Global Journal of Computational Intelligence Research.
Volume 1, Number 1 (2011), pp. 21-30
© Research India Publications
http://www.ripublication.com/gjcir.htm

Face Detection in Still Images using Neural Network and Back Propagation Algorithm

Ajit Pratap Singh Gautam

Associate Professor, Department of Computer Application
Technical Education and Research Institute, P.G. College, Ghazipur, India

Abstract

Human face analysis for detection and recognition is a growing field of interest in computer science and research. Its importance is clear with its applicability in different fields of real life and is obvious from security reason to computer aided identification. Face detection is the first step towards face recognition. Here, a neural network- based face detection system is designed. This paper is an effort to present a neural network-based algorithm to detect frontal view of faces in gray scale images. A bootstrap algorithm for training the network is used which adds false detection to the training set with progress of training.

Introduction

Analysis of human face for detection and recognition is a growing field of interest in computer science and research. The importance of it is clear with its applicability in different fields of real life and from security reasons to computer aided identification. This, indeed, is a result of the demand of current multimedia technology inasmuch as MPEG group is more inclined towards object oriented representation and comparison.

Moreover, the problem of face detection is difficult altogether. Face detection is a problem in which the machine must be able to classify between face and non-face images. For a given image as input, the output declares if the input contains a face or not. If it is so, it gives the location where it occurs.

Basically, face detection is the first step in face recognition and also the most difficult one. Once face detection is complete the face recognition may be solved with the help of available algorithms. Face detection in a general image has no definite algorithm which ensures hundred percent accuracy. The problem of face detection is typically a machine learning problem assuming the human face to be represented by some non-linier function.

Among the heuristics based approaches, feature vector based approach, color histogram and color segmentation approach performed well for colored images, but they are working mostly for images under clear background. Instead, the machine learning approaches, where the human face is approximated as a non-liner function., performs better even under a complex background.

If way take gray-scale upright frontal image of a human face and cut out the average intensity of the face. That is the background, than that portion which varies heavily from their surroundings can be identified. The portion vary hugely from the surroundings are the requisite parameters that help us developing the necessary non-linear function of the human face. After the parameters to develop the non-linear function is available, for machine learning we are employing the following methods:

1. Neural Network based learning
2. Levenberg-Marquartd Algorithm
3. Support Vector Machine

Since all human faces are quite similar to look at, we can expect the learning modal to successful coverage.

System Design

We have used feed forward Neural Network and Back Propagation algorithm for learning process. Moreover, instead of going for fully connected network, we can use a partially connected network expressing the relation between input and output for different parameters. Advantageous to use a partially connected network over a fully connected network as we have better performance on time, memory and space by we dealing with lesser numbers of edges also in the hidden layer, the number of nodes decreases. Hence, calculation involved in this process gets further reduced and simplified.

The basic problem with face detection in a general image is to train the network for negative examples. This is because; the number of examples containing faces which represent the set of human face may be obtained and restricted in size. Whereas, finding a representative set of examples, not containing face is the negative example set, is rather impossible and can grows very large in size soon. Hence, we restrict the input image size for training the network to 20 X 20 pixels. The images containing a larger face can be scaled down detected, if a network can dectect the presence of the face under this restriction. Faces of smaller size cannot be identified by the network, but it seems practically to have very low chance of obtaining a face of even smaller size. This also restricts the size of the input layer of the network to 400 nodes.

We are using upright, frontal view of human face of a given size, where eyes are brought horizontal position by rotating it suitable and scaling the image, so the distance between center of the upper lips and the horizontal line joining the eyes is 12 pixel. This restricts an exposure to the most important part for identifying the human face and also helps decide the topology for the hidden layer of a network. For negative example, we first create some random pixel value images and add them to the training set. Since this merely represent represents the possible negative set, after training on these images, we use boot-strap method by randomly running the network on images not containing faces, and selecting the sub-images for which the network gives positive output, and add them to the negative training set.

From the given training data, the input layers get fixed with 400 nodes. The network will have a single output node, giving output greater than or less than 0.5 to represent the presence or non presence of a human face in the image fed into the input nodes as input. Since the positive examples are matter of interest, that is what we want to identify, and also those are symmetrically aligned, ie, centered around the point at the middle of the line joining the center point of upper lip and the horizontal line joining the eyes, and the distance of this horizontal line from the center of upper lip is always 12 pixel. We take three different sets of hidden units to capture different features of face like eyes, lips, nose etc. The first set is a 2 x 2 block which looks at the 10 x10 blocks of the input image, divided through horizontal and vertical lines through the image which is a 20 x 20 block. The lower two of these nodes will probably capture the two corners of the lips and each of the upper two should contain 0ne of the eyes as a part of it. We take second set of hidden nodes as a 4 x 4 block which will be looked at individual 5x5 blocks of the input nodes obtained through three horizontal equidistant lines through image. Finally we go for a third set of 6 hidden nodes which looks overlapping horizontal strips of the input image, each of size 5 x 20, to capture the " horizontal eyes strips" and the horizontal strip of lips. Hence, in all we get 26 hidden nodes divided into three sets. Therefore, the total number of nodes in the net is 427 and total number of edges is 1426 (weights)

The output of the Neural Network is refined through some post-processing steps removing false detection etc. This is also important since the trained network alone is not sufficient to identify faces. Other possibilities are to use more than one network and attribute to them.

Data are also very important issue in this problem. There is no standard known formula to represent human face as an object. So we have taken a data driven approach to solve face recognition problem. We can best approximate human face from a set of 'example data', which we know both the input and output. For each of these 'example data', information corresponding featured image as well as the class it belongs to should be available.

All images at gray scale level have also been considered. Although the color information of a human face can be treated as a facial feature and it does not carry lots of information for isolation of a face from its background. We are neglecting it for simplicity and speedup reason. We also restrict ourselves to a single layer of hidden units, since this is fund to be sufficient for identifying frontal upright faces. This restriction helps obtaining higher speed of learning, as the number of multiplication and addition needs depends directly on the number of hidden layers and hidden units.

The Learning Algorithm

Here simple Gradient Descent Back Algorithm is used as the learning algorithm to learn the parameters, ie., the weights. This is again a state-of the- art choice. This is very popular in machine learning due to its simplicity and fast convergence criterion. The algorithm runs down the error surface, which is assumed to be parabolic in nature, performing a gradient decent search in the weight space. Error surface is the surface that describes the error on each example as a function of the weights in the network. In each iteration, the algorithm moves the network in the direction of steepest decent on the error surface. The Back-Propagation Algorithm

```
For all epochs do
For all images in the images in the initial set, do
    1. Set the input vector
    2. Set target
    3. Feed forward through the network and calculate output.
    4. Calculate error and output ‘delta’
    5. Calculate hidden ‘delta’
    6. Adjust hidden weights
    7. Adjust input weights
End
End
```

Training and Training Data

The training data consist of a tagged set of images faced and non- faced both. Each one is of 20 x 20 pixel. We consider the images only at gray scale level. For every face image in training set, the eyes and the center of the upper lip are manually located. Face images are adequately rotated and normalized so that the eyes are placed in the horizontal line and the distance between the centers of the upper lip from this horizontal line is 12 pixel. The 20 x 20 image block is centered at the middle point on this vertical line, that is, 6 pixel above the center of the upper lips.

Before feeding the images to the input nodes of neural network, all the images are pre-processed to correct the lighting effects and then histogram equalized to increase the contrast of that image. The lighting correction removes the average intensities of the pixel of the image by estimating a liner ramp which is subtracted from the original image. Now this feature image pixel values are scaled down using the pre-computational table to reduce the input pixel values from [0,255] to [0,1] to create the input vector to be fed to the neural network as input. The same image processing operations are also done for the negative examples as well to maintain similarity for all input vectors regardless of their type and class. We use sigmoid function as the threshold function. This forces us to initialize the weights with small numbers near zero since the sigmoid curve is very much stable and gives constant gradient near zero.

The training process can be summarized as follows:

1. Select the positive and negative training data
2. Train network for these data.
3. Run the network on some images not containing human face.
4. Add those sub- images to negative training set which gives positive output.
5. Go back to step 1 for all epochs or till net converges.

Testing and Test Data

A training time embedded testing has to be performed from within the training module to estimate the performance of the network. This is done on an image list in general. This provides a stopping criterion for the training module.

Testing procedure:

1. Start with original image size.
2. Apply the network at every possible location to detect presence of a face.
3. Search for faces rotated in a small angle by rotating the input image itself.
4. Scale down the image by a constant factor and go back to step 2 if scaled image size is higher than networks input size; stop otherwise.
5. (Additional step) Apply neighborhood information around every detected face.
6. (Additional step) Combine results from different network.

Implementation of the Training Module

The input to this module is the training image data files. From the design of the system, we have concrete, fixed structure for neural network at the heart of algorithm. The learning rate and momentum are also kept to be fixed at 0.3 both, but can easily be modified to take user defined values. We also have two arrays, Weights and Rrev_Weights, to account for the edge weights and previous change in the edge weights in the last iteration. These are the parameters we want to learn. The out of the system is a file containing the final values from the Weight array of the network which defines the classifier.

We have 427 units and 1426 edges in the network, distributed in three layers. Each unit is of the form

```
sruct Unit_Type{
float act;                        /*Activation*/
float Bias;                       /* Bias of the Unit*/
int NoOfSources;                  /*Number of predecessor Units*/
UNIT_TYPE **sources;            /*predecessor units*/
float *weights;             /*weights from predecessor units*/
float *prev_weights;        /*previous change in waights*/
} Unit, pUnit;
```

The neural network is also specified by a structure as follows:

```
struct NEURAL_NET{
int input_n;            /* number of input units*/
int hidden_n                /*number of hidden outputs*/
int out_n;                  /*number of output units*/
pUnit  *Input_Units;
pUnit  *Hidden_Units;
pUnit  *Output Units  // all the units
float *hidden_delta;            /*storage for hidden unit error*/
float *output delta;                    /*storage for output unit error*/

float eta,momentam;             /*storage for target vector*/
foat target;                         /*actual output observed*/

float output;
} BPNN;
```

All the images are subjected to lighting correction and a linear scaling of the scaling of the pixel values from [0,255] to [0.0,-1.0]. While lighting correction, the objective is to remove the background effect from the original image, first followed by an enhancement is contrast to identify the object features. We fit the linear ramp for the 20 x 20 blocks which approximates the pixel values at every location to define the background. This operation is done within a mask not considering the boundary pixels to avoid boundary to avoid boundary artifacts which may be present in the image templates due to cropping.

The pixel values are needed to be selected to [0.0-1.0] range to feed into the neural network. The number of random location to produce for boot strapping from any image and a particular location is taken to be constant and equal to five.

Implementation of Testing Module

Input to the testing module is an image to be tested for the presence of a human face, and a trained network. Here the actual input is the trained weights rather than the complete network. In this stage we do not need the pre-weights field any more. We use JPEG images instead of PGM images for testing purpose because of is highly scalable property of JPEG images. Using a JPEG image library, scaling images at any scale is just a function call. We have used 0.9 as scaling factor for constructing the image pyramid.

With the choice of proper threshold, the false detection can be eliminated or reduced. We noticed that the overlapped detection usually points the same face. By overlapping two faces we mean that the areas of the faces are overlapped. The biggest size detection is kept in the face list and rest all is deleted. These two post processing tool bring improvement in final result.

Result obtained during training

The size of the training set varies with training progresses due to the addition of negative images from the boot-strap set. Hence, to gather results on training set, we must go for a relative measure of the correct detection rate, false detection rate and failure detection rate instead of their actual values. The table of training result is as follows.

Table I (Network 4.1)

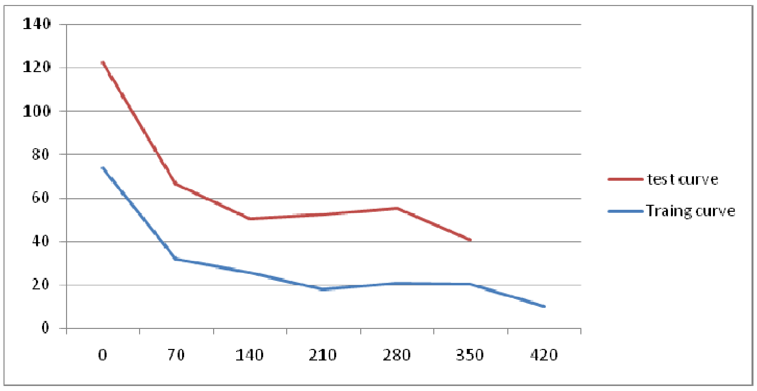
Seed=123456 Initial # of Images in Training set = 110 Boot-Strap Set=43 Test Set=49

Epoch	Total# In training set	% correct detection	%wrong detection	Total# in validation Set	% correct detection	% wrong detection
0	110	26.36	73.64	-	-	-
70	244	68.03	31.97	49	51.02	48.98
140	250	74.40	25.60	49	65.31	34.69
210	255	82.35	17.65	49	75.51	24.49
280	261	79.31	20.69	49	65.31	34.69
350	268	79.85	20.15	49	65.31	34.69
420	273	90.11	9.89	49	79.59	20.41

Table II (Network 4.2)

Seed= 210181 Initial # of Images in Training set= 110, Boot-Strap Set= 43, Test= 49

Epoch	Total # in training set	% correct detection	% wrong Detection	Total # in validation set	% correct detection	% failure detection
0	110	26.36	73.64	-	-	-
70	245	68.16	31.31	49	68.16	31.84
210	256	69.14	30.86	49	48.98	5102
350	269	82.16	17.84	49	67.35	32.65
490	282	82.98	17.84	49	61.22	38.78
630	290	85.17	14.83	49	69.39	30.61
770	302	85.76	14.24	49	65.31	34.69
910	312	91.35	8.65	49	77.55	22.45
980	319	95.30	4.70	49	83.67	16.33



Training Curve

Result Obtained Durning Test

These result are obtained by applying the trained network on JPEG test images of any size. Both the previosly trained netowks have been used on different test images which were not present in the training set.

The following results are obtained for a single netwotk system:

Table III: Thresh hold use = 8.

Network	Image	# location net applied	#of detection by the net	#detection after thresh holding	#detection on removal of overlap	Final location
Net 4.1	24.jpg (112 x 116)	33049	143	98	2	(35,76) (81,22)
Net 4.2	24.jpg (112 x 116)	3304 9	189	90	2	(36,76) (81,21)
Net 4.1	1.jpg (163 x 131)	63786	108	41	1	(54,66)
Net 4.2	1.jpg (163 x 131)	63786	118	29	1	(54,64)
Net 4.1	30.jpg (100 x100)	22746	125	77	3	(56,20) (37,74) (54,46)
Net 4.2	30.jpg (100 x100)	22746	87	48	1	(50,47)

Following are the result obtained for AND-ing multiple networks:

Table IV

S.No.	#Nets	Nets Used	#Images	#Faces	# Correct Detection	# False Detection	Failure Detection
1.	2	Object, Miscellaneous	22	68	54	39	14
2.	3	Objects, Sky, Miscellaneous	22	68	54	30	14
3.	4	Objects, Miscellaneous, Trees, Houses	22	68	58	8	10
4.	5	Objects, Miscellaneous, Trees, Houses, 4.3	22	68	54	4	14

Though it may seem from the tabulated result that the error percentage is very quite high, actually it is very near to zero. This is because, while calculating the percentage correct or false detection, we must take into account the negative sub-images as well. Since every image has almost 100% negative sub images and almost all of them are detected correctly, that is, as negative image, even if some of the faces are wrongly detected and other faces could not be detected, the performance of the system is close to accurate.

Quantitatively, we have total locations = 12761023
Face locations = 68, non face locations = 12760955
Faces correctly detected = 54, missed to detect = 14
Non-faces wrongly detected = false detection = 4
Non-faces correctly detected = 12760951
Hence total correct detection = 54+12760971 = 12761005 and
Total wrong detection = 14=4=18
This gives the clear idea about the system performance.

Conclusion

We address the problem of Human Face Detection here in this paper. We provide a thorough understanding of the problem followed by a detail design and implementation, which is further experimentally validated. As we can see from the result this face detection system is not robust on its own; that is, the learning procedure does not ensure us 100 % correct detection. Hence, we applied some post processing steps as showed in previous sections.

An important point to observe here is that the accuracy of the system heavily depends on the selection of the thresh holding parameter. Better result can be obtained using adaptive threshold. Generally we can expect more detection for a large face since it can be identified in many subsequent scales which forms the image pyramid. Again large faces are expected to occur more commonly in a large image. Hence we can apply adaptive thresh holding, which apply small value for the threshold for small images, medium value for medium sized image and relatively large value for a large image and so on. We also find here that choosing a very large threshold to remove false detection may result in failure in detection. Instead, this system use multiple network and arbitrate between them to produce actual output. This property can be efficiently used to remove false detection simple by AND-ing or Voting among different network, or by taking a weighted multiplexer. Color histogram based approach, edge detection based approach etc are also potential techniques for performance optimization, which if applied carefully, can improve speed of the system as well.

Reference

[1] A. J Colmenarez, B. Fery and T.S. huang : “Detection and tracking of faces and facial features:”: proceeding International Conference on Image processing,1999

[2] A. Schneiderman and T.Khande: “ A statistical Model for 3d object detection applied to faces and cars”, Proceeding computer vision and Pattern recognition, 2000, IEEE.

[3] David A. Forsyth, University of California, Berkeley and Jean Ponce, University of Illinois, Urbana Champaign” Computer Vision a Modern Approach “.

[4] D. E. Benn, and M.S. Nixon and J.N, Carter “ Extending concentricity analysis by deformable templates for improved eye extraction” Proceeding Second International Conference on Audio and Video –Based Biometric Person Authentication (AVBPA)1999

[5] Eric Hjelmas and Boon Kee Low,” Face Detection: A Survey”, Science Direct-Computer Vision and Image Understanding.

[6] Gaungzheng Yang and Thomas S Huang, “Human Face Detection in a Complex Background.

[7] Henry A. Rowley, Sumeet Baluja, Takeo Kande, “ Neural Network Based Face Detection, IEEE Transaction on Pattern Recognition and Machine Lerning.

[8] Lucas Finschi, ” An implementation of the Levenberg-Marquardt Algorithm”, Institute of Operation Research, Zurich, 1996.

[9] M. Abdel-Mottaleb, and A Elgammal, “Face Detection in Complex Environment”, Proceeding International Conference in Image Processing, 1999.

[10] Ming Husan Yang, David Kriegman and Narendra Ahuja:”Detecting Faces in Images : A survey, IEEE Transactions’ on Pattern analysis and Machine Intelligence

[11] M.J. Jones and J. Rehg,” Statistical color Models With Application to Skin Detection” Computer Vision and Pattern Recognition,1999,IEEE.

[12] Rapheal Feraud, Oliver Bernire, Jean E. Viallet and Michel Collobret,”A fast and accurate Face Fetector Based on Neural Network”, IEEE Transaction Pattern Recognition and Machine Learning.