3DCNN-based Real-time Driver Fatigue Behavior Detection in Urban Rail Transit

Yiqing Liu¹, Tao Zhang¹, and Zhen Li²,³
¹School of Information Department, Beijing University of Technology, Beijing, China, 100124
²School of Electronics and Information Engineering, Beijing Jiaotong University, Beijing, China, 100044
³National Engineering Laboratory for Urban Rail Transit Communication and Operation Control, Beijing, China, 100044

Corresponding author: Yiqing Liu (e-mail: liuyiqing95@sina.com).

This work was supported by the Beijing University of Technology.

ABSTRACT With the rapid development of urban rail transit, traffic safety has become the focus of attention and people are paying increasing attention to the prevention of fatigue driving. "Gesture and oral instructions of urban rail traffic drivers" is operational actions of drivers written in the Chinese metro operation specification. It is a method to prevent drivers from fatigue driving and ensure safety. However, there is a lack of scientific detection methods. We combine the standard traffic operational actions with fatigue action to construct a fatigue detection system that is suitable for the urban rail transit industry. The system includes a dynamic tracking model for the large-scale operation of rail transit drivers and a dual-input action discrimination model based on a three-dimensional convolutional neural network (3DCNN). The model sets the skipping frame and continuous frame as two inputs of the model, and extracts five channels of information from the two inputs. Dual-input multi-channel information enables the model to learn not only the spatial and temporal information of the entire action, but also the subtle changes of the action. First, we trained and validated the dual-input model based on a 3DCNN using the open dataset KTH, which contains several variations. Then, the model trained on KTH was migrated to our data using the transfer learning method, which saved training time and achieves an accuracy of 98.41%. This transfer learning scheme can also be applied when new categories are encountered in practice. Finally, we discussed and envisaged the future optimization of the system.

INDEX TERMS Action recognition, dual-input model, fatigue driving monitoring, three-dimensional convolutional neural network.

I.INTRODUCTION As the main representative of green travel, urban rail transit is the primary approach to solve urban traffic congestion, and has become the key area of development for public transport in large and medium-sized cities. By the end of 2018, 493 cities in 72 countries and regions had built urban rail transit, with the operating distance exceeding 26,100 km [1]. Urban rail transit has been built in 35 cities in mainland China, with 185 operating lines, which includes 20 new lines compared with 2017 [2]. In 2018, the total number of passenger traffic completed in mainland China was 210.7 billion, which is an increase of 14% compared with 2017 [2]. Because of the huge pressure of road traffic, costs in terms of time and money, and increasingly severe environmental problems, an increasing number of people are choosing rail transit. In some mega-cities, such as Beijing, Moscow, Shanghai, New York, and London, urban rail transit represents more than 50% of the transportation tasks for the entire city and has become a vital public transport tool in people's lives.

In the process of the rapid development of urban rail transit, driving safety has become the focus of attention. How to ensure the driving safety of urban rail transit has become an important research topic. In ensuring the driving safety of urban rail transit, drivers have an important responsibility. Their accurate actions and sober consciousness often determine the safety of passenger transportation [3]. Driver fatigue is one of the most important causes of accidents. In 2014, a fatigue-related driving accident occurred on the Chicago Metro: a subway derailed because the driver was dozing. According to a Federal Railway Administration survey, the proportion of accidents caused by fatigue driving is 30%–40% [4]. There are many factors that lead to driver fatigue, including work-related factors and non-work-related factors [5]. A minimum driver configuration, monotonous driving action [6], and alternate bright and dark driving environment [7] are the main reasons for driver fatigue.
Simultaneously, a driver's personal habits [8], work stress, and long working hours [9] also influence driver fatigue. A recognized effective approach to manage sleepiness is to stop driving and take a rest; however, the strict schedule of urban rail transit does not allow drivers to do this. Some traditional approaches are to alleviate the fatigue of train drivers by improving the management system and work plan. "Anti-sleeping-soundly" equipment in a train also alleviates driver fatigue, to a certain extent [10]. However, because of the long driving time, drivers may not be sensitive to anti-sleeping-soundly equipment, and this equipment cannot monitor the fatigue status of drivers in real time [11].

Using advanced technology and equipment to monitor drivers to reduce accidents and unsafe factors caused by fatigue driving has become an important mission for rail transit operation management and safety assurance.

The successful experience of Chinese railways has been used for reference and popularized in the operation and management of China's urban rail transit, and the "gesture and oral instructions of urban rail traffic driver" is operational actions of drivers written in the Chinese metro operation specification. These requirements are for the driver to make a second confirmation of each driving command, that is, pointing confirmation, oral confirmation, and watching and examine confirmation, to ensure that any driving operation is foolproof. The commands that need to be confirmed include route safety, correct signal, correct turnout, train speed, train status, and driving mode. This is part of the standardized operation of China's rail transit, and also an effective management measure to prevent fatigue driving by urban rail transit drivers. However, this very effective "gesture and oral instructions of urban rail traffic driver," which has standardized the operating behavior, is only applied to the on-site management of track operation, and is not included in the real-time fatigue monitoring system.

At the present time, real-time fatigue detection methods can be divided into those based on physiological signals [12]-[15], vehicle parameters or driving behavior [16]-[18], [41]-[43], and machine vision [48], [49]. Compared with the other two methods, the machine vision-based fatigue detection method has become the mainstream method of fatigue detection because of its non-contact, efficient, and accurate characteristics. This method mainly captures a video frame of the driver's face using a camera, and then segments the image of the human eyes and mouth from the video frame according to feature points. The driver's eye blinking frequency, mouth opening and closing frequency, and percentage of eyelid closure over the pupil over time algorithm [19] are analyzed to monitor the degree of driver fatigue. Jin et al. used human eye information to develop a driver fatigue detection system based on a support vector machine, with an accuracy rate of 85.41% [20]. Friedrichs et al. used human eye information to identify driver fatigue with an accuracy rate of 82.5% [21]. In practice, if the driver wears glasses or a hat with a brim, this affects the image segmentation and feature extraction of the eyes. If the driver wears a mask or sunglasses and other occlusive objects, the human eyes and mouth cannot be effectively located, and feature extraction cannot be performed. Advanced Safety Concepts has developed a head displacement sensor, which can accurately collect the head position and assess the driver's fatigue degree according to the change of position [22]. However, the development of the equipment is complex and the cost is too high for it to be used on a large scale. Zhang proposed a method of fatigue driving detection based on a convolutional neural network (CNN) using infrared video [23]. Although it solved the problem of eye condition extraction caused by the driver wearing glasses, the action of wearing glasses affected detection. Li proposed a fatigue driving detection method based on head posture features, which solved the problem of eye and mouth feature extraction, but it was still vulnerable to the influence of skin color disturbance, such as on the arms [24]. Xing et al. used the random forest method and maximum information coefficient to evaluate the importance of these features for driver behavior recognition. Then, the seven tasks were identified using a feedforward neural network (FFNN), which effectively identified the driver's actions. [44] Cao proposed a driver activity detection model based on a variety of CNNs, which effectively distinguished whether the driver was distracted, and the accuracy rate was 91.4% [45]. Ma proposed a driver fatigue detection system based on dual-stream CNN, which combined LSTM and CNN to recognize the information in a time series. [46]

Fatigue detection of face information and head information has many influencing factors. Researchers have attempted to introduce behavior detection into fatigue driving detection. There is relatively less research on action-based fatigue detection than face-based fatigue detection. Ma [47] et al. proposed a convolution-based driver fatigue detection method. The convolutional three-stream network model performed well because it used two-dimensional (2D) feature extraction and the final three-dimensional (3D) convolutional fusion method. Since the 3DCNN model was proposed by Ji [25], its superior ability to learn space-time characteristics [26] have been applied in many fields. In medicine, Payan et al. applied a 3DCNN to the prediction of Alzheimer's disease [27]. Ghafoorian et al. applied a 3DCNN to the automatic detection of vascular origin defects [28]. Additionally, Daniel and Maturana proposed a volume 3DCNN and applied it to LiDAR landing area detection [29]. Molchanov et al. proposed a vehicle driving gesture recognition model based on a 3DCNN, and classified gestures on the Viva challenge dataset. The accuracy rate of that model reached 77.4% [30]. Cao et al. proposed a 3DCNN model with a recursive spatiotemporal change module, which can effectively process self-centered motion [31].

The "gesture and oral instructions of urban rail traffic driver," which is standard operating behavior, is also an effective approach to prevent fatigue driving; however, there is a lack of effective detection. Simultaneously, urban rail transit is different from highway traffic, and the cab is more open. Drivers can operate in a wider range according to their
work needs. The large-scale operation of urban rail transit drivers may result in the inability of a fixed camera to capture the driver's actions, so the camera needs to be adjusted according to different scenarios, but current cameras lack intelligence. In this paper, a fatigue behavior detection framework for rail transit drivers is proposed, which combines the "gesture and oral instructions of urban rail traffic driver" with normal fatigue actions. The behavior dynamic tracking module adjusts the pan-tilt of the camera mainly through the PID algorithm to ensure that the driver's action is always in the center of the picture, and transmits the picture containing the action to the behavior detection module. The behavior detection module mainly uses the 3DCNN model of multi-channel information fusion with two inputs to discriminate the action. The input of the model is frames acquired at a certain interval (skipping frames) and frames acquired continuously (continuous frames). This model can recognize the action effectively and is not affected by the change of the picture. According to the action type identified by the model and the current state of the vehicle, the detection system synthetically determines whether the driver is fatigue driving. Compared with the traditional fatigue detection framework, the proposed monitoring framework not only increases the dynamic tracking of action recognition according to the characteristics of their driving actions. Considering the driver as a whole, we track the driver's work area, collect the driver's behavior, and identify the existence of fatigue driving by identifying human behavior. A 3DCNN model is chosen for human behavior recognition because it extracts information in time and space very well, and has high accuracy in terms of behavior recognition.

Because of the large working area of rail transit drivers, fixed cameras would fail to collect some behaviors. We choose a pan-tilt camera controlled by the Raspberry Pi and the control board, which has two degrees of freedom. We develop a behavior dynamic tracking program for identification and tracking, and program it in Raspberry Pi. This creates smarter, more portable pan-tilt camera equipment.

First, the position of the upper part of the frame is identified according to the Haar features [38] - [40], and the rotating camera keeps the person in the center of the image. Second, the face size is recognized according to Haar features. According to the recognized face size, the total picture size in the video frame is intercepted to maintain the proportion between the size of the face and the size of the entire picture. Finally, the picture is transmitted to the behavior recognition model.

As the core of the behavior recognition model, we redesign a dual-input 3DCNN model to identify the different behaviors of drivers during driving. One input is a picture captured by the skipping frame acquisition of the input video. The long time interval between frames and full use of the entire time sequence enables the model to obtain the overall behavior information of the characters in the video. Another input is the picture acquired by continuous frame acquisition of the input video, which enables the model to capture subtle changes in the behavior of the characters in the video. The actual environment of rail transit and optimizes fatigue detection. The system is integrated into the Raspberry Pi.

Notation: In this paper, \( f(\cdot) \) represents the rule extraction functions; \( f' \) represents the first derivative of a function; \( \Theta \) and \( (\cdot)^T \) represent the Kronecker product and transpose of the matrix, respectively; and \( \frac{\partial f(\cdot)}{\partial (\cdot)} \) denotes the partial derivative of \( f(\cdot) \) with respect to \( \cdot \). When describing a kernel, three numbers represent its length, width, and depth.

II. PROPOSED FRAMEWORK

In recent years, researchers have been studying how to reduce the effect of motion, glasses, clothing color, and other factors on fatigue driving detection based on machine vision, such as multi-information fusion assessment and an infrared camera. However, these studies focused on the detection of facial and head fatigue characteristics, and mainly focused on the driver. For mass transit, the effective real-time monitoring of its drivers is important. Unlike automobile drivers, the working environment of rail transit drivers is more open and the required operation is more frequent. We consider rail transit drivers as the research object, and design a system from dynamic tracking to the final state assessment of action recognition according to the characteristics of their driving actions. Considering the driver as a whole, we track the driver's work area, collect the driver's behavior, and identify the existence of fatigue driving by identifying human behavior. A 3DCNN model is chosen for human behavior recognition because it extracts information in time and space very well, and has high accuracy in terms of behavior recognition.

Because of the large working area of rail transit drivers, fixed cameras would fail to collect some behaviors. We choose a pan-tilt camera controlled by the Raspberry Pi and the control board, which has two degrees of freedom. We develop a behavior dynamic tracking program for identification and tracking, and program it in Raspberry Pi. This creates smarter, more portable pan-tilt camera equipment.

First, the position of the upper part of the frame is identified according to the Haar features [38] - [40], and the rotating camera keeps the person in the center of the image. Second, the face size is recognized according to Haar features. According to the recognized face size, the total picture size in the video frame is intercepted to maintain the proportion between the size of the face and the size of the entire picture. Finally, the picture is transmitted to the behavior recognition model.

As the core of the behavior recognition model, we redesign a dual-input 3DCNN model to identify the different behaviors of drivers during driving. One input is a picture captured by the skipping frame acquisition of the input video. Another input is the picture acquired by continuous frame acquisition of the input video, which enables the model to capture subtle changes in the behavior of the characters in the video.
two types of input information complement each other, and the model can fully learn the behavior information. We extract several pieces of information for each input, including gray level, gradient, and optical flow. Thus, the information in each frame is extracted, and the change information between frames is also extracted. The fusion of multi-information makes the model unaffected by the change of the picture when tracking, and it can effectively recognize the behavior in the video. First, the model is trained on an open behavior dataset to improve and modify the parameters according to the training results. Then, using the transfer learning method, the model is trained on the self-recorded dataset, and the driver behavior recognition model is obtained.

Finally, the behavior types identified by the model are transmitted to the decision program. In the assessment procedure, we access the state of the vehicle through the interface. When the driver's behavior is consistent with the current state of the vehicle, we determine that the driver has no problem. When the driver and the current state of the vehicle do not match or the driver does not exhibit fatigue behavior, we remind the driver of the existing problems. The overall structure is shown in Fig. 1.

In the following chapters, we describe our driver behavior tracking model and our driver behavior recognition model. To introduce the driver's action recognition model, we first explain the classical 3DCNN proposed by Ji in detail, and then introduce the driver's action recognition model based on our proposed 3DCNN. To introduce the driver fatigue behavior detection system based on real-time monitoring, we focus on the work performed in real-time acquisition.

**FIGURE 1.** Framework of the driver fatigue behavior detection system based on real-time monitoring.

**A. DRIVER BEHAVIOR DYNAMIC TRACKING MODEL**

To develop the driver behavior dynamic tracking model, we mainly focused on video acquisition for real-time monitoring. Additionally, we selected the AI vision platform suite of the AI Raspberry Group from Abbot Intelligent Technology Co., Ltd., including a two-degree-of-freedom camera platform, AI camera, Raspberry Pi, and steering control board, including PCA9685 as our video acquisition hardware. We defined the steering gear responsible for left-right movement in the platform as the X-axis steering gear, and the steering gear responsible for up-down movement as the Y-axis steering gear. We connected the X-axis rudder to S1 of the PCA9685 control board and the Y-axis rudder to S2 of the PCA9685 control board. Simultaneously, we plugged the steering control board into the 40pin GPIO pin of the Raspberry Pi. The camera was connected to the Raspberry Pi via USB. Our equipment was connected as shown in Fig. 2.
PID control algorithm. The values of $p_{wm_x}$ and $p_{wm_y}$, the error in the capture module of the system.

We determined the coordinates of the center point of the rectangular frame as $(x, y)$. At this time, the distance between the center of the character and the center of the picture was calculated. The difference was converted into a pulse value using the PID control algorithm, and the steering gear was controlled. Finally, according to the size of the character frame to capture the picture, so that the entire character frame to be intercepted in the original picture. $W = (64 \times w) / 13$ and $H = (64 \times h) / 13$ represent the width and height of the intercept box, respectively. The captured images were transmitted to the 3DCNN model for detection.

The above process effectively tracks the driver's behavior and ensures that the behavior areas of the characters are essentially consistent in the overall picture.

**B. DRIVER BEHAVIOR RECOGNITION MODEL**

1) 3DCNN MODEL ARCHITECTURE

A CNN is a type of FFNN with a deep structure and convolution calculation. It can learn features from the input data independently and does not need to conduct feature engineering. In CNNs, the idea of parameter sharing in hidden layers and the sparsity of connections between layers make the network less computationally expensive and easier to train [32], [33]. Since the emergence of Lenet [34], which was the first well-known deep neural network architecture for handwritten numeral classification, the CNN has made huge progress in solving many issues, such as image classification and image segmentation, in addition to scene marking and speech processing. However, it is limited to 2D input. The Lenet neural network can only process images; it cannot process video. Ji [25] proposed a new 3DCNN model in TPAMI 2013, which overcame the 2D limitation and made the application of the CNN more extensive. A 3DCNN can make the network less computationally expensive and easier to train [32], [33]. Since the emergence of Lenet [34], which was the first well-known deep neural network architecture for handwritten numeral classification, the CNN has made huge progress in solving many issues, such as image classification and image segmentation, in addition to scene marking and speech processing. However, it is limited to 2D input. The Lenet neural network can only process images; it cannot process video. Ji [25] proposed a new 3DCNN model in TPAMI 2013, which overcame the 2D limitation and made the application of the CNN more extensive. A 3DCNN can extract features from spatial and temporal dimensions, mainly by stacking multiple consecutive video frames to form an input video cube, and then convoluting in the input video cube using stereo convolutional kernels. In this structure, each feature map generated by convolution in the convolutional layer is connected to several adjacent continuous video frames in the previous layer.
FIGURE 3. Ji’s 3DCNN model.

Ji’s 3DCNN [25] includes an input layer, hardwire layer, three convolutional layers, two undersampling layers, and a fully connected layer, as shown in Fig. 3. The input layer of the model intercepts seven consecutive video frames and compresses the frame size of each video to $60 \times 40$ pixels. Seven consecutive video frames are processed by hardwire cores in the hardwire layer. The seven gray-scale images, seven $X$ and $Y$ directional gradient images, and six $X$ and $Y$ directional optical flow maps have 33 feature maps in five channels. Among them, the gray image reflects the overall information in the video frame, the gradient represents the edge distribution of the video frame, and the optical flow represents the trend of object movement. $C2$ and $C4$ are both 3D convolutional layers that use two $7 \times 7 \times 3$ 3D convolutional kernels and three $7 \times 6 \times 3$ 3D convolutional kernels, respectively. $S3$ and $S5$ are 2D maximum pooling layers, which are used for downsampling. $C6$ uses 2D convolutional layer to flatten the parameters, which is the same as the flatten layer. There are 128 feature maps in the $C6$ layer. Each feature map is fully connected with all 78 feature maps in the $S5$ layer so that each feature map size is $1 \times 1$ and the final feature vector is output. After multilayer convolution and downsampling, each of the seven consecutive frames of the input images is transformed into a 128-dimensional feature vector, which captures the motion information of the input video frames. The number of nodes in the output layer is the same as the number of types of behavior, and each node is fully connected to these 128 nodes in $C6$. In the output layer, a linear classifier is used to classify the 128-dimensional feature vectors to achieve behavior recognition and output the probability of each category.

FIGURE 4. The Design of the model, the name of each layer, and the setting of initial parameters for each layer. The input video information is divided into two kinds of input by hardwired kernel, one kind is skipping frame, the other

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/.
is continuous frame. In the yellow dotted frame above is the 3DCNN structure for processing skipping frame information, which includes three hidden layer groups. Each hidden layer group is composed of two convolution layers, one maxpooling layer and one dropout layer. In the yellow dotted frame below is the 3DCNN structure for processing continuous frame information, which includes three hidden layer groups. Each hidden layer group is composed of two convolution layers, one maxpooling layer and one dropout layer. The results of the upper and lower structures are merged by the Merge layer and then processed finally.

2) 3D CONVOLUTIONAL NEURAL NETWORK ARCHITECTURE BASED ON DRIVER BEHAVIOR

We redesign the dual-input multi-channel 3DCNN structure. First, we describe the design ideas and functions of each layer of the model, and then introduce back-propagation and weight updating. As shown in Fig. 4, the depth, structure, and parameters of the left and right convolution models are consistent. At the end of the last convolutional layer, the two convolutional models merge the data through the merge layer. Finally, the merged data are flattened and input into two successive fully connected layers.

We adopt two methods to extract video frames, long-jump frame extraction and continuous frame extraction, and extract video frames as two inputs of the model. We define the mode of frame skipping as: \( n_{frame} \in (0,1, ..., a) \), where \( n_{frame} \) denotes the total number of frames in the video and \( a \) denotes the depth of the model input. Using the method of long-jump frame fetching increases the interval between frames so that the trend of inter-frame action is more obvious. This method makes full use of the length of the video so that the model can fully learn the entire action process. Some details of the action can be obtained using the method of continuous frame selection. The combination of these two inputs can effectively learn more complete action information and reduce the effect of zoom on action information in video.

The captured video frames are input into two hardwire layers H1 and H2. The hardwire core of the two hardwire layers is the same. The hardwire core still extracts five channels of information: gray level, \( X \) and \( Y \) direction gradient, and \( X \) and \( Y \) direction optical flow. However, we extract 41 frames from the input video. Using the 41 frames, an optical flow graph is extracted from each pair of frames. The total number of \( 40 \times 40 \) \( X \)-direction optical flow maps and \( 40 \times 40 \) \( Y \)-direction optical flow maps are extracted. Simultaneously, the gray image, \( X \)-direction gradient image, and \( Y \)-direction gradient image are extracted from the first 40 frames. Each channel contains 40 images. In the hardwire layer, we use the first-order Sobel operator edge detection method to extract the gradient channel information in the \( X \) and \( Y \) directions of the video frames. We use the Gunner Farneback’s optical flow function to extract the dense optical flow matrix of the front and back frames [35], and then divide it into \( X \) and \( Y \) direction optical flow channel information. The final input video generates a matrix of size \( (40, 40, 40, 5) \), where the three 40s represent the frame height, frame width, and depth, and 5 represents the five channels, as shown in Fig. 5.

The captured video frames are input into two hardwire layers H1 and H2. The hardwire core of the two hardwire layers is the same. The hardwire core still extracts five channels of information: gray level, \( X \) and \( Y \) direction gradient, and \( X \) and \( Y \) direction optical flow. However, we extract 41 frames from the input video. Using the 41 frames, an optical flow graph is extracted from each pair of frames. The total number of \( 40 \times 40 \) \( X \)-direction optical flow maps and \( 40 \times 40 \) \( Y \)-direction optical flow maps are extracted. Simultaneously, the gray image, \( X \)-direction gradient image, and \( Y \)-direction gradient image are extracted from the first 40 frames. Each channel contains 40 images. In the hardwire layer, we use the first-order Sobel operator edge detection method to extract the gradient channel information in the \( X \) and \( Y \) directions of the video frames. We use the Gunner Farneback’s optical flow function to extract the dense optical flow matrix of the front and back frames [35], and then divide it into \( X \) and \( Y \) direction optical flow channel information. The final input video generates a matrix of size \( (40, 40, 40, 5) \), where the three 40s represent the frame height, frame width, and depth, and 5 represents the five channels, as shown in Fig. 5.

The captured video frames are input into two hardwire layers H1 and H2. The hardwire core of the two hardwire layers is the same. The hardwire core still extracts five channels of information: gray level, \( X \) and \( Y \) direction gradient, and \( X \) and \( Y \) direction optical flow. However, we extract 41 frames from the input video. Using the 41 frames, an optical flow graph is extracted from each pair of frames. The total number of \( 40 \times 40 \) \( X \)-direction optical flow maps and \( 40 \times 40 \) \( Y \)-direction optical flow maps are extracted. Simultaneously, the gray image, \( X \)-direction gradient image, and \( Y \)-direction gradient image are extracted from the first 40 frames. Each channel contains 40 images. In the hardwire layer, we use the first-order Sobel operator edge detection method to extract the gradient channel information in the \( X \) and \( Y \) directions of the video frames. We use the Gunner Farneback’s optical flow function to extract the dense optical flow matrix of the front and back frames [35], and then divide it into \( X \) and \( Y \) direction optical flow channel information. The final input video generates a matrix of size \( (40, 40, 40, 5) \), where the three 40s represent the frame height, frame width, and depth, and 5 represents the five channels, as shown in Fig. 5.

In the model, two convolutional layers are used continuously to increase the effective receptive field because two convolutional layers have fewer parameters than one convolutional layer with the same effective receptive field. For the two \( 3 \times 3 \times 3 \) convolutional kernels, the receptive field is \( 2 \times (3-1) \times (3-1) \times (3-1) \), which is the same as that for a \( 5 \times 5 \times 5 \) convolutional kernel, but the total number of parameters used for two \( 3 \times 3 \times 3 \) convolutional kernels is \( 2 \times (3 \times 3 \times 3) = 27 \) channels, which is only 43.2% of the total number of parameters used for the \( 5 \times 5 \times 5 \) convolutional kernel. The continuous use of two convolutional layers also increases the nonlinear transformation operation, which further improves the generalization ability of the model.

All the convolutional layers use 3D convolutional kernels. The 3D convolution equations are as follows:

\[
Y_{ij}^{X,Y,Z} = f \left( \sum_{m=0}^{Q-1} \sum_{p=0}^{P-1} \sum_{r=0}^{R-1} W_{ijm}^{(x+p)(y+q)(z+r)} \right),
\]

where \( x, y, \) and \( z \) denote the values of position \((x, y, z)\) of the \( z \)-th image in the video block; \( P, Q, \) and \( R \) denote the height,
width, and number of pictures in the video block; \( v_{ij} \) denotes the output of position \((x, y, z)\) of the \(j\)-th feature block in layer \(i\) after the neuron operation; and \( w_{ipqr} \) denotes the weight of position \((p, q, r)\) in the \(m\)-th image of the \(j\)-th feature block in layer \(i\). Each convolutional layer contains the ReLU excitation function. The ReLU excitation function is a frequently-used rectified linear unit, where ReLU \((v) = \max(0, v)\). When input \(v < 0\), the output is 0, and when \(v > 0\), the output is \(v\). The ReLU activation function is more efficient for gradient descent and back-propagation, so that the neural network can avoid the problems of gradient explosion and vanishing gradient. The function structure of ReLU which reduces the entire calculation cost of neural network is relatively simple. In the model, with the exception of the \text{conv3d}_6 layer and \text{conv3d}_12 layer convolutional kernel sizes of \((2,2,2)\) and the other convolutional kernel sizes of \((3,3,3)\), all convolutional kernel step sizes are \((1,1,1)\).

The pooling layer is max-pooling 3D with a size of \((2,2,2)\) and the formula is

\[
v_{xyz} = \max_{0 \leq x < X, 0 \leq y < Y, 0 \leq z < Z} \{ v_{x+i,y+j,z+k} \},
\]

where \(u\) is the 3D input vector of the pooling layer; \(V\) is the output of the pooling layer; and \(s\), \(t\), and \(r\) are the sampling steps in three directions. After sampling, the size of the feature graph decreases and the amount of computation decreases greatly. Simultaneously, the network becomes more robust to changes in the time and space domains.

Dropout refers to the temporary discarding of some neural network units from the network according to a certain probability in the process of model training, which can effectively reduce the amount of calculation, prevent model overfitting, and increase the generalization ability of the model. In the model, 25% of \text{dropout}_1, \text{dropout}_2, \text{dropout}_3, \text{dropout}_4, \text{dropout}_5, and \text{dropout}_6 layers are discarded, and 50% of \text{dropout}_7 layers are discarded.

After \text{dropout}_3 and \text{dropout}_6, the output data of the two convolution models are merged through the merge layer. The merge layer takes the final number of filters of two convolution models are merged through the merge layer.

The merge layer takes the final number of filters of two convolution models are merged through the merge layer.

The number of neurons in the first fully connected layer is \(256\), and the excitation function is the sigmoid function. The number of neurons in the second fully connected layer is \(256\), and the output category is connected with \(256\) neurons in the upper layer.

Finally, classification is performed by the softmax classifier. Softmax regression is the development of logistic regression in multiple classification problems. In a CNN, the softmax classifier is used to assess the probability of sample types and select the corresponding category of neurons with the largest output value as the classification result.

Next, we describe the back-propagation of the model. We use the cross-entropy loss function set in the Keras framework as the target function of the model and the Adam function as the gradient descent function. The loss function of the output layer is formulated as \(\text{loss} = -\sum_{i=1}^{n} y_i \log P_i\), where \(y_i\) denotes the real category of the video, \(P_i\) denotes the predicted probability value, and \(N\) denotes the number of classes, and back propagation is expressed as \(\frac{\partial \text{loss}}{\partial a_k} = \sum_{i=1}^{N} y_i \frac{\partial P_i}{\partial a_k}\). The backward propagation of the fully connected layer is \(\frac{\partial \text{loss}}{\partial a} = \omega \frac{\partial \text{loss}}{\partial v} \odot f'(\omega a)\), where \(v\) denotes the output value, \(a\) denotes the input value, \(\omega\) denotes the weight matrix, \(\frac{\partial \text{loss}}{\partial a} \odot f'(\omega a)\) denotes the error transferred to the upper layer, and \(\frac{\partial \text{loss}}{\partial v}\) denotes the error transferred to the lower layer, and the updating formula of the weight is expressed as \(\omega_{ij} = \omega_{ij} - lr \times \frac{\partial \text{loss}}{\partial v_{ij}} \times a_{j1} \times f'\left(\sum_{l=1}^{N} o_{lq} a_{lq}\right)\), where \(lr\) denotes the learning rate, \(T\) denotes the vector size of the output, and \(N\) denotes the vector size of the input. The reverse propagation of the pooling layer is formulated as \(\frac{\partial \text{loss}}{\partial a} = \text{upsample} \left(\frac{\partial \text{loss}}{\partial v}\right) \odot f'(\omega a)\), where the upsampling function represents upsampling. The upsampling denotes that the maximum pooling value is put back to the original position and positions without value are filled with \(0\). The back-propagation of the convolutional layer is expressed as \(\frac{\partial \text{loss}}{\partial a} = \frac{\partial \text{loss}}{\partial v} \odot \text{rot180}(w) \odot f'(\omega a)\), where the weight renewal formula is \(\omega = \omega - lr \times \frac{\partial \text{loss}}{\partial v} \times \text{rot360}(a) \times f'(\omega a)\), and \text{rot360} is about, up and down, and around and around.

III. EXPERIMENTS AND DISCUSSION

A. EXPERIMENTS 1

1) DATASET PREPARATION

We selected an open dataset, that is, KTH, as our first experimental dataset. The current video database contains six types of human actions (walking, jogging, running, boxing, hand waving, and hand clapping) performed several times by 25 subjects in four scenarios: outdoors s1, outdoors with scale variation s2, outdoors with different clothes s3, and indoors s4. The KTH dataset contains 599 videos. We chose this dataset because the videos in the dataset include scene changes and camera focus changes, which are similar to an actual rail transit scene.

2) TRAINING

In the training process, we used the train_test_split method in scikit-learn to cross-validate. The training set and validation set were randomly divided into 8:2 scaled datasets, and the random seed set was 3. The sample was divided into a training set that contained 479 videos and a verification set that contained 120 videos. Simultaneously, we added...
early_stopping to detect the loss in the verification set for 50 rounds and stop the iteration when the loss increased after 50 rounds. Early_stopping is a method for truncating iterations to prevent overfitting, that is, it stops the iteration before the model converges iteratively to the training dataset to prevent overfitting.

Following the model structure in Fig. 4, we used the Keras framework to implement the training process. The following describes the training process and results of models for different parameters. After each training instance, we drew the relationship between the epoch and loss, and also drew graphs with the epoch on the X-axis and accuracy on the Y-axis.

![Model Loss](image1)

![Model Accuracy](image2)

**FIGURE 6.** Comparison of the accuracy and loss in the training results.

Fig. 6a shows that the loss function of the model converged to approximately 0.8, whereas Fig. 6b shows that the accuracy of the model was approximately 55%. The failure of the model was caused by the large number of hidden layers in each convolution model, which resulted in gradient diffusion or gradient explosion of the model. The reason that there were too many hidden layers was the requirement to process 40 deep input data. We added a batch normalization layer after each convolutional layer, and a batch normalization layer after two fully connected layers because the batch normalization layer normalizes the output data of the upper layer so that the mean and variance of the data are fixed, which is conducive to a more uniform data distribution. The batch normalization layer not only solved the problem of poor training results, such as gradient dispersion and gradient explosion, but also made the model more robust, enhanced the generalization ability, and accelerated the training speed. As shown in Fig. 7, the accuracy function of the model converged and the loss function performed well. However, we aimed to further improve the accuracy of the model and reduce the loss value.

![Model Loss](image3)

![Model Accuracy](image4)

**FIGURE 7.** Comparison of the accuracy and loss in the training results.
Next, we modified the parameters of the first fully connected layer in the model and increased the number of neurons from 256 to 1,024. Figs. 7a, b, and c show the training results of 256 neurons, 512 neurons, and 1,024 neurons, respectively. Figs. 8a and b show that the number of neurons increased from 256 to 512, which improved the loss and accuracy of the model. Figs. 7b and c show that the accuracy of the model that resulted from increasing the number of neurons improved again, and the loss function did not perform well. We chose 1,024 neurons because the accuracy was higher and the loss was essentially the same compared with 256 and 512 neurons.

FIGURE 8. Comparison of experimental results for a varying number of neurons in the dense layer.

FIGURE 9. Comparison of the experimental results for a varying number of convolutional kernels in the convolutional layer.
Fig. 9a reflects the results of model training, in which the number of convolution kernels in the first group was 4, the number of convolution kernels in the second group was 8, and the number of convolution kernels in the third group was 16. The number of these three sets of convolution kernels is defined as the number of original convolution kernels. Next, we doubled the number of convolution kernels in the three groups of convolution layers, and increased the number of convolution kernels from the original convolution kernels to 32, 64, and 128 in the first, second, and third groups, respectively. Fig. 9 shows that the loss and accuracy of the model improved very well. Fig. 9d shows that the loss function of the model verification set was 0.3 and precision was 91.67%. Compared with the original parameter training results, that is, the results in Fig. 9a, accuracy increased by 3% and loss decreased by 0.04.

We found that the accuracy of the model was 91.67%, as shown in Figs. 9c and d. However, the convergence line of the accuracy curve in Fig. 9d was better than that in Fig. 9c, and the convergence speed was also better than that in Fig. 9c. The loss of the model in Fig. 9d was 0.03 higher than that in Fig. 9c. We attempted to make the convolution kernels of the first, second, and third groups 64, 128, and 256, respectively, to improve the results of the model. This promotion could not be conducted during training because the parameters exceeded the display memory. Hence, we reduced the number of convolution kernels in each group to half of the original so that the number of convolution kernels in the first, second, and third groups were 48, 96, and 192, respectively. Finally, the loss function of the model verification set was 0.2464 and the accuracy was 92.5%, as shown in Fig. 10.

Fig. 11 shows our final model structure and parameters.
3) COMPARISON

We compared the dual-input model with the single-convolution model for continuous frames and the single-convolution model for skipping frames. The three models used the same training samples. Table I shows that the dual-input model was better than the single model. The accuracy of the dual-input model was more than 20% higher than the skip frame input model, and the loss function was 0.6 smaller. The accuracy of the dual-input model was similar to the continuous frame input model, and the loss function was less than 0.01.

<table>
<thead>
<tr>
<th>COMPARISON</th>
<th>accuracy</th>
<th>loss</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dual-input model</td>
<td>92.5%</td>
<td>0.2464</td>
</tr>
<tr>
<td>Skipping frame input model</td>
<td>74.16%</td>
<td>0.8644</td>
</tr>
<tr>
<td>Continuous frame input model</td>
<td>91.67%</td>
<td>0.2721</td>
</tr>
</tbody>
</table>

B. EXPERIMENTS 2

1) DATASET PREPARATION

According to the driver's driving action standard for rail transit, three driving actions were selected: start-up action, stop action, and confirmation signal action. The start-up action involves extending the fingers of the right-hand and putting them close to the face, and extending the forearm forward from 90 degrees to 180 degrees. The stop action involves placing the fingers of the right hand on the face, with the arm swinging slightly greater and less than 90 degrees. The confirmation signal action involves pointing two fingers together at the front of the face to incline upward 45 degrees until the arm is straight. Simultaneously, we selected two fatigue movements: rubbing the eyes and yawning. All actions in the database are shown in Fig. 12.
We recruited 19 participants to record these five movements, each performing five actions as a group, for a total of two groups. A tripod was used to fix the camera position and the recording background was a fixed scene. In post-production of the dataset, we modified the video length and video frame size so that each video length was approximately 5 seconds and the frame size was 1000 × 1000. A total of 630 videos were generated.

2) TRAINING

We trained the model according to the model structure and model parameters in Table II, and continued to use the train_test_split method in scikit-learn for cross-validation. We used 80% of the dataset for training set and 20% for validation, and the random seeds were set to 3. After sample splitting, the training set contained 504 samples and the verification set contained 126 samples. The training results are shown in Fig. 13.

We used the transfer learning method to train our dataset directly, with the model trained on KTH. The accuracy of the model verification set reached 98.41% and the loss function value was 0.1007.

3) COMPARISON WITH OTHER MODELS

We compared the dual-input model with our 2DCNN model. For this 2DCNN model, like all our 3DCNN model structures, we replaced 3D convolutional kernel and 3D maxpooling kernel in the 3DCNN model with 2D versions. We also chose the classic Ji’s 3DCNN model for comparison. We reproduced Ji’s model with Keras, and the structure and parameters of the model remained unchanged. The three models used the same sample. Fig. 13 shows the results of the comparison. The accuracy of the dual-input model was 98.41%. The accuracy of the 2DCNN model and Ji’s 3DCNN model were 84.12% and 93.33%, respectively.

<table>
<thead>
<tr>
<th></th>
<th>accuracy</th>
<th>loss</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dual-input model</td>
<td>98.41%</td>
<td>0.1007</td>
</tr>
<tr>
<td>2DCNN model</td>
<td>84.12%</td>
<td>0.6090</td>
</tr>
<tr>
<td>Ji’s 3DCNN model</td>
<td>93.33%</td>
<td>0.3593</td>
</tr>
</tbody>
</table>
C. DISCUSSION

In training, when we used the traditional fit method, the computer ran out of memory. This is because the fit method stores all data into memory, and then batches them for model training. Because of the large amount of data, we used the fit generator method to generate batches of data for the training model through the Python generator. Compared with the fit method, the fit_generator method frees model training from the memory limitation and only stores data in memory for each batch. We implemented data extraction and splitting in the CPU and the calculate model in the GPU. This is because the GPU took 15 seconds to calculate each epoch, and the CPU took 593 seconds. Data extraction and splitting in the CPU did not occupy graphics card resources.

In practice, we implemented the trained model into Raspberry Pi to integrate the equipment and system. The entire set of equipment is portable and can be well integrated into the field environment. Dynamic behavior tracking effectively improved the video quality input into the recognition model, and improved the recognition ability of the model. The higher the quality of video input into the recognition model, the better the recognition accuracy of the model. After these improvements, the model effectively improved the accuracy of behavior recognition in the real environment. The system is not only applicable to urban rail transit, but also high-speed rail, train, tram, and other public transport modes with a large passenger flow. In an actual scenario, new categories will be collected, and new categories can be added using the of transfer learning method.

IV. CONCLUSION AND FUTURE WORK

According to the actual scenario of rail transit, in this paper, we established a multi-module fatigue detection system that includes a dynamic tracking module, action recognition module, and decision module. For the first time, the rail transit "gesture and oral instructions of urban rail traffic driver" operation and fatigue behavior were combined as the criteria to determine whether the driver is tired. The dynamic behavior tracking module solved the problem of an open environment and large operation area for rail transit drivers, and captured the required picture in real time. The behavior recognition module was based on a new dual-input 3DCNN model, which can effectively recognize the driver's action. The new dual-input 3DCNN model combined the skipping frame convolution model and the continuous frame convolution model, which learned all the information and the minute information of the action very well. The model not only improved the accuracy of behavior recognition but also was not affected by the change of the picture. For each input 3D convolution model, the depth of the input and depth of the model increased. Simultaneously, the batch normalization layer was added after the convolution layer, and the accuracy of each model improved. In particular, the batch normalization layer not only solved the problem of the poor training effect, such as gradient discretization and gradient explosion, but also made the model more robust, enhanced the generalization ability, and accelerated the training speed. Compared with the original fatigue detection method, this method not only identified whether the driver was tired but also effectively identified whether the driver was driving according to the norm. The system was integrated into the Raspberry Pi. The entire system was more portable and intelligent.

The current functions of our system can solve the problem of fatigue detection for rail transit drivers, but some functions can be added in future research. This will make our system more functional. The new functions are as follows:

1. The speech recognition function can be added in the system to improve the comprehensive performance of the system. The "Rules for Drivers of Rail Transit" state that drivers should also shout out the command content when conducting driving command actions. Therefore, the driver's voice information is also very important to detect the driver's understanding of instructions. We can further improve the driver detection model. First, an effective voice detection model can be established to identify the driver's command call in a noisy environment. Next, we can combine the sound detection model with the behavior detection model to achieve a comprehensive driver detection model. Finally, we propose a comprehensive criterion, including sound and action, to determine whether the driver understands the instructions clearly. Real-time and comprehensive detection of urban rail transit driver's driving status is achieved.

2. Establishing the generative countermeasure network can improve the accuracy of the model. The generative antagonism network is composed of a generator and discriminator, which improve accuracy by antagonizing each other. Our recognition model is a better discriminator. Next, we need to build a generator that can generate continuous meaningful video.

3. The zero shot learning method can be applied to the model to improve the learning ability of the new type of model. Existing machine learning models can only identify the type given by the training set, and cannot identify the new type. We propose zero shot learning to recognize new classes using machine learning, and it can also solve the problem of retraining the model to obtain new samples. Zero shot learning maps the semantic space to the image space by learning the semantics of training samples. Thereafter, even if a new category is encountered, the model can identify the category as long as the semantic knowledge of the category is provided. There will be many drivers' actions, so it is impossible to collect and mark all of them. Therefore, we hope to apply the zero shot learning method to our detection, which can form a more effective driver detection model.

REFERENCES

[3] A. J. Filtness, and A. Naweed, "Causes, consequences and


