This is the Pre-Published Version.

This is the peer reviewed version of the following article: Guo, F, Qian, Y, Wu, Y, Leng, Z, Yu, H. Automatic railroad track components inspection using real-time instance segmentation. Comput Aided Civ Inf. 2021; 36(3): 362–377, which has been published in final form at https://doi.org/10.1111/mice.12625. This article may be used for non-commercial purposes in accordance with Wiley Terms and Conditions for Use of Self-Archived Versions. This article may not be enhanced, enriched or otherwise transformed into a derivative work, without express permission from Wiley or by statutory rights under applicable legislation. Copyright notices must not be removed, obscured or modified. The article must be linked to Wiley's version of record on Wiley Online Library and any embedding, framing or otherwise making available the article or pages thereof by third parties from platforms, services and websites other than Wiley Online Library must be prohibited.

Automatic railroad track components inspection using real-time instance segmentation

Feng Guo, Yu Qian*

Department of Civil and Environmental Engineering, University of South Carolina, Columbia, SC, USA

Yunpeng Wu

School of Traffic and Transportation, Beijing Jiaotong University, Beijing, China

Zhen Leng

Department of Civil and Environmental Engineering, The Hong Kong Polytechnic University, Hong Kong, China

&

Huayang Yu

Department of Civil and Environmental Engineering, South China University of Technology, Guangdong, China

Abstract: *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 study. 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*

1 INTRODUCTION

Periodic inspection on the railroad track components is essential to maintain railroad safety and keep efficient operations. According to the Federal Railroad Administration (FRA) safety database (FRA, 2018a), there were 546 accidents associated with track defects in 2018, resulting in over $97 million financial loss and uncountable negative social impact. Out of the 546 accidents, there are 48 accidents caused by missing spikes, clips, and broken rails, leading to around $10 million damages. In the United States, FRA mandates regular track inspections as part of the early warning strategy (FRA, 2018b). Unfortunately, to date, most of the track inspection work, except for the track geometry measurement, is still very labor- and time-intensive, especially for inspecting missing track components. Due to the nature of the manual inspection, the results of inspecting

missing components could be very low-efficient and expensive. Based on Liu et al. (2014), except for the labor cost, the speed of inspection is around 15 to 20 mph and the average inspection cost per hour per vehicle is almost 300 dollars. Even with the manned-inspection vehicles, the expected annual inspection cost is easily millions of dollars, let alone a considerable amount of the missing track components is manually inspected by walking crew. This issue is more pronounced with the Class I railroad mainlines due to the saturated traffic volume and limited windows for inspection and maintenance, leading to accidents and even derailments. For example, 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 (Hardway, 2014). Therefore, the automated rail track components inspection is very meaningful for the railroad industry as pointed out by the FRA (Saadat et al., 2018).

In the past few years, many efforts have been devoted to developing automatic track inspection systems. Yang et al. (2011) demonstrated a Direction Field (DF)-based method for detecting absent fasteners. The DF was extracted as the feature element for recognition and the weight coefficient matrix was obtained based on the Linear Discriminant Analysis (LDA). Their detection approach performed well on low-resolution images with 320×240 pixels taken from high-speed railways, but the performance on high-resolution images was not mentioned. Li et al. (2012) reported a real-time visual inspection system (VIS) for discrete surface detects. VIS can detect rail defects with a speed of 216 km/h. But the system was not tested on other track components. Resendiz et al. (2013) used Gabor filters and the multiple signal classification (MUSIC) to perform periodicity detection on the track components. However, their proposed method cannot handle both 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 methods such as support-vector machine (SVM) and AdaboostSTM, STM was more robust and can achieve a higher precision on the detection of fasteners with different orientations and illumination conditions. Unfortunately, STM was very demanding in computational power, thus, it cannot perform an end-to-end test. Aytekin et al. (2015) developed a real-time railway fastener detection system using a high-speed 3-D laser range finder. On one hand, the system can detect the railway fasteners with a speed of 100 km/h. On the other hand, the system was not tested with other track components.

Earlier computer vision algorithms such as edge detection, Adaboost, SVM, and others, have been applied to detect track components and improved inspection efficiency. However, there is still room for improvement in terms of accuracy, efficiency, speed, end-to-end test, and robustness. Recently, the convolutional neural network (CNN), which automatically learns input features efficiently, has been extremely successful in computer vision development. It has become popular with the increased size of training data and improved computation power (Pan & Yang, 2020). Since the early 2000s, CNNs have dominated object detection, semantic segmentation, instance segmentation, multiple object tracking (MOT), and other similar work (LeCun et al., 2015). Several CNNs with high accuracy and efficiency, have been successfully developed and adopted in the field. In terms of object detection, there are mainly two categories, one-stage detector and two-stage detector (Jiao et al., 2019). YOLO (Redmon et al., 2016) is the typical one-stage detector that works in real-time with high inference speed, and 30 FPS is a key factor to determine whether a model can perform "live task" or not. For the two-stage detectors, Fast R-CNN (Girshick, 2015) is the most typical one, which has high accuracy on the object localization and recognition. In terms of instance segmentation, Mask R-CNN (He et al., 2017), a state-of-the-art model, is very accurate on object detection, whereas its processing speed is relatively low. To fill the gap, YOLACT (Bolya, Zhou, Xiao & Lee, 2019) separated instance segmentation into two parallel tasks and achieved over 30 FPS processing speed on MS COCO (Lin et al., 2014) with only one GPU.

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; Rafiei and Adeli, 2017; Cha et al., 2018; Rafiei and Adeli, 2018; Perez-Ramirez et al., 2019; Yeum et al., 2019; Cao et al., 2020). 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, Chang, & Jamshidi, 2020), and crack evaluation of a high-rise bridge by using a modified SegNet (Jang, An, Kim, & Cho, 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. (Jeong, Jo, & Ditzler, 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 (X. Zhang, Rajan, & Story, 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. (2015, 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.

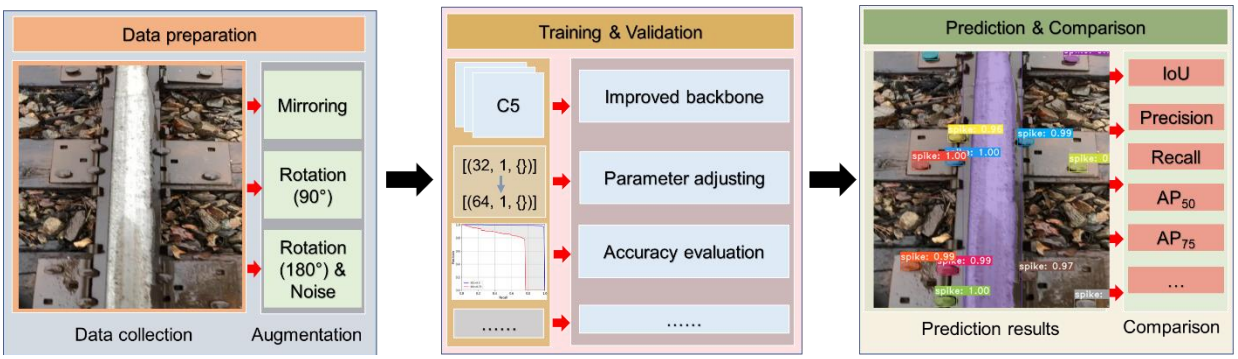


Figure 1 Content of this study

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. (2019) 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 study 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 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.

2 METHODOLOGY

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 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.

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

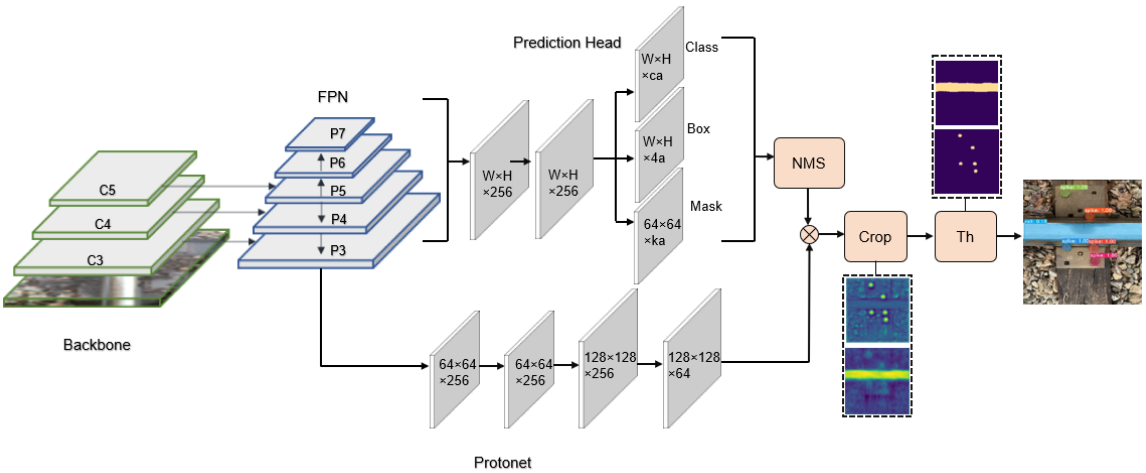


Figure 2 The main structure of the proposed models

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.

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 3 shows the bottleneck design for ResNet-50 and ResNet-101. As shown in Figure 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 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 2. Due to the limitation on the space, C1 and C2 are not plotted in Figure 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.

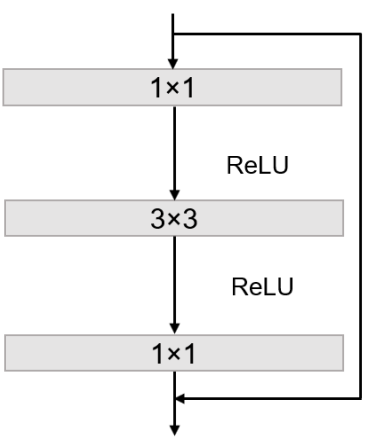


Figure 3 Structure of bottleneck design

2.2.2 Res2Net-50 & Res2Net-101

Res2Net (Gao et al. 2019) is a new backbone architecture which can improve the multi-scale representation capability at a granular level. Figure 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 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 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 (1). During the following model training and evaluation in this study, w is assigned to 26 and s is assigned to 4.

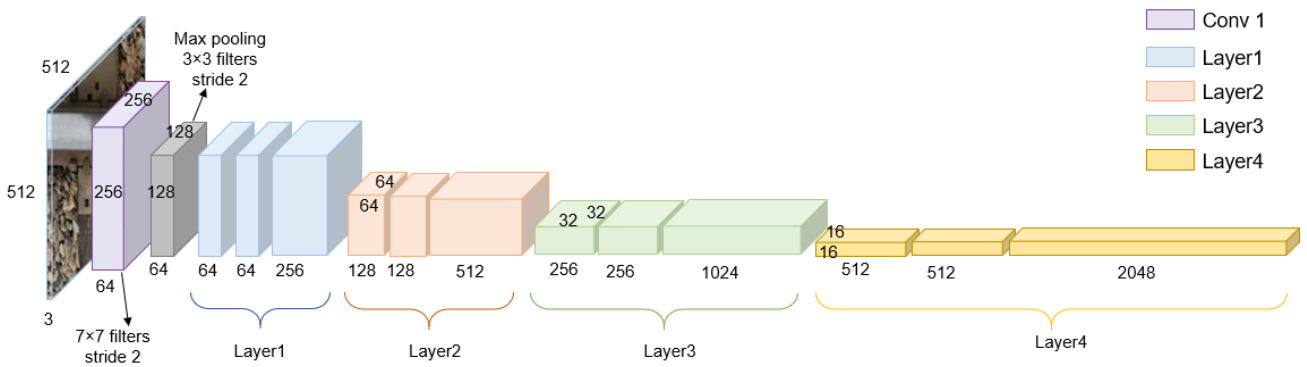


Figure 4 Main structure of ResNet-50

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

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 1 presents the detailed parameters of the proposed YOLACT-Res2Net-50 backbone. As shown in Figure 4, there are five stages: Conv1, Layer1, Layer2, Layer3, and Layer4. The main difference is the bottleneck structures shown in Figure 3 and Figure 5. It also can be referred in the filter size shown in Table 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.

2.3 FPN structure

To detect objects on multiple scales, Feature Pyramid Network (FPN) (Lin. 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 1, {*C*3, *C*4, *C*5} 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 {*P*3, *P*4, *P*5},

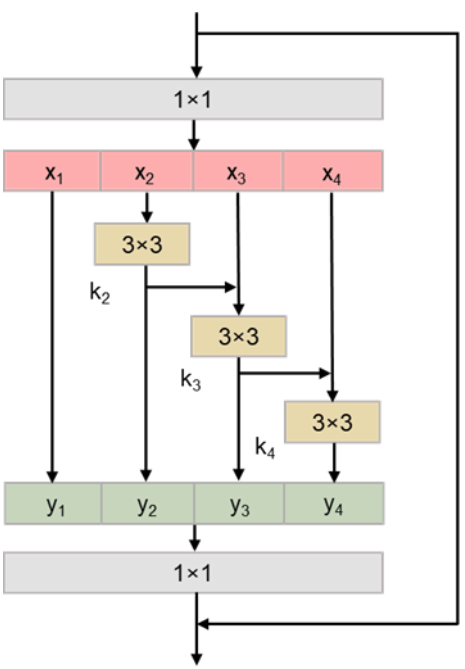


Figure 5 Structure of. Res2Net bottleneck (scale=4)

corresponding to {*C*3, *C*4, *C*5}. In the YOLACT type models, to increase the detection performance on the small objects, *P*5 is upsampled to *P*6 and *P*7 with one-fourth dimensions; meanwhile, *P*2 is omitted.

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 *P*3 layer in the FPN. The architecture of Protonet can be seen in Figure 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

Table 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

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 6. The high-resolution prototypes are beneficial for mask quality and detection performance on small objects.

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 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 (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.

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

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.

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 (3). For classification loss and box regression loss, the functions are shown in Equation (4) and (5). The corresponding weights for these three loss functions are 1, 1.5, and 6.125, respectively.

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

where M is the assembled masks, M_{gt} is the ground truth masks

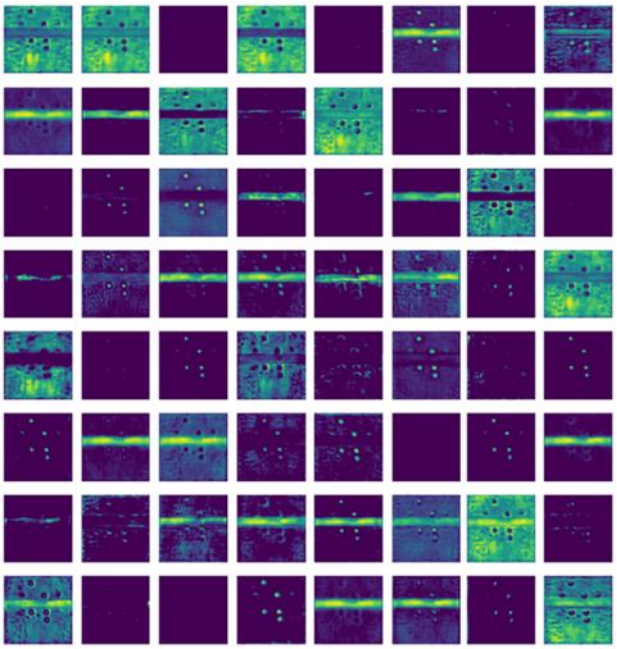


Figure 6 Prototype image generation

$$\hat{g}_j^w = \log(\frac{g_j^w}{d_i^w}) \qquad \hat{g}_j^h = \log(\frac{g_j^h}{d_i^h})$$
$$\hat{g}_j^{cx} = (g_j^{cx} - d_i^{cx})/d_i^w \qquad \hat{g}_j^{cy} = (g_j^{cy} - d_i^{cy})/d_i^h \qquad (4)$$
$$L_{loc}(x,l,g) = \sum_{i\in Pos}^{N} \sum_{m\in\{cx,cy,w,h\}} x_{ij}^k smooth_{L1}(l_i^m - \hat{g}_j^m)$$

where l is the predicted box, g is the ground truth, (c_x, c_y), w, and h is the center, width, and the height of the default bounding box (d). N is the number of matched defaulted boxes. If N=0, then the loss is 0. x_{ij}^k= {1,0} to be an indicator of matching the i-th default bounding box and j-th ground truth bounding box of category k.

$$L_{conf}(x,c) = -\sum_{i\in pos}^{N} x_{ij}^p \log(\hat{c}_i^p) - \sum_{i\in Neg} \log(\hat{c}_i^0)$$
$$\text{where } \hat{c}_i^p = \frac{\exp(c_i^p)}{\sum_p \exp(c_i^p)} \qquad (5)$$

where c is the softmax loss over multiple classes confidences.

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 3.

3.1 Data set 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 (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 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 (Zhang, Rajan & Story, 2019; Li, Zhao & Zhou, 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 3.

(a) (b)

Figure 7 Example of original jpg image and label result (a) Ground truth (b) instance label visualization

3.2 Training and validation

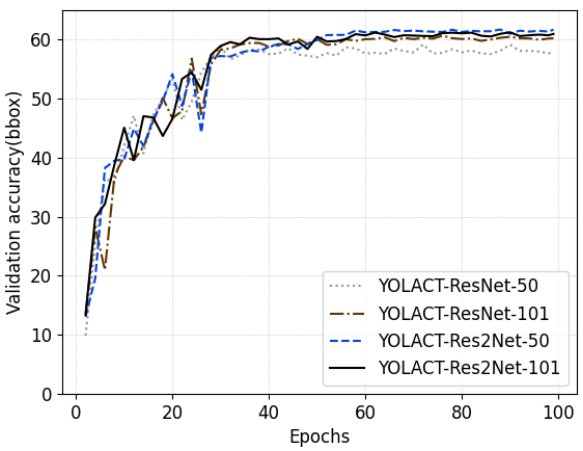
Transfer learning is a convenient timesaving method to train deep learning models. Since multiple models need to be trained and evaluated, transfer learning, other than training individual models from scratches, is employed in the test. Pre-trained weights for the backbones of the proposed models and 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 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.

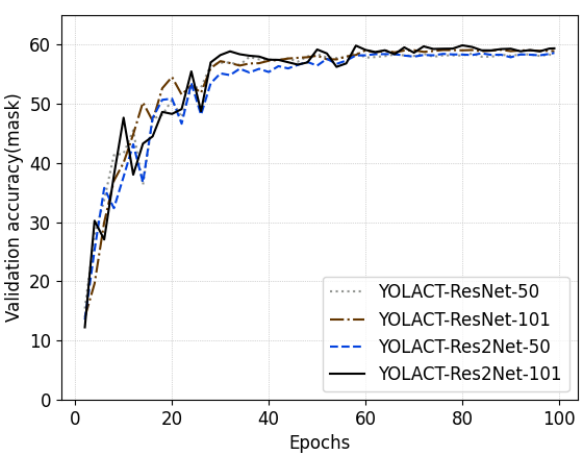
To configure and expedite the model training, the Pytorch library developed by Facebook and the packages of CUDA 10.2 with Cudnn 7.6.5 developed by NVIDIA are included in this study. All the training processes are accomplished in a lab server. The server system is CentOS 7.2 with Inter i7 CPU. The GPU is NVIDIA 1080 Ti with driver 440.33. The training time for each model takes around 1 to 1.5 hours. Figure 8 shows the validation accuracy of a test model on a randomly selected training set. Figure 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 8 (b), the validation accuracies of the mask are close among

Table 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



(a)



(b)

Figure 8 Representative validation accuracy of original YOLACT models and proposed YOLACT-Res2Net-50 and YOLACT-Res2Net-101

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.

3.3 Detection performance 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

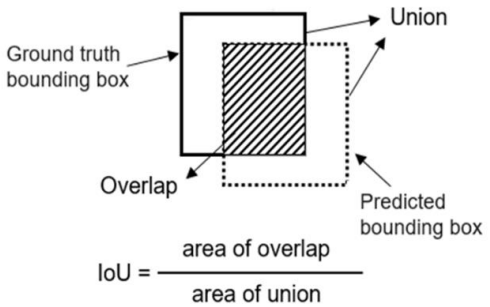


Figure 9 The definition of IoU

measures the overlap between the predicted boundary and the ground truth. Figure 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 (6) and (7). From our training logs, the representative PR curves of the improved models and the original models can be seen in Figures 11 and 12. Table 3 shows the COCO mAP results of all trained models.

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

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

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 10 and 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 10 and Figure 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 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. While, 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 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.

3.4 Influence of the light condition

In the field practice, environmental conditions are complex, and the track components are relatively small, making visual inspections very challenging. Besides, the

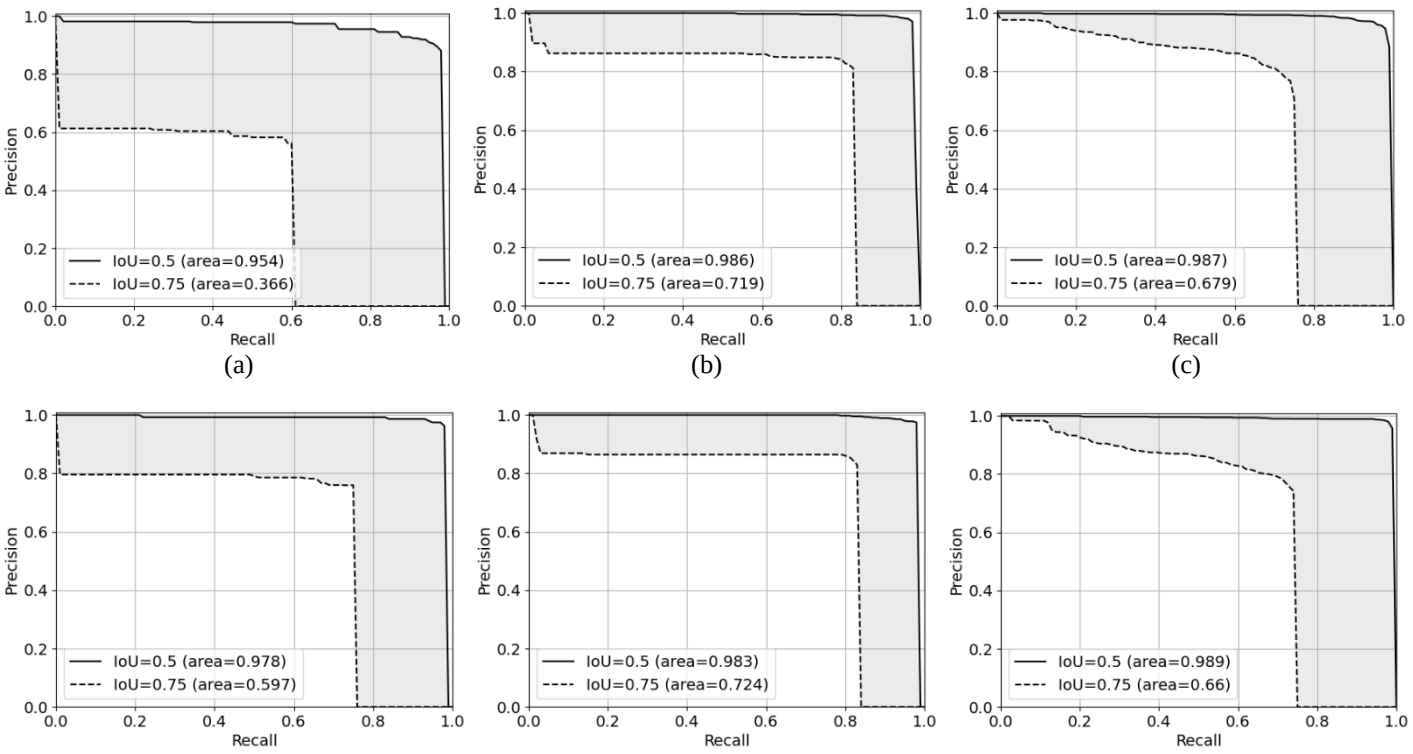


Figure 10 Representative precision-recall curves of YOLACT-ResNet-50 and YOLACT-ResNet-101 on each category. (a)-(c): rail, clip, and spike on YOLACT-ResNet-50; (d)-(f): rail, clip, and spike on YOLACT-ResNet-101

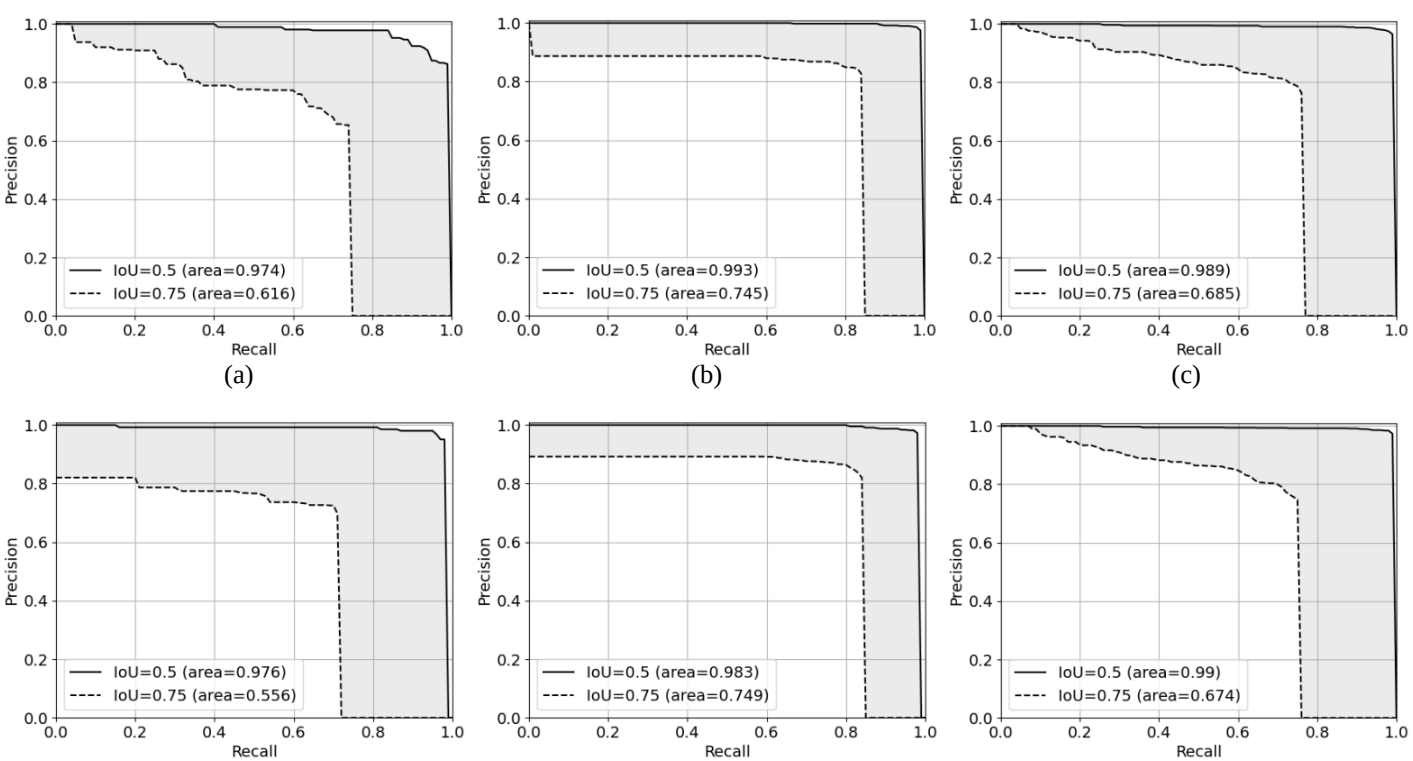


Figure 11 Representative precision-recall curves of YOLACT-Res2Net-50 and YOLACT-Res2Net-101 on each category. (a)-(c): rail, clip, and spike on YOLACT-Res2Net-50; (d)-(f): rail, clip, and spike on YOLACT-Res2Net-101

Table 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.

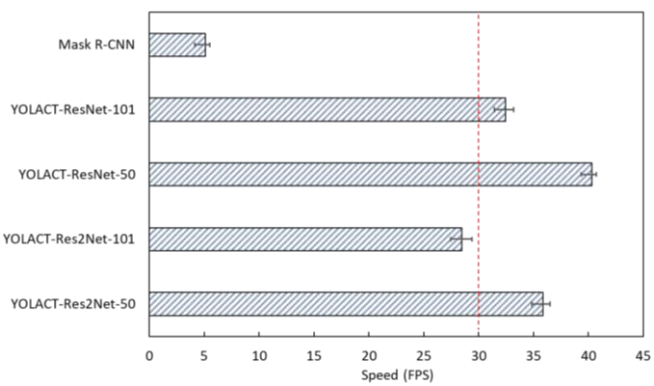


Figure 12 Detection speed of different models

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 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 13. Furthermore, the specific image presented in Figure 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 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.

4 CONCLUSIONS

This study 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

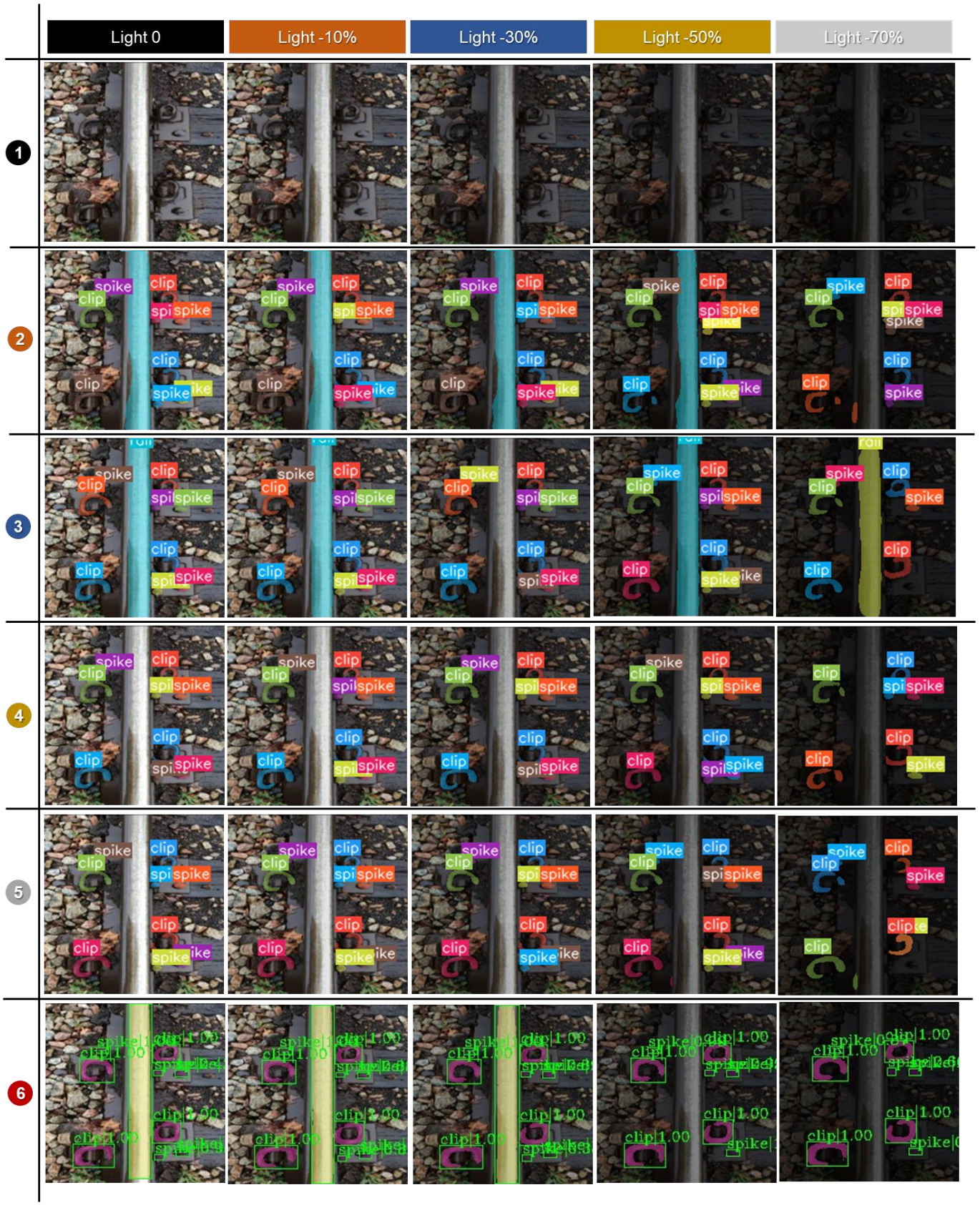


Figure 13 Representative detection results on the different light condition 1: Ground truth; 2: YOLACT-Res2Net-50; 3: YOLACT-Res2Net-101; 4: YOLACT-ResNet-50; 5: YOLACT-ResNet-101; 6: Mask R-CNN

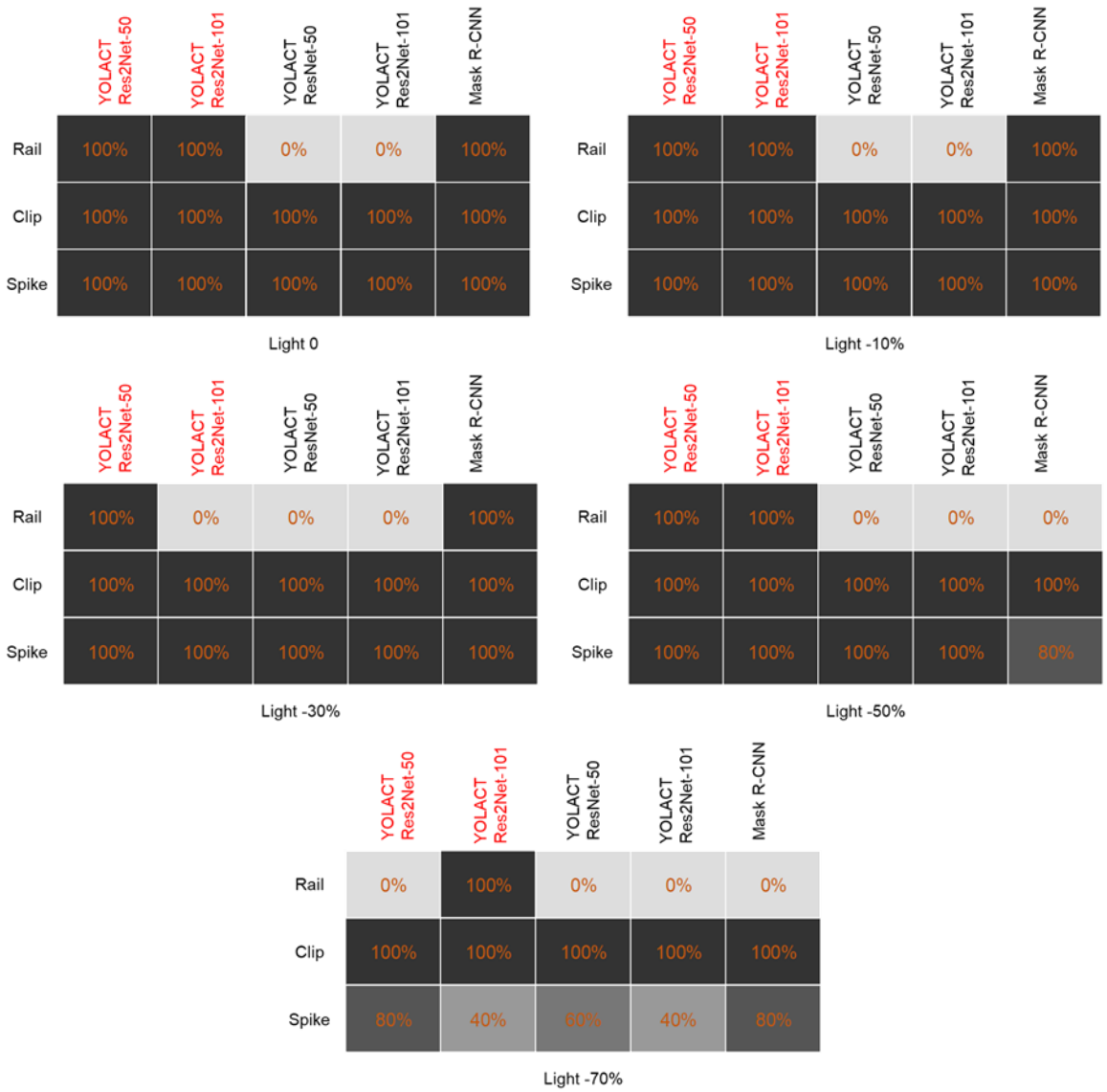


Figure 14 Detection accuracy under different illuminations.

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, our study 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 study 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.

ACKNOWLEDGEMENTS

This study is partially supported by the faculty startup funding provided by the College of Engineering and Computing at the University of South Carolina.

REFERENCES

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

Aytekin, Ç., Rezaeitabar, Y., Dogru, S., & Ulusoy, I. (2015). IEEE Transactions on Systems, Man, and Cybernetics: Systems, 45(7), 1101-1107.

Bolya, D., Zhou, C., Xiao, F., & Lee, Y. J. (2019). YOLACT: real-time instance segmentation. Paper presented at the Proceedings of the IEEE International Conference on Computer Vision.

Cao, R., Leng, Z., Hsu, S. C., & Hung, W. T. (2020). Modelling of the pavement acoustic longevity in Hong Kong through machine learning techniques. Transportation Research Part D: Transport and Environment, 83, 102366.

Cha, Y. J., Choi, W., Suh, G., Mahmoudkhani, S. & Büyüköztürk, O. (2018), Autonomous Structural Visual Inspection Using Region-Based Deep Learning for Detecting Multiple Damage Types, Computer-Aided Civil and Infrastructure Engineering, 33(9), 731-747.

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.

Dake, L. (2016). Mosier Oil Train Derailment Costs Near $9 Million.

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.

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

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.

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.

Giben, X., Patel, V. M., & Chellappa, R. (2015, September). Material classification and semantic segmentation of railway track images with deep convolutional neural networks. In 2015 IEEE International Conference on Image Processing (ICIP) (pp. 621-625). IEEE.

Girshick, R. (2015). Fast R-CNN. Paper presented at the Proceedings of the IEEE international conference on computer vision.

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. (2016). Deep residual learning for image recognition. Paper presented at the Proceedings of the IEEE conference on computer vision and pattern recognition.

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.

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.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. nature, 521(7553), 436-444.

Li, Q., & Ren, S. (2012). A real-time visual inspection system for discrete surface defects of rail heads. IEEE Transactions on Instrumentation and Measurement, 61(8), 2189-2199.

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.

Liu, X., Dick, C. T., & Saat, M. R. (2014). Optimizing ultrasonic rail defect inspection to improve transportation safety and efficiency. In T&DI Congress 2014: Planes, Trains, and Automobiles (pp. 765-774).

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., 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.

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.

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.

Rafiei, M. H. & Adeli, H. (2017), A Novel Machine Learning-Based Algorithm to Detect Damage in High-Rise Building Structures, The Structural Design of Tall and Special Buildings, 26(18), e1400.

Rafiei, M. H. & Adeli, H. (2018), A Novel Unsupervised Deep Learning Model for Global and Local Health Condition Assessment of Structures, Engineering Structures, 156, 598-607.

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.

Resendiz, E., Hart, J. M., & Ahuja, N. (2013). Automated visual inspection of railroad tracks. IEEE transactions on intelligent transportation systems, 14(2), 751-760.

Saadat, S., Sherrock, E., & Zahaczewski, J. (2018). Autonomous Track Geometry Measurement Technology Design, Development, and Testing (No. DOT/FRA/ORD-18/06). United States. Federal Railroad Administration. Office of Research, Development, and Technology.

Sawadisavi, S., Edwards, J. R., Resendiz, E., Hart, J. M., Barkan, C. P., & Ahuja, N. (2009). Machine-vision inspection of railroad track. Paper presented at the Proceedings of the TRB 88th Annual Meeting, Washington, DC.

Tom Roadcap, M. D., J. Riley Edwards. (2018). Broken Spikes in Premium Fastening Systems.

Wada, K. (2016). labelme: Image Polygonal Annotation with Python.

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.

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.

Xia, Y., Xie, F., & Jiang, Z. (2010). Broken railway fastener detection based on adaboost algorithm. Paper presented at the 2010 International Conference on Optoelectronics and Image Processing.

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.

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.

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, 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.