BLINK COUNTER USING CONVOLUTION NEURAL NETWORK

Group 3 - Bui Tuan Huy, Huynh Minh Triet, Le Tat Thanh
AIP391 - TrungNQ
10/07/2022

ABSTRACT

A real-time algorithm to detect eye blinks in a video sequence from a phone camera is proposed. Simple landmark detectors, trained on Closed Eyes in The Wild datasets exhibit excellent robustness against a head orientation with respect to a camera, varying illumination and facial expressions. We show that the landmarks are detected precisely enough to reliably estimate the level of the eye opening. The proposed algorithm therefore estimates the landmark positions, extracts eyes frame and resize the cropped image. Finally, a convolution neural network classifier detects eye blink was used to predict. Based on the experimental outcome the projected method achieves an accuracy of estimate 98%. The prototype developed serves as a base for further development of this process to achieve better safety and daily life usage.

I. INTRODUCTION

Detecting eye blinks is important for instance in systems that monitor a human operator vigilance, e.g., driver drowsiness, in systems that warn a computer user staring at the screen without blinking for a long time to prevent the dry eye and the computer vision syndromes, in human computer interfaces that ease communication for disabled people, or for anti-spoofing protection in face recognition systems. Existing methods are either active or passive. Active methods are reliable but use special hardware, often expensive and intrusive, e.g., infrared cameras and illuminators, wearable devices, glasses with a special close-up camera observing the eyes. While the passive systems rely on a standard remote camera only. Many methods have been proposed to automatically detect eye blinks in a video sequence. Several methods are based on a motion estimation in the eye region. Typically, the face and eyes are detected by a Viola-Jones type detector. Next, motion in the eye area is estimated from optical flow, by sparse tracking, or by frame-to-frame intensity differencing and adaptive thresholding. Finally, a decision is made whether the eyes are or are not covered by eyelids. A different approach is to infer the state of the eye opening from a single image, as e.g., by correlation matching with open and closed eye templates, a heuristic horizontal or vertical image intensity projection over the eye region, a parametric model fitting to find the eyelids, or active shape models. A major drawback of the previous approaches is that they usually implicitly impose too strong requirements on the setup, in the sense of a relative face-camera pose (head orientation), image resolution, illumination, motion dynamics, etc. Especially the heuristic methods that use raw image

intensity are likely to be very sensitive despite their real-time performance. However nowadays, robust real-time facial landmark detectors that capture most of the characteristic points on a human face image, including eye corners and eyelids, are available, see Fig. 1. Most of the state-of-the-art landmark detectors formulate a regression problem, where a mapping from an image into landmark positions or into another landmark parametrization is learned. These modern landmark detectors are trained on "in-the-wild datasets" and they are thus robust to varying illumination, various facial expressions, and moderate non-frontal head rotations. An average error of the landmark localization of a state-of-the-art detector is usually below five percent of the inter-ocular distance. Recent methods run even significantly super real-time. Therefore, we propose a simple but efficient algorithm to detect eye blinks by using a landmark detector with dlib and haarcascade's face detector. Finally having CNN acts as a classifier trained on examples of blinking and non-blinking patterns.

The contributions of the paper are:

1. Ability of advanced landmark detectors to reliably distinguish between the open and closed eye states is quantitatively demonstrated on a challenging dataset and for various face image resolutions.
2. A real-time eye blink detection algorithm which integrates a landmark detector and a classifier is proposed.

The rest of the paper is structured as follows: The algorithm is detailed in Sec. 2; evaluation is presented in Sec. 3. Finally, Sec. 4 concludes the paper.

II. PROPOSED WORK

This work is about developing an artificial intelligence (AI) model that is capable of identifying a person and telling whether his/her eyes are open

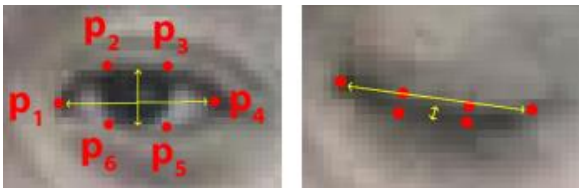

Figure 1: Open and closed eyes with landmarks

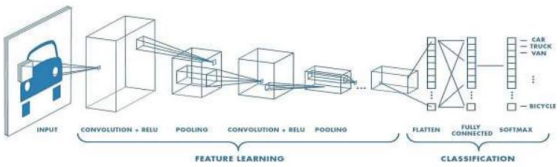

Figure 2: Eye blink detection model

or close. The model developed is a deep learning model and should take the images as input and result in an output specifying the eyes in the image are open or closed. This work aims to provide a solution for eye blink detection in automotive safety. The approach followed in this work follows retraining on the previously build models, to utilize the metrics of those models to achieve better performance. The general flow of CNN is given in Figure 2.

The basic components mentioned in Figure 1 consists of the operations as follows:

– **The convolution steps**

To mine, the characteristics of the entered image CNN are employed. Convolution gives the spatial association among pixels by understanding image characteristics employing small shapes of entered facts. The Convolution phase is shown in Figure 2.

– Design (optimizer, loss function and layers) CNN consists of an input layer, an output layer, and various hidden layers. Each sequence of layers in the CNNs hidden layers 'Convolve' with exponentiation. ReLU is the very frequently employed activation function in CNN and is shown in Figure 3. Activation functions are mathematical estimates that regulate the CNNs output.

– Optimizer

Optimizers are methods that are employed to transform the characteristics of any CNN in terms of learning proportions and weights to lessen the shortfalls. The feature extraction layer of this work makes use of Adam. Optimization policies are accountable for dropping the losses and to deliver the utmost precise consequences possible. The Adam uses dW and db for each epoch and it also does the momentum exponentially weighted average as shown in (1) and (2):

$$V_{dw} = \beta 1 \cdot V_{dw} + (1 - \beta 1) \cdot dW (1)$$

$$S_{db} = \beta 1 \cdot V_{db} + (1 - \beta 1) \cdot db$$

$$S_{dw} = \beta 2 \cdot S_{dw} + (1 - \beta 2) \cdot dW^2 (2)$$

$$S_{db} = \beta 2 \cdot S_{db} + (1 - \beta 2) \cdot db^2$$

Implementing bias correction in typical Adam's implementation is next. Here, Vcorrected and Scorrected means after the correction of the bias (shown in (5) and (6)).

$$V_{dWcorrected} = V_{dW}/(1 - \beta 1t) (5)$$

$$V_{dbcorrected} = V_{db}/(1 - \beta 1t)$$

$$S_{dWcorrected} = S_{dW}/(1 - \beta 2t) (6)$$

$$S_{dbcorrected} = S_{db}/(1 - \beta 2t)$$

Lastly, the update will be performed.

$$W = W - lr \cdot \left(\frac{V_{dWcorrected}}{\sqrt{S_{dWcorrected}} + \varepsilon} \right)$$

$$b = b - lr \cdot \left(\frac{V_{dbcorrected}}{\sqrt{S_{dbcorrected}} + \varepsilon} \right)$$

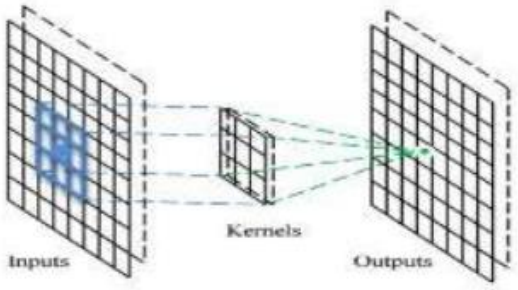

Figure 3. Convolution step

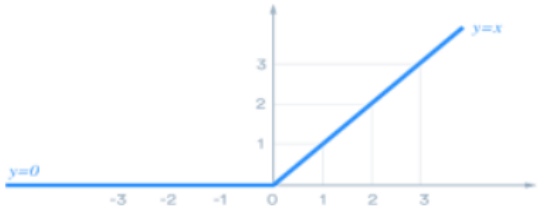

Figure 4. ReLU activation function

-The epsilon $'\varepsilon'$ is a very small number to avoid dividing by zero ($\approx 10\text{^} - 8$)

-Two Betas 1 and 2 are hyperparameters which control the two weighted averages.

-The learning rate (lr) is a range of values to be tested to see what works best for the problem.

- Loss Function

Loss function supports improving the limitations of the CNNs by calculating the loss. This work makes use of Binary cross-entropy for its binary nature, it measures the prediction from the true value (which is either 0 or 1) is for every class and then finds the mean of these class-wise faults to acquire the ultimate loss. The binary cross-entropy loss is defined using as below.

$$CE = \sum_{i=1}^{c'=2} t_i \log(f(s_i)) = -t_1 \log(f(s_1)) - (1 - t_1) \log(1 - f(s_1))$$

Here, C' = 2, C1, and C2 are two classes assumed, t1 and s1 are the value for C1, t2=1 - t1, and s2=1 - s1 is the value of C2. The loss can be represented as.

$$CE = \begin{cases} -\log(f(s_1)) & if\ t_1 = 1 \\ -\log(1 - f(s_1)) & if\ t_1 = 0 \end{cases}$$

– Pooling

CNN may comprise general or universal pooling layers to rationalize the fundamental calculation. Pooling layers diminish the magnitudes of the facts by merging the yields of neuron groups at one layer with particular neuron in the subsequent layer. Generic pooling unites minor clusters, naturally 2x2. Universal pooling proceeds on every other neuron in the convolutional layer. Pooling might calculate a maximum or a mean. The pooling phase is shown in Figure 5.

– Fully connected layer

This is typically the layer in any CNN upon which the Flatten and Dense layers are applied. The pooled features are transformed into a specific column by the Flatten function. This column is dispatched to the fully connected layer. The fully connected layer is inserted into CNN with the help of Dense. A fully connected layer links each neuron in a single layer with each other neuron in the next layer and is similar to MLP. The flattened matrix moves around this layer to categorize the images.

III. DESIGN AND IMPLEMENTATION

1.Design:

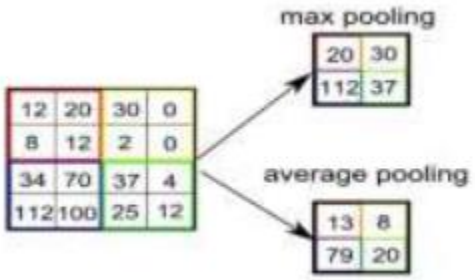

Figure 5. Pooling step

CNN's are used to implement this projected work. CNN's are versions of MLPs and uses fully connected layer. The Design of this work is given briefly in the high-level design diagram in Figure 6, were the loading of the current dataset (closed eyes in the wild – (CEW)) and adding of feature extraction layer on the CNN are mentioned.

The steps followed are:

Step 1: Start.

Step 2: Load the Dataset (closed eyes in the wild). - In this work, we have employed the open dataset published by 'Xiaoyang Tan' from Nanjing University of Aeronautics and Astronautics. A sample of the images belonging to the set closed eyes or open eyes is shown in Figure 6 and Figure 7.

Step 3: Resize and Reshape every image in the Dataset.

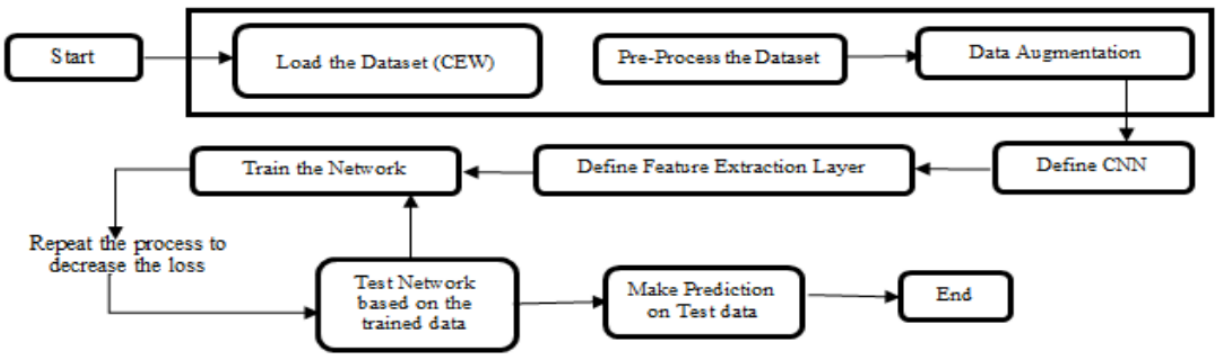

Figure 6. Processing Diagram

Step 4: Rotate and flip every image for better performance during training.

Step 5: Select the model and architecture according to the problem statement. - The CNN used for this model and the output of the model is binary i.e., eyes closed or eyes open.

Step 6: Write the top Feature Extraction Layer to detect the state of the eyes.

Step 7: Training - The last layers are fully connected network layers followed by 'Sigmoid activation' for classification in the output layer. The weights in the top layers of the pre-trained prototypes are adjusted further to upsurge the functioning. This will adjust the weights to associate with the feature related to the dataset.

Step 8: Testing the model and tuning of hyper-parameters. Repeat step 7.

- The fine-tuning approach was employed in training the model. CNN's are similar to normal NN. They consist of neurons with learnable biases and weights. Each neuron takes the data and does the dot product. The complete network gives one differential score function and still has a loss function.

Step 9: Make Predictions on completely new unseen data.

Step 10: End. Save the model and deploy.

2.Prototyping and Testing:

The testing phase of the project was initially focused on developing a mobile app to test the model trained. Finally, a webcam-based prototype was developed to test the model trained for the problem statement, using dlib facial landmarks to detect faces in the video stream and then pre-process the detected face for inference with the model. The results after the training show graphs about the loss and accurateness details throughout the training and validation phase.

Figure 7. Closed eyes and open eyes

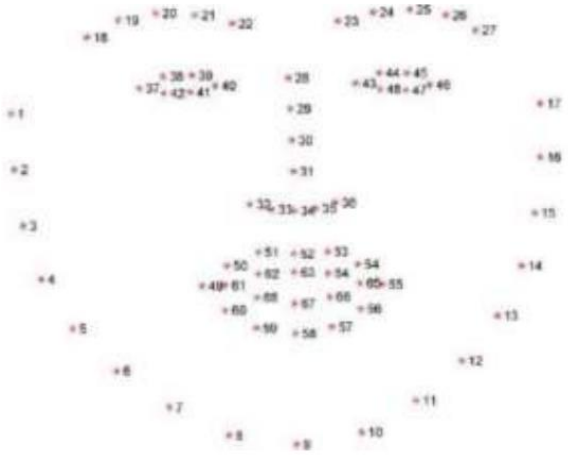

Figure 8. dlib 68-point facial landmarks

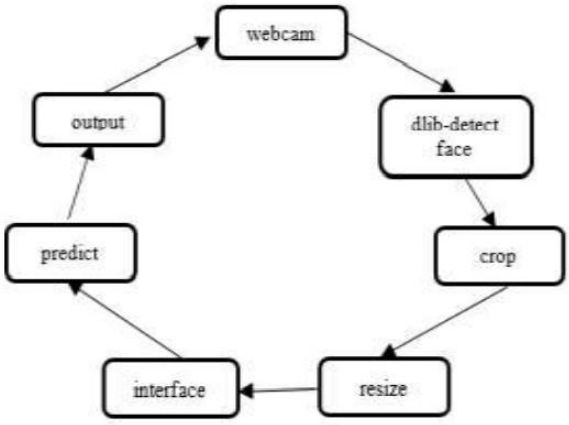

Figure 9. Flow of demo

The webcam approach takes the live video input from the webcam and was used for testing the model. Since the model was trained with only images of face data with eyes closed or open, the model can only inference on face images. To detect and crop faces from the input stream, dlib facial detector was used to detect faces in the input feel. The dlib facial detector works on the facial landmarks. The dlib facial landmark is a collection of points and is used to detect various parts of the face using 68 points on the face. These facial landmarks used by dlib facial detector are given in Figure 8 showing how the points make us the face of any human. The whole flow of the demo application is given in Figure 9.

The step followed are:

– Webcam: any webcam with a picture quality of 720p or greater.

– dlib: the facial detector to detect faces in the webcam feed, to locate the face in the wild.

– Crop: The location of faces is obtained from dlib facial detector and is cropped consisting of only the eyes and nothing else.

– Resize: The model is more efficient to process images of dimensions 244*244, so the image of the face is resized accordingly.

– Inference: The image is converted into tensors and inferences ready for prediction.

– Predict: The image is passed for prediction with a model trained for.

– Output: The output given out by the model is translated into open or close and displayed on the screen with a counter of how many time eyes have blinked. If either of the eyes is detected, the program will return none.

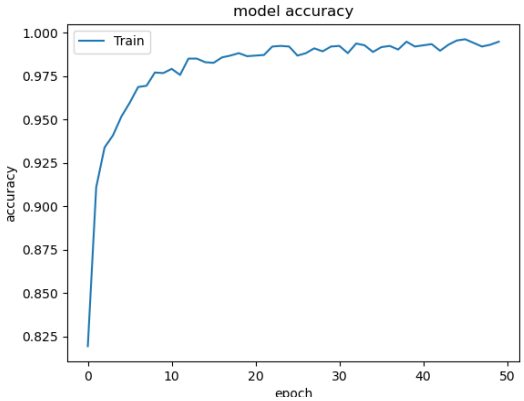

Figure 10. Accuracy during training throughout 50 epochs

Figure 11. Real-time demo, count when both eyes open after finishing a cycle

3.Result and Discussion:

Results obtained during the training of the network is given in Figure 10. The accuracy mentioned here is concerning the validation data given during the training time and is subject only to the data in the dataset. The output is a demonstration of a student in the group. It is given in Figure 11.

IV. CONCLUSION

The eye blink detection work finally served its purpose of detecting the state of the eyes in a given image. This work aims to provide a solution for reducing accidents caused by human error and as a daily life helper for people with and without any disorders. Many eye blink detection applications developed are serving these purposes, this is a model to help improve the performance of the detection rate as well as learning the progress throughout the problem. In future work, we try to concentrate on detecting eyes on the faces with sunglasses; only one eye closed and side view.

We see a limitation that a fixed blink duration for all subjects was assumed, although everyone's blink lasts differently. The results could be improved by an adaptive approach. Another limitation is in the eye-opening estimate. While the model is estimated from a 2D image, it is fairly insensitive to a head orientation, but may lose discriminability for out of plane rotations. Sometimes, the results may suffer from counter jumping number as Asian eyes are not as big as those in the dataset. An approach to the problem might be to add more landmarks around the eyes or using 3D landmarks.

REFERENCES

https://khawlajlassi.medium.com/deep-learning-optimization-techniques-2969dd05cd53
http://parnec.nuaa.edu.cn/_upload/tpl/02/db/731/template731/pages/xtan/ClosedEyeDatabases.html
https://www.researchgate.net/publication/350813427_Eye_blink_detection_using_CNN_to_detect_drowsiness_level_in_drivers_for_road_safety
https://pyimagesearch.com/2017/04/24/eye-blink-detection-opencv-python-dlib/
https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/
https://en.wikipedia.org/wiki/Convolutional_neural_network