Fall 2021

Computer Vision-Based Automatic Railroad Crossing Monitoring and Track Inspection

Feng Guo

COMPUTER VISION-BASED AUTOMATIC RAILROAD CROSSING MONITORING AND TRACK INSPECTION

by

Feng Guo

Bachelor of Science
Ludong University, 2014

Master of Engineering
Beijing University of Civil Engineering and Architecture, 2017

Submitted in Partial Fulfillment of the Requirements

For the Degree of Doctor of Philosophy in

Civil Engineering

College of Engineering and Computing

University of South Carolina

2021

Accepted by:

Yu Qian, Major Professor

Yi Wang, Committee Member

Dimitris Rizos, Committee Member

Robert L Mullen, Committee Member

Tracey L. Weldon, Interim Vice Provost and Dean of the Graduate School

DEDICATION

To my father Yongfa Guo, my mother Yinmin Xie, my little brother, Song Guo, and my fiancée Jun Dai

.

ACKNOWLEDGEMENTS

I have imagined writing these acknowledgments many times. When I began drafting this part, I was a bit sad as I realized that simple sentences cannot express my sincere gratitude for the people who I have met over the past 3.5 years. Without their individual and collective help, I would not have been able to finish this tough journey.

I would like to begin by thanking my advisor, Dr. Yu Qian. He is an excellent mentor with great knowledge, patience, passion, humor, and empathy. He not only encouraged me to critically examine my research; he also instilled in me a positive attitude towards life, a curiosity about science, and a high standard of work. However, I am sorry for having always sent him emails late at night and on weekends. I would also like to thank my co-advisor, Dr. Yi Wang. He is a cool man who has helped me understand lifelong learning and the meaning of research. His dedication to research will always motivate me to explore the unknown. I would like to express my appreciation to my committee members as well: Dr. Dimitris Rizos and Dr. Robert L. Mullen. I would not have completed this dissertation without their constructive guidance.

My thanks also go to my friends and colleagues at the University of South Carolina: Ai Li, Zhong Li, Ruixiao Sun, Junlin Ou, Haizhou Yang. Siwadol Dejphumee, Pitak Rutti, Huan Jiang, Hongxiao Yu, Zhuocheng Jiang, Ningqiao Li, Rui Xin, Xiaoqun Yan, Yunpeng Wu, Xu Wu, Yufeng Gong, Shihao Huang, Huaqiang Guo, and Youzhi Tang. Special thanks to my country fellow, Dr. Jing Li – she has driven me to reflect on my experience, and her attitude towards life will inspire me moving forward. I also want

to thank Dr. Xinxiang Zhang; I will never forget the night he helped me debug my code, and his eloquence is truly impressive.

My internship at Palo Alto Research Center opened my eyes to the thrilling world of computer science. I am forever grateful to my mentor and colleagues, Dr. Bob Price, Dr. Raja Bala, and Jonathan Ju.

I also greatly appreciate the support from my previous advisors and friends, Dr. Zhi Suo, Dr. Zhen Leng, Dr. Shanshan Jin, Dr. Rui Li, Yuefeng Shi, Peng Liu, and Qi Man. Special thanks to Dr. Huayang Yu for always being helpful.

This study is supported by the National Academy of Science (NAS) under the IDEA program, Federal Railroad Administration (FRA), City of Columbia, and CSX Corporation. Their support is sincerely appreciated.

In closing, I want to give my highest gratitude to my parents, my little brother, and my fiancée. Without their endless support and love, I would not have been able to come to the U.S. for my Ph.D. study, nor would I have survived during my darkest visa check period. Whenever I felt I could not continue and wanted to give up, their phone calls empowered me to keep going and complete this degree. This year is the eighth year since I met my fiancée. I am greatly indebted to her and cannot wait to marry her.

ABSTRACT

Currently, there are many imminent challenges in the railroad infrastructure system of the United States, impacting the operation, safety, and management of railroad transportation. In this work, three major challenges which are overcrowded traffic congestion at the grade crossing, low-efficiency and accuracy on inspection of missing or broken rail track components, and dense rail surface defects without quantification, respectively are studied. The congested railroad grade crossing not only introduces significant traffic delays to travelers but also brings potential safety concerns to the first responders. However, limited studies have been devoted on developing an intelligent traffic monitoring system which is significant to deliver real-time information to the travelers and the first responders to improve the traffic operation and safety at the railroad grade crossing. Except to improve the railroad safety related with travelers and the first responders in the first half, the rest of this dissertation focuses on the track safety related to railroad track components and surface defects. The missing or broken components such as spikes, clips, and tie plates can endanger the safety and operation of railroads. Even though various types of inspection approaches such as ground penetrating radar, laser, and LiDAR have been implemented, the operation needs rich experience and extensive training. Meanwhile, track inspections still heavily rely on manual inspection which is low-accurate, low-efficient, and highly subjective. Moreover, rail surface defects negatively impact riding comfort, operational safety, and could even lead to train derailments. During the past decades, there have been many efforts to detect rail surface

defects. Unfortunately, previous approaches for detecting and quantifying of rail surface defects are also limited by the high requirements of specialized equipment and personnel training.

The main focus of this work is to design and develop computer vision models to address the technical and practical challenges mentioned above. To cope with each challenge, different models including the object detection model, the instance segmentation model, and the semantic segmentation model have been successfully designed and developed. To train, validate, and test different models, three customized image datasets based on the traffic videos at the grade crossing, railroad component images, and dense rail surface defects images have been built. Specifically, a dense traffic detection net (DTDNet) is developed integrating the Transformer Attention (TA) module for better modeling of global context information and the learning-to-match detection head for optimizing object detection and localization using a likelihood probability fashion. A unique grade crossing traffic image dataset including congested and normal traffic during both daytime and nighttime is established. The proposed DTDNet and other state-of-the-art (SOTA) models have been trained, tested, and compared. The proposed DTDNet outperforms other SOTA models in the test cases. Regarding the automatic track components inspection, the real-time instance segmentation model and the YOLOv4-hybrid model have been designed, trained, tested, and evaluated. The first public rail components image database has been built and released online. Compared to the original YOLACT model and the Mask R-CNN model, the training performance has been improved with the improved instance segmentation model. The detection accuracy on the bounding box and the mask has been improved and the inference speed can

achieve the real-time speed. With respect to the YOLOv4-hybrid model, it outperforms other SOTA models on the training performance and the field tests with missing or fake rail track components. As for the rail surface defect inspection and quantification, the optimized Mask R-CNN model and the newly proposed lightweight Deeplabv3Plus model using Lovász-Softmax loss (LDL model) have been trained, tested, evaluated, and compared on our rail surface defects image database. Experimental results confirm the robustness and superiority of our model on defect segmentation. Besides, an algorithm is proposed to quantify rail surface defect severities at different levels using our rail surface defects image data.

Overall, this dissertation helps to improve the railroad safety by developing and implementing advanced computer vision-based models for better tracking monitoring and inspections.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1 INTRODUCTION

1.1 Background

Convolutional neural networks (CNNs), a subset of artificial intelligence (AI), have recently come to play a key role in the development of computer vision. CNNs can learn input features efficiently and are able to cope with the growing size of training data while offering strong computational power (Pan & Yang, 2020). Substantial progress in neural networks and computer vision has enabled the success of various civil engineering efforts based on machine learning and computer vision; such tasks involve pavement crack identification, health condition monitoring, and concrete structural damage evaluation (Perez-Ramirez et al., 2019; Yeum, Choi, & Dyke, 2019).

However, research and applications are limited regarding the use of computer vision techniques to address pressing challenges in the railroad system. Examples include rail track component inspection, dense traffic instance detection, and rail surface defect identification and quantification. In terms of dense traffic instance detection, the rapid development of deep learning has led various scholars (Fan, Brown, & Smith, 2016; Hu et al., 2018; Xu, Yu, Wang, Wu, & Ma, 2017; Yu, Yang, & Chen, 2021) to apply cutting-edge models such as You Only Look Once (YOLO) (Redmon, Divvala, Girshick, & Farhadi, 2016) and Faster R-CNN (Ren, He, Girshick, & Sun, 2016) to detect traffic or count vehicles. Yet these approaches often fail to achieve satisfactory results due to the complicated contexts at grade crossings.

Track component detection remains a challenging task due to complex environmental conditions, small objects, and limited training data. Additionally, railroad tracks can appear to be quite similar but have variation. For instance, spikes and clips may be quite different from each other depending on type. Also, the same components'

appearance can change based on the surrounding environment, considering that the tracks cross through multiple remote/rural areas.

Rail defects are the leading cause of freight train derailment in the United States. Investigation of a derailment case by the National Transportation Safety Board (Zakar & Mueller, 2016) focused on a severe accident in Columbus, OH: a broken rail caused more than $1.2 million in damage with evidence of rolling contact fatigue. Different stages of such fatigue manifest as small cracks, flaking, and spalling on the top of the rail. Thus, it is no exaggeration to say that a tiny defect that is initially overlooked on the rail surface may contribute to a broken rail in the future, with the potential for great losses.

1.2 Research motivation

This work aims to improve railroad safety from two perspectives: (1) human-related safety at congested grade crossings; and (2) infrastructure-related concerns, such as missing or broken rail track components and dense rail surface defects. With respect to railroad transportation management, for both historical and practical reasons, trains have the right-of-way at railroad-highway grade crossings (Estes & Rilett, 2000). Serious traffic congestion is common in highly populated areas during rush hours when a train slowly passes or completely stops at a grade crossing (Soleimani, Mousa, Codjoe, & Leitner, 2019). Unpredictable crossing blockages can produce delays for motorists and raise serious concerns for first responders (Ma, Hao, Xiang, & Yan, 2018; Park et al., 2016). To better manage and ease congested traffic at grade crossings, the first step is to implement an intelligent transportation system to automatically evaluate the congestion level accurately and efficiently (Lv, Duan, Kang, Li, & Wang, 2014). In other words, it is important to quickly and correctly determine how many vehicles are waiting in the queue

and how long it will take for the traffic flow to return to normal. This information should also be shared with the public and first responders.

Periodic inspection of railroad track components is essential to railroad safety and operations. According to the Federal Railroad Administration (FRA) safety database (FRA, 2018a), 546 accidents were associated with track defects in 2018, resulting in over $97 million in financial losses and countless social consequences. Out of these 546 accidents, 48 were caused by missing spikes, clips, or broken rails, amounting to roughly $10 million in damages. In the United States, FRA mandates routine track inspections as part of an early warning strategy (FRA, 2018b). Unfortunately, most track inspection work aside from track geometry measurement remains labor- and time-intensive, especially when inspecting missing track components. The results of inspecting missing components can also be inefficient and expensive given the nature of manual inspection. Liu et al. (2014) found that, apart from labor costs, inspection is completed at a speed of around 15–20 mph and the average inspection cost per hour per vehicle is nearly $300. Even with manned-inspection vehicles, the expected annual inspection cost is easily millions of dollars—and a considerable number of missing track components still need to be manually inspected by walking crews. This issue is more pronounced for Class I railroad mainlines due to the dense traffic volume and limited windows for inspection and maintenance, leading to accidents and potential derailments. For example, broken spikes caused a 120-car Norfolk Southern train derailment in Vandergrift, PA, which spilled between 3,000 and 4,000 gallons of crude oil (Hardway, 2014). Therefore, automated rail track component inspection is highly meaningful for the railroad industry as the FRA has pointed out (Saadat, Sherrock, & Zahaczewski, 2018).

The health of railroad infrastructure is paramount to train operations and track safety: reducing the risk of accidents due to unfavorable infrastructure conditions is of great interest to the public, railroads, and government. According to an FRA report (FRA, 2020a) and prior studies (Liu, Barkan, & Saat, 2011; Liu, Turla, & Zhang, 2018), rail defects represent a key factor in serious train accidents. These defects are also partly responsible for more than 70% of train derailments on freight mainlines (X. Liu et al., 2018). Furthermore, rail surface defects directly affect patrons' ride comfort and normal train operations due to additional excitations caused by surface irregularities. Such defects also introduce the risk of rail breakage, which could lead to catastrophic track failures. Yet inspection approaches rarely provide highly reliable results (FRA, 2020c), and inspection equipment is expensive and difficult to operate. Thus, a large proportion of inspection work relies heavily on visual inspections, which are labor-intensive, inefficient, and inherently subjective. As the number of freight shipments increases, the speed and accuracy of track inspection fail to satisfy rapidly growing inspection demands. A cost-effective, highly robust, reliable, and accurate inspection system is therefore urgently needed.

1.3 Research objective

This research seeks to develop AI solutions that leverage cutting-edge computer vision techniques such as object detection, instance segmentation, and semantic segmentation approaches to efficiently and accurately inspect/detect rail track components, rail surface defects, and dense traffic instances. These goals will be accomplished by:

- Proposing a real-time detector to identify dense traffic instances, including densely packed vehicles, small pedestrians, and long trains at grade crossings.
- Exploring the feasibility of the attention mechanism in modeling long-distance relationships for large object detection and examining detection performance using a maximum likelihood estimation (MLE) procedure to detect densely packed objects at grade crossings.
- Building the world's first public railroad components dataset, containing different rail track components, for free access and research purposes. This dataset may prompt the implementation of cutting-edge deep learning models in railroad applications.
- Developing real-time instance segmentation and object detection models with fast speed and high accuracy for automatic inspection of rail track components.
- Discussing the impacts of different lighting conditions on railroad component inspection.
- Proposing an optimized instance segmentation model for rail surface defect segmentation and a simple yet efficient approach for rail surface defect quantification.

Put simply, this study aims to develop intelligent systems using computer vision techniques for dense traffic instance detection, rail track component identification, and rail surface defect segmentation to address urgent needs in railroad infrastructure management and operations.

1.4 Research scope

To achieve the research goals mentioned above, the open-source library MMDetection and the Pytorch library were adopted for model training, validation, testing, and comparison. A deep learning workstation with four Nvidia 2080 Ti graphics processing units (GPUs) was used to support these tasks. The libraries of CUDA (v10.2) and CuDNN (v7.5.2), developed by Nvidia, were installed and configured in the workstation to provide high-performance GPU acceleration.

For dense traffic instance detection at grade crossings, an image database was assembled using video clips collected from a surveillance camera installed on the top of a power pole beside a grade crossing. Three object classes (i.e., vehicle, pedestrian, and train) are included in this study. A labeling tool, LabelImg, was used for ground truth labeling and annotation generation. A transformer attention module and MLE strategy were designed and integrated into the proposed model. Several state-of-the-art models were evaluated and compared using MMdetection library. The precision-recall curve, precision, recall, and inference speed were considered during model comparison and evaluation.

A public image database including rails, spikes, and clips was built to develop a real-time instance segmentation and object detection model for rail track component inspection. The Res2Net bottleneck, which can represent multi-scale features at a granular level with an increased receptive field, was embedded into the proposed model. Images were saved from video frames recorded on an iPhone® 8 with a 12-megapixel main camera which has a single wide-angle lens with an f/1.8 aperture. To prevent overfitting, the training images were processed with image augmentation, including

mirroring, rotation (90°), and a combination of rotation (180°) and Gaussian noise. A popular labeling tool, labelme (Ketaro Wada, 2016), was employed to generate annotation files. The mean average precision (mAP), inference speed, and average precision (AP) at the intersection over union equal to 0.5/0.75 were adopted as evaluation metrics to assess the training performance. To evaluate and compare inspection performance in the field, a complex environment was simulated under different lighting conditions.

A rail surface defect image database, including a rail and defects, was constructed for rail surface defect identification and evaluation. To ensure clean labeling in the prediction images, “surface defects” were called “defects.” The original image resolution was 1920×1080 pixel^2. These images were converted to 512×512 pixel^2 to accommodate the resolution requirements in the training process. Two backbones and three learning rates were optimized in the Mask R-CNN model. A novel semantic segmentation model with a lightweight backbone and Lovász-Softmax loss was also applied to better represent defects. Additionally, an algorithm was proposed to evaluate the severity of rail defects considering the ratio between the rail and the defect. The developed model includes a patch-wise training strategy to improve the training efficiency and accuracy. To enhance the detection performance and reduce possible overfitting, source images were subjected to image augmentations including image rotation, mirroring, and Gaussian noise. mAP, AP_{50} and AP_{75} constituted the evaluation metrics for model assessment. The rail orientation and different lighting conditions were also discussed. Traditional image processing and the optimized model were compared.

Training results with different patch sizes were identified, and rail severity was divided into four levels based on the collected data.

1.5 Research outline

To address the aforementioned issues and bridge the gap between academia and industry, this study is intended to enhance the railroad system via smart management and operations using proposed computer vision-based models such as object detection and segmentation neural networks. This research includes five studies. They respectively cover dense traffic detection at railroad grade crossings (**Chapter 3**), automatic railroad track component inspection using real-time instance segmentation (**Chapter 4**), real-time railroad track component inspection based on the improved YOLOv4 framework (**Chapter 5**), automatic rail surface defect inspection based on Mask R-CNN (**Chapter 6**), and a novel lightweight semantic segmentation model for rail surface defect detection and quantification (**Chapter 7**).

The dissertation is organized as follows: Chapter 2 presents a literature review on general object detection, instance segmentation, rail track component inspection, dense traffic instance detection, and rail surface defect segmentation. Chapter 3 outlines the proposed model, including training, validation, testing, and in-field application for dense traffic object detection at railroad grade crossings. Chapter 4 introduces the model design, training, validation, and testing with a real-world dataset for real-time rail track component inspection at railroad grade crossings using the proposed instance segmentation model. Chapter 5 depicts real-time rail track component inspection using the improved YOLOv4 model for better speed in real-world applications. Chapter 6

describes rail surface defect inspection using the instance segmentation model, Mask R-CNN. Chapter 7 presents a novel lightweight semantic segmentation model for enhanced speed and accuracy of rail surface defect inspection and quantification. Chapter 8 summarizes the findings of prior chapters and details recommendations for future research.

CHAPTER 2 LITERATURE REVIEW

This chapter provides a literature review of fundamental concepts in CNNs and three major research topics: dense traffic detection and counting, railroad track inspection, and railroad surface defect inspection and quantification. Regarding dense traffic detection and counting, general concepts including backbones, the feature pyramid network, detection head, and loss functions are introduced. The development of dense object detection and counting using the traditional image processing approach and CNN is reviewed as well. In terms of railroad track component inspection, image processing techniques such as the direction field (DF), linear discriminant analysis (LDA), and local normalization are analyzed and compared based on their advantages and disadvantages. This chapter also described prior use of CNN methods to inspect railroad track components with deep CNNs (DCNNs) and Faster R-CNN. Traditional image processing techniques and CNNs are presented to automatically analyze and detect rail surface defects. Lastly, an algorithm evaluating the severity of rail surface defects is proposed for quantification and rail track assessment.

2.1 Background on CNNs

A simplified CNN, as shown in Figure 2.1, includes three layers: the convolutional layer, pooling layer, and fully connected layer. The convolutional layer is responsible for feature extraction with a number of filters which can process image input and discern relevant features. The pooling layer, which follows the convolutional layer, reduces an image's dimensions by downsampling the operation, maintaining scale variance, and minimizing overall computation. The fully connected layer is typically placed before the output layer and represents the last few layers; it can classify high-dimensional features generated by previous feature maps which are concatenated into a

one-dimensional vector (Sony, Dunphy, Sadhu, & Capretz, 2021; Wang & Liu, 2017). Many deep learning libraries such as Pytorch (Paszke et al., 2019), TensorFlow (Abadi et al., 2016), and Keras (Géron, 2019) can be adopted for CNN model training.

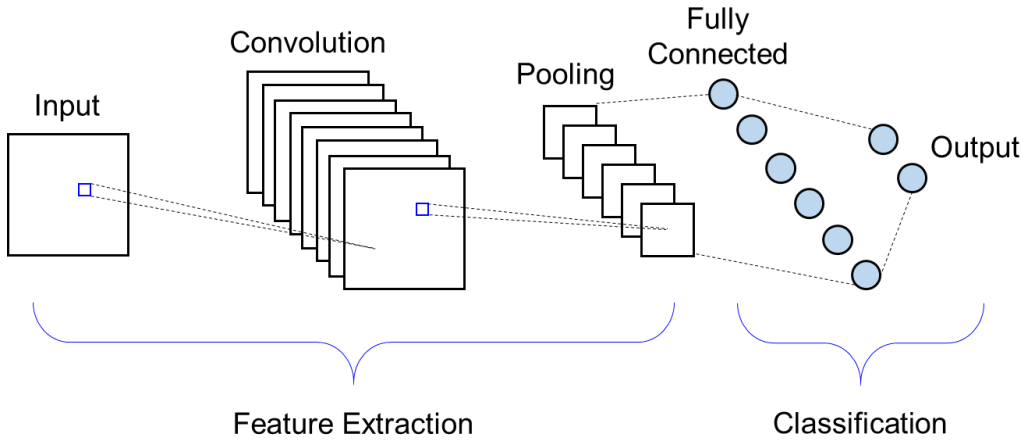


Figure 2.1. General CNN structure.

Owing to the swift development of GPUs and parallel computing, many CNN applications have emerged in engineering domains such as facial recognition, self-driving vehicles, medical image analysis, and structural health assessment that can benefit society. The steps of CNN model implementation cover data acquisition, model training, and model testing. International researchers have created freely available online image datasets with annotations such as COCO (Lin et al., 2014), Pascal VOC (Everingham, Van Gool, Williams, Winn, & Zisserman, 2010), and ImageNet (Krizhevsky, Sutskever, & Hinton, 2012). Yet individuals may still need to build customized datasets for specific scenarios (i.e., a dense traffic image database, railroad track components image database, and rail track surface defects image database in this case). Most model training is based on supervised learning, such that image data must be labeled before being fed into the neural network. Popular labeling tools include labelme (Russell, Torralba, Murphy, &

Freeman, 2008; Kentaro Wada, 2016), labelimg (Tzutalin, 2018), and VGG image annotator (Dutta & Zisserman, 2019). The graphical interface of labelme with Python is presented in Figure 2.2. Labels can be in different formats, such as .xml, .txt, and .json. The format of the original image and annotations must be paired correctly in the training procedure.

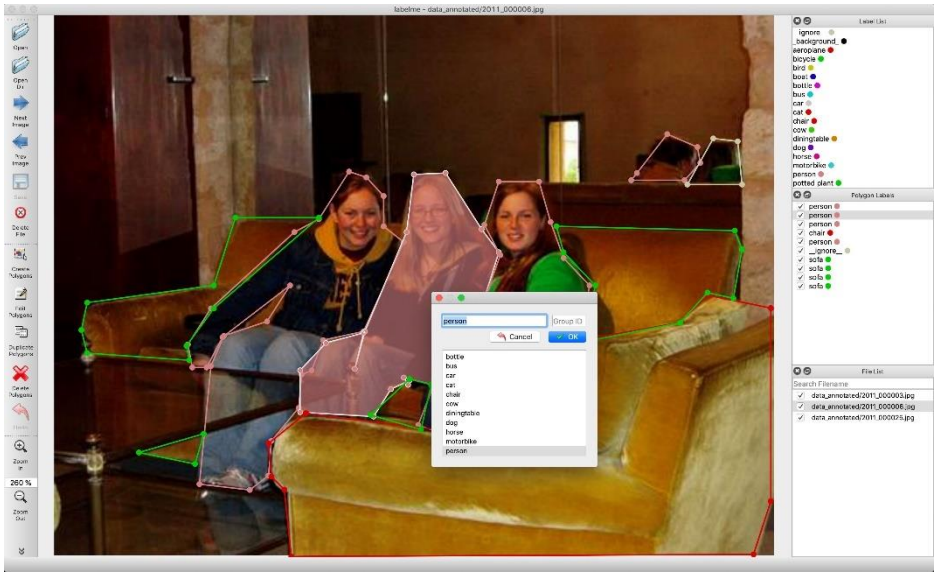


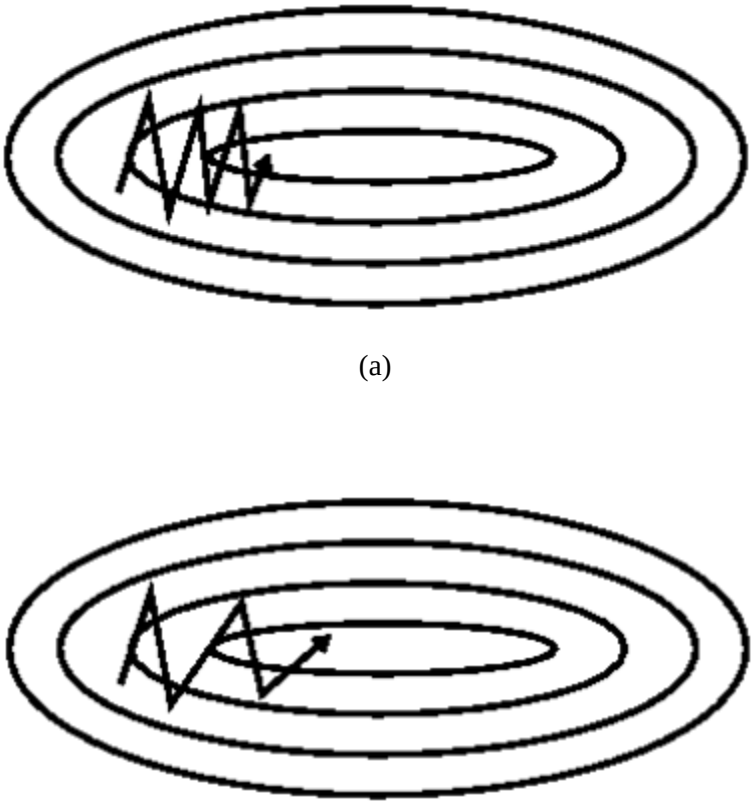
Figure 2.2. Graphical interface of labelme with Python (Kentaro Wada, 2016).

It is challenging to train a CNN model on a large image dataset. Loss functions, which compute the distance between current and predicted output, often have trouble converging. In addition, multiple powerful GPUs with large memories are needed for parallel computing to accelerate training (Aloysius & Geetha, 2017). When completing a specific task, it is more convenient to train the model using transfer learning instead of training it from scratch. Transfer learning reduces the time and computational cost because only the last few layers and the classifier layer (e.g., softmax classifier) are retrained.

Roughly, the training process involves adjusting the parameters of each layer and the model weight to make the output close to the ground truth. During the training process, hyperparameters related to backpropagation and gradient descent (e.g., the learning rate, batch size, number of epochs, and momentum) need to be set properly to optimize final predictions. The backpropagation algorithm refers to a computational process to look for the minimum of the error function in a weight space using gradient descent. Gradient descent requires the continuity and differentiability of the loss function because the gradient of the loss function must be calculated at each iteration step (Rojas, 1996). Gradient descent specifically involves repeatedly evaluating the gradient (commonly referred to as derivatives) and then performing a parameter update (Karpathy, 2016).

Compared with gradient descent, stochastic gradient descent (SGD) is more popular in application scenarios because it optimizes gradient descent in a stochastic approximation manner. With SGD, examples are randomly chosen at each iteration to compute the gradient rather than computing the gradient from the entire dataset; as such, the computational burden declines substantially and the convergence speed improves (Bottou, 2012). However, SGD often generates noisy steps towards the minima, resulting in large oscillation and slower convergence. Momentum offers a way to increase the dimensions whose gradient points are in the same directions while reducing updates to dimensions whose gradient points change directions (Ruder, 2016). Figure 2.3 depicts SGD with and without momentum. The learning rate determines how quickly the model updates its parameters (Karpathy, 2016). Batch size refers to how many samples are fed into the network after each update. Lastly, the number of epochs reflects the number of

times the entire dataset passes forward and backward through the network (Radhakrishnan, 2017).



(a)

(b)

Figure 2.3. (a) SGD without momentum; (b) SGD with momentum (Ruder, 2016).

2.2 Dense traffic detection and counting

2.2.1 Dense object detection

Object detection is a computer vision technique enabling identification of the object classes in images. Advances in object detection theory and applications have led to studies applying cutting-edge object detection in numerous contexts. Popular object detection frameworks can be divided into two categories: anchor-based approaches and anchor-free approaches (Zhang, Chi, Yao, Lei, & Li, 2020). Dense object detection is

more challenging than general object detection due to class imbalances, object size imbalances, dense sampling issues at object locations, and aspect ratios (Chabot, Pham, & Chaouch, 2019; Lin, Goyal, Girshick, He, & Dollár, 2017). Lin et al. (2017) proposed the idea of focal loss (FL) to balance the foreground and background. This method minimizes the weight of easy samples on the total loss and focuses on the training of hard negatives. Additionally, by shaping cross-entropy loss, the extreme foreground–background class imbalance is addressed. Redundant negatives can be prevented during detector training. A simple yet efficient detector featuring FL, RetinaNet (see Figure 2.4), was subsequently developed. RetinaNet matched the inference speed of one-stage detectors with greater detection accuracy than the two-stage detectors available in 2017.

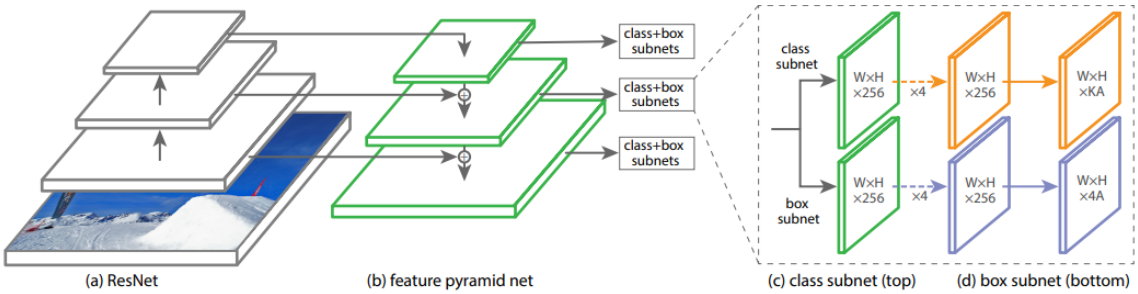


Figure 2.4. Structure of RetinaNet (Lin, Goyal, et al., 2017).

Following this idea, FL-related enhancements have been proposed such as class-imbalance loss (Tan, Guo, & Xiao, 2019), ω-FL (Ji, Kong, Wang, & Pang, 2019), generalized FL (GFL) (Li et al., 2020), and varifocal loss (Zhang, Wang, Dayoub, & Sünderhauf, 2020). Kant (2020) also extended RetinaNet by adding a Gaussian loss as an auxiliary branch in parallel with the original object classification and bounding box regression branches.

A recent work (Li et al., 2021) focusing on dense object detection and reliable localization quality estimation integrated a lightweight distribution-guided quality predictor and GFL version 2. This method makes use of real localization quality and distribution statistics in the dense object detection framework. Experimental results indicated accuracy improvements around 2 AP but without any loss in inference speed.

Notably, despite the number of CNN-based object detectors, most have been trained and tested on public datasets that include an array of object classes. These object detectors are hence distinct from customized datasets such as the dense traffic image database applied in this research. Using a general object detector trained on a public dataset would not be feasible in this case; rather, a specialized real-time detector for dense traffic at grade crossings is needed.

2.2.2 Dense object counting

Dense traffic counting is a subset of crowd counting, which usually involves estimating the number of pedestrians or dense vehicles (Liu, Salzmann, & Fua, 2019) in crowded scenes where occlusions make counting by detection impossible. Objects' exact coordinates are not needed in this counting task, which simply focuses on the number of objects. Many related studies (Kong, Gray, & Tao, 2006; Li, Yang, Zhu, Chen, & Guan, 2020; Ranjan, Le, & Hoai, 2018; Sindagi & Patel, 2017) have implemented counting based on density map estimation; in this case, a given crowd scene is mapped directly to its density in a simple yet efficient fashion.

With respect to dense traffic counting, Zhang et al. (2017) suggested using optimization-based and FCN-based methods to count the number of vehicles and estimate the vehicle density with low-resolution surveillance videos of busy traffic. Typically, the

manually selected hyperparameters on a Gaussian kernel or an adaptive bandwidth are used to generate a density map in an arbitrary domain for crowd recognition and counting.

Li et al. (2020) later formulated counting-driven attention networks (CODAN) for vehicle detection and counting in congested scenes. They mentioned that large scale variations, inconsistent distributions, and a diverse visual appearance render dense vehicle detection and counting (Figure 2.5[a]) more challenging than crowd counting (Figure 2.5[b]). To rectify the problem of crowd-counting methods being unsuitable for dense vehicle detection and to address constraints to mining spatial awareness, CODAN was proposed for vehicle detection and counting. Experimental results on four benchmark datasets were promising.

(a)

(b)

Figure 2.5. (a) Crowd scene; (b) Dense vehicle scene (Li et al., 2020).

Li et al. (2020) developed a vehicle counting network named the counting long short-term memory network. Experimental findings indicated that the proposed framework could attain a real-time image processing speed with 960 × 540 resolution images while maintaining a sound level of accuracy.

Wu et al. (2021) established an encoder-decoder network called the average up-sample convolution neural network (AU-CNN): a simple yet efficient up-sample module was proposed to gradually recover feature maps to their original size in the decoder system. Three public benchmarks were tested, namely ShanghaiTech, UCF_CC_50, and UCF_QNRF. Experiments demonstrated robust counting performance with the AU-CNN model.

2.3 Railroad track inspection

2.3.1 Track component inspection with image processing

Extensive effort has been devoted to developing automatic track inspection systems in the past few years. Yang et al. (2011) documented a DF-based method to detect absent fasteners as illustrated in Figure 2.6. The DF was extracted as the feature element for recognition, and the weight coefficient matrix was obtained based on LDA. Experimental results showed that the computation was efficient and robust under a complex environment. This detection approach also performed well on low-resolution images with 320 × 240 pixels taken from high-speed railways; however, the performance on high-resolution images was not discussed.

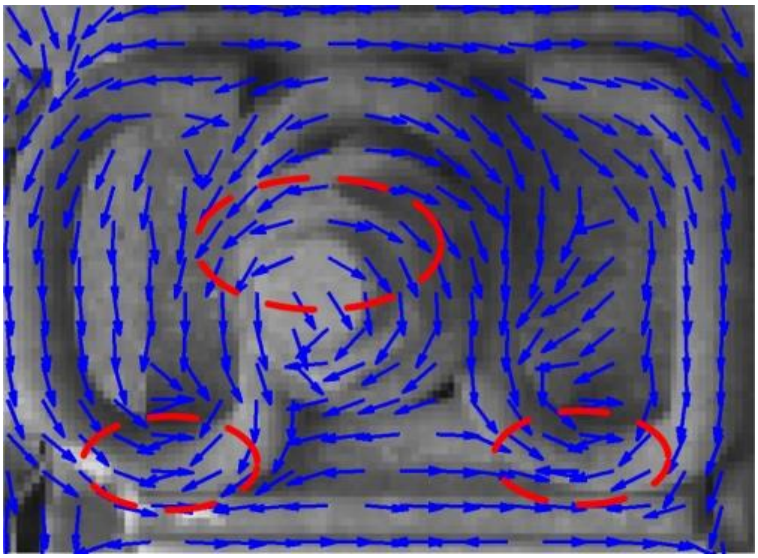


Figure 2.6. Region of fastener with DF method (Yang et al., 2011).

Resendiz et al. (2013) used Gabor filters and multiple signal classification (MUSIC) to perform periodicity detection on track components. Three algorithms were respectively put forth for component detection on railroad tracks, turnout detection, and

tie detection. MUSIC was proposed to detect periodicity in a 1-D signal. A particular Gabor filter was employed to conduct transform a panoramic image to Gabor representations. In practice, to simplify the test, only repeated railroad track components in the horizontal direction were considered. However, this method could not handle detection and segmentation simultaneously.

Feng et al. (2013) proposed a new probabilistic structure topic model (STM) to detect partially worn and missing fasteners. Compared with other approaches such as support-vector machine and AdaboostSTM, STM appeared more robust and could achieve higher precision when detecting fasteners with different orientations and lighting conditions. Multiple kinds of fasteners used in practice were recognized in each orientation, and the conditions of fasteners under different lighting conditions could be modeled independently. Yet STM required heavy computational power and was therefore incapable of an end-to-end test.

2.2.2 Convolutional neural network–based rail track component inspection

CNNs, which automatically learn input features efficiently, have recently enjoyed success in the field of computer vision. Many studies in civil engineering have applied CNNs for infrastructure damage detection/inspection. For instance, in bridge damage detection, bridge health inspections have included the Bayesian optimized deep learning model (Liang, 2019). Others have performed concrete bridge surface damage detection by using the improved YOLOv3 (Zhang, Chang, & Jamshidi, 2020) and crack evaluation of a high-rise bridge by using a modified SegNet (Jang, An, Kim, & Cho, 2020).

Meanwhile, few studies have implemented cutting-edge CNN models for the purposes of railroad track inspection and detection. Gibert et al. (2016) designed a new model leveraging the scalability of DCNNs to inspect rail ties and fasteners. The overall network design was based on the Caffe framework; the non-linear activation functions in different layers adopted rectified linear units (ReLUs). The shared features were found to save computational time, and the classification results were encouraging. However, more visualized results are needed.

Wei et al. (2019) compared dense scale invariant feature transform (Dense-SIFT), DCNNs, and Faster R-CNN for railroad fastener classification, localization, and detection. All acquired images came from Beijing Metro Line 6. The image data were pre-processed from grayscale to binary-scale. Spatial pyramid decomposition was adopted for feature extraction to produce better accuracy. Overall, Faster R-CNN demonstrated the lowest processing time and did not call for high computational power.

2.4 Rail surface defect inspection and quantification

2.4.1 Rail surface defect inspection with conventional image processing

Mandriota et al. (2004) utilized and compared three feature extractors – Gabor, wavelet, and Gabor wavelet filters – to extract rail defect textures. Their images were obtained from a 36h-long sequence acquired by a DALSA line scanner. However, a main problem in their study was that the three filters each required a large number of feature images, which are typically difficult to obtain.

Jie et al. (2009) proposed a rail head surface defect detection framework that included image pre-processing, defect locating, defect identifying, post-processing, and a

geometrical defect locating method. Yet noise in images greatly compromises defect detection, and the proposed approach to remove image noise required extensive validation. The framework thus warrants further investigation.

Li et al. (2012a) devised a real-time visual inspection system (VIS) to detect discrete rail surface defects. VIS comprises an image acquisition subsystem (IAS; see Figure 2.7) that contains a DALSA Spyder 2 line-scan camera with the resolution of 1024 pixels and a maximum line rate of 65000 lines/s. VIS aims to solve four challenging problems related to (a) limited features for recognition; (b) lighting inequality; (c) variation in rail surface reflection; and (d) the requirements for high-speed detection. VIS was found to have a recall rate of 93.1% and 80.41% on Type I and Type II defects, respectively.

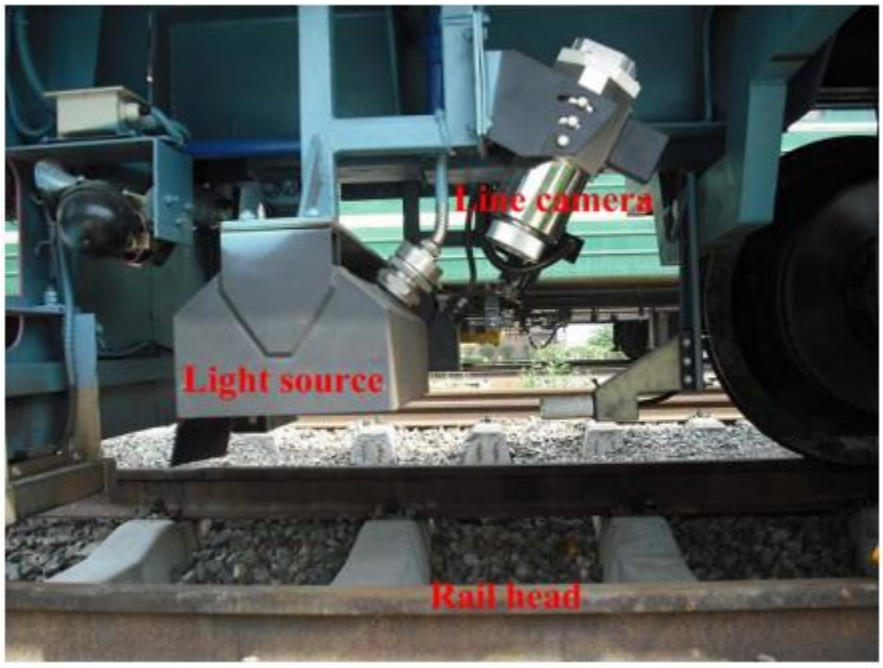


Figure 2.7. IAS system (Li & Ren, 2012a).

Li et al. (2012b) constructed an intelligent vision detection system for inspecting rail surface defects and addressed two issues of improving image quality and automatic thresholding by using the local Michelson-like contrast and the proportion emphasized maximum entropy thresholding algorithm. Although this method could effectively address the inspection of rail surface defects, its ability to differentiate rusted rail areas from areas with actual defects needs to be improved.

2.4.2 Rail surface defect inspection with deep learning

Of note, the above approaches focus on traditional hand-crafted feature learning to identify and classify rail surface defects. Technicians must therefore possess have rich experience in feature selection and training parameter adjustment. They also need access to a large amount of training data. Compared to those methods, deep learning approaches are more flexible and can automatically extract and learn problem-specific features from the original data without subjectively defining hand-crafted features. CNNs' ongoing development have contributed to key tasks in civil engineering, such as pavement crack detection (Yang et al., 2019; Zhang et al., 2017; Zhang, Cheng, & Zhang, 2018), concrete crack detection (Dung, 2019; Kim & Cho, 2018; Zhang, Rajan, & Story, 2019), and structural health monitoring (Azimi & Pekcan, 2020; Bao, Tang, Li, & Zhang, 2019; Kang & Cha, 2018). However, few studies have employed deep learning models to identify and characterize rail surface defects.

Faghih-Roohi et al. (2016) proposed using DCNNs to learn the features of rail surface defects. Three neural network structures (i.e., a small, medium, and large DCNN) were trained with two kinds of activation functions: Tanh and ReLU. The experimental

results showed that DCNN could detect rail surface cracks with high accuracy, and the large DCNN with ReLU outperformed the other models. The larger DCNN model also required a longer training time.

Shang et al. (2018) proposed a two-stage approach with a CNN to localize and classify rail surface defects. In the first stage, the training image was cropped to better focus on the rail part. In the second stage, a fine-tuned CNN was applied to extract rail defect features. Feng et al. (2020) designed M2-Y3 and M3-Y3 models for rail defect detection using YOLO (Redmon et al., 2016) and MobileNet (Howard et al., 2017). Although their model focused on identifying rail surface defects, it could not discern the shape of a given defect. Zhang et al. (2020) developed a multiple context information network to evaluate no-service rail surface defects (NRSDs); their network specifically integrated multiple pieces of contextual information, a pyramid pooling module, an attention mechanism, and information fusion. Both original and artificial NRSDs were tested, with results revealing reasonable segmentation performance. Figure 2.8 provides examples of man-made NRSDs.

Figure 2.8. Examples of man-made NRSDs (Zhang et al., 2020).

By combining unsupervised learning with an improved Gaussian mixture model for rail surface defect segmentation, coupled with supervised learning with Faster R-CNN for rail surface defect localization, Lu et al. (2020) proposed SCueU-Net: their network incorporates U-Net and the saliency cues method to automatically detect rail surface defects. Unfortunately, only 201 samples were included in their study after data augmentation. Potential overfitting during training was not addressed.

2.5 Summary

This chapter presented an overview of computer vision techniques related to transportation infrastructure, especially on railroads. Three major topics were considered: dense traffic detection and counting, railroad track inspection, and rail surface defect inspection and quantification. Dense traffic detection and counting were also described, specifically methods informed by CNN-based approaches entailing different network designs, loss functions, and uses.

This chapter also offered a thorough review of the literature on railroad track inspection with a focus on research featuring traditional image processing techniques and deep learning networks. The advantages and disadvantages of several approaches were analyzed on the basis of algorithmic complexity, inference speed, and image resolution. Rail surface defect inspection methods were revisited in this chapter as well; pre-designed feature extractors and CNN-based auto feature extractors were considered under different environmental conditions. Associated challenges and accomplishments were outlined in each section. The following chapter describes the proposed frameworks to address current problems related to railroad infrastructure.

CHAPTER 3 DENSE TRAFFIC DETECTION AT HIGHWAY-RAILROAD GRADE CROSSINGS[1]

[1] Feng Guo, Zhuocheng Jiang, Yi Wang, Chen Chen, and Yu Qian, Dense traffic detection at highway-railroad grade crossings, Submitted to IEEE Transactions Intelligent Transportation System.

In the United States, highway-railroad grade crossings are easily congested, which not only causes significant traffic delays to travelers but also brings potential threats to the first responders for emergencies. Unfortunately, very limited research efforts have been dedicated to developing practical systems that can assess traffic conditions at overcrowded grade crossings. The main challenge in evaluating the congestion conditions at the crossings is the different instance classes (i.e., vehicle, train, and pedestrian) that need to be accurately detected, especially when densely packaged.

In this chapter, a novel convolutional neural network (CNN) named dense traffic detection net (DTDNet) is developed. DTDNet proposes to integrate the Transformer Attention (TA) module for better modeling of global context information and the learning-to-match detection head for optimizing object detection and localization using a likelihood probability fashion. To train and test DTDNet, a unique grade crossing traffic image dataset including congested and normal traffic during both daytime and nighttime is established. Experimental results on the dataset show that the proposed DTDNet achieves the maximum mean average precision (mAP) value, 0.832, outperforming the other state-of-the-art (SOTA) models. Field test results with low mean average error (MAE), mean relative error (MRE), and root mean squared error (RMSE) which are 2.200, 1.890, and 0.280, respectively suggest the proposed model has a satisfying and robust performance in the field application under different environments.

3.1 Introduction

Due to historical and practical reasons, trains have the right-of-way at railroad-highway grade crossings in the United States (Estes & Rilett, 2000). It is common to see serious traffic congestion in highly populated areas during rush hours because a train is slowly passing by or completely stops at the grade crossing (Soleimani et al., 2019). The unpredictable crossing blockages not only introduce significant delays to motorists but also raise serious concerns to first responders on duty (Ma et al., 2018; Park et al., 2016). To better manage and ease congested traffic at grade crossings, the very first step is to have an intelligent transportation system (ITS) to automatically evaluate the congestion level in an accurate yet efficient manner (Lv et al., 2014). In other words, it is important to quickly and accurately determine how many vehicles are waiting in the queue and how long it would take for the traffic to get back to normal and to share the information with the public and the first responders.

Leveraging rapid development of deep learning, several studies (Fan et al., 2016; Hu et al., 2018; Xu et al., 2017; Yu et al., 2021) have applied cutting-edge models, such as YOLO (Redmon et al., 2016) and Faster R-CNN (Ren et al., 2016) to traffic detection or vehicle counting on the street. However, these traffic detection or vehicle counting approaches often fail to achieve satisfactory results due to the following reasons: (1) limited and low-resolution video source which cannot preserve enough details and leads to misdetection by convolutional neural networks (CNNs).

To control the budget, most of the transportation agencies, such as the department of transportation (DOT) have to use low-cost low-resolution cameras. For example, the video shared by a local agency in this study only has a resolution of 640×480, which is considerably lower than a high-quality video resolution (i.e., 2K or 4K); (2) the scale varies dramatically due to the large differences in the distances between the camera and the objects of interest. In other words, similar vehicles that are close to the camera or far away from the camera have quite different sizes on the same image (see Figure 3.1(a)). Most CNNs are very sensitive to the scale changes. It is difficult for a network to accommodate all scales simultaneously while maintaining the optimal confidence scores; (3) varied features during the daytime and the night.

Few studies have reported the detection performance during the night. The reason behind this could be dense traffic congestions often happen during the daytime, and traffic congestions during the night are not as important as those during the daytime. However, the reality is the congestions during the nighttime could be comparable to the congestions during the daytime (see Figure 3.1(b)). Thus, nighttime congestions should not be underestimated; and (4) serious imbalance of different object classes. There are multiple object classes (e.g., train, pedestrian, semi-truck, sedan, etc.) in the complex scenario of a grade crossing. Typically, the vehicles dominate the scene while the pedestrian and train occupy a small ratio, which introduces a serious class imbalance issue for object detection. Furthermore, the occlusion between different vehicles and oversized objects, such as the semi-trucks, makes the detection and classification more challenging.

(a)

(b)

Figure 3.1. Visualization of a dense traffic scene at a grade crossing. (a) an example daytime view of the congested traffic; (b) an example nighttime view of the congested traffic.

To address the above four challenges, the proposed dense traffic detection net (DTDNet) integrates the attention mechanism and maximum likelihood estimation (MLE) to detect congested vehicles under low resolution images and significantly varied object scales. Different features of the vehicle object class during daytime and nighttime are selected for training to make the accurate detection under nighttime possible. In addition, Focal Loss (FL) (Lin, Goyal, et al., 2017) is used to cope with the imbalance of object distribution.

Specifically, as shown in Figure 3.2, there are three stages in this chapter. In stage 1, the dense traffic data at a specific grade crossing has been collected and pre-processed. In stage 2, DTDNet has been built, trained, and compared with seven state-of-the-art (SOTA) models. In stage 3, DTDNet has been tested for dense traffic detection and vehicle counting at a real grade crossing. The proposed approach paves the way towards grade crossing traffic management, and other relevant railroad applications, such as pedestrian trespassing detection and train arrival recognition.

The contributions of this chapter are summarized as follows:

- The shortfall of detecting multiple traffic instances at the highway-railroad grade crossings of SOTA models has been filled with our proposed DTDNet.
- The feasibility of using the transformer attention (TA) module to characterize long-distance relationships and the MLE procedure to detect heavily packed objects has been discussed and tested.
- The detection performance under different environmental conditions including daytime, nighttime, and haze weather has been evaluated and compared on our proposed model and other SOTA models.
- Detection-based vehicle counting has been tested. Experimental results based on a railroad grade crossing image database have verified our model's robustness and superiority.

In short, our proposed DTDNet outperforms the other SOTA models in terms of detecting and counting vehicles in daytime, nighttime and the light haze condition at the congested highway-railroad grade crossing, which may benefit the industry on future field applications such as pedestrian monitoring, vehicle congestion management, and train arrival recognition.

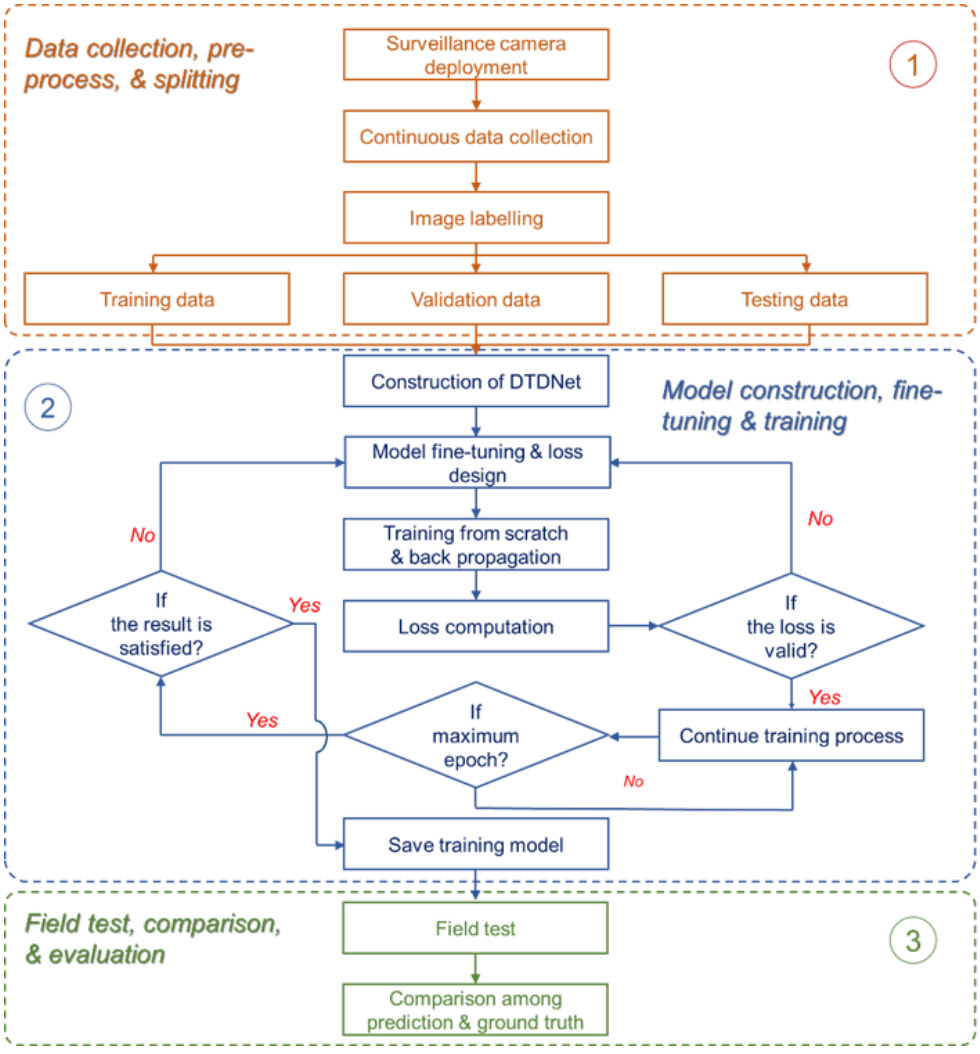


Figure 3.2. The flow chart of chapter 3.

3.2 Proposed method -- DTDNet

There are three main parts of the proposed DTDNet which are baseline, the TA module, and the FreeAnchor head for addressing the difficulties in the dense traffic at the grade crossing. The details of each component are presented in the following.

3.2.1 Baseline

In this chapter, the main structure of our proposed DTDNet is based on RetinaNet as shown in Figure 3.3 which includes the backbone, the Feature Pyramid Network (FPN), and the detection head. In short, the backbone is responsible for feature extraction. FPN is used to generate rich, multi-scale convolutional features through a top-down pathway and lateral connections. The detection head, including two subset networks, is designed for object classification and bounding box regression.

ResNet (He, Zhang, Ren, & Sun, 2016) is a popular CNN backbone because it is convenient to optimize and can improve the accuracy with an increased neural network depth. ResNet-50 which includes 50 layers is utilized for the design of the proposed DTDNet. The feature pyramid levels from P3 to P5 in FPN are computed from the output of the corresponding residual stages from C3 to C5 in ResNet using lateral connections and a top-down pathway. To facilitate visualization, Figure 3.3 does not show P6 and P7. Regarding the detection head, three aspect ratios (1:2, 1:1, and 2:1) have been selected in each pyramid level. That is, there are nine anchors per level and these anchors can cover 32 to 812 pixels per image.

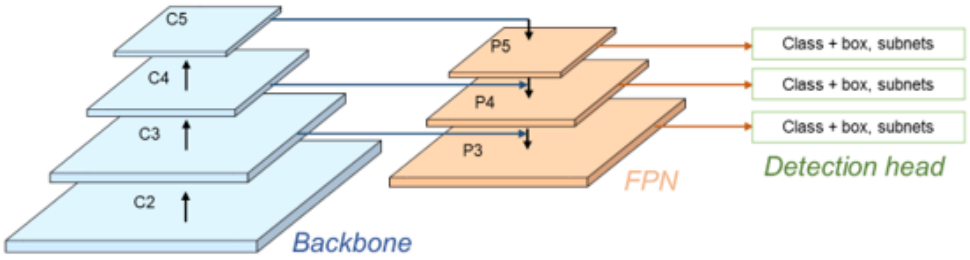


Figure 3.3. Overview of RetinaNet.

To deal the issue of imbalanced classes between the foreground and the background, improve detection accuracy of dense and small objects in our specific scene, FL is implemented in the proposed DTDNet. Before introducing FL, the cross-entropy (CE) loss for binary classification needs to be explained first. Equation (3-1) describes the CE loss in a binary fashion.

$$\mathrm{CE}(p,y)=\begin{cases} -\log(p), & \text{if } y=1 \\ -\log(1-p), & \text{otherwise} \end{cases} \tag{3-1}$$

$y\in[\pm 1]$ represents the ground truth class and $p\in[0,1]$ means the estimated probability of the model when label $y=1$.

To simplify the notation expression, p_t is defined in Equation (3-2) and the CE loss can be reorganized in Equation (3-3).

$$p_t=\begin{cases} p, & \text{if } y=1 \\ 1-p, & \text{otherwise} \end{cases} \tag{3-2}$$

$$\mathrm{CE}(p_t)=-\log(p_t) \tag{3-3}$$

To balance the positive and negative cases, the weighting factor $\alpha\in[0,1]$ is commonly used for class 1 and $1-\alpha$ is for class -1. There is a need to classify the easy and hard examples.

To address this issue, the loss function is reshaped by adding a modulating factor $(1-p_t)^{\gamma}$ on CE loss to FL. The definition of FL in practice can be expressed in Equation (3-4):

$$\mathrm{FL}(p) = -\alpha_t (1-p_t)^{\gamma} \log(p_t), \tag{3-4}$$

where $\alpha \in [0,1]$ is a weighting factor.

3.2.2 Transformer attention module

Attention mechanism is initially developed in natural language processing (NLP) but quickly becomes popular in the CNN design, due to its significant contribution to improving the accuracy by focusing on relevant elements rather than irrelevant parts from an input image (Bahdanau, Cho, & Bengio, 2014; Carion et al., 2020; Zhu, Cheng, Zhang, Lin, & Dai, 2019). *Transformer* is a simple network structure entirely relying on self-attention to compute representations of its input and output. It has been successfully applied in object detection and semantic segmentation (Sun, Cao, Yang, & Kitani, 2020; Zhao et al., 2018; Zheng et al., 2021). The reason to unify a TA module into our proposed model is twofold: (1) It is an architecture for sequence prediction, which can model pairwise intersections between elements in a sequence, therefore restricting the duplicate prediction in the end of the queue (Carion et al., 2020). (2) It is expressive of modeling long-range relationships (i.e., global context information), which can benefit the detection accuracy of long object such as the train at the grade crossing. Therefore, compared to other plain models, the implement of TA module can improve the feature descriptions on both dense vehicles and the train with a long distance.

Based on Zhu et al. (X. Zhu et al., 2019), there are four attention factors $[\varepsilon_1, \varepsilon_2, \varepsilon_3, \varepsilon_4]$ which are: (1) the query (image pixels) and key (regions of interest) content, (2) the query content and its relative position, (3) the relative position, and (4) the query content needs to be considered in the attention module. The TA module included in our model contains four terms aforementioned and their summation to formulate the attention weight. Considering a query element and a set of key elements, the outputs of multiple attention features y_q can be computed as:

$$y_q = \sum_{m=1}^{M} W_m [\sum_{k \in \Omega_q} A_m(q, k, z_q, x_k) \odot W_m^{'} x_k] \tag{3-5}$$

where W_m and $W_m^{'}$ are the weights with learnable abilities. m is the attention head. q denotes a query element with content z_q. k represents a key element with content x_k. Ω_q denotes the key region of the query. $A_m(q, k, z_q, x_k)$ represents the attention weight in the m-*th* attention head.

Specifically, the sum of four terms $\{\varepsilon_j\}_{j=1}^{4}$ can be used to compute the attention weight of each query-key pair as follows:

$$A_m^{Trans}(q, k, z_q, x_k) \propto \exp(\sum_{j=1}^{4} \varepsilon_j) \tag{3-6}$$

There are eight attention heads in the experimental setting. It is necessary to point out that the first two terms (ε_1 and ε_2) in the attention factors are sensitive to the query content, while the last two terms (ε_3 and ε_4) are independent of the query content. ε_1 describes the compatibility of the query and key content. ε_2 measures the query content and its relative position. ε_3 denotes the key content only. ε_4 involves the relative position only. The following Equations (3-7) to (3-10) describe the four items one by one.

$$\varepsilon_1 = z_q^{\mathrm{T}} U_m^{\mathrm{T}} V_m^C x_k \tag{3-7}$$

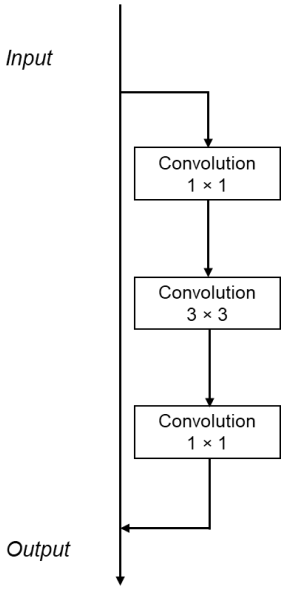
$$\varepsilon_2 = z_q^{\mathrm{T}} U_m^{\mathrm{T}} V_m^R R_{k-q} \tag{3-8}$$

$$\varepsilon_3 = u_m^{\mathrm{T}} V_m^C x_k \tag{3-9}$$

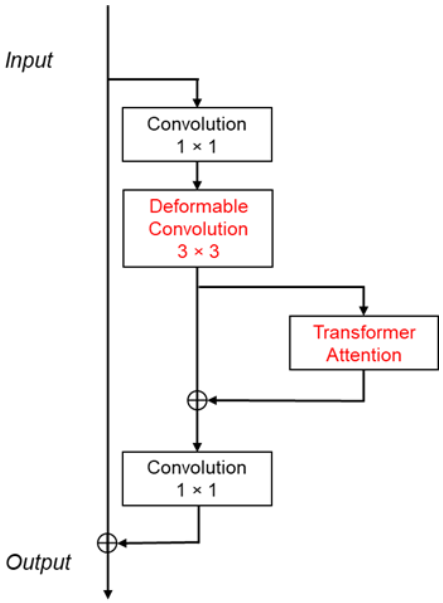
$$\varepsilon_4 = v_m^{\mathrm{T}} V_m^R R_{k-q} \tag{3-10}$$

where U_m and V_m^C are embedding matrices with the learnable property for the query and content, respectively. R_{k-q} represents the relative position of $k-q$ with projecting computation. u_m is a learnable vector which involves the key content and is irrelevant to the query content. v_m is a learnable vector which denotes the global errors among the query and the key content.

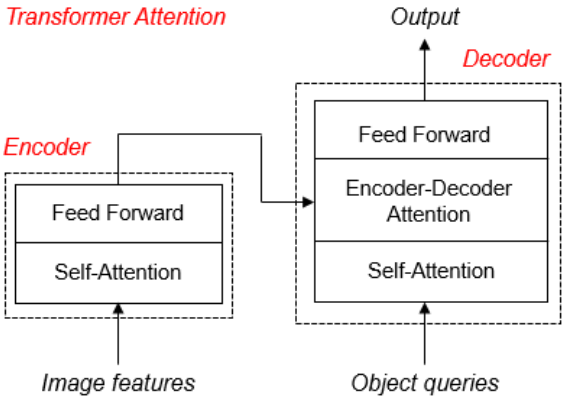
Figure 3.4 presents the structural design of the TA module integrated in the residual block of ResNet-50. Compared to the traditional design of the residual block as shown in Figure 3.4 (a), the TA module in the proposed DTDNet connects with other convolution output in series and follows the output of 3×3 deformable convolution in the residual block. Based on (X. Zhu et al., 2019), to avoid interruption (i.e., unmatched keys and values) of the initial behavior of choosing a pre-trained model in the training, the output of the TA module is multiplied by a learnable scalar which is initialized to zero. The applied residual block in the proposed neural network is shown in Figure 3.4 (b) using TA plus deformable convolution to achieve better accuracy on the dense traffic object detection in the crossing scene. Figure 3.4 (c) presents the structure of transformation module which includes encoder and decoder.



(a) The traditional residual block



(b) The proposed residual block



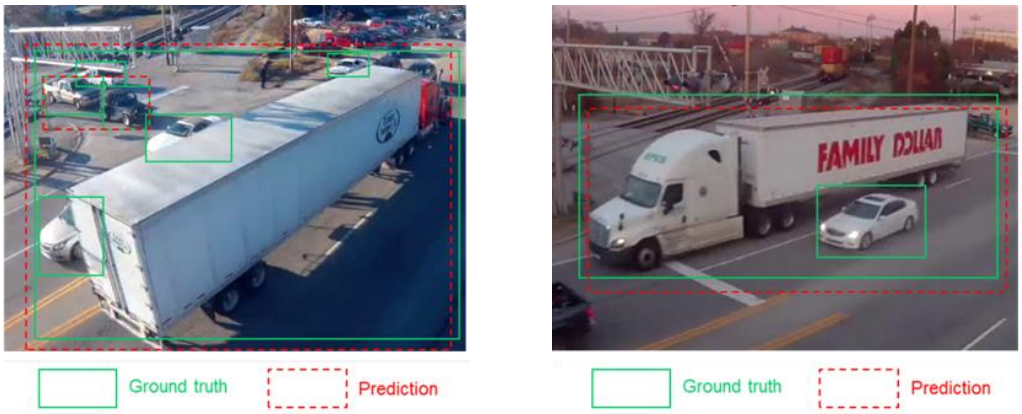
(c) The transformation attention module

Figure 3.4. The design of residual blocks.

3.2.3 FreeAnchor head

The intersection over union (IoU) is a leading voice in the design of anchor-based detectors. However, based on (Zhang, Wan, Liu, Ji, & Ye, 2019), IoU fails on two perspectives. On the one hand, it is not best choice for the match of anchors or features when there are multiple objects exist simultaneously. On the other hand, it can fail on

coping with the representative features on slender objects. As shown in Figure 3.5, there are a number of passenger cars but the semi-truck trailer occupies a large spatial area. On the one side, these passenger cars are close to each other and overlap in the image. On the other side, the centroid of the trailer is close to the centroids of the cars, which confounds the detector in locating and classifying the passenger cars.



(a) (b)

Figure 3.5. Illustration of false detections with hand-crafted IoU criterion at a highway-railroad grade crossing.

To address the aforementioned issue and better match the objects and anchors for visual object detection, the proposed DTDNet adopts the FreeAnchor (Xiaosong Zhang et al., 2019) method which optimizes the anchor assignment mechanism using a learning-to-match approach to achieve high recall and precision rates. Meanwhile, it is compatible with the non-maximum suppression (NMS) procedure which is critical to select the most appropriate bounding box for the object. Specifically, the anchor bag is constructed by selecting top-ranked anchors (n) from candidate anchors for each object.

A novel loss function, aiming to break the limitations of pre-assigned anchors using MLE to train detectors, is derived and presented by Equations (3-11) to (3-16). At first, the training loss of RetinaNet based on IoU criterion shown in Equation (3-11) is converted into a probability as Equation (3-12) presented.

$$L(\theta) = \sum_{a_j \in A_+} \sum_{b_i \in B} C_{ij} L_{ij}^{cls}(\theta) + \beta \sum_{a_j \in A_+} \sum_{b_i \in B} C_{ij} L_{ij}^{loc}(\theta) + \sum_{a_j \in A_-} L_j^{bg}(\theta) \tag{3-11}$$

where B denotes ground truth annotation. $C_{ij} \in \{0,1\}$ indicates whether object b_j matches anchor a_j based on the threshold of the IoU. If the threshold is smaller than a_j and b_j , then, $A\_ \in A$ and $C_{ij} = 0$. Otherwise, $A_+ \in A$ and $C_{ij} = 1$. $L_{ij}^{cls}(\theta)$ and $L_j^{bg}(\theta)$ use Binary Cross Entropy (BCE) loss for classification. $L_{ij}^{loc}(\theta)$ uses Smooth L1 loss for localization. bg represents background.

Following MLE procedures, Equation (3-12) can be derived from Equation (3-11).

$$\begin{aligned} P(\theta) &= e^{-L(\theta)} \\ &= \prod_{a_j \in A_+} (\sum_{b_i \in B} C_{ij} e^{-L_{ij}^{cls}(\theta)}) \prod_{a_j \in A_+} (\sum_{b_i \in B} C_{ij} e^{-L_{ij}^{loc}(\theta)}) \prod_{a_j \in A_-} e^{-L_j^{bg}(\theta)} \\ &= \prod_{a_j \in A_+} (\sum_{b_i \in B} C_{ij} p_{ij}^{cls}(\theta)) \prod_{a_j \in A_+} (\sum_{b_i \in B} C_{ij} p_{ij}^{loc}(\theta)) \prod_{a_j \in A_-} p_j^{bg}(\theta) \end{aligned} \tag{3-12}$$

where $p^{L_{ij}^{cls}}(\theta)$ and $p_j^{bg}(\theta)$ represent the confidence of classification. $p^{L_{ij}^{loc}}(\theta)$ indicates the confidence of localization. A lower loss function value in $L(\theta)$ represents a higher value in the likelihood probability in $P(\theta)$.

After constructing the anchor bag and optimizing the recall rate and precision based on MLE, the detection likelihood can be computed in Equation (3-13). Then, the detection customized loss function is derived and shown in Equation (3-14).

$$
\begin{aligned}
P'(\theta) &= P_{recall}(\theta) \times P_{precision}(\theta) \\
&= \prod_i \max_{a_j \in A_i} (p_{ij}^{cls}(\theta) p_{ij}^{loc}(\theta)) \\
&\quad \times \prod_j (1 - P\{a_j \in A\_\}(1 - p_j^{bg}(\theta)))
\end{aligned}
\tag{3-13}
$$

$$
\begin{aligned}
L'(\theta) &= -\log P'(\theta) \\
&= -\sum_i \log(\max_{a_j \in A_i} (p_{ij}^{cls}(\theta) p_{ij}^{loc}(\theta))) \\
&\quad -\sum_j \log(1 - P\{a_j \in A\_\}(1 - p_j^{bg}(\theta)))
\end{aligned}
\tag{3-14}
$$

Because neither the anchors with low confidence score at the early training stage nor the anchors with high confidence score are suitable for training, the Mean-max function (see Equation (3-15)) is used to select the anchors. Substituting Mean-max function in Equation (3-15) and applying FL, the loss function of the FreeAnchor head in our proposed framework can be obtained by Equation (3-16).

$$
\text{Mean-max}(X) = \frac{\sum_{x_j \in X} \frac{x_j}{1 - x_j}}{\sum_{x_j \in X} \frac{1}{1 - x_j}}
\tag{3-15}
$$

$$
\begin{aligned}
L''(\theta) &= -\omega_1 \sum_i \log(\text{Mean} - \max(X_i)) \\
&\quad + \omega_2 \sum_j FL(P\{a_j \in A\_\}(1 - P_j^{bg}(\theta)))
\end{aligned}
\tag{3-16}
$$

where $X_i = \{P_{ij}^{cls}(\theta) P_{ij}^{loc}(\theta) \mid a_j \in A_i\}$ represents the likelihood set corresponding to an anchor bag A_i . $\omega_1 = \frac{\alpha}{\|B\|}$, $\omega_2 = \frac{1 - \alpha}{n\|B\|}$, $FL(x) = -x^{\gamma} \log(1 - x)$. α and γ are from FL in Equation (3-4).

For this chapter, the inputs are images with highly congested traffic. After the feature extraction in the backbone with the TA module, the features are encoded into a certain representation with more focus on the congested area. Through FPN, the ability of feature extraction at different scales is significantly improved using the top-down pathway and lateral connections. In the proposed DTDNet as shown in Figure 3.6, the feature map size decreases 0.5 times each layer through the bottom-up in the backbone

and increases 2 times through top-down in FPN. The TA module can benefit the long object detection such as the train instance in our case. The original detection head in RetinaNet that follows the IoU criterion is replaced by the learning-to-match-based detection head that uses the MLE procedure for training. At the end, the detected, well classified, and localized congested traffic objects in each image are the outputs of the proposed neural network architecture.

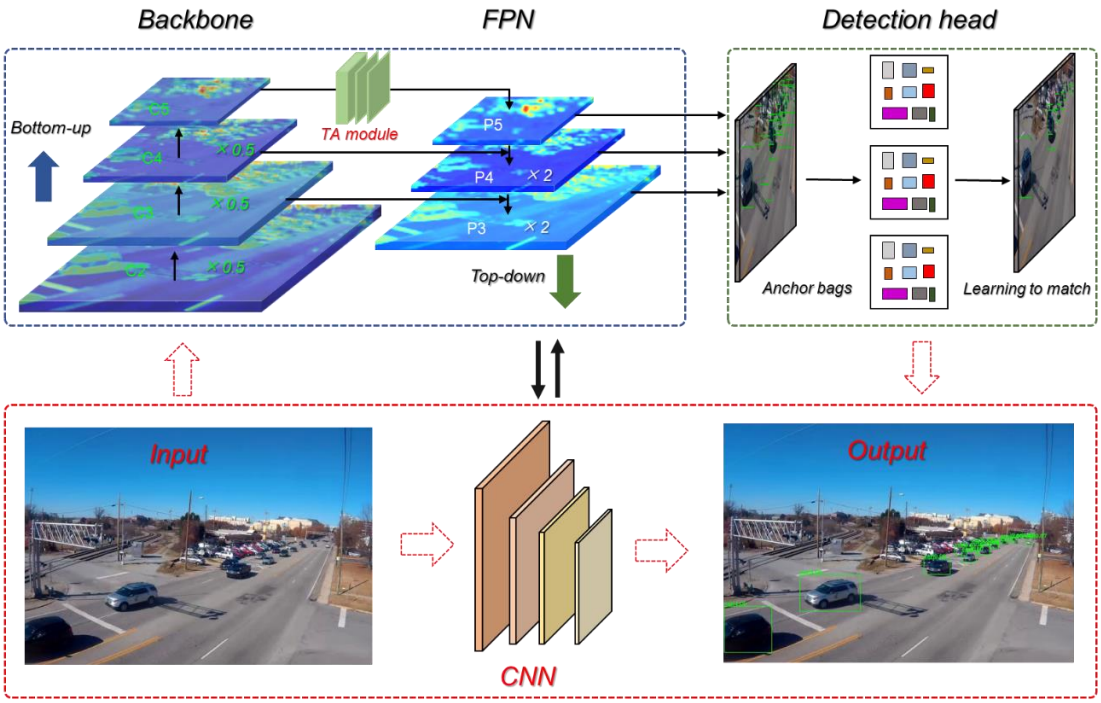


Figure 3.6. Overview of DTDNet.

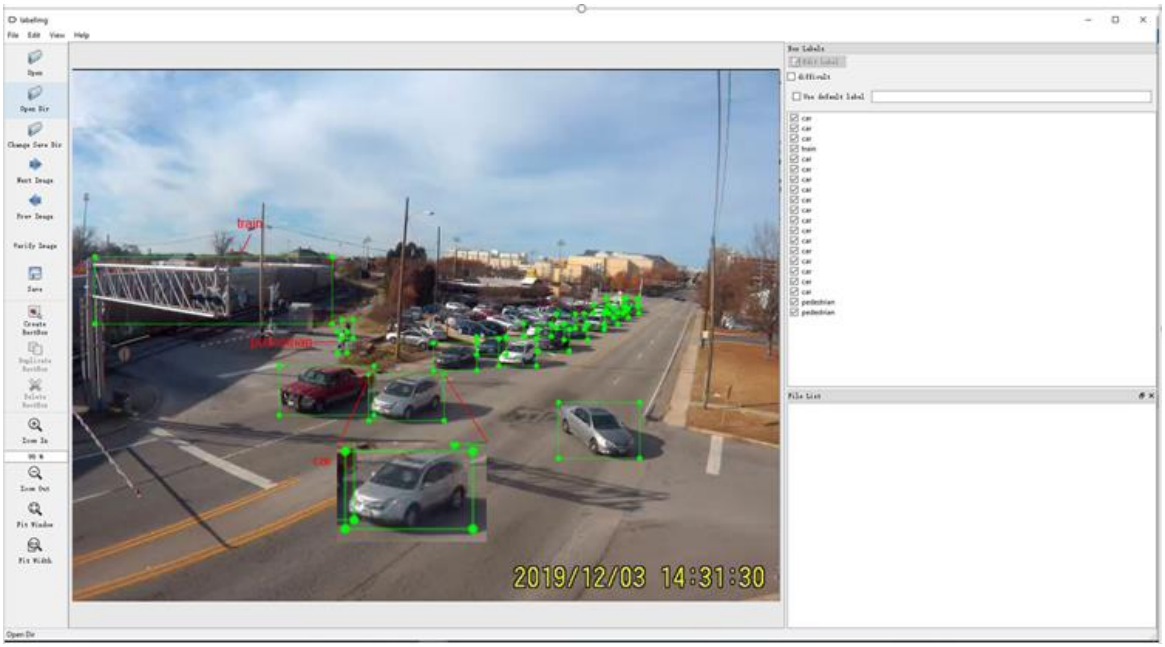
3.3 Experiments and results

3.3.1 Dataset and experimental setup

The image database was built through the video clips collected from a surveillance camera installed on the top of a power pole beside a grade crossing, near the main campus of the University of South Carolina (UofSC), Columbia, SC. The 100 hours of traffic surveillance video clips were collected at 640×480 and 10 frames per second

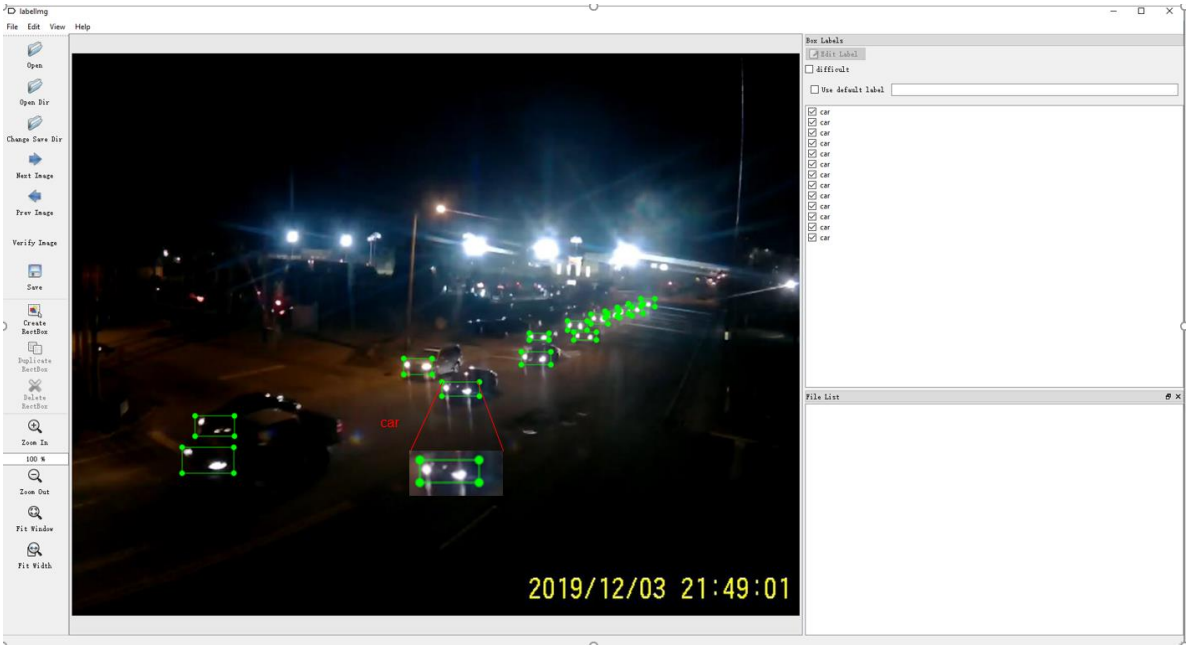
(FPS) in November and December 2019. There were 2,358 images in the image database, which covered both the daytime and night. No rain and foggy weather happened during data collection. Three object classes (i.e., vehicle, pedestrian, and train) were included in this study. A labeling tool, LabelImg[2], was used for ground truth labeling and annotation generation. The image labeling details were illustrated in Figure 3.7. Based on labeling results, there were 4835 vehicles, 477 pedestrians, and 36 trains of the ground truth data, reflecting the serious imbalanced object distribution.

In detail, Figure 3.7 (a) and (b) depict the grade crossing images under daytime condition and nighttime condition, respectively.



(a)

[2] https://github.com/tzutalin/labelImg



(b)

Figure 3.7. Image labeling with LabelImg. (a) daytime scene; and (b) night scene.

Specifically, training set, validation set, and test set were included during training. They were used to fit the parameters, predict the responses for the observation, and provide an unbiased evaluation on a final model, respectively. The ratio between the training set, validation set, and testing set, was 6:2:2. Although the congestion during the night could be comparable to the congestion during the daytime, the frequency of congestion happening during the night was relatively low. Therefore, the case ratio between the daytime and the night was around 4:1. In the daytime, the whole body of a vehicle was labelled. To better reflect the feature of a vehicle during the night, only the headlights were labelled. To fairly compare training and testing performance, all the models were trained from scratch. In other words, no transfer learning or a pre-trained weight was used in the experiments.

All the models were trained and tested on MMDetection[3] which was an open-source object detection toolbox based on the Pytorch framework developed by Facebook Inc. Besides the proposed model, seven well-recognized SOTA models under different categories were also considered for comparison, including ATSS (S. Zhang et al., 2020), SSD (Liu et al., 2016), Faster R-CNN (Ren et al., 2016), RetinaNet (Lin, Goyal, et al., 2017), FreeAnchor (Xiaosong Zhang et al., 2019), FSAF (Zhu, He, & Savvides, 2019), and Cascade R-CNN (Cai & Vasconcelos, 2018). The detectors used in these seven models can be grouped into two categories. On the one hand, they can be classified as single-stage or two-stage. On the other hand, they can be organized as an anchor-based detector or an anchor-free detector. Figure 3.8 illustrates the classification of each model.

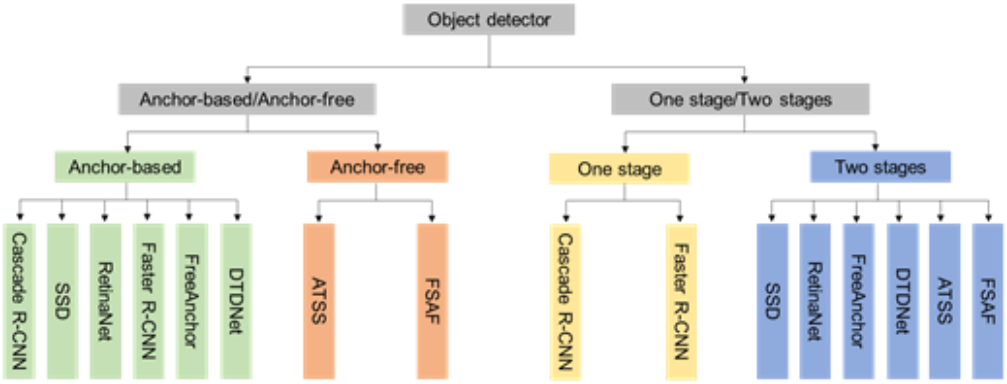


Figure 3.8. Classification of detectors trained in this study.

For this study, the MMDetection version is 2.0 and the Pytorch library version is 1.6. The experiments are performed on a deep learning workstation with four Nvidia 2080 Ti GPUs. The libraries of CUDA and cuDNN developed by Nvidia, have been installed and configured in the workstation, providing high-performance GPU acceleration. The CUDA version and cuDNN version are 10.2 and 7.5.2, respectively.

[3] https://github.com/open-mmlab/mmdetection

For each test, because the dataset is relatively small, a single GPU is used during the experimental training and testing. According to (Kisantal, Wojna, Murawski, Naruniec, & Cho, 2019), the detection accuracy can benefit from a large-size image. Hence, the training scale per image has been augmented from [512, 512] to [1333, 800]. To better present the training strategies, Algorithm 3-1 summarizes the pseudocode of the training steps.

Algorithm 3-1 Training scheme of DTDNet

Input:

Given input image data collected at the railroad grade crossing $I_{gt} = [I_1, I_2 ... I_n]$.

Annotation data of ground truth: $L \leftarrow I_{gt}$.

Output:

W_{dtd} : DTDNet model weight

1: Load training set, validation set, and test set

2: **for** the training set images of I_{gt} and their corresponding labels in L **do**

Image resize: $[1333, 800] \leftarrow [640, 480]$

Random flip, Normalization

collect I_{train}, I_{label}

end for

3: W, $bias$ $\leftarrow$ Initialize model parameters

4: **for** i =1: **do**

Forward propagation, Loss computation, Backward propagation

Until $\chi \leftarrow$ Maximum epoch

end for

5: **return** W_{dtd}

3.3.2 Ablation study

To investigate the functionality of each component of DTDNet, ablation tests were conducted. Detailly, this section demonstrates precision and inference speed results of four network designs. Table 3.1 and Figure 3.9 show test results with different components of DTDNet. The reason for choosing the image size of 1333×800 to train and test models is that a large image size can benefit detection accuracy. However, it

needs to mention that the enlarged image size also sacrifices inference speed. In Table 3.1, it can be easily found that the overall mAP value (83.2) of our model is 6.9%, 6.4%, and 2.6% higher than RetinaNet, RetinaNet + TA, and RetinaNet + FreeAnchor head, respectively, indicating a promising detection accuracy on overall object classes.

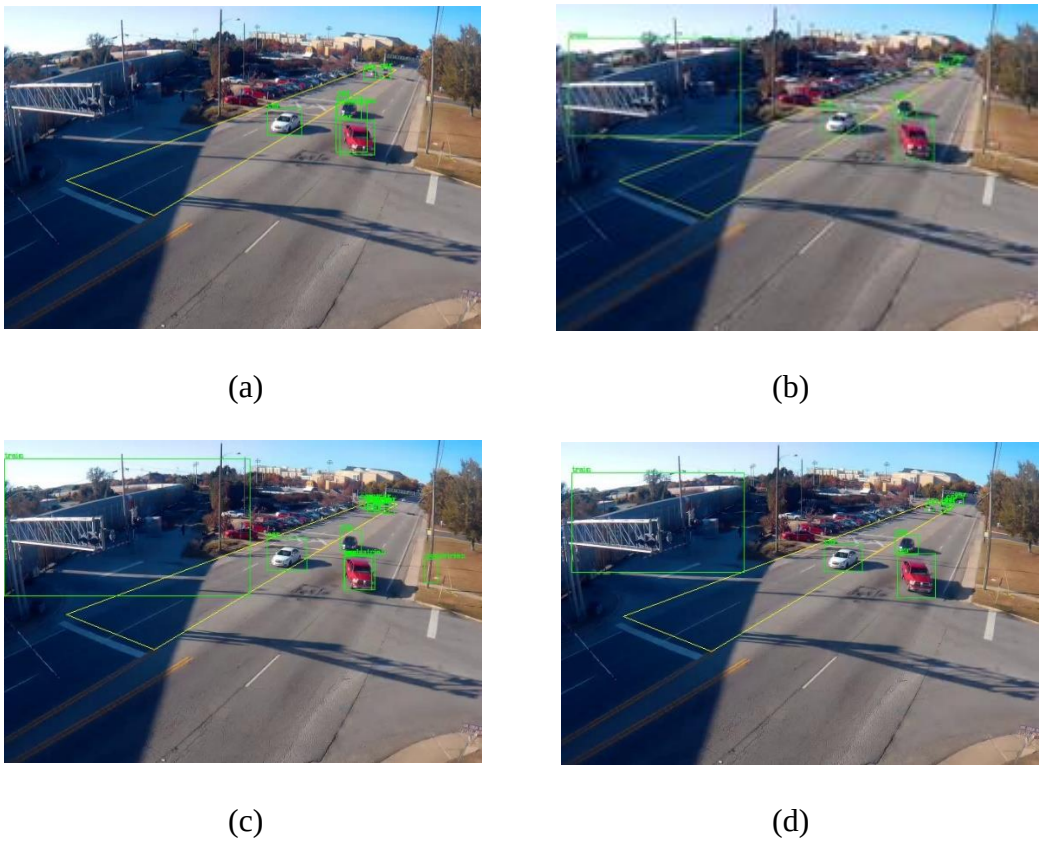
Specifically, regarding vehicle detection with the baseline, we find that the TA module contributes a little to improving detection accuracy but the FreeAnchor head significantly improves vehicle detection accuracy. This point also can be proved by the fact that more vehicles are detected in Figure 3.9 (c) than Figure 3.9 (b). Compared to the baseline, the models include TA module and the FreeAnchor head improve vehicle detection accuracy 1.0% and 10.0%, respectively. As for pedestrian detection, the TA module leads to a decrease on the pedestrian detection accuracy and the FreeAnchor head only brings 1.0% increment. Note that with both TA module and FreeAnchor head, our model is 3.1% higher than the baseline on the pedestrian detection accuracy.

The TA module presents significant impact on modeling the long object such as the train instance which is important for traffic management at the crossing. In Figure 3.9 (b) and (d), it can be found that the location and size of the detected train are more accurate than in Figure 3.9 (a) and (c). The improvement is attributed to the TA that can effectively model long-range correlations for better understanding the global context. Compared to the baseline, the models include TA module and the FreeAnchor head contributes 4.0% and 1.2% to the improvement of train detection accuracy, respectively. Through comparing the inference speed, we find that the inclusion of the TA module brings 11.2% inference speed decreases compared to the baseline. Thus, the inference

speed performance still needs to be improved and more work will be focused on the optimization of inference speed in the future.

Table 3.1. Ablation test results with different network component under training image size of 1333×800.

Model	mAP (%)	FPS	Vehicle	Pedestrian	Train
RetinaNet	77.8	19.8	74.1	73.9	85.4
RetinaNet + TA	78.2	17.8	74.9	70.6	89.0
RetinaNet + FreeAnchor head	81.1	20.0	82.3	74.7	86.4
DTDNet	83.2	13.7	83.1	76.3	90.1



(a) (b)

(c) (d)

Figure 3.9. Ablation test results under the scene of a train passing the grade crossing. (a) RetinaNet; (b) RetinaNet + TA; (c) RetinaNet + FreeAnchor head; (d) DTDNet.

3.3.3 Evaluation metrics and experimental results

Precision-recall (PR) curve is a popular tool to visualize the relationship between precision and recall (Guo, Qian, Wu, Leng, & Yu, 2021). Typically, precision and recall are opposite to each other. In other words, high precision is associated with low recall, and vice versa. In practice, the averaged precision (AP) calculated by integrating the area under each curve is adopted to evaluate the model accuracy. Equations (3-17) to (3-20) demonstrate the computation of precision, recall, and AP, respectively. The PR curves of above eight models including DTDNet (the proposed network) are plotted in Figure 3.10.

$$precision = \frac{TP}{TP + FP} \tag{3-17}$$

$$recall = \frac{TP}{TP + FN} \tag{3-18}$$

$$AP = \int_0^1 p(r)dr \tag{3-19}$$

$$mAP = \frac{1}{N}\sum AP_i \tag{3-20}$$

where TP is true positive, FP is false positive, and FN is false negative. N is the number of object classes.

By comparing the results in Figure 3.10, we can find that SOTA models fail to reach the expected performance as previous researchers claimed when they are applied to the scene of congested traffic at the grade crossing in this study. As for the AP results of vehicle detection, our model reaches the highest value of 0.831 while the second highest value is 0.828 coming from FSAF. SSD has the lowest AP value (0.405), indicating unsatisfactory performance for this object class. Concerning pedestrian detection, our proposed model reaches the AP value of 0.763. Compared to RetinaNet, FreeAnchor and FSAF, our model is 3.2%, 2.1%, and 5.0% higher, respectively, indicating promising

results on the small object detection. It needs to mention that SSD has the worst performance of pedestrian detection with an AP value of 0.399 and our model is 91.2% higher. With respect to the AP values of train detection, our model achieves the maximum value, 0.901, which is 1.7% higher compared to the lowest AP value from SSD. It is easy to find that the large object (train) has better experimental results compared to the other two smaller object classes by each model. No doubt the detection of small and dense objects is a more challenging task.

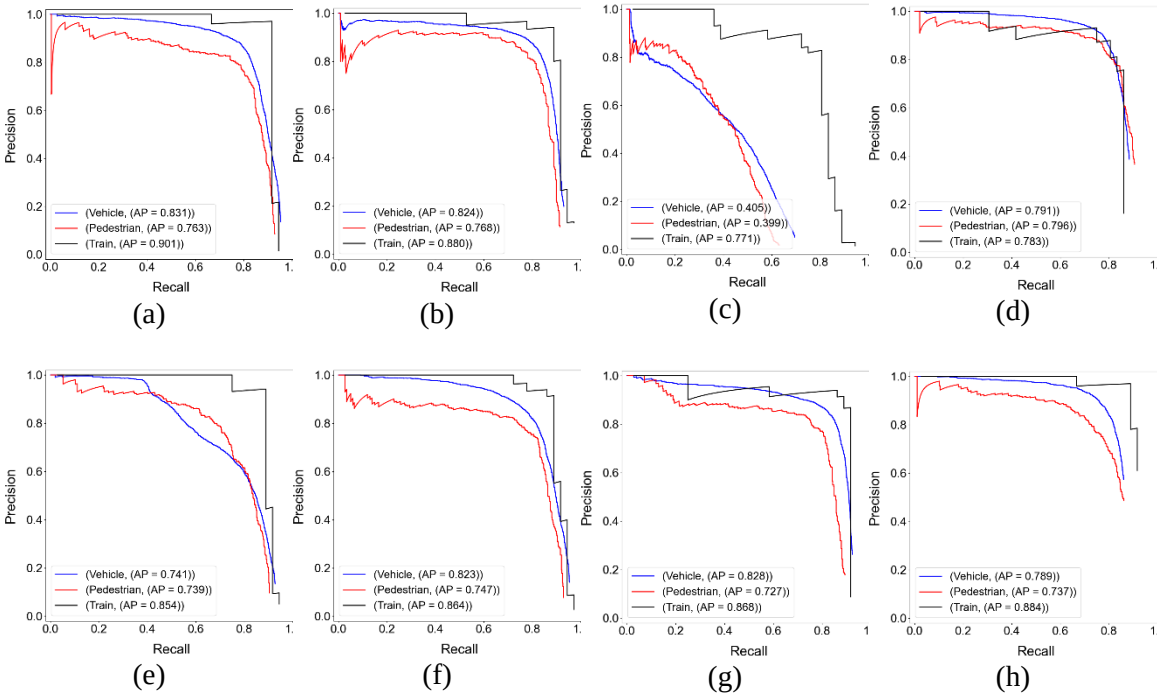


Figure 3.10. Precision-Recall curve of different models. (a) DTDNet; (b) ATSS; (c) SSD; (d) Faster R-CNN; (e) RetinaNet; (f) FreeAnchor; (g) FSAF; (h) Cascade R-CNN.

Except for AP, recall is another useful parameter to evaluate training results. As defined in Equation (3-18), recall reflects the ratio of the detected positive case over the ground truth case. The recall values are shown in Figure 3.11. The recall values of different classes from the proposed DTDNet are marked with red numbers and arrows for

easy comparison with other SOTA models. Regarding the detection of vehicles, the proposed model reaches the maximum recall value, 0.953. The recall value of the proposed DTDNet is 37.3% higher compared to the lowest recall value of SSD, and is 2.4%, 3.1%, 0%, 2.8%, 1.1%, and 1.1% higher than ATSS, FSAF, FreeAnchor, RetinaNet, Cascade R-CNN, and Faster R-CNN, respectively.

Regarding the training results of the pedestrian detection, the recall value of the proposed DTDNet is 0.929, which is 47.0% higher than SSD. It also outperforms ATSS and FSAF by 1.6% and 4.0%, respectively. The promising recall results on both vehicle and pedestrian detection suggest high accuracy potentially in the field application.

As shown in Figure 3.11, most of the detectors perform well on the large object detection, which is possibly because more features are available in the region of interest (ROI) for training. Our model reaches the second highest recall value (0.944) on the vehicle class and is 2.9% lower than ATSS and FreeAchor. Interestingly, SSD has the worst detection performance on both the vehicle and the pedestrian classes, while it still has good performance on the train detection. Overall, our model shows the best detection performance in terms of both precision and recall values on the railroad grade crossing image dataset.

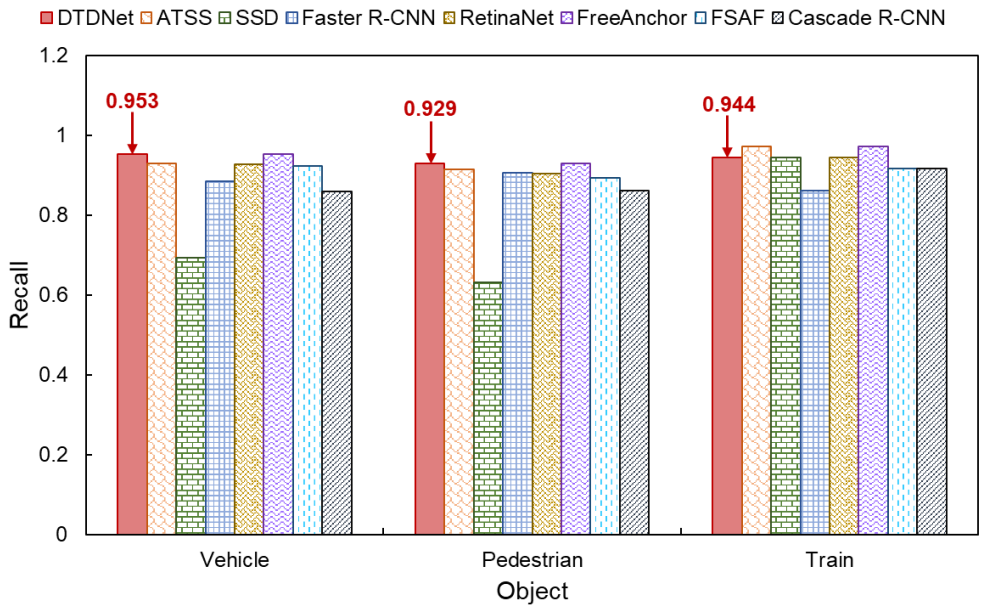


Figure 3.11. Recall values of different models.

3.3.4 Vehicle counting performance

Different from the evaluation of object detection, three parameters, mean absolute error (MAE), root mean squared error (RMSE), and mean relative error (MRE), are chosen to evaluate the vehicle counting performance because they are common evaluation metrics to measure the error between the prediction and the ground truth (Hsieh, Lin, & Hsu, 2017; Li, Li, Wu, Chen, & Ngan, 2019; W. Li et al., 2020; Li, Zhang, & Chen, 2018). RMSE could have a relatively higher weight of large errors because the errors are squared first and then averaged. Meanwhile, MRE can provide a measure of the relative size of the error in the field application. The definitions of MAE, RMSE, and MRE are shown in Equation (3-21), (3-22), and (3-23).

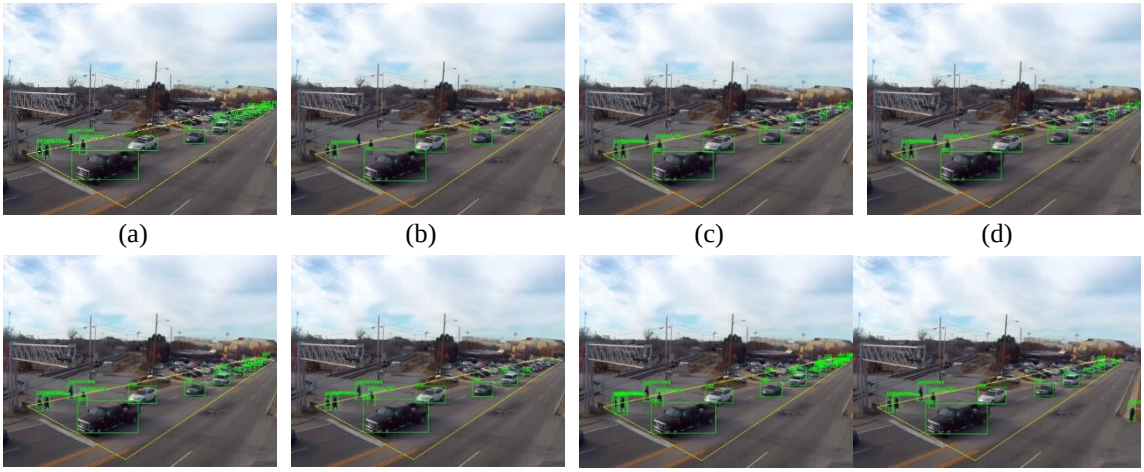
$$\mathrm{MAE} = \frac{1}{n}\sum_{j=1}^{n}\left|y_j - \hat{y}_j\right| \tag{3-21}$$

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{j=1}^{n}(y_j - \hat{y}_j)^2} \tag{3-22}$$

$$\text{MRE} = \frac{1}{n}\sum_{j=1}^{n}\frac{\left|y_j - \hat{y}_j\right|}{y_j} \tag{3-23}$$

where y_j corresponds to the ground truth number of vehicles in the ROI, $\hat{y}_j$ is the number of detected vehicles in the ROI. *n* is the total number of frames in the test.

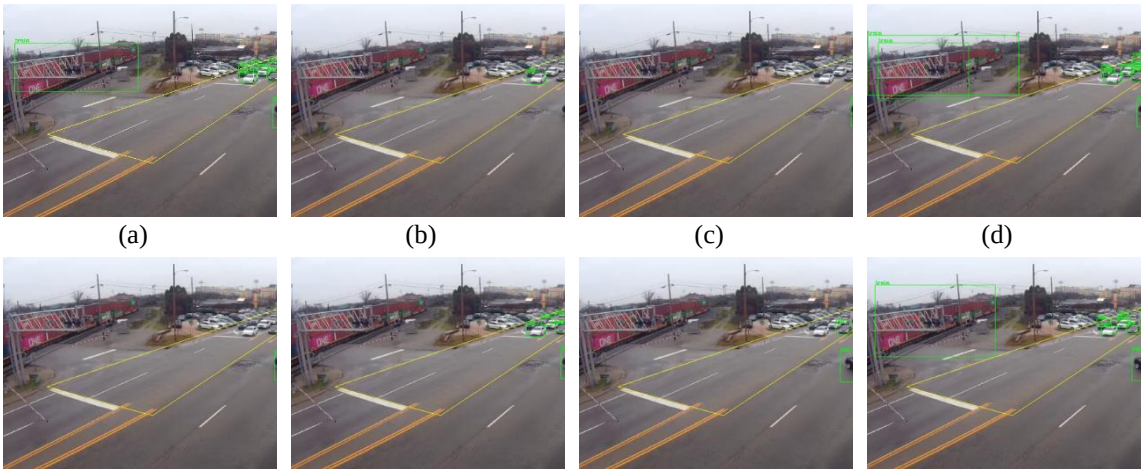
In this study, the ROI for vehicle detection and counting is specified by the quadrilateral area demarcated with the yellow lines, as shown in Figure 3.12, Figure 3.13, Figure 3.14 and Figure 3.15, at the monitored grade crossing. It is worth mentioning that the ROI needs to be adjusted each time when the camera position and/or orientation is changed.



(a) (b) (c) (d)

(e) (f) (g) (h)

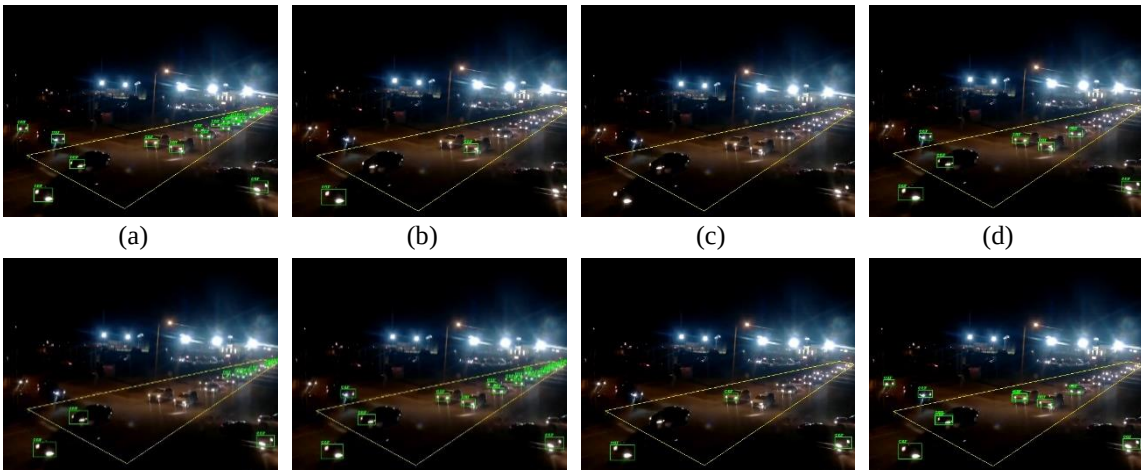
Figure 3.12. Field test results of during the daytime (without a train). (a) DTDNet; (b) ATSS; (c) SSD; (d) Faster R-CNN; (e) RetinaNet; (f) FreeAnchor; (g) FSAF; (h) Cascade R-CNN.



(a) (b) (c) (d)

(e) (f) (g) (h)

Figure 3.13. Field test results of during the daytime (with a train). (a) DTDNet; (b) ATSS; (c) SSD; (d) Faster R-CNN; (e) RetinaNet; (f) FreeAnchor; (g) FSAF; (h) Cascade R-CNN.



(a) (b) (c) (d)

(e) (f) (g) (h)

Figure 3.14. Field test results during the night. (a) DTDNet; (b) ATSS; (c) SSD; (d) Faster R-CNN; (e) RetinaNet; (f) FreeAnchor; (g) FSAF; (h) Cascade R-CNN.

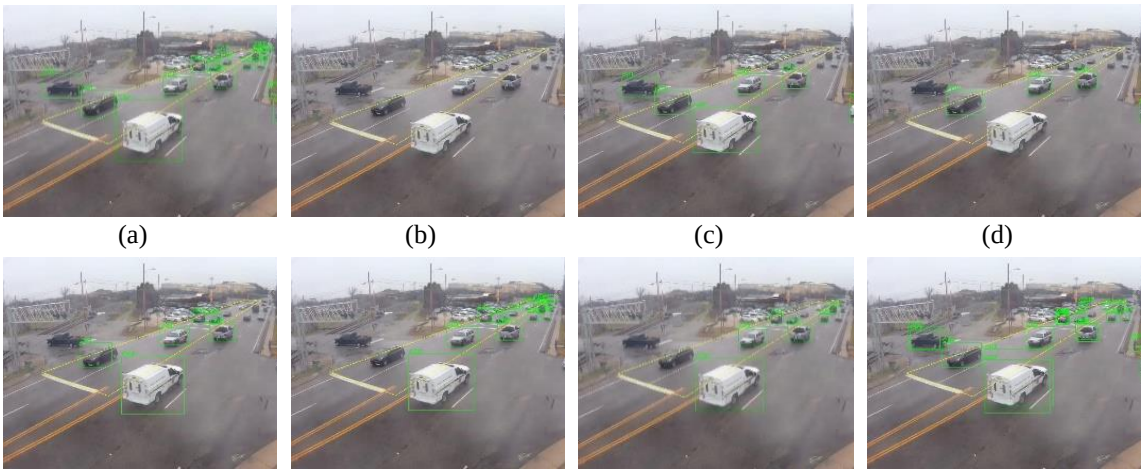


Figure 3.15. Field test results under the haze condition. (a) DTDNet; (b) ATSS; (c) SSD; (d) Faster R-CNN; (e) RetinaNet; (f) FreeAnchor; (g) FSAF; (h) Cascade R-CNN.

It needs to be mentioned that the battery life of the surveillance camera is 50 hours only, and each installation may introduce slight changes of the camera angle. In addition, all object classes can be detected in the whole image, but only the detected vehicles in the ROI were counted in this study. Figure 3.12, Figure 3.13, Figure 3.14 and Figure 3.15 present the detection performance during the daytime, nighttime and the light haze condition, respectively. It is clear to see that our proposed method can classify different objects well no matter it is a “bulky” train, a “tiny” pedestrian, or the “indistinguishable” vehicle at the end of the queue. Compared with other models, the proposed DTDNet can efficiently focus on the objects and accurately distinguish positive samples by using the attention module and the free anchor matching design, respectively. Results shown in Figure 3.12 to Figure 3.14 and Table 3.2 to Table 3.4 confirm the proposed model outperforms other SOTA models in all scenes, daytime or nighttime, busy or normal traffic, indicating a robust and salient performance for field applications.

From Figure 3.15, it is easy to find that under a light haze condition, our proposed model can detect traffic instances in a high accuracy manner. Clearly, the introduction of haze degrades the performance of other SOTA models and the number of detected vehicles at the end of the queue is decreased. Table 3.5 confirms our proposed model outperforms other models on this specific light haze condition. Even the number of detections of Free Anchor and Cascade RCNN is close to the ground truth. There are a lot of false positives and missing out of ROI. Meanwhile, it also indicates a direction for further improvement. The comparison under other bad weather conditions will be presented once the data is available.

Regarding the detection-based counting performance, we tested DTDNet on six randomly selected video clips with heavy congestion conditions during the daytime. The reason for not considering congestions during the night is twofold: 1) there is a low probability of congestion during the night and 2) the density level of congestion is not as severe as daytime. 50 frames in each video clip are randomly selected, and the ground true of vehicle counts in each clip is manually counted. Prediction results by DTDNet and the ground truth are plotted in Figure 3.16. The prediction curves are generally in good agreement with the ground truth, further confirming excellent counting performance of our proposed model. To further quantify the results in Figure 3.16 and better present the detection and counting performance, the aforementioned three indicators including MAE, RMSE, and MRE are calculated and summarized in Table 3.6 for each video clip.

Table 3.2. Example results during the daytime based on Figure 3.12.

	Vehicle	Pedestrian	Train
Ground Truth	20	4	0
DTDNet	18	4	0
ATSS	5	1	0
SSD	6	3	0
Faster R-CNN	12	4	0
RetinaNet	5	4	0
FreeAnchor	23	5	0
FSAF	6	2	0
Cascade R-CNN	9	4	0

Table 3.3. Example results during the daytime based on Figure 3.13.

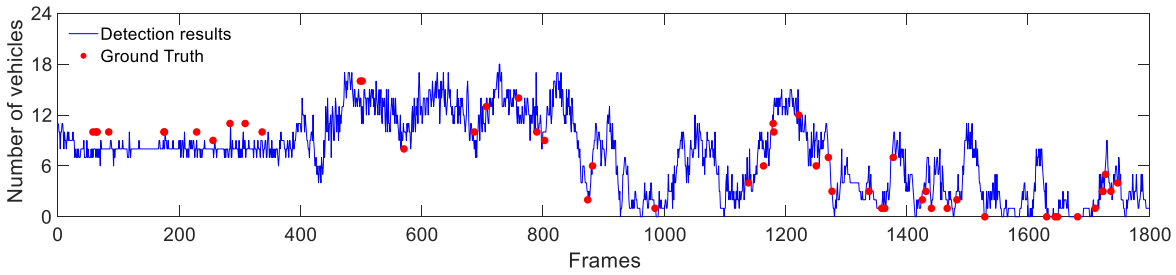
	Vehicle	Pedestrian	Train
Ground Truth	12	0	1
DTDNet	10	0	1
ATSS	1	0	0
SSD	0	0	0
Faster R-CNN	8	0	2
RetinaNet	2	0	0
FreeAnchor	8	0	0
FSAF	2	0	0
Cascade R-CNN	6	0	1

Table 3.4. Example results during the night based on Figure 3.14.

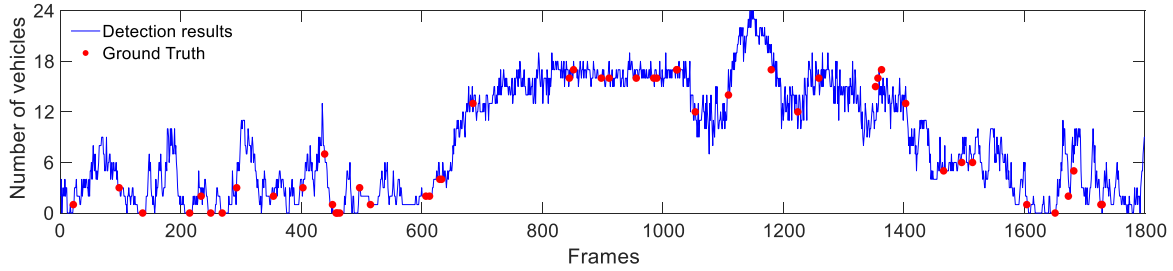
	Vehicle	Pedestrian	Train
Ground Truth	15	0	0
DTDNet	16	0	0
ATSS	1	0	0
SSD	0	0	0
Faster R-CNN	4	0	0
RetinaNet	7	0	0
FreeAnchor	17	0	0
FSAF	1	0	0
Cascade R-CNN	6	0	0

Table 3.5. Example results during the night based on Figure 3.15.

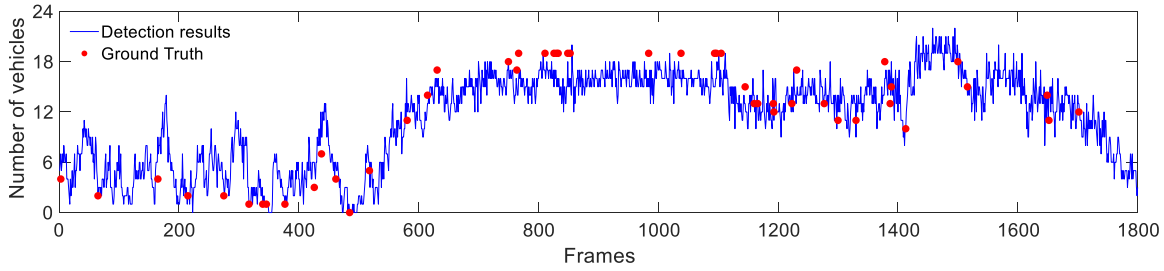
	Vehicle	Pedestrian	Train
Ground Truth	9	0	0
DTDNet	6	0	0
ATSS	0	0	0
SSD	2	0	0
Faster R-CNN	2	0	0
RetinaNet	4	0	0
FreeAnchor	6	0	0
FSAF	3	0	0
Cascade R-CNN	7	0	0


Detection results
Ground Truth
Number of vehicles
Frames
0
6
12
18
24
0
200
400
600
800
1000
1200
1400
1600
1800

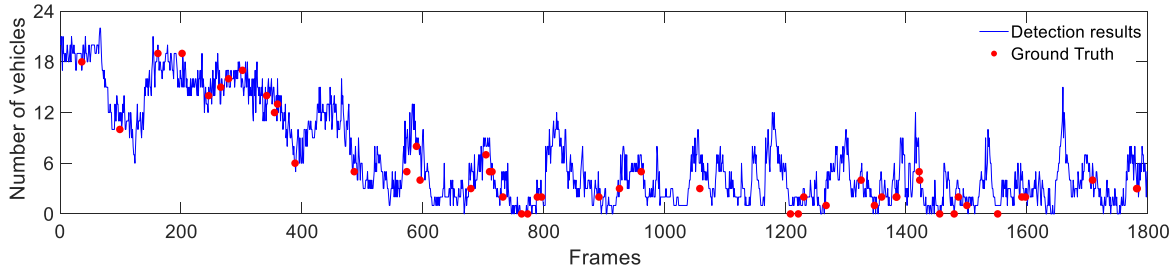
(a)


Detection results
Ground Truth
Number of vehicles
Frames
0
6
12
18
24
0
200
400
600
800
1000
1200
1400
1600
1800

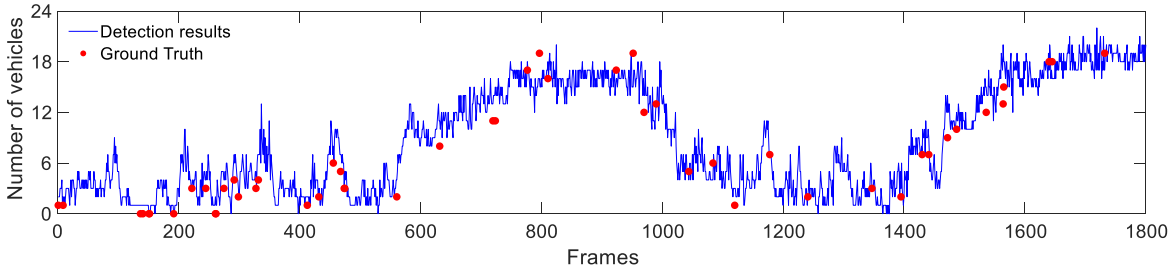
(b)


Detection results
Ground Truth
Number of vehicles
Frames
0
6
12
18
24
0
200
400
600
800
1000
1200
1400
1600
1800

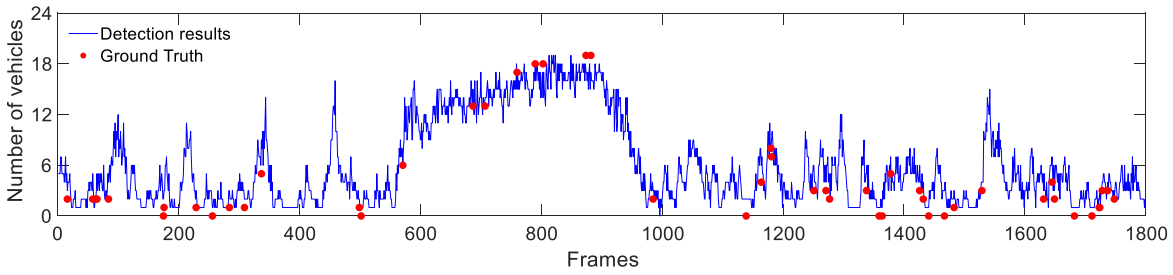
(c)


Detection results
Ground Truth
Number of vehicles
Frames
0
6
12
18
24
0
200
400
600
800
1000
1200
1400
1600
1800

(d)


Detection results
Ground Truth
Number of vehicles
Frames
0
6
12
18
24
0
200
400
600
800
1000
1200
1400
1600
1800

(e)



(f)

Figure 3.16. Comparison between prediction and ground truth on congested scene at the grade crossing.

The quantitative results in Table 3.6 convincingly show that our proposed model can perform well in congested conditions during the daytime. Regarding MAE results, clip No.2 has the minimum value, which is 0.67, and meanwhile clip No. 3 reaches the maximum MAE (2.20). As for RMSE results, clip No.2 obtains the minimum value which is 1.32, and clip No.6 obtains the maximum RMSE (1.89). In addition to the low values of MAE and RMSE, all MRE values are smaller than 0.3. To put it into a nutshell, the proposed model can accurately evaluate congestion conditions at the grade crossing, and the counting results have low errors compared to the ground truth.

3.4 Summary

This study presents a new DTDNet for congested traffic detection and experimental results of its detection-based counting in a specific grade crossing. The major contributions include the shortfall of detecting traffic instances at grade crossings of SOTA models has been filled with our proposed DTDNet. The feasibility of using the transformer attention (TA) module to characterize long-distance relationships and the MLE procedure to detect heavily packed objects has been discussed and tested. The detection performance under different environmental conditions including daytime,

nighttime, and haze weather has been evaluated and compared on our proposed model and other SOTA models. Detection-based vehicle counting has been tested. Experimental results based on a railroad grade crossing image database have verified our model's robustness and superiority.

Specifically, the backbone uses ResNet-50 which incorporates a TA module for better focusing on long-range relationships and bridging different forms of representation. Four attention factors are considered in the TA module. The learning-to-match-based detection head is integrated into our proposed model. It uses the MLE procedure rather than the conventional hand-crafted IoU criterion to match the object and its corresponding anchor in a flexible manner. A total of eight models, including seven SOTA models, were trained and evaluated. The indicators of recall and precision were used for assessing training and validation performance. The field test results of detection-based counting were evaluated by MAE, RMSE, and MRE.

Training and evaluation results indicate that our proposed model outperforms other models on both recall and precision rates. Our model achieves the maximum mAP value (0.832) of detecting vehicle, train, and pedestrian. The assessment of field congestion using detection-based counting shows that the proposed model can achieve promising results. With six randomly selected video clips at the grade crossing, the values of MAE, RMSE, and MRE are smaller than 2.200, 1.890, and 0.280, respectively. Overall, our proposed model would be a possible solution for congested traffic assessment at the grade crossing with a low-cost camera having a limited resolution.

CHAPTER 4 AUTOMATIC RAILROAD TRACK COMPONENTS INSPECTION USING REAL-TIME INSTANCE SEGMENTATION[4]

[4] Guo, Feng, Yu Qian, Yunpeng Wu, Zhen Leng, and Huayang Yu. Automatic railroad track components inspection using real-time instance segmentation. *Computer-Aided Civil and Infrastructure Engineering* 36, no. 3 (2021): 362-377.

In the United States, to ensure the railroad safety and keep its efficient operation, regular track inspections on track component defects are required by the Federal Railroad Administration (FRA). Various types of inspection equipment have been applied, such as ground penetrating radar, laser, and LiDAR, but they are usually very expensive and require extensive training and rich experience to operate. To date, track inspections still heavily rely on manual inspections which are low-efficient, subjective, and not as accurate as desired, especially for missing and broken track components, such as spikes, clips, and tie plates.

To address this issue, a real-time pixel-level rail components detection framework to inspect track timely and accurately is proposed in this chapter. The first public rail components image database, including rails, spikes, and clips, is built and released online. A real-time pixel-level detection framework with improved real-time instance segmentation models is developed. The improved models leverage fast object detection and highly accurate instance segmentation. Backbones with more granular levels and receptive fields are implemented in the proposed models. Compared with the original YOLACT and Mask R-CNN models, the proposed models are able to: 1) achieve 59.9 bbox mAP, and 63.6 mask mAP with the customized dataset, which are higher than the other models, and 2) achieve a real-time speed which is over 30 FPS processing a high-resolution video (1080×1092) with a single GPU. The fast processing speed can quickly turn inspection videos into useful information to assist track maintenance. The railroad track components image dataset can be accessed at https://github.com/jonguo111/Rail_components_image_data

4.1 Introduction

With the significant progress in neural network and computer vision, infrastructure damage detection methods, based on machine learning and computer vision, have been successfully applied in civil engineering (Adeli, 2001). For instance, in bridge damage detection, researchers have conducted bridge health inspections by using the Bayesian optimized deep learning model (Liang, 2019), concrete bridge surface damage detection by using the improved YOLOv3 (C. Zhang et al., 2020), and crack evaluation of a high-rise bridge by using a modified SegNet (Jang et al., 2020). For pavement assessment and crack detection, CrackNet and CrackNet-V for pixel-level cracking detection on 3D asphalt images were developed (Fei et al., 2019; A. Zhang et al., 2017). Jeong et al. (2020) assessed the pavement roughness by using an optimized CNN.

For concrete structure damage evaluation, there were studies on the reinforced concrete building damage detection using ResNet-50 and ResNet-50-based YOLOv2 (Pan & Yang, 2020), pixel-level multiple damage detection of concrete structure by using a fine-tuned DesNet-121 (Li, Zhao, & Zhou, 2019), and concrete crack detection by using context-aware semantic segmentation (Xinxiang Zhang, Dinesh Rajan, et al., 2019). Moreover, health condition monitoring of civil infrastructure has widely been using CNNs, such as infrastructure condition assessment using DCNNs (Wu et al., 2019) and the estimation of wind-induced responses using a CNN model (Oh, Glisic, Kim, & Park, 2019).

However, few studies implement the cutting-edge CNN models on railroad track inspection and detection. Gibert et al. (2016) attempted to use the DCNN model, which was developed for semantic image segmentation, in railway ties and fasteners inspection. The target objects needed to be classified on multiple levels and cannot perform a real-time and end-to-end test. Wu et al. (2018) built a novel visual inspection model for rail surface defects using UAV images and gray stretch maximum entropy. Due to limited performance under visibility and environmental variations, the model has few field applications. Later, Wu et al. (2020)proposed a deep learning-based method to improve the inspection on track fasteners using UAV images. However, it only focused on object detection but not segmentation, leaving it hard to characterize the damage shape of a certain track component.

Up to now, track component detection is still a very challenging task due to the complex environmental condition, small or tiny objects, and limited training data. Besides, the railroad track could appear to be very similar, but there would also be some variations. For example, the spikes and clips may be quite different from each other depending on the types. Also, the appearance of the same components would change based on the surrounding environment, considering the track would go through different remote/rural areas.

This chapter proposes a computer vision-based pixel-level track components detection system by using the improved one-stage instance segmentation model and prior knowledge, aiming to inspect the rail components in a rapid, accurate, and convenient fashion. The proposed network extracts the input features from the improved backbone, predicts objects in different scales utilizing feature pyramid network, and generates high-quality masks by assembling prototype generation and mask coefficient. As Figure 4.1 shows, three major tasks are conducted in this study: 1) data preparation, 2) training & validation, and 3) prediction & comparison with other state-of-the-art models. In this study, the contributions and novelties include: 1) The first public railroad components dataset, which includes a total of 1000 images, is built and released online for free access. It may prompt the implementation of cutting-edge deep learning models in the railroad application. 2) Real-time instance segmentation models with fast speed and high accuracy are firstly improved and utilized in railroad research. Testing results show the improved models outperform the original models. In the future, when it is implemented on a mobile computing board which has enough computational power, the current "walking inspection" in the railroad could be replaced, and the future inspection work can be more efficient and accurate. 3) The effects of different illumination conditions on predictions are discussed. The testing results verified the illumination condition would influence the performance of the models, and the improved models work better than the original models under low-visibility conditions.

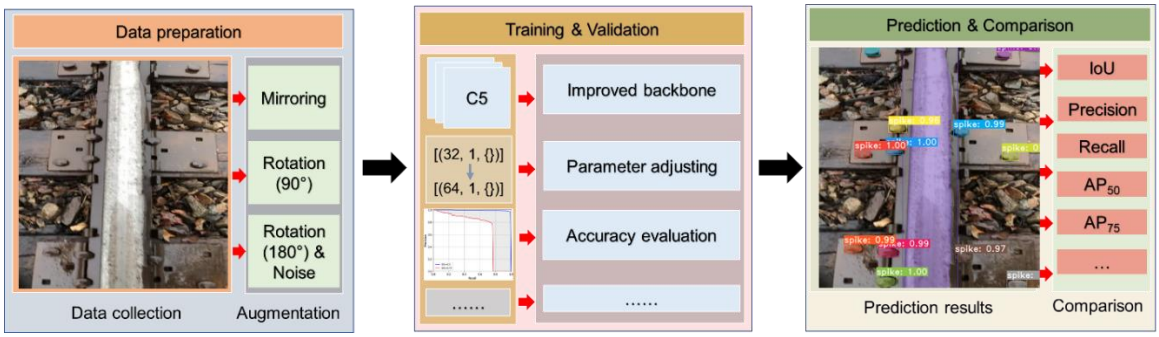


Figure 4.1. Content of this chapter.

4.2 Methodology

4.2.1 Proposed neural network architecture

To accurately identify multiple railroad track components, YOLACT-Res2Net-50 and YOLACT-Res2Net-101, which adapt a new backbone architecture compared to the original models, are proposed and evaluated in this study. Figure 4.2 presents the main structure of the proposed models. Specifically, the main structure includes backbone (feature extractor), feature pyramid network (FPN), prediction head (generating anchors), and Protonet (predicting *k* prototype masks). In general, instance segmentation is more difficult than object detection since it heavily relies on feature localization to generate masks, resulting in low speed and impractical in field applications. Nevertheless, the YOLACT type model separates the instance segmentation into two parallel tasks. One is responsible for generating prototype masks using the Protonet (a fully convolutional network) over the entire image, and the other one focuses on predicting anchors and mask coefficients by using prediction head. These two tasks are assembled by a linear combination, and the outputs are generated with a threshold. In this way, the model improves inference speed and mask quality.

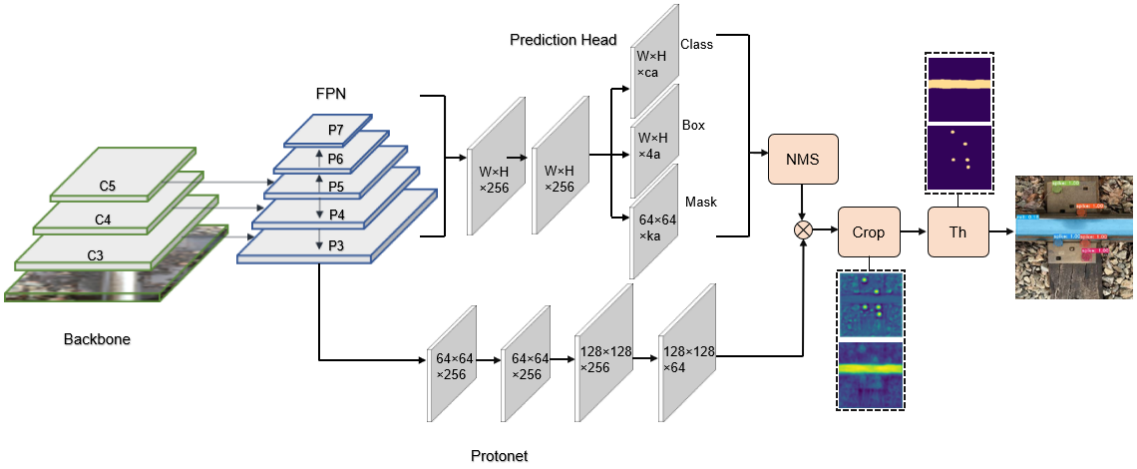


Figure 4.2. The main structure of the proposed models.

4.2.2 Backbone structure

In object detection, the backbone acts as the main feature extractor, which takes images or videos as input and yields corresponding feature maps (Jiao et al., 2019). According to the specific needs of detection accuracy and efficiency, different backbones can be developed for a model after a modification or tuning. For high accuracy, a deep and densely connected backbone, such as the ResNet and DenseNet, can be employed in the model. Considering the speed and efficiency, lightweight backbones, such as the MobileNet and EfficientNet, would be preferred. In this study, to improve the detection performance, a new backbone, Res2Net, with a stronger multi-scale representation capability is implemented into the proposed models, YOLACT-Res2Net-50 and YOLACT-Res2Net-101. More details are presented in the following sections.

4.2.2.1 ResNet-50 & ResNet-101

ResNet-50 and ResNet-101 backbone (He et al., 2016) are adopted in the original YOLACT models. As the name indicates, ResNet-50 and ResNet-101 include 50 layers and 101 layers, respectively. To reduce the inference computations, the bottleneck structure is introduced in the ResNet. Figure 4.3 shows the bottleneck design for ResNet-50 and ResNet-101. As shown in Figure 4.3, with the bottleneck design, the first 1×1 convolution reduces a 256-dimension channel to a 64-dimension channel, and it is recovered by a 1×1 convolution at the end.

Figure 4.4 shows the main structure of the ResNet-50. It consists of five stages, which are Conv1, Layer1, Layer2, Layer3, and Layer4, respectively, corresponding to C1 to C5 shown in Figure 4.2. Due to the limitation on the space, C1 and C2 are not plotted in Figure 4.2. From Layer1 to Layer4, each block contains three convolutional layers, which represent the bottleneck module. Specifically, there are 3, 4, 6, and 3 stacked blocks in ResNet-50. Similarly, in ResNet-101, there are 3, 4, 23, and 3 stacked blocks. Furthermore, after Conv1, Layer1, Layer2, Lay3, and Layer4, the input image size becomes 1/2, 1/4, 1/8, 1/16, and 1/32 of the original image size, respectively.

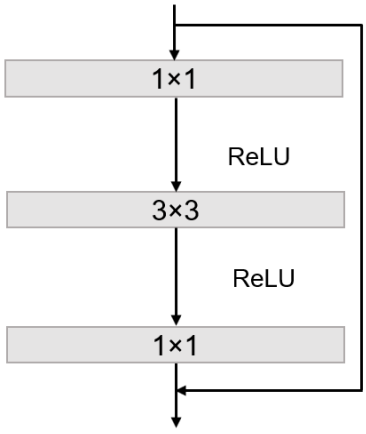


Figure 4.3. Structure of bottleneck design.

4.2.2.2 Res2Net-50 & Re2sNet-101

Res2Net (Gao et al. (2019) is a new backbone architecture which can improve the multi-scale representation capability at a granular level. Figure 4.5 shows the architecture of the Res2Net bottleneck which plays an important role in the new backbone. In this bottleneck structure, the original 3×3 filter of *n* channels shown in Figure 4.3 is replaced with a set of smaller filter groups. Each group has *w* channels. Note, $n=w\times s$, where *s* represents the scale. As shown in Figure 4.5, following the 1×1 convolution, the feature maps are evenly split into *s* subsets. x_i is one of the subsets which has 1/s number of channels and the same spatial size with inputs. For each feature subset x_i ($i \geq 2$), there is a 3×3 convolution corresponding to it, namely as K_i (). While for x_1 and $y_1 = x_1$, there is no convolution. Each output feature map, y_i, is the output of K_i (). The calculations are summarized in Equation (4-1). During the following model training and evaluation in this study, *w* is assigned to 26 and *s* is assigned to 4.

$$y_i = \begin{cases} x_i & i = 1; \\ K_i(x_i) & i = 2; \\ K_i(x_i + y_{i-1}) & 2 < i \leq s. \end{cases} \tag{4-1}$$

where y_i is the output feature map, x_i is the input feature map, K_i is the convolution corresponding to x_i.

To better show the improved network architecture, Table 4.1 presents the detailed parameters of the proposed YOLACT-Res2Net-50 backbone. As shown in Figure 4.4, there are five stages: Conv1, Layer1, Layer2, Layer3, and Layer4. The main difference is the bottleneck structures shown in Figure 4.3 and Figure 4.5. It also can be referred in the filter size shown in Table 4.1. The original filters are changed from [1×1, 3×3, 1×1] to [1×1, 3×3, 3×3, 3×3, 1×1]. Meanwhile, from x_2 to x_4, there are convolutional processes

with each kernel. This way, as the literature (Gao et al., 2019) mentioned, the range of receptive fields for each network layer will increase. Therefore, the model will have better detection performance. Besides, it is worth noting that the introduced feature sets cause changes in the output channels.

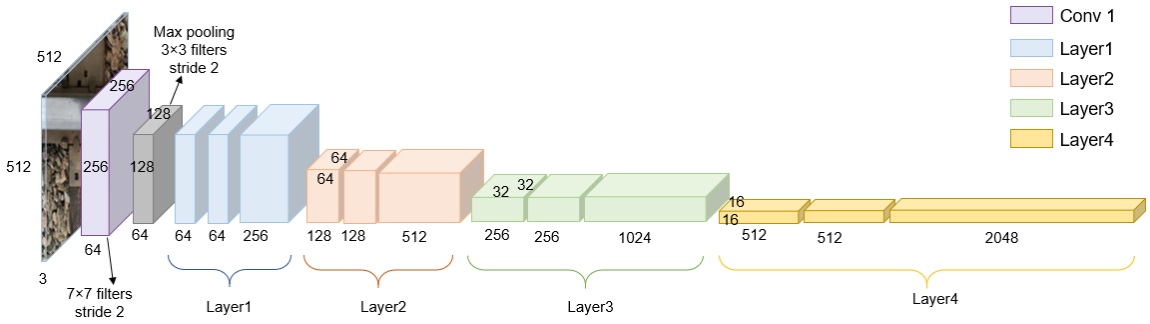


Figure 4.4. Main structure of ResNet-50.

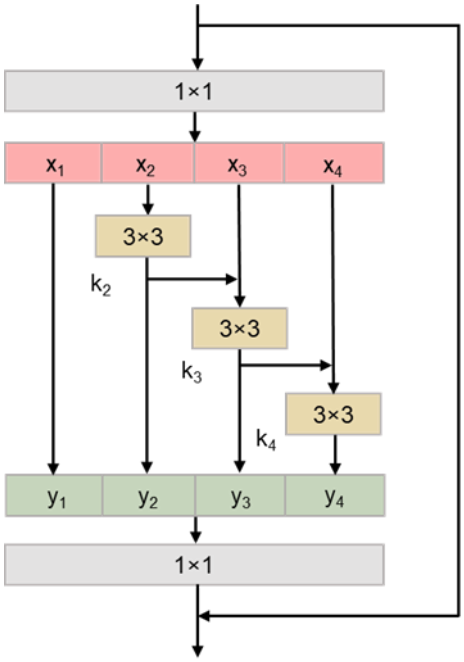


Figure 4.5. Structure of. Res2Net bottleneck (scale=4).

Table 4.1. The detailed specifications of backbone of proposed Res2Net-50.

Layer	Type	Filter size	Stride	Output channels	Output size
Input image					512
Conv1		7×7	2	64	256
	Max pooling	3×3	2	64	128
Layer1	bottleneck	$\begin{bmatrix} 1\times1,\ 104 \\ 3\times3,\ 26 \\ 3\times3,\ 26 \\ 3\times3,\ 26 \\ 1\times1,\ 256 \end{bmatrix}\times3$		256	128
Layer2	bottleneck	$\begin{bmatrix} 1\times1,\ 208 \\ 3\times3,\ 52 \\ 3\times3,\ 52 \\ 3\times3,\ 52 \\ 1\times1,\ 512 \end{bmatrix}\times4$		512	64
Layer3	bottleneck	$\begin{bmatrix} 1\times1,\ 416 \\ 3\times3,\ 104 \\ 3\times3,\ 104 \\ 3\times3,\ 104 \\ 1\times1,\ 1024 \end{bmatrix}\times6$		1024	32
Layer4	bottleneck	$\begin{bmatrix} 1\times1,\ 832 \\ 3\times3,\ 208 \\ 3\times3,\ 208 \\ 3\times3,\ 208 \\ 1\times1,\ 2048 \end{bmatrix}\times3$		2048	16

4.2.3 FPN structure

To detect objects on multiple scales, Feature Pyramid Network (FPN) (Lin, Dollár, et al., 2017) has widely been using in many object detection and segmentation models. Typically, the composition of an FPN includes a bottom-up pathway, a top-down pathway, and lateral connections. The bottom-up pathway is the feed-forward computation for the backbone to extract features in the inputs. The assembly of convolution layers with the same output feature size is denoted as the stage in the FPN. Specifically, the backbone, as shown in Figure 4.1, {$C3$, $C4$, $C5$} is the output of the last

residual blocks in the stage of Layer2, Layer3, and Layer4, respectively. When layers go up, the spatial resolution decreases. In terms of the top-down pathway, it constructs the high-resolution layers from higher layers in the pyramid which are semantically strong, but not precise. Hence, the later connections are then used to merge the features from the bottom-up pathway and top-down pathway for a better prediction on the object locations. The original set of feature output in the FPN is {$P3$, $P4$, $P5$}, corresponding to {$C3$, $C4$, $C5$}. In the YOLACT type models, to increase the detection performance on the small objects, $P5$ is upsampled to $P6$ and $P7$ with one-fourth dimensions; meanwhile, $P2$ is omitted.

4.2.4 Prototype generation

To improve the operation speed, instance segmentation is achieved by two parallel tasks in the original and improved models. One of the parallel tasks, generating prototype masks, is completed by Protonet. It is worth noting Protonet is a fully connected network (FCN), which is attached to the $P3$ layer in the FPN. The architecture of Protonet can be seen in Figure 4.2. In this branch, k Protonet masks without loss computations are proposed for the entire image. To improve the instance segmentation performance, we increase the k from 32 to 64 in the proposed models. A nonlinear activation function, ReLU, is used to keep the outputs from Protonet unbounded and generate more interpretable prototypes.

It also needs to mention that the number of prototype masks are independent of the number of categories; thus, it can lead to a distributed representation for the generated prototypes. Example prototype images generated by the proposed YOLACT-Res2Net-50 are shown in Figure 4.6. The high-resolution prototypes are beneficial for mask quality and detection performance on small objects.

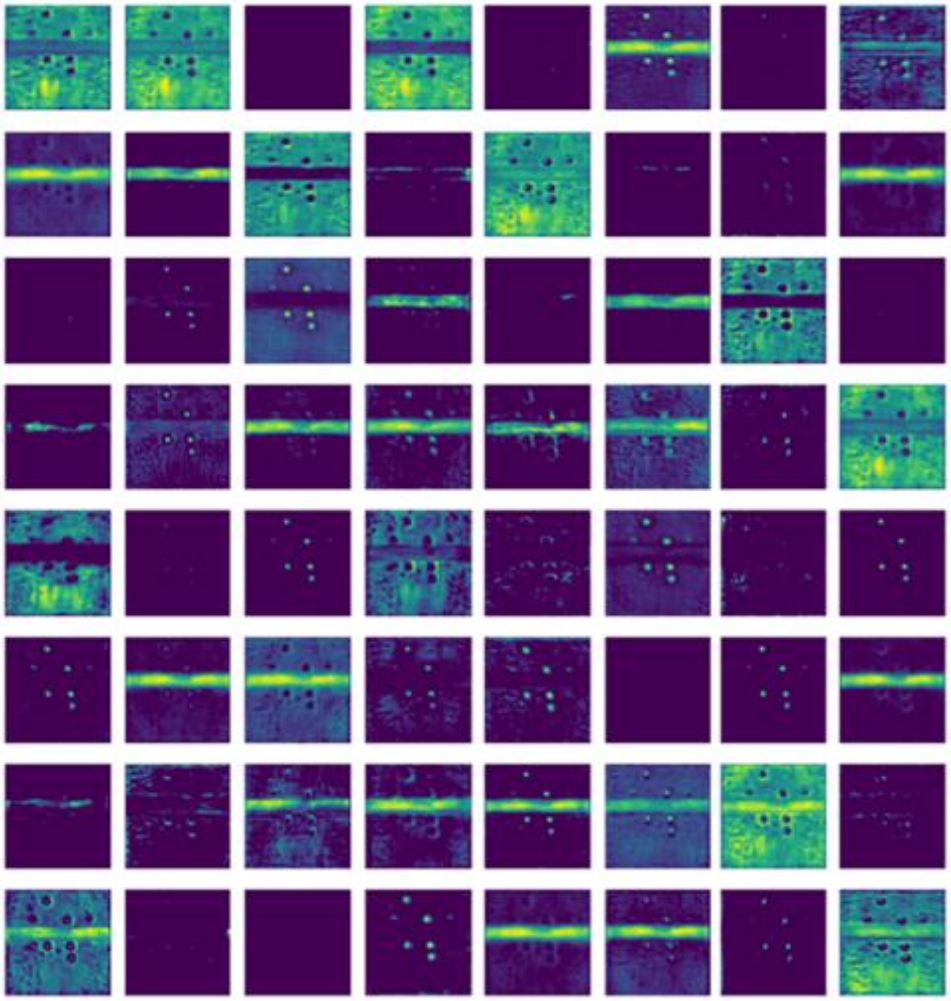


Figure 4.6. Prototype image generation.

4.2.5 Mask coefficient and assembly

The other parallel task of instance segmentation is to generate mask coefficients in anchor-based object detectors (prediction head). Unlike RetinaNet, the original and improved models use a shallower predictor and adopt a mask coefficient branch. As Figure 4.2 shows, in the prediction head, there are $4+c+k$ coefficients per anchor. To subtract the generated prototypes, the tanh activation function is applied. The masks are generated by assembling Protonet output and mask coefficients using a linear combination. A sigmoid nonlinearity is applied to generate final masks. Equation (4-2) shows the mentioned steps. During the training and evaluation process, the final masks are cropped with the ground truth bounding boxes and predicted bounding boxes, respectively.

$$M = \sigma(PC^T) \tag{4-2}$$

where P is an $h \times w \times k$ matrix of prototype masks and C is a $n \times k$ matrix of mask coefficients for n instances surviving NMS and score thresholding.

4.2.6 Loss functions

Three loss functions, including mask loss, classification loss, and box regression loss, are used in data training. Specifically, the mask loss applies pixel-wise binary cross-entropy (BCE) loss function to calculate the loss between the assembled Masks M and the ground truth masks M_{gt}. Mask loss is expressed in Equation (4-3). For classification loss and box regression loss, the functions are shown in Equation (4-4) and (4-5). The corresponding weights for these three loss functions are 1, 1.5, and 6.125, respectively.

$$L_{mask} = BCE(M, M_{gt}) \tag{4-3}$$

$$\hat{g}_j^w = \log(\frac{g_j^w}{d_i^w}) \qquad \hat{g}_j^h = \log(\frac{g_j^h}{d_i^h})$$

$$\hat{g}_j^{cx} = (g_j^{cx} - d_i^{cx}) / d_i^w \qquad \hat{g}_j^{cy} = (g_j^{cy} - d_i^{cy}) / d_i^h \tag{4-4}$$

$$L_{loc}(x, l, g) = \sum_{i \in Pos}^{N} \sum_{m \in \{cx, cy, w, h\}} x_{ij}^k smooth_{L1}(l_i^m - \hat{g}_j^m)$$

$$L_{conf}(x, c) = -\sum_{i \in pos}^{N} x_{ij}^p \log(\hat{c}_i^p) - \sum_{i \in Neg} \log(\hat{c}_i^0) \tag{4-5}$$

$$\text{where } \hat{c}_i^p = \frac{\exp(c_i^p)}{\sum_p \exp(c_i^p)}$$

where c is the softmax loss over multiple classes confidences.

4.3 Experiments and results

To validate the performance of the proposed models and compare them with the original models having their default backbones, five models are trained and tested in this study. Specifically, the original models are YOLACT-ResNet-50 and YOLACT-ResNet-101. The improved models are named as YOLACT-Res2Net-50 and YOLACT-Res2Net-101. In addition, Mask R-CNN, which represents the high mask quality and the high accuracy on object detection, is trained and evaluated, aiming to improve the comparison between Mask R-CNN and the improved models. For the original models and the improved models, the training processes are completed in the same module. For Mask R-CNN, MMDetection (Chen et al., 2019) which is an open-source object detection toolbox based on Pytorch, is adopted for friendly usage, training, and evaluation. The detection results generated from different models are evaluated based on MS COCO evaluation metric (Lin et al., 2014) aiming to compare the results fairly and comprehensively. The

validation curves generated from the training and validation process of the original and improved models are plotted and discussed. For Mask R-CNN, since there are differences between different training modules, only AP, AP_{50}, and AP_{75} are compared and evaluated in Table 4.3.

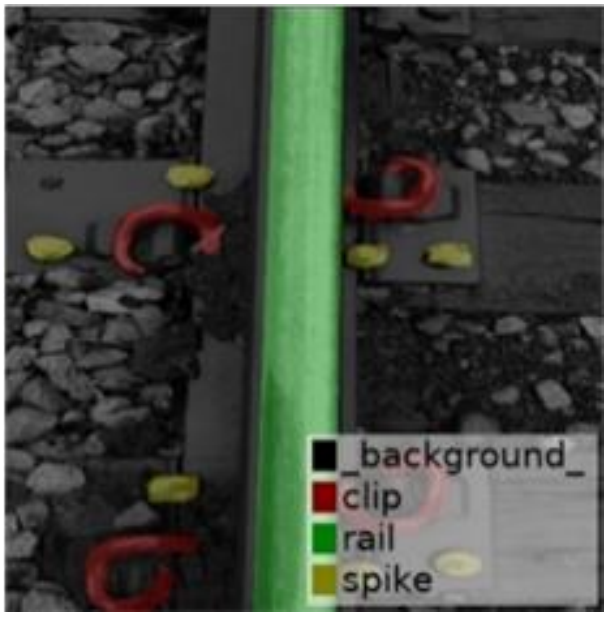
4.3.1 Dataset preparation

The images are saved from video frames recorded on an iPhone® 8 smartphone with a 12-megapixel main camera which has a single wide-angle lens with an f/1.8 aperture. The videos are taken from a railroad section besides 300 Main St. Columbia, SC. The section is between GPS coordinates [33.988208, -81.025973] and [33.989474, -81.025942]. The smartphone is held in hand and the video is taken at a walking speed along the track. A total of 30 minutes of video is recorded and saved on the smartphone. The original video resolution is 1920×1080 and the converted image size is set to 512×512 to meet the training image size requirement. Three types of rail components, including rail, spike, and clip are included in the image database. To prevent overfitting, the training images are processed with image augmentation, including mirroring, rotation (90°), and the combination of rotation (180°) and gaussian noise. A popular labeling tool, labelme (Ketaro Wada, 2016) is employed to generate the annotation files.

The output JSON files are converted to COCO format based on the prepared code for training, validation, and evaluation. Figure 4.7 shows the ground truth and the labeling mask. Note that the background is category 0. The rail, clip, and spike represent category 1, category 2, and category 3, respectively. The category IDs should be correctly associated with the class names. Otherwise, the detection results will have the wrong

labels. Following the general ratio of the cross-validation principle and previous studies (S. Li et al., 2019; Xinxiang Zhang, Dinesh Rajan, et al., 2019), the ratio between the training set and test set is set to 8:2. A total of 1000 images, which are released online for free access, are used for training and test. To reduce the bias and ensure the training processes are statistically significant, the 5-fold cross-validation is performed in the training procedure. Specifically, 1000 images are randomly split into 5 folds and each fold contains 200 images. Each group is taken as a test set and the remaining groups are considered as the training set. Totally, 25 training and tests are for the entire dataset. The evaluation results including the mean values and standard deviations are shown in Table 4.3.

(a)



(b)

Figure 4.7. Example of original jpg image and label result (a) Ground truth (b) instance label visualization.

4.3.2 Training and validation

Transfer learning is a convenient timesaving method to train deep learning models. Since multiple models need to be trained and evaluated, transfer learning, other than training individual models from scratches, is employed in the test. Pre-trained weights for the backbones of the proposed models and original models are implemented in the model initialization stage. Because the new backbones are adapted in the original models, the dictionary of key and value in the pre-trained weight file needs to be updated with the proposed network structure. To avoid program running errors and make sure the training is successfully started, new functions are written to filter the unused layers (such as some batch normalization layers) in pre-trained weight files and make the proposed architecture correspond to the original settings.

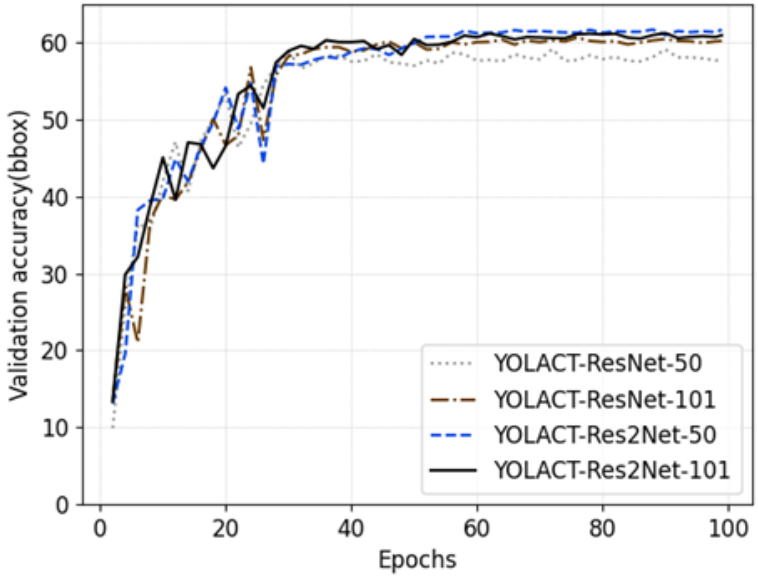
Generally, the training process aims to minimize the overall loss by optimizing the model parameters (Wang & Cheng, 2020). In other words, the lower the overall loss is, the better the model is. In this study, the popular stochastic gradient descent (SGD) optimizer is applied to train the improved models. Table 4.2 shows the training hyperparameters. The training iteration is 10k and the initial learning rate is 10^{-3}. The learning rate is a vital hyperparameter in model performance. A small value will result in a long training process, and a large value will lead to hasty and unstable training. In this study, the initial learning rate is divided by 10 at iterations 2k, 6k, and 7k by using a weight decay of 5×10^{-4} and a momentum of 0.9.

Table 4.2. Training hyperparameters for our proposed models.

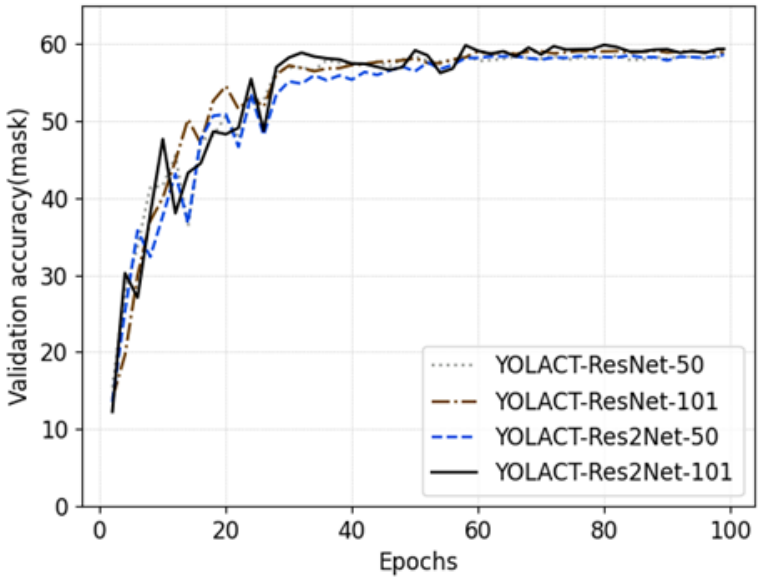
Hyperparameters	Value
Input size	512 × 512
Initial learning rate	10^{-3}
Weight decay	5×10^{-4}
Momentum	0.9
Iterations	10k
Batch size	8

To configure and expedite the model training, the Pytorch library developed by Facebook and the packages of CUDA 10.2 with Cudnn 7.6.5 developed by NVIDIA are included in this study. All the training processes are accomplished in a lab server. The server system is CentOS 7.2 with Inter i7 CPU. The GPU is NVIDIA 1080 Ti with driver 440.33. The training time for each model takes around 1 to 1.5 hours. Figure 4.8 shows

the validation accuracy of a test model on a randomly selected training set. Figure 4.8 (a) clearly shows the proposed models outperform the original model in terms of the validation accuracy of the bounding box. The original model has the lowest validation accuracy value, which is 57.65, while the proposed YOLACT-Res2Net-50 has the highest validation accuracy value, which is 61.65. In Figure 4.8 (b), the validation accuracies of the mask are close among these four models. Still, the proposed YOLACT-Res2Net-101 has the highest value of 59.32 and the original model has the lowest value of 57.95.



(a)



(b)

Figure 4.8. Representative validation accuracy of original YOLACT models and proposed YOLACT-Res2Net-50 and YOLACT-Res2Net-101.

4.3.3 Detection performance and evaluation

In this study, COCO mAP (mean average precision), a common metric in measuring the accuracy of object detectors, is applied to evaluate the detection performance of different models. Before analyzing the mAP results, its important components of intersection over union (IoU) and average precision (AP) need to be explained. IoU measures the overlap between the predicted boundary and the ground truth.

Figure 4.9 shows the definition of IoU. Generally, the IoU threshold of 0.5 is to determine if the prediction is a true positive or a false positive. AP is the averaged precision across all values of recall between 0 and 1, and it can be calculated by taking the area under the precision recall (PR) curve. Note AP is averaged over all categories, therefore there is no difference between mAP and AP in this study. The calculation of precision and recall are introduced in Equation (4-6) and (4-7). From our training logs, the representative PR curves of the improved models and the original models can be seen in Figures 4.11 and 4.12. Table 4.3 shows the COCO mAP results of all trained models.

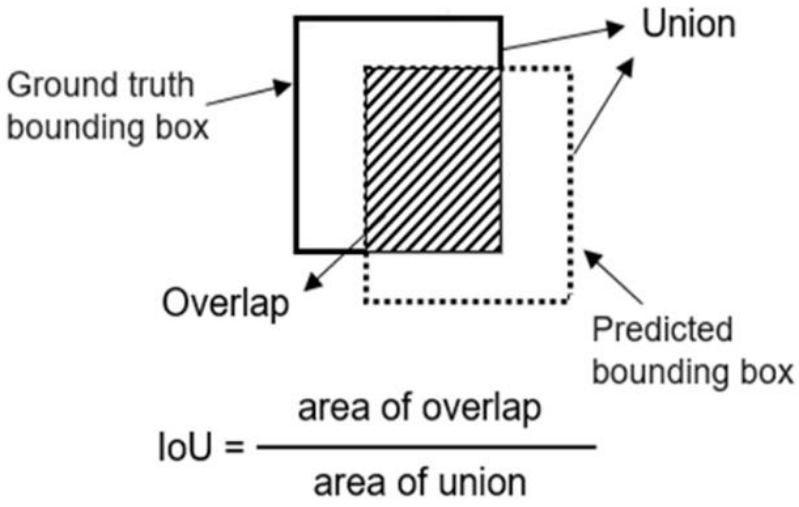


Figure 4.9. The definition of IoU.

$$Precision = \frac{TP}{TP + FP} \tag{4-6}$$

$$Recall = \frac{TP}{TP + FN} \tag{4-7}$$

where TP is true positive, TN is true negative, FP is false positive, and FN is false negative.

Typically, the precision-recall curve shows the relationship between precision and recall of different thresholds. Figures 4.10 and 4.11 are plotted with the thresholds of 50% and 75%, respectively. A high area corresponds to a high recall and a high precision in each class. Meanwhile, a high precision indicates the detection has more relevant results than the irrelevant cases and a high recall means the model returns most of the relevant results. In Figure 4.10 and Figure 4.11, with a threshold of 50%, all the areas are over 0.97 except for the result of YOLACT-ResNet-50. When the threshold is 75%, the best detection of the rail is from YOLACT-Res2Net-50, of which the area is 0.616. Similarly, the best detection on the clip is from YOLACT-Res2Net-101, and the best detection on the spike is from YOLACT-Res2Net-50.

Based on these results, it is reasonable to believe the improved models outperform the original models. To evaluate the detection performance on a wider scale, the MS COCO evaluation matric with AP_{50} and AP_{75} are performed and shown in Table 4.3. It shows that the proposed models, YOLACT-Res2Net-50 and YOLACT-Res2Net-101 have competitive performance in the detection of bounding boxes and masks. For detecting bounding boxes, considering the AP values, the proposed models outperform the original models by 3 AP and 2.5 AP, respectively. While Mask R-CNN achieves the highest AP value, 63.9. With a 50% IoU threshold, there is not much difference in AP values between different models. However, the improved models perform better when the IoU threshold is 75%.

The proposed models outperform the original models by 5.8 AP and 7 AP, respectively. Meanwhile, Mask R-CNN has the highest AP value, 76.7. Regarding the standard deviation (SD), all the models have low SD values with different training and testing sets, indicating the bounding box prediction results are solid, reliable, and statistically significant. The improved models are more effective on the bounding box prediction compared to the original ones.

In terms of the instance segmentation performance, the proposed models are able to improve the mask accuracy compared to the original models and Mask R-CNN. For the AP values, YOLACT-Res2Net-101 has the highest mask AP value, which is 2.6 higher than the value of the original one. YOLACT-Res2Net-50 improves AP by 4 compared to its original model. Regarding the performance of instance segmentation with an IoU threshold of 50%, the proposed models both achieve AP_{50} of 97.3 which are higher than the values of the original models and Mask R-CNN. When the IoU threshold is 75%, YOLACT-Res2Net-50 achieves the AP_{75} of 69.7 which is 3.2 higher compared to the value of its original model. While, the proposed YOLACT-Res2Net-101 model has a lower AP_{75} value which is 3.9 lower compared to its original model. Since the AP value computation needs different thresholds, although the proposed model performs a bit worse with one of the thresholds, overall it still performs very well. For the SD values on mask detection, it can be found that they are lower on the AP50 but are higher on the AP75. This needs to be considered for future improvement.

The detection strategy of Mask R-CNN is to propose lots of candidate proposals, so Mask R-CNN performs better on the bounding box and it has been proved to be effective as shown in Table 3. However, as a trade-off, from Figure 4.12 it can be seen

the detection speed of Mask R-CNN is much lower compared to YOLACT models. The average inference speed of Mask R-CNN is 5.3 FPS with a standard deviation of 0.36, while the original and improved models inference time is close to or over 30 FPS, indicating a possible real-time application in the field inspection. The inference speeds of the proposed YOLACT-Res2Net-50 and YOLACT-Res2Net-101 models are 35.9 FPS (SD=0.65) and 28.4 FPS (SD=0.92), respectively. They are slightly slower compared to the original models which have 40.3 FPS (SD=0.40) and 32.4 FPS (SD=0.76). The possible reason could be that more receptive fields cost more computational power and this leads a potential optimization research in the future. In short, the proposed YOLACT-Res2Net-50 performs the best on bounding box detection in a real-time speed, and YOLACT-Res2Net-101 has the best mask accuracy.

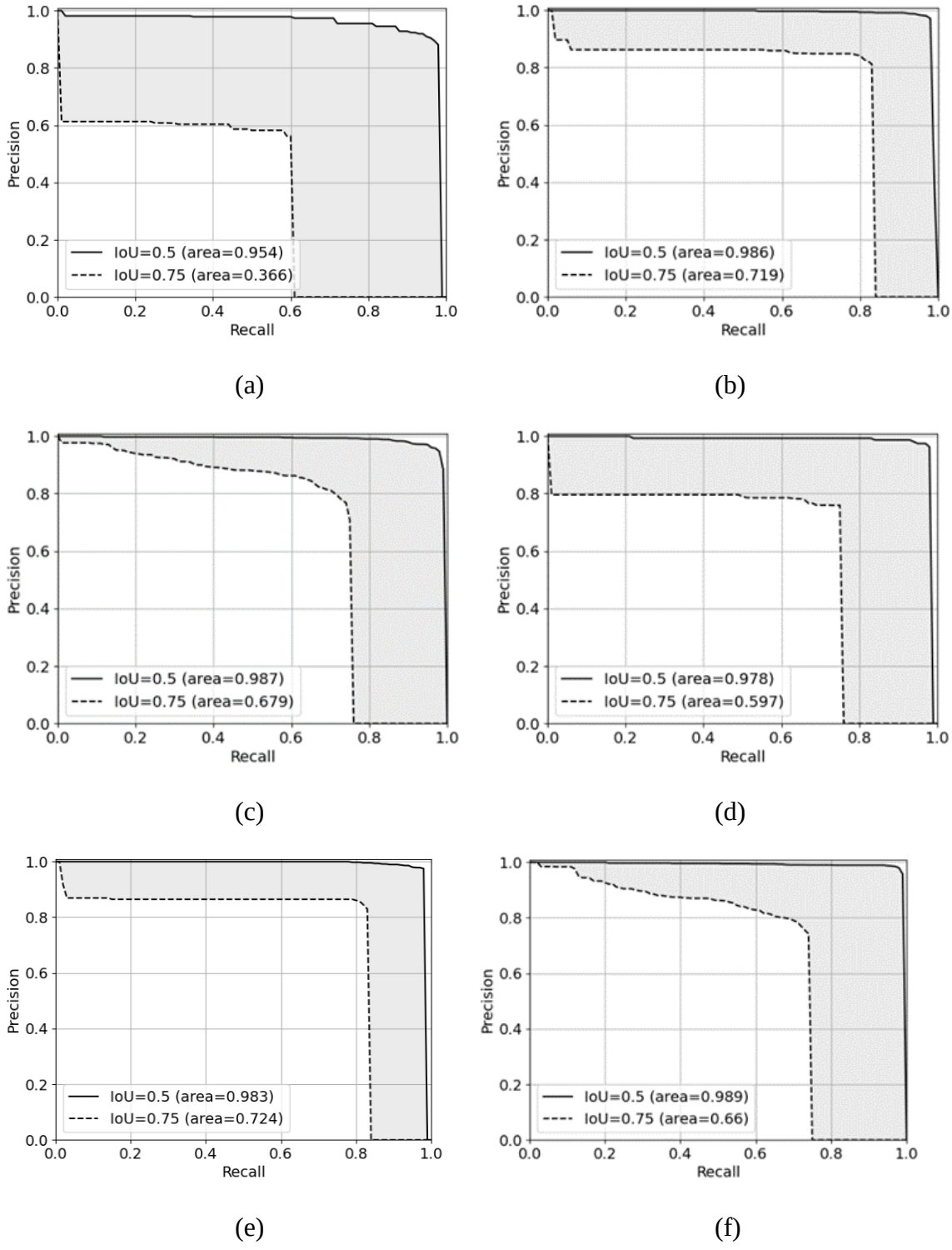


Figure 4.10. Representative precision-recall curves of YOLACT-ResNet-50 and YOLACT-ResNet-101 on each category. (a)-(c): rail, clip, and spike on YOLACT-ResNet-50; (d)-(f): rail, clip, and spike on YOLACT-ResNet-101.

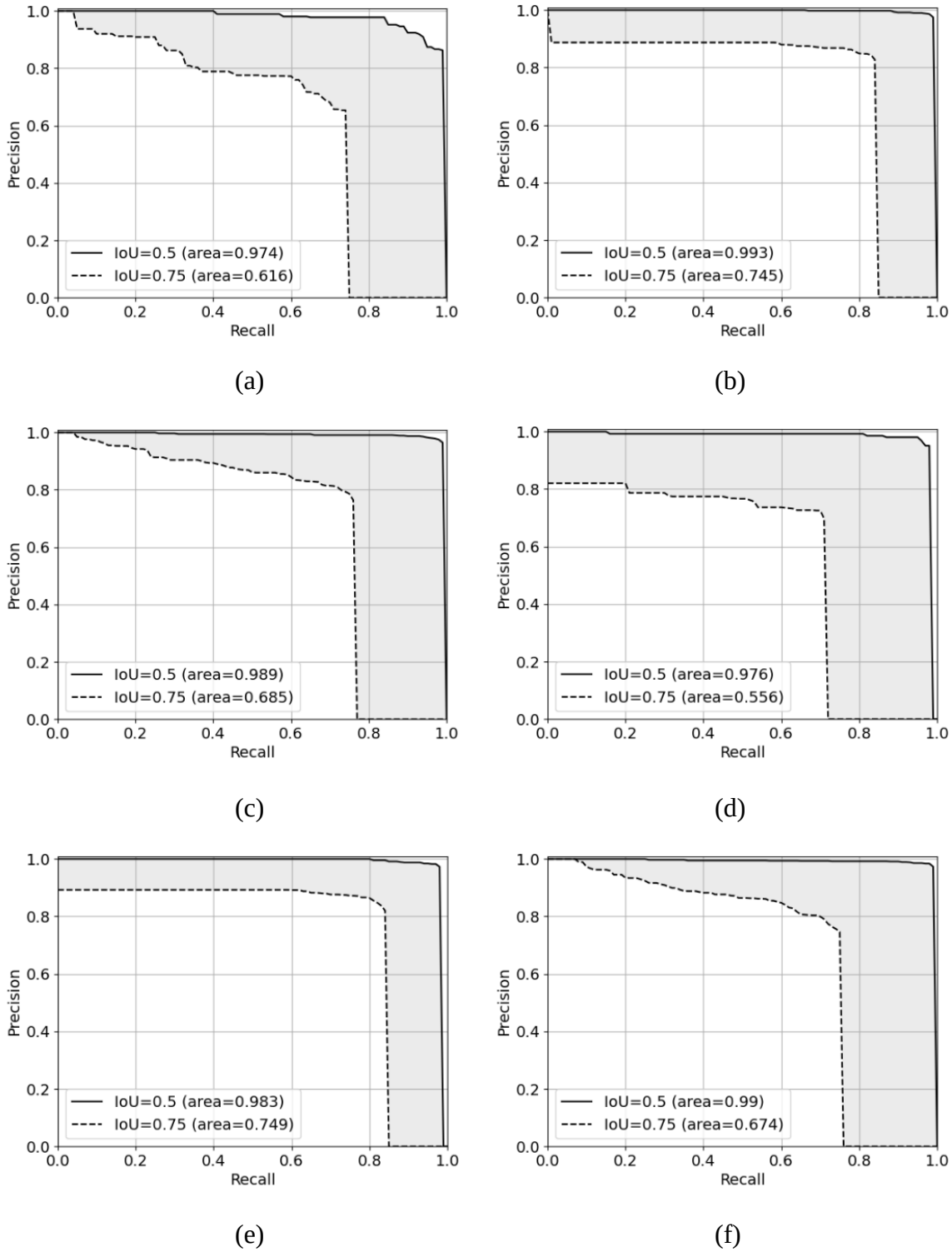


Figure 4.11. Representative precision-recall curves of YOLACT-Res2Net-50 and YOLACT-Res2Net-101 on each category. (a)-(c): rail, clip, and spike on YOLACT-Res2Net-50; (d)-(f): rail, clip, and spike on YOLACT-Res2Net-101.

Table 4.3. COCO mAP results with different models in this study on custom dataset.

	Method	AP	SD	AP_{50}	SD	AP_{75}	SD
bbox	YOLACT-Res2Net-50	59.4	2.4	97.7	1.6	65.6	6.2
	YOLACT-Res2Net-101	59.9	2.2	97.9	1.2	67.3	5.2
	YOLACT-ResNet-50	56.4	1.8	97.1	1.4	59.8	5.1
	YOLACT-ResNet-101	57.4	1.9	97.1	1.9	60.3	3.9
	Mask R-CNN	63.9	3.4	97.6	1.5	76.6	7.7
mask	YOLACT-Res2Net-50	63.2	7.3	97.3	1.7	69.7	12.8
	YOLACT-Res2Net-101	63.6	6.8	97.3	1.9	64.4	12.7
	YOLACT-ResNet-50	59.6	6.6	96.5	1.8	66.5	12.0
	YOLACT-ResNet-101	61.0	6.9	96.5	2.5	68.3	12.3
	Mask R-CNN	61.6	6.6	97.2	1.6	64.4	12.7

Note: AP50 is AP @ IoU = 0.5; AP75 is AP @ IoU = 0.75; SD is standard deviation.

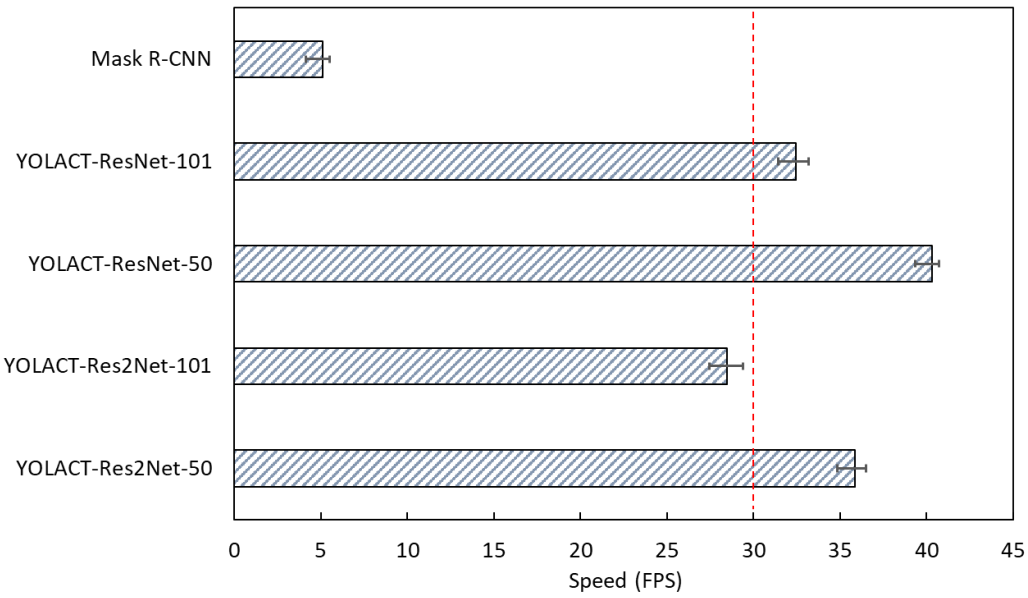


Figure 4.12. Detection speed of different models.

4.3.4 Influence of light condition

In the field practice, environmental conditions are complex, and the track components are relatively small, making visual inspections very challenging. Besides, the inspection window has been reducing due to the busier timetables. Therefore, any detection model has to be robust enough to accommodate harsh environmental conditions for field applications. One of the typical challenges in the field is the light condition. To test the detection performance under different light conditions, five different light intensities are used on the 24-bit depth images, which are original light, light-10%, light-30%, light-50%, and light-70%. Figure 4.13 shows the testing results under the selected five visibility conditions.

Looking at the ground truth with naked eyes, there are obvious differences between the normal and dimmed conditions. As the light decreases, the image background becomes darker, and the rail components are blended into the background. It

is indeed challenging to distinguish the rail components by naked eyes without sufficient light as shown in the first row in Figure 4.13. Furthermore, the specific image presented in Figure 4.13 is taken during a rain, making it more troublesome for detection. In this particular image, there are five spikes, one rail, and four clips. The results of the detection accuracy of each model under different light conditions are presented in Figure 4.14. For YOLACT-ResNet-50, in the first four light conditions, it successfully detects all the spikes and the clips. However, it cannot detect the rail and add a mask on it. In the darkest condition, light-70%, it missed two spikes. YOLACT-ResNet-101 is similar to YOLACT-ResNet-50. It also fails to detect the rail, meanwhile, it misses three spikes under the light-70% condition. Regarding the proposed YOLACT-Res2Net-50, except for the last condition, it successfully detects the rails and adds the masks on them. Under the light-70% condition, it misses one spike. The proposed YOLACT-Res2Net-101 also performs well under each condition. It detects the rails and adds mask on them under four different light conditions, but it misses three spikes in the darkest condition. It is worth noting that the last model, Mask R-CNN has good performance in different light conditions. It detects three rails out of all rails. Meanwhile, it just misses two spikes in the darkest condition.

Overall, our improved models outperform the original models and Mask R-CNN under five lighting conditions. It should be mentioned that the test image is randomly selected from the image set. To some extent, it can reflect the real performance in the field practice. Currently, limited by the training data, other types of track components are not included. In the future, the detection performance can be improved with more data and further enhancement of the model.

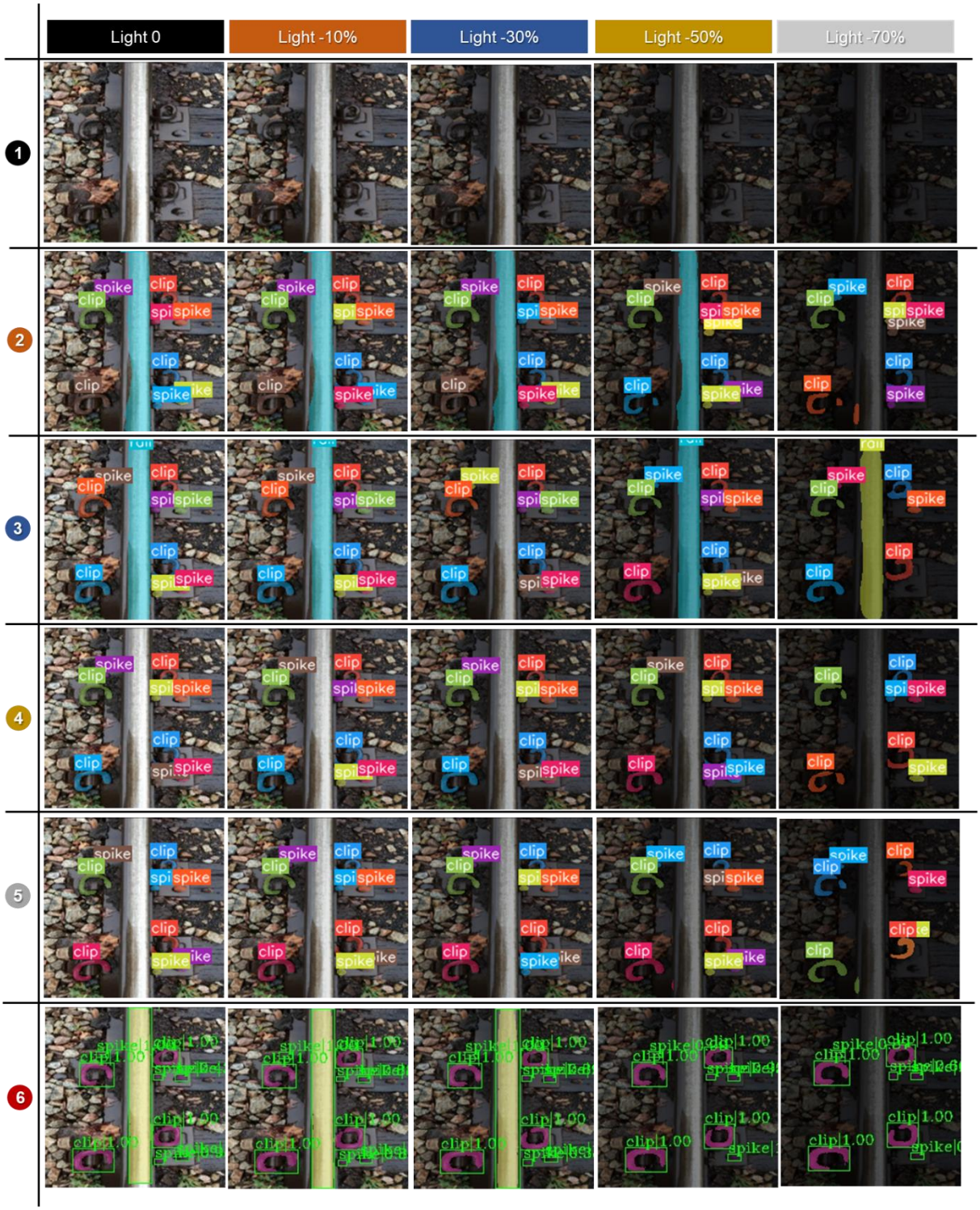


Figure 4.13. Representative detection results on the different light condition 1: Ground truth; 2: YOLACT-Res2Net-50; 3: YOLACT-Res2Net-101; 4: YOLACT-ResNet-50; 5: YOLACT-ResNet-101; 6: Mask R-CNN.

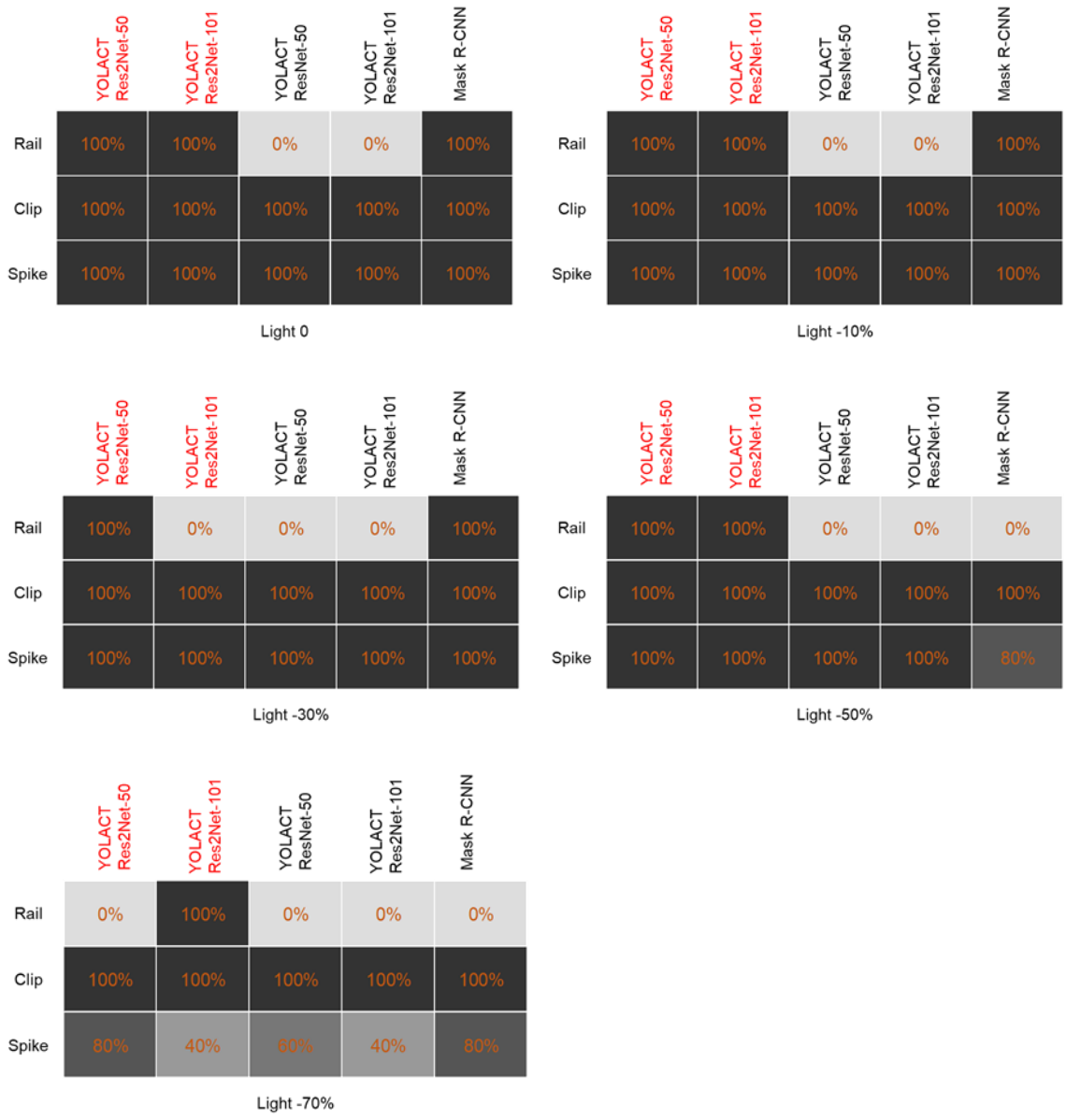


Figure 4.14. Detection accuracy under different illuminations.

4.4 Summary

This chapter presents the improved real-time instance segmentation models and their application on the railroad track component detection. The improved models are developed based on a fully convolutional model which includes backbone, FPN, Protonet, and prediction head. The input features are extracted by the improved backbone and the FPN structure is responsible for detecting objects on different scales. The instance mask is generated by two parallel tasks. One is accomplished by the Protonet and the other one is achieved by the prediction head which also generates the anchors for bounding boxes. To accelerate the detection speed, the fast NMS is applied. During the training, the first track components image database is built. A total of 1,000 images are used for training and validation. Cross-validation is performed to validate these experiments are statistically significant. Five models, including the two proposed models and three other popular models, are trained and evaluated based on precision-curve and MS COCO evaluation metrics. Experimental results show our proposed models outperform the state-of-the-art models on detection accuracy.

To our best knowledge, this chapter is the first attempt to apply real-time instance segmentation with high accuracy and real-time speed (>30 FPS) on a single GPU, which is a more challenging task compared with object detection, semantic segmentation, and previous instance segmentation (inference speed < 30 FPS), in the railroad inspection and even civil engineering. Under the real-time speed condition that requires a processing speed above 30 FPS, the proposed models outperform the original models and the Mask R-CNN model in terms of detection accuracy of the bounding box and mask. In the future, when the improved model implemented on a cost-efficient mobile computing

board that has enough computational power, the inspection on the rail track components can be more cost-effective, efficient, and accurate. Under the different light conditions, our proposed models outperform the other models, proving the robustness on low visibility conditions. This chapter demonstrated the possibility of applying the cutting-edge deep learning technology into the railroad track components inspections, paving the road for future applications. However, there is indeed room for improvement. Future extensions will focus on building the first comprehensive railroad image database and improving detection without compromising speed.

CHAPTER 5 A COMPUTER VISION-BASED REAL-TIME RAILROAD TRACK COMPONENTS INSPECTION FRAMEWORK BASED ON YOLOV4[5]

[5] Guo, Feng, Yu Qian, Yunpeng Wu, Zhen Leng, and Huayang Yu. Automatic railroad track components inspection using real-time instance segmentation. *Computer-Aided Civil and Infrastructure Engineering* 36, no. 3 (2021): 362-377.

According to the Federal Railroad Administration (FRA) database, track component failure is one of the major factors causing train accidents. To improve railroad safety and reduce accident occurrence, tracks need to be regularly inspected. Many computer-aided track inspection methods have been introduced over the past decades, however, inspecting missing or broken track components still heavily relies on manual inspections. To address those issues, this chapter proposes a real-time and cost-effective computer vision-based framework to inspect track components quickly and efficiently. The cutting-edge convolutional neural network, YOLOv4 is trained, tuned, and evaluated based on the images in a public track components image database. Compared with other one-stage object detection models, the customized YOLOv4-hybrid model can achieve 94.4 mean average precision (mAP) and 78.7 frames per second (FPS), which outperforms other models in terms of both accuracy and processing speed. It paves the way for developing portable and high-speed track inspection tools to reduce track inspection cost and improve track safety.

5.1 Introduction

In the United States, regular track inspections are mandatory by the federal railroad administration (FRA) due to the fact that infrequent or inadequate railroad inspection is one of the major causes of railroad accidents (FRA, 2018). Considering the tens of thousands of miles of railroad tracks and the busy operations, it is impossible and impractical to inspect each track segment on a daily basis. For the last few decades, rail inspections solely rely on the available personnel and their experience. To improve the railroad track safety and increase the inspection efficiency, automatic inspection methods and equipment have been developed over the years, such as using lasers to measure the track geometry and rail profiles (Liu, Li, Wu, & Meng, 2014), ground penetration radar (GPR) to assess ballast fouling conditions (Leng & Al-Qadi, 2010; Roberts, Rudy, Al-Qadi, Tutumluer, & Boyle, 2006), ultrasonic or Eddy current to identify rail internal defects (Hackel, Stein, Maindorfer, Lauer, & Reiterer, 2015; Rizzo et al., 2010), and Lidar to detect track fouling (Artagan, Ciampoli, D'Amico, Calvi, & Tosti, 2019). Those automatic inspection methods greatly improve track safety and reduce the accident frequency. However, those automatic inspection methods typically require expensive and special equipment as well as skillful operators, which limits the application in the field.

Thus, till now, most of the railroad track inspection work, except for the track geometry measurement, is still very labor- and time-intensive, especially for inspecting missing or broken track components. The dilemma between track operation and track inspection has been more pronounced with the increasing transportation demand. On one side, the rapid tonnage accumulation causes more track damage on the track components which requires more frequent track inspections. On the other side, a busier schedule

leaves very tight window for inspection operations. This issue is more problematic with the Class I railroad mainlines due to the saturated traffic volume. Over the past decades, missing and broken railroad track components, such as spikes, clips, rails, ties, and tie plates, are among of the leading factors that causing railroad accidents. Uncountable financial loss and even fatalities are caused by a limited number of missing or broken track components. For instance, in 2014, broken spikes caused a 120-car Norfolk Southern train derailment at Vandergrift, PA, which spilled between 3,000 and 4,000 gallons of crude oil. Another example is a Union Pacific train carrying 96 cars has derailed near Mosier, Oregon, resulting in 42,000 gallons of Bakken oil being spilled and a severe oil train fire due to broken spikes (Dake, 2016; A. Hardway, 2014; Tom Roadcap, 2018) . Thus, an automatic rail components inspection system with high accuracy, fast processing speed, and low cost, is in urgent need.

The challenges of railroad track inspection are the work needs to be completed in a very limited time and requires relatively easy operations by the railroad personnel. Considering the engineering difficulties and inspired by the successful applications of the third generation of You Only Look Once (YOLO) detectors, which are the state-of-the-art object detection algorithm in the deep learning(Zhao, Zheng, Xu, & Wu, 2019). This chapter aims to develop a fast, accurate, yet low computation demand track inspection method. In this work, a real-time railroad track components inspection framework based on the just-released YOLOv4 (Wu, Lv, Jiang, & Song, 2020) is proposed for track components inspection. The contributions of this study are: (1) An improved YOLOv4 model is firstly proposed for the railroad track components inspection; (2) Influence of different activation functions and loss functions on the performance of YOLOv4 trained

with a customized dataset is tested and compared; (3) The comparison between the improvedYOLOv4 and other State-of-The-Art (SOTA) models on the detection performance has been tested, summarized, and discussed; (4) The impact of different image sizes and illumination conditions on detection performance has been illustrated; and (5) The detection performance of YOLOv3 and the modified YOLOv4 on "missing components" or "fake components" is compared. Note the former two contributions are focusing on the model improvement, while the latter three contributions are directly related with the field application of automatic railroad track inspections.

5.2 Methodology

5.2.1 Proposed neural network architecture

In this study, to efficiently and accurately inspect the rail track components with a real-time speed on a single GPU, the newly developed one-stage object detection framework, YOLOv4 (Bochkovskiy, Wang, & Liao, 2020), is modified and trained based on the needs of railroad track inspection. Then, the developed models are evaluated and compared with SOTA models. The overall methodology is described in Figure 5.1.

Data preparation: the training files mainly include images and the corresponding annotation files which are built using the labelme (Ketaro Wada, 2016), an open-source labeling tool for the model training.

Model training: three object classes, rail, clip, and spike are selected for the training purpose. A public available data set with a total of 1,000 images are used for training (Guo, Qian, Wu, et al., 2021). To better fit the training needs of the customized image data, the original activation function of YOLOv4 is modified to improve the

prediction performance. The original activation function of YOLOv4 is replaced with two different activation functions and one combination of the hybrid activation functions, aiming to construct three different YOLOv4 models. A total of five models including the three modified YOLOv4 models, original YOLOv4, and original YOLOv3 are trained with the PyTorch library.

Performance evaluation: The precision-recall (PR) curve, precision, recall, F1 score, mean average precision (mAP), and inference time are used to evaluate and compare the training results of different models. The influence of different image sizes and illumination conditions on the prediction performance are evaluated and discussed. The capability to detect "missing components" and "fake components" are also investigated.

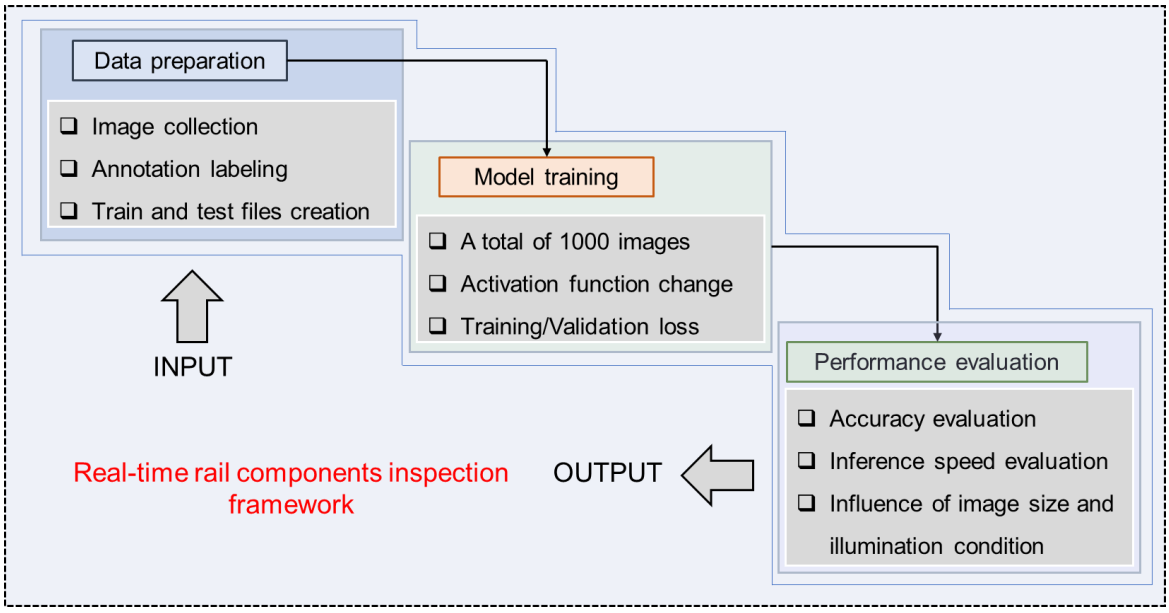


Figure 5.1. Overview of proposed methodology in this study.

5.2.2 Network architecture

The YOLOv4 model aims to optimize the speed and accuracy on real-time object detection based on YOLOv3. To balance the speed and accuracy, the backbone of Cross-Stage-Partial-connections (CSP) Darknet-53 is utilized in the new network architecture of YOLOv4. Based on the introductions of YOLOv4, CSP Darknet-53 has better performance on COCO dataset (Lin et al., 2014). To increase the receptive field, which affects the unit of the network, spatial pyramid pooling (SPP) block (He, Zhang, Ren, & Sun, 2015), and the modified path aggregation network (PANet) (Liu, Qi, Qin, Shi, & Jia, 2018), are integrated into YOLOv4. For the detection head, YOLOv3 head is assembled in the new model, aiming to predict objects in multiple scales. In this study, there are a total of three object classes. Therefore, the number of filters = (classes + 5) × 3 = 24. Figure 5.2 presents the overview of the YOLOv4 network architecture.

As shown in Figure 5.2, the skeleton of YOLOv4 mainly includes CSP Darknet-53, SPP block, PANet, and the prediction head. Specifically, CSP Darknet-53 assembles Darknet-53 and CSPNet, which includes the partial dense block and the partial transition layer to enhance the variability of the learned features within different layers. The detailed parameters of output features are presented in Figure 5.2. The SPP block is used to increase the receptive field and separate the most significant context features without sacrificing inference speed. Same as YOLOv3, there are three scales in the detection head. Since the inputs are 512 × 512, the parameters of the detection head in YOLOv4 are 64 × 64 × 24, 32 × 32 × 24, and 16 × 16 × 24, respectively.

There are other improvements implemented in YOLOv4, important components are Weighted-Residual-Connections (WRC) (Shen, Gan, & Zeng, 2016), Cross mini-Batch Normalization (CmBN) (Yao, Cao, Zheng, Huang, & Lin, 2020), Mish Activation (Misra, 2019), Complete Interaction over Union (CIoU) loss (Zheng et al., 2020b), Mosaic data augmentation, and DropBlock regularization (Ghiasi, Lin, & Le, 2018). Generally, it is believed these improvements can effectively improve the detection speed and accuracy on the COCO dataset (Lin et al., 2014). Unfortunately, the reality is the COCO dataset hardly meets the specific needs of the field applications, especially for civil engineering applications which highly demand suitable image data to complete specific tasks, such as asphalt pavement crack detection, concrete pavement crack detection, and railroad defects identification.

It is worth noting that, other than the dataset provided by Guo et al.(2021), there has been no other public image dataset related to rail track components such as rail, clip, and spike, which are critical track components to ensure the track integrity and safe operation. Besides, few studies are using YOLOv4 to work on a small dataset and discuss the role of different activation functions in prediction performance with a customized dataset of railroad engineering. In fact, based on the training experience of the authors and relevant studies (Misra, 2019), different activation functions do impact the prediction performance. Therefore, in the following part, the influences of different activation functions on training results and prediction performance with the customized dataset are investigated in detail.

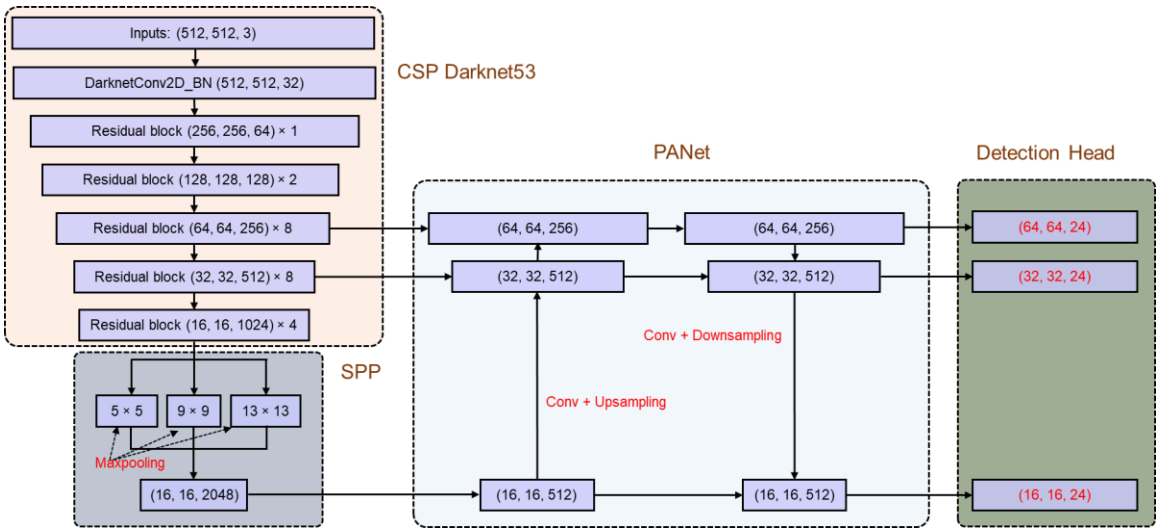


Figure 5.2. Overview of the YOLOv4 network architecture.

5.2.2.1 Activation functions

The *activation function* is a critical part of the neural network and has proven key to high performance among a wide range of tasks. It determines whether a neuron in the neural network should be activated or not, and is characterized by a variety of properties (derivative, monotonicity, etc.) for a customized training (Eger, Youssef, & Gurevych, 2019). Therefore, the choice of activation functions plays an important role in the training dynamics and performance (Ramachandran, Zoph, & Le, 2017). To train the model efficiently and make the prediction more accurate on our customized dataset, popular activation functions including Mish, Swish, and Leaky-ReLU, are implemented and configured for the training files. The following will introduce the selected different activation functions in detail.

Mish activation function

Mish is a novel activation function proposed by Diganta (Misra, 2019). It is defined in Equation (5-1). Like other popular activation functions, it can be easily implemented in the PyTorch and TensorFlow frameworks with well-developed commands. Specifically, it is bounded below and unbounded above with a range of [$\approx$-0.31, ∞). The properties of smooth, non-monotonic, unbounded above, and bounded below are important to improve the training performance. Based on the training experience, using Mish could obtain an improved prediction, but there are longer training time and more memory cost. The graph of mish can be seen in Figure 5.3 (a). Even though Mish is successfully used on the newly released YOLOv4 model with the COCO dataset, it has high computational expense and costs more time during training, which means it might not be the most suitable activation function on a featured dataset.

$$f(x) = x \cdot tanh(\ln(1 + e^{x})) \tag{5-1}$$

Swish activation function

Swish is an activation function proposed by Prajit (Ramachandran et al., 2017). It performs well on kinds of challenging datasets under popular deep learning libraries. Based on the experiment results reported by Prajit (Ramachandran et al., 2017), it can outperform ReLU on ImageNet by 0.9% for Mobile NASNet-A and 0.6% for Inception-ResNet-v2, respectively. The design of this activation function is inspired by the application of the sigmoid function on the long short-term memory (LSTM) and highway networks. Similar to Mish, it is bounded below, unbounded above, non-monotonic, and smooth. According to Prajit (Ramachandran et al., 2017), the non-monotonicity property

distinguishes Swish from other popular activation functions such as ReLU. Meanwhile, smoothness is useful for model generation and optimization. Equation (5-2) gives the definition of Swish and Figure 5.3 (b) presents the graph. From Figure 5.3 (d), it can be found that Swish has little variance from Mish, which can be found by comparing Figure 5.3 (a) and (b).

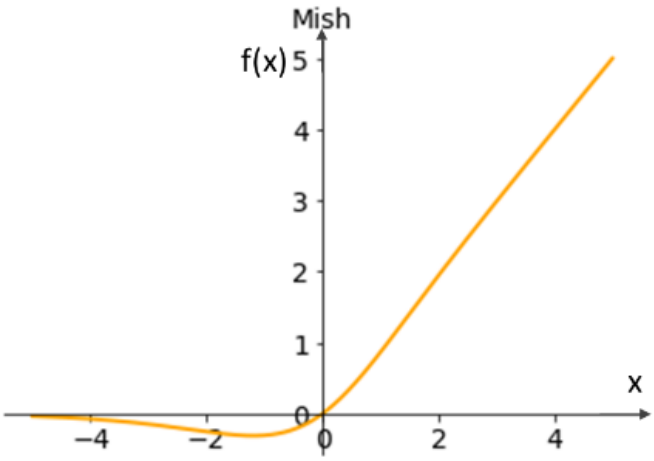
$$f(x) = x \cdot \sigma(x) \tag{5-2}$$

where $\sigma(x) = (1+\exp(-x))^{-1}$ is the sigmoid function.

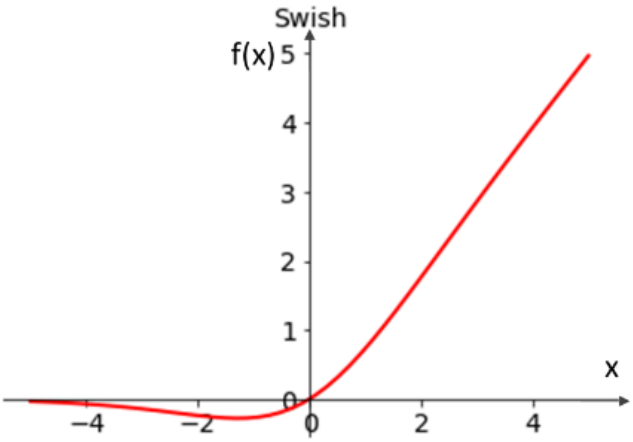
Leaky-ReLU

The Leaky Rectified Linear Unit (Leaky-ReLU) (Maas, Hannun, & Ng, 2013) is one of the most commonly used activation functions in current deep CNNs. Compared with previous activation functions such as tanh and sigmod, it can address the issue of gradient vanishing and keep the weight updates alive along the propagation process. The definition of Leaky-ReLU is shown in Equation (5-3). There is an alpha parameter which is used for solving the problem of dead neurons brought by its predecessor, ReLU. The alpha parameter can ensure the gradients would not be zero during the entire training process so that the training performance can be improved. Even though there are a number of activation functions trying to replace Leaky-ReLU, such as parametric rectified linear unit (PReLU), concatenated rectified linear unit (CReLU), and randomized leaky rectified linear unit (RReLU), none of which can achieve the popularity like Leaky-ReLU. The graph of Leaky-ReLU is shown in Figure 5.3(c).

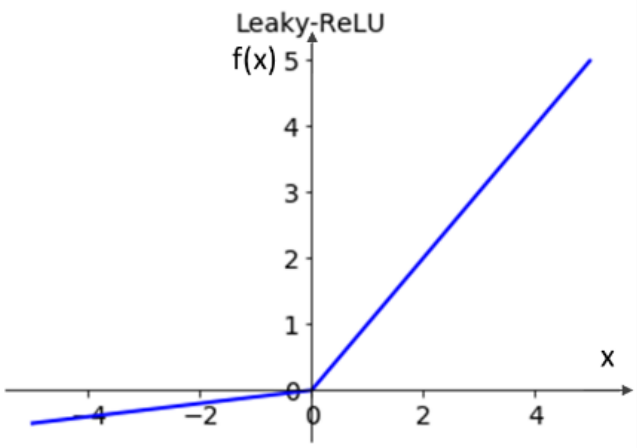
$$f(x)=\begin{cases} x & \text{for } x \geq 0 \\ \alpha x & \text{for } x < 0 \end{cases} \tag{5-3}$$



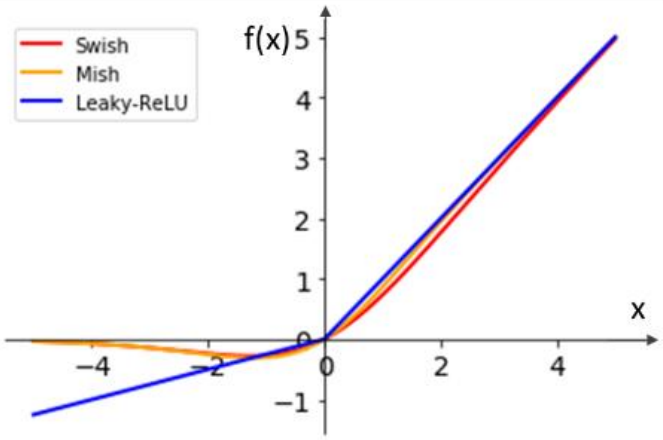
(a)



(b)



(c)



(d)

Figure 5.3. Plots of different activation functions (a) Swish (b) Mish (c) Leaky-ReLU (d) Overview of three activation functions in the same coordinate system.

5.2.2.2 Loss functions

In current object detection models, bounding box regression is a popular approach to predict the localization boxes on the input images. The previous generation YOLO detector, YOLOv3 (Darknet version) computes the bounding box loss through mean squared error loss (MSE) which needs the center point coordinates, height, and width of the predicted and ground truth bounding boxes. However, MSE loss cannot consider the

integrity of the object itself but only treats these parameters as independent variables. To achieve a better performance, for the YOLOv3 (PyTorch version) model trained in this study, the bounding box regression is the generalized IoU (GIoU) which takes the converge area, shape, and orientation all into consideration. However, GIoU needs more iterations for converging, and it still could produce inaccurate results depending on the target and the input image.

To improve the bounding box regression in terms of speed and accuracy, a novel loss function CIoU (Zheng et al., 2020b) with a faster convergence speed and better performance on bounding box regression is adopted in YOLOv4. For making use of the typically ignored geometric information such as the overlap area, aspect ratio, and central point distance in the bounding box regression, CIoU imposes the consistency of aspect ratios for bounding boxes. The CIoU loss is shown in Equation (5-4). The equation of IoU can be seen in Equation (5-5). The definition of trade-off parameter α can be seen in Equation (5-6). The consistency of aspect ratio *v* can be seen in Equation (5-7).

$$L_{CIoU} = 1 - IoU + \frac{\rho^2(\mathbf{b}, \mathbf{b}^{gt})}{c^2} + \alpha v \tag{5-4}$$

$$IoU = \frac{\left| B \cap B^{gt} \right|}{\left| B \cup B^{gt} \right|} \tag{5-5}$$

$$\alpha = \frac{v}{(1 - IoU) + v} \tag{5-6}$$

$$v = \frac{4}{\pi^2}(arctan\frac{w^{gt}}{h^{gt}} - arctan\frac{w}{h})^2 \tag{5-7}$$

where **b** and $\mathbf{b}^{gt}$ is the centroid of B and B^{gt} (see Equation (5-4)), c is the diagonal length of the smallest enclosing box covering ground truth and prediction bounding

boxes. $\rho(\cdot)$ is the Euclidean distance, α is a positive trade-off parameter, ν computes the consistency of aspect ratio. $B^{gt} = (x^{gt}, y^{gt}, w^{gt}, h^{gt})$ is the centroid coordinate, width, and height of ground truth bounding box, and $B = (x, y, w, h)$ is the centroid coordinate, width, and height of the prediction bounding box. w^{gt} is the width of the ground truth bounding box, w is the width of the prediction bounding box. h^{gt} is the height of the ground truth bounding box, h is the height of the prediction bounding box.

5.3 Experiment and results

In this chapter, to achieve the goal of inspecting rail track components in a real-time speed on a single GPU with high accuracy and efficiency, four different types of activation functions are implemented in the backbone of YOLOv4 for training and testing with a customized image dataset. YOLOv3 is trained as the control group to evaluate the prediction performance of the original and modified YOLOv4 models. The four different types of activation functions implemented in the backbone of YOLOv4 are Mish, Swish, Leaky-ReLU, and a combination of Swish and Mish. Accordingly, the models in this study are named as YOLOv4 (the original model which uses Mish), YOLOv4-swish (the modified model which uses Swish), YOLOv4-leaky (the modified model which uses Leaky-ReLU), YOLOv4-hybrid (the modified model which uses a combination of Swish and Mish), and YOLOv3. Note, the combination of Swish and Mish functions means the Mish activation function in the first two residual blocks (see Figure 5.2) in YOLOv4 are replaced by Swish activation function. The reason for making this modification is to take the advantages of both YOLOv4-swish and YOLOv4, which are expected to have a high F1 value and high mAP value, respectively. To evaluate and compare the predicted

results on the same scale, all training works are based on a GitHub repo developed by Ultralytics LLC (Glenn Jocher, 2020). Even though the training process and hyperparameters may vary a little, the prediction results are evaluated on the same metric as discussed in following sections.

5.3.1 Data preparation

As mentioned earlier, a public track component image dataset was built by the authors for the training and validation (Guo, Qian, Wu, et al., 2021). There are three object classes in this dataset, which are rail, clip, and spike, respectively. The images are saved from video frames recorded by a smartphone. The videos are taken along a railroad section near the campus of the University of South Carolina. The original video resolution is 1920×1080. The video is saved frame by frame and the size of the converted images is 512×512. To avoid overfitting in training, image augmentations including flip, contrast, rotation, shift, and Gaussian noise are conducted on this dataset.

To train the YOLO family models, the image data needs to be manually labeled first. The annotated images not only serve as the ground truth but also facilitate evaluating the training accuracy by comparing them with predicted results generated by the trained models. A total of 1,000 images are labeled using the popular annotation tool, labelme (Ketaro Wada, 2016). An example of the labeling process is presented in Figure 5.4. The dataset is randomly separated into two groups, one with 800 images is used for training, the other one with 200 images is used for validation. The testing set is the same as the validation set. The output of labelme is in JSON format, while the labels for the training of YOLOv3 and v4 need to be in txt format. Therefore, a conversion is

performed before the training. After that, the txt format labels are stored separately with the image files.

Figure 5.4. Labeling process in the labelme.

5.3.2 Training and validation

When the labeling process of the image data is completed, all JSON files generated from labelme are converted into txt files which contains the object class ID and normalized ground truth box coordinates. In this study, all the models are trained from the scratch since few studies are using different activation functions in YOLOv4 and no pre-trained weights are available so far. Specifically, the training processes in this study are completed on a workstation with four NVIDIA 2080 Ti GPUs. Note that only a single GPU is used to train each individual model. The operating system is Ubuntu 18.04, and the NVIDIA driver version is 440.64. To accelerate the training process and leverage the advantages of the powerful parallel computing capability of NVIDIA graphics cards, the packages of CUDA (version 10.2) and cuDNN (version 7.6.5) are applied. The training

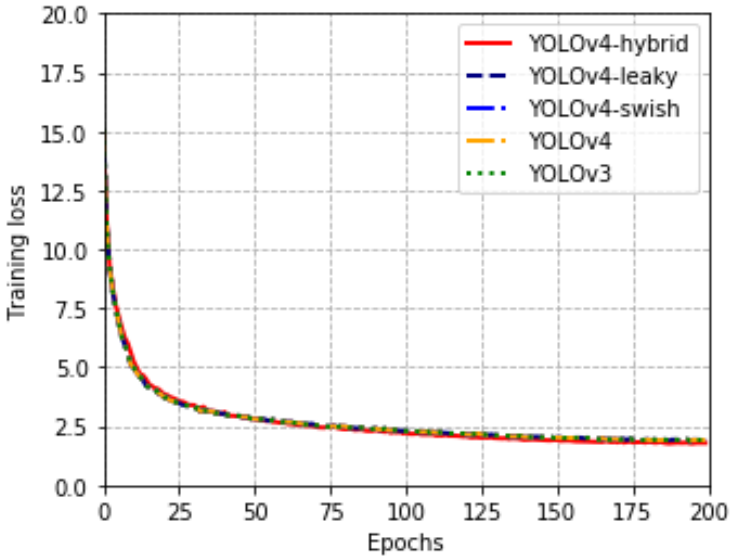
framework is based on PyTorch library which is published by Facebook AI. In this study, the PyTorch version is 1.5.0 and python version is 3.7. The hyperparameters for all the models are summarized in Table 5.1. It needs to mention that the YOLOv4 and YOLOv3 have slightly different optimized parameter settings.

To compare the performance of different activation functions on the customized dataset, the optimized parameters are adopted in this study as shown in Table 5.1. To compare the influence of different activation functions, all parameters are set to be the same for different YOLOv4 models. Specifically, the input size is the height and width of the training images. Momentum is a parameter for improving training accuracy and speed. Decay is used to prevent overfitting by causing the weight to exponentially decay to zero. The learning rate is to control the training speed. Batch size is the number of samples for training in each iteration. The training epoch refers to one cycle through the training dataset.

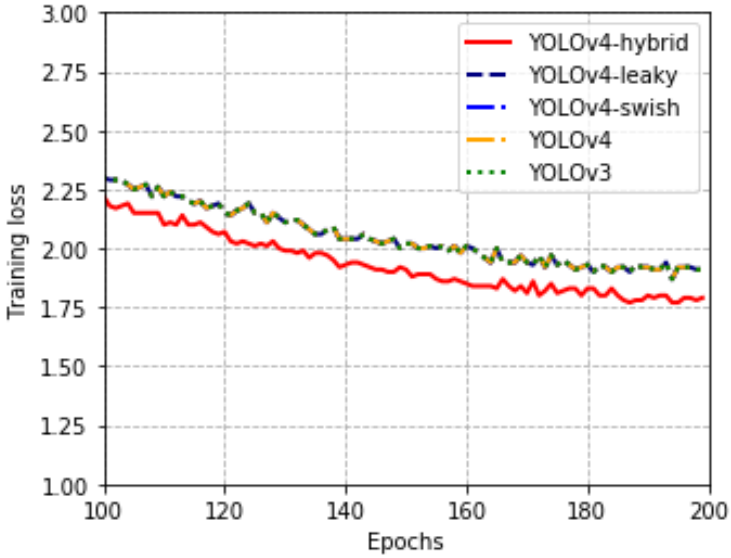
Table 5.1. The hyperparameters of training models.

Model	Input size	Learning rate	Decay	Momentum	Epochs	Batch size
YOLOv4-hybrid	512×512	0.00261	0.0005	0.949	200	8
YOLOv4-leaky	512×512	0.00261	0.0005	0.949	200	8
YOLOv4-swish	512×512	0.00261	0.0005	0.949	200	8
YOLOv4	512×512	0.00261	0.0005	0.949	200	8
YOLOv3	512×512	0.001	0.0005	0.9	200	8

Figure 5.5 and Figure 5.6 present the training loss and validation loss of different models, respectively. In Figure 5.5(a), the training losses of different models are close to each other. To better show the differences, the training loss from epoch 100 to epoch 200 is selected to have detailed comparisons. Figure 5.5(b) shows that YOLOv4-hybrid has the lowest training loss among the five models. Typically, a lower loss indicates a better training result. Also, it is interesting to find that there is small variance between YOLOv4 and YOLOv3 on this customized dataset, indicating a similar prediction performance of them. Figure 5.6 shows an obvious difference of validation loss, which is around 0.2 between YOLOv4-hybrid and other models. Since the validation set is also defined as the testing set, the validation loss could be more reliable compared to training loss on predicting the performance of different models. So according to Figure 5.6, it can be roughly concluded that YOLOv4-hybrid performs better than other models.



(a)



(b)

Figure 5.5. Training loss of different models.

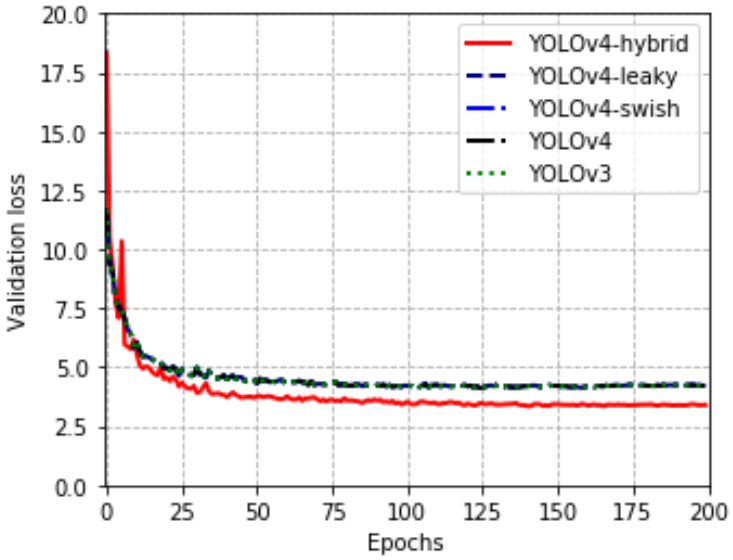


Figure 5.6. Validation loss of different models.

5.3.3 Evaluation metrics

In order to measure the prediction performance of different models, the precision-recall (PR) curve including precision and recall, mAP, and F1 score are used as the evaluation metrics. Precision and recall are two important factors in the drawing of the PR curve and the calculation of mAP. Precision refers to the percentage of prediction results that are relevant instances. Recall refers to the percentage of total relevant results that are correctly classified by the trained model. F1 score considers both precision and recall and conveys the balance between precision and recall. A well-trained model should have a high F1 score. Generally, a high precision value is associated with a low false negative value, and a high recall value is associated with a low false negative value. The PR curve describes the trade-off between the precision and recall. The larger area under the PR curve represents a higher AP. In other words, the trained model can perform well on the prediction if AP is high. The definition of precision, recall, and F1 score are given by Equation (5-9) to (5-11).

The mAP is a common parameter for accuracy evaluation on different object detection models. Specifically, it is the average of average precision (AP) for each object class. The AP is an area under a PR curve over the IoU which measures the overlap between ground truth and prediction. The definition of IoU can be referred to Figure 5.7. The PR curves of testing models are shown in Figure 5.8. Equation (5-12) shows the calculation on AP. Typically, when IoU is equal to or larger than 0.5, the prediction is recognized as correct. In this study, the threshold of IoU is 0.5. After the AP is computed, the mAP can be calculated according to Equation (5-13).

$$precision = \frac{TP}{TP + FP} \tag{5-9}$$

$$Recall = \frac{TP}{TP + FN} \tag{5-10}$$

$$F_1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{5-11}$$

$$AP = \int_0^1 p(r)dr \tag{5-12}$$

$$mAP = \frac{1}{N} \sum AP_i \tag{5-13}$$

where TP is true positive which is an outcome that the trained model can correctly predict the positive class, FP is false positive which is an outcome that the trained model falsely predicts the positive class, FN is false negative which is an outcome that the trained model falsely predicts the negative class. N is number of object classes.

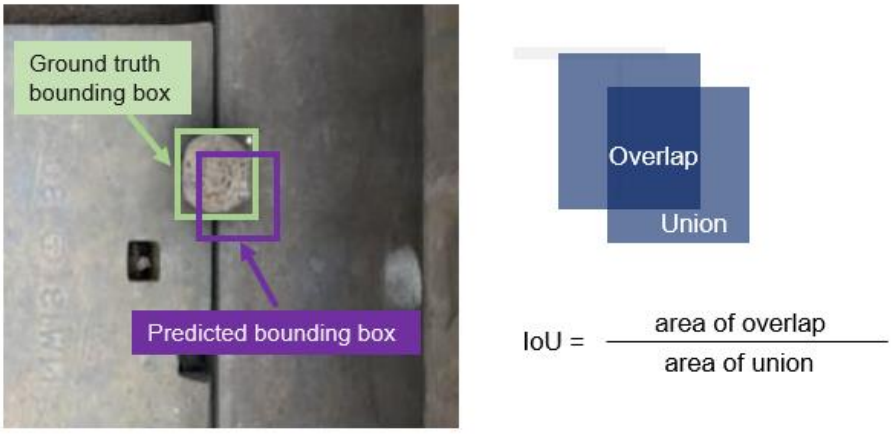


Figure 5.7. Definition of IoU.

Figure 5.8 shows the PR curves of the different models. In each plot, there are three object classes which are rail, clip, and spike. Both the precision and recall range from 0 to 1. Generally, with the increase of the recall, the precision decreases. Under each PR curve, the area means the AP of a specific class with an IoU threshold. In this study, the IoU threshold is set to 0.5 and the area can be referred to the column of mAP @ 0.5 in Table 5.2.

In Table 5.2, there are five parameters which are precision, recall, mAP @ 0.5, F1 score, and inference time to assess the prediction performance of the five models. In the column of precision, the hybrid model scores the highest precision value within all objects which are 1.6% and 1.5% higher than the value of YOLOv4 and YOLOv3, respectively. Besides, the hybrid model has the highest precision in the rail and clip, which are 6.7% and 0.4% higher than YOLOv4 and 10.8% and 1.5% higher than YOLOv3, respectively. YOLOv3 has the highest precision in the spike, which is over 90%.

In the column of recall, the hybrid model hits the highest recall values within the overall class, rail, and spike, which are 96.0%, 100.0%, and 89.4%, respectively. Specifically, the corresponding recall values of the hybrid model are 1.1%, 0, and 2.6% higher than YOLOv4 and 1.0%, 0, and 3.0% higher than YOLOv3. Meanwhile, YOLOv3 reaches the highest recall value, 99.0%, for the clip.

Regarding the column of mAP @ 0.5, YOLOv4-hybrid obtains the highest values among all objects and the spike, which are 1.3% and 0.7% higher compared to YOLOv4, and 1.8% and 2.1% higher than YOLOv3. YOLOv4-leaky and YOLOv4-swish score the highest mAP values in the clip and rail, respectively. It can be found that the Leaky-ReLU or Swish activation functions can help improve the mAP value but there is a limited effect.

Focusing on the column of the F1 score, the hybrid model still outperforms other models. It gains the highest F1 values among all objects, rail, and clip, which are 89.6%, 87.7%, and 95.1%. Correspondingly, they are 1.7%, 4.3%, and 0.5% higher compared to YOLOv4, and 1.9%, 7.2%, and 6.0% higher than YOLOv3. YOLOv3 obtains the highest F1 value for the spike, which is 88.1%. Since the F1 score considers both precision and recall, the F1 score is typically selected for comparing the performance of different models (Liu, Cao, Wang, & Wang, 2019; A. Zhang et al., 2019; A. Zhang et al., 2017). From the result of the F1 score presented in Table 5.2, the hybrid model has the best prediction performance over the rest models.

As regards the inference time which refers to the time using a pre-trained model to make predictions, Table 5.2 shows it varies from 9.6 ms to 13.2 ms. YOLOv3 has the fastest inference time for a single frame, which is 9.6 ms. Compared with YOLOv3, the

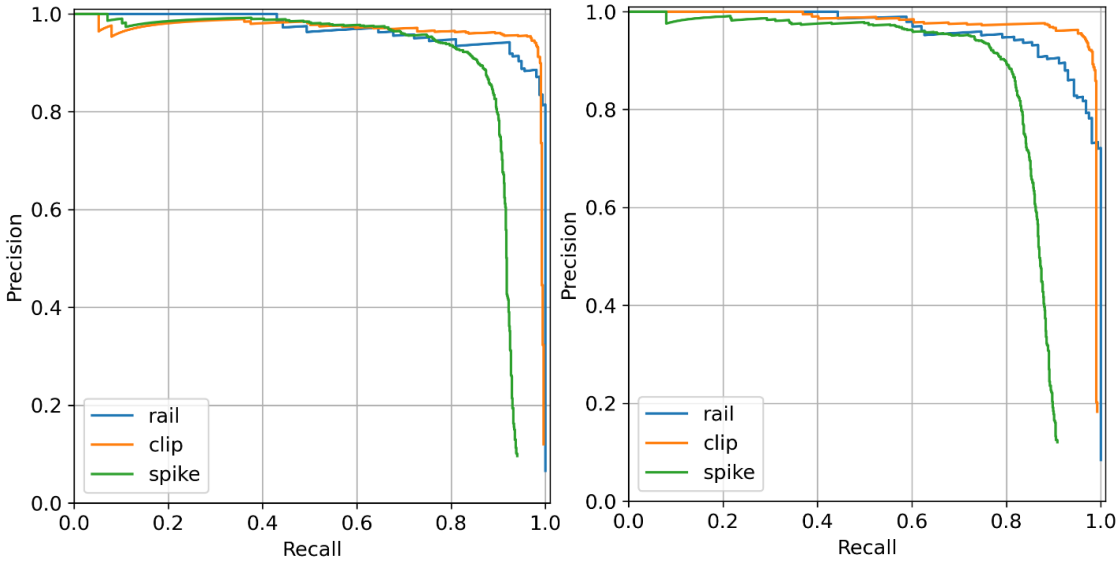
original YOLOv4 and the hybrid model are 3.6 ms and 3.1 ms slower. When the inference time is converted to the frame rate as shown in Table 5.2, all the models are much faster than the requirement of real-time speed which is 30 frames per second (FPS) (Redmon et al., 2016), indicating it may satisfy the inspection car's speed on the railroad, which is generally between 15-20 miles per hour (mph) (Liu, Lovett, Dick, Rapik Saat, & Barkan, 2014). Note for the inspection vehicles with a higher speed, the frame rate needs to be higher to the real-time processing.

One interesting finding is that the overall performance of YOLOv3 and YOLOv4 has a small variance with this customized image dataset. As for the F1 value, YOLOv4 is only 0.002 higher compared to YOLOv3. In terms of precision and recall, the performance of YOLOv4 is even poorer than YOLOv3. This is in contrast to the belief of the original YOLOv4 report which was trained on the popular large image dataset (Bochkovskiy et al., 2020). The possible reason is that the different image datasets may impact the prediction of different models. In this customized dataset, the environment is more challenging than that in the dataset associated with the original YOLOv4 report, such as similar shapes between the ballast and the spike top, and the frequent repetition of the same objects, like spikes and clips. All these factors can impact the performance of YOLOv4, and the users should build a customized dataset and identify the most suitable activation function whenever possible to meet their specific needs.

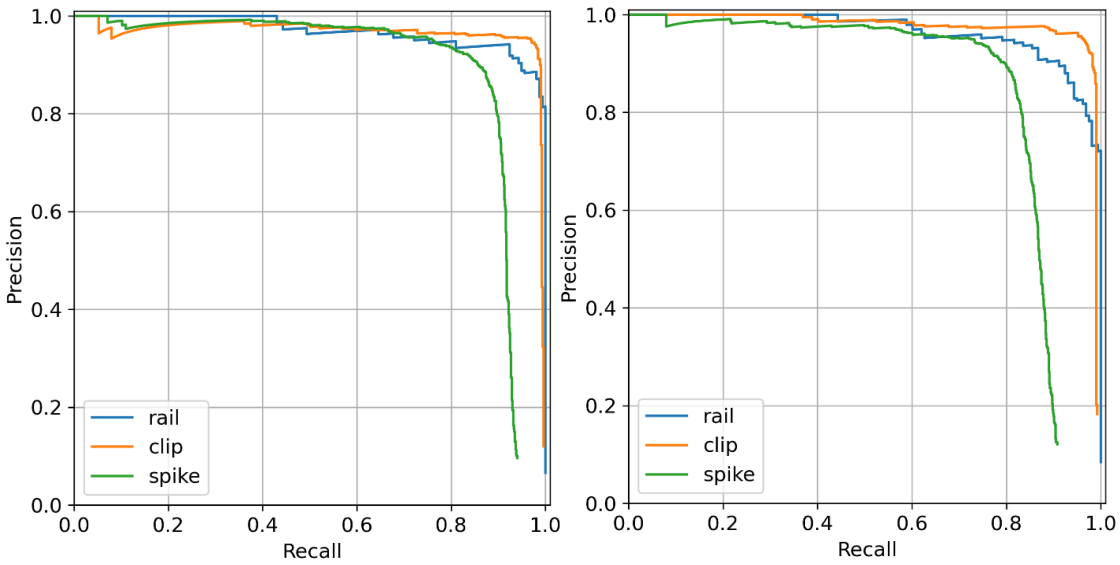
Additionally, the impact of different loss functions (Zheng et al., 2020a) including CIoU, GIoU, and Distance IoU (DIoU) on the prediction performance has been investigated as shown in Table 5.3. It is easy to find that with different loss functions, the inspection performance varies. The bold numbers indicate better performance with each

parameter (i.e., Precision, Recall etc.). With the same loss function, the improved model, YOLOv4-hybrid, performs better either from the accuracy indicator or the processing speed indicator.

Regarding the same model, YOLOv4 or YOLOv4-hybrid, the loss function does impact the performance but compared to the proposed activation function structure, they present very a limited impact on the detection accuracy. For example, with respect to the F1 scores of YOLOv4-hybrid (DIoU), YOLOv4(DIoU), and YOLOv4(GIoU), which are 87.9%, 83.4%, and 82.1% respectively. Taking YOLOv4(GIoU) as the baseline, the proposed model can improve the performance by 4.5% but the loss function only varies the performance by 1.3%. Both activation and loss functions could impact the model performance, but the impact from the activation functions is significantly more pronounced. This suggests that the enhanced performance of the proposed model is the benefit of using the hybrid activation function, rather than the selection of a loss function.

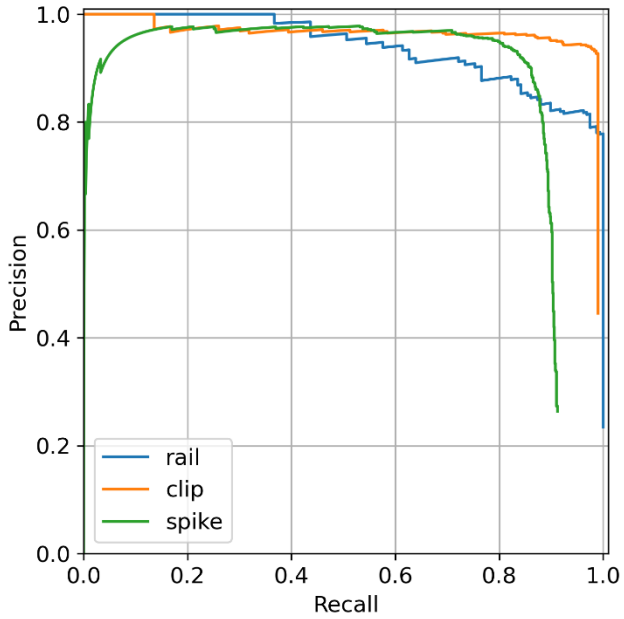


(a) (b)



(c)

(d)



(e)

Figure 5.8. Precision-recall curves of testing models. (a) YOLOv4-hybrid (b) YOLOv4-leaky (c) YOLOv4-swish (d) YOLOv4 (e) YOLOv3.

Table 5.2. Performance indicators.

Model	Class	Precision (%)	Recall (%)	mAP@0.5 (%)	F1 (%)	Inference time (ms)
YOLOv4-hybrid	All	***84.2***	***96.0***	***94.4***	***89.6***	12.7 (78.7 FPS)
	Rail	***78.2***	***100.0***	96.7	***87.7***	
	Clip	***91.9***	98.4	97.3	***95.1***	
	Spike	82.6	***89.4***	***89.3***	85.9	
YOLOv4-leaky	All	82.7	93.4	92.9	87.1	10.7 (93.5 FPS)
	Rail	70.5	100.0	96.3	82.7	
	Clip	91.3	98.3	***98.0***	94.7	
	Spike	86.3	81.8	84.3	84.0	
YOLOv4-swish	All	76.5	94.2	93.2	83.7	12.1 (82.6 FPS)
	Rail	60.7	100.0	***97.1***	75.6	
	Clip	91.3	97.1	96.6	94.1	
	Spike	77.6	85.6	86.0	81.4	
YOLOv4	All	82.6	94.9	93.1	87.9	13.2 (75.6 FPS)
	Rail	71.5	100.0	93.1	83.4	
	Clip	91.5	97.9	97.7	94.6	
	Spike	84.8	86.8	88.6	85.8	

YOLOv3	All	82.7	95.0	92.6	87.7	9.6 (104.1 FPS)
	Rail	67.4	100.0	93.8	80.5	
	Clip	90.4	***99.0***	96.7	94.5	
	Spike	***90.2***	86.0	87.2	***88.1***	

Table 5.3. Influence of different loss functions.

Model	Class	Precision (%)	Recall (%)	mAP@0.5 (%)	F1 (%)	Inference time (ms)
YOLOv4-hybrid (CIoU)	All	***84.2***	***96.0***	***94.4***	***89.6***	12.7 (78.7FPS)
	Rail	***78.2***	***100.0***	***96.7***	***87.7***	
	Clip	***91.9***	***98.4***	97.3	***95.1***	
	Spike	82.6	***89.4***	***89.3***	***85.9***	
YOLOv4 (CIoU)	All	82.6	94.9	93.1	87.9	13.2 (75.6FPS)
	Rail	71.5	100.0	93.1	83.4	
	Clip	91.5	97.9	97.7	94.6	
	Spike	***84.8***	86.8	88.6	85.8	

YOLOv4-	All	***82.6***	94.9	***93.5***	***87.9***	
hybrid	Rail	***71.2***	100.0	***95.1***	***83.2***	12.4
(DIoU)	Clip	***91.2***	98.4	97.8	***94.7***	(80.6FPS)
	Spike	***85.4***	86.4	***87.6***	***85.9***	
YOLOv4	All	74.8	***95.5***	93.0	83.4	
(DIoU)	Rail	61.9	100.0	94.7	76.5	13.1
	Clip	88.5	***99.0***	***97.9***	93.5	(76.3FPS)
	Spike	73.9	***87.6***	86.5	80.1	

YOLOv4-	All	***78.0***	***95.7***	94.1	***84.8***	
hybrid	Rail	***56.7***	***100.0***	95.0	***72.4***	12.7
(GIoU)	Clip	***90.4***	***98.1***	96.5	***94.1***	(78.7FPS)
	Spike	***87.0***	***89.1***	***90.7***	***88.0***	
YOLOv4	All	73.2	95.5	***94.2***	82.1	
(GIoU)	Rail	56.7	100.0	***95.7***	72.4	13
	Clip	81.8	97.7	***96.6***	89.0	(76.9FPS)
	Spike	81.1	88.9	90.2	84.8	

5.3.4 Comparison with other state-of-the-art models

In this study, to compare the detection performance of YOLOv4-hybrid with other SOTA models (Zhao et al., 2019), five models including Faster R-CNN, RetinaNet, SSD, Cascade R-CNN, and Mask R-CNN have been trained and tested. Specifically, these tests have been conducted on MMdetection which is a deep learning toolbox integrated with the SOTA object detection models. The default training parameters for each model have been kept. The experiment results with different performance indicators are plotted in Figure 5.9.

As shown in Figure 5.9, it can be easily found that regardless of the performance indicator, Faster R-CNN and RetinaNet perform the worst on the proposed railroad components dataset. The precision values of Faster R-CNN and RetinaNet are only 13% and 17.3%, respectively. With respect to recall values, they are merely 21.2% and 22.7%, respectively. These values are significantly lower than the corresponding values of YOLOv4-hybrid. As for the indicator values of SSD and Cascade R-CNN, it can be found that both of these two models have high mAP@0.5 values, which are 97.6% and 98.5%, respectively. They are 3.2% and 4.1% higher than the mAP@0.5 value of YOLOv4-hybrid. However, regarding the rest indicators which are quite behind YOLOv4-hybrid. For example, the precision values of SSD and Cascade R-CNN are 61.9% and 71.5%, which are 22.3% and 12.7% lower than the precision value of YOLOv4-hybrid. Besides, Cascade R-CNN has the lowest detection speed among all the models, which is 27.7 FPS, indicating its limited potential for field practice.

With respect to F1 score, Mask R-CNN is the only one close to 80%, which is 75.5%, indicating it has a relatively balanced performance compared to other four models However, it is still 14.1% lower than YOLOv4-hybrid. In addition, its inspection speed is also not promising, which is only 5.2 FPS, 73.5 FPS lower than YOLOv4-hybrid. To put it into a nutshell, compared to other SOTA models, the proposed model has a well-balanced detection performance on either detection accuracy or speed, indicating a possible adequate solution for the future automatic inspection of railroad components.

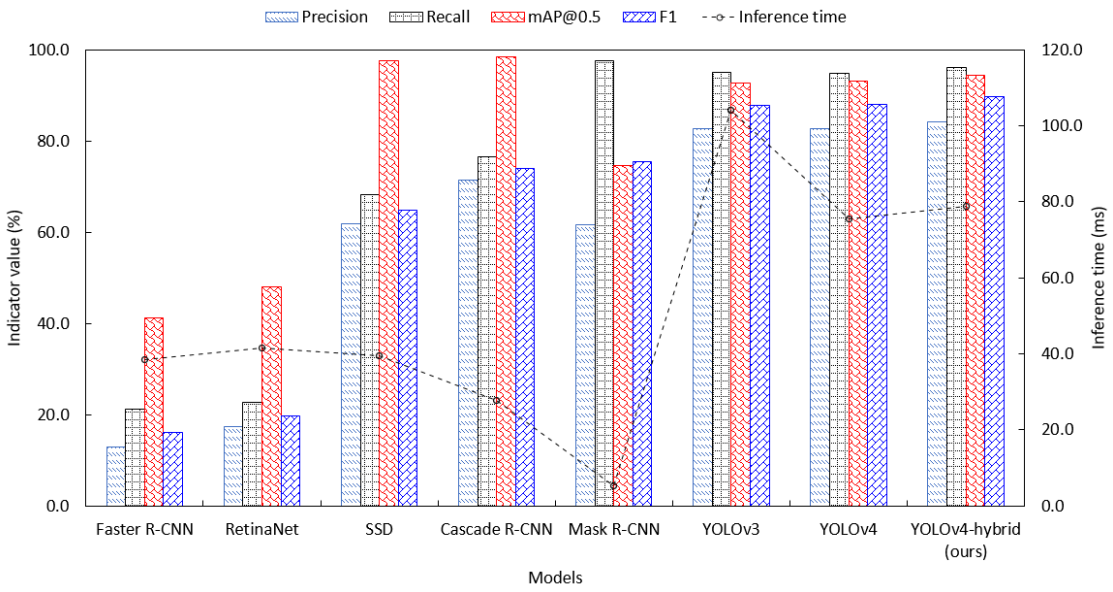


Figure 5.9. Performance comparison between YOLOv4-hybrid and other SOTA models.

The recall is an important indicator of the detection performance, which can return and present the ratio of the corrected components that can be detected over all the ground truth instances. To illustrate the detection effect with different recall values, Figure 5.10 shows the detection results with the aforementioned models. The first row shows the detection results by using YOLO detectors. The improved YOLO-hybrid model has detected all the components but the YOLOv3 and YOLOv4 missed two and

one spike, respectively. The second row presents three models (Mask R-CNN, Cascade R-CNN, and SSD) with high recall values and all of them missed one or two components, indicating similar detection results as YOLOv3 and YOLOv4. But, as shown in Figure 5.9, these three models generally have low precision values which means they may cause more false positive instances in the practice with more testing cases. Besides, regarding the detection speed, they are much slower compared to YOLO detectors. The third row gives the detection results using Faster R-CNN and RetinaNet. Both of them cannot work well in the railroad component inspection. Specifically, Faster R-CNN has missed six components while RetinaNet has missed five components. Overall, a higher recall value can return better detection results and it will benefit the engineering practice.

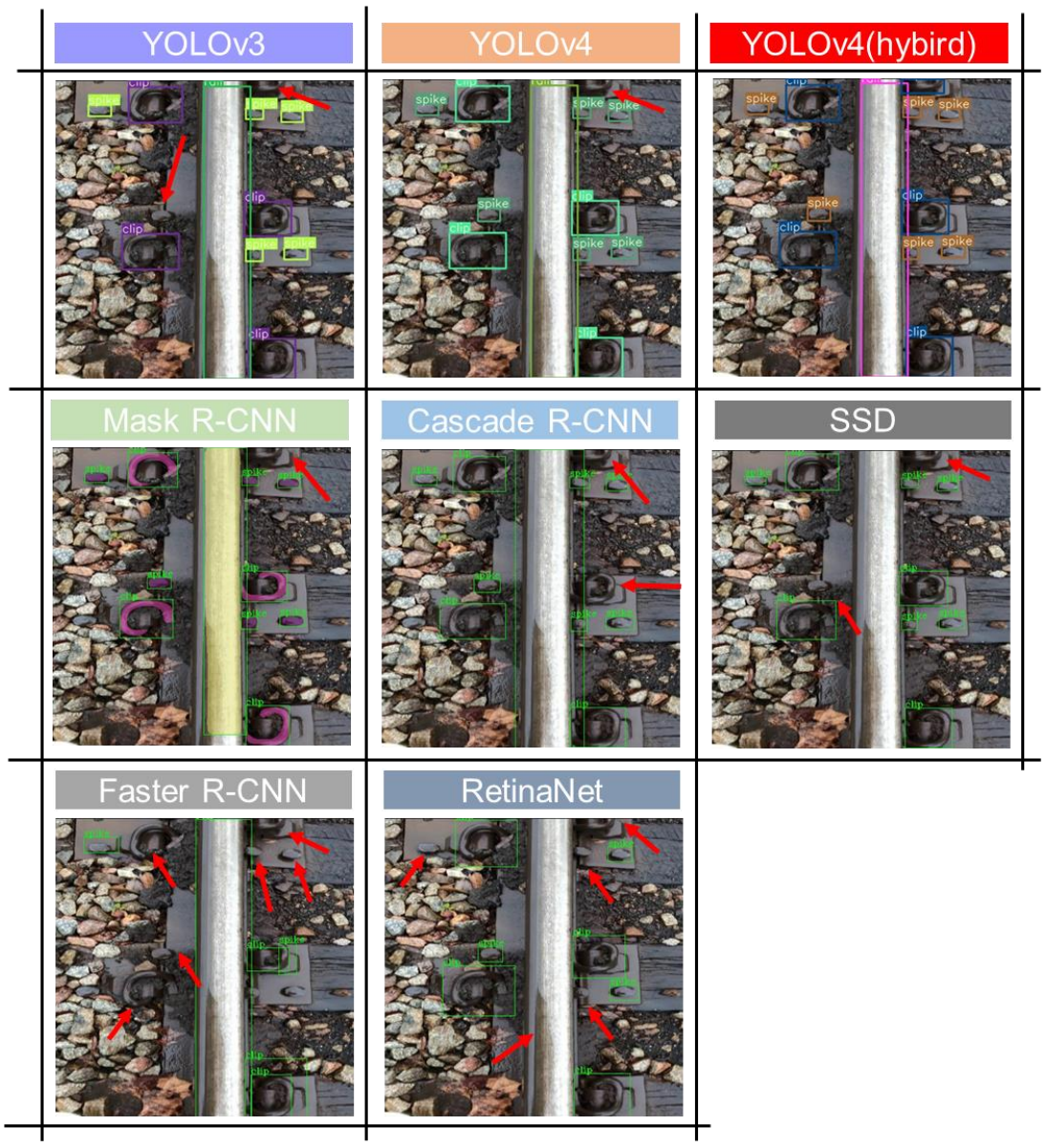


Figure 5.10. Prediction results on different models with the high and low recall values (red arrows point out the missed detection).

It needs to mention that typically, as shown in Figure 5.10, the segmentation-based approach (i.e., Mask R-CNN) can generate the mask on the instance to present its shape. The advantages are they can provide more information related to the instance and meanwhile the models have a good inspection performance. But the high computation cost may limit the potential for the field application with a mobile computing platform. Indeed, for the specific tasks for the railroad (i.e., finding missed rail components), the object detection method with bounding boxes on instances such as our improved YOLOv4, has the superior prediction performance as shown in Figure 5.9 and Figure 5.10. With an appropriate configuration, it can be utilized and deployed in the railroad practice with its fast speed and high accuracy.

5.3.5 Impacts of different image sizes and illumination conditions on prediction performance

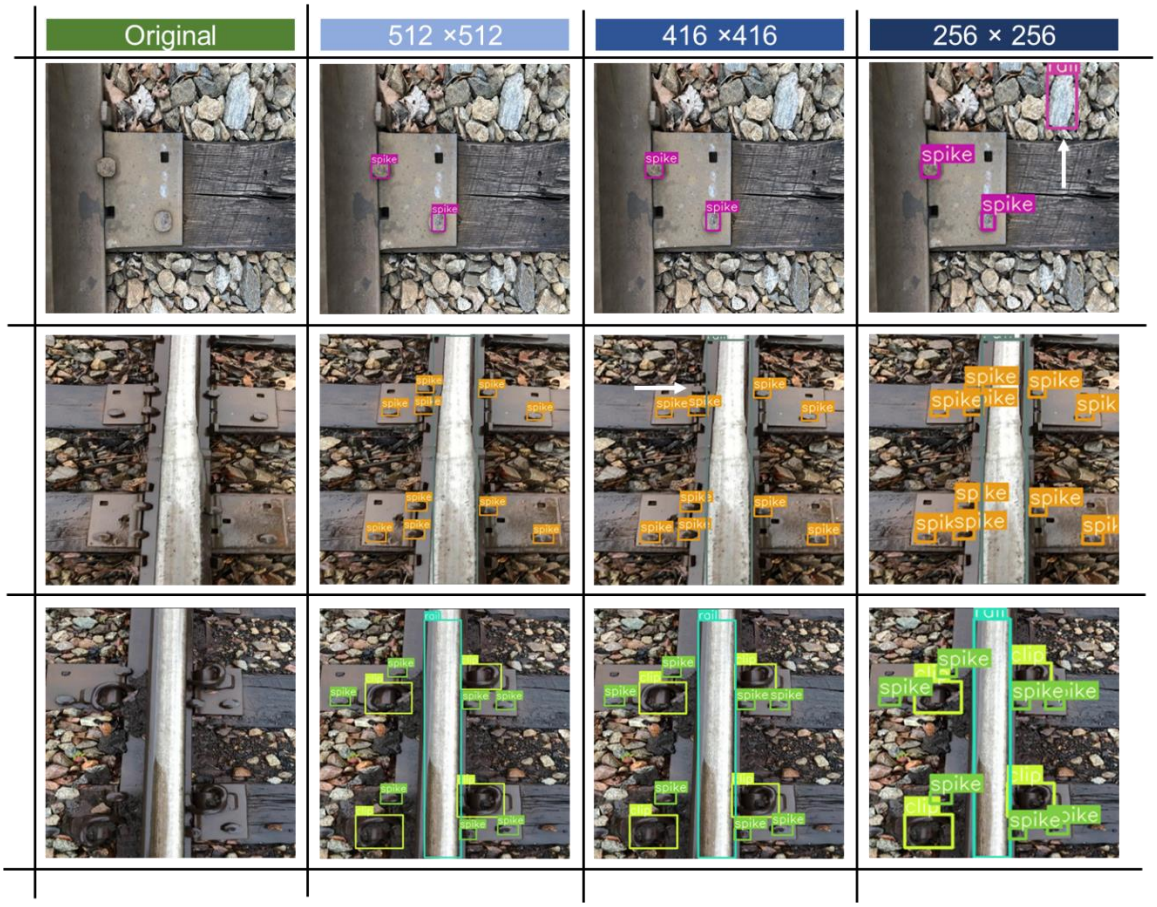
To evaluate the impact of different image sizes and illumination conditions on the detection performance of the proposed YOLOv4-hybrid and test its performance with simulated “missing” and “fake” components, three different image sizes, three different illumination conditions, and three “special cases” have been investigated in this study. The selected three images sizes are 512 × 512, 416 × 416, and 256 × 256, respectively. After the preprocessing, the original brightness is reduced to -30%, -50%, and -70%, respectively, to mimic the different visibility conditions. To validate the capability of the proposed model in inspecting potential “missing” track components in the field and test the robustness of the model with “fake” railroad track components, special cases of “fake

ballast", "fake spike", and "disappeared clip" are created to challenge the proposed YOLOv4-hybrid. The detailed results are shown as follows.

Figure 5.11(a) shows the detection performance with different image sizes. To better reflect the real field scenarios, the original images include three different railroad track conditions which are "spike only", "spike and rail", and "spike, rail, and clip", respectively. The reason for mixing different track conditions is to demonstrate the complexity of the track and different densities of the track components in the field. It should be mentioned that the general "walking inspection" practices in the field not only is low-efficient and labor-intensive but also has low-accuracy due to the complicated environment and countless railroad track components could lead to eye fatigue. With different image sizes, Figure 5.11(a) shows the inspection with the proposed YOLOv4-hybrid is promising and most of the track components can be successfully detected. Under the "spike only" condition, the detections are correct on the image sizes of 512 × 512 and 416 × 416. However, when the image size is reduced to 256 × 256, there is a false detection for the rail. The model classifies a large size ballast as rail. Regarding the "spike and rail" condition, there is only one missing detection for the spike with the image size of 416 × 416. In the "spike, rail, and clip" condition, all detections are correct. Overall, the image size is not significant in the prediction performance.

Figure 5.11(b) presents the detection results under different illumination conditions. Three preprocessed illumination conditions are included, and all the false detections are pointed out by yellow arrows. In the "spike only" condition, there are false detections under "light -50%" and "light -70%". In the "light -50%" condition, there is a redundant spike detection. In the "light -70%" condition, the detection misses a spike.

Further, in the “spike and rail” condition, there are two missing detections when the illumination is down to “light -70%”. Specifically, it misses both the spike and rail. With respect to the condition of “spike, rail, and clip”, when the light condition is reduced to “light -70”, the detection misses two spikes and incorrectly takes one clip as a spike. Overall, it can be found that the inspection is more sensitive to the variation of visibility rather than image size. Under the very dimming condition, the missing and false inspection can be occurred, which is unfavorable for field practice.



(a)

(b)

Figure 5.11. The impacts of image size and illumination on the prediction performance (a) prediction performance on the images with different sizes (b) prediction performance on the images under different illumination conditions.

To simulate the missing track components scenario and test the robustness of YOLOv4-hybrid, images edited by Photoshop are fed into the model to check the performance. Figure 5.12 shows the edited cases of “fake ballast”, “fake spike”, and “missing clip” with the stamp tool in Photoshop. According to Figure 5.12, even there are “missing parts” and “fake parts”, all detection results are correct. In the “spike only” condition, a spike is replaced by a “fake ballast” which has a very similar shape of a spike head, but the model correctly rejects it as a spike. In the “spike and rail” condition, two spikes are manually “copied” and added into the revised image and the newly added

spikes are correctly recognized by the model. Moreover, in the “spike, rail, and clip” condition, no redundant label is assigned to the area of the “disappeared clip” and the rest of the detection results are all correct. In summary, the proposed YOLOv4-hybrid model can accurately detect the railroad components before and after certain changes on the railroad track such as missing or fake components.

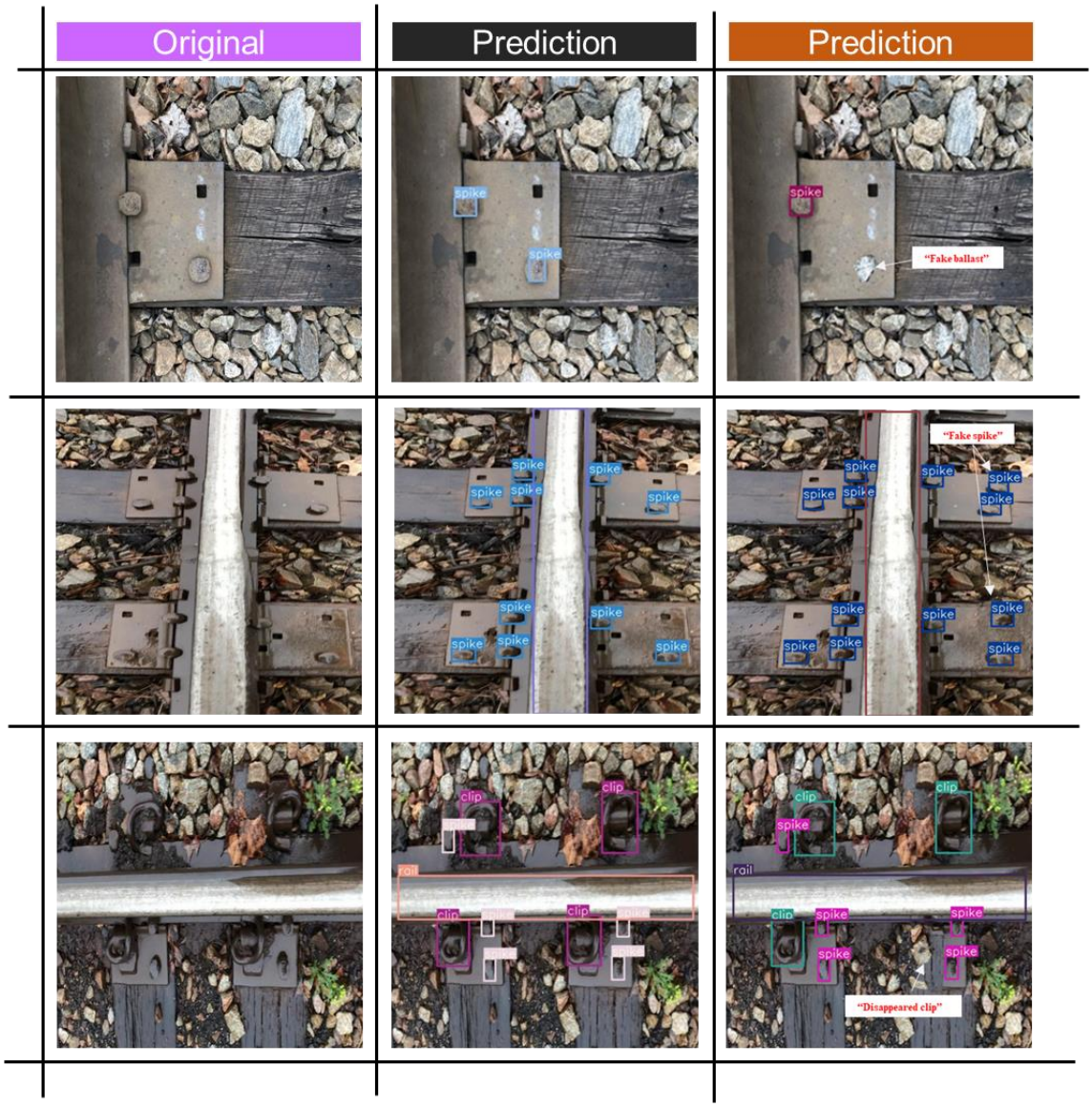


Figure 5.12. Prediction performance on the “fake” railroad track components.

5.4 Summary

In this chapter, a real-time railroad track components inspection framework is proposed, aiming to assist the railroad industry to save inspection cost, improve inspection efficiency and accuracy, and prompt railroad safety. The proposed framework is based on a newly released one stage object detection model YOLOv4. The original Mish activation functions in the four residual blocks of the backbone are replaced with different activation functions, including Swish, Leaky-ReLU, and a combination of Mish and Swish. A total of 1,000 images including rail, spike, and clip are utilized for training and validation. The modified models and the original YOLOv3 and YOLOv4 models are trained and evaluated based on PyTorch library, PR curve, F1 score, mAP, and inference time. Experimental results indicate that the hybrid model, YOLOv4-hybrid, which adopts a hybrid activation function outperforms the other models on precision, recall, mAP, and F1 score, showing potential better inspection performance in the field practice.

The influence of different image sizes and illumination conditions on the prediction performance is investigated. Test results depict that the developed YOLOv4-hybrid is more sensitive to the illumination conditions rather than image size. Under very dimming conditions, the model could miss railroad track components and false alarm could occur. To verify robustness of the proposed YOLOv4-hybrid in “missing and fake component” scenarios, edited images are used to evaluate the performance of the model. Experimental results indicate the developed YOLOv4-hybrid can accurately distinguish the changes before and after image modification.

CHAPTER 6 AUTOMATIC RAIL SURFACE DEFECTS INSPECTION BASED ON MASK R-CNN[6]

[6] Guo, Feng, Yu Qian, Dimitris Rizos, Zhi Suo, and Xiaobin Chen. Automatic Rail Surface Defects Inspection Based on Mask R-CNN. *Transportation Research Record* (2021): 03611981211019034..

Rail surface defects have negative impacts on the riding comfort and track safety, and could even lead to accidents. Based on the safety database (2020) of FRA, rail surface defects have been among the main factors causing derailments. During the past decades, there have been many efforts to detect such rail surface defects. However, the applications of the earlier methods are limited by the high requirements of specialized equipment and personnel training. To date, rail surface defect inspection is still a very labor-intensive and time-consuming process, which hardly satisfies the field maintenance expectations. Therefore, a cost-effective and user-friendly automatic system that can inspect the rail surface defects with high accuracy is in urgent need. To address this issue, this study proposes a computer vision-based instance segmentation framework for the rail surface defect inspection. A rail surface database having 1,040 images (260 source images & 780 augmented images) has been built. The classic instance segmentation model, Mask R-CNN has been re-trained for inspecting rail surface defects with the customized dataset. The influences of different backbones and learning rates are investigated and discussed. The experimental results indicate the ResNet101 backbone has the best inspection capability. With a learning rate of 0.005, the re-trained Mask R-CNN model can achieve the best performance on the bounding box and mask predictions. Sixteen images are used to test the inspection performance of the developed model. The results show the proposed approach in this study is promising. The proposed approach has potential for future field applications.

6.1 Introduction

The health condition of the railroad infrastructure plays an important role in train operation and track safety. Reducing the accident risks due to unfavorable infrastructure conditions is of great interest to the public, railroads, and government. According to the FRA report (FRA, 2020a) and prior studies (Liu et al., 2011; Liu et al., 2018), rail defects has been one of the main factors causing serious train accidents, as well as one of the major causes responsible for over 70% train derailments on freight mainlines (Liu et al., 2018). On one hand, rail surface defects can directly affect the riding comfort and the normal train operations due to the additional excitations caused by the surface irregularities. On the other hand, it introduces potential risks for the rail breakage that could result in catastrophic track failures. However, the current inspection approaches often cannot provide very reliable results (FRA, 2020c) and the inspection equipment is expensive and difficult to operate. Thus, considerable amount of the inspection work heavily relies on visual inspections, which are labor-intensive, inefficient, and subjective. With the freight shipment increases, current track inspection speed and accuracy cannot satisfy the rapidly growing track inspection demand. Therefore, a cost-effective, highly robust, reliable, and accurate inspection system is in urgent need.

Over the past decades, many efforts have focused on the inspection of rail surface defects and most of these works rely on rail texture classification, identification, and analysis. Mandriota et al. (2004) utilized and compared three feature extractors, i.e., Gabor, wavelet, and Gabor wavelet filters, to extract rail defect textures. Their results showed that the Gabor filter outperforms the other two filters on the rail defects identification. However, one of the main problems of their study is that these three filters

require a large number of feature images which are typically difficult to obtain. Jie et al. (2009) proposed a rail head surface defect detection framework which included image pre-processing, defect locating, defect identifying, post-processing, and a geometrical defect locating method. Test results suggest that their defect locating method is more effective and robust compared to the pixel-level feature extraction method, and the framework has high precision and could meet real-time inspection requirements of the field applications.

However, noise in the images has significant negative effects in the defect detection, while the proposed way to remove image noise requires extensive validation and needs to be investigated further. Li et al. (Li & Ren, 2012b) proposed an intelligent vision detection system (VDS) for inspecting rail surface defects and addressed two issues of improving image quality and automatic thresholding by using the local Michelson-like contrast (MLC) and the proportion emphasized maximum entropy (PEME) thresholding algorithm. The results show that high recall values on Type I and Type II defects are achieved.

Although the proposed approach can effectively address the issue of inspecting rail surface defects, the problem of distinguishing the rusted rail area from the area of the real defect needs to be improved. Li et al. (Li & Ren, 2012a) developed a real-time visual inspection system (VIS) for the detection of discrete rail surface defects. VIS aims to solve four difficult problems related to: *a*) limited features for recognition; *b*) illumination inequality; *c*) variation of rail surface reflection; and *d*) the requirements on high-speed detection. The proposed local normalization (LN) method and the defect localization based on the projection profile (DLBP) algorithm are used to address these

four challenges. The results show that VIS can have a recall rate of 93.1% and 80.41% on Type I and Type II defects, respectively. They also mentioned the detection speed can be improved, and the LN method needs to be more robust for field applications.

It is worth noting that the above approaches are focusing on traditional hand-crafted feature learning to identify and classify rail surface defects, which requires technicians to have rich experience in feature selection, training parameter adjustment, and a large amount of training data. Compared to those methods, deep learning approaches are more flexible and can automatically extract and learn problem-specific features from the original data without subjectively defining any hand-crafted features. Leveraging the fast development of convolutional neural networks (CNN), many recent developments have been successfully applied to civil engineering, such as pavement crack detection (Yang et al., 2019; Zhang et al., 2017; Zhang et al., 2018), concrete crack detection (Dung, 2019; Kim & Cho, 2018; Xinxiang Zhang, Dinesh Rajan, et al., 2019), and structural health monitoring (Azimi & Pekcan, 2020; Bao et al., 2019; Kang & Cha, 2018), etc. However, few studies used deep learning models to identify and characterize rail surface defects. Faghih-Roohi et al. (Faghih-Roohi et al., 2016) proposed to use deep convolutional neural networks (DCNN) to learn features of rail surface defects. Three neural network structures, including small, medium, and large DCNN, are trained with two kinds of activation functions, i.e., Tanh and ReLU. Their experimental results indicate that DCNN can detect rail surface cracks with high accuracy, while the large DCNN with ReLU performs better than other models. Also, they mentioned that the larger DCNN model required longer training time. Song et al. (Yanan, Hui, Li, & Hang, 2018) proposed to use YOLOv3 to detect rail surface flaws. YOLOv3 is a one-stage

detector with real-time speed for detection. It only detects the target objects with bounding boxes while it does not characterize the defect shape nor quantifies the sizes of the detected rail surface defects.

The current challenges of identifying rail surface defects are: 1) The rail surface defects are not of any regular shape, 2) The features are difficult to extract with a hand-crafted feature design, and 3) The inspection work does not produce reliable results. To address the above issues, this chapter proposes to train the computer vision-based instance segmentation model, Mask R-CNN (He, Gkioxari, Dollár, & Girshick, 2017), for the identification of the rail surface defects. The contributions of this chapter are: (i) The development of a customized rail surface defect image database which includes 260 source images and 780 augmented images for deep learning model training, evaluation, and test; (ii) Fine-tuning the Mask R-CNN model with two backbones (ResNet50 & ResNet101) and three learning rates(0.02, 0.01, and 0.005) for better inspection and identification performance on rail surface defect; (iii) The inspection performance evaluation of Mask R-CNN model on rail surface defects with different severity levels and different orientations of rails; (iv) The performance comparison between Mask R-CNN and Otsu's method on the rail surface defects inspection; and (v) The impact of different light conditions on the rail surface defects inspection with Mask R-CNN.

6.2 Methodology and Mask R-CNN model

6.2.1 Methodology overview

In this chapter, to accurately inspect the rail surface defects, the instance segmentation model, Mask R-CNN with two different backbones, and three different learning rates are trained and evaluated. Figure 6.1 shows the overall methodology of this study.

This chapter has three tasks: (i) Data preparation, (ii) Model training and evaluation, and (iii) Inspection results on the rail and its surface defects. The data preparation tasks include data collection, image augmentation, and data labelling with the generation of annotation files. The popular labelling tool, labelme (Ketaro Wada, 2016)is used in the labeling process. For the model training and evaluation, two backbones and three learning rates are selected and used in this study, aiming to find the optimal parameters for the inspection. Finally, regarding the inspection performance, the original images containing rail surface defects are used to test and evaluate the robustness of the proposed method.

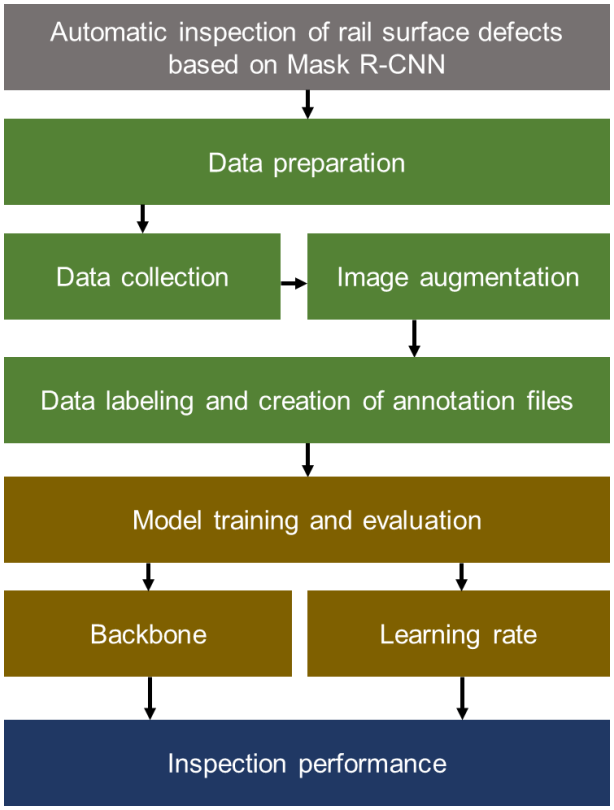


Figure 6.1. The methodology of this study.

6.2.2 Mask-RCNN architecture

Mask R-CNN is an instance segmentation model which is developed by He *et al.* (He et al., 2017). Compared to the original two-stage detector Faster R-CNN (Ren, He, Girshick, & Sun, 2015), it adds a parallel branch for recognizing the mask for each instance. It is worth noting that the semantic segmentation model can distinguish multiple objects of the same class as a single entity, while the target function of Mask R-CNN is more challenging since it segments each instance based on semantic segmentation. Similar to Faster R-CNN, it adopts the two-stage procedure consisting of the Region Proposal Network (RPN) in the first stage and parallelized class, box and mask detection in the second stage. The loss L in Mask R-CNN is defined by the three loss functions of classification loss L_{cls}, bounding box loss L_{box}, and mask loss L_{mask}. The sum of the three

loss functions is the total loss, i.e. $L = L_{cls} + L_{box} + L_{mask}$. Mask R-CNN network architecture, shown in Figure 6.2, includes two parts: (1) the backbone for feature extraction over the input image to generate feature maps; (2) the prediction head for bounding box recognition (including classification and regression) and mask generation. Theoretically, any CNN can be used as the backbone for feature extraction, but Mask R-CNN (He et al., 2017) proves that the feature pyramid network (FPN) (Lin, Dollár, et al., 2017) architecture is the most suitable type that can achieve both speed and accuracy.

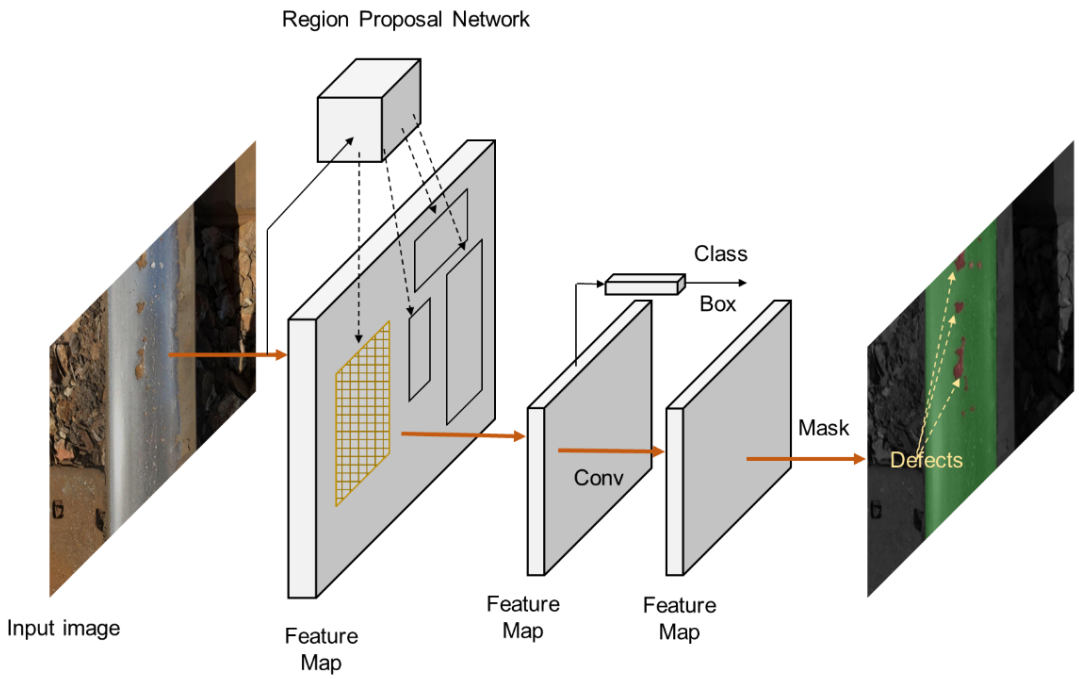


Figure 6.2. The overview of Mask R-CNN architecture.

Some of the existing object detectors (Liu et al., 2016; Redmon et al., 2016) perform worse on small objects than on large objects because lower layers provide accurate location information and less semantic information. As the layers increase, the feature semantic information becomes abundant, but the object location information is not accurate. The proposed backbone structure of Mask R-CNN, depicted in Figure 6.3,

addresses this issue. The Mask R-CNN backbone consists of the ResNet (He et al., 2016) with 50 or 101 layers as the bottom-up pathway, the FPN as the top-down pathway, and lateral connections, and is depicted in Figure 6.3. The bottom-up pathway in ResNet facilitates the feature extraction. Including the FPN structure into the backbone can help maintain strong semantic features at different scales since it constructs higher resolution layers from the top layers. The lateral connections function as bridges connecting the feature maps and the reconstruction layers for better predictions of object locations. Regarding the prediction head, Mask R-CNN extends the box head of the previously developed Faster R-CNN for classification and regression and adds a parallel mask branch for the mask prediction.

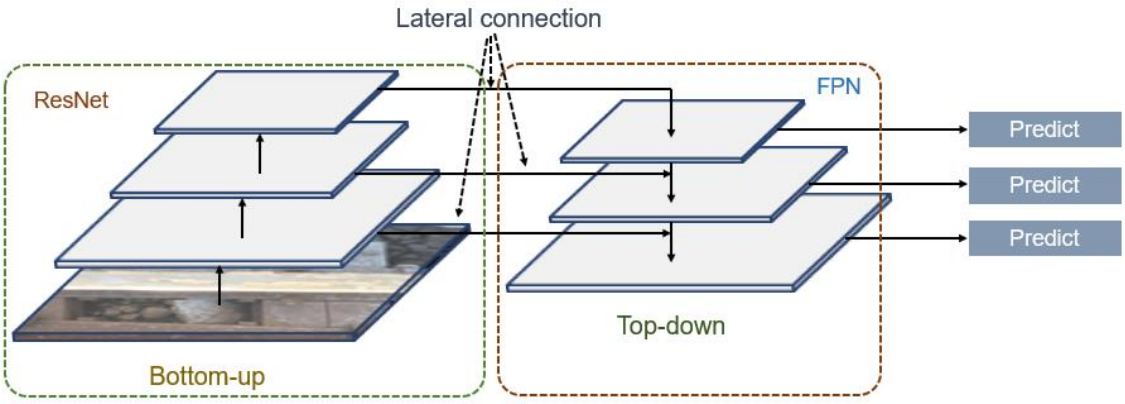
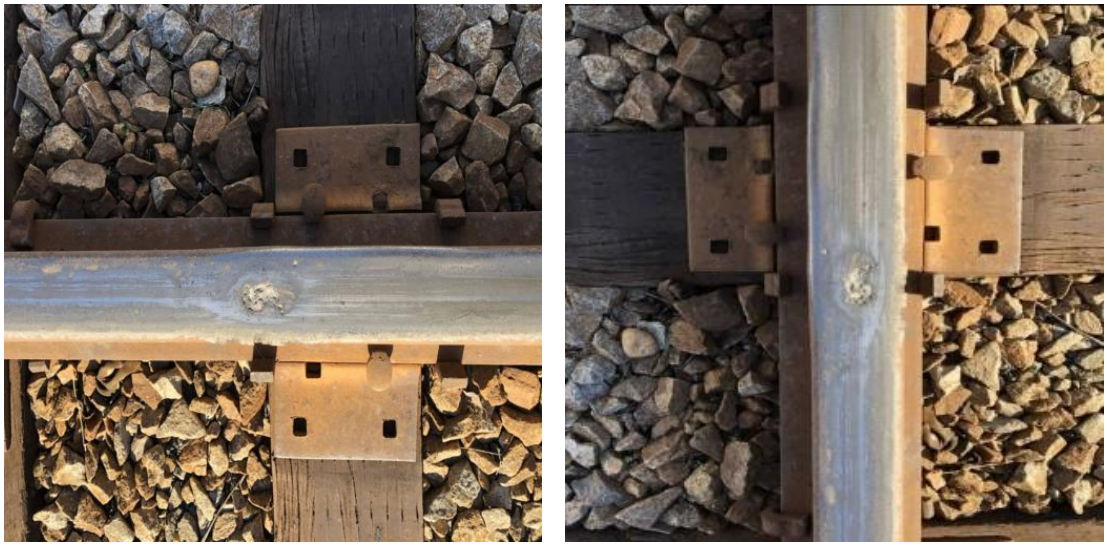


Figure 6.3. The overview of the backbone structure of Mask R-CNN.

6.3 Data preparation

A dataset with rail surface defect images (260 original images and 780 augmented images), has been built. Two object classes are included in this study, i.e., the “rail” class and the “defect” class. To keep the labeling clean and neat in the prediction images, “surface defect” is labeled as “defect. The images have been taken by an iPhone8 smartphone from two rail sections in different time near the campus of the University of

South Carolina. One is a 50 meters rail section close to Whaley St and the other one is a 100 meters rail section close to Assembly St, Columbia, South Carolina. The height between the camera and the rail surface is 20 ± 5 cm. The original image resolution is 1920×1080 pixel^2. Because of the resolution requirements on the training image, the original images are converted to 512×512 pixel^2 resolution. Overfitting refers to the trained model only performs well on the training dataset but loses its accuracy with any other dataset, which is typically due to the insufficient useful information of the training dataset. In this study, to improve the detection performance and reduce possible overfitting, image augmentations, including image rotation, mirroring, and Gaussian noise, are conducted on the source images. On one hand, image augmentation can generate similar images to enrich the dataset. On the other hand, the model would not be overfitted with more data. Examples of the original and the augmented images can be seen in Figure 6.4.



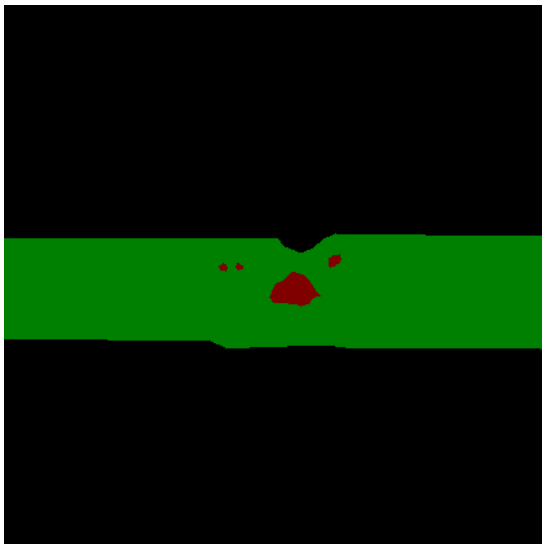
(a) (b)

(c) (d)

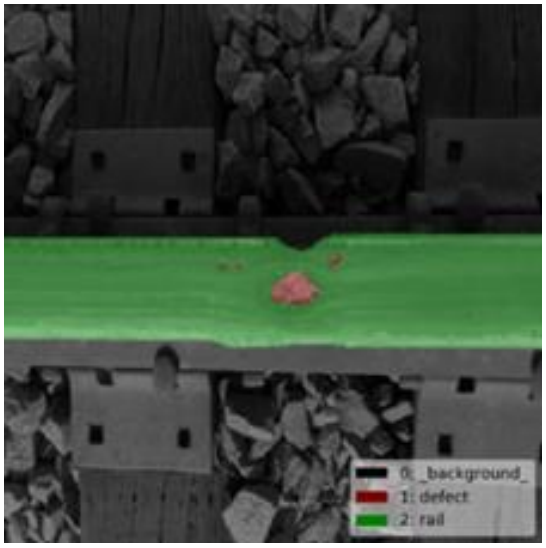
Figure 6.4. Source image and augmented image (a) source image; (b) 90 rotation (c) mirroring (d) 180 rotation and Gaussian noise.

Before the training of Mask R-CNN, the prepared image data needs to be manually labelled first. A popular image labelling tool, labelme, is used for data labeling. A total of 1,040 images after the augmentation are labeled as two classes, which are rail and defect. It is worth noting that the output file is in a JSON format which contains the location information of each manually labeled shape. All labeled images and JSON files are entered into the neural network for training and validation. The ratio of the training set and validation set is 3:1, which means there are 780 images for training and 260 images for validation. The test set is the same as the validation set. To better show the features contained in the JSON files, an example JSON file is converted to the source image, generated mask image, and its corresponding visualization image and is shown in Figure 6.5.

(a)



(b)



(c)

Figure 6.5. The converted results of a JSON file (a) source image; (b) mask file; (c) visualization of mask file.

6.4 Model training and evaluation

6.4.1 Model training

In this chapter, two different backbones, ResNet50 (50 convolutional layers) and ResNet101 (101 convolutional layers), and three different learning rates are used to train and evaluate the models. MMDetection (Chen et al., 2019), an open-source object detection toolbox that is based on the PyTorch library, is used to train Mask R-CNN models in different settings. All the training, validation, and testing tasks are completed in a workstation equipped with four NVIDIA 2080 Ti GPUs. It is worth noting that only one single GPU is called during all training, validation, and testing processes because the size of the image data is small, and it does not need multiple GPUs for computing. The operating system is Ubuntu 18.04, and the NVIDIA driver version is 440.64. To leverage the powerful computation capability of the graphic card, CUDA (version 10.2) and cuDNN (version 7.6.5) are used in the training. Specifically, CUDA is the NVIDIA's

language for expediting computation applications and cuDNN is a library that provides highly tuned implementations for different layers in deep neural networks. The PyTorch version is 1.5.0 (for Linux) and the python version is 3.7.

Table 6.1 shows the hyperparameters of each training. Note that the input size is the height and weight of an image. One epoch indicates one cycle that the entire dataset passes through the neural network with updated weight once. In this study, each model is trained by 12 epochs. The batch size is the number of images fed into the neural network in each training iteration. Momentum is used to accelerate the speed of converging and it is set to 0.9. Decay is used to prevent weights from growing too large to introduce overfitting. The value of decay is set to 0.0005. The learning rate is an important parameter to control the update rate of the training weights. If the learning rate is set too high, the model would not converge and may cause unfavorable divergent behavior. If the learning rate is set too low, the training progress would be very slow and may lead to marginal updates in the neural network. Three learning rates are tested on the two different backbones, aiming to explore an optimal hyperparameter combination for the customized dataset on the Mask R-CNN framework.

Table 6.1. Hyperparameters of each training.

Backbone	Learning rate	Input size	Epoch	Batch size	Momentum	Decay
ResNet50	0.02	512 × 512	12	8	0.9	0.0005
	0.01	512 × 512	12	8	0.9	0.0005
	0.005	512 × 512	12	8	0.9	0.0005
ResNet101	0.02	512 × 512	12	8	0.9	0.0005
	0.01	512 × 512	12	8	0.9	0.0005
	0.005	512 × 512	12	8	0.9	0.0005

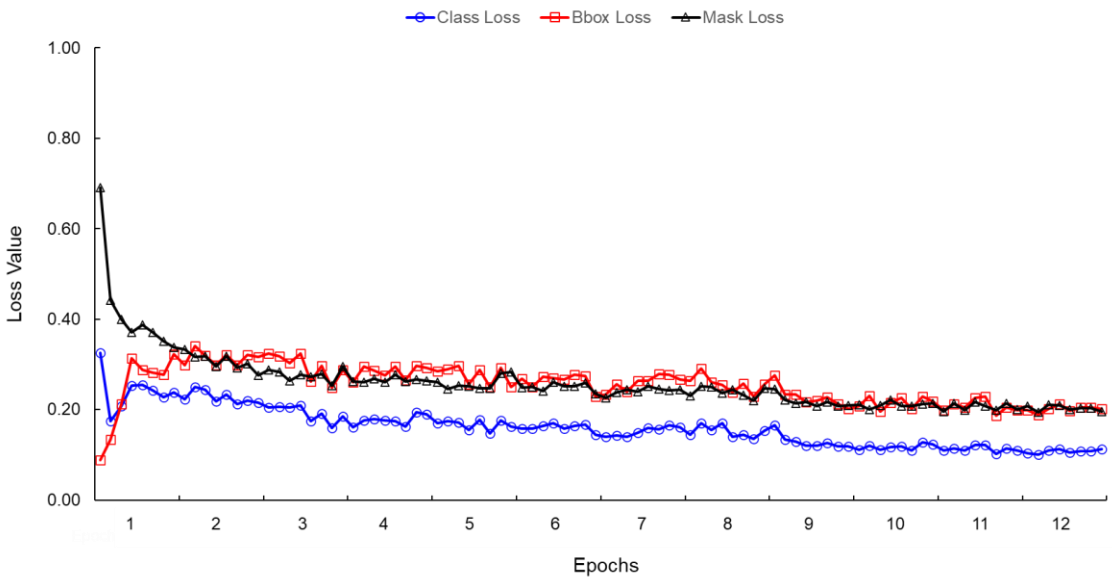


Figure 6.6. A representative training loss over epochs.

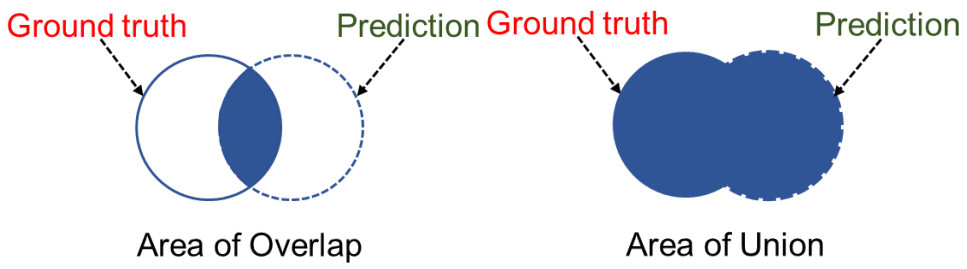
Figure 6.6 shows a representative loss graph depicting the loss changes over training iterations. As mentioned in the previous section, the three losses in the Mask R-CNN model are bounding box loss, mask loss, and regression loss. It is evident that the mask loss has the highest loss values and the class loss has the lowest loss values. It is also interesting to observe that in the initial stage, the bounding box loss has a low value,

but with the training progressing, the loss values grow bigger. At the end of the training, the values of the class loss, bounding box loss, and mask loss are 0.109, 0.205, and 0.204, respectively. Therefore, the total loss can be 0.518, which is the summation of all three losses. Note that, typically, the total loss should be less than one if there are good training and configuration.

6.4.2 Model evaluation

To evaluate the prediction performance in different settings, the parameters of mean average precision (mAP), AP_{50}, and AP_{75} are used for comparison. Precision is the percentage of correct positive predictions for overall predictions. Specifically, the mAP is the mean value of average precision (AP) for each object class. The intersection over the union (IoU) is defined in Equation (6-1). As shown in Figure 6.7**,** IoU measures the overlap between the ground truth and the prediction result for either a bounding box or a mask and distinguishes the positive case and negative case in the training and testing. Specifically, if the IoU with a ground-truth box above and equal to 0.5 is a positive case, and negative otherwise. AP_{50} refers to the AP value when the IoU threshold is 50%. Similarly, AP_{75} indicates the AP value when the IoU threshold is 75%. If the prediction is perfect, the IoU would equal to 1. In contrast, if the prediction is missed, the IoU is 0. Generally, the IoU above 50% is considered as a good prediction for object detection.

$$IoU = \frac{Area\ of\ Overlap}{Area\ of\ Union} \tag{6-1}$$



(a) (b)

Figure 6.7. The definition of overlap and union. (a) area of overlap, (b) area of union.

Figure 6.8 and Table 6.2 present the mAP, AP_{50}, and AP_{75} results with different backbones and learning rates on the prediction of the bounding box and mask. In the plots and tables, the learning rate is named as LR. Specifically, three repeated tests have been performed on each configuration, aiming to accurately evaluate each configuration's performance in a statistical manner. The mean value (MV) and standard deviation (SD) have been calculated after each parallel test set. It is worth noting that a high indicator value on MV and a low indicator value on SD mean a good prediction performance on either bounding box or mask. Typically, the mAP value is the smallest because it includes each object class. AP_{50} value is higher than AP_{75} value since the lower IoU threshold will introduces more positive cases in the experiments and generates the higher indicator value.

Figure 6.8 (a) and (b) depict the bounding box prediction results with the backbone of ResNet101 and ResNet50, respectively. For the bounding box prediction results shown in Table 6.2, the ResNet101 (LR=0.005) has the maximum MV on mAP which is 65.43% (SD=0.25), 12.86% higher than the maximum MV on mAP with

ResNet50. Regarding to the prediction results with ResNet50, it is clear that with the increase of LR on ResNet50, the Mask R-CNN's bounding box prediction performance improves. For example, with ResNet50, when the LR is 0.02, there are the maximum MVs on mAP, AP_{50} and AP_{75}, which are 52.57, 80.33, and 50.90, respectively.

However, the configuration of Mask R-CNN with ResNet50 and LR=0.005 has the minimum MVs on mAP, AP_{50} and AP_{75}, which are 34.30% (SD =0.59), 68.37% (SD=0.26), and 30.47% (SD=1.32), respectively. In other words, a low learning rate with ResNet50 produces worse performance on bounding box prediction. Interestingly, there are little differences on the bounding box prediction results with different configurations on ResNet101. For the results shown in Table 6.2, the indicator values are pretty close with different LR values on ResNet50. Overall, the Mask R-CNN model with the backbone of ResNet101 performs best on bounding box prediction on our dataset and different LR values will not introduce a large performance gap on bounding box predication.

Figure 6.8 (c) and (d) show the mask prediction results with the backbones of ResNet50 and ResNet101, respectively. Similar to the bounding box prediction results, the configuration of Mask R-CNN with ResNet101(LR=0.005) achieves the best MV results on mAP, AP_{50} and AP_{75}, which are 64.50% (SD=0.22), 90.37% (SD=0.12), and 63.37% (SD=0.66). As for the configuration with ResNet50, when the LR is 0.02, Mask R-CNN achieves the best MV result on mAP, AP_{50} and AP_{75}, which are 54.53% (SD=0.56), 81.07% (SD=0.05), and 52.37% (SD=0.59), respectively. It is worth noting that under the configuration of ResNet50 backbone, with the increase of the LR, the indicator values of mask prediction results increase.

Regarding the mask prediction performance under the configuration of Mask R-CNN with ResNet101 and different LR values, it is easy to find there are similar performances with each configuration. In detail, the gaps between the maximum MVs and minimum MVs on the mAP, AP_{50} and AP_{75} are 0.1%, 0.37%, and 0.97%, respectively. Similar to the bounding box prediction results, it shows different LR would not introduce much difference in mask prediction under the model configuration with ResNet101. In short, it can conclude that the Mask R-CNN setting of ResNet101(LR=0.005) has the best performance on the prediction of the bounding box and mask on this customized rail surface defects dataset.

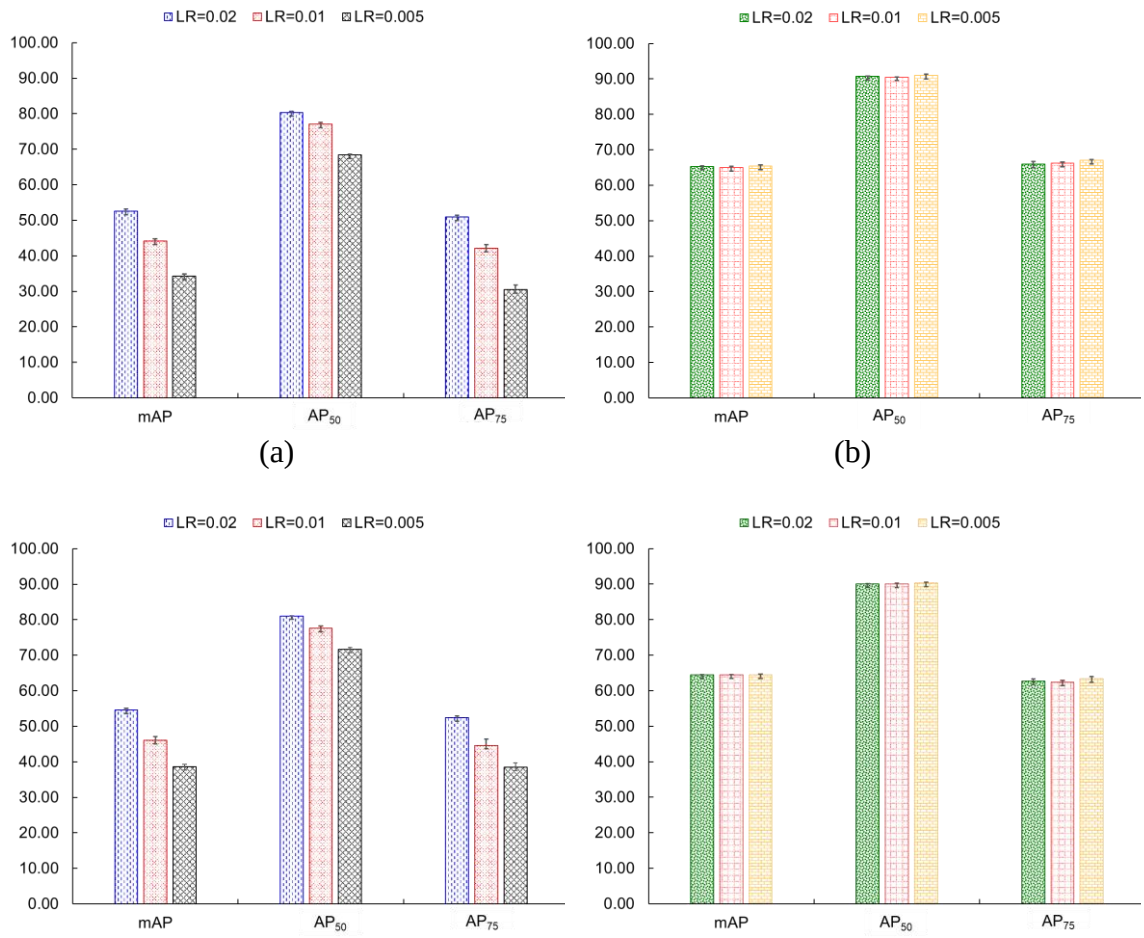


Figure 6.8. AP results of Mask R-CNN models with different backbones and learning rates. (a) bounding box results of ResNet101; (b) bounding box results of ResNet50; (c) mask results of ResNet101; (d) mask results of ResNet50.

Table 6.2. Parallel test results of Mask R-CNN models with different backbones and learning rates.

Backbone	Learning rate	Bbox			Mask		
		mAP	AP_{50}	AP_{75}	mAP	AP_{50}	AP_{75}
ResNet101	LR=0.02	65.50	90.60	66.80	64.60	89.90	62.50
		65.10	90.60	66.10	64.30	90.20	63.60
		65.40	90.90	65.10	64.30	90.00	61.80
	Mean Value (MV)	65.33	90.70	66.00	64.40	90.03	62.63
	Standard Deviation (SD)	0.17	0.14	0.70	0.14	0.12	0.74
	LR=0.01	65.40	90.50	66.60	64.30	90.30	62.00
		64.90	90.50	66.20	64.60	90.00	63.10
		64.70	90.60	66.00	64.30	89.70	62.10
	Mean Value (MV)	65.00	90.53	66.27	64.40	90.00	62.40
	Standard Deviation (SD)	0.29	0.05	0.25	0.14	0.24	0.50
	LR=0.005	65.70	90.50	67.10	64.70	90.20	64.10
		65.10	91.30	66.90	64.20	90.40	62.50
		65.50	91.10	67.20	64.60	90.50	63.50
	Mean Value (MV)	65.43	90.97	67.07	64.50	90.37	63.37

	Standard Deviation (SD)	0.25	0.34	0.12	0.22	0.12	0.66
ResNet50	LR=0.02	53.20	80.50	51.40	55.30	81.10	53.20
		52.70	80.70	50.30	54.30	81.10	52.00
		51.80	79.80	51.00	54.00	81.00	51.90
	Mean Value	52.57	80.33	50.90	54.53	81.07	52.37
	Standard Deviation	0.58	0.39	0.45	0.56	0.05	0.59
	LR=0.01	44.50	77.40	41.90	46.80	78.20	45.10
		44.70	77.50	43.40	46.70	78.00	46.60
		43.20	76.40	40.90	44.80	76.70	42.20
	Mean Value (MV)	44.13	77.10	42.07	46.10	77.63	44.63
	Standard Deviation (SD)	0.66	0.50	1.03	0.92	0.66	1.83
	LR=0.005	34.90	68.50	31.50	38.80	71.60	39.00
		33.50	68.00	28.60	37.80	71.10	37.00
		34.50	68.60	31.30	39.30	72.40	39.60
	Mean Value (MV)	34.30	68.37	30.47	38.63	71.70	38.53
	Standard Deviation (SD)	0.59	0.26	1.32	0.62	0.54	1.11

Note: all the parameters' unit is percentage (%)

6.5 Inspection performance

To evaluate the prediction performance of Mask R-CNN on the rail surface defect, the parameter setting of the ResNet101 backbone with the learning rate of 0.005, which achieves the best performance as discussed in the last section, is used to inspect the rail surface defects. A total of sixteen testing images are tested in this section. Figure 6.9 and Figure 6.10 show the testing images and their corresponding prediction results with different orientations and different defect severities. Figure 6.11 presents the comparison results of Mask R-CNN and Otsu's method. Figure 6.12 depicts the inspection performance under different light conditions.

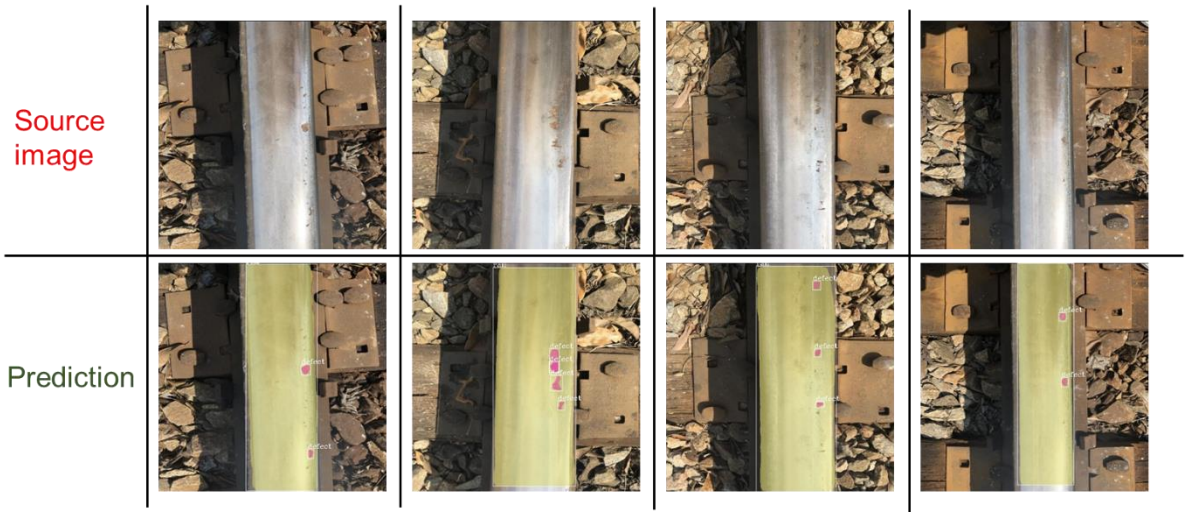
6.5.1 Influence of rail orientation and different defect severities

In Figure 6.9 (a), the rail has a vertical orientation. While another horizontal section of rail is shown in Figure 6.9 (b). The reason to have different orientations is to investigate if the orientation has an impact on the prediction performance. In Figure 6.10 (a), there are relatively mild defect conditions. In Figure 6.10 (b), there are relatively severer defect conditions processed by the enlarged defect parts in the images. The reason behind this is to explore the inspection performance on different defect severities. In Figure 6.9 (a), some obvious defects can be seen, and the developed model predicts well those defects, especially in the second source image which has a continuously defected area. However, as the evaluation metrics in the previous section indicate, the performance on small defects detection could be further improved as suggested by the last two pictures of Figure 6.9 (a). However, those very tiny defects may not impact the track performance very much unless they grow larger, in which case they would be detected by the proposed

model. In Figure 6.9 (b), it clearly shows that almost all defects are successfully detected by Mask R-CNN, indicating promising inspection results.

In Figure 6.10 (a), with the relatively mild defect conditions, it can be found that our trained Mask R-CNN model can perform well. There are at least seven defects on each original image shown in Figure 6.10 (a) and the re-trained model have detected the defects' shapes and locations with an accurate manner, indicating its promising performance on the mild defect conditions. As for Figure 6.10 (b), to furtherly investigate the model's performance on the relatively more severe conditions, which is common in the field, four images with dense and packed defects have been utilized for test. Each image has at least eight dense defects with small or large shapes. Obviously, it can be found that the re-trained model can detect dense rail surface defects very well on either their shape and locations.

Overall, the orientation of the rail has no impact on the detection of the rail surface defect, suggesting a camera could be mounted at an arbitrary orientation for inspection. The re-trained model can detect different defect severities on the rail surface. The results also indicate Mask R-CNN has promising performance for rail surface defect detection and the potential for field applications, but further improvements can be done for very small defects detections and also the implementation of the model with a computing board needs to be developed in the near future.

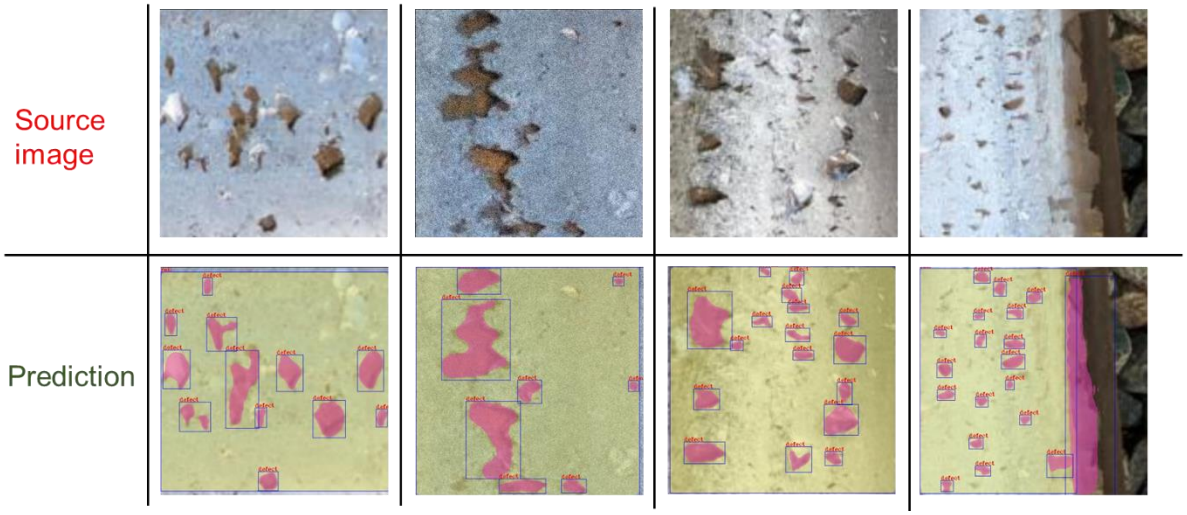


(a)

(b)

Figure 6.9. Inspection performance of Mask R-CNN on the rail surface defect with different orientations. (a) Images with vertical orientation, (b) Images with horizontal orientation.

(a)



(b)

Figure 6.10. Inspection performance of Mask R-CNN on the rail surface defect with different defect severities. (a) Images with relatively mild defect conditions, (b) Images with relatively severer defect conditions (enlarged to show details).

6.5.2 Comparison of inspection performance between Mask R-CNN and Otsu's method

To evaluate the inspection performance and segmentation effect between Mask R-CNN and the traditional image processing algorithm, Otsu's method, which is commonly used for obtaining segmentation results with simple thresholding to separate foreground and background. It needs to mention that the threshold is the average of mean levels of

foreground and background divided by it (Xu, Xu, Jin, & Song, 2011). The reasons to choose Otsu's method are twofold: (1) It is a representative method which is simple and widely studied in other engineering domains; (2) It has similar functionality with Mask R-CNN and both of them can be used for image segmentation, which is useful for defect size analysis in the future study. Figure 6.11 presents the comparison results between Mask R-CNN and Otsu's method.

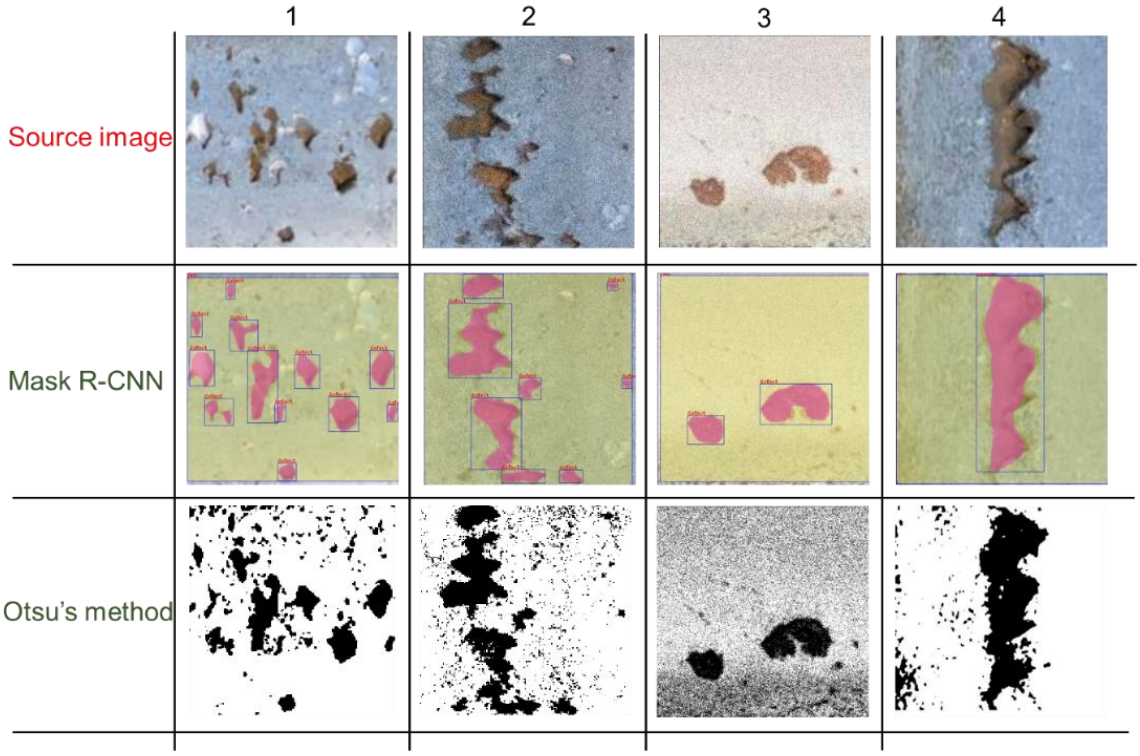


Figure 6.11. Performance comparison between Mask R-CNN and Otsu's method.

In Figure 6.11, compared to Otsu's method, Mask R-CNN has better performance on both dense defect conditions (columns 1 and 2) and sparse defect conditions (columns 3 and 4). In columns 1 and 2, there are 13 defects and 8 defects identified by Mask RCNN. Meanwhile, although Otsu's method can separate the area of large defects, it still introduces a lot of noise on the testing images especially for the image in column 2. In

columns 3 and 4, the sparse defect cases, it can be concluded that the fine-tuned Mask R-CNN model can also inspect and distinguish the rail and the defects well. Regarding the performance of Otsu's method in columns 3 and 4, it can be found that Otsu's method cannot separate the foreground and background (i.e., column 3) since their grayscale is similar and the noise is still significant in the sparse case. Besides, Otsu's method cannot deliver the type of object class in the inspection results. For example, taking a view on the inspection result using Otsu's method in column 4, if there is no label to mark it as surface defect, it is a bit hard to judge.

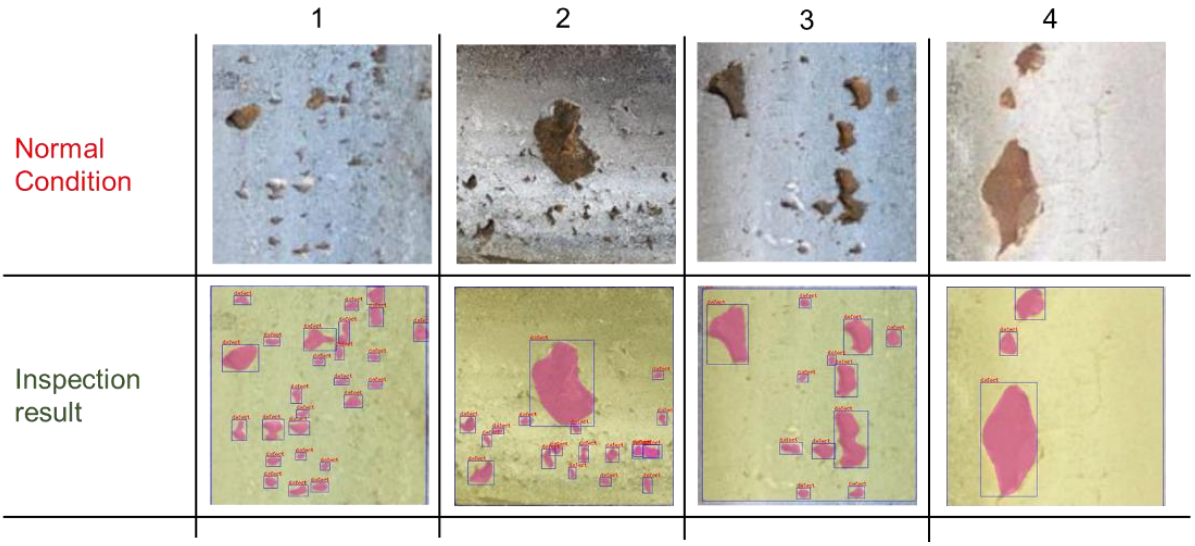
6.5.3 Inspection performance under different light conditions

In the real practice, there are many environmental noises which can negatively impact the inspection results. Those factors need to be considered include but not limited to rust, shadow, mud-spot, over-exposure, and dust. Due to the limitation of the available testing data, many of those factors are not covered. Inspired by previous studies (Guo, Qian, & Shi, 2021; Guo, Qian, Wu, et al., 2021; Wu et al., 2018), the inspection results of rail surface defects under different light conditions are discussed in this section.

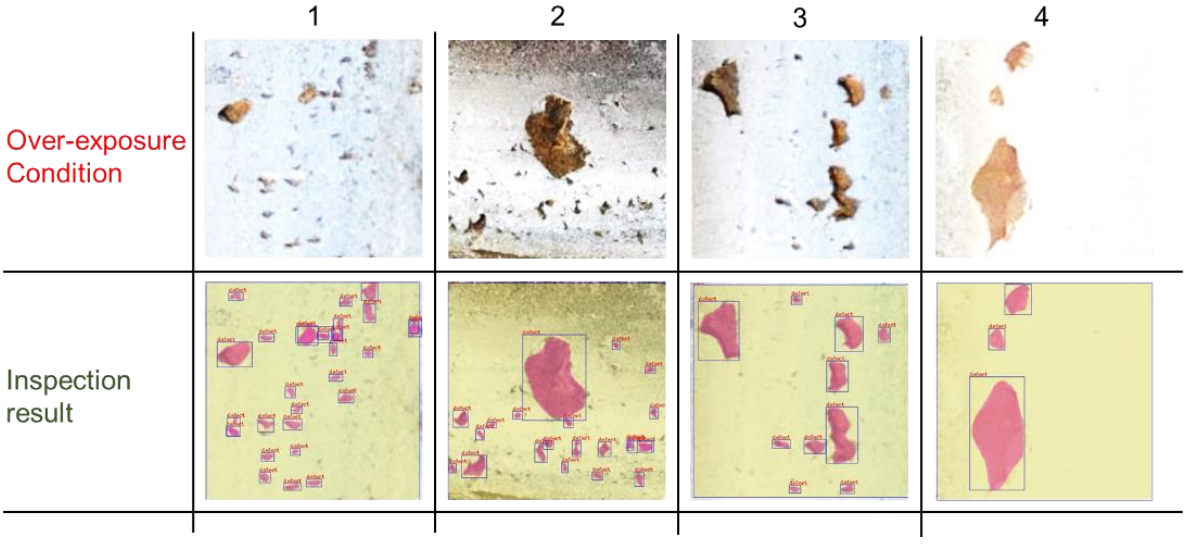
Figure 6.12. shows the inspection performance of rail surface defects under three light conditions which are normal, over-exposure, and weak-light conditions, respectively. Even it can be found that the fine-tuned Mask R-CNN can perform well under all the three light conditions, the inspection results are still influenced by the light variation. Specifically, in Figure 6.12 (b), there are mislabeled defect results in column 1 and missed defect results in column 3. In Figure 6.12 (c), it is easy to find that each mask is a little bit larger than the results in Figure 6.12 (a) and (b). It may be due to the

boundary of defects is not as obvious as in previous conditions and it will introduce the errors in the defects size evaluation in the future study.

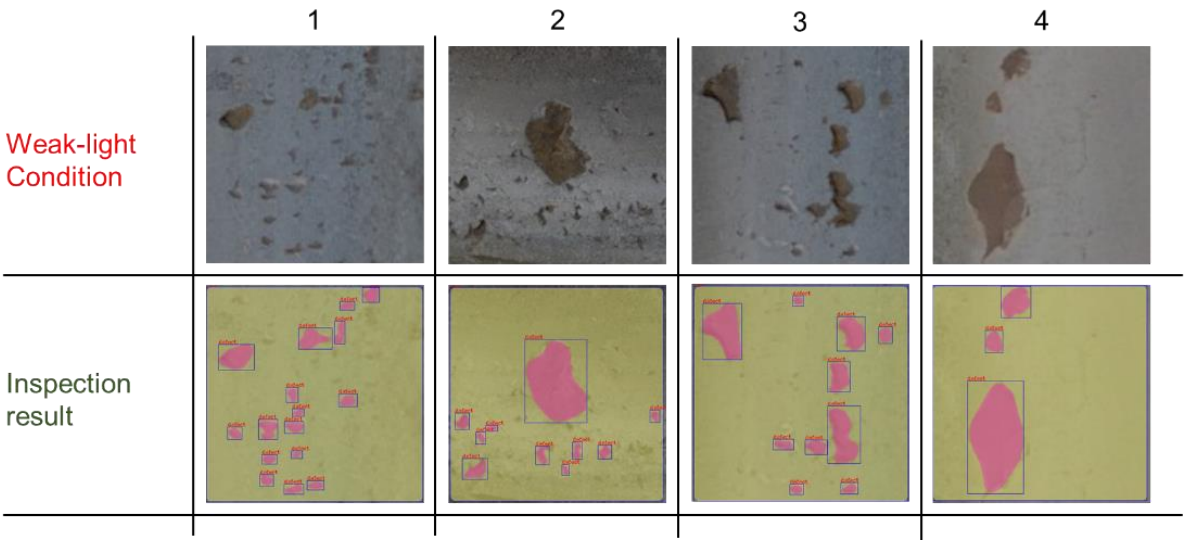
To put it into a nutshell, all testing results indicate Mask R-CNN has promising performance for rail surface defect detection and the potential for field applications, but further improvements can be done for defects size evaluation and grading and the implementation of the model with a computing board needs to be developed in the near future.



(a)



(b)



(c)

Figure 6.12. Inspection performance under different light conditions (a) Normal condition; (b) Over-exposure condition; (c) Weak-light condition.

6.6 Summary

In this chapter, an automatic inspection framework using Mask R-CNN to inspect the rail surface defect is proposed, aiming to improve the inspection accuracy and efficiency, save the labor cost, and improve the railroad safety. Two different backbones, ResNet50 and ResNet 101, are implemented into Mask R-CNN to test their feature extraction capability on a customized rail surface defect dataset. Three different learning rates which are 0.02, 0.01, and 0.005 are selected for testing the optimal learning speed on this customized dataset. A total of 1,040 images, including two object classes, rail and defect, are utilized for training and evaluation. Different parameter configurations are trained and evaluated based on the MMDetection toolbox. The evaluation metrics including mAP, AP_{50}, and AP_{75} are used for the model evaluation. Parallel tests have been performed and the test results have been analyzed in a statistical manner using the parameters of MV and SD. Test results indicate that under the configuration of the

ResNet101 backbone and the learning rate of 0.005, Mask R-CNN can achieve the highest MVs on mAP with respect to the bounding box and mask predictions, which are 65.43 (SD=0.25) and 64.50 (SD=0.22), respectively. A total of sixteen images with different defect severities have been tested on the optimal parameter settings of Mask R-CNN. The comparison between fine-tuned Mask R-CNN model and Otsu's method has been conducted. Experimental results present a better performance than Otsu's method and show a promising performance on the rail surface defect inspection regardless of the rail orientation, different defect severities, and different light conditions.

To our best knowledge, this is the first attempt using the instance segmentation model, Mask R-CNN to inspect or predict the rail surface defects. The results indicate a possible solution by using Mask R-CNN to inspect rail surface defect in future applications. However, the prediction performance of the developed model can be further improved as more training data are used.

CHAPTER 7 A NOVEL LIGHTWEIGHT SEMANTIC SEGMENTATION MODEL FOR RAIL SURFACE DEFECTS DETECTION AND QUANTIFICATION [7]

[7] Guo, Feng, Yu Qian. Submitted to *Transportation Research Record*.

Rail surface defect is an important railroad track quality indicator and impacts the overall track safety. The rail surface condition needs to be inspected regularly to evaluate defects such as rail surface defect, following the FRA Track Inspector Rail Defect Reference Manual, Title 49 Code of Federal Regulations (CFR), and Track Safety Standards (TSS), to make proper maintenance plan. Unfortunately, rail surface defect defection has not been fully automated yet and significant manual inspection is still involved, which could be labor-intensive but low-efficient. Earlier efforts have developed some automatic rail surface defect detection systems, but the accuracy and equipment cost have limited their applications in the field. To address the needs in the track inspection, this study proposes a Lightweight Deeplabv3Plus model using Lovász-Softmax loss (LDL model) to automatically detect rail surface defects. The ResNet-18 backbone is adopted for the encoder design to reduce the computational cost and maximize the inference speed. Meanwhile, the patch-wise training strategy is conducted and the Lovász-Softmax loss is implemented to improve the inspection accuracy. To quantify different severity levels, a rail surface defect severity index using the ratio between the defect area and the rail surface area is proposed. The developed model is compared with another five state-of-the-art (SOTA) models to validate the performance. Experimental results confirm the proposed model reaches the highest mIoU, outperforming other available models. This work provides a feasible solution for the future implementation of automatic railroad track inspection.

7.1 Introduction

Rail is a key component of the railroad track and well-maintained rails play critical role in safe operation. According to the report (FRA, 2020b) from FRA and a study conducted by Liu (Liu, 2017), rail defect is the leading cause of freight train derailments in the United States. For instance, the National Transportation Safety Board (NTSB) concluded a piece of broken rail with the evidence of rolling contact fatigue (RCF) led to the severe accident happened in Columbus, OH that caused over $1.2 million damage (Zakar & Mueller, 2016). Small rail surface defects (RSDs), cracks, flaking, and spalling on top of the rail may be the results of different stages of RCF or other types of more severe rail defects (Stewart & Ahmed, 2002). In other words, close attention should be paid to monitor RSD initiation and development during track inspections because even small RSDs may contribute to more serious type of rail defects or even rail breakage in the future, and should be carefully inspected and evaluated.

Over the past decades, many studies (Faghih-Roohi et al., 2016; Li & Ren, 2012a; Min, Xiao, Dang, Yue, & Cheng, 2018) have developed different methods to improve the rail defect inspection. Up to date, the common practices include but not limit to human visual inspection, acoustic emission test (Ye, Stewart, & Roberts, 2019), Eddy current test (Pohl, Erhard, Montag, Thomas, & Wüstenberg, 2004; Rajamäki, Vippola, Nurmikolu, & Viitala, 2018), and ultrasonic (Pohl et al., 2004). These approaches are typically developed for rail internal defect inspections and have limited applications on the rail surface defect detection. Moreover, other than human visual inspection which is labor- and time-consuming, most of the approaches need expensive specialized equipment and well-trained technicians. Recently, with the rapid development of

convolutional neural networks (CNNs) and computer vision (CV) techniques, cost-effective yet accurate surface defect inspection has become possible and successful applications can be found in pavement engineering and structure engineering (Guo, Qian, & Shi, 2021; Guo, Qian, Wu, et al., 2021; A. Zhang et al., 2017; Xinxiang Zhang, Dinesh Rajan, et al., 2019; Zhang, Story, & Rajan, 2021; Xinxiang Zhang, Ye Wang, et al., 2019). Fortunately, more attention has been paid to the development of computer vision-based rail surface defect and other track component inspections.

Current CV-based track inspection mainly focuses on missing component (i.e., rail, spike, and clip) defection and component defect identification. For example, Guo *et al.* (Guo, Qian, & Shi, 2021; Guo, Qian, Wu, et al., 2021) developed and applied the improved YOLACT and YOLOv4 models on the multi-class track components inspection with high recall and average precision (AP) results. Qi *et al.* (Qi, Xu, Wang, Cheng, & Chen, 2020) proposed MYOLOv3-Tiny for a real-time detection on track fasteners with reduced computational complexity and high detection precision. Compared to CV-based track components inspection, track component defect detection is more challenging. Specifically, the rail surface defect inspection task has the following difficulties: (1) The developed model needs to be adaptive to random noise brought by the complicated field conditions, such as reflection from other track components; (2) The issue of imbalanced instances, which means the ratio between the aera of rail surface defect and rail surface area is very small and this will cause model training failures; (3) The state-of-the-art models are developed for general detection purpose which require high computational resources but may yield low accuracy in specific railroad applications; (4) There is no literature to clearly benchmark the severity of rail surface

defects for CV models and most of the studies just evaluate their recall and precision without any quantitative evaluation which cannot provide useful information for maintenance planning.

7.2 Related work

In general, there are two types of CV-based methods to inspect the rail surface defect which are the conventional image processing approach and the deep learning approach, respectively. Related articles are reviewed and summarized in this part.

7.2.1 Methodology overview

Conventional image processing needs hand crafted features specified by experienced technicians to process the image data. This approach is not friendly for customization and extension, especially when the application environment changes or it is used with a large dataset. Li et al. (Li & Ren, 2012a) proposed a real-time visual inspection system (VIS) with four core components which are image acquisition system, track extraction algorithm, local normalization (LN) method, and projection profile (DLBP). Because only the local information was used, this model has large space to be improved in terms of detection speed and precision. Gan et al. (Gan, Li, Wang, & Yu, 2017) developed an automatic VIS with the coarse extractor and fine extractor, aiming to address the large variation issue of the rail surface defect appearance. Besides, Gan et al. (Gan, Wang, Yu, Li, & Shi, 2018) proposed the background-oriented defect inspector (BODI) to improve the inspection performance with a random sampling stage for acquiring the rail background features. In Yu et al. (Yu et al., 2018), the coarse-to-fine

model (CTFM) was proposed to identify the rail surface defect in three levels which were sub-image level, region level, and pixel level. Same as Li et al. (Li & Ren, 2012a) and Gan et al. (Gan et al., 2017), Type-I and Type-II rail surface defects were tested. Deutschl et al. (Deutschl, Gasser, Niel, & Werschonig, 2004) formulated the spectral image difference procedure (SIDP) with shading correction and replacement of the difference by another function. Zhuang et al. (Zhuang, Wang, Zhang, & Tsui, 2018) advanced the feature-based linear iterative crack aggregation (FLICA) which can automatically locate the rail surface crack and obtain the boundary of a rail surface crack. Leveraging the superpixel classifier, Wang et al. (Wang, Zhuang, & Zhang, 2019) designed a bilevel superpixel-based framework for rail surface crack inspection. The simple linear iterative clustering (SLIC) module is responsible for the generation of superpixels of raw rail images and the bag-of-words (BoW) was used for developing superpixel classifier.

7.2.2 Rail surface defect inspection with deep learning

Deep learning approaches are more convenient on training customized data because there is no pre-defined features and associated restrictions. Meanwhile, the neural network is enabled to extract and learn features by itself. Taking the advantage of the rapid evolving deep convolutional neural networks (DCNNs), Faghih-Roohi et al. (2016) successfully applied DCNN on the rail defect identification to save computational cost for feature extraction. Shang et al. (2018) proposed a two-stage approach with CNN to localize and classify the rail surface defect. In the first stage, the training image was cropped to better focus on the rail part. In the second stage, a fine-tuned CNN was applied to extract rail defect features. Feng et al. (2020) designed M2-Y3 and M3-Y3

models for rail defect detection the framework of You Only Look Once (YOLO) (Redmon et al., 2016) and MobileNet (Howard et al., 2017). However, their model focused on judging the existence of rail surface defect but cannot depict the shape of any specific defect. Integrating multiple context information, pyramid pooling module, attention mechanism, and information fusion, Zhang et al. (2020) developed multiple context information network (MCnet) for the evaluation of no-service rail surface defects (NRSDs). Two types of NRSDs including both the original and artificial types were tested and the results indicated reasonable segmentation performance. Combining the unsupervised learning with an improved Gaussian mixture model for the rail surface defect segmentation, and the supervised learning with Faster R-CNN for rail surface defect localization, Lu et al. (2020) proposed SCueU-Net for incorporating U-Net and saliency cues method to automatically detect rail surface defect. Unfortunately, only a total of 201 samples were included in that study even after data augmentation. The potential overfitting issue during training was not addressed.

7.2.3 Scope and objective

To address the aforementioned challenges in rail surface defect detection, this chapter aims to develop a novel lightweight semantic segmentation model that is able to quantify rail surface defect severity and maintain high computational efficiency. The specific contributions are summarized as follows:

- A Lightweight Deeplabv3Plus model using Lovász-Softmax loss (LDL model) is proposed using the ResNet18backbone for a real-time inference speed in the

encoder design. A patch-wise training strategy is implemented to better capture local features and improve training efficiency.

- The Lovász-Softmax loss which can directly optimize the mean intersection over union (IoU) is adopted in the LDL model to train multi-class objects with better semantic segmentation accuracy.
- A total of 1,078 rail surface defect images collected from the field are trained, tested, and validated on six models for comparison.
- The rail surface defect severity has been quantified and rated based the ratio of the defect area over the rail surface area. Different severity indexes can be customized by the end-user.

7.3 Methodology

There are four parts in the proposed rail surface defect inspection framework, which includes 1) a lightweight backbone for a better inference speed and high accuracy for field applications, 2) a patch-wise training strategy to keep the local information and avoid the feature loss in the downsampling, 3) a loss function to optimize the jaccard loss and improve the relevance to small objects (i.e., tiny defects), and 4) an index using the severity ratio with the number of pixels is proposed to quantify rail surface defect severity levels.

7.3.1 Baseline

Deeplabv3Plus (Chen, Zhu, Papandreou, Schroff, & Adam, 2018), an encoder-decoder framework developed by Google Inc, is selected as the baseline. Specifically, Deeplabv3Plus makes good use of the spatial pyramid pooling module for encoding multi-scale contextual information at an arbitrary resolution and recoveries detailed information of object boundaries by a simple yet effective decoder module. On the one hand, Deeplabv3Plus inherit the advantages of its ancestor, DeepLabv3 (Chen, Papandreou, Schroff, & Adam, 2017), which employs the atrous spatial pyramid pooling (ASPP) to extract multiple scale image-level features, as the proposed encoder. On the other hand, Deeplabv3Plus incorporates a novel decoder architecture (see Figure 7.1) and the modified aligned Xception model to boost the semantic performance and improve the computation efficiency. The structure of Deeplabv3Plus is given in Figure 7.1.

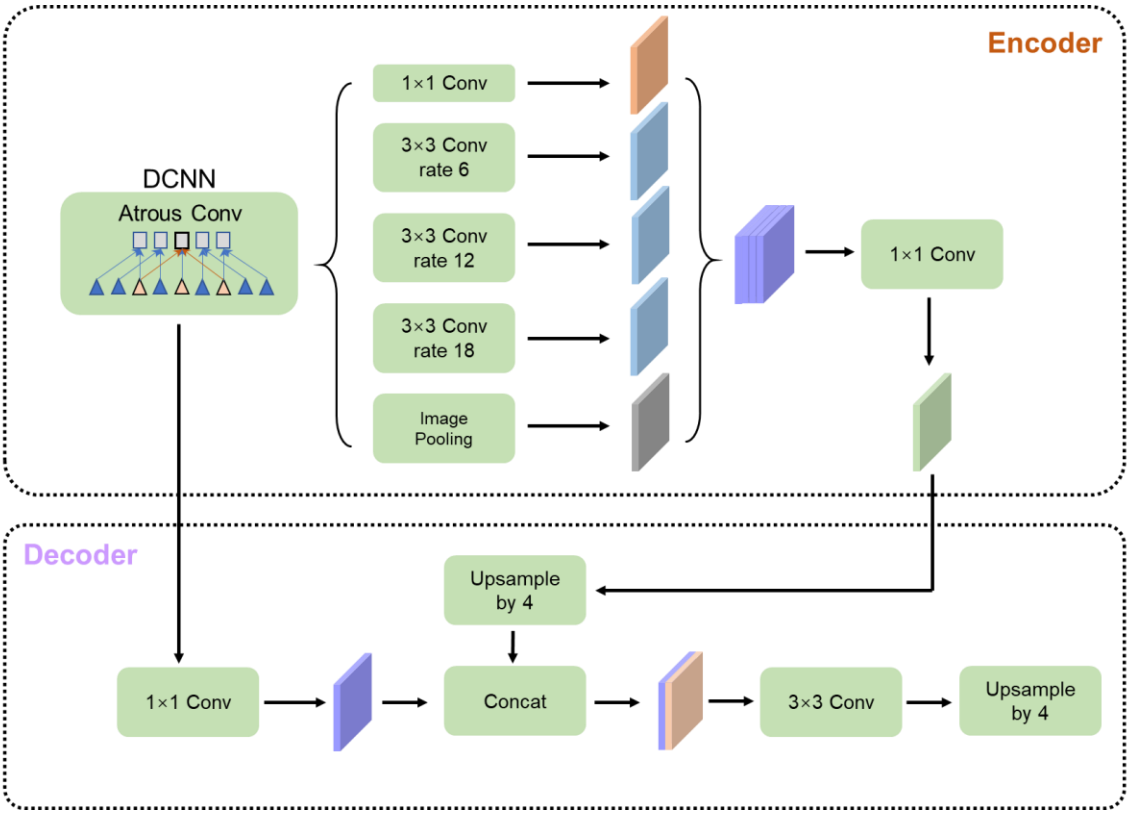
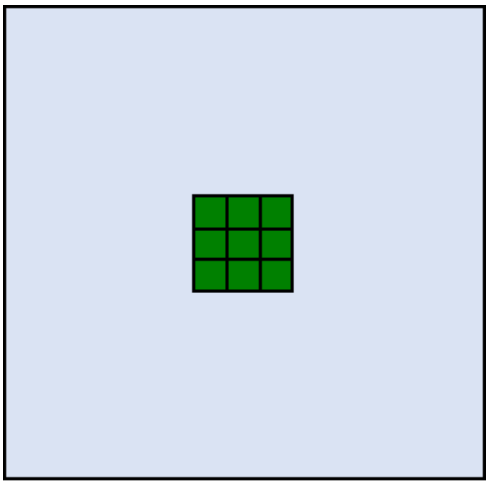


Figure 7.1. The Deeplabv3Plus architecture.

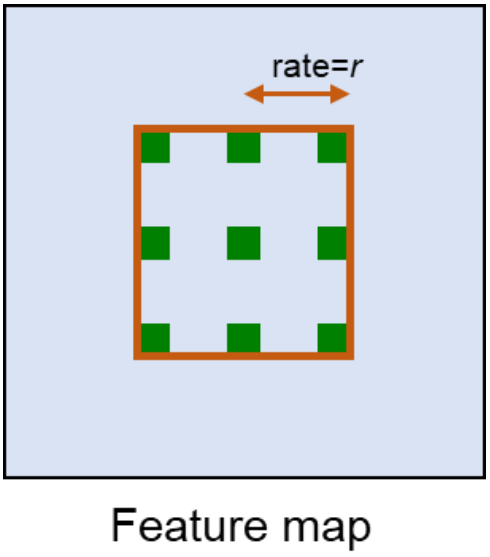
As shown in Figure 7.2, the atrous convolution (see Figure 7.2 (b)) is responsible for reducing the computation complexity, enlarging the receptive field, and explicitly controlling the resolution of features calculated by DCNN. Meanwhile, it keeps the standard convolution operation (see Figure 7.2 (a)) for extracting multiple features without extra parameters. For instance, when dealing with 2D signals, the atrous convolution can be applied over the input feature map x to generate the output feature map y on each location i via a filter w which has a kernel size of k. The computation process is shown in Equation (7-1).

$$y[i] = \sum_{k} x[i + r \cdot k] w[k] \tag{7-1}$$

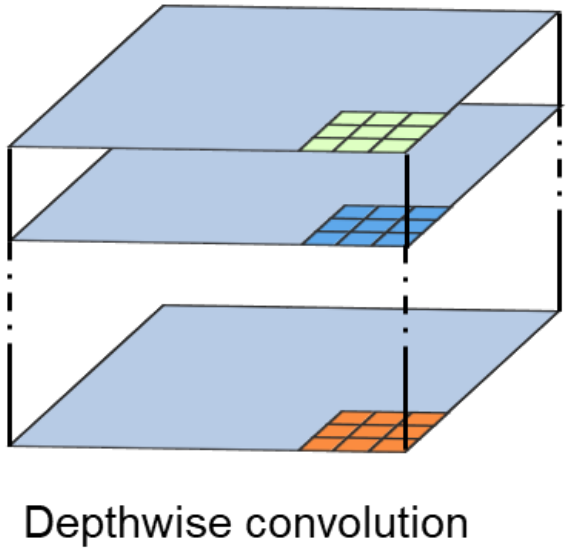
where the r represents the atrous rate that corresponds to the stride with the input signal. It needs to mention that the standard convolution can be treated as a special case of atrous convolution with $r=1$.



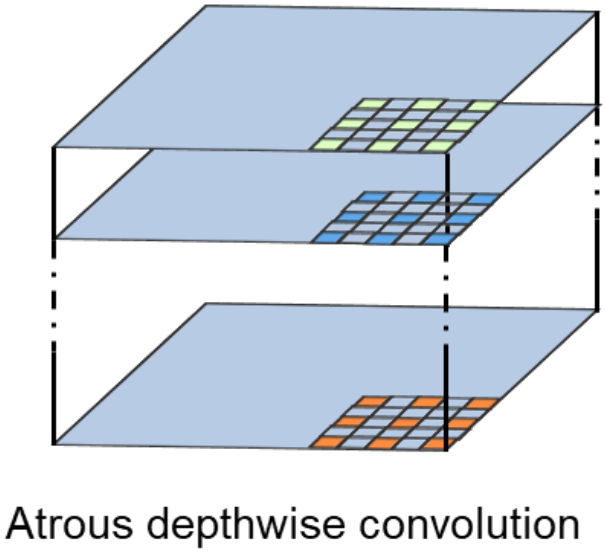
(a)



(b)



(c)



(d)

Figure 7.2. Different convolution operations. (a) Standard convolution, (b) Atrous convolution with a rate *r*, (c) Depthwise convolution, (d) Atrous depthwise convolution.

It needs to mention that in the proposed model, the atrous depthwise convolution (see Figure 7.2 (d)), which leverages the depthwise convolution (see Figure 7.2 (c)) and atrous convolution for effectively reducing the computation complexity and maintaining

a superior performance, has been implemented in the original Deeplabv3Plus. In addition, the modified Xception model with more layers, replaced max pooling operations with depthwise separable convolutions and extra batch normalization has also been added into the original Deeplabv3Plus.

7.3.2 ResNet18 backbone

Because of the promising feature extraction performance, ResNet 50 (He et al., 2016) and ResNet 101(He et al., 2016) are popular backbones for the CNN design to improve model performance on large image datasets that include millions of images and many different instance classes. However, strong and versatile backbones also require higher computational recourse and time. Focusing on the specific task of rail surface defect inspection, a backbone with a large number of layers and complicated bottleneck designs may not be necessary considering the very limited object classes and relatively simple context compared to a general large database. For the sake of tailoring the network design and expedite the training process to avoid redundancy, ResNet 18, a backbone with much less layers and parameters compared to ResNet 50 and ResNet 101, is selected as the backbone to build the encoder structure in the proposed model. Figure 7.3 gives the general architecture of ResNet 18 (He et al., 2016).

There are four convolutional stacks which are Conv2, Conv3, Conv4, and Conv5 in the whole structure. Correspondingly, the output channels are 64, 128, 256, and 512, respectively. Before the four convolutional stacks, there is a single convolution layer with a 3×3 kernel and a stride of 2. The number of the output channels of the first layer is 3×3. To be consistent, the kernel sizes in the four convolutional stacks are also 3×3. To

simplify the schematic of ResNet 18, the batch normalization layer after each convolution layer and ReLU activation are not shown in Figure 7.3.

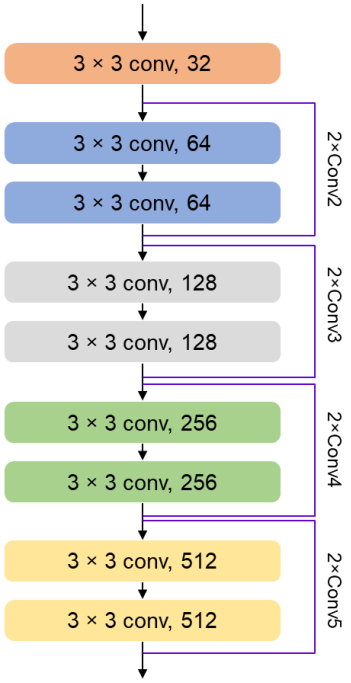


Figure 7.3. The architecture of ResNet 18.

7.3.3 Patch-wise training strategy

Semantic segmentation is a pixel-level classification of cluster pixels of the same object instance in an image. In general, with a large image size and multiple object classes, the training requires more computational resource, GPU capacity, and time. Down-sampling/resize the input to small ones is helpful to keep computational expense in control, but this approach causes more local information loss on each image and therefore hurts the final training performance (Xinxiang Zhang, Dinesh Rajan, et al., 2019). For the specifically application in this study, the rail surface defects usually are very small and the true information can be easily lost when having small input sizes. To balance the

accuracy and training speed, this study proposes a patch-wise instead of pixel-level training strategy to optimize the training speed and meanwhile achieve better accuracy.

Figure 7.4 illustrates the details of the proposed patch-wise training strategy. In the first step, all images and corresponding annotations are cropped into sub-images for training and validation. In the experimental design, there are four different cropping sizes which are 64×64, 128×128, 256×256, and the whole image having no cropping, respectively. The reason to have different cropping sizes is different cropping sizes impact the training accuracy and speed a lot. More details are discussed in the following section. In the following steps, sub-images and corresponding annotations are fed into the improved Deeplabv3Plus model for training and validation. The final semantic segmentation results can be visualized after calling the trained weights in the testing algorithm. Specifically, there is no crop process in the testing algorithm and there is no limit, such as the image size must be divided by 32 which is commonly used in the deep learning training process, on the testing input. It is worth noting that because the images in this study are not as large as a high-resolution remote sensing image which is typically over 2,000×2,000, and have limited object instance classes, overlapping between the sub-images is not necessary. Moreover, to augment the image data to increase the dataset adequacy, the random flipping with a flip probability of 0.5 is adopted in the training with the patch-wise training strategy.

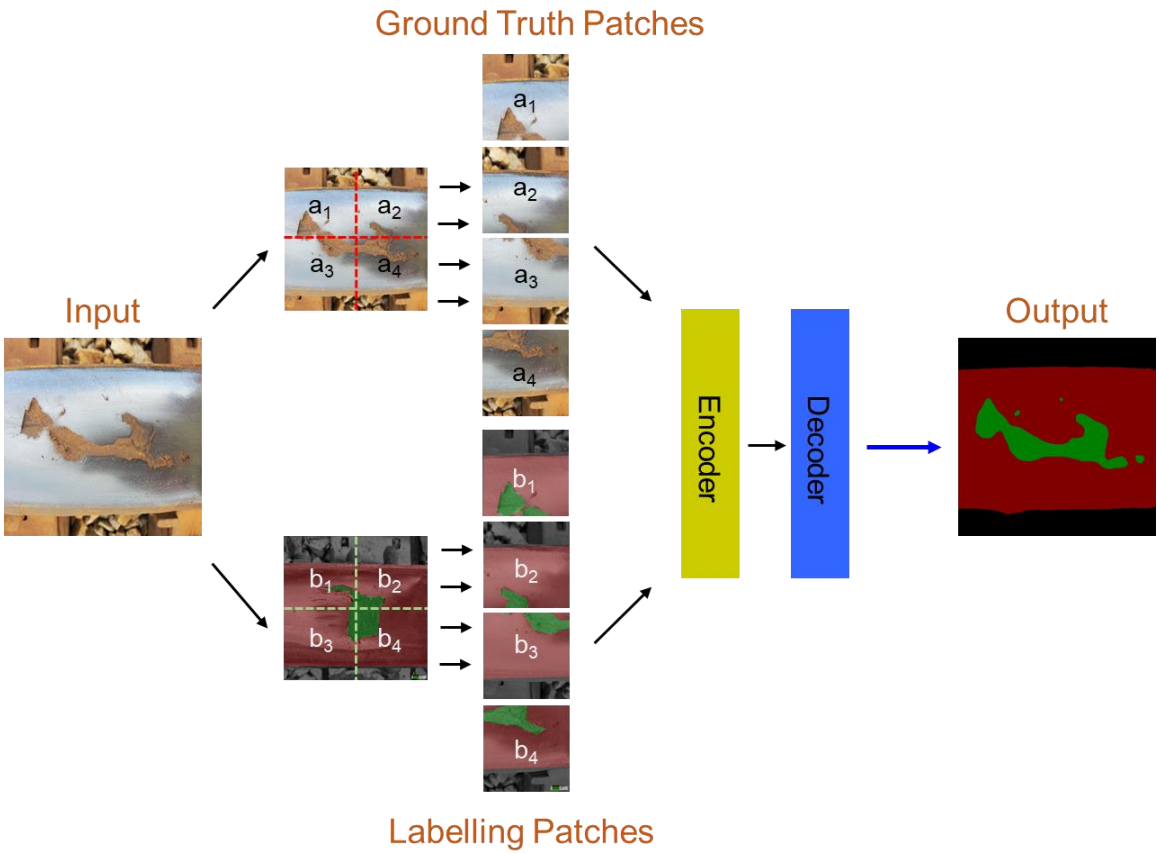


Figure 7.4. Patch-wise training strategy.

7.3.4 Lovárz Loss Function

A commonly used measure metric in the semantic segmentation task is Jaccard index which is also refereed as the intersection over union (IoU), which can bring appropriate relevance to small objects and the counting of false negatives. To further improve the semantic segmentation performance on a large dataset, Lovárz loss function (Berman, Triki, & Blaschko, 2018) is proposed for direct optimization of the mean IoU loss in a model. Specifically, given the object class c, the ground truth label y^*, and the prediction $\tilde{y}$, the Jaccard index $J_c(y^*, \tilde{y})$ can be defined in Equation (7-2).

$$J_c(y^*, \tilde{y}) = \frac{|\{y^* = c\} \cap \{\tilde{y} = c\}|}{|\{y^* = c\} \cup \{\tilde{y} = c\}|} \tag{7-2}$$

where the intersection ratio is in [0,1] and the special case that $0/0 = 1$.

Correspondingly, the loss function can be defined in Equation (7-3). In a semantic segmentation task with multiple classes, a *softmax* layer is used to map the probability distributions for the loss computation.

$$\Delta_{J_c}(y_i^*, \tilde{y}) = 1 - J_c(y_i^*, \tilde{y}) \tag{7-3}$$

Following the cross-entropy loss fashion, the pixel errors $m(c)$ for class $c \in C$ expressed by a vector can be defined as in Equation (7-4).

$$m_i(c) = \begin{cases} 1 - f_i(c) & \text{if } c = y_i^* \\ f_i(c) & \text{otherwise} \end{cases} \tag{7-4}$$

The loss surrogate is constructed to Δ_{J_c} and the Jaccard index for class c is computed based on the vector of errors $m(c)$ in Equation (7-5). Considering the multiple classes in the semantic segmentation task, the surrogates are averaged and the Lovász-Softmax loss can be defined in Equation (7-6).

$$loss(f(c)) = \overline{\Delta_{J_c}}(m(c)) \tag{7-5}$$

$$loss(f) = \frac{1}{|c|} \sum_{c \in C} \overline{\Delta_{J_c}}(m(c)) \tag{7-6}$$

7.3.5 Assessment of Rail Defect Severity

As mentioned earlier, there are some studies to detect the rail surface defect using either traditional image processing or deep learning-based technologies. Unfortunately, most of the earlier efforts focus on determining the existence of rail surface but few of them can quantify the severity. The reason behind is the complex and random shapes the defects, making the previous model not be able to accurately capture the shapes. To overcome the technical difficulty, we propose a pixel-aware approach to quantify the rail surface defect severity for field quantifications. Detailed procedures can be referred to Algorithm 7.1.

To assess the rail surface defect severity, all test images are processed by the proposed LDL model first to acquire semantic segmentation outputs. Afterward, the defect area $_L$, rail area $\tilde{R}$, the defect ratio $_r$ which is the defect area over the rail area, and the severity index ξ will be computed based on the pixel values. Based on the dataset in this study, the defect ratio $_r$ and even its distribution can be captured. According to the distribution of the calculated defect ratio $_r$ from the dataset in this study, the severity levels can be classified into three categories and a severity index ξ can be used to indicate the severity level of the inspected rail surface defects. With the semantic segmentation settings in this study, the pixel values of the background, the rail surface, and the defect part, equal to zero, one, and two, respectively. It is worth noting that only the number of pixels is considered for calculation and the label values are only used for the classification of different types of rail defects. Figure 7.5 illustrates an example of the distribution of the rail surface defect at the pixel level.

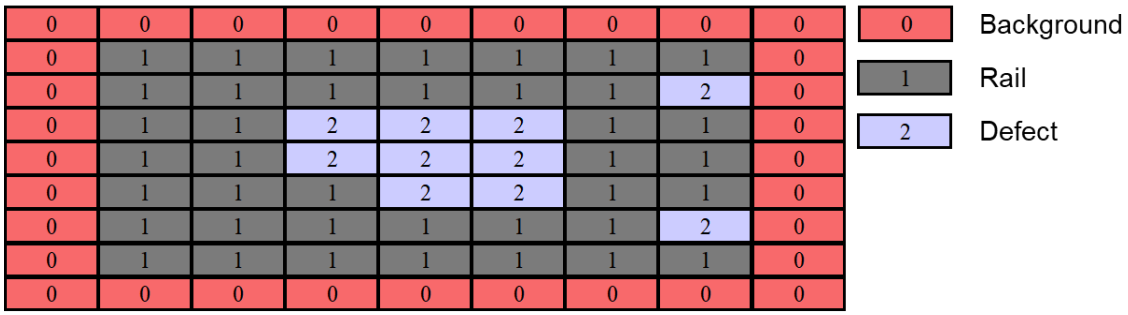


Figure 7.5. Illustration of the distribution of rail surface defects in the pixel level.

Algorithm 7.1 Assessment of rail defect severity

Input:

$_{P}$: Predicted images using LDL model

Output:

$_{r}$: The ratio of the defect area to the rail area,

ξ: Severity level of the input

1: Load predicted images

2: Build a matrix $_{M}$ to save defect area $_{L}$, rail area $\tilde{R}$, the ratio of the defect area to the rail area $_{r}$, and the severity index ξ

3: **for** each predicted image $_{i}$ in $_{I}$ **do**

Use the built-in *imread* function to read the predicted image;

Find the defect pixel dp and sum them up $\sum_{m,n} dp$;

Find the rail pixel $\tilde{r}p$ and sum them up $\sum_{m,n} \tilde{r}p$;

$L \leftarrow \sum_{m,n} dp$, $\tilde{R} \leftarrow \sum_{m,n} \tilde{r}p$, $r = L / \tilde{R}$;

Calculate the distribution of $_{r}$;

Set threshold of ξ for different severity levels;

Determine the severity level of each image;

Collect $_{L}$, $\tilde{R}$, $_{r}$, ξ and fill them into $_{M}$

end for

7.4 Experiments and results

In this chapter, a workstation equipped with quadra NVIDIA 2080 Ti GPUs is used for model training, testing, and validation. Specifically, the CUDA version is 10.2 and the cuDNN version was 7.5.2. All training work is based on the Pytorch library, which is developed by Facebook, Inc. MMSegmentation, an open-source semantic segmentation toolbox is adopted for all the experiments. Specifically, there are 20 epochs/20k iterations for each training. The optimizer which can control the weights and learning rate to reduce loss is set as the stochastic gradient descent (SGD). As we only perform the training with a single GPU, the learning rate is 0.0025 which equals 1/4 of the default settings in the MMSegmentation. The momentum which can accumulate the exponential decaying and improve both the training speed and accuracy is set as 0.9. The weight decay which can suppress any irrelevant components and improve generalization is set at 0.0005.

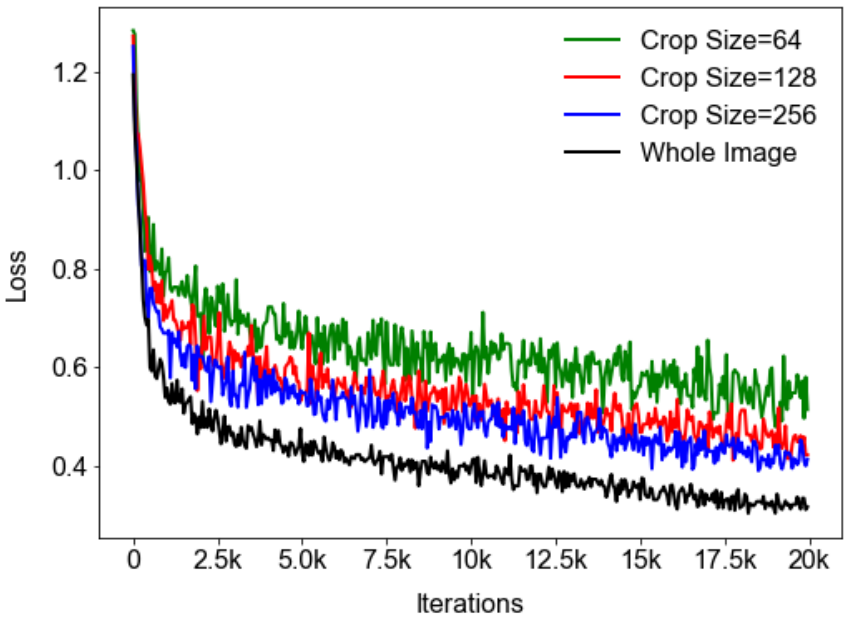
A total of 1,078 rail surface defect images are labeled using the annotation tool, labelme. All the images are taken from a fixed distance (17.8 cm or 7 inches) between the cell phone camera and the rail surface from the same angle. All surface defect images are collected along the track that are close to the Department of Civil and Environmental Engineering at the University of South Carolina. The training set and validation set are randomly divided with a ratio of 4:1.

Note that the validation set also serves as the testing set. To explicitly present the defect severity of rail track surface, two object instance classes (i.e., rail and defect) are included in the labeling process. In addition, to evaluate the training performance, the commonly used metric *mean intersection over union* (mIoU) is selected for semantic segmentation quantification. The results with our patch strategy and comparison with the state-of-the-art (SOTA) models can be referred to as follows.

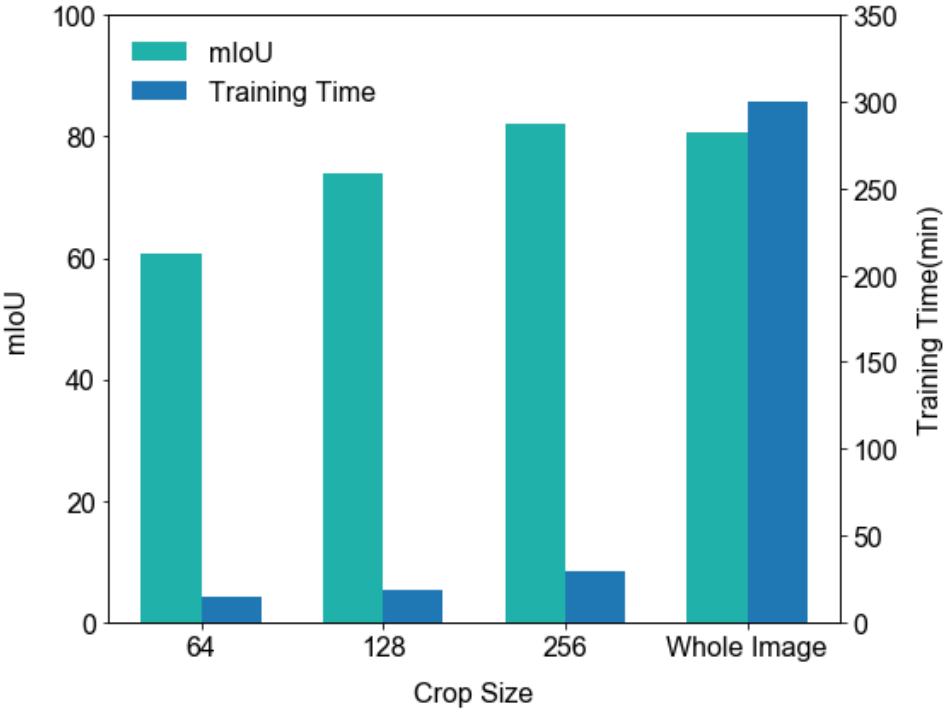
7.4.1 Training Results with Different Patch Sizes

In the experiments, to examine the impact of patch sizes, four different patch sizes including 64×64, 128×128, 256×256, and the whole image are included. Figure 7.6 presents the training results with different patches using the proposed LDL model. Specifically, Figure 7.6 (a) shows the loss changes over iterations. It needs to be mentioned that the loss here is the total loss including encode loss and decode loss. Additionally, it is clear that the crop size does impact the loss reduction in each training. With the increase in crop size, the loss decreases. Specifically, when the crop size is 64, the total loss scores the maximum value (0.5135) at the end of each training. When the crop size is 256, the total loss reaches a lower value (0.4129) at the end of each training. While, when the whole image is fed at its full-size directly with no preprocessing, the total loss achieves the minimum value (0.3163) which is 38.4% and 23.3% lower compared to the results with crop sizes are 64 and 256, respectively. In general, a low loss result indicates good training performance and Figure 7.6 (a) appears to suggest the whole image training is better than the patch strategy. To validate if the whole image training is actually better than the patch strategy or not, Figure 7.6 (b) is provided.

As shown in Figure 7.6 (b), there are two factors need to be considered. The first factor is *mIoU*, which is the indicator for accuracy. The definition can be referred to (Guo, Qian, Rizos, Suo, & Chen, 2021). The second factor is the *training time*, which is the indicator for training speed. With the patch strategy, it is easy to find the accuracy is associated with the loss for each patch size. As mentioned above, a lower loss indicates a higher accuracy. With a patch size of 64×64, LDL model reaches a relatively lower mIoU value which is 60.79. With a patch size of 128, the mIoU value (73.85) increases by 21.4% from that with a patch size of 64×64. When the patch size is 256×256, the maximum mIoU value (82.1) which is 11.1% and 35.1% higher compared to the case with a crop size of 64×64 and 128×128, respectively. From the perspective of the training speed, it is not hard to image a larger patch size would cause a lower training speed because a larger patch occupies more pixels and computational resource. With a patch size of 64×64, the training time is 15 minutes. When the patch sizes are 128×128 and 256×256, the training time is 19 minutes and 29 minutes, respectively. This shows a significant training time cost as the patch size increases. In addition, a full-size image does have good training accuracy due to the lower training loss shown in Figure 7.6 (a). However, the training time is 300 minutes, which is significantly higher than any case using a patch strategy. Clearly, the marginal benefit in accuracy is not worth the significant loss in training speed when compare the case using 256×256 patches and the whole image. To put it in a nutshell, this section proves the proposed patch strategy has promising balanced performance on loss reduction, training accuracy, and training speed.



(a)



(b)

Figure 7.6. Training results with different patches. (a) Loss results with different crop sizes, (b) mIoU and training time cost with different crop sizes.

7.4.2 Comparison with SOTA models

To fairly compare semantic segmentation performance with other SOTA models, all models are equipped with ResNet-18 backbone. The original models are FCN (Long, Shelhamer, & Darrell, 2015), GCNet (Cao, Xu, Lin, Wei, & Hu, 2019), PSPNet(Zhao, Shi, Qi, Wang, & Jia, 2017), DeepLabv3 (Chen et al., 2017), and DeepLabv3Plus (Chen et al., 2018), respectively. The proposed model and reconfigured models are named as LDL, FCN-Res18, GCNet-Res18, PSPNet-Res18, DeepLabv3-Res18, and DeepLabv3Plus-Res18, respectively. Four types of mIoU are used for comparison which are a total mIoU, IoU-background, IoU-rail, and IoU-defect. Corresponding results are shown in Figure 7.7. It needs to be clarified that the reason to clearly divide the IoU into different categories is to identify the specific contribution the LDL model can bring.

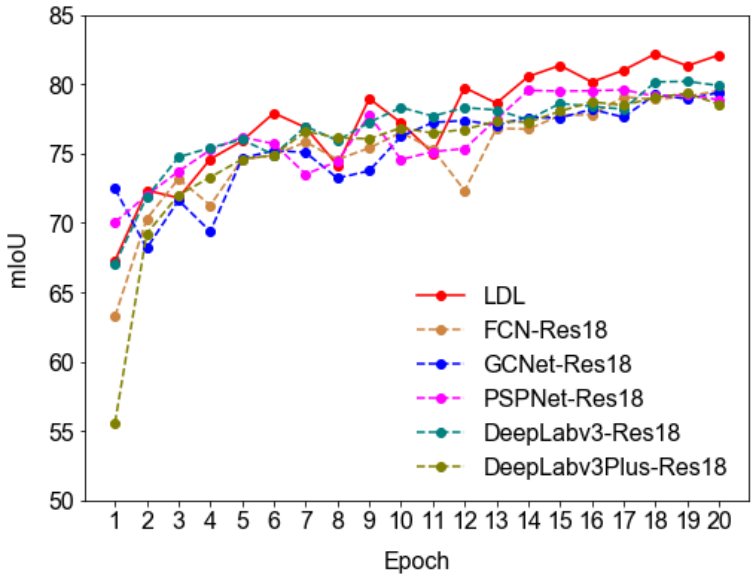
As shown in Figure 7.7 (a), the proposed model scores the highest mIoU (82.10) at the end of the model training. It is 3.2%, 3.4%, 4.1%, 2.8%, and 4.5% higher than FCN-Res18, GCNet-Res18, PSPNet-Res18, DeepLabv3-Res18, and DeepLabv3Plus-Res18, respectively. Since all training processes have experienced long iterations, it is reasonable to believe the proposed LDL outperforms other SOTA models on inspection accuracy using the index of mIoU. To identify the specific part the LDL model contributes the most, the IoU results of background, rail, and defect are plotted in Figure 7.7 (b), (c), and (d), respectively.

From Figure 7.7 (b), the proposed LDL model maximizes the IoU-Background to 86.86. It is 2.0%, 2.9%, 3.5%, 2.5%, and 3.3% higher than FCN-Res18, GCNet-Res18, PSPNet-Res18, DeepLabv3-Res18, and DeepLabv3Plus-Res18, respectively. In addition, the IoU-Background of DeepLabv3Plus-Res18 is very low in the initial stage compared

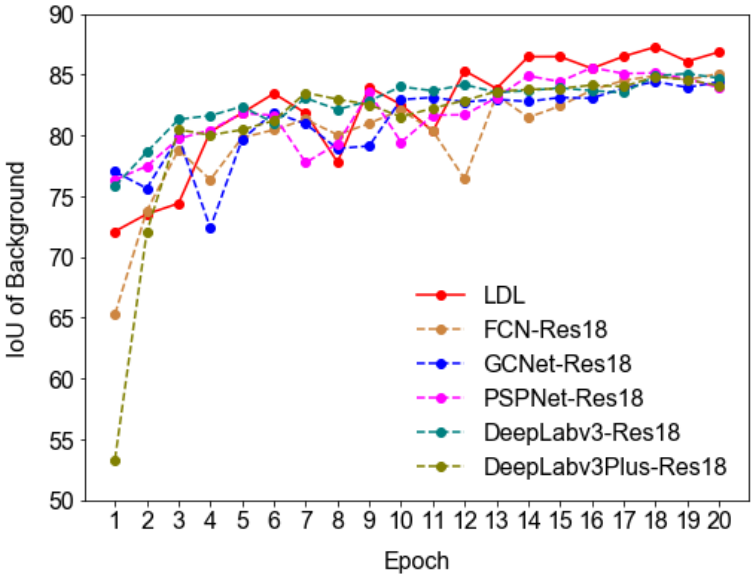
to other models. While, even having the same baseline of DeepLabv3Plus, the proposed LDL model does not show large fluctuations and increases steadily. As approaching 20 epochs, the proposed LDL model still increases while other models tend to be stable, indicating the proposed LDL model has better semantic segmentation results in the background.

Comparing the IoU-Rail results as shown in Figure 7.7 (c), the performances are close for all the models. The results of IoU-Rail from LDL, FCN-Res18, GCNet-Res18, PSPNet-Res18, DeepLabv3-Res18, and DeepLabv3Plus-Res18 are 93.5, 93.33, 93.13, 93.14, 93.23, and 93.13, respectively. Although all the values are close to each other, the proposed LDL model still has the maximum value. Figure 7.7 (c) indicates all models have promising results in detecting the rail.

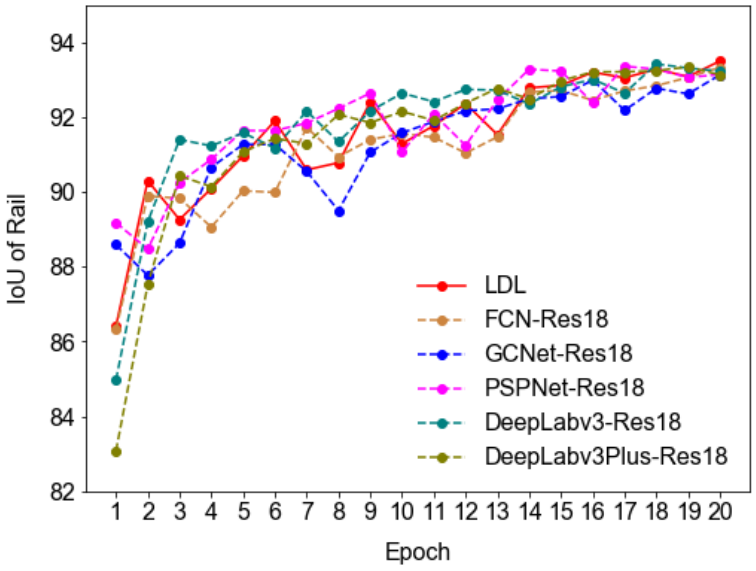
Figure 7.7 (d) gives the primary reason the proposed LDL model scores the maximum mIoU in Figure 7.7 (a). The proposed LDL model generally has higher values in IoU-Defect throughout the training process. It reaches the maximum IoU-Defect of 65.94, which is 9.6%, 8.7%, 10.7%, 6.8%, and 12.8% higher than FCN-Res18, GCNet-Res18, PSPNet-Res18, DeepLabv3-Res18, and DeepLabv3Plus-Res18, respectively. The curve of LDL model clearly stands out from the other models, indicating a significant improvement in the segmentation performance of defects. To improve the segmentation capability of rail surface defect is the main objective of this study and the results in Figure 7.7 shows the proposed model has met the research objective.


mIoU
Epoch
LDL
FCN-Res18
GCNet-Res18
PSPNet-Res18
DeepLabv3-Res18
DeepLabv3Plus-Res18

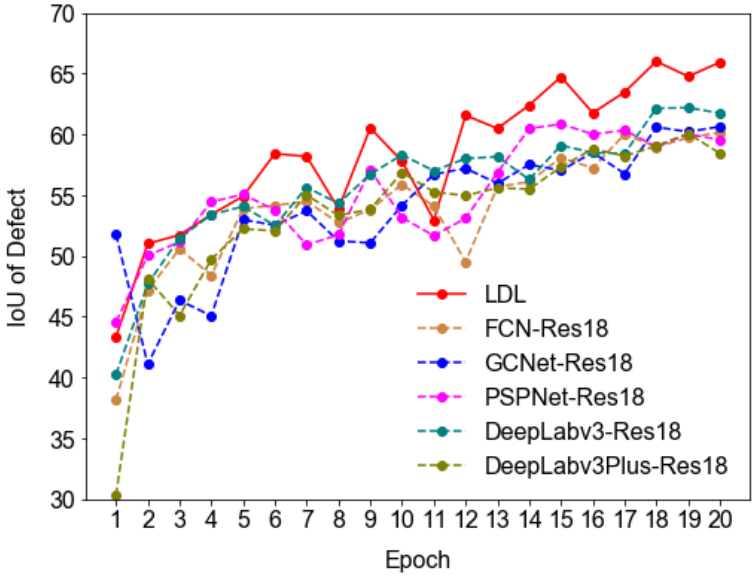
(a)


IoU of Background
Epoch
LDL
FCN-Res18
GCNet-Res18
PSPNet-Res18
DeepLabv3-Res18
DeepLabv3Plus-Res18

(b)



(c)



(d)

Figure 7.7. Comparison between different models. (a) mIoU, (b) IoU-Background, (c) IoU-Rail, (d) IoU-Defect.

7.4.3 Visualization and comparison of experimental results

In this section, visual evidence is provided to compare the semantic segmentation results from different models. Specifically, the most left column in Figure 7.8 is the ground truth. Besides the ground true, images from the left to the right, are the predicted results from LDL, FCN-Res18, GCNet-Res18, PSPNet-Res18, DeepLabv3-Res18, DeepLabv3Plus-Res18, respectively. Five randomly selected rail images with different shapes and densities of defects are given in Figure 7.8 examples for comparison. In the first row, there is a long, large, and connected defect in the rail surface. The proposed LDL model is able to well preserve the shape of the defect. However, FCN-Res18, GCNet-Res18, PSPNet-Res18, DeepLabv3-Res18 cannot outline the defect not the rail edges clearly. DeepLav3Plus-Res18 also performs well with this particular image.

For the second row in Figure 7.8, there are several individual surface defects. Similar to the result offered in the first row, LDL model and DeepLabv3Plus-Res18 can present the best results, but LDL model can preserve the rail edges better and preserve more details of the defects. While the other three models have poor performance. Similar observations can be made from the rest three rows in Figure 7.8. Overall, the proposed LDL model has promising performance to preserve the shapes of rail surface defects.

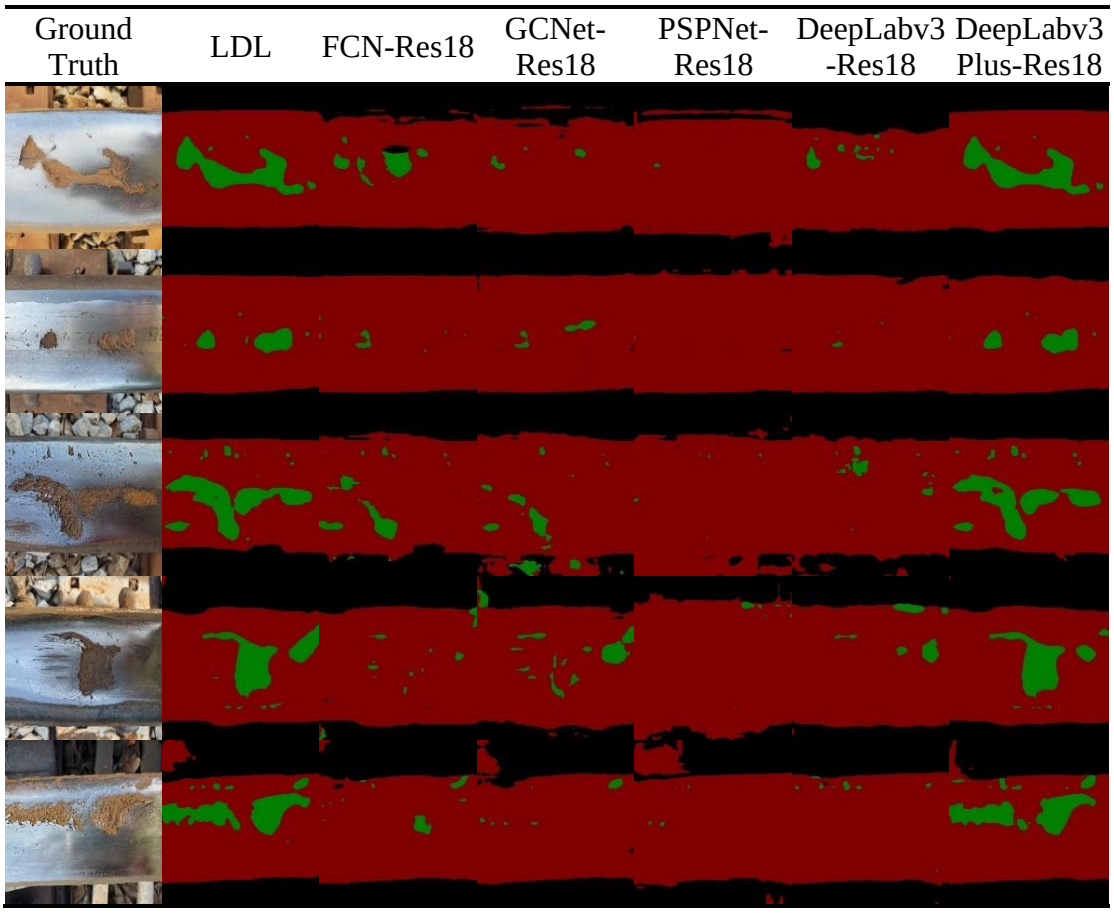
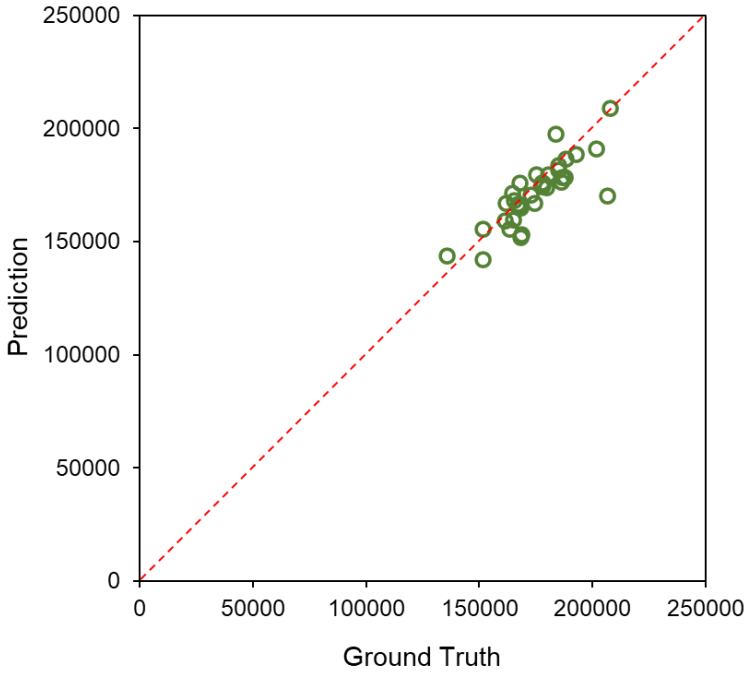


Figure 7.8. Visualized results of rail surface defects using different models.

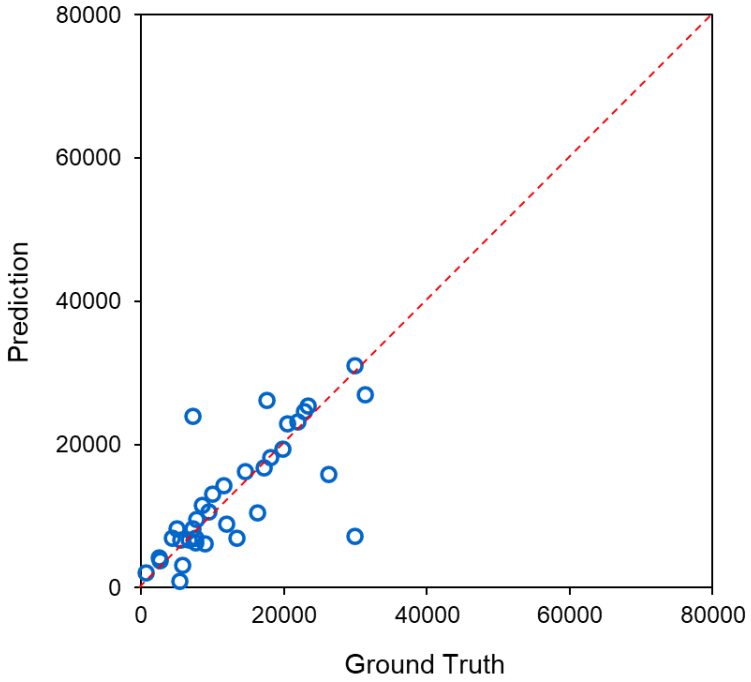
7.4.4 Rail severity assessment results

To able to preserve the shapes of the rail surface defect is the steppingstone to quantify the defect severity. By comparing the number of predicted pixels and ground truth pixels of the rail and the rail surface defects, Figure 7.9 (a) and (b) are plotted with diagonal lines using results from randomly selected 35 images processed by the LDL model. Ideally, if the data point aligns in the diagonal line, the prediction is 100% accurate. In Figure 7.9 (a), the predicted rail pixel numbers range from 142,013 to 209,122. Correspondingly, the ground truth pixels of the rail range from 135,760 to 207,684. All the 35 points are close to the diagonal line, indicating a good prediction result on the rail detection. Figure 7.9 (b) gives the comparison between the number of

predicted rail surface defect pixels and the ground truth rail surface defect pixels. Similar to the result shown in Figure 7.9 (a), most of the data points are concentrated along the diagonal line with only a few of them located a little bit away, indicating a reliable rail surface defect prediction result.



(a)



(b)

Figure 7.9. Comparison between the number of ground truth pixels and predicted pixels (a) rail surface defect pixels, (b) rail pixels.

After the numbers of pixels of the ground truth and the prediction are obtained, the rail surface defect severity can be assigned according to the ratio of the rail surface defect area over the rail surface area. It needs to be mentioned the severity level thresholds can be defined by the end-user to the model can provide default values according to the overall severity distribution based on all the processed images. Based on the limited data points in this study, the rail surface defect ratios are distributed as shown in Figure 7.10. Accordingly, 6% is selected as the interval between different severity levels as shown in Table 7.1. The proposed LDL model can process the track images and directly provide Table 7.1 to assist automatic track inspections.

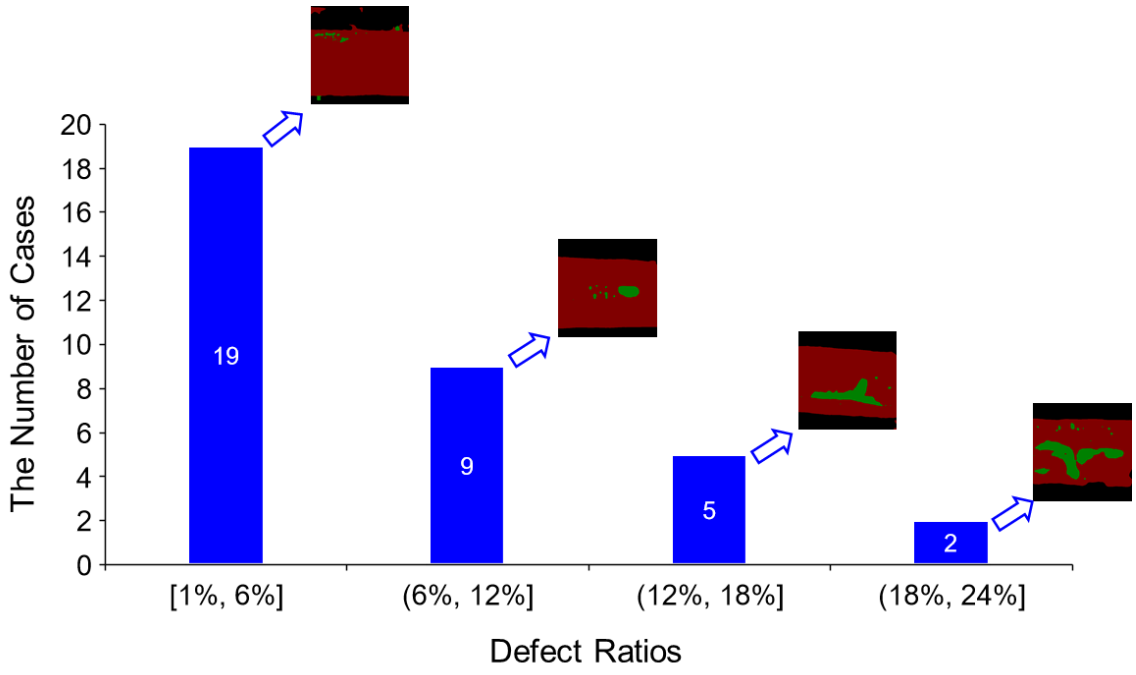


Figure 7.10. The distribution of rail surface defect ratios in our experiments.

Table 7.1. Rail surface defect severity levels.

Severity level	Rail surface defect ratio (RSFR)
Low	<6%
Medium	6%~12%
High	12%~18%
Super High	>18%

7.5 Summary

In this chapter, a novel lightweight semantic segmentation model for rail surface defects detection and quantification, namely the LDL model. The LDL model is developed based on the framework of the DeepLabv3Plus model. To balance both efficiency and accuracy for potential field inspection with limited computing resource, the backbone of ResNet-18 is integrated into the model architecture and a patch-wise training strategy is utilized. The Lovász loss function is adopted to tailor the proposed model for better semantic segmentation performance on complex rail surface defect shapes. A total of 1,078 images including two instance classes (i.e., rail and defect) are used for the model training, testing, and validation. Another five SOTA models are trained and compared with the proposed model using the metric of mIoU, IoU of background, IoU of rail, and IoU of the defect. To evaluate the rail surface defect severity, an algorithm is designed for the computation of the rail surface defect ratio between the rail defect and the rail at a pixel level.

Based on the limited results in this study, the proposed model can effectively and accurately detect and quantify rail surface defects and outperforms the other SOTA models. The improvement is mainly due to the proposed model can better preserve the shape of the defects.

Training results with different patches indicate the patch size of 256×256 has the optimal result on both mIoU and training speed. The comparison between the number of pixels of the ground truth label and predicted images proves that there is a good agreement between the prediction results using LDL and the ground truth label.

The proposed model has potential to be implemented to assist in automatic track inspection to provide quantitative information for rail maintenance planning

CHAPTER 8 CONCLUDING REMARKS AND RECOMMENDATIONS FOR FUTURE STUDY

8.1 Concluding remarks

This ongoing work contributes to the development and application of cutting-edge computer vision techniques including object detection, instance segmentation, and semantic segmentation in the railroad engineering domain. Leveraging the base model of the object detection, the new DTDNet has been proposed for dense traffic detection and tracking at the congested railroad grade crossings. The models including DTDNet and other SOTA models have been trained based on a crossing image database. The field detection and counting performance have been evaluated and compared under daytime, nighttime, and haze weather condition.

Regarding the rail track components inspection, an improved real-time instance segmentation model has been developed based on the original YOLACT model and Res2Net model. A customized rail track components image database including the rail, the clip, and the spike, has been established for training, validation, and testing. To improve the inference speed and assist the railroad industry to keep the inspection accuracy, a one-stage object detection model named YOLOV4-hybrid has been proposed based on the new design of activation functions in the framework. Different evaluation metrics including PR curve, F1 score, mAP, and inference time etc. have been considered.

With respect to the rail surface defect inspection, the optimized Mask R-CNN model and the proposed LDL have been trained, evaluated, and compared. The evaluation metrics including AP50, AP75, mIoU, IoU of background, IoU of rail, and IoU of the defect have been adopted to evaluate the training performance. The rail

surface defect severity with four different levels has been evaluated based on the developed algorithm.

Based on the limited experimental results, major findings of this work can be summarized as follows:

1. The TA module can bridge different forms of feature representations and model the long-range relationships. The learning to match-based detection head with the MLE procedure rather than the conventional hand-crafted IoU criterion can have a better detection accuracy.
2. Training and evaluation results with the dense traffic detection models indicate that DTDNet outperforms other models on both recall and precision rates. The proposed DTDNet achieves the best performance of detecting vehicle, train, and pedestrian.
3. The detection-based counting approach shows a promising counting results and accuracy. Regarding the three different errors (i.e., MAE, RMSE, and MRE), the testing on six randomly selected video clips indicates a robust and superior performance in the field.
4. The first attempt on railroad engineering even civil engineering using the proposed real-time instance segmentation model has been successfully deployed. The inference speed can achieve the real-time level which is over 30 FPS.
5. The proposed real-time instance model outperforms the original model and the classic instance segmentation model (i.e., Mask R-CNN) on both the detection accuracy of the bounding box and the mask.

6. The inspection performance of the proposed real-time instance segmentation model has been tested and compared on different illumination conditions. Experimental results confirm our model's robustness on low visibility conditions.
7. The new real-time object detection framework named YOLOv4-hybrid has been successfully developed with the combined activation function of Mish and Swish. The model comparison has been evaluated based on the YOLOv3 model and the YOLOv4 models with different activation functions.
8. The influence of two different factors on the track component detection including the illumination and the image size has been discussed. Experimental results confirm that the proposed YOLOv4-hybrid model is more sensitive to the illumination conditions rather than the image size.
9. The inspection performance with edited images including the missing or fake rail track components show that the proposed YOLOv4-hybrid model can have the robust results in the field.
10. Two backbones with three learning rates including 0.02, 0.01, and 0.005 have been used to optimize Mask R-CNN model on the customized rail surface defect image database. Experimental results indicate the configuration of ResNet101 with the learning rate of 0.005 achieves the best accuracy on both bounding box and mask predictions.
11. Compared to the traditional image processing algorithm (e.g., Otsu's method), the optimized Mask R-CNN has better performance on the rail surface defect inspection regardless of the different illuminations, rail orientations, and defect severities.

12. Regarding the proposed LDL model, under the optimized condition of the patch size (256 × 256), the proposed LDL model can have better performance on both mIoU and training speed.
13. The proposed LDL model can accurately detect and quantify the rail surface defects and outperforms SOTA models with different indicators such as mIoU (82.10%), IoU-background (86.86%), IoU-Rail (93.5%), and IoU-Defect (65.94).
14. Using limited random rail surface defect images, the visualization results confirm that the proposed model can depict a better outline and whole view of the background, the rail, and the defect.
15. Four rail surface defect severity levels have been proposed based on the limited image samples. Specifically, the low, medium, high, and super high severity levels correspond to the different rail surface defect ratios of less than 6%, between 6% and 12%, between 12% and 18%, and above 18%.

8.2 Recommendations for future study

This work provides a comprehensive study of developing and utilization of the cutting-edge computer vision models on the automatic railroad grade crossing monitoring and rail track inspection. The following contents are recommended for future research:

1. More environmental conditions include haze, fog, rain need to be considered in the dense traffic detection at the grade crossing.
2. The power efficiency of on-board dense traffic inspection needs to be considered in the deployment.
3. A light-weighted real-time instance segmentation or object detection model with better accuracy and inspection speed should be developed for future application due to the limited inspection time window in the railroad.
4. The semi-supervised learning should be considered in the future since there are limited labelling access and huge unknown data of rail track components.
5. Different types and shapes of the rail track components should be considered in the future. Also, the inspection of different damage levels of the rail track components is an interesting topic for the field application.
6. More standardized rail surface defect image data is needed. Building a rail surface defect image database can prompt more research efforts on this topic and benefit the railroad industry.

REFERENCES

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Isard, M. (2016). *Tensorflow: A system for large-scale machine learning.* Paper presented at the 12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16).

Adeli, H. (2001). Neural networks in civil engineering: 1989–2000. *Computer-Aided Civil and Infrastructure Engineering, 16*(2), 126-142.

Aloysius, N., & Geetha, M. (2017). *A review on deep convolutional neural networks.* Paper presented at the 2017 International Conference on Communication and Signal Processing (ICCSP).

Artagan, S. S., Ciampoli, L. B., D'Amico, F., Calvi, A., & Tosti, F. (2019). Non-destructive assessment and health monitoring of railway infrastructures. *Surveys in Geophysics*, 1-37.

Azimi, M., & Pekcan, G. (2020). Structural health monitoring using extremely compressed data through deep learning. *Computer-Aided Civil and Infrastructure Engineering, 35*(6), 597-614.

Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Bao, Y., Tang, Z., Li, H., & Zhang, Y. (2019). Computer vision and deep learning–based data anomaly detection method for structural health monitoring. *Structural Health Monitoring, 18*(2), 401-421.

Berman, M., Triki, A. R., & Blaschko, M. B. (2018). *The lovász-softmax loss: A tractable surrogate for the optimization of the intersection-over-union measure in neural networks.* Paper presented at the Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.

Bochkovskiy, A., Wang, C.-Y., & Liao, H.-Y. M. (2020). Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*.

Bottou, L. (2012). Stochastic gradient descent tricks. In *Neural networks: Tricks of the trade* (pp. 421-436): Springer.

Cai, Z., & Vasconcelos, N. (2018). *Cascade r-cnn: Delving into high quality object detection.* Paper presented at the Proceedings of the IEEE conference on computer vision and pattern recognition.

Cao, Y., Xu, J., Lin, S., Wei, F., & Hu, H. (2019). *Gcnet: Non-local networks meet squeeze-excitation networks and beyond.* Paper presented at the Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops.

Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., & Zagoruyko, S. (2020). *End-to-end object detection with transformers.* Paper presented at the European Conference on Computer Vision.

Chabot, F., Pham, Q.-C., & Chaouch, M. (2019). Lapnet: Automatic balanced loss and optimal assignment for real-time dense object detection. *arXiv preprint arXiv:1911.01149*.

Chen, K., Wang, J., Pang, J., Cao, Y., Xiong, Y., Li, X., . . . Xu, J. (2019). MMDetection: Open mmlab detection toolbox and benchmark. *arXiv preprint arXiv:1906.07155*.

Chen, L.-C., Papandreou, G., Schroff, F., & Adam, H. (2017). Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*.

Chen, L.-C., Zhu, Y., Papandreou, G., Schroff, F., & Adam, H. (2018). *Encoder-decoder with atrous separable convolution for semantic image segmentation.* Paper presented at the Proceedings of the European conference on computer vision (ECCV).

Dake, L. (2016). Mosier Oil Train Derailment Costs Near $9 Million. Retrieved from https://www.opb.org/news/series/oil-trains/mosier-derailment-cost-9-million/#:~:text=So%20far%2C%20an%20email%20obtained,derailment%20at%20about%20%248.9%20million.

Deutschl, E., Gasser, C., Niel, A., & Werschonig, J. (2004). *Defect detection on rail surfaces by a vision based system.* Paper presented at the IEEE Intelligent Vehicles Symposium, 2004.

Dung, C. V. (2019). Autonomous concrete crack detection using deep fully convolutional neural network. *Automation in Construction, 99*, 52-58.

Dutta, A., & Zisserman, A. (2019). *The VIA annotation software for images, audio and video.* Paper presented at the Proceedings of the 27th ACM international conference on multimedia.

Eger, S., Youssef, P., & Gurevych, I. (2019). Is it time to swish? Comparing deep learning activation functions across NLP tasks. *arXiv preprint arXiv:1901.02671*.

Estes, R. M., & Rilett, L. R. (2000). Advanced prediction of train arrival and crossing times at highway-railroad grade crossings. *Transportation research record, 1708*(1), 68-76.

Everingham, M., Van Gool, L., Williams, C. K., Winn, J., & Zisserman, A. (2010). The pascal visual object classes (voc) challenge. *International journal of computer vision, 88*(2), 303-338.

Faghih-Roohi, S., Hajizadeh, S., Núnez, A., Babuska, R., & De Schutter, B. (2016). *Deep convolutional neural networks for detection of rail surface defects.* Paper presented at the 2016 International joint conference on neural networks (IJCNN).

Fan, Q., Brown, L., & Smith, J. (2016). *A closer look at Faster R-CNN for vehicle detection.* Paper presented at the 2016 IEEE intelligent vehicles symposium (IV).

Fei, Y., Wang, K. C., Zhang, A., Chen, C., Li, J. Q., Liu, Y., . . . Li, B. (2019). Pixel-level cracking detection on 3D asphalt pavement images through deep-learning-based CrackNet-V. *IEEE Transactions on Intelligent Transportation Systems*.

Feng, H., Jiang, Z., Xie, F., Yang, P., Shi, J., & Chen, L. (2013). Automatic fastener classification and defect detection in vision-based railway inspection systems. *IEEE Transactions on Instrumentation and Measurement, 63*(4), 877-888.

Feng, J. H., Yuan, H., Hu, Y. Q., Lin, J., Liu, S. W., & Luo, X. (2020). Research on deep learning method for rail surface defect detection. *IET Electrical Systems in Transportation, 10*(4), 436-442.

FRA. (2018). Accident Causes. Retrieved from https://safetydata.fra.dot.gov/OfficeofSafety/publicsite/Query/inccaus.aspx

FRA. (2018a). Train accidents by cause from FRA F 6180.54. Retrieved from https://safetydata.fra.dot.gov/OfficeofSafety/publicsite/Query/inccaus.aspx

FRA. (2018b). Track and Rail and Infrastructure Integrity Compliance Manual. Retrieved from

https://railroads.dot.gov/sites/fra.dot.gov/files/fra_net/17940/CM%20Vol%20II%20Ch1%202018.pdf

FRA. (2020a). Accident Data as Reported by Railroads. Retrieved from https://railroads.dot.gov/accident-and-incident-reporting/overview-reports/accident-data-reported-railroads

FRA. (2020b). Accident Data, Reporting, and Investigations. Retrieved from https://railroads.dot.gov/railroad-safety/accident-data-reporting-and-investigations

FRA. (2020c). Rail Defect Detection. Retrieved from https://railroads.dot.gov/program-areas/track-and-structures/inspection-techniques

Gan, J., Li, Q., Wang, J., & Yu, H. (2017). A hierarchical extractor-based visual rail surface inspection system. *IEEE Sensors Journal*, *17*(23), 7935-7944.

Gan, J., Wang, J., Yu, H., Li, Q., & Shi, Z. (2018). Online rail surface inspection utilizing spatial consistency and continuity. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, *50*(7), 2741-2751.

Gao, S., Cheng, M.-M., Zhao, K., Zhang, X.-Y., Yang, M.-H., & Torr, P. H. (2019). Res2net: A new multi-scale backbone architecture. *IEEE transactions on pattern analysis and machine intelligence*.

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

Ghiasi, G., Lin, T.-Y., & Le, Q. V. (2018). *Dropblock: A regularization method for convolutional networks.* Paper presented at the Advances in Neural Information Processing Systems.

Gibert, X., Patel, V. M., & Chellappa, R. (2016). Deep multitask learning for railway track inspection. *IEEE Transactions on Intelligent Transportation Systems, 18*(1), 153-164.

Glenn Jocher, Y. K., guigarfr, Josh Veitch-Michaelis, perry0418,Ttayu, … Dustin Kendall. (2020). Ultralytics/yolov3: 43.1mAP@0.5:0.95 on COCO2014. Retrieved from https://doi.org/10.5281/zenodo.3785397

Guo, F., Qian, Y., Rizos, D., Suo, Z., & Chen, X. (2021). Automatic Rail Surface Defects Inspection Based on Mask R-CNN. *Transportation research record*, 03611981211019034.

Guo, F., Qian, Y., & Shi, Y. (2021). Real-time railroad track components inspection based on the improved YOLOv4 framework. *Automation in Construction, 125*, 103596.

Guo, F., Qian, Y., Wu, Y., Leng, Z., & Yu, H. (2021). Automatic railroad track components inspection using real-time instance segmentation. *Computer-Aided Civil and Infrastructure Engineering, 36*(3), 362-377.

Hackel, T., Stein, D., Maindorfer, I., Lauer, M., & Reiterer, A. (2015). *Track detection in 3D laser scanning data of railway infrastructure.* Paper presented at the 2015 IEEE International Instrumentation and Measurement Technology Conference (I2MTC) Proceedings.

Hardway. (2014). Train carrying oil, propane derails in Vandergrift. Retrieved from https://www.wtae.com/article/train-carrying-oilpropane-derails-in-vandergrift/7464992

Hardway, A. (2014). Train carrying oil, propane derails in Vandergrift. Retrieved from https://www.wtae.com/article/train-carrying-oil-propane-derails-in-vandergrift/7464992

He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2017). *Mask r-cnn.* Paper presented at the Proceedings of the IEEE international conference on computer vision.

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE transactions on pattern analysis and machine intelligence, 37*(9), 1904-1916.

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

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*.

Hsieh, M.-R., Lin, Y.-L., & Hsu, W. H. (2017). *Drone-based object counting by spatially regularized regional proposal network.* Paper presented at the Proceedings of the IEEE International Conference on Computer Vision.

Hu, X., Xu, X., Xiao, Y., Chen, H., He, S., Qin, J., & Heng, P.-A. (2018). SINet: A scale-insensitive convolutional neural network for fast vehicle detection. *IEEE Transactions on Intelligent Transportation Systems, 20*(3), 1010-1019.

Jang, K., An, Y. K., Kim, B., & Cho, S. (2020). Automated crack evaluation of a high-rise bridge pier using a ring-type climbing robot. *Computer-Aided Civil and Infrastructure Engineering*.

Jeong, J. H., Jo, H., & Ditzler, G. (2020). Convolutional neural networks for pavement roughness assessment using calibration-free vehicle dynamics. *Computer-Aided Civil and Infrastructure Engineering*.

Ji, Z., Kong, Q., Wang, H., & Pang, Y. (2019). *Small and Dense Commodity Object Detection with Multi-Scale Receptive Field Attention.* Paper presented at the Proceedings of the 27th ACM International Conference on Multimedia.

Jiao, L., Zhang, F., Liu, F., Yang, S., Li, L., Feng, Z., & Qu, R. (2019). A Survey of Deep Learning-Based Object Detection. *IEEE Access, 7*, 128837-128868.

Jie, L., Siwei, L., Qingyong, L., Hanqing, Z., & Shengwei, R. (2009). *Real-time rail head surface defect detection: A geometrical approach.* Paper presented at the 2009 IEEE International Symposium on Industrial Electronics.

Kang, D., & Cha, Y. J. (2018). Autonomous UAVs for structural health monitoring using deep learning and an ultrasonic beacon system with geo-tagging. *Computer-Aided Civil and Infrastructure Engineering, 33*(10), 885-902.

Kant, S. (2020). Learning Gaussian Maps for Dense Object Detection. *arXiv preprint arXiv:2004.11855*.

Karpathy, A. (2016). Cs231n convolutional neural networks for visual recognition. *Neural networks, 1*(1).

Kim, B., & Cho, S. (2018). Automated vision-based detection of cracks on concrete surfaces using a deep learning technique. *Sensors, 18*(10), 3452.

Kisantal, M., Wojna, Z., Murawski, J., Naruniec, J., & Cho, K. (2019). Augmentation for small object detection. *arXiv preprint arXiv:1902.07296*.

Kong, D., Gray, D., & Tao, H. (2006). *A viewpoint invariant approach for crowd counting.* Paper presented at the 18th International Conference on Pattern Recognition (ICPR'06).

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems, 25*, 1097-1105.

Leng, Z., & Al-Qadi, I. L. (2010). Railroad ballast evaluation using ground-penetrating radar: laboratory investigation and field validation. *Transportation Research Record, 2159*(1), 110-117.

Li, C., Yang, T., Zhu, S., Chen, C., & Guan, S. (2020). *Density map guided object detection in aerial images.* Paper presented at the Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops.

Li, Q., & Ren, S. (2012a). A real-time visual inspection system for discrete surface defects of rail heads. *IEEE transactions on instrumentation and measurement, 61*(8), 2189-2199.

Li, Q., & Ren, S. (2012b). A visual detection system for rail surface defects. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 42*(6), 1531-1542.

Li, S., Chang, F., & Liu, C. (2020). Bi-directional dense traffic counting based on spatio-temporal counting feature and counting-lstm network. *IEEE Transactions on Intelligent Transportation Systems*.

Li, S., Zhao, X., & Zhou, G. (2019). Automatic pixel-level multiple damage detection of concrete structure using fully convolutional network. *Computer-Aided Civil and Infrastructure Engineering, 34*(7), 616-634.

Li, W., Li, H., Wu, Q., Chen, X., & Ngan, K. N. (2019). Simultaneously detecting and counting dense vehicles from drone images. *IEEE Transactions on Industrial Electronics, 66*(12), 9651-9662.

Li, W., Wang, Z., Wu, X., Zhang, J., Peng, Q., & Li, H. (2020). *CODAN: Counting-driven Attention Network for Vehicle Detection in Congested Scenes.* Paper presented at the Proceedings of the 28th ACM International Conference on Multimedia.

Li, X., Wang, W., Hu, X., Li, J., Tang, J., & Yang, J. (2021). *Generalized focal loss v2: Learning reliable localization quality estimation for dense object detection.* Paper presented at the Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.

Li, X., Wang, W., Wu, L., Chen, S., Hu, X., Li, J., . . . Yang, J. (2020). *Generalized focal loss: Learning qualified and distributed bounding boxes for dense object detection.* Paper presented at the arXiv preprint arXiv:2006.04388.

Li, Y., Zhang, X., & Chen, D. (2018). *Csrnet: Dilated convolutional neural networks for understanding the highly congested scenes.* Paper presented at the Proceedings of the IEEE conference on computer vision and pattern recognition.

Liang, X. (2019). Image-based post-disaster inspection of reinforced concrete bridge systems using deep learning with Bayesian optimization. *Computer-Aided Civil and Infrastructure Engineering, 34*(5), 415-430.

Lin, T.-Y., Dollár, P., Girshick, R., He, K., Hariharan, B., & Belongie, S. (2017). *Feature pyramid networks for object detection.* Paper presented at the Proceedings of the IEEE conference on computer vision and pattern recognition.

Lin, T.-Y., Goyal, P., Girshick, R., He, K., & Dollár, P. (2017). *Focal loss for dense object detection.* Paper presented at the Proceedings of the IEEE international conference on computer vision.

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., . . . Zitnick, C. L. (2014). *Microsoft coco: Common objects in context.* Paper presented at the European conference on computer vision.

Liu, C., Li, N., Wu, H., & Meng, X. (2014). Detection of high-speed railway subsidence and geometry irregularity using terrestrial laser scanning. *Journal of Surveying Engineering, 140*(3), 04014009.

Liu, S., Qi, L., Qin, H., Shi, J., & Jia, J. (2018). *Path aggregation network for instance segmentation.* Paper presented at the Proceedings of the IEEE conference on computer vision and pattern recognition.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., & Berg, A. C. (2016). *Ssd: Single shot multibox detector.* Paper presented at the European conference on computer vision.

Liu, W., Salzmann, M., & Fua, P. (2019). *Context-aware crowd counting.* Paper presented at the Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.

Liu, X. (2017). Optimizing rail defect inspection frequency to reduce the risk of hazardous materials transportation by rail. *Journal of Loss Prevention in the Process Industries, 48*, 151-161.

Liu, X., Barkan, C. P., & Saat, M. R. (2011). Analysis of derailments by accident cause: evaluating railroad track upgrades to reduce transportation risk. *Transportation Research Record, 2261*(1), 178-185.

Liu, X., Dick, C., & Saat, M. (2014). *Optimizing ultrasonic rail defect inspection to improve transportation safety and efficiency.* Paper presented at the T&DI Congress 2014: Planes, Trains, and Automobiles.

Liu, X., Lovett, A., Dick, T., Rapik Saat, M., & Barkan, C. P. (2014). Optimization of ultrasonic rail-defect inspection for improving railway transportation safety and efficiency. *Journal of Transportation Engineering, 140*(10), 04014048.

Liu, X., Turla, T., & Zhang, Z. (2018). Accident-cause-specific risk analysis of rail transport of hazardous materials. *Transportation Research Record, 2672*(10), 176-187.

Liu, Z., Cao, Y., Wang, Y., & Wang, W. (2019). Computer vision-based concrete crack detection using U-net fully convolutional networks. *Automation in Construction, 104*, 129-139.

Long, J., Shelhamer, E., & Darrell, T. (2015). *Fully convolutional networks for semantic segmentation.* Paper presented at the Proceedings of the IEEE conference on computer vision and pattern recognition.

Lu, J., Liang, B., Lei, Q., Li, X., Liu, J., Liu, J., . . . Wang, W. (2020). SCueU-net: Efficient damage detection method for railway rail. *IEEE Access, 8*, 125109-125120.

Lv, Y., Duan, Y., Kang, W., Li, Z., & Wang, F.-Y. (2014). Traffic flow prediction with big data: a deep learning approach. *IEEE Transactions on Intelligent Transportation Systems, 16*(2), 865-873.

Ma, C., Hao, W., Xiang, W., & Yan, W. (2018). The impact of aggressive driving behavior on driver-injury severity at highway-rail grade crossings accidents. *Journal of Advanced Transportation, 2018*.

Maas, A. L., Hannun, A. Y., & Ng, A. Y. (2013). *Rectifier nonlinearities improve neural network acoustic models.* Paper presented at the Proc. icml.

Mandriota, C., Nitti, M., Ancona, N., Stella, E., & Distante, A. (2004). Filter-based feature selection for rail defect detection. *Machine Vision and Applications, 15*(4), 179-185.

Min, Y., Xiao, B., Dang, J., Yue, B., & Cheng, T. (2018). Real time detection system for rail surface defects based on machine vision. *EURASIP Journal on Image and Video Processing, 2018*(1), 1-11.

Misra, D. (2019). Mish: A self regularized non-monotonic neural activation function. *arXiv preprint arXiv:1908.08681*.

Oh, B. K., Glisic, B., Kim, Y., & Park, H. S. (2019). Convolutional neural network-based wind-induced response estimation model for tall buildings. *Computer-Aided Civil and Infrastructure Engineering, 34*(10), 843-858.

Pan, X., & Yang, T. (2020). Postdisaster image-based damage detection and repair cost estimation of reinforced concrete buildings using dual convolutional neural networks. *Computer-Aided Civil and Infrastructure Engineering, 35*(5), 495-510.

Park, P. Y., Jung, W. R., Yeboah, G., Rempel, G., Paulsen, D., & Rumpel, D. (2016). First responders' response area and response time analysis with/without grade crossing monitoring system. *Fire Safety Journal, 79*, 100-110.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., . . . Antiga, L. (2019). Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems, 32*, 8026-8037.

Perez-Ramirez, C. A., Amezquita-Sanchez, J. P., Valtierra-Rodriguez, M., Adeli, H., Dominguez-Gonzalez, A., & Romero-Troncoso, R. J. (2019). Recurrent neural network model with Bayesian training and mutual information for response prediction of large buildings. *Engineering Structures, 178*, 603-615.

Pohl, R., Erhard, A., Montag, H.-J., Thomas, H.-M., & Wüstenberg, H. (2004). NDT techniques for railroad wheel and gauge corner inspection. *NDT & e International, 37*(2), 89-94.

Qi, H., Xu, T., Wang, G., Cheng, Y., & Chen, C. (2020). MYOLOv3-Tiny: A new convolutional neural network architecture for real-time detection of track fasteners. *Computers in Industry, 123*, 103303.

Radhakrishnan, P. (2017). What are Hyperparameters ? and How to tune the Hyperparameters in a Deep Neural Network? Retrieved from https://towardsdatascience.com/what-are-hyperparameters-and-how-to-tune-the-hyperparameters-in-a-deep-neural-network-d0604917584a

Rajamäki, J., Vippola, M., Nurmikolu, A., & Viitala, T. (2018). Limitations of eddy current inspection in railway rail evaluation. *Proceedings of the Institution of Mechanical Engineers, Part F: Journal of Rail and Rapid Transit, 232*(1), 121-129.

Ramachandran, P., Zoph, B., & Le, Q. V. (2017). Searching for activation functions. *arXiv preprint arXiv:1710.05941*.

Ranjan, V., Le, H., & Hoai, M. (2018). *Iterative crowd counting.* Paper presented at the Proceedings of the European Conference on Computer Vision (ECCV).

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). *You only look once: Unified, real-time object detection.* Paper presented at the Proceedings of the IEEE conference on computer vision and pattern recognition.

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

Ren, S., He, K., Girshick, R., & Sun, J. (2016). Faster R-CNN: towards real-time object detection with region proposal networks. *IEEE transactions on pattern analysis and machine intelligence, 39*(6), 1137-1149.

Resendiz, E., Hart, J. M., & Ahuja, N. (2013). Automated visual inspection of railroad tracks. *IEEE Transactions on Intelligent Transportation Systems, 14*(2), 751-760.

Rizzo, P., Cammarata, M., Bartoli, I., di Scalea, F. L., Salamone, S., Coccia, S., & Phillips, R. (2010). Ultrasonic guided waves-based monitoring of rail head: laboratory and field tests. *Advances in Civil Engineering, 2010*.

Roberts, R., Rudy, J., Al-Qadi, I., Tutumluer, E., & Boyle, J. (2006). *Railroad ballast fouling detection using ground penetrating radar–a new approach based on scattering from voids.* Paper presented at the Ninth European Conference on NDT.

Rojas, R. (1996). The backpropagation algorithm. In *Neural networks* (pp. 149-182): Springer.

Ruder, S. (2016). An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*.

Russell, B. C., Torralba, A., Murphy, K. P., & Freeman, W. T. (2008). LabelMe: a database and web-based tool for image annotation. *International journal of computer vision, 77*(1-3), 157-173.

Saadat, S., Sherrock, E., & Zahaczewski, J. (2018). *Autonomous Track Geometry Measurement Technology Design, Development, and Testing*. Retrieved from

Shang, L., Yang, Q., Wang, J., Li, S., & Lei, W. (2018). *Detection of rail surface defects based on CNN image recognition and classification.* Paper presented at the 2018 20th International Conference on Advanced Communication Technology (ICACT).

Shen, F., Gan, R., & Zeng, G. (2016). *Weighted residuals for very deep networks.* Paper presented at the 2016 3rd International Conference on Systems and Informatics (ICSAI).

Sindagi, V. A., & Patel, V. M. (2017). *Cnn-based cascaded multi-task learning of high-level prior and density estimation for crowd counting.* Paper presented at the 2017

14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS).

Soleimani, S., Mousa, S. R., Codjoe, J., & Leitner, M. (2019). A comprehensive railroad-highway grade crossing consolidation model: a machine learning approach. *Accident Analysis & Prevention, 128*, 65-77.

Sony, S., Dunphy, K., Sadhu, A., & Capretz, M. (2021). A systematic review of convolutional neural network-based structural condition assessment techniques. *Engineering Structures, 226*, 111347.

Stewart, S., & Ahmed, R. (2002). Rolling contact fatigue of surface coatings—a review. *Wear, 253*(11-12), 1132-1144.

Sun, Z., Cao, S., Yang, Y., & Kitani, K. (2020). Rethinking Transformer-based Set Prediction for Object Detection. *arXiv preprint arXiv:2011.10881*.

Tan, G., Guo, Z., & Xiao, Y. (2019). *PA-RetinaNet: Path Augmented RetinaNet for Dense Object Detection.* Paper presented at the International Conference on Artificial Neural Networks.

Tom Roadcap, M. D., J. Riley Edwards. (2018). Broken Spikes in Premium Fastening Systems. Retrieved from https://railtec.illinois.edu/wp/wp-content/uploads/pdf-archive/4.3_Roadcap.pdf

Tzutalin, D. (2018). LabelImg.

Wada, K. (2016). Labelme: Image polygonal annotation with python. *GitHub repository*.

Wang, L., Zhuang, L., & Zhang, Z. (2019). Automatic detection of rail surface cracks with a superpixel-based data-driven framework. *Journal of Computing in Civil Engineering, 33*(1), 04018053.

Wang, M., & Cheng, J. C. (2020). A unified convolutional neural network integrated with conditional random field for pipe defect segmentation. *Computer-Aided Civil and Infrastructure Engineering, 35*(2), 162-177.

Wang, Z., & Liu, J. (2017). *A review of object detection based on convolutional neural network.* Paper presented at the 2017 36th Chinese Control Conference (CCC).

Wei, X., Yang, Z., Liu, Y., Wei, D., Jia, L., & Li, Y. (2019). Railway track fastener defect detection based on image processing and deep learning techniques: A comparative study. *Engineering Applications of Artificial Intelligence, 80*, 66-81.

Wu, D., Fan, Z., & Cui, M. (2021). Average up-sample network for crowd counting. *Applied Intelligence*, 1-13.

Wu, D., Lv, S., Jiang, M., & Song, H. (2020). Using channel pruning-based YOLO v4 deep learning algorithm for the real-time and accurate detection of apple flowers in natural environments. *Computers and Electronics in Agriculture, 178*, 105742.

Wu, R. T., Singla, A., Jahanshahi, M. R., Bertino, E., Ko, B. J., & Verma, D. (2019). Pruning deep convolutional neural networks for efficient edge computing in condition assessment of infrastructures. *Computer-Aided Civil and Infrastructure Engineering, 34*(9), 774-789.

Wu, Y., Qin, Y., Wang, Z., & Jia, L. (2018). A UAV-based visual inspection method for rail surface defects. *Applied sciences, 8*(7), 1028.

Wu, Y., Qin, Y., Wang, Z., Ma, X., & Cao, Z. (2020). Densely pyramidal residual network for UAV-based railway images dehazing. *Neurocomputing, 371*, 124-136.

Xu, X., Xu, S., Jin, L., & Song, E. (2011). Characteristic analysis of Otsu threshold and its applications. *Pattern recognition letters, 32*(7), 956-961.

Xu, Y., Yu, G., Wang, Y., Wu, X., & Ma, Y. (2017). Car detection from low-altitude UAV imagery with the faster R-CNN. *Journal of Advanced Transportation, 2017*.

Yanan, S., Hui, Z., Li, L., & Hang, Z. (2018). *Rail surface defect detection method based on yolov3 deep learning networks.* Paper presented at the 2018 Chinese Automation Congress (CAC).

Yang, F., Zhang, L., Yu, S., Prokhorov, D., Mei, X., & Ling, H. (2019). Feature pyramid and hierarchical boosting network for pavement crack detection. *IEEE Transactions on Intelligent Transportation Systems, 21*(4), 1525-1535.

Yang, J., Tao, W., Liu, M., Zhang, Y., Zhang, H., & Zhao, H. (2011). An efficient direction field-based method for the detection of fasteners on high-speed railways. *Sensors, 11*(8), 7364-7381.

Yao, Z., Cao, Y., Zheng, S., Huang, G., & Lin, S. (2020). Cross-iteration batch normalization. *arXiv preprint arXiv:2002.05712*.

Ye, J., Stewart, E., & Roberts, C. (2019). Use of a 3D model to improve the performance of laser-based railway track inspection. *Proceedings of the Institution of Mechanical Engineers, Part F: Journal of Rail and Rapid Transit, 233*(3), 337-355.

Yeum, C. M., Choi, J., & Dyke, S. J. (2019). Automated region-of-interest localization and classification for vision-based visual assessment of civil infrastructure. *Structural Health Monitoring, 18*(3), 675-689.

Yu, H., Li, Q., Tan, Y., Gan, J., Wang, J., Geng, Y.-a., & Jia, L. (2018). A coarse-to-fine model for rail surface defect detection. *IEEE Transactions on Instrumentation and Measurement, 68*(3), 656-666.

Yu, W., Yang, T., & Chen, C. (2021). *Towards resolving the challenge of long-tail distribution in uav images for object detection.* Paper presented at the Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision.

Zakar, F., & Mueller, E. (2016). Investigation of a Columbus, Ohio train derailment caused by fractured rail. *Case Studies in Engineering Failure Analysis, 7*, 41-49.

Zhang, A., Wang, K. C., Fei, Y., Liu, Y., Chen, C., Yang, G., . . . Qiu, S. (2019). Automated pixel-level pavement crack detection on 3D asphalt surfaces with a recurrent neural network. *Computer-Aided Civil and Infrastructure Engineering, 34*(3), 213-229.

Zhang, A., Wang, K. C., Li, B., Yang, E., Dai, X., Peng, Y., . . . Chen, C. (2017). Automated pixel-level pavement crack detection on 3D asphalt surfaces using a deep-learning network. *Computer-Aided Civil and Infrastructure Engineering, 32*(10), 805-819.

Zhang, C., Chang, C. c., & Jamshidi, M. (2020). Concrete bridge surface damage detection using a single-stage detector. *Computer-Aided Civil and Infrastructure Engineering, 35*(4), 389-409.

Zhang, D., Song, K., Xu, J., He, Y., Niu, M., & Yan, Y. (2020). MCnet: Multiple Context Information Segmentation Network of No-Service Rail Surface Defects. *IEEE Transactions on Instrumentation and Measurement, 70*, 1-9.

Zhang, H., Wang, Y., Dayoub, F., & Sünderhauf, N. (2020). Varifocalnet: An iou-aware dense object detector. *arXiv preprint arXiv:2008.13367*.

Zhang, K., Cheng, H., & Zhang, B. (2018). Unified approach to pavement crack and sealed crack detection using preclassification based on transfer learning. *Journal of Computing in Civil Engineering, 32*(2), 04018001.

Zhang, S., Chi, C., Yao, Y., Lei, Z., & Li, S. Z. (2020). *Bridging the gap between anchor-based and anchor-free detection via adaptive training sample selection.* Paper presented at the Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.

Zhang, S., Wu, G., Costeira, J. P., & Moura, J. M. (2017). *Understanding traffic density from large-scale web camera data.* Paper presented at the Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.

Zhang, X., Rajan, D., & Story, B. (2019). Concrete crack detection using context-aware deep semantic segmentation network. *Computer-Aided Civil and Infrastructure Engineering, 34*(11), 951-971.

Zhang, X., Story, B., & Rajan, D. (2021). Night Time Vehicle Detection and Tracking by Fusing Vehicle Parts From Multiple Cameras. *IEEE Transactions on Intelligent Transportation Systems*.

Zhang, X., Wan, F., Liu, C., Ji, R., & Ye, Q. (2019). Freeanchor: Learning to match anchors for visual object detection. *arXiv preprint arXiv:1909.02466*.

Zhang, X., Wang, Y., Wang, H., Zhang, Y., Wu, H., & Zhao, M. (2019). *Bend Detection of Bridge Chords in UAV Images via Region-Based Deep Semantic Segmentation*

Network. Paper presented at the 2019 IEEE 17th International Conference on Industrial Informatics (INDIN).

Zhao, H., Shi, J., Qi, X., Wang, X., & Jia, J. (2017). *Pyramid scene parsing network.* Paper presented at the Proceedings of the IEEE conference on computer vision and pattern recognition.

Zhao, H., Zhang, Y., Liu, S., Shi, J., Loy, C. C., Lin, D., & Jia, J. (2018). *Psanet: Point-wise spatial attention network for scene parsing.* Paper presented at the Proceedings of the European Conference on Computer Vision (ECCV).

Zhao, Z., Zheng, P., Xu, S., & Wu, X. (2019). Object detection with deep learning: A review. *IEEE transactions on neural networks and learning systems, 30*(11), 3212-3232.

Zheng, C., Zhu, S., Mendieta, M., Yang, T., Chen, C., & Ding, Z. (2021). 3D Human Pose Estimation with Spatial and Temporal Transformers. *arXiv preprint arXiv:2103.10455*.

Zheng, Z., Wang, P., Liu, W., Li, J., Ye, R., & Ren, D. (2020a). *Distance-IoU loss: Faster and better learning for bounding box regression.* Paper presented at the Proceedings of the AAAI Conference on Artificial Intelligence.

Zheng, Z., Wang, P., Liu, W., Li, J., Ye, R., & Ren, D. (2020b). *Distance-IoU Loss: Faster and Better Learning for Bounding Box Regression.* Paper presented at the AAAI.

Zhu, C., He, Y., & Savvides, M. (2019). *Feature selective anchor-free module for single-shot object detection.* Paper presented at the Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.

Zhu, X., Cheng, D., Zhang, Z., Lin, S., & Dai, J. (2019). *An empirical study of spatial attention mechanisms in deep networks.* Paper presented at the Proceedings of the IEEE/CVF International Conference on Computer Vision.

Zhuang, L., Wang, L., Zhang, Z., & Tsui, K. L. (2018). Automated vision inspection of rail surface cracks: A double-layer data-driven framework. *Transportation research part C: emerging technologies, 92*, 258-277.