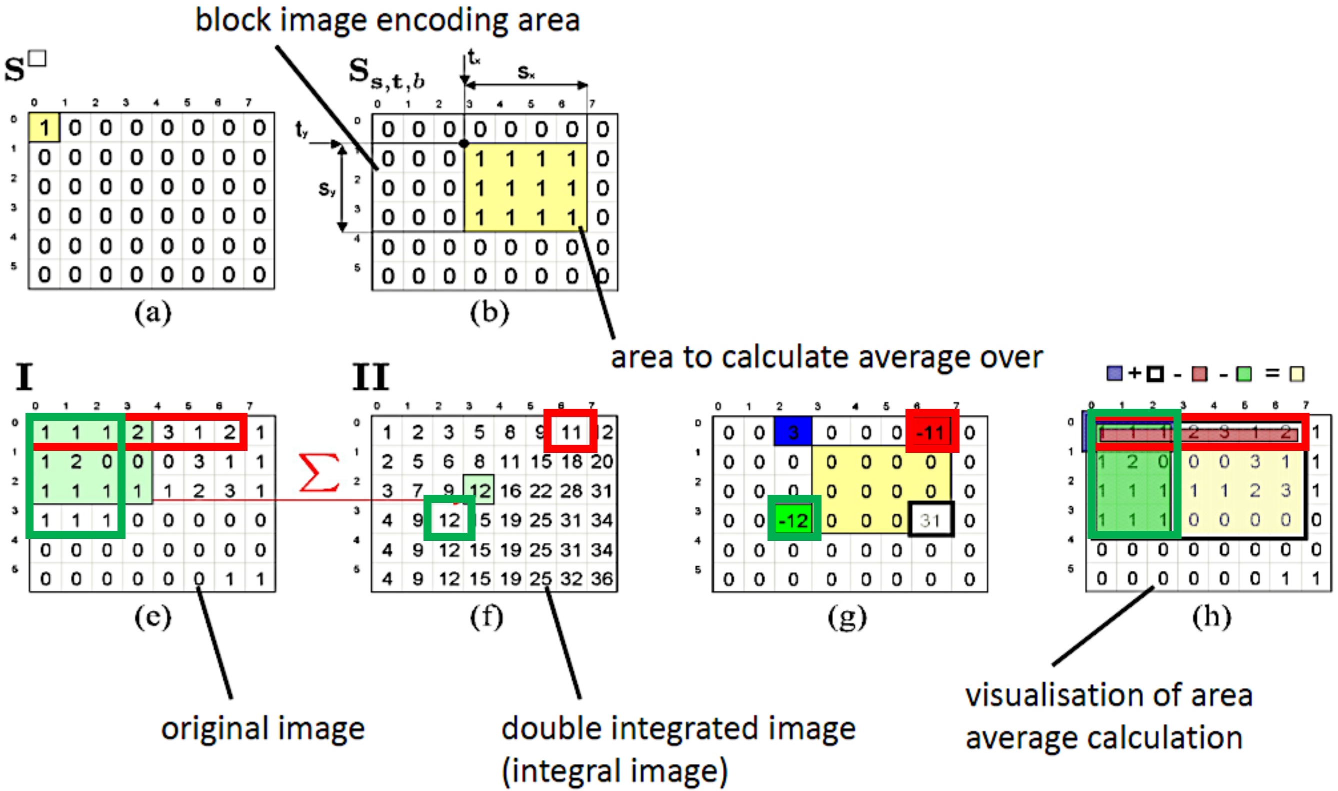
area to calculate average over

original image

double integrated image (integral image)

visualisation of area average calculation

Calculating the Avg Pixel Value of Large Block Fast



block image encoding area
$\mathbf{S}^{\square}$
$\mathbf{S}_{\mathbf{s},\mathbf{t},b}$
t_x
s_x
t_y
s_y
(a)
(b)
area to calculate average over
I
II
Σ
🟦 + □ - 🟥 - 🟩 = 🟨
(e)
(f)
(g)
(h)
original image
double integrated image (integral image)
visualisation of area average calculation

(a) $\mathbf{S}^{\square}$

	0	1	2	3	4	5	6	7
0	1	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0

(b) $\mathbf{S}_{\mathbf{s},\mathbf{t},b}$

	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
1	0	0	0	1	1	1	1	0
2	0	0	0	1	1	1	1	0
3	0	0	0	1	1	1	1	0
4	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0

(e) I

	0	1	2	3	4	5	6	7
0	1	1	1	2	3	1	2	1
1	1	2	0	0	0	3	1	1
2	1	1	1	1	1	2	3	1
3	1	1	1	0	0	0	0	0
4	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	1	1

(f) II

	0	1	2	3	4	5	6	7
0	1	2	3	5	8	9	11	12
1	2	5	6	8	11	15	18	20
2	3	7	9	12	16	22	28	31
3	4	9	12	15	19	25	31	34
4	4	9	12	15	19	25	31	34
5	4	9	12	15	19	25	32	36

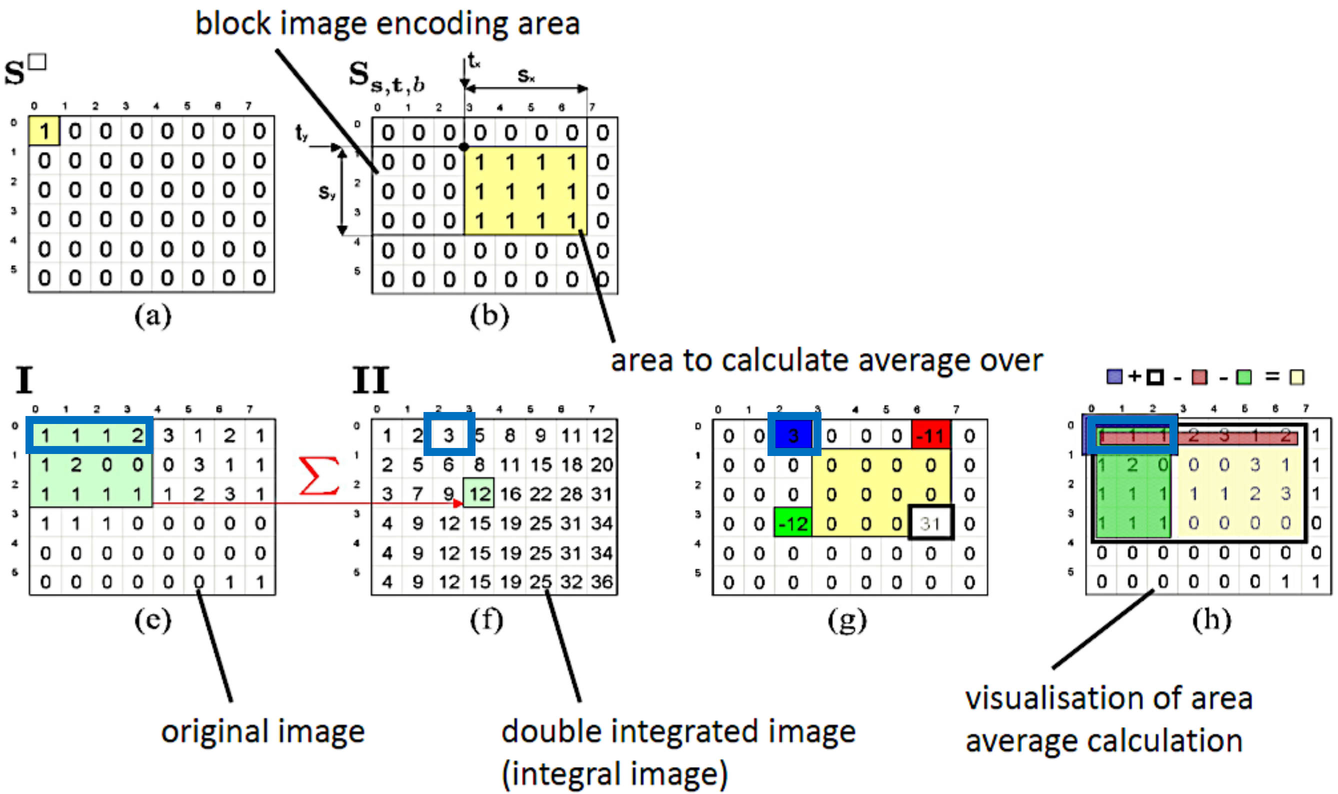
(g)

	0	1	2	3	4	5	6	7
0	0	0	3	0	0	0	-11	0
1	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0
3	0	0	-12	0	0	0	31	0
4	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0

(h)

	0	1	2	3	4	5	6	7
0	1	1	1	2	3	1	2	1
1	1	2	0	0	0	3	1	1
2	1	1	1	1	1	2	3	1
3	1	1	1	0	0	0	0	0
4	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	1	1

Calculating the Avg Pixel Value of Large Block Fast



block image encoding area

area to calculate average over

original image

double integrated image
(integral image)

visualisation of area
average calculation

$\mathbf{S}^{\square}$

	0	1	2	3	4	5	6	7
0	1	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0

(a)

$\mathbf{S}_{\mathbf{s},\mathbf{t},b}$

	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
1	0	0	0	1	1	1	1	0
2	0	0	0	1	1	1	1	0
3	0	0	0	1	1	1	1	0
4	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0

(b)

I

	0	1	2	3	4	5	6	7
0	1	1	1	2	3	1	2	1
1	1	2	0	0	0	3	1	1
2	1	1	1	1	1	2	3	1
3	1	1	1	0	0	0	0	0
4	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	1	1

(e)

$\sum$

II

	0	1	2	3	4	5	6	7
0	1	2	3	5	8	9	11	12
1	2	5	6	8	11	15	18	20
2	3	7	9	12	16	22	28	31
3	4	9	12	15	19	25	31	34
4	4	9	12	15	19	25	31	34
5	4	9	12	15	19	25	32	36

(f)

	0	1	2	3	4	5	6	7
0	0	0	3	0	0	0	-11	0
1	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0
3	0	0	-12	0	0	0	31	0
4	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0

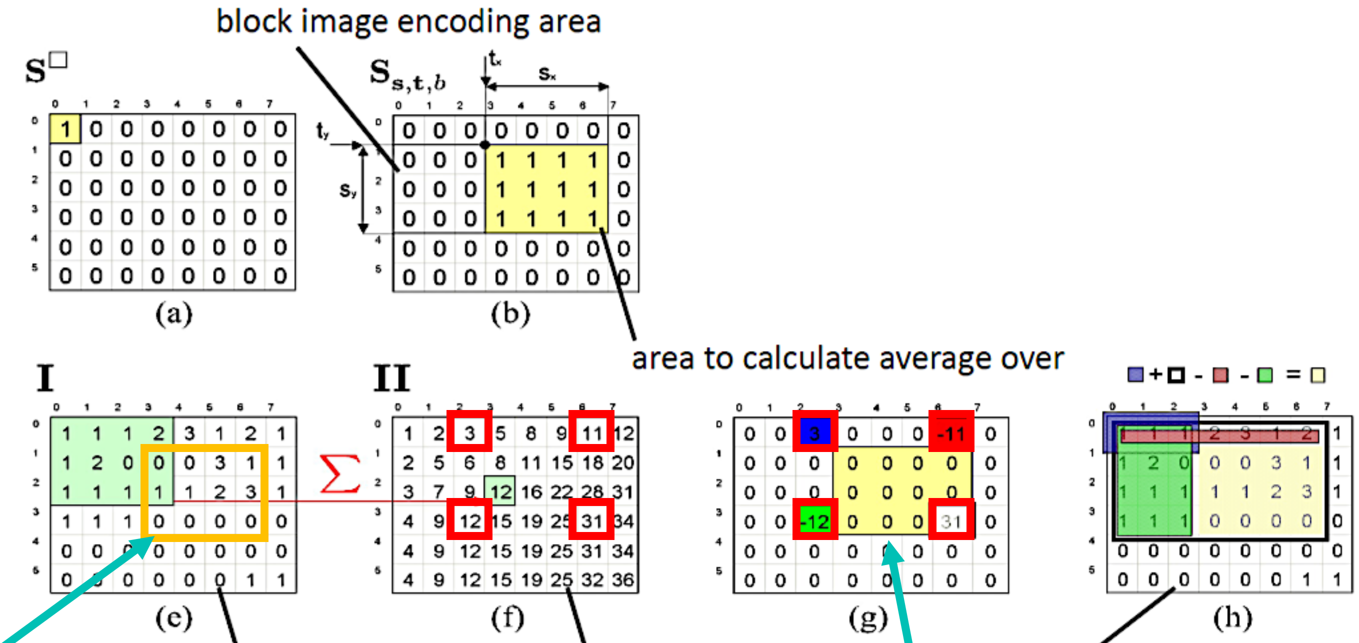
(g)

🟦 + □ - 🟥 - 🟩 = 🟨

	0	1	2	3	4	5	6	7
0	1	1	1	2	3	1	2	1
1	1	2	0	0	0	3	1	1
2	1	1	1	1	1	2	3	1
3	1	1	1	0	0	0	0	0
4	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	1	1

(h)

Calculating the Avg Pixel Value of Large Block Fast



block image encoding area

(a)

	0	1	2	3	4	5	6	7
0	1	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0

(b)

	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
1	0	0	0	1	1	1	1	0
2	0	0	0	1	1	1	1	0
3	0	0	0	1	1	1	1	0
4	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0

area to calculate average over

(e)

	0	1	2	3	4	5	6	7
0	1	1	1	2	3	1	2	1
1	1	2	0	0	0	3	1	1
2	1	1	1	1	1	2	3	1
3	1	1	1	0	0	0	0	0
4	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	1	1

(f)

	0	1	2	3	4	5	6	7
0	1	2	3	5	8	9	11	12
1	2	5	6	8	11	15	18	20
2	3	7	9	12	16	22	28	31
3	4	9	12	15	19	25	31	34
4	4	9	12	15	19	25	31	34
5	4	9	12	15	19	25	32	36

(g)

	0	1	2	3	4	5	6	7
0	0	0	3	0	0	0	-11	0
1	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0
3	0	0	-12	0	0	0	31	0
4	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0

(h)

	0	1	2	3	4	5	6	7
0	1	1	1	2	3	1	2	1
1	1	2	0	0	0	3	1	1
2	1	1	1	1	1	2	3	1
3	1	1	1	0	0	0	0	0
4	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	1	1

0 + 0 + 3 + 1 +1 + 1 + 2 + 3 + 0 + 0 + 0 + 0 = 11

original image

double integrated image (integral image)

3 + 31 - 11 - 12 = 11

visualisation of area average calculation

Feature Extraction

x=2, y=2

x

y	0	1	2	3	4	5	6	7	8	9
0	4	1	1	0	10	0	1	0	0	5
1	3	5	7	1	4	1	8	9	4	0
2	6	4	5	0	1	0	7	8	3	2
3	2	3	4	5	1	1	6	3	4	5
4	9	9	10	1	5	3	1	4	5	4
5	2	2	1	7	4	0	2	1	5	6
6	8	1	1	4	2	3	2	3	1	0
7	0	1	7	0	3	5	6	3	4	1
8	1	5	6	3	5	9	10	4	2	0
9	1	8	3	4	6	3	6	3	3	5

Image

x

y	0	1	2	3	4	5	6	7	8	9
0	4	5	6	6	16	16	17	17	17	22
1	7	13	21	22	36	37	46	55	59	64
2	13	23	36	37	52	53	69	86	93	100
3	15	28	45	51	67	69	91	111	122	134
4	24	46	73	80	101	106	129	153	169	185
5	26	50	78	92	117	122	147	172	193	215
6	34	59	88	106	133	141	168	196	218	240
7	34	60	96	114	144	157	190	221	247	270
8	35	66	108	129	164	186	229	264	292	315
9	36	75	120	145	186	211	260	298	329	357

Integral Image

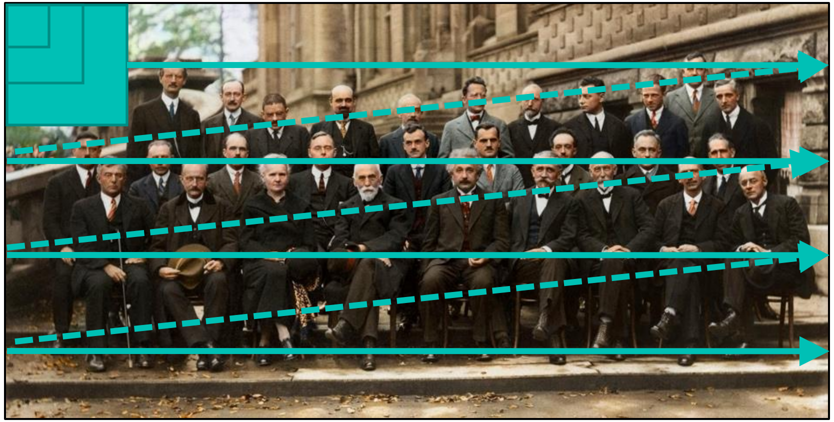
-1	-1	-1	1	1	1
-1	-1	-1	1	1	1
-1	-1	-1	1	1	1
-1	-1	-1	1	1	1
-1	-1	-1	1	1	1
-1	-1	-1	1	1	1

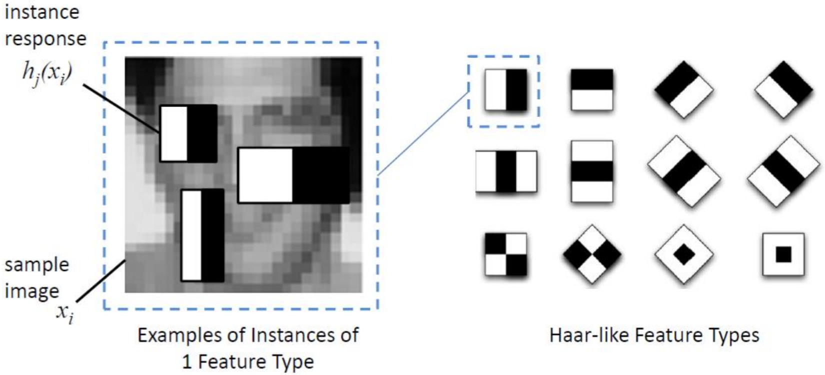
Haar filter

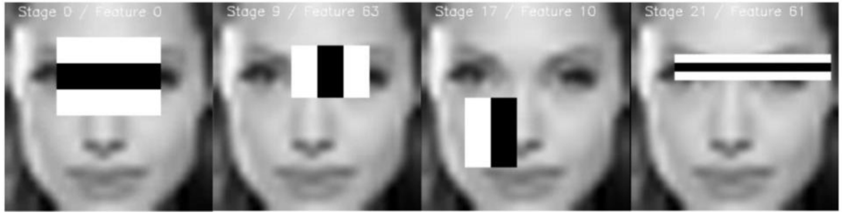
-5-0-1-4-5-1-10-1-5-1-7-4-1-4-2-7-0-3+0+7+8+1+6+3+3+1+4+0+2+1+3+2+3+5+6+3 = **-3**

- (144+13-36-60) + (221+36-55-144) = **-3**

Viola & Jones' Real-time Method

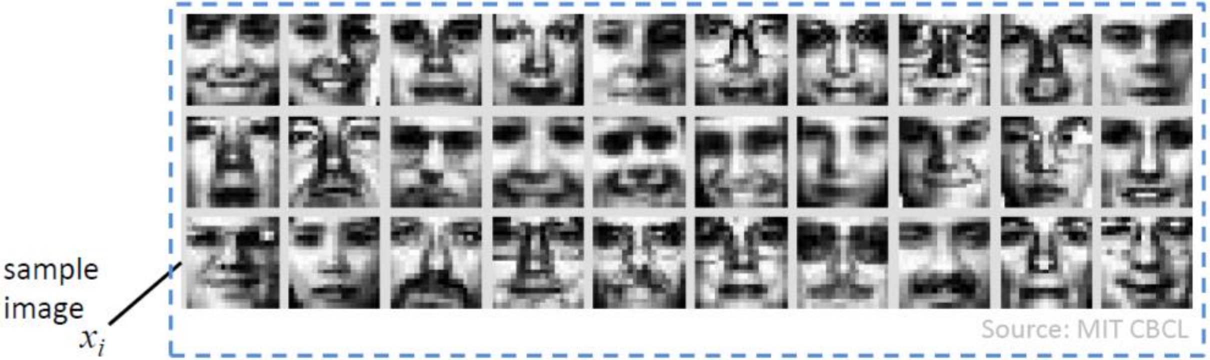






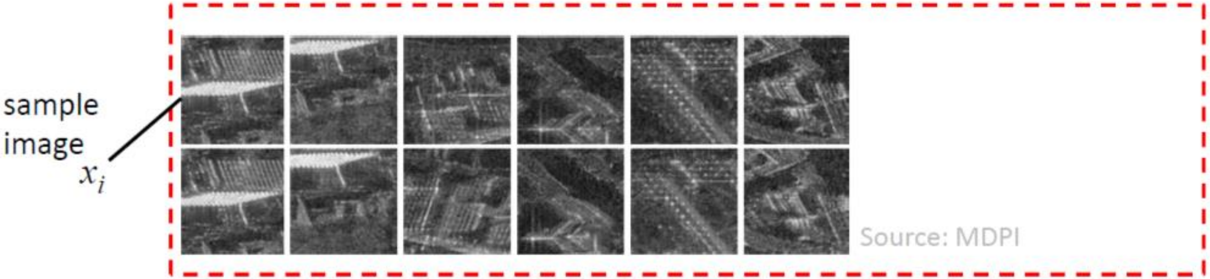
https://medium.com/@Andrew_D./computer-vision-viola-jones-object-detection-d2a609527b7c

Training



Source: MIT CBCL

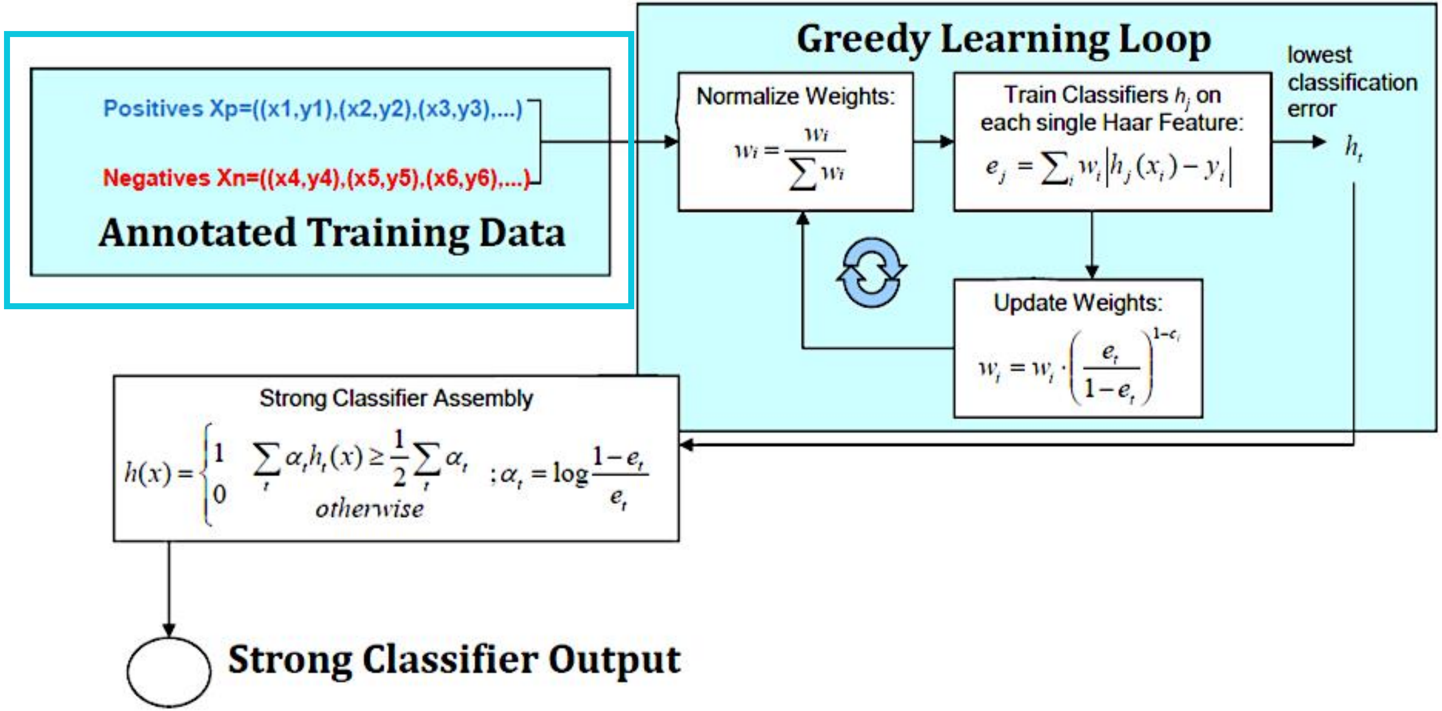
Positive Samples (e.g. FACE) ... $(x_i ,y_i =1)$, $w_i =1$



Source: MDPI

Negative Samples (e.g. NO-FACE) ... $(x_i ,y_i =0)$, $w_i =1$

AdaBoost Classifier



Positives Xp=((x1,y1),(x2,y2),(x3,y3),...)

Negatives Xn=((x4,y4),(x5,y5),(x6,y6),...)

Annotated Training Data

Greedy Learning Loop

Normalize Weights:

$$w_i = \frac{w_i}{\sum w_i}$$

Train Classifiers h_j on each single Haar Feature:

$$e_j = \sum_i w_i \left| h_j(x_i) - y_i \right|$$

lowest classification error

h_t

Update Weights:

$$w_i = w_i \cdot \left(\frac{e_t}{1-e_t} \right)^{1-e_i}$$

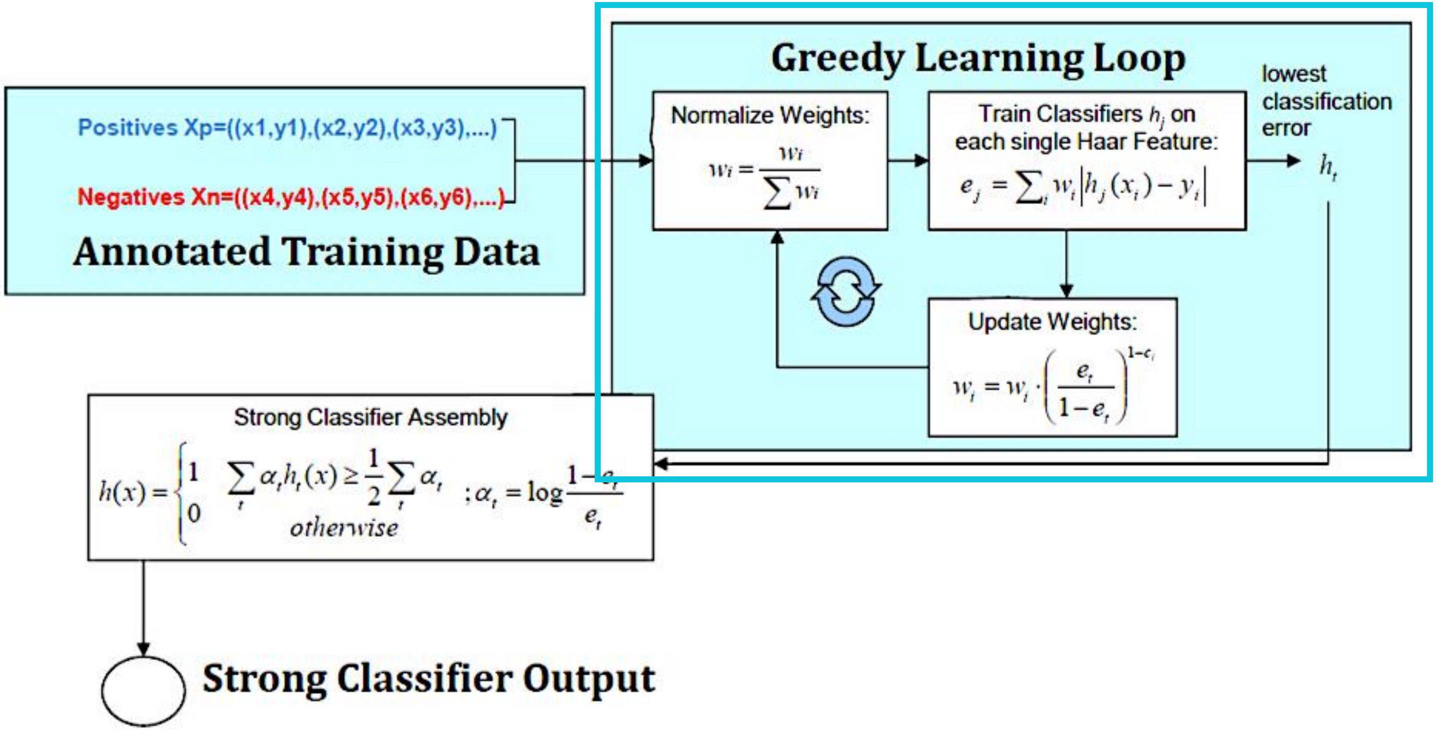
Strong Classifier Assembly

$$h(x) = \begin{cases} 1 & \sum_t \alpha_t h_t(x) \geq \frac{1}{2} \sum_t \alpha_t \\ 0 & otherwise \end{cases} \quad ; \alpha_t = \log \frac{1-e_t}{e_t}$$

Strong Classifier Output

(also see paper by Viola and Jones 2004)

AdaBoost Classifier



Positives Xp=((x1,y1),(x2,y2),(x3,y3),...)

Negatives Xn=((x4,y4),(x5,y5),(x6,y6),...)

Annotated Training Data

Greedy Learning Loop

Normalize Weights:

$$w_i = \frac{w_i}{\sum w_i}$$

Train Classifiers h_j on each single Haar Feature:

$$e_j = \sum_i w_i \left| h_j(x_i) - y_i \right|$$

lowest classification error

h_t

Update Weights:

$$w_i = w_i \cdot \left(\frac{e_t}{1 - e_t} \right)^{1 - \epsilon_i}$$

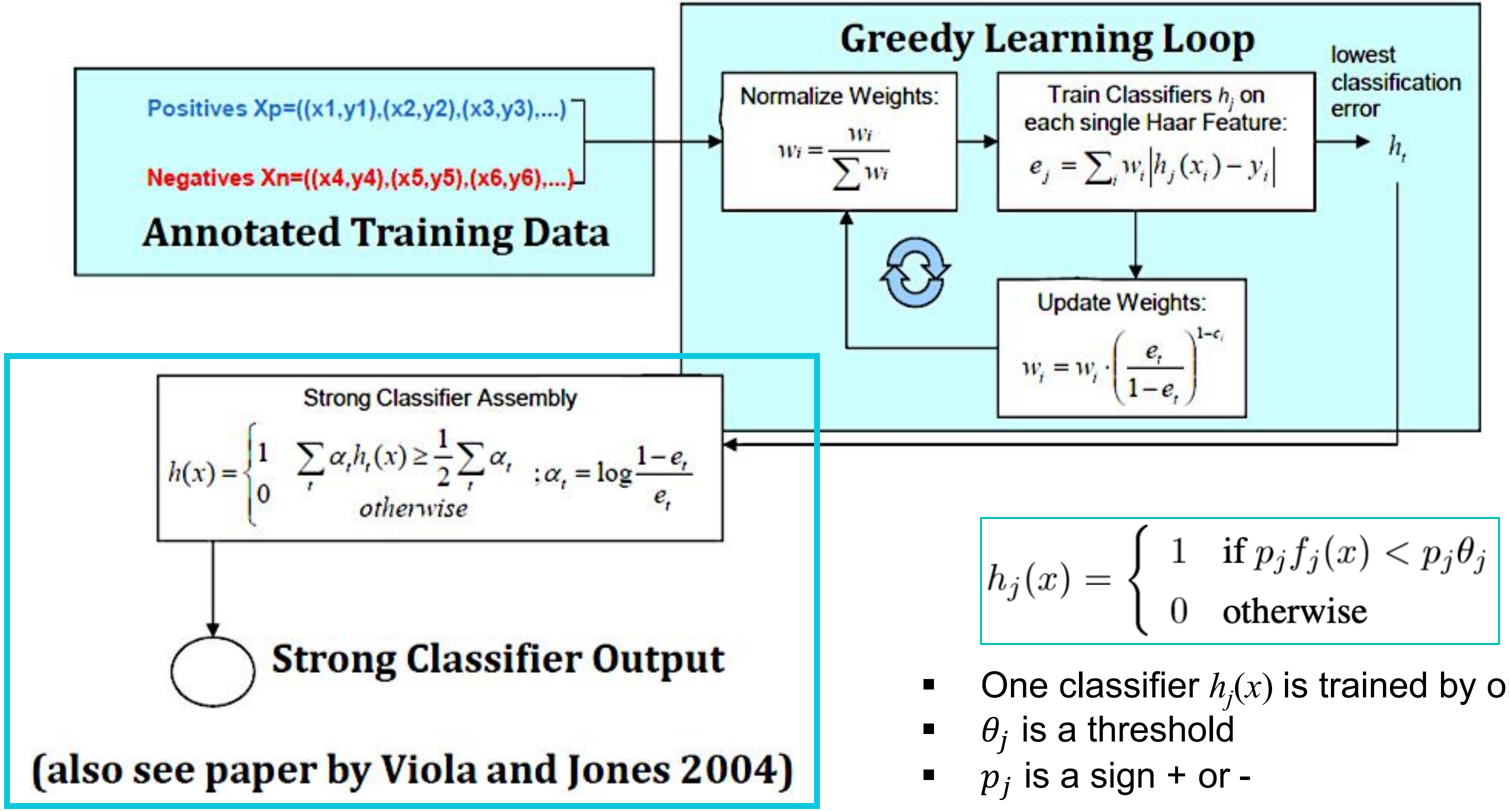
Strong Classifier Assembly

$$h(x) = \begin{cases} 1 & \sum_t \alpha_t h_t(x) \geq \frac{1}{2} \sum_t \alpha_t \\ 0 & otherwise \end{cases} \quad ; \alpha_t = \log \frac{1 - e_t}{e_t}$$

Strong Classifier Output

(also see paper by Viola and Jones 2004)

AdaBoost Classifier



Positives Xp=((x1,y1),(x2,y2),(x3,y3),...)

Negatives Xn=((x4,y4),(x5,y5),(x6,y6),...)

Annotated Training Data

Greedy Learning Loop

Normalize Weights:

$$w_i = \frac{w_i}{\sum w_i}$$

Train Classifiers h_j on each single Haar Feature:

$$e_j = \sum_i w_i \left| h_j(x_i) - y_i \right|$$

lowest classification error

h_t

Update Weights:

$$w_i = w_i \cdot \left(\frac{e_t}{1-e_t} \right)^{1-c_i}$$

Strong Classifier Assembly

$$h(x) = \begin{cases} 1 & \sum_t \alpha_t h_t(x) \geq \frac{1}{2} \sum_t \alpha_t \\ 0 & otherwise \end{cases} \quad ; \alpha_t = \log \frac{1-e_t}{e_t}$$

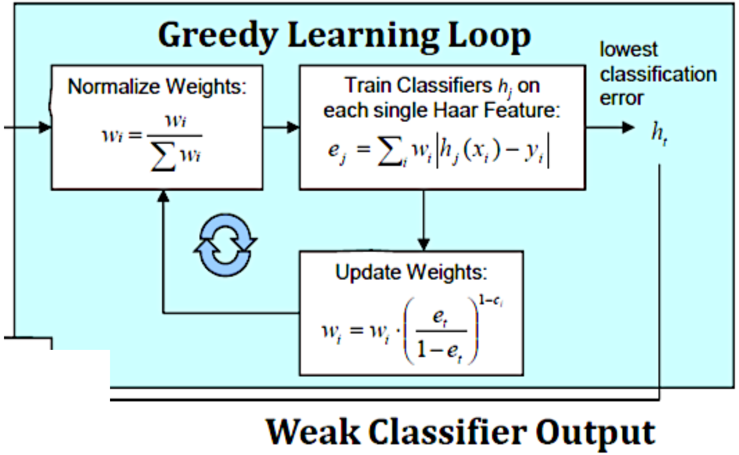
Strong Classifier Output

(also see paper by Viola and Jones 2004)

$$h_j(x) = \begin{cases} 1 & \text{if } p_j f_j(x) < p_j \theta_j \\ 0 & \text{otherwise} \end{cases}$$

- One classifier $h_j(x)$ is trained by one feature $f_j(x)$
- θ_j is a threshold
- p_j is a sign + or -

AdaBoost Classifier



Greedy Learning Loop

Normalize Weights:

$w_i = \frac{w_i}{\sum w_i}$

Train Classifiers h_j on each single Haar Feature:

$e_j = \sum_i w_i |h_j(x_i) - y_i|$

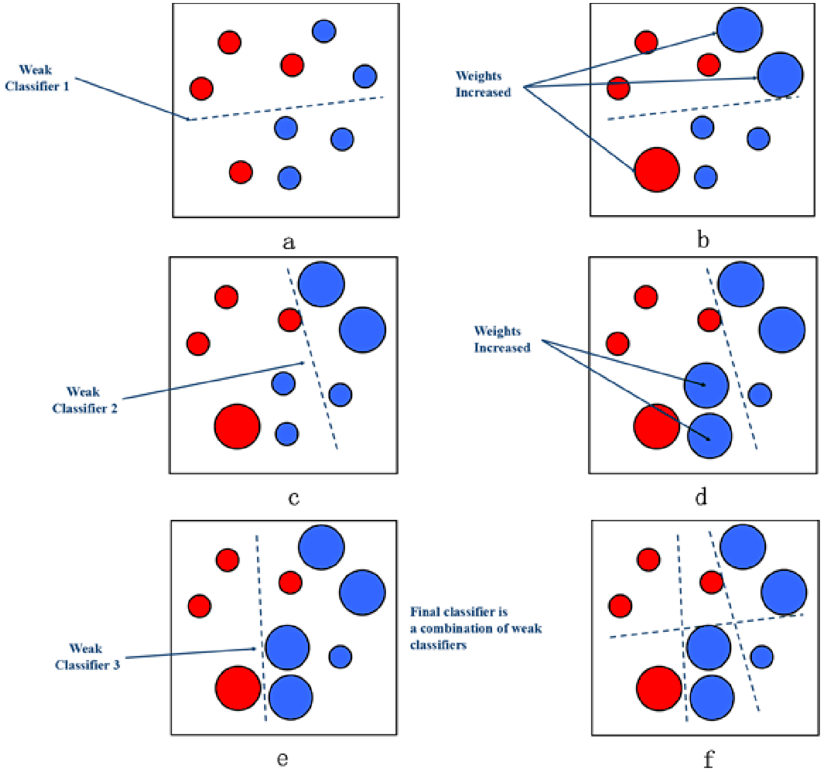
lowest classification error

h_t

Update Weights:

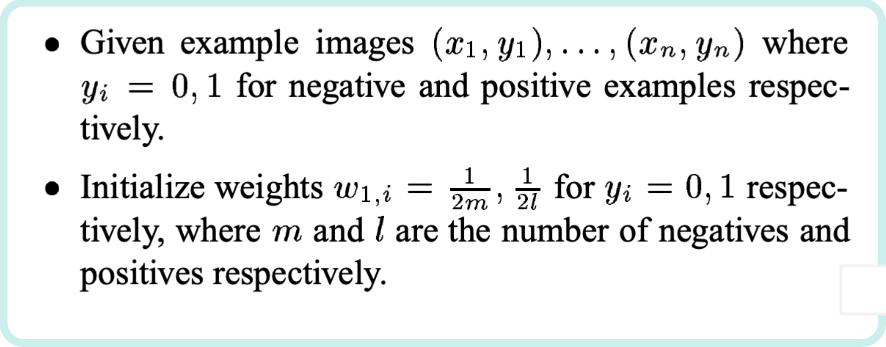
$w_i = w_i \cdot \left(\frac{e_t}{1-e_t}\right)^{1-\epsilon_i}$

Weak Classifier Output

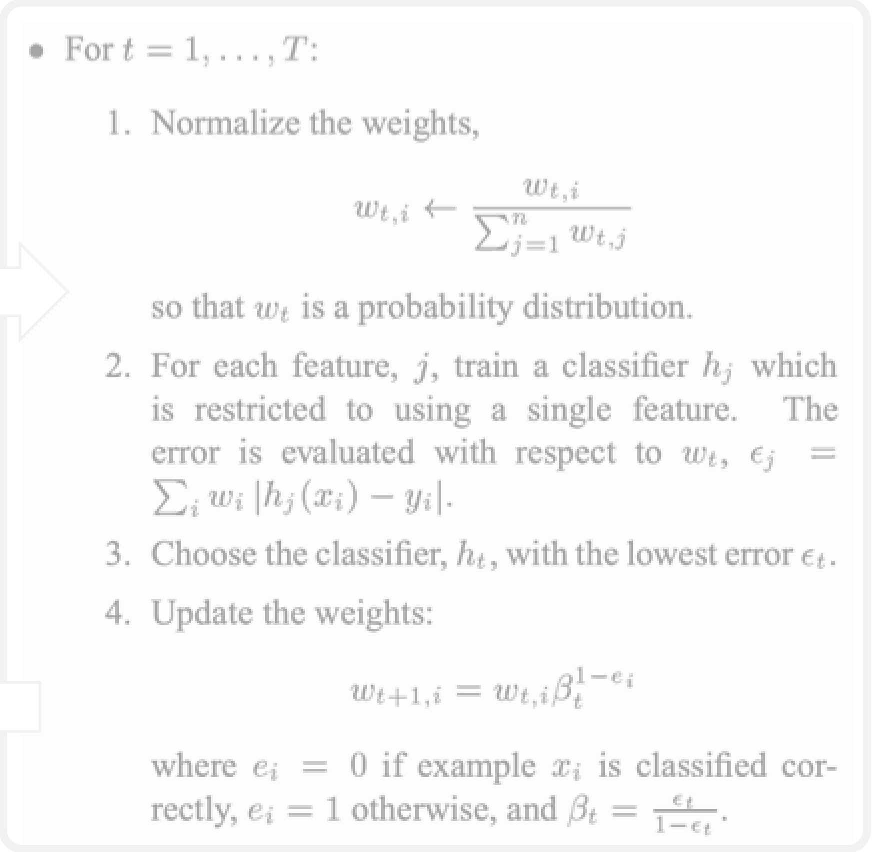


https://www.codeproject.com/Articles/4114375/Step-by-Step-Guide-to-Implement-Machine-Learning

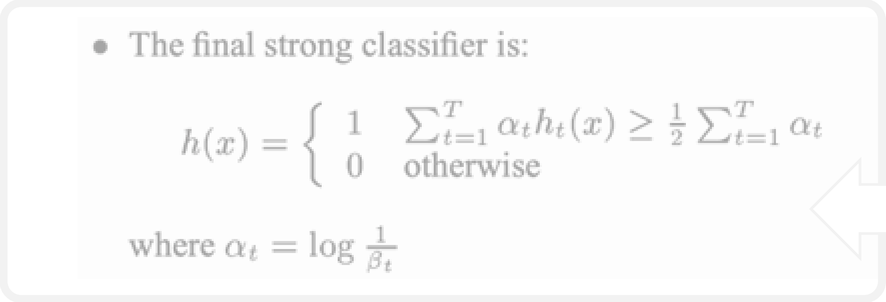
Adaboost Algorithm



- Given example images $(x_1, y_1), \ldots, (x_n, y_n)$ where $y_i = 0, 1$ for negative and positive examples respectively.
- Initialize weights $w_{1,i} = \frac{1}{2m}, \frac{1}{2l}$ for $y_i = 0, 1$ respectively, where m and l are the number of negatives and positives respectively.

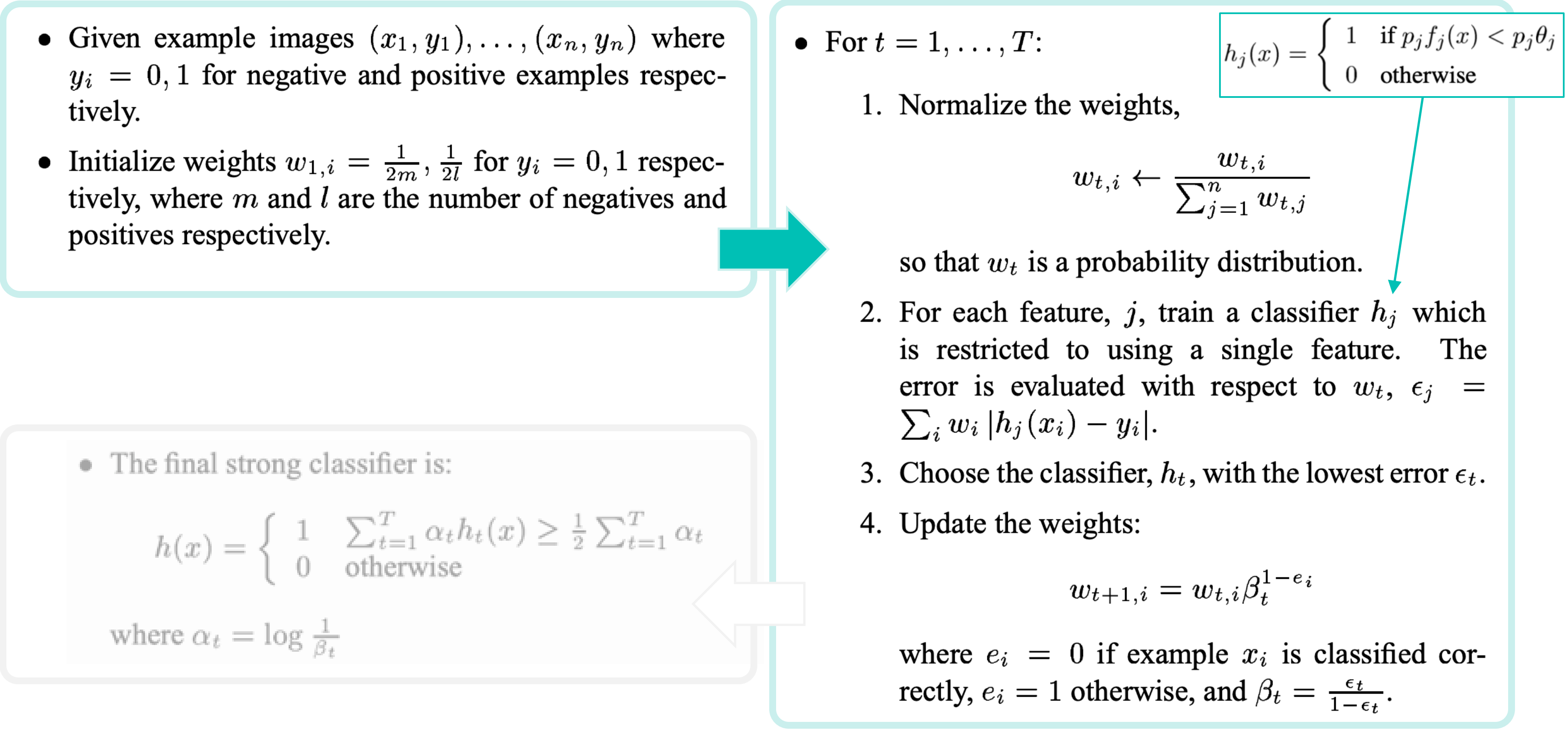


- For $t = 1, \ldots, T$:
  1. Normalize the weights,
     $$w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}}$$
     so that w_t is a probability distribution.
  2. For each feature, j, train a classifier h_j which is restricted to using a single feature. The error is evaluated with respect to w_t, $\epsilon_j = \sum_i w_i \left| h_j(x_i) - y_i \right|$.
  3. Choose the classifier, h_t, with the lowest error ϵ_t.
  4. Update the weights:
     $$w_{t+1,i} = w_{t,i} \beta_t^{1-e_i}$$
     where $e_i = 0$ if example x_i is classified correctly, $e_i = 1$ otherwise, and $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$.



- The final strong classifier is:
  $$h(x) = \begin{cases} 1 & \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0 & \text{otherwise} \end{cases}$$
  where $\alpha_t = \log \frac{1}{\beta_t}$

Adaboost Algorithm



- Given example images $(x_1, y_1), \dots, (x_n, y_n)$ where $y_i = 0, 1$ for negative and positive examples respectively.
- Initialize weights $w_{1,i} = \frac{1}{2m}, \frac{1}{2l}$ for $y_i = 0, 1$ respectively, where m and l are the number of negatives and positives respectively.

- For $t = 1, \dots, T$:

  1. Normalize the weights,
  $$w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}}$$
  so that w_t is a probability distribution.
  2. For each feature, j, train a classifier h_j which is restricted to using a single feature. The error is evaluated with respect to w_t, $\epsilon_j = \sum_i w_i \, |h_j(x_i) - y_i|$.
  3. Choose the classifier, h_t, with the lowest error ϵ_t.
  4. Update the weights:
  $$w_{t+1,i} = w_{t,i}\beta_t^{1-e_i}$$
  where $e_i = 0$ if example x_i is classified correctly, $e_i = 1$ otherwise, and $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$.

$$h_j(x) = \begin{cases} 1 & \text{if } p_j f_j(x) < p_j \theta_j \\ 0 & \text{otherwise} \end{cases}$$

- The final strong classifier is:
$$h(x) = \begin{cases} 1 & \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0 & \text{otherwise} \end{cases}$$
where $\alpha_t = \log \frac{1}{\beta_t}$

Adaboost Algorithm

- Given example images $(x_1, y_1), \dots, (x_n, y_n)$ where $y_i = 0, 1$ for negative and positive examples respectively.
- Initialize weights $w_{1,i} = \frac{1}{2m}, \frac{1}{2l}$ for $y_i = 0, 1$ respectively, where m and l are the number of negatives and positives respectively.

- For $t = 1, \dots, T$:

  1. Normalize the weights,

     $$w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}}$$

     so that w_t is a probability distribution.

  2. For each feature, j, train a classifier h_j which is restricted to using a single feature. The error is evaluated with respect to w_t, $\epsilon_j = \sum_i w_i \left| h_j(x_i) - y_i \right|$.

  3. Choose the classifier, h_t, with the lowest error ϵ_t.

  4. Update the weights:

     $$w_{t+1,i} = w_{t,i} \beta_t^{1-e_i}$$

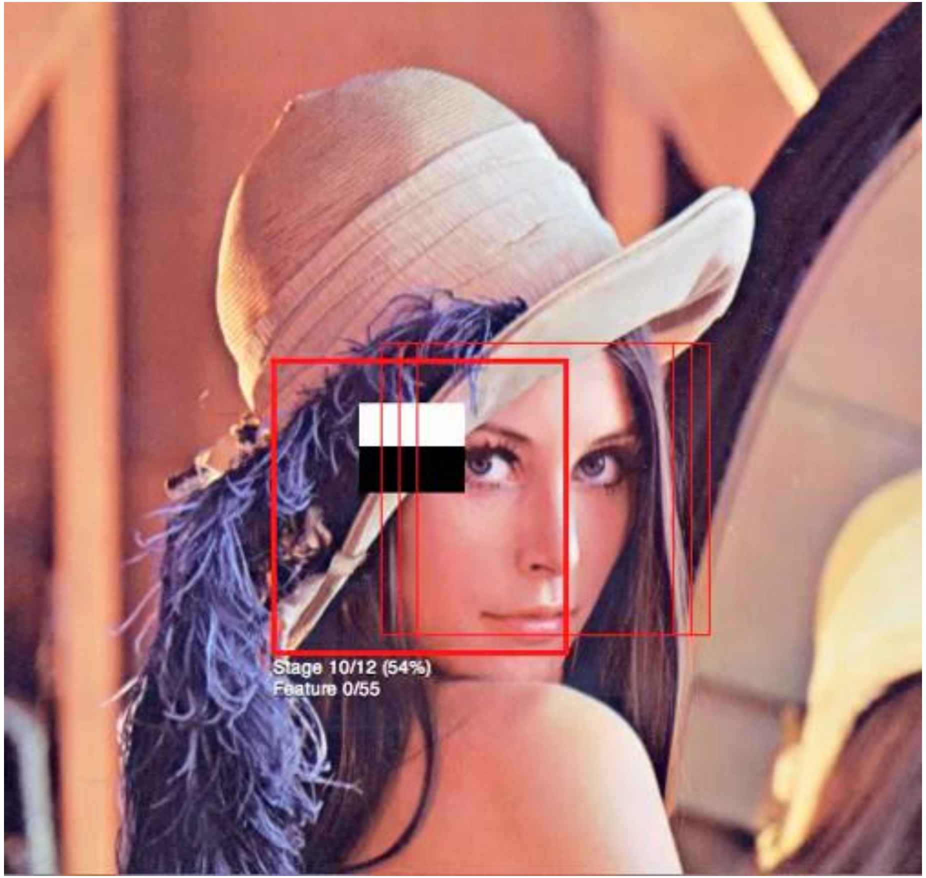
     where $e_i = 0$ if example x_i is classified correctly, $e_i = 1$ otherwise, and $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$.

- The final strong classifier is:

  $$h(x) = \begin{cases} 1 & \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0 & \text{otherwise} \end{cases}$$

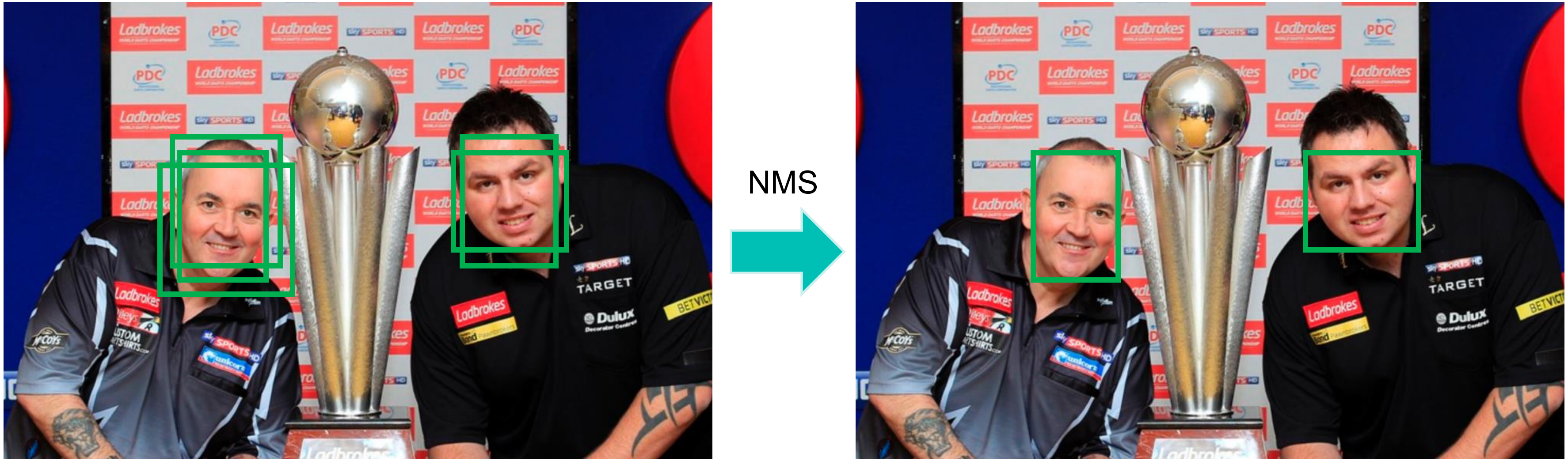
  where $\alpha_t = \log \frac{1}{\beta_t}$

Visualisation



Non-Maximum Suppression (NMS)

- A post-processing step to remove redundant detections.



NMS

Non-Maximum Suppression (NMS)

- A post-processing step to remove redundant detections.

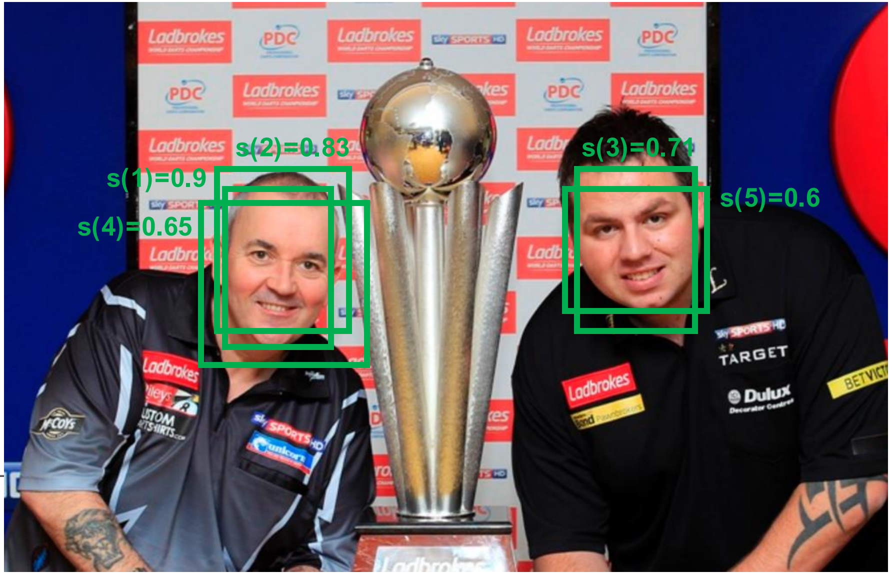
Algorithm 1 Non-Maximum Suppression Algorithm

Require: Set of predicted bounding boxes B, confidence scores S, IoU threshold τ, confidence threshold T

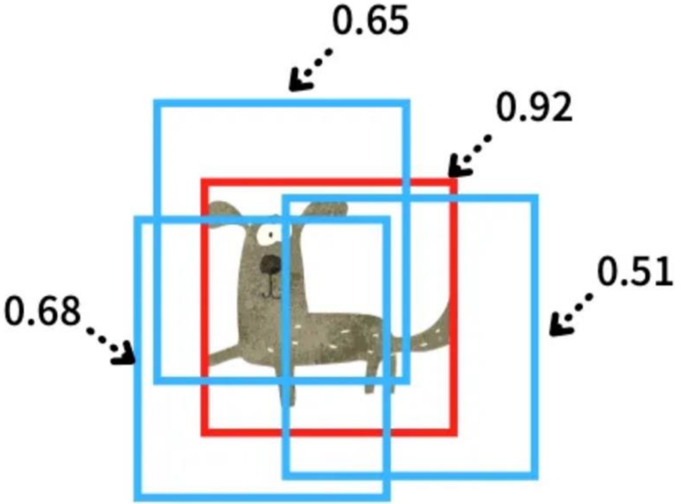
Ensure: Set of filtered bounding boxes F

```
$F \leftarrow \emptyset$
Filter the boxes: $B \leftarrow \{b \in B \mid S(b) \geq T\}$
Sort the boxes $B$ by their confidence scores in descending order
while $B \neq \emptyset$ do
    Select the box $b$ with the highest confidence score
    Add $b$ to the set of final boxes $F$: $F \leftarrow F \cup \{b\}$
    Remove $b$ from the set of boxes $B$: $B \leftarrow B - \{b\}$
    for all remaining boxes $r$ in $B$ do
        Calculate the IoU between $b$ and $r$: $iou \leftarrow IoU(b, r)$
        if $iou \geq \tau$ then
            Remove $r$ from the set of boxes $B$: $B \leftarrow B - \{r\}$
        end if
    end for
end while
```

https://browse.arxiv.org/pdf/2304.00501.pdf

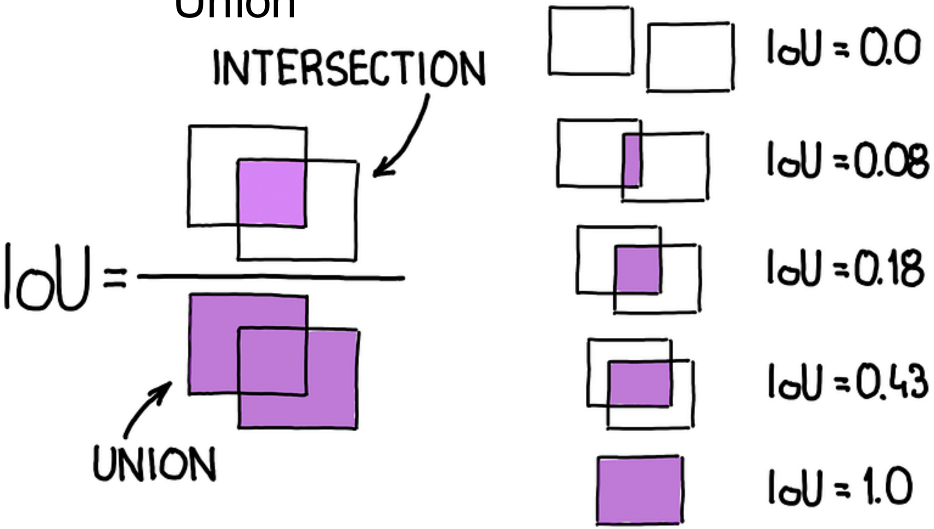


Non-Maximum Suppression (NMS)



1. take the largest probability box

Intersection over Union



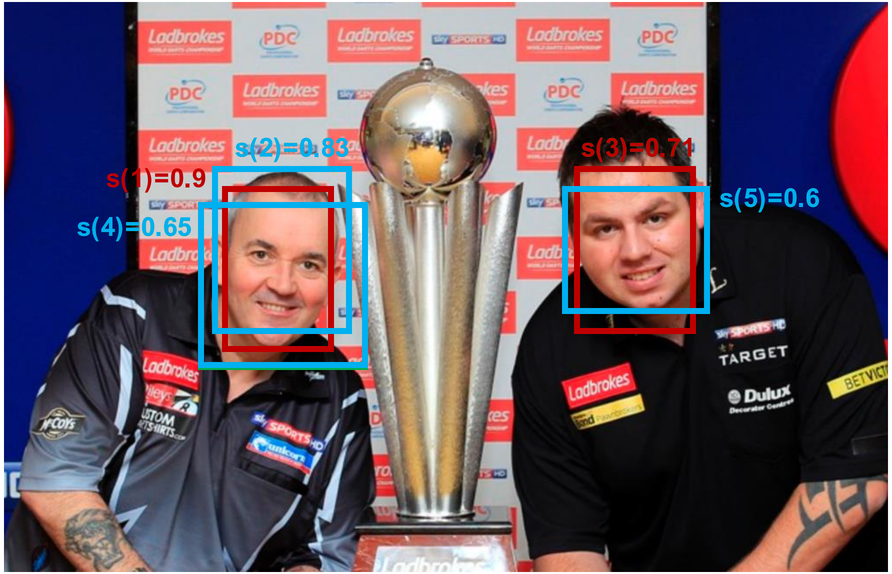
2. remove others with IoU score < threshold value.

Non-Maximum Suppression (NMS)

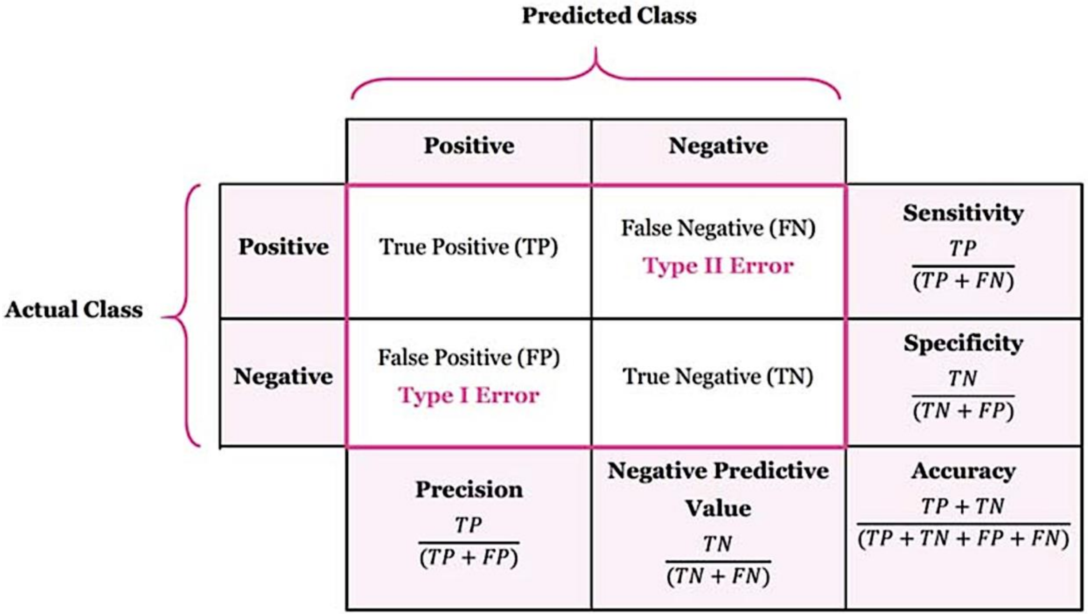
1. **Filter Bounding Boxes by Probability:**

2. **Sort Bounding Boxes by Probability:**

3. **Non-Maximum Suppression Loop:**

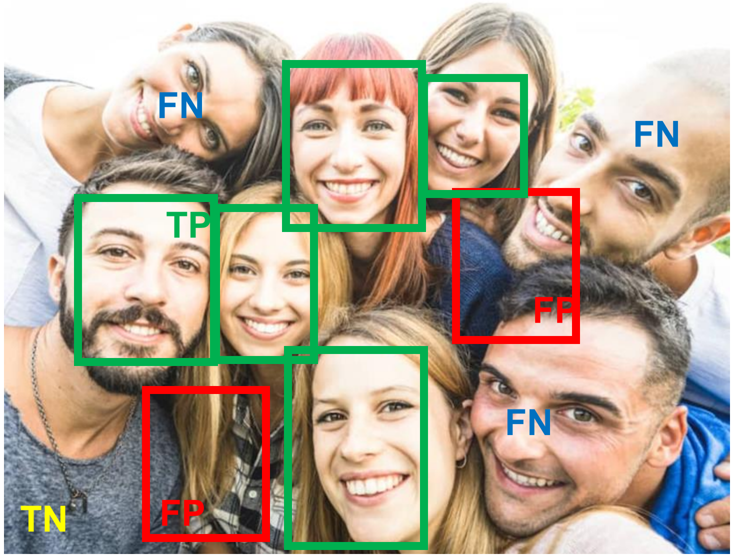


Performance Considerations



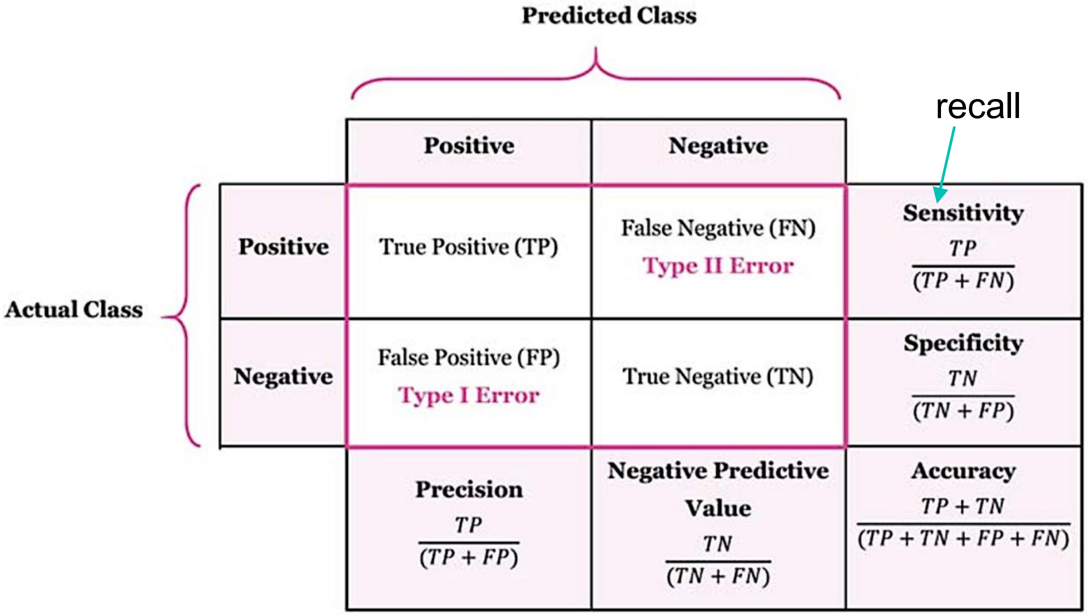
		Predicted Class		
		Positive	**Negative**	
Actual Class	**Positive**	True Positive (TP)	False Negative (FN) Type II Error	**Sensitivity** $\frac{TP}{(TP+FN)}$
	Negative	False Positive (FP) Type I Error	True Negative (TN)	**Specificity** $\frac{TN}{(TN+FP)}$
		Precision $\frac{TP}{(TP+FP)}$	**Negative Predictive Value** $\frac{TN}{(TN+FN)}$	**Accuracy** $\frac{TP+TN}{(TP+TN+FP+FN)}$

https://medium.com/@m.virk1/classification-metrics-65b79bfdd776



https://www.quickanddirtytips.com/articles/people-or-persons/

Performance Considerations



		Predicted Class		
		Positive	Negative	
Actual Class	Positive	True Positive (TP)	False Negative (FN) Type II Error	Sensitivity $\frac{TP}{(TP+FN)}$
	Negative	False Positive (FP) Type I Error	True Negative (TN)	Specificity $\frac{TN}{(TN+FP)}$
		Precision $\frac{TP}{(TP+FP)}$	Negative Predictive Value $\frac{TN}{(TN+FN)}$	Accuracy $\frac{TP+TN}{(TP+TN+FP+FN)}$

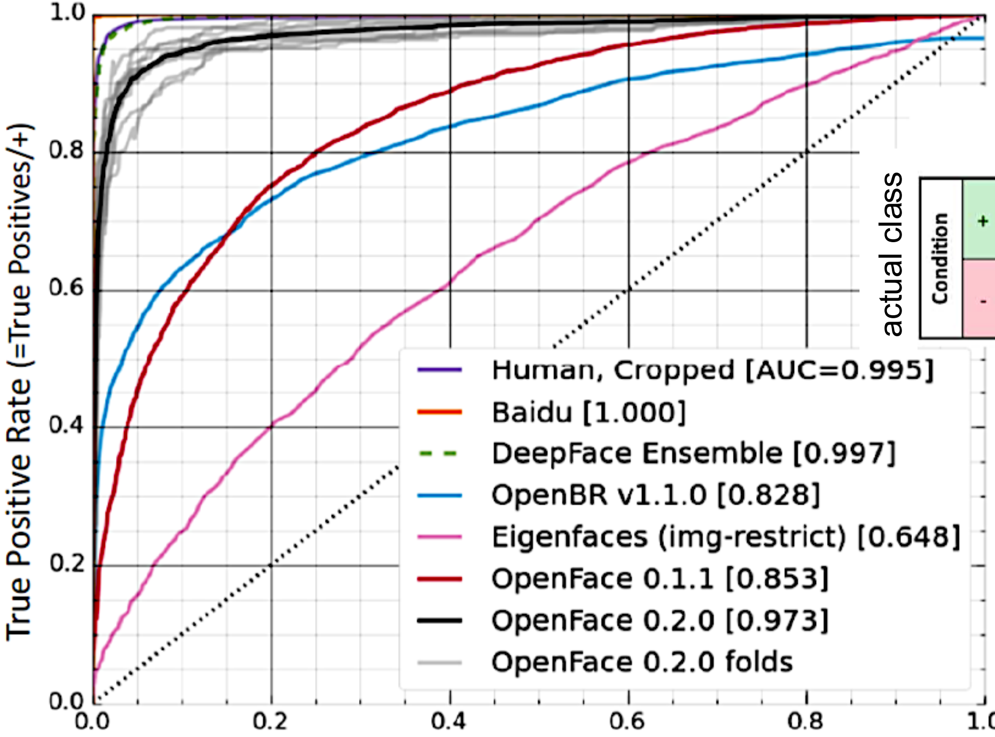
recall

https://medium.com/@m.virk1/classification-metrics-65b79bfdd776

F-score

$$F_1 = 2\frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} = \frac{2\text{tp}}{2\text{tp} + \text{fp} + \text{fn}}$$

Receiver Operating Characteristic (ROC) Curve



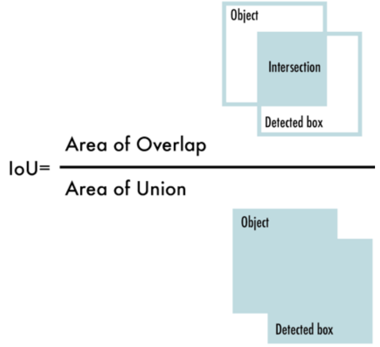
From B Amos (2016)

predicted class

actual class		Classification	
		Positive	Negative
Condition	+	True Positive	False Negative
	-	False Positive	True Negative

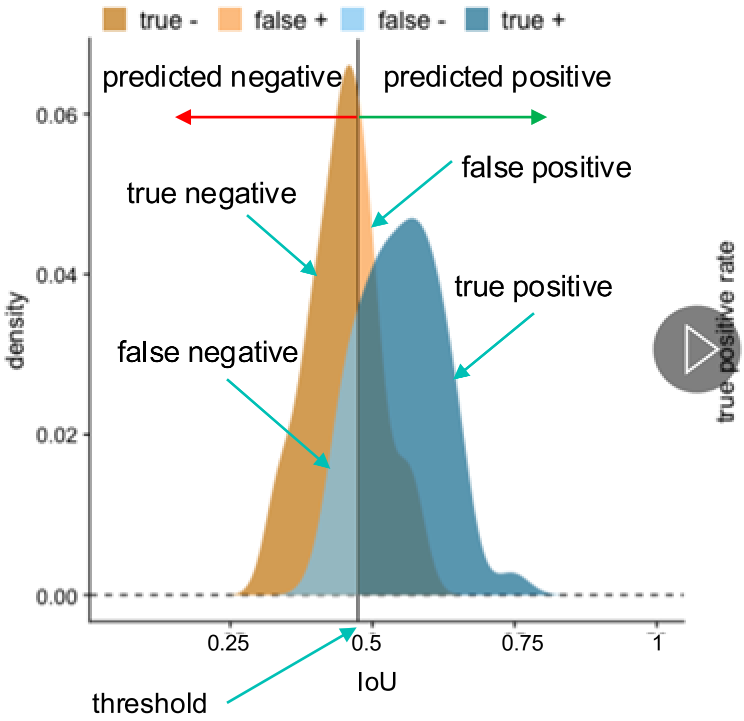
From R Wingate (2016)

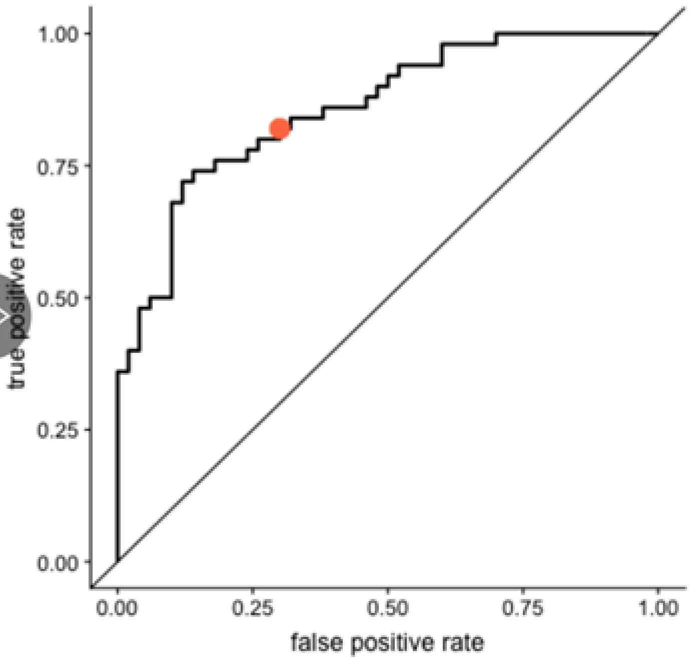
Intersection over Union



$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$

Receiver Operating Characteristic (ROC) Curve

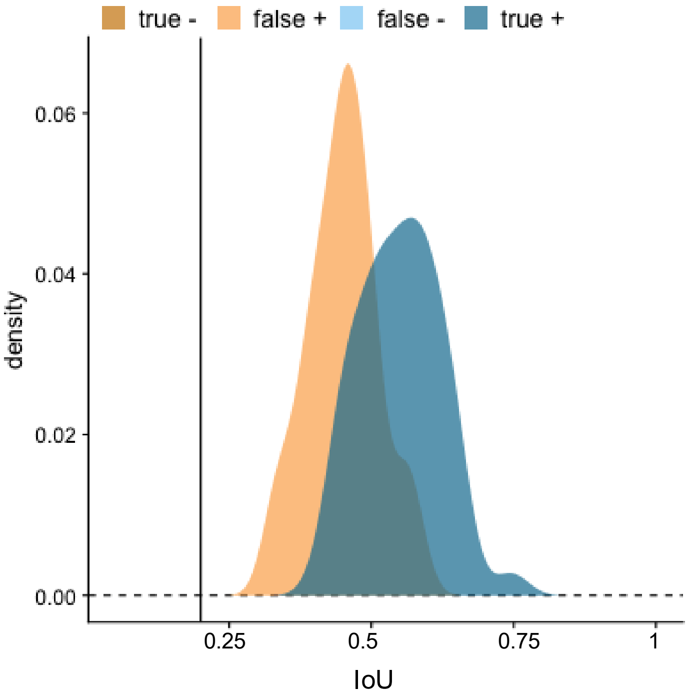


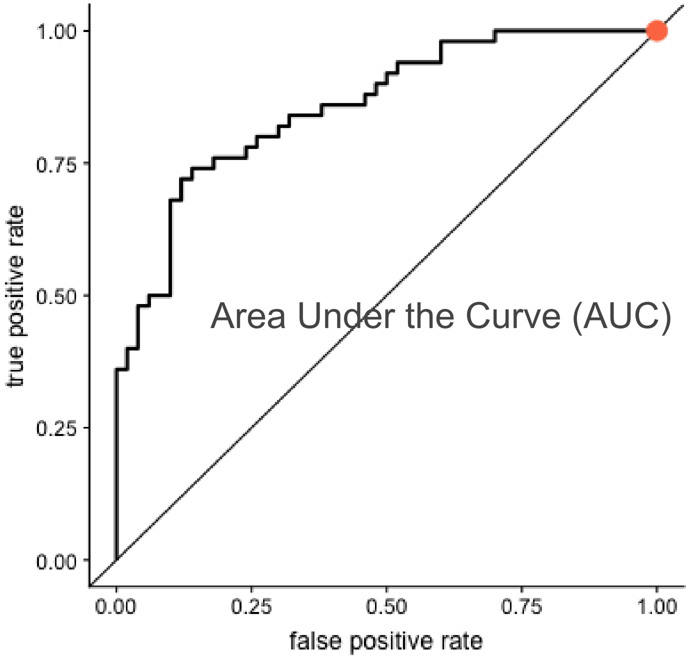


blue: actual positives (e.g. faces)
yellow: actual negatives (e.g. background)

https://paulvanderlaken.com/2019/08/16/roc-auc-precision-and-recall-visually-explained/

Receiver Operating Characteristic (ROC) Curve

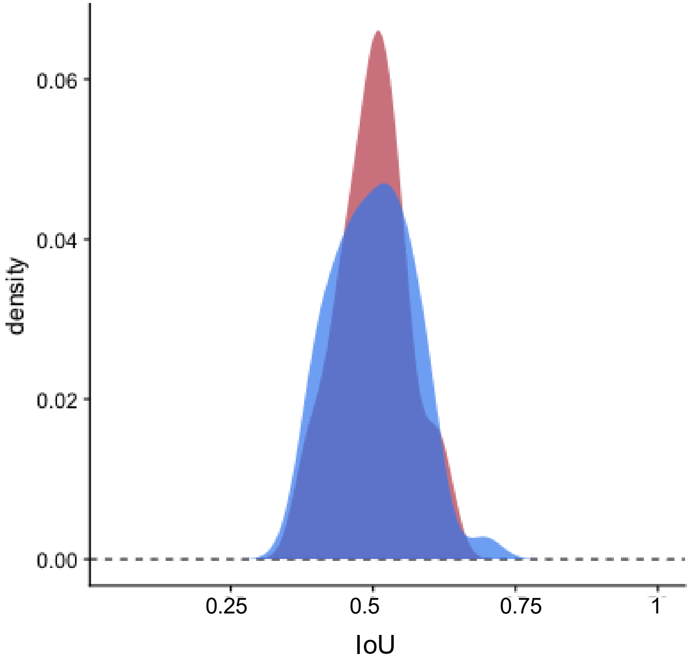


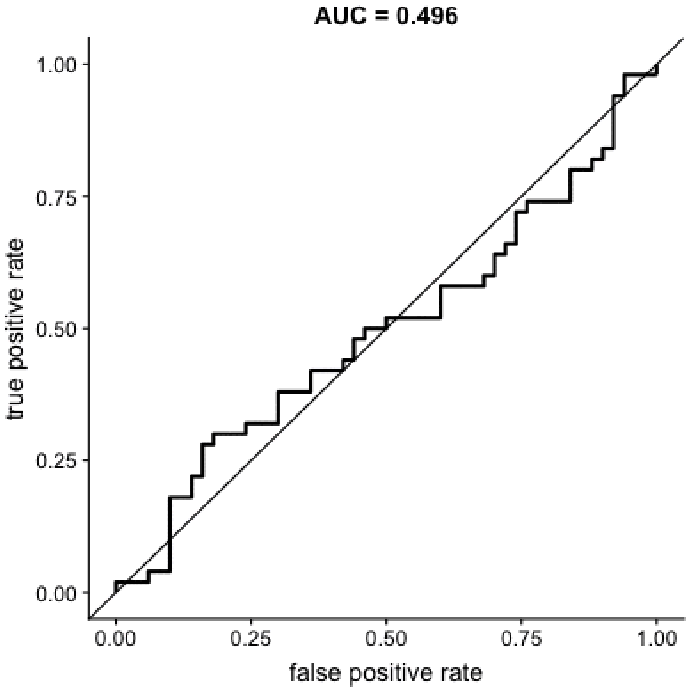


blue: actual positives (e.g. faces)
yellow: actual negatives (e.g. background)

https://paulvanderlaken.com/2019/08/16/roc-auc-precision-and-recall-visually-explained/

Receiver Operating Characteristic (ROC) Curve





https://paulvanderlaken.com/2019/08/16/roc-auc-precision-and-recall-visually-explained/

Next

Stereo : Epipolar Geometry