Hindawi
Advances in Civil Engineering
Volume 2023, Article ID 1959962, 12 pages
https://doi.org/10.1155/2023/1959962

Research Article

Safety Helmet Wearing Detection Based on Jetson Nano and Improved YOLOv5

Zaihui Deng, Chong Yao, and Qiyu Yin

Wuhan Textile University, Wuhan 430200, China

Correspondence should be addressed to Chong Yao; 2012045@wtu.edu.cn

Received 17 November 2022; Revised 18 March 2023; Accepted 5 April 2023; Published 11 May 2023

Academic Editor: Suraparb Keawsawasvong

Aiming to address the current shortcomings of the existing safety helmet wearing detection algorithms, including a slow reasoning speed, a large model size, and high hardware requirements, this study proposes an improved safety helmet wearing detection network named YOLOv5-SN, which is suitable for embedded deployment on Jetson Nano. First, the backbone of the YOLOv5 network is modified using the model lightweight method introduced by ShuffleNetV2. Next, the model size and number of parameters in the trained model are reduced to about one-tenth of those of the YOLOv5 network, and the reasoning speed is improved by 72 ms/f when tested on Jetson Nano. Then, the modified model is optimized using the quantification and layer fusion operations, further reducing the computing power and accelerating the reasoning speed. Finally, the YOLOv5-SN network is obtained by improving the YOLOv5 model, and the optimized model is deployed on Jetson Nano for testing. The average reasoning speed of the YOLOv5s-SN network reaches 32.2 ms/f, which is 84.7 ms/f faster compared to that of the YOLOv5s model. This demonstrates an obvious advantage of the proposed model in reasoning speed compared to the existing YOLOv5 models. Finally, the proposed model can perform real-time and effective target detection on the Jetson Nano embedded terminal.

1. Introduction

In the rapid development of the modern world, much infrastructure, including buildings, subways, and electric power systems, has been constructed every year. However, any accident at the construction site can be fatal and catastrophic. According to 2020 Notice on Production Safety Accidents of Municipal Housing Projects issued by the Ministry of Housing and Urban-Rural Development of China [1], a total of 689 safety accidents took place at the construction sites of municipal housing projects across China in 2020, resulting in 794 deaths. Among these fatal accidents, 407 were caused by workers or objects falling from a high height, accounting for 59.07% of the total. Thus, a large portion of falling-related accidents could be attributed to falling objects. In these accidents, the protection provided by a safety helmet is beyond doubt. In the study of [2], it was concluded that among the workers who had fallen victim to fatal injury in safety accidents, 47.3% of them were not wearing a helmet at the moment of the accident. The existing manual supervision methods mainly include viewing a monitoring video and patrolling. However, these methods have the disadvantages of high labor cost, low efficiency, high probability of missing targets, and poor real-time performance. Due to the rapidly increasing population of construction workers and the lack of safety education for workers, nonconforming behaviors related to helmet-wearing have not been uncommon. Some of the workers wear safety helmets only when a safety officer is present, and some workers often inadvertently take off safety helmets during work hours. Therefore, when an object falls from a high position, it could inflict irreparable injury to the workers without a helmet. With the development of artificial intelligence (AI) technology, different helmet-wearing detection systems have been deployed at construction sites and have become an effective solution to the safety supervision problem. A helmet-wearing detection system can be used to detect workers entering the construction site efficiently and accurately without interruption. Once a worker without a helmet appears in the monitoring area, the detection

system can immediately identify him/her and sends off an alert. Upon receiving the alert, the safety officer at the site can address the nonconforming behaviors regarding helmet wearing timely, which can greatly reduce the casualties in accidents of a falling object and is of great significance to the green and safe development of the construction industry.

Recently, computer vision and deep learning technologies have advanced rapidly. The detection algorithms can be roughly divided into two types, namely, two-stage target detection algorithms and one-stage target detection algorithms. The well-known two-stage target detection algorithms include R-CNN [3], faster R-CNN [4], and mask R-CNN [5]. These algorithms have high detection accuracy, but their detection speed is relatively slow. Therefore, they are not suitable to be deployed on embedded terminals and perform real-time target detection tasks. On the contrary, one-stage target detection algorithms can directly predict the probability of the target category and the coordinates of the target position using regression-based methods and obtain the final result in one go. These algorithms have the advantage of high detection speed. Well-known one-stage target detection algorithms include YOLO algorithms (the latest version is YOLOv5) [6–8], single shot multibox detector (SSD) algorithm [9], and RetinaNet [10]. The YOLOv5 algorithms can achieve higher detection accuracy than the previous generations of YOLO algorithms while ensuring high detection speed. Therefore, they are well qualified for real-time safety helmet-wearing detection in various industrial scenes.

In the field of target detection, including safety helmet wearing detection, many methods have been proposed for improving the YOLO algorithms' performances. Benjumea et al. [11] improved the YOLOv5 model to address the problem of low performance of the YOLO model in detecting small targets. By modifying the model structure, the authors obtained a series of models with different scales (YOLO-Z series). The YOLO-Z series algorithms had good performance in small target detection, thus suiting well the task of safety helmet wearing detection. However, the improvement in the mean average precision (mAP) was achieved at the cost of inference speed. Kaushal [12] added additional detection and convolution layers to the original YOLO model to design the Rapid-YOLO algorithm and introduced an improved loss function to increase the detection accuracy of the proposed model. Jin et al. [13] improved the YOLOv5 algorithm in terms of the efficiency of helmet-wearing detection and prevention of the error accumulation effect on the detection accuracy. The authors first used the k-means++ algorithm to improve the size-matching degree of the prior anchor box and then introduced the depthwise coordinate attention mechanism in the backbone network to enable the model to learn the weight of each channel independently. These improvements strengthened the information dissemination between features, improving the model's ability to distinguish the foreground from the background. In this way, the average detection accuracy of the improved model surpassed that of the original YOLOv5 algorithm. Han et al. [14] proposed a target detection algorithm based on the SSD to improve the detection accuracy of the existing helmet wearing detection methods. This algorithm used a spatial attention mechanism for low-level features and a channel attention mechanism for high-level features, which further refined the feature information of the target area. In addition, a feature pyramid and a multi-scale sensing module are adopted to improve algorithm robustness for targets of different scales, and a method for achieving an adaptive adjustment of the scale distribution between the layers of the anchor box based on the feature map size was developed.

However, fewer studies have proposed improved helmet-wearing detection models with a faster reasoning speed. Amudhan and Sudheer [15] proposed a convolution neural network model with low computational complexity, which can extract low-level features from the shallow layer and transfer them to the deep layer, thus improving small-target detection accuracy and reasoning speed. When deployed on Jetson Nano, this model can outperform the YOLOv3-tiny, YOLOX, and some other models. Zhao et al. [16] proposed an improved YOLO-S model, where model complexity was reduced by modifying the backbone network and removing redundant channels, thus improving the reasoning speed.

If the existing detection models are used in real-time monitoring, it is typically necessary to transfer the monitoring video to a GPU that can provide the necessary computing power for object detection. However, the hardware cost of such a monitoring system is very high, and a severe network delay might occur during video transfer. Still, if object detection could be carried out directly at the embedded terminal, the deployment cost would be significantly reduced. In addition, such a monitoring system could be easily expanded, and the video transfer delay could be avoided. Therefore, it is of great importance to study how to deploy a helmet-wearing detection model on embedded terminals. Namely, deployment of the existing models on embedded terminals for helmet-wearing detection has been limited by several problems. First, the existing detection models are usually based on a large and complex detection network with a large number of parameters and thus have a slow reasoning speed. Therefore, it will be impossible to realize real-time monitoring by deploying such detection models on embedded platforms with low computing power. As an excellent first-stage model, the YOLOv5 model has high detection accuracy, but its performance is limited when deployed on platforms with low computing power. Even the YOLOv5s model, which is currently the fastest model in the YOLOv5 family, can achieve a speed of only 116 ms/f (about 9 fps) on Jetson Nano. Moreover, the YOLOv5m model, which has higher complexity than the YOLOv5s models, can only operate at a speed of lower than 5 fps; the YOLOv5l model is too complex to be deployed on platforms with low computing power. Therefore, it is necessary to develop a lightweight YOLOv5 model to meet the current deployment requirements of platforms with low computing power.

Most of the above-mentioned improvement measures aim to improve model accuracy by increasing detection network complexity, which affects the reasoning speed.

Although recent research has developed much effort to improve the reasoning speed, the proposed models still cannot meet the requirements of real-time monitoring when deployed on embedded terminals. Currently, there have been fewer studies on lightweight YOLOv5 models, but this topic should be further researched to deploy a detection network on embedded terminals.

In order to solve the aforementioned problems, this paper proposes a lightweight helmet wearing detection model named YOLOv5-SN, which is suitable for embedded terminal deployment on Jetson Nano.

The main contributions of this study are as follows:

(1) Images from safety helmet wearing datasets (SHWDs) are used, which is a standard dataset for helmet-wearing target detection. These images are resized to make their size uniform and are used to train the YOLOv5 and YOLOv5-SN models to achieve the performance levels of the YOLOv5 series models on the experimental dataset and quantify the improvement achieved by the proposed YOLOv5-SN models.

(2) The YOLOv5 series models are improved using the ShuffleNetV2 method, thus significantly reducing the parameter number and model complexity. The experimental results show that the parameter number and model size of the YOLOv5-SN models are approximately one-tenth of those of the YOLO models. The improved models are tested on the Jetson Nano embedded terminal, and the results show that the YOLOv5-SN model can process a single frame with a reasoning speed of 72 ms/f, which is faster than that of YOLOv5s.

(3) Optimization operations, including quantization and layer fusion, are performed on the trained YOLO models and YOLOv5-SN model to deploy them effectively on Jetson Nano. The optimized models are tested on Jetson Nano embedded terminal. The test results show that after model optimization, the reasoning speed of the YOLOv5-SN model is 84.7 ms/f faster than that of the YOLOv5s model, indicating that the YOLOv5-SN model can achieve better performance than the YOLOv5s in processing real monitoring images. Therefore, this model is suitable to be deployed on embedded terminals for real-time helmet-wearing detection.

2. Dataset and Preprocessing

The dataset used in the experiment of this study was SHWD. It contained a large amount of data for helmet-wearing detection and head detection, namely, 7,581 images were obtained from Google and Baidu. The software tool LabelImg was used to label objects in images. The label format was PASCAL VOC. The images included 9,044 marked helmet-wearing targets and 111,515 marked normal head targets. The SHWD helmet dataset contained large-, medium-, and small-sized helmet-wearing targets acquired under different values of density (including multiple scales), illumination, and occlusion; also, some targets were captured with blur background. Before model training, the dataset was randomly divided into training, verification, and test sets according to the ratio of $7:2:1$.

The SHWD images were preprocessed before model training; particularly, all images were resized to 640×640 pixels. Figure 1 shows several preprocessed images in the dataset.

3. Optimization and Deployment of Safety Helmet Wearing Detection Model

3.1. Algorithm-Related Work

3.1.1. YOLOv5 Algorithm. The construction sites are characterized by a complex environment and a multitude of different obstacles. In addition, during the night, the lighting is poor, and shadows increase the difficulty of visual detection. As a result, performing helmet-wearing detection at construction sites requires high detection accuracy. If a hardware platform of a detection system is the Jetson Nano embedded terminal, the reasoning speed of the detection algorithm must satisfy certain requirements. Therefore, the requirements for accuracy and speed should be considered comprehensively in the field of helmet-wearing detection.

The one-stage target detection algorithms extract data features directly from image data to predict types and positions of targets, and bounding boxes can be predicted right after the image data are input in a detection model. Therefore, these algorithms have a relatively high processing speed and are suitable for helmet-wearing detection tasks. As well-known one-stage target detection algorithms, the YOLO series algorithms have high performance in terms of reasoning speed and accuracy. Compared to the early YOLO models, such as the YOLOv3 and YOLOv4 models, the YOLOv5 model has much higher detection speed and accuracy. Therefore, this study uses the YOLOv5 model to develop the YOLOv5-SN detection algorithm, which can achieve excellent performance when deployed on Jetson Nano embedded terminal.

The overall structure of the YOLOv5 model is presented in Figure 2. The YOLOv5 model consists of the input module, backbone, neck module, and output module. The basic units of these modules include Focus, CBL, CSP, and SPP. The focus performs slicing operations; CBL includes convolution, batch normalization, and Leaky ReLU activation functions, and CSP includes residual unit (Resunit) and CBL component.

3.1.2. ShuffleNetV2 Algorithm. The ShuffleNet [17] is an efficient convolutional neural network model proposed by Megvii Technology. This model aims to reduce the computational complexity of deep networks so that they can be deployed and operated on mobile devices.

Ma et al. [18] proposed the ShuffleNetV2 for lightweight networks adopting four criteria, which are as follows: equal channel width minimizes the memory access cost (MAC), excessive group convolution operations increase the MAC value, network fragmentation reduces the parallelism degree, and the element-wise operations are non-negligible.

Figure 1: Examples of safety helmet wearing images after data preprocessing.

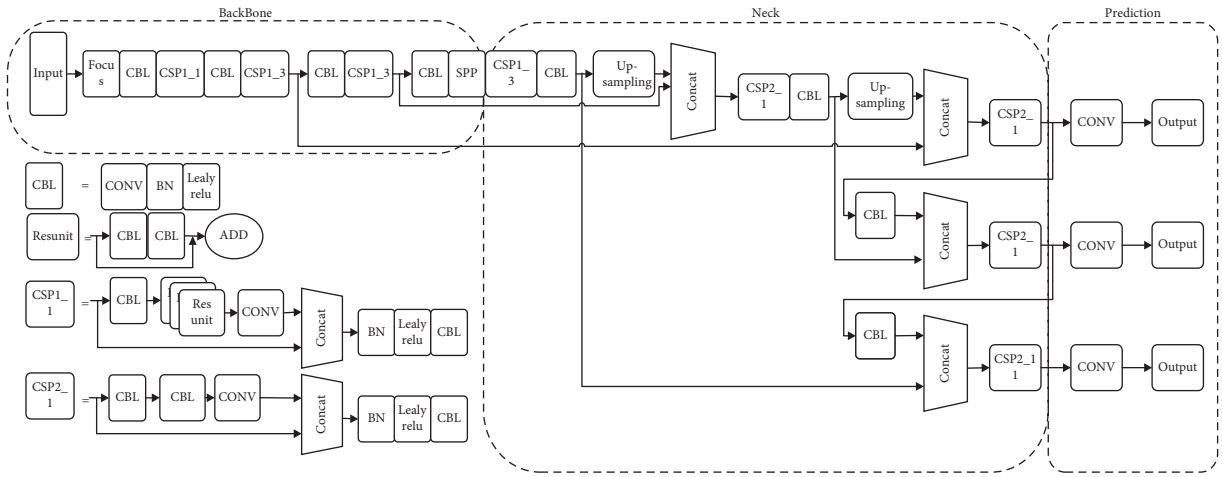

Figure 2: The overall structure of the YOLOv5 model.

The group convolution mentioned in the lightweight network criterion adopts the sparse channel connection mode, which reduces the parameter quantity and computational complexity compared to the conventional convolution. The group convolution process is shown in Figure 3.

In conventional convolution, channels are densely connected, so the entire input data (i.e., all channels) undergo the convolution operation together to generate a feature map. Suppose the number of convolution kernels is **N**, and the number of channels at the output is also **N**; then, the following equation can be obtained:

$$\mathbf{P} = \mathbf{K}^2 \times \mathbf{C} \times \mathbf{N}, \tag{1}$$

where **P** is the parameter quantity, **K** is the convolution kernel size, and **C** is the number of input feature channels.

In group convolution, **C** input feature channels are divided into **G** groups, and the convolution kernels are divided into **G** groups accordingly, so each convolution kernel has **C**/**G** channels. Then, group convolution is performed, and the output data of **G** groups are merged to obtain a feature map of **N** channels, which can be expressed as follows:

$$\mathbf{P} = \mathbf{K}^2 \times \frac{\mathbf{C}}{\mathbf{G}} \times \mathbf{N}, \tag{2}$$

where **G** is the number of groups.

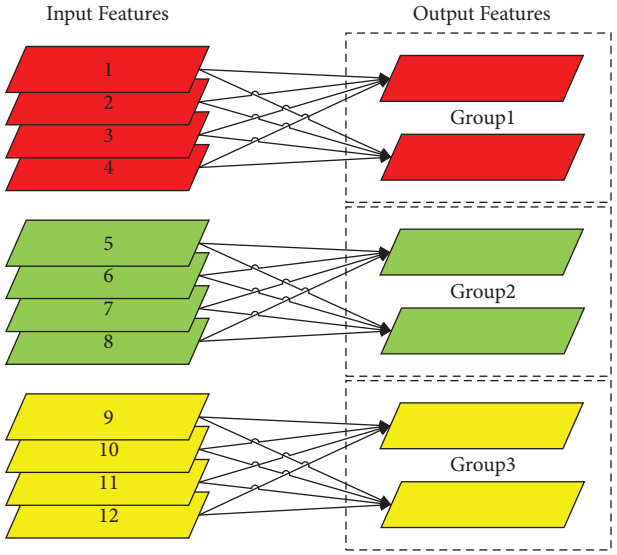

Figure 3: The group convolution.

Based on (1) and (2), the number of parameters and computational complexity of group convolution are 1/**G** of those of conventional convolution, indicating that the group convolution operation can significantly reduce both the parameter quantity and the computational complexity. In addition, group convolution can address the overfitting problem to a certain extent. However, the feature extraction capability of group convolution is comparatively lower than that of conventional convolution due to a lack of information exchange between different groups.

To solve this problem, the channel shuffle operation has been introduced in the ShuffleNet model. The principle of channel shuffle operation is shown in Figure 4.

The channel shuffle operation shuffles features extracted in groups, mixing the information obtained from different groups and realizing information exchange between different channels, thus ensuring sufficient information fusion of all channels. This operation addresses the lack of information exchange in group convolution without increasing computational complexity.

The workflow of the ShuffleNetV2 model is shown in Figure 5. As shown in Figure 5, after the feature mapping data are input into the ShuffleNetV2 model, the channel segmentation operation is performed on the input features. Considering that too many branches can result in an overly fragmented network structure, thus reducing the network speed, the features are divided into two branches by the channel split operation. The left branch remains intact, serving as a flag group, and the right branch is composed of three convolutions with equal channel widths, which can minimize the memory access cost. Unlike the ShuffleNetV1 model, where excessive group convolution can increase the memory access cost, the ShuffleNetV2 model does not adopt group convolution for these three convolutions; instead, a simple 1×1 convolution is used in the ShuffleNetV2 model. After the convolution operation is completed, the Concat operation is performed to concatenate the two

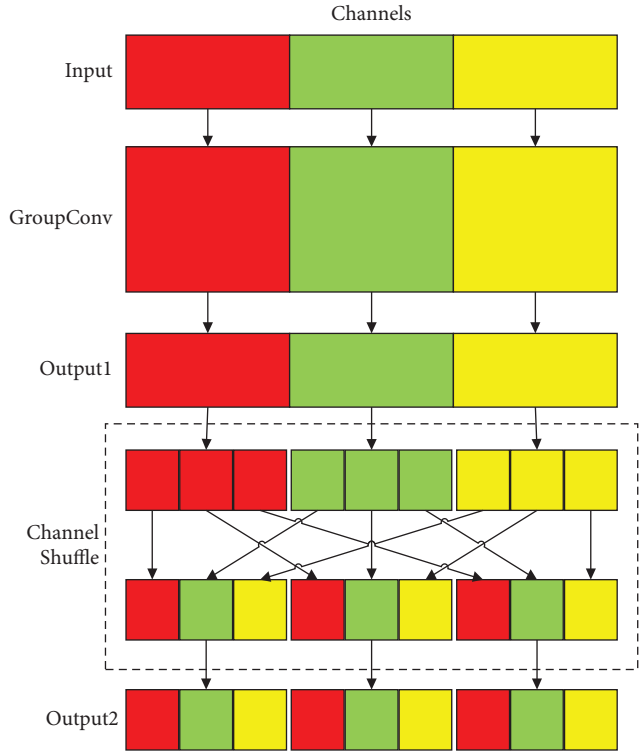

Figure 4: The channel shuffle process.

branches' output data, which ensures that the number of channels remains unchanged. Finally, the channel shuffle operation is performed before outputting the data to enable information exchange between the two branches.

3.1.3. Improved YOLOv5-SN Algorithms. The YOLOv5 algorithm has certain deficiencies, including a large number of parameters, a long reasoning time, a bulky model, and difficult deployment. In this study, the backbone of the ShuffleNetV2 model is used as a backbone in the improved YOLOv5-SN algorithm to achieve the goal of a lightweight network structure without affecting the feature extraction capability significantly. This design reduces the model size while ensuring a certain level of accuracy, thus meeting the lightweight network criterion of the ShuffleNet model.

The structure of the YOLOv5-SN model is shown in Figure 6, where it can be seen that it is mainly composed of four parts: input module, backbone, neck module, and prediction module. The input module is responsible for inputting the image data of helmet-wearing targets into the model; the backbone extracts the helmet-wearing features; the neck module realizes the fusion of helmet-wearing features; the prediction module is responsible for predicting helmet-wearing targets.

This study constructs three versions of the YOLOv5 model, as well as their corresponding YOLOv5-SN models, which are used for comparison. The performance improvements achieved by the improved YOLOv5-SN algorithms compared to the original YOLO models are analyzed regarding the reasoning speed, number of parameters, and model size.

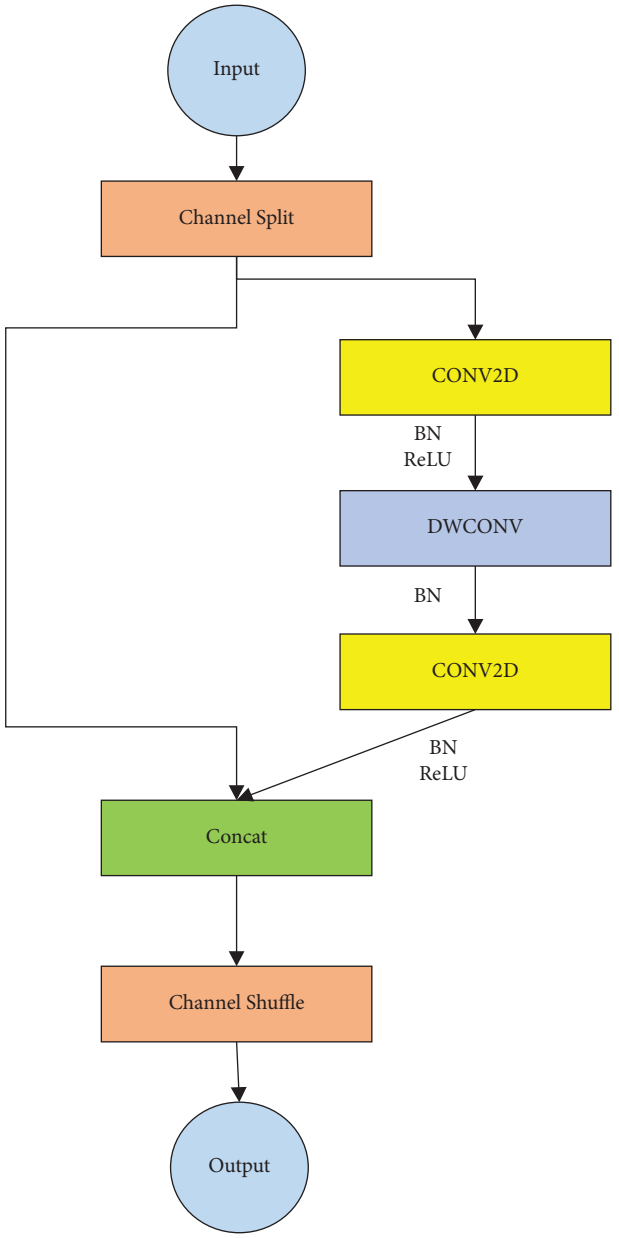

Figure 5: The workflow of the ShuffleNetV2 model.

3.2. Model Optimization and Deployment

3.2.1. Model Optimization. Although deploying the improved YOLOv5-SN models directly on the embedded terminals can provide better performance compared to the original YOLO models, the reasoning speed of the YOLOv5-SN models still does meet the requirements of real-time processing. To solve this problem, this study reduces network complexity by performing optimization operations, thus improving the reasoning speed after the deployment. The model optimization operations mainly include model quantization and layer fusion.

The model quantization refers to the process of transforming the weights, activation values, and other parameters of a trained deep neural network model from high accuracy to low accuracy. For instance, from 32-bit float type to 8-bit int type. The expected accuracy of the model after the quantization operation is close to that before the quantization operation.

Performing quantization operation on a model can reduce the occupation of memory and storage spaces as well as shortens the time required to access the memory, which optimizes the calculation time significantly. Moreover, the model quantization reduces the power consumption in reasoning; for instance, the energy consumption of an 8-bit int model is quite different from that of a 32-bit float model. From the viewpoint of a processor, many processors exhibit higher efficiency in processing integer data than float data. In summary, the model quantization operation can effectively improve the reasoning speed of a model, meeting the real-time detection requirement and thus creating suitable conditions for deploying the model on Jetson Nano embedded terminals.

The purpose of layer fusion is to fuse network layers. Aiming to improve the detection accuracy of deep learning networks, recent studies have tended to design highly complex network models consisting of a large number of layers, including both convolution layers and fully-connected layers. However, during reasoning, the forward propagation needs to pass through every layer; so in a complex network, a large number of convolution and fully connected layers need to perform CUDA core calls and tensor reading or writing, which significantly slows down the reasoning process. The layer fusion operation aims to eliminate some of the unused layers to avoid unnecessary calculations and merge multiple layers into a single layer, thereby improving reasoning efficiency. The layer fusion process used in model optimization is shown in Figure 7, where it can be seen that the original convolution, offset, and ReLU layers are fused into a new layer named the CBR layer. The layer fusion operation can speed up the forward propagation on GPU, thus improving reasoning efficiency.

However, the aforementioned optimization operations modify only the reasoning process of trained models but not the model structure, so the accuracy achieved during the training process is not affected by the optimization operation.

3.2.2. Model Deployment. In this study, a deep learning server was used to train the YOLO series models and their corresponding improved YOLOv5-SN models, and the processes of model optimization, deployment, and testing were performed on Jetson Nano embedded terminal.

First, the model was trained on the deep learning server, obtaining the model of the pt format. The pt format is a model format that is obtained after the model is trained using the Pytorch deep learning-based framework written in Python. Then, the trained model was transferred to the Jetson Nano development board for the first round of deployment. The speed and accuracy of the model were tested and recorded. Furthermore, the model was optimized on the Jetson Nano development board, and through layer fusion and model quantification operation, the format of the model

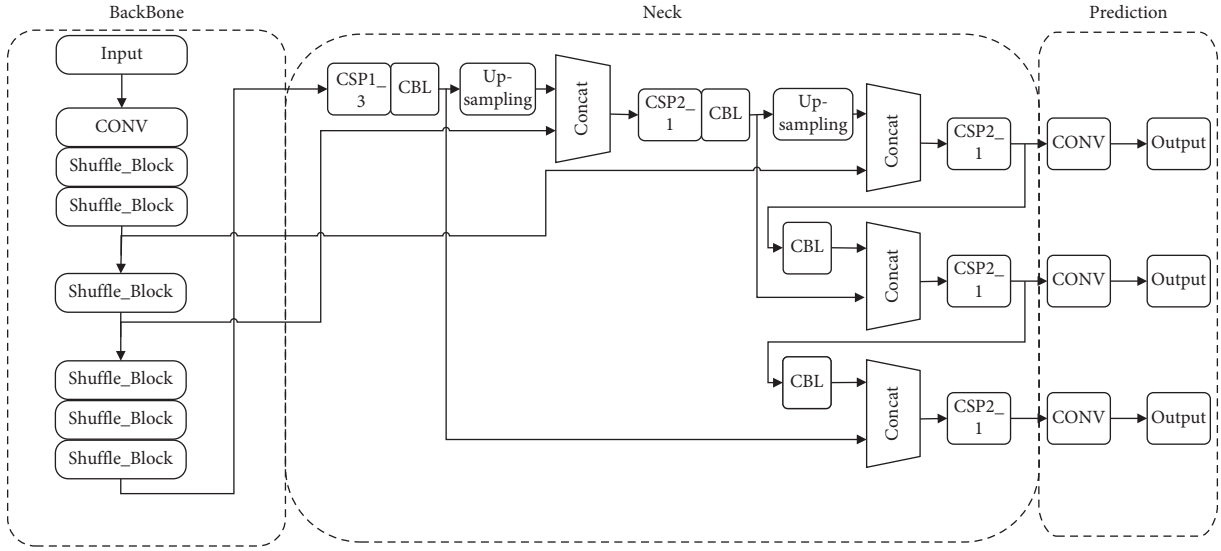

Figure 6: The overall structure of the YOLOv5-SN model.

was converted from the pt format (Pytorch framework) to the TensorRT format (engine). Finally, the converted model was again deployed on the Jetson Nano development board, and the speed and accuracy of the model were tested and recorded. The operation of the model deployed on the Jetson Nano development board is presented in Figure 8.

3.3. Model Evaluation Metrics. In this study, the metrics for model performance evaluation included single frame image reasoning time, precision, recall, parameter quantity, and model size. The precision was used to measure the accuracy of the model detection, and its calculation formula is given in (3). The recall rate was used to evaluate the comprehensiveness of the model detection, and it is expressed by (4), where TP represents the real case, FN is the pseudonegative case, and FP denotes the pseudopositive case.

$$\mathbf{P} = \frac{\mathbf{TP}}{\mathbf{TP} + \mathbf{FP}}, \tag{3}$$

$$\mathbf{R} = \frac{\mathbf{TP}}{\mathbf{TP} + \mathbf{FN}}. \tag{4}$$

4. Experimental Results Analysis

4.1. Training and Embedded Platform. The TeslaA100, 40 GB memory, Ubuntu 20.04 LTS 64-bit operating system, Python, and Pytorch deep learning-based framework were used for model training.

The model reasoning and model optimization were performed on the Jetson Nano development board. The GPU had the NVIDIA MAXWELL architecture and was equipped with 128 NVIDIA CUDA cores. The CPU was a four-core ARM CORTEX-A57 MPCORE, and the memory was 4 GB 64-bit LPDDR4. The running environment on the Jetson Nano included the Ubuntu18.04 operating system and Python3.6.

4.2. Training Parameters Setting. The model training followed the principle of the consistent settings of model parameters. The training cycle included 300 epochs, and the batch size was set to 64. The model was saved after each epoch, and the best model at the end of the training process was selected as the final model.

4.3. Training Model Results Analysis. The variations in the box loss and object loss values of the YOLOv5s, YOLOv5l, and YOLOv5m models and their improved YOLOv5-SN versions with the number of training epochs are presented in Figures 9 and 10. Although complete fitting of the loss value curves was not achieved within 300 epochs, the declining speeds of the loss values clearly indicated the tendency of the complete fitting.

The variations in precision and recall of the models with the number of epochs are presented in Figures 11 and 12, respectively. Usually, with an increase in the number of epochs, the model accuracy increases, and the loss value decreases. However, in this study, after 150 epochs, the values of precision and recall tended to be stable, indicating that further training could not improve the model performance.

Based on the comprehensive examination of variations in the models' accuracy, mAP, and recall rate values, the optimal weights of each model were selected from the results of 300 epochs as the final model training result and used for subsequent performance assessment. The test results are shown in Table 1. Compared with the original YOLOv5 models, the improved YOLOv5-SN models were much better in terms of the number of parameters and model size. It should be noted that parameter quantity refers to the total number of model parameters, and in a convolution neural network, the parameter quantity is mainly defined by weights in each convolution layer. The parameter quantity can affect the memory occupation, the speed of model initialization, and model size, which can

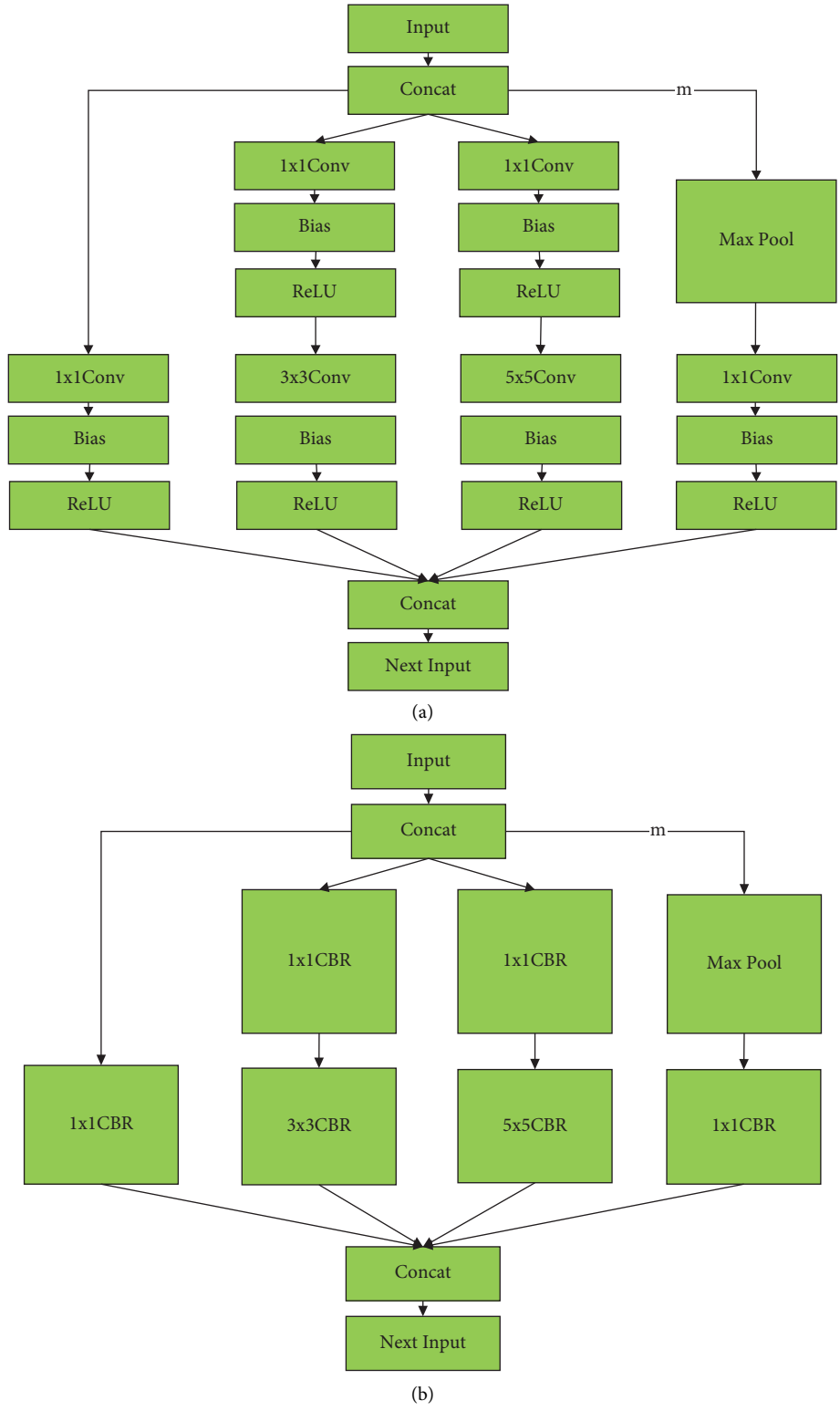

Figure 7: The model layer fusion operation. (a) Before model layer fusion operation. (b) After model layer fusion operation.

further affect model performance in reasoning. For the YOLOv5s model, the parameter quantity of the original YOLOv5s model was 7.01 M, while the parameter quantity of the YOLOv5s-SN (improved version) was 0.84 M, which was only about 12% of that of the YOLOv5s model. The reduction in parameter quantity led to the reduction in the model size. The size of the YOLOv5s-SN model was 2.0 MB, which was only 13.8% of that of the YOLOv5s model of 14.4 MB. Similar improvements were achieved for the other improved YOLOv5 models. The sharp

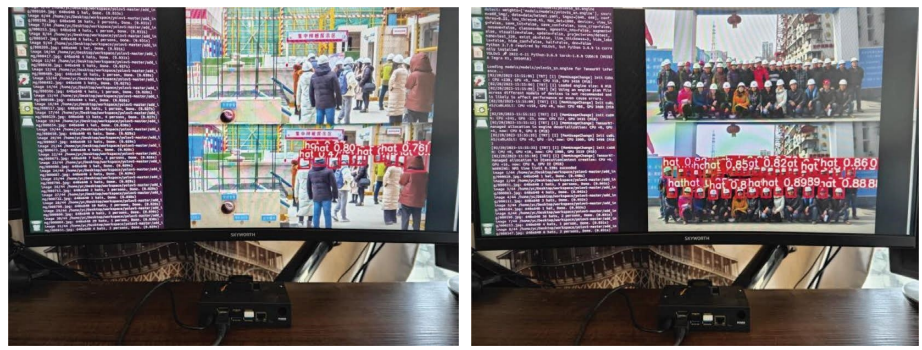

Figure 8: The deployment on Jetson Nano.

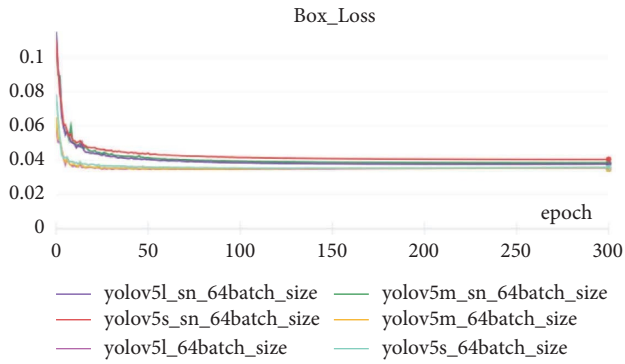

Figure 9: The results of the bounding box.

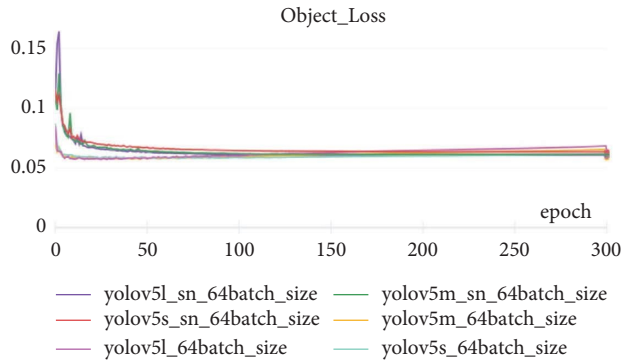

Figure 10: The results of the bounding object loss.

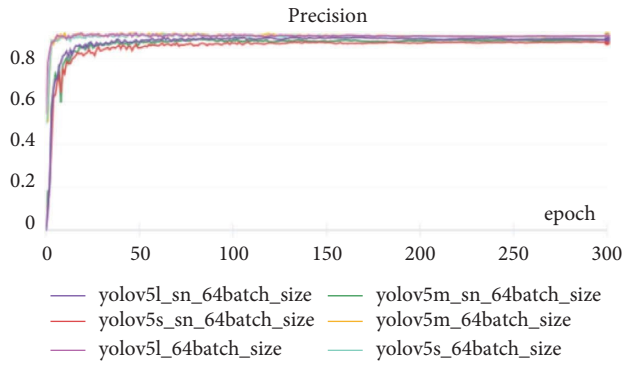

Figure 11: Precision results of the models.

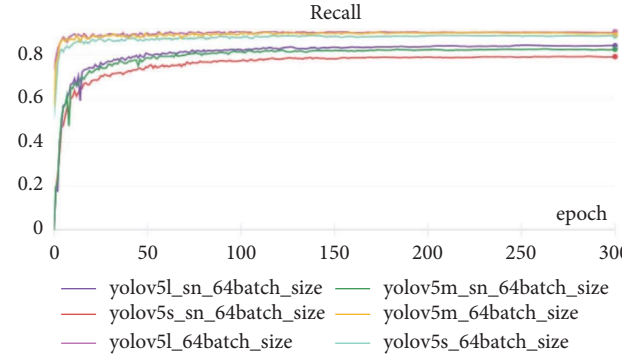

Figure 12: Recall results of the models.

decrease in parameter quantity led to a decrease in network complexity and the reduction of the convolution network weight number, which reduced the computational complexity and made model deployment on platforms with low computing power easier. The precision decreased by roughly 3% on average, and the recall was reduced by 7%–8% on average. Thus, the lightweight design of the improved YOLOv5-SN algorithms met the real-time requirement.

4.4. Visualization of Test Results. After the optimal model obtained by lightweight improvement and model optimization was deployed on Jetson Nano, it was used to perform helmet-wearing detection under different conditions. The prediction result is presented in Figure 13, where tag "hat" indicates that a helmet was correctly worn, and tag "person" indicates that the person did not wear a helmet. The confidence score of the classification was attached to the tail end of each classification result. In Figure 13, it can be seen that the proposed detection system could accurately detect workers in the video image, as well as the helmet-wearing status of each worker.

The actual detection performance of the YOLOv5s-SN model for different targets in different scenes is presented in Figure 14, where it can be seen that the YOLOv5s-SN model achieved good detection performance under the conditions of poor detection angle, target occlusion, dense small targets, and low illumination.

Table 1: The results of model performance.

Models	Precision	Recall rate	Parameter (million)	Model size (MB)
YOLOv5s	90.5	88.7	7.01	14.4
YOLOv5s-SN	87.7	79.2	0.84	2.0
YOLOv5m	91.5	89.4	20.85	42.2
YOLOv5m-SN	88.8	82.6	2.02	4.4
YOLOv5l	90.5	90.7	46.11	92.8
YOLOv5l-SN	89.3	84.3	3.79	8.0

Figure 13: Detection results obtained on images of a real construction site.

(a) (b) (c)

(d) (e) (f)

Figure 14: The detection results of the YOLOv5-SN model in different scenes. (a) Poor detection angle. (b) Dense small targets. (c) Target occlusion. (d) Back-showing dense small targets. (e) Single target. (f) Low-light targets.

4.5. Test Results and Analysis after Model Deployment. Based on the previous analysis, it can be concluded that improving a detection model by performing network lightweight and model optimization could significantly accelerate the reasoning speed of the model. To prove the effectiveness of network lightweight and model optimization, an ablation experiment was conducted. First, the YOLOv5 series models were improved by performing network lightweight, and the YOLOv5-SN series models were developed and deployed on Jetson Nano for testing. The test results are shown in Table 2. For the YOLOv5s model, the reasoning speed of the original YOLOv5s model was 116.9 ms/f, and the reasoning speed of the improved YOLOv5s-SN model was

Table 2: Detection performance of the models improved by the network lightweight improvement method.

Models	Model format	Processing speed (ms/f)	Precision (%)	Recall rate (%)
YOLOv5s	pt	116.9	90.5	88.7
YOLOv5s-SN	pt	44.9	87.7	79.2
YOLOv5m	pt	277.1	91.5	89.4
YOLOv5m-SN	pt	64.6	88.8	82.6
YOLOv5l	pt	500.9	90.5	90.7
YOLOv5l-SN	pt	93.1	89.3	84.3

Table 3: Detection performance of the models improved by the model optimization method.

Models	Model format	Processing speed (ms/f)	Precision (%)	Recall rate (%)
YOLOv5s	pt	116.9	90.5	88.7
YOLOv5s (after optimization)	Engine	67.3	90.4	88.8
YOLOv5m	pt	277.1	91.5	89.4
YOLOv5m (after optimization)	Engine	171.2	90.4	90.4
YOLOv5l	pt	500.9	90.5	90.7
YOLOv5l (after optimization)	Engine	318.5	90.6	90.6

Table 4: Detection performance of the models that have been improved using both the two improvement methods.

Models	Model format	Processing speed (ms/f)	Precision (%)	Recall rate (%)
YOLOv5s-SN	pt	44.9	87.7	79.2
YOLOv5s-SN (after optimization)	Engine	32.2	88.8	78.0
YOLOv5m-SN	pt	64.6	88.8	82.6
YOLOv5m-SN (after optimization)	Engine	56.5	88.0	83.2
YOLOv5l-SN	pt	93.1	89.3	84.3
YOLOv5l-SN (after optimization)	Engine	83.9	88.6	84.7

44.9 ms/f, which was an improvement of 72 ms/f. Improvements of a similar extent were also achieved for the other YOLOv5 models. The results in Table 2 demonstrate that the network lightweight improvement method was effective.

Next, the YOLOv5 models were improved using the model optimization method; namely, the models with the pt format were optimized into the engine-format models and deployed on Jetson Nano for testing. The test results are shown in Table 3. For the YOLOv5s model, the reasoning speed of the original YOLOv5s model was 116.9 ms/f, and the reasoning speed of the improved YOLOv5 engine model increased to 67.3 ms/f, which represented an improvement of 49.6 ms/f. It is worth noting that the improvement achieved by the model optimization method had little effect on the detection accuracy and recall rate of the model.

Finally, the YOLOv5 models were improved using both the network lightweight improvement method and the model optimization method. The improved models were deployed on Jetson Nano for testing, and the obtained test results are shown in Table 4. As shown in Table 4, the improved YOLOv5s-SN model with the engine format achieved a high reasoning speed of 32.2 ms/f, surpassing the reasoning speed of the YOLOv5s model of 116.9 ms/f by 84.7 ms/f. The results in Table 4 demonstrate that combining the two improvement methods could provide better model performance than when the two methods were used separately.

5. Conclusions

This study introduces an improved safety helmet-wearing detection model named YOLOv5-SN, aiming to address the shortcomings of the existing YOLOv5 models, including a large number of model parameters, slow reasoning speed, and redundant network structure. The network lightweight method proposed along with the ShuffleNetV2 model is adopted to reduce the parameter quantity and model size. Considering that the reasoning speed of a model slows down after the model is deployed on a Jetson Nano terminal because of unused layers, compilation, and increased calculation time, the quantization and layer fusion operations are performed on the trained model to fuse the redundant layers and quantify the model precision. The results show that when the models improved by the two optimization methods are deployed on the Jetson Nano terminal, the helmet detection speed is significantly accelerated. In the model improvement process, the reduction of convolution layer number, parameter quantity, and network complexity causes the detection accuracy to decline to a certain extent. Finally, the models improved using the proposed method can meet the real-time detection requirements. In this study, the real-time helmet-wearing detection at construction sites is realized by deploying the target detection model on an embedded terminal, which contributes to the intelligent level improvement of construction sites.

One of the future research directions could be to improve model detection accuracy by using a knowledge distillation method. In addition, more attention could be paid to target features by introducing an attention mechanism to minimize false-positive and false-negative errors.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the Natural Science Foundation of Hubei Province of China under Grant no. 2015CFB652.

References

[1] M. O. H. U. R. D. China, "Report on production safety accidents of housing and municipal engineering in 2020," 2021, https://www.mohurd.gov.cn/gongkai/fdzdgknr/zfhcxjsbwj/202210/20221026_768565.html.

[2] Y.-S. Ahn, "Comparison of unintentional fatal occupational injuries in the Republic of Korea and the United States," *Injury Prevention*, vol. 10, pp. 199–205, 2004.

[3] R. Girshick, J. Donahue, and T. Darrell, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 580–587, Columbus, OH, USA, June 2014.

[4] R. Girshick, "Fast R-CNN," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1440–1448, Cambridge, MA, USA, June 2015.

[5] K. He, G. Gkioxari, and P. Dollar, "Mask R-CNN," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, pp. 386–397, 2020.

[6] J. Redmon, S. Divvala, and R. Girshick, "You only look once: unified, real-time object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 779–788, Las Vegas, NV, USA, June 2016.

[7] J. Redmon and A. Farhadi, "YOLOv3: an incremental improvement," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 89–95, Salt Lake City, UT, USA, June 2018.

[8] A. Bochkovskiy, C. Y. Wang, and H. Liao, "YOLOv4:Optimal speed and accuracy of object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition(CVPR)*, Seattle, WA, USA, June 2020.

[9] W. Liu, D. Anguelov, and D. Erhan, *SSD: Single Shot Multibox Detector",European Conference on Computer Vision*, Springer, Cham, 2016.

[10] T. Y. Lin, P. Goyal, and R. Girshick, "Focal loss for dense object detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, pp. 2999–3007, 2017.

[11] A. Benjumea, I. Teeti, and F. Cuzzolin, "YOLO-Z: improving small object detection in YOLOv5 for autonomous vehicles," *arXiv: 2112.11798*, 2021.

[12] M. Kaushal, "Rapid -YOLO: a novel YOLO based architecture for shadow detection," *Optik*, vol. 260, Article ID 169084, 2022.

[13] Z. Jin, P. Qu, and C. Sun, "Dwca-Yolov5: An improve single shot detector for safety helmet detection," *Journal of Sensors*, vol. 2021, Article ID 4746516, 12 pages, 2021.

[14] G. Han, M. Zhu, and X. Zhao, "Method based on the cross-layer attention mechanism and multiscale perception for safety helmet-wearing detection," *Computers & Electrical Engineering*, vol. 95, 2021.

[15] A. N. Amudhan and A. P. Sudheer, "Lightweight and computationally faster Hypermetropic Convolutional Neural Network for small size object detection," *Image and Vision Computing*, vol. 119, 2022.

[16] H. Zhao, X. Tian, and Z. Yang, "YOLO-S: a new lightweight helmet wearing detection model," *Journal of East China Normal University*, vol. 2021, 2021.

[17] X. Zhang, X. Zhou, and M. Lin, "Shufflenet: an extremely efficient convolutional neural network for mobile devices," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 6848–6856, Salt Lake City, UT, USA, June 2018.

[18] N. Ma, X. Zhang, and H. T. Zheng, "Shufflenet v2:Practical guidelines for efficient cnn architecture design," in *Proceedings of the European conference on computer vision (ECCV)*, pp. 116–131, Munich, Germany, September 2018.