Analysis of the Performance of HOG and CNNs for Detecting Construction Equipment and Personal Protective Equipment

Seyedeh Forough Karandish

A Thesis

in

The Department

of

Concordia Institute for Information Systems Engineering

Presented in Fulfillment of the Requirements

For the Degree of Master of Applied Science (Quality Systems Engineering) at

Concordia University

Montreal, Quebec, Canada

April 2019

CONCORDIA UNIVERSITY

School of Graduate Studies

This is to certify that the thesis prepared

By: Seyedeh Forough Karandish

Entitled: Analysis of the Performance of HOG and CNNs for Detecting Construction Equipment and Personal Protective Equipment

and submitted in partial fulfillment of the requirements for the degree of

Master of Applied Science (Quality Systems Engineering)

complies with the regulations of the University and meets the accepted standards with respect to originality and quality.

Signed by the final Examining committee:

Dr. Farnoosh Naderkhani Chair

Dr. Mazdak Nikbakht Examiner

Dr. Jia Yuan Yu Examiner

Dr. Amin Hammad Supervisor

Approved by _____

Chair of Departmental or Graduate Program Director

_____ _____

Dean of Faculty

ABSTRACT

Analysis of the Performance of HOG and CNNs for Detecting Construction Equipment and Personal Protective Equipment

Seyedeh Forough Karandish

Concordia University, 2019

The construction industry remains one of the most dangerous working environments in terms of fatalities and accidents. High numbers of accidents and loss-time injuries, leads to a decrease in productivity in this industry. Therefore, new technologies are being developed to improve the safety of construction sites. Object detection on construction sites has a huge impact on the construction industry. Many researchers studied productivity, safety and project progress. However, few efforts have been made to improve the robustness of the related datasets for detection purposes. In the meantime, it is noticed that the lack of a custom dataset leads to low accuracy and also an increase in the cost and time of training dataset preparation.

In this research, we first investigated the generation of synthetic images using 3D models of construction equipment to use them as the datasets for training purposes, namely: excavators, loaders and trucks, and then sensitivity analysis is applied. We compared the performance of CNNs and other conventional methods for classifying construction equipment. In the second part, the detection of personal protective equipment for construction workers was studied. For this purpose, several object detection architectures from the TensorFlow object detection model zoo have been evaluated to find the best and most robust detection model. The dataset used in this study contains real images from construction sites. The performance evaluation of trained object detectors are measured in terms of mean average precision. The test results from this study showed that (1) synthetic images have a significant effect on the final detection results; and (2) comparing various object detection architectures, Faster_rcnn_resnet101 was the most suitable model in terms of accuracy of detection.

ACKNOWLEDGEMENT

My greatest appreciation goes to my supervisor, Dr. Amin Hammad for his intellectual and personal support, encouragement and patience. His guidance, advice and criticism was my most valuable asset during my studies. Overall, I feel very fortunate having the opportunity to know him and work with him.

I would like to acknowledge the contributions of Dr. Zhenhua Zhu for his generous guidance and support during my internship.

I would, also, like to thank Mr. Matthew Man and Dr. Walid Ahmed for providing me the opportunity to work on a real project and be part of their Machine Learning team.

Finally, I must express my very profound gratitude to my parents for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them.

This study is wholeheartedly dedicated to my beloved parents, Farahnaz Kavoosfar and Seyed Hossein Karandish for their unlimited love and unwavering support.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

Abbreviation	Descriptor
2D	Two Dimensional
3D	Three Dimensional
AI	Artificial Intelligence
ANN	Artificial Neural Network
APPC	Automated Project Performance Control
ATR	Automated Target Recognition
AWCBC	Association of Workers Compensation Boards of Canada
AWS	Amazon Web Service
CNN	Convolutional Neural Network
COCO	Common Object in Contact
CPU	Central Processing Unit
CSV	Comma Separated Values
CV	Computer Vision
DNN	Deep Neural Network
FPS	Frames per Second
GDP	Gross Domestic Product
GPS	Global Position Systems
GPU	Graphical Processing Unit
HASARD	Hazardous Area Signaling and Ranging Device
HOG	Histogram of Oriented Gradients
HSV	Hue, Saturation, Value

NIOSH	National Institute for Occupational Safety and Health
NN	Neural Network
PCA	Principle Component Analysis
PPE	Personal Protective Equipment
PPI	Project Performance Indicators
RCNN	Regional-based Convolutional Neural Network
RELU	Rectified Linear Unit
RESNET	Residual Network
RF	Radio Frequency
RFCN	Region-based Fully Convolutional Network
RFID	Radio-Frequency Identifier
RGB	Red Green Blue
RPN	Region Proposal Network
RTS	Robotic Total Station
SIFT	Scale-Invariant Feature Transforms
SIG	Synthetic Image Generation
SSD	Single Shot Detector
SVM	Support Vector Machine
Tanh	Hyperbolic TangentTensorFlow
TF	TensorFlow
TFRecord	TensorFlow Record
WBC	Worker Compensation Band
XML	Extensible Markup Language

CHAPTER 1 INTRODUCTION

1.1 General Information

The construction industry remains one of the most dangerous working environments in terms of fatalities and accidents. According to the United States department of labor, the fatal injury rate for the construction industry is significant and requires special considerations (Osha Statistics, 2006). The report highlights the fact that 252,000 construction sites across the country, with approximately 6.5 million workers, have higher fatal accidents than the national average of all industries. The immediate tragic consequences of such accident on construction sites are the increase in costs and the decrease in progress in construction projects. The significance of the special considerations is thus evident. In order to tackle this crucial problem, it is important to identify the main causes of the problem in a sound and thorough investigation, and then use state-of-the-art technology to remedy the origin.

A large number of studies have been conducted to identify the roots causes of construction accidents (Sawacha et al. 1999). The main focus of these construction accident investigations is to identify what type of accidents occur and how they occurred. The empirical factors influencing unsafe behaviors and accidents reported in the content of the research are listed as the absence of knowledge or training, the lack of supervision leading which leads to the lack of task safety, error of judgement, negligence and apathy are just some factors affecting construction safety performance. In addition, most accidents may be associated with some type of carelessness and may include unsafe working conditions, inappropriate use of tools and/or equipment and lack of personal protective equipment (Sawacha et al. 1999). However, in order to have a more succinct classification of the original causes, they can be summarized in three main categories (Abdelhamid and Everett, 2000). Three root causes of accidents fail to identify an unsafe condition, decide to work after the worker identifies an existing unsafe condition, and decide to act unsafe regardless of the initial working conditions. Since decision - making on the site is outside the scope of this research, the main focus of this research is the first root cause, which is the identification of the potential existing unsafe condition.

This research aims to detect the potential existing hazards on construction sites using state-of-the-art technology. In order to achieve this, we have used an automated detection mechanism based on Computer Vision (CV) technology to identify potential hazards on construction sites. To achieve this objective, the research is divided into the following steps:

(1) Collecting, preparing and cleaning the input data, this step also requires labeling to be fed into the model; (2) Developing Deep Learning-based models to implement object detection algorithms; (3) Checking the output of the implemented models and the accuracy of the results, followed by improvi ng the model to increase the model's accuracy.

1.2 Research Objectives

This research aims to achieve the following objectives: (1) generating synthetic images as the input data; (2) Analyzing the performance of equipment detection on construction sites using synthetic and real images; (3) improving the safety of workers by detecting their PPEs on construction sites.

1.3 Thesis Organization

The thesis structured as follows:

Chapter 1 *Introduction*: this chapter introduces the research topic and objectives and presents the structure of the thesis.

Chapter 2 *Literature Review*: this chapter reviews the existing literature on the impact of CV-based methods on safety of construction sites and providing input data.

Chapter 3 *Proposed Methodology and implementation for comparing the performance of CNN and Other Conventional Methods for classifying equipment on construction sites using synthetic images*: this chapter proposes the generation of synthetic images as input data. The factors involved in the sensitivity analysis are organized. In addition, different methods are used for the detection of various objects on construction sites. The evaluation and performance of object detection methods are tested to ensure that detectors are accurate on a real construction project.

Chapter 4 *Comparison of the performance of TensorFlow object detection models for the detection of personal protective equipment on construction sites*: This chapter consists of the methodology and implementation of the proposed strategy, which is presented through case studies and the analysis of the collected data.

Chapter 5 *Conclusions and Future Work*: this chapter summarizes the present work and concludes the findings. This chapter also includes the limitations and future work.

CHAPTER 2 Literature Review

2.1 Introduction

The present chapter focuses on reviewing the recent deep learning algorithms in the field of Computer Vision. First the problem statement is clarified and then the methodologies used in the literatures are investigated. We will review how Two Dimensional (2D) images can affect the visual detection through computers. Afterwards, the existing research related to object detection and classification on construction sites in terms of safety monitoring is comprehensively investigated. Lastly, the limitations and concerns of the existing methods are summarized at the end of this chapter.

2.2 Safety Status and Monitoring in Construction Projects

By analyzing statistical information of workplace injuries, diseases, and fatalities in the construction industry of Canada, it can be clearly seen that construction sites have the highest rate of fatality among all other industries. Statistics carried out by the Association Of Workers Compensation Boards Of Canada (AWCBC), shows that among all existing industries in Canada, the construction industry has the highest rate in total number of fatalities. Figure 2-1 shows the numbers of fatalities in all industries in 2016.

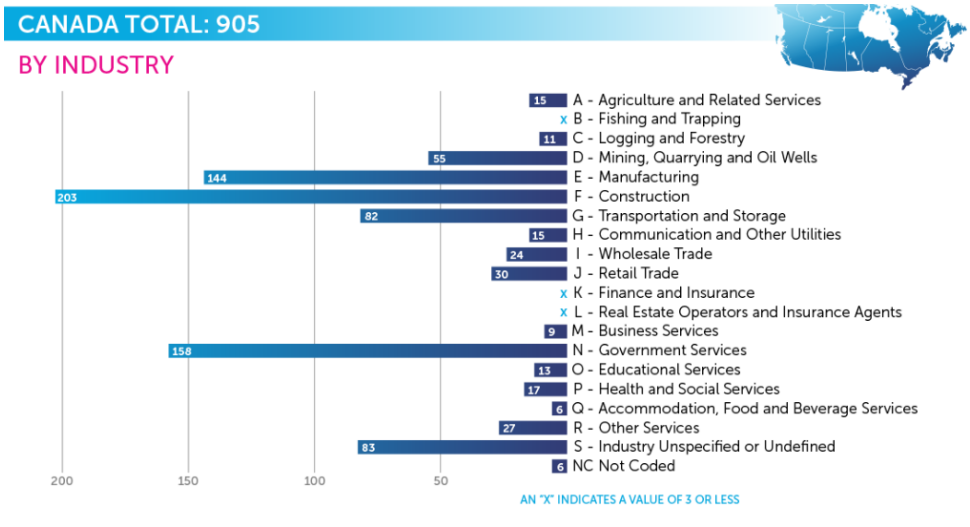


Figure 2-1 2016 Lost time claims in Canada (AWCBC Statistics, 2016)

In terms of the provincial and territorial classification of Canada, the province of Alberta had the highest rate of fatalities, whereas Ontario, British Columbia, and Quebec had the maximum number of fatalities in the construction industry after Alberta (AWCBC Statistics). Furthermore, the number of fatality claims admitted by Workers Compensation Board of Alberta (WCB) raised by 15.2% in

2016, from 125 in 2015 to 144 in 2016. More than one third of fatalities and injuries were reported by construction and construction trade services.

2.3 Potential Construction Hazards

There had been researches investigating a variety of hazards leading to fatalities and injuries in construction sites. Table 2-1 demonstrates all types of hazards that can occur on construction sites. Accordingly, three main causes of construction accidents at work sites are due to the dynamic nature of the work environment on construction sites, the presence of different contractors in the same field, as well as the presence of heavy machinery on construction sites.

1) Construction sites are highly dynamic and changing constantly: by increasing the complexity of modern construction projects, there is a higher need of using computers in order to accomplish effective planning and management. The computing power of computers can be used to detect and automate the danger detection process. Since the environment is changing constantly due to dynamic work environment, there is a need for machine learning algorithms combined with Artificial Intelligence (AI) in order to detect the objects in the construction site.

2) Several small contractors may be close to each other and execute different types of work. This implies that, there should be a central unit in the construction site to observe the movement of each group of workers and machinery at the same time. Otherwise, managing different groups of labors and machinery on the same construction environment, working on different projects may cause fatalities as well as serious injuries.

3) The positions and poses of equipments and machineries are frequently changed on construction sites. It is important to monitor both the position and poses of the equipments and machineries at every point in time. The position and dynamic nature of construction sites were discussed in previous research, however the pose of the machinery is also important. Therefore, in order to reduce fatalities and injuries on construction sites, which in turn increases the trust and moral of the workers, priority should be given to the detection of the placement and positioning of machineries. Nevertheless, new hazards are constantly emerging on construction sites (Reese *and Edison*, 2006).

Table 2-1 Potential construction hazards (Reese and Edison, 2006)

Premature explosions	Rollover
Hand/ arm vibration	Gases
Concrete handling	Noise
Compressed air equipment	Cave-in
Working with sharp objects	Burns
Working without guards	Mist
Carrying heavy materials	Fumes
Wet/ slippery surface	Electrocution

2.4 Existing Safety Practices and Involvement of Technology

Several studies have been dedicated to developing consistent alert systems to use location information for safety improvement for construction work zones. Schiffbauer, (2002) Proposed alert systems using RFID or millimeter wave technologies in order to avoid contact between vehicles and workers. However, RFID range is limited to 10 to 20 meters. Moreover, RF-based methods, such as cellular-based systems and sensor-based detection have been tested and used for exchanging the location of vehicles via wireless communication. These methods can be useful in small-scale mining sites, but it is unpractical to cover the large mining sites, which (Sun et al. 2009; Ni et al. 2014).

Another research investigated by Teizer, (2008) determined that it is possible to improve construction safety through alerting workers on-foot and/or equipment operators in real-time when they are too-close to an unidentified or other construction recourses by using of remote sensing and proactive technologies such as Radio Frequency (RF), Ultra-Wideband (UWB), and imaging technologies. The results of the research have shown very promising safety improvement in outdoor construction environment. Moreover, the real-time pro-active RF warning and the alert technology was used to be effective in aiding the safety needs in the construction environment. Accordingly, several blind spots of heavy equipment, such as trucks, excavators, graders were selected, and 3D laser scanner was used to measure and take their blind spots. Then, testing was done along with a warning system under ideal conditions. When the warning system was activated, the distances among the worker and the equipment would be measured by the Robotic Total Station (RTS). This shows the least distances expected to bring construction equipment to a fully safe stop condition before accidents occur with the workers, crew, or other objects.

Similarly, Zhang et al. (2009) suggested a multi-agent system to identify potential accidents or conflicts related to movements of equipment on construction sites. Yards (2008) has applied active RFID tags to monitor workers as they embark and disembark along the entry bridges to various ships. Also, RF-based tags kept significant data about the workers such as pictures and allowed to count workers or search and save their personnel life at the time of emergency.

Another research done by Teizer et al. (2010) depicted two heavy construction equipment and a worker in between them. A commercial Three-Dimensional (3D) laser scanner was used to take a dense range point cloud with overlaid true color values. Blind spots to an articulated dump truck can then be measured using an automated blind spot measurement device. Figure 2-2 illustrates an example of a working crew placed in blind spot of a heavy machinery as an example of a possible hazardous situation.

Figure 2-2 Hazardous situation of worker and equipment measured using a 3D laser scanner (Teizer et al. 2010)

2.5 Productivity Monitoring on Construction Sites

The construction industry has a substantial role in the economic development of a country. It is also illustrated that the construction industry produces about six to nine percent of the Gross Domestic Product (GDP) of industrialized countries. Similarly, this industry creates more than half of the fixed capital formation as infrastructure and public utilities capital works needed for economic development (Chitkara, 1998). Hence, the performance of the construction industry has an important influence on the economy. Therefore, research projects had been placed to measure the performance of this key factor on GDP.

Navon and Shpatnitsky (2005) investigated automated monitoring of road construction by GPS to measure the development of a real-time productivity and progress of construction project. The research proposed automated project performance control of construction projects. Based on the project management's opinion, a project is a success when all the project performance indicators (PPI) are met. As an example, the lowest possible expense with the maximum quality, material consumption, and labor productivity without any accidents, can be achieved as quickly as possible. Consequently, a control system is vital for project managers.

Automated Project Performance Control (APPC) and the traditional method of collecting data are two proposed methods to enhance the productivity of construction. In the first method, measuring devices are used indirectly to evaluate given data such as analog thermometers and the GPS. In the traditional approach, data is collected manually, which is inaccurate and time-consuming. These two methods were compared from the productivity point of view. A related method to estimate the productivity of construction sites, which applies GPS, was proposed by Li et al. (2005). Development of human pose analyzing algorithms for the determination of construction productivity in real-time was suggested by Peddi et al. (2009). The key point of this research is to classify construction worker poses into three categories, namely, effective work, ineffective work, and contributory work. This method was trained by neural network classifications to determine the worker's status. Then, the results of this method are compared with incoming images to compare it with the manual method. The integration of construction-process simulation and vision-based methods has shown promising results for productivity analysis of the earthmoving process in a tunnel. Hence, convolutional neural networks (CNNs) used for detection purposes of construction equipment resulted in a mean average precision of 99.09% and a context error of 1.6% in the detection of an earthmoving equipment. The context of information was used as an input dataset for process simulation. Productivity estimation, cost analysis, resource allocation plan, and work schedules were the results of the study of Kim et al. (2018) as shown in Figure 2-3.

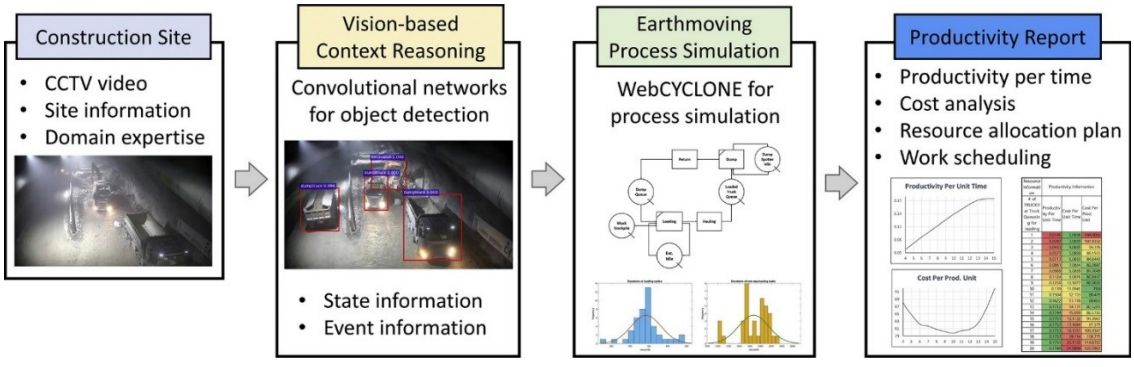


Figure 2-3 Graphical process of CV-based construction productivity (Kim et al. 2018)

The importance of a visual understanding of productivity measurement is clear from the discussion. This research focuses more on this aspect of the on-going attempts toward the effective and optimized workplaces in construction fields; however, the focus of visual recognition in this thesis is on safety. To reach this goal, this research aims at understanding and utilizing computer vision methods.

2.6 Computer Vision Techniques

Researchers in computer vision use mathematical techniques to discover the 3D appearance and structure of the objects through images. The goal of computer vision is to establish methods that allow computers to see and understand images, analyze them, and finally reach machine vision just like the human vision. What differentiates computer vision from the current field of digital image processing is a need to extract three-dimensional structures from images with the objective of achieving full scene of understanding and analysis. Several attempts were done by researchers in the 1970s, such as, motion estimation, optical flow, non-polyhedral and polyhedral modeling, extraction of edges from images, and labeling of lines (Szeliski R., 2011). However, afterwards, researchers developed more sophisticated mathematical analysis and quantitative approach in the field of computer vision. These models include intensities and shading variations, texture and focus, the concept of scale-space processing and contour models known as snakes (Yin et al. 2012). Furthermore, the interaction with computer graphics increased dramatically in the image-based rendering, panoramic image stitching and light-field rendering (Szeliski (2010)). The final movement, which now leads a lot of visual recognition, is the application of machine learning to computer vision. Also, computer vision occasionally is seen as a part of artificial intelligence which shares other subjects such as pattern recognition and learning techniques. Artificial intelligence takes advantage of Neural Networks, which is implemented and modeled based on the human brain. The model simulates the brain and uses techniques such as deep learning. Deep learning is a process where the machine can learn, and the

process is based on a set of deep neural networks working in a sequence. Hence, the complexity of the models pays off with the high accuracy of the result. This makes computer vision a very applicable and sophisticated model. Basically, topics that can be covered in the application of Computer vision are scene reconstruction, event detection, video tracking, object recognition, object pose estimation, learning, indexing, motion estimation, and image restoration. In the following sections, the achievement of computer vision and related studies are explained.

2.6.1 Image Processing

Image improvement is basically processing images to increase their usefulness. The technologies of image processing could be divided into two categories: Image Enhancement and Image Restoration. These two technologies are closely related to each other. The procedure of image enhancement is the procedure of improving the content of information in the images for human viewers and prepare "superior" input for other automated image processing techniques. Hence, this method uses a certain performance index to measure the quality of images, and subsequently enhance the quality of the images. The principal goal of image enhancement is to change the characteristics of an image to make it more applicable for a given task (Maini, 2010). Digital image restoration is the process of taking noisy images and estimate the clean, and original image. The main concept of this method is to reduce noise and recover resolution loss to restore a better image based on known degeneration models and make it more comprehensible for computers (Katsaggelos, 2012).

2.6.2 Object Recognition

Object recognition is one of the substantial challenges in the field of computer vision. However, a drawback of this method is detecting and localizing objects from different classes such as people or cars in the static images. The enormous differences in the appearance of objects in the same category, cause difficulty in detection. Even in the same category the diversity in shape and other visual properties could have a huge impact on the detection, such as different poses of people or different types and colors of cars (Felzenszwalb et al. 2010). Furthermore, it is a challenge to find all the objects in the given image and assigning the right label to the object from a set of given labels. The labeling problem based on models of known objects can be discussed. Usually, an image containing one or several objects of interest based on a set of labels related to a set of models that are identified to the system. Each label should be assigned to zones, or a set of zones, in the image. Each zone contains the detected object. The challenge might seem trivial for a human being, however, to teach a computer to

overcome such a challenge is not as simple. Besides image recognition, there is the segmentation problem. Segmentation problem has a very close relation to the object recognition problem. The problem is to recognize at least a part of the object in order to complete the segmentation, and without segmentation, object recognition is impossible. Since image recognition requires to achieve the complete picture after segmentation problem is done.

Many advanced methods have been introduced for the object recognition problem. Schmid and Mohr (1997) used the Harris corner detector to find interest spot, and then generate a local image at each descriptor at each interest spot from an orientation-invariant vector of derivative-of-Gaussian image measurement. This method is used for robust object recognition by considering various matching descriptors that consists of object-based orientation and location constraints in order to detect the object.

Another method proposed by Felzenszwalb and Huttenlocher (2005) discussed the pictorial structure models, which focuses on a matching problem, where each part has a separate match cost in a set of the neighborhood. In this model, the geometric arrangement was taken by "springs" that are linking pairs of parts together, which allows for qualitative definition of appearance and it is useful for generic recognition problems.

In addition, we can categorize the object recognition challenges into 2D and 3D sub-challenges. If the object can be seen always in one stable position, then it can be two-dimensional. There are two possible scenarios in this case which are: 1) The object will not be occluded, like in remote sensing and industrial applications, 2) Objects may be occluded by other objects of interest or be partially visible, as in bin of part problems. If the images of objects can be obtained from arbitrary viewpoints, then an object may appear very different in its 2D views.

The perspective effect and viewpoint of the image are two important factors in object recognition which have to be considered very carefully. Additionally, another option that can affect object recognition is that the models are 3D and the images have only 2D information. Similarly, it is very important to consider that objects may be separated from other objects (Jain et al. 1995).

Based on research of Kanade (2000), multiple detectors can be used for object detection. Therefore, to deal with variation in a pose in any category or class, view-based approach can be used with multiple

detectors, suggesting that each detector has a specialized angle of the object. This model is commonly accepted for various shapes and positions of cars and faces.

In order to decrease the errors in the object recognition, model parameters have to be selected for training. This strategy has a huge improvement on the boundary of negative and positive images (Dalal and Triggs, 2005).

2.7 Model-Based Synthetic Images

The demand of having a large imagery dataset for developing and testing an algorithm leads to the idea of synthetic images as a complementarity of real images. The idea behind synthetic images is to generate new images from the current set of images. Since, the more the data, the more accurate the result is. In machine learning and AI, it is clear that synthetic images play a key role in data set generation.

The background of Synthetic Image Generation (SIG) modeling was several studies, such as sensor design, analyst training, mission rehearsal, algorithm development, and evaluation. One of the most important outcomes of SIG modeling tools has been the need for various sets of imagery for training of Automatic Target Recognition (ARTs) algorithms (Schott, 1999).

Shotton et al. (2013) asserted an algorithm for developing body part recognition as an average representation of human pose estimation from a single depth image. Therefore, very large amount of realistic synthetic images of humans of different shapes and sizes in highly varied poses form the datasets used as training dataset. By using computer graphics to synthesize a very large dataset of training image pairs, one can train a classifier that estimates body part labels from test images invariant to pose, body shape, clothing, and other irrelevances.

Soltani et al. (2016) suggested a synthetic image technique to track the pose of the equipment for the safety and productivity evaluation purposes. Hence, by creating a 3D model of equipment and using CV-Based object recognition algorithms, similar equipments in the construction site could be found. Additionally, this method has a huge impact on improving the accuracy of object recognition and reducing the training time.

The importance of color in the field of image processing has been always considered as a challenging task. Several researchers (Ruderman et al. 1998; Reinhard et al. 2001) investigated on Red Green Blue (RGB) channel to create synthetic images by taking one image with another that looks similar.

Another main problem in the field of image recognition stems from the measurement of optical flow. Hence, using synthetic and real images sequences suggested to evaluate different optical flow techniques. The core of this method is to calculate the 2-D motion field projection of the 3-D velocities of surface points onto the imaging surface from spatiotemporal pattern of image intensity (Barron et al. 1994).

Colored images and optical flow are not the only challenges researchers have encountered in the field of image recognition. The variance between diffuse and specular reflection causes a problem in many computervisiontasksThereisa necessity on segmentation to deliver limitation on the diffuse component of specular reflections. Hence, to overcome this limitations, color analysis and multi-baseline stereo suggest to evaluate the separation and the true depth of specular reflection simultaneously. Therefore, to have accurate separation and depth, this algorithm was implemented on synthetic and real image sequences, where synthetic images are used to evaluate the ground truth data (Li et al. 2002).

Image segmentation and grouping remains a challenging problems in the field of CV since the perceptual grouping has a significant role in human visual perception. Thus, Felzenszwalb and Huttenlocher, (2004) proposed a graph-based segmentation method with the use of local neighborhoods in constructing the graph and then implemented the results on both synthetic and real images.

2.8 Application of Computer Vision in Construction Projects

By reviewing several studies based on the CV methods in construction projects, three categories emerged as follows: (1) determining activity and productivity of project; (2) tracking resources in the construction site; and (3) monitoring the safety and health of workers in the construction area. The highlights and limitations of each category will be discussed in the following.

1) Application of CV for Monitoring the Productivity of Construction Sites

New automated methods for productivity estimation aims to detect the types, locations, and activities of construction equipment based on sensory data. CV is one of the most promising automated methods that provides an affordable opportunity for estimating the productivity of construction sites since it only requires regular surveillance cameras for data collection, which are available on many construction sites.

The availability of the surveillance cameras on construction sites opens the opportunity for applying CV-based methods to monitor the productivity of the equipment in addition to monitoring the safety and security of the sites (Soltani et al. 2017).

Zhang et al. (2009) explored CV methods in helping project management to qualify and measure progress in their construction projects. Accordingly, an integrated information system was developed to control the productivity and improvement of construction through digital images (using a 3D model of the building). These images were captured on site for the purpose of semi-automated work progress and computing of interim payments along with a warning function system for the potential delays. However, the limitation of this method is that it cannot provide a complete picture of work progress.

Several other approaches have been presented for measuring construction productivity. Choy and Ruwanpura (2006) investigated their research on situation-based simulation modeling for construction productivity. Other methods using neural network were developed (Chao and Skibniewski, 1994; Dissanayake et al. 2005).

2) Object Recognition Methods for Monitoring the Construction Resources

Productivity improvement is considered as the most important part of construction management. Hence, in order to develop the productivity and analyze construction operation, it is important to gather information and data about resources in construction sites. Grau et al. (2009) proposed a video interpretation method that can automatically extract information about resources in construction sites. In essence, video interpretation model was merged with the CV-based hierarchy for analyzing and data collection about resources and productivity of construction project.

Gong and Caldas, (2011) developed a method for object recognition and tracking on the construction sites by combining CV with operating process modeling algorithms. According to this research, the three main algorithms for object recognition are Background subtraction, Haar features, and color-based recognition method.

Background subtraction methods are extensively used for the purpose of object detection from cameras (Piccardi, 2004). Several methods for performing background subtraction have been proposed, which include mixtures of Gaussian for outdoor scene where light intensity changes rapidly (Stauffer and Grimson, 1999), Codebook-based method for scenes with complex moving

objects by using filter concepts (Kim et al. 2005), and Bayesian model-based method for foreground object recognition from videos which contains complex background (Li et al. 2003).

Another method that can be suggested for object recognition is Cascade of simple Haar Features (Viola and Jones. 2001). The main goal of this method is to divide video images into a set of overlapping windows at a different scale and then make a decision whether a window contains a target object or not. Moreover, the result of this decision is based on edge features and line features.

Combination of Haar–Histogram of Oriented Gradients (HOG) and Blob-HOG method together, derived a new method for estimating the ability for recognizing heavy equipment such as dump trucks in construction sites. Hence, Haar-HoG method contains Haar detectors to speed up classification for eliminating most true negative and decrease the number of search windows, then HoG detector is used to reject the false alarm and keep true positives.

In a second approach, a Bayesian-based model was used for foreground and background filter to detect both gradual and abrupt movements in captured videos. However, this method creates some noises which are eliminated using a predetermined threshold to create unified shape called a Blob. The results of this study shows that the two cascade approaches can be beneficial for the application of construction management from the productivity point of view, as well as locating resources in the sites, and safety. Moreover, Haar-HOG method shows better performance due to higher detection rates and lower false positives (Rezazadeh and McCabe., 2011).

To develop a realistic color-based object recognition, it is vital to understand that light condition, view angles, and glare, have a significant impact on object recognition. Gong and Caldas (2011) proposed a Gaussian model to represent the color distribution for the safety vests and background scenes by using sample pixel for probabilistic estimation and recognition of safety vests and background. Also, there are many other color spaces for sample pixels such as Lab, Luv, HSV and the most common of them is RGB.

Another method proposed by Zou and Kim (2007) for accurate analysis of equipment in construction sites is using Hue, Saturation, and Value (HSV) color space. Accordingly, the image color space is used as the base for image segmentation and tracking algorithm. Moreover, HSV brings huge impact over the RGB color space in order to recognize and detect excavators on construction sites. Hue is used to distinguish excavators with different colors whereas saturation is used to distinguish an

excavator from its background differentiating between light and dark colors. Furthermore, both above characteristics are used in simple thresholding and calculating object centroid methods to achieve more accurate results when images are colored.

3) Application of CV for Health and Safety Monitoring in Construction Projects

To eliminate potential hazard on construction sites, continuous monitoring of unsafe conditions and actions is essential. Hence, CV methods have been used for the extraction of safety-related information of site from images and videos as a robust and automated field of observation (Seo et al. 2015). Generally, unsafe conditions and unsafe acts are two main reasons for accidents. Thus, monitoring unsafe conditions and acts plays a key role to prevent resulting safety and health concerns by removing them in the process (Hinze and Godfrey, 2003).

The most important element in construction sites is human's safety. Thus, one of the fundamental equipment for workers in construction sites is a hard hat to protect their life. Detecting hard hats based on the real-time surveillance cameras is one of the new topics in CV to address this issue. Figure 2-4 shows the three main parts of this new method. The first step is to detect the person's face based on the Haar-like face features. The second, shows the skin color detection and motion detection which is used to decreasing the false alarm of faces. The third step is using the color information above the face regions to detect the hard hat (Du et al. 2011).

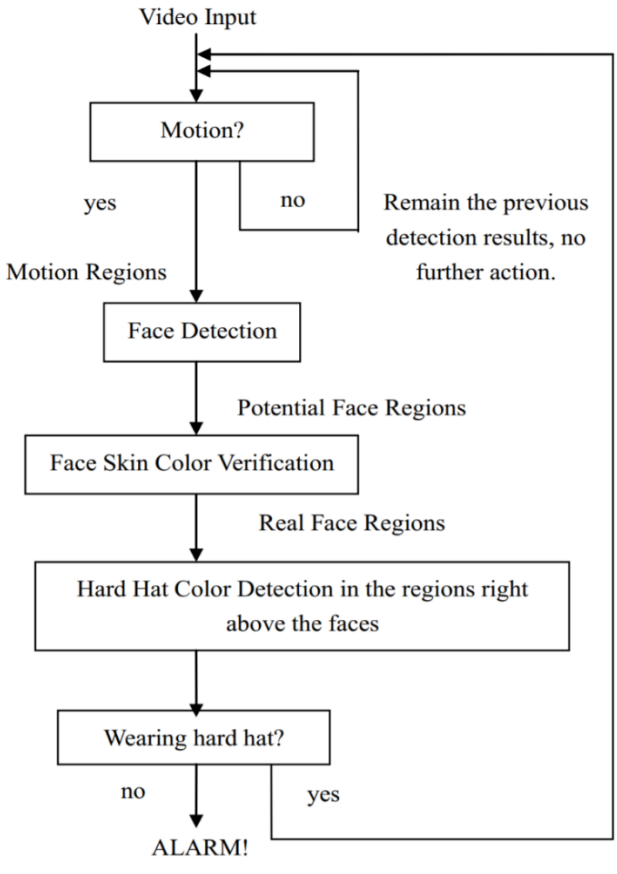


Figure 2-4 The process of hard hat detection (Du et al. 2011)

Park and Brilakis (2012) developed a new method for construction worker detection, which localizes construction workers in the video frame. Accordingly, they used motion, shape, and color cues to limit the detection areas to moving objects, people, and construction workers. The three cues are known as background subtraction, HOG and HSV color histogram, which are used in this on-site tracking. This method is used for safety management purposes to have onsite tracking of workers on construction sites. Figure 2-5 shows the process of this method. Firstly, this method detects foreground blobs where objects in motion are expected to be. Given the stable camera views, the variance of pixel values among a background model and the incoming frames is the major sign to recognize motion. Secondly, it identifies the regions matching to human bodies from the foreground blobs based on their patterns of HOG features. Thirdly, the detection regions that effect from the second step are classified into construction workers and non-workers. It uses color histograms and a K-Nearest Neighbors (KNN) classifier to characterize fluorescent colors of safety gear.

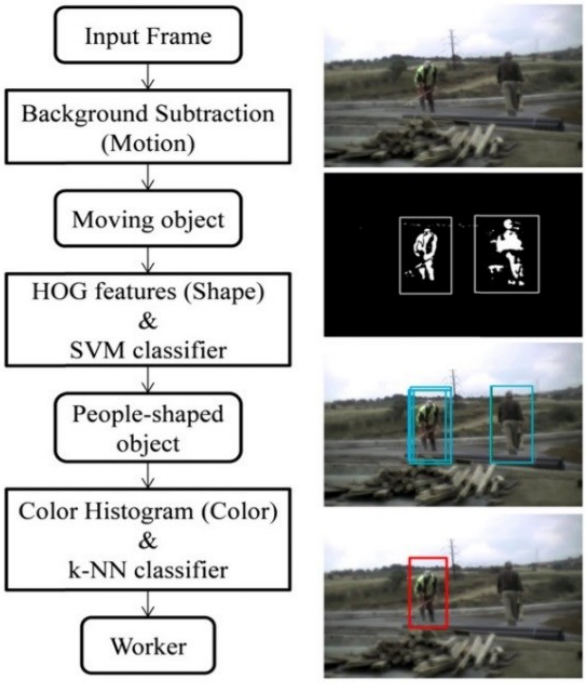


Figure 2-5 Automation in construction (Park and Brilakis, 2012)

Han et al. (2011) suggested dimension reduction and kernel Principal Component Analysis (PCA) techniques to analyze the behavior and motion of workers during an activity in construction sites. Li and Lee (2011) proposed a method on how 3D skeleton video image can be reconstructed within 2D skeleton images recorded from two network surveillance cameras. The results of this method shows that obtaining 3D skeleton video with coordinates of joints has enough information to be used for motion analysis, motion recognition of non-ergonomic issues, such as musculoskeletal disorders, motion visualization, postures and movements of workers on construction sites. Han et al. (2010) proposed behavioral observation with CV technology to detect unsafe acts. Therefore, CV methods for motion detection used to constantly monitor workers' behavior and automatically analyze their level of unsafe acts.

Teizer and Vela (2009) discussed four tracking techniques, namely density mean-shift, Bayesian segmentation, active contours, and graph cuts to determine the most appropriate tracking method for automatically tracking workers within construction site by the video camera. Hence, mean-shift tracking is a statistical template-based tracking algorithm and is only able to track the overall position of the workers. The remaining methods are segmentation-based tracker methods, which is able to track the shape of the worker under the appropriate imaging condition. According to Teizer and Vela (2009),

Bayesian segmentation has the most promising approach since the boundary of the target region can be found more precisely and can achieving more accurate results in image-plane tracking. According to Soe et al. (2015), CV-based safety and health monitoring can be divided into three categories based on the types of information achieved from collected images or videos: object detection, object tracking, and action recognition. Figure 2-6 shows how data can be collected in 2D and 3D images by surveillance cameras, portable cameras, flash LADAR, RGB-D Sensors, and stereo vision camera.

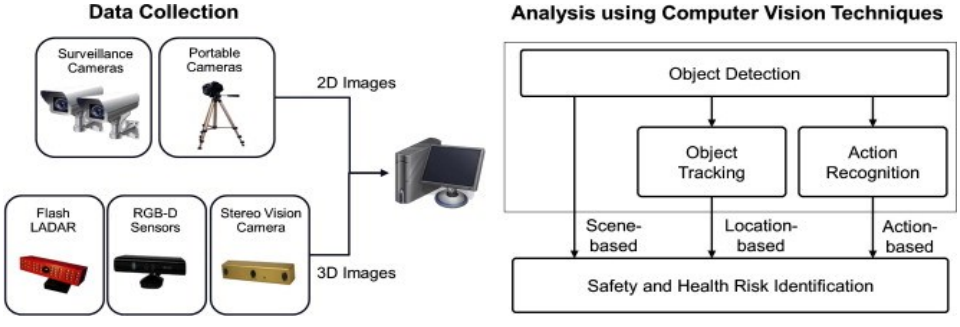


Figure 2-6 General framework for CV-based safety and health monitoring (Seo et al. 2015)

2.9 Deep Learning

Deep learning is an advanced technology which plays a key role in AI, machine learning, and big data and, has shown excellent and robust performance. The idea of deep learning came from the study of Artificial Neural Network (ANN). The biological nervous system of an animal brain, formation, and function leads to the idea of the NN computing systems. The structure of deep learning, contains various layers of simple modules, and a great part of them are applied for learning and several of these are used to process non-linear input-output mapping. The architecture of multi-layer learns through training data.

Researchers believed that for the low dimensionality data, it is better to use machine learning and for unstructured and high dimensional data, deep learning is recommended (Challet, 2017). Complex deep learning models arise from the limitation of the linear model in terms of explaining high dimensional complicated representation models. One of the other differences between machine learning and deep learning is that machine learning uses single-level representation learning, whereas deep learning uses various processing layers to learn the finest features required to signify the data. For more complex models with high dimensional data, such as image recognition, text processing, and speech recognition, deep learning is a great choice due to the high level of learning (Goodfellow et al. 2016). Computation process and big dataset for training models were two main challenges for deep learning

in the past. Available open sources dataset, faster processing computing devices, and superior performance of GPU plays an important role in deep learning.

2.10 Neural Networks

A Neural Network (NN) behaves by acquiring experimental knowledge, through a learning process which makes it easy to use due to its parallel distributed processor. A NNs knowledge is stored within inter-neuron connection strengths known as weights (Haykin, 2009). Neurons or nodes work together to produce an output. The neurons are the basic building blocks of the brain and artificial NN. Identified layers are a mixture of neurons. Each neuron gets their inputs (x) from the sources connected with them and multiples it by the weight (w) of its connection and then sums it up as illustrated in Figure 2-7. Then, bias is added to this value. The activation function processes this sum of weighted connection and the bias. It normalizes the result and outputs it as y. In Deep Neural Networks (DNN) there are usually many hidden layers.

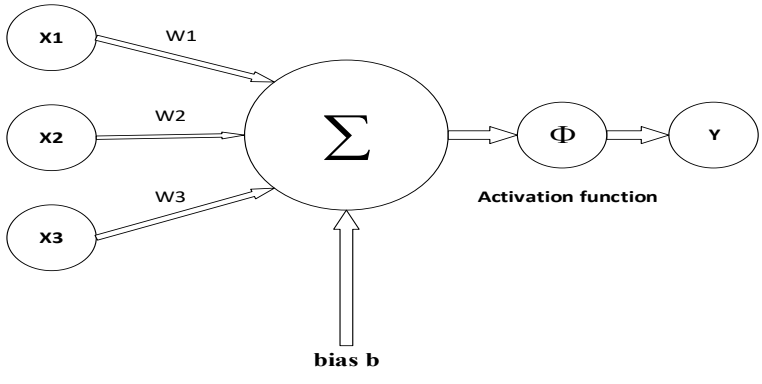


Figure 2-7 A simple model of artificial neuron (Haykin, 2009)

2.10.1 Activation Function

In convolutional neural layers, the activation function is utilized to bring non-linearity to a linear output. There are different activation functions in CNNs with different uses, cases and functionality. In this research, we focus on the three mostly used activation functions, namely ReLU, tanh, and sigmoid. A rectified linear unit (ReLU) is mostly used in CNN architectures and maps only positive values, while negative values are mapped to zero. Hyperbolic tangent (tanh) and sigmoid functions are correlated activation functions, which usually play a central role in classification problems. The main difference between these two is that the sigmoid function maps the values between zero and one,

whereas tanh has a range between -1 and 1. Figure 2-8 compares these three activation functions (Gomez, 2015).

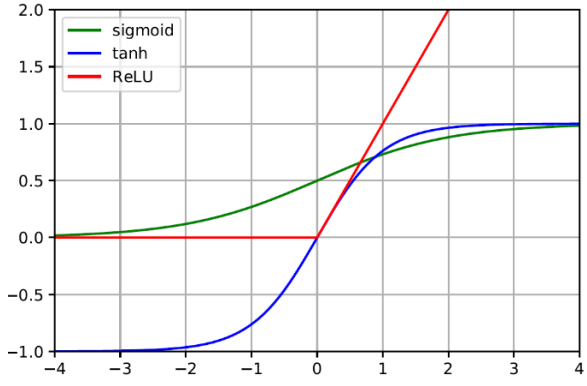


Figure 2-8 Comparison of sigmoid, tanh and ReLU (Gomez, 2015)

2.10.2 Fully Connected Layer

The fully connected layer is one of the topologies in NN connections between layers. As the name indicates, a fully connected layer is a layer where each pair of node in this layer is connected. It is common to call a fully connected layer a dense layer. In order to extract more complicated features in the learning process, fully connected layers will be added to the network. Since in fully connected layers, every two nodes in the layer are connected, it helps to find the correlation between every two possible features, and as a result extract more complex results. However, this complexity comes at a higher computational cost. Consequently, it is common to have only one or two fully connected layers in a neural network structure.

2.10.3 Backpropagation

The error in neural networks is the difference between the actual output and the output generated by the network. The goal is to reduce the error to the minimum possible value. In order to reduce the error, the network updates the weights of the neurons. The backpropagation algorithm is used to solve this issue by making the training error as small as possible. This is done by iteratively passing batches of data through the network and updating the weights. This mechanism is also known as stochastic gradient descent (Goodfellow et al. 2016).

2.11 Convolutional Neural Network

CNNs are an ANN that involves multiple layers of neurons. A typical topology of the CNN consists of convolution, nonlinearity, subsampling repeatedly connected with fully connected layers. Modern

CNN framework contains many numbers of layers (deep layers) with convolutional and subsampling layers followed by one or more fully connected layers. Figure 2-9 shows an input image passing through convolutional layers followed by maxpooling layers, fully connected layers, and output layer.

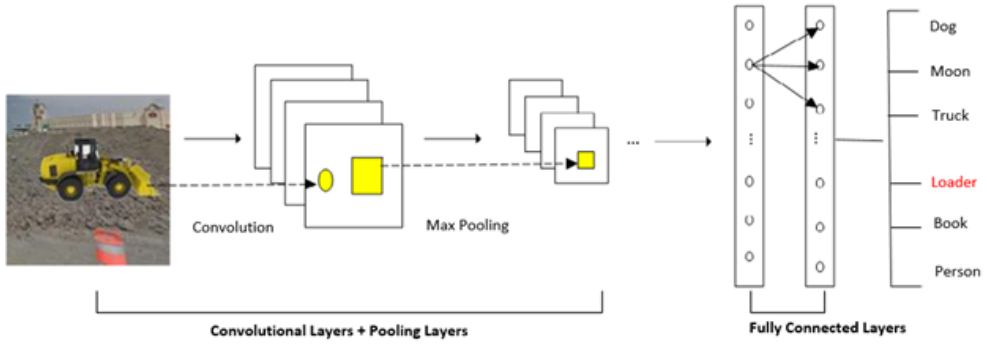


Figure 2-9 A multi-layer convolutional neural network (Krizhevsky et al. 2012)

The input image is put through the series of convolution and pooling operations followed by the fully connected layer to generate the output. To prevent overlaps in CNN, the convolution filter of fixed size window is utilized at every point of the image, preventing overlap. Feature maps are the outcomes gotten from these filters. Max pooling is used on all positions of these feature maps to decrease the dimensionality. Then, the convolution filter is used once again followed by the maxpooling. These procedures can be done continuously. Convolutional layers are usually linear; therefore, they may not be able to represent feasible nonlinearity. The output layer obtains the data from the fully connected layer about the output class related with the score of each class category (Kim, 2017).

2.11.1 Modern Convolutional Object Detectors Architectures

A lot of progress has been made in recent years on object detection due to the use of CNNs. In the following sections, we are going to introduce the structure of modern object detectors based on three networks, namely Faster R-CNN, R-FCN, and Multibox SSD.

2.11.1.1 Faster Regional-based Convolutional Neural Network

Two classes of RCNN are Fast RCNN (Girshick, 2015) and Faster RCNN. As the names indicate they are aimed at improving the running time of the algorithm, while preserving the accuracy of the model. Among all these models, Faster RCNN is becoming the most dominant and popular architecture in object detection (Ren et al. 2015). RCNN is built up based on a three-step process. In the first step, the algorithm looks for the object in the input image, known as region proposal. In the second step, the

CNN starts to perform on top of these detected regions. In the last step, the algorithm runs SVM on the CNN output in order to apply classification. In Faster RCNN, the algorithm is applied by only two steps. The first step is running over the image to detect and extract the feature maps. Then, the Region Proposal Network (RPN), using these featured maps, finds the set of object proposal boxes along with their scores. In the second step, the area of the image is extracted and a bounding box with probability of accuracy in prediction is added to the image. This is done using the extracted features fed to all layers of the network. Finally, it classifies the object and outputs the results. In Figure 2-10, the whole process develops on the unique single network that lets the system apply convolutional filters with the detection network. It is believed that the Faster RCNN architecture has very promising results on difficult object recognition and classification (Geiger et al. 2012).

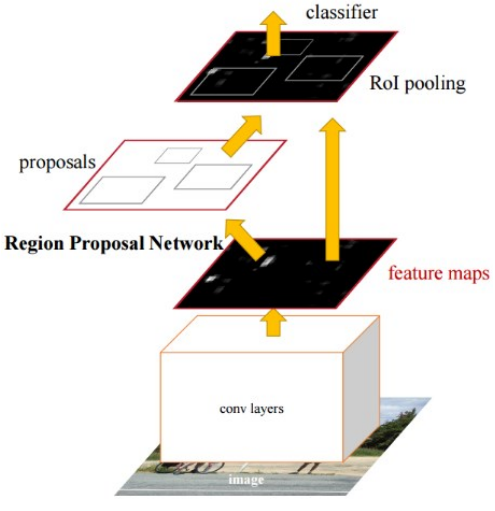


Figure 2-10 Unified network for object detection *applied* in faster RCNN (Ren et al. 2015)

2.11.1.2 Region-Based Fully Convolutional Network

The Region-based Fully Convolutional Network (RFCN), which includes region proposal and region classification, is almost the same as the Faster RCNN, but with a very slight difference. RFCN uses features that are taken from the last layer of features preceding region proposals. By using this method, it is easier to reduce the amount of memory applied in region computation. Also, other researches showed that RFCN combined with Resnet101 feature extractor has better performance compared to Faster RCNN (Geiger et al. 2012; Dai et al. 2016). Figure 2-11 illustrates the architecture of RFCN. It starts by applying the convolutional filter over the image. Then the feature map is applied to create a score of positive-sensitive score maps. After that, the fully convolutional regional proposal network (RPN) generates the Regions of Interest (RoI) from the previous layer output (feature maps). Each

generated RoI is divided into subregions (bins) as scored maps. In this step, the score of each bin is evaluated with the corresponding position of the object. This procedure is repeated to all the bins and for all the classes showed in the image.

When the bins could be modeled accurately enough with the sub-region of the object, the average score can be calculated for each class. At the last stage, max pooling is applied. The reason which makes RFCN model faster than the Faster RCNN is that RFCN uses fully convolutional network and distributes the computation over the networks (Dai et al. 2016).

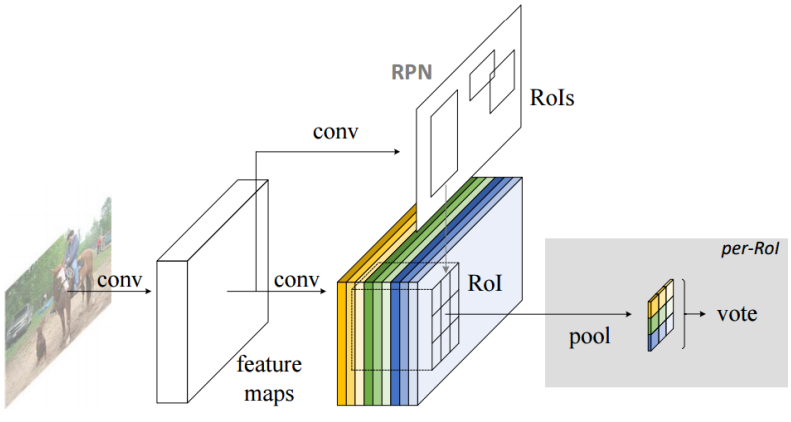


Figure 2-11 The architecture of RFCN (Dai et al. 2016)

2.11.1.3 Single Shot MultiBox Detector

The problem of object detection is solved by a Single Shot multi-box Detector (SSD) architecture through a single pass of a feedforward convolutional network. SSD does not produce region proposals and resampling of image segments. These two factors make SSD very fast model and help to reduce computational time (Fuentes et al. 2017). Also, this model can use feature maps from different convolutional layers, which leads to ease of use.

Moreover, this network generates a huge number of bounding boxes that show the scores of the object class in those boxes. Non-maximum suppression is applied to remove boxes under a specific number of thresholds. The higher the threshold, the more uncertain detections will be removed without a trial. Thus, only the boxes with superior confidence values are used for classification. Compared to the other architectures, SSD is faster due to the fact that it does all the steps in one shot. However, the accuracy of detection is less than other architectures. The SSD has fixed-size bounding box because its architecture is based on the feedforward CNN. Figure 2-12 shows an input image to the SSD architecture that passed through a series of the convolutional layers and then down sampled. The output

of the last convolutional layer is linked to the SSD layer. Several collections of feature maps at various scales are gained from the convolutional layers with the prediction of object classes and a set of bounding boxes. The comparison between the predicted box and the ground truth of the object leads to find the best one between them and the one with the higher Intersection over Union (IOU).

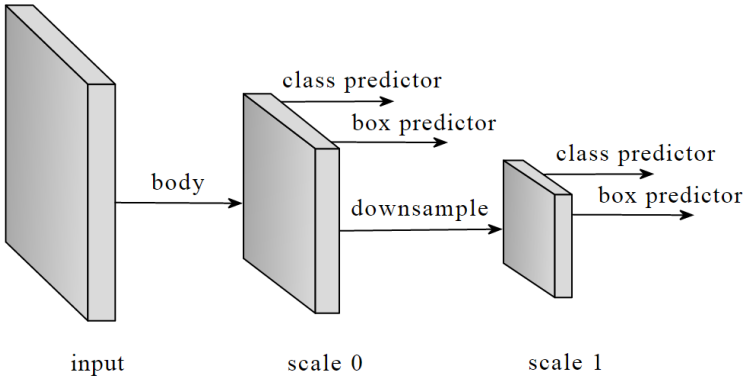


Figure 2-12 SSD Layers consist of small convolutional layer which is attached on top of base layer (Gluon, 2017)

Gevers and Stokman, (2004) suggested kernel density histogram to contrast high color invariant histogram for object recognition by focusing on normalized color RGB against contrast colors which is extensively used in computer vision.

2.12 Feature Extractors

The object detection model is organized by the combination of meta-architecture including a feature extractor and a classifier as shown in Figure 2-13. The main structure of the object detection model is based on applying feature extractor to extract the information of the object from the data. In the first step, the input image is given to the feature extractor, then in the second step, the extracted features from the first step are given to the classifier that categorizes the class and the location of the object within the image.

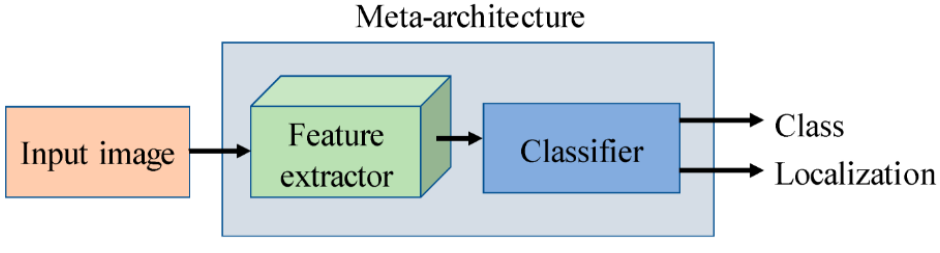
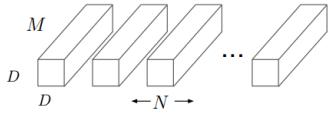


Figure 2-13 Object detection model architecture (Fuentes et al. 2017)

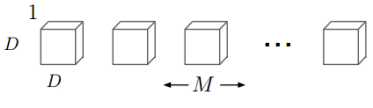
The main advantage of feature extractor is that it has a very deep architecture that helps to improve the accuracy while simplifying the computational time. There are various feature extractors available for object detection namely, VGGNet, ResNet, MobileNet, and Inception, which are considered as the most popular feature extractors. However, in this study, we used MobileNet and ResNet feature extractors from the above-mentioned meta-architecture.

2.12.1 Mobile Network

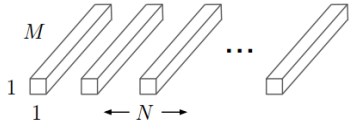
Mobile network (MobileNet) known for its light network, which is very useful for mobile devices embedded systems used in real-time computation. The theory of mobile network architecture relies on dividing the typical convolutional filter into two parts. The first part is depthwise convolution, and the second part is pointwise convolution. Comparing the computation complexity, the normal convolutional filter has more computation complexity than depthwise and pointwise convolutions. The computation cost of the convolution depends on various parameters such as the input network (M), size of the output network (N), feature map size and the kernel size (D). The computation complexity of the normal convolutional filter illustrated in Figure 2-14. The standard convolutional filters in (a) are replaced by two layers: depthwise convolution in (b) and pointwise convolution in (c). This classification improved the computation speed (Howard et al. 2017). MobileNet uses 3 × 3 depthwise separable convolutions which use between 8 to 9 times less computation than standard convolutions at the cost of only a small reduction in accuracy.



(a) Standard Convolutional Filters



(b) Depthwise Convolutional Filters

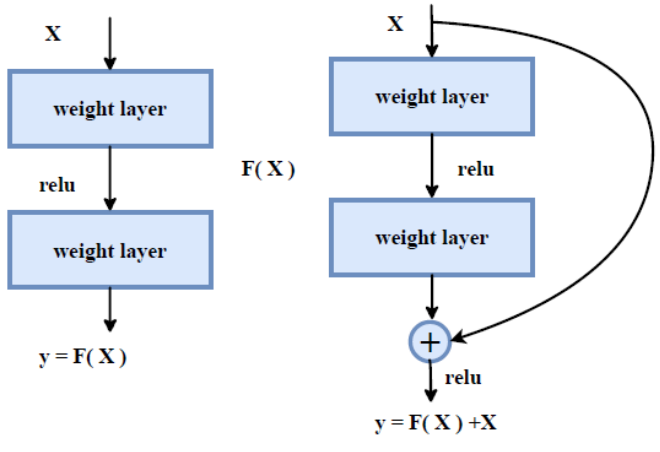


(c) Pointwise Convolutional Filters

Figure 2-14 MobileNet feature extractor (Howard et al. 2017)

2.12.2 Residual Network

Residual Network (ResNet) is based on residual learning theory. The ResNet is basically a 150-deep CNN made by equal "residual" blocks. Network depth is of crucial importance in NN architectures, but deeper networks are more difficult to train. The residual learning framework eases the training of these networks and enables them to be substantially deeper - leading to improved performance in both visual and non-visual tasks. ResNet is considered as record breaker architecture for image classification, object detection, and semantic segmentation. With the network depth increasing, the augmentation in depth of the deep NN improves the accuracies till some point and after that, it starts degradation. Residual learning attempts to simplify this matter of accuracy degradation in NN. The structure of the residual network is to use blocks that re-route the input, and add to the concept learned from the previous layer. The idea is that during learning, the next layer will learn the concepts of the previous layer plus the input of that previous layer. This would work better than just learning a concept without a reference that was used to learn that concept. In Figure 2-15, F(X) shows the residual function of non-linear CNN layers. In the plain block (a), it is difficult to get identity mapping by pushing the residual function to zero. It is easier to get identity mapping in the residual block (b) than in the plain block (a) (He et al. 2016).



(a) Plain block (b) A Residual block

Figure 2-15 A residual block, the fundamental building block of residual networks (He et al. 2016)

2.13 Assessment Criteria

Various assessment matrices exist in order to measure the performance of machine learning algorithms. The object detection model can be used to find the accurate location of the object in

addition to recognizing the object itself. The IoU measure, also known as bounding box overlap, is popular among other assessment approaches (Everingham et al. 2015). The performance measure of the object detection model is quantitative and explains how many objects are detected correctly or wrongly (False alarm). The overall quality of the object detection model is calculated in terms of IoU and precision. For a large scale multi-class dataset, the mean Average Precision (mAP) is used to measure the performance of the method in the whole dataset (Everingham et al. 2010). Apart from the accuracy measure, the computation complexity of the model is of main concern. The performance of the trained detector on unseen data, training time of the model and processing speed are major issues that should to check while selecting the object detection model.

2.13.1 Accuracy Metrics

In machine learning, precision and recall are important factors to evaluate the performance of a classifier. Precision is the ratio of retrieved items over all the presented items. In the object detection case, precision is the sum of the correctly detected objects divided by the total population of the objects that are detected by the detector as shown in Equation 2-1.

$$\boldsymbol{Precision} = \frac{True\ Positive}{True\ Positive + False\ Positive} \qquad \text{Equation 2-1}$$

Recall is actually calculates how many of positive or relevant items that are captured through labeling it as positive (True positive).

$$\boldsymbol{Recall} = \frac{True\ Positive}{True\ Positive + False\ Negative} \qquad \text{Equation 2-2}$$

Another two indices which shows the total performance are F_1-score and Accuracy as follow:

$$\boldsymbol{F_1\text{-}score} = \frac{Precision * Recall}{Precision + Recall} * 2 \qquad \text{Equation 2-3}$$

To understand the terms used in performance measurement, it is important to consider the binary classification problem. The output of the classifier is positive or negative based on whether it is

classified correctly or not. The detection of the single object can be considered as a classification task. The clear and concise way to understand the idea behind the binary classification accuracy measure is using the confusion matrix. Table 2-2 shows the confusion matrix, also known as contingency table. The true class conditions are the ground truth of the data while predicted class conditions are the prediction made by the classifier. The model prediction result will be one of four outcomes, true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN). Each instance holds two parts, the first part is whether the prediction is true or false, and the second part is the predicted class (positive or negative). If the true class conditions and predicted class conditions are matched, it is considered as TP. If both the real class and prediction class are negative than the result is considered as TN. If the ground truth of the class is positive but the prediction is negative, than this instance is called FP. If the prediction is positive but the ground truth is negative than that instance is called FN.

Table 2-2 The table of confusion matrix

		True Class Condition	
		Positive	Negative
Predicted Class Condition	Positive	True Positive (TP)	False Negative (FN)
	Negative	False Positive (FP)	True Negative (TN)

The diagonal from the upper left to lower right is the correct classification. While the other diagonal is the false classification.

The population of positive class (P) is the sum of all positive instances in data, calculated as, $P = TP+FN$. The population of negative class (N) is the amount of real negative instances in data, $N = TN + FP$. Total population is sum of positive and negative classes, $Total\ (T) = P + N = TP + FN + FP + TN$. The amount of false prediction is, $F = FN + FP$.

Many performance measurement can be calculated based on this contingency table. Table 2-3 contains the list of most commonly used metrices and their calculations. The calculation of TPR, sensitivity, and recall are same. Also, PPV and precision calculation are same. The accuracy (ACC) is the measure of the fraction of correctly predicted instances over total population of instances. The high ACC value is always good for the classification model but is not sufficient to prove that the model is performing well (Tikkainen, 2014; Adhikari, 2018).

Table 2-3 Calculation of various performance metrics based on the confusion matrix

Term	Calculation
True Positive Rate (TPR), Sensitivity, Recall	$TPR = \frac{TP}{P} = \frac{TP}{TP+FN}$
True Negative Rate (TNR)	$TNR = \frac{TN}{N} = \frac{TN}{TN+FP}$
Positive Predictive Value (PPV), Precision	$PPV = \frac{TP}{TP+FP}$
Negative Predictive Value (NPV)	$NPV = \frac{TN}{TN+FN}$
Accuracy (ACC)	$ACC = \frac{TP+TN}{P+N}$

Mean Average Precision (mAP)

The mAP is an evaluation protocol used by the TensorFlow Object Detection API for the measurement and comparison of object detection model accuracy. According to the PASCAL metrics, mAP is calculated as the mean of all object classes for the average precision (AP). For each object class, average precision is the area under the Precision-Recall curve. The precision-recall curve shows the tradeoff between precision and recall for different thresholds.

Calculating Average Precision (AP)

In object detection, first the predicted bounding boxes (for all the images) are sorted according to their confidence score, and the precision and recall values are calculated for each confidence score. Afterwards, a Precision-Recall curve is plotted for each of these precision recall values at different scores. As pointed out above, AP summarizes the area under the Precision-Recall curve in one number. AP is the averaged precision across all recall values between 0 and 1. However, the PASCAL VOC metric is based on interpolated average precision. Interpolated average precision is defined as "the mean precision at a set of eleven equally spaced recall levels (Everingham et al. 2010)." AP is calculated based on the formula given below:

$$AP = \frac{1}{11} \sum_{r \epsilon \{0, 0.1, 0.2.., 1\}} p_{interp}(r) \qquad \text{Equation 2-4}$$

In interpolated AP, precision at each recall level corresponds to the maximum precision across all the recall values higher than recall level. Precision at recall is calculated as follows:

$$p_{interp}(r) = \max_{\tilde{r}:\tilde{r}\geq r} p(\tilde{r}),$$
$$\text{where } p(\tilde{r}) = \text{Precision measured at recall } \tilde{r}, \qquad \text{Equation 2-5}$$
$$\tilde{r} = \text{Recall values higher than recall level r}$$

2.13.2 Detection Speed

Computers integrated with high-end GPUs process graphical contents faster than basic computers that only have CPUs or less powered graphics card (Murray, 2017). Frames per second (FPS) is the frequency of consecutive images, also known as frames, appearing on the display. FPS is the widely used metric in broadcasting, films, image and video processing, games, and computer graphics. Computation complexity of the model is one of the main concerns in real-time object detection. There is no clear answer to how much inference speed is considered real-time detection. However, a model is said to be real-time if, its output is faster than or as fast as to the input (Murray, 2017). The inference speed of 1 FPS means every second a single frame is displayed and video of 1 minute contains 60 consecutive frames. The processing time calculation gives the general information about the computation complexity of the algorithm. Deep learning frameworks are complex in nature and computationally heavy. Hence, it is hard to achieve real-time inference on less powerful devices. We considered the FPS as the measure of the computational complexity of the model. As we are running all models in the same working environment, the FPS results give a better understanding of computation cost of experimented models. We compare the computation complexity of experimented object detection detectors during training and evaluation on the test dataset.

2.14 Overfitting

The main challenge for the machine learning algorithm is to perform well on unseen data. The ability to perform well on previously unseen data or data other than training set is called generalization. The generalization error or test error is the expected value of the error measure on new data. For the good performance of an algorithm, the generalization error must be as low as possible. The training error can be calculated from the training set. The determining factor of the machine learning algorithm is its ability to reduce the training error and its generalization gap. Generalization gap is the distance

between the training error and test error. Under fitting and Overfitting terms are used to describe the machine learning challenges. Underfitting occurs when the model has high training error. Underfitting refers to a model that cannot model or generalize training data to new data. Overfitting happens when the generalization gap is high. Overfitting is the case when the model performs well on the input (train) data but poorly generalize on unseen data. Regularization is a technique used to make modification on the learning algorithm to mitigate the test error but not the training error. Regularization attempts to construct the model structure as simple as possible to avoid the overfitting effect. The behavior of training and test (generalization) error is different at the different levels of model capacity. Capacity is the ability of the model to fit the wide range of functions. In the beginning, both the training and generalization errors are high. This area is known as the underfitting zone. When the capacity increases, the training error decreases but the generalization error starts increasing. At the overfitting zone, the training error is lower but the generalization error is higher, making the generalization gap bigger. The optimal capacity is the boundary line to distinguish the underfitting zone and overfitting zone (Goodfellow et al. 2016).

2.15 TensorFlow Object Detection API

The TensorFlow object detection API is an open-source software library released from Google Brain team (Huang et al. 2017), which gives access to an open source framework for constructing, training and deploying of various object detection models. This library provides numerical computation to be used on different devices and servers.

TensorFlow model zoo provides a collection of detection models pre-trained on the COCO dataset, the Kitti dataset, the Open Images dataset, the AVA v2.1 dataset, and the iNaturalist Species Detection Dataset. The COCO dataset contains photos of 91 different class of objects with a total 2.5 million labeled instances and 328,000 images Lin et al. (2014), Kitti captures locations in Germany in rural areas and on highways, consisting of different classes on road environment (Geiger et al. 2012). TensorFlow Object Detection API aims to make it easy to construct, train and deploy object detection models.

2.16 Object Classification Descriptors

The bag of words (BoWs) model is one of the most popular methods for object categorization. The ultimate point consists of quantizing each key point extracted into one visual word and then representing each image by a visual word histogram. This process of vector quantification enables the

image to be represented by a histogram of the visual words that is often called bag of words and converts the problem of object classification to text classifying problem (Zhang et al. 2010a).

The idea behind using the Local Binary Pattern (LBP) features is that a face can be considered a micro pattern composition. LBP is the first order circular derivative pattern in nature of images, a micro pattern generated by the binary gradient direction concatenation. However, the pattern of the first order cannot extract more detailed information from the input object. In fact, the operator can capture more detailed information about discrimination (Zhang et al. 2010b).

The Histogram of Oriented Gradients (HOG) is used for the purposes of object detection in computer vision and image processing. The technique counts gradient orientation occurrences in localized parts of an image. This method is similar to that of edge orientation histograms, scale - invariant transformers and shape contexts, but differs because it is calculated on a dense grid of evenly separated cells and uses local contrast standardization for higher accuracy (Dalal & Triggs, 2005).

The VGG-F network is an eight layer deep neural network designed and trained for classification of images. The network was trained with gradient descent with momentum on ILSVRC data (Wozniak et al. 2018).

2.17 Summary and Conclusions

This chapter reviewed the concepts, methods, techniques, algorithms, and technologies involved in object detection as a keystone in improving the safety and efficiency of construction sites. While Most of the researches only studied the productivity and monitoring resources in the construction sites, the safety with its undeniable importance is still suffering from a lack of studies. Furthermore, the technologies to reduce potential risks and hazards were discussed in detail as well as safety research using computer vision methods. Finally, different object detection technologies were also reviewed and evaluated to select the most suitable methods to gain the required information.

CHAPTER 3 Comparing the performance of CNN and other Conventional methods for classifying equipment on construction sites using synthetic images

3.1 Introduction

In the previous chapters we discussed the problem statement and past research approaches. The main problem was the high rate of fatalities and injuries on construction sites. Researches and current approaches in discussion were going toward automated systems. The objectives are to analyze the real-time videos of construction sites and detect objects, in order to predict the ongoing hazards or near-miss events. In this chapter, we will discuss the methodology of this research to achieve the goal which is comparing the performance of CNN and HOG for classifying equipment on construction sites using synthetic images.

3.2 Proposed Method

The result obtained from researches conducted about construction sites showed the significance of real-time analytics and detection of the potential hazards. In order to reduce the collisions between construction equipment, machinery, workers, and crews, computer vision is one of the promising methods to detect potential accidents beforehand. Soltani, (2017) proposed excavator pose estimation for safety monitoring using computer vision.

This chapter aims to achieve the following objectives: (1) automatically generating synthetic images of three different types of construction equipment using the 3D models; (2) adding background to the generated images; (3) automatically annotating the generated images; and (4) evaluating and comparing the performance of object detection with training and testing images. In the following, the methodology will be discussed in detail.

3.3 Synthetic Data Generation

Synthetic images can be useful when providing a large and accurate dataset are not applicable. Also, they can be generated from different orientations or points of view. To create synthetic images, in the first place, a 3D model of the object is chosen, and a dataset of synthetic images is created and used as a large and reliable dataset to training and testing purposes. The construction equipments selected for this study include the three most used equipments in construction sites, namely, excavators, loaders, and trucks. Figure 3-1 shows samples of real images used for training

equipment. Images are taken from different construction sites where different types of equipment with different poses and angels are used.

(a) Excavators (b) Loaders (c) Trucks

Figure 3-1 Sample of real images used for training

3.3.1 Equipment Conditions and Camera Views

It is assumed that the 3D models of equipment are available and can be imported to a 3D modeling tools (e.g. 3Ds Max (Autodesk, 2014)). The first important factor that should be considered in the sensitivity analysis is the desired equipment on construction site. The object itself, has the most important role on object detection. The more accurate and more realistic the targets are, the higher accuracy in detection at the end. Here, we consider the four most effective factors for the desired equipment namely: (1) camera views, (2) poses (3) shape, and (4) color of equipments.

(1) Camera View: Usually, equipment would be seen from the street level views. However, it is far different for the cameras which are installed at a high elevation to have a wide range of views of the environment. Hence, if it assumed that if we know the location of cameras on top of the equipments we can generate synthetic images of all possible view angels for each equipment. In this study, eight viewing angles from 5 to 75 degree were considered for each equipment and poses as shown in Figure 3-2.

Where:

Number of images for each pose * Angle = Total number of generated synthetic images

360 * 8 = 2,880

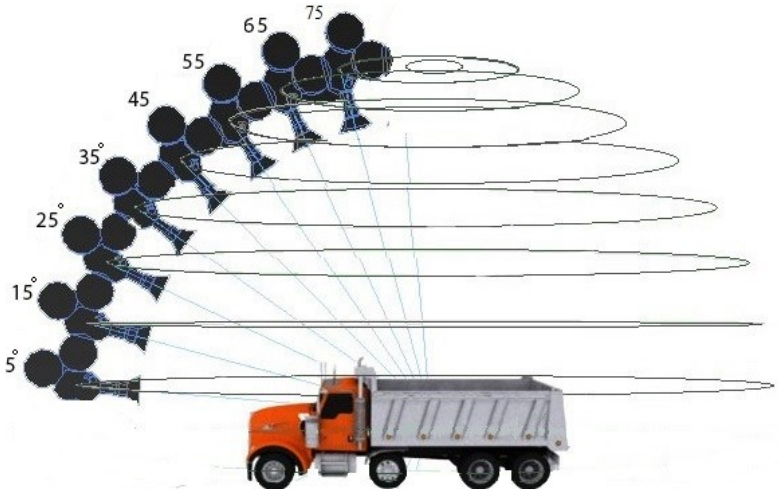


Figure 3-2 Camera positions and 3D model in virtual environment

Figure 3-3 shows the various viewing angles for each equipment.

Figure 3-3 Generated viewing angles for each equipment

Degree / Equipment	5	15	25	35	45	55	65	75
Excavator								
Loader								
Truck								

(2) Poses of equipment: The poses of equipment working on construction sites can change constantly. For example, the arm and the bucket of the loader and excavators change their angels during operations. Therefore, different poses of equipment are considered to improve the detection.

The first equipment is excavator with 27 different poses under various poses of the boom, arm, and bucket considered for it. However, only the most seven common poses which mostly show the digging process were generated. Table 3-1 shows all possible poses and the highlighted boxes show the

selected poses for generating synthetic images. Figure 3-4 shows all generated poses for the excavator from a sample of camera views.

Table 3-1 All possible and selected poses for generating synthetic images for the excavator

Number	Boom	Arm	Bucket	Number	Boom	Arm	Bucket	Number	Boom	Arm	Bucket
1	up	up	up	10	Middle	up	up	19	down	up	up
2	up	up	Middle	11	Middle	up	Middle	20	down	up	Middle
3	up	up	down	12	Middle	up	down	21	down	up	down
4	up	Middle	up	13	Middle	Middle	up	22	down	Middle	up
5	up	Middle	Middle	14	Middle	Middle	Middle	23	down	Middle	Middle
6	up	Middle	down	15	Middle	Middle	down	24	down	Middle	down
7	up	down	up	16	Middle	down	up	25	down	down	up
8	up	down	Middle	17	Middle	down	Middle	26	down	down	Middle
9	up	down	down	18	Middle	down	down	27	down	down	down

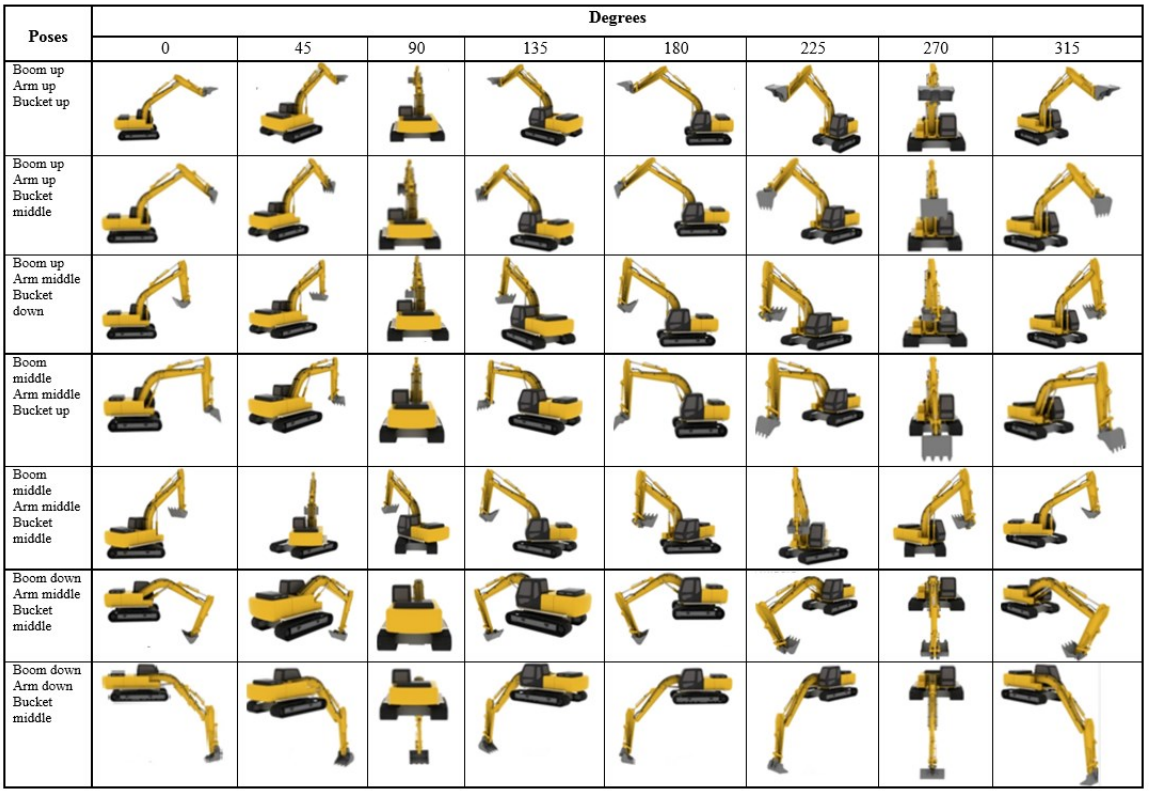


Figure 3-4 Different poses of excavator

The second equipment is loader, which has nine different poses under various conditions of the arm and bucket. However, only the three most common poses were generated for it. Table 3-2 shows all possible poses and the highlighted boxes show the selected poses for generating synthetic images. Figure 3-5 shows all generated poses for the loader from a sample of camera views.

Table 3-2 All possible and selected poses for generating synthetic images for the loader

Number	Boom	Bucket	Number	Boom	Bucket	Number	Boom	Bucket
1	up	up	4	Middle	up	7	down	up
2	up	Middle	5	Middle	Middle	8	down	Middle
3	up	down	6	Middle	down	9	down	down

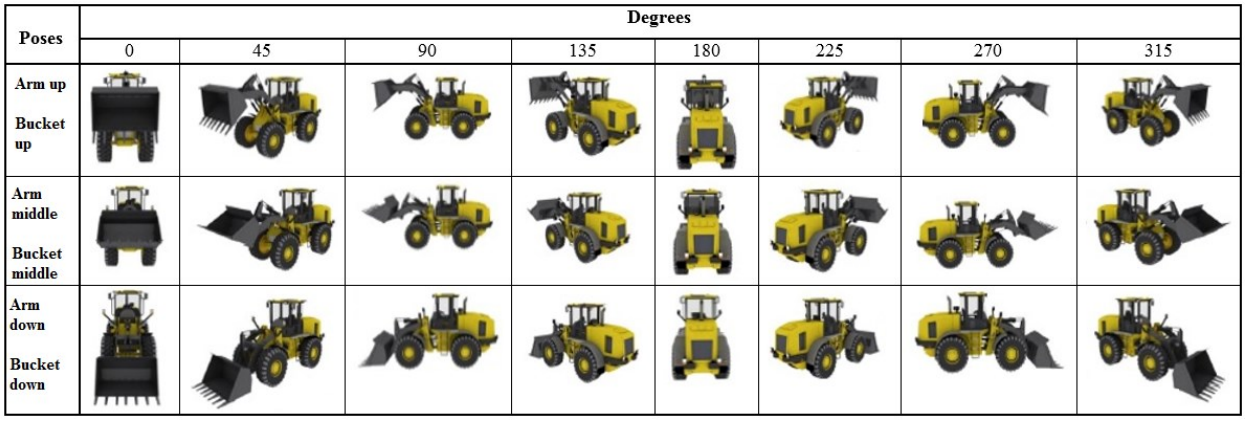


Figure 3-5 Different poses of Loader

The last equipment is the truck for which only one pose is considered. Figure 3-6 shows examples of the generated views for the truck.

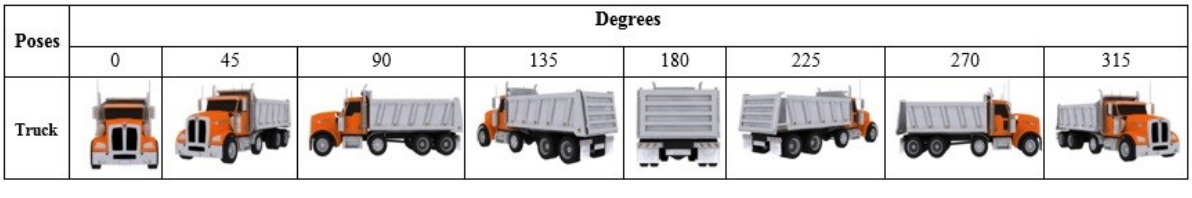


Figure 3-6 Different poses of truck

In total, seven different poses were considered for the excavator, three for the loader, and one for truck. Each equipment and poses were generated from 0-360 degrees. Therefore, 360 images generated for each pose and angle. Table 3-3 shows the numbers of generated images through 3Ds Max for each pose and equipment.

Table 3_3 Total number of generated synthetic images

Equipment type	Poses	Number of images	Total number of generated images
Excavator	Boom up, Arm up, Bucket up	2,880	20,160
	Boom up, Arm up, Bucket middle	2,880	
	Boom up, Arm middle, Bucket down	2,880	
	Boom middle, Arm middle, Bucket up	2,880	
	Boom middle, Arm middle, Bucket middle	2,880	
	Boom down, Arm middle, Bucket middle	2,880	
	Boom down, Arm down, Bucket middle	2,880	
Loader	Arm up, Bucket up	2,880	8,640
	Arm middle, Bucket middle	2,880	
	Arm down, bucket down	2,880	
Truck	---------	2,880	2,880
Total	---------		31,680

(3) Shape of equipment: The shape and size of equipment are considered as important factors for object detection. It could be any brand of equipment (Volvo, Caterpillar, Komatsu, Hitachi, etc.) and they have different shapes and appearances. As it can be seen in Figure 3-7, four excavators of different brands have different colors, shapes, and appearances. Hence, for detection, it is important to consider which equipment brand is working on the site.

(a) Excavator brand Caterpillar (b) Excavator brand Hitachi (c) Excavator brand Komatsu (d) Excavator brand Volvo

Figure 3-7 Samples of excavators from different brands

(4) Color of equipment: Beside the shape and appearance of equipment, the color of equipment can be different. Therefore, the color of equipment can be customized based on the equipment on construction

sites. Moreover, to create more realistic images that match with their background, equipments are generated under two different weather conditions: snowy and dry weathers. Hence, synthetic snow was created to cover the surface and body of the equipment. In this method, first the different poses of equipment were created, then they were colored based on the desired colors. After that, synthetic snow was used to cover the surface of the equipment. Therefore, the system is trained based on all varieties of colors that the equipment has on site. As depicted in Figure 3-8, three different colors of loaders are considered to improve the quality of dataset for training.

(a) Loader with yellow bucket (b) Loader with gray bucket (c) Loader with gray color

Figure 3-8 Sample of the synthetic image with different colors

3.3.2 Adding background

In the construction sites, equipment never appears in isolation. They are surrounded with other objects and environment, providing a strong contextual information to be used by visual systems. There is a consensus that objects appearing in a consistent or familiar background are detected more accurately and processed more quickly than objects appearing in an inconsistent scene (Aude & Torralba, 2007). Studies have shown that the background of the image has impacts at various levels, such as semantic, spatial configuration, and poses. Biederman, (2017) studied that sematic (object presence, position, and size) and physical (consistent support and interposition with other objects) object-scene relationships have an impact on the detection of a target object within the image. After generating images in 3Ds Max, different images, which are similar to the real construction sites, were used as the background of the equipments. This process can generate more synthetic images from the same input similar to the actual images from construction sites and increase the accuracy of the computer vision training. If the training is done with white background images, in the implementation it causes inaccurate results. Figure 3-9 shows sample background images used in generation phase.

Figure 3-9 Sample images of construction background used in synthetic images

3.3.2.1 Environmental Conditions

In large scale image processing, sensitivity analysis is important to study the effect of the artificially generated images. Assuming that we have specific knowledge and information about equipment in construction sites. The customization can be considered to get more accurate results and make the synthetic images as close as possible to the real images. Since the process of generating synthetic images happens in the virtual environment, it is possible to apply sensitivity analysis. Two main factors are considered for synthetic image generation. The first factor is generating desired equipment in 3Ds Max, which is discussed in section 3.3.1, and the second factor is adding environmental condition to the images. Following our discussion on the effect of background images toward boosting the accuracy of model, weather and illumination conditions are considered to integrate as a background of image.

Weather Conditions: Usually, in a country like Canada, we experience different weather conditions during the year. Low visibility weather condition is an important factor on construction sites because it increases the risk of accidents. Low visibility, such as fog, mist, dust, freezing drizzle or snowy weather conditions can be considered as an influential reason that can affect the accuracy of the model. Therefore, based on different seasons of the year, different weather conditions can be added to the background of synthetic images to make them as real as possible. However, in this research we just considered snowy and dry weather. Figure 3-10 shows dry and snowy weather condition with an excavator working on the construction site.

(a) Dry weather condition (b) Snowy weather condition

Figure 3-10 Synthetic images of dry weather conditions of construction sites

Illumination Conditions: Depending on the construction schedule, equipment may work in the day or night shifts. Usually, heavy equipments start working in the morning and finish their job in the evening. The cameras monitor different objects at different time intervals, which might affect equipment detection. Therefore, considering time scheduling can help to adjust specific illumination and light in the synthetic images for the day shift and night shift. Figure 3-11 shows similar background images used for creating day and night time conditions.

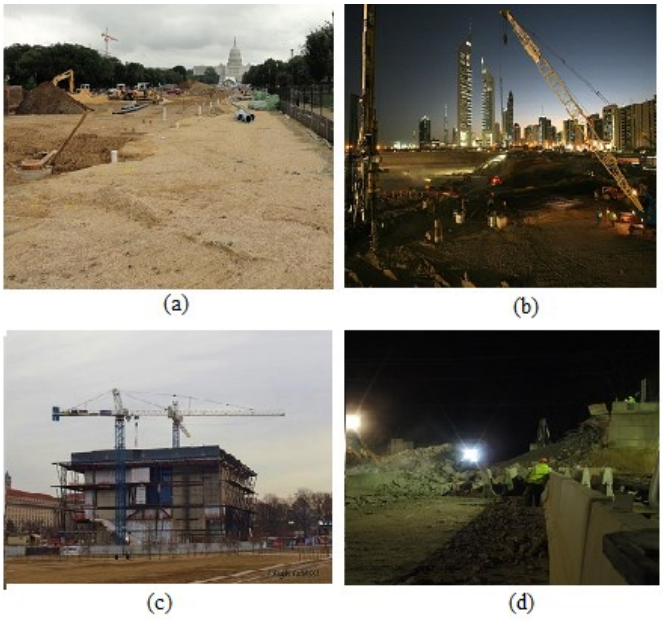


Figure 3-11 Day time and night time construction background sample images

The light condition and the position of sun over equipment may change constantly during the day. So, different light conditions are considered for equipment and their background. Having shadow on some parts of equipment can influence the accuracy of the model significantly if the model is just trained with images under perfect sunlight condition. Therefore, it is of a great importance to consider images with various light conditions. Figure 3-12 shows a sample of generated images with less light and more light over the equipment respectively.

(a) Less light (b) More light

Figure 3-12 Sample of generated synthetic images under different light condition

3.3.3 Annotation of Images

After generating images using 3D max, the raw images with single white color background are considered for the annotation process. The Sobel-filter is applied to the images to calculate the edge of the object and then change the RGB images to the gray image. Next, Dilate-filter is applied to reduce the black gaps. Soille (1999) applied an algorithm to fill out the empty parts in the surrounding edge, and connect the image boarder to pixels which are lighter than their surroundings. As mentioned in the Section 3.3.2, based on the information from construction sites, related background images can be added to the synthetic images. Figure 3-13 shows three generated synthetic images which is annotated and added to their backgrounds.

(a) (b) (c)

Figure 3-13 Generated synthetic images using various equipment types and backgrounds

3.4 Implementation and Case Study

3.4.1 Conventional Versus CNN-based Methods

The proposed framework was implemented in Matlab 9.1 (Mathworks, 2016) on a mobile station with Intel i7 Quad-Core processor, 32 gigabytes Random-Access Memory (RAM), and NVIDIA Quadro K2000M graphics card. As explained in Section 2.16, in this test, five main types of classifiers were investigated: Bag of Words (BoW), Local Binary Pattern (LBP), and HOG integrated with Support Vector Machine (SVM) as the conventional methods and AlexNet integrated with SVM, independent AlexNet, and independent Visual Geometry Group-face (VGG-f) as CNN-based methods. As shown in Table 3-4, one configuration with 500 clusters was used for extracting BoW features while four cell sizes for LBP features and six cell sizes for HOG features were tested. Three layers were selected from AlexNet considering the available computation resources in this research. Two layers were close to the end of the network and one layer was near the starting border of the network. Selecting the features from both end of the network helps to compare the effectiveness of the feature at different level of the CNN architecture. The features obtained from three different layers of the original AlexNet network were used separately for training the SVM classifiers. Due to the limitation of the computation capacity, only the first convolutional layer was tested while the other two layers were from the fully connected layers at the end of the network structure. The last two classifiers had similar structure to AlexNet and VGG-f, respectively, except the last layer (classification layer), which has three classes instead of 1000 classes in the original model. Also, in another scenario, AlexNet and VGG-f were fed and tested by gray scale images. For training each of the aforementioned classifier, the datasets of synthetic images with 6,000 images (first row) shown in Table 3-5 were used while for testing the dataset of real images with 1,169 images was used (second row). However, from the 7 generated poses for the excavator, only one pose (arm middle, boom middle, and bucket up) has been selected for testing and training.

The accuracies of correct detections for each class from the confusion matrix are shown in Table 3-4 in addition to the average accuracy of the all classes. The averages accuracy of BoW was 38%, while the best achieved average accuracies for LBP and HOG were 41% and 47%, respectively. The results show that all the conventional classifiers had more difficulties for classifying the loaders. Investigating the poor accuracies for the loader shows that the 3D model of the loader used for creating the synthetic images looks similar to a brand new loader which has usually a yellow bucket. However, the bucket of the real loaders looks very dark (close to black or dark brown color). Opposite to HOG-based

method, it is highly possible that CNN can be sensitive to the color of the object during training phase. On the other hand, the best average accuracies of AlexNet-SVM, AlexNet, and VGG-f were 83%, 78%, and 68%, respectively. It can be concluded from this analysis that CNN-based methods outperformed the conventional methods.

Table 3-4 Results of comparative analysis between conventional and CNN-based classifiers (Soltani et al. 2017)

Method	Size of Images	Additional Information	Accuracy (%)			
			Excavator	Loader	Truck	Average
Bag of Words	128×128	500 Clusters	78	3	33	38
LBP-SVM	128×128	4×4 Cells	72	3	39	38
	128×128	8×8 Cells	72	5	38	38
	128×128	16×16 Cells	76	5	41	41
	128×128	32×32 Cells	51	10	56	39
HOG-SVM	128×128	2×2 Cells	69	1	35	35
	128×128	4×4 Cells	70	1	39	37
	128×128	8×8 Cells	75	3	43	40
	128×128	16×16 Cells	73	5	61	46
	128×128	32×32 Cells	55	21	65	47
	128×128	64×64 Cells	41	45	35	40
AlexNet-SVM	227×227	conv 1 Layer	51	42	60	51
	227×227	fc7 Layer	76	78	93	83
	227×227	fc8 Layer	72	60	96	76
AlexNet	227×227	Colored	74	48	99	74
	227×227	Grayscale	95	60	78	78
VGG-f	224×224	Colored	70	40	94	68
	224×224	Grayscale	92	10	52	51

Table 3-5 Image datasets specifications (Soltani et al. 2017)

		Excavator	Loader	Truck	Total
Synthetic		2,000	2,000	2,000	6,000
Real		555	267	347	1,169
Real		100	100	100	300
Mixed	**Real**	555	267	347	2,338
	Synthetic	555	267	347	
Mixed	**Real**	100	100	100	600
	Synthetic	100	100	100	

3.4.2 Performance Evaluation of CNN-based Method on Different Datasets

At this stage, the best two groups of classifiers (trained for three classes of equipment) resulting from the previous test in Section 3.4.1 were used for a more detailed analysis and testing. The two groups are the AlexNet-SVM classifiers with 'fc7' layer and the independent AlexNet classifiers.

The purpose of this test is to find how the accuracy of each classifier can be affected using the five datasets shown in Table 3-5. There are two different datasets of real images, one with 1,169 images and the other with a small number of images (300 in total) to find the impact of the number of the training images. Also, two more datasets are prepared by adding the synthetic images with the same number of the real images in each of two previous datasets. The advantage of the synthetic images is that they are taken from all views around the equipment while the real images are mostly captured from low heights and views. A sample of real and synthetic images are shown in Figure 3-14. One the other hand, the synthetic images may look artificial and that may have negative effects on the CNN-based classifiers compared to conventional methods that did not show such effects on the accuracy (Soltani et al. 2016). Therefore, using the mixed datasets can evaluate the generality of the classifiers on different brands, colors, and views of the equipment.

Twenty scenarios are created and tested, which are shown in Table 3-6. Starting with AlexNet-SVM classifiers, the results of the classifiers trained by synthetic images and tested on real images are 83% and 86%. The accuracy achieved by classifying the dataset with smaller number of images shows better performance as expected. In the next scenarios, the larger dataset of the real images was used for training and the smaller datasets of real and mixed images were classified. The results show that adding synthetic images from various views reduces the accuracy. By using the two larger datasets of mixed images for the training, the accuracies were improved for both previously tested datasets. In the next comparison, the smaller dataset of real images was used to mimic the situations where there is a very limited number of available training images. This classifier is applied on two larger datasets of real and mixed images and the accuracies dropped significantly. In the last two scenarios, the smaller dataset of the mixed images was used for training and applied on the larger datasets of real and mixed images. It is clear from this test that we can improve the quality of training (and the resulting accuracy) by adding synthetic images to the training dataset when the number of real images is small.

In the next test, similar comparisons were repeated for evaluating the independent AlexNet CNN. The trend of the results is close to the previous test except in the scenario where the larger dataset of real images is used for training and applied on the smaller datasets of real and mixed images. Opposite to

the AlexNet-SVM classifier, the independent AlexNet outperformed on the smaller dataset of the mixed images compared to the smaller dataset of the real images. Additionally, using the larger dataset of mixed images shows its maximum performance on the smaller datasets of the real and mixed images. However, achieving the accuracy of 100% does not mean that this classifier is able to reach the same accuracy on the larger datasets and it simply proves that adding synthetic images to real images during the training can improve the performance of the AlexNet classifier.

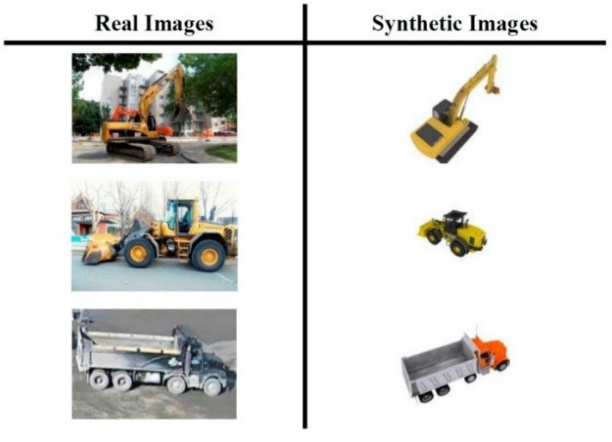


Figure 3-14 Sample of real and synthetic images (Soltani et al. 2017)

Table 3-6 Results of analysis on AlexNet-SVM and AlexNet classifiers (Soltani et al. 2017)

Method	Training Images		Testing Images		Accuracy (%)			
	Type	Number	Type	Number	Excavator	Loader	Truck	Average
AlexNet-SVM	Synthetic	6,000	Real	1,169	76	78	93	83
	Synthetic	6,000	Real	300	95	83	79	86
	Real	1,169	Real	300	89	100	95	94
	Real	1,169	Mixed	600	94	88	93	91
	Mixed	2,338	Real	300	94	100	96	97
	Mixed	2,338	Mixed	600	97	100	98	98
	Real	300	Real	1,169	54	66	100	73
	Real	300	Mixed	2,338	78	69	97	81
	Mixed	600	Real	1,169	83	90	99	90
	Mixed	600	Mixed	2,338	91	95	99	95
AlexNet	Synthetic	6,000	Real	1,169	74	48	99	74
	Synthetic	6,000	Real	300	95	51	97	81
	Real	1,169	Real	300	88	100	95	94
	Real	1,169	Mixed	600	99	94	97	97
	Mixed	2,338	Real	300	100	100	100	100
	Mixed	2,338	Mixed	600	100	100	100	100
	Mixed	600	Real	1,169	77	88	99	88
	Mixed	600	Mixed	2,338	88	94	100	94
	Real	300	Real	1,169	81	67	100	83
	Real	300	Mixed	2,338	100	71	96	89

3.4.4 Sensitivity Analysis

The purpose of this test is to find out how the accuracy of the AlexNet classifier can be affected by using a dataset containing different poses of an excavator with different background images. Therefore, seven generated excavator poses as shown in Section 3.3.1 with different background images such as snowy, dry, summer, night and day as mentioned in Section 3.3.2.1 were used to create the datasets shown in Table 3-7.

As shown in Table 3-7, there are two datasets with 4,000 and 2,000 synthetic images and one with 555 real images. Two different datasets of 2,500 and 1,110 mixed images are also prepared by adding synthetic images with the same number of real images in each. Seven scenarios are created and tested, as shown in Table 3-7. The result of AlexNet trained by 4,000 synthetic images of an excavator with different poses and backgrounds and tested on 555 real images shows the accuracy of 92.79%. The

accuracy achieved by training 2,000 synthetic images of the excavator on 555 and 300 real images of the excavator shows 94% and 100% respectively. However, we expect to achieve better accuracy by adding more synthetic images to the training dataset, but the result shows less accuracy on the large dataset of 4,000 synthetic images compared to the 2,000 synthetic images dataset. This can be due to overfitting and too much understanding of the training dataset. In the next scenarios, the dataset of 555 real images was used for training, and the datasets of 200 mixed images were used for testing and show 98% accuracy. By using the large dataset of 2500 mixed images for training and 555 real images for testing, the accuracy reached 99,82%. A mixed dataset of 1,110 training images, tested on 200 and 100 real images, shows 100 %accuracy for both. Comparing the results of the AlexNet classifier for the excavator from Table 3-6 and Table 3-7, we can conclude that adding synthetic images of the excavator from various poses with different backgrounds to the dataset can improve the performance of the AlexNet classifier.

Table 3-7 The results of sensitivity analysis on AlexNet classifier for the excavator

Method	Training Images		Testing Images		Accuracy (%)
	Type	Number	Type	Number	Excavator
AlexNet	Synthetic	4,000	Real	555	92.79
	Synthetic	2,000	Real	555	94
	Synthetic	2,000	Real	300	100
	Real	555	Mixed	200	98
	Mixed	2500	Real	555	99.82
	Mixed	1,110	Mixed	200	100
	Mixed	1,110	Real	100	100

3.5 Summary, Conclusions and Future Work

This section examined a new method of creating and annotating synthetic images using 3D equipment models to identify images of construction sites.

By reviewing the results of this case study, it can be concluded that the automatic annotation method using synthetic images can produce the same results of real images of construction sites as a dataset for training and testing purposes. By using this method, any construction equipment can be created and applied for training of vision-based detection of construction equipment on construction sites. This method is also not expensive and can save a lot of time, as there is no need for physical photography

of equipment on sites. The synthetic images created on the 3D model of construction equipment can be used to train the model, which helps to find the object.

This study examined the applicability of CNN-based methods while comparing them with conventional descriptors. In the first analysis, BoW, LBP, HOG, AlexNet were investigated independently and integrated with SVM and VGG - f networks. In the second test, AlexNet and AlexNet - SVM were compared using different image datasets, including real, synthetic and mixture of real and synthetic images with different numbers of images in the training and testing datasets The results show that it is better to add synthetic images to the real images for the training of classifiers, especially when there is a limited number of real images available. Including synthetic images in the real images, the classifier takes into account various views of the target objects that are not available in the real image dataset. In addition, by applying sensitivity analysis, we can conclude that adding different equipment poses with different background images can significantly improve the model's accuracy. However, training and applying of CNN methods requires a very powerful computer configuration, especially when developing a very deep network.

As a limitation and future work for this study, we can point to the mismatching of generated synthetic images with their background, and calculation costs. In this study, for adding a background to the annotated images, some mismatching between the background and annotated image happened. The reason is that some background images are taken from street level and synthetic images are generated with different viewing angles. Sometimes, attaching these two together can look far from reality and cause the synthetic images to be mismatched. However, Furht (2011) proposed the Augmented Reality technique to match the viewing angle of the synthetic images and the background. Moreover, due to computation cost, only a small sample of generated synthetic images (6,000) used for training and testing.

CHAPTER 4 Comparison of the performance of TensorFlow object detection models for the detection of personal protective equipment on construction sites

4.1 Introduction

In the previous chapter, we discussed the performance of conventional versus CNN-based methods as means for the classification of equipment on construction sites using synthetic images. In this chapter, we will use the state-of-the-art CNNs techniques for object detection of Personal Protective Equipments (PPEs). The goal is to find the best CNN architecture available for the detection of PPEs on construction sites. The open source TensorFlow Object Detection API is used to facilitate the building, training, and deployment of object detection models. There are generally four ways to improve deep learning performance, such as (1) improve performance with data, (2) improve performance with algorithms, (3) improve performance with algorithm tuning, and (4) enhance the performance with ensemble method.

This chapter aims to achieve the following objectives: (1) improving the performance by customizing the dataset; and (2) Improve detection performance through fine tuning and compare different object detection algorithms from TensorFlow.

4.2 Proposed Method

The main goal of this section is to evaluate and compare existing object detection methods in construction site scenarios. Three classes of objects are considered as the target of this study for detection on construction sites, namely safety vests, workers with hardhats, and workers without hardhats. The methods selected here for the said evaluation and comparison have already shown promising detection performance within the computer vision community. Results and findings from this research are expected to help improving construction safety. The overall steps employed for the evaluation and comparison are illustrated in Figure 4-1.

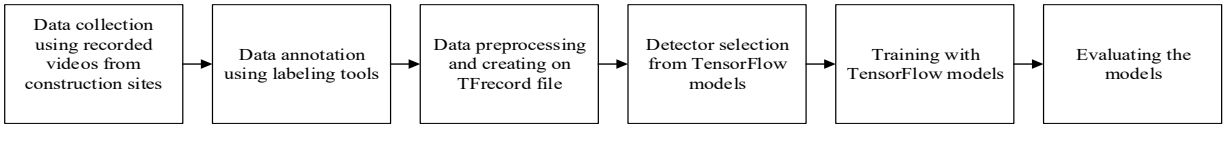


Figure 4-1 The overall steps for object detection

4.2.1 Data Collection

The first stage of data collection was the recording of videos from construction sites, such as civil infrastructures, residential buildings, and municipal facilities. The frames that showed objects of interest were extracted from the series of recorded videos. Table 4-1 shows the number of collected frames and their corresponding classes. This datasets were used in various training steps.

Table 4-1 Dataset Specification

Total number of frames	Number of frames per classes		
	Hard Hat	Safety Vest	Without Hard Hat
30,500	13,000	9,500	8,000

The deep learning implementation requires much more training data than the basic machine learning algorithms. Throughout the collection of our object detection dataset, various challenges arose. One of the major challenges was the collection of common objects suitable for the dataset considering the quality and clarity of images. Yet another challenge encountered during data acquisition was the collection of wide-ranging sets of images of the same object.

In the videos used to build the input datasets, there were one or many objects of the three classes. Figure 4-2 presents some representative images from our dataset showing: (a) workers with hardhats and safety vests, and (b) workers without hardhat. The models need to be trained to identify the PPEs of the construction workers within various construction sites.

(a) Workers with hardhat and safety vests (b) Worker without hardhat

Figure 4-2 Sample images of different objects of interest for detection purposes

4.2.2 Data Annotation

Data annotation is the process of defining the bounding box and labeling the data for supervised machine learning technique. The Labelme image annotation tool was used to annotate the images by the third party company. However, manually annotating all objects and their instances with the corresponding class labels could be considered as a drawback image annotation. The information of the images and corresponding boxes were stored in a file and saved in the Extensible Markup Language (XML) format. The XML file created using the Labelme tool contains the image file with the information of the rectangular boxes surrounding the objects and the corresponding class label of each object. Finally, all annotations were stored in a Comma-Separated Values (CSV) file. Figure 4-3 shows an image with the bounding boxes around objects of interest in the image. Different classes of objects of interest are shown in differentially colored bounding box.

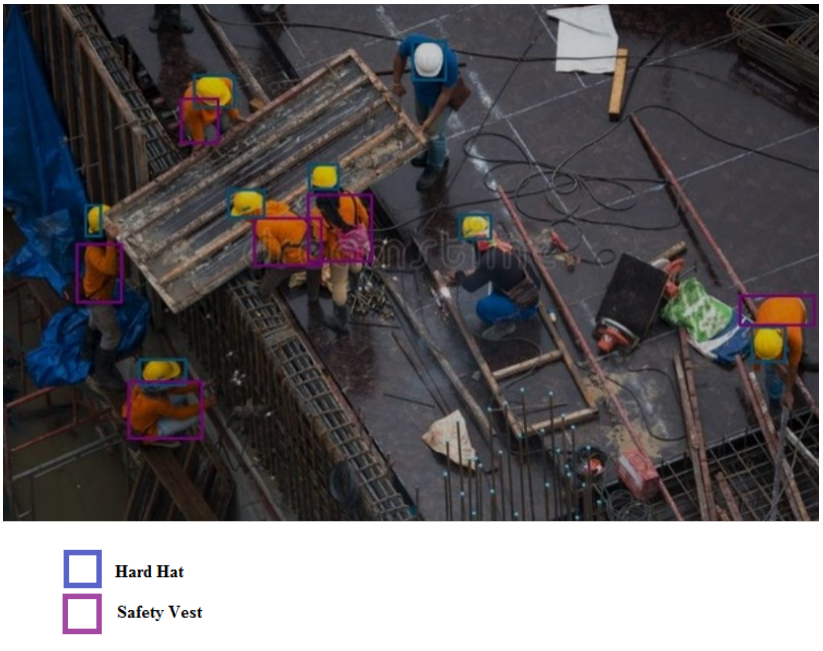


Figure 4-3 Sample image of multi-class object annotation with Labelme (Screenshot)

4.2.3 Data Preprocessing

After all the frames captured from the video and then were annotated, the new annotated dataset can be used for further processing. The dataset has to be divided into the training and testing subsets. In this study, the dataset was split randomly into 80-20. The next step is to change the annotated CSV file format into a format which can be supported by the TensorFlow library.

The TensorFlow library uses its native file format TFrecord (.record) to perform the batch operation. While some platforms do most of the batch processing directly from the images, TensorFlow uses a single file for the batch operation. Images in TFrecord files are transformed into Numpy (Numerical Python library) arrays. The TFrecord files facilitate the processing of large datasets to match the network architecture. This file format is the record-oriented binary format which is used in many TensorFlow applications that handle training and testing datasets.

As shown in Figure 4 - 4, the first step in the process of creating TFrecord file is to convert images and related labels to the appropriate data type (labeled images). Next, this data type is converted into feature sets. Then, the feature sets are transformed into serialized data, and ultimately written into .TFRecord file format using the TFRecordWriter function.

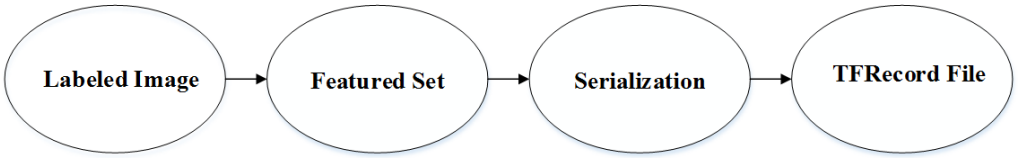


Figure 4-4 The process of creating TFRecord file

4.2.4 Detector Selection

In this study, three different models from the TensorFlow object detection model zoo were selected, namely SSD-MobileNet-V1-coco, Faster-RCNN-ResNet101-coco, and RFCN-ResNet101-coco.

These models were analyzed on our collected dataset of PPEs. Table 4-2 shows the list of models amongst the existing ones that we used in our research. The structure of each model and their feature extractors are discussed in Sections 2.11 and 2.12. Other useful properties, such as speed of execution, mean average precisions (mAP), and output type are provided in Table 4-2. The speed of execution was measured in milliseconds (ms), showing runtime in ms per 600x600 image, as shown in Table 4 - 2. The mean average precision was used to calculate the detector performance as reviewed in Section 2.13.1 and shown in Table 4-2. In addition, the default fine - tuned and configuration file for each model provided by the TensorFlow model zoo was used for training different models in this case study. Here, higher values of mAP are better, and we only report bounding box mAP rounded to the nearest integer. Boxes output were used to show the final results.

Table 4-2 List of models used in this research (Lin et al. 2014)

Model Name	Speed (ms)	mAP	Output
SSD-MobileNet-V1-coco	30	21	Boxes
Faster-RCNN-ResNet101-coco	106	32	Boxes
RFCN-ResNet101-coco	92	30	Boxes

4.2.5 Environment Requirements and Specifications

The system requirements necessary to work with object detection models may vary depending on the model, operating system, and list of libraries one may be using. The main requirements to train and implement the TensorFlow object detection API are explained in this section. The following list shows the libraries and devices that are required for the object detection using TensorFlow object detection API.

- Python, version 3.6
- Open Computer Vision (OpenCV) libraries
- Lebelme annotation tools
- CUDA library
- CUDA Deep Neural Network (cuDNN) libraries
- TensorFlow Object Detection API libraries
- Pre-trained Model Zoo Object Detection Models
- Computing Source to Train Deep Learning

All of our experiments were done on a local computer using 1 GPU model GeForec GTX 1070, 12 CPUs, and 32 GB Memory. Moreover, TensorFlow GPU version 1.10.0 was installed on the local machine and the object detection API library was configured on top of it.

4.3 Tensorflow Object Detection API

Pre-trained models have been fine-tuned for our dataset using manually labeled images for the PPEs of the workers on construction sites. Three pre-trained models for object detection namely, SSD-MobileNet-V1, Faster RCNN-ResNet101, and RFCN-ResNet101 were selected as shown in Table 4-2. A provided configuration file was used as a basis for the model configuration.

4.3.1 Object Detection with SSD-MobileNet-V1

The pre-trained SSD-MobileNet-V1 model is fine-tuned with the training and testing datasets as discussed in Section 4.2.2 and 4.2.3. The provided checkpoint from configuration file for SSD-MobileNet-V1 was used as a starting point for the fine-tuning process. The configuration file also needs training record and test record file locations. Training record is a TFRecord file created for the training dataset, and a test record is a file created for testing dataset. In addition to the configuration file, a label map object file is required. The structure of an object label map file is shown in the Figure 4-5. The first item refers to the workers without hardhat (NOHH), the second one shows workers with hardhat (HardHat) and the last one shows the workers with High Visibility Vest (HIVIS).

```
item {
    id: 1
    name: 'NOHH'
}
item {
    id: 2
    name: 'HARDHAT'
}
item {
    id: 3
    name: 'HIVIS'
}
```

Figure 4-5 the structure of label map file

The training record and testing record file locations are added along with the object label map file in the configuration file. The configuration file is also updated with the number of classes which is three in our case.

The SSD model is fine-tuned for 65,000 steps as shown in Figures 4-6. Figure 4-6 shows the decline in total loss throughout the process of fine-tuning. The total loss values are added to the training log files after a particular interval of time. The log files are then passed as input to Tensorboard, a visualization tool to plot the computational graphs. All the plots presented below are captured from Tensorboard visualizations. As shown in Figure 4-6, the total loss decreases rapidly and indicates different values. The values reported in faded orange are the actual total loss values, while the darker orange is obtained by 0.9 smoothing.

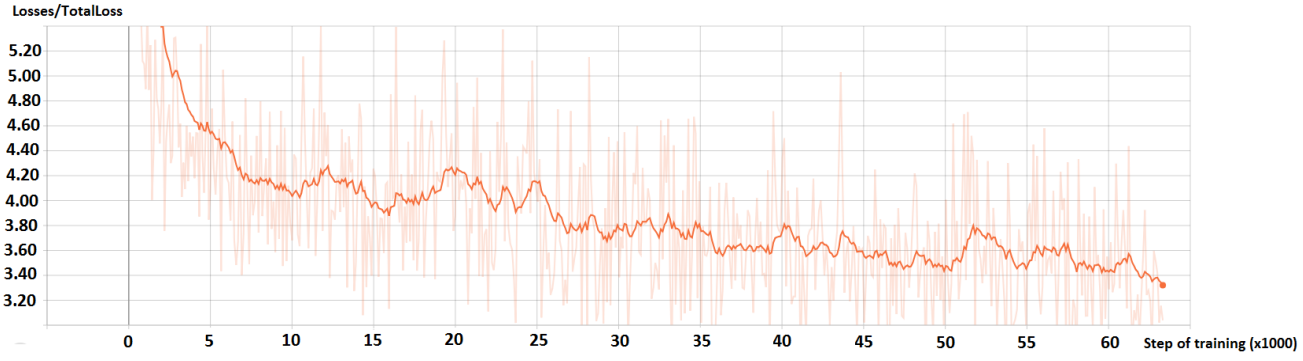


Figure 4-6 Total loss decrease with fine-tuned SSD-MobileNet-V1 (TensorBoard Screenshot)

4.3.2 Object Detection with Faster-RCNN-ResNet101

In this section, the Faster-RCNN- ResNet101 is fine-tuned to the previously created datasets of training and testing. We have selected the provided configuration file as a basis similar to fine-tuning of SSD-MobileNet-V1. We have also updated the number of classes, added training and testing record places as well as the location of the object label map.

As shown in Figures 4-7, the Faster RCNN model is fine-tuned for 65,000 steps. Figure 4-7 shows the declining trend of total loss during the training process. The total loss values show a fluctuating trend as the batch size is 1, and a loss for each iteration can be slightly different from the previous iteration. However, the main point is that the losses decreased overall during training.

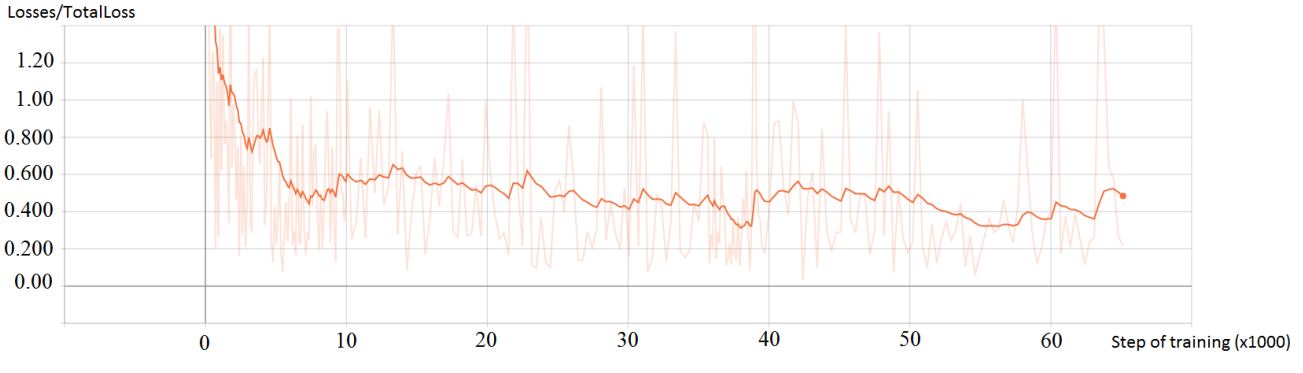


Figure 4-7 Total loss decrease with fine-tuned Faster-RCNN-ResNet101 (TensorBoard Screenshot)

4.3.3 Object Detection with RFCN-ResNet101

This section discusses the fine-tuning of the RFCN-ResNet101 model. Like the fine tuning of the SSD and Faster RCNN model, we also update RFCN training and testing locations, object label map, and class number. Similar to Faster RCNN and SSD, the RFCN model is also trained for 65,000 steps as shown in Figures 4-8. Figure 4-8 shows the total loss decline over the whole tuning process. The total loss shows a general reduction trend.

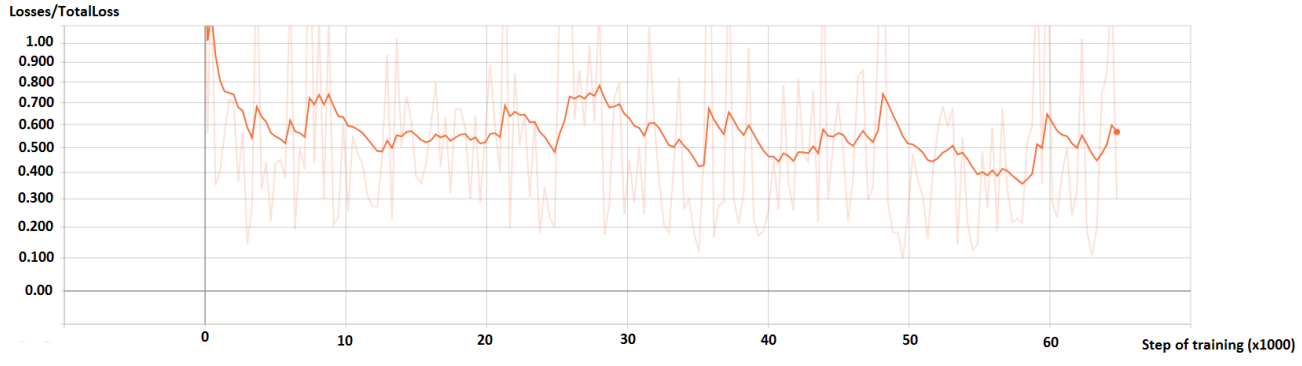


Figure 4-8 Total loss decrease with fine-tuned RFCN-ResNet101 (TensorBoard Screenshot)

4.4 Results

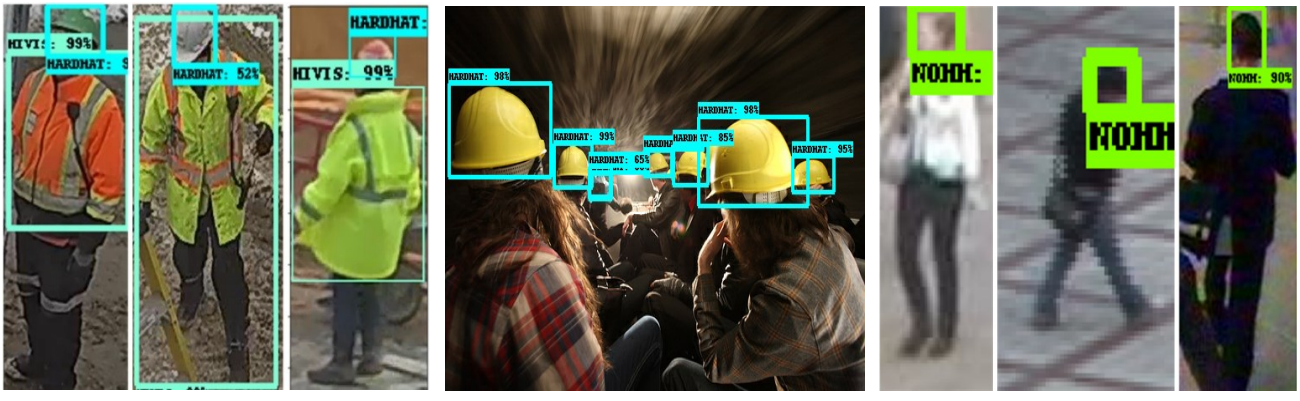
In this section, we compare the performance of fine-tuned model based on AP and mAP. Each model has been tested on 1000 images containing the PPEs. Tables 4-3 presents the performance of each category and mean Average Precision values for the fine-tuned model at 65,000 steps of training for three classes of PPEs.

Table 4-3 The results of comparative analysis on TensorFlow object detection models

Model	AP			mAP
	Hardhat	Vest	No Hardhat	
SSD-MobileNet-V1	0.9225	0.9195	0.9020	0.9147
Faster-RCNN-ResNet-101	0.9612	0.9021	0.9471	0.9368
RFCN-ResNet-101	0.9618	0.8749	0.9514	0.9294

For SSD-MobileNet-V1, trained up to 65,000 steps, the AP are 0.9225, 0.9195, and 0.9020 respectively for hardhat, vest, and no hardhat. Like SSD model, we also compared the AP values of the Faster-RCNN-ResNet-101 fine-tuned model for three classes of PPEs. The fine-tuned model is trained in 65,000 steps as well. The AP are 0.9612, 0.9021, and 0.9471 respectively for hardhat, vest,

and no hardhat for Faster-RCNN model. The last model is RFCN-ResNet-101 which we have AP of 0.9618, 0.8749, and 0.9514 respectively for hardhat, vest, and no hardhat. The value in mAP is also 0.9147 for SSD-Mobilenet-V1, 0.9368 for Faster-RCNN-ResNet-101, and 0.9294 for RFCN-ResNet-101. Moreover, Figure 4-9 shows detected object with the above mentioned models.



(a) SSD-MobileNet-V1 (b) Faster-RCNN-ResNet-101 (c) RFCN-ResNet-101

Figure 4-9 The results of detected objects with mentioned models

4.5 Discussion

In this section the results of the object detection models in the previous section are discussed. By evaluating different criteria of detectors, it is not easy to choose the best detection model from the list of models. Moreover, other criteria such as speed have significant impacts on the object detection operation. Table 4-4 shows the speed of each object detector in seconds per frame (SPF). From the Table 4-4 we can conclude that the time of inference for the SSD model is the least, i.e., the SSD model is the fastest among all the models. Moreover, as expected, the mAP value is the highest for Faster RCNN which makes it the most accurate of all. Additionally, Faster RCNN is approximately 2% more accurate than the SSD model. We can conclude that the Faster RCNN is superior in terms of mAP of object detection, although this model is computationally heavy compared to the SSD model. The RFCN architecture has a quite good accuracy only about 1% less than Faster RCNN. However, the speed of testing for RFCN is less than Faster RCNN and more than SSD. The detection speed for Faster RCNN is about six times higher than SSD and four times higher than RCNN, which leads to a very heavy model to detect objects in real time. Straightforward deployment capability on a mobile and embedded device is a huge advantage of the SSD model.

Table 4-4 Results of the testing speed and accuracy for three fine-tuned models

Models	Testing time in second		Testing Speed vs. Accuracy	
			Frame per Second	mAP
SSD-MobileNet-v2	540	540S/1000	0.54	0.9147
FRCNN-ResNet-101	6,480	6,480S/1000	6.48	0.9368
RFCN-ResNet-101	2,880	2,880S/1000	2.88	0.9294

4.6 Summary and Conclusion

In this study, we investigated the performance of the common object detection models obtained from the TensorFlow object detection model zoo when used to detect PPEs. Different object detection models were trained and tested on the custom dataset. A diverse range of the object datasets from the construction sites were gathered and labelled. The models were trained, and their performance was evaluated using accuracy metrics by reviewing the above-mentioned model systems in object detection.

We conduct tests on Faster RCNN, RFCN and SSD architectures combined with MobileNet and ResNet101 feature extractors to find the best model for object detection. The annotation and preprocessing part is another crucial part of the object detection process and must be done carefully.

To wrap up this study, we can say that the Faster RCNN architecture with the ResNet101 feature extractor has better results with regards to AP, mAP, and considering its high computation cost. The SSD MobileNet compared to the Faster RCNN ResNet101 is faster, but less accurate. The RFCN model has good mAP but less accurate than Faster RCNN. The SSD MobileNet model could be a suitable model to be used on mobile devices but the performance may not satisfying in terms of accuracy of detection.

CHAPTER 5 CONCLUSION AND FUTURE WORKS

5.1 Summary of Research

Object detection technologies are becoming increasingly important at modern construction sites and could be used in productivity analysis, material detection and safety monitoring. However, few efforts have been made to evaluate the accuracy and robustness of these object detection methods in construction scenarios. This study proposed a new method for generating synthetic images of construction equipment under various conditions. Several equipment poses, colors and equipment types were generated. Different weather conditions attached to the synthetic images as a background. These methods helped to reduce the time and cost of preparing large datasets. Generating synthetic images of construction equipment also enhanced the performance and accuracy of object detectors trained by synthetic images. In addition, this study proposed a comparative research of the 2D visual object detection technique in the construction environment. For comparison purposes, various object detection methods from the computer vision community were selected. These methods have been tested on images that include various building resources of interest, such as excavators, loaders, trucks, hard hats, safety jackets and hard hat workers. All images were annotated and labeled manually to build the ground truths and characterized by quantitative and detailed methods of object detection.

5.2 Research Contributions

The contribution of this research are as follows:

(1) Providing a new method for generating and annotating construction equipment using their 3D models and real images of construction sites. This method helped in providing image datasets in a short time and inexpensively. With regard to this contribution, the following conclusions can be drawn:

- ✓ Various equipment, size, color, poses, and types of construction equipment considered for the synthetic image generation.
- ✓ Different background conditions, such as weather, lighting, day or night time considered as the background of synthetic images.
- ✓ The synthetic images were annotated automatically using the proposed method.

(2) Comparison of the applicability of CNN-based methods with conventional descriptors.

- ✓ CNN- based method clearly outperformed the conventional methods.

- ✓ AlexNet and AlexNet-SVM were compared using different images datasets including real, synthetic, and mixture of real and synthetic images with different numbers of images in the training and testing dataset.
- ✓ Mixture of real and synthetic images for training of the classifier, especially when there is a limited number of real images available, significantly improved classifier accuracy.
- ✓ Including synthetic images in the real images, the classifier takes into account various views of the target objects that are not available in the real image dataset.

(3) Sets of pre-trained detection models of COCO dataset were used to detect PPE's on construction sites. The following conclusions were reached:

- ✓ Customize the algorithms of TensorFlow object detection model zoo for custom object detection
- ✓ Detecting PPE's of workers on 2D images of construction sites.

5.3 Limitation and Future Work

This research experiment is a step towards the real-time object detection of different class objects using surveillance cameras on construction sites. The performance of the detectors on datasets, videos and the mobile implementation motivates further research. Future studies can consider pixel-wise segmentation as a replacement for the bounding box annotation method of objects in the dataset. This could provide greater accuracy than the existing bounding box annotation technique.

The amount of data in our dataset is not adequate and is imbalanced to simplify all collected object classes. To achieve a better generalization result, additional images are required. Automatic annotation could be an appropriate attempt to decrease the time limit in the collection of the labeled dataset for data annotation. More images can be collected and trained to increase the total accuracy of detectors. Different object detection architectures can be studied for training and evaluation of the functionality of detectors. Tracking real-time objects, counting objects of interest and locating objects using other deep learning algorithms could be other opportunities to work on in the future. Two main concerns in real-time object detection are the accuracy of detection and computational complexity. Therefore, we can conduct more research on the detection model architecture.

While this research has successfully achieved its objectives, the following limitations remain to be addressed in the future:

- ✓ Although the PPE's detectors have achieved mean average precision of 0.9368, the final results of the test on real ongoing construction projects show that the detectors can perform less than their best performances. This can happen because of occlusion between objects or poor camera quality on the construction site.
- ✓ Some background images are taken from street level and synthetic images are generated with different viewing angles. Sometimes, attaching these two together may look far from reality and cause the synthetic images to be mismatched. However, Furht (2011) proposed the Augmented Reality technique to match the viewing angle of the synthetic images and the background.

REFERENCES

Abdelhamid, T. S., & Everett, J. G. (2000). Identifying root causes of construction accidents. *Journal of construction engineering and management*, 52-60.

ADHIKARI, B. P. (2018). *CAMERA BASED OBJECT DETECTION FOR INDOOR SCENES*. Retrieved from https://medium.com/@14prakash

ADHIKARI, B. P. (2018). *CAMERA BASED OBJECT DETECTION FOR INDOOR SCENES.* Tampere University of Technology.

Authors, T. T. (2017). *TenserFlow Object Detection API*. Retrieved from https://github.com/TensorFlow/models/tree/master/research/object_detection

Barron, J. L., Fleet, D. J., & Beauchemin, S. S. (1994). Performance of optical flow techniques. *International journal of computer vision*, 43-77.

Biederman, I. (2017). On the semantics of a glance at a scene. *In Perceptual organization*, pp. 213-253.

Challet, F. (2017). *Deep Learning with Python.* New York: Manning Publication.

Chao, L. C., & Skibniewski, M. J. (1994). Estimating construction productivity: Neural-network-based approach. *Journal of Computing in Civil Engineering*, 234-251.

Chitkara, K. K. (1998). *Construction project management.* Tata McGraw-Hill Education.

Choy, E., & Ruwanpura, J. Y. (2006). Predicting construction productivity using situation-based simulation models. *Canadian Journal of Civil Engineering*, 1585-1600.

Dai, J., Li, Y., He, K., & Sun, J. (2016). R-fcn: Object detection via region-based fully convolutional networks. *Advances in neural information processing systems*, 379-387.

Dai, J., Li, Y., He, K., & Sun, J. (2016). R-fcn: Object detection via region-based fully convolutional networks. *Advances in neural information processing systems*, (pp. 379-387).

Dalal, N., & Triggs, B. (2005). Histograms of oriented gradients for human detection. *Computer Vision and Pattern Recognition.* San Diego, CA, USA, USA: IEEE.

Dissanayake, M., Robinson Fayek, A., Russell, A. D., & Pedrycz, W. (2005). A hybrid neural network for predicting construction labour productivity. *In Computing in Civil Engineering*, 1-12.

Du, S., Shehata, M., & Badawy, W. (2011). Hard hat detection in video sequences based on face features, motion and color information. *Computer Research and Development (ICCRD), 2011 3rd International Conference* (pp. 25-29). IEEE.

Edwin Sawacha, Shamil Naoum, Daniel Fong. (1999). Factors affecting safety performance on construction sites. *International Journal of Project Management*, 309-315.

Everingham, M., Eslami, S. A., Van Gool, L., Williams, C. K., Winn, J., & Zisserman, A. (2015). The pascal visual object classes challenge: A retrospective. *International journal of computer vision*, 98-136.

Everingham, M., Van Gool, L., Williams, C. K., Winn, J., & Zisserman, A. (2010). he pascal visual object classes (voc) challenge. *International journal of computer vision*, 303-338.

F., C. (2017). *Deep Learning with Python.* Greenwich, CT, USA: Manning Publications Co.

Felzenszwalb, P. F., & Huttenlocher, D. P. (2004). Efficient graph-based image segmentation. *International journal of computer vision*, 167-181.

Felzenszwalb, P. F., & Huttenlocher, D. P. (2005). Pictorial structures for object recognition. *International journal of computer vision*, 55-79.

Fuentes, A., Yoon, S., Kim, S. C., & Park, D. S. (2017). A robust deep-learning-based detector for real-time tomato plant diseases and pests recognition.

Fuentes, A., Yoon, S., Kim, S. C., & Park, D. S. (2017). *A robust deep-learning-based detector for real-time tomato plant diseases and pests recognition.* Sensors, 17(9), 2022.

Furht, B. (2011). *Handbook of augmented reality* (Vol. Vol. 71). New York: Springer.

G. B. Team. (2018). *About tensorflow.* Retrieved from https://www.tensorflow.org

Geiger, A., Lenz, P., & Urtasun, R. (2012). Are we ready for autonomous driving? the kitti vision benchmark suite. *Computer Vision and Pattern Recognition* (pp. 3354-3361). IEEE.

Geiger, A., Lenz, P., & Urtasun, R. (2012). Are we ready for autonomous driving? the kitti vision benchmark suite. *In Computer Vision and Pattern Recognition.* IEEE.

Georgios Zoumpourlis, Alexandros Doumanoglou, Nicholas Vretos, Petros Daras. (2017). Non-linear Convolution Filters for CNN-based Learning. *IEEE International Conference*, (pp. 4771-4779).

Gevers, T., & Stokman, H. (2004). Robust histogram construction from color invariants for object recognition. *IEEE transactions on pattern analysis and machine intelligence*, 113-118.

Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2016). Region-based convolutional networks for accurate object detection and segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 142-158.

Gluon. (2017). *Object detection using convolutional neural networks*. Retrieved from http://gluon.mxnet.io/chapter08_computer-vision/

Gomez, V. V. (2015). *Object Detection for Autonomus Driving Using Deep Learning.* UNIVERSITAT POLITÈCNICA DE CATALUNYA.

Gong, J., & Caldas, C. H. (2011). An object recognition, tracking, and contextual reasoning-based video interpretation method for rapid productivity analysis of construction operations. *Automation in Construction*, 1211-1226.

Grau, D., Caldas, C. H., Haas, C. T., Goodrum, P. M., & Gong, J. (2009). Assessing the impact of materials tracking technologies on construction craft productivity. *Automation in construction*, 903-911.

Han, S., Lee, S., & Peña-Mora, F. (2010). Framework for a resilience system in safety management: a simulation and visualization approach. *Computing in Civil and Building Engineering, Proceedings of the International Conference.*

Han, S., Lee, S., & Peña-Mora, F. (2011). Application of dimension reduction techniques for motion recognition. *Computing in Civil Engineering*, 102-109.

Haykin, S. (2009). *Neural Networks and Learning Machine.* New York: Pearson Prentic Hall.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. *Proceedings of the IEEE conference on computer vision and pattern recognition*, (pp. 770-778).

Hinze, J., & Godfrey, R. (2003). An evaluation of safety performance measures for construction projects. *Journal of Construction Research*, 5-15.

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., ... & Adam, H. (2017). *Mobilenets: Efficient convolutional neural networks for mobile vision applications.* arXiv preprint arXiv:1704.04861.

Huang, J., Rathod, V., Sun, C., Zhu, M., Korattikara, A., Fathi, A., Murphy, K. (2017). Speed/accuracy trade-offs for modern convolutional object detectors. *In IEEE CVPR.*

Huang, J., Rathod, V., Sun, C., Zhu, M., Korattikara, A., Fathi, A., ... & Murphy, K. (2017). Speed/accuracy trade-offs for modern convolutional object detectors. *IEEE CVPR.*

Ian Goodfellow and Yoshua Bengio and Aaron Courville. (2016). *Deep Learning.* MIT Press.

J.R. Schott, S. B. (1999). An Advanced Synthetic Image Generation Model and its Application to Multi/Hyperspectral Algorithm Development. *Canadian Journal of Remote Sensing*, 99-111.

Jain, R., Kasturi, R., & Schunck, B. G. (1995). *Machine vision.* New York: McGraw-Hill.

Joseph Redmon, Ali Farhadi. (2018). *An Incremental Improvement*. Retrieved from arXiv:1804.02767v1.

Kanade, H. S. (2000). A statistical method for 3D object detection applied to faces and cars. *Proceedings IEEE Conference on Computer Vision and Pattern Recognition.* Hilton Head Island, SC, USA: IEEE.

Katsaggelos, A. K. (2012). *Digital Image Restoration.* Springer Publishing Company.

Kim, H., Bang, S., Jeong, H., Ham, Y., & Kim, H. (2018). Analyzing context and productivity of tunnel earthmoving processes using imaging and simulation. *Automation in Construction*, 188-198.

Kim, P. (2017). *MATLAB Deep Learning: With Machine Learning, Neural Networks and Artificial Intelligence.* New York: Apress.

Li, C., & Lee, S. (2011). Computer vision techniques for worker motion analysis to reduce musculoskeletal disorders in construction. *Computing in Civil Engineering*, 380-387.

Li, H., Chen, Z., Yong, L., & Kong, S. C. (2005). Application of integrated GPS and GIS technology for reducing construction waste and improving construction efficiency. *Automation in Construction*, 323-331.

Li, S. Z., Zhu, L., Zhang, Z., Blake, A., Zhang, H., & Shum, H. (2002). Statistical learning of multi-view face detection. *In European Conference on Computer Vision* (pp. 67-81). Springer, Berlin, Heidelberg.

Lin, T. Y., Goyal, P., Girshick, R., He, K., & Dollár, P. (2018). Focal loss for dense object detection. *IEEE transactions on pattern analysis and machine intelligence*.

Lin, T. Y., Goyal, P., Girshick, R., He, K., & Dollár, P. (2018). Focal loss for dense object detection. *IEEE transactions on pattern analysis and machine intelligence*.

Lin, T. Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., ... & Zitnick, C. L. (2014). Microsoft coco: Common objects in context. *European conference on computer vision* (pp. 740-755). Springer, Cham.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y., & Berg, A. C. (2016). Ssd: Single shot multibox detector. *European conference on computer vision* (pp. 21-37). Springer, Cham.

Murray, S. (2017). Real-Time Multiple Object Tracking-A Study on the Importance of Speed.

Navon, R., & Shpatnitsky, Y. (2005). Field experiments in automated monitoring of road construction. *Journal of Construction Engineering and Management*, 487-493.

Ni, W., Collings, I. B., Liu, R. P., & Chen, Z. (2014). Relay-assisted wireless communication systems in mining vehicle safety applications. *IEEE Transactions on Industrial Informatics*, 615-627.

Oliva, Aude, and Antonio Torralba. (2007). The role of context in object recognition. *Trends in cognitive sciences, 11*(12), 520-527.

Park, M. W., & Brilakis, I. (2012). Construction worker detection in video frames for initializing vision trackers. *Automation in Construction*, 15-25.

Peddi, A., Huan, L., Bai, Y., & Kim, S. (2009). Development of human pose analyzing algorithms for the determination of construction productivity in real-time. *In Construction Research Congress*, 11-20.

Piccardi, M. (2004). Background subtraction techniques: a review. *IEEE international conference*, 3099-3104.

R. B. Girshick. (2015). *Fast R-CNN.* [Online]. Available: http://arxiv.org/abs/1504.08083.

Raman Maini, H. A. (2010). A Comprehensive Review of Image Enhancement. *JOURNAL OF COMPUTING, VOLUME*, 8-11.

Rasmussen, C. B., Nasrollahi, K., & Moeslund, T. B. (2017). R-FCN Object Detection Ensemble based on Object Resolution and Image Quality. *International Joint Conference on Computational Intelligence* (pp. 110-120). SCITEPRESS Digital Library.

Rasmussen, C. B., Nasrollahi, K., & Moeslund, T. B. (2017). R-FCN Object Detection Ensemble based on Object Resolution and Image Quality. *International Joint Conference on Computational Intelligence* (pp. 110-120). SCITEPRESS Digital Library.

Redmon, J. &. (2018). *An incremental improvement.* arXiv preprint arXiv:1804.02767.

Reese, C. D., & Eidson, J. V. (2006). *Handbook of OSHA construction safety and health.* Crc Press.

Reinhard, E., Adhikhmin, M., Gooch, B., & Shirley, P. (2001). Color transfer between images. *IEEE Computer graphics and applications*, 34-41.

Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, (pp. 91-99).

Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, (pp. 91-99).

Ren, S., He, K., Girshick, R., & Sun, J. (2017). Faster R-CNN: towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 1137-1149.

Rezazadeh Azar, E., & McCabe, B. (2011). Automated visual recognition of dump trucks in construction videos. *Journal of Computing in Civil Engineering*, 769-781.

Ruderman, D. L., Cronin, T. W., & Chiao, C. C. (1998). Statistics of cone responses to natural images: implications for visual coding. *JOSA A*, 2036-2045.

S. Ren, K. He, R. B. Girshick, and J. Sun,. (2015). Faster R-CNN: towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, (pp. 91-99).

Sawacha, E. N. (1999). Factors affecting safety performance on construction sites. *International journal of project management*, 309-315.

Schiffbauer, W. (2002). Active proximity warning system for surface and underground mining applications . *National Institute for Occupational Safety and Health.* Pittsburg, PA: NIOSHTIC-2 No. 20021434.

Schmid, C., & Mohr, R. (1997). Local grayvalue invariants for image retrieval. *IEEE transactions on pattern analysis and machine intelligence*, 530-535.

Seo, J., Han, S., Lee, S., & Kim, H. (2015). Computer vision techniques for construction safety and health monitoring. *Advanced Engineering Informatics*, 239-251.

Shotton, J., Sharp, T., Kipman, A., Fitzgibbon, A., Finocchio, M., Blake, A., ... & Moore, R. (2013). Real-time human pose recognition in parts from single depth images. *Communications of the ACM*, 116-124.

Soltani, M. M. (2017, June). Excavator Pose Estimation for Safety Monitoring by Fusing Computer Vision and RTLS Data. *Doctoral dissertation, Concordia University*. Montreal, Quebec, Canada: Concordia University.

Soltani, M. M., Karandish, S. F., Ahmed, W., Zhu, Z., & Hammad, A. (2017). Evaluating the Performance of Convolutional Neural Network for Classifying Equipment on Construction Sites. *nternational Symposium on Automation and Robotics in Construction* .

Soltani, M. M., Zhu, Z., & Hammad, A. (2016). utomated annotation for visual recognition of construction resources using synthetic images. *Automation in Construction*, 14-23.

Statistic, O. (n.d.). Retrieved from https://www.osha.gov/Publications/OSHA3252/3252.html

Stauffer, C., & Grimson, W. E. L. (1999). Adaptive background mixture models for real-time tracking. *IEEE*.

Sun, E., Nieto, A., & Li, Z. (2009, November). Real-time Google Earth 3D assisted driving system in surface mining operations. In Computer-Aided Industrial Design & Conceptual Design. *IEEE 10th International Conference*, 2095-2099.

Szeliski, R. (2010). *Computer vision: algorithms and applications.* Springer Science & Business Media.

Szeliski, R. (2011). *Computer Vision Algorithms and Applications.* Washington : Springer London Dordrecht Heidelberg New York.

Team, G. B. (2018). *About tensorflow*. Retrieved from https://www.tensorflow.org/

Teizer, J. (2008). 3D range imaging camera sensing for active safety in construction. *The Electronic Journal of Information Technology in Construction*, 103-117.

Teizer, J., & Vela, P. A. (2009). Personnel tracking on construction sites using video cameras. *Advanced Engineering Informatics*, 452-462.

Teizer, J., Allread, B. S., Fullerton, C. E., & Hinze, J. (2010). Autonomous pro-active real-time construction worker and equipment operator proximity safety alert system. *Automation in construction*, 630-640.

Thao, N., Eun Ae, P., Jiho, H., Dong Chul, P., & Soo Young, M. (2014). *Object detection using scale invariant feature transform.* Springer International Publishing Switzerland.

Tikkainen, T. (2014). *Cell detection from microscope image using histogram of oriented.* Finland: Master's thesis, Tampere University of Technology.

Viola, P., & Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. In Computer Vision and Pattern Recognition. *Proceedings of the 2001 IEEE Computer Society Conference on* .

Wozniak, P. A. (2018). Scene Recognition for Indoor Localization of Mobile Robots Using Deep CNN. *International Conference on Computer Vision and Graphics* (pp. 137-147). Springer, Cham.

Yards, A. (2008, October 21). Retrieved from bhttp://www.vilant.com: http://www.vilant.com/2368?dsid9= 7g81jll4m5r7hbq0sg1racrg46

Yin, Z., Kanade, T., & Chen, M. (2012). Understanding the phase contrast optics to restore artifact-free microscopy images for segmentation. *Medical image analysis*, 1047-1062.

Zhang, C., Hammad, A., & Bahnassi, H. (2009). Collaborative multi-agent systems for construction equipment based on real-time field data capturing. *Journal of Information Technology in Construction (ITcon)*, 204-228.

Zhang, Y., Jin, R., & Zhou, Z. H. (2010a). Understanding bag of words: a statistical framework. *International Journal of Machine Learning and Cybernetics, 1(1-4)*, 43-52.

Zhang, B., Gao, Y., Zhao, S., & Liu, J. (2010b). Local derivative pattern versus local binary pattern: face recognition with high-order local pattern discriptor. *IEEE transactions on image processing, 19(2)*, 533-544.

Zou, J., & Kim, H. (2007). Using hue, saturation, and value color space for hydraulic excavator idle time analysis. *Journal of computing in civil engineering*, 238-246.

APPENDICES

Appendix A. Matlab Code of Auto Annotation (Soltani, 2017)

```
clc
clear
% Margine
m = 20;
% Number of Conditions
CD = 3;
tic
MainBackImg = 'C:\Users\umroot\Documents\MATLAB\My PhD Codes\Papers\Conference\Excavator3DModel\BackGround\';
JPG = '*.jpg';
BackImList = dir(fullfile(MainBackImg,JPG));
% Number of Backgrounds
NoG = length(BackImList);

MainImg = 'C:\Users\umroot\Documents\MATLAB\My PhD Codes\Papers\Conference\SensivityAnalysis\8of16\+45\';
JPG = '*.jpg';
ImList = dir(fullfile(MainImg,JPG));
% Number of Images
NoI = length(ImList);

NewImgBG = 'C:\Users\umroot\Documents\MATLAB\My PhD Codes\Papers\Conference\SensivityAnalysis\8of16\+45\NewGenImg\';

parfor i=1:NoI
imgfile = fullfile(MainImg,ImList(i).name);
I1 = imread(imgfile);
I = rgb2gray(I1);
DipperMat(i).imageFilename = strrep (imgfile, '/', '\');
figure, imshow(I1), title('original image');
I = rgb2gray(imread('Dipper.jpg'));
figure, imshow(I), title('gray image');

se90 = strel('line', 3, 90);
se0 = strel('line', 3, 0);

[~, threshold] = edge(I, 'sobel');
fudgeFactor = 0.1;
BWs = edge(I,'sobel', threshold * fudgeFactor);
figure, imshow(BWs), title('binary gradient mask');

BWsdil = imdilate(BWs, [se90 se0]);
figure, imshow(BWsdil), title('dilated gradient mask');

BWdfill = imfill(BWsdil, 'holes');
figure, imshow(BWdfill);title('binary image with filled holes');

BWnobord = imclearborder(BWdfill, 4);

figure, imshow(BWnobord), title('cleared border image');

seD = strel('diamond',1);
BWfinal = imerode(BWnobord,seD);
BWfinal = imerode(BWfinal,seD);
figure, imshow(BWfinal), title('segmented image');

BWoutline = bwperim(BWfinal,8);
Segout = I;
Segout(BWoutline) = 100;

figure, imshow(BWoutline), title( );
```

```
regdata = regionprops(BWfinal,'BoundingBox','Area');
Area = cat(1, regdata.Area);
BoundingBox = cat(1, regdata.BoundingBox);
Best = find(Area(:) == max(Area(:)));
bbox = BoundingBox(Best,:);
DipperMat(i).objectBoundingBoxes = [bbox(1,1)-m bbox(1,2)-m bbox(1,3)+2*m
bbox(1,4)+2*m];
x = bbox(1, 1); y = bbox(1, 2); w = bbox(1, 3); h = bbox(1, 4);
bboxPolygon = [x-m, y-m, x+w+m, y-m, x+w+m, y+h+m, x-m, y+h+m];
videoFrame = insertShape(I1, 'Polygon', bboxPolygon,'Color','black');
figure, imshow(videoFrame), title(' Bounding Box');
end


parfor i=1:NoG
BG = [];
BG = imread(fullfile(MainBackImg,BackImList(i).name));
figure, imshow(LogZero)
for j=1:NoI
LogOne = [];
LogOne = imread(fullfile(MainImg,ImList(j).name));
figure, imshow(LogOne)
I = rgb2gray(LogOne);
figure, imshow(I)
I2 = [];
I2 = ~im2bw(I,0.95);
I2 = repmat(I2,[1 1 3]);
LogOne(I2==0)=0;
figure, imshow(LogOne)
I3 = [];
I3 = ~I2;
LogZero = BG;
LogZero(I3==0)=0;
figure, imshow(LogZero)
for k=1:CD
        LogOne = imadjust(LogOne,[0; 1],[(k-1)*0.1; (1.1-(k*0.1))],(1.1-(k*0.1)));
        figure, imshow(LogOne)
        LogOne(I2==0)=0;
        figure, imshow(LogOne)
        K=imadd(LogZero,LogOne);
        figure, imshow(K)
        New_k = k + (CD*(j-1)) + (NoI*CD*(i-1));
        imwrite(K,fullfile(NewImgBG,sprintf('IMAGE%04d.jpg',New_k)));

                NewTemp(i,j,k).imageFilename =
                fullfile(NewImgBG,sprintf('IMAGE%04d.jpg',New_k));
            NewTemp(i,j,k).objectBoundingBoxes = DipperMat(j).objectBoundingBoxes
        end
    end
end
New_k=0;
for i=1:NoG
    for j=1:NoI
        for k=1:CD
            New_k = New_k+1;
            Temp(New_k).imageFilename = NewTemp(i,j,k).imageFilename;
            Temp(New_k).objectBoundingBoxes =
                NewTemp(i,j,k).objectBoundingBoxes;
        end
    end
end
toc
DipperMat = [DipperMat, Temp];
save(fullfile(MainImg,sprintf('ROI.mat')),'DipperMat');
% 0 on background 1 on foreground
% I gray original image
% I2 Background with logical One on foreground
% I3 Background with logical Zero on foreground
```

Appendix B. Matlab Code of Auto Cropping (Soltani, 2017)

```
clc
clear

tic
load('C:\Users\umroot\Documents\MATLAB\My PhD Codes\Papers\PartRecognition\ExcavatorCompleteModel\Dipper\E\OnlyDipper\E-5-55D45-135\ROI.mat');

MainImg = 'C:\Users\umroot\Documents\MATLAB\My PhD Codes\Papers\PartRecognition\ExcavatorCompleteModel\Dipper\E\OnlyDipper\E-5-55D45-135\NewGenImg\';
JPG = '*.jpg';
ImList = dir(fullfile(MainImg,JPG));
% Number of Images
NoI = length(ImList);

PreImg = 'C:\Users\umroot\Documents\MATLAB\My PhD Codes\Papers\PartRecognition\ExcavatorCompleteModel\Dipper\E\OnlyDipper\E-5-55D45-135\';
JPG = '*.jpg';
ImPreList = dir(fullfile(PreImg,JPG));
% Number of Images
NoPI = length(ImPreList);

NewImgBG = 'C:\Users\umroot\Documents\MATLAB\My PhD Codes\Papers\PartRecognition\ExcavatorCompleteModel\Dipper\E\OnlyDipper\E-5-55D45-135\Cropped\';


parfor i=1:NoI+NoPI
        imgfile = DipperMat(i).imageFilename;
        I1 = imread(imgfile);
        bbox = DipperMat(i).objectBoundingBoxes;
        Icrop = imcrop(I1, bbox);
        imwrite(Icrop,fullfile(NewImgBG,sprintf('Cropped%04d.jpg',i)));
end
toc
```

Appendix C. Matlab Code of AlexNet (Soltani et al. 2017)

```
digitDatasetPath = 'E:\Forough\Excavator-7poses\Training_set\Synthetic-6000-excavator7poses';
digitData = imageDatastore(digitDatasetPath,'IncludeSubfolders',true,'LabelSource','foldernames');
testDataPath = 'E:\Forough\Excavator-7poses\Testing-set\testing20%';
testData = imageDatastore(testDataPath,'IncludeSubfolders',true,'LabelSource','foldernames');
digitData.countEachLabel
minSetCount = min(digitData.countEachLabel{:,2})
 %trainingNumFiles = round(minSetCount/20);
trainingNumFiles = 267;
rng(1)
[trainDigitData,testDataPath] = splitEachLabel (digitData,trainingNumFiles,'randomize');
NewNet = alexnet;
layers = [imageInputLayer([128 128 3]);

        NewNet.Layers(2);NewNet.Layers(3);NewNet.Layers(4);
        NewNet.Layers(5);NewNet.Layers(6);NewNet.Layers(7);
        NewNet.Layers(8);NewNet.Layers(9);NewNet.Layers(10);
        NewNet.Layers(11);NewNet.Layers(12);NewNet.Layers(13);
        NewNet.Layers(14);NewNet.Layers(15);NewNet.Layers(16);



        fullyConnectedLayer(512);NewNet.Layers(18);NewNet.Layers(19);
        fullyConnectedLayer(512);NewNet.Layers(21);NewNet.Layers(22);
        fullyConnectedLayer(3);softmaxLayer();classificationLayer()];
options = trainingOptions('sgdm','MaxEpochs',10,'InitialLearnRate',0.001);
tic
convnet = trainNetwork(trainDigitData,layers,options);
toc
YTest = classify(convnet,testData);
TTest = testData.Labels;
accuracy = sum(YTest == TTest)/numel(TTest)
Excavator = sum(YTest(1:555) == TTest(1:555))/numel(TTest(1:555))
Loader = sum(YTest(556:822) == TTest(556:822))/numel(TTest(556:822))
Truck = sum(YTest(823:1169) == TTest(823:1169))/numel(TTest(823:1169))
```

Appendix D. Python Code of TensorFlow Object Detection Model SSD-MobileNet-V1 (Huang et al, 2017)

```
# SSD with Mobilenet v1 configuration for MSCOCO Dataset.
# Users should configure the fine_tune_checkpoint field in the train config as
# well as the label_map_path and input_path fields in the train_input_reader and
# eval_input_reader. Search for "PATH_TO_BE_CONFIGURED" to find the fields that
# should be configured.

model {
  ssd {
    num_classes: 3
    box_coder {
      faster_rcnn_box_coder {
        y_scale: 10.0
        x_scale: 10.0
        height_scale: 5.0
        width_scale: 5.0
      }
    }
    matcher {
      argmax_matcher {
        matched_threshold: 0.5
        unmatched_threshold: 0.5
        ignore_thresholds: false
        negatives_lower_than_unmatched: true
        force_match_for_each_row: true
      }
    }
    similarity_calculator {
      iou_similarity {
      }
    }
    anchor_generator {
      ssd_anchor_generator {
        num_layers: 6
        min_scale: 0.2
        max_scale: 0.95
        aspect_ratios: 1.0
        aspect_ratios: 2.0
        aspect_ratios: 0.5
        aspect_ratios: 3.0
        aspect_ratios: 0.3333
      }
    }
    image_resizer {
      fixed_shape_resizer {
        height: 300
        width: 300
      }
    }
    box_predictor {
      convolutional_box_predictor {
        min_depth: 0
        max_depth: 0
        num_layers_before_predictor: 0
        use_dropout: false
        dropout_keep_probability: 0.8
        kernel_size: 1
        box_code_size: 4
        apply_sigmoid_to_scores: false
        conv_hyperparams {
          activation: RELU_6,
          regularizer {
            l2_regularizer {
              weight: 0.00004
            }
          }
```

```
      initializer {
       truncated_normal_initializer {
        stddev: 0.03
        mean: 0.0
       }
      }
      batch_norm {
       train: true,
       scale: true,
       center: true,
       decay: 0.9997,
       epsilon: 0.001,
      }
     }
    }
  }
  feature_extractor {
   type: 'ssd_mobilenet_v1'
   min_depth: 16
   depth_multiplier: 1.0
   conv_hyperparams {
    activation: RELU_6,
    regularizer {
     l2_regularizer {
      weight: 0.00004
     }
    }
    initializer {
     truncated_normal_initializer {
      stddev: 0.03
      mean: 0.0
     }
    }
    batch_norm {
     train: true,
     scale: true,
     center: true,
     decay: 0.9997,
     epsilon: 0.001,
    }
   }
  }
  loss {
   classification_loss {
    weighted_sigmoid {
    }
   }
   localization_loss {
    weighted_smooth_l1 {
    }
   }
   hard_example_miner {
    num_hard_examples: 3000
    iou_threshold: 0.99
    loss_type: CLASSIFICATION
    max_negatives_per_positive: 3
    min_negatives_per_image: 0
   }
   classification_weight: 1.0
   localization_weight: 1.0
  }
  normalize_loss_by_num_matches: true
  post_processing {
   batch_non_max_suppression {
    score_threshold: 1e-8
    iou_threshold: 0.6
    max_detections_per_class: 100
```

```
      max_total_detections: 100
    }
    score_converter: SIGMOID
   }
  }
}

train_config: {
 batch_size: 24
 optimizer {
  rms_prop_optimizer: {
   learning_rate: {
    exponential_decay_learning_rate {
     initial_learning_rate: 0.004
     decay_steps: 800720
     decay_factor: 0.95
    }
   }
   momentum_optimizer_value: 0.9
   decay: 0.9
   epsilon: 1.0
  }
 }
 fine_tune_checkpoint: "ssd_mobilenet_v1_coco_2018_01_28/model.ckpt"
 from_detection_checkpoint: true
 # Note: The below line limits the training process to 200K steps, which we
 # empirically found to be sufficient enough to train the pets dataset. This
 # effectively bypasses the learning rate schedule (the learning rate will
 # never decay). Remove the below line to train indefinitely.
 num_steps: 65000
 data_augmentation_options {
  random_horizontal_flip {
  }
 }
 data_augmentation_options {
  ssd_random_crop {
  }
 }
}

train_input_reader: {
 tf_record_input_reader {
  input_path: "data/train.record"
 }
 label_map_path: "training/label_map.pbtxt"
}

eval_config: {
 num_examples: 1000
 # Note: The below line limits the evaluation process to 10 evaluations.
 # Remove the below line to evaluate indefinitely.
 max_evals: 10
}

eval_input_reader: {
 tf_record_input_reader {
  input_path: "data/test.record"
 }
 label_map_path: "training/label_map.pbtxt"
 shuffle: false
 num_readers: 1
}
```

Appendix E. Python Code of TensorFlow Object Detection Model R-FCN-Resnet101 (Huang et al, 2017)

```
# R-FCN with Resnet-101 (v1),  configuration for MSCOCO Dataset.
# Users should configure the fine_tune_checkpoint field in the train config as
# well as the label_map_path and input_path fields in the train_input_reader and
# eval_input_reader. Search for "PATH_TO_BE_CONFIGURED" to find the fields that
# should be configured.

model {
  faster_rcnn {
    num_classes: 3
    image_resizer {
      keep_aspect_ratio_resizer {
        min_dimension: 600
        max_dimension: 1024
      }
    }
    feature_extractor {
      type: 'faster_rcnn_resnet101'
      first_stage_features_stride: 16
    }
    first_stage_anchor_generator {
      grid_anchor_generator {
        scales: [0.25, 0.5, 1.0, 2.0]
        aspect_ratios: [0.5, 1.0, 2.0]
        height_stride: 16
        width_stride: 16
      }
    }
    first_stage_box_predictor_conv_hyperparams {
      op: CONV
      regularizer {
        l2_regularizer {
          weight: 0.0
        }
      }
      initializer {
        truncated_normal_initializer {
          stddev: 0.01
        }
      }
    }
    first_stage_nms_score_threshold: 0.0
    first_stage_nms_iou_threshold: 0.7
    first_stage_max_proposals: 300
```

```
  first_stage_localization_loss_weight: 2.0
  first_stage_objectness_loss_weight: 1.0
  second_stage_box_predictor {
   rfcn_box_predictor {
    conv_hyperparams {
     op: CONV
     regularizer {
      l2_regularizer {
       weight: 0.0
      }
     }
     initializer {
      truncated_normal_initializer {
       stddev: 0.01
      }
     }
    }
    crop_height: 18
    crop_width: 18
    num_spatial_bins_height: 3
    num_spatial_bins_width: 3
   }
  }
  second_stage_post_processing {
   batch_non_max_suppression {
    score_threshold: 0.0
    iou_threshold: 0.6
    max_detections_per_class: 100
    max_total_detections: 300
   }
   score_converter: SOFTMAX
  }
  second_stage_localization_loss_weight: 2.0
  second_stage_classification_loss_weight: 1.0
 }
}
train_config: {
 batch_size: 1
 optimizer {
  momentum_optimizer: {
   learning_rate: {
    manual_step_learning_rate {
     initial_learning_rate: 0.0003
     schedule {
      step: 900000
```

```
          learning_rate: .00003
        }
        schedule {
          step: 1200000
          learning_rate: .000003
        }
      }
    }
    momentum_optimizer_value: 0.9
  }
  use_moving_average: false
 }
 gradient_clipping_by_norm: 10.0
 fine_tune_checkpoint: "rfcn_resnet101_coco_2018_01_28/model.ckpt"
 from_detection_checkpoint: true
 # Note: The below line limits the training process to 200K steps, which we
 # empirically found to be sufficient enough to train the pets dataset. This
 # effectively bypasses the learning rate schedule (the learning rate will
 # never decay). Remove the below line to train indefinitely.
 num_steps: 65000
 data_augmentation_options {
  random_horizontal_flip {
  }
 }
}
train_input_reader: {
 tf_record_input_reader {
  input_path: "data/train.record"
 }
 label_map_path: "training/label_map.pbtxt"
}
eval_config: {
 num_examples: 1000
 # Note: The below line limits the evaluation process to 10 evaluations.
 # Remove the below line to evaluate indefinitely.
 max_evals: 10
}
eval_input_reader: {
 tf_record_input_reader {
  input_path: "data/test.record"
 }
 label_map_path: "training/label_map.pbtxt"
 shuffle: false
 num_readers: 1
}
```

Appendix F. Python Code of TensorFlow Object Detection Model Faster-RCNN-Resnet101 (Huang et al, 2017)

```
# Faster R-CNN with Resnet-101 (v1), configuration for MSCOCO Dataset.
# Users should configure the fine_tune_checkpoint field in the train config as
# well as the label_map_path and input_path fields in the train_input_reader and
# eval_input_reader. Search for "PATH_TO_BE_CONFIGURED" to find the fields that
# should be configured.
model {
  faster_rcnn {
    num_classes: 3
    image_resizer {
      keep_aspect_ratio_resizer {
        min_dimension: 600
        max_dimension: 1024
      }
    }
    feature_extractor {
      type: 'faster_rcnn_resnet101'
      first_stage_features_stride: 16
    }
    first_stage_anchor_generator {
      grid_anchor_generator {
        scales: [0.25, 0.5, 1.0, 2.0]
        aspect_ratios: [0.5, 1.0, 2.0]
        height_stride: 16
        width_stride: 16
      }
    }
    first_stage_box_predictor_conv_hyperparams {
      op: CONV
      regularizer {
        l2_regularizer {
          weight: 0.0
        }
      }
      initializer {
        truncated_normal_initializer {
          stddev: 0.01
        }
      }
    }
    first_stage_nms_score_threshold: 0.0
    first_stage_nms_iou_threshold: 0.7
    first_stage_max_proposals: 300
    first_stage_localization_loss_weight: 2.0
    first_stage_objectness_loss_weight: 1.0
    initial_crop_size: 14
    maxpool_kernel_size: 2
    maxpool_stride: 2
    second_stage_box_predictor {
      mask_rcnn_box_predictor {
        use_dropout: false
        dropout_keep_probability: 1.0
        fc_hyperparams {
          op: FC
          regularizer {
            l2_regularizer {
              weight: 0.0
            }
          }
          initializer {
            variance_scaling_initializer {
              factor: 1.0
              uniform: true
              mode: FAN_AVG
```

```
          }
         }
        }
       }
      }
      second_stage_post_processing {
       batch_non_max_suppression {
        score_threshold: 0.0
        iou_threshold: 0.6
        max_detections_per_class: 100
        max_total_detections: 300
       }
       score_converter: SOFTMAX
      }
      second_stage_localization_loss_weight: 2.0
      second_stage_classification_loss_weight: 1.0
     }
    }
    train_config: {
     batch_size: 1
     optimizer {
      momentum_optimizer: {
       learning_rate: {
        manual_step_learning_rate {
         initial_learning_rate: 0.0003
         schedule {
          step: 900000
          learning_rate: .00003
         }
         schedule {
          step: 1200000
          learning_rate: .000003
         }
        }
       }
       momentum_optimizer_value: 0.9
      }
      use_moving_average: false
     }
     gradient_clipping_by_norm: 10.0
     fine_tune_checkpoint: "faster_rcnn_resnet101_coco_2018_01_28/model.ckpt"
     from_detection_checkpoint: true
     data_augmentation_options {
      random_horizontal_flip {
      }
     }
    }
    train_input_reader: {
     tf_record_input_reader {
      input_path: "data/train.record"
     }
     label_map_path: "training/label_map.pbtxt"
    }
    eval_config: {
     num_examples: 1000
     # Note: The below line limits the evaluation process to 10 evaluations.
     # Remove the below line to evaluate indefinitely.
     max_evals: 10
    }
    eval_input_reader: {
     tf_record_input_reader {
      input_path: "data/test.record"
     }
     label_map_path: "training/label_map.pbtxt"
     shuffle: false
     num_readers: 1
    }
```

Appendix G. Python code of TensorFlow Object Detection for Training (Huang et al, 2017)

```
# Copyright 2017 The TensorFlow Authors. All Rights Reserved.
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#     http://www.apache.org/licenses/LICENSE-2.0
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
# ==============================================================================
r"""Training executable for detection models.
This executable is used to train DetectionModels. There are two ways of
configuring the training job:
1) A single pipeline_pb2.TrainEvalPipelineConfig configuration file
can be specified by --pipeline_config_path.
Example usage:
    ./train \
        --logtostderr \
        --train_dir=path/to/train_dir \
        --pipeline_config_path=pipeline_config.pbtxt

2) Three configuration files can be provided: a model_pb2.DetectionModel
configuration file to define what type of DetectionModel is being trained, an
input_reader_pb2.InputReader file to specify what training data will be used and
a train_pb2.TrainConfig file to configure training parameters.
Example usage:
    ./train \
        --logtostderr \
        --train_dir=path/to/train_dir \
        --model_config_path=model_config.pbtxt \
        --train_config_path=train_config.pbtxt \
        --input_config_path=train_input_config.pbtxt
"""

import functools
import json
import os
import tensorflow as tf

from object_detection.builders import dataset_builder
from object_detection.builders import graph_rewriter_builder
from object_detection.builders import model_builder
from object_detection.legacy import trainer
from object_detection.utils import config_util

tf.logging.set_verbosity(tf.logging.INFO)

flags = tf.app.flags
flags.DEFINE_string('master', '', 'Name of the TensorFlow master to use.')
flags.DEFINE_integer('task', 0, 'task id')
flags.DEFINE_integer('num_clones', 1, 'Number of clones to deploy per worker.')
flags.DEFINE_boolean('clone_on_cpu', False,
                     'Force clones to be deployed on CPU.  Note that even if '
                     'set to False (allowing ops to run on gpu), some ops may '
                     'still be run on the CPU if they have no GPU kernel.')
flags.DEFINE_integer('worker_replicas', 1, 'Number of worker+trainer '
                     'replicas.')
flags.DEFINE_integer('ps_tasks', 0,
                     'Number of parameter server tasks. If None, does not use '
                     'a parameter server.')
flags.DEFINE_string('train_dir', '',
```

```
            'Directory to save the checkpoints and training summaries.')

flags.DEFINE_string('pipeline_config_path', '',
            'Path to a pipeline_pb2.TrainEvalPipelineConfig config '
            'file. If provided, other configs are ignored')

flags.DEFINE_string('train_config_path', '',
            'Path to a train_pb2.TrainConfig config file.')
flags.DEFINE_string('input_config_path', '',
            'Path to an input_reader_pb2.InputReader config file.')
flags.DEFINE_string('model_config_path', '',
            'Path to a model_pb2.DetectionModel config file.')

FLAGS = flags.FLAGS


@tf.contrib.framework.deprecated(None, 'Use object_detection/model_main.py.')
def main(_):
  assert FLAGS.train_dir, '`train_dir` is missing.'
  if FLAGS.task == 0: tf.gfile.MakeDirs(FLAGS.train_dir)
  if FLAGS.pipeline_config_path:
    configs = config_util.get_configs_from_pipeline_file(
        FLAGS.pipeline_config_path)
    if FLAGS.task == 0:
      tf.gfile.Copy(FLAGS.pipeline_config_path,
                os.path.join(FLAGS.train_dir, 'pipeline.config'),
                overwrite=True)
  else:
    configs = config_util.get_configs_from_multiple_files(
        model_config_path=FLAGS.model_config_path,
        train_config_path=FLAGS.train_config_path,
        train_input_config_path=FLAGS.input_config_path)
    if FLAGS.task == 0:
      for name, config in [('model.config', FLAGS.model_config_path),
                           ('train.config', FLAGS.train_config_path),
                           ('input.config', FLAGS.input_config_path)]:
        tf.gfile.Copy(config, os.path.join(FLAGS.train_dir, name),
                      overwrite=True)

  model_config = configs['model']
  train_config = configs['train_config']
  input_config = configs['train_input_config']

  model_fn = functools.partial(
      model_builder.build,
      model_config=model_config,
      is_training=True)

  def get_next(config):
    return dataset_builder.make_initializable_iterator(
        dataset_builder.build(config)).get_next()

  create_input_dict_fn = functools.partial(get_next, input_config)

  env = json.loads(os.environ.get('TF_CONFIG', '{}'))
  cluster_data = env.get('cluster', None)
  cluster = tf.train.ClusterSpec(cluster_data) if cluster_data else None
  task_data = env.get('task', None) or {'type': 'master', 'index': 0}
  task_info = type('TaskSpec', (object,), task_data)

  # Parameters for a single worker.
  ps_tasks = 0
  worker_replicas = 1
  worker_job_name = 'lonely_worker'
  task = 0
  is_chief = True
  master = ''
```

```
  if cluster_data and 'worker' in cluster_data:
    # Number of total worker replicas include "worker"s and the "master".
    worker_replicas = len(cluster_data['worker']) + 1
  if cluster_data and 'ps' in cluster_data:
    ps_tasks = len(cluster_data['ps'])

  if worker_replicas > 1 and ps_tasks < 1:
    raise ValueError('At least 1 ps task is needed for distributed training.')

  if worker_replicas >= 1 and ps_tasks > 0:
    # Set up distributed training.
    server = tf.train.Server(tf.train.ClusterSpec(cluster), protocol='grpc',
                             job_name=task_info.type,
                             task_index=task_info.index)
    if task_info.type == 'ps':
      server.join()
      return

    worker_job_name = '%s/task:%d' % (task_info.type, task_info.index)
    task = task_info.index
    is_chief = (task_info.type == 'master')
    master = server.target

  graph_rewriter_fn = None
  if 'graph_rewriter_config' in configs:
    graph_rewriter_fn = graph_rewriter_builder.build(
        configs['graph_rewriter_config'], is_training=True)

  trainer.train(
      create_input_dict_fn,
      model_fn,
      train_config,
      master,
      task,
      FLAGS.num_clones,
      worker_replicas,
      FLAGS.clone_on_cpu,
      ps_tasks,
      worker_job_name,
      is_chief,
      FLAGS.train_dir,
      graph_hook_fn=graph_rewriter_fn)


if __name__ == '__main__':
  tf.app.run()
```

Appendix H. Python Code of TensorFlow Object Detection for Evaluating (Huang et al, 2017)

```
# Copyright 2017 The TensorFlow Authors. All Rights Reserved.
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#     http://www.apache.org/licenses/LICENSE-2.0
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
# ==============================================================================
r"""Evaluation executable for detection models.
This executable is used to evaluate DetectionModels. There are two ways of
configuring the eval job.
1) A single pipeline_pb2.TrainEvalPipelineConfig file maybe specified instead.
In this mode, the --eval_training_data flag may be given to force the pipeline
to evaluate on training data instead.
Example usage:
    ./eval \
        --logtostderr \
        --checkpoint_dir=path/to/checkpoint_dir \
        --eval_dir=path/to/eval_dir \
        --pipeline_config_path=pipeline_config.pbtxt

2) Three configuration files may be provided: a model_pb2.DetectionModel
configuration file to define what type of DetectionModel is being evaluated, an
input_reader_pb2.InputReader file to specify what data the model is evaluating
and an eval_pb2.EvalConfig file to configure evaluation parameters.
Example usage:
    ./eval \
        --logtostderr \
        --checkpoint_dir=path/to/checkpoint_dir \
        --eval_dir=path/to/eval_dir \
        --eval_config_path=eval_config.pbtxt \
        --model_config_path=model_config.pbtxt \
        --input_config_path=eval_input_config.pbtxt
"""
import functools
import os
import tensorflow as tf

from object_detection.builders import dataset_builder
from object_detection.builders import graph_rewriter_builder
```

```
from object_detection.builders import model_builder
from object_detection.legacy import evaluator
from object_detection.utils import config_util
from object_detection.utils import label_map_util

tf.logging.set_verbosity(tf.logging.INFO)

flags = tf.app.flags
flags.DEFINE_boolean('eval_training_data', False,
                     'If training data should be evaluated for this job.')
flags.DEFINE_string(
    'checkpoint_dir', '',
    'Directory containing checkpoints to evaluate, typically '
    'set to `train_dir` used in the training job.')
flags.DEFINE_string('eval_dir', '', 'Directory to write eval summaries to.')
flags.DEFINE_string(
    'pipeline_config_path', '',
    'Path to a pipeline_pb2.TrainEvalPipelineConfig config '
    'file. If provided, other configs are ignored')
flags.DEFINE_string('eval_config_path', '',
                    'Path to an eval_pb2.EvalConfig config file.')
flags.DEFINE_string('input_config_path', '',
                    'Path to an input_reader_pb2.InputReader config file.')
flags.DEFINE_string('model_config_path', '',
                    'Path to a model_pb2.DetectionModel config file.')
flags.DEFINE_boolean(
    'run_once', False, 'Option to only run a single pass of '
    'evaluation. Overrides the `max_evals` parameter in the '
    'provided config.')
FLAGS = flags.FLAGS


@tf.contrib.framework.deprecated(None, 'Use object_detection/model_main.py.')
def main(unused_argv):
  assert FLAGS.checkpoint_dir, '`checkpoint_dir` is missing.'
  assert FLAGS.eval_dir, '`eval_dir` is missing.'
  tf.gfile.MakeDirs(FLAGS.eval_dir)
  if FLAGS.pipeline_config_path:
    configs = config_util.get_configs_from_pipeline_file(
        FLAGS.pipeline_config_path)
    tf.gfile.Copy(
        FLAGS.pipeline_config_path,
        os.path.join(FLAGS.eval_dir, 'pipeline.config'),
        overwrite=True)
  else:
```

```
  configs = config_util.get_configs_from_multiple_files(
      model_config_path=FLAGS.model_config_path,
      eval_config_path=FLAGS.eval_config_path,
      eval_input_config_path=FLAGS.input_config_path)
  for name, config in [('model.config', FLAGS.model_config_path),
                       ('eval.config', FLAGS.eval_config_path),
                       ('input.config', FLAGS.input_config_path)]:
    tf.gfile.Copy(config, os.path.join(FLAGS.eval_dir, name), overwrite=True)

  model_config = configs['model']
  eval_config = configs['eval_config']
  input_config = configs['eval_input_config']
  if FLAGS.eval_training_data:
    input_config = configs['train_input_config']

  model_fn = functools.partial(
      model_builder.build, model_config=model_config, is_training=False)

  def get_next(config):
    return dataset_builder.make_initializable_iterator(
        dataset_builder.build(config)).get_next()

  create_input_dict_fn = functools.partial(get_next, input_config)

  categories = label_map_util.create_categories_from_labelmap(
      input_config.label_map_path)

  if FLAGS.run_once:
    eval_config.max_evals = 1
  graph_rewriter_fn = None
  if 'graph_rewriter_config' in configs:
    graph_rewriter_fn = graph_rewriter_builder.build(
        configs['graph_rewriter_config'], is_training=False)
  evaluator.evaluate(
      create_input_dict_fn,
      model_fn,
      eval_config,
      categories,
      FLAGS.checkpoint_dir,
      FLAGS.eval_dir,
      graph_hook_fn=graph_rewriter_fn)
if __name__ == '__main__':
  tf.app.run()
```