Vol.29 计算机集成制造系统 ISSN
No. 7 **Computer Integrated Manufacturing Systems** 1006-5911

Semantic Segmentation Model for Edge Devices Based on Model Compression and Model Acceleration

Tzu-Lien Tzou[1], Chieh-Ying Lai[2], Meng-Fong Tsai[3], Wen-Chieh Lee[2], Chung-Ho Huang[4], Meng-Hsiun Tsai[2*]
[1]Occupational Safety and Health Administration, Taiwan (R.O.C)
[2,3]National Chung Hsing University. Taiwan (R.O.C)
[4]National Taipei University of Technology, Taiwan (R.O.C)

Abstract:
This study aims to adopt different models and apply various model compression and acceleration techniques to improve the performance of segmentation and computing efficiency of the semantic segmentation models for small objects such as safety guardrails. Due to the scarcity of data, caused by restrictions of construction site safety regulations and difficulty of data labeling, this study adopts the method of data augmentation to assist the training process of the model. In addition, in response to the model with hardware performance and a large amount of model parameters, it is found that using input images of different sizes for different models can ensure its segmentation performance and successfully perform guardrail identification according to the experiments. As a result, all models achieve above 0.54 in IoU. In this study, Ghost Module is chosen as the acceleration method, and experiments have confirmed that this acceleration method can help improve the computing efficiency and allow the performance of segmentation of the model up to an IoU of 0.65. Although running on edge devices cannot achieve the level of real-time segmentation, after model acceleration, the time required for an image is still significantly decreased by more than 110 percent. Also, since the guardrail is a static object, there is no need for the fast identification frequency. Finally, in order to further reduce the computational complexity of the model, this study uses model pruning to compress the overall model size. According to the results of the experiments, it is found that there is indeed a problem of redundant weights in the model. After removing a certain degree of redundant weights by the L1 norm and adopting fine-tuning, it can effectively improve the model's ability to segment guardrails

Keywords: Semantic Segmentation, Model Acceleration, Model Compression, Artificial Intelligence.
DOI: 10.24297/j.cims.2023.7.4

1. Introduction

The construction industry is characterized by hazards in the workplace and variety of complex project types, leading to a high incidence rate of major occupational disasters compared to other industries. According to the Occupational Safety and Health Administration of the Ministry of Labor of Taiwan [1], the number of major occupational accidents in the construction industry is equivalent to in the sum of all other industries. These accidents are largely caused by human negligence, including workers unsafe behavior, inadequate use of personal protective equipment, and improper installation of guardrails and safety nets.

In the past, monitoring workers' clothing and behavior, and ensuring compliance with safety regulations at construction sites was done manually. However, it was labor-intensive and financially demanding, and there was often delayed and inaccurate information caused by the negligence of supervisors. Later on, sensing technologies, such as radio frequency identification (RFID), global positioning system (GPS), and ultra-wideband (UWB) were used to aid in automated real-time monitoring of construction sites [2-5]. Nevertheless, these methods were often expensive and prone to inaccurate information due to interference from metal products or electronic signals. With the advancement of artificial intelligence in computer vision, the automation and real-time monitoring of construction sites have gained attention from academics. Currently, computer vision technologies are being increasingly applied to the construction industry for occupational accident prevention. Utilizing surveillance video or filming equipment provides supervisors with faster and more comprehensive information and enables the preliminary identification of dangerous behaviors during operations, reducing the likelihood of occupational accidents.

Computer vision mainly includes image recognition, object detection, and image segmentation. In the construction industry, object detection is the most commonly used approach, particularly for detecting personal safety equipment [6]. Unlike object detection, there have been few cases of image segmentation being applied in the construction industry. Moreover, compared to other objects on the construction sites, safety guardrails are relatively small objects without a fixed format, causing uncertainty in their identification.

This study aims to improve the performance of guardrail detection by adopting different semantic segmentation models, utilizing model acceleration and compression techniques, and enhancing the efficiency of automated supervision in the field of construction.

2. Related Work

This study focuses on enhancing the performance of semantic segmentation models and reducing the computational cost by employing both model acceleration and compression techniques. This chapter will be divided into three sections. First, a review of deep learning approaches including semantic segmentation, model acceleration, and compression techniques will be presented. Secondly, the existing research on construction safety based on deep learning methods will also be reviewed. Finally, the target object of this study, safety guardrails, will be discussed in detail.

2.1. Deep Learning

Deep Learning [7] is a branch of Machine Learning, which is a type of feature learning based on Neural Networks (NNs). DNNs [8] performed linear transformations in order to automatically extract features that effectively represent the data, making them applicable to various domains. So far, there have been numerous deep neural network frameworks, which can primarily be categorized into two types: Convolutional Neural Networks (CNNs) [9] and Recurrent Neural Networks (RNNs) [10]. Among them, CNNs are widely used in computer vision tasks, such as image recognition, image classification, and object detection. RNNs, on the other hand, are frequently utilized in text and audio applications, such as natural language processing and speech recognition.

2.1.1 Convolutional neural networks

The Convolutional Neural Network (CNN) is a type of deep neural network framework that uses convolution operations to replace traditional matrix multiplication. The key features of CNNs are "weight sharing" and "preserving spatial information". The earliest convolutional neural network, LeNet-5, was proposed by Yann LeCun in 1998 [9]. The Convolutional Layer, Pooling Layer, and Fully Connected Layer in LeNet-5 have become the building blocks for subsequent convolutional neural network architectures.

Although convolutional neural networks were initially applied to simple text recognition, the training of these models was challenging due to the limitations of computer hardware in the 1990s. It wasn't until the 2000s that Kumar Chellapilla et al. [10] utilized GPUs to accelerate the training process of CNNs, and the availability of the large image database ImageNet [11] enabled the application of CNNs to a wider range of fields. Since then, there have been numerous well-known CNN architectures, such as AlexNet [12], VGGNet [13], and ResNet [14], which have achieved great success in image classification tasks. The R-CNN series [15-18] and YOLOs [19-22] are widely recognized models for object detection tasks, while FCN [23], Unet [24], and DeepLabs [25-28] have broad applications in image segmentation."

2.1.2. Related research of model acceleration and model compression

Since the 2010s, Convolutional Neural Networks (CNNs) have begun to flourish and have been widely adopted in the field of computer vision. With the increasing demand for higher model performance, academics have proposed more advanced neural network architectures, leading to an increase in the number of network layers. However, the changes also resulted in issues regarding the efficiency of these networks, such as the memory space and inference time. These efficiency concerns need to be addressed if the models are to be deployed in real-world applications or on mobile devices.

Recently, two main approaches, model acceleration and model compression have been proposed to address the efficiency issues. The methods of model acceleration aim to reduce the computational complexity of the model and accelerate its speed by designing a more efficient network computing method. In 2016, SqueezeNet proposed by Forrest et al. [29] initiated a major research direction for designing lightweight model structures. The Fire Module was introduced in SqueezeNet to replace traditional convolutional layers while achieving the same performance as AlexNet [12], with only 2.14% of AlexNet's parameters. In 2017, a research team at Google proposed a module called Depth-wise Separable Convolution in their paper MobileNet, which enabled a neural network model that could be used on mobile devices [30].

Additionally, the methods for model compression mainly focused on reducing the number of parameters in models through various techniques, thereby decreasing both storage and computational costs. Mainstream model compression methods include model pruning, knowledge distillation, and model quantization. Model pruning was first introduced by HanSong et al. [31, 32] in 2015 and it involves identifying and removing redundant parameters to compress the model. On the other hand, knowledge distillation, proposed by Hinton et al. [33], trained a smaller student model using a more complex and larger teacher model so that the student model can eventually perform similarly to the teacher model while having fewer parameters. Unlike model pruning, knowledge distillation does not cause irreversible damage to the model architecture. Finally, model quantization is the process of mapping values from one domain of definition to another, typically from a higher-precision domain to a lower-precision domain. Since convolutional neural networks are relatively less sensitive to the noise generated by quantization, quantization is an effective method for reducing model storage [34].

2.2. Related research on construction safety

In the construction industry, serious falling incidents are prone to occur when workers are at height. To prevent such incidents, workers are required to wear personal protective equipment (PPE) and various anti-fall facilities are set up in the work environment. PPE includes helmets, full-body safety harnesses, and hooks; while anti-fall facilities include guardrails, pillars, lanyards, and safety nets that provide support to workers when working at heights. In recent years, several studies have explored the application of deep learning technologies, particularly convolutional neural networks, in the field of construction safety [6].

In 2018, Qi Fang et al. [35] proposed a detection method for personal protective equipment using a combination of computer vision and an ASC classifier to identify the scene of aerial work. In the same year, Zdenek Kolar et al. [36] proposed a method for detecting safety barriers in images through the usage of deep convolutional neural networks and transfer learning. In 2020, Nipun D. Nath et al. [37] proposed a method based on YOLOV3 for real-time detection of personal safety equipment for construction workers. In 2021, Zifeng Wang et al. [38] utilized a semantic segmentation model for objects in construction sites. Since the use of semantic segmentation for automated visual understanding of construction sites has been rarely mentioned in previous literature, the authors employed the DeepLabV3+ model for the task of segmenting various objects categories in construction sites, along with a robotic system equipped with an RGB depth camera. This paper demonstrates the feasibility of using semantic segmentation models for construction sites, as well as the potential for automatic detection by robots.

2.3. Safety guardrail

The main detection target of this study is safety guardrails used at construction sites. According to article 19 in the Ministry of Labor' s Standards for Construction Safety and Health Installations [39], the roofs, steel beams, openings, stairs, and other equipment on the construction site that are higher than 2 meters must have guardrails installed as required by law. As outlined in Article 20 of the same Standards [39], guardrails consist of upper railings, middle railings, toe boards, and poles with a height of over 90 cm. The distance from the upper and lower openings to the floor surface must not be greater than 55 cm. Guardrails can be made of wood or steel pipes, and regardless of the material used, the strength of its poles, rods, and anchors should ensure that the entire guardrail has the ability to support the upper railing, with a strength to withstand a load of 75 kilograms in any direction without significant deformation.

Method

This chapter will outline the structure of the study. First, the dataset utilized in this study, along with the data processing techniques such as data labeling, augmentation, and preprocessing, will be discussed in Section 3.1. In Section 3.2 and 3.3, the three models and two model compression methods used in this study will be detailed. Finally, the evaluation and validation metrics employed in this study will be described in Section 3.4.

3.1. Data Collection

The dataset used in this study was collected from the "2021 Artificial Intelligence Application and Promotion of Hazard Identification in Construction Engineering-Steel Structure Engineering Edge Computing Assisted Artificial Intelligence Safety Identification Technology Improvement Project" of the Occupational Safety and Health Administration at the Ministry of Labor [40]. The guardrail images were taken at construction sites that cooperated with the project. This section will be divided into three subsections. Section 3.1.1 will illustrate the data labeling method used in the study, Section 3.1.2 and Section 3.1.3 will illustrate the data augmentation and preprocessing methods adopted in the study.

3.1.1 Data Labeling

The dataset used in this study primarily consists of two colors of guardrails: yellow and silver. The toe board part of the guardrail at the construction site is often obscured by other objects or its color is too similar to the floor or steel laminate. To concentrate the model on identifying a single object type, this study only labels the railing part of the guardrail in the image. The labeling result is shown in Figure 1, where the green part represents the guardrail and the black part represents the background.

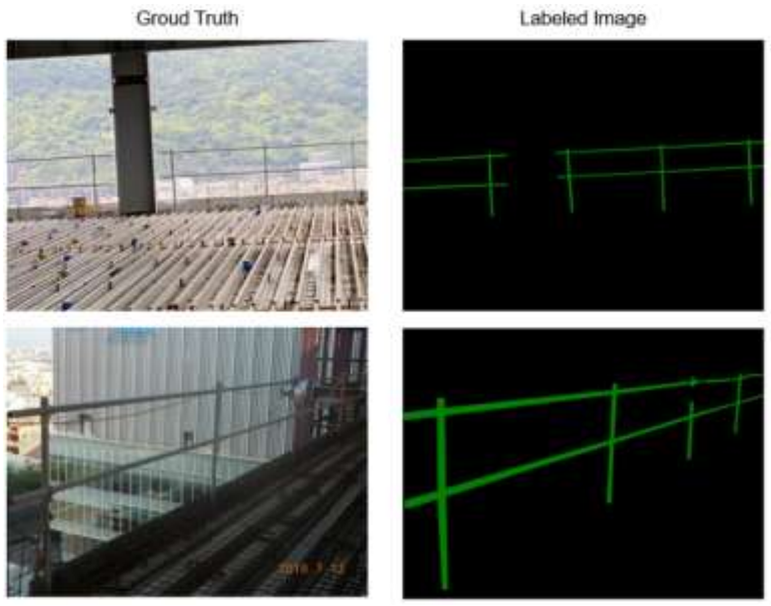

Figure 1. Dataset

3.1.2. Data augmentation

Due to the need for a large amount of data to avoid overfitting or underfitting when training deep learning models, in addition to the scarcity of data sources and the manpower required for data labeling, it is necessary to employ data augmentation to assist in the training process. Data augmentation is a technique that generates additional training data by using image processing or deep learning methods on a limited amount of original data.

The data augmentation method used in this study is based on image processing, including ten-fold image cropping and random affine transformation. Ten-fold image cropping is performed by cropping the center of the original image into five images of a fixed size ratio. These five augmented images are obtained by cropping the upper left, lower left, upper right, lower right and the center parts of an image. Finally, these five images are flipped horizontally. Random affine transformation is a method that randomly applies different affine transformations, such as translation, scaling, rotation, and flipping, to the training data in each iteration, providing the model with different data each time to achieve data augmentation.

3.1.3. Data Preprocessing

In this study, data filtering and image scaling were performed during the preprocessing stage. The data filtering process removes two types of data. The first type is images with a resolution lower than 1000 x 1000. Since the guardrails are small and thin, having a low resolution of the original image can cause difficulties in data labeling and model training, so these images are discarded. The other type is images that don't contain guardrails after the data augmentation process. Since the proportion of guardrails in the original image is low, some images obtained from ten-fold cropping may not contain guardrails at all, so these images are also discarded. During the image scaling stage, all training data is scaled to a fixed size, based on the quality of platforms used in this study and the ease of model training. The dataset is also divided into training and testing sets. The training images are scaled to (512, 512) and (256, 256), and the testing images are scaled to (960, 720).

3.2. Models

The processed training data in this study will be used by various semantic segmentation models to segment the safety guardrails. The models used are Unet++, DeepLab V3+, and EDANet. The following subsections will provide a detailed explanation of each semantic segmentation model.

3.2.1. Unet++

Unet++ is an improved model based on Unet [24], proposed by Zongwei Zhou et al. in 2018 [41]. It was initially used for improving the fineness of medical image segmentation. The key of Unet++ is its denser residual connection compared to its predecessor, Unet. As the model is prone to losing details and features at different levels of the input image during the process of down-sampling and up-sampling, the denser residual connections provide the model with different levels of feature maps, leading to more accurate and refined outputs.

3.2.2. DeepLab V3+

DeepLab V3+ is an improved semantic segmentation model based on DeepLab V3 [27], proposed by Liang-Chieh Chen et al. in 2018. The main difference from the previous version, DeepLab V3, is that it introduces an encoder-decoder architecture and improves its backbone network. The key of the model lies in the use of Atrous Spatial Pyramid Pooling (ASPP), which expands the receptive field of each layer of feature maps, thereby improving the overall ability of the model to capture detailed features. This also leads to a better result in segmenting the edges of objects in images. The decoder part of the model has discarded the use of bilinear interpolation for up-sampling and instead applies convolution operations to obtain features at different levels for multi-stage up-sampling to obtain the final output.

3.2.3. EDANet

EDANet is a semantic segmentation model designed for real-time segmentation, proposed by Shao-Yuan Lo et al. in 2019 [42]. Compared to other semantic segmentation models, EDANet has lower computational complexity and a smaller number of parameters, allowing for real-time performance. Additionally, in order to maintain the accuracy of segmentation, EDANet adopts an asymmetric convolution structure, which combines dilated convolution and dense connections.

3.3. Model acceleration and model compression

In order to improve the operational efficiency of the model and maintain a satisfactory level of accuracy, this study adopts two methods of model acceleration and compression, namely Ghost Module and Model Pruning. The former aims to improve the convolution operation in the neural network, thereby reducing the computational cost of the overall model; while the latter aims to improve the running speed of the model by removing redundant weights. The following subsections will discribe each model acceleration and compression method in detail and illustrate how they are applied in this study.

3.3.1. Ghost Module

The Ghost Module is the core block in GhostNet, a lightweight neural network proposed by the Google research team in 2020 [43]. The Ghost Module is an optimized module that can replace the traditional convolution operation, which is usually computationally expensive. By applying the Ghost Module, the computational cost generated by the convolution operation is reduced, thereby improving the calculation speed and maintaining the accuracy of the model.

In Ghost Module, the convolution operation is divided into three parts:
(1) First, execute the convolution operation on the feature map whose input size (width, height, number of channels) is (w,h,c), and the output is (w^',h^',m), where m=n/s, s is the compression ratio, and n is the number of traditional convolution output channels.
(2) Next, execute a linear transformation (Φ) on each channel of the feature map obtained in the previous step to obtain a feature map with the same output size (w^',h^',m).
(3) Finally, execute matrix concatenation of the result from steps (1) and (2) to obtain a final output with a size of (w^',h^',n), which has the same size as traditional convolutional operation.

The following formula below represents the calculation speed-up ratio (r_s) achieved after the traditional convolution operation was replaced by the Ghost Module:

$$r_s = \frac{n \cdot h' \cdot w' \cdot c \cdot k \cdot k}{\frac{n}{s} \cdot h' \cdot w' \cdot c \cdot k \cdot k + (s-1) \cdot \frac{n}{s} \cdot h' \cdot w' \cdot d \cdot d} = \frac{c \cdot k \cdot k}{\frac{1}{s} \cdot c \cdot k \cdot k + \frac{s-1}{s} \cdot d \cdot d} \approx \frac{s}{s+c-1} \approx s \qquad (1)$$

The following formula represents the calculation compression ratio (r_c) achieved after the Ghost Module replaces the traditional convolution operation:

$$r_c = \frac{n \cdot c \cdot k \cdot k}{\frac{n}{s} \cdot c \cdot k \cdot k + (s-1) \cdot \frac{n}{s} \cdot d \cdot d} \approx \frac{s \cdot c}{s+c-1} \approx s \qquad (2)$$

This study adopts the following changes to models introduced in Section 3.2:

(1) Unet++: The convolutional layers in Unet++ are replaced with the Ghost Modules. As shown in Figure 2, Figure 2(a) is the original convolution block, and Figure 2(b) is the modified block.

(2) DeepLab V3+: The traditional convolution layers in DeepLab V3+ is replaced with the Ghost Modules. In addition, since the convolution used in the backbone network is dilated convolution, it remains unchanged. The modified model architecture is shown in Figure 3.

(3) EDANet: The down-sampling block and the traditional convolutional layer in the EDA blocks are replaced with the Ghost Modules. Figure 4 illustrates the modified architecture of the down-sampling block, while Figure 5 shows the modified architecture of the EDA block.

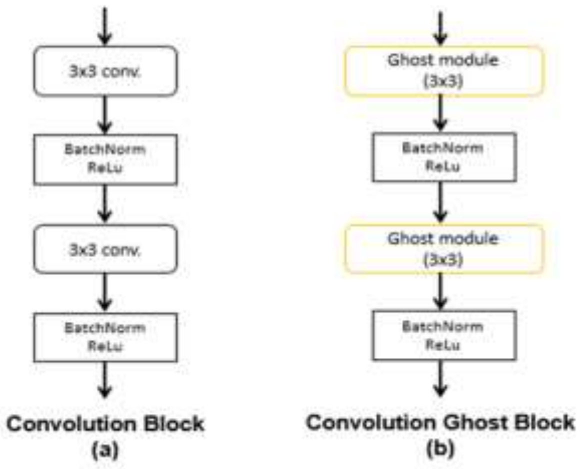

Figure 2. modified convolution block in Unet++

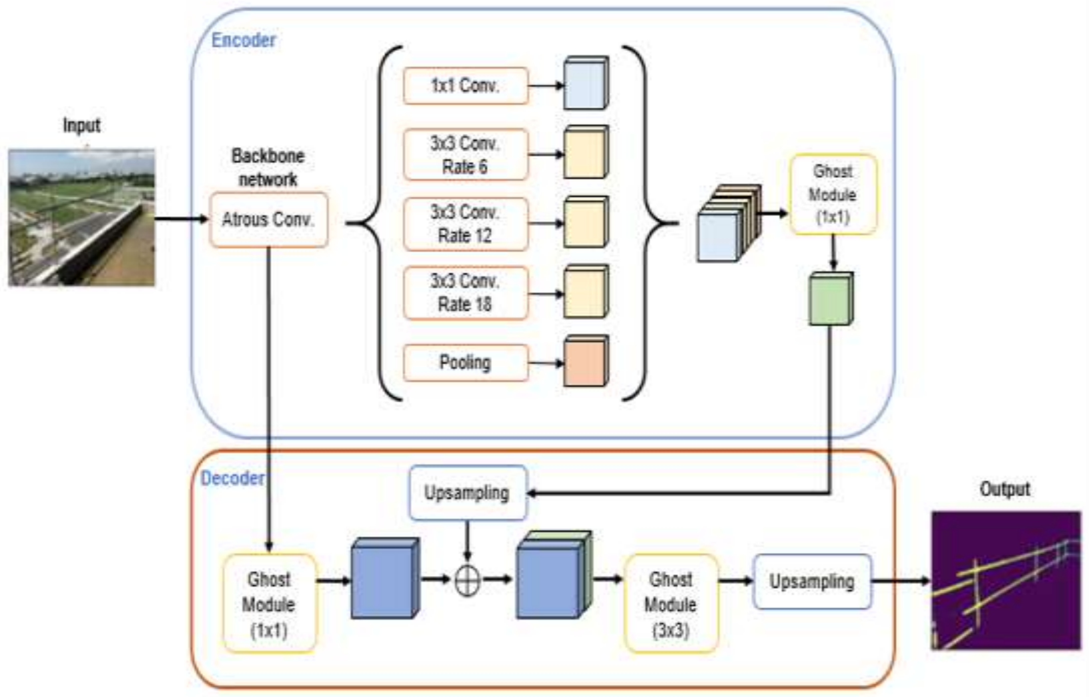

Figure 3. modified model architecture of DeepLab V3+

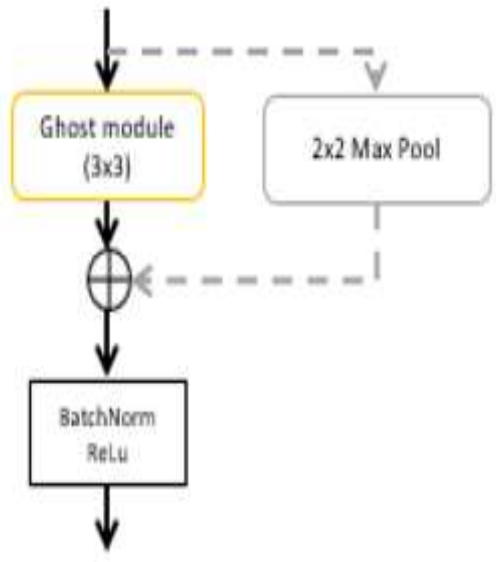

Figure 4. modified down-sampling block of EDANet

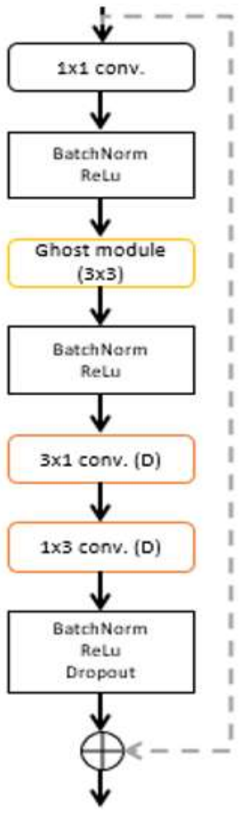

Figure 5. modified EDA block of EDANet

3.3.2. Model Pruning

Model pruning is a common method for model compression that involves finding and removing less important weights in the model to achieve compression.

In this study, the three steps of model pruning proposed by Han Song et al. [32] are used for model compression to ensure that the model retains a certain level of accuracy after some weights have been removed. The steps include initializing and training the model in a conventional manner, applying model pruning to set a certain percentage of weights in the model to zero, and retraining the pruned model to improve its accuracy. This process of pruning and retraining can be repeated multiple times until the model converges.

In terms of semantic segmentation, as the convolution operation holds the majority of parameters, this study focuses primarily on model pruning for each convolution layer in the models. For the pruning stage, two methods are adopted to select the weights to be removed in the models. The first method is Random selection, which prunes weights of the models randomly, and the other method is to identify and prune the less important weights based on the L1 norm.

3.4. Validation and evaluation metrics

This section will detail the evaluation metrics used in this study, including Intersection over Union (IoU), which is widely used in semantic segmentation model studies, Floating-point Operations (FLOPs) for measuring the acceleration and compression effects of the model, and Inference Time.

3.4.1. Intersection over Union

Intersection over Union, or IoU for short, is a widely used metric in semantic segmentation model evaluations. It measures the accuracy of the models' prediction by calculating the proportion of overlap between the predicted and ground truth segmentations for each category. The calculation formula is as follows:

$$\boldsymbol{IoU_{class}} = \frac{\boldsymbol{target} \cap \boldsymbol{prediction}}{\boldsymbol{target} \cup \boldsymbol{prediction}} = \frac{\boldsymbol{TP}}{\boldsymbol{FN} + \boldsymbol{FP} + \boldsymbol{TP}} \qquad (3)$$

Where TP refers to the true positive, which represents the total number of pixels that are correctly predicted as belonging to a certain category in an image; FN represents the false negative, which indicates the total number of pixels that actually belong to the category but were not correctly predicted; and FP refers to the false positive, representing the total number of pixels that do not belong to the category but were predicted as belonging to it. The calculated IoU value will be between 0 and 1, with 0 indicating that no pixels were correctly predicted, and 1 indicating that the predictions were a perfect match.

3.4.2. Floating-point operations

Floating-point operations, or FLOPs for short, represent the total number of floating-point operations per second for a model or an algorithm. It is an important indicator to measure the complexity of a deep learning model or algorithm. The FLOPs unit used in this study is gigaFLOPs (GFLOPs).

3.4.3. Inference time

Inference time is the time elapsed from input to the generation of output when a trained deep learning model is applied to new data. The length of time is influenced by both the complexity of the model and the performance of the hardware. In this study, inference time is utilized as one of the metrics to compare the acceleration and compression of each semantic segmentation models. The unit used is frames per second (FPS), which indicates the total number of inferences per second made by the model. In general, a model can be considered suitable for real-time calculation when its FPS reaches 30 or higher.

3. Results and Discussion

This chapter presents the experimental results based on the research method outlined in Chapter 3. Section 4.1 provides details of the dataset, experimental environment, and model configuration; while Section 4.2 uses the evaluation metrics mentioned in the previous chapter to analyze and present the results of various experiments.

4.1. Experimental environment

4.1.1. Dataset

In this study, a total of 88 original guardrail images were collected and used as the dataset. Out of the 88 images, 12 were designated as test data and the remaining 73 were used for training. After data augmentation and preprocessing, the training data set was increased to 792 images. However, 11 images were removed from the dataset due to low resolution or absence of guardrails after undergoing 10-fold image cropping. The details of the guardrail dataset are provided in Table 1.

Table 1. overview of the dataset

Category		Number of Images
Original data	before	88
	deleted	3
	after	85
Testing data		12
Training data	Data augmentation	803
	deleted	11
	after	792

4.1.2. Model configuration

The computer operating system utilized in the experiments in this study is Ubuntu 16.04. The graphics card used is NVIDIA GeForce RTX 2080 Ti, and the RAM is 16GB. The hyperparameters and input image size during the training of each semantic segmentation model in the following experiments are described in Table 2.

Table 2. Overview of model configuration

Model	Type	Value
Unet++	Number of epochs	25
	Batch size	8
	Learning rate	0.0005
	Input shape	(256, 256)
DeepLab V3+ Xception	Number of epochs	25
	Batch size	8
	Learning rate	0.0005
	Input shape	(256, 256)
DeepLab V3+ MobileNet V2	Number of epochs	25
	Batch size	16
	Learning rate	0.0005
	Input shape	(512, 512)
EDANet	Number of epochs	25
	Batch size	16
	Learning rate	0.0005
	Input shape	(512, 512)

4.3. Experimental result

The experiments in this study will be divided into three parts in order to explore the balance between accuracy and operational efficiency of different semantic segmentation models. First, we will compare the segmentation results of the four semantic segmentation models outlined in Section 3.2. Secondly, we will compare the performance of the combination of the four models with the model acceleration method described in Section 3.3.1. Finally, we will evaluate the effectiveness of the model compression method and different compression ratios described in Section 3.3.2 on the models with better performance from the previous experiments.

4.3.1. Semantic segmentation model comparison experiment

This section focuses on the comparison of the segmentation performance of different semantic segmentation models, and the impact of varying input sizes on these models. The models being compared are the four models introduced in Section 3.2: Unet++, DeepLab V3+, and EDANet, where DeepLab V3+ contains two backbone neural networks, MobileNet V2 and Xception.

In this section, the impact of using images of varying input sizes on the segmentation results will be examined. This experiment is conducted for two reasons. Firstly, hardware limitations and model size may restrict the use of high-resolution images during training. If an image with a higher resolution than the original training image is used in the testing stage, it may negatively affect the model's segmentation performance. Secondly, as the guardrails are small objects, it is anticipated that by resizing the original data to match the training input size, the model will be able to predict the guardrail more accurately. The experiment compares the results of four models using two different testing sets, with image sizes of (960, 720) referred to as the control group and the same size as the training input referred to as the experimental group.

The experiment compares the performance of four semantic segmentation models using two sets of testing images with different sizes, and the output will be rescaled to the size as same as the original image (960, 720). The evaluation metric used in this experiment is Intersection over Union (IoU). The results, as shown in Table 3, indicate that for the DeepLab V3+ model with Xception as the backbone, using images of the same size as the original training images leads to a higher IoU value of 0.5599 compared to 0.3973 for the control group. This suggests that using images of the same size in the inference stage improves the performance of semantic segmentation. The reason for this result is likely due to the limitations of the hardware and the large number of parameters in the DeepLab V3+ model, which cannot accurately predict the pixel position of the guardrail when using larger images than the original training data. However, for the other three models, using images of the same size as the original training images resulted in improved segmentation results. As an example, Figure 6 compares the results of the four models, where (a) represents the output of the control group and (b) represents the output of the experimental group."

According to the experimental results in this section, for subsequent experiments, the DeepLab V3+ model using Xception as the backbone will be tested on images of the same size as the original training images. Meanwhile, the other three models, Unet++, DeepLab V3+ with MobileNet V2 as the backbone, and EDANet, will be tested on images with a size of (960, 720).

Table 3. Result of semantic segmentation model comparison experiment

Model	Size of Training	Experimental IoU	Control IoU
Unet++	(256, 256)	0.2803	0.5380
DeepLab V3+ Xception	(256, 256)	0.5599	0.3973
DeepLab V3+ MobileNet V2	(512, 512)	0.4421	0.5675
EDANet	(512, 512)	0.3134	0.5411

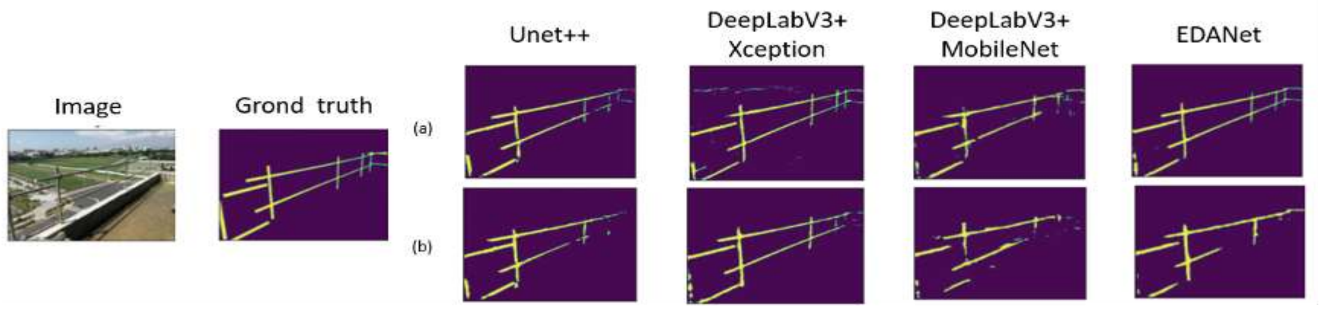

Figure 6. Result of semantic segmentation model comparison experiment

4.3.2. Model acceleration comparison experiment

This section focuses on evaluating the effectiveness of the Ghost Module, a model acceleration method outlined in Section 3.3.1, on improving the efficiency of semantic segmentation models while retaining a certain level of accuracy. The section is divided into two sub-sections to elaborate on the two experiments. The impact of the model acceleration on the model's segmentation performance is discussed in detail in Section 4.3.2.1; while the operation speed comparison on different devices is described in detail in Section 4.3.2.2. There are eight models compared in this section, including the four models from Section 4.3.1, and four models with the traditional convolution operations replaced by the Ghost Module. The four models are Unet++ with Ghost Module, DeepLabV3+ Xception with Ghost Module, DeepLabV3+ MobileNetV2 with Ghost Module, and EDANet with Ghost Module, referred to as Unet++ G, DeepLabV3+ XG, DeepLabV3+ MG, and EDANet G respectively. The comparison and evaluation metrics for these models include the number of model parameters, IoU, FLOPs, and inference time.

First of all, a comparison of model parameters was performed. Table 4 presents a comparison of the parameters of the original models and the models after employing the Ghost Module. As shown in Table 4, except for the DeepLabV3+ model with Xception as the backbone, the models' parameters significantly reduced after employing the Ghost Module. Among them, Unet++ and EDANet had the most noticeable reductions of 46.49% and 34.84%, respectively. In the case of the DeepLabV3+ model, the significant use of dilated convolution in its backbone network limited the reduction in model parameters.

Table 4. Comparison of model parameters

Model	Number of Parameters	Ratio
Unet++	8,596,703	-46.49%
Unet++ G	4,599,921	
DeepLabV3+ Xception	54,607,521	+7.78%
DeepLabV3+ XG	58,854,241	
DeepLabV3+ MobileNetV2	5,810,913	-11.75%
DeepLabV3+ MG	5,128,001	
EDANet	681,367	-34.84%
EDANet G	443,970	

4.3.2.1. Segmentation performance comparison experiment

The training results are presented in Table 5. The EDANet, Unet++, and DeepLabV3+ with MobileNetV2 as the backbone models show some improvement after the use of the Ghost Module. However, the DeepLabV3+ model with Xception as the backbone does not show any improvement in segmentation performance.

Table 5 provides the results of FLOPs and inference time. Unet++ significantly reduces the number of FLOPs by over 50%, but its improvement in inference time is limited and does not meet the requirement for real-time segmentation. EDANet, on the other hand, reduces FLOPs by approximately 30% with the Ghost Module, resulting in a segmentation speed of 93 images per second. The addition of the Ghost Module to the DeepLabV3+ model with MobileNetV2 as the backbone results in a slight increase in both FLOPs and inference time. Meanwhile, the DeepLabV3+ model with Xception as the backbone shows a decrease in FLOPs but no significant change in inference time after the Ghost Module

Figure 7 presents the output results of eight models, with (a) being the output result of the original models and (b) being the output result of the models with Ghost Module. As shown as

Figure 7, the employment of the Ghost Module in EDANet and Unet++ has resulted in the reduction of noise in the segmentation performance and improved coherence of the predictions. Additionally, the DeepLabV3+ model with MobileNet V2 as the backbone neural network demonstrates improved ability in capturing thinner guardrails.

Based on the above analysis, the replacement of traditional convolution operations with the Ghost module to accelerate the model has shown to be beneficial in terms of segmentation performance and operational efficiency. However, the DeepLabV3+ model using Xception as the backbone has not demonstrated improved results in terms of segmentation performance and running speed, even after being accelerated. As such, this model will not be used in further experiments.

Table 5. Result of segmentation performance comparison experiment

Model	IoU	FLOPs (Unit: GMac)	Inference time (Unit: FPS)
Unet++	0.5380	196.73	6.02
Unet++ G	0.6349	70.11	10.72
DeepLabV3+ Xception	0.5599	82.96	13.72
DeepLabV3+ XG	0.5050	78.54	13.29
DeepLabV3+ MobileNetV2	0.5675	29.13	50.58
DeepLabV3+ MG	0.5978	18.56	56.47
EDANet	0.5411	4.46	58.01
EDANet G	0.6503	3.13	93.02

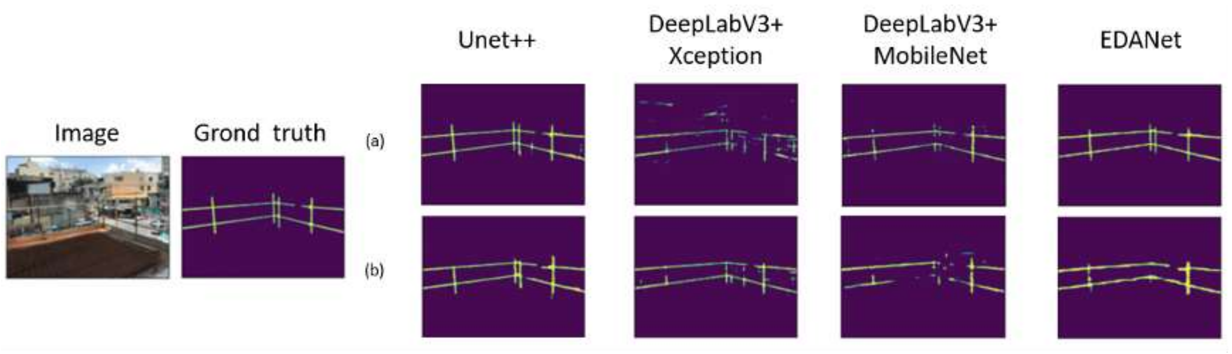

Figure 7. Result of segmentation performance comparison experiment

4.3.2.2. Comparison experiment of different devices

This section compares the computational speed of the eight models discussed previously through two different devices, namely a server and an edge device. The server used in this study is the same device used for training, while the edge device is the NVIDIA Jetson Nano. The purpose of this experiment is to compare the inference time required by the models for image segmentation on each device and to assess the effectiveness of the model acceleration method.

The experimental results are shown in Table 6. The four models using the lightweight network on the server can achieve real-time segmentation performance, while the Unet++ model with more complex models and the DeepLabV3+ model using Xception as the backbone cannot achieve real-time segmentation. On the other hand, in terms of the edge device, since the memory space of the Jetson Nano is not enough to support complex models for high-resolution image segmentation, certain models and image sizes cannot be loaded. It can be seen from Table 6 that the ranking of the obtained running results is similar to that of the server side, but due to the limitation of the device itself, although the models were accelerated, there is still no model that can be used for real-time segmentation.

Table 6. Result of comparison experiment of different devices

Model	Inference Time (Unit: FPS)			
	Size of Input Image			
	Server	Edge Device		
	(960, 720)	(960, 720)	(480, 640)	(256, 256)
Unet++	6.0237			0.0636
Unet++ G	10.7193			0.0854
DeepLabV3+ Xception	13.7231		0.0171	0.0989
DeepLabV3+ XG	13.2908		0.0154	0.0801
DeepLabV3+ MobileNetV2	50.5817	0.0277	0.0679	0.2394
DeepLabV3+ MG	56.4653	0.0321	0.0850	0.2187
EDANet	58.0046	0.0298	0.0985	0.2413

EDANet G	93.0233	0.0628	0.1150	0.2488

4.3.3. Model compression comparison experiment

This section discusses the effect of model pruning on the operational efficiency and performance of semantic segmentation models. In this experiment, two kinds of model pruning will be used with different compression ratios. In addition to the three baseline models involved in the experiment, Unet++, DeepLabV3+ using MobileNet V2 as the backbone, and EDANet, as well as three models with Ghost Module, are Unet++ with Ghost module, DeepLabV3+ MobileNetV2 with Ghost module and EDANet with Ghost module, which will be referred to as Unet++ G, DeepLabV3+ MG and EDANet G below. After model pruning, the model will lose a certain degree of segmentation ability. Therefore, in the experiments in this section, the model will be pruned three times, and it will be retrained with four iterations to maintain the segmentation ability of the model after each pruning. In this section, model parameters, IoU, and FLOPs will be used as the measurement and evaluation metrics of the experiment.

Table 7 shows the parameters of each model after pruning in different ratios, and Table 8 shows the FLOPs of each model after compression. As can be observed from both Tables 7 and 8, due to the varying proportion of convolutional layers used in different models in the overall architecture, the reduction in model parameters and FLOPs is slightly lower than the pruning ratio.

Table 7. The number of parameters in the model compression comparison experiment

Model	Pruning Ratio				
	Original	0.1	0.3	0.5	0.7
	Parameter Num.	Parameter Num.	Parameter Num.	Paramete r Num.	Paramet er Num.
Unet++	8,596,703	7,727,227	6,012,115	4,299,150	2,586,37 5
Unet++ G	4,599,921	4,097,398	3,188,380	2,274,843	1,369,17 0
DeepLabV3+ MobileNetV2	5,810,913	5,094,426	3,965,818	2,842,801	1,720,15 2

DeepLabV3+ MG	5,128,001	4,499,743	3,503,118	2,512,371	1,521,854
EDANet	681,367	613,862	478,853	343,842	208,833
EDANet G	443,970	399,029	311,638	224,252	136,863

Table 8. The FLOPs in model compression comparison experiment

Model	Pruning Ratio				
	Original	0.1	0.3	0.5	0.7
	FLOPs	FLOPs	FLOPs	FLOPs	FLOPs
Unet++	196.73	176.98	137.65	98.32	58.99
Unet++ G	70.11	62.88	48.90	34.93	20.94
DeepLabV3+ MobileNetV2	29.13	26.17	20.35	14.54	8.72
DeepLabV3+ MG	18.56	16.67	12.97	9.26	5.56
EDANet	4.46	4.01	3.12	2.23	1.34
EDANet G	3.13	2.82	2.19	1.56	0.94

In the first experiment, model pruning was performed by randomly selecting weights from the convolutional layers of the model. The pruning ratios used in this stage were 0.1, 0.3, 0.5, and 0.7. The results of the IoU for each model in the test dataset are shown in Table 9. It can be observed that removing a certain portion of the model's weights can still maintain its segmentation performance. However, since the random method was used to select the weights to be removed, the model's important weights for segmentation may be removed when the pruning ratio is high, resulting in a decline in segmentation accuracy.

Table 9. Results of random in model compression comparison experiment

Model	Pruning Ratio				
	Original	0.1	0.3	0.5	0.7
	IoU	IoU	IoU	IoU	IoU
Unet++	0.5380	0.5477	0.5316	0.5186	0.4916
Unet++ G	0.6349	0.6371	0.6128	0.6433	0.6072
DeepLabV3+ MobileNetV2	0.5675	0.5541	0.5532	0.5887	0.5051
DeepLabV3+ MG	0.5978	0.5892	0.5669	0.5722	0.5577
EDANet	0.5411	0.5539	0.5305	0.5675	0.5114
EDANet G	0.6503	0.5794	0.6043	0.6201	0.6114

Next, in order to avoid removing important weights when the model is pruned, the second experiment utilized the L1 norm to select the less important weights in the model for pruning. The pruning ratio used in this stage is the same as in the previous experiment. The aim was to investigate whether weight selection based on their significance would result in improved segmentation accuracy and computing speed compared to random selection. The results are shown in Table 10. When the pruning ratio was 0.1 and 0.3, each model has a similar level of segmentation ability as before pruning. When the compression ratio reaches 0.5, except for the EDANet and DeepLabV3+ models that use Ghost Module, the segmentation capabilities of the other models have improved. Finally, when the compression ratio reaches 0.7, most models get a higher value of IoU. among these, the ability of Unet++ to improve is the most prominent with nearly 0.7 IoU.

Table 10. Results of L1 norm in model compression comparison experiment

Model	Pruning Ratio				
	Original	0.1	0.3	0.5	0.7
	IoU	IoU	IoU	IoU	IoU

Unet++	0.5380	0.5156	0.5323	0.5693	0.5636
Unet++ G	0.6349	0.6172	0.6399	0.6718	0.6991
DeepLabV3+ MobileNetV2	0.5675	0.5424	0.5418	0.5831	0.5533
DeepLabV3+ MG	0.5978	0.5579	0.5758	0.5663	0.6145
EDANet	0.5411	0.5559	0.5634	0.5832	0.5887
EDANet G	0.6503	0.5891	0.6201	0.6168	0.6335

In conclusion, the model contains redundant weights. Regardless of the method used to identify which weights should be removed, fine-tuning the remaining weights can still result in a notable improvement in segmentation performance and a reduction in model complexity.

4. Conclusion and Future Work

5.1. Conclusion

This study applies several semantic segmentation models to detect guardrails in construction sites and proposes a combination of model acceleration and compression methods to enhance the computational efficiency of semantic segmentation models, which typically have a large number of parameters and high complexity. This study is expected to assist the site personnel in assessing the positioning and placement of guardrails according to regulations, thereby reducing the rate of accidents at construction sites. Furthermore, this study also reduces the computing power requirements of edge devices and enables further automation of industrial safety protection.

First of all, this study applies four semantic segmentation models (Unet++, DeepLabV3+ with Xception as the backbone, DeepLabV3+ with MobileNetV2 as the backbone, and EDANet) to detect safety guardrails. The Ghost Module is used to reduce model complexity and the number of parameters in the segmentation models. The results show that Unet++ had the largest reduction in parameters (over 40%) and increased IoU in guardrail segmentation to 0.65 after applying the Ghost Module. The models also experienced more than a 110% increase in computing speed on edge devices. This study also prunes the models using L1 norm, and find that most models perform better after removing a certain number of parameters, with Unet++ achieving an IoU of nearly 0.7. In conclusion, the semantic segmentation models can effectively improve segmentation performance and computational speed through the acceleration of the Ghost Module and model pruning, with Unet++ showing the most refined performance and EDANet having both adequate performance and high computational efficacy.

5.2. Future work

The dataset used in this study only includes a single type of safety guardrails. In order to extend the proposed model to other types of guardrails, additional relevant images need to be collected. This study can be further extended to encompass the image segmentation of other static objects in construction sites, such as safety lanyards and safety nets. Furthermore, other fields that require real-time detection can also utilize the model acceleration and compression techniques to enhance the accuracy and computational speed of the segmentation or detection models, beyond the construction field.

Acknowledgements
Organizer: Occupational Safety and Health Administration, MOL
Executive: National Taipei University of Technology,National Chung Hsing University
Funding: Occupational Safety and Health Administration, MOL
Project number: 111-059.

Reference

1. Akinosho, T. D., Oyedele, L. O., Bilal, M., Ajayi, A. O., Delgado, M. D., Akinade, O. O., & Ahmed, A. A. (2020). Deep learning in the construction industry: A review of present status and future innovations. Journal of Building Engineering, 32, 101827.
2. Bochkovskiy, A., Wang, C. Y., & Liao, H. Y. M. (2020). Yolov4: Optimal speed and accuracy of object detection. arXiv preprint arXiv:2004.10934.
3. Chen, L. C., Papandreou, G., Kokkinos, I., Murphy, K., & Yuille, A. L. (2014). Semantic image segmentation with deep convolutional nets and fully connected crfs. arXiv preprint arXiv:1412.7062.
4. Chen, L. C., Papandreou, G., Kokkinos, I., Murphy, K., & Yuille, A. L. (2017). Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. IEEE transactions on pattern analysis and machine intelligence, 40(4), 834-848.

5. Chen, L. C., Papandreou, G., Schroff, F., & Adam, H. (2017). Rethinking atrous convolution for semantic image segmentation. arXiv preprint arXiv:1706.05587.
6. Chen, L. C., Zhu, Y., Papandreou, G., Schroff, F., & Adam, H. (2018). Encoder-decoder with atrous separable convolution for semantic image segmentation. In Proceedings of the European conference on computer vision (ECCV) (pp. 801-818).
7. Cheng, T., Venugopal, M., Teizer, J., & Vela, P. A. (2011). Performance. evaluation of ultra wideband technology for construction resource location tracking in harsh environments. Automation in Construction, 20(8), 1173-1184.
8. Chellapilla, K., Puri, S., & Simard, P. (2006, October). High performance convolutional neural networks for document processing. In Tenth international workshop on frontiers in handwriting recognition. Suvisoft.
9. Chellapilla, K., Puri, S., & Simard, P. (2006, October). High performance convolutional neural networks for document processing. In Tenth international workshop on frontiers in handwriting recognition. Suvisoft.
10. Deng, J., Dong, W., Socher, R., Li, L. J., Li, K., & Fei-Fei, L. (2009, June). Imagenet: A large-scale hierarchical image database. In 2009 IEEE conference on computer vision and pattern recognition (pp. 248-255). IEEE.
11. Fiesler, E., Choudry, A., & Caulfield, H. J. (1990, August). Weight discretization paradigm for optical neural networks. In Optical interconnections and networks (Vol. 1281, pp. 164-173). SPIE.
12. Fang, Q., Li, H., Luo, X., Ding, L., Luo, H., & Li, C. (2018). Computer vision aided inspection on falling prevention measures for steeplejacks in an aerial environment. Automation in Construction, 93, 148-164.
13. Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). 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).
14. Girshick, R. (2015). Fast r-cnn. In Proceedings of the IEEE international conference on computer vision (pp. 1440-1448).
15. Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). 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).
16. Girshick, R. (2015). Fast r-cnn. In Proceedings of the IEEE international conference on computer vision (pp. 1440-1448).
17. Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., ... & Adam, H.

(2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861.

18. Han, S., Pool, J., Tran, J., & Dally, W. (2015). Learning both weights and connections for efficient neural network. Advances in neural information processing systems, 28.
19. Han, S., Mao, H., & Dally, W. J. (2015). Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. arXiv preprint arXiv:1510.00149.
20. Hinton, G., Vinyals, O., & Dean, J. (2015). Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531, 2(7).
21. Hinton, G. E., Osindero, S., & Teh, Y. W. (2006). A fast learning algorithm for. deep belief nets. Neural computation, 18(7), 1527-1554.
22. He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).
23. Han, K., Wang, Y., Tian, Q., Guo, J., Xu, C., & Xu, C. (2020). Ghostnet: More features from cheap operations. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (pp. 1580-1589).
24. He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2017). Mask r-cnn. In Proceedings of the IEEE international conference on computer vision (pp. 2961-2969).
25. He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).
26. Huang, C. H., "110年度人工智慧於營建工程危害辨識應用及推廣-鋼構工程邊緣運算輔助人工智慧安全辨識技術提升計畫(No:110-050)" (2021), Occupational Safety and Health Administration
27. He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2017). Mask r-cnn. In Proceedings of the IEEE international conference on computer vision (pp. 2961-2969).
28. Iandola, F. N., Han, S., Moskewicz, M. W., Ashraf, K., Dally, W. J., & Keutzer, K. (2016). SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and< 0.5 MB model size. arXiv preprint arXiv:1602.07360.
29. Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems, 25.
30. Kelm, A., Laußat, L., Meins-Becker, A., Platz, D., Khazaee, M. J., Costin, A. M., ... & Teizer, J. (2013). Mobile passive Radio Frequency Identification (RFID) portal for automated and

rapid control of Personal Protective Equipment (PPE) on construction sites. Automation in construction, 36, 38-52.

31. Kolar, Z., Chen, H., & Luo, X. (2018). Transfer learning and deep convolutional neural networks for safety guardrail detection in 2D images. Automation in Construction, 89, 58-70.
32. Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems, 25.
33. LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), 2278-2324.
34. Lo, S. Y., Hang, H. M., Chan, S. W., & Lin, J. J. (2019). Efficient dense modules of asymmetric convolution for real-time semantic segmentation. In Proceedings of the ACM Multimedia Asia (pp. 1-6).
35. LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. nature, 521(7553), 436-444.
36. Lee, H. S., Lee, K. P., Park, M., Baek, Y., & Lee, S. (2012). RFID-based real-time. locating system for construction safety management. Journal of Computing in Civil Engineering, 26(3), 366-377.
37. Long, J., Shelhamer, E., & Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 3431-3440).
38. Nath, N. D., Behzadan, A. H., & Paal, S. G. (2020). Deep learning for site safety: Real-time detection of personal protective equipment. Automation in Construction, 112, 103085.
39. National Regulatory Database，” 營造安全衛生設施標準(2021)” ，from the website of National Regulatory Database https://law.moj.gov.tw/LawClass/LawAll.aspx?pcode=N0060014
40. Occupational Safety and Health Administration， ” 2021年勞動檢查統計年報” ，from the website of Occupational Safety and Health Administration, https://www.osha.gov.tw/1106/1164/1165/1168/34345/
41. Pradhananga, N., & Teizer, J. (2013). Automatic spatio-temporal analysis of. construction site equipment operations using GPS data. Automation in Construction, 29, 107-122.
42. Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. Advances in neural information processing systems, 28.
43. Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. Advances in neural information processing

systems, 28.

44. Ronneberger, O., Fischer, P., & Brox, T. (2015, October). U-net: Convolutional networks for biomedical image segmentation. In International Conference on Medical image computing and computer-assisted intervention (pp. 234-241). Springer, Cham.
45. Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 779-788).
46. Redmon, J., & Farhadi, A. (2017). YOLO9000: better, faster, stronger. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 7263-7271).
47. Redmon, J., & Farhadi, A. (2018). Yolov3: An incremental improvement. arXiv preprint arXiv:1804.02767.
48. Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.
49. Wang, Z., Zhang, Y., Mosalam, K. M., Gao, Y., & Huang, S. L. (2022). Deep semantic segmentation for visual understanding on construction sites. Computer-Aided Civil and Infrastructure Engineering, 37(2), 145-162.
50. Zhou, Z., Rahman Siddiquee, M. M., Tajbakhsh, N., & Liang, J. (2018). Unet++: A nested u-net architecture for medical image segmentation. In Deep learning in medical image analysis and multimodal learning for clinical decision support (pp. 3-11). Springer, Cham.