Survey of Image Based Eye Closure Detection Methods

Minh-Tri Ho
Advanced Program in Computer Science
Faculty of Information Technology
University of Science, VNU-HCM
hmtri@apcs.vn

Trong-Tung Nguyen
Advanced Program in Computer Science
Faculty of Information Technology
University of Science, VNU-HCM
ntrtung@apcs.vn

Hieu Dao
Advanced Program in Computer Science
Faculty of Information Technology
University of Science, VNU-HCM
dhieu@apcs.vn

***Abstract*—Being able to detect eye closure can have several practical applications e.g. a driver drowsiness detection system, a student attention monitoring system, etc. We compare between several methods to determine whether a person is closing their eyes or not based on a picture of their face, which can come in various face angles and lighting conditions. We find that the best method is either an ensemble of linear classifiers that is fast and has 86.94% accuracy, or a feature extraction and classification method using the MobileNet convolutional neural network that is five times slower but with 93.57% accuracy. Which method is more appropriate depends on the specific situation. Our work can be the basis for more sophisticated solutions that fit more closely with real-world usage e.g. multiple faces in one image, video data rather than image data,**

1. Introduction

As one of the most significant features of the human face, the eyes play an important role in face detection and face recognition [1, 2, 3, 4]. Through the eyes, we can obtain information about a person's attentions and emotional states [1, 3, 5]. Therefore, much effort has been made to utilize the eyes in facial expression analysis and human-computer interaction technologies [1, 5]. It can be said that proper analysis of the eyes will be of great help in a variety of different situations.

In this paper, we explore a range of different methods to determine whether a person's eyes in an image are open or closed. An eye detection algorithm can use our methods to achieve possibly better accuracy by working separately on open and closed eyes. This algorithm can then be used, for example, as part of a gaze tracking functionality of an attentive user interface. Another possible application of our methods would be to calculate the blinking rate of a person, which can be used in a driver drowsiness detection system or a student attention monitoring system.

Our methods work on face images and require the faces to be located beforehand. Out of the explored methods, there are two noteworthy ones. The first method uses a linear classifier directly on the input images. This method is fast, lightweight, and has 86.94% accuracy. However, because it is basically comparing the input image to a template image, this method will only work well if the input is a frontal or near-frontal face image. The second method uses the existing MobileNet convolution neural network to extract features from the images as a form of transfer learning [6]. These extracted features will be the input of a linear classifier. This method is five times slower and uses more memory than the first method but achieves 93.57% accuracy. Also, it generalizes better to cases where the inputted face deviates substantially from the template face. Depending on the specific circumstance, the first or the second method will be more appropriate.

The methods we explore are organized in Fig 1. The rest of this paper is organized as follows: section 2 introduces the background and related works, section 3 talks about the dataset and our hardware resources, section 4 describes the Viola-Jones based method [7], section 5 discusses the traditional computer vision methods, section 6 gives the details of the machine learning methods, and section 7 concludes the paper.

2. Background and related works

The existing image based eye closure detection methods can be divided into two main groups: feature-based methods and appearance-based methods.

The feature-based methods aim to solve the problem by relying on distinctive characteristics of the eye. Some methods might use geometrical characteristics such as the circular shape of the iris [8], the elliptical shape of the eyelid [9], or the horizontal symmetry of an open eye [10]. Other methods make use of the intensity and saturation distributions, for example, utilizing the contrast between the darkness of the iris and the whiteness of the sclera [8, 11], or the distinctive skin color of the eyelid [12]. Yet other methods experiment with building a template for the eye, either by using classifiers to learn eye contour positions [13], using existing methods to locate facial landmarks specific to the eyes [14], or some other method [15].

The appearance based methods differ from the feature based methods in that they attempt to extract and use mid- and low-level features of the eye region. That is, rather than describing specific, high-level details such as the iris or

the sclera, the mid- and low-level features would describe basic shape, texture, or color information. Some examples of mid- and low-level features extracted are Local Binary Patterns (LBP) [16, 17], Histogram of Oriented Gradients (HOG) [16], Gabor wavelets [16, 18, 19], or simply the pixel values of the image [16, 20]. These features are then given to a machine learning model such as Adaboost [16, 20, 21], Support Vector Machine (SVM) [16, 17, 18], or neural network [16, 19] to automatically find the underlying relation between the mid- and low-level features and the eye closure status.

3. Methods

3.1. Overview

The methods we explore are organized in Fig. 1. Beside a simple method using the existing Viola-Jones object detection framework, our methods can be divided into manual methods and machine learning methods. With the manual methods, we experiment with several ways a person might detect eye closure and attempt to translate these into concrete algorithms. We also investigate several machine learning methods, from the more simple and traditional ones like linear classifiers to the more complicated and modern ones like feature extraction with convolutional neural networks.

3.2. Viola-Jones based method

Our first proposed method is to use the Viola-Jones object detection framework [7]. This framework is often used for face detection but it can also be used for eye detection. We observe that when using the Viola-Jones function of the OpenCV library, the eyes can only be reliably detected when they are open. Therefore, we propose classifying the eyes as open when they can be detected by Viola-Jones and closed otherwise. As can be seen in 4.2, this method has surprisingly high accuracy. Therefore, we can think of this method as a baseline for the other methods in this paper.

3.3. Traditional computer vision methods

As mentioned above, the Viola-Jones framework is not specifically designed for classification [7]. Therefore, we believe that we can improve on the result of the method using Viola-Jones by using methods designed for classification. We begin by exploring the techniques of traditional computer vision because these techniques are simple and can help us express how a person might intuitively solve the problem of eye closure detection.

3.3.1. Preprocessing. To make it easier to work with the data, we implement a few preprocessing steps. Firstly, we crop the eye area from the face image, the specific implementation of which is in 4.3.1. Next, we turn the image into greyscale. Finally, we threshold the image so that each pixel is now only one of two values – 0 or 255. The first threshold type used is binary threshold, but we find that the result of this threshold is unsatisfactory. Hence, we apply another threshold – adaptive threshold, and get a better result (Fig. 2).

3.3.2. Sclera detection. We observe that one of the distinguishing features of the eyes is the white region surrounding the cornea, called the sclera. This region is only visible when the eyes are opened and disappears when the eyes are closed. Furthermore, it can be observed that the area of the sclera is roughly the same from person to person. We attempt to capture these facts in a classifying algorithm: If there is a white region within a specific range of area in the image then the eyes are opened; otherwise, the eyes are closed.

However, even after choosing the best possible range, this method still gives rather low accuracy, as can be seen in 4.3.2. One reason for this is because the sclera often spills into the surrounding regions, as can be seen in Fig. 3. When this happens, we cannot calculate the area of only the sclera and our algorithm will not work. This may be improved by employing edge detection or image dilation to isolate the sclera. We believe another reason why this algorithm is unlikely to achieve good results is because it is too crude: it only cares that the image contains a white region within a specific area range, not where the region is. Several things that might improve this would be to make sure that the found region is in an appropriate position in the image or to check if there are two regions with similar areas and symmetric with regard to some line (the two eyes symmetric with regard to the nose).

3.3.3. Black pixel percentage. We observe that when the eyes are open, the eye region is generally more cluttered than when the eyes are closed. This can be seen most clearly in thresholded images (Fig. 4). One feature of this clutterness is that there are more black pixels in the eye region. If the other regions of the face have - on average - the same amount of black pixels between open and closed eyes then we hypothesize that we can use the percentage of black pixels as a distinguishing feature. Our classifying algorithm is: If the percentage of black pixels in the image is above a certain threshold then the eyes are open; otherwise, the eyes are closed.

However, as can be seen in 4.3.3, this algorithm's result is still not satisfactory. One reason is because it is only using a global measure of the image without taking into account the local features. So if, for example, the person in the image is wearing glasses (as can be seen in Fig. 5) then the percentage of black pixels will increase substantially. So even if the person's eyes are closed, they might still be misclassified as open.

3.3.4. Pupil detection. Another easily recognizable feature of the eye is the pupil. We propose to locate the pupil based on its circular shape by using the circle Hough Transform [22]. For this method, our classifying algorithm is: If a circle is detected then the eyes are open; otherwise, the eyes are

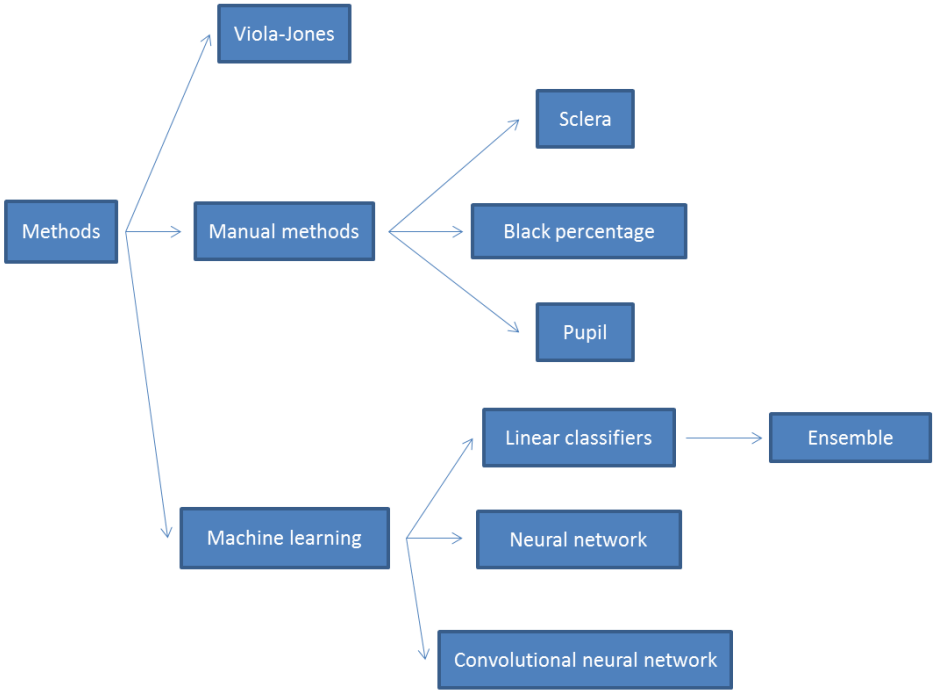
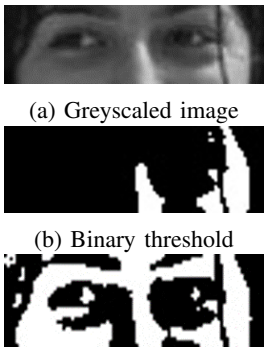


Figure 1: The explored methods



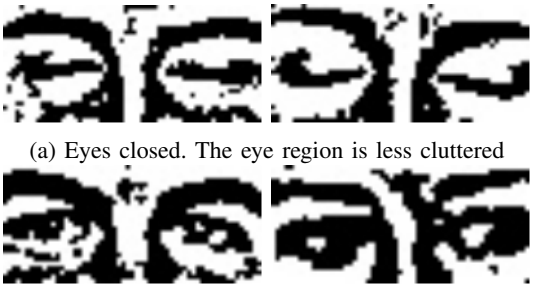
(a) Greyscaled image

(b) Binary threshold

(c) Adaptive threshold

Figure 2: Comparison of the two types of threshold

Figure 3: The sclera spills into the surrounding region



(a) Eyes closed. The eye region is less cluttered

(b) Eyes open. The eye region is more cluttered

Figure 4: Comparison of the eye region between open and closed eyes

Figure 5: Examples of people wearing glasses in the dataset

closed. It is our expectation that the detected circles will be the pupils.

However, as can be seen in 4.3.4, the result of this method is still not adequate. One reason is that in our dataset, and in realistic conditions, it is quite rare for the

Figure 6: Examples where the pupil is not circular

pupil to be perfectly circular. But instead, the pupil might take on an oval shape or in some cases might just be a stripe (Fig. 6). Clearly, a circle detection method cannot work in cases like these.

3.3.5. Conclusion. One conclusion that we drew from the three methods is that it is difficult to achieve good results using the basic techniques of traditional computer vision. This is because even though it is easy for humans to distinguish between open and closed eyes, it is difficult to translate this into specific algorithmic form.

Machine learning describes a group of methods that has seen widespread use in computer vision in recent years. These methods has the benefit that we do not have to manually find the important features to solve the problem as in sclera detection or pupil detection above but rather, the machine will learn the features on its own. We will now explore several methods of machine learning.

3.4. Machine learning methods

3.4.1. Logistic regression.

Overview about this learning model. The general mechanism of this learning model is taking the dot product between optimized parameter vector and each of the image in the dataset as input for loss function of the model. This loss function represents the similarity between our parameter vector and the original dataset. The target of this learning model is to find the optimized vector that can cause loss function's value to be as small as possible. The mathematical notations used in this model is explained here. Let's denote $\theta = [w_1, ..., w_n]$ to be column optimized parameter vector. The i^{th} image also known as i^{th} training example of size $n1{*}n2{*}3$ (with $n1{*}n2$ is the size of original image) , which has been unrolled into a column-vector of $n1{*}n2{*}3$ elements is denoted as x^i, a column-vector denoted as y_i represents the actual labels for the i^{th} training image (1 for open eyes and 0 for closed eyes). Denoting the dot product between optimized theta and training example x^i as $\theta^T x^i$. The result of this dot product is then put into the Sigmoid function to return value from range 0 to 1. The Sigmoid function that takes dot product between optimized parameter and training image i^{th} is denoted as $h_\theta(x^{(i)})$. The lost function that this learning model tries to optimize has the following form

$$J(\theta) = \frac{-1}{m}\sum_{i=1}^{m}[y^i \log(h_\theta(x^{(i)})) + (1-y^{(i)})\log(1-h_\theta(x^{(i)}))] \quad (1)$$

$$\theta = \arg\min_\theta(J(\theta)) \quad (2)$$

Training phase with Gradient Descent vs Momentum Gradient Descent. Dataset is divided into 3 set: training set , cross-validation set and test set. Learning model starts to learn from training set to find optimized parameter. The optimized parameter is then validated again by the cross-validation set to tune the best hyper-parameter for the model. In this model, hyper-parameter we choose is γ which will be described soon. One of the popular methods for finding optimized parameter for 1 is Gradient Descent. However, the dimension of the training dataset is too large that it might take a lot of time to converge to the global minimum. Therefore, an improved version of Gradient Descent known as Momentum Gradient Descent (MGD) is applied with expectation of faster convergence. The idea behind MGD is rather comprehensive. The progress of updating parameter vector to find the global minimum can be considered as a ball trying to reach the lowest point. Without momentum, the speed of convergence of the ball depends only on the slope of hill at current position. With momentum, the speed of convergence depends both on the slope of hill and current velocity. If current velocity is large enough, the ball goes downhill faster or even it can jump over the local minimum.
The progress of training with MGD comprises of many parameters that have to be randomly initialized first:α known as learning rate or step size,k known as number of iterations that MGD has to run through, optimized vector parameter θ, list of γ known as scaled factor for previous velocity and hyper-parameter for this model, $v_0 = 0$ known as initial velocity. Then the progress of training is repeated as follows
Training parameter step:
for gamma **in** gammalist:
+Repeat

$$v_t := \gamma v_{t-1} + \alpha \nabla_\theta J(\theta) \quad (3)$$

$$\theta := \theta - v_t \quad (4)$$

Calculating error using gamma value and pre-trained parameter and appending onto cross-validation error list

+Pick gamma value which has the **smallest** error on cross-validation error list with $\nabla_\theta J(\theta)$ as gradient vector of loss function. This process will be terminated when the loss function has approached to a specific small value or after k iterations.

Predicting Phase. The last phase of this learning model is using pre-trained vector θ to predict unseen images. The progress of prediction comprising of many steps is described in Figure 7 : face detection using Viola Jones, unrolling into a column-vector of $n1 * n2 * 3$ elements, taking the dot product between the pre-trained vector and the image's column-vector, the score of dot product is pushed into a Sigmoid function to return values from range 0 to 1,

setting threshold $\in = 0.5$, the outcomes of Sigmoid function will be compared with $\in$ value to predict the state of eyes.

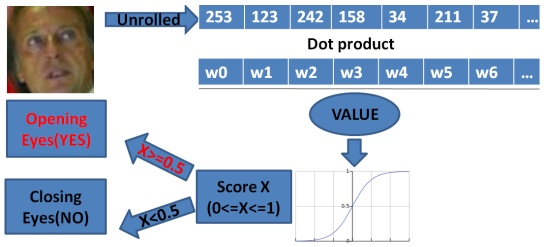


Figure 7: Predicting Progress

3.4.2. Softmax classifier. Alongside the logistic regression classifier, we also experiment with a second linear classifier called the Softmax classifier [23]. One reason of the experimentation is to see if there are any differences between the linear classifiers. Another reason is because the Softmax classifier is more directly related to the neural network, which we will discuss in 3.3.4.

The basic difference between the two linear classifiers is that whereas the logistic regression classifier takes the dot product of the N-dimensional input, where N is the dimensionality of the unrolled input image, with an N-dimensional weight vector and uses this value to classify, the Softmax classifier takes the dot product of the N-dimensional input with two N-dimensional weight vectors (Fig. 8). The two resulting values will be the scores of the two classes and the class with the higher score is chosen. The name Softmax comes from the loss function, which interprets the class scores as unnormalized log probability and aims to maximize the probability of the correct class (Fig. 9) [23]. Therefore, one advantage of this classifier is that we can output the classification and also our confidence in the classification, which can be useful in many cases.

The Softmax classifier achieves comparable accuracy to the logistic regression classifier and takes a similar amount of time to train. The two resulting weight vectors can be visualized as two template images (Fig. 10). Basically our classifier is comparing the input image to the two template images using dot product distance and classifying the input image the same class as the more similar template image.

3.4.3. Linear classifiers ensemble. We experiment with a method commonly used to slightly boost classifiers accuracy: combining them together [24]. We decide to combine the logistic regression classifier and the Softmax classifier to see if there is any accuracy improvement. There are many ways to combine classifiers into an ensemble but we choose to use another Softmax classifier to classify the output of the two linear classifiers. More specifically, for each image, we use the logistic regression classifier to calculate the image score – the output of the Sigmoid function – and we use the Softmax classifier to calculate the images two class scores. We combine these three values into a vector and this vector will now represent the original image. With each image as a three-dimensional vector, we use a Softmax classifier to classify the images.

The ensemble does increase on the individual classifiers accuracy as expected. However, the accuracy increase might be a bit too small, not worth the extra time and memory training and testing two classifiers. We believe one reason why this increase is small is that the ensemble classifier is supposed to combine the strengths of the individual classifiers [24]. But because our two linear classifiers are too alike and therefore have mostly overlapping strengths, the ensemble classifier cannot be much useful here.

3.4.4. Neural network. A neural network is a generalization of the linear classifier [25]. Whereas a Softmax linear classifier calculates the scores for the different categories using s = W*x, a 2-layer neural network would instead calculate s = W2 * max(0, W1*x). Here the function max(0, -) is a non-linearity and is one of many choices. Similarly, a 3-layer neural network would calculate s = W3 * max(0, W2 * max(0, W1*x)) and so on for deeper networks. A linear classifier can be thought of as a 1-layer neural network.

From the neuron view, a neural network consists of one or more hidden layers between the input and output layers [25]. Each neuron in the hidden layer is connected to each neuron in the previous and next layer.

What differentiates a neural network from a linear classifier is that a linear classifier is only good at classifying linearly separable data whereas a neural network, because of its non-linearity, can classify non-linearly separable data. In fact, it is proven that with enough neurons in the hidden layer, a 2-layer neural network can approximate any arbitrary function [26].

However, when we experiment with a 2-layer neural network and a 3-layer neural network, we find that the accuracy increases very slightly. And because of the added computation, training and testing the neural networks take much more time compared to the linear classifiers. The reason for the low increase in accuracy is partially specific to the dataset we use so we explore this further in 4.4.4.

3.4.5. Feature extraction using convolutional neural network. In the previous subsection, when we say neural network we actually mean fully connected neural network, in which every neuron in a layer is connected to every neuron in the previous layer. There is another kind of neural network called a convolutional neural network (CNN) that is very popular in recent years. It has been shown to help achieve very good results on various image recognition tasks [27]. We propose a way to apply CNN to our specific problem.

One way to apply CNN would be to train a CNN architecture completely on our dataset. However, this would likely require a lot of time and memory. Another way that is also popular uses transfer learning, where we take a CNN trained on a different dataset and take out its last few layers [27]. The remaining network would take in an image and outputs a feature vector which usually has lower dimensionality than the image. This technique is called

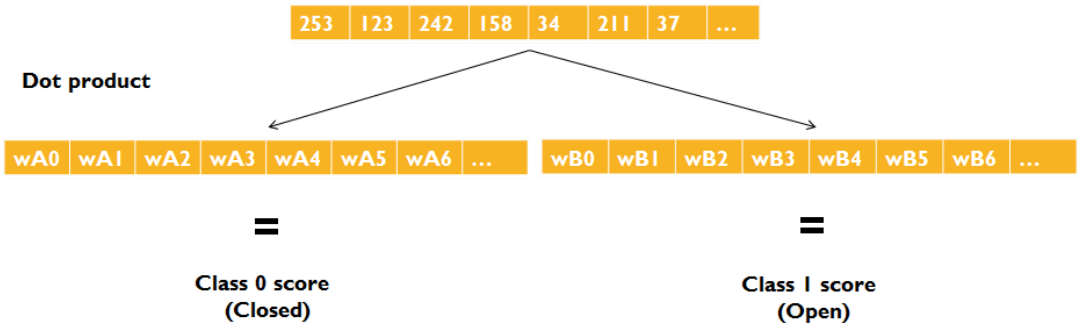


Figure 8: Two class scores are calculated from the unrolled image vector

$$L_i = -\log\left(\frac{e^{f_{y_i}}}{\sum_j e^{f_j}}\right)$$

Figure 9: The Softmax loss of one sample

Figure 10: Visualization of the two weight vectors corresponding to open and closed eyes

feature extraction and it has been shown that the feature vector contains meaningful content [27].

For our problem, we use MobileNet as the CNN for feature extraction [6]. We choose this CNN because it has good performance but is also lightweight compared to other CNNs, appropriate for the low memory capacity of our machine. One effect of this is that our resulting method will be fast and lightweight compared to if we used other CNNs. We take out the fully connected layer at the end of MobileNet and keep the convolutional layers. We then use this network to extract feature vectors from the images and a Softmax classifier is used to classify the feature vectors.

This method achieves the highest accuracy among our methods. This confirms that the extracted features are semantically meaningful and the transfer learning technique is valid. However, even with the MobileNet CNN, the feature extraction phase still takes considerable time, making speed a significant limitation.

4. Experimental results

4.1. Dataset and Hardware

We use the Closed Eyes in the Wild (CEW) dataset of Nanjing University of Aeronautics and Astronautics, China [16]. This dataset includes 2423 images of frontal faces, among which are 1192 images with closed eyes and 1231 images with open eyes a slightly uneven distribution. The data exhibits various environment changes including lighting condition, blur, occlusion, and face orientation.

All of our methods are run on an Intel Core i5-8250U CPU at 1.60 GHz and with 4.00 GB of RAM. This hardware is for general purposes and is not specialized for intensive computing like machine learning. Therefore, methods that run efficiently on our hardware are likely to run efficiently on other commodity hardware.

4.2. Viola-Jones based method

By classifying the eyes as open when they can be detected by Viola-Jones and closed otherwise, we can achieve an accuracy of 83.16% on the CEW dataset without any specific finetuning, which is impressive considering that Viola-Jones is designed for detection, not classification [7]. The method is also rather fast, taking 10 seconds to classify the 2423 images in the dataset.

4.3. Traditional computer vision methods

4.3.1. Preprocessing. For the CEW dataset, we find that we can reliably crop the eye area from the image using horizontal slice with the upper and lower end at 25 / 32 and 15 / 32 of the height of the image, respectively. There values are fixed and have been manually determined by us to be not too wide but also not too narrow. We use the adaptiveThreshold function from the OpenCV library and choose Gaussian as the threshold mode.

Figure 11: Output of HoughCircles() as we vary param2

4.3.2. Objective function. Each of the traditional computer vision methods has a number of parameters and we need to choose the parameters that maximize a certain objective function. Because of the uneven nature of the dataset and also the low accuracy of the methods, this objective function needs to be chosen with care. We have experimented with letting the objective function be accuracy or instead, the average of precision and recall. However, both of these result in the algorithms classifying open eyes disproportionately more than closed eyes or the other way around. We finally decide to let the objective function be the minimum of precision and recall, which gives the most appropriate results.

4.3.3. Sclera detection. The areas of the white regions in an image can be calculated using breadth first search. To find the appropriate area range for the sclera, we iterate over all the possible combinations of the lower and upper end of the range, with a step size of 1% of the image's area for each of the end. The combination of the ends that leads to the best result is chosen to be the range. The returned range is 1% - 2% and the accuracy is 51.80%, not much better than randomly choosing between open and closed eyes, as explained in 3.2.2.

4.3.4. Black pixel percentage. To find the appropriate black pixel percentage threshold, we iterate over all possible values with a step size of 1% of the images area and choose the one that gives the best result. The chosen threshold is 59%, which gives 58.27% accuracy. Although the accuracy is approximately equal to the percentage of open eyes images in the dataset –leading to suspicions that this accuracy might be achieved by classifying most images as open eyes, the number of images labeled as open and closed eyes are actually 1308 and 1115, respectively. We can see that the algorithm is labeling the images with an appropriate ratio of open and closed eyes and that it performs better than random.

4.3.5. Pupil detection. We use the HoughCircles() function in the OpenCV library as the implementation. Of the parameters of the function, a notable one is param2. If we set this parameter too low then many false circles will be detected, creating noise; if we set it too high then the pupil might not be detected (Fig. 11). We empirically find 16 to be a reasonably good value for this parameter.

This method achieves 58.19% accuracy. As in the black pixel percentage method, our method does not classify all images as open eyes but rather assigns an appropriate ratio between open and closed eyes, indicating that this method is in the right direction and is better than random.

4.3.6. Overfitting. For each of the previous methods, we choose the parameters that maximize the objective function on the entire dataset and then evaluate the method on the entire dataset again. This might lead to the problem of overfitting. However, we believe that this is highly unlikely because of the small number of parameters in each method two, one, and three, respectively. To make sure, we test the methods with the chosen parameters on random samples of the dataset of size 1000. We find that the results are consistent and do not vary much between each test and between the samples and the overall dataset. Therefore, it can be said that our methods are not overfitting.

4.4. Machine learning methods

4.4.1. Logistic Regression Model. The training phase is conducted on our own datasets which is divided into training,cross validation and test set with ratio 60/20/20 and costs about 10 minutes for training to find optimize parameter with k=2200 iterations .

It is shown that the hyper-parameter $\gamma = 0.9$ is good for running MGD [28] . But in this method, we randomly choose many other different value of γ to see whether there is any better γ which perform well on our cross-validation set. The table of measurements of randomly picking 5 different gamma values is given in the table.

Table Of Measurements	
Gamma Value	Cross-Validation Error
0.00541174	0.246
0.66344759	0.15
0.54015045	0.156
0.82001168	0.15
0.93190692	0.164

As you can see in the table With $\gamma = 0.0054015045$, $\gamma = 0.82001168$ and $\gamma = 0.66344759$, model performs well on the unseen data with accuracy approximately about 0.85, all of three gamma above would be nice for using to predict on a new input image.

4.4.2. Softmax classifier. Like the logistic classifier, we use momentum to augment the stochastic gradient descent and use a 1400/500/523 train/val/test split. It takes 51 seconds for the Softmax classifier to train 250 iterations and the classifier achieves 85.96% test accuracy, comparable to the logistic regression classifier.

4.4.3. Linear classifiers ensemble. After collecting the outputs of the logistic regression classifier and the Softmax classifier into a three-dimensional vector, we train a second Softmax classifier on the resulting vector. Because the dimensionality of the new input is much less than that of the original image, 3 compared to 30000, the ensemble Softmax classifier trains much faster, taking 31 seconds to train for 10000 iterations. The classification accuracy is 86.94%, slightly better than the individual logistic regression classifier and Softmax classifier.

4.4.4. Neural network. We experiment with 2-layer neural networks with up to 200 neurons in the hidden layer and 3-layer neural networks with up to 200 x 1000 neurons in the two hidden-layers. Training for 250 iterations, the neural networks take up to three minutes, about three times as long as the linear classifiers training time. The highest achieved accuracy is 86.55%, only slightly higher than the accuracy of the linear classifiers.

We believe the reason for this low increase in accuracy is that there are too few samples in our dataset compared to the dimensionality of each sample. We can imagine each sample as a vector and can see that the dataset of 2423 vectors is too sparse for the 30000-dimensional vector space. Intuitively, it is likely that the dataset is close to being linearly separable. Another way to verify this is in finding out that our Softmax linear classifier, a very simple model, is able to achieve close to 100% accuracy on the training set. Because the dataset is closed to being linearly separable, when training on the training set, both the linear classifier and the neural network will end up linearly separating the training samples and the non-linearity of the neural network will give it no advantage.

4.4.5. Feature extraction using a convolutional neural network. We resize the input images to a size of 224 x 224 x 3 because this is the expected input size of MobileNet [6]. With the MobileNet CNN, we take out the fully connected layer at the end and keep the convolutional layers. The reason we do this is because it is believed that whereas the fully connected layers are specific to the dataset the original ImageNet dataset MobileNet is trained on, the convolutional layers on the other hand can extract meaningful and generalizable semantic content from the image [27]. After taking out the last layer, the remaining convolutional layers of MobileNet are used to extract 1280-dimensional feature vectors from images of size 224 x 224 x 3. Finally, a Softmax classifier is used to classify these feature vectors.

This method achieves 93.57% accuracy, the highest among our methods. However, even with the MobileNet CNN, the feature extraction phase still takes considerable time, requiring five minutes to extract features for the entire dataset. Even though for training we technically only need to extract once and can store the feature vectors to reuse later making the extracting time somewhat negligible for training, at test time we still have to extract features for each image. Therefore, the speed of this method is still a significant limitation.

4.4.6. Summary. Here is a table listing the accuracy and training time of the methods.

Model	Accuracy	Training time
Viola-Jones	83.16%	NaN
Sclera	51.80%	NaN
Black ratio	58.27%	NaN
Pupil	58.19%	NaN
Logistic	85.01%	1m
Softmax	85.96%	51s
Ensemble	86.94%	31s (+ Logistic + Softmax)
ANN	86.55%	3m
CNN	93.57%	5m

5. Conclusion

We have explored several methods for detecting eye closure in an image. Out of these, the most notable are the fast linear classifiers with decent accuracy and the slower convolutional neural networks with higher accuracy. Depending on the use case, there will be a more appropriate method. Some potential directions to continue from our work include applying data augmentation or dimensionality reduction to better make use of the neural networks, or experimenting with different CNN architectures to analyze their trade-offs.

References

[1] D. W. Hansen and Q. Ji. "In the Eye of the Beholder: A Survey of Models for Eyes and Gaze". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 32.3 (Mar. 2010), pp. 478–500. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2009.30.

[2] Kun Peng, Liming Chen, and Kukharev Georgy. "A Robust Algorithm for Eye Detection on Gray Intensity Face without Spectacles". In: *Journal of Computer Science and Technology - JCST* 5 (Oct. 2005).

[3] Zhiwei Zhu and Qiang Ji. "Robust real-time eye detection and tracking under variable lighting conditions and various face orientations". In: *Computer Vision and Image Understanding* 98 (Apr. 2005), pp. 124–154. DOI: 10.1016/j.cviu.2004.07.012.

[4] Peng Wang ; M.B. Green ; Qiang Ji ; J. Wayman. "Automatic Eye Detection and Its Validation". In: *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Workshops*. Sept. 2005, pp. 164–164. DOI: 10.1109/CVPR.2005.570.

[5] Tanmay Rajpathak, Ratnesh Kumar, and Eric Schwartz. "Eye Detection Using Morphological and Color Image Processing". In: (Jan. 2009).

[6] Andrew G. Howard et al. "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications". In: *CoRR* abs/1704.04861 (2017). arXiv: 1704.04861. URL: http://arxiv.org/abs/1704.04861.

[7] Paul Viola and Michael Jones. "Robust Real-Time Object Detection". In: *International Journal of Computer Vision - IJCV*. Vol. 57. Jan. 2001.

[8] Hong Liu, Yuwen Wu, and Hongbin Zha. "Eye state detection from color facial image sequence". In: *Proceedings of SPIE - The International Society for Optical Engineering* 4875 (July 2002). DOI: 10.1117/12.477054.

[9] L. Yunqi et al. "Recognition of Eye States in Real Time Video". In: *2009 International Conference on Computer Engineering and Technology*. Vol. 1. Jan. 2009, pp. 554–559. DOI: 10.1109/ICCET.2009.105.

[10] T. Danisman et al. "Drowsy driver detection system using eye blink patterns". In: *2010 International Conference on Machine and Web Intelligence*. Oct. 2010, pp. 230–233. DOI: 10.1109/ICMWI.2010.5648121.

[11] M. Eriksson and N. P. Papanikotopoulos. "Eye-tracking for detection of driver fatigue". In: *Proceedings of Conference on Intelligent Transportation Systems*. Nov. 1997, pp. 314–319. DOI: 10.1109/ITSC.1997.660494.

[12] Axel Panning, Ayoub Al-Hamadi, and Bernd Michaelis. "A color based approach for eye blink detection in image sequences". In: Nov. 2011, pp. 40–45. DOI: 10.1109/ICSIPA.2011.6144085.

[13] Liting Wang et al. "Eye Blink Detection Based on Eye Contour Extraction". In: vol. 7245. Feb. 2009, p. 72450. DOI: 10.1117/12.804916.

[14] Tereza Soukupova. "Eye Blink Detection Using Facial Landmarks". In: 2016.

[15] Ying-Li Tian, Takeo Kanade, and Jeffrey Cohn. "Dual-state Parametric Eye Tracking". In: *Proceedings of the 4th IEEE International Conference on Automatic Face and Gesture Recognition (FG'00)*. Mar. 2000, pp. 110–115.

[16] Fengyi Song et al. "Eyes closeness detection from still images with multi-scale histograms of principal oriented gradients". In: *Pattern Recognition* 47 (Sept. 2014), pp. 2825–2838. DOI: 10.1016/j.patcog.2014.03.024.

[17] Rui Sun and Zheng Ma. "Robust and Efficient Eye Location and Its State Detection". In: *Advances in Computation and Intelligence*. Ed. by Zhihua Cai et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 318–326. ISBN: 978-3-642-04843-2.

[18] Erkang Cheng et al. "Eye state detection in facial image based on linear prediction error of wavelet coefficients". In: *2008 IEEE International Conference on Robotics and Biomimetics*. Feb. 2009, pp. 1388–1392. DOI: 10.1109/ROBIO.2009.4913203.

[19] Ying-Li Tian, Takeo Kanade, and Jeffrey Cohn. "Eye-State Action Unit Detection by Gabor Wavelets". In: *Proceedings of the Third International Conference on Advances in Multimodal Interfaces (ICMI 2000)*. Oct. 2000.

[20] Mahdi Rezaei and Reinhard Klette. "3D Cascade of Classifiers for Open and Closed Eye Detection in Driver Distraction Monitoring". In: vol. 6855. Aug. 2011, pp. 171–179. DOI: 10.1007/978-3-642-23678-5_19.

[21] H. Wang, L. B. Zhou, and Y. Ying. "A novel approach for real time eye state detection in fatigue awareness system". In: *2010 IEEE Conference on Robotics, Automation and Mechatronics*. June 2010, pp. 528–532. DOI: 10.1109/RAMECH.2010.5513139.

[22] Richard O. Duda and Peter E. Hart. "Use of the Hough Transformation To Detect Lines and Curves in Pictures". In: *Commun. ACM* 15 (Jan. 1972), pp. 11–15. DOI: 10.1145/361237.361242.

[23] John S. Bridle. "Probabilistic Interpretation of Feedforward Classification Network Outputs, with Relationships to Statistical Pattern Recognition". In: Jan. 1990, pp. 227–236. DOI: 10.1007/978-3-642-76153-9_28.

[24] Thomas G. Dietterich. "Ensemble Methods in Machine Learning". In: *Multiple Classifier Systems*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, pp. 1–15. ISBN: 978-3-540-45014-6.

[25] Andrej Karpathy. *CS231n Convolutional Neural Networks for Visual Recognition*. 2018. URL: http://cs231n.github.io/neural-networks-1/ (visited on 03/19/2019).

[26] G. Cybenko. "Approximation by Superpositions of a Sigmoidal Function". In: *Math. Control Signals Systems* 2 (Feb. 1989), pp. 303–314. DOI: 10.1007/BF02551274.

[27] Ali Sharif Razavian et al. "CNN Features off-the-shelf: an Astounding Baseline for Recognition". In: *CoRR* abs/1403.6382 (2014). arXiv: 1403.6382. URL: http://arxiv.org/abs/1403.6382.

[28] Sebastian Ruder. "An overview of gradient descent optimization algorithms". In: *CoRR* abs/1609.04747 (2016). arXiv: 1609.04747. URL: http://arxiv.org/abs/1609.04747.