Received May 25, 2020, accepted June 3, 2020, date of publication June 10, 2020, date of current version June 23, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3001159

HCF: A Hybrid CNN Framework for Behavior Detection of Distracted Drivers

CHEN HUANG[1,2], XIAOCHEN WANG[1], JIANNONG CAO[3], (Fellow, IEEE), SHIHUI WANG[1,2], AND YAN ZHANG[1,2]

[1]School of Computer Science and Information Engineering, Hubei University, Wuhan 430062, China
[2]Hubei Engineering Research Center of Educational Informationalization, Wuhan 430062, China
[3]Department of Computing, The Hong Kong Polytechnic University, Hong Kong

Corresponding author: Chen Huang (huang@hubu.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61977021, in part by the Natural Science Foundation of Hubei Province under Grant 2018CFB692, in part by the Science and Technology Innovation Program of Hubei Province under Grant 2018ACA13, and in part by the Science and Technology Innovation Major Program of Hubei Province under Grant 2019ACA144.

ABSTRACT Distracted driving causes a large number of traffic accident fatalities and is becoming an increasingly important issue in recent research on traffic safety. Gesture patterns are less distinguishable in vehicles due to in-vehicle physical constraints and body occlusions from the drivers. However, by capitalizing on modern camera technology, convolutional neural network (CNN) can be used for visual analysis. In this paper, we present a hybrid CNN framework (HCF) to detect the behaviors of distracted drivers by using deep learning to process image features. To improve the accuracy of the driving activity detection system, we first apply a cooperative pretrained model that combines ResNet50, Inception V3 and Xception to extract driver behavior features based on transfer learning. Second, because the features extracted by pretrained models are independent, we concatenate the extracted features to obtain comprehensive information. Finally, we train the fully connected layers of the HCF to filter out anomalies and hand movements associated with non-distracted driving. We apply an improved dropout algorithm to prevent the proposed HCF from overfitting to the training data. During the evaluation, we apply the class activation mapping (CAM) technique to highlight the feature area involving ten tested classes of typical distracted driving behaviors. The experimental results show that the proposed HCF achieves the classification accuracy of 96.74% when detecting distracted driving behaviors, demonstrating that it can potentially help drivers maintain safe driving habits.

INDEX TERMS Distracted drivers, convolutional neural network, transfer learning, fusion model.

I. INTRODUCTION

According to the report from World Health Organization (WHO) [1] in 2020, approximately 1.35 million people worldwide have died from traffic accidents each year. Losses from road traffic collisions are equal to approximately 3% of the GDP in most countries.

Complex and dynamic road traffic systems consist of four elements: people, cars, roads, and the environment. Traffic accidents result from the uncoordinated effects of these four elements [2]. Research shows that, more than 90% of traffic accidents are attributable to driver error [3], including the dominating factors such as fatigue, distraction, and drunkenness. Among these, distracted driving has become an increasingly important cause of traffic accidents in recent years. For example, distracted driving can cause wrong lane changes, which will lead to serious traffic accidents [4], [5]. According to the preliminary definition of the International Organization for Standardization (ISO) [6], distracted driving is ‘‘attention given to a non-driving related activity, typically to the detriment of driving performance’’ The National Highway Traffic Safety Administration (NHTSA) defines distracted driving as ‘‘any activity that diverts attention from driving, including talking or texting on the phone, eating and drinking, etc.’’ [7]. The current popularity of on-board electronics such as navigation systems and smart phones has introduced additional factors that induce distracted driving behavior by drivers. Therefore, it is necessary to carry out in-depth research on distracted driving behaviors, determine their occurrence mechanisms, and propose corresponding

The associate editor coordinating the review of this manuscript and approving it for publication was Lorenzo Ciani.

solutions. By doing so, we can reduce the frequency of distracted driving behaviors and improve driving safety.

To detect distracted driver behaviors, several approaches have been proposed based on physiological parameters, vehicle driving state and vision [8]. There are two requirements for a driving behavior detection system. **First**, regardless of the detection approach adopted, drivers should not be affected by the measuring instruments. Although approaches based on physiological parameters can achieve high accuracy, such measurements require numerous instruments that may disturb the driver. **Second**, current detection methods for driving behaviors consistently lag occurrences. We need to warn the drivers just at the point when they start to get distracted—not warn after their driving behavior has already become abnormal. Computer vision approaches meet the above two requirements. Based on the improvements in computing hardware, camera technology and neural networks, features can be fully extracted from complex images.

Convolutional neural networks (CNNs) are widely used for complex image processing tasks. In recent years, various methods, such as AlexNet [9], VGGNet [10], Inception [11], ResNet [12], and DenseNet [13], have been proposed to address problems such as image classification and recognition. These methods can extract features from images and use them to perform classification. However, recognizing distracted drivers' behavior using only one pretrained model is prone to overfitting, which causes detection failures in practice.

In this paper, to accurately detect distracted driving behaviors, we propose a hybrid CNN framework (HCF) consisting of three modules: a cooperative CNN module, a feature concatenation module, and a feature classification module.

First, in the cooperative CNN module, three pretrained models, ResNet50, Inception V3 and Xception, are combined to extract multiscale behavior features in parallel.

Second, in the feature concatenation module, the deep behavior features extracted by the cooperative CNN module are deeply fused into a set of one-dimensional vectors. These fused features contain high-level semantics and lay the foundation for the subsequent image classification.

Third, in the feature classification module, the neurons in the fully connected layer of the HCF capture the key elements of the fused feature vectors and use them as basis for classification. The fused features are classified into different distracted driving behaviors.

Our main contributions are summarized below.

(1) We propose a hybrid CNN framework (HCF) to detect distracted driving behavior. The HCF consists of a cooperative CNN module, a feature concatenation module and a feature classification module. To the best of our knowledge, this work is the first one to comprehensive study the driving behavior detection with a hybrid deep pretrained framework, which can better adapt the confined space in the vehicle without disturbing the drivers and make an accurate decision in a real-time manner.

(2) We design and implement the cooperative CNN module, which integrates three pretrained models, ResNet50, Inception V3 and Xception, to perform cooperative transfer learning. The novelty of cooperative transfer learning is to exact the detailed image features in parallel, which can prevent overfitting and significantly enhance the deep learning capability.

(3) We further propose an efficient momentum-based training rate optimization (MTRO) algorithm to solve the convergence problem with the weighted sum as the search direction, which can remarkable improve the CNN training speed and alleviate the tedious and time-consuming iteration process.

(4) The experimental results show that the proposed HCF outperforms current driving behavior detection approaches with regard to detection accuracy, average image processing time and robustness for ten types of typical distracted driving behaviors. The HCF can achieve high performance for a classification accuracy of 96.74%.

The remainder of this paper is organized as follows. In Section 2, we introduce prior related works. The detailed design of the HCF for the detection of distracted drivers is provided in Section 3. Section 4 describes the model evaluation and optimization. Our experimental results and analysis are reported in Section 5, and Section 6 presents conclusions and the outlook for future works.

II. RELATED WORKS

This section summarizes the academic research on distracted driving detection in recent years. Using cell phones is a main cause of driver distraction. Beck and Park [14] compared the impact of editing text messages on traditional mobile phones and smartphones on road traffic safety in a simulated driving environment. The results showed that smartphones increase driving risks as measured by key performance parameters. Esfahani *et al.* [15] used a driving simulator method to study the impact of reading and texting on driving performance. Studies have shown that such behaviors result in significant reductions in driving safety and should be prohibited. Hoang *et al.* [16] proposed an automatic system that determined the driver's actions through a dashboard-mounted camera. The observed features are input into a hidden conditional random field (HCRF) model. This method is able to effectively recognize mobile phone usage by drivers.

Tran *et al.* [17] used pretrained models to extract distracted drivers features and adopted a support vector machine (SVM) as a classifier. They compared the performances of CNNs for extracting the features input into the SVM to detect distracted driver behaviors. Chawan *et al.* [18] used VGG16, VGG19 and Inception models to classify distracted drivers. Baheti *et al.* [19] proposed a modified VGG model that used regularization techniques, their approach achieved a classification accuracy of 95.54%.

Yang *et al.* [20], [21] proposed a driving-related recognition system based on the deep CNN model. They used Kinect cameras to collect images of distracted drivers and the raw images are processed with a GMM-based segmentation

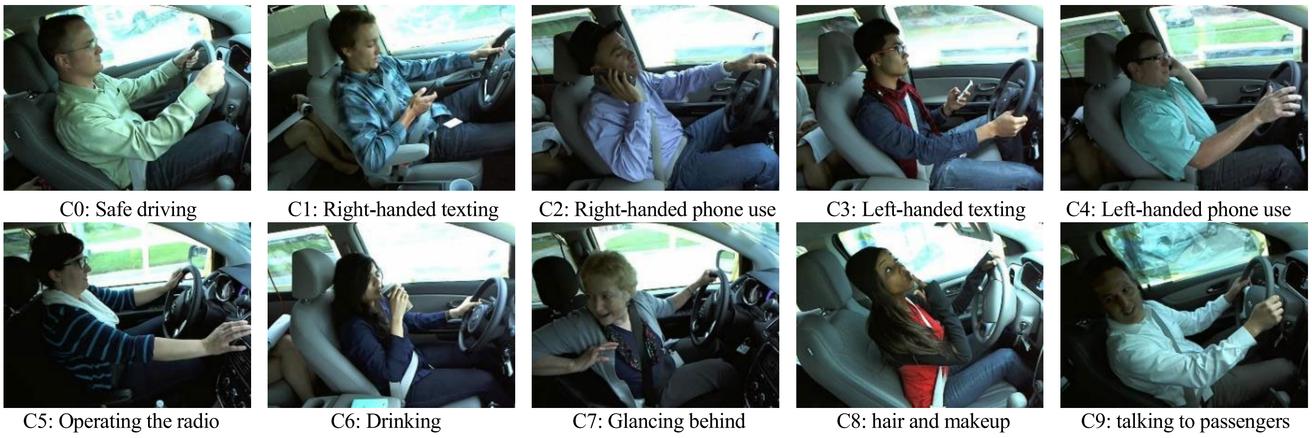


FIGURE 1. Ten common driver behaviors while driving.

algorithm. Then CNN models are used as a binary classifier which achieves 91% accuracy.

Yan [22] proposed a CNN-based model that used unsupervised learning and transfer learning to classify activities of normal driving, answering a cellphone call, eating, and smoking. Ramzan *et al.* [23] proposed using a hierarchical classification approach and treating driving behavior in spatio-temporal reference frame terms rather than as a static image. The overall prediction accuracy for this method reached 89.62%. Shahverdy *et al.* [24] proposed a deep learning-based method for acquiring distracted driving data and constructed an analysis system called DarNet, which achieved a classification accuracy of 87.02% on their collected dataset.

Kaggle organized a competition to classify driver behavior images [25] because early distracted driving datasets included only a few categories that covered common distracted driving behaviors. In contrast, the Kaggle dataset contains ten different distracted driver behaviors while driving (see Fig. 1). Majdi *et al.* [26] proposed Drive-net, a method that uses a combination of a CNN and a random decision forest to classify driver images. Driver-net achieved a detection accuracy of 95% on the Kaggle dataset. Ou and Karray [27] proposed a driver distraction recognition system that used generative adversarial networks (GANs) and demonstrated that generative models can generate images of drivers in different driving scenarios. By using these images to augment training, they improved the system's image classification performance by 11.45%.

Currently, computer vision techniques are widely used to extract key features from images. Such classification tasks are completed by various deep neutral network models running on high-performance computers to achieve better recognition accuracy.

III. RESEARCH METHODS

In this section, we introduce our proposed hybrid CNN framework (HCF), which includes three parts: a cooperative CNN module, a feature concatenation module and a feature classification module. Fig. 2 shows the architecture of the HCF.

First, in the cooperative CNN module, distracted behavior features are extracted in parallel, using the three integrated pretrained models: ResNet50, Inception V3 and Xception. These three models are trained on the ImageNet dataset and then transfer learning is applied for fine tuning [28].

Second, the feature concatenation module is used to deeply fuse the extracted features from the cooperative CNN module, generating the input for the feature classification module.

Third, the feature classification module is used to train the weights of the feature vectors. After training, a classification result can be obtained for each driving behavior.

A. COOPERATIVE CNN MODULE

The cooperative CNN module integrates three pretrained models that extract features from images of distracted drivers in parallel to enhance the deep learning capability. These pretrained models are ResNet50, Inception V3 and Xception.

Before inputting the training image to the cooperative pretrained models, we need to preprocess the original images ($640 \times 480 \times 3$) to satisfy the input requirements of ResNet50 ($224 \times 224 \times 3$), Inception V3 ($299 \times 299 \times 3$), and Xception ($299 \times 299 \times 3$). The steps used to preprocess the original images are as follows:

First, the original image is rescaled and flipped horizontally. For ResNet50, the rescaled image size is 320×240. For Inception V3 and Xception, the rescaled image size is 480×360.

Second, the original images and flipped images are expanded and then one expanded image is randomly cropped. For ResNet50, the cropped size is 224×224. For Inception V3 and Xception, the cropped size is 299×299.

Third, according to the requirements of the pretrained model, we freeze the specific layers in the CNN and start the training procedure.

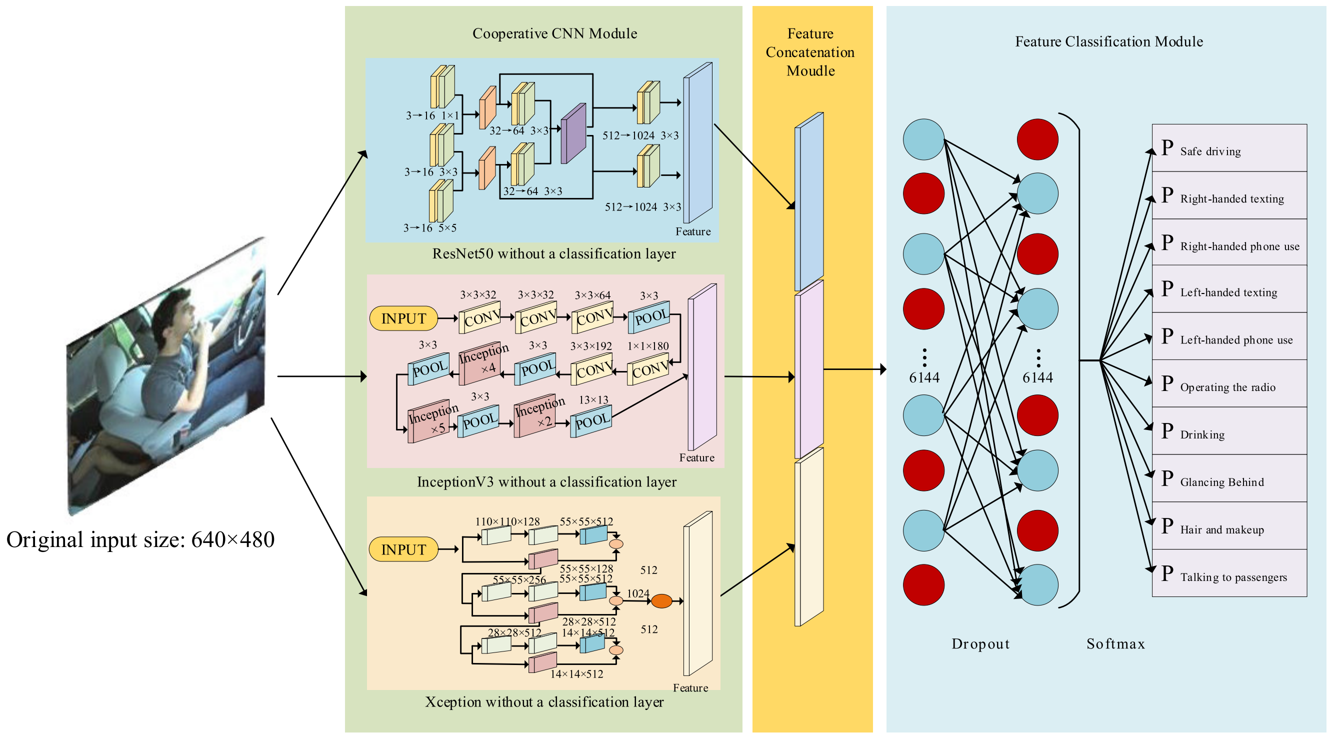


FIGURE 2. The architecture of hybrid CNN framework.

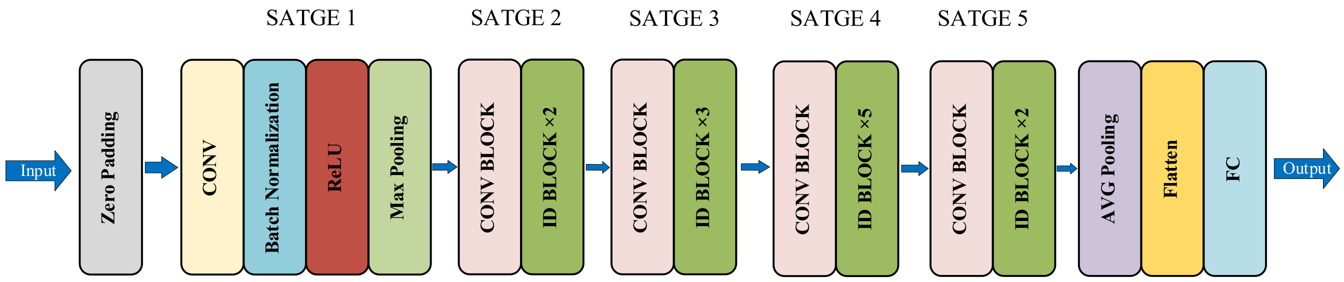


FIGURE 3. The architecture of ResNet50.

1) ResNet50 MODEL

The ResNet50 model effectively solves the problem of the increase in training errors caused by increasing the number of layers in a neural network. ResNet50 includes 5 main stages. Fig. 3 shows the architecture of the ResNet50 model.

The steps of the training procedure for ResNet50 are as follows:

First, in stage 1, a convolutional kernel of 7×7 is used to extract image features. Batch normalization, activation and max pooling are completed, and the output of stage 1 is a $55 \times 55 \times 64$ feature map in the CNN.

Second, from stages 2–5, two types of residual blocks are added to the CNN. A CONV block is a scaled residual block. Each CONV Block includes three convolutional kernels of 1×1 or 3×3. An ID block represents a residual block that does not change size. The sizes of the output feature maps from stages 2–5 are $55 \times 55 \times 256$, $28 \times 28 \times 512$, $14 \times 14 \times 1024$ and $7 \times 7 \times 2048$, respectively.

Finally, after the five stages, a global average pooling layer (AVG Pooling) with a kernel size of 7×7 is used to convert the output feature map into a 2,048-dimensional feature vector. This feature vector is input into the feature concatenation module.

2) INCEPTION V3 MODEL

Inception V3 clusters the sparse convolution kernel structure into multiple dense sub-convolution kernel combinations. Fig. 4 shows the architecture of Inception V3. The core units (the pink blocks in Fig. 4) of Inception V3 are shown in Fig. 5. Convolutional kernels of different sizes, (e.g., 1×1, 3×3 and 5×5) are used to obtain different receptive fields.

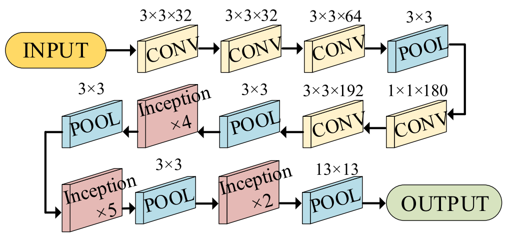


FIGURE 4. The architecture of Inception V3.

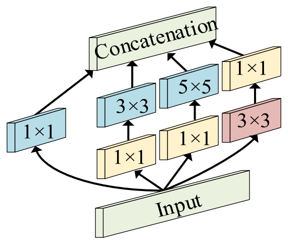


FIGURE 5. Core unit of Inception V3.

The training procedure steps for Inception V3 are as follows:

First, for the input image, 3 convolution layers with kernels of 3×3 and one max pooling layer are used to extract low latitude features of the image. Then two convolution layers with kernels of 1×1 and 3×3 and one max pooling layer are used to further extract features. A feature map of $35 \times 35 \times 192$ can be obtained.

Second, the 11 core units of Inception V3 and 3 pooling layers are used to extract high-dimensional features. A feature map of $8 \times 8 \times 2048$ can be obtained.

Finally, we obtain a 2,048-dimensional feature vector from the feature map. This feature vector will be input into the feature concatenation module.

3) XCEPTION MODEL

The Xception architecture includes 36 convolutional layers that form the feature extraction basis of the CNN. These 36 convolutional layers are arranged into 14 modules, all of which (except for the first and last modules) are surrounded by linear residual connections.

The training procedure steps for Xception are as follows:

First, the original images are input into the convolutional kernels of (3, 3, 64) and (3, 3, 128) to extract features. The calculation performed in the convolution layers is:

$$A_L = f(W_L{}^{*}A_{L-1} + b_L) \tag{1}$$

where $f(\cdot)$ is the activation function of the convolution layer, A_L represents the output of the L-th convolution layer W_L represents the convolutional kernel, the asterisk "$*$" denotes the convolution operation, and b_L represents the offset parameter.

Second, the feature maps are input into the depthwise separable convolution for further feature extraction. Depthwise separable convolution is used to reduce the number of parameters and reduce the calculation complexity [29].

Finally, a 2048-dimensional feature vector is obtained from the feature map. The feature vector will be input into the feature concatenation module.

B. FEATURE CONCATENATION MODULE

The feature concatenation module is used to deeply fuse the feature vectors output by the cooperative CNN module.

The features extracted by the three cooperative pre-trained models include different semantics. The three feature

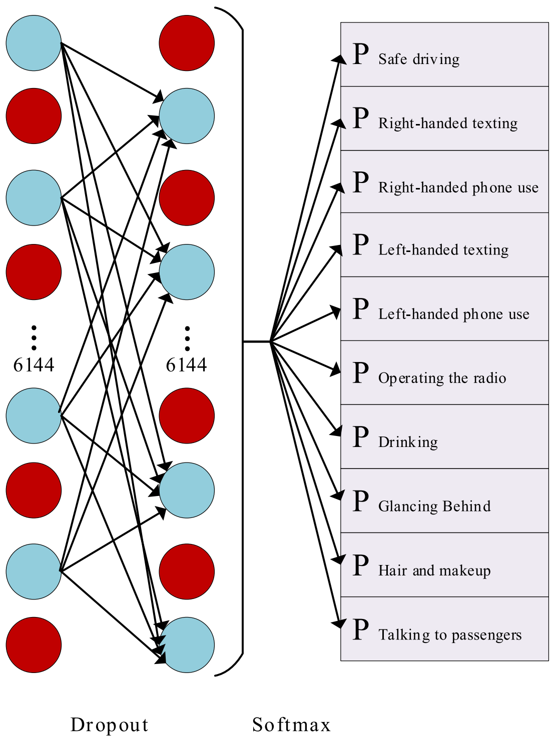


FIGURE 6. Feature classification module.

$1 \times 1 \times 2048$ vectors are merged to generate a vector of $3 \times 1 \times 2048$. Then, this multidimensional vector is flattened into a 1×6144 vector, which satisfies the input requirement for the feature classification module.

C. FEATURE CLASSIFICATION MODULE

The feature classification module is used to train the weights of the feature vectors. The feature classification module includes two fully connected layers, as shown in Fig. 6.

To reduce interdependent learning among the CNN neurons, we introduce dropout technology [30] to temporarily ignore some neurons according to a certain probability during the training procedure. We set the dropout rate to 0.5.

The feature classification steps are as follows:

First, each neuron at the first fully connected layer connects to the 6,144 nodes of the feature vector, and local information is integrated with the classification characteristics.

Second, dropout technology is adopted to reduce overfitting on the training data.

Third, in machine learning, the softmax classifier [31] is used to produce a one-hot vector consistent with the probabilities of the ten types of districted driving behaviors in the Kaggle dataset.

Finally, the classification results of the ten types of distracted driving behaviors are obtained.

IV. MODEL EVALUATION AND OPTIMIZATION

We use the loss function to evaluate the performance of the HCF during training. The loss function evaluates the error between the output of the HCF and the given target value. Meanwhile, to speed up HCF training, we propose a new momentum-based training rate optimization (MTRO) algorithm based on the Adam algorithm [11].

A. LOSS FUNCTION FOR PERFORMANCE EVALUATION

We generally evaluate the performance of deep learning CNN models by the numerical magnitude of the loss function.

To improve model generalizability, regularization terms are added to the loss function $\tilde{J}(\theta)$ as follows:

$$\tilde{J}(\theta) = J(\theta) + a\phi(\theta) \tag{2}$$

where θ is the measurable state. The difference between the real value and the output value is represented as $J(\theta)$, which is the standard loss function. $\phi(\theta)$ is the regularization term, $a \in [0, \infty]$ is the regularization term coefficient, and a is used to measure the weight of the regularization term in the loss function $\tilde{J}(\theta)$. The larger a is, the greater the weight of the regularization term is.

The commonly used loss functions are listed below:

1) THE MEAN SQUARED LOSS FUNCTION

$$J = \frac{1}{2}\left\|y - \hat{y}\right\|_2^2 \tag{3}$$

where y is the real value, $\hat{y}$ is the actual output value of the model and J is the mean squared loss function.

2) CROSS-ENTROPY LOSS FUNCTION

In binary classification, the number of classes M equals 2, and the cross-entropy loss function can be described as follows:

$$J = \frac{1}{N}\sum_i -[y_i \cdot \log(p_i) + (1 - y_i) \cdot \log(1 - p_i)] \tag{4}$$

where N is the number of samples, y_i is the real value of sample i, $\hat{y}_i$ is the output value of sample i, and p_i is the probability that sample i is positive.

When $M > 2$ (i.e. multiclass classification), we calculate a separate loss for each class label and sum the result:

$$J = \frac{1}{N}\sum_i -\sum_{c=1}^{M} y_{ic}\log(p_{ic}) \tag{5}$$

where M is the number of classes, and y_{ic} is the binary indicator (0 or 1). If class c is the correct classification for observation i, p_{ic} refers to the predicted probability observation i of class c.

3) EXPONENTIAL LOSS FUNCTION

The exponential loss function is

$$J = \frac{1}{N}\sum\nolimits_{i=1}^{n}\exp[-y_i\hat{y}_i] \tag{6}$$

where exp is the exponential function.

4) ABSOLUTE LOSS FUNCTION

$$J = \left|y - \hat{y}\right| \tag{7}$$

These functions have different metrics for the target deep learning method. When the partial derivative is small, the parameter updating rate of the mean squared loss function becomes slow. Then, we can choose the cross-entropy loss function to evaluate the performance of the proposed HCF.

B. OPTIMIZATION TO IMPROVE THE TRAINING SPEED

The objective function of a deep neural network is a complex, high-dimensional, nonconvex random function, and the corresponding optimal solution can be only continuously approximated through iterations. Hence, we use the optimizer to speed up the convergence of the deep neural network. We introduce momentum into the Adam algorithm [30] and propose a momentum-based training rate optimization (MTRO) algorithm.

In the MERO algorithm, we weight the sum of the search direction of the historical iteration point and that of the current iteration point and use the weighted sum as the search direction for the next iteration point.

The MTRO algorithm is a norm-based algorithm. Compared with the Adam algorithm, MTRO is more stable and has a faster convergence speed. The detailed process of MTRO is shown in Algorithm 1.

V. EXPERIMENTAL RESULTS AND ANALYSIS

A. THE EXPERIMENTAL ENVIRONMENT

1) DATASET DESCRIPTION

The dataset of distracted driving behaviors comes from the State Farm insurance company for a Kaggle challenge, which includes 22,424 images for training and 79,726 images for testing. Because the testing set images are unlabeled, we perform our experiments on the training set. The size of each image is 640 × 480. The dataset includes the images of 26 drivers, and the distracted driving behaviors in the dataset are divided into ten classes.

The ten classes of distracted driving behaviors include: safe driving, right-handed texting, right-handed phone use, left-handed texting, left-handed phone use, operating the radio, drinking, glancing behind, hair and makeup, and talking to passengers. We label these ten classes of distracted driving behaviors as c0–c9, the number of samples for each driver, and the number of distracted driving behavior samples in each class are shown in Figs. 7 and 8, respectively.

Algorithm 1 Momentum Based Training Rate Optimization (MTRO)

```
Input: initial learning rate
Input: exponential decay rate for moment estimation ($\rho_1, \rho_2 \in [0, 1]$)
Input: small constant for numerical stability ($\delta$)
Input: initial parameter ($\theta$)
Input: initial MTRO algorithm's change value ($P_0^{MTRO}$)
Input: initial Adam algorithm's iterative direction ($P_0^{Adam}$)
Output: parameters that can minimize the
        objective function
1:  Initialize 1st and 2nd moment variables $s = 0$, $r = 0$
2:  Initialize time step Initialize time step $t = 0$
3:  while (stop criterion not reached) do
4:        Sample a minibatch of m examples from the
          training set $\{x^{(1)}, \ldots, x^{(m)}\}$ with
          corresponding targets $y^{(i)}$;
5:        Calculate the gradient;
6:        $t \leftarrow t + 1$;
7:        Update biased first moment estimate;
8:        Update biased second moment estimate;
9:        Correct bias in first moment;
10:       Correct bias in second moment;
11:       Compute update;
12:       Apply update;
13: end while
```

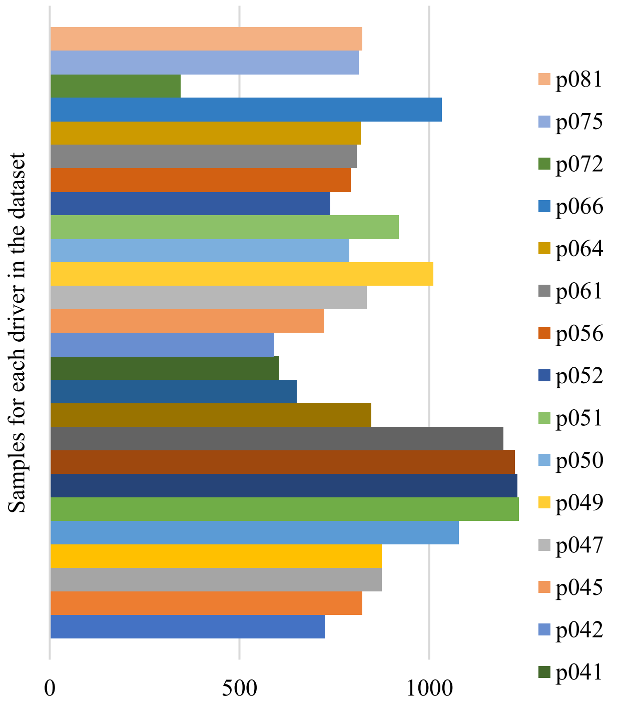


FIGURE 7. Number of samples associated with each driver in the dataset.

In the adopted dataset, the number of distracted driving behaviors samples for each class and the number of samples for each driver are roughly even, which is beneficial for our research. We can train the models with this dataset to achieve more accurate detection of distracted driving behaviors.

We selected images of some of the drivers (P014, P021, P026, and P049) as a verification set (total: 4,320 samples), and the other images are included in the training set (total: 18,104 samples). The ratio of images in the training set and the validation set was kept at approximately 8:2.

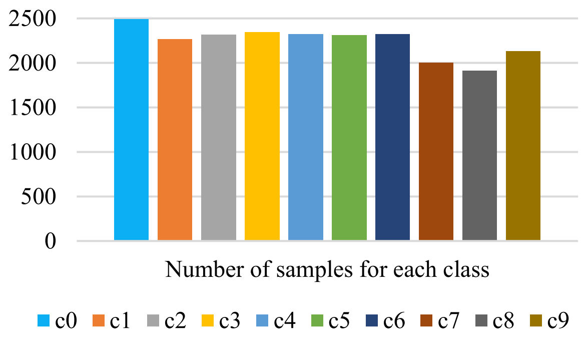


FIGURE 8. Number of samples of distracted driving behavior for each class in the dataset.

2) EXPERIMENTAL SETUP

The proposed HCF was tested on a workstation with a 3.2 GHz CPU, 32 GB of RAM, and a Tesla K80 GPU running the Ubuntu 18.04 operating system. The Linux kernel version was 5.1. The code was mainly implemented in the Python language, and we adopted the deep learning TensorFlow framework [32].

B. TRAINING STRATEGY

The parameters of the ResNet50, Inception V3 and Xception models were pretrained on ImageNet, and we transferred the parameters directly to our distracted behavior detection task. However, the features of the State Farm dataset are different from those of the ImageNet dataset. Thus, these pretrained models must be fine-tuned to adapt them to the State Farm dataset.

The strategy for training the proposed HCF is as follows:

(1) ResNet50: For the pretrained ResNet50, we froze the weights from layers 0 to 151 and trained only the weights from layer 152 to the top layer.

(2) Inception V3: We froze the weights of the pretrained Inception V3 layers 0 to 171 and trained only the weights from layer 172 to the top layer.

(3) Xception: We froze the weights of the pretrained Xception from layers 0 to 116 and trained the weights from layer 117 to the top layer.

During training, we used the MTRO algorithm and set the initial learning rate to 0.001. The image batch size was set to 64, and each model was trained for 60 epochs on the GPU. We applied dropout to reduce overfitting. A softmax classifier is used to complete the feature classification, and the final

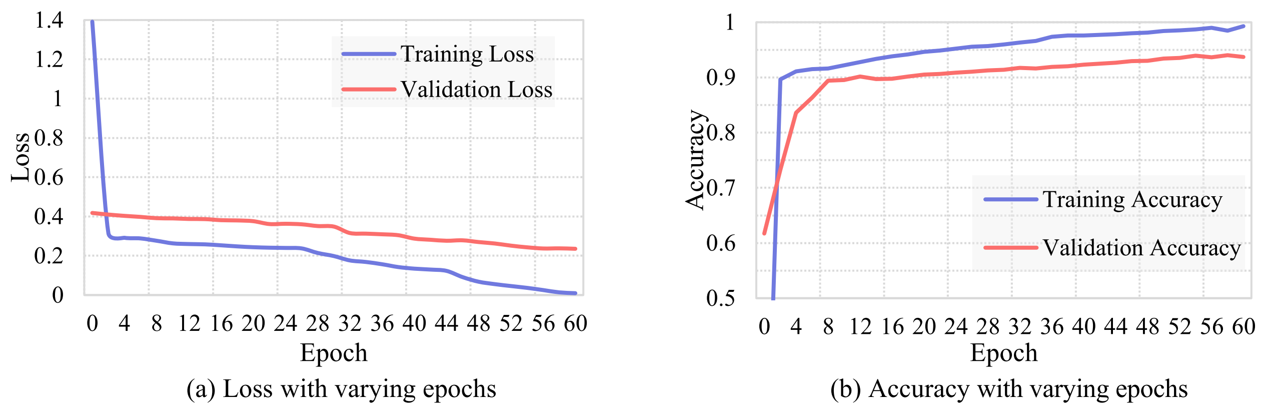


FIGURE 9. **The classification results using ResNet50.**

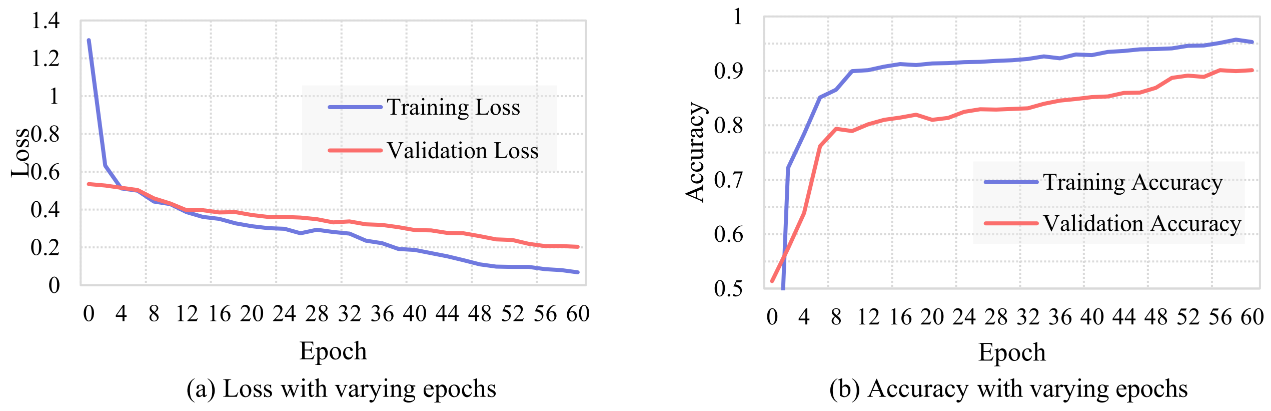


FIGURE 10. **The classification results using Inception V3.**

output is the probability corresponding to the ten classes of distracted driving behaviors in the State Farm dataset.

C. COMPARISON WITH SINGLE PRE-TRAINED MODELS

In the proposed HCF, we fine-tuned the three pretrained models to extract the distracted drivers' behavior features cooperatively. To demonstrate the improvement exhibited by the HCF, we compared the performance of a single pretrained model separately with that of the HCF.

1) THE LOSS AND ACCURACY

The experimental results of the single pretrained models are as follows. Under varying epochs, the training and validation set losses and accuracies of ResNet50, Inception V3 and Xception are presented in Figs. 9, 10, and 11, respectively.

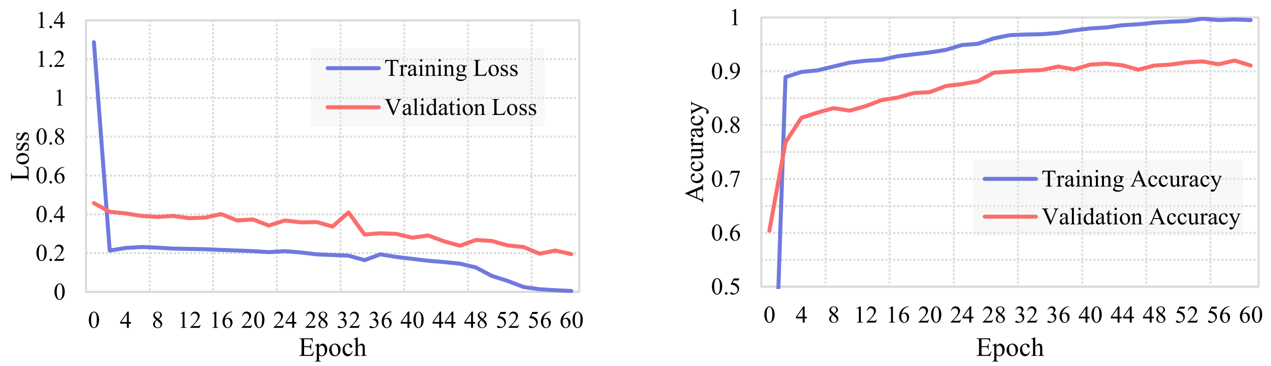
Among the three pretrained models, ResNet50 achieves the highest accuracy but also the largest loss. Inception V3 has the lowest accuracy and the second largest loss. Xception achieves the best loss, but its accuracy is lower than ResNet50.

For the task of distracted behavior detection, ResNet50 achieves an accuracy of 99.29% on the training set and 93.72% on the validation set. For Inception V3, these two accuracy rates are 95.31% and 90.13%, respectively, and for Xception, they are 99.53% and 91.08%, respectively.

Thus, the above three pretrained models are able to recognize distracted driving behaviors in the State Farm dataset. However, these pretrained models also all have large losses; consequently, they are incapable of identifying images of distracted drivers without this specific dataset.

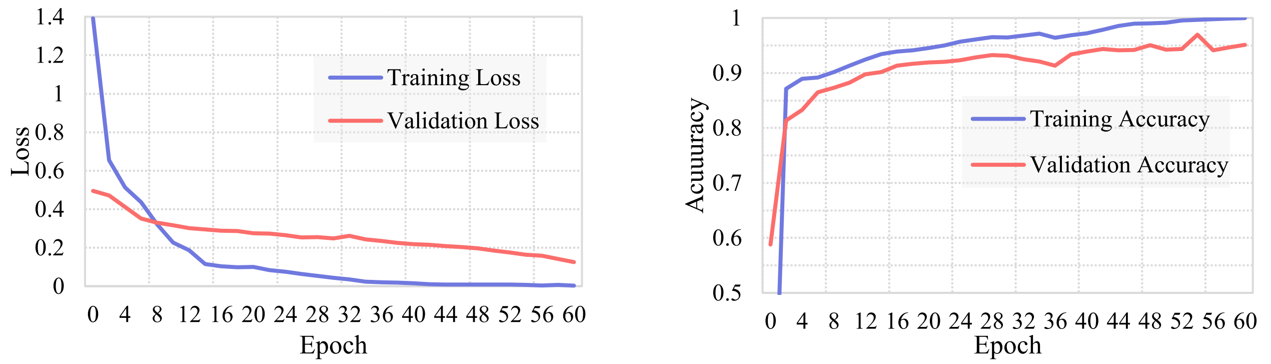
The performance of the HCF is shown in Fig. 12. It can be observed that the HCF not only improves the recognition accuracy for distracted driver behaviors (99.95% on the training set and 96.74% on the validation set) but also reduces the loss.

Table 1 lists the detection results for the ten classes of distracted driving behaviors separately predicted by ResNet50, Inception V3, Xception and the proposed HCF. As shown in Table 1, the overall behavior recognition accuracy of ResNet50 achieves an average accuracy of 93.72%. The Inception V3 model achieves an average accuracy of 90.13%, and the Xception model achieves an average accuracy of 91.08%. the HCF outperforms these three models with regard to detection accuracy. Among the various classes of distracted driving behavior, all the models achieved the most accurate detection results on the ''Texting Right''.



(a) Loss with varying epochs (b) Accuracy with varying epochs

FIGURE 11. The classification results using Xception.



(a) Loss with varying epochs (b) Accuracy with varying epochs

FIGURE 12. Distracted driver classification results using HCF.

Interestingly, the detection accuracy for ‘‘Talking Right’’ is obviously higher than that for ‘‘Talking Left’’ and is almost equal to the classification accuracies for ‘‘Texting Right’’ and ‘‘Texting Left’’. The worst result occurs for safe driving behavior.

2) THE CONFUSION MATRIX

Fig. 13 illustrates the confusion matrixes for ten classes of distracted driving behaviors using ResNet50, Inception V3, Xception and the proposed HCF. The yellow diagonal shows the percentage of correctly detected cases for each class. The leftmost column shows the actual label of each class, and the top row shows the output label. From the confusion matrixes in Fig. 13(a), Fig. 13(b) and Fig. 13(c), it can be observed that the model confuses the ‘‘Reaching Behind’’ (C7) and ‘‘Talking to Passengers’’ (C9) classes. Additionally, it shows that the three single pretrained models may classify distracted behaviors as safe driving, which is dangerous for practical use. The proposed HCF never mistook a distracted behavior as ‘‘Safe Driving’’; thus, it is more reliable in practice.

3) CLASS ACTIVATION MAPPING

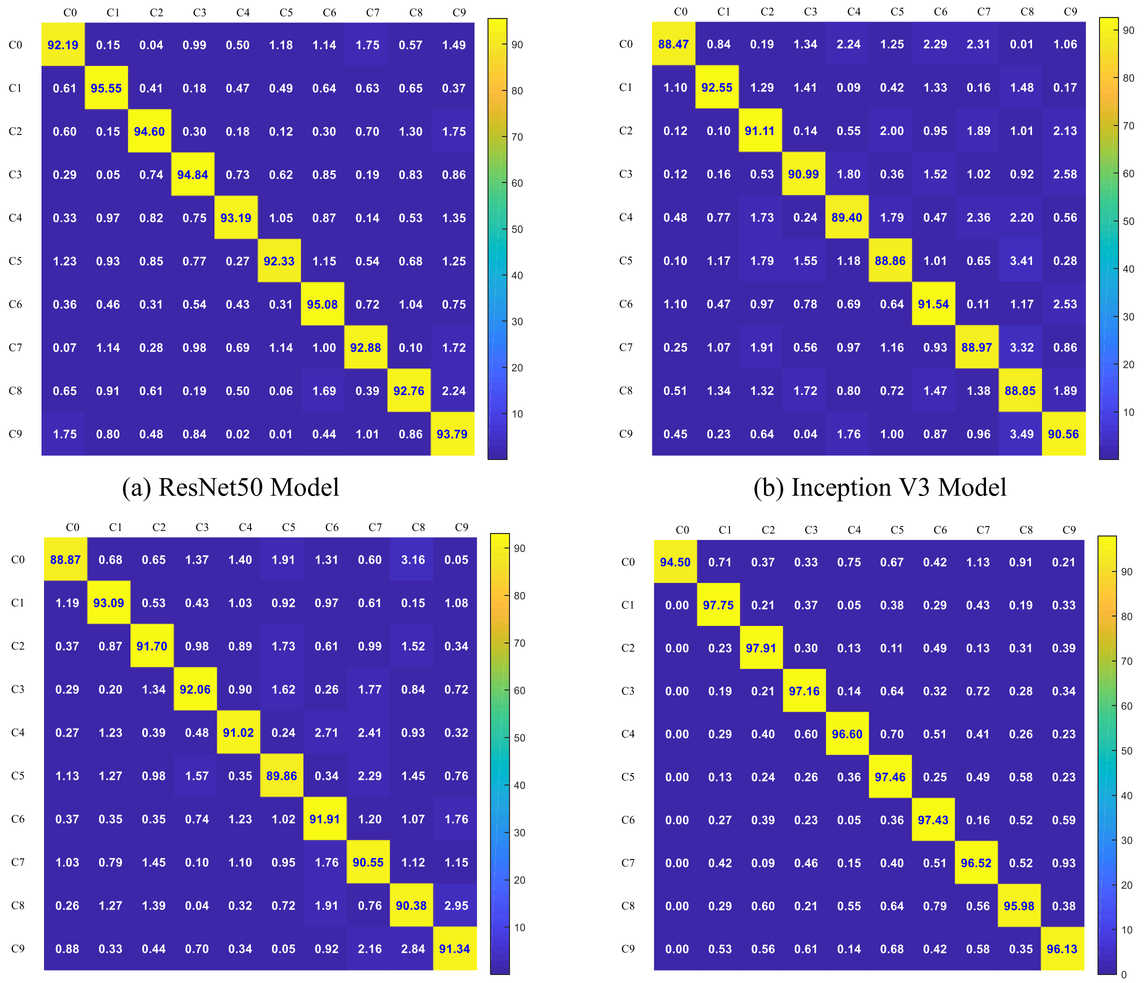
We use the class activation mapping (CAM) [33] technique to highlight the detection area on which each pretrained model focuses. Fig. 14 shows the heatmap created by CAM for ResNet50, Inception V3, and Xception, and Fig. 15 shows the heatmap created by CAM for the HCF. According to the highlighted area, we can determine the detection results for the extracted features; then, the classification decision can be made for the distracted driving behaviors.

As shown in Figs. 14 and 15, the single pretrained models extract fewer features, and they always focus only on whether the hands are on the steering wheel. The HCF extracts more features than the three pretrained models. Additionally, the detection results by the HCF are closer to the features observed by humans.

When the misclassifications in the confusion matrixes (shown in Fig. 13) are combined with the heatmap, we can explain the results. For example, ‘‘Reaching Behind’’ and ‘‘Talking to Passengers’’ are confused because the drivers are sitting sideways in both situations. From the detection results shown in Table 1, we can find that the features of ‘‘Talking Right’’, ‘‘Texting Right’’ and ‘‘Texting Left’’ all focus on the hand and mobile phone. However, the features of ‘‘Talking Left’’ focus on whether the left hand is on the wheel, which brings a lower accuracy than does ‘‘Talking Right’’. As the CAM results

TABLE 1. Classification accuracy of using different models.

Driver Status	ResNet50 Accuracy (%)	Inception V3 Accuracy (%)	Xception Accuracy (%)	Hybrid CNN Framework Accuracy (%)
Safe driving	92.19	88.47	88.87	**94.50**
Right-handed texting	95.55	92.55	93.09	**97.75**
Right-handed phone use	94.60	91.11	91.70	**97.91**
Left-handed texting	94.84	90.99	92.06	**97.16**
Left-handed phone use	93.19	89.40	91.02	**96.60**
Operating the radio	92.33	88.86	89.86	**97.47**
Drinking	95.08	91.54	91.91	**97.43**
Glancing behind	92.88	88.97	90.55	**96.52**
Hair and makeup	92.76	88.85	90.38	**95.98**
Talking to passengers	93.79	90.56	91.34	**96.13**
Average	93.72	90.13	91.08	**96.74**



(a) ResNet50 Model

	C0	C1	C2	C3	C4	C5	C6	C7	C8	C9
C0	92.19	0.15	0.04	0.99	0.50	1.18	1.14	1.75	0.57	1.49
C1	0.61	95.55	0.41	0.18	0.47	0.49	0.64	0.63	0.65	0.37
C2	0.60	0.15	94.60	0.30	0.18	0.12	0.30	0.70	1.30	1.75
C3	0.29	0.05	0.74	94.84	0.73	0.62	0.85	0.19	0.83	0.86
C4	0.33	0.97	0.82	0.75	93.19	1.05	0.87	0.14	0.53	1.35
C5	1.23	0.93	0.85	0.77	0.27	92.33	1.15	0.54	0.68	1.25
C6	0.36	0.46	0.31	0.54	0.43	0.31	95.08	0.72	1.04	0.75
C7	0.07	1.14	0.28	0.98	0.69	1.14	1.00	92.88	0.10	1.72
C8	0.65	0.91	0.61	0.19	0.50	0.06	1.69	0.39	92.76	2.24
C9	1.75	0.80	0.48	0.84	0.02	0.01	0.44	1.01	0.86	93.79

(b) Inception V3 Model

	C0	C1	C2	C3	C4	C5	C6	C7	C8	C9
C0	88.47	0.84	0.19	1.34	2.24	1.25	2.29	2.31	0.01	1.06
C1	1.10	92.55	1.29	1.41	0.09	0.42	1.33	0.16	1.48	0.17
C2	0.12	0.10	91.11	0.14	0.55	2.00	0.95	1.89	1.01	2.13
C3	0.12	0.16	0.53	90.99	1.80	0.36	1.52	1.02	0.92	2.58
C4	0.48	0.77	1.73	0.24	89.40	1.79	0.47	2.36	2.20	0.56
C5	0.10	1.17	1.79	1.55	1.18	88.86	1.01	0.65	3.41	0.28
C6	1.10	0.47	0.97	0.78	0.69	0.64	91.54	0.11	1.17	2.53
C7	0.25	1.07	1.91	0.56	0.97	1.16	0.93	88.97	3.32	0.86
C8	0.51	1.34	1.32	1.72	0.80	0.72	1.47	1.38	88.85	1.89
C9	0.45	0.23	0.64	0.04	1.76	1.00	0.87	0.96	3.49	90.56

(c) Xception Model

	C0	C1	C2	C3	C4	C5	C6	C7	C8	C9
C0	88.87	0.68	0.65	1.37	1.40	1.91	1.31	0.60	3.16	0.05
C1	1.19	93.09	0.53	0.43	1.03	0.92	0.97	0.61	0.15	1.08
C2	0.37	0.87	91.70	0.98	0.89	1.73	0.61	0.99	1.52	0.34
C3	0.29	0.20	1.34	92.06	0.90	1.62	0.26	1.77	0.84	0.72
C4	0.27	1.23	0.39	0.48	91.02	0.24	2.71	2.41	0.93	0.32
C5	1.13	1.27	0.98	1.57	0.35	89.86	0.34	2.29	1.45	0.76
C6	0.37	0.35	0.35	0.74	1.23	1.02	91.91	1.20	1.07	1.76
C7	1.03	0.79	1.45	0.10	1.10	0.95	1.76	90.55	1.12	1.15
C8	0.26	1.27	1.39	0.04	0.32	0.72	1.91	0.76	90.38	2.95
C9	0.88	0.33	0.44	0.70	0.34	0.05	0.92	2.16	2.84	91.34

(d) HCF

	C0	C1	C2	C3	C4	C5	C6	C7	C8	C9
C0	94.50	0.71	0.37	0.33	0.75	0.67	0.42	1.13	0.91	0.21
C1	0.00	97.75	0.21	0.37	0.05	0.38	0.29	0.43	0.19	0.33
C2	0.00	0.23	97.91	0.30	0.13	0.11	0.49	0.13	0.31	0.39
C3	0.00	0.19	0.21	97.16	0.14	0.64	0.32	0.72	0.28	0.34
C4	0.00	0.29	0.40	0.60	96.60	0.70	0.51	0.41	0.26	0.23
C5	0.00	0.13	0.24	0.26	0.36	97.46	0.25	0.49	0.58	0.23
C6	0.00	0.27	0.39	0.23	0.05	0.36	97.43	0.16	0.52	0.59
C7	0.00	0.42	0.09	0.46	0.15	0.40	0.51	96.52	0.52	0.93
C8	0.00	0.29	0.60	0.21	0.55	0.64	0.79	0.56	95.98	0.38
C9	0.00	0.53	0.56	0.61	0.14	0.68	0.42	0.58	0.35	96.13

FIGURE 13. Confusion matrixes using different approaches.

show, by using multiple pretrained models cooperatively, the HCF learns more representative features and achieves higher detection accuracy for the tested distracted driving behaviors.

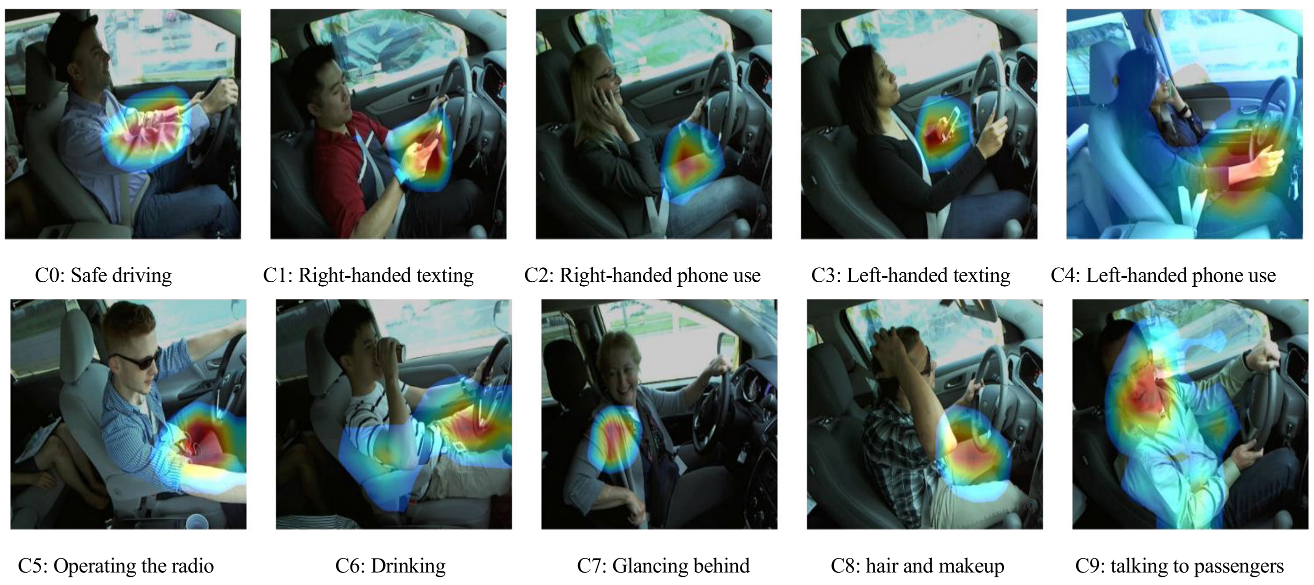


FIGURE 14. **Highlights of the feature area of distracted drivers, C0-C2 are using ResNet50, C3-C5 are using Inception V3, C6-C9 are using Xception.**

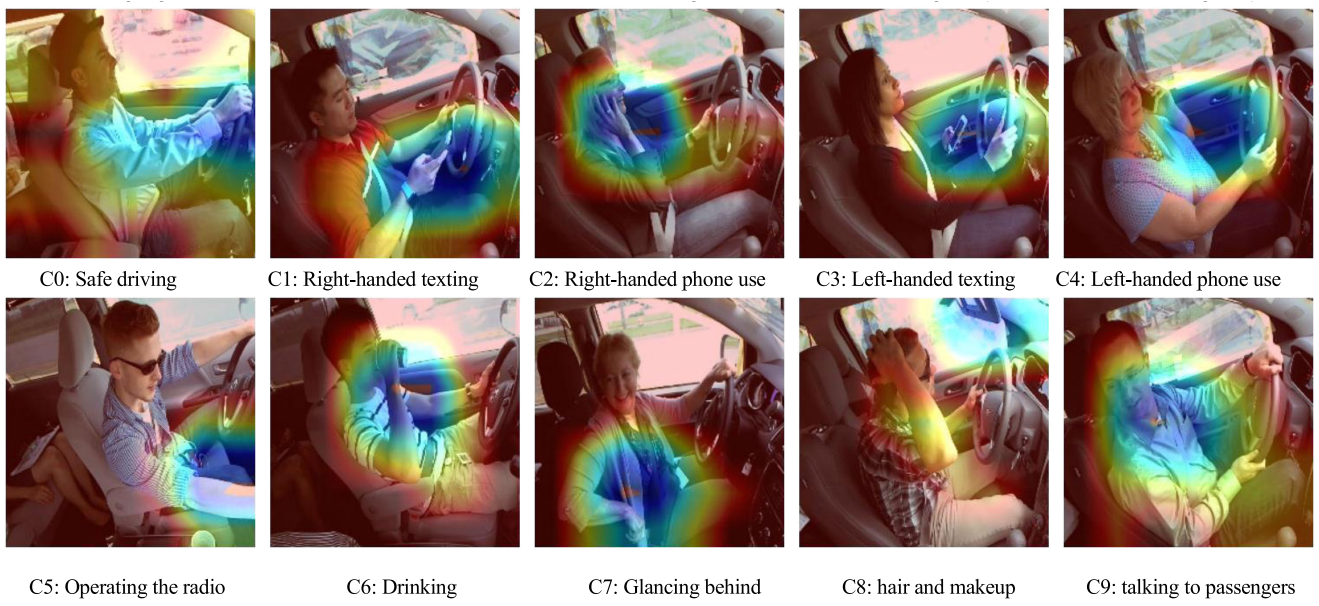


FIGURE 15. **Highlights of the feature area of distracted drivers extracted by the hybrid CNN framework.**

D. COMPARISON WITH OTHER APPROACHES

We selected several approaches for detecting distracted driving behaviors for comparison with the proposed HCF. Tran *et al.* [17] used pretrained models to extract distracted driver features and used an SVM as a classifier. Chawan *et al.* [18] used VGG16, VGG19 and Inception to classify distracted driver behaviors. Baheti *et al.* [19] proposed a modified VGG model that used regularization techniques. Moslemi *et al.* [34] proposed a 3D CNN and used optical flow to improve the detection accuracy for distracted driving behaviors.

1) ACCURACY

To compare with the proposed HCF with other approaches, we chose 80% of the images from the dataset for use as a training set and used the remaining 20% as the validation set. The results are listed in Table 2.

The results reveal that the CNN-based approaches perform much better than other approaches, which occurs because the CNN-based approaches learn more features. The approaches proposed in [17]–[19] are similar to the HCF; however. because we fine-tune specific layers of the cooperative pre-trained models, the HCF extracts more features. The approach

TABLE 2. Comparison of the detection accuracy.

Approach	Accuracy (%)
Handcrafted Features with SVM [17]	27.70
AlexNet [17]	72.60
VGG-16 [17]	82.50
ResNet152 [17]	85.00
VGG-19 [18]	77.00
Inception V3 [18]	73.00
Modified VGG [19]	95.54
3D-CNN [34]	94.40
HCF	**96.74**

proposed in [34] uses a 3D-CNN and optical flow to improve the distracted driver monitoring performance. However, this model was not trained sufficiently.

2) AVERAGE PROCESSING TIME

The execution efficiency of the deep learning algorithm is affected by the running state of the computer. The heat dissipation status and the number of parallel tasks both affect the real-time running speed of the computer, which results in the difference in the image processing time even under the same experimental conditions.

To alleviate the impact of the experimental conditions and accurately measure the average processing time, we have conducted 10 groups of experiments for each algorithm. We randomly selected 100 images from the dataset in each experiment group. The comparison results are listed in Table 3. The time refers to the average runtime required to process one image from the selected 100 images. The average value of each 10 groups of experiments is taken as the final average processing time.

We introduce momentum into the Adam strategy and designed a momentum-based training rate optimization (MTRO) algorithm as show in Algorithm 1. As an improved Adam strategy, MTRO is more stable and has a faster convergence speed, which can speed up the convergence of the proposed HCF.

As shown in Table 3, in all 10 groups of experiments, the proposed HCF achieves the best performance in 9 groups of experiments compared with other approaches. The proposed HCF requires approximately 41 ms to process one image, which obtains the optimal value of the average processing time. Among these approaches, the performance of the proposed HCF is much more stable in all 10 groups of experiments, which can satisfy the practical requirements better for the unpredictable distracted driving behaviors without considering the class label.

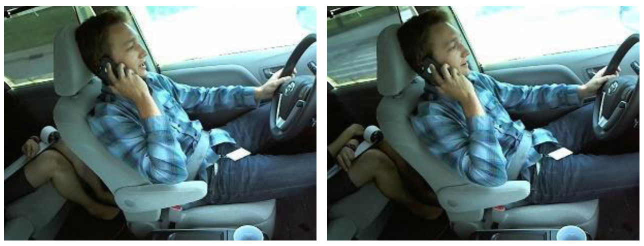


FIGURE 16. Different images of one driver in the dataset.

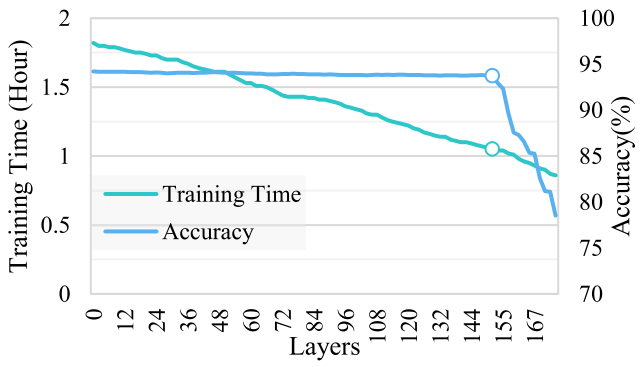


FIGURE 17. Training strategy of ResNet50.

E. DISCUSSION

1) IMPACT OF DATASET DIVISION

First, we randomly selected 80% of the dataset as the training set and used the remaining 20% as the validation set. This approach can introduce overfitting on the validation set because the dataset consists of a sequence of frames drawn from the video, and randomly dividing the dataset into an 8:2 ratio causes the driver images in the validation set and the training set to be highly similar. Fig. 16 shows two different images of one driver in the State Farm dataset. Clearly, these images could easily be confused with each other.

Hence, we adopted the images numbered P014, P021, P026, and P049 as the verification set (total: 4,320) and included the other images in the training set (total: 18,104). In this way, the images in the training set and validation set are not as apt to be confused with each other.

2) IMPACT OF TRAINING STRATEGY

We used transfer learning to fine-tune the three pretrained deep CNN models in the HCF. In transfer learning, the fine-tuning mainly focuses on tuning only a few layers while preserving the main characteristics of the pretrained convolutional layers. In the HCF, the parameters of the pretrained models are not directly transferred to our behavior detection task; instead, we fine-tune the three pretrained models to extract suitable features. In these experiments, we tested different numbers of frozen layers to find the best training strategy.

TABLE 3. Comparison of the average processing time.

Approach	Average runtime for one image (s)										Average processing time (s)
	Test1	Test2	Test3	Test4	Test5	Test6	Test7	Test8	Test9	Test10	
AlexNet	0.124	0.119	0.137	0.167	0.145	0.162	0.121	0.131	0.129	0.152	0.139
VGG-16	0.098	0.071	0.074	0.063	0.075	0.071	0.056	0.082	0.078	0.079	0.075
ResNet152	0.159	0.149	0.152	0.139	0.144	0.170	0.149	0.158	0.160	0.151	0.153
VGG-19	0.104	0.073	0.082	0.085	0.076	0.083	0.063	0.077	0.087	0.081	0.081
Inception V3	0.087	0.087	0.069	0.098	0.081	0.073	0.083	0.076	0.096	0.093	0.084
Modified VGG	**0.025**	0.072	0.075	0.068	0.059	0.074	0.058	0.063	0.054	0.057	0.061
3D-CNN	0.094	0.081	0.064	0.072	0.095	0.102	0.094	0.106	0.095	0.119	0.092
HCF	0.037	**0.042**	**0.038**	**0.043**	**0.039**	**0.042**	**0.038**	**0.043**	**0.041**	**0.042**	**0.041**

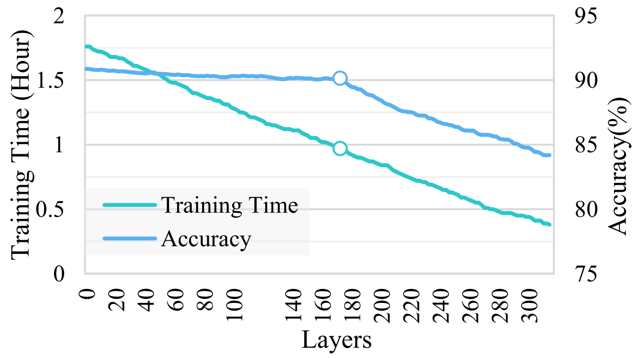


FIGURE 18. Training strategy of Inception V3.

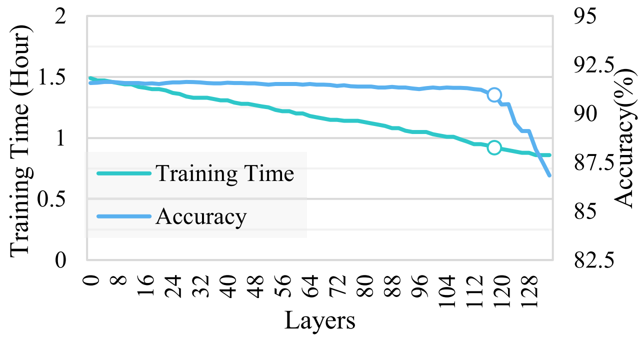


FIGURE 19. Training strategy of Xception.

The experimental results are shown in Figs. 17–19. When the number of frozen layers is small, training requires more time. However, the performance of the single pretrained models do not improve as the training time increases. Conversely, when the number of frozen layers is large, the accuracy decreases.

To consider the balance between the training time and the accuracy, for the HCF, we chose 151, 171 and 115 as the number of frozen layers for ResNet50, Inception V3 and Xception, respectively.

VI. CONCLUSION AND FUTURE WORK

Distracted driving behaviors are a primary cause of traffic accidents. Hence, it is necessary to find methods to effectively identify distracted driving behaviors. In this paper, we propose a hybrid CNN framework (HCF) to recognize distracted driver behaviors. Features are extracted at different scales by three cooperative pretrained CNN models; then, the features are concatenated to obtain the feature maps. Subsequently, we train the fully connected layer to classify each distracted driving behavior. During the training procedure, we apply dropout technology to prevent the training model from overfitting to the training data. We apply CAM to highlight the detection area results. The results show that the proposed HCF achieves good performance for recognizing distracted driver behaviors, reaching a classification accuracy of 96.74%.

The dataset in this paper only provides the images from the right-hand side. However, the performance of the proposed HCF may degrade when the camera is set in different places in the vehicle. In addition, the HCF is greatly influenced by the light, which indicates that the HCF cannot detect the distracted driving behavior accurately at night.

Considering the above limitations, in the future, we will analyze the driving behavior from different camera angles and work towards reducing the computation time and the number of parameters. Additionally, the goal is to not only to recognize distracted driving behaviors but also to prevent them.

REFERENCES

[1] World Health Organization. (2020). *Road Traffic Injuries*. [Online]. Available: https://www.who.int/news-room/fact-sheets/detail/road-traffic-injuries

[2] W.-C. Yang, Z.-X. Chen, M. Zhang, G.-G. Guo, and W.-B. Zhang, "The analysis of road condition causes of traffic accidents on mountainous highway and its safety countermeasures," in *Proc. CICTP*, Jul. 2019, pp. 3784–3796.

[3] M. Chen, Y. Tian, G. Fortino, J. Zhang, and I. Humar, "Cognitive Internet of vehicles," *Comput. Commun.*, vol. 120, pp. 58–70, May 2018.

[4] Y. Xing, C. Lv, H. Wang, D. Cao, and E. Velenis, "An ensemble deep learning approach for driver lane change intention inference," *Transp. Res. C, Emerg. Technol.*, vol. 115, Jun. 2020, Art. no. 102615.

[5] Y. Xing, C. Lv, H. Wang, H. Wang, Y. Ai, D. Cao, E. Velenis, and F.-Y. Wang, "Driver lane change intention inference for intelligent vehicles: Framework, survey, and challenges," *IEEE Trans. Veh. Technol.*, vol. 68, no. 5, pp. 4377–4390, May 2019.

[6] A. D. McDonald, H. Alambeigi, J. Engström, G. Markkula, T. Vogelpohl, J. Dunne, and N. Yuma, "Toward computational simulations of behavior during automated driving takeovers: A review of the empirical and modeling literatures," *Hum. Factors, J. Hum. Factors Ergonom. Soc.*, vol. 61, no. 4, pp. 642–688, Jun. 2019.

[7] *National Highway Traffc Safety Administration Traffc Safety Facts*. Accessed: Dec. 19, 2019. [Online]. Available: https://www.nhtsa.gov/risky-driving/distracted-driving

[8] H. Mårtensson, O. Keelan, and C. Ahlström, "Driver sleepiness classification based on physiological data and driving performance from real road driving," *IEEE Trans. Intell. Transp. Syst.*, vol. 20, no. 2, pp. 421–430, Feb. 2019.

[9] X. Zhang, X. Zhou, M. Lin, and J. Sun, "ShuffleNet: An extremely efficient convolutional neural network for mobile devices," in *Proc. CVPR*, Jun. 2018, pp. 6848–6856.

[10] J. Su, D. V. Vargas, and K. Sakurai, "One pixel attack for fooling deep neural networks," *IEEE Trans. Evol. Comput.*, vol. 23, no. 5, pp. 828–841, Oct. 2019.

[11] X. Yuan, P. He, Q. Zhu, and X. Li, "Adversarial examples: Attacks and defenses for deep learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 9, pp. 2805–2824, Sep. 2019.

[12] W. Li, W. Tao, J. Qiu, X. Liu, X. Zhou, and Z. Pan, "Densely connected convolutional networks with attention LSTM for crowd flows prediction," *IEEE Access*, vol. 7, pp. 140488–140498, 2019.

[13] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Mar. 24, 2020, doi: 10.1109/TNNLS.2020.2978386.

[14] D. Beck and W. Park, "Perceived importance of automotive HUD information items: A study with experienced HUD users," *IEEE Access*, vol. 6, pp. 21901–21909, 2018.

[15] H. N. Esfahani *et al.*, "Prevalence of cell phone use while driving and its impact on driving performance, focusing on near-crash risk: A survey study in Tehran," *J. Transp. Saf. Secur.*, vol. 1, no. 1, pp. 1–21, 2019.

[16] T. H. N. Le, Y. Zheng, C. Zhu, K. Luu, and M. Savvides, "Multiple scale faster-RCNN approach to driver's cell-phone usage and hands on steering wheel detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2016, pp. 46–53.

[17] D. Tran, H. M. Do, W. Sheng, H. Bai, and G. Chowdhary, "Real-time detection of distracted driving based on deep learning," *IET Intell. Transp. Syst.*, vol. 12, no. 10, pp. 1210–1219, Dec. 2018.

[18] P. P. M. Chawan, S. Satardekar, D. Shah, R. Badugu, and A. Pawar, "Distracted driver detection and classification," *J. Eng. Res. Appl.*, vol. 8, no. 4, pp. 60–64, 2018.

[19] B. Baheti, S. Gajre, and S. Talbar, "Detection of distracted driver using convolutional neural network," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2018, pp. 1032–1038.

[20] Y. Xing, C. Lv, H. Wang, D. Cao, E. Velenis, and F.-Y. Wang, "Driver activity recognition for intelligent vehicles: A deep learning approach," *IEEE Trans. Veh. Technol.*, vol. 68, no. 6, pp. 5379–5390, Jun. 2019.

[21] Y. Xing, C. Lv, Z. Zhang, H. Wang, X. Na, D. Cao, E. Velenis, and F.-Y. Wang, "Identification and analysis of driver postures for in-vehicle driving activities and secondary tasks recognition," *IEEE Trans. Comput. Social Syst.*, vol. 5, no. 1, pp. 95–108, Mar. 2018.

[22] C. Yan, "Vision-based driver behaviour analysis," Univ. Liverpool, Liverpool, U.K., Tech. Rep. 706848, 2016.

[23] M. Ramzan, H. U. Khan, S. M. Awan, A. Ismail, M. Ilyas, and A. Mahmood, "A survey on state-of-the-art drowsiness detection techniques," *IEEE Access*, vol. 7, pp. 61904–61919, 2019.

[24] M. Shahverdy, M. Fathy, R. Berangi, and M. Sabokrou, "Driver behavior detection and classification using deep convolutional neural networks," *Expert Syst. Appl.*, vol. 149, Jul. 2020, Art. no. 113240.

[25] *State Farm Distracted Driver Detection*. Accessed: Aug. 2, 2016. [Online]. Available: https://www.kaggle.com/c/state-farm-distracted-driver-detection

[26] M. S. Majdi, S. Ram, J. T. Gill, and J. J. Rodríguez, "Drive-Net: Convolutional network for driver distraction detection," in *Proc. IEEE SSIAI*, Apr. 2018, pp. 1–4.

[27] C. Ou and F. Karray, "Enhancing driver distraction recognition using generative adversarial networks," in *Proc. IEEE Trans. Intell. Vehicles*, early access, Dec. 19, 2019, doi: 10.1109/TIV.2019.2960930.

[28] B. Zhou, H. Zhao, X. Puig, T. Xiao, S. Fidler, A. Barriuso, and A. Torralba, "Semantic understanding of scenes through the ADE20K dataset," *Int. J. Comput. Vis.*, vol. 127, no. 3, pp. 302–321, Mar. 2019.

[29] Y. Luo and N. Mesgarani, "Conv-TasNet: Surpassing ideal time–frequency magnitude masking for speech separation," *IEEE/ACM Trans. Audio, Speech, Language Process.*, vol. 27, no. 8, pp. 1256–1266, Aug. 2019.

[30] C. Cangea, P. Veličković, and P. Lio, "XFlow: Cross-modal deep neural networks for audiovisual classification," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Nov. 8, 2019, doi: 10.1109/TNNLS.2019.2945992.

[31] Y. Luo, Y. Wong, M. Kankanhalli, and Q. Zhao, "$\mathcal{G}$-softmax: Improving intraclass compactness and interclass separability of features," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 2, pp. 685–699, Feb. 2020.

[32] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," *J. Comput. Phys.*, vol. 378, pp. 686–707, Feb. 2019.

[33] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba "Learning deep features for discriminative localization," in *Proc. CVPR*, Jun. 2016, pp. 2921–2929.

[34] N. Moslemi, R. Azmi, and M. Soryani, "Driver distraction recognition using 3D convolutional neural networks," in *Proc. 4th IPRIA*, Mar. 2019, pp. 145–151.

CHEN HUANG received the B.Eng. and Ph.D. degrees in communication and information system from the Huazhong University of Science and Technology, Wuhan, China, in 2005 and 2010, respectively. He is currently an Associate Professor with the Department of Computer and Information Engineering, Hubei University, Wuhan. His research interests include the Internet of Things, autonomous driving, machine learning, and big data analysis in brain–computer interface.

XIAOCHEN WANG received the B.Sc. degree in computer science from Hubei University, Wuhan, China, in 2020. His research interest includes Internet of vehicles, machine learning, and software engineering.

JIANNONG CAO (Fellow, IEEE) received the B.Sc. degree in computer science from Nanjing University, China, in 1982, and the M.Sc. and Ph.D. degrees in computer science from Washington State University, USA, in 1986 and 1990, respectively. He is currently the Chair Professor of the Department of Computing, The Hong Kong Polytechnic University, Hong Kong. He is also the Director of the Internet and Mobile Computing Laboratory, in the department, and the Director of the University Research Facility in Big Data Analytics. He has coauthored five books in mobile computing and wireless sensor networks, coedited nine books, and published over 600 papers in major international journals and conference proceedings. His research interests include parallel and distributed computing, wireless networks and mobile computing, big data and cloud computing, pervasive computing, and fault tolerant computing. He is a Distinguished Member of ACM and a Senior Member of the China Computer Federation (CCF).

SHIHUI WANG received the B.Sc. degree in computer science from Wuhan University, Wuhan, China, in 1986, and the M.Sc. degree in software engineering from Zhengzhou University, Zhengzhou, China. He is currently a Professor with the Department of Computer and Information Engineering, Hubei University, Wuhan. He is also the Director of the Education Information Engineering Technology Research Center of Hubei Province. His research interests include big data analysis, artificial intelligence, and virtual reality. He is also a Senior Member of the China Computer Federation (CCF).

YAN ZHANG received the B.Sc. and M.Sc. degrees in computer science from Hubei University, Wuhan, China, in 1997 and 2002, respectively, and the Ph.D. degree in software engineering from Beihang University, Beijing, China. He is currently a Professor with the Department of Computer and Information Engineering, Hubei University. He is also the Vice-Director of the Education Information Engineering Technology Research Center of Hubei Province. His research interests include information security, big data analysis, and software defect detection. He is also a Senior Member of the China Computer Federation (CCF).