

Article

# A Real-Time Subway Driver Action Sensing and Detection Based on Lightweight ShuffleNetV2 Network

Xing Shen <sup>1,2</sup>  and Xiukun Wei <sup>1,2,\*</sup> <sup>1</sup> School of Traffic and Transportation, Beijing Jiaotong University, Beijing 100044, China<sup>2</sup> State Key Laboratory of Advanced Rail Autonomous Operation, Beijing Jiaotong University, Beijing 100044, China

\* Correspondence: xkwei@bjtu.edu.cn

**Abstract:** The driving operations of the subway system are of great significance in ensuring the safety of trains. There are several hand actions defined in the driving instructions that the driver must strictly execute while operating the train. The actions directly indicate whether equipment is normally operating. Therefore, it is important to automatically sense the region of the driver and detect the actions of the driver from surveillance cameras to determine whether they are carrying out the corresponding actions correctly or not. In this paper, a lightweight two-stage model for subway driver action sensing and detection is proposed, consisting of a driver detection network to sense the region of the driver and an action recognition network to recognize the category of an action. The driver detection network adopts the pretrained MobileNetV2-SSDLite. The action recognition network employs an improved ShuffleNetV2, which incorporates a spatial enhanced module (SEM), improved shuffle units (ISUs), and shuffle attention modules (SAMs). SEM is used to enhance the feature maps after convolutional downsampling. ISU introduces a new branch to expand the receptive field of the network. SAM enables the model to focus on important channels and key spatial locations. Experimental results show that the proposed model outperforms 3D MobileNetV1, 3D MobileNetV3, SlowFast, SlowOnly, and SE-STAD models. Furthermore, a subway driver action sensing and detection system based on a surveillance camera is built, which is composed of a video-reading module, main operation module, and result-displaying module. The system can perform action sensing and detection from surveillance cameras directly. According to the runtime analysis, the system meets the requirements for real-time detection.

**Keywords:** action recognition; deep learning; driver detection; railway; action sensing and detection



**Citation:** Shen, X.; Wei, X. A Real-Time Subway Driver Action Sensing and Detection Based on Lightweight ShuffleNetV2 Network. *Sensors* **2023**, *23*, 9503. <https://doi.org/10.3390/s23239503>

Academic Editors: Sukho Lee and Dae-Ki Kang

Received: 19 October 2023  
Revised: 23 November 2023  
Accepted: 27 November 2023  
Published: 29 November 2023



**Copyright:** © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

## 1. Introduction

With the development of computer technology, researchers apply advanced artificial intelligence technology in the field of transportation, such as traffic sign detection in the field of road traffic [1,2], railway surface [3–6] and fastener [7,8] defect detection in the field of rail transit. With the rapid development of urban rail transit, the subway system has become the preferred mode of public transportation, which undoubtedly raises higher requirements for the safety of trains. However, there is limited research on subway driver action sensing and detection based on surveillance cameras. Subway drivers play a crucial role in the safe operation of trains. They need to confirm each step to ensure that no step is missed. The actions of drivers indicate that the current equipment is normally operating. Currently, the monitoring of driver actions is mainly carried out by two surveillance cameras installed in the driver cab. The manual inspection of surveillance videos is used to determine whether the driver has performed the corresponding actions. This method is both inefficient and costly. Therefore, it is of great significance to conduct research on the action sensing and detection of subway drivers based on surveillance cameras to realize the real-time automatic detection of driver action categories. It can help

reduce costs, enhance the operational safety of trains, and improve the intelligence level of urban rail transit monitoring systems.

Subway driver action sensing and detection (SDASD) belongs to spatial–temporal action detection (STAD), which aims to detect the spatial positions of individuals in the current frame and determine their action categories. In the past, complex handcrafted features have been used, such as spatial–temporal interest points, motion trajectories, etc., for video action recognition [9–13]. These methods achieved good results for simple actions. However, due to the complexity of designing and computing handcrafted features, these methods suffer from slow recognition speed and are not suitable for practical applications.

In recent years, with the rapid development of deep learning, scholars have started to utilize deep neural networks for STAD, which can be categorized into two-stage methods and one-stage methods. The mainstream approach is the two-stage method, where the first stage uses pre-trained object detectors [14–16] to generate human region proposals in the current frame. In the second stage, an action recognition network, often utilizing 3D CNNs to extract spatial–temporal features from video clips, is used for action recognition [17]. Gu et al. [18] introduce an AVA dataset and propose a STAD approach. The region proposal network adopts ResNet50, and the action recognition network uses the I3D network [19], which integrates RGB and optical flow, and finally performs action classification. Subsequently, several models are proposed to improve the performance of STAD, such as ACRN (Actor-Centric Relation Network) [20], STEP (Spatio-TEMPoral Progressive) [21], LFB (Long-term Feature Banks) [22], SlowFast [23], Context-Aware RCNN [24], ACARN (Actor-Context-Actor Relation Network) [25] and so on. These two-stage models achieve high accuracy, but the human region detection and action recognition are independent, making these models inefficient. In recent studies, several one-stage models are proposed, where a single network is used for both human region detection and action recognition. These models include WOO (Watch Only Once) [26], SE-STAD (Simple and Efficient Spatial–Temporal Action Detector) [27], and DOAD (Decoupled one-stage action detection network) [28]. However, one-stage models face challenges, such as unsatisfied precision. Deep learning-based algorithms have significantly improved the accuracy of STAD, but the deep convolution network models often have a large number of parameters and computation cost, leading to a slow detection speed that does not meet the real-time detection requirements.

To enable deep neural networks to run on devices with limited computational resources, some lightweight networks have been proposed, such as MobileNet [29–31] and ShuffleNet [32,33]. ShuffleNetV2 [33] reduces the number of parameters and model size significantly by introducing depthwise separable convolutions. It incorporates channel split and channel shuffle to facilitate information exchange between channels. ShuffleNetV2 [33] strikes a good balance between speed and precision.

As of now, there are no reported studies specifically focusing on SDASD based on surveillance cameras. Some researchers have conducted studies on driver action recognition. For instance, Hu et al. [34] propose the RepC3D model, which combines C3D [35] and RepVGG [36] for recognizing subway driver actions. Suo et al. [37] introduce an improved dense trajectory algorithm for driver action recognition. These studies primarily focus on video-level action recognition, where video clips are used for action classification, without explicitly detecting the region of drivers. Different from the above works, our main objective is to apply advanced artificial intelligence technology in the field of subway driver action detection, which is less relevant research work at present. We propose appropriate improvements on the basis of the existing model to further improve the detection performance. And then based on the improved model, we build a real-time driver action detection system to realize real-time video reading from the surveillance camera to carry out action detection, which makes it possible to deploy the system in the subway cab in the future.

This paper aims to achieve real-time sensing of the region of the subway driver and recognition of their action category based on surveillance cameras. For this purpose, a two-

stage model for subway driver action sensing and detection is proposed. In the first stage, the region of the driver is localized by employing a pre-trained lightweight network called MobileNetV2-SSDLite. The network generates driver candidate region proposals along with confidence scores, which are used for subsequent action recognition. In the second stage, an improved ShuffleNetV2 is proposed to extract the spatial-temporal features of the video clips and recognize the category of actions. To boost the performance of network, a spatial enhanced module is introduced to compensate for spatial information loss caused by downsampling. A new branch with larger convolutional kernels is added to expand the receptive field of the network and a shuffle attention module is used to help the network focus the attention on important channels and spatial positions. Experimental results show that the proposed model outperforms other models, achieving a mAP of 72.44%, 4.87% higher than the baseline. Furthermore, a subway driver action sensing and detection system based on surveillance cameras is built, which performs real-time action detection directly by reading video from surveillance cameras. It is composed of a video-reading module, main operation module and result-displaying module. The performance of the system shows that it meets the requirements for real-time detection. The main contributions are summarized as follows:

1. A real-time subway driver action sensing and detection model is proposed, which consists of a driver detection network and an action recognition network. The driver detection network is used to locate the region of the driver in the images, and the action recognition network is employed to recognize the category of the action.
2. A spatial enhanced module is introduced after the first convolution downsampling layer, compensating for the loss of spatial information and enhancing the spatial positions of the feature map. In addition, a dataset specifically for subway driver action sensing and detection is constructed.
3. A new branch with a large convolutional kernel in the shuffle units is proposed to expand the receptive field, which is crucial for the subsequent action recognition. In addition, the shuffle attention module is introduced to help the network focus the attention on important channels and spatial positions.
4. A real-time subway driver action sensing and detection system based on surveillance cameras is built, which reads video from surveillance cameras and performs SDASD directly. According to the runtime analysis, the system meets the requirements for real-time detection.

The rest of this paper is organized as follows. Section 2 introduces the problems studied in this paper. In Section 3, a lightweight two-stage model for subway driver action sensing and detection is proposed. Section 4 introduces the detailed experiments and results. In Section 5, a subway driver action sensing and detection system is introduced. Section 6 summarizes the conclusions and presents an outlook for future work.

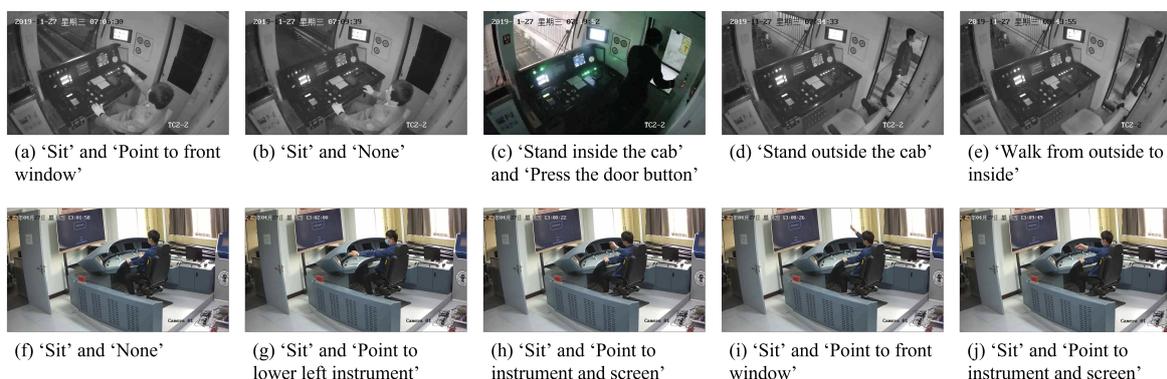
## 2. Problem Statements

Subway drivers play a crucial role in ensuring the safe operation of urban rail transit. They need to confirm each step with their fingers to ensure that no step is missed. The actions of the driver can be used to determine whether the equipment is normally operating or not. Currently, the monitoring of driver actions and their states is primarily performed through the installation of two surveillance cameras in the driver cab (one located at the bottom left corner and the other at the top right corner). The action monitoring uses manual inspection of the recorded videos to check the actions of driver. This manual approach is inefficient and costly.

With the trend of intelligent development in urban rail transit, more and more advanced artificial intelligence algorithms are being applied to detection tasks. However, there is limited research on monitoring subway driver actions. Existing studies mostly focus on classifying driver actions from 2D images, namely recognizing the action category in a single image. Such methods can only identify simple actions without temporal relationships and do not achieve comprehensive action recognition. An action is often composed

of multiple consecutive frames with temporal dependencies, and relying on a single image is insufficient for recognizing actions with temporal relationships. Suo et al. [37] study subway driver action recognition by video clips from the perspective of the video level. The actions can be categorized as arrival confirmation, departure confirmation, interval confirmation, platform closing confirmation, and no action. An improved dense trajectory-based method has been proposed for recognizing driver gesture actions. While this method achieves high accuracy, it suffers from slow speed and is used for pre-cropped action video clips; thus, it cannot provide the real-time localization of the driver region and the recognition of driver actions.

In this paper, the action sensing and detection of subway drivers based on surveillance cameras is studied, aiming to sense the region of the driver and recognize their current actions from surveillance videos. According to surveillance videos in subway driver cabs, driver actions and states are categorized into 11 classes as shown in Figure 1, including sitting (Sit), standing inside the cab (StinCab), standing outside the cab (StoutCab), walking from inside to outside (Wafri2O), walking from outside to inside (Wafro2I), pointing to the instrument and screen (Po2InSc), pointing to the front window (Po2FrWin), pointing to the lower left instrument (Po2LLin), pressing the door button (PrDoBu), pushing the instrument (PuIn), and no action (None). For the task of driver region localization, the lightweight object detection algorithm MobileNetv2-SSDLite is employed to locate the region of the driver in the image, and the coordinates and confidence scores are obtained. These coordinates and scores are used as regions of interest (ROIs) and fed to the action recognition network. For driver action recognition, an improved lightweight ShuffleNet2 is proposed to extract spatial-temporal features from multiple frames of input. The ROIs from the driver region localization task are mapped onto the final feature map, followed by ROI pooling to generate fixed-size features for ROIs, and then processed through fully connected layers for action recognition. To evaluate the real-time performance of the proposed model, a subway driver action sensing and detection system based on surveillance cameras is built, consisting of three parts: video-reading module, main operation module, and result-displaying module. The video-reading module stores the video streams in a reading queue, the main operation module samples frames from the queue and performs driver region localization and driver action recognition on the sampled frames, and the result-displaying module renders the results on the frames and composes the video for display on the screen. A surveillance camera is installed, and the system can perform action sensing and detection on the video streams from the surveillance camera. According to the runtime of each module, the system achieves real-time detection.



**Figure 1.** Action categories. The images in the first rows are from the surveillance video of the driver cab, and the images in the second rows are from simulated video.

### 3. Methodology

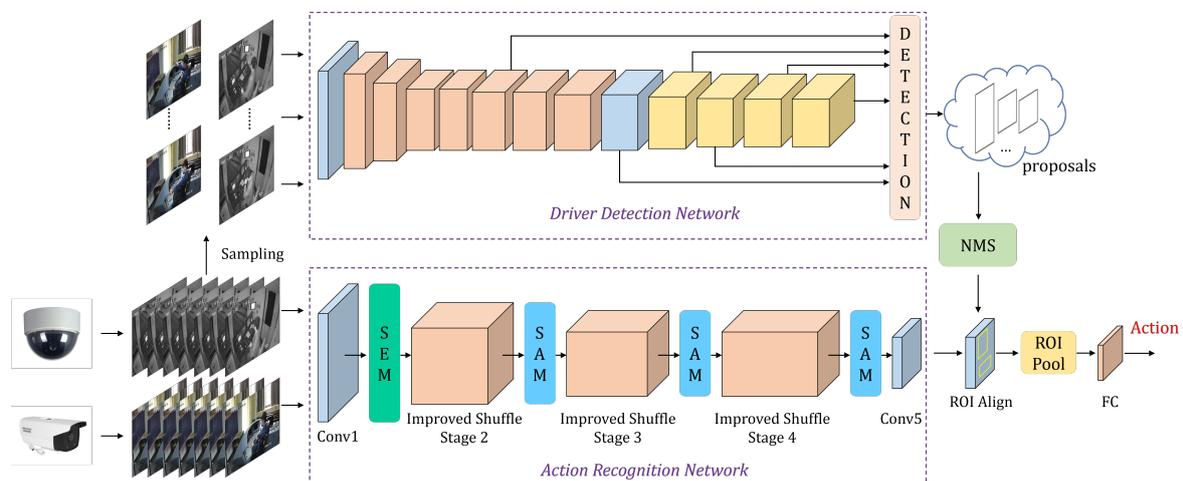
#### 3.1. Overall Framework

The overall framework of the model is shown in Figure 2. The model is composed of the driver detection network and the action recognition network. The driver detection

network generates proposals of the driver region, and the action recognition network recognizes the current action category of the driver. To achieve fast and accurate detection, lightweight networks are employed for both tasks. For the driver detection network, a pre-trained MobileNetV2-SSDLite [30] is utilized, which has a small number of parameters and model size, allowing it to quickly and accurately detect the region of the driver. For the action recognition network, an improved ShuffleNetV2 is proposed. The network also has a small number of parameters and model size, enabling the fast and accurate recognition of actions.

### 3.2. Driver Detection Network

In this paper, the pre-trained MobileNetV2-SSDLite is adopted for the driver detection network. It aims to sense and locate the region of the driver in the image and obtain the coordinates and confidence scores. The SSD [16] is a classic one-stage object detection algorithm that can simultaneously perform object localization and classification in a single stage. It combines the advantages of the anchor-based mechanism from region proposal algorithms and the regression-based algorithm in one-stage methods, resulting in high accuracy and fast detection speed. The original SSD uses VGG16 as a base network. However, the large number of parameters in the VGG16 makes it unsuitable for running on resource-limited embedded devices and mobile devices. To address this issue, the SSDLite object detection network based on MobileNetV2 is proposed, which reduces the number of parameters and computation. Specifically, the VGG16 is replaced with MobileNetV2 for feature extraction. Additionally, instead of using regular convolutions, the extra convolution layers in SSDLite utilize depthwise separable convolutions (DWConv) as the basic structure. The input size of image is  $320 \times 320$ . Predictions are made by using six different-sized feature maps when detecting, and proposals of the driver region along with confidence scores are obtained. These proposals are then selected by non-maximum suppression (NMS) to obtain the final proposals, used for subsequent action recognition.



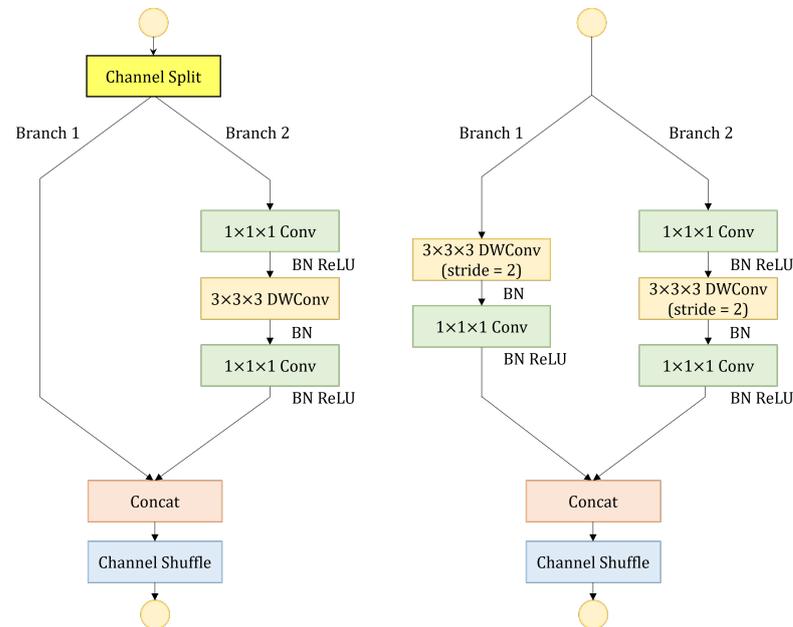
**Figure 2.** The overall framework of the model.

### 3.3. Action Recognition Network

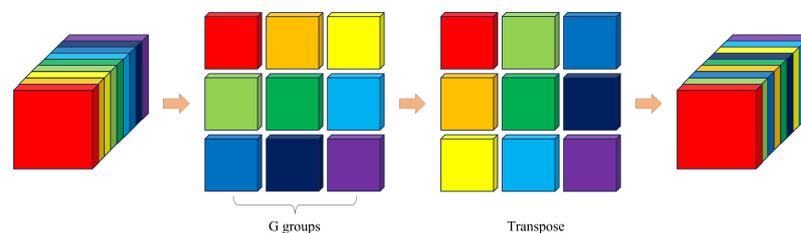
#### 3.3.1. ShuffleNetV2

Three-dimensional ShuffleNetV2 [33] is a lightweight network with small number of parameters and computation. The shuffle units of 3D ShuffleNetV2 are shown in Figure 3. When the stride is one, the channels are split into two branches. Branch 1 does not have any operations, while branch 2 adopts DWConv. The outputs of branch 1 and branch 2 are concatenated, followed by a channel shuffle module. When the stride is two, the input is processed by two branches. The outputs of branch 1 and branch 2 are concatenated. The channel shuffle operation allows for information exchange between channels. The channel shuffle module is shown in Figure 4. For a given feature map with a specific number of

channels, the channels are first divided into  $G$  groups. Then, the groups are transposed and rearranged to obtain the shuffled feature map. The channel shuffle operation does not introduce any parameters and achieves channel-wise information exchange through simple grouping and transposition operations.



**Figure 3.** The structure of the shuffle units.



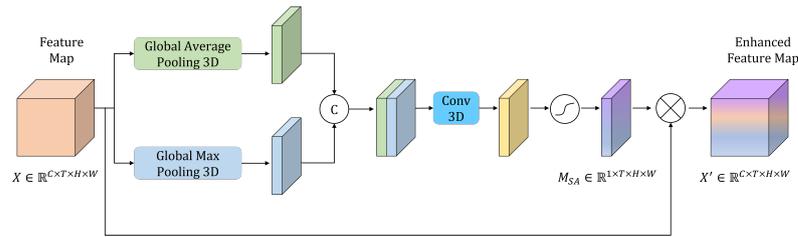
**Figure 4.** The structure of the channel shuffle.

### 3.3.2. Spatial Enhanced Module (SEM)

When the network performs convolutional downsampling, the size of the feature maps decreases. Though this allows for capturing high-level features, downsampling leads to spatial information loss. To reduce the information loss, a SEM is added after the first convolution to enhance the spatial representation capability of network. The specific structure is shown in Figure 5. Firstly, the global average pooling and global max pooling are adopted. The resulting feature maps are then concatenated and processed by a 3D convolution to extract features. An activation function is applied to enhance the representation ability of the network, and the feature maps are multiplied element-wise with the original feature maps to obtain the enhanced feature maps. The formulas are denoted as follows :

$$M_{SA} = \sigma(\text{conv}([\text{AvgPool}(X); \text{MaxPool}(X)])) \quad (1)$$

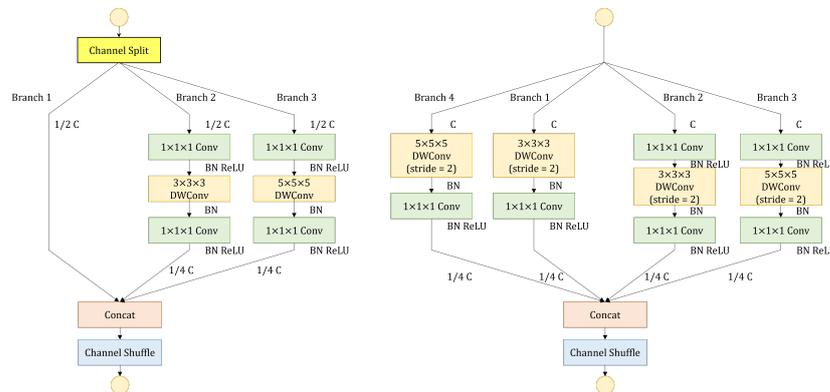
$$X' = X \otimes M_{SA} \quad (2)$$



**Figure 5.** The structure of the spatial enhanced module.

### 3.3.3. Improved Shuffle Units (ISUs)

The receptive field of the network can be expanded by using large convolution kernels, which is crucial for subsequent tasks. Inspired by it, a new branch with a  $5 \times 5 \times 5$  kernel size is added to the ShuffleNet units as shown in Figure 6. When the stride is 1, branch 3 is added to obtain a larger receptive field. The rest of branch 3 is the same as branch 2. To ensure that the final concatenation has the same number of channels, the output channels of the last convolution in branch 2 and branch 3 are set to  $1/4$  of the input channels. Branch 1, branch 2, and branch 3 are then concatenated to obtain feature maps, followed by a channel shuffle module. When the stride is 2, branch 3 and branch 4 are added with a bigger kernel size. The output channels of the last convolution in all branches are set to  $1/4$  of the input channels. Branch 1, branch 2, branch 3, and branch 4 are then concatenated to obtain the final feature maps, followed by a channel shuffle module. The introduction of improved shuffle units increases the number of computations, about 1.5 times to 2 times as much as the original shuffle units. The detailed calculation procedure is shown in Appendix A.



**Figure 6.** The structure of the improved shuffle units.

### 3.3.4. Shuffle Attention Module (SAM)

The attention module allows the network to focus on important features and suppress unimportant features. There are two main types of attention mechanisms: channel attention and spatial attention. Channel attention focuses on “what”, while spatial attention focuses on “where”. In this paper, to boost the representation ability in both the spatial and channel dimensions of the feature maps, the shuffle attention module (SAM) [38] is added after each improved shuffle stage. The structure of SAM is shown in Figure 7. First, the feature maps are divided into  $g$  groups along the channel dimension. Each group is further divided into two branches, namely, the channel attention branch and the spatial attention branch. These branches are responsible for generating different channel and spatial importance weights.

**Channel attention branch:** Instead of using the traditional SE [39] module, which introduces a large number of parameters, a simple combination of global average pooling, scale, and sigmoid is adopted. Firstly, global average pooling is used to embed global information, generating  $s \in \mathbb{R}^{\frac{c}{g} \times 1 \times 1 \times 1}$ .

$$s = F_{gp}(X_{k1}) = \frac{1}{T \times H \times W} \sum_{i=1}^T \sum_{j=1}^H \sum_{k=1}^W X_{k1}(i, j, k) \quad (3)$$

$$X'_{k1} = \sigma(F_c(s)) \cdot X_{k1} = \sigma(W_1s + b_1) \cdot X_{k1} \tag{4}$$

Spatial attention branch: Unlike the channel attention branch, spatial attention focuses on the spatial dimension. Firstly, Group Norm (GN) is used to obtain statistical information along the spatial dimensions. Then,  $F_c(\cdot)$  is applied to enhance the spatial attention. The formula is denoted as follows:

$$X'_{k2} = \sigma(W_2 \cdot GN(X_{k2}) + b_2) \cdot X_{k2} \tag{5}$$

Aggregation: After obtaining the channel attention weights and spatial attention weights, it is necessary to aggregate them. Firstly, a simple concatenation operation is adopted. Then, inter-group information exchange is performed by channel shuffle module.

The computation of the grouping operation in SAM is about  $\frac{1}{4g^2}$  of that of the non-grouping operation. The detailed calculation procedure is shown in Appendix B.

### 3.3.5. Network Structure

After a detailed introduction of each component of the network, the complete action recognition network is presented along with its structure and specific parameters in Table 1. The network consists of two standard convolutions, one max pooling, one global average pooling, one SEM module, three improved shuffle stages, three SAM modules, one ROI Align and Pooling, and one fully connected layer. In Table 1, T represents the number of input frames, and there is no downsampling in the temporal dimension.

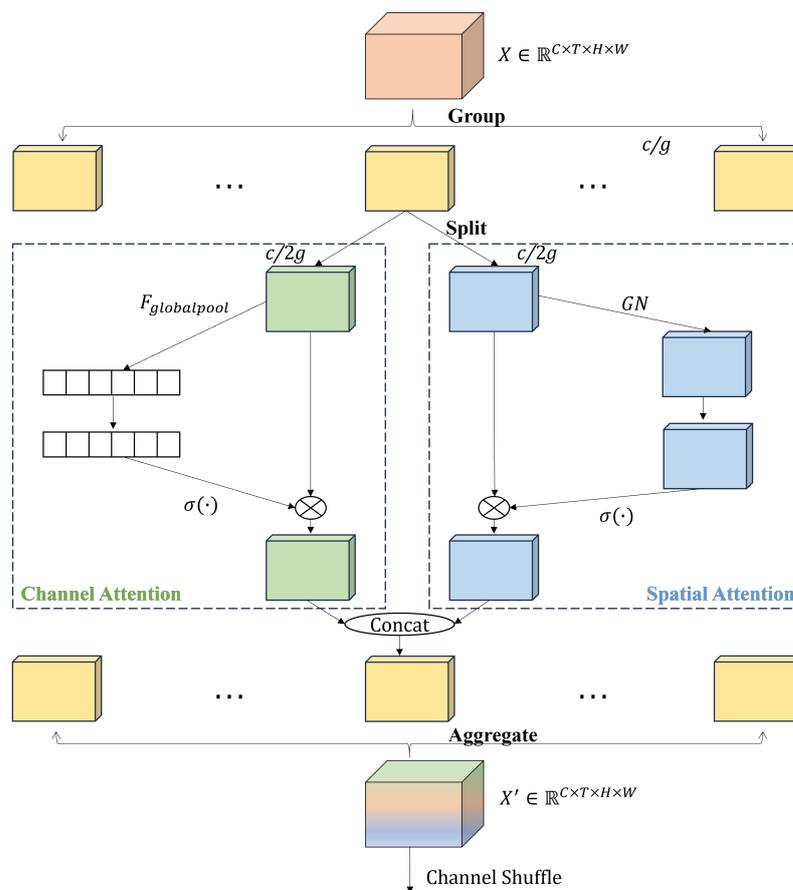


Figure 7. The structure of the shuffle attention module.

**Table 1.** Structure and parameters of the action recognition network.

Operator	Output Size	KSize	Stride	Rep.	Output Channels
Frames	$T \times 256 \times 256$	-	-	-	3
Conv1	$T \times 128 \times 128$	$3 \times 3 \times 3$	(1,2,2)	1	24
MaxPool	$T \times 64 \times 64$	$3 \times 3 \times 3$	(1,2,2)	1	24
SEM	$T \times 64 \times 64$	-	-	-	24
Improved Shuffle Stage2	$T \times 32 \times 32$	-	(1,2,2), (1,1,1)	1, 3	116
SAM	$T \times 32 \times 32$	-	-	-	116
Improved Shuffle Stage3	$T \times 16 \times 16$	-	(1,2,2), (1,1,1)	1, 7	232
SAM	$T \times 16 \times 16$	-	-	-	232
Improved Shuffle Stage4	$T \times 16 \times 16$	-	(1,1,1)	4	464
SAM	$T \times 16 \times 16$	-	-	-	464
Conv5	$T \times 16 \times 16$	$1 \times 1 \times 1$	(1,1,1)	1	1024
ROI Align & Pool	$1 \times 8 \times 8$	-	-	-	1024
Global Average Pool	$1 \times 1 \times 1$	-	-	-	1024
Fully Connected	-	-	-	-	12

KSize represents kernel size, Rep. represents repeat number.

## 4. Experiments

### 4.1. Dataset Preparation

The videos of the driver cab used in the experiment are from Beijing Metro Line 9, with a video resolution of  $1280 \times 720$ . The simulation video is recorded by the surveillance camera installed in the lab, with a video resolution of  $1920 \times 1080$ . Each raw surveillance video is about an hour long, and if it is fed directly into the model, it requires a huge amount of memory and computing resources. Therefore, in order to better sense and detect the category of the subway driver action, 328 video clips are cropped with a duration of 10 s from the original driver cab videos and simulation videos, of which 163 clips are from the actual surveillance video and 165 clips are from laboratory simulation videos. Each clip contains two action labels. The action labels includes sitting (Sit), standing inside the cab (StinCab), standing outside the cab (StoutCab), walking from inside to outside (Wafir2O), walking from outside to inside (WafirO2I), pointing to the instrument and screen (Po2InSc), pointing to the front window (Po2FrWin), pointing to the lower left instrument (Po2LLin), pressing the door button (PrDoBu), pushing the instrument (PuIn), and no action (None). The number of labeled actions is shown in Table 2. The driver detection network does not require additional datasets. As the proposed model works, the frames for the driver detection network are sampled from the input video clip and no additional datasets are required.

The annotation method refers to the AVA dataset format [18]. First, the video is extracted into a series of frames with an FPS of 30. The AVA dataset format does not label all frames but annotates 1 frame per second. Therefore, in the spatial–temporal detection dataset of the subway driver action, the first frame per second is annotated [40]. Since the first and last 2 s of the videos are not involved in detecting, only images with indexes of 61, 91, 121, 151, 181, 211, 241 are labeled for each video clip with a duration of 10 s. In order to quickly label the region of the driver, the pre-trained YOLOv5 [41] is used to detect the region of the driver, and the coordinates and confidence scores of the proposals are obtained as rough labeling, and then the rough labeling is imported into the VIA [42] labeling tool for action category labeling. The annotation process is shown in Figure 8.

**Table 2.** The number of labeled actions.

Action	Training Set	Testing Set	Total
Sit	953	402	1355
StinCab	157	89	246
StoutCab	326	147	473
Wafri2O	82	25	107
WafriO2I	91	23	114
Po2InSc	109	31	140
Po2FrWin	62	21	83
Po2LLin	61	22	83
PrDoBu	98	58	156
PuIn	268	150	418
None	465	181	646
<b>Total</b>	<b>2672</b>	<b>1149</b>	<b>3821</b>

**Figure 8.** The annotation process.

The detailed process by which MobileNetV2-SSDLite results are used by improved shuffleNetV2 to drive the action classification task is as follows.

Taking a cropped video clip for example, the first and last 2 s of the videos are not involved in detecting (AVA dataset format), and the index of 61, 91, 121, 151, 181, 211, and 241 frames are labeled. Therefore, taking these seven frames as the center, and sampling eight frames each, we can obtain seven clips for the model training, where each clip has eight frames.

Taking the frame with index 61 as an example, with 61 as the center and an interval of 8, a total of 8 frames are sampled, that is, the index corresponding to the sampled frames is (29, 37, 45, 53, 61, 69, 77, 85), and the action label is the same as the label of the frame with index 61. For the driver detection network, these eight frames are input in sequence to obtain the driver proposals (namely anchors) in the eight frames. Then, the non-maximum suppression (NMS) is then used to filter invalid anchors that exceed a fixed threshold. The retained anchors after NMS are then mapped on the feature map of the last convolution layer of the improved ShuffleNetV2 (namely the output feature map of conv5, as shown in Figure 1). For the action recognition network, these eight frames are taken as an input. The region of the driver on the feature map, namely the region of interest (ROI), can be obtained, and then the ROIs which have different sizes are transformed into fixed sizes by ROI pooling, and finally the fixed size features are sent to the fully connected layer for action classification.

#### 4.2. Evaluation Indicators and Experimental Details

The purpose of subway driver action sensing and detection is to sense and locate the region of the driver and recognize the category of the action, paying more attention to driver action recognition. A common evaluation indicator is mAP (mean Average Precision), which is the average of AP in all action categories.

In this paper, all the models are implemented by PyTorch and trained on 1 NVIDIA A6000 GPU. The CPU is Inter(R) Xeon(R) silver 4314 @2.4 GHz.

The input of the model is eight frames, which are sampled at equal intervals. The interval in this paper is set to eight, that is, one frame is sampled every eight frames. When model training, eight images are scaled to  $256 \times 256$  after sampling, and horizontal flip is introduced to augment the dataset. For model testing, the height of the image is scaled to 256, and the width is scaled proportionally. The parameters are set as follows: the optimizer is SGD, the initial learning rate is 0.01, the weight momentum is 0.00003, and the learning rate decay strategy is cosine annealing, where warmup\_iters is 500, warmup\_ratio is 0.1, the minimum learning rate is 0.00001, and the training epoch is 200.

#### 4.3. Experimental Results

For the driver detection network, the Intersection over Union (IoU) threshold is set to 0.5. For the action recognition network, the prediction score threshold is set to 0.9. The experimental results are shown in Table 3. The baseline is the original 3D ShuffleNetV2 [33]. As can be seen from Table 3, SEM, ISU, and SAM can improve the performance of the model. ISU brings the most obvious gain; the mAP can increase 4.13%, and the number of parameters is smaller than the baseline. When all modules are added to the network, the mAP increases to 72.44%, 4.87% higher than the baseline.

Table 3. Ablation experiments.

Baseline	SEM	ISU	SAM	mAP
✓				0.6757
✓	✓			0.6842
✓		✓		0.7170
✓			✓	0.7114
✓	✓	✓	✓	<b>0.7244</b>

Table 4 shows the AP of each action. It can be seen that the introduction of SEM, ISU and SAM improve the AP of most action categories, among which walking from inside to outside has the most obvious improvement, from 0.5143 to 0.8309, improved by 31.66%, followed by pressing the door button, which increases from 0.2172 to 0.3276, an improvement of 11.04%. Though the AP of walking from outside to inside, pointing to the instrument and screen, and pushing the instrument decrease slightly, other categories of actions have different amplitudes of increment. It can be concluded that the introduction of SEM, ISU and SAM improve the performance of the model. In addition, it is seen that the AP of pressing the door button action is low, owing to the small number of action instances in the actual surveillance video, and this action is not simulated due to the constraints of the operation console, which further leads to a smaller number of actions than other actions, resulting in the model not learning the feature of action well. Figure 9 shows the confusion matrix for the recognition results of the proposed model. The action recognition network can almost accurately classify all kinds of actions; only seven actions are classified incorrectly. Of these seven actions, one StinCab and one StoutCab are classified as Sit because the driver is close to the seat in the two video clips, leading to the wrong classification. One StinCab is classified as WafrI2O because the StinCab occurs at the junction where the two actions occur, resulting in a classification error. Two WafrO2I are classified as StoutCab because these actions are both outside the cab and the action after StoutCab is WafrO2I. One WafrI2O is classified as WafrO2I because at the door, WafrI2O

is similar to WafrO2I. One WafrO2I is classified as PuIn because in the real cab video, the action behind WaFrO2I is PuIn, resulting in a recognition error at the action connection. Overall, it can be seen that the proposed model can well recognize the action categories of the driver.

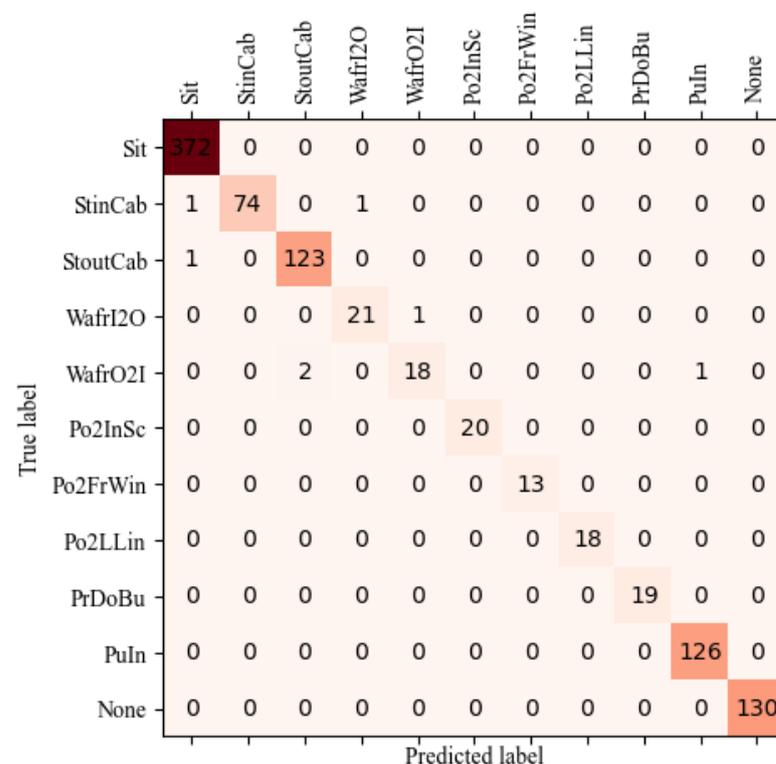
**Table 4.** The AP of actions.

Action	Baseline	+SEM	+ISU	+SAM	+SEM+ISU+SAM
Sit	0.8760	0.8672	0.8804	0.8680	<b>0.8817</b>
StinCab	0.7865	0.8090	0.8315	0.8090	<b>0.8315</b>
StoutCab	0.8231	<b>0.8503</b>	0.8360	0.8298	0.8296
WafirI2O	0.5143	0.8696	<b>0.8783</b>	0.8200	0.8309
WafirO2I	0.8239	0.7826	<b>0.8261</b>	0.7712	0.7666
Po2InSc	0.5839	0.4073	<b>0.6144</b>	0.4315	0.5823
Po2FrWin	0.5714	0.6190	0.5159	0.6667	<b>0.6122</b>
Po2LLin	0.7727	0.7246	0.7702	0.7727	<b>0.8182</b>
PrDoBu	0.2172	0.1724	0.2165	0.3575	<b>0.3276</b>
PuIn	0.7967	0.7463	<b>0.8240</b>	0.8047	0.7865
None	0.6670	0.6774	0.6936	0.6940	<b>0.7013</b>
mAP	0.6757	0.6842	0.7170	0.7114	<b>0.7244 (+4.87%)</b>

In order to prove that the SAM module is better than the SE module, we carried out a comparative experiment. Table 5 shows the comparison results. It can see that the SAM has fewer parameters and better performance than the SE module.

**Table 5.** Comparison between SE and SAM.

Model	Model Size	Parameters	GFlops	mAP
+SEM+ISU+SE	10.50 M	1,329,416	6.36	0.7230
+SEM+ISU+SAM	10.26 M	1,294,740	6.36	0.7244



**Figure 9.** The confusion matrix of improved ShuffleNetV2.

In order to evaluate the performance of the proposed network, 3D MobileNetV1 [29], 3D MobileNetV3 [31], SlowFast-R50 [23], SlowOnly-R50 [23], and SE-STAD [27] are selected for comparison. The experimental results are shown in Table 6. The results show that the proposed network has a higher mAP and a smaller number of parameters and model size than the compared networks. Table 7 shows the AP of each action. It can be seen that the APs of the proposed model are better than the compared models in most action categories, and the proposed network reaches the state of the art.

Table 6. Comparison with other networks.

Model	mAP	Parameters	Model Size	GFlops
3D MobileNetV1 [29]	0.6164	3,310,284	25.43 M	14.60
3D MobileNetV3 [31]	0.7189	1,165,964	9.07 M	3.39
SlowFast-R50 [23]	0.6095	33,671,220	257.46 M	193.57
SlowOnly-R50 [23]	0.6630	31,659,084	241.91 M	166.08
SE-STAD [27]	0.7068	40,650,557	310 M	213.65
<b>ours</b>	<b>0.7244</b>	<b>1,294,740</b>	<b>10.26 M</b>	<b>6.36</b>

Table 7. The AP of actions.

Action	3D MobileNetV1 [29]	3D MobileNetV3 [31]	SlowFast-R50 [23]	SlowOnly-R50 [23]	SE-STAD [27]	Ours
Sit	0.8742	0.8737	0.8753	0.869	0.9801	0.8817
StinCab	0.7978	0.8427	0.7978	0.8539	0.8202	0.8315
StoutCab	0.8298	0.8299	0.8639	0.8502	0.9048	0.8296
Wafri2O	0.76	0.8345	0.64	0.781	0.4	0.8309
WafriO2I	0.5652	0.7826	0.3913	0.6957	0.3478	0.7666
Po2InSc	0.4649	0.5755	0.5066	0.5145	0.7041	0.5823
Po2FrWin	0.1905	0.6444	0.4459	0.4238	0.7963	0.6122
Po2LLin	0.7273	0.7727	0.5455	0.6364	0.8636	0.8182
PrDoBu	0.1552	0.1551	0.1207	0.1207	0.1034	0.3276
PuIn	0.7929	0.8174	0.8266	0.8259	0.9267	0.7865
None	0.6231	0.7789	0.6906	0.7222	0.9282	0.7013
mAP	0.6164	0.7189	0.6095	0.663	0.7068	0.7244

Figure 10 shows the driver action detection results from the actual surveillance video in the driver cab. It can be seen that the model can precisely locate the region of the driver and recognize the category of action.



Figure 10. Cont.

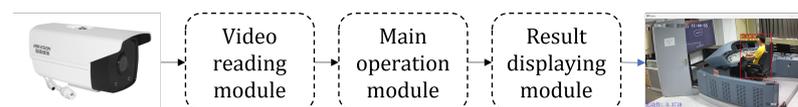


**Figure 10.** The action sensing and detection result of the subway driver.

## 5. Subway Driver Action Sensing and Detection System

### 5.1. System Structure

Based on the proposed model, a subway driver action sensing and detection system (SDASD) is built, aiming to read the video from the surveillance camera in real-time and conduct SDASD directly. The system structure is shown in Figure 11. The system is composed of a video-reading module, main operation module and result-displaying module.



**Figure 11.** The structure of subway driver action sensing and detection system.

#### 5.1.1. Video-Reading Module

The real-time video is read from the surveillance camera, and video frames are stored in the reading queue for the main operation module.

#### 5.1.2. Main Operation Module

Frames are sampled from the reading queue according to the given sampling strategy (that is, sampling 1 frame at an interval of 8, and sampling 8 frames in total). The eight frames are sent to the driver detection network to predict the region of the driver, and the coordinates and confidence scores of the proposals are obtained. The region proposals are drawn in the frames, which are taken as the ROIs in the subsequent action recognition. In addition, the eight frames are pre-processed for the action recognition network. In the last layer of convolution, the ROIs in the driver detection network are mapped on the feature map, and finally the ROI pooling is carried out to obtain the fixed-size feature map. Finally, the action categories and corresponding confidence scores are predicted through the fully connected layer.

### 5.1.3. Result-Displaying Module

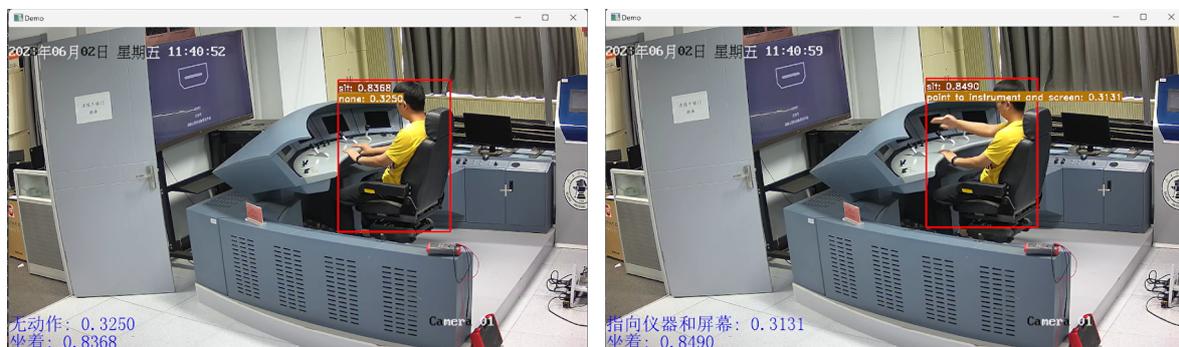
The result-displaying module mainly draws the action recognition results on the frames, including the action category and its corresponding confidence score, then synthesizes the video at a fixed frame rate, and finally displays it on the screen.

### 5.2. Performance Evaluation

To evaluate the performance of the system, experiments are conducted on a personal laptop with a Core (TM) i5-12500H 2.5 GHz CPU and an NVIDIA RTX 2050 (4 GB) GPU. Since the input images are sampled from 64 frames (about  $64/30 = 2.13$  s), that is, the model needs to complete the whole process (video-reading module, main operation module, and result-displaying module) within 2.13 s to meet the real-time detection requirements. Table 8 shows the time consumed of each module. It can be seen that the complete runtime of our model is between 0.6 s and 0.75 s, which is much less than 2.13 s. Compared with the other models, our model is better regarding total runtime. Therefore, it is concluded that the model meets the requirements of real-time action detection from surveillance cameras. The results of the real-time detection system are shown in Figure 12.

**Table 8.** The runtime of each module.

Model	Video-Reading Module	Main Operation Module	Result-Displaying Module	Total
3D MobileNetV1 [29]	280–350 ms	120–180 ms	300–350 ms	700–880 ms
3D MobileNetV3 [31]	250–350 ms	100–180 ms	280–350 ms	630–880 ms
SlowFast-R50 [23]	280–350 ms	360–450 ms	380–450 ms	1020–1250 ms
SlowOnly-R50 [23]	280–350 ms	220–280 ms	250–300 ms	750–930 ms
ours	250–300 ms	100–150 ms	250–300 ms	600–750 ms



**Figure 12.** The results of real-time detection system.

## 6. Conclusions

In this paper, a lightweight two-stage model for subway driver action sensing and detection based on surveillance cameras is proposed. It consists of the driver detection network and the action recognition network. The driver detection network adopts MobileNetV2-SSDLite, with the purpose of locating the region of the driver. The action recognition network employs the improved ShuffleNetV2 to extract spatial-temporal features and recognizes the category of action. The proposed network has a smaller number of parameters and model size than the compared networks. The experimental results show that the proposed network outperforms the compared networks, with a mAP of 72.44%, 4.87% higher than the baseline. Then a subway driver action sensing and detection system is built based on the proposed model to realize real-time detection from surveillance cameras. The system runs on a personal laptop; according to the runtime of the system, it takes 0.6 to 0.75 s for a whole process, which is less than the video duration of 2.13 s. It can be seen that the system meets the real-time detection requirements.

In the future research, we will further optimize our system, expand the dataset, and improve the performance. In addition, the action statistics function is taken into account for the subway driver action sensing and detection to count the number of actions completed by the driver.

**Author Contributions:** Conceptualization, X.S.; methodology, X.S.; writing—original draft preparation, X.S.; writing—review and editing, X.S. and X.W.; supervision, X.W.; project administration, X.W.; funding acquisition, X.W. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research was funded by the National Key R&D Program of China (grant number 2022YFB4301204).

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

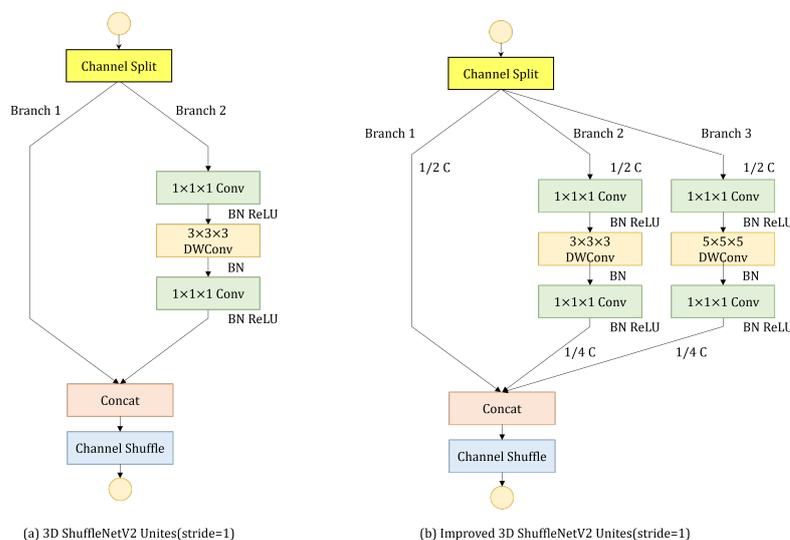
**Data Availability Statement:** Data are contained within the article.

**Conflicts of Interest:** The authors declare no conflict of interest.

### Appendix A

The additional computation resulting from the improved shuffle unit (ISU) is calculated as follows.

When the stride is one, the input shape is  $X \in \mathbb{R}^{C \times T \times H \times W}$ , and the output shape is  $X' \in \mathbb{R}^{C \times T \times H \times W}$ . The structures of the original shuffle units and improved shuffle units are shown in Figure A1.



**Figure A1.** The structures of the original shuffle units and improved shuffle units (stride = 1).

The computation of 3D ShuffleNetV2 (stride = 1):

$$\begin{aligned}
 \text{Comput\_original1} &= (2 \times \frac{C}{2} \times 1 \times 1 \times 1 - 1) \times \frac{C}{2} \times T \times H \times W \\
 &+ (\frac{2 \times \frac{C}{2} \times 3 \times 3 \times 3}{\frac{C}{2}} - 1) \times \frac{C}{2} \times T \times H \times W \\
 &+ (2 \times \frac{C}{2} \times 1 \times 1 \times 1 - 1) \times \frac{C}{2} \times T \times H \times W \\
 &= \frac{C}{2} \times T \times H \times W \times (2C + 51)
 \end{aligned}
 \tag{A1}$$

The computation of improved 3D ShuffleNetV2 (stride = 1):

$$\begin{aligned}
 \text{Comput\_improved1} &= (2 \times \frac{C}{2} \times 1 \times 1 \times 1 - 1) \times \frac{C}{2} \times T \times H \times W \\
 &+ (\frac{2 \times \frac{C}{2} \times 3 \times 3 \times 3}{\frac{C}{2}} - 1) \times \frac{C}{2} \times T \times H \times W \\
 &+ (2 \times \frac{C}{2} \times 1 \times 1 \times 1 - 1) \times \frac{C}{4} \times T \times H \times W \\
 &+ (2 \times \frac{C}{2} \times 1 \times 1 \times 1 - 1) \times \frac{C}{2} \times T \times H \times W \quad (\text{A2}) \\
 &+ (\frac{2 \times \frac{C}{2} \times 5 \times 5 \times 5}{\frac{C}{2}} - 1) \times \frac{C}{2} \times T \times H \times W \\
 &+ (2 \times \frac{C}{2} \times 1 \times 1 \times 1 - 1) \times \frac{C}{4} \times T \times H \times W \\
 &= \frac{C}{2} \times T \times H \times W \times (3C + 299)
 \end{aligned}$$

It can be seen that when the stride is one, the introduction of improved shuffle units increases the number of computations, about 1.5 times as much.

When the stride is two, the input shape is  $X \in \mathbb{R}^{C \times T \times H \times W}$ , and the output shape is  $X' \in \mathbb{R}^{C \times T \times \frac{H}{2} \times \frac{W}{2}}$ . The structures of the original shuffle units and improved shuffle units are shown in Figure A2.

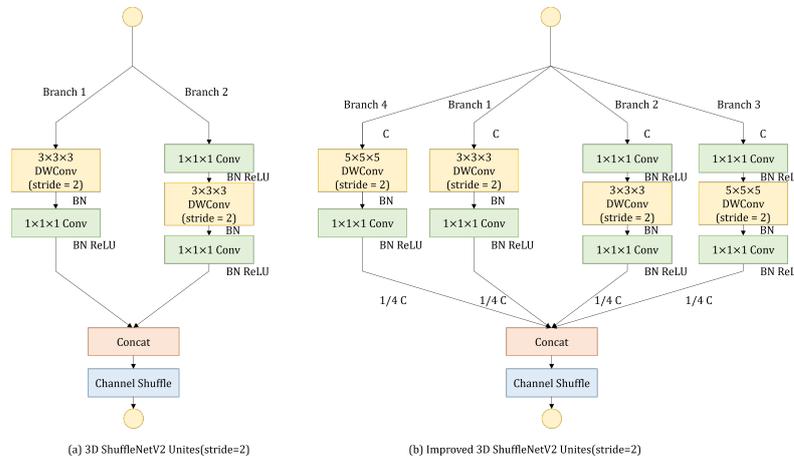


Figure A2. The structures of original shuffle units and improved shuffle units (stride = 2).

The computation of 3D ShuffleNetV2 (stride = 2):

$$\begin{aligned}
 \text{Comput\_original2} &= (2 \times C \times 1 \times 1 \times 1 - 1) \times C \times T \times H \times W \\
 &+ 2 \times (\frac{2 \times C \times 3 \times 3 \times 3}{C} - 1) \times C \times T \times \frac{H}{2} \times \frac{W}{2} \\
 &+ 2 \times (2 \times C \times 1 \times 1 \times 1 - 1) \times \frac{C}{2} \times T \times \frac{H}{2} \times \frac{W}{2} \quad (\text{A3}) \\
 &= C \times T \times \frac{H}{2} \times \frac{W}{2} \times (10C + 101)
 \end{aligned}$$

The computation of improved 3D ShuffleNetV2 (stride = 2):

$$\begin{aligned}
Comput\_improved2 &= 2 \times (2 \times C \times 1 \times 1 \times 1 - 1) \times C \times T \times H \times W \\
&+ 2 \times \left( \frac{2 \times C \times 3 \times 3 \times 3}{C} - 1 \right) \times C \times T \times \frac{H}{2} \times \frac{W}{2} \\
&+ 2 \times \left( \frac{2 \times C \times 5 \times 5 \times 5}{C} - 1 \right) \times C \times T \times \frac{H}{2} \times \frac{W}{2} \\
&+ 4 \times (2 \times C \times 1 \times 1 \times 1 - 1) \times \frac{C}{4} \times T \times \frac{H}{2} \times \frac{W}{2} \\
&= C \times T \times \frac{H}{2} \times \frac{W}{2} \times (18C + 595)
\end{aligned} \tag{A4}$$

It can be seen that the introduction of four branches increases the number of computations, about twice as much.

## Appendix B

The computation of the grouping operation and non-grouping operation is calculated as follows.

The computation of the grouping operation: ( $g$  groups)

(1) The computation of channel attention:

$$\left(2 \times \frac{C}{2g} \times 1 \times 1 \times 1 - 1\right) \times \frac{C}{2g} \times T \times H \times W \tag{A5}$$

(2) The computation of spatial attention:

$$\left(2 \times \frac{C}{2g} \times 1 \times 1 \times 1 - 1\right) \times \frac{C}{2g} \times T \times H \times W \tag{A6}$$

(3) The total computation of grouping operation:

$$2 \times \left(2 \times \frac{C}{2g} \times 1 \times 1 \times 1 - 1\right) \times \frac{C}{2g} \times T \times H \times W \tag{A7}$$

The computation of the non-grouping operation:

(1) The computation of channel attention:

$$(2 \times C \times 1 \times 1 \times 1 - 1) \times C \times T \times H \times W \tag{A8}$$

(2) The computation of spatial attention:

$$(2 \times C \times 1 \times 1 \times 1 - 1) \times C \times T \times H \times W \tag{A9}$$

(3) The total computation of non-grouping operation:

$$2 \times (2 \times C \times 1 \times 1 \times 1 - 1) \times C \times T \times H \times W \tag{A10}$$

The computation ratio of grouping and non-grouping operation:

$$\frac{group}{non\_group} = \frac{2 \times \left(2 \times \frac{C}{2g} \times 1 \times 1 \times 1 - 1\right) \times \frac{C}{2g} \times T \times H \times W}{2 \times (2 \times C \times 1 \times 1 \times 1 - 1) \times C \times T \times H \times W} \approx \frac{1}{4g^2} \tag{A11}$$

It can see that the grouping operation can reduce the number of computations. The computation of grouping operation is about  $\frac{1}{4g^2}$  of that of the non-grouping operation.

## References

1. Zhang, J.; Zheng, Z.; Xie, X.; Gui, Y.; Kim, G.J. ReYOLO: A traffic sign detector based on network reparameterization and features adaptive weighting. *J. Ambient. Intell. Smart Environ.* **2022**, *14*, 317–334. [[CrossRef](#)]
2. Zhang, J.; Zou, X.; Kuang, L.D.; Wang, J.; Sherratt, R.S.; Yu, X. CCTSDB 2021: A more comprehensive traffic sign detection benchmark. *Hum.-Centric Comput. Inf. Sci.* **2022**, *12*, 23.
3. Zhang, Z.; Zhang, L. Unsupervised Pixel-Level Detection of Rail Surface Defects Using Multistep Domain Adaptation. *IEEE Trans. Syst. Man Cybern. Syst.* **2023**, *53*, 5784–5795. [[CrossRef](#)]
4. Wei, D.; Wei, X.; Tang, Q.; Jia, L.; Yin, X.; Ji, Y. RTLSeg: A novel multi-component inspection network for railway track line based on instance segmentation. *Eng. Appl. Artif. Intell.* **2023**, *119*, 105822. [[CrossRef](#)]
5. Liu, Y.; Xiao, H.; Xu, J.; Zhao, J. A rail surface defect detection method based on pyramid feature and lightweight convolutional neural network. *IEEE Trans. Instrum. Meas.* **2022**, *71*, 1–10. [[CrossRef](#)]
6. Yang, H.; Wang, Y.; Hu, J.; He, J.; Yao, Z.; Bi, Q. Segmentation of track surface defects based on machine vision and neural networks. *IEEE Sens. J.* **2021**, *22*, 1571–1582. [[CrossRef](#)]
7. Su, S.; Du, S.; Lu, X. Geometric Constraint and Image Inpainting-Based Railway Track Fastener Sample Generation for Improving Defect Inspection. *IEEE Trans. Intell. Transp. Syst.* **2022**, *23*, 23883–23895. [[CrossRef](#)]
8. Bai, T.; Yang, J.; Xu, G.; Yao, D. An optimized railway fastener detection method based on modified Faster R-CNN. *Measurement* **2021**, *182*, 109742. [[CrossRef](#)]
9. Dollár, P.; Rabaud, V.; Cottrell, G.; Belongie, S. Behavior recognition via sparse spatio-temporal features. In Proceedings of the 2005 IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance, Beijing, China, 15–16 October 2005; pp. 65–72.
10. Wang, H.; Kläser, A.; Schmid, C.; Liu, C.L. Dense trajectories and motion boundary descriptors for action recognition. *Int. J. Comput. Vis.* **2013**, *103*, 60–79. [[CrossRef](#)]
11. Wang, H.; Schmid, C. Action recognition with improved trajectories. In Proceedings of the IEEE International Conference on Computer Vision, Sydney, NSW, Australia, 1–8 December 2013; pp. 3551–3558.
12. Peng, X.; Zou, C.; Qiao, Y.; Peng, Q. Action recognition with stacked fisher vectors. In Proceedings of the Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, 6–12 September 2014; Proceedings, Part V 13; Springer: Berlin/Heidelberg, Germany, 2014; pp. 581–595.
13. Nazir, S.; Yousaf, M.H.; Velastin, S.A. Evaluating a bag-of-visual features approach using spatio-temporal features for action recognition. *Comput. Electr. Eng.* **2018**, *72*, 660–669. [[CrossRef](#)]
14. Girshick, R.; Donahue, J.; Darrell, T.; Malik, J. Rich feature hierarchies for accurate object detection and semantic segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, USA, 23–28 June 2014; pp. 580–587.
15. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster r-cnn: Towards real-time object detection with region proposal networks. *Adv. Neural Inf. Process. Syst.* **2015**, *28*, 1–9. [[CrossRef](#)] [[PubMed](#)]
16. Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C.Y.; Berg, A.C. Ssd: Single shot multibox detector. In Proceedings of the Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, 11–14 October 2016; Proceedings, Part I 14; Springer: Berlin/Heidelberg, Germany, 2016; pp. 21–37.
17. Vahdani, E.; Tian, Y. Deep learning-based action detection in untrimmed videos: A survey. *IEEE Trans. Pattern Anal. Mach. Intell.* **2022**, *45*, 4302–4320. [[CrossRef](#)] [[PubMed](#)]
18. Gu, C.; Sun, C.; Ross, D.A.; Vondrick, C.; Pantofaru, C.; Li, Y.; Vijayanarasimhan, S.; Toderici, G.; Ricco, S.; Sukthankar, R.; et al. Ava: A video dataset of spatio-temporally localized atomic visual actions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 6047–6056.
19. Carreira, J.; Zisserman, A. Quo vadis, action recognition? A new model and the kinetics dataset. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 6299–6308.
20. Sun, C.; Shrivastava, A.; Vondrick, C.; Murphy, K.; Sukthankar, R.; Schmid, C. Actor-centric relation network. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 318–334.
21. Yang, X.; Yang, X.; Liu, M.Y.; Xiao, F.; Davis, L.S.; Kautz, J. Step: Spatio-temporal progressive learning for video action detection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 264–272.
22. Wu, C.Y.; Feichtenhofer, C.; Fan, H.; He, K.; Krahenbuhl, P.; Girshick, R. Long-term feature banks for detailed video understanding. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 284–293.
23. Feichtenhofer, C.; Fan, H.; Malik, J.; He, K. Slowfast networks for video recognition. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Seoul, Republic of Korea, 27 October–2 November 2019; pp. 6202–6211.
24. Wu, J.; Kuang, Z.; Wang, L.; Zhang, W.; Wu, G. Context-aware rcnn: A baseline for action detection in videos. In Proceedings of the Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, 23–28 August 2020; Proceedings, Part XXV 16; Springer: Berlin/Heidelberg, Germany, 2020; pp. 440–456.

25. Pan, J.; Chen, S.; Shou, M.Z.; Liu, Y.; Shao, J.; Li, H. Actor-context-actor relation network for spatio-temporal action localization. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Nashville, TN, USA, 20–25 June 2021; pp. 464–474.
26. Chen, S.; Sun, P.; Xie, E.; Ge, C.; Wu, J.; Ma, L.; Shen, J.; Luo, P. Watch only once: An end-to-end video action detection framework. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Montreal, QC, Canada, 10–17 October 2021; pp. 8178–8187.
27. Sui, L.; Zhang, C.L.; Gu, L.; Han, F. A Simple and Efficient Pipeline to Build an End-to-End Spatial-Temporal Action Detector. In Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, Waikoloa, HI, USA, 2–7 January 2023; pp. 5999–6008.
28. Chang, S.; Wang, P.; Wang, F.; Feng, J.; Shou, M.Z. DOAD: Decoupled One Stage Action Detection Network. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Vancouver, BC, Canada, 17–24 June 2023; pp. 3122–3131.
29. Howard, A.G.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W.; Weyand, T.; Andreetto, M.; Adam, H. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv* **2017**, arXiv:1704.04861.
30. Sandler, M.; Howard, A.; Zhu, M.; Zhmoginov, A.; Chen, L.C. Mobilenetv2: Inverted residuals and linear bottlenecks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 4510–4520.
31. Howard, A.; Sandler, M.; Chu, G.; Chen, L.C.; Chen, B.; Tan, M.; Wang, W.; Zhu, Y.; Pang, R.; Vasudevan, V.; et al. Searching for mobilenetv3. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Seoul, Republic of Korea, 27 October–2 November 2019; pp. 1314–1324.
32. Zhang, X.; Zhou, X.; Lin, M.; Sun, J. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 6848–6856.
33. Ma, N.; Zhang, X.; Zheng, H.T.; Sun, J. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 116–131.
34. Hu, X.; Wang, T.; Huang, J.; Peng, T.; Liu, J.; He, R. Subway Driver Behavior Detection Method Based on Multi-features Fusion. In Proceedings of the 2021 IEEE International Conference on Bioinformatics and Biomedicine (BIBM), Houston, TX, USA, 9–12 December 2021; pp. 3378–3385.
35. Tran, D.; Bourdev, L.; Fergus, R.; Torresani, L.; Paluri, M. Learning spatiotemporal features with 3d convolutional networks. In Proceedings of the IEEE International Conference on Computer Vision, Santiago, Chile, 7–13 December 2015; pp. 4489–4497.
36. Ding, X.; Zhang, X.; Ma, N.; Han, J.; Ding, G.; Sun, J. Repvgg: Making vgg-style convnets great again. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Nashville, TN, USA, 20–25 June 2021; pp. 13733–13742.
37. Suo, D.; Wei, X.; Wei, D. Gesture Recognition of Subway Drivers Based on Improved Dense Trajectory Algorithm. In Proceedings of the 2021 33rd Chinese Control and Decision Conference (CCDC), Kunming, China, 22–24 May 2021; pp. 1554–1559.
38. Zhang, Q.L.; Yang, Y.B. Sa-net: Shuffle attention for deep convolutional neural networks. In Proceedings of the ICASSP 2021–2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Toronto, ON, Canada, 6–11 June 2021; pp. 2235–2239.
39. Hu, J.; Shen, L.; Sun, G. Squeeze-and-excitation networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 7132–7141.
40. Yang, F. A Multi-Person Video Dataset Annotation Method of Spatio-Temporally Actions. *arXiv* **2022**, arXiv:2204.10160.
41. Jocher, G.; Changyu, L.; Hogan, A.; Yu, L.; Rai, P.; Sullivan, T. Ultralytics/yolov5: Initial Release. 2020. Available online: <https://zenodo.org/records/3908560> (accessed on 1 February 2023).
42. Dutta, A.; Gupta, A.; Zissermann, A. VGG Image Annotator (VIA). 2016. Available online: <https://www.robots.ox.ac.uk/~vgg/software/via/> (accessed on 2 February 2023).

**Disclaimer/Publisher’s Note:** The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.