EasyChair Preprint

№ 11300

Distraction Evaluation by Facial Landmark Detection with Lightweight Multi-Task Neural Network

Xie Xin-Hua and Wu Yi-Chao

EasyChair preprints are intended for rapid dissemination of research results and are integrated with the rest of EasyChair.

November 16, 2023

Distraction Evaluation by Facial Landmark Detection with Lightweight Multi-Task Neural Network

Xin-Hua Xei[1] and Yi-Chao Wu[1,*]

[1] Department of Electronic Engineering, National Yunlin University of Science and Technology, Yunlin, Taiwan
alanwu@yuntech.edu.tw

ABSTRACT

Recently, most of distraction detection research results focus on the driver's distraction detection in a car and the detection object is almost the one. However, the detection object may be more than two in some application scenarios. For example, using a camera to perform attention detection on multiple students, how to accurately assess students' attention without disrupting their learning becomes our primary research goal. In this paper, the behavior of student was defined as the student's distraction while the student turns his head, yawning, and eyes closing in the classroom. In this condition, most of existed researches only could be used for one-object detection but could not be used for multi-object detection through one detector. Moreover, less of researches could be used for facial landmark detection with multi-object. Hence, a distraction evaluation by facial landmark detection for multi-object by multi-task neural Network is required. To reduce the cost and space, and improve the ease of installation, our proposal is designed with the embedded system. Therefore, a distraction evaluation by facial landmark detection with lightweight multi-task neural network, DEFLD-LMTNN, was proposed in this paper to address the above issues. In DEFLD-LMTNN, the distraction detection could be applied for multi-object. When the behavior was evaluated as our defined distraction, the student will be marked as distracted in the monitor screen and an alert could be notified to teacher immediately. The teacher also could track student's learning status based on the number or frequency of distraction afterwards by our DEFLD-LMTNN. In the experimental results, the accuracy of DEFLD-LMTNN could be up to 90%. It was proved that distraction evaluation by facial landmark detection with lightweight multi-task neural network, DEFLD-LMTNN, proposed in this paper could be applied for distraction evaluation with multi-object in the embedded system with low cost and space.

Keywords: Distraction Detection, Multi-Object Detection, Facial Landmark Detection, Embedded System, Lightweight Multi-Task Neural Network

1. INTRODUCTION

Facial recognition technology had attracted a lot of attention recently due to the progress of information and communications technology. Thus, various fields of facial recognition technology were used in different applications, such as security, human-computer interaction, and AR/VR. Since the facial recognition is based on facial feature detection, how to propose the efficient and accurate facial feature extraction and recognition are important for the facial recognition. In daily life, facial expressions have always been a crucial means of conveying important messages in social interactions. Facial expressions could reflect an individual's emotions, concentration, and other states. With the development of deep learning technology, attention evaluation based on facial features has become a popular research topic. Traditional facial feature detection methods are based on machine learning for feature extraction. However two challenges are occurred. In the first challenge, the stability and accuracy of traditional methods could not be used in complex scenarios directly, such as low light conditions or occlusions. Secondly, the existed methods often required the significant computational resources not to be unsuitable for embedded device. Therefore, a new efficient facial feature detection lightweight model to address the above challenges is needed for distraction evaluation. In the current stage of research papers, many propose a series of model compression techniques to alleviate computational burdens. For instance, distillation technology [1] is to transfer the knowledge of a large complex model (teacher model) to a lightweight model (student model) for reduce computing requirements. Pruning techniques [2]-[3] reduces computational load by eliminating redundant weights and simplifying model structures. Weight binarization [4] restricting weights to $+1$ and -1, enabling accelerated inference in resource-constrained embedded systems. In this paper, we propose a lightweight multitask neural network, DEFLD-LMTNN, designed for facial detection. This network is divided into two tasks: face positioning network and face key points network. Our primary objective is to perform attention detection on multiple targets using a single camera in an embedded system without the need for any additional sensors. The remaining chapters of this thesis are structured as follows: Chapter 2 introduces relevant knowledge in the field, Chapter 3 presents the design of the DEFLD-LMTNN model architecture, Chapter 4 proposes our attention detection evaluation algorithm, Chapter 5 discusses the experimental results, and finally, Chapter 6 provides the conclusion and future prospects.

2. RELATE WORK

In the field of facial recognition and facial feature detection, precise facial key points are crucial. The challenge of facial key points detection lies in handling variations in various scenes, lighting conditions, and angles, while addressing factors such as facial occlusion and expression changes. These challenges have long been the difficulties faced by researchers. Currently, there are several outstanding architectures proposing solutions to these problems. For instance, RetinaFace [5] employs the technique of multi-scale feature map fusion, enabling effective handling of faces of different sizes and angles. Additionally, this architecture adopts the lightweight MobileNet [6] model for operation on embedded systems. MTCNN [7] is a multi-task neural network composed of

P-Net, R-Net, and O-Net. P-Net generates preliminary candidate face regions by sliding windows on feature maps of different scales. R-Net further filters the results of P-Net through a series of convolutions, and O-Net, similar to R-Net, aims to output the final results, including face confidence, face bounding box, and facial key points through deep network. YOLOV5Face [8] adopts the method of treating faces as objects for detection and directly regresses facial key points within the detection box. Furthermore, to simulate the differences in face size caused by different distances in real environments, this architecture is designed with detection heads of different scales. The aforementioned methods are all based on detecting 5 face facial key points. In contrast, the DEFLD-LMTNN proposed in this paper is based on 68 facial key points. In situations where the picture is obscured or blurred, Distraction detection can be judge by utilizing other key points, thus improving accuracy.

3. MODEL DESIGN

This chapter introduces the proposed lightweight multi-task neural network (DEFLD-LMTNN) for attention assessment, as illustrated in Fig. 1 The network consists of both a face positioning network and face key points network. Initially, the face position is determined by the face positioning network. Subsequently, the feature information from the located face serves as the input for the face key points network. Finally, the facial key points are obtained to assess whether students are experiencing distraction or fatigue.

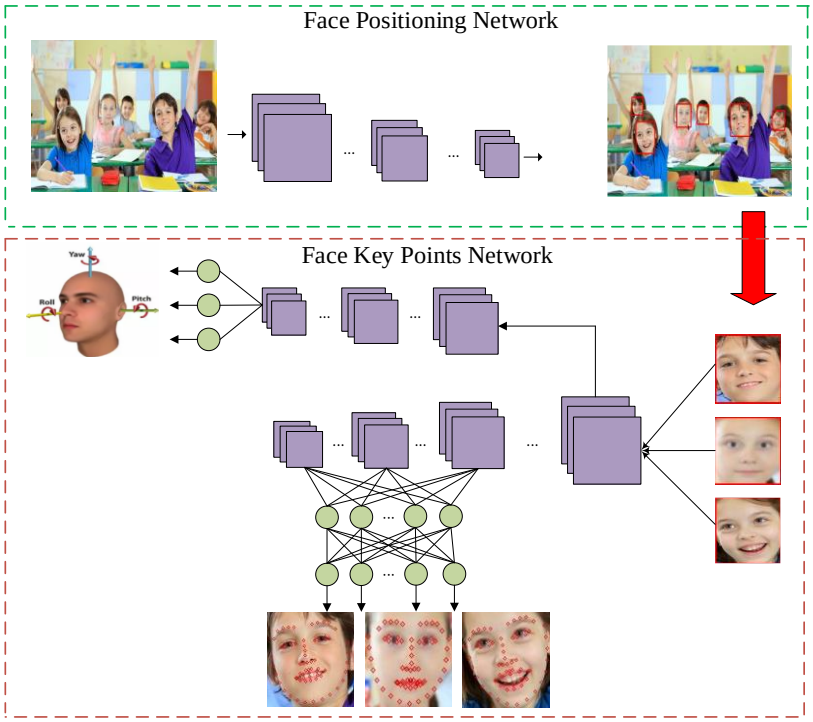

Fig. 1. Schematic diagram of the structure of DEFLD-LMTNN

3.1 Face Positioning Network

The first architecture in this paper is the face positioning network, which is used to locate faces in the images. Finding the position of faces is crucial for providing valuable information for the subsequent distraction detection. The face positioning network is designed with a backbone, neck, and detection head. The loss function consists of three components: confidence, class, and bounding box. Confidence and class components use (focal loss, FL) [9]. Focal loss (FL) is primarily used to address the issue of mismatched positive and negative samples and handling difficult samples. If the number of negative samples in a grid is larger than the positive samples, the task tends to focus on the negative samples, making it challenging to accurately learn the features of positive samples. *FL* was defined as (1), where the parameters ω and γ are the adjustable constants. Parameter ω controls the balance between positive and negative samples and γ manages the difficulty in distinguishing difficult samples.

$$FL = -\omega(1-p)^{\gamma} log(p) \tag{1}$$

The loss function *CIOU* (Complete-IOU) was defined as (2) [10]. It takes into account the distance between the real box and the predicted box, the overlapping area, and the distance between their center points. This approach further stabilizes the regression of the target box. As shown in Fig. 2, the orange box represents the predicted box, while the blue box represents the ground truth box. Parameters b and b_{gt} denote the centers of the two rectangular boxes, parameter c represents the diagonal of the minimum enclosing area of the two rectangular boxes, and parameter ρ represents the Euclidean distance between the centers of the two rectangular boxes. IOU (Intersection over Union) represents the overlap area between the predicted box and the real box

$$CIOU = 1 - IOU + \frac{\rho^2(b, b^{gt})}{c^2} + \alpha v \tag{2}$$

In *CIOU*, there are two adjustable parameters: α and v. α is used to penalize the v parameter. The v represents the aspect ratio between the predicted box and the real box. When IOU is large, indicating a significant overlapping area, increasing α affects the v value. Conversely, when IOU is small, indicating a small overlapping area, decreasing α influences the reduction of v.

$$\alpha = \frac{v}{(1 - IOU) + v} \tag{3}$$

From this, it can be deduced that if the overlapping area between the predicted box and the real box is small, the influence of *v* is relatively minor. In this scenario, the focus is on reducing the distance between the two. Conversely, if the overlapping area between them is large, the influence of *v* is significant. In this case, the emphasis is on adjusting the aspect ratio between the real box and the predicted box.

$$v = \frac{4}{\pi^2}(arctan\frac{w^{gt}}{h^{gt}} - arctan\frac{w}{h})^2 \tag{4}$$

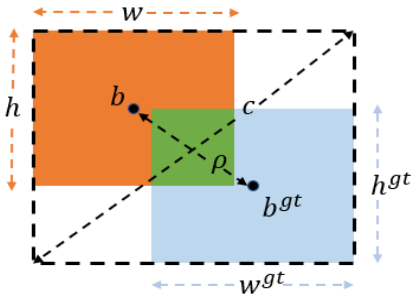

Fig. 2. Schematic diagram of CIOU loss function

3.1.1 Backbone Design

The backbone design of this study was inspired by the GoogleNet [11] network architecture. GoogleNet proposed the fusion of convolutional kernels of different sizes to obtain better feature representations, as shown in Fig. 3, Asymmetric Block is designed by our backbone. CBR (Conv + BatchNorm + ReLU) represents performing normalization calculation after convolution and then obtaining the result through ReLU. DWCBR (Depth-wise Conv + BatchNorm + ReLU) represents performing normalization calculation after depth-wise separable convolution and then obtaining the result through ReLU. ADD is the result of network residual operation. Concat is the result of channel merging. CB (Conv + BatchNorm) stands for normalization after convolution. DWCB (Depth-wise Conv + BatchNorm) stand for depth-wise separable convolution after normalization. The architecture of this paper first performs channel amplification through 1×1 CBR, subsequently two branches were designed with 3×3 and 5×5 DWCBR to extract deeper features. The networks in these two branches utilized asymmetric convolutions, splitting the kernel into $k \times 1$ and $1 \times k$, to focus on both horizontal and vertical features simultaneously. Finally, the feature maps from the two branches were merged. 1×1 CB was employed for the final channel output.

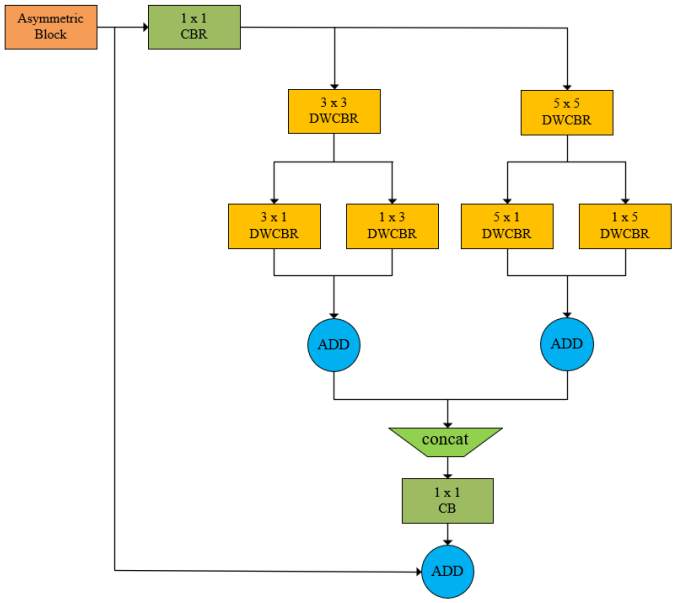

Fig. 3. Asymmetric Block Architecture Diagram

3.1.2 Neck Design

The neck architecture adopts the FPN [12] design. In this study, this paper takes input images with a size of 512×288, as shown in Fig. 4, the image undergoes a total of five down-samplings. The feature maps from L1 and L2 belong to the shallow layers of the network. Due to their large sizes, which would consume a significant amount of computational resources, only the feature maps from L3, L4, and L5 are utilized. By fusing feature maps of different sizes, the accuracy of target detection can be enhanced.

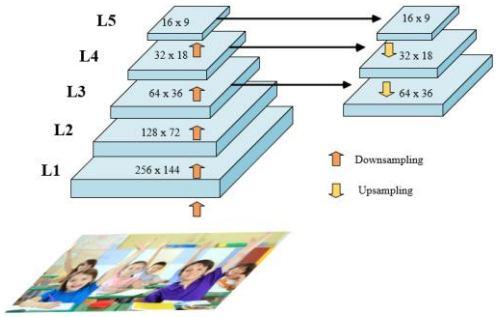

Fig. 4. Schematic diagram of downsampling

The neck neural network adopts the GhostNet [13] architecture, as shown in Fig. 5, in the Ghost_Branch, there are two branches: one consists of 3×3 DWCBR, and the other is composed of Ghost_Modules. The Ghost_Module comprises 1×1CBR and 3×3 DWCBR.

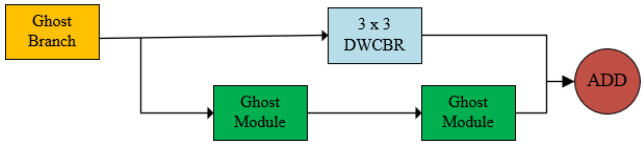

Fig. 5. Ghost Branch Architecture Diagram

3.1.3 Detected Head

After passing through the neck neural network, this paper generates three types of detection heads in different sizes: 16×9 for detecting large objects, 32×18 for detecting medium-sized objects, and 64×32 for detecting small objects. These detection heads include coordinates (x, y, w, h), confidence scores, and categories. Finally, the original image is segmented into regions corresponding to the sizes of these detection heads. Non-maximum suppression [14] is applied to these regions to obtain our final prediction results. As shown in Table1, the overall architecture of the face key points network in this paper is as follows: the ID represents the input network number, Layer Name refers to the network architecture introduced in the previous sections, among them, Asy_Block is the backbone design mentioned Asymmetric Block in Chapter 3.1.1, and Ghost_Branch and Ghost_Module are the neck design mentioned in Chapter 3.1.2. Output indicates the output image size, S represents the stride, Expand Ratio represents the number of hidden layers, and the model's channels are enlarged by a corresponding Expand Ratio multiplier based on the input channels. Output Channels represent the output channels of the network.

Table 1 Face Positioning Network Architecture

ID	Layer Name	Input Layer ID	Output Size	S	Expand Ratio	Output Channels
0	CBR	-	512x288	2	-	3
1	Asy_Block	0	256x144	2	-	16
2	Asy_Block	1	128x72	1	2	16
3	Asy_Block	2	128x72	2	-	16
4	Asy_Block	3	64x36	1	2	32
5	Asy_Block	4	64x36	2	-	32
6	Asy_Block	5	32x18	1	2	64
7	Asy_Block	6	32x18	2	-	64
8	Asy_Block	7	16x9	1	2	128
9	Ghost_Module	8	16x9	1	-	128
10	Upsample	9	32x18	-	-	128
11	Concat	5, 10	32x18	-	-	160
12	Ghost_Branch	11	32x18	1	-	64
13	Ghost_Module	12	32x18	1	-	32
14	Upsample	13	64x36	-	-	32
15	Concat	3, 14	64x36	-	-	48
16	Ghost_Branch	15	64x36	2	-	32
17	Ghost_Module	16	32x18	1	-	32
18	Concat	13, 17	32x18	-	-	64
19	Ghost_Branch	18	32x18	-	-	32
20	Ghost_Module	19	16x9	-	-	64
21	Concat	9, 20	16x9	-	-	192
22	Ghost_Branch	21	16x9	-	-	64
23	**Output**	**15**	**64x36**	**-**	**-**	**8**
24	**Output 1**	**18**	**32x18**	**-**	**-**	**8**
25	**Output 2**	**21**	**16x9**	**-**	**-**	**8**

3.2 Face Key Points Network

The neural network design for facial key points can be divided into Block 1, Block 2, Detected Head, and an Auxiliary Network. Through experiments, this paper discovered that utilizing the auxiliary network to predict the Euler angles in the 3D world and incorporating the Euler angle information trained from the auxiliary network into the loss function contributes to a faster convergence speed of the model.

3.2.1 Block 1 Network

The design of Block 1 is based on the design of MobileNet. As shown in Fig. 6, first 1×1 CBR is used to extend the number of neural network channels to generate more feature maps, and then a 3×3 DWCBR is used to do the sampling to reduce the amount of computation, and then the final channel is output by a liner 1×1 CB, and the last layer of 1×1 convolution adopts a linear approach without an activation function because the mapping of the high-dimensional channels in 3×3 back into the lowdimensional channels in 1×1 produces a large number of values less than 0, and the activation function used in ReLU will destroy the feature information.

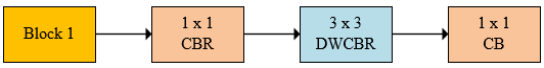

Fig. 6. Block 1 Architecture Diagram

3.2.2 Block 2 Network

The design of Block 2 is based on the GhostNet, MobileNet, and ShuffleNet-V2 [15] architectures. The MobileNet network architecture utilizes an inverted residual design, which first performs channel amplification through 1×1 convolution, followed by deeper channel feature extraction through 3×3 convolution. Finally channel dimensionality reduction is performed by 1×1 convolution. As shown in Fig. 7, first 1×1 CBR and 3×3 DWCBR are fused with the number of channels so that more useful features can be extracted later. Next, this paper uses the idea proposed by the GhostNet architecture to halve the number of input channels and divide it into two parts. The first part is reducing the number of channels by 1×1 CB. The second part is 3×3 DWCB. Finally, the number of channels produced by both are fused to reduce the computation and enrich the features. Finally, with this design, the size of the original number of channels is kept consistent with the number of input channels. It is based on ShuffleNet-V2 which proposes that equal inputs and equal outputs can minimize memory accesses.

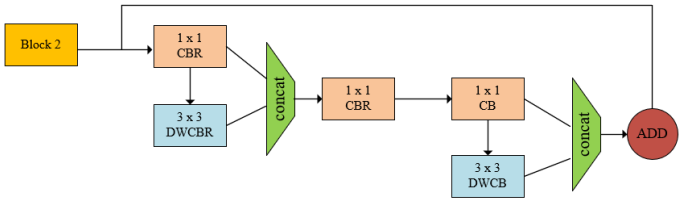

Fig. 7. Block 2 Architecture Diagram

3.2.3 Auxiliary Network

Through experiments with [16] , it is evident that incorporating auxiliary networks contributes to more accurate key points regression. Since key points regression tasks tend to be unstable during the initial stages of training, relying solely on 2D image information increases the model's complexity. Integrating 3D facial pose information into the process and merging 2D and 3D information through a designed loss function helps the model converge faster and more steadily. Facial poses can be inferred using Euler angles. In Fig. 8, the facial key points information used in this paper is displayed. By mapping annotated 2D key points *P36*, *P45*, *P33*, *P48*, and *P54* back to 3D space through camera coordinate system transformation, rotation matrices are obtained. Utilizing these rotation matrices, Euler angles can be calculated, leading to accurate predictions of head poses.

Fig. 8. Facial key point output positions and corresponding identifiers.

3.2.4 Detected Head

From the first three subsections, we can conclude that our network structure consists of Table2, this thesis has gone through five down-sampling, and after the second down-sampling, it will be divided into two streams of networks, the first one is to do the facial key points task regression, which utilizes the fusion of multi-dimensional feature maps to combine the results of the last three

down-sampling, and increases the accuracy of the model by fusing the shallow network and the deep network information, and the second one is to do the facial posture prediction, also through the three down-sampling, using the full connectivity layer to classify the three Euler angles, and the auxiliary network structure is only executed in the training stage, so it is not executed in the reasoning stage. For the face posture prediction, the three Euler angles are also categorized by using the full connectivity layer after three down-sampling. The auxiliary network architecture is only executed in the training stage, but not in the inference stage, so the auxiliary network algorithm can be eliminated in the inference stage. As shown in Table2, the overall architecture of the face key points network in this paper, 'Output' representing the 2D coordinates of the 68 facial key points used in this study. 'Output1' denotes the head pose trained by the auxiliary network.

Table 2 Face Key Points Network Architecture

ID	Layer Name	Input Layer ID	Output Size	S	Expand Ratio	Output Channels
0	CBR	-	112x112	2	-	3
1	CBR	0	56x56	1	-	64
2	Block 1	1	56x56	2	2	64
3	Block 2	2	28x28	1	2	64
4	Block 2	3	28x28	1	2	64
5	Block 2	4	28x28	1	2	64
6	Block 2	5	28x28	1	2	64
7	Block 1	6	28x28	2	2	128
8	Block 2	7	14x14	1	2	128
9	Block 2	8	14x14	1	2	128
10	Block 2	9	14x14	1	2	128
11	Block 2	10	14x14	1	2	128
12	Block 2	11	14x14	1	2	128
13	Block 2	12	14x14	1	2	128
14	Block 1	13	14x14	2	2	16
15	CBR	14	7x7	7	-	32
16	CBR	15	1x1	1	-	128
17	Fully Connect 1	14	1x1	-	-	16
18	Fully Connect 2	15	1x1	-	-	32
19	Fully Connect 3	16	1x1	-	-	128
20	Concat	17,18,19	1x1	-	-	176
21	**OUTPUT**	**20**	**1x1**	**-**	**-**	**136**
22	CBR	6	28x28	2	-	64
23	CBR	22	14x14	1	-	128
24	CBR	23	14x14	2	-	128
25	CBR	24	7x7	7	-	32
26	Fully Connect 4	25	1x1	-	-	128
27	**OUTPUT1**	**26**	**1x1**	**-**	**-**	**3**

3.2.5 Loss Function

The design of the loss function in this paper was inspired by the idea proposed in Wing loss [19], which incorporates the calculation of facial pose Euler angles in 3D space through an auxiliary network for backpropagation. The training of Wing loss adopts a two-stage approach. In the first stage, during the early training, there is a significant error between the predicted and actual values of key points. Therefore, the descent offset should be relatively large. In the second stage, in the middle to later stages of training, a few key points still exhibit significant errors between the actual and predicted values. If the loss function used during the early training is continued, key points with large errors in backpropagation would dominate the entire task, affecting the regression of other key points.

$$wing(x) = \begin{cases} w\,ln\left(1+\frac{|x|}{\epsilon}\right) & if\,|x| < w \\ |x| - C & otherwise \end{cases} \tag{5}$$

Wing loss defined as (5), the w and ϵ in Wing loss are adjustable parameters, the role of w is to control the nonlinear part in the interval $[-w, w]$, and the role of ϵ is to limit the curvature of the nonlinear region, $C = w - w\,ln(1 + w/\epsilon)$,which is to be used for connecting the linear and nonlinear parts, and the experimental results of this paper show that $w = 10, \epsilon = 2$ has a better effect on the overall network regression.

$$DL_{loss} = \frac{1}{M}\sum_{m=1}^{M}\sum_{n=1}^{N}\left(\sum_{k=1}^{K}(1-cos\,\theta^{k})\right)wing(x) \tag{6}$$

After regressing the key point positions through Wing loss, we also considered the disparities between the predicted Euler angles from the auxiliary network and the ground truth values. As shown in Equation 6, DL_{loss} is the method we proposed, Here M represents the total number of facial samples, N represents the 2D coordinates (x, y) of key points, and K represents the total number of Euler angles. In the initial stages of neural network training, due to inaccuracies in the predictions made by the auxiliary network, larger weight adjustments are propagated back during backpropagation if significant disparities exist. As the network gradually converges, most samples can be accurately predicted. However, for samples with exaggerated poses, training becomes more challenging. In such cases, the functionality of the auxiliary network can assign higher weights to these difficult-to-train samples, allowing for more effective training. Conversely simpler samples receive reduced weight adjustments.

4. METHOD

4.1 Data Preparation

WFLW [20] was used as the dataset in this paper. WFLW encompassed a wide range of facial expressions, poses, and lighting variations. To enhance the diversity of the data, the data augmentation techniques were used, such as rotation, blurring, and occlusion. In the rotation, it simulated the changes in head poses of the real world to allow our model to be better in various scenarios. Blurring was applied to enable the model to learn global features instead of localized details. Lastly, the random occlusion of partial face led the model to learn features from occluded regions. Through the data augmentation, the model could gain an understanding of different facial poses in real situations to improve the accuracy in practical applications.

4.2 Head Turning Detection Algorithm

The model architecture designed in Chapter Three is ultimately capable of detecting 68 facial key points. To infer the 3D head pose coordinates through these key points in 2D images, we need to perform a transformation in the camera coordinate system. First, we need to understand how each coordinate point in pixel space should be mapped to positions in real-world space. This mapping requires the assistance of camera intrinsic and extrinsic parameters. As shown in Fig. 9, In the camera model, there are a total of four coordinate systems: pixel coordinates refer to the initial positions of objects in image pixels, image coordinates represent the projection of the camera coordinate system onto positions in the image, and camera coordinates constitute a 3D coordinate system representing the position of the camera's optical center. These three coordinate systems constitute the camera's intrinsic parameters. The camera's extrinsic parameters correspond to the world coordinate system, representing the positions of objects in the real world. This set of parameters consists of a 3×3 rotation matrix and a 3×1 translation matrix.

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} fx & 0 & cx \\ 0 & fy & cy \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \tag{7}$$

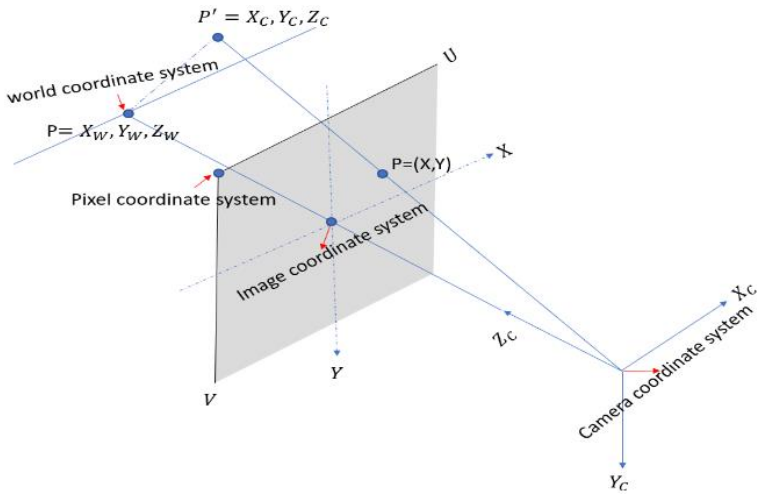

Fig. 9. Pixel coordinates, image coordinates, camera coordinates, and world coordinates schematic diagram.

After completing the camera coordinate transformation, we need to use external parameters to obtain a rotation matrix R, which is used to describe the positional relationship between the camera and the head. This paper employs the Euler angles method to determine the head's pose [21]. Euler angles are a way to describe the rotational orientation of an object in three-dimensional space. It decomposes the rotation into three angles: Pitch, Yaw, and Roll, and can be categorized into intrinsic (or internal) and extrinsic (or external) rotations. Intrinsic rotation involves rotating around one of the object's axes, while extrinsic rotation fixes one axis and rotates around the other two. In this paper, the rotation matrix obtained through the camera perspective utilizes extrinsic rotation. First, it fixes the rotation angle α around the X-axis, then the angle β around the Y-axis, and finally the angle γ around the Z-axis. Ultimately, by multiplying these three rotation matrices sequentially: $R_x(\alpha)$, $R_y(\beta)$, $R_z(\gamma)$, the rotation matrix R can be obtained. As shown in Fig. 10, the rotation around the x-axis is first fixed to obtain x', y', and z'. Subsequently, fixing the rotation around the y-axis results in x'', y'', and z''. Final fixing the rotation around the z-axis yields the final values of x''', y''', and z'''

$$R_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\alpha & -sin\alpha \\ 0 & sin\alpha & cos\alpha \end{bmatrix} \tag{8}$$

$$R_y(\beta) = \begin{bmatrix} cos\beta & 0 & sin\beta \\ 0 & 1 & 0 \\ -sin\beta & 0 & cos\beta \end{bmatrix} \tag{9}$$

$$R_z(\gamma) = \begin{bmatrix} cos\gamma & -sin\gamma & 0 \\ sin\gamma & cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{10}$$

$$R = \begin{bmatrix} cos\beta cos\gamma & -sin\gamma cos\beta & sin\beta \\ sin\alpha sin\beta cos\gamma + cos\alpha sin\gamma & sin\alpha sin\beta sin\gamma - cos\alpha cos\gamma & -sin\alpha cos\beta \\ sin\alpha sin\gamma - cos\alpha sin\beta cos\gamma & cos\alpha sin\beta sin\gamma + sin\alpha cos\gamma & cos\alpha cos\beta \end{bmatrix} \tag{11}$$

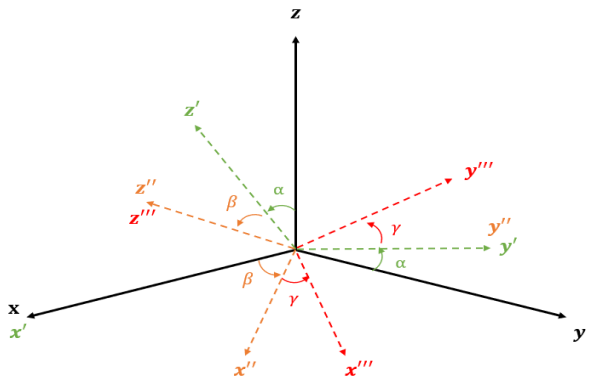

Fig. 10. Rotation matrix diagram

By using the rotation matrix, we can obtain the Pitch rotation around the X-axis, Yaw rotation around the Y-axis, and Roll rotation around the Z-axis. Through these three angles, we can predict whether the current head posture has caused a distracted state.

$$Pitch = arctan2(-R_{21}, R_{22}) \tag{12}$$

$$Yaw = arctan2(-R_{20}, \sqrt{(R_{00})^2 + (R_{10})^2}) \tag{13}$$

$$Roll = arctan2(R_{00}, R_{10}) \tag{14}$$

4.3 Fatigue Detection Algorithm

The fatigue detection algorithm in this paper involves detecting key points on the eyes and mouth calculating their Euclidean distances in 2D images. If these distances are smaller than a threshold value set by us, the system flags it as a state of fatigue. Taking Fig. 8 as an example, the thresholds for individual eyes are calculated from *P36* to *P47*. Finally, the average of the thresholds for both eyes is compared to the threshold value we set.

$$Left_{eye} = \frac{(P37 - P41) + (P38 - P40)}{2(P36 - P39)} \tag{15}$$

$$Right_{eye} = \frac{(P43 - P47) + (P41 - P46)}{2(P42 - P45)} \tag{16}$$

By utilizing six key points, namely *P48*, *P50*, *P52*, *P56*, *P58*, and *P54*, the current state of mouth openness can be obtained to determine if a person is yawning or not.

$$Yawn = \frac{(P50 - P58) + (P52 - P56)}{2(P48 - P54)} \tag{17}$$

During the experimentation, we believe it is not appropriate to solely rely on the threshold calculated from the current frame to determine whether a student is in a closed-eye state. This approach can lead to misjudgments, especially when a student is simply lowering their gaze. Therefore, this paper adopts an Adaptive Threshold as the criteria for determination. We record the eye thresholds from the student's previous five frames, calculate the average, and then compare it with our predefined threshold. Only when the average is below the predefined threshold, the student is considered to be in a state of fatigue.

$$Adaptive\ Threshold = \frac{1}{M}\sum_{m}^{M} frame_m \tag{18}$$

As shown in Figure 11, where the x-axis represents frames and the y-axis represents the threshold, the blue line represents closed-eye states, and the orange line represents the threshold set in this paper. It can be observed that between frame 0 and frame 110, all values are below the threshold we set, thus being determined as closed-eye states. Between frame 110 and frame 300, occasional values are below our set threshold, indicating instances of blinking. Through our adaptive threshold algorithm, these blinking instances will not be misclassified as closed-eye states, further improving the accuracy of the judgment.

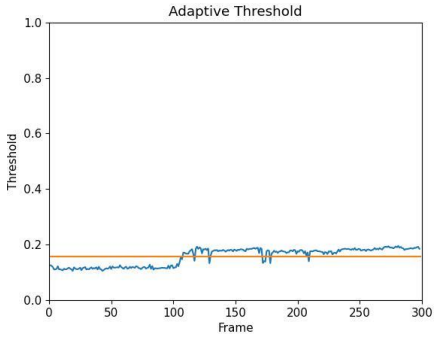

Fig. 11. Adaptive Threshold schematic diagram

5. EXPERIMENTAL ANALYSIS RESULTS

In this chapter, we will present the evaluation methods and comparative results of DEFLD-LMTNN. In this experiment, we utilized the NVIDIA GeForce RTX 4070 as the hardware platform for training our model, employing the WFLW dataset, which was divided into 7500 images for training and 2500 images for validation. Through data augmentation techniques such as rotation, blur, and occlusion, we expanded the training dataset to 75,000 images. Our model underwent a two-stage training process. In the initial pre-training stage, given the absence of weight references, we chose a larger batch size and learning rate to prevent the model from getting stuck in local minima and to enhance convergence speed. During the second training phase, we used the pre-trained weights as a reference and reduced the batch size and learning rate. Since the model had already converged in the pre-training stage, our goal was to guide the model to find the global minimum to improve accuracy. Next, we will introduce the evaluation methods and comparative results of the two models. Firstly, for the face positioning network, we employed the following evaluation metrics: Precision, Recall, F1-Score, Map, and FLOPs. The first three metrics require the use of a confusion matrix, consisting of *TP* (True Positive, predicted positive and actual positive), *TN* (True Negative, predicted negative and actual negative), *FP* (False Positive, predicted positive but actually negative), and *FN* (False Negative, predicted negative but actually positive). *Recall* is the proportion of correctly predicted positive samples among all actual positive samples. *F1-Score* is the weighted harmonic mean of precision and recall. FLOPs are millions of floating points operations per second, the unit is M. In this paper, we will modify the backbone architecture under the same dataset conditions to incorporate current state-of-the-art lightweight architectures, as shown in Table3, and compare them using MobileNet and GhostNet.

$$Precision = \frac{TP}{TP + FP} \tag{19}$$

$$Recall = \frac{TP}{TP + FN} \tag{20}$$

$$F1 - Score = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{21}$$

Table 3 Comparison of Results for Face Positioning Network in this Paper

Backbone	Precision	Recall	F1-Score	mAP@.5	mAP@.5:.95	FLOPs(M)
MobileNet	0.90	0.87	0.8847	0.937	0.661	295.05
GhostNet	0.868	0.814	0.8679	0.913	0.621	322.53
Asy_Block	**0.93**	**0.88**	**0.904**	**0.945**	**0.686**	**279.13**

Here is an overview of the evaluation metrics for the Face Key Points Network, including Accuracy, ION, IPN, FLOPs, and Parameter. Taking Fig. 8 as an example, *ION* involves normalizing the interocular distance *D (P36, P45)*.

$$ION = \frac{\sum_{i=1}^{N} ||x_{pred\_i} - x_{gt\_i}||_2}{N \times D} \tag{22}$$

IPN involves calculating the distance between the pupils of both eyes, As shown in Fig. 12, the calculation involves finding the center of points *P36* to *P41* and points *P42* to *P47*. Finally, the Euclidean distance between these two centers is computed

(Indicated by red line), resulting in the parameter *T*. By standardizing the pupil distance, for instance, using *ION* and *IPN*, we assess the difference between predicted and actual values. A smaller numerical value indicates a smaller difference between the two, thereby reflecting the accuracy of our predictions.

$$IPN = \frac{\sum_{i=1}^{N} ||x_{pred\_i} - x_{gt\_i}||_2}{N \times T} \tag{23}$$

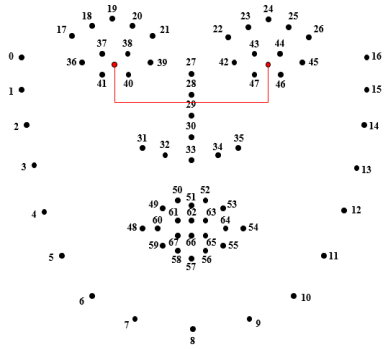

Fig. 12. IPN calculation diagram

This paper face key points network builds upon the ideas proposed in PFLD [22], aiming for improvements. Under the same dataset WFLW, we conducted comparisons using three different methods. The first method employed the model architecture and loss function used by the original authors. The second method utilized the same architecture but incorporated our custom loss function. The third method involved our own designed model architecture and loss function. The model architecture of PFLD adopts the MobileNet framework and employs the *L2* loss function. *L2* loss function defined as (24), it measures the mean squared difference between the true values and the predicted values. In contrast, our model architecture adopts the face key points network mentioned in Chapter 3, and the loss function follows our proposed DL_{loss}.

$$L2_{loss} = \frac{1}{N}\sum_{i=1}^{N}(x_i - x_i^*)^2 \tag{24}$$

Table 4 Comparison of Results for Face Key Points Network

Model / Loss function	Accuracy	ION	IPN	FLOPs(M)	Parameter(M)
MobileNet / L2	0.8702	0.065	0.0921	396.89	1.25
Face Key Points Network / L2	0.8916	0.0542	0.0767	396.89	1.25
Face Key Points Network / DL	**0.8992**	**0.0505**	**0.0715**	455.96	1.34

6. CONCLUSIONS AND FUTURE WORK

In this paper, we propose an attention assessment method based on a lightweight multitask neural network. In traditional classroom settings, teachers find it challenging to simultaneously monitor the learning status of every student. Our research achieves a one-to-many distraction detection scenario through facial feature extraction. With the model we designed, we can monitor students' attention states in real-time, including behaviors such as turning heads, yawning, and closing eyes. This method holds significant value in educational contexts. It not only assists teachers in identifying and promptly correcting distracted students but also provides a deeper understanding of students' learning conditions. With technological advancements, approaches similar to our facial feature-based attention assessment method are expected to find broader applications. We will continue to optimize our model, enhancing its accuracy and efficiency, and apply it to embedded systems to extend its usability across various fields. Our ongoing research and exploration in this area aim to contribute to the development of educational technology and enhance students' learning experiences.

REFERENCES

[1] Shan You, Chang Xu, Chao Xu, and Dacheng Tao, "Learning from multiple teacher networks ", In Knowledge Discovery and Data Mining, pp. 1285-1294, 2017.

[2] Jian-Hao Luo, Jianxin Wu, and Weiyao Lin, "Thinet: A filter level pruning method for deep neural network compression", In IEEE International Conference on Computer Vision, pp. 5058–5066, 2017.

[3] Song Han, Huizi Mao, and William J Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding", In International Conference on Learning Representations, 2016.

[4] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi, "XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks", In European Conference on Computer Vision, pp. 525-542, 2016.

[5] Jiankang Deng, Jia Guo, Yuxiang Zhou, Jinke Yu, Irene Kotsia, and Stefanos Zafeiriou, "RetinaFace: Single-stage Dense Face Localisation in the Wild", In Computing Research Repository, 2019.

[6] Mark Sandler, Andrew G. Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks". In IEEE Conference on Computer Vision and Pattern Recognition, pp. 4510-4520, 2018.

[7] Kaipeng Zhang, Zhanpeng Zhang, Zhifeng Li, and Yu Qiao, "Joint Face Detection and Alignment using Multi-task Cascaded Convolutional Networks", In Computing Research Repository, 2016.

[8] Delong Qi, Weijun Tan, Qi Yao, and Jingfeng Liu, "YOLO5Face: Why Reinventing a Face Detector", In European Conference on Computer Vision, pp. 228-244, 2022.

[9] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár, "Focal Loss for Dense Object Detection", In IEEE International Conference on Computer Vision, pp. 2999-3007, 2017.

[10] Zhaohui Zheng, Ping Wang, Dongwei Ren, Wei Liu, Rongguang Ye, Qinghua Hu, and Wangmeng Zuo, "Distance-IoU Loss: Faster and Better Learning for Bounding Box Regression", In Association for the Advancement of Artificial Intelligence, pp. 12993-13000, 2020.

[11] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich, "Going deeper with convolutions", In IEEE Conference on Computer Vision and Pattern Recognition, pp. 1-9, 2015.

[12] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie, "Feature Pyramid Networks for Object Detection", In IEEE Conference on Computer Vision and Pattern Recognition, pp. 936-944, 2017.

[13] Kai Han, Yunhe Wang, Qi Tian, Jianyuan Guo, Chunjing Xu, and Chang Xu, "GhostNet: More Features from Cheap Operations", In IEEE Conference on Computer Vision and Pattern Recognition, pp. 1577-1586, 2020.

[14] Jan Hosang, Rodrigo Benenson, and Bernt Schiele, "Learning non-maximum suppression", In IEEE Conference on Computer Vision and Pattern Recognition, pp. 6469-6477, 2017.

[15] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun, "ShuffleNet V2: Practical Guidelines for Efficient CNN Architecture Design", In European Conference on Computer Vision, pp. 122-138, 2018.

[16] Amin Jourabloo, Xiaoming Liu, "Pose-Invariant 3D Face Alignment", In IEEE International Conference on Computer Vision, pp. 3694-3702, 2015.

[17] Wayne Wu, Chen Qian, Shuo Yang, Quan Wang, Yici Cai, and Qiang Zhou, "Look at Boundary: A Boundary-Aware Face Alignment Algorithm", In IEEE Conference on Computer Vision and Pattern Recognition, pp. 2129-2138, 2018.

[18] Amit Kumar, and Rama Chellappa, "Disentangling 3D Pose in A Dendritic CNN for Unconstrained 2D Face Alignment", In IEEE Conference on Computer Vision and Pattern Recognition, pp. 430-439, 2018.

[19] Zhen-Hua Feng, Josef Kittler, Muhammad Awais, Patrik Huber, and Xiao-Jun Wu, "Wing Loss for Robust Facial Landmark Localisation with Convolutional Neural Networks", In IEEE Conference on Computer Vision and Pattern Recognition, pp. 2235-2245, 2018.

[20] Shuo Yang, Ping Luo, Chen Change Loy, and Xiaoou Tang, "WIDER FACE: A Face Detection Benchmark", In IEEE Conference on Computer Vision and Pattern Recognition, pp. 5525-5533, 2016.

[21] Aryaman Gupta, Kalpit Thakkar, Vineet Gandhi, and P J Narayanan, "Nose, eyes and ears: Head pose estimation by locating facial keypoints", In IEEE Sinnal Processing Society, pp. 1977-1981, 2019.

[22] Xiaojie Guo, Siyuan Li, Jinke Yu, Jiawan Zhang, Jiayi Ma, Lin Ma, Wei Liu, and Haibin Ling, "PFLD: A Practical Facial Landmark Detector", In Computing Research Repository, 2019.