Unsafe Construction Behavior Classification Using Deep Convolutional Neural Network

P. D. Hung[a],* **and N. T. Su**[a],**

[a]*FPT University, Hoa Lac High Tech Park, Hanoi, 10000 Vietnam*

**e-mail: hungpd2@fe.edu.vn*

***e-mail: su18mse13007@fsb.edu.vn*

Abstract—In the construction industry, about 80–90% of accidents are caused by the unsafe actions and behaviors of employees. Thus, behavior management plays a key role in enhancing safety. In particular, behavior observation is the most critical element for modifying workers' behavior in a safe manner. However, there is a lack of practical methods to measure workers' behavior in construction as current literature only focuses on a few unusual signs such as not wearing personal protective equipment. This paper proposes a system for recognizing workers' dangerous behaviors. To that end, an image dataset has been collected, labeled for three such behaviors. Based on the dataset obtained, the transfer-learning approach is used with three pretrained models, VGG19, Inception_V3 and InceptionResnet_V2. The results indicate that InceptionResnet_V2 performs better than VGG19_ and Inception_V3 for classifying unsafe behaviors and after 150 epochs, its accuracy reaches 92.44%.

Keywords: unsafe behavior in construction, convolutional neural network, transfer learning

DOI: 10.1134/S1054661821020073

1. INTRODUCTION

Considering that the unsafe actions and behaviors of employees in the construction industry account for about 80–90% of incidents [1], [2], and there is a meaningful correlation between behavior measurement and incident rates [3], worker behavior management therefore plays a critical role in the success of construction projects. Consequently, the most effective way to enhance safety can be considered behavior measurements (e.g. finding the frequency of unsafe actions and postures) and modifications (e.g. offering feedbacks, setting goals, and engaging workers) [4–8]. The industry, however, lacks cost-effective and robust methods to measure workers' behavior [9]. Behavior observation requires (1) a time-consuming and labor-intensive task in collecting and analyzing records [10], (2) plenty of data to deal with inconsistent and biased results [11], and (3) active worker participation in observing and reporting their own and colleague's unsafe behavior [12]. These requirements impose practical constraints on behavior measurement, and thus hinders the key to behavior-based safety management [13, 14].

Computer vision techniques have previously been used in construction-related research, focusing on the detection of construction workers, site machinery and on progress tracking [15–19]. In [20], Teizer described the status quo and challenges in computer vision in construction. Among computer vision techniques, histogram of oriented gradients (HOG) is one of the most widely used. Park and et al. [15] used HOG and the histogram of HSV (Hue, Saturation, Value) colors as inputs for a k-nearest neighbors (KNN) classifier. HOG, Histogram of Optical Flow (HOF), Motion Boundary Histogram (MBH) were used for the recognition of construction worker actions in [21]. Besides HOG, Haar Cascade [22] is another popular technique used in construction. Du et al. [23] used an approach based on the Haar Cascade to detect the workers' hard hats at construction sites. Kim et al. [24] used a combination of KNN and Scale Invariant Feature Transform (SIFT) algorithms to parse a complete image from a construction site. For traditional computer vision technologies, it is quite straightforward in that the extracted features are used to code one image, and then to conduct classification and clustering for labeling other images. However, the features are extracted by the predefined and special-purpose optimized models and these models can only be manually developed where high-dimensional features are required. Therefore, when multi-feature models are simultaneously considered, these traditional methods (HOG, HOF, MBH, etc.) lose their advantages or even may fail to perform the designed tasks.

Convolutional Neural Networks (CNN) have stood out as an effective method for solving image-based object detection and classification in construction-related problems [25]. Numerous studies have applied CNN-based algorithms to detect unsafe behavior in construction site. In [26], a Faster

Received October 8, 2020; revised January 16, 2021; accepted January 22, 2021

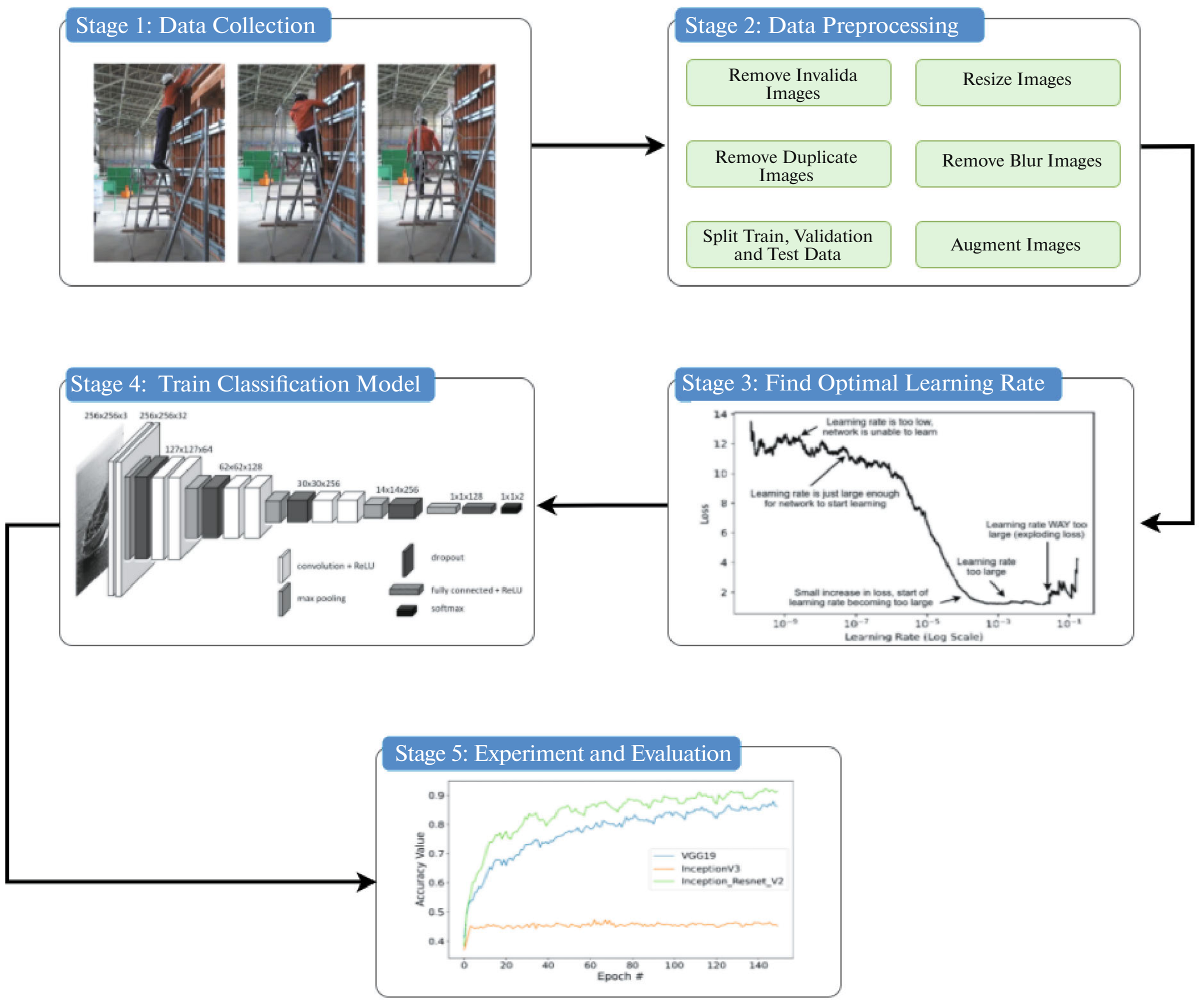

Stage 1: Data Collection
Stage 2: Data Preprocessing
Remove Invalida Images
Resize Images
Remove Duplicate Images
Remove Blur Images
Split Train, Validation and Test Data
Augment Images
Stage 3: Find Optimal Learning Rate
Learning rate is too low, network is unable to learn
Learning rate is just large enough for network to start learning
Learning rate WAY too large (exploding loss)
Learning rate too large
Small increase in loss, start of learning rate becoming too large
Loss
Learning Rate (Log Scale)
Stage 4: Train Classification Model
convolution + ReLU
max pooling
dropout
fully connected + ReLU
softmax
Stage 5: Experiment and Evaluation
Accuracy Value
Epoch #
VGG19
InceptionV3
Inception_Resnet_V2

Fig. 1. Overview of the proposed method.

R-CNN and deep CNN model were developed by W. Fang et al to identify workers and their harnesses. Precision and recall rates for the Faster R-CNN to detect workers were 99% and 95%, respectively, while those for the CNN to detect people not wearing their safety harnesses were 80% and 98%, respectively. In [27], Q. Fang et al introduced a deep-learning-based occlusion mitigation method for personal protective equipment (PPE) checking. Their experimental results have demonstrated that the method is robust under various conditions. In [28], Q. Fang et al developed an algorithm based on deep learning for non-hardhat-use (NHU) detection using more than 100.000 construction worker images. The results showed high precision and recall rate and that the proposed method can facilitate safety inspection and supervision.

When applying CNN to image classification problems, a pre-trained network, a network that was previously trained on a large-scale image dataset and had proved itself highly effective in dealing with small dataset cases, is often used. CNN architectures such as the VGG family [29], Inception_V3 [30], and Resnet50 [31] are typically trained on the ImageNet dataset [32] and have obtained very good results for general image classification. Inspired by such an accomplishment, K. Zdenek et al. [33] retrained the VGG-16 deep learning network on 4000 augmented images. The authors used the core part of a deep CNN called VGG-16 to transfer the image feature knowledge stored in the VGG-16 model to the guardrail detection model. Then, the MLP model was trained to process the output of the core VGG-16-based object detection. Its performance was shown to be better than a support vector machine (SVM) in the conclusion.

In summary, the previous studies were primarily limited to the detection of dangerous signs related to safety equipment such as helmets, protective clothes, guardrails. In addition, the datasets came from ideal laboratory conditions, not actual working environment.

The main contribution of this paper is a method for classifying three unsafe behaviors based on machine vision and deep learning technologies. The aim is to recognize a worker's dangerous behavior and to accelerate risk analysis and assessment with high accuracy.

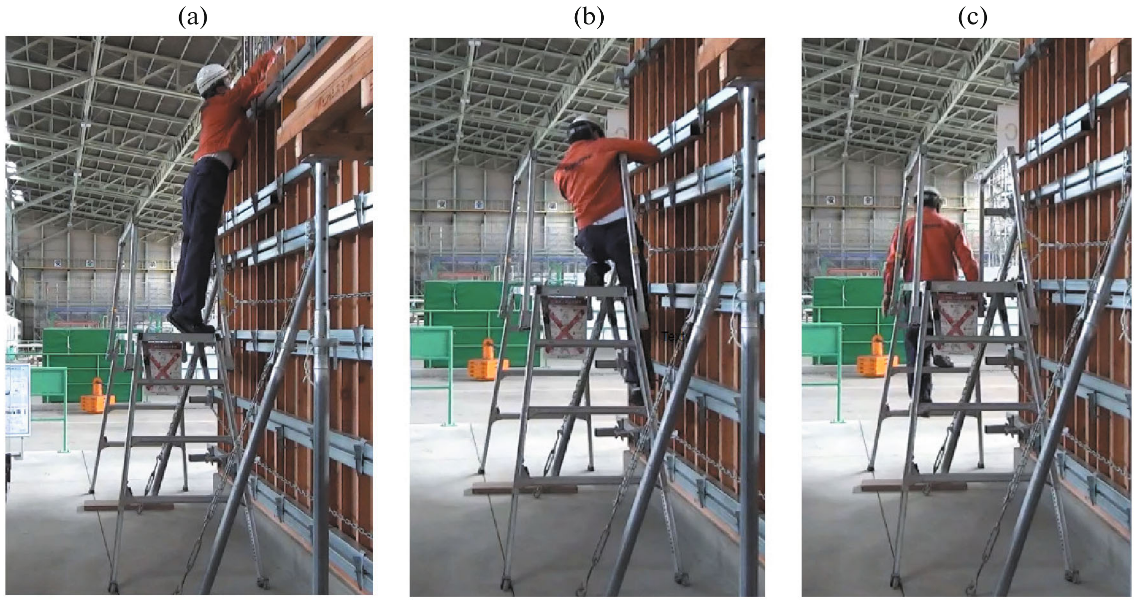


Fig. 2. Dataset sample. (a) Unsafe action 1: Human reach out, (b) Unsafe action 2: Human legs out, (c) Unsafe action 3: Human Climb Wrong.

The CNN architecture is used in combination with the transfer learning approach and optimizations of the model's helper parameters. In addition, a dataset with three labeled classes is made available to the research community on GitHub [34]. The study considers only the problem of classification of types of unsafe behavior. It is preceded by the problem of detection of any type of unsafe behavior, which must be also solved by an automated system coupled with video surveillance equipment.

The remainder of the paper is organized as follows. Section 2 describes the proposed method. Then, the results are analyzed in Section 3. Finally, brief conclusions are made in Section 4.

2. METHODOLOGY

2.1. Overview of the Proposed Method

The proposed method for classifying unsafe behavior includes five stages: (1) data collection, (2) data preprocessing, (3) learning rate optimization, (4) implementing the CNN with the transfer – learning approach and (5) experimentation and evaluation (Fig. 1). First, images are captured from an actual construction site in a variety of settings. Next, usable images are selected from the collected set during preprocessing. Then, optimal learning rate is figured out through an automatic finder algorithm. After that, the transfer – learning approach is used to enable the training of Deep Neural Networks without the need for millions of labeled data points typically for such complex models. Finally, the results are evaluated, analyzed and compared to a number of typical CNN architectures.

2.2. Data Collection

When using machine learning for digital image classification, the number and variety of the images in the dataset will greatly affect the classification accuracy. Currently, common datasets for classifying unsafe behaviors are either limited in quantity or are not collected from real working environment, therefore reducing the practicality of research based on such datasets. In this study, images are gathered from a real environment to obtain a more diverse selection. There is a total of 5000 images, captured using supervision camera from a real construction site in Japan, and divided into three dangerous behaviors as shown in Fig. 2.

2.3. Data Pre-processing

The collected images at 1200-by-1080 pixels are then pre-processed through the following steps:

• Step #1 – Detect and remove blurry images: This can be done by either of the two methods: Laplacian [35] operator or Fast Fourier Transform (FFT) [36]. Although the FFT method requires some manual tunings, it has proved to be more robust and reliable in blur detection than the variance of Laplacian method.

• Step #2 – Detect and remove irrelevant images: Images that do not contain unsafe actions are manually removed.

• Step #3 – Detect and remove duplicate images: Duplicate images introduce bias into dataset, making the deep neural network learn patterns specific to those. In addition, they hurt the model's ability to

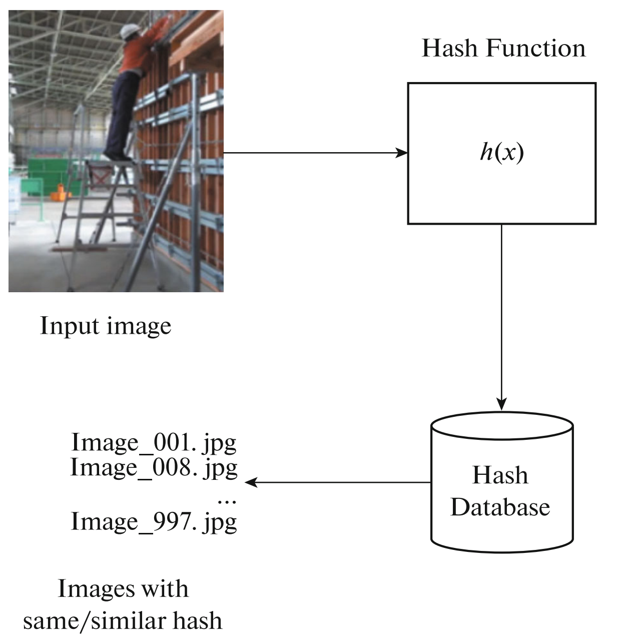


Fig. 3. Image-hashing function.

generalize to new images outside of what it was trained on. To detect and remove duplicate images, a method called "image hashing" is used [37] (Fig. 3).

The processed images are then separated into three parts: training, testing, and validation with a ratio of 8:1:1, respectively. Before training, the images are also augmented [38] and resized to fit model's input requirement.

2.4. Learning Rate Optimization

The automatic learning rate finder algorithm works through the following steps (Fig. 4):

• Step #1: Start by defining an upper and lower bound on the learning rate. The lower bound should be very small (1×10^{-10}) and the upper bound should be very large (1×10^{1}). At the lower bound, the learning rate should be too small for the network to learn, and at the upper bound, the learning rate should be too large, and the model would overfit.

• Step #2: Start training the network from the lower bound. After each batch update, the learning rate is exponentially increased.

• Step #3: Continue training until the learning rate hits the maximum value. Typically, this entire training process/learning rate increase takes 1–5 epochs.

• Step #4: After training is complete, a graph of loss versus learning rate is plotted, to identify the points where the learning rate is:

■ Large enough for the loss to decrease

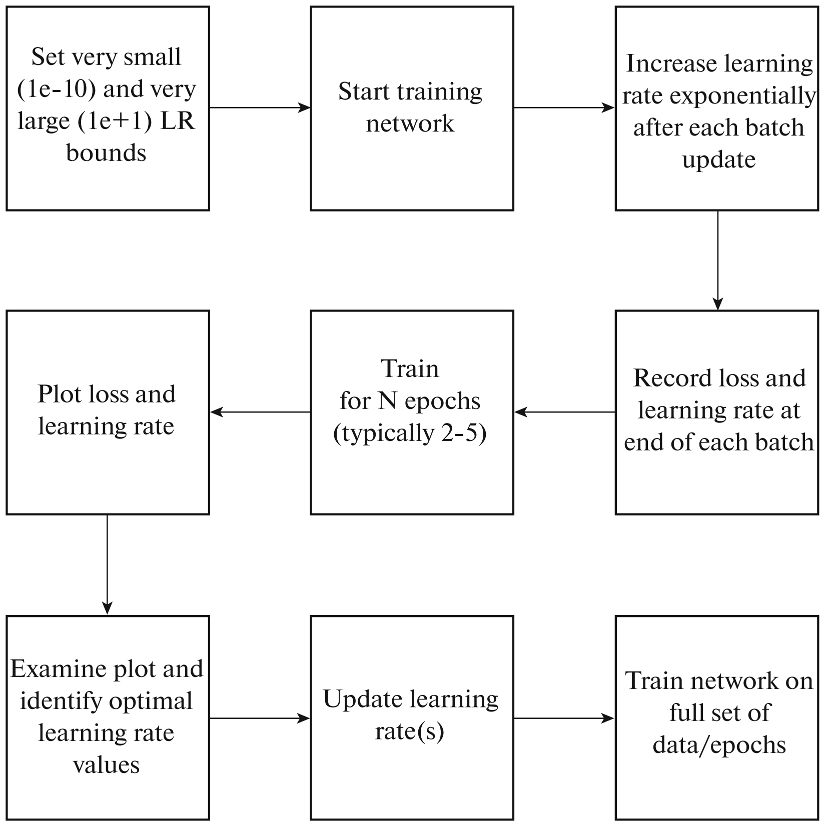

Set very small (1e-10) and very large (1e+1) LR bounds
Start training network
Increase learning rate exponentially after each batch update
Record loss and learning rate at end of each batch
Train for N epochs (typically 2-5)
Plot loss and learning rate
Examine plot and identify optimal learning rate values
Update learning rate(s)
Train network on full set of data/epochs

Fig. 4. Procedure for learning rate optimization.

■ Too large, to the point where loss starts to increase.

The following figure shows the visualized output of the learning rate finder algorithm on the dataset using a variation of VGG19 network (Fig. 5).

Notice that from 1×10^{-10} to 1×10^{-8} the loss only slightly decreases, meaning the learning rate is too small for the network to actually learn. Starting at approximately 1×10^{-7} the loss starts to decline – this is the smallest learning rate where the network can actually learn.

By the time we hit 1×10^{-6} the network is learning very quickly. At a little past 1×10^{-4}, there is a small increase in loss, but the big increase doesn't begin until 1×10^{-1}.

Finally, by 1×10^{1} the loss has exploded – the learning rate is now far too high for the model to learn.

Given this plot, through visual examination, the lower and upper bounds on the learning rate for the Cyclical Learning Rate (CLR) [39] can be determined to be 1×10^{-7} and 1×10^{-4} respectively.

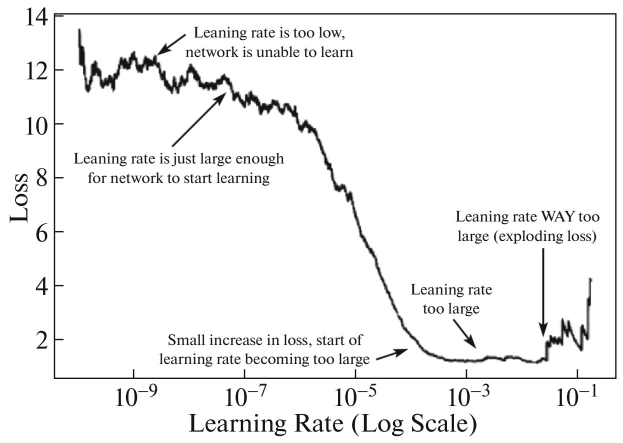


Fig. 5. Learning rate finder algorithm output.

2.5. Implementing Transfer-learning

A CNN architecture typically consists of several convolutional blocks and a fully connected layer. Each convolutional block is composed of a convolutional layer, an activation unit, and a pooling layer. A convolutional layer performs convolution operation over the output of the preceding layers using a set of filters or kernels to extract the features that are important for classification.

Although the development of a deep learning model for dangerous behavior recognition and classification is the key part of this research, training a model from scratch takes a considerable amount of time even with a workstation-level computer. For example, the training of the famous CNN classifier AlexNet [40], takes five to six days on NVIDIA GTX 580 3GB GPUs due to the large number of images. This long training time prevents quick validation of a trained classifier with various training options.

Transfer learning is an effective approach in reducing training time by fine-tuning a deep learning model that has previously been trained for a similar purpose. It is currently very popular in the field of Deep Learning because it enables the training of Deep Neural Networks with comparatively little data. In the present study, three different pre-trained CNN models including VGG19, Inception_V3 and InceptionResnet_V2 are experimented with to evaluate their performance.

The VGG network architecture was introduced by Simonyan and Zisserman in their 2014 paper [41]. This network is characterized by its simplicity, using only 3 × 3 convolutional layers stacked on top of each other in increasing depth. Volume size reduction is handled by max pooling. Two fully connected layers, each with 4.096 nodes are then followed by a softmax classifier. The number "19" stands for the number of weight layers in the network. VGG19 is a widely used convolutional architecture pre-trained on the ImageNet dataset. The second pre-trained model, Inception-v3, achieves state-of-the-art accuracy for recognizing general objects with 1000 classes, for example, "Zebra", "Dalmatian", and "Dishwasher". This model first extracts general features from input images and then classifies them based on those features [42]. Meanwhile, Inception-ResNet-v2 is a convolutional neural network that achieves a new level of accuracy on the ILSVRC image classification benchmark [43]. It is a variation of the earlier Inception V3 model with ideas borrowed from Microsoft's ResNet papers [44, 45]. Residual connections allow shortcuts in the model for researchers to successfully train even deeper neural networks, which have led to even better performance. The information for the VGG19, Inception_V3, and InceptionResnet_V2 models is illustrated in Table 1.

For transfer learning, first, only the convolutional part of a model up to the fully connected (FC) layers (i.e. excluding the top FC layers) is initiated. Then it is run on the training and validation image data only once and the output of the last layer before the FC layer, i.e. the output features, is saved. After that, a customized FC layer is trained on top of these output features. The output of the last convolutional layer is flattened and connected to the ReLU-activated units of the FC layer. The output layer consists of a single unit with a softmax activation [46], i.e. a function that turns numbers, a.k.a. logits, into probabilities that sum

Table 1. Pre-trained model properties

Model	Parameters	Depth
VGG19	143,667,240	26
Inception_V3	23,851,784	159
Inception Resnet_V2	55,873,736	572

```
Layer (type)                  Output Shape                  Param #
=====================================================================
flatten_1 (Flatten)           (None, 51200)                 0

dense_1 (Dense)               (None, 128)                   6553728

dropout_1 (Dropout)           (None, 128)                   0

dense_2 (Dense)               (None, 5)                     645
=====================================================================
Total params: 6,554,373
Trainable params: 6,554,373
Non-trainable params: 0
```

Fig. 6. Customized FC layers architecture.

```
Layer (type)                  Output Shape                  Param #
=====================================================================
input_3 (Input Layer)         (None, 227, 227, 3)           0

block1_conv1 (Conv2D)         (None, 227, 227, 64)          1792

block1_conv2 (Conv2D)         (None, 227, 227, 64)          36928

block1_pool (MaxPooling2D)    (None, 113, 113, 64)          0

block2_conv1 (Conv2D)         (None, 113, 113, 128)         73856

block2_conv2 (Conv2D)         (None, 113, 113, 128)         147584

block2_pool (MaxPooling2D)    (None, 56, 56, 128)           0

block3_conv1 (Conv2D)         (None, 56, 56, 256)           295168

block3_conv2 (Conv2D)         (None, 56, 56, 256)           590080

block3_conv3 (Conv2D)         (None, 56, 56, 256)           590080

block3_conv4 (Conv2D)         (None, 56, 56, 256)           590080

block3_pool (MaxPooling2D)    (None, 28, 28, 256)           0

block4_conv1 (Conv2D)         (None, 28, 28, 512)           1180160

block4_conv2 (Conv2D)         (None, 28, 28, 512)           2359808

block4_conv3 (Conv2D)         (None, 28, 28, 512)           2359808

block4_conv4 (Conv2D)         (None, 28, 28, 512)           2359808

block4_pool (MaxPooling2D)    (None, 14, 14, 512)           0

block5_conv1 (Conv2D)         (None, 14, 14, 512)           2359808

block5_conv2 (Conv2D)         (None, 14, 14, 512)           2359808

block5_conv3 (Conv2D)         (None, 14, 14, 512)           2359808

block5_conv4 (Conv2D)         (None, 14, 14, 512)           2359808

block5_pool (MaxPooling2D)    (None, 7, 7, 512)             0
---------------------------------------------------------------------
Total params: 20,024,384
Trainable params: 20,024,384
Non-trainable params: 0
```

Fig. 7. VGG19 pre-trained model architecture.

Table 2. Hyper-parameters

Hyper-parameter	Value
Epoch	150
Learning rate	CLR[1e-7, 1e-4]
Batch size	32

to one. It outputs a vector that represents the probability distributions of a list of potential outcomes, in this case are the three classes. The dropout layers are added after the activation layers to avoid overfitting. Meaning that the customized fully connected output layer of the network should have three classes expected instead of those in the pre-trained models.

Next, the model is compiled on the few last layers of the network in order to adjust the pre-trained weight using loss functions such as Categorical Cross-entropy and optimizers such as Adam [47]. The training is performed using the Keras framework with a TensorFlow backend, an open-source deep learning framework [48]. The hardware used is a personal computer with the following configurations: Intel® Core™ i7-7700HQ CPU (4 cores, 8 threads) @ 2.80 GHz, 16GB of RAM and a GeForce® GTX 1050 4GB GPU. Figure 6 shows the customized FC layers architecture, and Fig. 7 demonstrates the VGG19 pre-trained model architecture.

3. EXPERIMENTATION AND EVALUATION

The classifiers are trained with the following hyper-parameters (Table 2).

For evaluation, the training dataset is used to fit the model. The validation dataset is used to provide an unbiased evaluation of a model fit the training dataset while tuning the model hyper-parameters. The trained network is asked to predict the label for each image in our testing set, which used to provide an unbiased evaluation of a final model fit on the training dataset. These predictions are compared to the ground-truth labels, the category of the actual images of the testing set. From there, the number of predictions that are

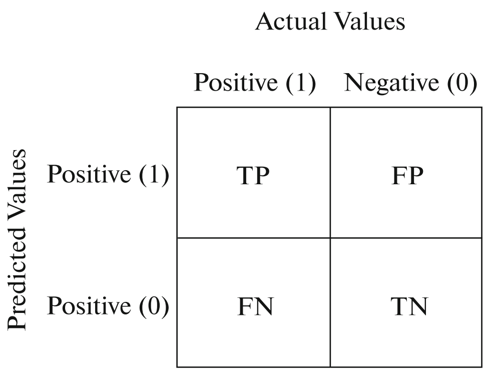


Fig. 8. Confusion matrix.

correct can be computed into aggregate reports presented as the confusion matrix (Fig. 8). Precision, recall, f-measure and accuracy are used to quantify the performance of the network as a whole.

Recall is the ratio of the Actual Positives the model managed to identify (True Positive).

$$\textbf{Recall} = \text{TP}/(\text{TP}+\text{FN})$$

Precision is how precise/accurate the model is, i.e. out of the predicted positive, how many of them are actual positive.

$$\textbf{Precision} = \text{TP}/(\text{TP}+\text{FP})$$

F1 Score is a balance between Precision and Recall.

$$\textbf{F1 Score} = 2 \times \text{Precision} \times \text{Recall}/(\text{Precision} + \text{Recall})$$

Accuracy is the ratio of predictions the model predicted correctly.

$$\textbf{Accuracy} = (\text{TP}+\text{TN})/(\text{TP}+\text{TN}+\text{FP}+\text{FN})$$

Comparing the three pre-trained models during the training process (Figs. 9, 10, 11, 12), the accuracy of InceptionResnet_V2 is always higher than those of Inception_V3 and VGG19. The accuracy of Inception_V3 is lower than that of VGG19. The pattern, is maintained during validation process, the accuracy of VGG19 is much higher than Inception_V3 but lower than InceptionResnet_V2. In terms of loss value, InceptionResnet_V2's loss value is always lower than the other two models. The same also happens here where the loss value of Inception_V3 is higher than VGG19 during training and evaluation, and the loss value of InceptionResnet_V2 is much lower than Inception_V3 and slightly lower than VGG19.

The training/validate accuracy curves and training/validation loss curves are shown in figures from Figs. 13 to 15. For two VGG19 and InceptionResnet_V2 models, it is clear that their accuracy and loss values increase and decrease gradually, respectively, throughout the training/evaluation process. Meanwhile, the accuracy and loss values of Inception_V3 do not seem to possess such trend.

Figures 16, 17, and 18 are the confusion matrices of the three models. It is easy to see that Inception_V3 cannot distinguish among three unsafe actions. The average percent of images predicted correctly as their actual labels by InceptionResnet_V2 is the higher than that in VGG19 model and Inception_V3.

Lastly, in Figs. 19, 20, 21, the average F1 score of InceptionResnet_V2 is greater than the other two models. The F1 scores of InceptionResnet_V2, VGG19 and Inception_V3 models are 0.91, 0.90 and 0.41, respectively.

In conclusion, InceptionResnet_V2 has shown the best results to solve the problem mentioned in this paper. The high accuracy of this model, 92.44%, also

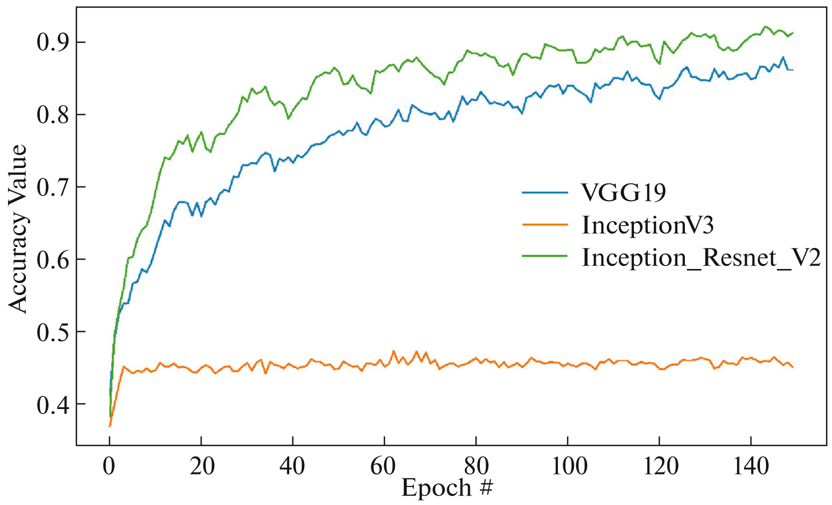


Fig. 9. Training accuracy curves of three pre-trained models.

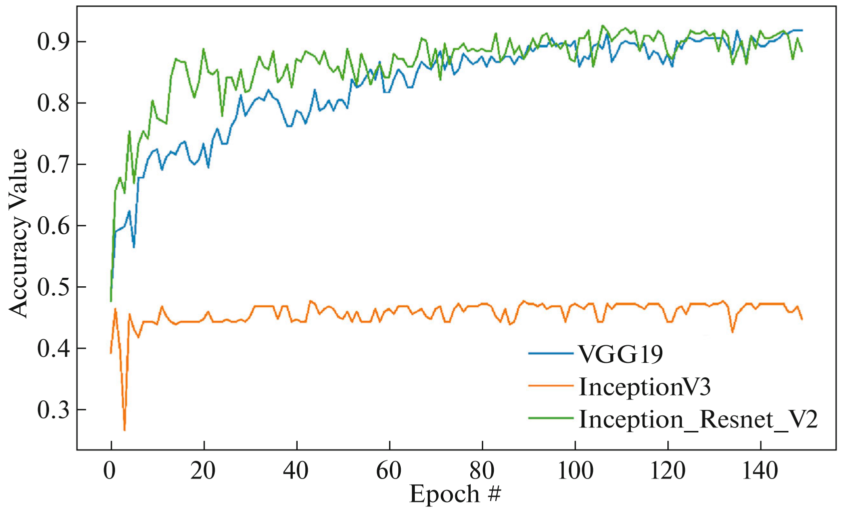


Fig. 10. Validation accuracy curves of three pre-trained models.

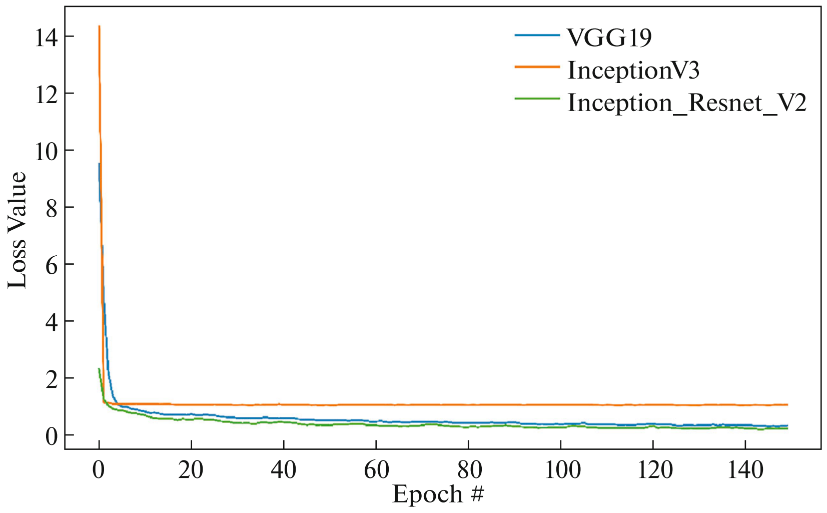


Fig. 11. Training loss curves of three pre-trained models.

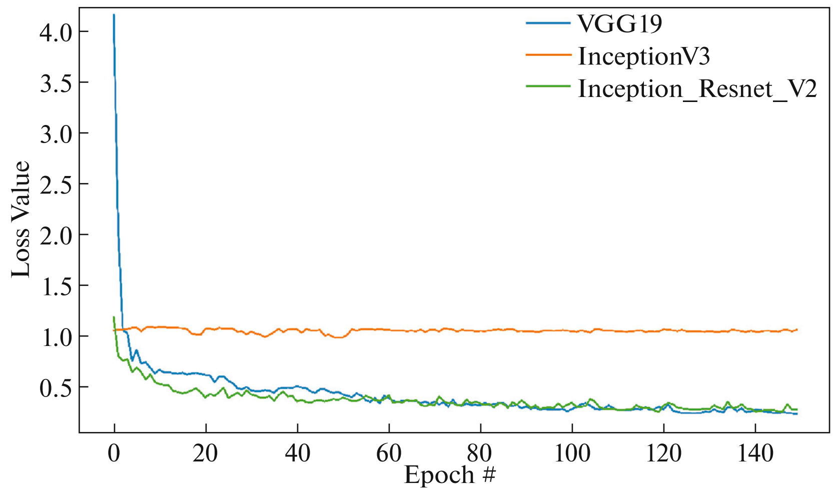


Fig. 12. Validation loss curves of the three models.

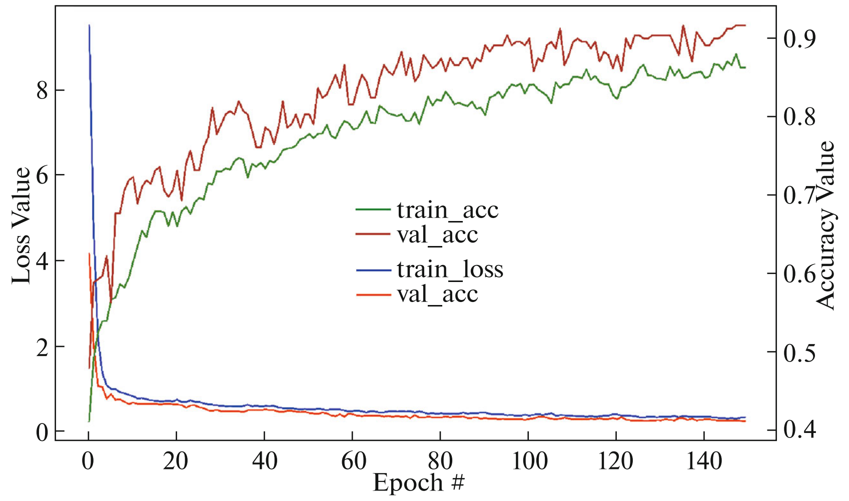


Fig. 13. Training/validation accuracy and training/validation loss curves of VGG19 model.

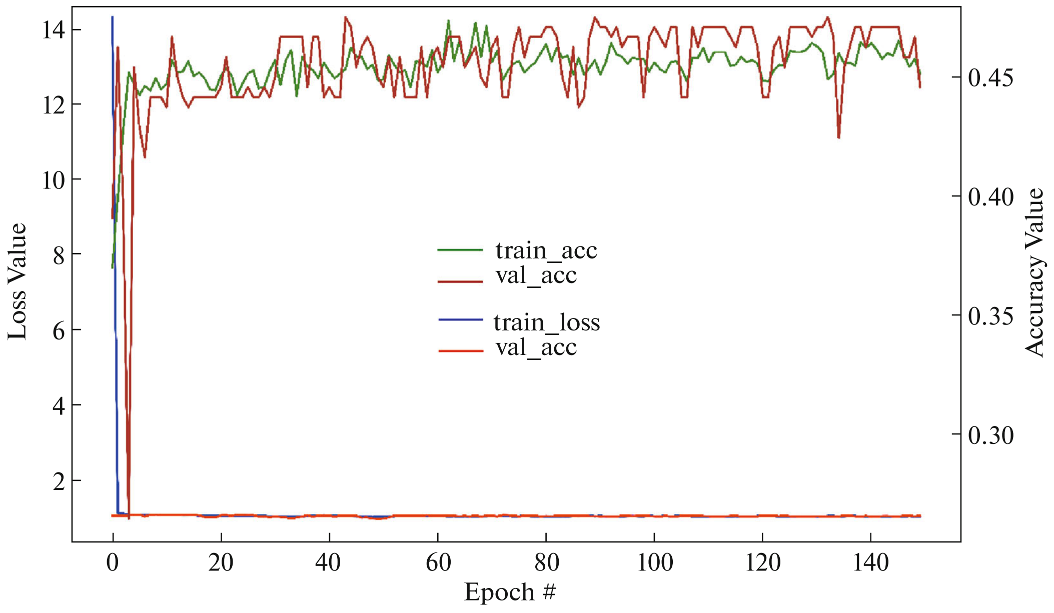


Fig. 14. Training/validation accuracy and training/validation loss curves of Inception_V3 model.

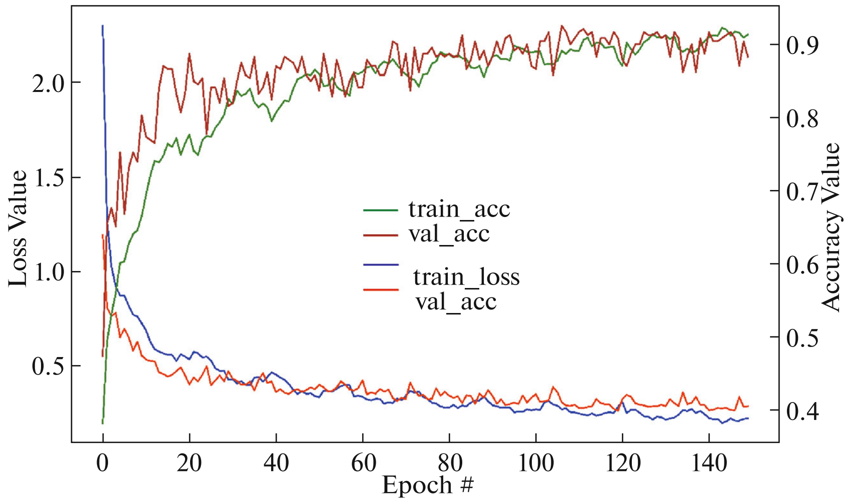
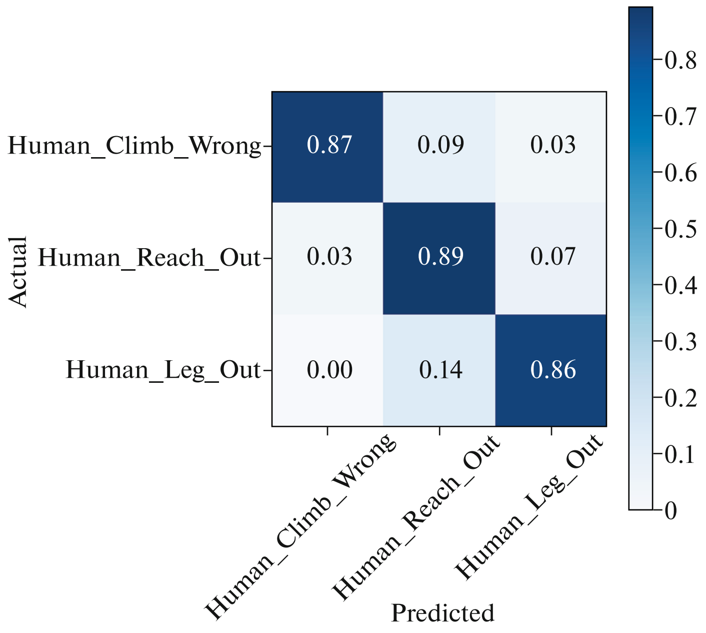


Fig. 15. Training/validate accuracy and training/validation loss curves of InceptionResnet_V2 model.



Actual \ Predicted	Human_Climb_Wrong	Human_Reach_Out	Human_Leg_Out
Human_Climb_Wrong	0.87	0.09	0.03
Human_Reach_Out	0.03	0.89	0.07
Human_Leg_Out	0.00	0.14	0.86

Fig. 16. Confusion matrix of VGG19 model.

confirms that the transfer learning approach is effective while also cut down on training time.

4. CONCLUSIONS AND PERSPECTIVE

In this study, a method for classifying workers' dangerous behaviors based on machine vision and deep learning technologies is proposed. A new dataset with three labeled classes is created and published on GitHub for the research community. This dataset was collected from a real construction environment using cameras, which is then reviewed by occupational safety experts. They are the quality assurance for this research.

Based on the image dataset obtained, the transfer-learning approach is used with three pre-trained models, VGG19, Inception_V3 and InceptionResnet_V2. Customized FC layers are defined and associated with the above models to perform classification. The results indicate that InceptionResnet_V2 performs better than VGG19 and Inception_V3 for classifying workers' dangerous behaviors. After 150 epochs, its accuracy reaches 92.44%, compared to the 91.16% and 47.06% of VGG19 and Inception_V3, respectively.

The proposed approach is not limited to the number of classes in the output classification. This challenge can be extended to building a large dataset that can cover more types of unsafe behavior in the future according to actual needs.

This work can serve as a reference for problems in Pattern Recognition and Object Classification. It can

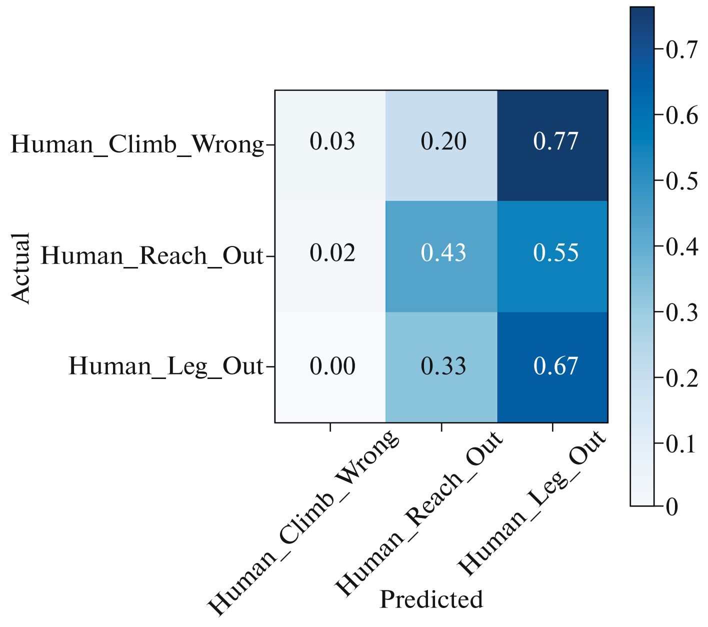


Fig. 17. Confusion matrix of Inception_V3 model.

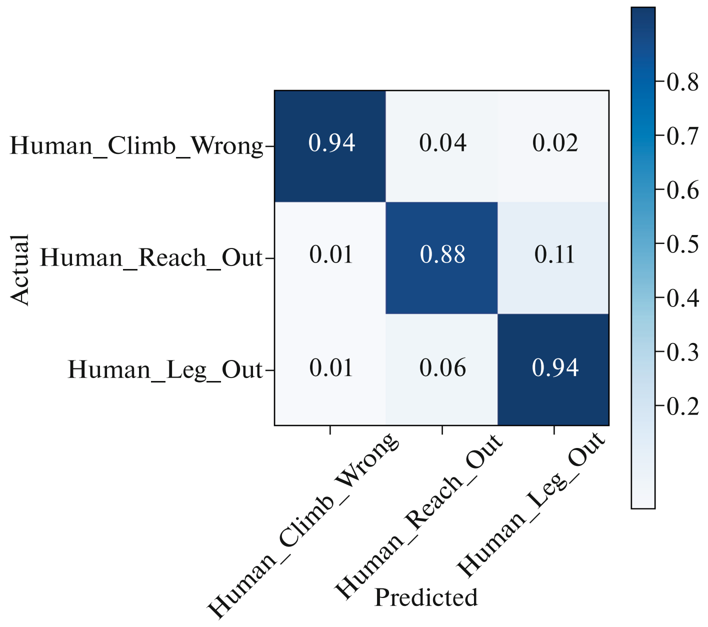


Fig. 18. Confusion matrix of InceptionResnet_V2.

	precision	recall	f1-score	support
Human_Climb_Wrong	0.94	0.91	0.93	222
Human_Reach_Out	0.92	0.84	0.89	317
Human_Leg_Out	0.84	0.92	0.88	252
accuracy			0.90	791
macro avg	0.90	0.90	0.90	791
weighted avg	0.90	0.90	0.90	791

Fig. 19. Classification report of VGG19.

```
                      precision    recall  f1-score   support

Human_Climb_Wrong          0.00      0.00      0.00       222
  Human_Reach_Out          0.40      1.00      0.58       317
    Human_Leg_Out          1.00      0.03      0.06       252

         accuracy                              0.41       791
        macro avg          0.47      0.34      0.21       791
     weighted avg          0.48      0.41      0.25       791
```

Fig. 20. Classification report of Inception_V3.

```
                      precision    recall  f1-score   support

Human_Climb_Wrong          0.98      0.94      0.96       222
  Human_Reach_Out          0.92      0.88      0.90       317
    Human_Leg_Out          0.86      0.94      0.89       252

         accuracy                              0.91       791
        macro avg          0.92      0.92      0.92       791
     weighted avg          0.92      0.91      0.91       791
```

Fig. 21. Classification report InceptionResnet_V2.

also be a reference for many fields in Deep learning, for example, Computer Vision, Parameterized Learning in Deep learning, Optimization Methods and Regularization in Deep learning, etc.

ACKNOWLEDGMENTS

We would like to thank our colleagues in the Information Technology Specialization Department of FPT University, Hanoi, Vietnam for their critical and relevant comments on the manuscript; Colleagues in the English Department who have helped to polish the English text. We would also like to extend our gratitude to experts at the Vietnam Institute for Building Science and Technology (IBST) for helping us with the methodological aspects of this study and with reviewing part of the data.

CONFLICT OF INTEREST STATEMENT

The authors declare no conflict of interest neither in financial nor in any other area.

COMPLIANCE WITH ETHICAL STANDARDS

Conflict of interests. The authors declare no conflict of interest neither in financial nor in any other area.

Statement of compliance with standards of research involving humans as subjects. All procedures performed in studies involving human participants were in accordance with the ethical standards of the institutional and/or national research committee and with the 1964 Helsinki Declaration and its later amendments or comparable ethical standards. Informed consent was obtained from all individual participants involved in the study.

REFERENCES

1. H.W. Heinrich, D. Petersen, and N. Roos, *Industrial Accident Prevention* (McGraw-Hill, New York, 1980).
2. L. Helen and S. Rowlinson, *Occupational Health and Safety in Construction Project Management* (Spon Press, London, 2005), pp. 157–158.
3. T. R. Krause, *The Behavior-Based Safety Process—Managing Involvement for an Injury-Free Culture* (Wiley, New York, 1990).
4. J. Komaki, K. D. Barwick, and L. R. Scott, "A behavioral approach to occupational safety: Pinpointing and reinforcing safety performance in a food manufacturing plan," J. Appl. Psych. **63** (4), 434–445 (1978).
5. A. R. Duff, I. T. Robertson, R. A. Phillips, and M. D. Cooper, "Improving safety by the modification of behavior," Constr. Manage. Econ. **12** (1), 67–78 (1994).
6. T. R. Krause, K. J. Seymour, and C. M. Sloat, "Long-term evaluation of behavior-based method for improving safety performance: A meta-analysis of 73 interrupted time-series replications," Saf. Sci. **32** (1), 1–18 (1999).
7. T. E. McSween, *Value-Based Safety Process: Improving Your Safety Culture with Behavior-Based Safety* (Wiley, Hoboken, NJ, 2003).
8. J. F. Godbey, "The effects of behavior-based safety techniques on behavior variation, targeted and non-targeted safety behaviors, and productivity and quality in manufacturing facilities," PhD Dissertation (Auburn Univ., Auburn, AL, 2006).
9. S. Han, S. Lee, S, and F. Peña-Mora, "Vision-based motion detection for safety behavior analysis in construction," in *Proc. of 2012 Construction Research Congress (CRC)* (ASCE, Reston, VA, 2012), pp. 1032–1041.
10. R. E. Levitt and N. M. Samelson, *Construction Safety Management* (McGraw-Hill, New York, 1987).

11. H. Laitinen, M. Marjamäki, and K. Päivärinta, "The validity of the TR safety observation method on building construction," Accid. Anal. Prev. **31** (5), 463–472 (1999).

12. S. Han, S. Lee, and F. Peña-Mora, "Application of dimension reduction techniques for motion recognition: Construction worker behavior monitoring," in *Proc. of 2011 ASCE Int. Workshop on Computing in Civil Engineering* (ASCE, Reston, VA, 2011), pp. 102–109.

13. J. S. Chhokar and J. A. Wallin, "Improving safety through applied behavior analysis," J. Saf. Res. **15** (4), 141–151 (1984).

14. J. L. Komaki, S. Zlotnick, and M. Jensen, "Development of an operant-based taxonomy and observational index of supervisory behavior," J. Appl. Psych. **71** (2), 260–269 (1986).

15. M. Park, E. Palinginis, and I. Brilakis, "Detection of construction workers in video frames for automatic initialization of vision trackers," in *Construction Research Congress* (2012). https://doi.org/10.1061/9780784412329.095.

16. I. Brilakis, M. W. Park, and G. Jog, "Automated vision tracking of project related entities," Adv. Eng. Inf. **25** (2011). https://doi.org/10.1016/j.aei.2011.01.003

17. V. Pətrəucean, I. Armeni, M. Nahangi, J. Yeung, I. Brilakis, and C. Haas, "State of research in automatic as-built modelling," Adv. Eng. Inf. **29**, 162–171 (2015). https://doi.org/10.1016/j.aei.2015.01.001

18. M. W. Park and I. Brilakis, "Construction worker detection in video frames for initializing vision trackers," Autom. Constr. **28**, 15–25 (2012). https://doi.org/10.1016/j.autcon.2012.06.001

19. Y. Ham, K. K. Han, J. J. Lin, and M. Golparvar-Fard, "Visual monitoring of civil infrastructure systems via camera-equipped unmanned aerial vehicles (UAVs): A review of related works," Vis. Eng. **4** (1) (2016). https://doi.org/10.1186/s40327-015-0029-z

20. J. Teizer, "Status quo and open challenges in vision-based sensing and tracking of temporary resources on infrastructure construction sites," Adv. Eng. Inf. **29**, 225–238 (2015). https://doi.org/10.1016/j.aei.2015.03.006

21. J. Yang, Z. Shi, and Z. Wu, "Vision-based action recognition of construction workers using dense trajectories," Adv. Eng. Inf. **30**, 327–336 (2016). https://doi.org/10.1016/j.aei.2016.04.009

22. P. Viola and M. J. Jones, "Robust real-time face detection," Int. J. Comput. Vis. **57**, 137–154 (2004). https://doi.org/10.1023/B:VISI.0000013087.49260.fb

23. S. Du, M. Shehata, and W. Badawy, "Hard hat detection in video sequences based on face features, motion and color information," in *Proc. 3rd International Conference on Computer Research and Development* (2011), pp. 25–29. https://doi.org/10.1109/ICCRD.2011.5763846.

24. H. Kim, K. Kim, and H. Kim, "Data-driven scene parsing method for recognizing construction site objects in the whole image," Autom. Constr. **71**, 271–282 (2016). https://doi.org/10.1016/j.autcon.2016.08.018

25. W. Rawat and Z. Wang, "Deep convolutional neural networks for image classification: A comprehensive review," Neural Comput. **29** (9), 2352–2449 (2017).

26. W. Fang, L. Ding, H. Luo, and P. E. D. Love, "Falls from heights: A computer vision-based approach for safety harness detection," Autom. Constr. **91**, 53–61 (2018). https://doi.org/10.1016/j.autcon.2018.02.018

27. Q. Fang, H. Li, X. Luo, L. Ding, H. Luo, and C. Li, "Computer vision aided inspection on falling prevention measures for steeplejacks in an aerial environment," Autom. Constr. **93**, 148–164 (2018). https://doi.org/10.1016/j.autcon.2018.05.022

28. Q. Fang, H. Li, X. Luo, L. Ding, H. Luo, T. M. Rose, and W. An, "Detecting non-hardhat-use by a deep learning method from far-field surveillance videos," Autom. Constr. **85**, 1–9 (2018). https://doi.org/10.1016/j.autcon.2017.09.018

29. K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," arXiv (2014). arXiv:1409.1556

30. C. Szegedy, V. Vanhoucke, S. Ioffe, et al., Rethinking the inception architecture for computer vision. https://www.cv-foundation.org/openaccess/content_cvpr_2016/papers/Szegedy_Rethinking_the_Inception_CVPR_2016_paper.pdf. Accessed June 22, 2020.

31. K. He, X. Zhang, S. Ren, et al., Deep residual learning for image recognition. https://www.cv-foundation.org/openaccess/content_cvpr_2016/html/He_Deep_Residual_Learning_CVPR_2016_paper.html. Accessed June 22, 2020.

32. J. Deng, W. Dong, R. Socher, et al., "ImageNet: A large-scale hierarchical image database," in *IEEE Conference on Computer Vision and Pattern Recognition* (Miami, FL, 2009).

33. K. Zdenek, C. Hainan, and L. Xiaowei, "Transfer learning and deep convolutional neural networks for safety guardrail detection in 2D images," Autom. Constr. **89**, 58–70 (2018). https://doi.org/10.1016/j.autcon.2018.01.003

34. https://github.com/tiensu/Construction_Site_Unsafe_Actions_Dataset.git.

35. R. C. Gonzalez and R. E. Woods, *Digital Image Processing,* 4th ed. (Pearson, 2018), chap. 3.

36. R. C. Gonzalez and R. E. Woods, *Digital Image Processing,* 4th ed. (Pearson, 2018), chap. 4.

37. H. Kim, S. Sohn, and J. S. Kim, "Revisiting Gist-PCA hashing for near duplicate image detection," J. Sign. Process Syst. **91**, 575–586 (2019). https://doi.org/10.1007/s11265-018-1360-0

38. Keras-team, "Image data preprocessing." https://keras.io/api/preprocessing/image/. Accessed June 22, 2020.

39. S. Leslie, "Cyclical learning rates for training neural networks," arXiv (2015). arXiv:1506.01186

40. A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proceedings of the 25th International Conference on Neural Information Processing Systems* (2012), pp. 1097–1105.

41. K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," arXiv (2014). arXiv:1409.1556

42. K. Gopalakrishnan, S. K. Khaitan, A. Choudhary, et al., “Deep convolutional neural networks with transfer learning for computer vision-based data-driven pavement distress detection,” Constr. Build. Mater. **157**, 322–330 (2017).

43. Stanford Vision Lab, Large Scale Visual Recognition Challenge 2012 (VOC2012). http://image-net.org/challenges/LSVRC/2012/. Accessed June 22, 2020.

44. K. He, X. Zhang, S. Ren, et al., “Deep residual learning for image recognition,” arXiv (2015). arXiv:1512.03385

45. K. He, X. Zhang, S. Ren, et al., “Identity mappings in deep residual networks,” arXiv (2016). arXiv:1603.05027

46. Uniqtech, Understand the Softmax Function in Minutes. https://medium.com/data-science-bootcamp/understand-the-softmax-function-in-minutes-f3a59641e86d. Accessed June 22, 2020.

47. D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” arXiv (2017). arXiv:1412.6980

48. Keras-Team, Keras: The Python Deep Learning library. https://keras.io/. Accessed June 22, 2020.

Phan Duy Hung received his Ph.D. degree from INP Grenoble France, in 2008. Since 2009, he has worked as a Lecturer, and served as the Head of Department and the Director of Master Program in the Software engineering at FPT University, Hanoi, Vietnam.
His current research interests include Digital Signal and Image processing, Internet of Things, Big-Data, Artificial Intelligence, Measurement and Control Systems and Industrial networking.

Nguyen Tien Su received his B. Sc from Hanoi University of Science and Technology, Vietnam in 2012, and MCS from FPT University, Vietnam 2020. Since 2013, he has worked as an engineer in the fields of Machine Learning, Image Processing and Software engineering.