**FINAL**  **PROJECT REPORT**

**Nhận dạng-18KHMT**

**2020 - 2021**

# 

# *Giáo viên hướng dẫn: Lê Hoàng Thái-Võ Hoài Việt*

|  |  |  |
| --- | --- | --- |
| STT | MSSV | Họ và Tên |
| 1. | 18127158 | Lê Thành Nam |
| 2. | 18127190 | Lê Hồng Quang |

Thông tin nhóm:

# Introduction Face Detection:

* Face detection is the first step in automatic face recognition. Its reliability has a great influence on the performance and usability of the entire facial recognition system. With an image or a video, an ideal face detector is capable of identifying and positioning all current faces regardless of their position, proportions, orientation, age, and expression. Moreover, detection must be carried out not depending on the lighting conditions, image content and video.
* Examples face(top) and nonface(bottom):



# AdaBoost-Based Methods:

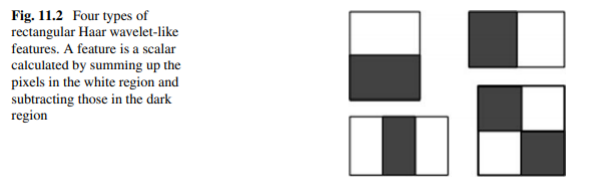
* With AdaBoost learning, a complex nonlinear strong classifier HM(x) is built as a simpler linear combination ofM.



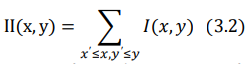
* Here x is a model that needs to be classified, hm(x) is Mweak classifiers, αm ≥ 0 are the combined values in R.In discrete version, hm(x) receives a discrete value in therange {-1,+1}, while in the version with real value , the output of hm(x) is a numberin R. HM(x) is the real value but the label prediction for xis obtained as ŷ(x)= sign[HM(x)].
* The AdaBoost learning process aims to learn a series of the best weak classifiers hm(x)and combining the best weights αm. A set of N labeled training examples {(x1, y1 ), ... , (xN, yN)} is assumeed to be available, here yi ∈ {+1, −1} is the class label for example xi ∈ R  n . A distribution [w1 ... wN] oftraining examples, here wi is associated with a training example (xi , yi ), which is calculated and updated throughout the learning process to show the distribution of training examples.
* After repeating m, more difficult examples ofclassification( xi , yi ) are given larger weights wi  (m) so that when at the iteration m+1, there will be more emphasis on these examples. AdaBoost assumes that a procedure is available to learn a weak classifier hm(x) from trainingexamples, with distribution [wi  (m) ].

1. **Local features**

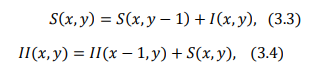
* Viola and Jones propose four basic types of radioless features for face detection.
* A block feature is located in a subregion of a subwindow and differs in shape(aspect ratio), size, and position inside the subwindow. With a subwindow size of 20x20 , there can be tens of thousands of such features for different shapes, sizes and positions. Feature k , take a directional value zk (x) ∈ R , which can be considered a transformation from n-dimensional space (n = 400 if a face example x is of size 20×20) to real line. These numberless numbers form a set of overcomplete features for low-dimensional facial patterns in essence.



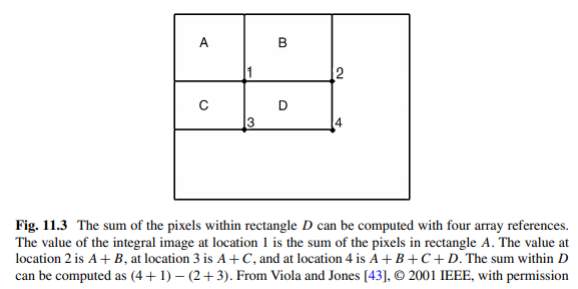
* A Haar-like feature can be viewed as a mask consisting of three values: 1 for pixels in the white area of the feature, - 1 for dark area pixels in the feature and 0 for pixels outside the feature area. Masks are the same size as subwindows. If we stack the pixels of a subwindow into a vector x, and arrange the mask associated with a feature into a vector m, this specific feature will have a feature value of mT x.
* Image integral II(x, y) at position x, y contains the sum of the pixels above and to the left of x, y, defined as follows:

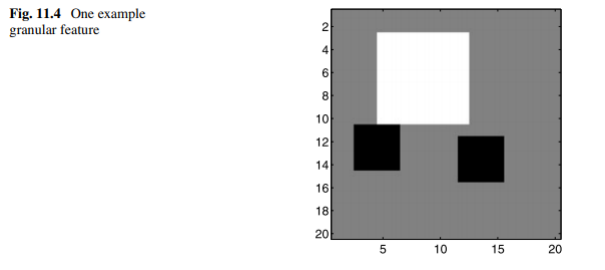


* The image can be calculated during a transfer through the original image by using the following recursive system pair:

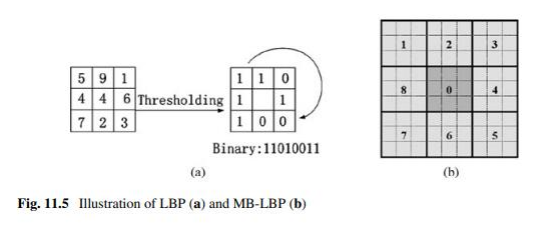


* Where S(x,y) is the sum of accumulated rows, S(x,−1) = 0 and II(−1,y) = 0.Using image analysis, any sum of the rectangles can be calculated in four array references, as illustrated in Figure. 11.3 . The use of image analysis leads to significant savings in calculations for features of different locations and sizes.
* With image integral, the change in intensity ina straight line D of any size and any position can be calculated effectively , forexample VD = √V ∗ V with V=(4+1)- (2+3) is the sum located in D , and the standardization of simple intensity can be done by dividing all pixel values in the subwindow by the variable.





* Figure 11.4 shows an example of the detailed features used. The mask corresponds to these specially structured detail features: most of the values of the mask are 0, while some non-duplicate square sub-regions in the mask have a value of all +1 (or all - 1). Square sub-regions are said to be 1 × 1, 2 × 2, 4 × 4 or 8 × 8, so that image analysis tricks can be used to quickly calculate detailed features.
* Subwindow in Pictures. 11.4 has a size of 20 ×20, so a feature will be equivalent to a 20 × 20 mask, where gray pixels correspond to the value 0, sub-region 8 × 8 corresponds to the value +1 in the mask and two sub-regions 4 × 4 correspond to the value - 1. By pre-calculating three image analyses for sub-regions of sizes 2, 4 and 8 respectively, detailed characteristic values can be calculated quickly.



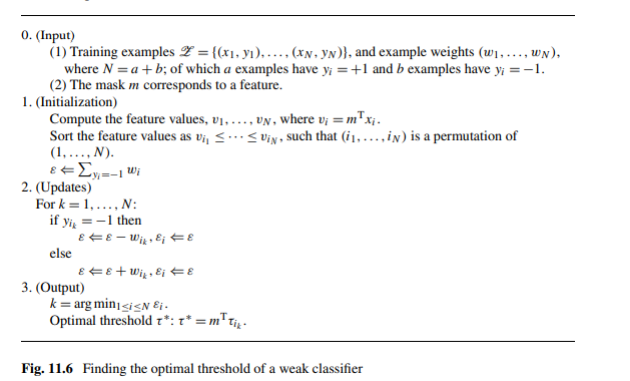
* Recently, other local features have also been proposed for use in facial recognition, where the local binary model (LBP) feature has shown promising results. As shown in The Picture. 11.5 (a), the lbp algorithm initially compares a pixel with its 8 neighbors, creating a '1' bit if the neighboring pixel has a higher strength value (and a bit '0' if vice versa). The LBP value is then a combination of these 8 bits. Multi-block LBP (MB-LBP) generalizes LBP by comparing the average of the intensity of the image in a region instead of comparing the single image intensity (illustrated in Figure 11.5(b)).

1. **Learning weak classifiers**

* As mentioned earlier, the AdaBoost learning process aims to learn a sequence of weak classifiers hm(x)and a combined weight of αm in (3.1). It solves the following three basic problems: (1) learning effective features from a large feature set; (2) formulate weak classifications, each based on one of the selected characteristics; and (3) promote weak classifications to build a strong classification.
* AdaBoost assumes that a "weak learner" process is available. The task of the procedure is to select the most important feature from a set of candidate features, based on the current strong classification learned so far, then build the best weak classification and combine it into the existing strong classification
* For discrete AdaBoost, a stump can be built in the following way. Suppose that we have built M - 1 weak classifier { hm( x) | m = 1,. . . , M - 1} and we want to build hM( x). The root hM( x) ∈ {- 1, + 1}is determined by comparing the selected feature zk∗( x) with a threshold of τk∗ as follows:



* In this form, ℎ𝑀(𝑥) is defined by two parameters: feature type 𝑧𝑘 ∗ và Threshold τ𝑘 ∗ .Both can be identified according to several criteria, for example, (1) minimum weight classification error, or (2) the lowest false alarm rate given a certain detection rate.
* Wu and his associates point out that only N+ 1 possioble τk values need to be evaluated and evaluated each τk  as O(1)only if we sort the feature values for zk to come first.This method to determine the optimal threshold of a weak classification is illustrated in the Figure. 11,6.
* Most of the calculations in Figure. 11.6 is used in the initials section. An important observation is that feature values only need to be calculated and sorted once, since they do not change during the AdaBoost process although the weight w changes at each iteration. By storing the arranged feature values for all features in a table, the AdaBoost training time is reduced from a few weeks to several hours .
* More features often result in higher detection accuracy. However, it also means that the sorted feature value table may be too large to be stored in the primary memory. Pham and Cham build weak layers using the average and standard deviations of feature values. With one feature, it is associated with the mask m and weight AdaBoost w ( M - 1), the average feature value is , i-, w-i-M −1.,m-T., x-i.., with xi as a set of training examples.



* The image integral trick can be used to accelerate the calculation of average characteristic values and standard deviations, i.e., providing a way to use the structures in the mask m and calculate mT x quickly. For x to be a subwindow image in the form of stacked vectors, and y is the corresponding analysis image,from (3.2 ),the transformation that produces y from x is linear, i.e. there exists a square matrix B such as y = Bx, and mT x = mTB −1  y.The average feature value is:

****

* In (3.6), the weighted average integral image can be calculated in linear time, and the transformed mask mTB −1 is sparse because of the structure in the mask m. Therefore, the average feature value can be calculated very quickly. Similarly, standard deviations (weighted) can also be calculated quickly.
* This method is faster than checking all that can be τk and has much smaller storage requirements. It is reported that the training speed is about twice as fast as the algorithm presented in the Figure. 11,6 . This method has much less storage requirements and therefore can train a strong classification with more local features

1. **Learning strong classifiers using AdaBoost**

* AdaBoost learns a series of weak classifications and motivates them to become a powerful H M efficiency by minimizing the upper limit on classification errors achieved by H M. The limit can be infernable as an exponthial loss function.

**(3.7)**

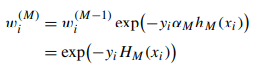
* That i is the index for training examples. AdaBoost builds hm(x) (m = 1,...,M) by phased mitigation (3.7).With HM−1(x) = current and new weak classification hM , the best combination factor αM for the new strong classification HM(x) = HM−1(x) + αMhM(x) minimizes costs.

(3.8)

* Với ,

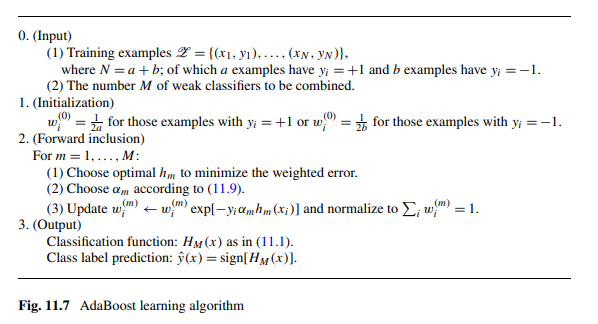
(3.10)

* Each example is rebalanced after repetition, wi(M−1) updated according to classification performance of HM:

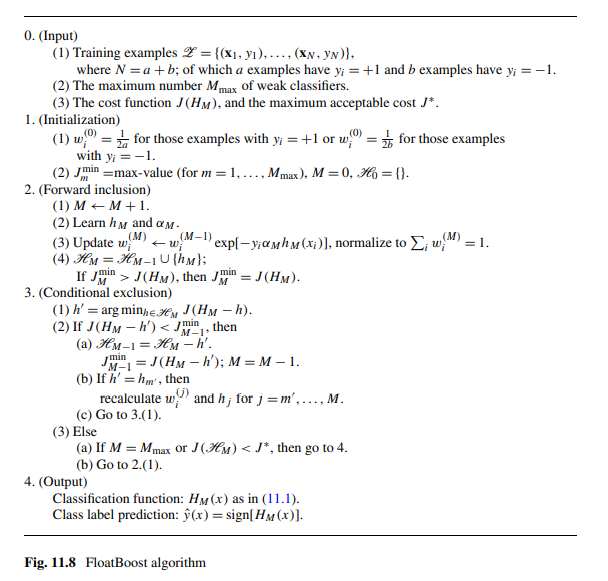
(3.11)

1. **Alternative Feature Selection Methods**

* In enhanced-based methods, weak classifications hM and their associated weight αM are simultaneously defined: hM is selected to minimize certain objective values (e.g., the weighted errorrate of the feature), andαM is a function of the target value. Wu points out that if these tasks (learning hM and setting αM) are separated intotwo sedning steps,it is possible to get a more accurate powerful classification than HM.
* Different methods can be used to select features and train weak classifications. Besides AdaBoost and other enhanced variants, showing a greedy Transition Feature Selection (FFS) method can successfully select a sub-set of features from a large feature group and learn the corresponding weak classifications. In accelerated methods, a feature is selected if its corresponding weak classification has a minimum weighted error rate. FFS uses another selection criterion that is directly related to the performance of a strong classification. In the FFS, if a strong classification of part HM−1 has been built,an hM∗feature is selected only in repeating M if it leads to the strongest classification accuracy, that is, H M−1 ∪ h M ∗ has the highest accuracy of all possible hM . FFS uses a majority of votes (i.e αi = 1 for all i). A table of feature values is stored to ensure rapid weak classification training. FFS trains faster than the AdaBoost method and achieves comparable detection accuracy but slightly lower than AdaBoost.



* The FloatBoost Learning Process is shown in Figure 11.8. It consists of several sections: training input, initial training, including transitions, conditional exclusions and outputs. In step 2 (including transitions), the most significant weak classifications are now added one by one, just like in AdaBoost. In step 3 (conditional exclusion), FloatBoost removes the least important weak taxa from the HM set of current weak classifications, depending on the removal conditions resulting in a lower cost than J min M−1. Suppose that the weak classification is eliminated as m′ th in HM , then hm′,...,h M−1 and αm must be learned. These steps are repeated until further removal is not possible.



1. **Asymmetric Learning Methods**

* Viola and Jones proposed the AsymBoost method , which is a modification of the AdaBoost algorithm. The essence of AsymBoost is to focus more on positive examples by changing the weight update rule (3.11) to

**(3.12)**

where C = if yi > 0 and C = if yi < 0 and K > 1 are a parameters that measure that asymmetry. We assume that the motivated learning process will repeat the T rings. In the T rings of the AdaBoost algorithm, positive examples are constantly assigned a higher weight than negative examples.

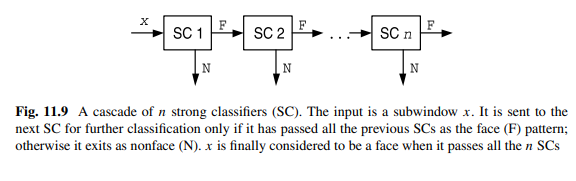
* Wu proposes the following learning objective for strong classifications in one tier: "for every node, design a classification with very high (e.g., 99.9%) detection rate and only moderate (e.g. 50%) false positive rate." Linear Asymmetric Classifier (LAC) is designed to α achieve this goal.
* With weak classifications h1, h2,...,hM , an example x is mapped to a reaction vector h(x) = (h1(x), h2(x), . . , hM(x)). LAC calculates the distributions of vectors h(x): μ+ and Σ+ are the average and codential matrix of h(x) when x is a set of faces. Similarly, μ− and Σ− are average matrix and varication is calculated using non-air sides. It is in point that the following LAC solution vector α∗ ∈ R M is globally optimal for casing learning goals according to certain reasonable assumptions:

**(3.13)**

Another way to establish α vector is to use Fisher's Discrimination Analysis (FDA). Experiments showed that the use of LAC or FDA to establish α vectors continuously improved layer detection accuracy, regardless of the weak classifications selected and trained with AdaBoost or FFS.

1. **Cascade of Strong Classifiers**

* Viola and Jones continued to expand this idea by training a floor consisting of a series of powerful classifications, as illustrated in Figure 11.9. A strong classification is trained using non-present examples of bootstrapped passing through previously trained falls. Usually, 10 to 20 strong classifications are tiered. On the other hand, subwindows do not pass a strong classification that is not further processed by the next strong classification. This strategy can significantly accelerate detection and reduce false alarms, with a slight sacrifice in detection speed.
* Various improvements have also been proposed for the floor structure. Xiao et al. argues that historical information is useful during tiered training, that is, we should not ignore the information presentin the strong classifications SC1,...,SCM−1 when we train the strong classification M-th SCM (that is the fact in Figure 11.7). The Boosting Chain framework is proposed to combine such historical information: the strong classification SCM−1 is considered the first "weak classification" in SCM . This modification to the floor frame reduces the number of weak classifications required and increases detection accuracy.



* Another attempt to modify the flooring frame is the soft waterfall method or similar ideas. Soft cascade is an extreme cascade structure: a strong "tedious" classification consisting of many weak classifications, similar to the strong classification in the cascade frame. Let c1(x), . . cT(x) be weak classifications that form a soft waterfall

(3.14)

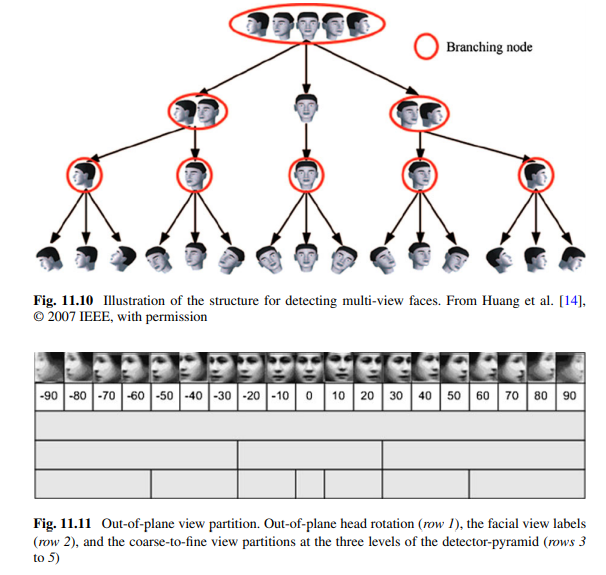
* A soft waterfall links an RT rejection threshold for each strong classification in part . If Ht(x) < rt , the input subwindow x is rejected as nonface,and the weak classification c t +1,...,cT is not evaluated. In other words, a soft waterfall similar to a tier structure requires only 1 weak classification per node. However, since historical information is preserved in Ht, soft falls achieve high detection performance.
* After the weak classifications c1,...,cT are trained, the soft waterfall method rearranges the order of these weak classifications. This step is performed using a separate set of authentication examples. An optimal order of weak classification and rt rejection threshold is selected to minimize both detection errors and the cost of calculating testing time .

# Dealing with Head Rotations:

* Multiview face detection will be able to detect face without forehead. Face detection methods typically handle two main types of U-turns: (1) off-plane rotation (left-right); (2) rotate on the plane
* Rowley proposed using two neuro-network differenties to detect rotating front faces in the plane. The first is the router network, which is trained to estimate the direction of a hypothetical face in the subwindow, although the window may contain a non-face model. Network inputs are strength values in a 20-× 20 that are pre-processed. The rotation angle is represented by an array of 36 output units, of which each unit represents an angle range. With orientation estimates, subwindows are derotated to make the face potentially vertical. The second neural network is a normal, vertical front-facing detector. Within the framework of floor detectors, detector pyramids have been proposed to detect and erest faces in different locations and have achieved the most modern detection performance

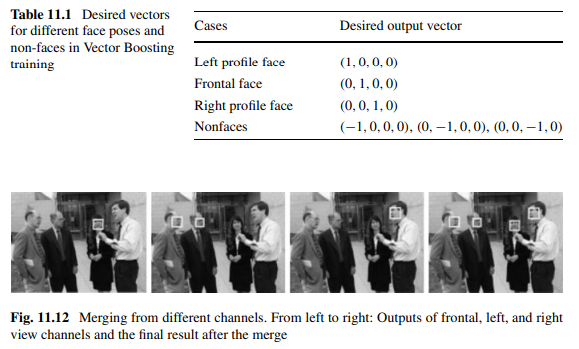
1. **Hierarchical Organization of Multi-view Faces**

* Width-First-Search structure of Huang (Figure 11.10) handles in-flight and off-plane rotations at the same time. They manually divide the face range into 15 different styles and arrange such styles according to the four-level tree structure. The top-level tree button includes all face nodes. The second level contains 3 buttons, corresponding to the left, front and right profile sides. The third level continues to fine-tune into 5 nodes, where the left and right profile sides are divided into 2 different nodes based on the angle of rotation outside the plane. The first 3 levels handle extra-plane rotations. Each node in the third level is divided into 3 nodes at the last level, processing rotation angles in different planes.
* The processing plant structure rotates beyond the plane in Θ = [−90◦,+90◦] and rotates in the plane in Φ2 = [−45◦,+45◦]. The full rotation range in the plane Φ = [−180◦,+180◦] is covered by rotating features 90 °, 180 ° and 270 °. It has been noticed that for these specific rotation angles, rotating features equivalent to m mask rotation involves a feature equal to a corresponding angle. The rotation features are more efficient than rotating the image.



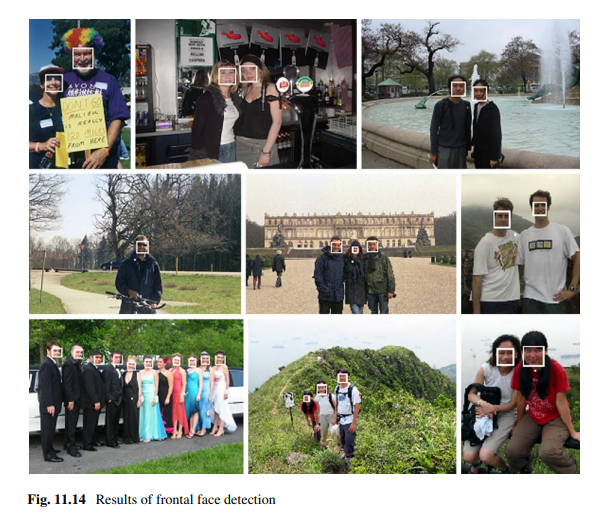
1. **From Face-Pose Hierarchy to Detector-Pyramid**

* A multi-view face detection detector pyramid can be derived directly from the facial positioning decentralized system. Each node in Figure 11.10 corresponds to a strong classification. For example, the root node determines whether a subwindows contains a left, front, or right profile face. It is important to note that multiple child nodes of the same mother node can be activated simultaneously (that is, look for the width first of the tree structure). Huang et al. made this choice based on the following argument: different face-to-face relationships compete with non-human people, and discrimination between them is less important (except at the last level). This hypothesis requires nodes in the top 3 levels to be multilayer classification and the proposed Vector Boosting algorithm to meet this particular requirement.
* For example, a part profile face with a 45° extra-plane rotation angle can be detected using both the right profile face button and the front button in the second layer. To achieve greater detection accuracy, it is reasonable to further examine two seedlings derived from both nodes. In the Vector Boosting classification for the root node, the desired output is the desired vector and vectors for different cases summarized in Table 11.1.
* Figure 11.11 leads to another detector pyramid. Instead of using a multilayer enhancement algorithm that generates vector output, use a strong binary realboost classification k if a node has a k-child node. Multiple heads of a node can be activated (that is, further examination of seedlings originating from a child node) if more than one positive output binary RealBoost classification. At the last level, multiple leaf buttons (corresponding to different face positiones) can work for a subwindow. The faces detected by seven channels at the last level of Figure 11.11 are united to get the final result. This is illustrated in Figure 11.12.



# Performance Evaluation

* Facial detection results from images are influenced by two basic factors: face/face classification and post-processing (merge). To understand how the system works, it is recommended to evaluate two components separately, with two types of test data. The first includes fixed-sized face icons (as used for training). This process is aimed at evaluatiing the performance of the face/face classification (including preliminary processing), without being affected by the ingest. The second type of test data includes normal images. In this case, the results of facial detection are affected by both the trained and united classification; overall system performance is evaluated.

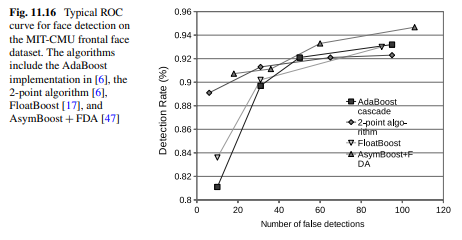


1. **Performance Measures**

* Face detection performance is primarily measured by two speeds: the correct detection speed (minus 1 minus the missed rate) and the wrong alarm speed. Performance can be observed by drawing on the characteristic curves of the operation of the machine (ROC).

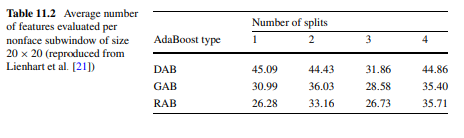


* The false alarm speed is calculated as a percentage of non-face subwindows but mis classified as faces. However, the number of false detections (remaining after multiple detections ingest) is a more relevant one as it reflects the effectiveness of post-processing and links directly to the final output of the facial detection system. Although false alarm speeds often positively correlation with the number of false findings, many authors are recently reporting the number of false detections (after post-processing) in the X axis of the ROC curve. Figure 11.16 shows several examples of ROC curves, including four recently evaluated methods on the standard MIT-CMU front-facing data set.



1. **Comparison of Cascade-Based Detectors**

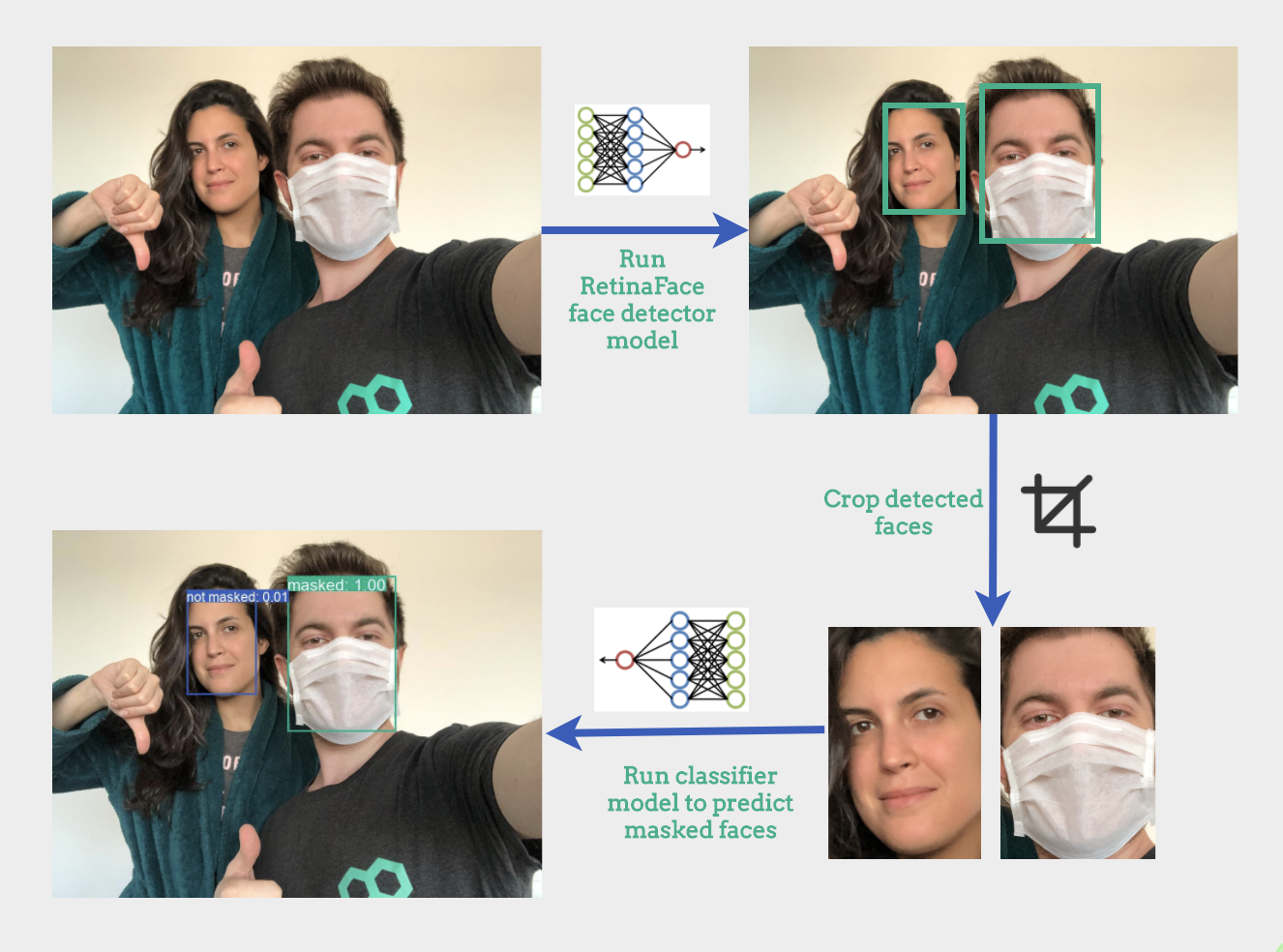
* Since tier-based methods (with local features) have so far provided the best facial detection solutions in terms of statistical proportions and speed, the following provide a comparative review of various enhancement algorithms (DAB: Discrete AdaBoost; RAB: Real AdaBoost; and GAB: Gentle AdaBoost), various sets of training preparation and classification of various weak. Results of providing experimental references to facial detection engineers.
* Enhance the algorithm. Three 20-stage tier classifications have been trained with DAB, RAB, and GAB using Viola and Jones' Haar-like feature set and stump as weak classifications. It is known that GAB outperforms the other two enhancement algorithms. In addition, a smaller resizing factor for scanning images benefits a high detection rate.



* Detection speed. Table 11.2 compares the weak classifications of CART plants with different number of nodes in terms of the effectiveness of removing nonface subwindows. RAB is the most effective. There are also reports that rab has the fastest detection speed. The reuse of historical information is confirmed to be effective in reducing testing time.
* Like Haar and other local features. Experiments and other works show that while the larger Haar-like feature set makes it more complex in both time and memory during the learning enhancement phase, achieved during the detection phase. Using the same training period, (with 295 920 local features) false alarm reporting is about 5% lower than the method in Figure 11.6 (with 40 000 local features). Other local features such as MB-LBP also show excellent detection results.
* Subwindow size. Subwindow sizes varying from 16×16 to 32×32, which have been used to detect faces. Experiments showed that subwindow sizes of 20 × 20 reached the highest detection speed with an absolute number of false alarms of 5 to 100 on the CMU tester of the front. Subwindow size 24 × 24 works better for false alarms under five.

# Face-mask Detection:

* Masks are important in minimizing the spread of Covid-19, and are highly recommended or even mandatory in many situations. In this project, we developed a process to detect faces that have not yet been exposed in the image. For example, this can be used to warn people not to wear masks when entering the building.
* General Model Workflow:



1. Data collection:

* Our training data is based on[VGGFace2 dataset](http://www.robots.ox.ac.uk/~vgg/data/vgg_face2/). This data set provides a set of faces captured in nature with different ethnicities, ages and emotions. We use 4941 images of this data set and apply artificial masks to half of them. This data will be used to train our mask/no\_mask classifier.
* Our authentication and testing data includes images of people with and without masks that we collect from various sources that provide images with a license to allow (e.g. [pexels.com](https://github.com/datarootsio/face-mask-detection/blob/master/www.pexels.com), [unsplash.com](https://github.com/datarootsio/face-mask-detection/blob/master/www.unsplash.com)). We have manually anidented all the faces in the collected images and labeled them as [concealed](https://www.makesense.ai/) or not (using the makesense.ai annotation tool). We collected 273 images containing 524 faces (246 masks and 278 without masks). The images are divided 50/50 on the authentic and tester. An overview of the collected data and the corresponding url and basic fact annotation can be found in [test\_validation\_metadata.csv](https://github.com/datarootsio/face-mask-detection/blob/master/data/test_validation_metadata.csv).

1. Data preprocessing:

* The labeled data of the masked face is difficult to come by, which is why we decided to place overall still limited real masked faces that we have collected in addition to confirmation and testing. Artificially created masks used for training are created as follows:

+Face detection in images

+Find face molds, more specifically we need the position of the nose and chin

+Apply images of masks to the face with position based on face molds

* This strategy is based on the description that you can find in the prajnasb/observation archive. We apply 13 masks with different shapes and colors to create training data that you can find in data templates/ masks. Below you can see an example of a mask that is being applied artificially.

1. Face detection:

* We used the RetinaFace facial detector to extract the face as it is modern in face localization in nature and works in real time on a single CPU core (Deng et al.). We used a pre-training and deployment model available at the RetinaFace repository. Note that we used the RetinaNetMobileNetV1 model, which is much faster than RetinaNetResNet50 and DSFDDetector.

1. Masked or Not Masked Classification:

* The model that we train to distinguish between the mask and the non-masked cross section consists of a MobileNetV1 base followed by a fully connected layer and a final output layer with sigmoid activation. We use pre-trained weight ImageNet for the MobileNetV1 facility and only fine-tune the last 4 layers of the complete model. Only VGGFace2 does not wear masks and data artificial masks used for training. We use two sets of authentication:

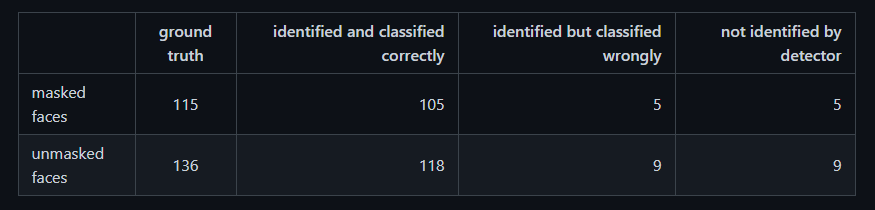
+ One includes data without mask and VGGFace2 artificial mask

+ A face consisting of faces that are masked and not really concealed from the set that we have collected on our own.

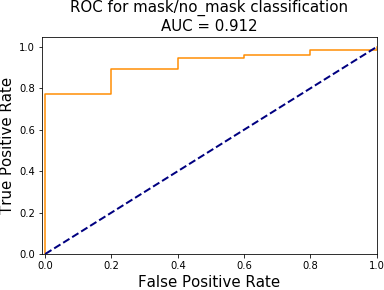
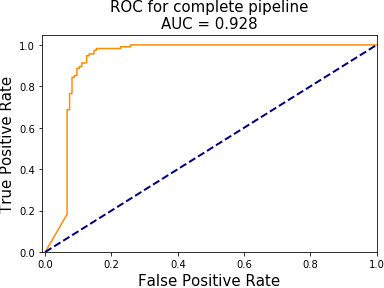
* This allows us to monitor performance on artificial and practical data separately. The final test set evaluation is performed only on real data.

1. Model Performance:

* Overall evaluation:
* Our 135-image tester contains 251 faces, of which 136 are not exposed.
* The following table summarizes the performance of the complete pipe (i.e. the face detector followed by the ererer). We apply the mask/no\_mask classification to the cut faces extracted by the face detector and compare the result labels with the ground truth labels of the appropriate truth-limiting boxes. A predictive face limit box matches the ground truth limit box if their intersection on union (IoU) > 0.5.



* The two most relevant indicators are true negative ratio (TNR) and false negative ratio (FNR). The first one shows us how many faces have not been discovered and how many times we have discovered the incorrectly identified face. 118 of the 136 unsolved faces were correctly identified, resulting in a true negative rate (TNR) of 87%. 5 out of 115 masked faces were incorrectly identified as not concealed, resulting in a false negative rate (FNR) of 4%. Note that faces that are not identified by the detector are not taken into use in these numbers. The pipeline also incorrectly identified 19 faces that did not match any of the faces on the ground.
* Although the previous statistics correspond to the mask / no\_mask classification threshold at 0.5, of course we can change this in exchange for a better trade-off between TNR or FNR. The figure below shows the ROC curve for the pipe. For the creation of this ROC curve, we looked at ground truth faces that were not detected by the face detector predicted to be masked. After all, the purpose is to detect a face that is not concealed: if the detector does not detect the face, it will have the same effect as predicting the masked face for most practical purposes. Faces detected by face detectors but do not exist in ground truth have not been taken into use in this ROC curve.



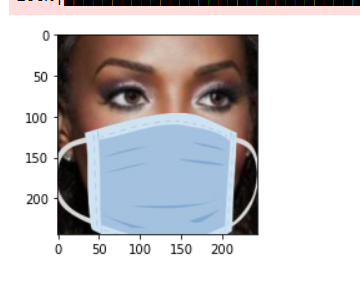
* Evaluation of the face detector:
* The facial detector accurately identified 94% of the ground real faces in the tester (i.e. for 237 of the 251 ground truth limit boxes there is a predictive limit box with IoU > 0.5). Of the 14 faces on the ground that it did not detect, 5 were masked and 9 were not wearing masks. The face detector also exported 13 limited boxes to the tester that did not correspond to the face, 6 of which were later classified as non-masked.
* Evaluation of the mask/no mask classifier:
* We have also evaluated the mask/non-mask classification separately based on our ground truth annotations. With a classification threshold of 0.5, this results in 90% accuracy. The figure below shows the ROC curve for the mask/mask-free classification:

# Results:



Use the no-wearer data set and wear a mask (real version) as a validation set

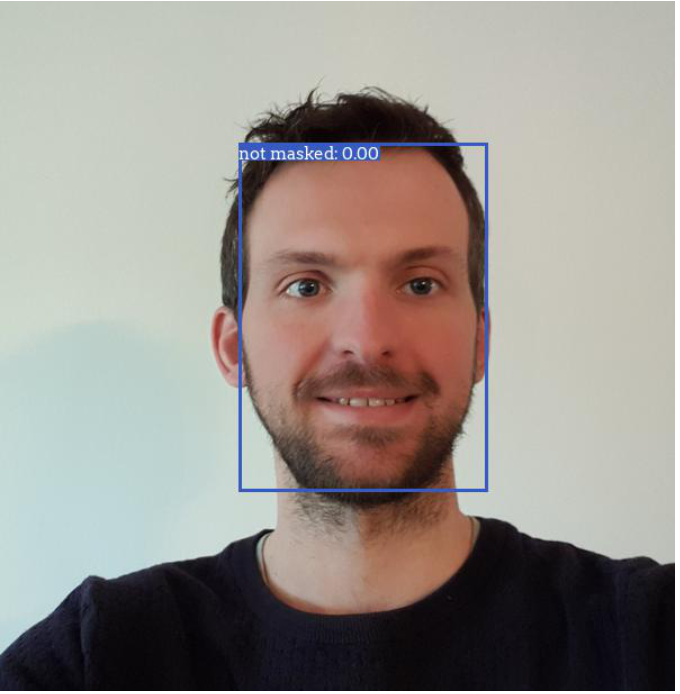
And shows the correctly predicted positive sample number and the wrong predicted positive sample number



Use images available in the Vggface2 dataset to create an image of the person wearing an "artificial" mask



* Test results of the "ex\_img\_annotated" image. Test run on the file "predict.ipynb"



* Input image test result: face-mask-detection/scripts/img /"toon\_original.jpg". On the file "predict.ipynb"

# References:

* + Dataroots,2021, <https://github.com/datarootsio/face-mask-detection>
  + Stan Z. Li ! Anil K. Jain,2011, Handbook of Face Recognition Second Edition