HUMAN-CENTERED COMPUTING AND VISUAL ANALYTICS FOR FUTURE OF WORK IN CONSTRUCTION

A Dissertation

by

NIPUN DEB NATH

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

August 2021

Major Subject: Civil Engineering

ABSTRACT

Artificial intelligence (AI) is revolutionizing various systems within the Architecture, Engineering, Construction, and Facilities Management (AEC/FM) domains. The rapid advancements in computational methods, engineering knowledge, and sensor technologies is transforming the current construction practices that are heavily reliant on human intervention. Therefore, visionaries are foreseeing that future construction works will be collaborated by humans and machines which will lead to unprecedented socio-economic outcomes in the safety, health, and productivity of construction workers. This Dissertation aims to advance the fundamental knowledge for effectively implementing human-machine collaboration in the construction site. Particularly, the ultimate objective of this Dissertation is to design AI-based autonomous systems for continuously monitoring workplace safety and productivity. Toward this goal, firstly, a content retrieval scheme is designed to analyze a large volume of construction imagery at a rapid speed. Next, an object recognition framework is developed to detect construction-related objects from digital images or videos in real-time. By further extending this framework, an automated safety monitoring system is subsequently designed to verify workers' compliance with the requirements related to personal protective equipment (PPE). Next, an AI-enabled image enhancement technique is developed to improve the quality of visual data to achieve better performance from the detection models in the preceding steps. Finally, an active vision system is proposed that enables an autonomous camera to intelligently navigates through a jobsite to monitor objects of interest for their safety and productivity.

DEDICATION

I dedicate this dissertation to my caring parents, Swapan Debnath and Alpana Debnath, and my loving wife, Projna Paromita.

ACKNOWLEDGEMENTS

There is a proverb that says – “if you want to go fast, go alone; if you want to go far, go together”. Definitely, doctoral study is a long journey that checks one’s perseverance from time to time. I am fortunate enough to have had many kind people by my side who endlessly helped and supported me to come this far.

Firstly, I would like to thank my research supervisor and doctoral committee Co-Chair, Dr. Amir H. Behzadan, for his sincere guidance throughout my entire graduate study. Also, I would like to thank my doctoral committee Chair, Dr. Stephanie Paal, and my committee member, Dr. Theodora Chaspari, for their incessant kind supports.

I am also privileged to be able to affiliate myself with several academic departments at Texas A&M University, including Construction Science (where my dissertation research was housed as a member of Connected Informatics and Built Environment Research (CIBER) Lab), Civil and Environmental Engineering (where my doctoral studies were concluded and my Ph.D. degree was conferred), Computer Science and Engineering (where I obtained my second Master’s degree), and Architecture (where my doctoral studies were initiated). My heartfelt thanks go to the faculties, staff, and my friends of these departments who made my journey at Texas A&M University a pleasant experience.

My cordial thanks also go to my friends and colleagues, particularly my peers in the CIBER Lab – Bahareh Alizadeh, Chih-Shen Cheng, Md Nazmus Sakib, and Yalong Pi – for their continuous encouragement.

Lastly, I thank my parents and my wife for their relentless patience and love.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supervised by a dissertation committee consisting of Dr. Stephanie Paal (Department of Civil and Environmental Engineering), Dr. Amir H. Behzadan (Department of Construction Science), Dr. Arash Noshadravan and Dr. Stefan Hurlebaus (Department of Civil and Environmental Engineering), and Dr. Theodora Chaspari (Department of Computer Science and Engineering).

The datasets used in this Dissertation are partially collected and labeled by Dr. Yalong Pi (post-doctoral researcher at Texas A&M Institute, and former CIBER Lab member), Mr. Kexin Feng (Ph.D. student in the Department of Computer Science and Engineering), and Mr. Chih-Shen Cheng (Ph.D. student in the Department of Civil and Environmental Engineering, and CIBER Lab member). All generated code and datasets are publicly available on CIBER Lab's GitHub page at https://github.com/ciber-lab.

All other works conducted for the dissertation were completed by the student independently.

Funding Sources

This work was supported in part by the U.S. National Science Foundation (NSF) through Grant Number CMMI 1800957. Data analysis and model training were performed on Texas A&M University's High Performance Research Computing (HPRC) clusters. The contents of this document are solely the responsibility of the author and do not necessarily represent the official views of the NSF, HPRC, or the individuals named above.

NOMENCLATURE

AI	Artificial Intelligence
AP	Average Precision
AR	Augmented Reality
BIM	Building Information Modeling
BLS	Bureau of Labor Statistics
BRISQUE	Blind (or Referenceless) Image Spatial Quality Evaluator
CNN	Convolutional Neural Network
CV	Computer Vision
DL	Deep Learning
DMLI	Dynamically Matching Local Information
DNN	Deep Neural Network
DQN	Deep Q-learning Network
ELU	Exponential Linear Unit
FC	Fully Connected
FN	False Negative
FP	False Positive
FPS	Frames Per Second
GAN	Generative Adversarial Network
GMM	Gaussian Mixture Model
IoT	Internet of Things
IoU	Intersection over Union

KNN	k-Nearest Neighbor
LR	Low-resolution
LSTM	Long Short-Term Memory
mAP	Mean Average Precision
ML	Machine Learning
MSE	Mean-Squared-Error
NB	Naïve Bayes
NSF	National Science Foundation
NN	Neural Network
OSHA	Occupational Safety and Health Administration
PPE	Personal Protective Equipment
PRID	Person Re-Identification
R-CNN	Region-based Convolutional Neural Network
ReLU	Rectified Linear Unit
R-FCN	Region-based Fully Convolutional Network
RFI	Request for Information
RL	Reinforcement Learning
RoI	Regions of Interest
RPN	Region Proposal Network
SR	Super-resolution
SSD	Single Shot MultiBox Detector
SVM	Support Vector Machine

TD	Temporal-Difference
TN	True Negative
TP	True Positive
UAV	Unmanned Aerial Vehicle
VR	Virtual Reality
YOLO	You-Only-Look-Once

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Page

CHAPTER I

INTRODUCTION

The landscape of future construction work will be significantly different from what we observe now. The rapid advancement in computational techniques, engineering knowledge, and sensor technologies is revolutionizing the current practices that are heavily reliant on manual work and direct human intervention. Inspired by this vision of the future of work, organizations engaged in frontier-level scientific research have predicted that the next generation of construction work will involve many levels of collaboration between humans and machines (NSF 2020; Autodesk 2020). While high-risk or routine tasks will be delegated to autonomous robots, for the cognitively demanding tasks, robots will work hand in hand with the human to maximize the productivity and quality of work, while minimizing project cost, duration, and risks. To achieve this goal, many researchers and practitioners are relying on CV, AI, and ML due to their remarkable success in numerous real-world applications across various domains.

From this perspective, the research presented in this Dissertation aims to pave the ground for new scientific discoveries and lay the theoretical foundation of human-centered algorithms powered by CV and AI/ML to autonomously monitor the risks and productivity in construction sites through self-governing machines. Achieving this goal, however, is not trivial and requires a rigorous and systematic multi-faceted study to integrate multiple processes and methods. Therefore, this Dissertation follows a bottom-up approach starting from the most fundamental procedures to the most advanced techniques. Chapters II and III cover the state-of-the-art and theoretical background of the most relevant AI/ML techniques used in this research. In Chapter IV, a content retrieval scheme is designed to analyze a large volume of construction imagery at a fast pace. Chapter V describes an object recognition framework designed to detect construction-related objects from digital imagery (photos and videos) in real-time. By further expanding upon this framework in Chapter VI, an automated safety monitoring system is designed to verify workers' compliance with the requirements related to PPE. In Chapter VII, an AI-enabled image enhancement technique is developed and evaluated to improve the quality of visual data in order to increase the performance of the previously developed detection models. Chapter VIII describes the details of an active vision system that allows a camera mounted on a UAV to autonomously navigate in a construction site to monitor the safety and productivity of the crew. Finally, Chapter IX provides an abridged summary along with a discussion of future research opportunities.

A detailed description of the background and motivation behind these applications is as follows. For an autonomous machine to work in collaboration with humans, within their physical proximity, it is necessary that the machine understands its surrounding in order to safely operate and accomplish the tasks delegated to it. In CV, the capacity to comprehend the surrounding is also known as scene understanding which includes the ability of an intelligent machine to recognize key objects in a complex scene followed by analyzing the contextual relationships among them (Li et al. 2009). For a vision-based autonomous system, an essential technical step toward achieving this goal is to have the ability to analyze visual data in real-time. Therefore, as the most fundamental step, this Dissertation aims to build and validate AI models that can process digital contents at a fast pace and recognize the vital objects within the scene. Apart from building the foundation of an autonomous machine with scene understanding, this AI-enabled image analysis tool has several real-world implications, including content retrieval and safety monitoring, that are discussed in the following paragraphs.

Visual data, such as images or videos, is one of the most common types of media to document construction fieldwork. Photos and videos are frequently used to prepare and document progress reports, RFI, safety training, productivity monitoring, and claims and litigation. Recently, the ubiquity of digital cameras, mobile devices (e.g., smartphones) with internet connectivity, and UAVs (a.k.a., drones) equipped with onboard cameras has exponentially increased the volume of visual data collected on a daily basis. If fully utilized, this big data can be readily used to increase the accuracy and timeliness of decision-making in construction. However, captured images and videos rarely contain rich metadata other than date, time, and (in some cases) geolocation information. Therefore, retrieving a set of specific visual contents from a large volume of image data (a.k.a., content retrieval) may turn into a non-trivial and resource-intensive task. A potential solution to this challenge is to create a semantic structure for the collection of images by using metadata tags that among others, describe the content (e.g., objects, scenes) and appearance (e.g., color, context). In pattern recognition and ML, automated generation of metadata tags for visual data using AI models is known as image/video classification. The majority of research in this direction has focused on recognizing everyday objects (e.g., flower, umbrella) and animals (e.g., cat, dog) in digital images (Krizhevsky et al. 2012; Simonyan et al. 2014). In the construction domain, previous studies have attempted at recognizing construction equipment, e.g., excavators (Zou et al. 2007), and materials (Brilakis et al. 2008) in digital imagery. However, these studies either focused on a specific application or employed a time-consuming analysis tool, rendering the content retrieval for large image collections for general applications futile. In contrast, this Dissertation creates and presents a large-scale image dataset, Pictor-v1, containing >2,000 images where each image is tagged with keywords based on the presence of general construction-related objects (e.g., building, equipment, worker). Next, a DL model, particularly, a CNN, is built and trained to perform image classification in real-time.

While image classification assigns a single label or multiple labels for the entire image, object detection localizes objects within the image and assigns a label for each of the detected objects. Compared to image classification, object detection provides more semantically rich information (e.g., the number of objects belonging to a particular class, and the spatial relationships among them) which can be utilized for more refined content retrieval. Furthermore, object detection constitutes the basis for scene understanding by providing distinctive information (e.g., type, color, position) about individual objects in the scene. To conduct object detection in this Dissertation, a second dataset, Pictor-v2 (with ~2,500 images), is created that contains instance-level annotations of construction-related objects (e.g., building, equipment, worker). Next, DL-based object detection models are examined to perform real-time detection of these objects. Since construction sites are dynamic and complex, generally, the images captured from these sites are visually diverse. Therefore, the performance of the above DL models is particularly scrutinized for different images under different visual conditions.

The next element of the Dissertation is focused on one of the most crucial applications in construction, i.e., safety monitoring. According to the BLS, in 2016-17 alone, the total number of fatal occupational injuries was the highest in construction compared to all other industries. During this period, a total of 991 fatal incidents (~19% of all the fatalities) were recorded (BLS 2019). Moreover, in 2017, 79,810 (~9% of total cases) non-fatal occupational injuries and illnesses occurred in construction which was also exorbitant (BLS 2019). Previous studies have found that many construction fatalities occur due to traumatic brain injuries (resulted from fall and electrocution) and collisions (resulted from struck by objects). Therefore, the U.S. OSHA requires that all workers properly use PPE at all times to prevent such accidents. The majority of previous studies in this area has used AI techniques to verify the use of only hard hats (Fang et al. 2018; Wu et al. 2019). Building upon past work, this Dissertation designs a more comprehensive framework that automatically monitors workers' proper usage of multiple PPE in real-time. For the purpose of training and testing AI models, another dataset, Pictor-v3, containing ~1,500 annotated images and ~4,700 instances of workers wearing various combinations of PPE components, is also developed.

Generally, vision-based algorithms (including object detection) perform extraordinarily better when the quality of the input image is high. For example, researchers have achieved remarkable improvement in recognizing human faces in everyday images, or retrieving biological information from medical images by using higher quality images (Li et al. 2018; Trinh et al. 2014). However, since construction projects often take place in harsh environments, the ability to collect good quality and well-lit imagery may be limited, which in turn, can significantly lower the performance of DL-based object detection models. Therefore, this Dissertation also proposes an AI-based image enhancement technique that particularly uses a GAN to increase the input image resolution for fast and reliable object detection.

The previously mentioned methods for object detection and safety monitoring could not successfully perform the intended task if the object of interest is considerably occluded by another object in the scene, as viewed by the camera. In fact, occlusion is one of the most prevailing challenges to CV-based algorithms (Hoiem et al. 2011). For example, a previous study has found a significant drop in the performance of facial recognition when the face is partially occluded (Ekenel et al. 2009). In the context of this research, a good example is safety monitoring when a worker's PPE requirement cannot be verified with high certainty due to the occlusion. A potential remedy to this problem is to revisit the construction site and capture more images from multiple viewing angles so that the worker appears adequately visible in the image. However, for a large active construction site with multiple active crews and moving equipment, it may not be feasible to set up, calibrate, and maintain multiple cameras to continuously collect occlusion-free images. Therefore, this Dissertation presents a smart active vision system that gives a UAV-mounted camera the ability to navigate the scene and operate autonomously in search of an optimal viewpoint from which sufficiently good and occlusion-free visual data can be captured to reliably verify workers' safety compliances. The camera mechanism is trained with RL and tested in both a simulated environment (feasibility testing) as well as a real-world setting (captured in a warehouse with 360-degree video) to validate the proposed method. This RL-based technique supports and promotes human-machine collaboration in construction sites by delegating the routine task of continuous monitoring of safety compliance to the autonomous robots, thus allowing workers and safety managers to dedicate time and effort to more cognitively demanding tasks such as high-level analysis and decision-making.

CHAPTER II

STATE OF THE ART

AI and Deep Learning

Traditional ML algorithms require careful and meticulous engineering of features that might be only relevant to specific tasks and set of classes (Kolar et al. 2018). However, for content-rich construction images that cover a large visual field containing diverse and complex categories of objects in various ambient conditions (e.g., lighting, landscape), automatic feature extraction through DL is more practical. Particularly, CNN-based algorithms have gained more traction due to their ability to self-learn features from a given dataset without demanding exorbitant computational power (Kolar et al. 2018; LeCun et al. 1998). LeCun et al. (1998) proposed a precursor to the modern CNN algorithms which can recognize handwritten digits in an image. However, recent CNN models are now capable of classifying images into 1,000 different categories (Krizhevsky et al. 2012; Simonyan et al. 2014), or identifying 9,000 different object types in images (Redmon et al. 2017). Nonetheless, these models are limited to detecting only everyday objects (e.g., printer, umbrella, bicycle, dog).

For object detection problems in CV, region-based CNN (a.k.a., R-CNN) (Girshick et al. 2014) is one of the most prevailing examples of state-of-the-art algorithms. Particularly, R-CNN uses selective search to identify RoI, followed by using CNN to extract features from each region, and finally applying SVM to classify the object in that region (Girshick 2015; Girshick et al. 2014). However, due to excessive time and space requirements to run this algorithm, several faster variants of it, e.g., Fast R-CNN (Girshick 2015), and Faster R-CNN (Ren et al. 2017) have been also proposed. For example, Faster R-CNN comprises of RPN, which is a fully convolutional network for proposing ROIs, followed by the Fast R-CNN algorithm for performing final object detection (Ren et al. 2017). While R-CNN and Fast/Faster R-CNN output rectangular bounding boxes for each detected object, another variant of R-CNN, namely Mask R-CNN (He et al. 2017), can output segmentation masks of irregular shapes. Particularly, Mask R-CNN has an extra branch to output the segmentation masks in addition to the existing branches of Faster R-CNN that output classification labels and bounding boxes (He et al. 2017). Another variant, R-FCN, eliminates the computationally extravagant fully connected layers in R-CNN and uses only the convolutional layers for faster yet accurate object detections (Dai et al. 2016). Unlike region proposal-based methods, YOLO (Redmon et al. 2016; Redmon et al. 2017, 2018) and SSD (Liu et al. 2016) algorithms combine the classification and localization tasks into one single neural network which significantly reduces the computational burden. According to a comparison of the performance of different algorithms performed by Liu et al. (2016), only the YOLO algorithm can perform detection in real-time and. Therefore, considering the ability of YOLO for fast yet accurate object detection, it is used in this Dissertation.

Previous studies have shown that vision-based AI algorithms perform poorly if the quality of the input image is low. Therefore, many researchers have attempted to improve the quality of input images to achieve better performance from AI models. For example, researchers have achieved a 30% improvement in facial recognition by deblurring low-quality photos (Li et al. 2018). Similarly, in medical imaging, high-resolution imagery is desirable to retrieve vital biological, anatomical, physical, and metabolic information which might be difficult to catch in a low-resolution (noisy or blurry) image (Trinh et al. 2014). Another area where higher resolution images capture more crucial pieces of evidence for future investigations includes video surveillance (e.g., for public safety, traffic monitoring, military reconnaissance) (Kumar et al. 2016). From these motivations, there have been previous attempts at investigating ways to enhance image quality. For instance, an example-based method, that relies on training images, is applied to learn a general process to enlarge an image and restore missing pixels with rich and fine details based on the spatial contexts in the given low-resolution image (Freeman et al. 2002). To this end, DL-based methods have achieved remarkable performance in this task (Ledig et al. 2017). Particularly, past research has found that GAN is uniquely reliable in producing realistic and natural up-sampled images (Ledig et al. 2017). Therefore, in this Dissertation, a GAN-based method is designed and validated to enhance the resolution (quality) of an input image.

AI/ML for Automation in Construction

Within the construction domain, various studies have utilized AI/ML algorithms to automate the process of visual recognition in construction site imagery. For example, Chi et al. (2011) used NB, and NN classifiers to detect workers, loaders, and backhoes. Son et al. (2014) used a voting-based ensemble classifier combining several base classifiers, e.g., SVM, NN, NB, decision tree, logistic regression, and KNN, to identify construction materials (e.g., concrete, steel, and wood) in an image. Dimitrov et al.

(2014), and Han et al. (2015) used one-vs-all multi-class SVM to classify major construction materials (around 20 types).

Recently, researchers in this domain have documented the use of CNN for visual recognition, however, primarily for construction safety. For examples, Kolar et al. (2018) used CNN to detect safety guardrails, Siddula et al. (2016) combined the GMM (Zivkovic 2004) with CNN to detect objects in roof construction, and Ding et al. (2018) integrated LSTM model (Hochreiter et al. 1997) with CNN to recognize unsafe behavior (e.g., climbing a ladder) of construction workers. For monitoring PPE compliance, most existing vision-based methods focus on identifying only hard hats. For example, Fang et al. (2018) used R-CNNs to detect if a worker is not wearing a hard hat. Wu et al. (2019) proposed an SSD-based algorithm to detect hard hats. Mneymneh et al. (2018) isolated moving workers (by detecting motion) in videos and identified if any hard hat was in or around the top area of a worker's detection box. Similarly, Xie et al. (2018) used fully convolution-based algorithms to detect workers' hard hats. However, to date, only a few studies have been directed at identifying multi-class PPE harnessing the power of CNN-based DL algorithms. Among the scarce examples of multiple PPE detection, a commercially available software, named *smartvid.io*, applies an AI-driven algorithm to detect multiple PPE components (e.g., hard hat, safety vest, gloves, safety goggles, and steel toe shoes) (smartvid.io 2017). While existing work facilitates the detection of only the presence of PPE components, it is also valuable to recognize the color of the PPE components. For example, in construction sites, color-coding schemes are widely used to differentiate roles, trades, or access rights (Highways England 2016). Therefore, having the ability to identify the color of PPE components can provide additional insights into the type of activities that are taking place in a particular location within the site, as well as monitoring site security and crew productivity.

Outside the immediate domain of construction safety, Luo et al. (2018) proposed a method that uses R-CNN to detect 22 classes of construction-related objects and predict construction activities based on the spatial relevance between the detected objects. Kim et al. (2018) used R-FCN to detect different types of construction equipment. However, the majority of the aforementioned studies use object detection algorithms that are computationally intensive and require heavy processing power to perform analyses on high volumes of visual data. Therefore, there is also a need, within the construction domain, to investigate fast algorithms for performing object detection in real-time. Although previous studies (Ren et al. 2015) have defined "real-time" as processing at least 5 FPS, more recent studies (Redmon et al. 2016; Redmon et al. 2017) prefer higher FPS to be considered as real-time. At such a fast rate, detection of PPE as a standalone application or in conjunction with other construction-related objects (e.g., building, equipment) allows the identification of more complex and subtle spatiotemporal relationships that might not be possible to detect otherwise. For example, an employee, not wearing PPE while working in close proximity to heavy construction equipment (e.g., excavator), is exposed to a high risk of being struck by the equipment. To accurately track object movements in a live-streamed video and predict an imminent collision, it is essential to have an extremely fast algorithm that can repeatedly process videos frame-by-frame and provide feedback in real-time. Moreover, since a fast algorithm is less computationally expensive, it will significantly reduce the upfront cost associated with securing and operating computational resources, allowing it to be launched on mobile devices and even on light-weight drones (Kyrkou et al. 2018).

Problem Statement and Contributions to the Body of Knowledge

The review of the literature highlights the limited number of studies that have systematically addressed the problem of recognizing common construction objects for general applications or detecting multiple PPE for verifying safety compliance. Moreover, current research in this area lacks a thorough investigation of crowdsourced image collection (in addition to web mining) to build training datasets for DL models. Furthermore, the tradeoff between detection speed and accuracy needs to be thoroughly examined since a faster model (i.e., YOLO) may pose weaknesses especially when tested under different visual conditions.

For the particular task of PPE detection, since past work has primarily focused on a single PPE type (i.e., detecting only hard hats), it is necessary to develop a general framework for detecting multiple PPE components. However, detecting multiple PPE components in isolation may not paint a full picture of whether a worker is properly wearing those PPE components. Therefore, more work is needed to design algorithms that not only do detect PPE components but also recognize the contextual relationship between them and the worker and, then, use this information to verify compliance with PPE requirements, all in real-time. Furthermore, since the color of the PPE component is useful to be recognized, an integrated approach to simultaneously verify the PPE compliance and identify the color of the PPE components is also desirable.

Previous work indicates that the major challenges that hinder the performance of AI-based visual recognition models include low-quality images and occluded objects of interest. In the general domain, past studies have investigated various methods for improving image quality. However, it is necessary to examine those methods with construction imagery and verify the performance of AI models specifically developed for construction-related applications. Furthermore, to date, there is no study in the construction domain that investigates the use of an active vision camera to intelligently search for occlusion-free views of construction objects (e.g., workers). In summary, the primary contributions of this Dissertation are as follows:

1. Develop large image datasets, through crowdsourcing and web mining, for various vision-based applications in the construction domain.
2. Examine the performance of object detection models trained and tested on various subsets of crowdsourced and web mined images.
3. Restructure the YOLO model to detect three common classes of construction-related objects, namely building, equipment, and worker, in construction sites, and compare the performance of YOLO-v2 and YOLO-v3, two variants of the YOLO algorithm.
4. Investigate the strengths and weaknesses of the YOLO model in detecting construction objects of different sizes and in environments with varying levels of crowdedness and lighting conditions and discuss potential ways to improve the performance of the YOLO model considering the revealed weaknesses.

5. Design YOLO-based algorithms to detect multiple PPE components (i.e., hard hat, safety vest), and verify workers' compliance with PPE-related requirements in real-time.
6. Develop algorithms to detect not only the presence of PPE on workers but also to recognize the color of the detected PPE component, simultaneously.
7. Develop GAN models to enhance the quality of construction imagery and compare the performance of YOLO models on low-quality images and GAN-improved images, for various construction-related applications.
8. Design the underlying methods and algorithms for an active vision camera capable of autonomously searching for occlusion-free views of objects of interest (e.g., workers, forklift) in a construction site, train the AI model through RL, and evaluate the model by testing it in a simulated environment of a construction site and a real-world setting of a warehouse (captured with 360-degree video).

CHAPTER III

THEORETICAL BACKGROUND

Convolutional Neural Network (CNN)

Similar to the traditional NN, CNN consists of a series of layers (i.e., input, hidden, and output layers). However, in CNN, the first few hidden layers are convolutional layers where convolution and pooling operations take place (LeCun et al. 2015). Each convolution operation outputs a numerical value by applying a filter (i.e., a matrix of weights) to a sub-region of an image (Kolar et al. 2018). A sample convolution operation involving a 3×3 filter is shown in Figure 1(a). A pooling operation, on the other hand, is performed to merge semantically similar features into one, thus reducing the size of the image (a.k.a., sub-sampling) (LeCun et al. 2015). Figure 1(b) illustrates max-pooling, one of the most commonly used pooling operations, where a 2D image is divided into fixed-sized sub-regions (i.e., kernels) and the maximum value in each sub-region is passed to the next layer. The remaining hidden layers are FC layers that are similar to traditional NN. Of note, while working with small training data, to prevent overfitting, some hidden units are often randomly turned off in a process called dropout (Hinton et al. 2012).

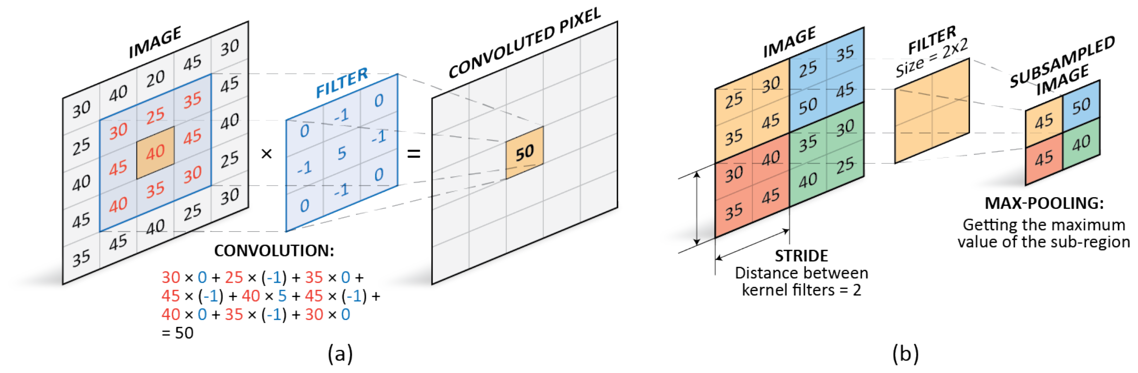


Figure 1. Example of (a) convolution operation performed with 3×3 filter and (b) max-pooling operation performed with a 2×2 filter. Reprinted with permission from Nath et al. (2019).

Activation function

The ReLU non-linear activation function is applied to the output of each hidden convolutional or FC layer to accelerate convergence (Krizhevsky et al. 2012). While the activation functions are the same at each hidden layer of single-label and multi-label classifier model, they are different at the output layer. For the single-label classifier model, the softmax activation function (Murphy 2012) (Equation 1) is used at the output layer, whereas for the multi-label classifier model, the Sigmoid activation function (Friedman et al. 2001) (Equation 2) is used.

$$\sigma_{\text{softmax}}(\boldsymbol{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \tag{1}$$

$$\sigma_{\text{sigmoid}}(z_i) = \frac{e^{z_i}}{e^{z_i} + 1} \tag{2}$$

Here, z_i is the output value of ith node in the output layer, and $\boldsymbol{z}$ is the vector output of the output later, i.e., $\boldsymbol{z} = \{z_1, \ldots z_K\}$.

Loss function

For single-label classification, multi-class cross-entropy (Friedman et al. 2001) is used as a loss function. This loss function is defined by Equation 3. On the other hand, for multi-label classification, the loss function is defined as the sum of binary cross-entropy (Buja et al. 2005) over all classes, as formulated in Equation 4.

$$L_{\text{single\_label}}(\boldsymbol{y}, \boldsymbol{p}) = -\sum_{i=1}^{N}\sum_{c=1}^{N_c} y_{i,c} \log(p_{i,c}) \tag{3}$$

$$L_{\text{multi\_label}}(\boldsymbol{y}, \boldsymbol{p}) = -\sum_{i=1}^{N}\sum_{c=1}^{N_c} \left[y_{i,c} \log(p_{i,c}) - (1 - y_{i,c}) \log(1 - p_{i,c})\right] \tag{4}$$

Here, N is the total number of samples, N_c is the total number of classes, $y_{i,c}$ and $p_{i,c}$ are the ground-truth label and predicted label, respectively, for the ith sample and cth class, and $\boldsymbol{y}$ and $\boldsymbol{p}$ are matrices containing all the ground-truth and predicted labels, respectively, i.e., $\boldsymbol{y} = [y_{i,c}]$ and $\boldsymbol{p} = [p_{i,c}]$ for $i = 1,2, \dots N$, and $c = 1,2, \dots N_c$. To note, the ground-truth labels ($y_{i,c}$) are presented as binary numbers with the value of one (1) indicating that the sample belongs to the corresponding class, and zero (0) meaning that it does not belong to that class.

You-Only-Look-Once (YOLO)

A schematic diagram of the YOLO algorithm is shown in Figure 2. As shown in this Figure, YOLO initially divides the input image into a $S \times S$ grid and predicts M bounding boxes of different shapes (a.k.a., anchor boxes) for each grid cell, each defined by a $(N + 5)$-dimensional vector, where N is the number of classes. The values t_x, t_y, t_w, and t_h are associated with *x*- and *y*-coordinates of the center, the width, and the height of the box, respectively. The value p_0 (a.k.a., objectness score) represents the probability that an object is present inside the bounding box. The remaining values are N conditional probabilities, $P(C_i|object)$, indicating the probability of the object belonging to the class C_i where $i = 1,..,N$, given that such object is present inside the box.

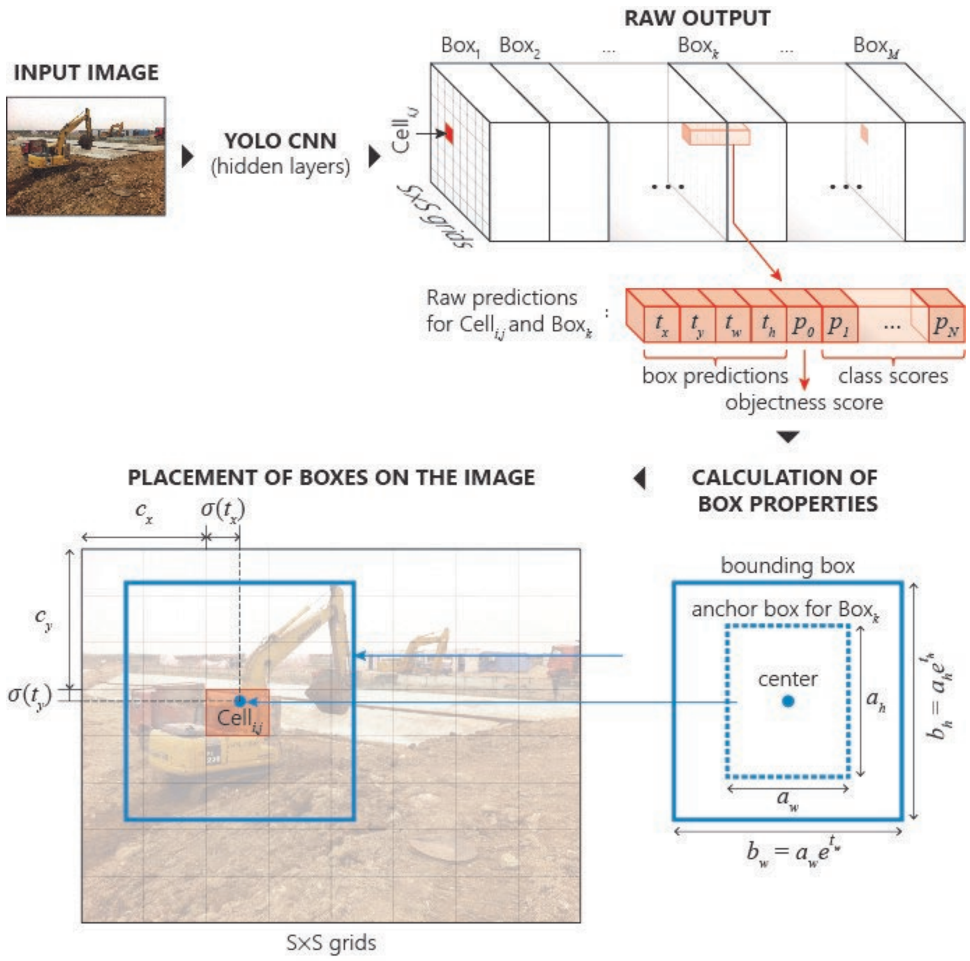


Figure 2. Schematic diagram of the YOLO algorithm. Reprinted with permission from Nath et al. (2020).

The architecture of the YOLO-v3 model is shown in Figure 3. This model consists of convolutional, residual, and output blocks. Particularly, it has three output layers to detect three different sizes of objects – large, medium, and small.

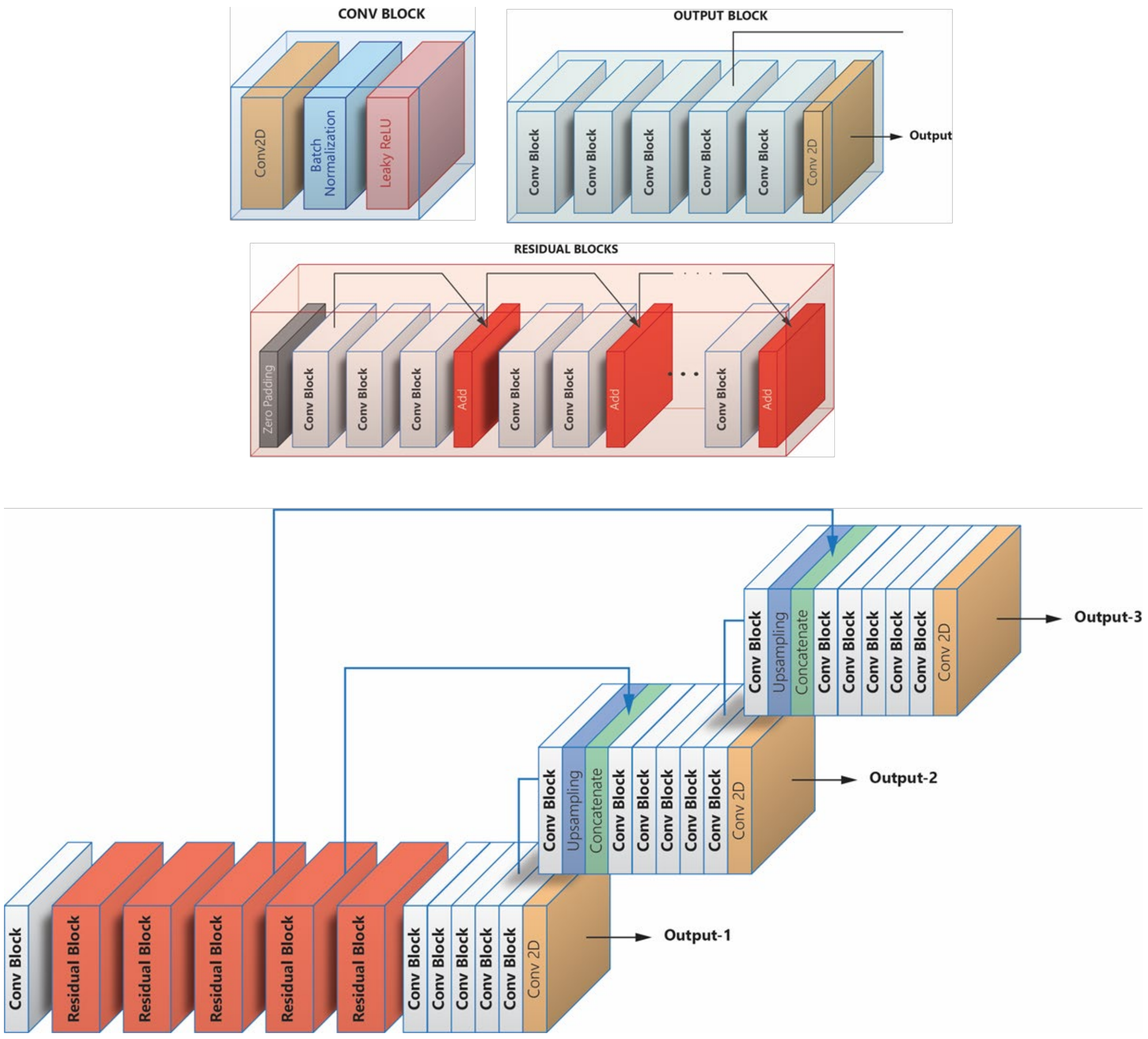


Figure 3. Architecture of the YOLO-v3 model.

Generative Adversarial Network (GAN)

A GAN model has two components, a generator G and a discriminator D (Goodfellow et al. 2014). For the image quality enhancement problem, given a low quality or low-resolution image I^L, the objective of the generator G is to generate the same image but with higher resolution and enhanced quality. This GAN-generated image is also called the super-resolved image (Ledig et al. 2017) and is denoted as I^S.

For each low-resolution image (I^L) used to train the network, there is a ground-truth high quality counterpart image (I^H). Given a I^S (generated) or I^H (real) image, the goal of the discriminator D is to accurately distinguish between them.

In a training scheme, known as *adversarial training*, G and D compete with one another to improve their performance (Goodfellow et al. 2014). In particular, during the training, G aims to generate I^S images similar to I^H images so that D fails to notice the differences between the two images. In contrast, D tries to learn more subtle differences between I^S and I^H images so that G cannot deceive it. From the perspective of game theory (Freund et al. 1996), this process constitutes a minimax game between two agents, G and D, and the game settles when each agent achieves the minimum level of competency that is perceived as maximum by the other agent (Goodfellow et al. 2014). Upon successful training, it is expected that the discriminator D can differentiate I^S and I^H images with high accuracy. However, more importantly, the generator G learns to generate high-quality I^S images that are difficult to tell apart from the I^H images, even by a high performing D.

Transfer Learning

For a particular dataset, a CNN model can be trained from scratch. However, to achieve best results, a large amount of training data coupled with the proper selection of optimal hyper-parameters (e.g., number of layers, number of nodes in each layer, filter size, number of epochs, learning rate, dropout) is required, which can make the training process extremely slow and time-consuming (Kolar et al. 2018). One way to overcome this challenge is to perform transfer learning, i.e., using a CNN model (e.g., GoogleNet, AlexNet, VGG-16) pre-trained with a different but related dataset (a.k.a., source dataset), and partly re-trained with the desired dataset (a.k.a., target dataset). Particularly, transfer learning allows the model to remember high- and mid-level features (e.g., edge, shape, color) learned from the source dataset and apply these features (with minor adjustment) to effectively distinguish the classes in the target dataset (Oquab et al. 2014). Building upon previous studies that have found significantly better and consistent performance using transfer learning (Oquab et al. 2014; Shin et al. 2016), for the image-level content retrieval, this Dissertation utilizes the VGG-16 architecture, pre-trained on the ImageNet dataset (Simonyan et al. 2014). This model is particularly selected for its wide adaptation in various domains, consistent performance comparable to the state-of-the-art techniques (Simonyan et al. 2014), and manageable size (i.e., only 16 convolutional layers) that allow porting the model on embedded systems (e.g., smartphones, drones, autonomous vehicles, handheld smart devices) with limited computational power (Alippi et al. 2018).

Reinforcement Learning (RL)

RL, a branch of ML, is a trial-and-error method of learning behavior (i.e., what actions to take in which situations) through continuous interaction with the environment (Chollet 2018; Géron 2019; Sutton et al. 2018; Szepesvári 2010). Examples include but are not limited to Google's AlphaGo learning to play a game named Go (Silver et al. 2017; Silver et al. 2016), self-learning robots learning how to walk (Yang et al. 2020; Haarnoja et al. 2018), and self-driving cars learning how to drive on roads (Liang et al. 2018; Pan et al. 2017).

RL terminology

In an RL problem, the learner is often referred to as an agent (Chollet 2018; Géron 2019; Sutton et al. 2018; Szepesvári 2010). Everything surrounding the agent, that can be interacted with, is called the environment (Chollet 2018; Géron 2019; Sutton et al. 2018; Szepesvári 2010). In the RL problem of this Dissertation, the moving camera is the agent while the construction site or warehouse and all objects within it comprise the environment. At any given time, the part of the environment that is accessible by the agent can be described in a mathematical form which is called the state of the environment (Géron 2019; Sutton et al. 2018). For example, the moving camera (i.e., agent) can see some part of the site (i.e., environment) and capture it as an image (i.e., a matrix of pixel values). The captured image is considered as the current state of the environment.

An RL problem is called continuous if the behavior of an RL agent can continue indefinitely (e.g., a robot walking) (Sutton et al. 2018). On the contrary, if the behavior can be broken down into episodes (e.g., a match in the AlphaGo game), the problem is called episodic (Sutton et al. 2018; Szepesvári 2010). The last state of an episode is called the terminal state (Sutton et al. 2018; Szepesvári 2010). In this Dissertation, the RL problem is episodic where one episode consists of starting from the initial camera position and moving to a position from where the object of interest is sufficiently visible.

In an RL problem, at one time step, the agent can perform an action by transitioning from the current state to the next state of the environment (Chollet 2018; Géron 2019; Sutton et al. 2018; Szepesvári 2010). For example, in the RL problem of this Dissertation, the camera can move “up”, “down”, “left”, “right”, or stay at the current position (a.k.a., “do nothing”). All such actions (except for “do nothing”) alter the position of the camera and, thus, allow it to capture a slightly different image (i.e., state) of the site (i.e., environment). If the camera decides to stay at the current position, i.e., when it selects the action “do nothing”, the job is considered done and the episode is terminated.

As soon as the agent performs an action, it receives feedback from the environment which indicates the merit of the action in meeting the ultimate learning goal of the problem. This feedback is described in a numerical form and referred to as reward (Chollet 2018; Géron 2019; Sutton et al. 2018; Szepesvári 2010). The RL problem should be mathematically formulated in such a way that maximizing the reward also leads to meeting the learning goal (Sutton et al. 2018; Van Seijen et al. 2017). For example, in the RL problem of this Dissertation, the goal of the camera is to learn to take actions (move or stay) in such a way that workers are adequately visible from its vantage point. To remove the notion of subjectiveness from this description of the goal, the term “adequately visible” will be precisely defined in the upcoming Section. Nonetheless, it is also important (and challenging) to define the reward function so that the camera can achieve “adequate visibility” (whatever it is) of the worker by maximizing the cumulative reward received through a series of actions taken in every episode.

RL vs. other ML problems

RL is different from commonly used supervised ML in many aspects. In supervised ML, the machine learns from a given set of training data labeled by a supervisor (Chollet 2018; Géron 2019). However, in RL, the agent actively searches for the training data (i.e., state and reward) in the environment which is, however, not labeled (i.e., it is not explicitly indicated what is the best action to take in each state) (Sutton et al. 2018). Moreover, in supervised ML, each sample is associated with a single ground-truth label (Chollet 2018; Géron 2019). For example, given an image of a cat, the ground-truth label for classifying the image will always be "cat" regardless of when it is used during the training. However, in RL, one sample might have multiple different consequences (Sutton et al. 2018). For example, given an image of an occluded worker in a construction site, for better visibility of the worker, the best action might be moving to left or right (or other directions) depending on what actions were taken before and will be taken next. Therefore, the decision in an RL problem does not depend only on the current state but also on the prior and subsequent states.

In practice, RL can be considered as online learning since not all training data are available at once (Szepesvári 2010); the RL agent explores the environment and collects the training data. In one iteration of training, the agent considers only the training data available up to that point. In case the environment changes over time (a.k.a., non-stationary environment), the agent only considers the newest training samples (Sutton et al. 2018). However, in the RL problem of this Dissertation, it is assumed that the environment is stationary (Sutton et al. 2018), i.e., does not change over time.

Q-value

In each state, the RL agent can take multiple actions. Previously, it was stated that the agent should take the action which will ultimately lead to the maximum reward, i.e., the action with the highest value. The value of the action in each state is called action value or Q-value (Géron 2019; Szepesvári 2010; Sutton et al. 2018), and is mathematically denoted as $Q(S_t, a)$, where S_t is the state at time t and a is the action. Therefore, the agent should take the action with the maximum Q-value, as shown in Equation 5.

$$a_{\text{greedy}} = \text{argmax}_a Q(S_t, a) \tag{5}$$

Assume, as shown in Figure 4, at state S_t, the agent takes action a and receives a reward r_t, termed as the immediate reward. This leads the agent to state S_{t+1}, at which point it will take action a' and receives a reward r_{t+1}, termed as the future reward. After n steps, i.e., at time $t + n$, the agent reaches to the terminal state and receives a reward r_{t+n}. The future reward received at the terminal state (i.e., r_{t+n}) is also referred to as the termination reward.

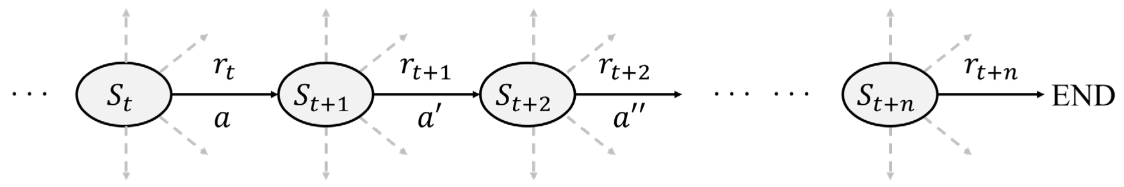


Figure 4. Rewards at each step of an episode.

Subsequently, the Q-value of action a at state S_t, $Q(S_t, a)$ is defined as the sum of all the rewards that will have been received, as expressed by Equation 6,

$$Q(S_t, a) = r_t + r_{t+1} + r_{t+2} + \cdots + r_{t+n} \tag{6}$$

However, with this definition, the agent might want to continue the task forever ($n = \infty$) to maximize the cumulative reward. Therefore, the concept of discounting is used (Géron 2019; Szepesvári 2010; Sutton et al. 2018) at each state by giving more importance to the immediate reward than the future rewards, such that the agent does not heavily rely on future rewards. Using a discount rate γ ($0 \leq \gamma \leq 1$), the Q-value can be redefined as in Equation 7,

$$Q(S_t, a) = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots + \gamma^n r_{t+n} \tag{7}$$

Equation 7 can be further rewritten by factoring γ in the second and subsequent terms in RHS, as shown in Equation 8, where $r_{t+1} + \gamma r_{t+2} + \cdots + \gamma^{n-1} r_{t+n}$ denotes the immediate reward received at state S_{t+1} for taking action a', plus the discounted rewards received subsequently. Therefore, $Q(S_t, a)$ can be expressed as the sum of the immediate reward at state S_t and $Q(S_{t+1}, a')$.

$$Q(S_t, a) = r_t + \gamma(r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots + \gamma^{n-1} r_{t+n}) = r_t + \gamma Q(S_{t+1}, a') \tag{8}$$

However, taking action a' at state S_{t+1} may not maximize the cumulative reward received at state S_t. Rather, at state S_{t+1}, the action with the highest Q-value, i.e., $\max_{a'} Q(S_{t+1}, a')$ should be taken. Thus, Equation 8 is further modified as shown in Equation 9,

$$Q(S_t, a) = r_t + \gamma \max_{a'} Q(S_{t+1}, a') \tag{9}$$

This recursive relationship has two important implications. First, one can estimate $Q(S_t, a)$ based on the immediate reward r_t and the Q-values of the actions in the next state. Using dynamic programming to estimate the values will significantly reduce computation time (Sutton et al. 2018). Secondly, it forms the basis of Q-learning which we will be discussed at length in the following Section.

Q-learning

The term Q-learning refers to the algorithm behind learning Q-values in RL (Sutton et al. 2018). As Equation 9 implies, given an estimation of $Q(S_{t+1}, a')$, one can approximate $Q(S_t, a)$. Here, the process of achieving a more accurate estimation of the Q-values in an iterative process, a.k.a., TD learning (Géron 2019; Szepesvári 2010; Sutton et al. 2018) is discussed. Assume old estimations of $Q^{old}(S_{t+1}, a')$ and $Q^{old}(S_t, a)$ are given, after one iteration of training, a new estimation of $Q(S_{t+1}, a')$ is obtained. Based on this new estimation, using Equation 9, the Q-value for S_t and a (a.k.a., TD target) is estimated using Equation 10,

$$\text{TD target} = r_t + \gamma \max_{a'} Q(S_{t+1}, a') \tag{10}$$

The difference between TD target (potentially better estimation of $Q(S_t, a)$) and $Q^{old}(S_t, a')$ is termed TD error or δ_t, as written in Equation 11,

$$\delta_t = r_t + \gamma \max_{a'} Q(S_{t+1}, a') - Q^{old}(S_t, a) \tag{11}$$

In the most simplistic approach, adding δ_t to the old estimate $Q^{old}(S_t, a)$ yields a new estimate of $Q(S_t, a)$. However, it is important to note that TD error in Equation 11 is based on the estimation of $Q(S_{t+1}, a')$, which might be affected by the stochasticity of the RL problem, making it inaccurate. Therefore, the TD error is multiplied by a factor α ($0 < \alpha < 1$), and the product is added to $Q^{old}(S_t, a)$ to obtain $Q^{new}(S_t, a)$, as expressed in Equation 12. The term α, a.k.a., the learning rate (a.k.a. step size) (Géron 2019; Szepesvári 2010; Sutton et al. 2018), allows the agent to incrementally learn from the newest experience one step at a time, rather than drastically change the previous learnings.

$$Q^{new}(S_t, a) = Q^{old}(S_t, a) + \alpha[r_t + \gamma \max_{a'} Q(S_{t+1}, a') - Q^{old}(S_t, a)] \tag{12}$$

Generally, a large value of α might destabilize the training by correcting the Q-values too much at each step of training (Géron 2019). On the other hand, a small value of α might lengthen the training time since at each time only a small correction is made (Géron 2019). Therefore, a careful experimentation with different values of α is needed to find an optimum α that balances the training stabilization and training time.

Deep Q-learning

A DNN model can be utilized to predict the Q-values. The model will take the state S_t as input, and outputs $Q(S_t, a)$ for each action a. The Q-learning for DNN is known as deep Q-learning (Van Hasselt et al. 2015; Silver et al. 2016; Géron 2019; Sutton et al. 2018). In DNN, the weights are updated based on the loss L. The update rule is shown in Equation 13, where W^{old} and W^{new} are old and new weights, respectively, and α is the learning rate.

$$W^{new} = W^{old} - \alpha \left[\frac{\partial L}{\partial W^{old}}\right] \tag{13}$$

Comparing Equation 12 and Equation 13, one can think of the estimation of Q-values (Equation 12) as a weight update task (Equation 13). This leads to the definition of loss L shown in Equation 14, which suggests that loss L would be the MSE. Therefore, in deep Q-learning, MSE is used as the loss function (Géron 2019; Sutton et al. 2018).

$$-\frac{\partial L}{\partial Q^{old}(S_t, a)} = r_t + \gamma \max_{a'} Q(S_{t+1}, a') - Q^{old}(S_t, a)$$

$$\Rightarrow \int \frac{\partial L}{\partial Q^{old}(S_t, a)} \mathrm{d}Q^{old}(S_t, a)$$

$$= -\int [r_t + \gamma \max_{a'} Q(S_{t+1}, a') - Q^{old}(S_t, a)] \, \mathrm{d}Q^{old}(S_t, a)$$

$$\Rightarrow L = \frac{1}{2}[r_t + \gamma \max_{a'} Q(S_{t+1}, a') - Q^{old}(S_t, a)]^2 + C$$

$$\Rightarrow L = \frac{1}{2}\delta_t^2 + C \tag{14}$$

Exploration vs. exploitation

It was previously established that through iterating Equation 12, an increasingly better estimation of $Q(S_t, a)$ can be achieved. This, however, requires the agent to take action a at state S_t many times. However, if the environment is too large (i.e., containing too many states and allowing too many actions), the excessively large computational time might hinder the ability to try out all possible state-action pairs several times. In fact, for a large environment, the agent might rarely arrive at some of the states and might leave out many states completely (Sutton et al. 2018). If action a is potentially the best action to take at S_t, it is worth to estimate $Q(S_t, a)$ multiple times. However, since this presumption may not be always true (at least, during training), other actions at S_t need to be also tried, leading to a dilemma known as the exploration vs. exploitation problem in RL (Géron 2019; Sutton et al. 2018). The term exploration means to try a state-action pair that has not been explored enough (or not explored at all) (Géron 2019; Sutton et al. 2018). On the other hand, exploitation means to exploit the prior knowledge and take the best-known action at a particular state for better estimation of the Q-value of that state-action pair (Géron 2019; Sutton et al. 2018).

The rule for selecting an action at a given state is referred to as policy (Géron 2019; Sutton et al. 2018). Based on current knowledge, selecting the action that has the highest Q-value is called the greedy policy, and the corresponding action is called the greedy action (Sutton et al. 2018). The greedy policy allows the agent to exploit the current knowledge. However, the agent must explore the environment as well. Therefore, a policy called ε-greedy policy is commonly used where at each state, the agent will explore a randomly selected action with probability ε ($0 \leq \varepsilon \leq 1$), and exploit this greedy action with probability $1 - \varepsilon$ (Géron 2019; Sutton et al. 2018).

At the beginning of training, the agent is encouraged to explore the environment more (Géron 2019). Throughout the training, however, and as the agent becomes more mature by gaining better knowledge about the environment, it can progressively favor exploitation (Géron 2019). This is done by selecting a high value of ε (assume, ε_{max}) at the very beginning of the training process, followed by linearly decaying ε so that after a certain number of iterations (assume, $n_{\text{iteration}}$), it reaches to a very small value (assume, ε_{min}). Equation 15 shows the formula for calculating ε_i, i.e., the value of ε at the ith iteration.

$$\varepsilon_i = \varepsilon_{\text{max}} - \frac{\varepsilon_{\text{max}} - \varepsilon_{\text{min}}}{n_{\text{iteration}}} * i \tag{15}$$

Performance Metrics

Precision, recall, and accuracy

To test the performance of a classifier model, unseen testing data are fed to the trained model. The performance of the classifier model in single-label and multi-label classification tasks is evaluated using well-established measures of accuracy, precision, and recall, as shown in Equations 16 through 18.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{16}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{17}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{18}$$

Here, TP, TN, FP, and FN refer to true positive (correctly classified to the class), true negative (correctly classified to other class), false positive (incorrectly classified to the class), and false negative (incorrectly classified to other class), respectively.

Intersection over union (IoU)

For the object detection problems, IoU metric represents the percentage of overlap between ground-truth boxes and predicted boxes (Nath et al. 2020) for an object, is measured using Equation 19 where *G* and *P* denote the ground-truth and predicted boxes, respectively.

$$\text{IoU} = \frac{\text{intersection}}{\text{union}} = \frac{G \cap P}{G \cup P} \tag{19}$$

Mean average precision (mAP)

To calculate mAP, first, IoU is measured using Equation 19. Next, all detections are ranked in order of their corresponding confidence level. This is followed by moving through the ranked sequence, from the highest to the lowest confidence level, and calculating the precision and recall values, at each position for a particular class. In object detection, TP, FN, and FP are calculated by comparing the IoU against a threshold value (e.g., 50%). Next, for each class, AP is calculated using Equation 20, where n is the total number of detections, i is the rank of a particular detection in the list sorted in descending order of confidence, $p(i)$ is the precision of the sub-list ranged from 1st to ith detection, and $\Delta r(i)$ is the change in recall from $(i-1)$th to ith detection. Finally, mAP is estimated by calculating the mean of APs of all possible classes.

$$AP = \sum_{i=1}^{n} p(i)\Delta r(i) \tag{20}$$

CHAPTER IV

IMAGE-LEVEL CONTENT RETRIEVAL[*]

In this Chapter, DL-based approaches for retrieving image-level contents will be described. These approaches can be applied to analyze a large volume of construction imagery and automatedly generate metadata tags describing the contents of each image.

Dataset Description

Single-label dataset (Pictor v.1.0)

The Pictor-v1 dataset contains 2,037 images web-mined from Google image search database by using the keywords such as “building under construction”, “construction equipment”, “truck”, “dozer”, “excavator”, “crane”, and “construction worker”. Next, each image is manually annotated with a single label – building, equipment, or worker – based on the most prominent object in the image. The Pictor-v1 dataset with single-label annotation is named Pictor-v1.0 and the statistics and examples are shown in Figure 5.

Multi-label dataset (Pictor v.1.1)

All images in the Pictor-v1 dataset are also manually annotated with multiple labels – building, equipment, and worker – depending on the presence of the objects in

[*] Part of the data and analyses reported in this chapter is reprinted with permission from “Single-and multi-label classification of construction objects using deep transfer learning methods” by Nath, Nipun, Theodora Chaspari, and Amir Behzadan. 2019. *Journal of Information Technology in Construction*, 24: 511-26.

the image and named Pictor-v1.1. The number of images and sample images per class label in the Pictor- v1.1 dataset are shown in Figure 6.

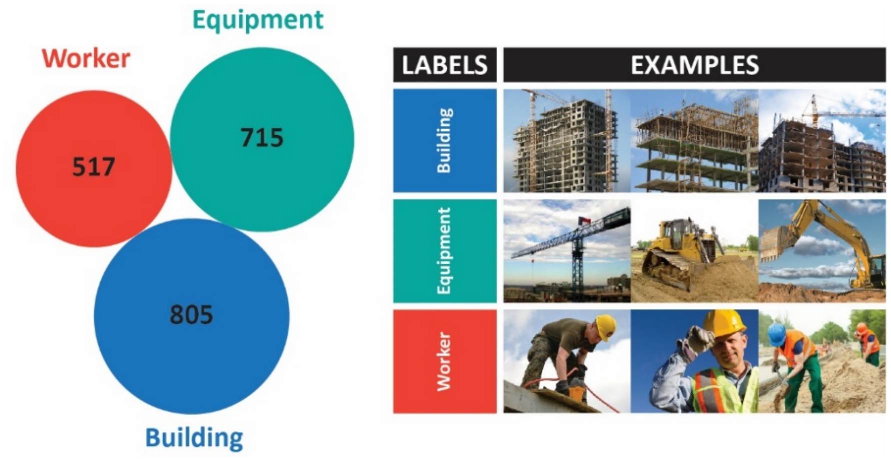


Figure 5. Number of images and sample images per class label in Pictor-v1.0 dataset. Reprinted with permission from Nath et al. (2019).

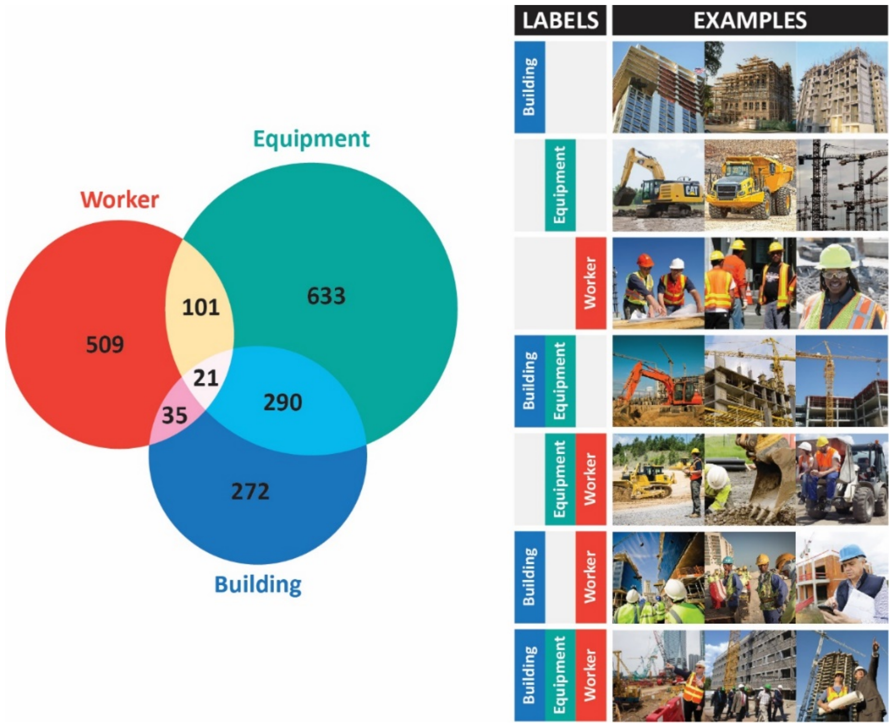


Figure 6. Number of images and sample images per class label in Pictor-v1.1 dataset. Reprinted with permission from Nath et al. (2019).

Proposed Methodology

Data pre-processing and splitting

Since the CNN models generally takes square-sized input images, for single-label classification, any rectangular image is cropped into a group of square images that cover the entire visual field of the original image while being equidistantly distributed along the longer dimension of the original image. An example is shown in Figure 7 where a portrait rectangular image is cropped into three square images. The number of cropped images is determined based on the smallest integer number greater than or equal to (i.e., ceiling of) the ratio between the longer and shorter dimensions of the original image. Next, all cropped images are resized to 128×128 images using the bi-cubic interpolation method (Zhang et al. 2011).

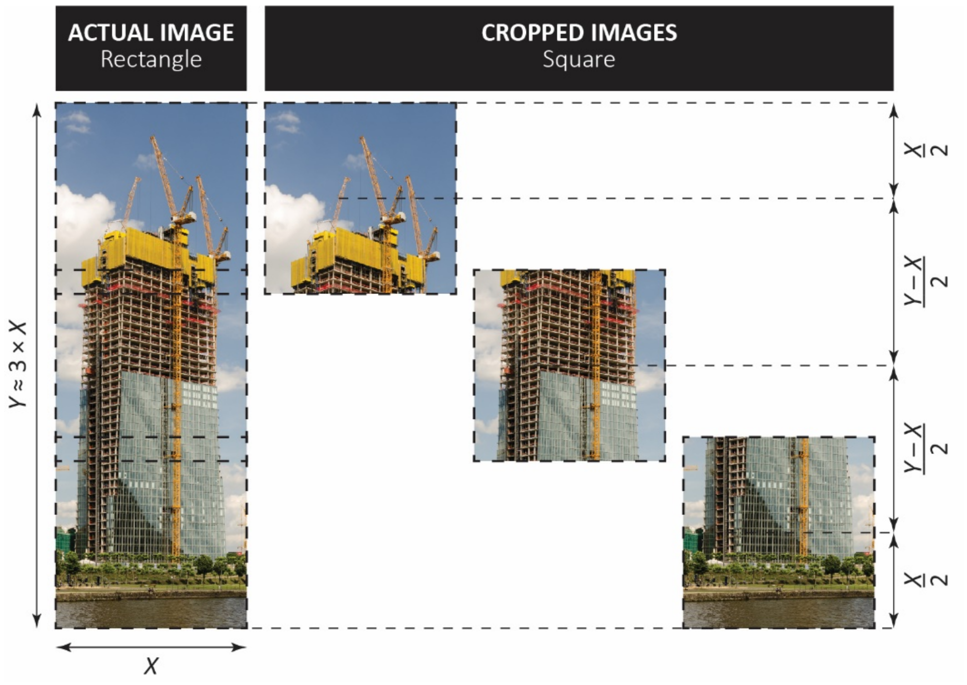


Figure 7. Example of cropping a rectangular image into a group of square-sized images. Reprinted with permission from Nath et al. (2019).

Data augmentation

Data augmentation is an effective technique to prevent classifier models from overfitting by providing randomly distorted training images to the model and thus, allowing the model to learn general features (Perez et al. 2017). In this Dissertation, during each epoch of training, training images are distorted by randomly scaling the image by ±20% and horizontally flipping the image randomly 50% of the time. Example of an actual image and randomly generated augmented images are shown in Figure 8. As shown in this Figure, data augmentation generates more training images with different orientations (e.g., bucket of the excavator facing left and right) and zoom-levels (e.g., the bucket appearing closer in some images, and farther in other images). It allows the model to learn to recognize the objects regardless of their orientation and distance with respect to the camera.

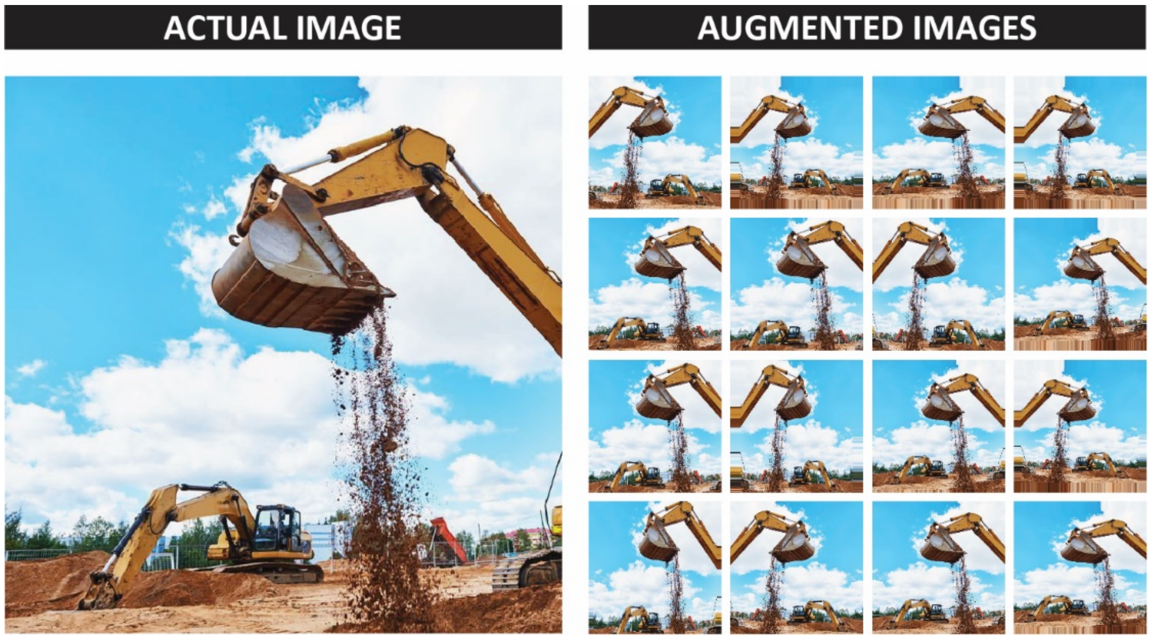


Figure 8. Example of data augmentation using random scaling and horizontal flipping. Reprinted with permission from Nath et al. (2019).

Architecture of the CNN

The designed CNN, for both single-label and multi-label classification, is based on the VGG-16 model (Simonyan et al. 2014). It consists of one input layer (i.e., 128×128 RGB images), 18 VGG-16 layers, 2 fully connected layers, and one output layer (e.g., labels or tags) as shown in Figure 9. The output layer yields a vector represented as "one-hot encoding" (Marinai et al. 2005) where each element of the vector represents one class and can have a value of either 1 (i.e., the input image belongs to that class) or 0 (i.e., the input image does not belong to that class).

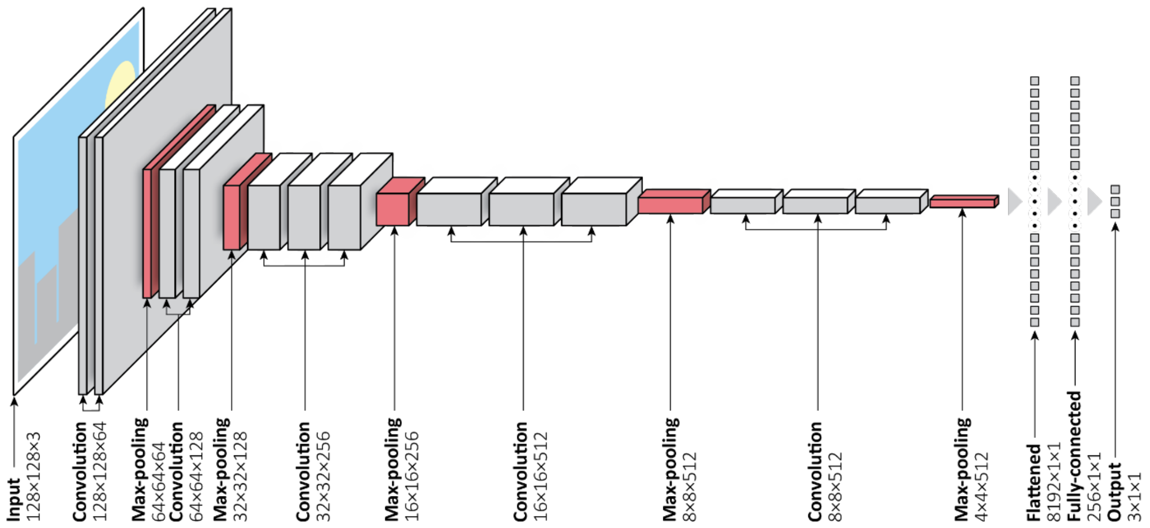


Figure 9. Architecture of the proposed CNN model. Reprinted with permission from Nath et al. (2019).

Activation and loss functions

For both models, at each hidden convolutional or FC layer, the ReLU non-linear activation function is applied to accelerate the convergence (Krizhevsky et al. 2012). However, for the single-label classifier model, the softmax (Equation 1) activation function (Murphy 2012) is used at the output layer, whereas for the multi-label classifier model, the Sigmoid (Equation 2) activation function (Friedman et al. 2001) is used. Also, for single-label classification, multi-class cross-entropy (Equation 3) (Friedman et al. 2001) is used as a loss function, while for multi-label classification, the loss function is defined as the sum of binary cross-entropy (Buja et al. 2005) over all classes (Equation 4).

Model training

To train the models, a transfer learning scheme is used which allows the models to remember high- and mid-level features (e.g., edge, shape, color) learned from a large dataset and apply these features (with minor adjustment) to effectively distinguish the classes in a smaller target dataset (Oquab et al. 2014). According to this scheme, the pre-trained weights from the ImageNet dataset (Simonyan et al. 2014) are used in the VGG-16 layers of the models. Next, only the weights in the FC layers are optimized using the RMSprop optimization algorithm (Tieleman et al. 2012). Then, the weight values of the last three convolutional layers and two fully-connected layers are updated using the stochastic gradient descent (SGD) algorithm (Bottou 2010) with a slow-learning rate (hyper-parameters, e.g., learning rate = 10^{-4}, and momentum = 0.9, are empirically selected).

Results and Discussion

The performance of the single-label classifier model on the Pictor-v1.0 dataset is summarized in Table 1. It shows that all classes are predicted with ~90% accuracy, precision, and recall. However, the precision of recognizing buildings (i.e., 89.1%) and the recall of recognizing a worker (i.e., 88.7%) is relatively lower. The underlying reason is that model tends to label the image with that class that has the larger visual footprint in the image. For example, as shown in Figure 10, an image with a building in the background and a worker in the foreground is labeled as a worker by the annotator as the worker appears more prominent to the human. However, the model would classify it as a building because of its larger footprint compared to the worker, lowering both the precision of building and recall of worker. This observation reveals one of the weaknesses of single-label classification and suggests performing multi-label classification. For the multi-label classification, the performance of the model on the Pictor-v1.1 dataset is summarized in Table 2, and examples are shown in Figure 11. The Table shows that all object classes are detected with >82% accuracy. However, it is noticeable that for all classes, the recall (>92%) is relatively higher than the precision (>73%). It indicates that the model sometimes predicts a class that is not present in the image (which lowers the precision). However, if a class is actually present, there is a high probability that the model will predict that class (which increases the recall).

Table 1 Performance metrics of the model for single-label classification. Adapted with permission from Nath et al. (2019).

Class	**Accuracy**	**Precision**	**Recall**
Building	95.2%	89.1%	95.2%
Equipment	89.5%	92.6%	89.5%
Worker	88.7%	94.0%	88.7%
Unweighted Average	91.1%	91.9%	91.1%
Weighted Average	91.2%	91.3%	91.2%

Table 2 Performance of the model for multi-label classification. Adapted with permission from Nath et al. (2019).

Class	Accuracy	Precision	Recall
Building	85.9%	73.7%	92.8%
Equipment	82.9%	76.0%	97.5%
Worker	89.2%	76.9%	94.5%
Unweighted Average	86.0%	75.5%	94.9%
Weighted Average	85.5%	75.6%	95.3%

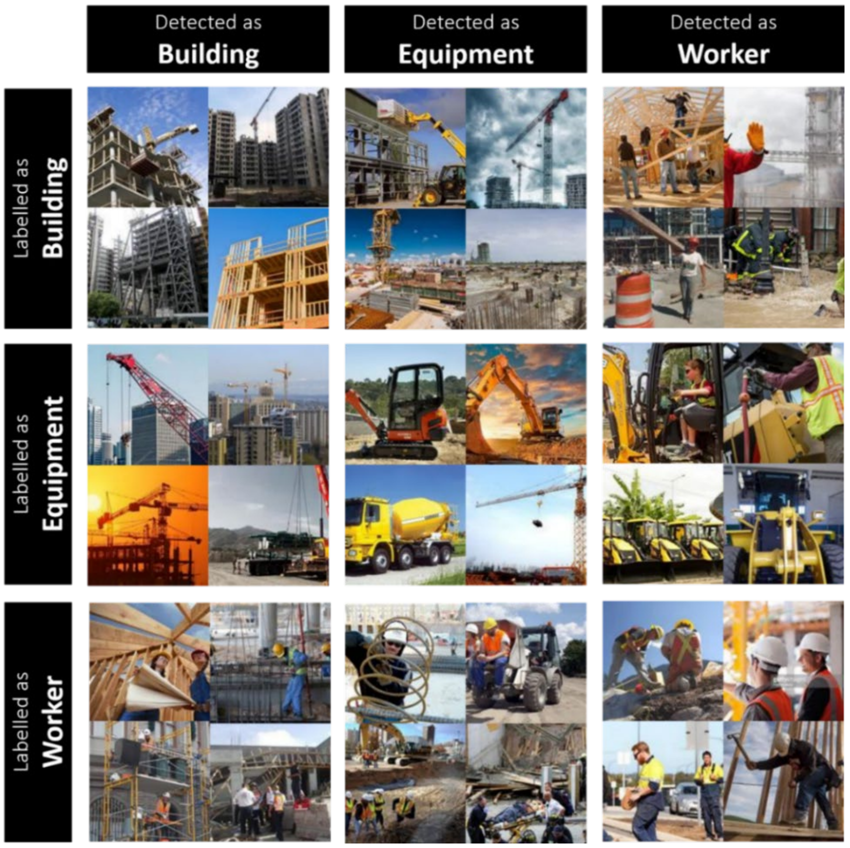


Figure 10. Visualization of the confused labels in single-label classification. Reprinted with permission from Nath et al. (2019).

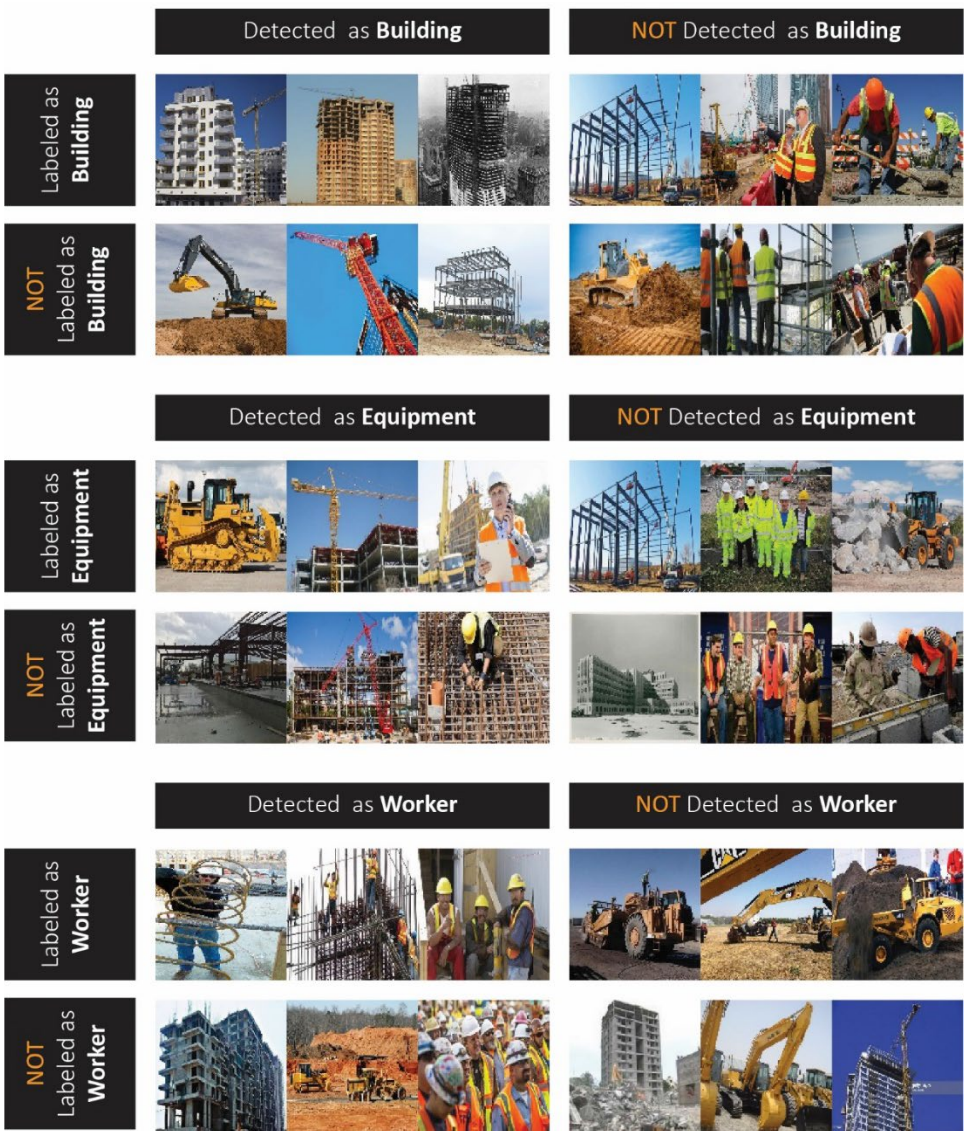


Figure 11. Visualization of the confused labels in multi-label classification. Reprinted with permission from Nath et al. (2019).

CHAPTER V

INSTANCE-LEVEL CONTENT RETRIEVAL*

This Chapter describes an object detection framework for retrieving instance-level contents in real-time. This framework can provide semantically richer information on the objects (e.g., the number, and the spatial relationships among them). Upon selecting the best-performing model, from different YOLO models, trained and tested on images collected through different methods, the Chapter will also discuss the strength and weakness of the model in detecting objects under different visual conditions.

Dataset Description

The Pictor-v2 dataset (Figure 12) contains instance-level annotations of buildings, equipment, and worker on the images collected through crowd-sourcing (Yuen et al. 2011) and web-mining (Kosala et al. 2000). As shown in Figure 13, the Pictor-v2 dataset contains 1,105 crowd-sourced images and 1,402 web-mined images. As shown in Figure 13(a), in total, there are 1,821 instances of building, 1,180 instances of equipment, and 2,611 instances of workers in the crowd-sourced images and 2,110 instances of building, 1,593 instances of equipment, and 2,257 instances of workers in the web-mined images of Pictor-v2 dataset. The number of images for each class in the

* Part of the data and analyses reported in this chapter is reprinted with permission from "Deep convolutional networks for construction object detection under different visual conditions" by Nath, Nipun, and Amir Behzadan. 2020. *Frontier's in Built Environment*, 6(97): 1-22.

randomly split training (~64%), validation (~16%), and testing (~20%) subsets of crowd-sourced and web-mined Pictor-v2 dataset are shown in Figure 13(b).

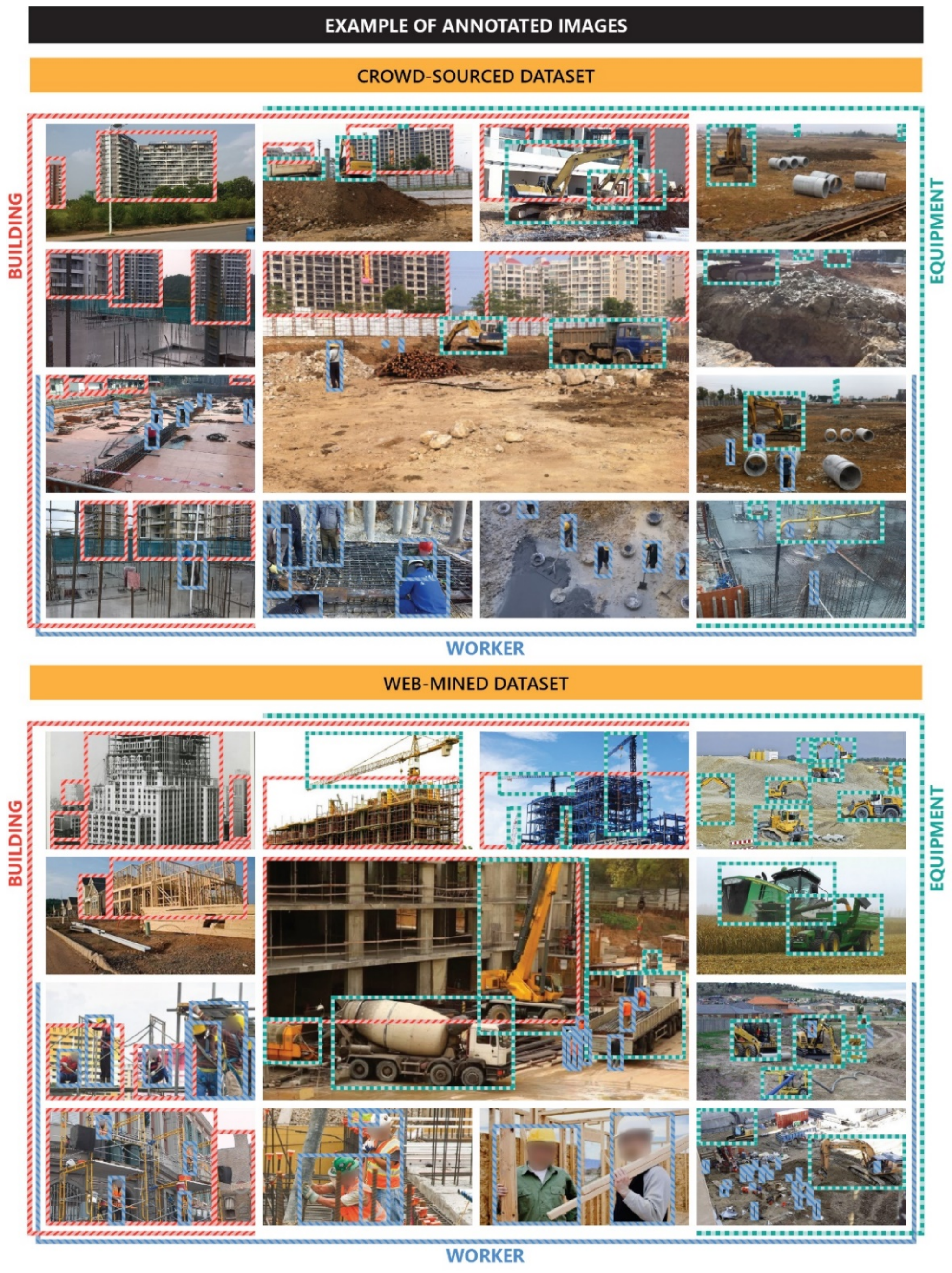
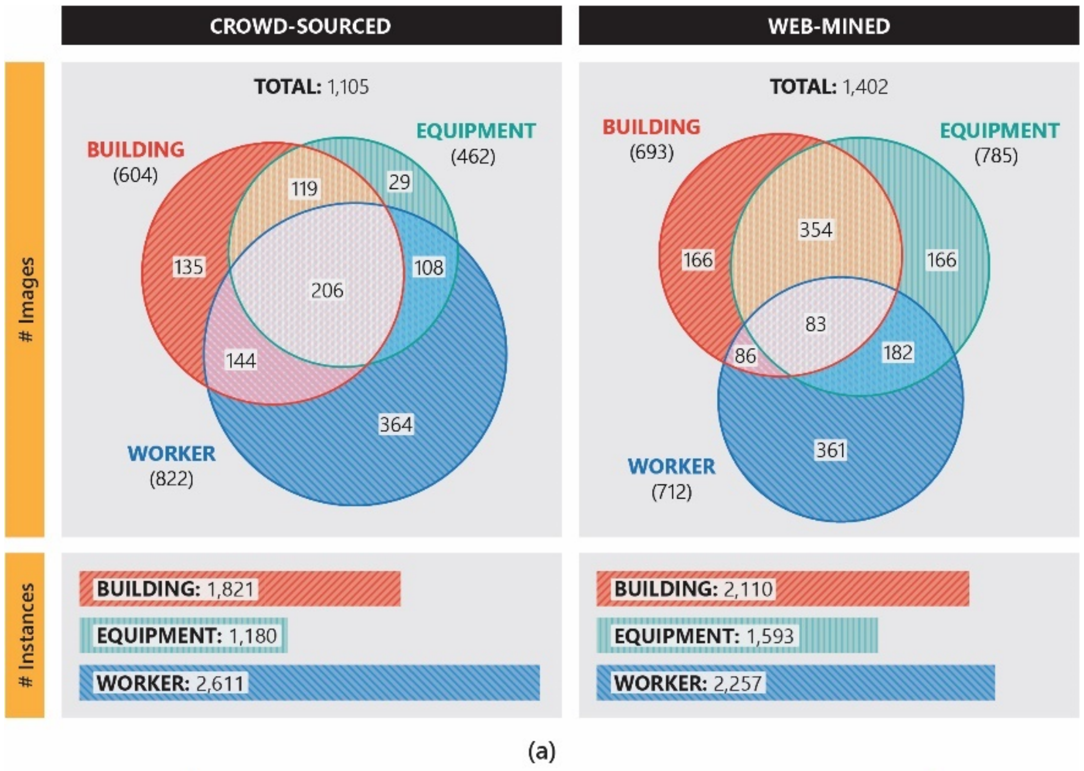


Figure 12. Examples of images and annotations in the Pictor-v2 dataset. Reprinted with permission from Nath et al. (2020).



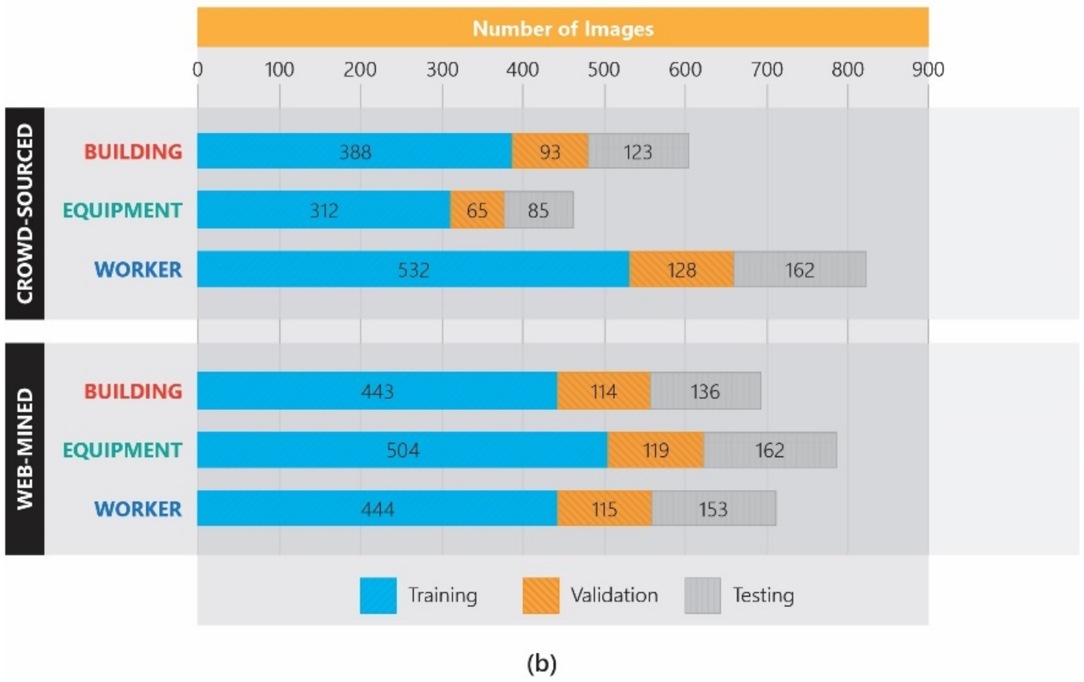


Figure 13. Number of images (a) per class labels retrieved through crowd-sourcing and web-mining in Pictor-v2 dataset, and (b) in the training, validation, and testing datasets. Reprinted with permission from Nath et al. (2020).

Proposed Methodology

Dataset pre-processing

Since YOLO takes 416×416 images as input, all images in the Pictor-v2 dataset are resized to 416×416 using bi-cubic interpolation (Zhang et al. 2011). During resizing, the original aspect ratio is preserved by padding the image equally on both sides along the shorter dimension. Next, each of the crowd-sourced and web-mined datasets is randomly split into training, validation, and testing subsets containing mutually exclusive 64%, 16%, and 20% of the entire dataset, respectively. The corresponding subsets (training, validation, and testing) of crowd-sourced and web-mined datasets are then merged to form the third combination, namely the "combined" subset. Next, k-means clustering (Redmon et al. 2017) is performed on all the rectangular boxes in the training dataset of each combination to obtain the desired number of anchor boxes.

Clustering

Since YOLO-v2 and YOLO-v3 models require five and nine anchor boxes, respectively, all the boxes in the training subset of each combination (crowd-sourced, web-mined, and combined) of the Pictor-v2 dataset are clustered into five (Figure 14(a)) and nine (Figure 14(b)) groups, using k-means clustering (Redmon et al. 2017), and a representative (centroid) from each group is selected as anchor box.

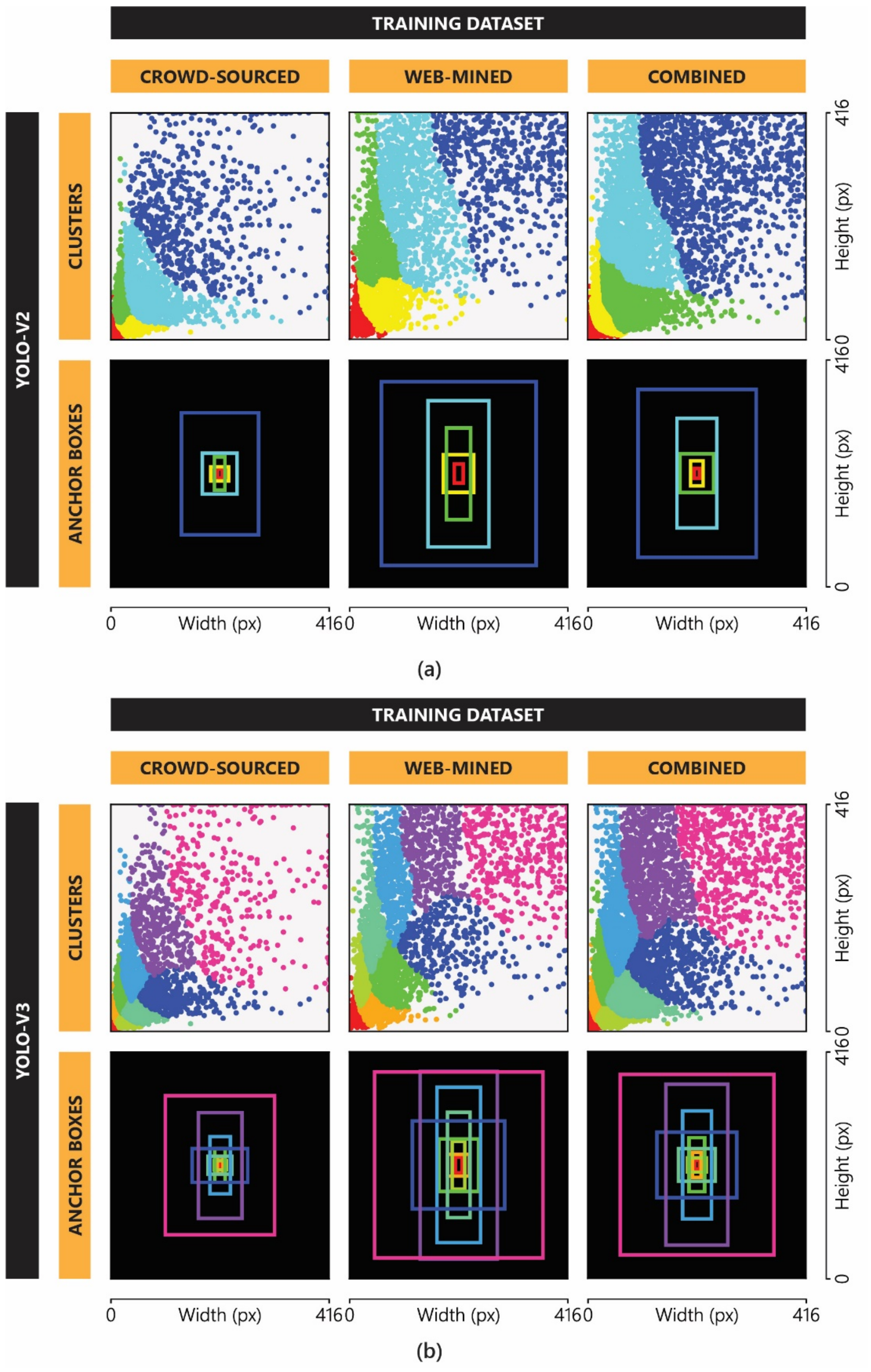


Figure 14. Clusters and corresponding anchor boxes for (a) YOLO-v2, and (b) YOLO-v3 implementation. Reprinted with permission from Nath et al. (2020).

It is evident from the anchor boxes in Figure 14 that the size of anchor boxes in the web-mined dataset is larger than those in the crowd-sourced dataset. This is rooted in the fact that web-mined images often contain a fewer number of objects, and the objects cover a larger visual field within the image. This setup imitates professional photographic arrangements where the appearance of the objects is of particular interest. On the contrary, the crowd-sourced images generally cover a larger field of view that captures more objects, each appearing smaller in the images. It also indicates that crowd-sourced images prioritize the amount of information over the appearance of objects in the image. The Figure also shows that the anchor boxes in the five-cluster groups are slenderer than the anchor boxes in the nine-cluster groups. This observation uncovers that when boxes are clustered into a fewer number of groups, slender objects (e.g., tall buildings, cranes, standing human) dominate, indicating the presence of a larger number of slender objects in the dataset.

Data augmentation

During training, real-time data augmentation is performed to prevent overfitting. Particularly, in every training step, each training image is randomly scaled up/down by ±30%, translated horizontally or vertically by ±30% (positive sign indicates translating to the right/downwards), and flipped in the horizontal direction in randomly selected 50% of the times. Also, hue, saturation, and value (brightness) of the training image are randomly changed (with uniform probability) in the range of [-10%, +10%], [-33%, 50%], and [-33%, 50%], respectively. Example of an actual image and generated images through random data augmentation are shown in Figure 15.

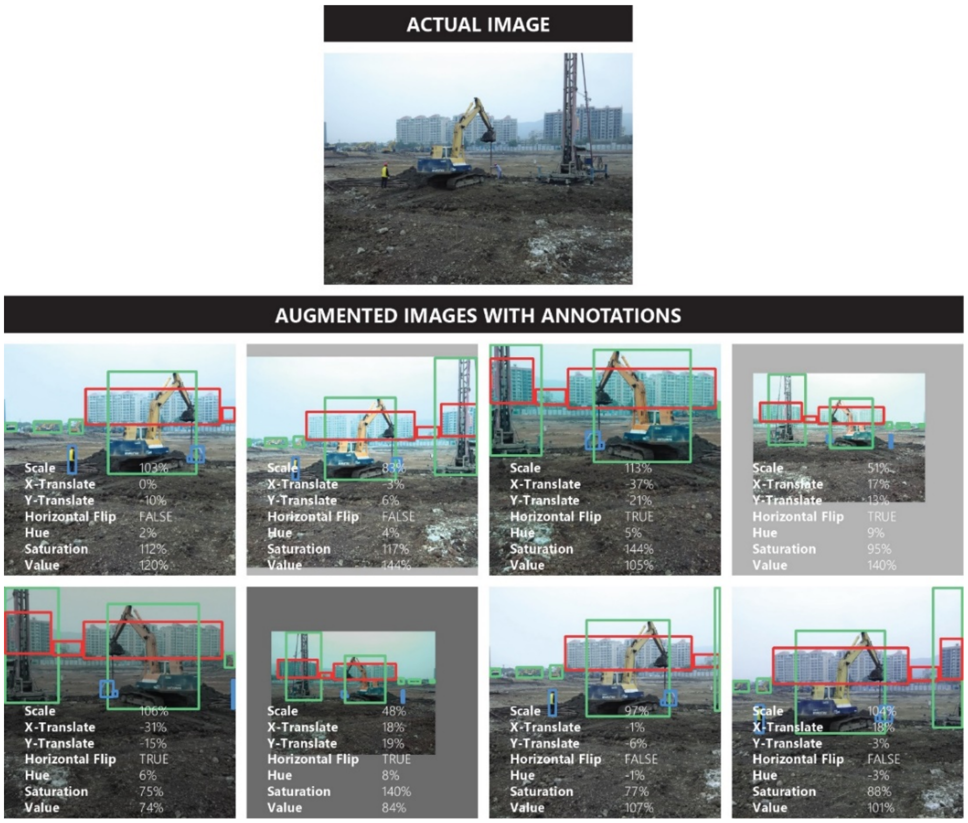


Figure 15. Example of actual and randomly augmented data. Reprinted with permission from Nath et al. (2020).

Model training

In this Dissertation, two different variations of the YOLO architecture, YOLO-v2 (Redmon et al. 2017) and YOLO-v3 (Redmon et al. 2018) are investigated with different combinations of subsets in the Pictor-v2 dataset (i.e., crowd-sourced, web-mined, and combined). All YOLO models are pre-trained on the COCO dataset (Chen et al. 2015), followed by re-training only the output layer(s) of the models on the training dataset of each combination for 25 epochs with a learning rate of 10^{-3} using Adam's (Kingma et al. 2014) optimizer. In the next step, all layers are fine-tuned with a slower learning rate using the same optimizer. To avoid overfitting during this fine-tuning phase, the loss on the validation data is continuously monitored and the learning rate is dynamically adjusted. Particularly, the fine-tuning phase is started with the initial learning rate of 10^{-4}. However, if the validation loss does not decrease for three consecutive epochs, the learning rate is reduced by a factor of 0.5. Moreover, if the validation loss does not decrease for ten consecutive epochs, the training is stopped. Additionally, during the re-training and fine-tuning phases, conventional data augmentation (e.g., translation, zoom in/out, horizontal flipping, and change of hue, saturation, and brightness of the image) (Perez et al. 2017) is performed.

Results and Discussion

Performance of the trained YOLO-v2 and YOLO-v3 models, in terms of AP for each class and mAP across all classes, are shown in Figure 16 and Figure 17, respectively. The Figures show that comparing crowd-sourced and web-mined subsets, the model performs better if it is trained and tested on a similar subset. In other words, the model performs poorly if it is trained on one subset and tested on another subset. It indicates significant visual differences between the images in crowd-sourced and web-mined subsets. However, both YOLO-v2 and YOLO-v3, models perform better when trained on the combined subset. It can be attributed to the higher number of training images in the combined data that allows the model to learn more generalizable features. However, it also indicates that the balance of diverse and challenging crowd-sourced images and well-structured web-mined images makes the model more robust. The Figures also show that, for any combination, the YOLO-v3 model performs better than the YOLO-v2 model. Particularly, the three output layers of the YOLO-v3 model allow the model to detect objects of various sizes. Therefore, YOLO-v3 models, trained on the combined subsets, are used for further analyses.

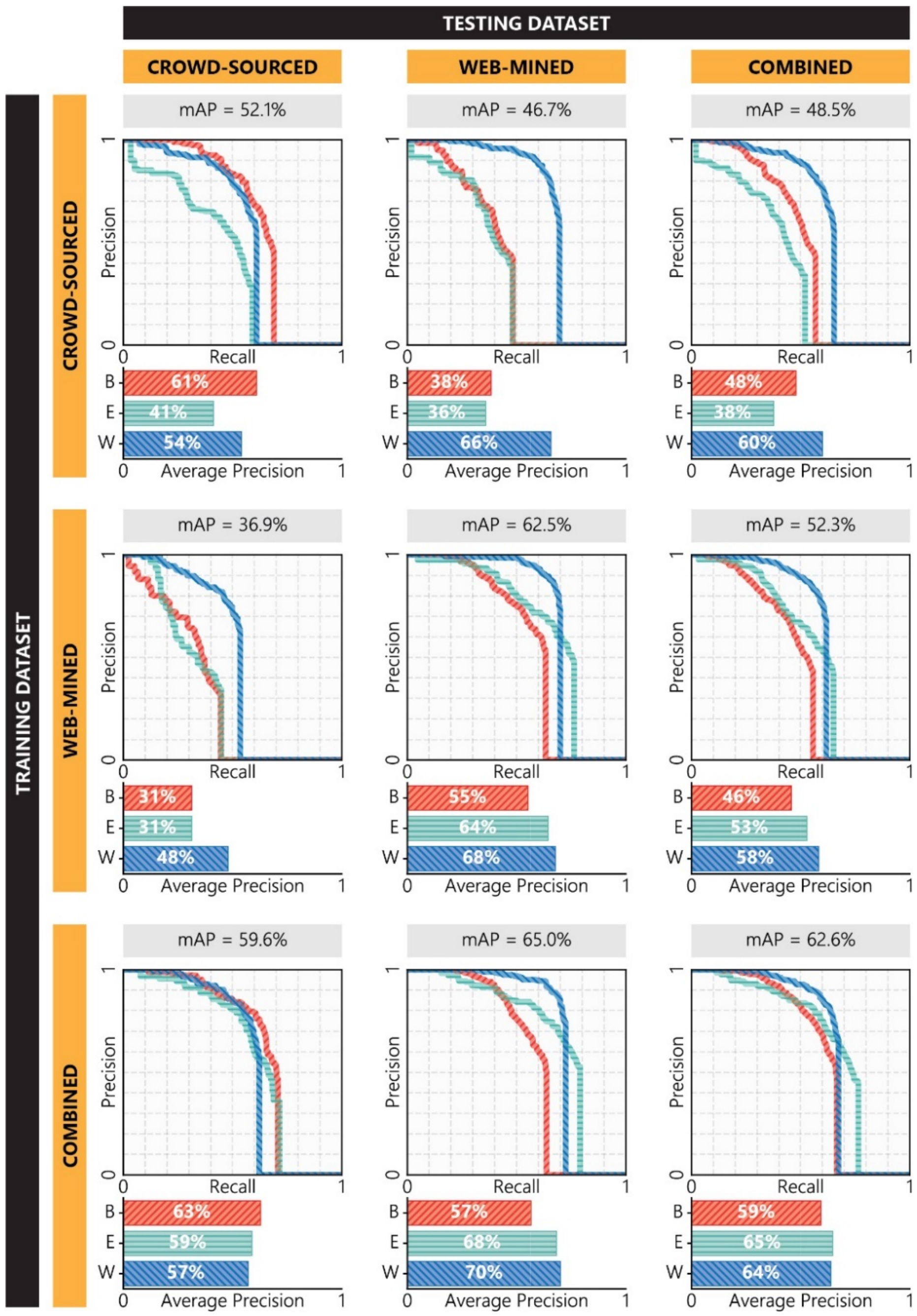


Figure 16. Performance of YOLO-v2 models trained and tested on different combinations of the Pictor-v2 dataset. Reprinted with permission from Nath et al. (2020).

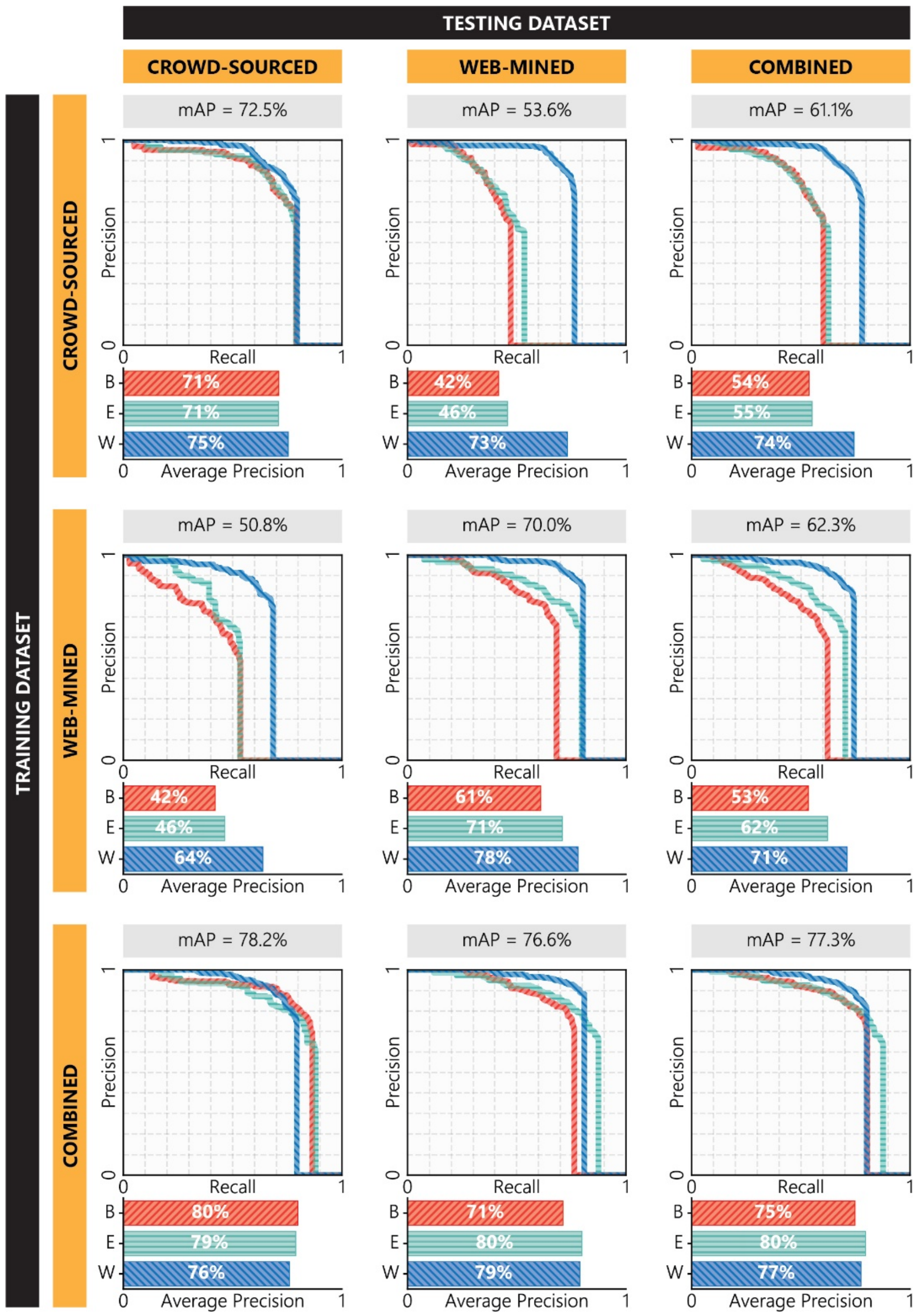


Figure 17. Performance of YOLO-v3 models trained and tested on different combinations of the Pictor-v2 dataset. Reprinted with permission from Nath et al. (2020).

Strengths and Weaknesses of Object Detection Model

To understand the strengths and weaknesses of the best model, YOLO-v3 trained and tested on the combined subset, the model is further tested under different visual conditions. For example, an object in the construction site imagery may appear in various sizes based on its real size, distance to the camera, and occlusion. Therefore, the object instances in the combined Pictor-v2 dataset are divided into two categories – "smaller" if the size is smaller than the median size, and "larger" otherwise. Moreover, construction sites generally consist of crowded spaces and occluded which can hinder the ability of the model to accurately find those objects. Therefore, if an object that has a bounding box overlapping with the bounding box of another object (i.e., the intersection over union, IoU $>$ 0%) is considered as a "more crowded" object. Furthermore, images captured in a poorly lit construction site may not contain content-rich information (due to less brightness and contrast). The amount of useful information in an image can be measured by Shannon entropy (Wu et al. 2013). Therefore, objects having Shannon entropy larger than the median value are considered as "well-lit" objects, while the others are considered as "poorly-lit" objects.

Examples of smaller vs. larger objects, more crowded vs. larger objects, and poorly lit vs. well-lit objects in Pictor-v2 dataset are shown in Figure 18. The performance of the YOLO-v3 models for various conditions is summarized in Table 3. As shown in the Table, the model performs substantially better in detecting large objects. Particularly, for detecting workers, the precision and recall of the model are 98%, indicating near human-level accuracy. Also, the Table shows that the model detects less crowded objects more accurately which is intuitive. Furthermore, in general, the model is better at detecting well-lit objects. However, if there is equipment in the image, the model is equally likely to detect the equipment regardless of the crowdedness or the lighting condition.

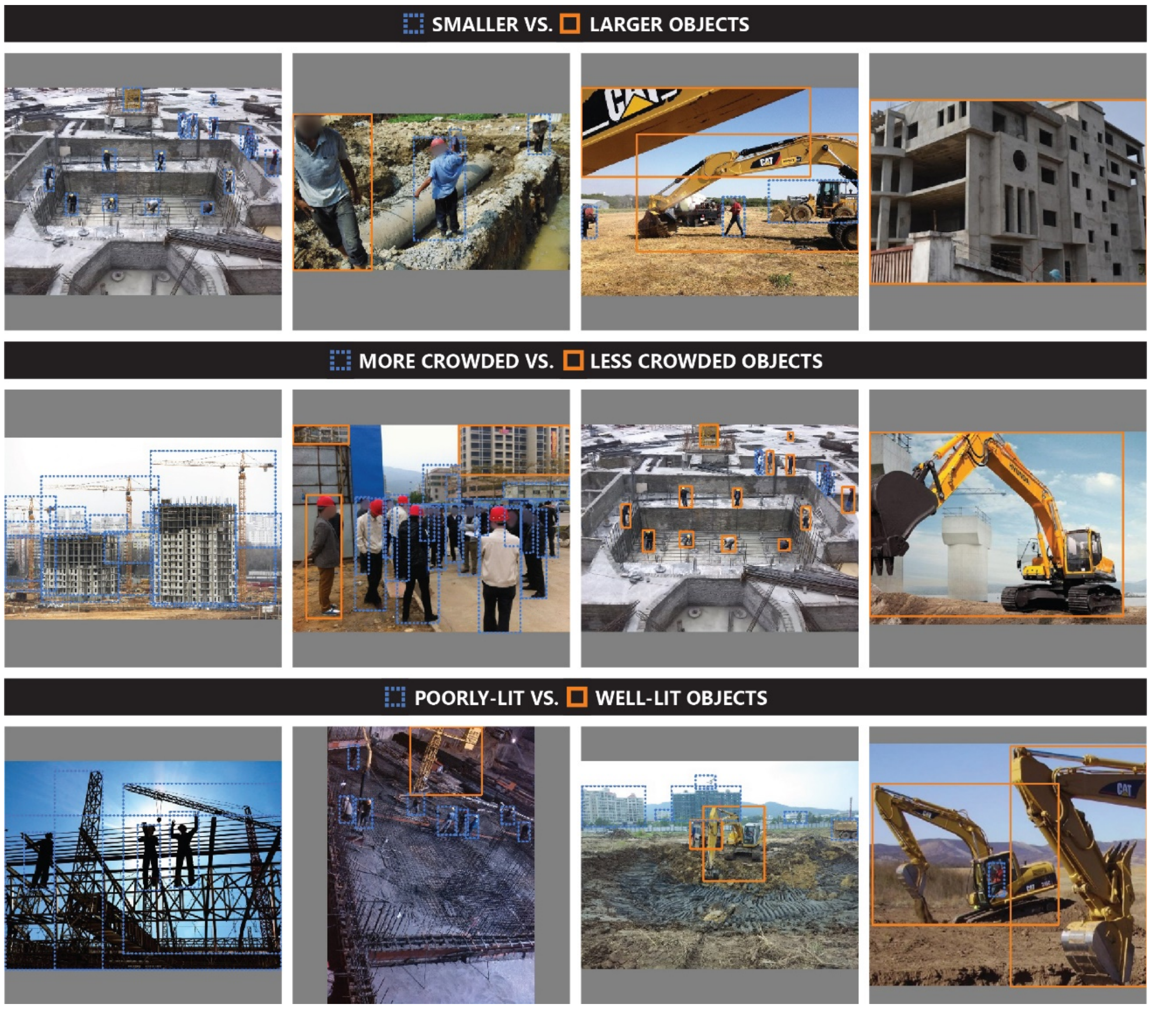

SMALLER VS. LARGER OBJECTS
MORE CROWDED VS. LESS CROWDED OBJECTS
POORLY-LIT VS. WELL-LIT OBJECTS

Figure 18. Examples of objects with different constraints (e.g., size, crowdedness, and lighting conditions) in Pictor-v2 dataset. Reprinted with permission from Nath et al. (2020).

Table 3. Performance of the best model in detecting building (B), Equipment (E), and worker (W) under different visual conditions. Reprinted with permission from Nath et al. (2020).

Criteria		Precision			Recall		
		B	E	W	B	E	W
Size	Large object	90%	80%	98%	94%	93%	98%
	Small object	77%	72%	83%	71%	80%	75%
	Difference	+13%	+8%	+15%	+23%	+13%	+23%
Crowdedness	Less crowded	84%	80%	87%	78%	83%	83%
	More crowded	80%	73%	85%	76%	83%	75%
	Difference	+4%	+7%	+2%	+2%	±0%	+8%
Lighting	Well-lit	83%	78%	93%	83%	83%	88%
	Poorly lit	79%	71%	77%	70%	83%	68%
	Difference	+4%	+7%	+16%	+13%	±0%	+20%

CHAPTER VI

VERIFICATION OF PPE COMPLIANCE[*]

Various safety and health organizations, such as OSHA, require that all workers properly use PPE at all times to avoid accidents. This Chapter describes DL-based approaches for an automated safety monitoring system that is designed to verify workers' compliance with the requirements related to PPE in real-time. The Chapter also demonstrates a process to recognize PPE components and their color, simultaneously, in real-time.

Dataset Description

The Pictor-v3 dataset contains the annotated object instances of the worker, hard hat, and safety vest, in 1,472 images. As shown in Figure 19, the dataset contains 774 crowd-sourced images and 698 web-mined images. Among the crowd-sourced, 240 images contain only worker (W), 517 images contain worker and hat (W+H), and 17 images contain worker, hat, and vest (W+H+V). Moreover, among the 2,496 workers in the crowd-sourced images, 873 workers do not wear any hat or vest (W), 1,583 workers wear hat (WH), 40 workers wear both hat and vest (WHV), and no worker wears only vest without hat (WV).

[*] Part of the data and analyses reported in this chapter is reprinted with permission from "Deep learning for site safety: Real-time detection of personal protective equipment" by Nath, Nipun, Amir Behzadan, and Stephanie G Paal. 2020. *Automation in Construction*, 112: 103085.

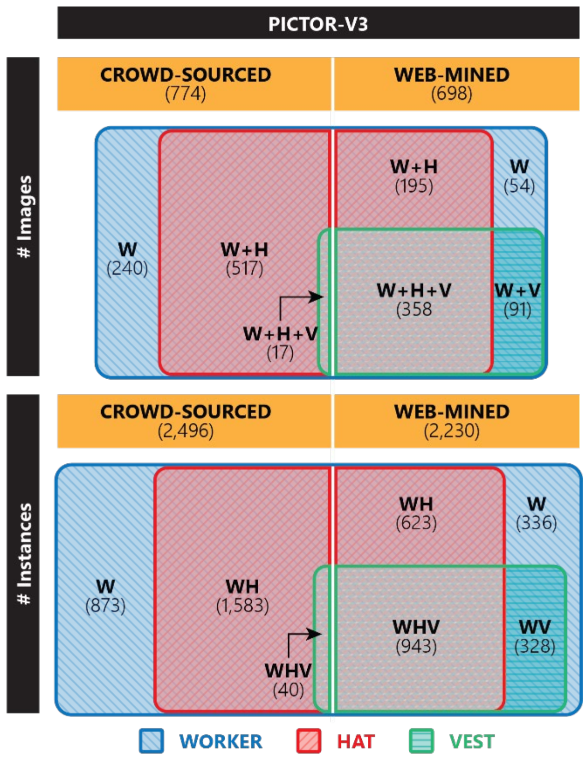


Figure 19. Number of images and number of instances per class label in the crowd-sourced and web-mined subsets of Pictor-v3 dataset. Reprinted with permission from Nath et al. (2020).

Worker and PPE Detections

Three approaches

In this Dissertation, three approaches are investigated to verify the PPE attire of workers. As shown in Figure 20, In Approach-1, a YOLO-v3-based model (named YOLO-v3-A1) is trained and tested to detect worker (W) and different PPE types, e.g., hat (H), and vest (V), individually. In contrast, in Approach-2, another YOLO-v3-based (named YOLO-v3-A2) model is designed to localize workers and directly classify them based on their PPE attire, i.e., W, WH, WV, and WHV. Finally, in Approach-3, a YOLO-v3-based model (named YOLO-v3-A3) is used to detect only workers (regardless of the PPE attire), then crops parts of the image that contain workers. Subsequently, different algorithms are applied to verify the PPE attire which will be discussed in the following Section.

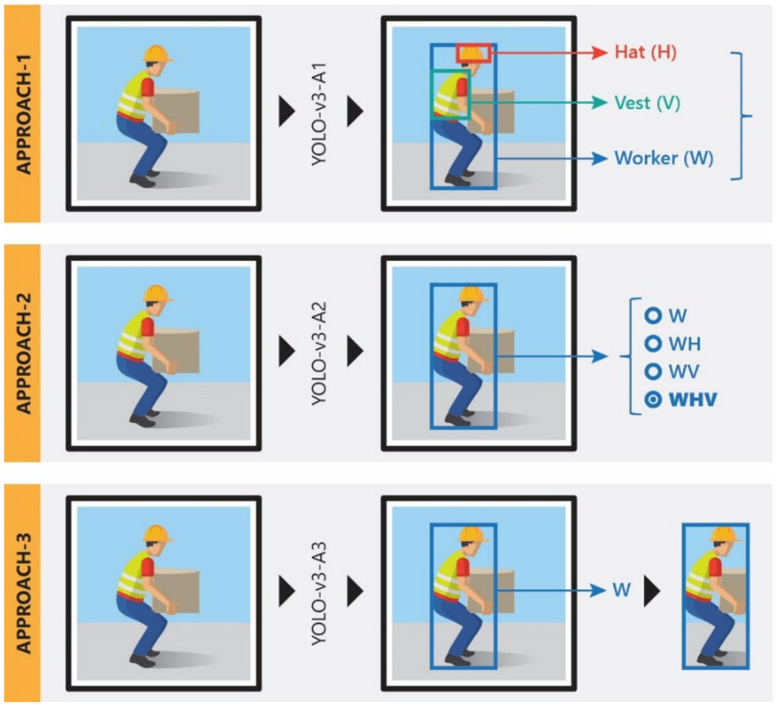


Figure 20. Schematic diagram of the three approaches. Reprinted with permission from Nath et al. (2020).

Data pre-processing

Data processing, splitting, and augmentation are performed following the similar methods described in Chapter V. Examples of augmented images are shown in Figure 21.

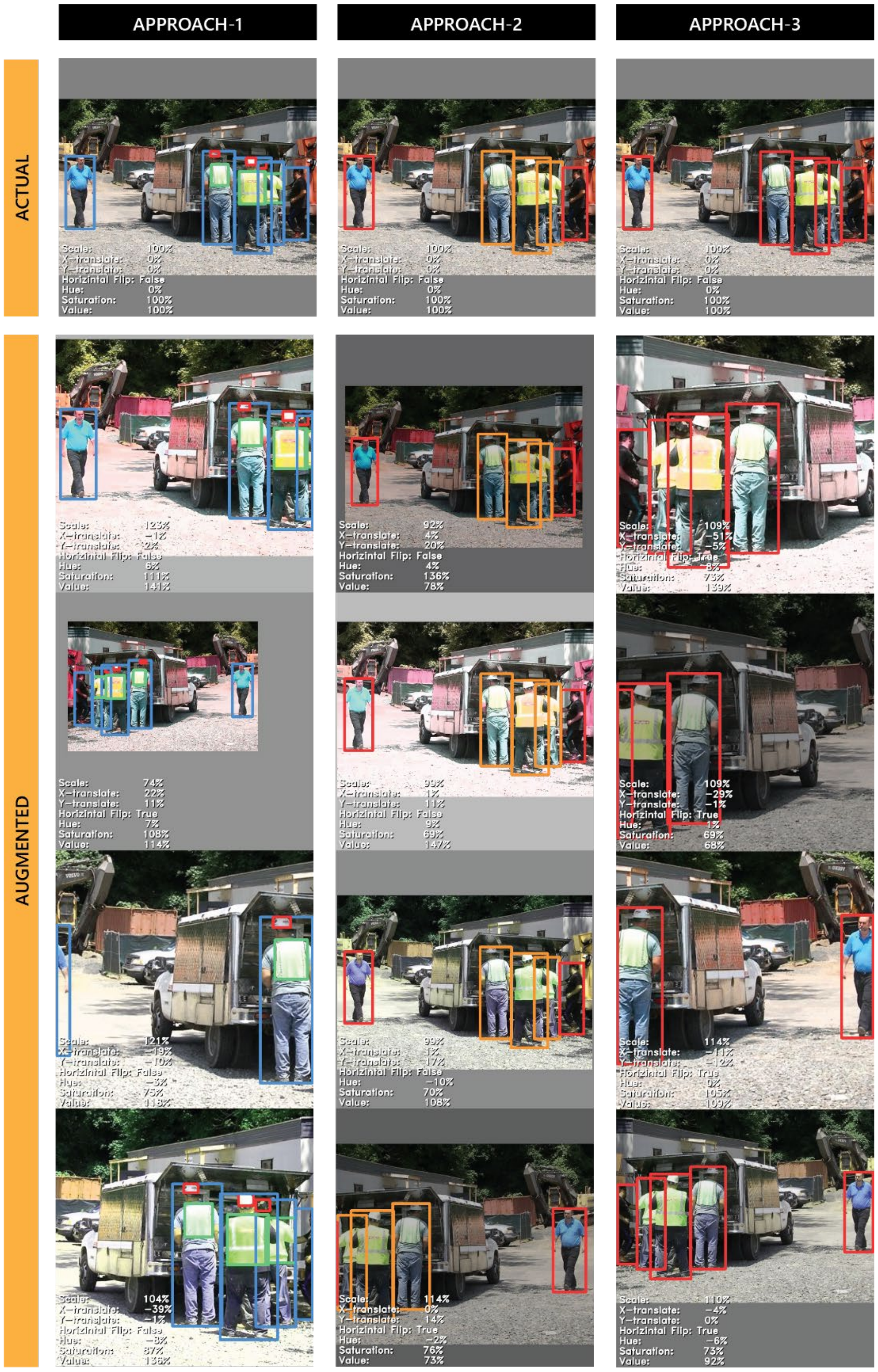


Figure 21. Examples of actual and randomly augmented data in three approaches. Reprinted with permission from Nath et al. (2020).

Clustering

Next, k-means clustering (Redmon et al. 2017) is performed on all the rectangular boxes in the training dataset of the Pictor-v3 to obtain nine anchor boxes for the models (Figure 22). All YOLO-v3 models are subsequently pre-trained on the COCO dataset (Chen et al. 2015).

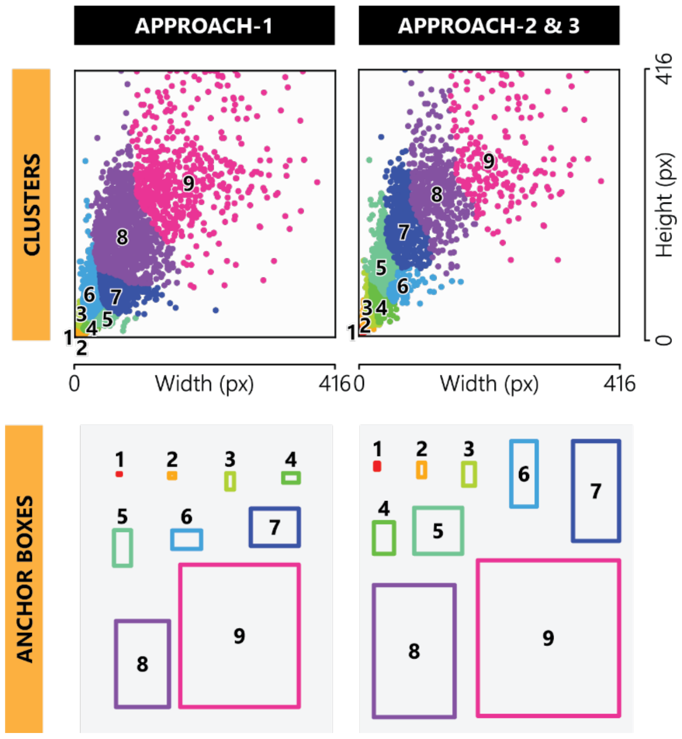


Figure 22. Nine clusters and corresponding anchor boxes for each approach. Reprinted with permission from Nath et al. (2020).

Model training

In all three approaches, the architecture of the pre-trained YOLO-v3 models (particularly, the last three output layers) are modified based on the number of considered classes in each approach. For example, in Approach-1, the YOLO-v3 model detects worker and n_P number of different PPE types, a total of $n_P + 1$ number of classes, individually. Therefore, each bounding box in each grid cell of the output layers of the YOLO-v3 architecture should be a $(n_P + 6)$-dimensional vector. For example, since there are 3 bounding boxes in each 13×13 grids of Output-1 layer, the dimension of the predicted tensor should be $13 \times 13 \times 3(n_P + 6)$. Similarly, for Output-2 and Output-3 layers, the dimensions of the predicted tensors should be $26 \times 26 \times 3(n_P + 6)$ and $52 \times 52 \times 3(n_P + 6)$, respectively. However, since the dimensions of output layers and the number and types of classes in the pre-trained YOLO-v3 models may not match with the problem at hand, all three output layers are replaced with new layers with required dimensions, and the weights in the newly added layers are randomly initialized. The modified model is referred to as YOLO-v3-A1. In Approach-2, the YOLO-v3 model directly detects worker's 2^{n_P} different combinations of PPE attire for n_P different PPE types. Therefore, the dimensions of the output layers are $13 \times 13 \times 3(2^{n_P} + 5)$, $26 \times 26 \times 3(2^{n_P} + 5)$, and $52 \times 52 \times 3(2^{n_P} + 5)$, respectively. With the same token as in Approach-1, the output layers of the YOLO-v3 model for Approach-2 are modified and referred to as YOLO-v3-A2 hereafter. In contrast, regardless of the number of considered PPE types, in Approach-3, the YOLO model detects only workers (i.e., one class) in the input image or video frame. Therefore, the dimensions of the output layers in the modified YOLO-v3 model for this approach (referred to as YOLO-v3-A3) are $13 \times 13 \times 3 * 6$, $26 \times 26 \times 3 * 6$, and $52 \times 52 \times 3 * 6$, respectively.

In YOLO-v3-A1, -A2, and -A3 models, all layers except the last three output layers contain pre-determined weights obtained by pre-training the models on COCO dataset. These pre-trained weights can extract useful features (e.g., colors, edges) that can effectively distinguish the classes (e.g., person, dog, car, apple, laptop, clock) present in the COCO dataset. However, in the newly added output layers, weights are initialized with random values and updated through re-training the models with the Pictor-v3 dataset for classifying the target classes. During this re-training process, the weights in all other layers are kept frozen (i.e., unchanged). The idea is to familiarize the models with the new target classes and allow them to learn how to use the pre-learned features to distinguish the target classes. The re-training process involves training for 25 epochs with a learning rate of 10^{-3} using Adam (Kingma et al. 2014) optimizer. Next, the entire model is fine-tuned by updating the weights in all layers, however, with a slower learning rate. This allows the models to slightly modify the pre-learned features to find more effective features that work better for detecting the target classes. To prevent the model from overfitting, fine-tuning is performed by continuously monitoring the validation loss after each epoch and adjusting the learning rate accordingly. Particularly, the following criteria are maintained in this step: fine-tuning of the model is started with an initial learning rate of 10^{-4}; the learning rate is reduced by half if the validation loss does not decrease for three consecutive epochs; training is terminated if the validation loss does not decrease for 10 consecutive epochs.

Performance of YOLO-v3 in three approaches

All the YOLO-v3 models are trained on the Pictor-v3 dataset following the transfer learning scheme. The mAPs of the YOLO-v3 models in three approaches are shown in Figure 23. As shown in the Figure, Approach-1 achieves 81.2% mAP. Among all three classes, worker class (W) has the highest AP, reaching 85%, which can be attributed to the large size of this object class compared to the other two classes – hat (H) and vest (V). The mAP is the lowest in Approach-2 (72.3%) which is not surprising since the target classes (i.e., W, WH, WV, and WHV) are visually very similar (i.e., all contain humans). Also, as expected, the mAP of Approach-3 is the highest (85.6%) since there is only one class to learn (i.e., W) and there is no chance of inter-class confusion.

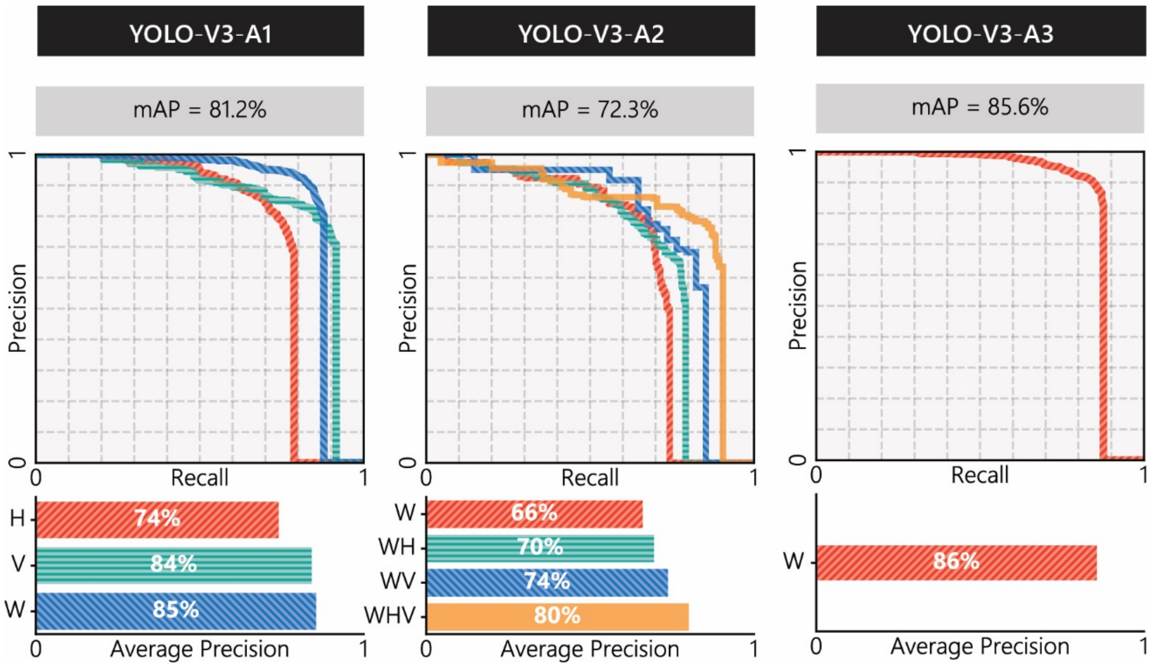


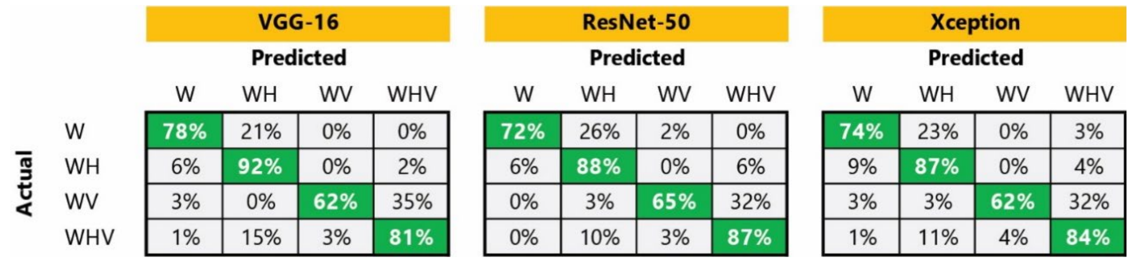
Figure 23. Performance of the YOLO-v3 models in three approaches. Reprinted with permission from Nath et al. (2020).

Classification of Worker Images for PPE Detections

CNN classifiers

After receiving the images of workers, from Approach-3, various classifier models can be applied to classify the image into four classes – worker wearing no hat or vest (W), worker wearing only hat (WH), worker wearing only vest (WV), and worker wearing both hat and vest (WHV). Particularly, three CNN classifier models based on three different architectures, VGG-16 (Simonyan et al. 2014), ResNet-50 (He et al. 2016), and Xception (Chollet 2017), are investigated for this Dissertation.

The accuracy of the VGG-16, ResNet-50, and Xception models in classifying worker images into W, WH, WV, and WHV classes are 78.2%, 77.8%, and 76.8%, respectively. The confusion matrices for these models are shown in Figure 24. The confusion matrices show that the CNN models tend to confuse the class W with the class WH, and the class WV with the class WHV. In both cases, the models false-positively detect hats in the images even if the worker is not wearing any hat. It can be attributed to the unbalanced dataset (where the samples of WH and WHV are higher than the samples of W and WV, respectively) and the small size of the hat.



		VGG-16				ResNet-50				Xception			
		Predicted				Predicted				Predicted			
		W	WH	WV	WHV	W	WH	WV	WHV	W	WH	WV	WHV
Actual	W	78%	21%	0%	0%	72%	26%	2%	0%	74%	23%	0%	3%
	WH	6%	92%	0%	2%	6%	88%	0%	6%	9%	87%	0%	4%
	WV	3%	0%	62%	35%	0%	3%	65%	32%	3%	3%	62%	32%
	WHV	1%	15%	3%	81%	0%	10%	3%	87%	1%	11%	4%	84%

Figure 24. Confusion matrices for VGG-16, ResNet-50, and Xception classifier models. Reprinted with permission from Nath et al. (2020).

Bayesian classifiers

In addition to the CNN classifiers, Bayesian inference models are investigated where the prior and posterior probabilities are calculated from the predictions of CNN classifiers (e.g., VGG-16, ResNet-50, and Xception) on trained data, and the posterior probability is used to classify the test images. Particularly, given a set of n classes $C = \{c_1, \dots c_n\}$ and CNN models $M_1, \dots, M_m$, a sample that is classified to class $c_i^{M_j} \in C$ by the CNN model M_j, the posterior probability $P(c_k^B|c_i^{M_1}, \dots, c_j^{M_m})$ represents the probability that the sample belongs to class $c_k^B \in C$. The posterior probabilities for all combination of $c_k^B, c_i^{M_1}, \dots, c_j^{M_m} \in C$ are calculated from the training dataset using the Bayesian formula in Equation 21.

$$P\left(c_k^B \middle| c_i^{M_1}, \dots, c_j^{M_m}\right) = \frac{P(c_k^B, c_i^{M_1}, \dots, c_j^{M_m})}{P(c_i^{M_1}, \dots, c_j^{M_m})} \tag{21}$$

During the inference phase, a test sample is classified by each CNN model as $c_i^{M_j}$ with probability $p^{M_j}(c_i)$. The final class (c^B) is determined by checking the posterior probabilities for all classes $c_k^B \in C$ and choosing the class (c^B) that yields the maximum probability, as shown in Equation 22. Also, the final probability is the maximum of the probabilities of the final class, i.e., $\max_j p^{M_j}(c^B)$.

$$c^B = \underset{c_k^B \in C}{\operatorname{argmax}} P\left(c_k^B \middle| c_i^{M_1}, \dots, c_j^{M_m}\right) \tag{22}$$

The performance of the Bayesian models, along with CNN models, is shown in Figure 25. The Figure shows that, in general, the Bayesian models perform slightly better than the CNN models. Particularly, the Bayesian model with all three classifiers performs the best (67.9% mAP).

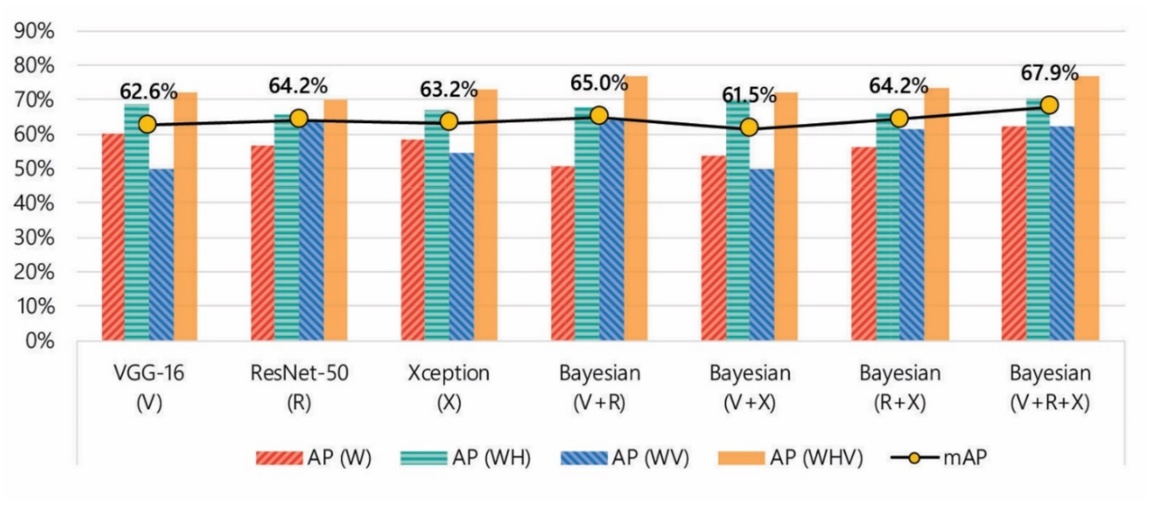


Figure 25. Comparison of the performance of CNN classifier and Bayesian models. Reprinted with permission from Nath et al. (2020).

Similarity Measure Technique for PPE Detections

Similarity measure technique

Given the images of a worker, the PPE attire can also be determined using the similarity measurement technique which is often used for PRID (Zhang et al. 2017) in the CV domain. In this technique, a query image of a person (with unknown information, e.g., identity or attire) is compared with the gallery images of other people (with known information). Then, from the best-matched gallery image, the required information of the person in the query image is retrieved. For PPE detection, the premise is that the people wearing similar PPE also look similar. Therefore, it is anticipated that given an image of a worker with unknown PPE attire, a trained PRID model will find the best-matching gallery image with similar PPE. Furthermore, the model has the potential to find the matching based on the color of the PPE as well. Therefore, for example, if the worker in the best-matched gallery image is wearing a red hard hat and a yellow safety vest (i.e., WHV), it is likely that the worker in the query image is also wearing a hard hat and a vest (i.e., WHV) of potentially same colors. From this motivation, a state-of-the-art PRID algorithm, AlignedReID (Luo et al. 2019), is adapted for detecting the PPE attire of workers in the Pictor-v3 dataset.

Dynamically matching local information (DMLI)

As shown in Figure 26, the AlignedReID algorithm divides an image into seven horizontal stripes, extracts local features from each stripe, and dynamically aligns two images from top to bottom using the shortest path in the distance matrix (Zhang et al. 2017). This process is termed DMLI (Luo et al. 2019). As Figure 26 illustrates, even if the persons in two images are in different poses, facing different directions, or occluded differently, the algorithm still finds similar attributes and calculates the similarity only based on the matching parts while ignoring the rest (e.g., irrelevant background).

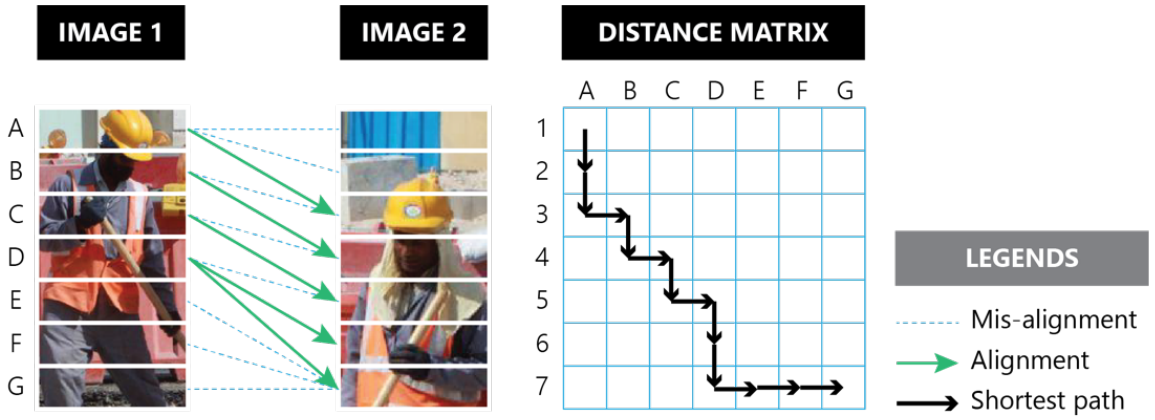


Figure 26. Example of aligning two images using DMLI.

Model training

A CNN model, based on ResNet-50 architecture, is used. The model extracts global and local features from two given images (one query image and another gallery image) and calculates the final distance between these two images as shown in Figure 27. The final distance represents the similarity between two images, where a smaller distance indicates a higher degree of similarity.

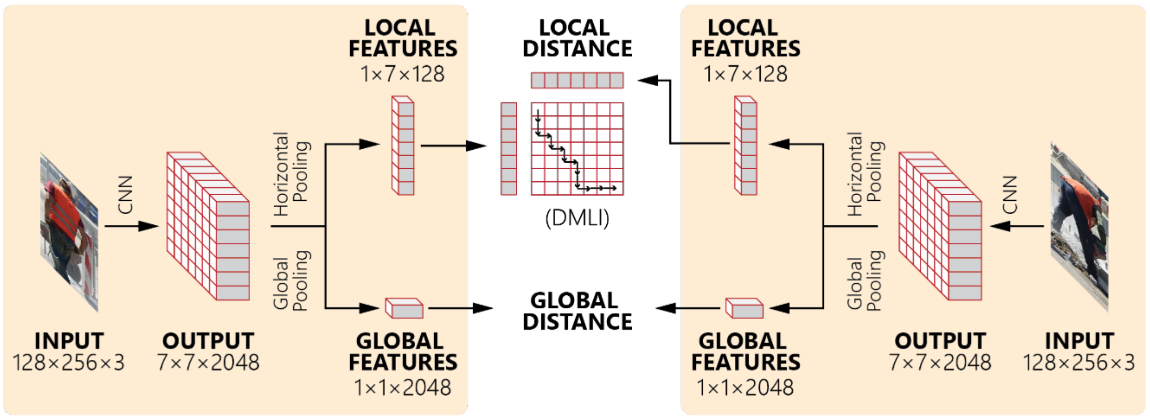


Figure 27. Calculation of local and global distances from two images.

The training protocol used in this Dissertation is inspired by the AlignedReID algorithm (Luo et al. 2019; Zhang et al. 2017). As shown in Figure 27, the CNN model, ResNet-50 (He et al. 2016), takes 128×256 RGB images as input and extracts a 7×7×2048 feature map. As shown in this Figure, horizontal pooling is then performed to extract 1×7×128 local features (128-dimensional feature for each of the 7 horizontal patches in the image) and global pooling is performed to extract 1×1×2048 global features of the image (Zhang et al. 2017).

The CNN model is trained on the training subset of the Pictor-v3 dataset with triplet loss (Zhang et al. 2017). In each training step, a batch of N images is fed to the model and (local and global) distances between each image pair are calculated. As illustrated in Figure 28, for each image in the batch, based on the global distances between the considered image (a.k.a., anchor) and other images, the farthest (most dissimilar) one from the same cluster (a.k.a. hardest positive) and the closest (most similar) one from a different cluster (a.k.a. hardest negative) are selected. The anchor image and its hardest positive and negative counterparts, altogether, is referred to as a *triplet*. For all valid triplets in the batch, the distances between each anchor and its positive and negative counterparts is used to calculate the triplet loss (Zhang et al. 2017). The sum of triplet losses for global and local distances along with softmax loss for global features is considered as the total loss (Luo et al. 2019). During training, the CNN model simultaneously learns to extract effective global and local features from the total loss (Luo et al. 2019). However, triplet loss, in particular, helps the model to learn subtle similarities between the inter-cluster images (i.e., anchors and their positive counterparts) and tenuous dissimilarities between intra-cluster images (i.e., anchors and their negative counterparts).

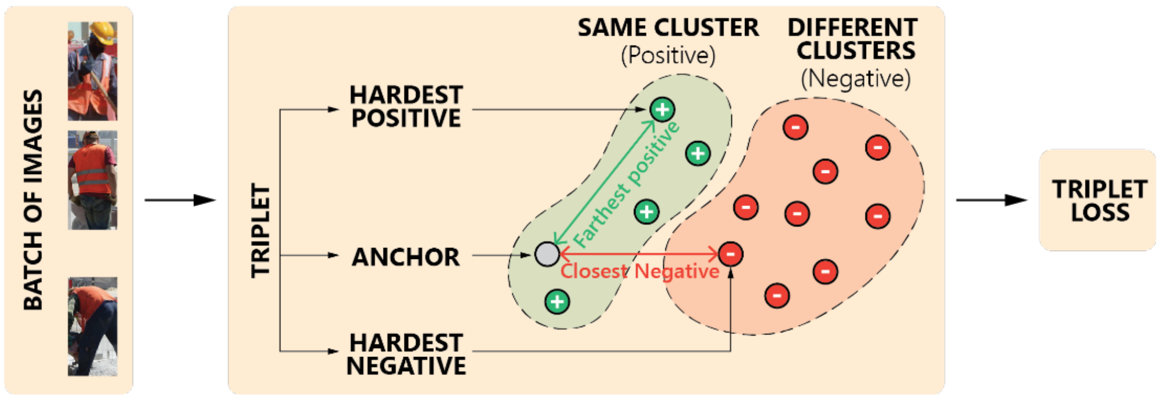


Figure 28. Schematic diagram of calculating triplet loss.

Result and discussion

The performance of this framework on the testing subset is shown in Figure 29. The Figure shows that the workers without any hat or vest (W), with hat (WH), with vest (WV), and with both hat and vest (WHV) are detected with 90% accuracy, which is higher than the accuracies of previously described CNN and Bayesian models. Moreover, the color of the PPE component is also recognized with 77% accuracy.

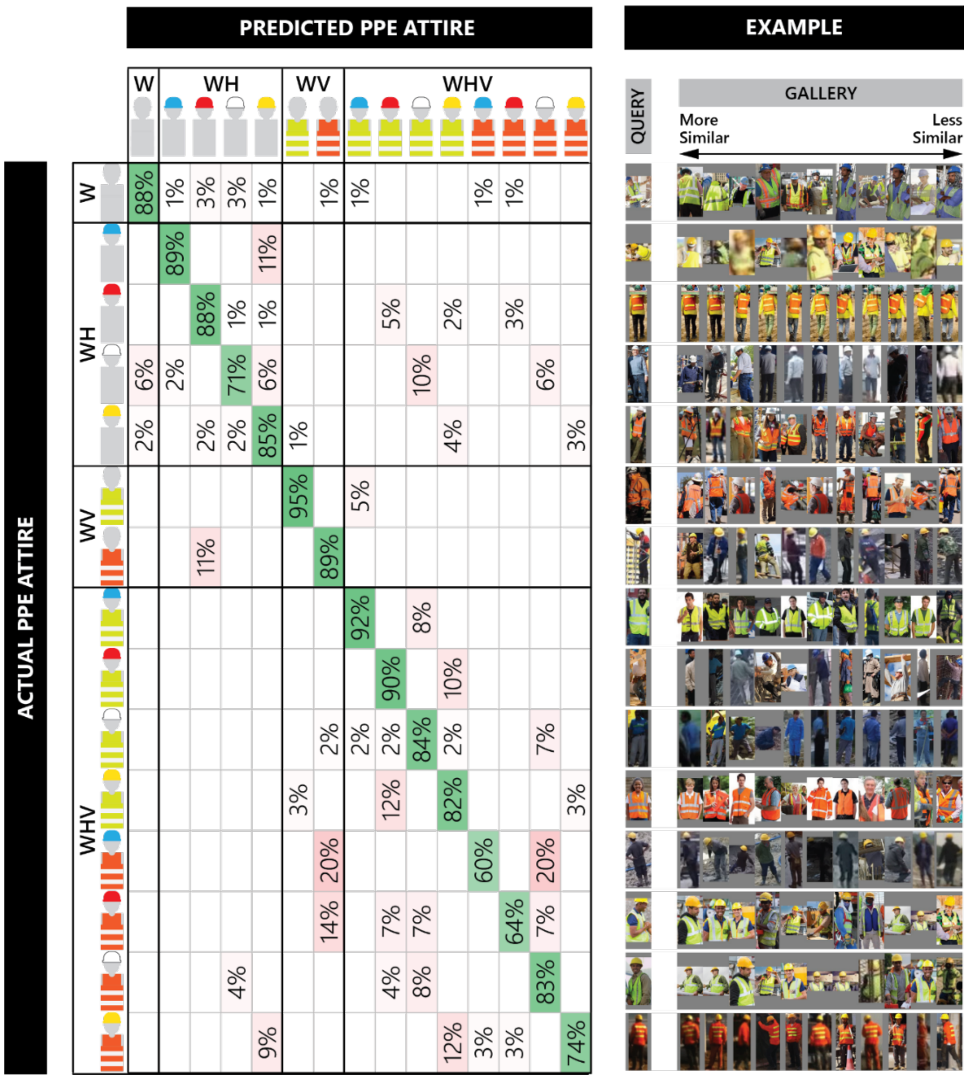


Figure 29. Confusion matrix and examples of detecting different PPE attire using PRID technique.

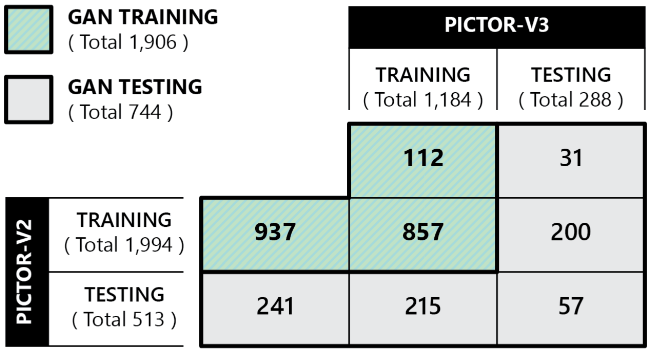
CHAPTER VII

IMAGE ENHANCEMENT FOR IMPROVED DETECTION PERFORMANCE

As CV-based algorithms perform better on high-quality images, this Chapter describes an AI-enabled image enhancement technique, using GAN, to improve the quality of visual data in order to increase the performance of the previously developed DL models.

Dataset Description

In this Chapter, a combination of two previously developed datasets, Pictor-v2 (Nath et al. 2020) and Pictor-v3 (Nath et al. 2020), as described in Chapters V and VI, is used. As shown in Figure 30, Pictor-v2 contains 1,994 training and 513 testing images, labeled with three object classes – building (B), equipment (E), and worker (W). On the other hand, Pictor-v3 contains 1,184 training and 288 testing images, also labeled with three object classes – hat (H), vest (V), and worker (W). The GAN model will be evaluated based on the performance of two YOLO models, namely YOLO-BEW and YOLO-WHV, which are trained on the training subsets of Pictor-v2 and Pictor-v3 datasets, respectively. To ensure that these YOLO models do not encounter any images on which they are already trained, the union of the testing subsets of Pictor-v2 and Pictor-v3 datasets is used for evaluating the GAN and YOLO models and, therefore, excluded from training the GAN models. This results in a total of 1,906 training and 744 testing images for the GAN models, as shown in Figure 30.



🟩 GAN TRAINING (Total 1,906)

⬜ GAN TESTING (Total 744)

		PICTOR-V3	
		TRAINING (Total 1,184)	TESTING (Total 288)
		112	31
PICTOR-V2	TRAINING (Total 1,994) 937	857	200
	TESTING (Total 513) 241	215	57

Figure 30. Number of images in the training and testing subsets.

Proposed Methodology

In this Dissertation, during the training, the weights of the discriminator D and generator G of the GAN model (Figure 31) are updated through backpropagation using Adam optimizer (Aggarwal 2018) with a starting learning rate of 1×10^{-4}. However, after 150th, 250th, and 300th epochs, the learning rate is reduced by a factor of 5. Finally, training is terminated after 330 epochs. Next, to investigate the influence of different image resolutions, I^L images are created by letterboxing the testing images, from Pictor-v2 and -v3 datasets, to sizes 52×52, 72×72, 96×96, 144×144, and 208×208 resolutions. Then, the GAN model (particularly, the generator G) is applied to the images which increases the resolution of each input image by a factor of 2×2. Next, two previously developed YOLO-v3 models are applied to the GAN-generated images to measure their object detection performance on the improved images. Particularly, YOLO-v3-BEW was trained on the Pictor-v2 dataset to detect common construction objects, namely buildings (B), equipment (E), and workers (W), as described in Chapter V. Another model YOLO-v3- WHV was trained on Pictor-v3 dataset to detect workers (W) and personal protective equipment (PPE), e.g., hat (H) and vest (V), as described in Chapter VI.

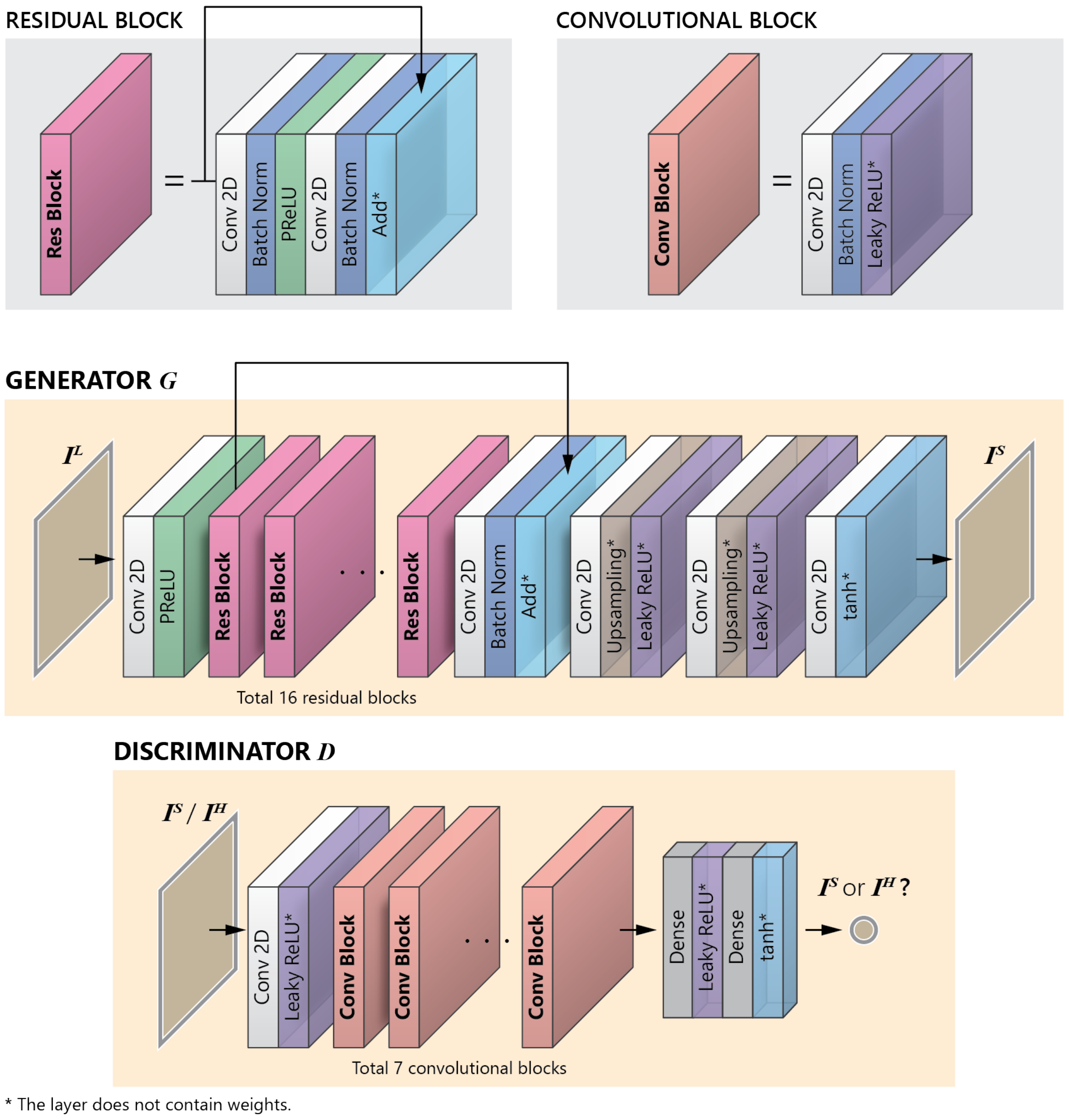


Figure 31. Architecture of the GAN models.

Loss functions

To train the generator and discriminator networks, three loss functions, namely the binary cross-entropy, content loss, and perceptual loss are used (Ledig et al. 2017). In particular, given an image I^H ($y^{True} = 1$) or I^S ($y^{True} = 0$), if the discriminator predicts the image as I^H with the probability of y^{Pred}, the binary cross-entropy loss (Aggarwal 2018) is defined by Equation 23.

$$L_{\text{binary}} = -\left[y^{True} \log\left(y^{Pred}\right) + \left(1 - y^{True}\right) \log\left(1 - y^{Pred}\right)\right] \quad (23)$$

To calculate content loss, a VGG-19 model, without the FC layers, and pre-trained on the ImageNet dataset, is used (Simonyan et al. 2014). Given the I^S and I^H images, the network extracts the two-dimensional feature maps F^S and F^H, respectively, each of size $w_F \times h_F$. The content loss is then defined as the Euclidean distance between these two feature maps (Ledig et al. 2017), as expressed in Equation 24.

$$L_{\text{content}} = \frac{1}{w_F h_F} \sum_{i=1}^{w_F} \sum_{j=1}^{h_F} \left(F_{i,j}^H - F_{i,j}^S\right)^2 \quad (24)$$

Finally, the perceptual loss is defined as the weighted average of content loss and binary cross-entropy loss (Ledig et al. 2017), as shown in Equation 25.

$$L_{\text{perceptual}} = L_{\text{content}} + 10^{-3} L_{\text{binary}} \quad (25)$$

Data pre-processing

To prepare the data for training and testing the GAN model, colors are linearly scaled (a.k.a., normalized) across all images so that all values remain in the range of

$[-1, +1]$. Next, images are divided into two subsets of training and testing. Only the training images are randomly augmented, as shown in Figure 32.

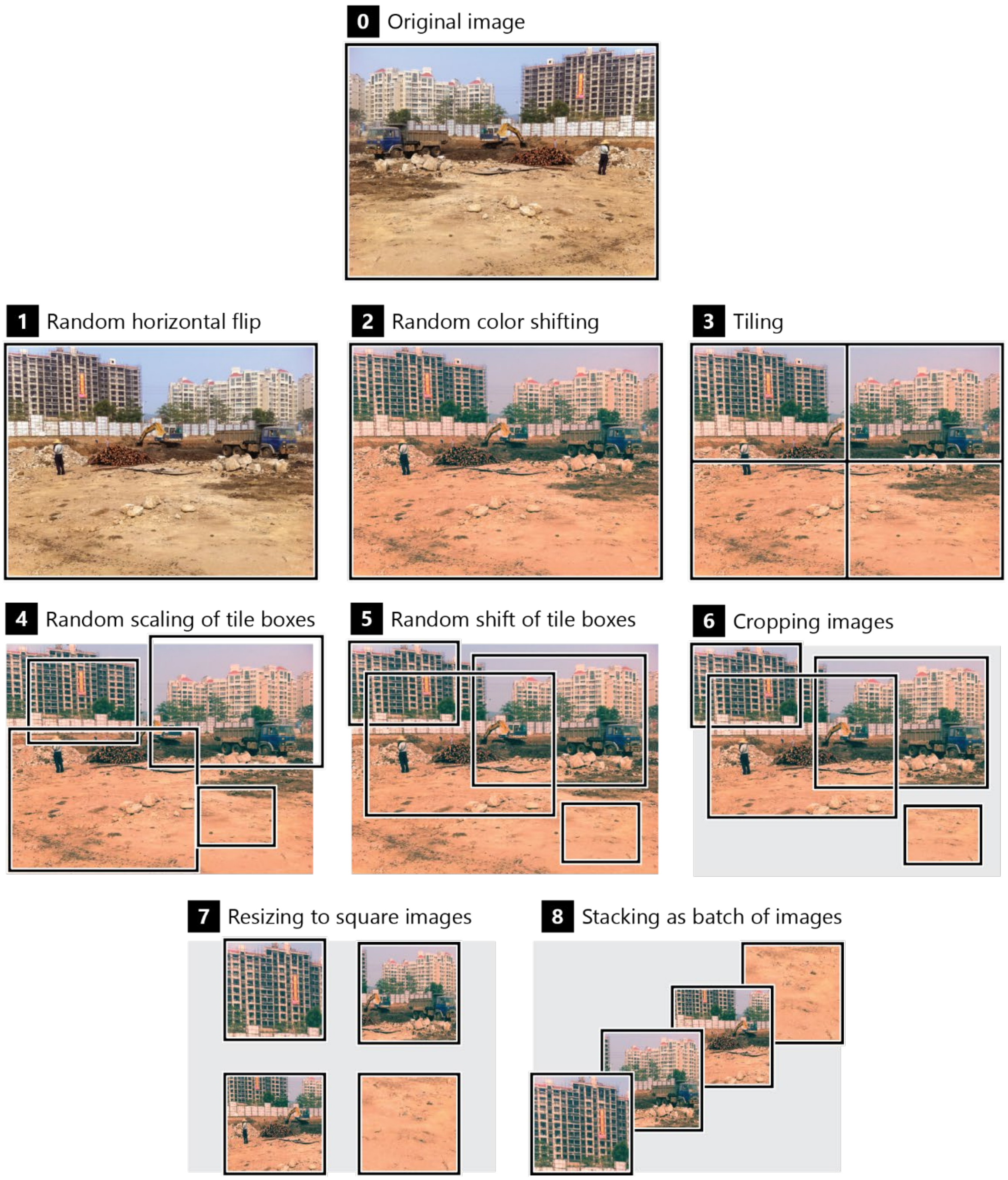

0 Original image
1 Random horizontal flip
2 Random color shifting
3 Tiling
4 Random scaling of tile boxes
5 Random shift of tile boxes
6 Cropping images
7 Resizing to square images
8 Stacking as batch of images

Figure 32. Preparation of training data through random augmentation.

First, randomly selected 50% of the original training images are flipped horizontally. Next, color-shifting is performed by multiplying the color of each pixel by a random value drawn from a normal distribution $N(1.0, 0.1)$, and adding another random value drawn from $N(0.0, 0.1)$ distribution. Following this step, $m \times n$ rectangular tiles are generated for each image. For the images with an aspect ratio (i.e., the ratio between image width and its height) between 0.5 to 2.0, $m = 2$ and $n = 2$ are used. For other aspect ratios (i.e., <0.5 or >2.0), 2 rectangular tiles, stacked along the longer dimension of the image, are generated. Next, each tile is randomly scaled by multiplying its box size with a factor drawn from $N(1.0, 0.25)$ distribution. Also, tile centers are further shifted along X and Y directions, each by a length from $N(1.0, 0.167)$ distribution, multiplied by their size along that direction. To note, the use of the normal distributions is based on Krizhevsky et al. (2017), however, the mean and standard deviation of these distributions are selected empirically by examining the dataset.

During the scaling and translation of the tile boxes, if any box is moved outside the image boundaries, it is trimmed so that the residual part remains inside the image. Next, the portion of the image within each tile box is cropped. Each cropped image is then resized to a 192×192 square-sized image which is treated as the high-resolution (ground-truth) image, or I^H. The I^H is subsequently resized to 48×48 resolution which serves as the low-resolution version, or I^L, corresponding to the I^H. Finally, I^L and I^H images are stacked in batches to allow the GPU to perform operations (e.g., convolutions) on all the images in one batch simultaneously, rather than treating each image individually, thus leading to a significantly lower computational time.

Model training

As shown in Figure 33, at each iteration of training, a single batch of training images (I^L) is first fed to the generator (G) and outputs (I^S) are recorded. Next, generated images (I^S) and corresponding ground-truth images (I^H) are assorted and fed to the discriminator (D) to check if it can distinguish between them. This task is analogous to binary classification where the job of the discriminator is to classify any given image into two classes: I^Sor I^H. Based on D's output, the binary cross-entropy loss is calculated using Equation 23, and subsequently, its weights are updated using backpropagation (Aggarwal 2018).

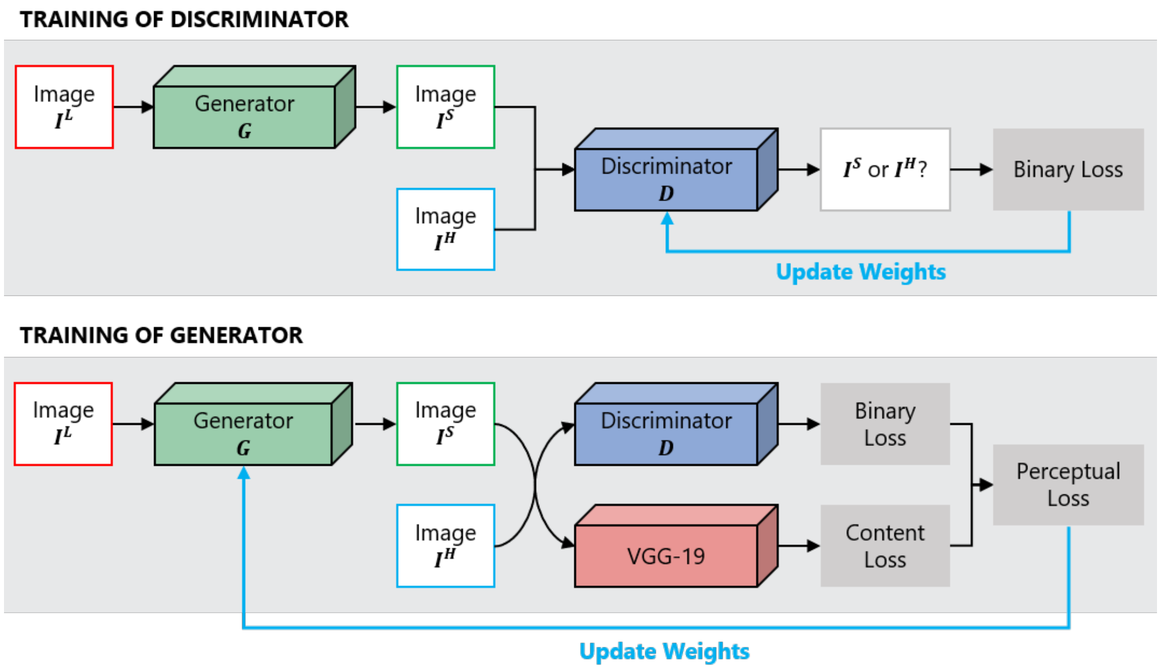


Figure 33. Schematic diagram of one iteration of training GAN models.

Next, the process is repeated with another batch of I^L images and binary loss is calculated, but without updating D's weights. This time, G's outputs (I^S) and corresponding ground-truths (I^H) are also fed to the VGG-19 model and the content loss is calculated using Equation 24. Based on the VGG-19's content loss and D's binary loss, the perceptual loss is subsequently calculated using Equation 25, and G's weights are updated. These sequential updates of D and G accomplish one iteration of training. The number of iterations in one epoch is equal to the number of training images divided by the number of images in one batch (a.k.a. batch size), rounded to the lowest integer.

Model testing

As mentioned earlier, two previously trained YOLO models – YOLO-BEW and YOLO- WHV – are tested on the low-resolution images (I^L) and super-resolved images (I^S) generated by the trained G model. Each model takes a 416×416 resolution image and outputs bounding boxes for detected objects. To investigate the influence of different image resolutions, I^L images are created by letterboxing the original testing images to sizes 52×52, 72×72, 96×96, 144×144, and 208×208 resolutions. Following this step, to test the performance of the object detection model on low-resolution images (referred to as Model-LR), I^L images are directly fed to the YOLO models and bounding boxes for detected objects are recorded. On the other hand, to test the performance of object detection model on GAN-improved images (referred to as Model-SR), first, each I^L image is broken down to 2×2 tiles and stacked into one batch of 4 images. Next, the batch is given to the trained G model to generate corresponding 4 super-resolved images, which are then tiled back to create the full image I^S. Finally, this I^S image is supplied to the YOLO models and detected bounding boxes are recorded.

To evaluate the quality of the generated I^S images, the content loss is calculated using Equation 24 to determine how much useful content is missing in I^S images compared to the I^H (ground-truth) images (Ledig et al. 2017). The lower value of content loss implies higher perceptual similarities between the two images. Additionally, another metric, called BRISQUE, is used (Mittal et al. 2012) to evaluates an image as a whole and measure the possible loss of naturalness in it. Similar to content loss, the lower score of BRISQUE indicates better quality of the image. Finally, the performance of YOLO object detection models is measured by calculating AP for each class and taking the average of these, a.k.a., mAP (Nath et al. 2020).

Results and Discussion

Quality of the generated images

Figure 34 shows the examples of I^L, I^S and I^H images and Table 4 lists the content loss and BRISQUE score of the I^L and I^S images for different resolutions. The Table shows that the higher the input resolution the lower the content loss. Intuitively, the higher resolution images preserve the contents of the original image. It can be seen that for 52×52, 72×72, and 96×96 images, I^S images have lower content loss than the corresponding I^L images, indicating that some of the contents lost in the I^L images (when resized from I^H images) are successfully retrieved by the GAN model in the I^S images. However, for higher resolutions, i.e., 144×144 and 208×208, $\boldsymbol{I^L}$ images have slightly better contents than the I^S images. One possible reason is that the GAN model is trained on low-resolution (48×48) images and, therefore, performs better on those images. The BRISQUE scores in Table 4 show that I^S images have smaller score (or distortion) and thus, higher naturalness compared to the I^L images. Moreover, the score does not vary much with the change in input resolution, indicating a consistent level of naturalness in the I^S images.

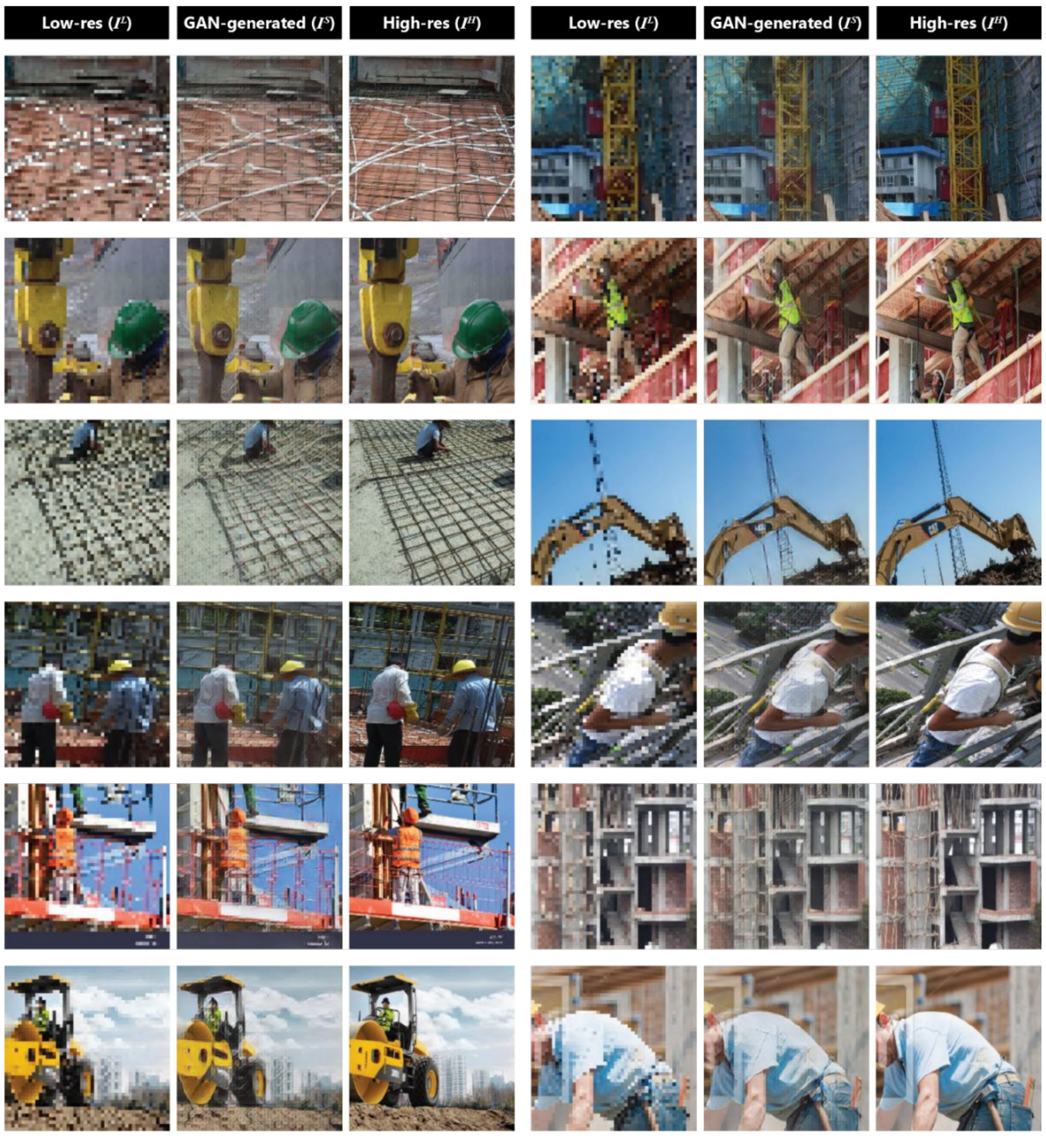


Figure 34. Examples of low-resolution (I^L), GAN-generated (I^S), and high-resolution (I^H) images.

Table 4. Content loss and BRISQUE score of the low-resolution (I^L) and GAN-improved (I^S) images (lower is better).

Input Size	Model	Content Loss	BRISQUE
52×52	I^L	61.3	85.4
	I^S	57.2	43.8
72×72	I^L	51.0	75.0
	I^S	48.5	45.1
96×96	I^L	40.8	68.5
	I^S	39.2	45.4
144×144	I^L	23.7	65.0
	I^S	26.4	45.7
208×208	I^L	12.2	60.9
	I^S	16.0	46.1

Performance of object detection

The mAP of the YOLO-v3-BEW and YOLO-v3- WHV models is illustrated in Figure 35(a) and Figure 35(b), respectively. The Figure shows that with the increase of input size, the performance of both models improves. Particularly, for lower resolution input images, the models perform remarkably better with the SR images than with the counterpart LR models. For example, for 52×52 images, compared to the LR images, the YOLO-v3-BEW and YOLO-v3- WHV models are 14% and 18% better, respectively, with the SR images. However, for higher resolution images, the difference in the performances for LR and SR images slightly drops. For example, for 208×208 images, compared to the LR images, the YOLO-v3-BEW and YOLO-v3-WHV models are only 4% and 5% better, respectively, with the SR images. It indicates that when the input resolution is lower, the GAN model can generate significantly better content-rich images. However, if the image resolution is already sufficiently good, it leaves less improvement room for the GAN model. Nonetheless, the results indicate that the GAN has a significant potential to enhance the quality of images which subsequently allows the YOLO models to perform better in detecting objects in the enhanced images.

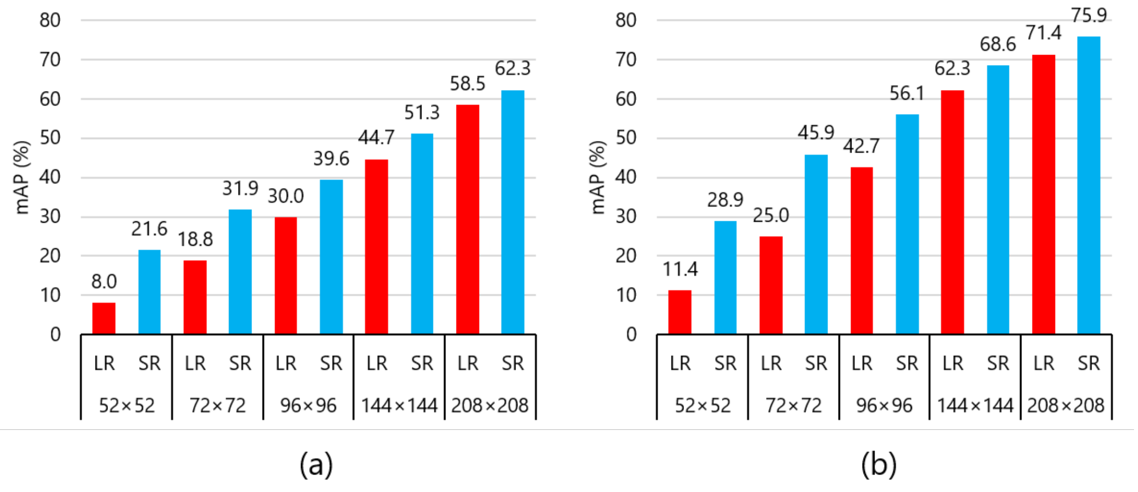


Figure 35. Performance of the YOLO-v3-BEW (a) and YOLO-v3-WHV (b) models with LR and SR input images.

Example of object detection with GAN image

Figure 36 shows an example of object detection by YOLO-WHV model for I^L and I^S images with an input size 96×96. The Figure shows that for both images, workers $W1$, $W3$, and $W4$, as well as their PPE components (hat and vest) are detected correctly. However, for worker $W2$, the model missed the hat and incorrectly detected the yellow bucket as a hat in I^L image. Meanwhile, in the I^S image, the model not only did detect $W2$'s hat and vest correctly but also detected the vest with higher confidence (85%) compared to the 49% confidence in the corresponding detection in I^L image. This example highlights some of the primary reasons why YOLO models achieve better mAP on the I^S images than the counterpart I^L images.

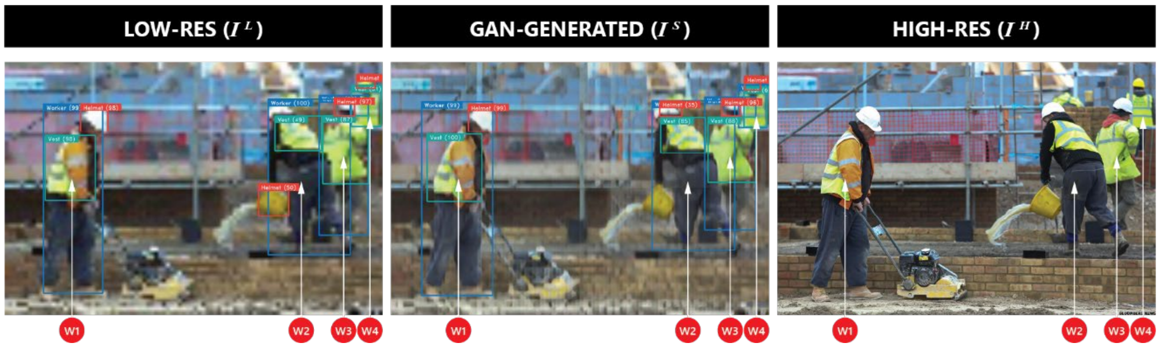


Figure 36. An example of object detection by YOLO-WHV for low-resolution (I^L) and GAN-generated (I^S) images with an input size of 96×96.

CHAPTER VIII

ACTIVE VISION SYSTEM

As mentioned earlier, occlusion is one of the prevailing issues that hinder the performance of CV-based algorithms. This Chapter describes the details of an active vision system that allows a camera to autonomously navigate in an environment to collect occlusion-free images of the objects of interest.

Intelligent Visual Data Acquisition

Simulated environment

In order to obtain occlusion-free images of a congested workspace (e.g., construction jobsite), one potential solution is to perform smart spatiotemporal navigation by mounting an ordinary camera on a UAV with AI capabilities, in the form of RL. The AI model in this scenario will allow the camera to adjust its position and angle of view for the best target visibility in the scene. To demonstrate the key steps of the proposed method, as shown in Figure 37, a single construction worker standing on the site is considered as the target object. The environment is created in Autodesk® Maya® using commercially available 3D models of a construction site and a worker. As shown in the Figure, the workspace is a 17.5×17.5 square foot area with the worker standing near the center. A 3D camera, which serves as the RL agent, is created and positioned at 27.5 feet away from the center of the site. This RL camera can move in a circular orbit of 55-foot diameter while maintaining its aim at the center of the site.

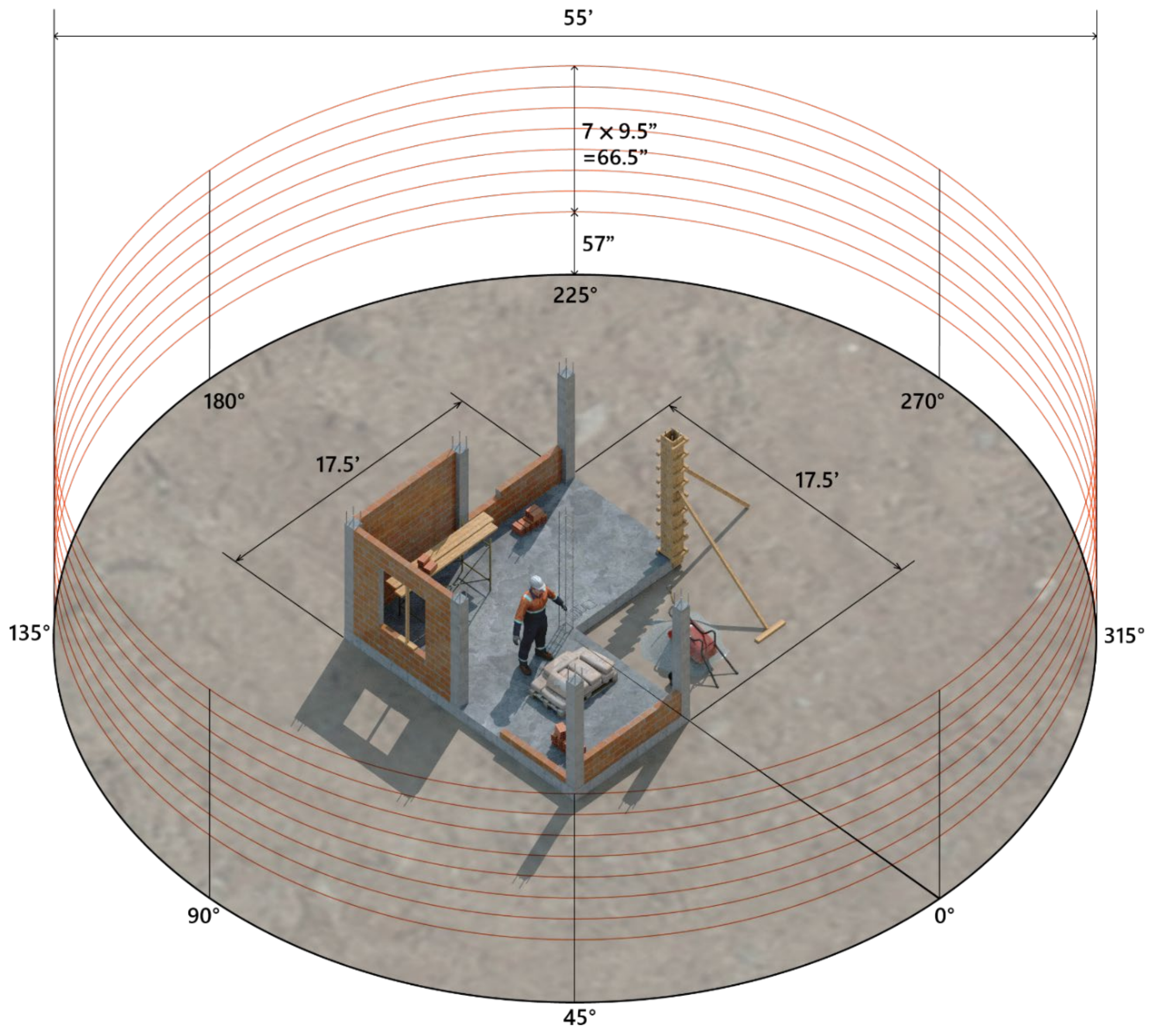


Figure 37. 3D model of the worker and the construction site for the RL experiment.

The space, in this experiment is assumed to be discrete. At each time step, the RL camera can move 1° arc length. Therefore, there are a total of 360 positions (referred to as horizontal positions hereafter) where the camera can position itself. The camera can also move vertically. There are 8 discrete elevations on which the camera can fly. As shown in Figure 37, the lowest elevation is at 57 inches above the ground, and the remaining elevations are 9.5 inches apart from each other. Hereafter, these elevations are referred to as vertical positions. The RL camera can take the "left" or "right" action to move to the next left or right horizontal position, respectively. Similarly, the RL camera can take "up" or "down" action to fly to the immediate upper and lower vertical position, respectively. The camera can also decide to "do nothing" and hover at its current position.

Mathematical formulations

Visibility

The definition of visibility of a worker in an image can be derived based on the end-use of that image. In this Dissertation, it is assumed that the image will be used to train deep learning (DL) algorithms for object detection in support of tasks such as worker's activity recognition (Nath et al. 2018; Han et al. 2013; Aggarwal et al. 2014; Zhao et al. 2012), or safety monitoring through continuous measurement of PPE compliance (Nath et al. 2020; Park et al. 2015; Fang et al. 2018). Previous studies have found that the performance of DL algorithms in extracting contextual information from images is positively correlated with the size of objects in that image (Nath et al. 2020; Redmon et al. 2018). Therefore, the visibility of an object is defined in terms of its footprint in the image, hereafter referred to as immersive visibility, and mathematically expressed as the ratio of the pixel area of the object to the total pixel area of the image, represented as a percentage.

Assume, a gray-scale image M_{worker} of height H and width W represents the mask of the worker (as shown in Figure 38). A pixel $0 \leq M_{\text{worker}}(i,j) \leq 1$ represents the pixel at position (i,j). If this pixel belongs to the worker, $M_{\text{worker}}(i,j) = 1$, and if it does not belong to the worker, $M_{\text{worker}}(i,j) = 0$. It must be noted that if antialiasing is used to increase the degree of realisticness of the image by smoothing edges on curved lines (Szeliski 2019), some pixels (that fall on object edges) may partially contain the worker, in which case $M_{\text{worker}}(i,j)$ will contain fractional values between 0 and 1 (Figure 38). With this notation, we can define the immersive visibility of a worker using Equation 26. For example, immersive visibility $v_i = 1.75\%$ means that the worker covers 1.75% of the total pixels in the entire image.

$$v_{\text{immersive}}(\%) = \frac{1}{HW}\sum_{i=1}^{H}\sum_{j=1}^{W} M_{\text{worker}}(i,j) * 100 \tag{26}$$

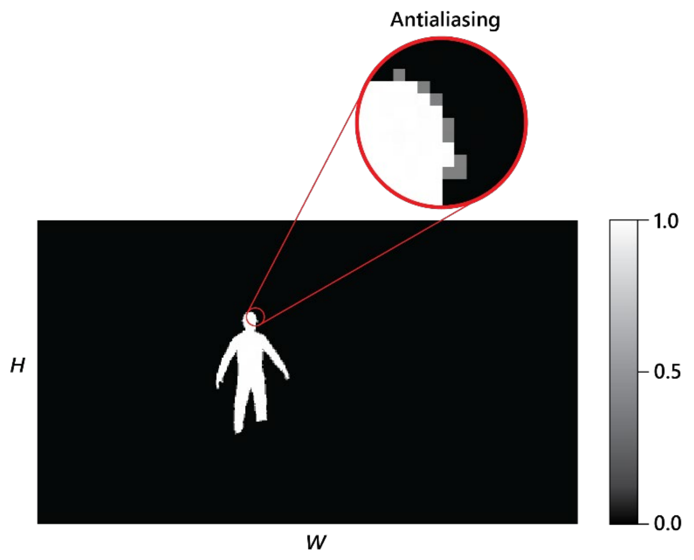


Figure 38. Mask of workers and antialiasing effect.

Also, in some applications, it is important that certain parts of the object be completely (or sufficiently) visible to perform desired recognition task. For example, to recognize PPE attire (hard hat and safety vest) of a worker, the worker's head and trunk must be completely visible to the camera. Therefore, visibility can also be defined in terms of completeness (referred to as complete visibility hereafter). In this way, a body part (i.e., head, trunk, left hand, right hand, left leg, or right leg) is called completely visible in the image if no other object (even another body part) blocks that body part, i.e., in other words, there is no occlusion. Assume, as shown in Figure 39, a gray-scale image $M_{\text{current},b}^{(m)}$ represents the mask of body part b of the worker in the current frame m. Also, another gray-scale image $M_{\text{complete},b}^{(m)}$ represents the mask of body part b in the same frame but in the absence of any occlusion (i.e., when the body part b is completely visible). Mathematically, the complete visibility of body part b is defined as shown in Equation 27,

$$v_{\text{complete},b}^{(m)} = \frac{\sum_{i=1}^{H} \sum_{j=1}^{W} M_{\text{current},b}^{(m)}(i,j)}{\sum_{i=1}^{H} \sum_{j=1}^{W} M_{\text{complete},b}^{(m)}(i,j)} \tag{27}$$

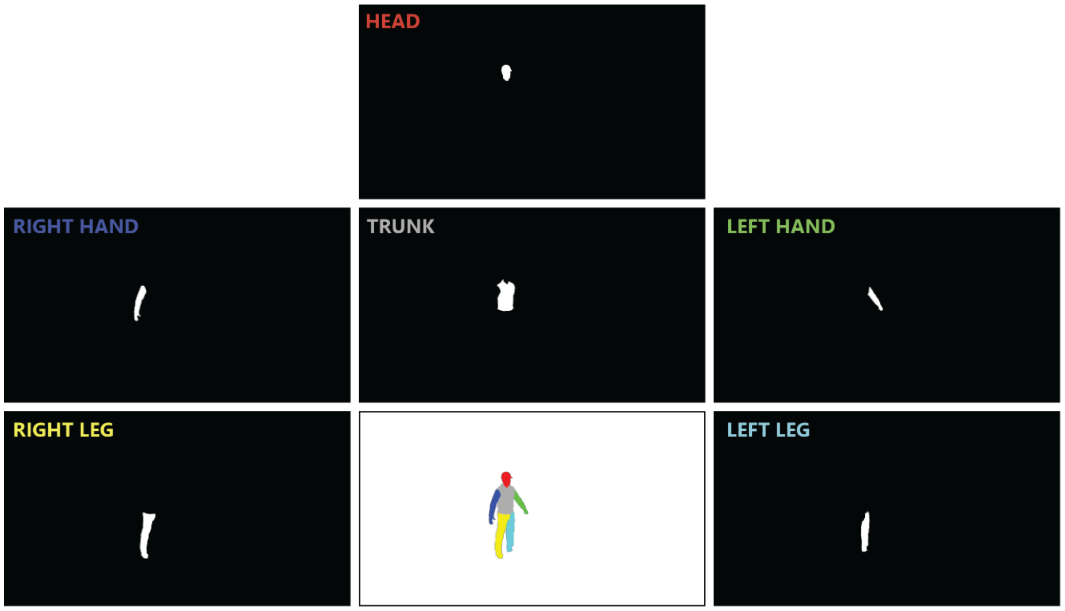


Figure 39. Mask of different body parts of the worker.

Therefore, by definition, the maximum value of $v_{\text{complete},b}^{(m)}$ is 1, which indicates that body part b is completely visible in frame m. If the camera moves in a circular orbit with the worker at the center (i.e., the worker is always at a constant distance from the camera), and there exists at least one point in the orbit from where the body part b is completely visible, then, in the image captured from that point, the body part will have the maximum pixel area. Thus, for this special case, visibility $v_{\text{complete},b}^{(m)}$ is defined as shown in Equation 28, where $M_b^{(m)}$ represents the mask of body part b in frame m, and $n \in F$ represents any frame n in the set of all frames F.

$$v_{\text{complete},b}^{(m)} = \frac{\sum_{i=1}^{H} \sum_{j=1}^{W} M_b^{(m)}(i,j)}{\max\limits_{n \in F}\left[\sum_{i=1}^{H} \sum_{j=1}^{W} M_b^{(n)}(i,j)\right]} \tag{28}$$

Once the complete visibility of k body parts, $b_1, b_2, \dots, b_k$, are calculated, complete visibility of the worker, $v_{\text{complete}}^{(m)}$, can be obtained by taking L_2 norm (a.k.a. Euclidean distance) of complete visibility of all body parts, as shown in Equation 29. The use of L_2 norm instead of L_1 norm (i.e., $\sum_{i=1}^{k} v_{\text{complete},b_i}^{(m)}$) is because mathematically, L_2 norm offers better stability, particularly, in an optimization task. For example, between two points in a high-dimensional space, there might be multiple paths where L_1 distance is minimum. However, there is only one path (i.e., a straight line connecting those two points) where L_2 distance is minimum.

$$v_{\text{complete}}^{(m)} = \sqrt{\sum_{i=1}^{k} \left(v_{\text{complete},b_i}^{(m)}\right)^2} \tag{29}$$

From Equation 29, it can be inferred that the maximum value of v_c is $\sqrt{k}$, corresponding to the case where all k body parts of the worker are completely visible. It must be noted that while all body parts are considered to be equally important in Equation 29, it is possible to assign different weights w_i $(0 \leq w_i \leq 1)$ to each body part b_i, as shown in Equation 30, depending on the end-use of the model. For example, to detect workers' PPE attire, it is logical to give higher weights to the head and trunk compared to other body parts.

$$v_{\text{complete}}^{(m)} = \sqrt{\sum_{i=1}^{k} \left(w_i \cdot v_{\text{complete},b_i}^{(m)}\right)^2} \tag{30}$$

Reward and discount factor

This Section describes the mathematical grounds for selecting the reward function and numerical value of discount factor γ to calculate Q-values using Equation 31. First, assume the agent starts from state S_0 and takes the action $a \in \{\text{up, down, left, right}\}$. After time t, the agent reaches the terminal state and receives a reward r_t. For simplicity, it can be assumed that the rewards received in the intermediate states are negligible compared to the terminal reward. Therefore, Equation 31 can be deduced, as follows.

$$Q(S_0, a) = \gamma^t r_t \tag{31}$$

From Equation 31, it can be seen that the Q-value is directly proportional to the terminal reward. Since the goal of the agent is to maximize the Q-value, this implies that the agent will try to maximize the terminal reward. If the objective is to maximize the worker's visibility, a reasonable approach would be to set the terminal reward as the visibility of the worker at the terminal state. Assume, v_t is the visibility (immersive or complete, based on the application) of worker at time t and $r_t = v_t$. Therefore, Equation 31 can be rewritten as shown in Equation 32.

$$Q(S_0, a) = \gamma^t v_t \tag{32}$$

Nonetheless, at state S_0, the agent could have decided to do nothing which will terminate the episode and give a terminal reward $r_0 = v_0$, where v_0 is the visibility (immersive or complete) of the worker at time $t = 0$. This is expressed in Equation 33,

$$Q(S_0, \text{do nothing}) = r_0 = v_0 \tag{33}$$

At state S_0, the agent will be motivated to select action a instead of doing nothing only if $Q(S_0, a) > Q(S_0, \text{do nothing})$, or $\gamma^t v_t > v_0$. By setting the two sides equal, an equation for γ can be derived, as shown in Equation 34.

$$\begin{gathered} \gamma^t v_t = v_0 \\ \Rightarrow \gamma = \left(\frac{v_t}{v_0}\right)^{-\frac{1}{t}} \end{gathered} \tag{34}$$

If $\gamma < 1$ and $t > 0$, then $v_t/v_0 > 1$ indicating an improvement to visibility (immersive or complete). The mathematical relations, shown in Equations 32 through 34, can be interpreted as follows: If the agent believes that the visibility of the worker can be improved by more than v_t/v_0 within t timesteps, it will move in the direction that leads to visibility v_t; otherwise, it will stay put at its current position. Clearly, the value of v_t/v_0 and t in Equation 34 can be manually set to calculate γ. However, these values should be select based on the desired outcome. For example, selecting a v_t/v_0

close to 1 can make the agent too greedy (and perhaps, unstable), i.e., even for slight improvement, the camera might continuously change its position. On the other hand, a large v_t/v_0 value can make the agent too ambitious, i.e., the camera might not move unless there is a potential for huge improvement of visibility. If the value of t is very small, the camera might be too impatient, i.e., if it cannot gain the desired improvement within a few steps, it might not do anything. In contrast, a large t value can make the camera too patient and optimistic, i.e., it might continuously search for improvements forever.

Model training

The expectation is that a fully trained RL camera, if it determines that the worker is occluded by any object, would automatically adjust its position by moving around the site so that from the new position the worker is better visible to the camera. For this experiment, a DQN model (Géron 2019) (Figure 40) will be used which takes the RGB image captured by the camera (i.e., the state of the environment) and returns the Q-values corresponding to each action – “left”, “right”, “up”, “down”, and “do nothing”. The maximum reward that can be achieved by taking an action at a particular time is called the Q-value of that action at that time (Géron 2019; Szepesvári 2010; Sutton et al. 2018). Therefore, the camera would take the action which has the highest Q-value.

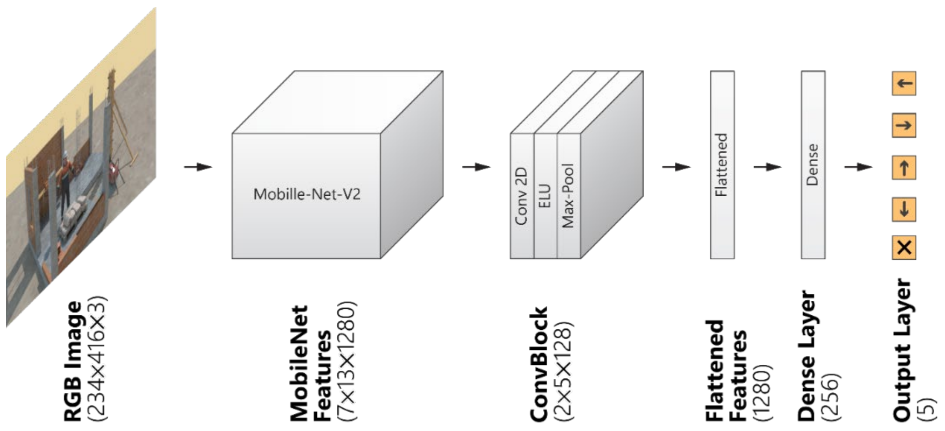


Figure 40. Layers and dimensions of the features in the DQN model.

The reward function for this RL experiment is defined based on the visibility of the worker. However, the visibility of a worker is mathematically defined by the total percentage of the pixel area the worker occupies in the image. Therefore, by maximizing the reward, the RL camera would maximize the visibility of the worker.

During the training, an ε-greedy policy is followed which balances the exploration versus exploitation dilemma (Géron 2019; Sutton et al. 2018) for the RL camera. Particularly, with a probability ε, the RL camera will explore the construction site by taking randomly selected actions at different time steps. However, for the other times (with probability $1 - \varepsilon$), the RL camera will exploit its current knowledge about the construction site and will select the action that is known best to maximize the reward function.

At each episode of training, the RL camera is started from a random position in the environment. Then, it is given a maximum number of time steps (a.k.a., time-out) to find a better position to view the worker. At the end of the episode, a reward is given based on the final visibility of the worker and the time taken to achieve this visibility.

Next, the weights of the DQN model are updated based on the reward value achieved. This process is iterated a number of times until the RL camera gets sufficiently trained so that it consistently achieves a high reward in each episode. For example, the preliminary results of the training are shown in Figure 41 (immersive visibility) and Figure 42 (complete visibility). The Figures show the average reward received in 1,000-episode intervals. It can be seen that after ~200,000th episodes, the RL camera continues to achieve better rewards.

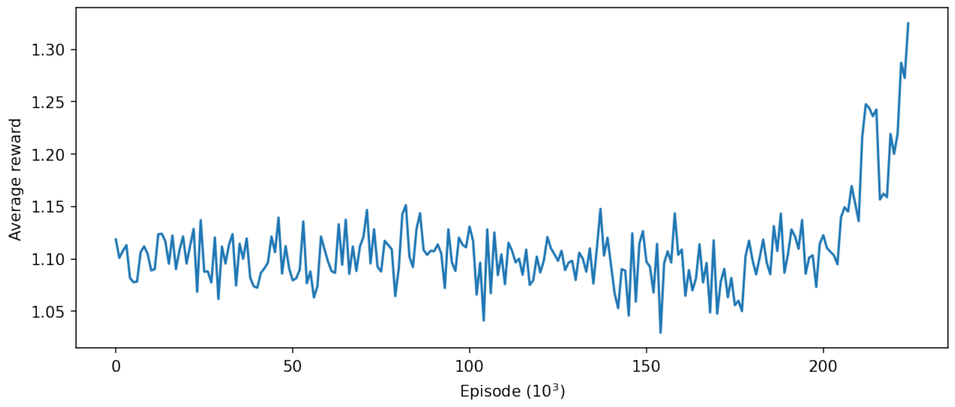


Figure 41. Average reward received in 1,000-episode intervals during training with immersive visibility.

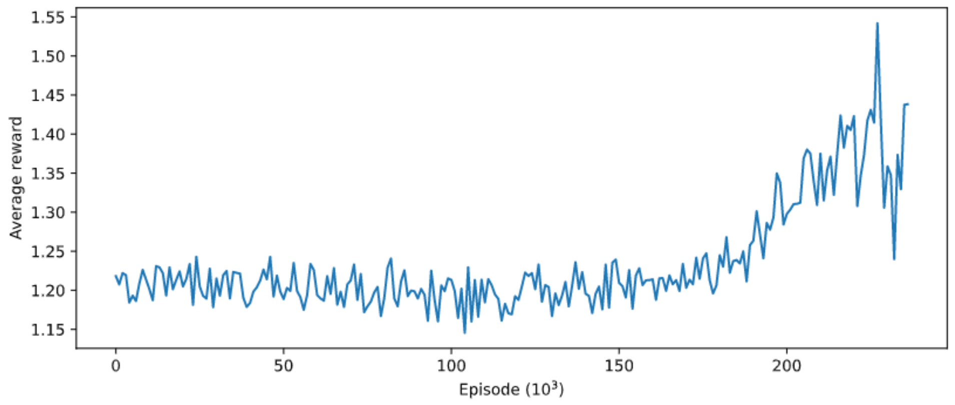


Figure 42. Average reward received in 1,000-episode intervals during training with complete visibility.

Performance evaluation

Figure 43 (immersive visibility) and Figure 44 (complete visibility) display initial visibility (v_O) versus final visibility (v_P) for different camera positions. The 45° equality line represents the positions where $v_P = v_O$, i.e., the visibility of the worker remained the same.

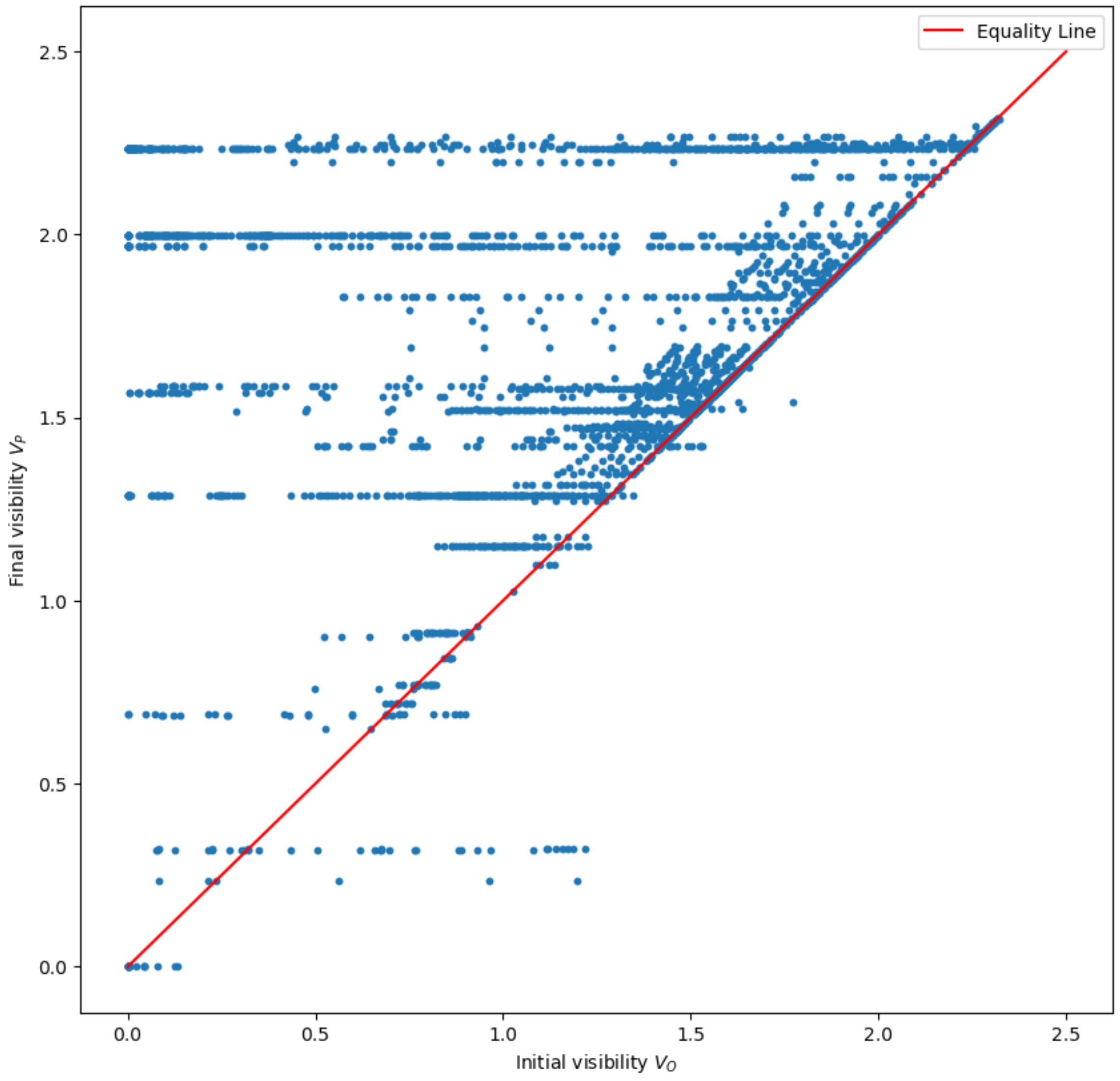


Figure 43. Improvement in immersive visibility for various initial positions of the camera.

In Figure 43, 96.77% of the points are on or above the equality line (i.e., $v_P \geq v_O$) and 80.42% of the points are above the equality line (i.e., $v_P > v_O$). Also, in Figure 44, 89.41% of the points are on or above the equality line (i.e., $v_P \geq v_O$) and 72.19% of the points are above the equality line (i.e., $v_P > v_O$). These results imply that in most of the times, the camera was successful in improving visibility. There are also a limited number of points below the equality line, i.e., $v_P < v_O$, i.e., depicting cases where visibility was decreased as a result of camera movement.

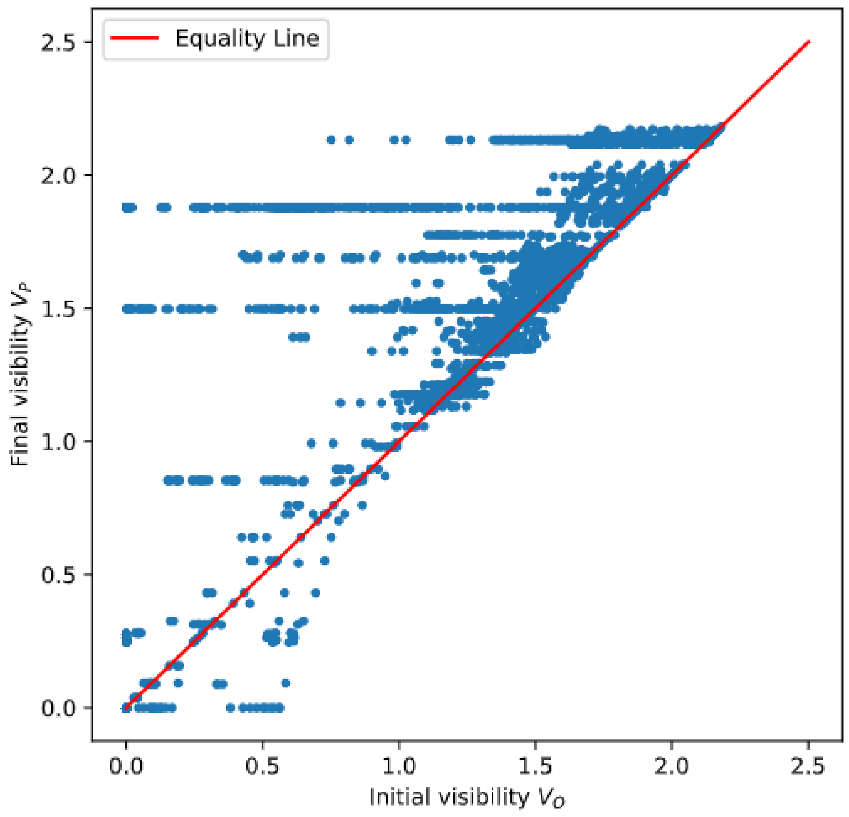


Figure 44. Improvement in complete visibility for various initial positions of the camera.

Surveillance of Warehouse Operations

Environment

In the construction domain, jobsite safety and accident prevention is one of the most active areas of research with real implications (OSHA 2019). Therefore, to complement the previous experiment, another scenario is created and tested, this time in a real-world setting, to find forklifts in an active warehouse environment. Particularly, an RL agent with active vision capability is trained to locate and monitor forklifts in a dynamic space. The framework can be implemented in a jobsite for alerting workers of imminent contact collisions (e.g., due to the worker and forklift moving too close to each other, or the worker moving into the blind spot of the forklift).

The environment is generated from a 360° video of warehouse operation, which is referred to as Pictor-360 video. Figure 45 shows a sample from the Pictor-360 video. As shown in the Figure, the frame encompasses a 360° panoramic view of a warehouse. The resolution of each frame is 5120×1080 from which a window of 420×420 portion is cropped to mimic the camera view of an autonomous drone. At each step, the camera can rotate 11.25° to left or right (i.e., rotation along yaw axis) and 5° in the up or down direction (i.e., rotation along pitch axis). These actions result in translating the 420×420 window 160 pixels along the horizontal direction (when the drone moves left or right) and 60 pixels along the vertical direction (when the drone moves up or down), as shown in Figure 45.

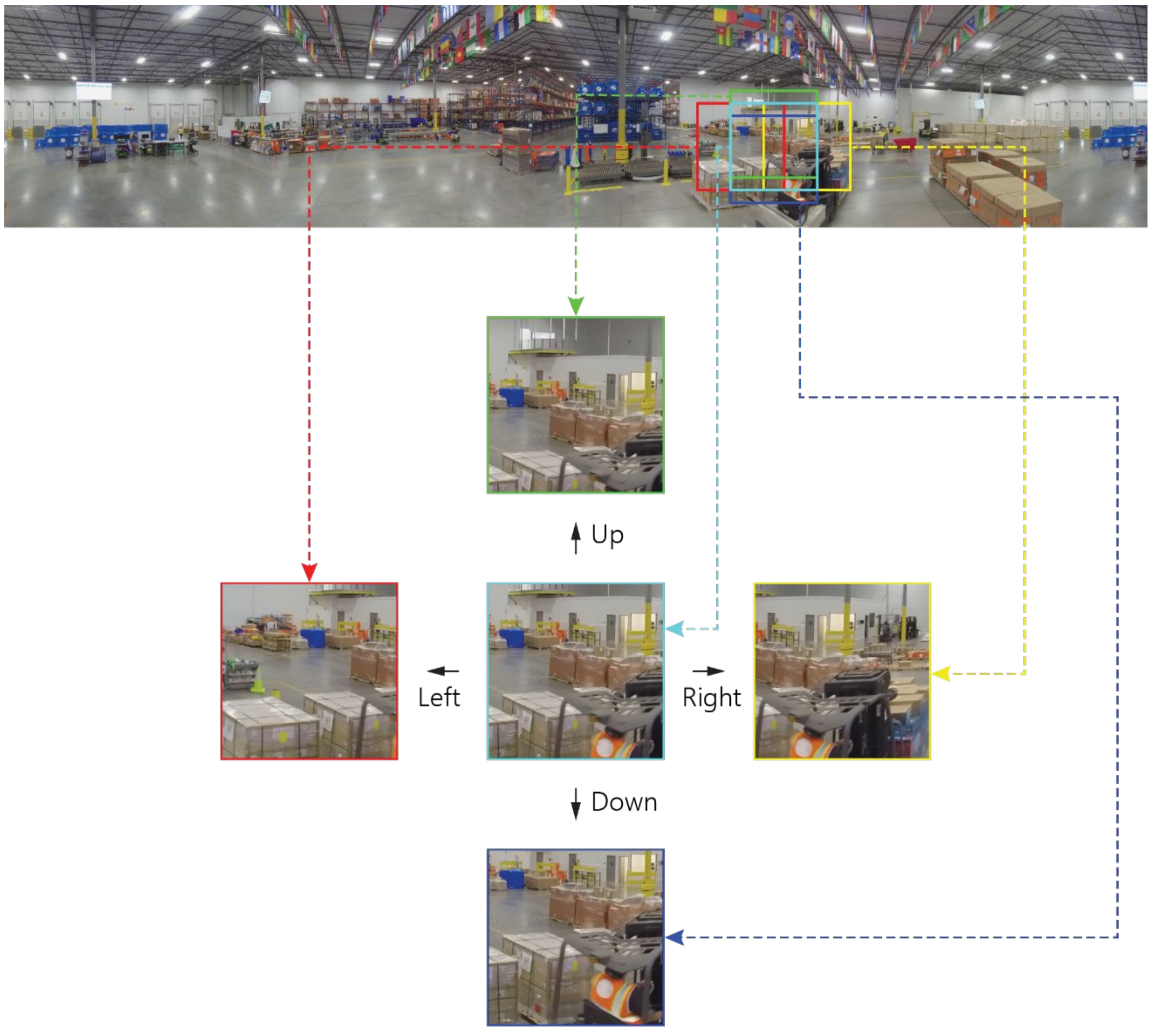


Figure 45. A sample frame of the Pictor-360 video and extracted camera views by taking different actions.

Model training

In this experiment, a VGG-16 model is selected to extract features because of its high performance in classifying numerous real-world objects. The architecture of the model is shown in Figure 46. The VGG-16 network is amended by three convolution blocks, each consisting of a convolution with ELU activation function, followed by a max-pooling layer. Features are then flattened and connected with a dense layer with 64 nodes. The final output layer consists of 5 nodes to predict individual Q-values corresponding to the five actions.

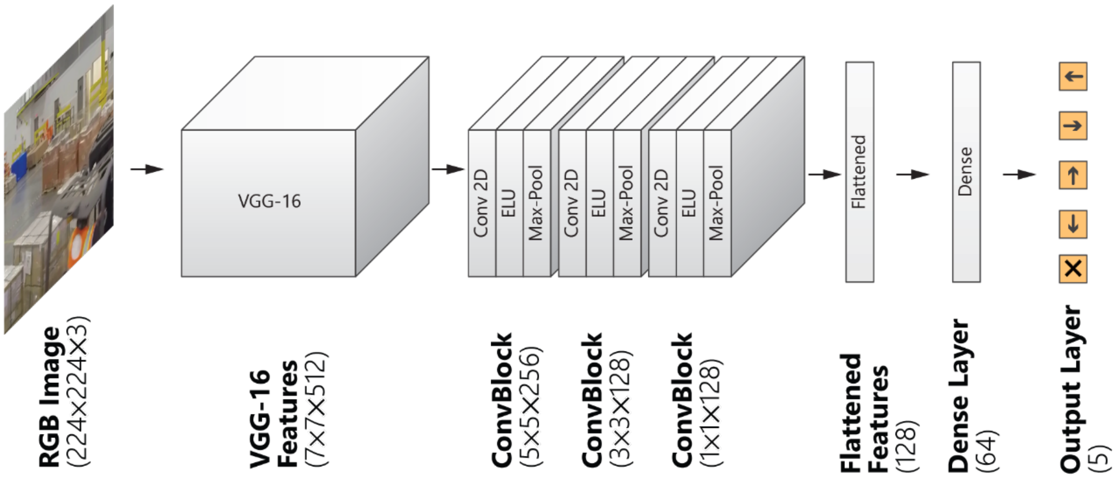


Figure 46. The architecture of the DQN model for the Pictor-360 experiment.

Model training follows similar steps to the previous RL experiment. However, in this experiment, the reward is defined based on the IoU between the box of the forklifts and the viewing window of the RL camera. The hyperparameters for this experiment are listed in Table 5. The outcome of training is shown in Figure 47, which illustrates the average reward received by the RL agent in 1,000-episode intervals. It can be seen that the model achieves higher rewards with the progress of training.

Table 5. Hyperparameters for the Pictor-360 experiment.

Category	**Hyperparameter**	**Value**
Environment	Number of states	268,416
	Number of actions	5
	Image resolution	420×420
	Time-out	100
Reward	Step reward	10×IoU
	Discount factor, γ	0.995
Policy	$\varepsilon_{\mathrm{max}}$	1
	$\varepsilon_{\mathrm{min}}$	0.1
	$n_{\mathrm{iteration}}$	2,000,000
Experience	Deque memory size	100,000
	Batch size	20
Training	Number of iterations	2,000,000
	Warm-up period	5,000 steps
	Target model update interval	5,000 steps
	Optimizer	Adam
	Learning rate	0.0001

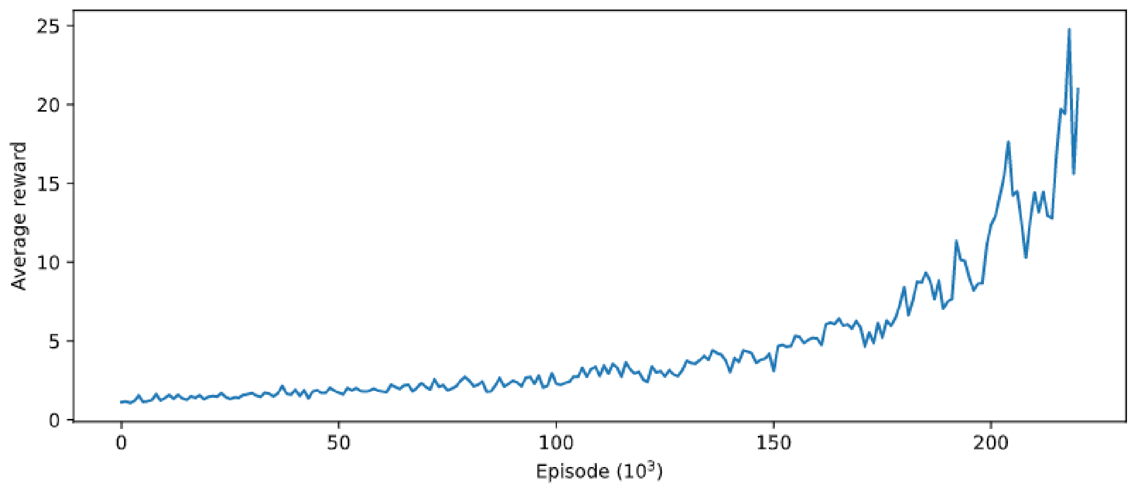


Figure 47. The average reward received in 1,000-episode intervals during training for the Pictor-360 experiment.

Performance evaluation

Figure 48 displays the initial IoU (I_O) versus maximum IoU (I_m) from testing the RL camera starting from randomly selected 100 different positions. The 45° equality line represents the positions where $I_m = I_O$, i.e., the visibility of the worker remained the same. In this experiment, all (100%) of the points are on or above the equality line (i.e., $I_m \geq I_O$) and 42% of the points are above the equality line (i.e., $I_m > I_O$). This means that in 42% of the times the RL agent found a position from where the forklift was more visible, indicating the effectiveness of the RL agent in successfully locating the object of interest in complex and dynamic real-world settings. It is worth noting that in the remaining 58% of cases, the visibility of the forklift was preserved at the same level, and in no cases, the movement of the RL agent resulted in lower visibility.

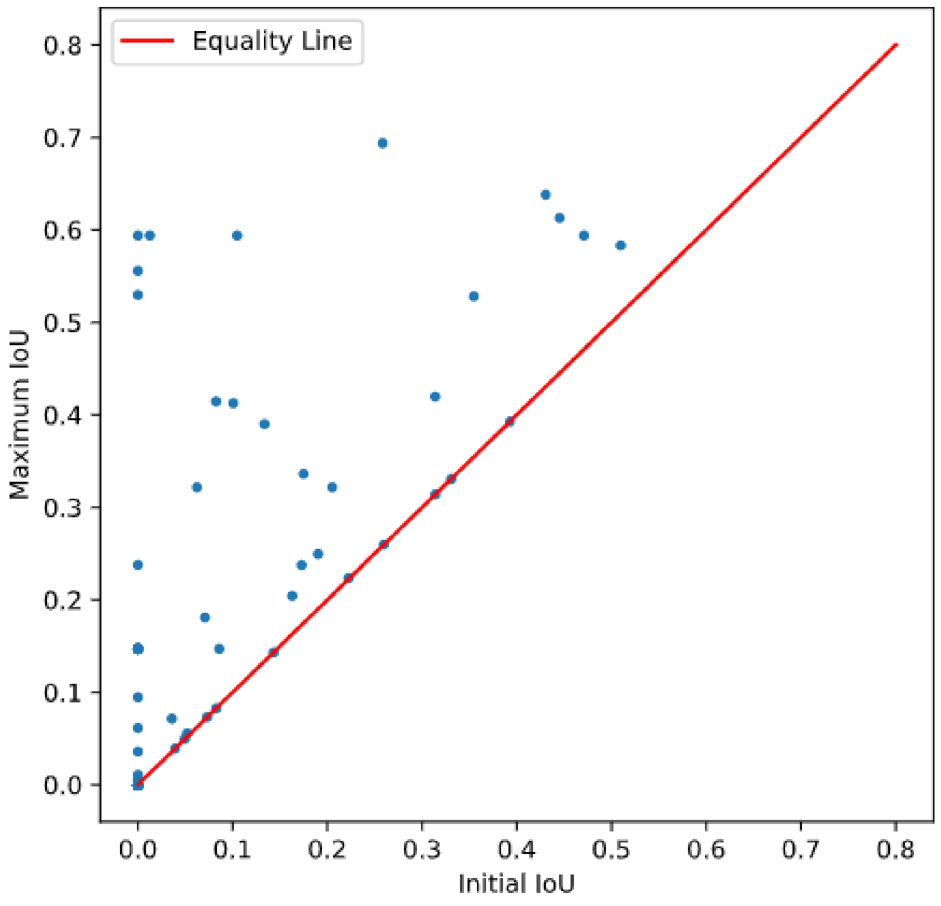


Figure 48. Improvement in IoU, for finding forklifts in the warehouse, for various initial positions of the camera.

CHAPTER IX

CONCLUSION

Summary and Discussion

The research presented in this Dissertation aimed to advance the knowledge necessary for effectively implementing human-machine collaboration in the construction site. Particularly, the ultimate objective of this Dissertation was to design and evaluate human-centered AI-based solutions to fundamental challenges in construction including content and information retrieval, jobsite safety, and intelligent visual data acquisition. To achieve this overarching goal, first, fundamental methods most closely related to the scope of the research were thoroughly investigated in order to identify key domain-specific challenges and potential solutions. Next, following a bottom-up approach, a host of AI-enabled computing and visual analytics were designed and tested with several large construction-related datasets that were generated as part of this research.

In Chapter IV, in order to rapidly retrieve contents from a large volume of construction imagery, single-label classification was performed on the Pictor-v1.0 dataset. Particularly, a VGG-16-based classifier model was trained and found to be ~90% accurate in assigning single-label tags (i.e., building, equipment, worker) to individual images. However, when a single image contained multiple object types, the model showed a tendency to assign the class that had a larger visual footprint in the image, a behavior that might disagree with that of a human annotator in some cases. For example, the model classified an image as one that contained a building (appearing as a large object but in the background) whereas a human annotator noticed a worker standing in the foreground as the main content of the image.

Therefore, in Chapter IV, multi-label classification was performed on the Pictor-v1.1 dataset where another VGG-16-based classifier assigned multiple labels (i.e., building, equipment, or worker) to each individual image if those objects were present and visible. Although the model was 86.0% accurate on average, it could only indicate the presence of an object without precisely locating that object within the coordinate frame of the image. Therefore, in Chapter V, object detection was performed to localize the objects of interest and classify them at the same time. Particularly, YOLO algorithm was used given its known ability to perform real-time object detection without noticeably compromising accuracy. Two variants of the YOLO algorithm, YOLO-v2 and -v3, were applied to various combinations (crowdsourced, web mined, and combined) of the Pictor-v2 dataset. To note, the crowdsourced and web mined portions of the dataset were visually distinguishable; while crowdsourced images were more challenging and less structured, web mined images were cleaner and more structured. It was found that the YOLO models perform best when trained and tested on images from the same category (crowdsourced, web mined). However, the performance increased when the models were trained with a combination of crowdsourced and web mined images. This finding indicates that using diverse images for training helps the model learn generalizable features which allows it to perform better in object detection. It was also found that, for any combination of training and testing images, the YOLO-v3 model performs better than the YOLO-v2 model, a behavior that can be attributed to the three output layers of the YOLO-v3 model that are specifically designed for detecting three different object sizes – large, medium, and small. This also indicates that the size of the object is one of the important factors that can influence the performance of object detection models. To investigate this further, the YOLO-v3 model was tested for objects under various visual conditions, e.g., if the object was large or small, crowded or not, and well-lit or poorly lit. Results showed that the model tends to perform better when the object is large, not crowded, and well-lit. Particularly, for detecting workers, the model achieved near-human level precision and recall (~98%) when the worker appeared large in an image.

Motivated by the satisfactory performance of the YOLO-v3 model for general applications, Chapter VI used this model in a more specific task of verifying workers' compliance with PPE requirements. Particularly, three approaches were proposed. In the first approach, worker, hat, and vest objects were detected separately, resulting in 81.2% mean average precision (mAP). The second approach achieved 72.3% mAP in detecting the worker based on their PPE attire – W (no hat or vest), WH (with only hat), WV (with only vest), and WHV (with both hat and vest). The third approach only detected workers, however, with 85.6% mAP. In this case, the detected portion of the image (containing the worker object) was subsequently cropped and forwarded to another algorithm to perform PPE detection. Applying the VGG-16, ResNet-50, and Xception classifier models to the cropped worker images yielded 78.2%, 77.8%, and 76.8% accuracy in detecting PPE attire, respectively. Next, a Bayesian scheme was used to combine the individual detections from each classifier and resulted in improved performance (67.9% mAP as opposed to 62.6%, 64.2% and 63.2% mAPs for VGG-16, ResNet-50, and Xception models, respectively) in classifying the input image into W, WH, WV, and WHV. While these models were able to detect the presence of PPE components, another technique, based on PRID was also investigated to detect the color of the PPE components. By comparing the query image of a worker (with unknown PPE) with a gallery of images of workers (with known PPE), the method was able to infer the PPE attire of the query worker from the PPE attire of the best-matched (i.e. most similar) gallery worker. The model achieved 90% accuracy in determining the PPE attire of workers, and 77% accuracy in recognizing the color of each PPE component.

While the abovementioned methods demonstrated promising results in support of building autonomous systems for monitoring crew safety and productivity, all could be in vain if the images were of low quality. Therefore, in Chapter VII, a GAN model was trained and tested on the combination of Pictor-v2 and -v3 datasets to enhance image quality. Next, previously trained YOLO-v3 models were applied to GAN-improved images. It is found that the YOLO-v3 model, trained for detecting buildings, equipment, and worker, performed 4%-14% better with GAN-improved images compared to the low-quality counterparts. Similarly, the YOLO-v3 model, trained for detecting worker, hat, and vest, performed 5%-18% better with GAN-improved images in contrast to low-quality images.

Finally, in Chapter VIII, an intelligent vision-based navigation technique was designed to help a digital capture device (i.e., camera) obtain occlusion-free views of objects of interest (e.g., worker) in an environment. Two sets of experiments were conducted to quantify and analyze the capability of a drone-mounted (flying) camera in performing an active vision task first in a simulated 3D model of a construction workspace, and then in the 360-degree video of an active warehouse operation. Through the application of RL, the mathematical foundation of the reward function, and systematic training of a DQN, and the approach for measuring the performance of the algorithm were explained. It was found that in both experiments the RL model learned to perform the intended task by improving the received reward during training.

Concluding Remarks

Construction is one of the largest global industries with a major impact on the global economy (BLS 2019). However, it is also one of the least digitalized work domains with a substantial portion of the industry (mostly, the small businesses) reluctant to adopt new technologies (Peltier et al. 2012; Gandhi et al. 2016). The most direct consequence of this lack of technology integration is that a vast majority (almost 70%) of construction projects fail to finish within the estimated time and budget. Also, numbers show that about 37% of the assumptions made during the planning phase turn out to be untrue when the actual execution starts (Mieritz 2012). Moreover, frequent illnesses, injuries, and fatalities have marked construction as one of the most hazardous occupations in the world (BLS 2014). Therefore, there is an urgent need for a disruption in this industry to transform the current practices by providing the workforce with better means (informed by operations-level data) to perform the job faster, safer, and with better quality. Market research and technology trends have shown that new technologies such as BIM, 3D printing, VR/AR, IoT, and robotics – that enable automation in construction – have the potential to revolutionize the industry and improve productivity and safety by a significant margin (Manyika et al. 2017). Particularly, researchers and practitioners are foreseeing that human-machine collaboration in the construction site holds the key to improving safety, productivity, quality of work, and return on investment (NSF 2020; Autodesk 2020). Therefore, this Dissertation was an attempt to lay out the theoretical foundations of several AI-based applications that will enable and promote successful human-machine partnership in construction. In the future, the research presented in this Dissertation can be further extended, for example, by contributing to the design of autonomous robots with situation awareness for performing complex construction tasks. These robots can be trained through imitation learning where they learn an intricate task simply by observing the demonstration given by a human expert (Billard et al. 2011; Bonardi et al. 2020; Finn et al. 2017). Besides developing theoretical and practical models for construction robotics, it is also equally important to enable trust in AI and reskill the current workforce to prepare them for adopting new technologies (National Academies 2017; Siau et al. 2018). Toward this goal, by enabling and promoting a synergic relationship between the AI-powered machines and their human partners, a safer and more productive construction workplace can be established that can lead to positive socio-economic outcomes for all stakeholders.

REFERENCES

Aggarwal, Charu C. 2018. *Neural networks and deep learning* (Springer: Berlin, Germany). DOI: https://doi.org/10.1007/978-3-319-94463-0.

Aggarwal, Jake K, and Lu Xia. 2014. "Human activity recognition from 3D data: A review", *Pattern Recognition Letters*, 48: 70-80. DOI: https://doi.org/10.1016/j.patrec.2014.04.011.

Alippi, Cesare, Simone Disabato, and Manuel Roveri. 2018. "Moving convolutional neural networks to embedded systems: the Alexnet and VGG-16 case." In *Proceedings of the 17th ACM/IEEE International Conference on Information Processing in Sensor Networks*, 212-23. IEEE Press. DOI: https://doi.org/10.1109/ipsn.2018.00049.

Autodesk. 2020. "Future of work", Accessed January 19, 2021. https://www.autodesk.com/future-of-work.

Billard, Aude, and Daniel Grollman. 2011. "Imitation learning (of robots)." In *Encyclopedia of the Sciences of Learning* (Springer: Boston, MA). DOI: https://doi.org/10.1007/978-1-4419-1428-6_758.

BLS. 2014. "Nonfatal occupational injuries and illnesses requiring days away from work." Accessed November 1, 2016. https://www.bls.gov/news.release.

BLS. 2019. "Fatal occupational injuries counts and rates by selected industries", Accessed January 20, 2021. https://www.bls.gov/news.release/cfoi.t04.htm.

BLS. 2019. "Industries at a glance: Construction", Accessed January 20, 2021. https://www.bls.gov/iag/tgs/iag23.htm.

Bonardi, Alessandro, Stephen James, and Andrew J Davison. 2020. "Learning one-shot imitation from humans without humans", *IEEE Robotics and Automation Letters*, 5(2): 3533-39. DOI: https://doi.org/10.1109/lra.2020.2977835.

Bottou, Léon. 2010. "Large-scale machine learning with stochastic gradient descent." In *Proceedings of the Computational Statistics*, 177-186. Springer. DOI: https://doi.org/10.1007/978-3-7908-2604-3_16.

Brilakis, Ioannis K, and Lucio Soibelman. 2008. "Shape-based retrieval of construction site photographs", *Journal of Computing in Civil Engineering*, 22(1): 14-20. DOI: https://doi.org/10.1061/(asce)0887-3801(2008)22:1(14).

Buja, Andreas, Werner Stuetzle, and Yi Shen. 2005. "Loss functions for binary class probability estimation and classification: Structure and applications". Accessed January 20, 2021. http://www-stat.wharton.upenn.edu/~buja/PAPERS/paper-proper-scoring.pdf.

Chen, Xinlei, Hao Fang, Tsung-Yi Lin, Ramakrishna Vedantam, Saurabh Gupta, Piotr Dollár, and C Lawrence Zitnick. 2015. "Microsoft COCO captions: Data collection and evaluation server", *arXiv preprint arXiv:1504.00325*.

Chi, Seokho, and Carlos H Caldas. 2011. "Automated object identification using optical video cameras on construction sites", *Computer-Aided Civil and Infrastructure Engineering*, 26(5): 368-80. DOI: https://doi.org/10.1111/j.1467-8667.2010.00690.x.

Chollet, Francois. 2018. *Deep Learning with Python* (Manning Publications Co.: Shelter Island, NY).

Chollet, François. 2017. "Xception: Deep learning with depthwise separable convolutions." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1251-58. IEEE. DOI: https://doi.org/10.1109/cvpr.2017.195.

Dai, Jifeng, Yi Li, Kaiming He, and Jian Sun. 2016. "R-FCN: Object detection via region-based fully convolutional networks." In *Proceedings of the Advances in Neural Information Processing Systems*, 379-87.

Dimitrov, Andrey, and Mani Golparvar-Fard. 2014. "Vision-based material recognition for automated monitoring of construction progress and generating building information modeling from unordered site image collections", *Advanced Engineering Informatics*, 28(1): 37-49. DOI: https://doi.org/10.1016/j.aei.2013.11.002.

Ding, Lieyun, Weili Fang, Hanbin Luo, Peter ED Love, Botao Zhong, and Xi Ouyang. 2018. "A deep hybrid learning model to detect unsafe behavior: integrating convolution neural networks and long short-term memory", *Automation in Construction*, 86: 118-24. DOI: https://doi.org/10.1016/j.autcon.2017.11.002.

Ekenel, Hazım Kemal, and Rainer Stiefelhagen. 2009. "Why is facial occlusion a challenging problem?" In *Proceedings of the International Conference on Biometrics*, 299-308. Springer. DOI: https://doi.org/10.1007/978-3-642-01793-3_31.

England, Highways. 2016. "Health and safety for major road schemes: Safety helmet colours", Accessed 24 July, 2021.

https://www.gov.uk/government/publications/health-and-safety-for-major-road-schemes-safety-helmet-colours.

Fang, Qi, Heng Li, Xiaochun Luo, Lieyun Ding, Hanbin Luo, Timothy M Rose, and Wangpeng An. 2018. "Detecting non-hardhat-use by a deep learning method from far-field surveillance videos", *Automation in Construction*, 85: 1-9. DOI: https://doi.org/10.1016/j.autcon.2017.09.018.

Finn, Chelsea, Tianhe Yu, Tianhao Zhang, Pieter Abbeel, and Sergey Levine. 2017. "One-shot visual imitation learning via meta-learning." In *Proceedings of the Conference on Robot Learning*, 357-68. PMLR.

Freeman, William T, Thouis R Jones, and Egon C Pasztor. 2002. "Example-based super-resolution", *IEEE Computer graphics and Applications*, 22(2): 56-65. DOI: https://doi.org/10.1109/38.988747.

Freund, Yoav, and Robert E Schapire. 1996. "Game theory, on-line prediction and boosting." In *Proceedings of the Ninth Annual Conference on Computational Learning Theory*, 325-32. DOI: https://doi.org/10.1145/238061.238163.

Friedman, Jerome, Trevor Hastie, and Robert Tibshirani. 2001. *The elements of statistical learning* (Springer: New York City, NY). DOI: https://doi.org/10.1007/b94608.

Gandhi, Prashant, Somesh Khanna, and Sree Ramaswamy. 2016. "Which industries are the most digital (and why)", *Harvard Business Review*, 1: 45-48.

Géron, Aurélien. 2019. *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems* (O'Reilly Media: Sebastopol, CA).

Girshick, Ross. 2015. "Fast R-CNN." In *Proceedings of the IEEE International Conference on Computer Vision*, 1440-48. DOI: https://doi.org/10.1109/iccv.2015.169.

Girshick, Ross, Jeff Donahue, Trevor Darrell, and Jitendra Malik. 2014. "Rich feature hierarchies for accurate object detection and semantic segmentation." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 580-87. DOI: http://dx.doi.org/10.1109/cvpr.2014.81.

Goodfellow, Ian, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. "Generative adversarial networks", In *Proceedings of the Advances in Neural Information Processing Systems*, 2672-80.

Haarnoja, Tuomas, Sehoon Ha, Aurick Zhou, Jie Tan, George Tucker, and Sergey Levine. 2018. "Learning to walk via deep reinforcement learning", *arXiv preprint arXiv:1812.11103*.

Han, Kevin K, and Mani Golparvar-Fard. 2015. "Appearance-based material classification for monitoring of operation-level construction progress using 4D BIM and site photologs", *Automation in Construction*, 53: 44-57. DOI: https://doi.org/10.1016/j.autcon.2015.02.007.

Han, SangUk, Madhav Achar, SangHyun Lee, and Feniosky Peña-Mora. 2013. "Empirical assessment of a RGB-D sensor on motion capture and action recognition for construction worker monitoring", *Visualization in Engineering*, 1: 6. DOI: https://doi.org/10.1186/2213-7459-1-6.

He, Kaiming, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. 2017. "Mask R-CNN." In *Proceedings of the IEEE International Conference on Computer Vision*, 2980-88. IEEE. DOI: https://doi.org/10.1109/iccv.2017.322.

He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. "Deep residual learning for image recognition." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 770-78. DOI: https://doi.org/10.1109/cvpr.2016.90.

Hinton, Geoffrey E, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. 2012. "Improving neural networks by preventing co-adaptation of feature detectors", *arXiv preprint arXiv:1207.0580.*

Hochreiter, Sepp, and Jürgen Schmidhuber. 1997. "Long short-term memory", *Neural Computation*, 9(8): 1735-80. DOI: https://doi.org/10.1162/neco.1997.9.8.1735.

Hoiem, Derek, Alexei A Efros, and Martial Hebert. 2011. "Recovering occlusion boundaries from an image", *International Journal of Computer Vision*, 91: 328-46. DOI: https://doi.org/10.1007/s11263-010-0400-4.

Kim, Hongjo, Hyoungkwan Kim, Yong Won Hong, and Hyeran Byun. 2018. "Detecting construction equipment using a region-based fully convolutional network and transfer learning", *Journal of Computing in Civil Engineering*, 32(2): 04017082. DOI: https://doi.org/10.1061/(asce)cp.1943-5487.0000731.

Kingma, Diederik P, and Jimmy Ba. 2014. "Adam: A method for stochastic optimization", *arXiv preprint arXiv:1412.6980.*

Kolar, Zdenek, Hainan Chen, and Xiaowei Luo. 2018. "Transfer learning and deep convolutional neural networks for safety guardrail detection in 2D images", *Automation in Construction*, 89: 58-70. DOI: https://doi.org/10.1016/j.autcon.2018.01.003.

Kosala, Raymond, and Hendrik Blockeel. 2000. “Web mining research: A survey”, *ACM SIGKDD Explorations Newsletter*, 2(1): 1-15. DOI: https://doi.org/10.1145/360402.360406.

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E Hinton. 2012. “Imagenet classification with deep convolutional neural networks.” In *Proceedings of the Advances in Neural Information Processing Systems*, 1097-105.

Kumar, Praveen, Ayush Singhal, Sanyam Mehta, and Ankush Mittal. 2016. “Real-time moving object detection algorithm on high-resolution videos using GPUs”, *Journal of Real-Time Image Processing*, 11: 93-109. DOI: https://doi.org/10.1007/s11554-012-0309-y.

Kyrkou, Christos, George Plastiras, Theocharis Theocharides, Stylianos I Venieris, and Christos-Savvas Bouganis. 2018. “DroNet: Efficient convolutional neural network detector for real-time UAV applications.” In *Proceedings of the Design, Automation & Test in Europe Conference & Exhibition*, 967-72. IEEE. DOI: https://doi.org/10.23919/date.2018.8342149.

LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton. 2015. “Deep learning”, *Nature*, 521: 436. DOI: https://doi.org/10.1038/nature14539.

LeCun, Yann, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. “Gradient-based learning applied to document recognition”, In *Proceedings of the IEEE*, 86: 2278-324. DOI: https://doi.org/10.1109/5.726791.

Ledig, Christian, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, and Zehan Wang. 2017. “Photo-realistic single image super-resolution using a generative adversarial network.” In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 4681-90. DOI: https://doi.org/10.1109/cvpr.2017.19.

Li, Li-Jia, Richard Socher, and Li Fei-Fei. 2009. “Towards total scene understanding: Classification, annotation and segmentation in an automatic framework.” In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2036-43. IEEE. DOI: https://doi.org/10.1109/cvpr.2009.5206718.

Li, Pei, Loreto Prieto, Domingo Mery, and Patrick Flynn. 2018. “Face recognition in low quality images: a survey”, *arXiv preprint arXiv:1805.11519*.

Liang, Xiaodan, Tairui Wang, Luona Yang, and Eric Xing. 2018. "CIRL: Controllable imitative reinforcement learning for vision-based self-driving." In *Proceedings of the European Conference on Computer Vision*, 584-99. DOI: https://doi.org/10.1007/978-3-030-01234-2_36.

Liu, Wei, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. 2016. "SSD: Single shot multibox detector." In *Proceedings of the European Conference on Computer Vision*, 21-37. Springer. DOI: https://doi.org/10.1007/978-3-319-46448-0_2.

Luo, Hao, Wei Jiang, Xuan Zhang, Xing Fan, Jingjing Qian, and Chi Zhang. 2019. "AlignedReID++: Dynamically matching local information for person re-identification", *Pattern Recognition*, 94: 53-61. DOI: https://doi.org/10.1016/j.patcog.2019.05.028.

Luo, Xiaochun, Heng Li, Dongping Cao, Fei Dai, J Seo, and S Lee. 2018. "Recognizing diverse construction activities in site images via relevance networks of construction-related objects detected by convolutional neural networks", *Journal of Computing in Civil Engineering*, 32(3): 04018012. DOI: https://doi.org/10.1061/(asce)cp.1943-5487.0000756.

Manyika, James, Susan Lund, Michael Chui, Jacques Bughin, Jonathan Woetzel, Parul Batra, Ryan Ko, and Saurabh Sanghvi. 2017. "Jobs lost, jobs gained: What the future of work will mean for jobs, skills, and wages." Accessed June 18, 2021. https://www.mckinsey.com/featured-insights/future-of-work/jobs-lost-jobs-gained-what-the-future-of-work-will-mean-for-jobs-skills-and-wages#.

Marinai, Simone, Marco Gori, and Giovanni Soda. 2005. "Artificial neural networks for document analysis and recognition", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(1): 23-35. DOI: https://doi.org/10.1109/tpami.2005.4.

Mieritz, Lars. 2012. "Gartner Survey Shows Why Projects Fail". Accessed June 18, 2021. https://thisiswhatgoodlookslike.com/2012/06/10/gartner-survey-shows-why-projects-fail.

Mittal, Anish, Anush Krishna Moorthy, and Alan Conrad Bovik. 2012. "No-reference image quality assessment in the spatial domain", *IEEE Transactions on Image Processing*, 21(12): 4695-708. DOI: https://doi.org/10.1109/tip.2012.2214050.

Mneymneh, Bahaa Eddine, Mohamad Abbas, and Hiam Khoury. 2018. "Vision-based framework for intelligent monitoring of hardhat wearing on construction sites", *Journal of Computing in Civil Engineering*, 33(2): 04018066. DOI: https://doi.org/10.1061/(asce)cp.1943-5487.0000813.

Murphy, Kevin P. 2012. *Machine learning: A probabilistic perspective* (MIT press: Cambridge, MA).

Nath, Nipun D, Theodora Chaspari, and Amir H Behzadan. 2019. "Single-and multi-label classification of construction objects using deep transfer learning methods", *Journal of Information Technology in Construction*, 24: 511-26. DOI: https://doi.org/10.36680/j.itcon.2019.028.

Nath, Nipun D, and Amir H Behzadan. 2020. "Deep convolutional networks for construction object detection under different visual conditions", *Frontier's in Built Environment*, 6: 97. DOI: https://doi.org/10.3389/fbuil.2020.00097.

Nath, Nipun D, Amir H Behzadan, and Stephanie G Paal. 2020. "Deep learning for site safety: Real-time detection of personal protective equipment", *Automation in Construction*, 112: 103085. DOI: https://doi.org/10.1016/j.autcon.2020.103085.

Nath, Nipun D, Theodora Chaspari, and Amir H Behzadan. 2018. "Automated ergonomic risk monitoring using body-mounted sensors and machine learning", *Advanced Engineering Informatics*, 38: 514-26. https://doi.org/10.1016/j.aei.2018.08.020.

National Academies. 2017. "United States' Skilled Technical Workforce Is Inadequate to Compete in Coming Decades; Actions Needed to Improve Education, Training, and Lifelong Learning of Workers". Accessed June 18, 2021. https://www.nationalacademies.org/news/2017/05/united-states-skilled-technical-workforce-is-inadequate-to-compete-in-coming-decades-actions-needed-to-improve-education-training-and-lifelong-learning-of-workers.

NSF. 2020. "Future of work at the human-technology frontier", Accessed January 19, 2021. https://www.nsf.gov/news/special_reports/big_ideas/human_tech.jsp.

Oquab, Maxime, Leon Bottou, Ivan Laptev, and Josef Sivic. 2014. "Learning and transferring mid-level image representations using convolutional neural networks." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1717-24. DOI: http://dx.doi.org/10.1109/cvpr.2014.222.

OSHA. 2019. "Commonly used statistics", Accessed January 20, 2021. https://www.osha.gov/oshstats/commonstats.html.

Pan, Xinlei, Yurong You, Ziyan Wang, and Cewu Lu. 2017. "Virtual to real reinforcement learning for autonomous driving", *arXiv preprint arXiv:1704.03952*.

Park, Man-Woo, Nehad Elsafty, and Zhenhua Zhu. 2015. "Hardhat-wearing detection for enhancing on-site safety of construction workers", *Journal of Construction*

Engineering and Management, 141(9): 04015024. DOI: https://doi.org/10.1061/(asce)co.1943-7862.0000974.

Peltier, James W, Yushan Zhao, and John A Schibrowsky. 2012. "Technology adoption by small businesses: An exploratory study of the interrelationships of owner and environmental factors", *International Small Business Journal*, 30(4): 406-31. DOI: https://doi.org/10.1177/0266242610365512.

Perez, Luis, and Jason Wang. 2017. "The effectiveness of data augmentation in image classification using deep learning", *arXiv preprint arXiv:1712.04621*.

Redmon, Joseph, Santosh Divvala, Ross Girshick, and Ali Farhadi. 2016. "You only look once: Unified, real-time object detection." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 779-88. IEEE. DOI: http://dx.doi.org/10.1109/cvpr.2016.91.

Redmon, Joseph, and Ali Farhadi. 2017. "YOLO9000: better, faster, stronger", *arXiv preprint arXiv: 1612.08242*.

Redmon, Joseph, and Ali Farhadi. 2018. "YOLOv3: An incremental improvement", *arXiv preprint arXiv:1804.02767*.

Ren, Shaoqing, Kaiming He, Ross Girshick, and Jian Sun. 2015. "Faster R-CNN: Towards real-time object detection with region proposal networks", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6): 1137-49. DOI: https://doi.org/10.1109/tpami.2016.2577031.

Shin, Hoo-Chang, Holger R Roth, Mingchen Gao, Le Lu, Ziyue Xu, Isabella Nogues, Jianhua Yao, Daniel Mollura, and Ronald M Summers. 2016. "Deep convolutional neural networks for computer-aided detection: CNN architectures, dataset characteristics and transfer learning", *IEEE Transactions on Medical Imaging*, 35(5): 1285-98. DOI: https://doi.org/10.1109/tmi.2016.2528162.

Siau, Keng, and Weiyu Wang. 2018. "Building trust in artificial intelligence, machine learning, and robotics", *Cutter Business Technology Journal*, 31: 47-53.

Siddula, Madhuri, Fei Dai, Yanfang Ye, and Jianping Fan. 2016. "Unsupervised feature learning for objects of interest detection in cluttered construction roof site images", *Procedia Engineering*, 145: 428-35. DOI: https://doi.org/10.1016/j.proeng.2016.04.010.

Silver, David, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, and Marc Lanctot. 2016. "Mastering the game of Go with deep neural networks and tree search", *Nature*, 529: 484-89. DOI: https://doi.org/10.1038/nature16961.

Silver, David, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, and Adrian Bolton. 2017. "Mastering the game of go without human knowledge", *Nature*, 550: 354-59. DOI: https://doi.org/10.1038/nature24270.

Simonyan, Karen, and Andrew Zisserman. 2014. "Very deep convolutional networks for large-scale image recognition", *arXiv preprint arXiv:1409.1556*.

Smartvid.io Inc. 2017. "smartvid.io". Accessed January 20, 2021. https://www.smartvid.io/.

Son, Hyojoo, Changmin Kim, Nahyae Hwang, Changwan Kim, and Youngcheol Kang. 2014. "Classification of major construction materials in construction environments using ensemble classifiers", *Advanced Engineering Informatics*, 28(1): 1-10. DOI: https://doi.org/10.1016/j.aei.2013.10.001.

Sutton, Richard S, and Andrew G Barto. 2018. *Reinforcement learning: An introduction* (MIT press: Cambridge, MA). DOI: https://doi.org/10.1109/tnn.1998.712192.

Szeliski, Richard. 2019. "Computer vision: algorithms and applications", *Instructor*, 201901.

Szepesvári, Csaba. 2010. *Algorithms for reinforcement learning* (Morgan & Claypool Publishers: San Rafael, CA). DOI: https://doi.org/10.2200/s00268ed1v01y201005aim009

Tieleman, Tijmen, and Geoffrey Hinton. 2012. "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude", *Coursera: Neural networks for machine learning*, 4: 26-31.

Trinh, Dinh-Hoan, Marie Luong, Francoise Dibos, Jean-Marie Rocchisani, Canh-Duong Pham, and Truong Q Nguyen. 2014. "Novel example-based method for super-resolution and denoising of medical images", *IEEE Transactions on Image Processing*, 23(4): 1882-95. DOI: https://doi.org/10.1109/tip.2014.2308422.

Van Hasselt, Hado, Arthur Guez, and David Silver. 2015. "Deep reinforcement learning with double q-learning", *arXiv preprint arXiv:1509.06461*.

Van Seijen, Harm, Mehdi Fatemi, Joshua Romoff, Romain Laroche, Tavian Barnes, and Jeffrey Tsang. 2017. "Hybrid reward architecture for reinforcement learning." In *Proceedings of the Advances in Neural Information Processing Systems*, 5392-402.

Wu, Jixiu, Nian Cai, Wenjie Chen, Huiheng Wang, and Guotian Wang. 2019. "Automatic detection of hardhats worn by construction personnel: A deep learning approach and benchmark dataset", *Automation in Construction*, 106: 102894. DOI: https://doi.org/10.1016/j.autcon.2019.102894.

Wu, Yue, Yicong Zhou, George Saveriades, Sos Agaian, Joseph P Noonan, and Premkumar Natarajan. 2013. "Local Shannon entropy measure with statistical tests for image randomness", *Information Sciences*, 222: 323-42. DOI: https://doi.org/10.1016/j.ins.2012.07.049.

Xie, Zaipeng, Hanxiang Liu, Zewen Li, and Yuechao He. 2018. "A convolutional neural network based approach towards real-time hard hat detection." In *Proceedings of the IEEE International Conference on Progress in Informatics and Computing*, 430-34. IEEE. DOI: https://doi.org/10.1109/pic.2018.8706269.

Yang, Yuxiang, Ken Caluwaerts, Atil Iscen, Tingnan Zhang, Jie Tan, and Vikas Sindhwani. 2020. "Data efficient reinforcement learning for legged robots." In *Proceedings of the Conference on Robot Learning*, 1-10. PMLR.

Yuen, Man-Ching, Irwin King, and Kwong-Sak Leung. 2011. "A survey of crowdsourcing systems." In *Proceedings of the IEEE Third International Conference on Privacy, Security, Risk and Trust and IEEE Third International Conference on Social Computing*, 766-73. IEEE. DOI: https://doi.org/10.1109/passat/socialcom.2011.203.

Zhang, Xuan, Hao Luo, Xing Fan, Weilai Xiang, Yixiao Sun, Qiqi Xiao, Wei Jiang, Chi Zhang, and Jian Sun. 2017. "AlignedReID: Surpassing human-level performance in person re-identification", *arXiv preprint arXiv:1711.08184*.

Zhang, Yongbing, Debin Zhao, Jian Zhang, Ruiqin Xiong, and Wen Gao. 2011. "Interpolation-dependent image downsampling", *IEEE Transactions on Image Processing*, 20(11): 3291-96. DOI: https://doi.org/10.1109/tip.2011.2158226.

Zhao, Yang, Zicheng Liu, Lu Yang, and Hong Cheng. 2012. "Combing RGB and depth map features for human activity recognition." In *Proceedings of The 2012 Asia Pacific Signal and Information Processing Association Annual Summit and Conference*, 1-4. IEEE.

Zivkovic, Zoran. 2004. "Improved adaptive Gaussian mixture model for background subtraction." In *Proceedings of the 17th International Conference on Pattern Recognition*, 28-31. DOI: https://doi.org/10.1109/icpr.2004.1333992.

Zou, Junhao, and Hyoungkwan Kim. 2007. "Using hue, saturation, and value color space for hydraulic excavator idle time analysis", *Journal of computing in Civil Engineering*, 21(4): 238-46. DOI: https://doi.org/10.1061/(asce)0887-3801(2007)21:4(238).