A Pull-Reporting Approach for Floor Opening Detection Using Deep-Learning on Embedded Devices

Sharjeel Anjum[a], Rabia Khalid[a], Muhammad Khan[a], Numan Khan[a], and Chansik Park [a,*]

[a] Department of Architectural Engineering, Chung-Ang University, Seoul 06974, South Korea.
E-mail: sharjeelanjum@cau.ac.kr, rabiakhalid@cau.ac.kr, muhammadkhan607@cau.ac.kr, numanpe@gmail.com, cpark@cau.ac.kr

Abstract –

The construction site is prone to a considerable number of accidents due to its dense and complex nature. Accidents due to falling from opening at the construction site are leading reasons for severe injuries and sometimes fatalities. Openings and Holes are made on floors and roofs during the building construction or destruction. Despite being aware of the hazards associated with openings, gaps, and holes when working at heights, many workers fail to cover the openings or remove the covers for ease of work. The current inspection procedure relies on manual practices that are error-prone, time-consuming, expensive, and difficult for a site manager to monitor. Therefore, the authors propose a pull-reporting approach to resolve this issue by utilizing a computer vision detector model embedded in the android mobile to facilitate the safety manager. The proposed application uses YOLOv4 trained weights on a custom dataset obtained from the recorded videos at the Korean Scaffolding Institute with various view angels and various degrees of occlusion and data crawling techniques. The weights are then deployed on edge devices using TensorFlow API, Java programming, and maintain a real-time database of unsafe behavior. The developed system can identify, classify, and record the fully opened openings (FOO) and partially opened openings (POO) at the construction site along with geo-coordinates details in real-time.

Keywords –

Edge Computing; Worker Driven Approach; Construction Hazards; Trade Worker Safety; Safety Monitoring

1 Introduction

In recent decades, the construction industry has grown, resulting in higher firm profits, financial accessibility, and commodity demand. Despite its prominence, it has long been recognized as one of the world's most dangerous industries due to its severe accident rate than other industries [1]. Falls from heights (FFH) are a severe construction industry issue [2]. According to research, FFH is responsible for around 48 percent of major injuries and 30 percent fatalities. Most FFH events occur due to the fully opened opening (FOO) and partially opened opening (POO) at the temporary supporting platforms, e.g., scaffolding and on-ground openings. There are several safety regulations and procedures in place to protect workers' falls from height; for instance, the gaps, holes, and openings should be covered so that the person working near the edges could be safe during his task.

Even though covering the openings and gaps is a legal requirement and employees are aware of their risk of falling, workers have shown a reluctance to utilize these regulations, as could be understood by many accidents that already happened in construction. Extra efforts besides their tasks and the constraints it imposes on mobility have been identified as the reasons for non-compliance.

The detection of wall openings is similar to the detection of slab holes. The location of the wall element must be considered in the special situation: whether it is an interior or exterior wall [3]. To avoid any fall, the wall opening should also be protected [3]. Construction and safety managers require realistic means to monitor and ensure that the openings, holes, and gaps are entirely covered with the desired strength material. On the other hand, the safety inspection procedure can be time-consuming and intermittently commenced because traditional practices rely on the push-inspection approach or manual inspection process where the safety managers enforce the safety rules through top to bottom monitoring style. Correspondingly, safety rule compliance about the hole, gap, and opening covering is challenging if done with the traditional practices, and the possibility of falls from height remains a severe hazard. Therefore, this paper presents an intuitive pull-reporting approach which means that the worker is involved in this process to report

to the safety manager by using an android-based application to automate the manual inspection process.

2 Literature Review

Construction sites have a distinct, dynamic, and complex working environment and non-standardized design and work processes that may expose employees to hazards. Researchers and experts in construction safety and health management have put a lot of effort into preventing falls from height[2].

Proactive and Passive strategies can be used to prevent and minimize the seriousness of injuries generated from FFH [4]. Preventative methods that focus on safety training and education are known as proactive methods. The creation of short-term training programs and implementation of specific fall protection training programs are two examples. Passive strategies are based on the analysis of fall accident data to build future preventive measures. For instance, using accident records and data from routine safety inspections to identify factors contributing to deadly occupational falls [4]. Fixed safety equipment (e.g., guardrails and opening covers), fall arrest systems (e.g., full-body harness), and travel restraint systems (e.g., belts); are FFH preventative measures generated from an examination of accident data. On the contrary, enforcing rules may boost the usage of safety protective equipment and is said to be a reactive strategy to address the safety issue.

Several researchers have been attracted to use computer vision in their fields, such as in the construction industry for worker safety monitoring, progress monitoring, and worker action recognition to automate the manual procedures at the construction site [4–9]. Therefore, computer vision-based methods have become widely used in project progress monitoring [8], productivity analysis [5, 6], and safety monitoring [10]. Recently researchers have been focusing on computer-vision-based safety monitoring of the worker. Khan et al. [1] proposed a Mask R-CNN-based detection algorithm to check the worker's safety while working on the mobile scaffolding and achieved an overall 86% accuracy on the testing dataset. Weili Fang et al. [10] used Mask R-CNN-based algorithm for the recognition of the unsafe behavior of construction workers traversing structural support during the construction. This approach first detects and segments the worker and support, and then an overlapping detection module is used to find a relationship between workers and structural support. [10] During the testing of the algorithm, they have achieved precision and recall of 75% and 90%, respectively.

Nath et al. [6] develop three Deep learning (DL) models on YOLO architecture for the detection of PPEs (hard hats and safety vests) from images. KNN search optimization technique was created by Heidari et al. [11] to compute anchor positions that fulfill fall clearance and swing hazard.

Holes that pose a potential safety hazard(s) can be found inside buildings, on work platforms, on roofs, during roadway/bridge construction, in shops or warehouses, and other outdoor working environments. Roof ducts/drains, skylights, unfinished stairs or missing steps, unsupported ceilings/walkways, and manholes are some examples of holes/openings or gaps found on construction projects. Unprotected holes in the floor, deck, or roof have caused several serious injuries. However, falls through holes can be easily avoided with proper planning and the personal attention of the trade workers and safety manager.

3 System Development

This section presents the dataset preparation, model training, deployment on devices, and firebase database.

3.1 Dataset Preparation

A large amount of digital image data with various patterns is needed to train a vision intelligence-based detection model. As vision intelligence approaches emerge in the construction sector, obtaining labeling datasets from open-source websites remains difficult. Therefore, images of the hatch opening with enough variations were collected from three sources; (1) Google search engine, (2) random frames from YouTube videos, and (3) recorded videos in a Korean Scaffolding Institute. In this paper, different types of keywords for crawling techniques are used, such as "hole opening", "hatch opening", "partial opening on construction", "opening on construction site". As another source for image data collection, multiple videos have been recorded at the Korean Scaffolding Institute for FOO and POO in Seoul, Korea. Random frames were extracted using Fast Forward MPEG (Ffmpeg) tool in python. Data cleaning is an important step for the model's training; therefore, a two-step process is used for data cleaning to select the useful dataset to train, validate, and test the deep-learning model. The goal of the two-step process is to generate a useful dataset for the deep learning model. In the first step of data cleaning, unsuitable/unclear images such as incorrect exposure were removed.

There are eight simple techniques for avoiding overfitting, but two of them were used in this study: the Hold-Out and Data Augmentation [12]. The Roboflow platform was used to pre-process and label the 852 images. According to the Hold-out (data) technique, the total image data should be divided into train and test datasets. The common ratio is 80:20, but in this case, the Hold-out technique was used with a ratio of 87:8:5 for training, testing, and validation, respectively. Thus, image data from the training set of 1657 images, a test set

of 163, and a validation set of 80 images were retrieved. Data augmentation techniques such as crop, rotation, shear, hue, saturation, brightness, and exposure were applied to the training set to increase the dataset [12]. As a result of the augmentation process, the total number of image data after the augmentation is 1900 labeled images across the two classes (1) Opening, (2) partial opening. The dataset's labeling can be seen in the picture below:

Figure 1. Dataset Preparation

3.2 Model Training

Object detection with deep learning techniques has currently attracted many researchers worldwide, owing to its applications in our daily life. For example, business analytics, self-driving vehicles, face identification, and medical image analysis depend on object detection [13]. GPUs, CPUs, IoT devices, and embedded computers are needed to create these everyday applications. To detect FOO and POO on the construction site, the pre-trained YOLOv4 object detection algorithm is used. The blog [10] stated the rule of thumb is 1000 images for image recognition, but this number can be reduced significantly in the pre-trained model. The YOLO is implemented in Darknet, an open-source framework written in C language, and Compute Unified Device Architecture (CUDA) is used for parallel computation. Darknet establishes the network's fundamental architecture and serves as the basis for YOLO training. This architecture is simple, fast to set up, and supports both Graphical Processing Unit (GPU) and Central Processing Unit (CPU) computation [14]. The YOLO (You Only Look Once) network is an algorithm that operates to detect an object in a single stage. It processes images with a single CNN and can measure classification results and object location coordinates directly. The detection speed has been significantly improved thanks to end-to-end object positioning and classification [15].

YOLOv4 network uses CSPDarknet53 for the image features extraction and network training. [16]. After that, Path Aggregation Network (PANet) was applied as a neck network to improve the extracted features fusion, and the head of YOLOv3 is utilized to realize object detection. Fig. 2 shows the architecture of YOLOv4. The key modules of the YOLOv4-based FOO and POO detection model are as follows:

- A convolution layer, a batch normalization layer, and a Leaky-ReLU activation function made up the CBL (Convolution, Batch Normalization, and Leaky-ReLU) module.
- Both the CBL and CBM (Convolution, Batch Normalization, and MISH) modules extracted the features. The difference was that the CBM's used MISH activation mechanisms instead of Leaky-ReLU [17].
- By splitting low-level features into two sections and then fusing cross-level features, the CSP (Center and Scale Prediction) module could improve CNN's learning ability [18].

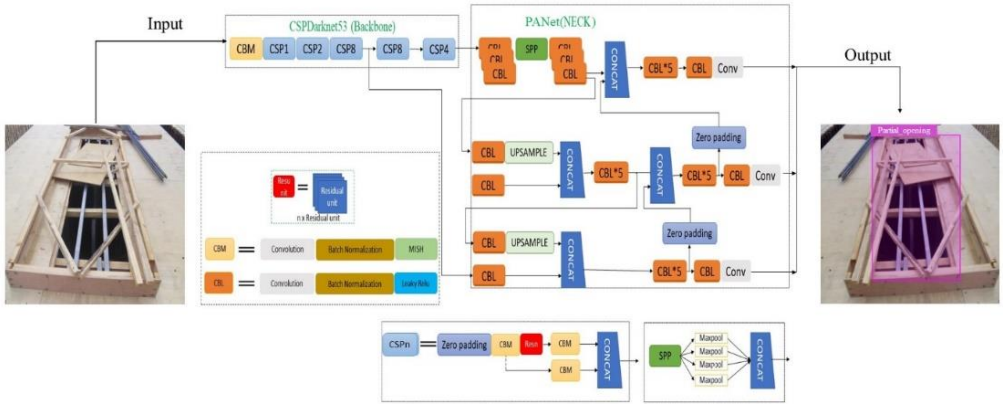

Figure 2. The architecture of YOLOv4 [19]

The POO and FOO detection based on YOLOv4; the main steps involved in the training are as follows:

- As mentioned above, this paper uses a dataset from three sources; (1) Google search engine, (2) random frames from YouTube videos, and (3) recorded videos in a Korean Scaffolding Institute. All the images are uploaded to the Roboflow platform to label the dataset into two classes, "opening" and "partial opening," with the ratio of 87:8:5.
- The batch size, learning rate, number of classes, mini-batch size, and the number of convolutional kernels in the previous layers are modified. For the detection of an object, the network input size was set to 416 pixels × 416 pixels, the learning rate set to 0.00065, batch size to 64, mini-batch size to 16, the step size to 8000-9000, filters to 21, momentum and decay rate to 0.949 and 0.0005, respectively. The parameters used in the configuration file of the YOLOv4 are shown in Table 1. At every 10,000 steps, the training process produced several models; the best model fit was used in our testing and deployed on Android devices. The training, validation, and testing set were used for training, validation, and testing of the proposed model, respectively.

Table 1. Parameters of FOO and POO detection model

Parameters	Values
Input Size	416 x 416
Batch Size	64
Learning Rate	0.00065
Momentum	0.949
Decay	0.0005
Iterations	4000
Classes	2

The Loss curve during training is depicted in Fig. 3, and it can be seen that the model learning performance was higher, and the convergence speed of the training curve was faster at the beginning of the FOO and POO detection. The blue curve represents the training loss or error on the training dataset. The mean average precision at the 50% Intersection-over-Union threshold (mAP@0.5) is indicated by the red curve that determines if the model generalizes successfully on a previously unseen dataset or validation set. The graph shows that the model has the highest mean-average-precision score of 89 percent map (best model) during training at 3600th iteration, and the final map reported as 87.7% on the last iteration. The slope of the training curve steadily decreased after the 200 iterations in training. The number of training iterations and the training curve can be considered as inversely proportional as training iterations increase, the training curve decreases.

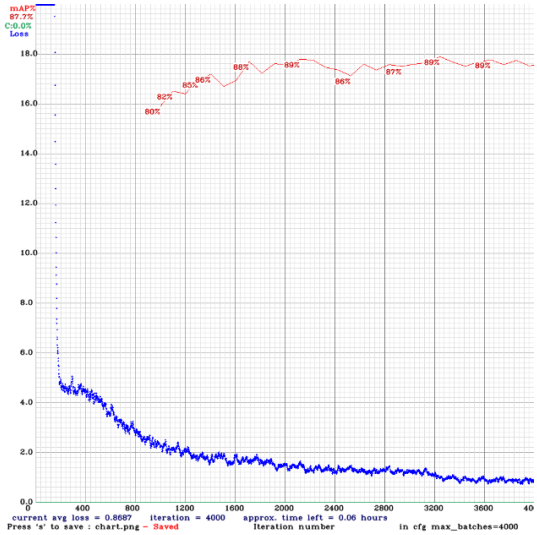

Figure 3. Loss Curve

3.3 Deployment on Devices

This section describes the methodology for deploying the trained YOLOv4 model and converting the YOLOv4 object detection model to TensorFlow Lite for on-device inference. TensorFlow Lite consists of tools used for machine learning on edge devices such as smartphones and IoT. TensorFlow Lite is the TensorFlow framework designed to visualize inference on small devices, meant to avoid a round-trip data computation to and from the server capability of real-time detection and work without internet connection. [20]. Fig .4 explains the process of FOO and POO object detection-based mobile applications. As we explained earlier about the dataset preparation and training, after that, the YOLOv4 based FOO and POO object detection model was deployed to an android application. There are two main steps to deploy a Deep Learning model on edge devices:

- The first one is converting the model into TensorFlow Lite format; For the conversion of YOLOv4 (TensorFlow model) into TensorFlow Lite, the TensorFlow Lite (TfLite) converter is used. TfLite converter takes the YOLOv4 (TensorFlow) model as input and generates the TfLite model (an optimized Flat Buffer format identified by .tflite file extension). During conversion, the quantization is applied to reduce the model size because the android studio cannot deploy a model that is larger than 250MB.
- The second essential step in the deployment is to Run inference. Inference refers to executing the model to make predictions based on input data, but

it requires Metadata. TensorFlow Lite metadata describes the model, including license words, input information (pre-processing), normalization, output information (post-processing), mapping labels. Normalization is a popular data pre-processing technique that converts values into a common scale without distorting differences in value ranges. As a result, the normalization parameter. In this case, the normalization parameters for each float model

though this is strengthened in a wall opening that should also be protected [3]. This information will then be uploaded to a database with the worker's current location (longitude and latitude)

The Geocoder API is used to determine the worker's current position using Geocoding process. The Geocoding process converts the addresses into latitude and longitude-based coordinates. These geocoded

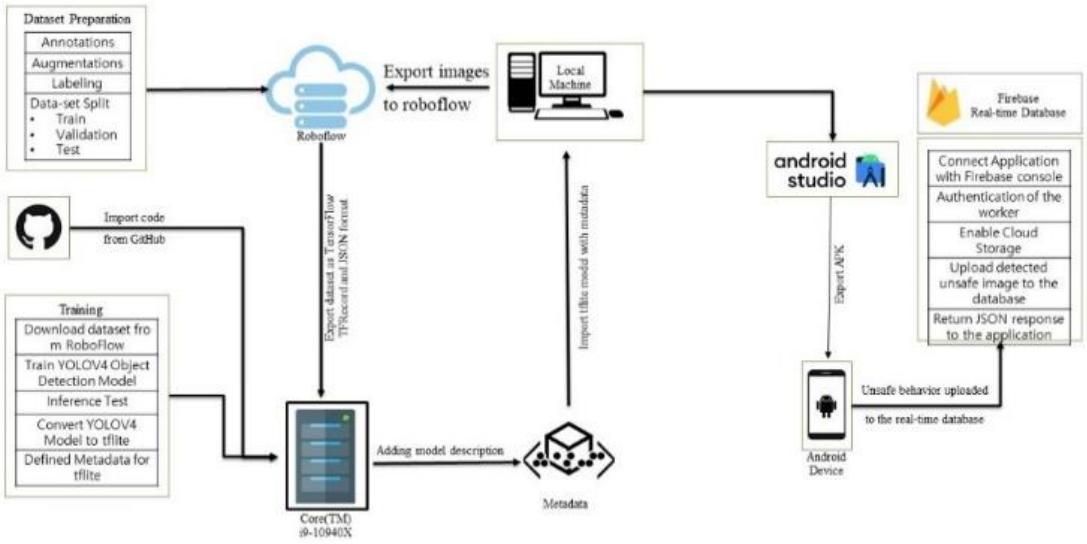

Fig. 4 Process flow of FOO and POO Detection

mean and standard deviation were 127.5. After that, the model is exported to an Android Studio[20].

-
- Now the quantized model is exported into an Android Studio Integrated Development Environment (IDE). The application is built on top of an open-source platform with a TensorFlow backend that was cloned from the GitHub repository. We have modified both the back end and the visual front end. In the backend, the real-time Firebase Database is used to record the unsafe behavior of FOO and POO on the construction site. This application encourages workers to be involved in this process, called a worker Driven approach or Pull-Reporting Approach. Supposing that the worker walks around the construction site, opens an application, and reports any unsafe activity associated with FOO and POO. The application may also recognize a vertical opening as a hole,

coordinates are further used for the placement of the markers on a map. The algorithmic component shows how the object identification and unsafe actions are uploaded to the Database using java language with android studio, as shown below:

Table 2. Algorithm

Title: Pseudo Code for FOO and POO detection
Input: Video Frames **Output:** Detect Unsafe Behavior with current location.
1. Load the .tflite model and labels files from the assets folder. 2. Set Minimum Confidence = 0.5 3. Start Camera activity to get input. 4. Extract Frame from the video.

5. While (frame != 0):
   Send frame to Detector class:
   1) If (Detected result != null and Detected result confidence >= Minimum Confidence):
      i. Draw Rectangle on Detected frame
      ii. Get the Current Location of the Worker using Geocoder API
      iii. Get the Title of Detected Result
      iv. If (Detected Result Title == "Opening"):
         - Upload Detected frame with the current location of the worker to the Opening folder in Database.
         - Return to step 4.
      v. Else:
         - Upload Detected frame with the current location of the worker to the Partial Opening folder in Database.
         - Return to step 4.
   2) Else:
      - Return to step 4.
6. Close an application to stop the process.

First, to explain the algorithm, the TensorFlow Lite model is loaded, and the labels file from the assets folder to perform object detection. The detected result confidence is compared with the minimum set confidence that is 0.5. The process of getting and passing frames to the Detector class operated continuously until the application close. The frames are being extracted from the video. An individual frame is passed to the Recognition function declared and defined in the Detector java class that returns the result. An algorithm compares the detected result confidence with the minimum confidence. The detector java class will process the frame if the detected result confidence is greater than the minimum confidence and will draw a rectangular box on the detected result. An application will get the worker's current location by using Geocoder API. Further, it checks whether the detected frame is labelled as "opening" or "partial opening" by comparing the label of the detected result. The detected result will be uploaded to the designated folder in the firebase database based on the mark, and this process will continue until the application is running.

Both options are provided in an application such as video detection as well as static image detection. Fig. 5 visualizes the four buttons, and if a worker clicks the gallery button, the gallery will open, and the user can pick an image; if the user clicks the camera button, the algorithm will identify the result and upload it to the designated folder with the current location of the Worker into Firebase database. Firebase notifies the worker that the data has been uploaded until the unsafe behavior has been registered.

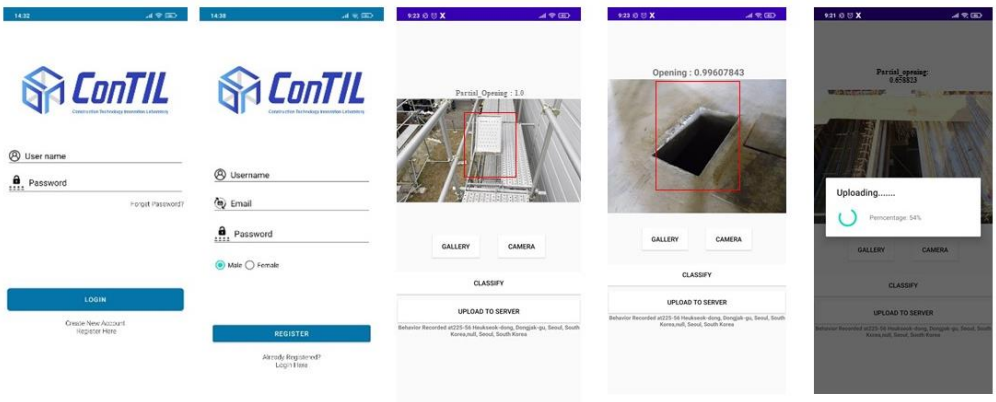

Figure 5. Android-based application

4 Results and Evaluation

To check the feasibility of the proposed model, the validation set containing 80 images of fully opened openings (FOO) and partially opened openings (POO) were used. The results can be seen in Table 3 that presents the outcomes of the performance indicators used

to evaluate the proposed model. The qualified model has a recall of 82 percent and precision of 92.35 percent, with an overall mean average precision of 87.70 percent and an intersection over union of 76.40 percent (IoU). The test data produced promising results, and the model was chosen to detect data from actual construction sites. The results demonstrated that the model could achieve high precision on real-time data.

Table 3. Performance Matrices of trained model

Evaluation Indexes	Test Results
Precision	92%
Recall	82%
Average IOU	76.40%
mAP	87.70%
F1 Score	86%
Average Loss	0.86

5 Conclusion

In this paper, the authors propose a pull-up reporting approach based on an android operating system to detect FOO and POO on the construction site to overcome the manual inspection process. YOLOv4 detector is trained on our custom pre-processed dataset and then later converted to the TensorFlow lite model by using TensorFlow API to deploy and perform inference on the edge devices. The qualified model shows that our developed application can perform better and facilitate the automation process of construction site management. The trained model has an F1 score of 86% percent, which shows how accurately our model performs predictions on the custom dataset. Moreover, a rewarding functionality and more case scenarios such as missing planks and missing guardrails would be added to the application to reward the worker who facilitates the safety manager in identifying risk at the construction site.

Acknowledgment

This study was financially supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government Ministry of Science and ICT (MSIP) [No. NRF-2020R1A4A4078916] and the "National R&D Project for Smart Construction Technology (No.21SMIP-A158708-02)" funded by the Korea Agency for Infrastructure Technology Advancement under the Ministry of Land, Infrastructure and Transport, managed by the Korea Expressway Corporation

References

[1] N. Khan, M.R. Saleem, D. Lee, M.W. Park, C. Park, Utilizing safety rule correlation for mobile scaffolds monitoring leveraging deep convolution neural networks, Comput. Ind. 129 (2021) 103448. https://doi.org/10.1016/j.compind.2021.103448.

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

[3] S. Zhang, K. Sulankivi, M. Kiviniemi, I. Romo, C.M. Eastman, J. Teizer, BIM-based fall hazard identification and prevention in construction safety planning, Saf. Sci. 72 (2015) 31–45. https://doi.org/10.1016/j.ssci.2014.08.001.

[4] E. Nadhim, C. Hon, B. Xia, I. Stewart, D. Fang, Falls from Height in the Construction Industry: A Critical Review of the Scientific Literature, Int. J. Environ. Res. Public Health. 13 (2016) 638. https://doi.org/10.3390/ijerph13070638.

[5] P. Chansik, L. Doyeop, K. Numan, An Analysis on Safety Risk Judgment Patterns Towards Computer Vision Based Construction Safety Management, in: Periodica Polytechnica Budapest University of Technology and Economics, 2020: pp. 31–38. https://doi.org/10.3311/ccc2020-052.

[6] D. Lee, N. Khan, C. Park, Stereo vision based hazardous area detection for construction worker's safety, Proc. 37th Int. Symp. Autom. Robot. Constr. ISARC 2020 From Demonstr. to Pract. Use - To New Stage Constr. Robot. (2020) 935–940. https://doi.org/10.22260/isarc2020/0129.

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

[8] Z. Kolar, H. Chen, X. Luo, Transfer learning and deep convolutional neural networks for safety guardrail detection in 2D images, Autom. Constr. 89 (2018) 58–70. https://doi.org/10.1016/j.autcon.2018.01.003.

[9] M.-W. Park, N. Elsafty, Z. Zhu, Hardhat-Wearing Detection for Enhancing On-Site Safety of Construction Workers, J. Constr. Eng. Manag. 141 (2015) 04015024. https://doi.org/10.1061/(asce)co.1943-7862.0000974.

[10] W. Fang, B. Zhong, N. Zhao, P.E.D. Love, H. Luo, J. Xue, S. Xu, A deep learning-based approach for mitigating falls from height with

computer vision: Convolutional neural network, Adv. Eng. Informatics. 39 (2019) 170–177. https://doi.org/10.1016/j.aei.2018.12.005.

[11] A. Heidari, S. Olbina, S. Glick, Automated positioning of anchors for personal fall arrest systems for steep-sloped roofs, Buildings. 11 (2021) 1–26. https://doi.org/10.3390/buildings11010010.

[12] 8 Simple Techniques to Prevent Overfitting | by David Chuan-En Lin | Towards Data Science, (n.d.).

[13] C.-Y. Wang, H.-Y.M. Liao, Scaled-YOLOv4: Scaling Cross Stage Partial Network, 2021.

[14] YOLO v4 or YOLO v5 or PP-YOLO? Which should I use? | Towards Data Science, (n.d.).

[15] J. Redmon, S. Divvala, R. Girshick, A. Farhadi, You only look once: Unified, real-time object detection, Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. 2016-Decem (2016) 779–788. https://doi.org/10.1109/CVPR.2016.91.

[16] A. Bochkovskiy, C.Y. Wang, H.Y.M. Liao, YOLOv4: Optimal Speed and Accuracy of Object Detection, ArXiv. (2020).

[17] K. He, X. Zhang, S. Ren, J. Sun, Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition, IEEE Trans. Pattern Anal. Mach. Intell. 37 (2015) 1904–1916. https://doi.org/10.1109/TPAMI.2015.2389824.

[18] C.Y. Wang, H.Y. Mark Liao, Y.H. Wu, P.Y. Chen, J.W. Hsieh, I.H. Yeh, CSPNet: A new backbone that can enhance learning capability of CNN, in: IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. Work., IEEE Computer Society, 2020: pp. 1571–1580. https://doi.org/10.1109/CVPRW50498.2020.00203.

[19] D. Wu, S. Lv, M. Jiang, H. Song, Using channel pruning-based YOLO v4 deep learning algorithm for the real-time and accurate detection of apple flowers in natural environments, Comput. Electron. Agric. 178 (2020) 105742. https://doi.org/10.1016/j.compag.2020.105742.

[20] Jim Su, How to Train a Custom TensorFlow Lite Object Detection Model, (2020). https://blog.roboflow.com/how-to-train-a-tensorflow-lite-object-detection-model/ (accessed April 17, 2021).