DOI: 10.1111/mice.12500

ORIGINAL ARTICLE

COMPUTER-AIDED CIVIL AND INFRASTRUCTURE ENGINEERING WILEY

Concrete bridge surface damage detection using a single-stage detector

Chaobo Zhang | **Chih-chen Chang** | **Maziar Jamshidi**

Department of Civil and Environmental Engineering, Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong

Correspondence
Chih-chen Chang, Department of Civil and Environmental Engineering, Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong.
Email: cechang@ust.hk

Abstract

Early and timely detection of surface damages is important for maintaining the functionality, reliability, and safety of concrete bridges. Recent advancement in convolution neural network has enabled the development of deep learning-based visual inspection techniques for detecting multiple structural damages. However, most deep learning-based techniques are built on two-stage, proposal-driven detectors using less complex image data, which could be restricted for practical applications and possible integration within intelligent autonomous inspection systems. In this study, a faster, simpler single-stage detector is proposed based on a real-time object detection technique, You Only Look Once (YOLOv3), for detecting multiple concrete bridge damages. A field inspection images dataset labeled with four types of concrete damages (crack, pop-out, spalling, and exposed rebar) is used for training and testing of YOLOv3. To enhance the detection accuracy, the original YOLOv3 is further improved by introducing a novel transfer learning method with fully pretrained weights from a geometrically similar dataset. Batch renormalization and focal loss are also incorporated to increase the accuracy. Testing results show that the improved YOLOv3 has a detection accuracy of up to 80% and 47% at the Intersection-over-Union (IoU) metrics of 0.5 and 0.75, respectively. It outperforms the original YOLOv3 and the two-stage detector Faster Region-based Convolutional Neural Network (Faster R-CNN) with ResNet-101, especially for the IoU metric of 0.75.

1 | INTRODUCTION

Highway bridges are an integral part of the transportation system that form the backbone of the modern metropolises. The functionality, reliability, and safety of these bridges are important to the well-being of society. For this reason, it is necessary to detect defects that might appear on these bridges as early as possible to prevent any further losses in their structural capacity and durability. Among all nondestructive evaluation techniques that can be used to identify and monitor defects, visual inspection remains the most adopted approach, despite that the internal condition of a structural element cannot be assessed by merely using visual inspection techniques and other in-depth methods should be brought into the process for a more comprehensive investigation. Besides, defects on the surface are the most observable indicator of possible structural deterioration or damage. It was concluded that surface defects are a good indicator for the general condition of a structural member and are a key part of many visual condition assessment manuals (Koch, Georgieva, Kasireddy, Akinci, & Fieguth, 2015).

Despite its popularity, manual visual inspection of bridges could be labor intensive and hazardous in locations with low accessibility. The results of visual inspection could also

be subjective and unreliable. These drawbacks bring about the needs of developing more objective and autonomous approaches that require little or no direct human intervention. Computer vision-based inspection techniques, facilitated with images acquired from unmanned aerial vehicles, have been proposed under such a circumstance (Ellenberg, Kontsos, Bartoli, & Pradhan, 2014; H. Kim, Sim, & Cho, 2015). Once images have been collected from a bridge, plenty of image processing techniques can be used to extract relevant information for defect detection, classification, and assessment (Koch et al., 2015). These techniques vary in the level of complexity and the way they process images. Among them, machine learning approaches are more advanced tools that utilize the extracted features of images to do specific tasks, such as classification, regression, clustering, and so on. These machine learning-based approaches have been utilized for detecting various types of defects, including cracks (Nishikawa, Yoshida, Sugiyama, & Fujino, 2012; Prasanna, Dana, Gucunski, & Basily, 2012; Zalama, Gómez-García-Bermejo, Medina, & Llamas, 2014), spalling (Dawood, Zhu, & Zayed, 2017; German, Brilakis, & DesRoches, 2012), and corrosion (O'Byrne, Schoefs, Ghosh, & Pakrashi, 2013). Although the machine learning techniques showed significant improvement on efficiency and robustness over the traditional image processing techniques, these methods were still based on handcrafted low-level features and require pre- and post-processing. This process can hardly be generalized and may not be applicable to actual images with large variations, such as those acquired from robotic platforms.

Recently, deep learning methods have gained a lot of attention and have been applied in various areas of science and engineering. Deep learning takes advantage of a large-scale database and allows computational models that are composed of multiple processing layers to learn representations of data (LeCun, Bengio, & Hinton, 2015). Owing to this excellent capability, various data forms have been exploited for deep learning-based structural health monitoring (Cha, Choi, & Büyüköztürk, 2017; Dai & Cao, 2017; Rafiei & Adeli, 2017, 2018). Particularly, in the area of image processing using deep learning techniques, one of the most important advancements in recent years is the development of convolutional neural networks (CNNs; Krizhevsky, Sutskever, & Hinton, 2012). CNNs can enhance the object detection and classification capability of image processing techniques as they can learn the appropriate features automatically from the training data without the need of prior image processing or feature extraction. CNN was employed for crack segmentation by classifying each pixel as either damaged or intact (X. Yang, Li, et al., 2018; A. Zhang et al., 2017). To find the extent of a defect in a large image, a primitive approach is to do raster scanning of the image with a fixed-size sliding window and then apply the trained CNN on each small window patch. Following this idea, Cha et al. (2017) proposed a deep CNN model to classify whether a concrete crack was presented in each image patch. L. Yang et al. (2017) fine-tuned the VGG-16 model (Simonyan & Zisserman, 2014) to classify two types of defect: cracks and spalling. The challenge of such sliding window-based approach, however, is to find a proper window size when dealing with defects of various scales. Also, this approach comes with a high computational cost because CNN classifier must be applied many times for every single window in each image.

To improve the efficiency of detecting and locating multiple objects, the CNN-based object detectors can be enhanced by directly predicting object regions. In terms of network architecture, these object detectors can be divided into two main categories: two-stage and single-stage detectors. Region-based CNN (R-CNN) is a two-stage architecture that uses region proposals instead of sliding windows to find objects in an image (Girshick, Donahue, Darrell, & Malik, 2014). The R-CNN approach uses a selective search approach (Uijlings, Van De Sande, Gevers, & Smeulders, 2013) to generate region proposals and then extracts features for each potential bounding box separately for classification and box regression. However, the region proposals generation step using the selective search is still slow and has limited accuracy (Ren, He, Girshick, & Sun, 2015). Also, due to the significant repetitive computation for overlapping regions, the R-CNN approach can be inefficient and time-consuming. To overcome this issue, Ren et al. (2015) proposed the Faster R-CNN by replacing the traditional handcrafted proposals generation step with a region proposal network (RPN). These two-stage detectors have recently been used to detect structural defects. For example, I. Kim, Jeon, Baek, Hong, and Jung (2018) combined the R-CNN with morphological postprocessing to detect and quantify cracks in a concrete bridge. The detected crack region from the R-CNN was, however, fragmented and the detection accuracy was affected by the extracted potential crack regions. Cha, Choi, Suh, Mahmoudkhani, and Büyüköztürk (2018) used the Faster R-CNN architecture to detect five types of surface damage in concrete and steel structures. They showed that a high accuracy could be achieved for a dataset of images with distinctive features. Li, Yuan, Zhang, and Yuan (2018) also employed a modified Faster R-CNN network to identify three types of concrete defects using a dataset that included various background information and relatively small defects. Although above studies demonstrated the suitable accuracy of two-stage detectors, their detection was not very fast due to the incorporation of region proposal as the intermediate step. The speed of the detection process can become important as the number of images increases. This is particularly true for inspection videos, where processing every frame can become overwhelmingly slow. On the other hand, in a broader scope, high-speed detection is essential for the development of real-time robotic inspection systems, which

seems to be the future direction of the industry (Lattanzi & Miller, 2017).

Due to the above limitations, single-stage detectors were proposed as an end-to-end network that combined object classification and localization in one single convolution network. There are two popular models in this category: the Single Shot MultiBox Detector (SSD) (Liu et al., 2016) and the You Only Look Once (YOLO) (Redmon, Divvala, Girshick, & Farhadi, 2016). Both SSD and YOLO removed the proposal-generating step and predicted multiple bounding boxes and class probabilities simultaneously. Hence, their processing speed was considerably faster than that of the two-stage approaches. Maeda, Sekimoto, Seto, Kashiyama, and Omata (2018) applied the SSD to detect road surface defects in real time. They demonstrated that the SSD could obtain relatively high accuracy for some classes of defect. However, some researchers found that directly locating objects using such single-stage detectors could compromise the accuracy of the detection (Li et al., 2018; L. Wang et al., 2017). In particular, the earliest version of YOLO could hardly detect objects with unusual aspect ratios or configurations (Redmon et al., 2016). To remedy this shortcoming, Redmon and Farhadi (2017, 2018) introduced several improvements to enhance the performance of single-stage detectors, including applying the batch normalization (BN) (Ioffe & Szegedy, 2015), using good candidate anchors to predict bounding boxes in YOLOv2, and employing the feature pyramid network (FPN) (Lin et al., 2017) and Darknet-53 backbone in YOLOv3. Results showed that the latest version of YOLO (i.e., YOLOv3) could achieve a comparable accuracy to the two-stage detector, Faster R-CNN, without sacrificing its computational efficiency.

Besides the development of network architectures, many studies have been devoted to the improvement of the training and testing process, making deep learning methods more applicable to different scenarios. First, training a deep network requires large annotated image datasets to achieve high predictive accuracy. However, for some applications, such as identifying and classifying structural damages, acquiring relevant images and annotating them for training is not easy. In such a scenario, it is common to pretrain a CNN model on a very large common object dataset and then use the pretrained weights either as an initialization or a fixed feature extractor for the task of interest (Oquab, Bottou, Laptev, & Sivic, 2014; Yosinski, Clune, Bengio, & Lipson, 2014). This process is called transfer learning (TL) and was widely used in deep learning-based structural damage recognition (Gao & Mosalam, 2018; Gopalakrishnan, Khaitan, Choudhary, & Agrawal, 2017; Kucuksubasi & Sorguc, 2018; Li et al., 2018). Also, it is well known that normalizing the input data makes training faster and enables very deep networks to converge (LeCun, Bottou, Orr, & Müller, 2012) because the change in the distributions of layers' inputs poses a problem that the layers need to continuously adapt to the new distributions. To remedy this problem, different normalization layers had been developed and adopted in deep networks to ease optimization, such as BN (Ioffe & Szegedy, 2015), layer normalization (Ba, Kiros, & Hinton, 2016), and instance normalization (Ulyanov, Vedaldi, & Lempitsky, 2016). Moreover, the class imbalance encountered during training of object detectors would decrease the model performance as the vast number of easily classified examples may overwhelm the training process. Several effective ways were proposed to address the class imbalance issue, including resampling the training examples (Ren et al., 2015; Shrivastava, Gupta, & Girshick, 2016) and reweighting the loss for different classes (Lin, Goyal, Girshick, He, & Dollár, 2018). In addition, many other efforts contributed to the optimization of algorithms for specific task (Molina-Cabello, Luque-Baena, López-Rubio, & Thurnhofer-Hemsi, 2018; Torres, Galicia, Troncoso, & Martínez-Álvarez, 2018; P. Wang & Bai, 2018) and the reduction of models' memory and computational cost (Ortega-Zamorano, Jerez, Gómez, & Franco, 2017; Sandler, Howard, Zhu, Zhmoginov, & Chen, 2018).

The objective of this study is to investigate the applicability of the state-of-the-art single-stage detector, YOLOv3, for identifying multiple classes of defects for concrete bridges and to improve its performance in terms of detection accuracy. The original YOLOv3 is first configured and trained using an image dataset that consists of 2,206 actual images containing four major types of defects of concrete bridges. A new TL method with fully pretrained weights from a geometrically similar dataset is then proposed to mitigate some issues associated with the insufficient training data and the low localization accuracy. Additional techniques, including the batch renormalization (BR) and the focal loss (FL), are also adopted to further improve the accuracy of detection. The performance of the improved YOLOv3 is validated and compared with that of the original YOLOv3 and the representative two-stage detector, Faster R-CNN with ResNet-101. The rest of the article is organized as follows. Section 2 provides an overview of the YOLOv3 architecture. In Section 3, procedures for generating database and preliminary network performance are discussed. In Section 4, the proposed improvements for YOLOv3 are elaborated. Cross-validation and testing results are presented in Section 5, followed by the concluding remarks in Section 6.

2 | YOLOv3 ARCHITECTURE

YOLOv3 is a single-stage CNN consisting of one backbone and one head subnet (Figure 1). The backbone of YOLOv3 is called Darknet-53, which contains 23 residual units (He, Zhang, Ren, & Sun, 2016) and is responsible for computing the convolutional feature maps over an entire input image. Each residual unit contains one 3×3 and one 1×1

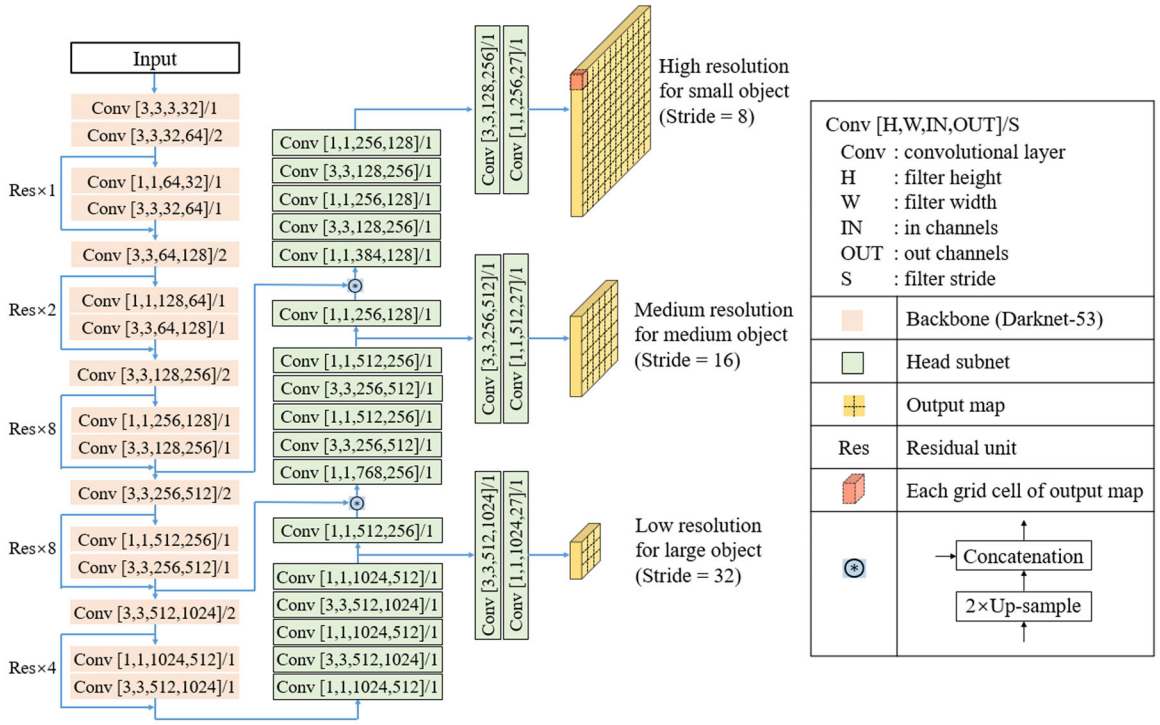


FIGURE 1 Architecture of YOLOv3 with multiscale predictions

convolutional layer, where each convolutional layer consists of three sequential operations: convolution, BN (Ioffe & Szegedy, 2015), and the Leaky ReLU activation (Maas, Hannun, & Ng, 2013). At the end of each residual unit, an element-wise addition is performed between the input and output vectors. The residual unit eases the training of deep networks and can effectively solve the vanishing gradient problem. Also, downsampling operation is performed in five separate convolutional layers with a stride of 2 mainly to reduce the feature map dimensionality (width and height) for computational efficiency. The backbone Darknet-53 is pretrained using the ImageNet dataset (Deng et al., 2009) and performs on par with state-of-the-art classifiers but with less computational cost (Redmon & Farhadi, 2018).

The head subnet of YOLOv3 is built on top of the backbone and adopts an FPN (Lin et al., 2017) to detect objects at three different scales. The FPN augments a standard convolutional network via a top-down pathway and lateral connections. By doing so, it efficiently constructs a multiscale feature pyramid with rich features at all levels from a single resolution input image. YOLOv3 incorporates the FPN and constructs a three-level pyramid with downsampling strides of 32, 16, and 8. Each level of the pyramid can be used for detecting objects at a different scale. The lower resolution feature maps with larger strides produce a very coarse representation of the input image and can be used for large object detection. On the other hand, the higher resolution feature maps have more fine-grained features and can be used for the detection of smaller objects. After extracting features from these three different scales, YOLOv3 predicts objects' location and type at each scale by adding a few more convolutional layers.

YOLOv3 predicts three bounding boxes at each grid cell on three different-resolution output feature maps. Each predicted box has one confidence variable (t^c), four class variables (t^i, $i = 1, 2, 3, 4$), and four coordinate variables (t^x, t^y, t^w, t^h). These predicted variables are, respectively, transformed to objects' confidence, class probabilities, and locations to generate predicted results. Object confidence C reflects the probability of a box containing an object and is computed using a sigmoid function (σ)

$$C = \sigma(t^c) = \frac{1}{1 + e^{-t^c}} \tag{1}$$

As the damage classes are mutually exclusive in the bridge damage dataset, the softmax classifier is used to transform the output class variables in a multiclass probability distribution (Bishop, 2006). The probability of being each damage type p^i corresponds to

$$p^i = \frac{e^{t^i}}{\sum_{i=1}^{4} e^{t^i}} \tag{2}$$

For location prediction, YOLOv3 predicts center coordinates of bounding box relative to the location of the grid cell in such a way that the center coordinates are between 0 and 1. As shown in Figure 2, if the grid cell is offset from the top left

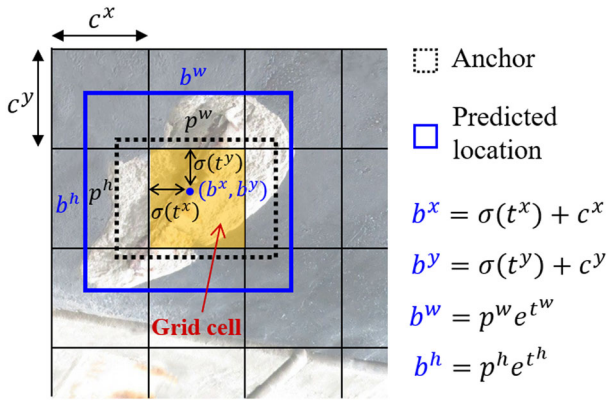


FIGURE 2 Bounding box with anchor and predicted location

corner of the image by (c^x, c^y), then the predicted bounding box has center coordinates (b^x, b^y). The width b^w and height b^h of the predicted bounding box are regressed from a set of predefined reference boxes called anchors. To generate prediction, for the case that the corresponding anchor has width p^w and height p^h, the predicted width b^w and height b^h of the object are as shown in Figure 2.

With these definitions, YOLOv3 minimizes an objective function, which is a weighted sum of the confidence loss, classification loss, and localization loss. The cross entropy (CE) loss (Goodfellow, Bengio, & Courville, 2016) is used to measure the performance of confidence and class predictions, whose output is a probability value between 0 and 1. For the localization loss, sum-squared loss is used as the predicted values need to regress to a specific target value not necessarily between 0 and 1. During training, YOLOv3 assigns one particular anchor for each ground-truth box. In other words, only the anchor that has the highest IoU with the ground-truth box will be responsible for predicting the object. Hence, the loss function only penalizes classification errors and localization errors if that anchor is responsible for the ground-truth box. This leads to specialization among the bounding box predictors (Redmon et al., 2016).

3 | DATASET AND INITIAL EVALUATION

3.1 | Dataset

An image dataset containing a total of 2,206 inspection images of highway concrete bridges in Hong Kong was established for this study. Among them, 75% of the images with 1,280 × 960 pixels resolution were acquired from the Hong Kong Highways Department and the rest with 4,000 × 3,000 pixels resolution were taken by the authors. The graphical image annotation tool LabelImg (Lin, 2015) was used to label damage objects, including their damage types and bounding box coordinates. Four different types of damage are considered in this study: crack, pop-out, spalling, and exposed rebar. Spalling refers to the type of damage where the concrete surface has peeled off but rebars are not exposed, while pop-out is used to indicate those damages where the concrete cover is still in place. To evaluate the network performance, the dataset was randomly split into training data and testing data with a ratio of 8:2, respectively. Figure 3 shows some examples of the annotated images with different types of damage.

To determine the anchor dimensions, YOLOv3 implements *k*-means clustering on ground-truth boxes in the training data to find good candidates for anchors automatically. The good candidates are dimensions of anchors that lead to the largest overlap between the anchors and ground-truth boxes. The number of anchors (clusters) could be adjusted and more anchors seem to make the task of predicting boxes easier but requires larger computation cost. In this study, as recommended by Redmon and Farhadi (2018), nine anchors were used and uniformly divided into three groups based on their area. Each anchor group can be assigned to a specific level in the pyramid network, where the anchors with smaller scales are assigned to the feature map with a higher resolution. In this way, specific feature maps learn to be responsive to particular scales of the objects. Multiscale training with seven image sizes: MS = {416, 448, 480, 512, 544, 576, 608} is performed to give the network the ability to predict well across a variety of input dimensions. To obtain the good candidate anchors, *k*-means clustering is also considered on these seven different input sizes. For the bridge damage dataset, the computed nine clusters (width × height) were: (29 × 22), (30 × 95), (97 × 37), (39 × 267), (105 × 101), (290 × 59), (227 × 139), (126 × 282), and (411 × 209). These nine anchors can offer an average IoU of 0.583 with ground-truth boxes and 70.4% of the ground-truth boxes have an anchor with IoU larger than 0.5.

Data augmentation is a way to increase the performance of an object detector and can reduce the probability of overfitting on a small dataset. Considering the actual operation of bridge inspection, the following four aspects were considered for data augmentation. First, there is image blurring or degradation because of camera movement during the image capturing process. The motion blur is more severe for camera installed on a flight system (e.g., UAV) due to wind effect. The shift in the image can be simulated by a linear motion blur function with a given motion direction and length (Hallermann & Morgenthal, 2014; Moghaddam & Jamzad, 2006). Second, due to random illumination of the scenes, the real inspection images have large brightness variations. This effect can be simulated by changing the pixel values of "lightness" in HSL (hue, saturation, lightness) color space. Third, salt-and-pepper noise is used to model defects in the charge-coupled device image sensor or in the transmission of the image (Koziarski & Cyganek, 2017). Finally, due to varying viewpoint and camera-damage

FIGURE 3 Sample images with ground-truth boxes and class labels

distance in the real-world application, the damage objects have a very large variation in the scale and aspect ratio. For this effect, scaling and cropping the image randomly can make the model more robust to the various damage sizes and shapes. These four aspects help to generate training samples that cover the span of image variations and help the network to be less sensitive to changes in these properties. In summary, each training image is augmented by the following sequence (Figure 4): (a) randomly scaling and cropping with a probability of 1/2; (b) horizontally or vertically flipping the image with a probability of 1/3; (c) randomly manipulating the images by applying the motion blur, changing brightness or adding salt-and-pepper noise with a probability of 1/4; and (d) inputting a padded training image to the network. Note that the data augmentation was carried out during the training process and the raw images were used for validation and testing. Meanwhile, the probability of augmentation effect in each step is assigned based on the equal probability as shown in Figure 4.

3.2 | Training YOLOv3

To evaluate the performance and potential application of the YOLOv3, the model was trained using the training data with the proposed data augmentation procedure and its accuracy was evaluated on the testing data. Training and testing were performed on a PC running the Ubuntu 17.04 operating system with an Intel® Core™ i7-7700 CPU and a GeForce GTX 1060 6GB GPU. The interpreted high-level programming language Python and the open-source software library TensorFlow were used for the experimental study. The YOLOv3 network was trained using the Adam optimizer (Kingma & Ba, 2014) with a batch size of 2 and a weight decay of 0.0001. The

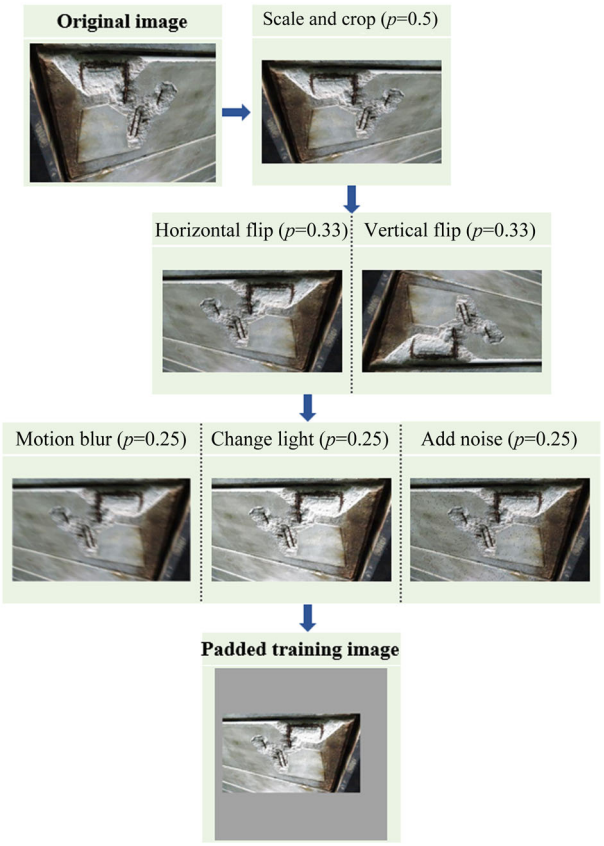


FIGURE 4 Data augmentation procedure (p: probability)

total number of training epochs was fixed at 80 with learning rates of 10^{-3} for the first 25 epochs, 10^{-4} for the second 25 epochs, and 10^{-5} and 10^{-6} for the last two 15 epochs. A class-specific confidence threshold of 0.25 (confidence $C \times$ class probability p^i) was used to give out predicted results (Redmon & Farhadi, 2018). Average precision (AP) and mean AP (mAP) were used to evaluate the detection performance, where AP summarizes the precision/recall curve for a given class by calculating the area under the curve (Everingham, Van Gool, Williams, Winn, & Zisserman, 2010) and mAP is the average of calculated APs for all classes.

Rather than training from scratch, TL was applied for fine-tuning YOLOv3, where the backbone was initialized with the pretrained Darknet-53 weights on ImageNet dataset. And the convolutional layers pertinent to head subnet without pretrained weights are initialized with Xavier initialization (Glorot & Bengio, 2010). From the authors' previous works, the trained YOLOv3 model with TL had much better performance than the model without TL on the aforementioned dataset for mean average precision at IoU = 0.5 (C. Zhang & Chang, 2019). In this study, mean average precision at both IoU = 0.5 (mAP_{50}) and IoU = 0.75 (mAP_{75}) was considered for evaluating the detection accuracy. The stricter IoU metric of 0.75 was used to count for special tasks that require more accurate localization (Lin et al., 2014). A higher localization accuracy is particularly important for those tasks that need to perform damage segmentation or quantification inside the detected region (I. Kim et al., 2018). The testing results were evaluated on different input image sizes as those used in multiscale training. Table 1 summarizes the mean average precision at both IoU = 0.5 and IoU = 0.75 under different testing image sizes. As shown, there is a threefold gap between mAP_{50} and mAP_{75} for all image sizes. This indicates that the overlap between most predictions and corresponding ground-truth boxes is just over 0.5, which could be a problem for subsequent tasks like damage segmentation. It should also be mentioned that the achieved detection speed is much slower than the reported value in the original YOLOv3 paper (Redmon & Farhadi, 2018), which is mainly due to the discrepancy in the GPU performance.

3.3 | Comparison with Faster R-CNN

To better understand the performance and applicability of the YOLOv3 for bridge damage detection, its accuracy and detection speed were compared with the most commonly used object detector in structural inspection domain, the Faster R-CNN (Cha et al., 2018; Cheng & Wang, 2018; Li et al., 2018; Liang, 2018). The Faster R-CNN consists of two steps: (a) an RPN takes an image as the input and outputs a set of rectangular object proposals; and (b) a Fast R-CNN detector takes the set of proposed regions from RPN and their corresponding feature vector to predict the class probabilities and the offset values for the bounding boxes. In Faster R-CNN, the base convolutional layers for feature extraction are shared between RPN and Fast R-CNN. In this experiment, the ResNet-101 (He et al., 2016) is chosen as the shared base layers, which is pretrained on ImageNet dataset. To fine-tune the Faster R-CNN, the RPN and Fast R-CNN networks are merged into one network and the end-to-end jointly training method is used as in Huang et al. (2017).

To train the Faster R-CNN, the shorter sides of the input images were scaled to 600 pixels and the same data augmentation procedure employed in training YOLOv3 was applied. For the anchors, three scales with box areas: 46^2, 109^2, and 235^2, and three aspect ratios: 1:1, 1:3, and 3:1 were used. These anchor sizes were carefully chosen based on the distribution of object scales with percents of object at 16.7%, 50%, and 83.3%. These nine anchors could offer an average IoU of 0.572 with ground-truth boxes and 67.4% of the ground-truth boxes had an anchor with IoU larger than 0.5. Adam optimizer was used and the total number of training epoch was set to be 120. The learning rate for the first 60 epochs was 10^{-4} and it was divided by 10 for every 30 epochs of the remaining 60 epochs. Note that the testing results for the Faster R-CNN were obtained using 300 RPN proposals.

Table 1 summarizes the comparison of testing performance between the YOLOv3 and the Faster R-CNN. It is observed

TABLE 1 Testing results of the original YOLOv3 and Faster R-CNN

Model	Train image size	Test image size	Testing time (s)	mAP_{50}	mAP_{75}
Faster R-CNN	Shorter side = 600		0.537	0.691	0.367
		416	0.117	0.610	0.208
		448	0.130	0.637	0.215
		480	0.144	0.633	0.224
YOLOv3	MS	512	0.160	0.662	0.240
		544	0.183	0.664	0.251
		576	0.196	0.649	0.260
		608	0.209	0.648	0.256

Note: Multiscale training is performed for the YOLOv3 with seven image sizes: MS = {416, 448, 480, 512, 544, 576, 608}.

that the testing speed of the Faster R-CNN is about three to five times slower than the YOLOv3 when tested on a PC with the aforementioned specifications. In terms of detection accuracy, the mAP_{50} and mAP_{75} of the Faster R-CNN are about 2.9% and 12.7% higher than that of the YOLOv3 at input image size of 512. Even though the YOLOv3 shows a similar accuracy to the Faster R-CNN when tested on the COCO dataset (Redmon & Farhadi, 2018), the mAP_{75} of the Faster R-CNN is much higher than that of the YOLOv3 on the bridge damage dataset. This is possibly due to the small damage dataset, which makes transferred feature layers impact more on accuracy. For the Faster R-CNN, only the final fully connected layers and a small head subnet of RPN are initialized without pretrained weights, which have fewer weight parameters than the head part of the YOLOv3. Also, the Faster R-CNN benefits from the two stages of bounding box regression, which offers a more accurate predicted location than the YOLOv3. In the following, some improvements are investigated to enhance the detection accuracy of the YOLOv3, particularly with the focus on localization accuracy.

4 | IMPROVEMENTS

There are lots of factors affecting the TL performance and the common practice of applying TL for object detectors may not perform well when fine-tuning on a relatively small and dissimilar dataset like structural inspection images. Another issue is the imbalance between the background (i.e., no damage) and foreground during network training. This may also be a case for different damage classes in the training dataset. Also, insufficient GPU memory may result in a small training batch size, which might diminish the effectiveness of BN and lead to poor testing results. To mitigate these issues, some improvements on the YOLOv3 are proposed in this section.

4.1 | Transfer learning

Due to the remarkable success of using ImageNet pretrained CNNs for TL, extensive efforts have been made on understanding the factors that affect the TL performance (Huh, Agrawal, & Efros, 2016). There are generally two groups of factors affecting the transferability between source task (pretrained model) and target task (target model): (a) model attributes, such as network architecture (Azizpour, Razavian, Sullivan, Maki, & Carlsson, 2016) and number of transferred layers (Yosinski et al., 2014); (b) data attributes, such as the size and distribution of pretrained data (Azizpour et al., 2016; Sun, Shrivastava, Singh, & Gupta, 2017) and the similarity between pretrained and target data (Cui, Song, Sun, Howard, & Belongie, 2018; Lim, Salakhutdinov, & Torralba, 2011; Yosinski et al., 2014). These studies suggested that a better TL performance could be achieved with more transferred feature layers. Moreover, a higher similarity between pretrained and target data leads to better performance on the target task. Possible source of similarity between pretrained and target data can be class overlap between two tasks (Huh et al., 2016; Oquab et al., 2014) and shared visual attributes, such as color, texture, and object shapes among different classes (Cui et al., 2018; Lim et al., 2011; G. Wang, Forsyth, & Hoiem, 2013).

In this study, inspired by the fact that a better performance could be achieved with a pretrained model with higher similarity and more transferred feature layers, a novel TL method was proposed. Instead of directly fine-tuning all layers using the damage dataset, the fine-tuning procedure is carried out in two steps. First, a larger common object dataset with a higher geometric similarity to the damage dataset is extracted from COCO dataset (Lin et al., 2014) to fine-tune the network. Then, the fully pretrained weights for all layers are restored and fine-tuned again using the damage dataset. Figure 5 illustrates the commonly used (TL-A) and the proposed (TL-B) TL methods. During the first fine-tuning step of TL-B, the anchors from k-means clustering of the damage dataset were used. That is, because a specific scale of anchors is assigned to each level and specific feature maps learn to be responsive to particular scales of the objects. Therefore, this can help to make the object scales in pretrained weights and the damage dataset to be similar in terms of each level of feature maps.

In this method, the object's geometric attributes, including scale and aspect ratio, are selected as the similarity metric

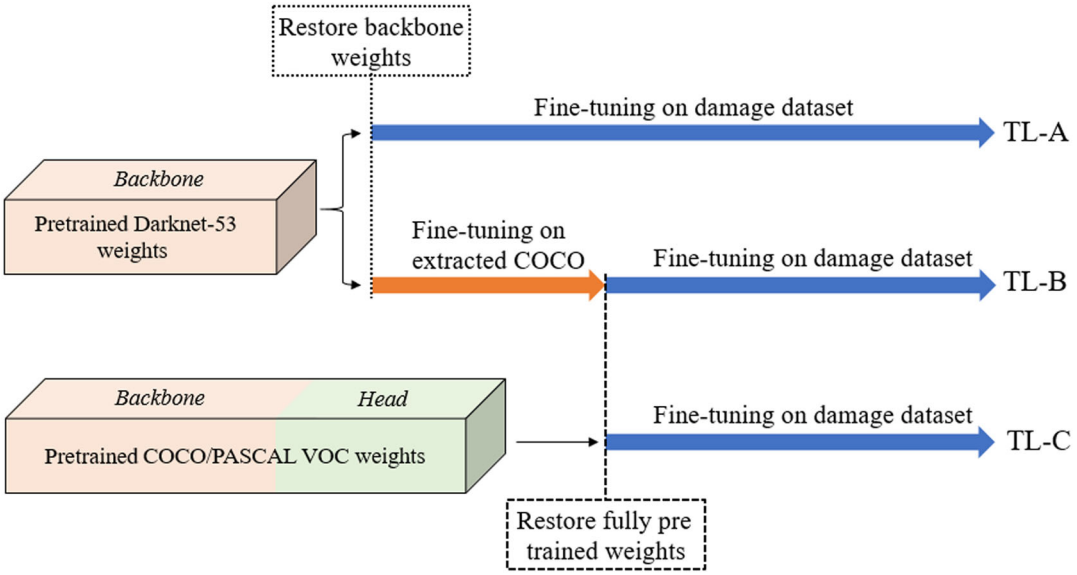
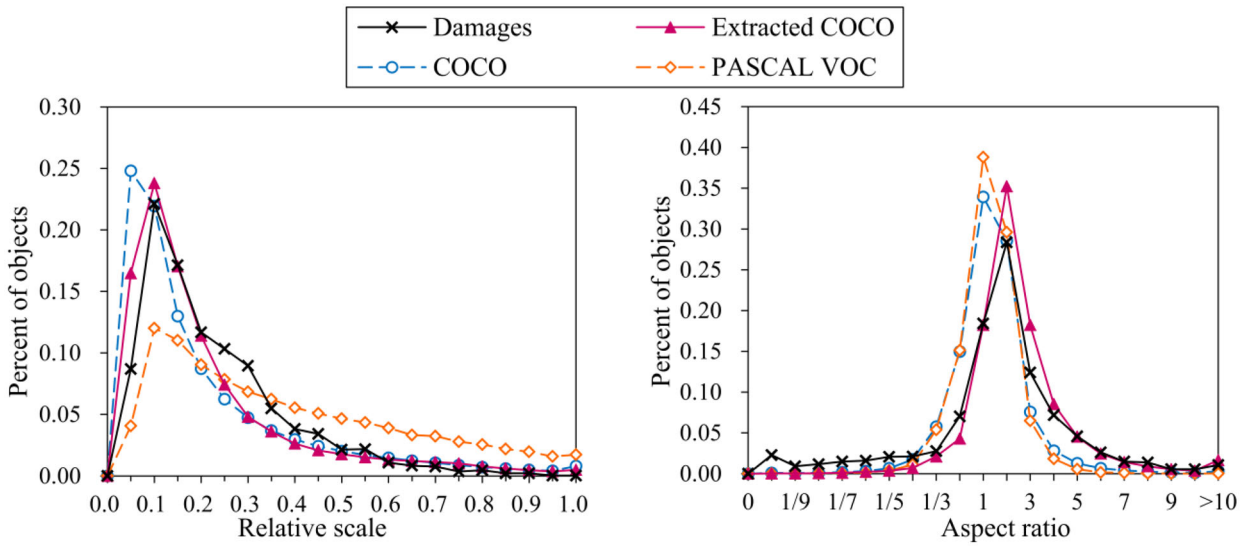


FIGURE 5 Schematic diagram of different transfer learning methods



(a) Distribution of object scales (b) Distribution of aspect ratios

FIGURE 6 Comparison between different datasets

between pretrained and target dataset. Compared with other sources of similarity, it is natural to expect a higher correlation between this metric and the localization accuracy of target task. Actually, the difference in the object scale between pretrained and damage datasets may result in a large domain-shift when fine-tuning from a pretrained network. This makes the optimization process difficult (Redmon & Farhadi, 2017; Singh & Davis, 2018). Hence, those classes with the closest distribution of object scales and aspect ratios to the damage dataset are extracted from COCO dataset for the first fine-tuning step. The distribution of object scales is presented in Figure 6a as the percent of object in 20 uniformly distributed relative scale segments, where the relative scale is the object size normalized by the image size. Similarly, in Figure 6b, the distribution of object aspect ratios is plotted as the percent of object for each of the 20 aspect ratio segments {0–1/10, 1/10–1/9, …, 9–10, >10}. The similarity of the distributions between each COCO class and damage dataset is measured and sorted through the Euclidean distance. The total data number of the extracted four COCO classes (namely, bench, fork, skateboard, and truck) was 17,952, which had the minimum sum of the Euclidean distance for object scales and aspect ratios.

Figure 6 displays the comparison of the data distributions between bridge damages, extracted four COCO classes, COCO, and PASCAL VOC datasets. It is obvious that the

extracted four COCO classes have the highest similarity of object scales and aspect ratios to the bridge damages, while the PASCAL VOC has the least similarity. To evaluate the effect of such similarity on TL performance, another method called TL-C was studied wherein all layers were initialized with the fully pretrained COCO or PASCAL VOC weights (Figure 5). In the later experimental studies, the model starts with the pretrained backbone weights and the fully pretrained weights were trained for a total of 80 and 50 epochs, respectively.

4.2 | Batch renormalization

BN (Ioffe & Szegedy, 2015) is adopted in YOLOv3 to normalize the convolutional output of each layer, which helps to stabilize training, speed up convergence, and regularize the model. The input vector of each BN layer is a 4D tensor of size $N \times H \times W \times C$ (Figure 7), where N indicates the number of samples (mini-batch size), H and W represent the spatial height and width, and C is the number of channels. During training, BN normalizes the input vector along each channel independently as follows:

$$y_{ij} = \gamma_j \left(\frac{x_{ij} - \mu_{Bj}}{\sqrt{\sigma_{Bj}^2 + \epsilon}} \right) + \beta_j \; i \in [1,\ N],\ j \in [1,\ C] \tag{3}$$

where x_{ij} and y_{ij} represent the input and the output vector of each BN layer for sample i along each channel j; μ_{Bj} and σ_{Bj}^2 are the mini-batch mean and variance computed along each channel j across all samples (Figure 7); γ_j and β_j are a pair of parameters learned along with the original model parameters; and ϵ is a small real number to avoid dividing by zero. At latter testing time, the moving averages of mean and variance (as an estimate of the statistics of the entire training set) are used for the normalization step:

$$y_{ij} = \gamma_j \left(\frac{x_{ij} - \mu_j}{\sqrt{\sigma_j^2 + \epsilon}} \right) + \beta_j \; i \in [1,\ N],\ j \in [1,\ C] \tag{4}$$

where μ_j and σ_j^2 are the moving averages of the mean and variance along each channel j as shown in Figure 7.

However, the effectiveness of BN diminishes when the mini-batch mean (μ_{Bj}) and variance (σ_{Bj}^2) diverge frequently from the mean (μ_j) and variance (σ_j^2) over the entire training set. Consequently, the mean and variance are different during training and testing, which can decrease the testing performance. This occurs commonly when the training batch size is small due to limited GPU memory. Such restriction on batch size is more severe for the detection and segmentation tasks as they benefit from large size of input images (P. Luo, Ren, Peng, Zhang, & Li, 2018; Wu & He, 2018). For instance, in this study, the computer can only afford a batch size of 2 ($N = 2$). BR (Ioffe, 2017) tries to tackle the issue of differing statistics at training and testing time directly. During testing time, the BR has the same form of normalization step as that of the BN (Equation 4). However, at training time, the BR augments the normalization step of BN by an affine transformation as follows:

$$y_{ij} = \gamma_j \left(\frac{x_{ij} - \mu_{Bj}}{\sqrt{\sigma_{Bj}^2 + \epsilon}} \times r_j + d_j \right) + \beta_j \; i \in [1,\ N],\ j \in [1,\ C] \tag{5}$$

$$r_j = \frac{\sigma_{Bj}}{\sigma_j} \in \left[\frac{1}{r_{\max}},\ r_{\max} \right] \tag{6}$$

$$d_j = \frac{\mu_{Bj} - \mu_j}{\sigma_j} \in \left[-d_{\max},\ d_{\max} \right] \tag{7}$$

where r_j and d_j are treated as constants during the gradient computation; μ_j and σ_j^2 are the current moving mean and variance, respectively; and $r_{\max}$ and $d_{\max}$ are two hyperparameters introduced to control the transition between BN and BR (default values are used as in Ioffe, 2017). Using mini-batch statistics (μ_{Bj} and σ_{Bj}^2) and affine transformation at training time and moving averages (μ_j and σ_j^2) at testing time ensures that the output of BR is the same during both phases. The BR was originally evaluated on classification model and it has not been applied in the other tasks, such as object detection and segmentation. Hence, it is worthwhile to investigate the performance and applicability of BR for object detectors.

4.3 | Focal loss

An issue with training single-stage detectors is that there is an imbalance between foreground and background samples, for example, around 1:8,000 for a 512 × 512 image with two objects in the YOLOv3. As a result, easily classified background samples comprise the majority of the confidence loss and dominate the gradient. Additionally, the class imbalance between different damages in the training data can have a similar effect. In our case, the number of damage instances belonging to exposed rebar and crack (72%) is much larger than the other two damages (28%) (Table 2). The confidence and class imbalances cause two problems: (a) training is not very efficient as most locations are easy background samples that provide no useful learning signal; and (b) the background and damage classes with a larger number of samples may overwhelm training and lead to degenerate models. To mitigate these imbalance problems, the commonly used approach is applying different fixed weighting factors

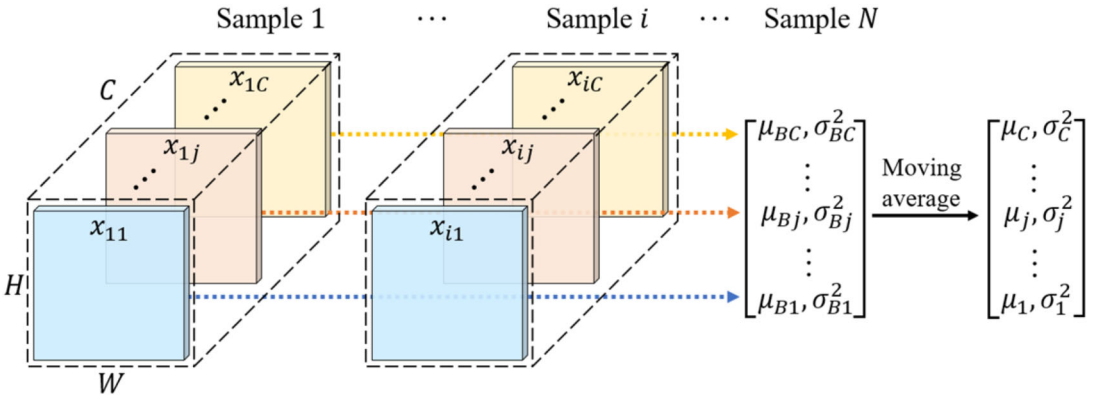


FIGURE 7 Mini-batch mean and variance of the input vector

TABLE 2 Number of annotated instances per class

Damage type	Number of instances	Percentage (%)
Crack	1,099	35.1
Pop-out	474	15.1
Spall	411	13.1
Rebar	1,146	36.6

for the loss of different classes. However, the weighting factors in such method should be carefully chosen, and they do not differentiate between easy/hard examples. In this study, FL (Lin et al., 2018) is applied to the original loss function, which dynamically scales the loss of different classes. The FL addresses the imbalance using a modulating factor to down-weight easy samples by discounting their contribution to the total loss even if their number is large. In other words, the FL performs training on a sparse set of hard samples. The FL was originally applied for multiclass prediction, where each class was treated as a binary classification problem with sigmoid output (Lin et al., 2018). However, in YOLOv3, the confidence (foreground and background) and class predictions are separated and they should take different forms of FL. For confidence loss of each prediction, the FL applied for a CE with sigmoid outputs is as below:

$$FL_{\text{conf}} = \begin{cases} -(1-y)^{\gamma}\log(y) & \text{if } \hat{y}=1 \text{ (foreground)} \\ -y^{\gamma}\log(1-y) & \text{if } \hat{y}=0 \text{ (background)} \end{cases} \quad (8)$$

For classification loss of each prediction, the FL for a CE with softmax outputs is as follows:

$$FL_{\text{class}} = -(1-y)^{\gamma}\log(y) \quad \text{if } \hat{y}=1 \text{ (positive class)} \quad (9)$$

where y and $\hat{y}$ are the respective predicted and ground-truth values; $(1-y)^{\gamma}$ and y^{γ} are the modulating factors; and γ is the focusing parameter that smoothly adjusts the downweighting rate of easy samples. When $\gamma = 0$, FL is equivalent to CE, and as γ increases the downweighting effect of the modulating factor is likewise increased (in this study, $\gamma = 2$ in all experiments). Figure 8 demonstrates an example of applying FL for confidence loss with a ratio of 1:1,000 between foreground and background samples. In this example, the predicted value of background samples was illustrated as moving from 0.6 to the target value 0 and the foreground prediction was assumed to be 0.3 and 0.6, respectively. It is seen that the background samples almost always take up all the confidence loss when using the original CE loss function. However, the FL dramatically downweights the loss for background samples when they become easily classified (e.g., background prediction ≤ 0.2). In this case, the FL could put more focus on hard, misclassified foreground samples.

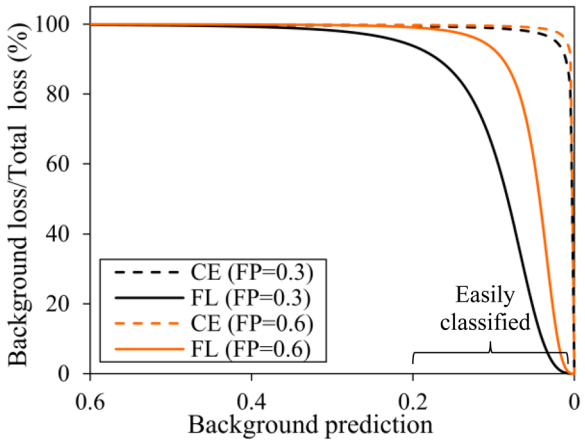


FIGURE 8 Illustration of focal loss effect on confidence loss

5 | RESULTS AND DISCUSSION

To study the performance of the proposed improvements on the original YOLOv3, the cross-validation and testing experiments were performed with the same training setting described in Section 3.2. Cross-validation is commonly used to tune the model hyperparameters and compare the performances of different model designs (Brownlee, 2018). The testing experiment uses the testing data, which is a holdout

TABLE 3 Cross-validation results for different transfer learning methods

Model	Fully pretrained weights	Total training epochs	Max mAP_{50}	Max mAP_{75}
TL-A	–	80	0.638	0.244
TL-B	Extracted COCO	80	0.714	0.410
TL-B*	Extracted COCO	80	0.702	0.386
TL-C1	COCO	50	0.694	0.348
TL-C2	PASCAL VOC	50	0.661	0.291

*represents using the anchors from *k*-means clustering of the extracted four COCO classes in the first fine-tuning step of TL-B.

dataset to provide an unbiased evaluation of the final model (Brownlee, 2017).

5.1 | Cross-validation results

The training data were randomly partitioned into five equal sized subsamples according to the fivefold cross-validation principle. Each subsample had the same proportion of each damage class. The cross-validation process was then repeated five times, with each of the five subsamples used exactly once as the validation data. The three aforementioned improvements were evaluated sequentially. To evaluate the proposed TL method, five experiments were conducted as follows: (a) TL-A: the backbone was initialized with the pretrained Darknet-53 weights and then all layers were fine-tuned using damage dataset for 80 epochs (common practice of applying TL for an object detector); (b) TL-B: following the procedure explained in Section 4.1, the backbone was initialized with the pretrained Darknet-53 weights and all layers were first fine-tuned on the extracted four COCO classes for the first 30 epochs. Then, all layers were fined-tuned again using damage dataset for the remaining 50 epochs; (c) TL-B*: similar to TL-B but using the anchors from *k*-means clustering of the extracted four COCO classes in the first fine-tuning step; (d) TL-C1: all layers were initialized with the pretrained COCO weights and then all layers were fine-tuned using damage dataset for 50 epochs; (e) TL-C2: similar to TL-C1 but using the pretrained PASCAL VOC weights. The pretrained COCO and PASCAL VOC weights are publicly accessible and they were originally trained for 80 and 20 classes, respectively. To restore these two fully pretrained weights for all layers, their last class prediction channels of each scale were cropped with only the first four class channels retained. Also, the training setting for fine-tuning the TL-C1 and TL-C2 was the same as the second fine-tuning step (50 epochs) for TL-B and TL-B*. A summary of the five TL experiments is given in Table 3.

Figure 9a,b shows the comparison of validation accuracy at IoU = 0.5 and IoU = 0.75 over last 50 training epochs, respectively. The validation accuracy is evaluated after each training epoch with an input image size of 512 and is presented as the average value of fivefold cross-validation. It is seen that the validating accuracy shows a noisy curve and such fluctuation is mainly due to the inherent feature of Adam optimizer and the small training batch size. Comparing the five TL experiments, the models with fully pretrained weights (TL-B, TL-B*, TL-C1, and TL-C2) converge much faster and have a higher accuracy than the TL-A, which is only initialized with pretrained backbone weights. The TL-B using fully pretrained weights from a relatively larger and geometrically similar dataset offers the highest accuracy around $mAP_{50} = 71.4\%$, which is 7.6% higher than the commonly used TL-A (Table 3). For the strict evaluation metric, the TL-B gives an $mAP_{75} = 41.0\%$, which is 16.6% higher than the TL-A. Comparing the model with TL-B and TL-B*, using the anchors from *k*-means clustering of the damage dataset during the first fine-tuning step offers a 1.2% and 2.4% higher accuracy for the mAP_{50} and mAP_{75}, respectively. This can be attributed to the reduction of domain-shift in each level of feature maps when using the same anchors in pretrained and target models. Also, the mAP_{50} and mAP_{75} of the model with fully pretrained COCO weights (TL-C1) are about 3.3% and 5.7% higher than that with fully pretrained PASCAL VOC weights (TL-C2), but 2.0% and 6.2% lower than the model with TL-B. In terms of convergence speed, TL-B is the fastest among all TL cases as shown in Figure 9. These results indicate that a better TL performance especially for localization accuracy could be achieved when fine-tuned with fully pretrained weights from a geometrically similar dataset.

TABLE 4 Cross-validation results for batch renormalization and focal loss

Improvements	Model	Max mAP_{50}	Max mAP_{75}
Batch renormalization (BR)	TL-B & BR	0.751	0.433
Focal loss (FL)	TL-B & BR & FL	0.770	0.465

Figure 10 and Table 4 summarize the validation accuracy for YOLOv3 using BR and FL when training with the developed TL method TL-B. Comparing the validation accuracy for YOLOv3 using BN and BR, the validation accuracy mAP_{50} and mAP_{75} are increased by 3.7% and 2.3%, respectively. Also, the FL gives a further 1.9% and 3.2% gain on the accuracy mAP_{50} and mAP_{75}, respectively. A possible reason for a larger increase in mAP_{75} is that FL downweights the confidence and classification loss from easy samples and thus penalizes more on the localization loss. Hereafter, the

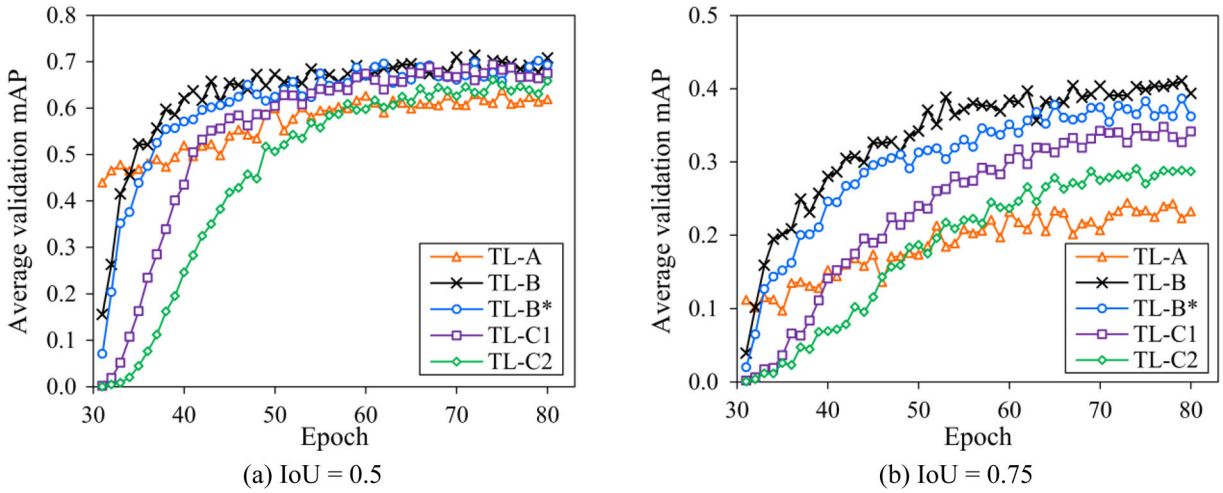


FIGURE 9 Cross-validation curves for different transfer learning methods

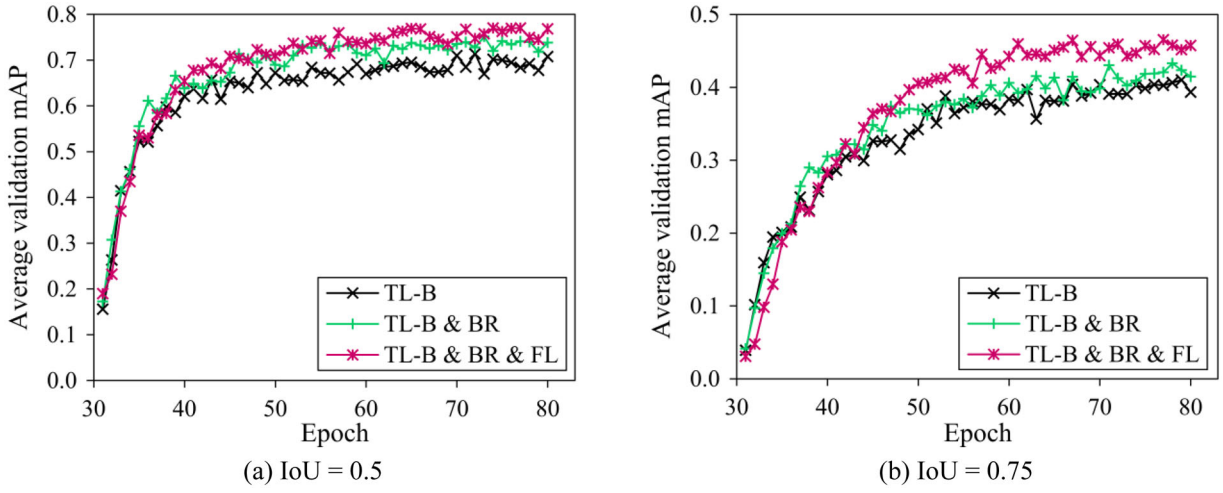


FIGURE 10 Cross-validation curves for batch renormalization and focal loss

YOLOv3 with TL-B, BR, and FL improvements is denoted as the "improved" YOLOv3 and the YOLOv3 with only TL-A as the "original" YOLOv3. Note that there is no need to evaluate all combinations of the three improvements as the final model is always the same despite that different evaluation sequence may slightly affect the accuracy value of each improvement. Also, it should be noted that in this study the obtained accuracy improvements are limited to the bridge inspection images dataset. Therefore, other datasets may give different accuracy values and more experiments with various object detection datasets are needed to fully validate the generalization of the improved YOLOv3.

5.2 | Testing results

To obtain the testing results of the improved YOLOv3, the model was trained using all training data and its accuracy was evaluated on the testing data. Here, the testing performance for the original YOLOv3 was also provided as a reference. Figures 11 and 12 along with Table A1 summarize the testing accuracy for each class and the mean average precision at both IoU = 0.5 and IoU = 0.75 under different testing image sizes. For the improved YOLOv3, at the input image size of 512, the AP_{50} values for crack, pop-out, spalling, and exposed rebar are 76.5%, 82.7%, 73.9%, and 86.6%, respectively. For the same input image size, the mAP_{50} and mAP_{75} of the improved YOLOv3 are 79.9% and 47.2%, respectively, which are about 13.7% and 23.2% higher than the respective values of the original model. The much higher increase in mAP_{75} indicates that the improved YOLOv3 with the novel TL method and improvements gives a more accurate location prediction. Comparing the average precision between different classes (Figure 11), the exposed rebar has a higher accuracy than the other three classes for IoU = 0.5. Other than having a relatively larger dataset, this could be because the texture representing the exposed rebar tends to be more consistent compared to the other damages. The average precisions of crack and spalling are lower, which can be attributed to the

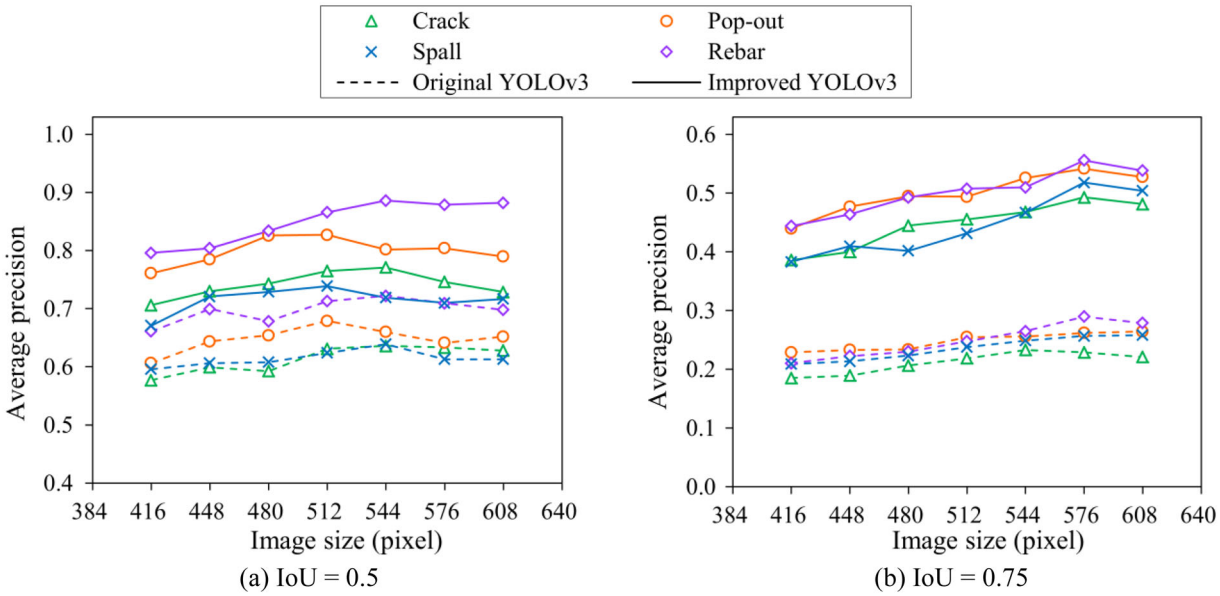


FIGURE 11 Average precision for each class

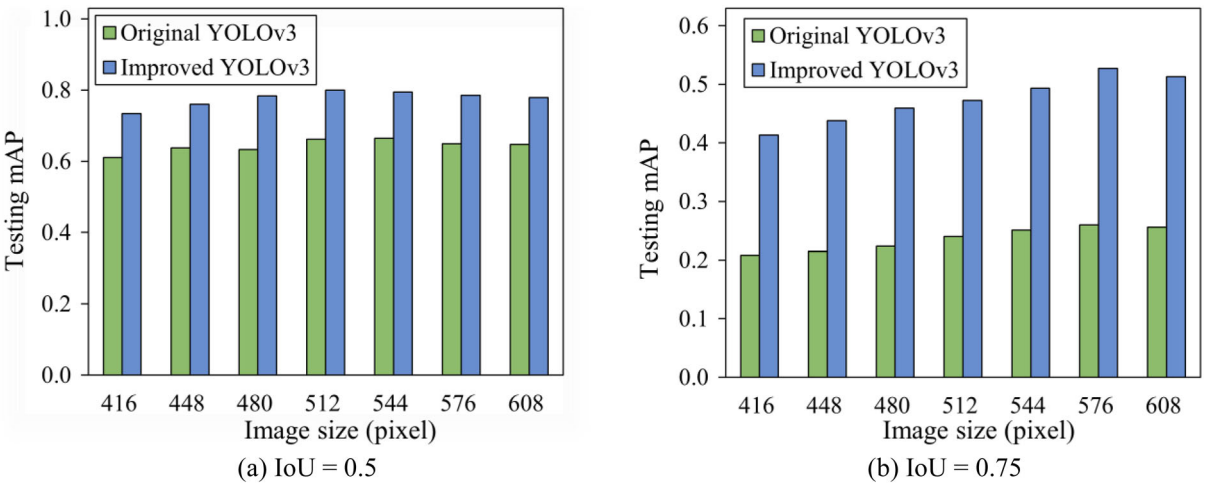


FIGURE 12 Testing accuracy of the original and improved YOLOv3

insufficient features of crack and small scale and dataset size of spalling.

Figure 12 shows the mean average precision versus multiple image sizes. As shown, the accuracy difference among various image sizes is not very significant. This indicates that the multiscale training enables the network to predict across a variety of input dimensions. It is, however, observed that there are a few drops in the accuracy when the image size increases. This phenomenon can be attributed to several factors: (a) the effective receptive field (W. Luo, Li, Urtasun, & Zemel, 2016) on the lowest resolution feature map that is responsible for detecting large objects is not large enough to predict the scaled-up objects; (b) there is a mismatch between image size and the anchor size when the image size becomes very large; and (c) the possible instability from the multiscale training.

Figure 13 displays detection examples of the improved YOLOv3. The testing results show a consistent performance for the inspection images taken under various conditions (e.g., locations, viewpoints, camera-damage distance, lighting, blur, background, and so on). Figure 14 demonstrates the examples that the trained model failed to detect and locate the damage. In Figure 14a, the network could not distinguish between spalling and exposed rebar as they both have similar features of exposed concrete aggregates. Similarly, the contour of some pop-out damages resembles a crack, which could cause both to be detected for a single damage (Figure 14b). Nonetheless, for both cases, the network is still successful in detecting the bridge damages for the inspection task. In addition, some diagonal or complex cracks were not well localized due to their indistinctive features and the large redundant (background) regions of the bounding box

FIGURE 13 Detection samples from the improved YOLOv3 model

(Figures 14c,d). The noise introduced by the redundant regions can interfere or even submerge the crack feature information that is important for accurate localization. This problem could probably be resolved by predicting an additional angle information for generating a rotational bounding box (X. Yang, Sun, et al., 2018), which has less redundant detection regions and can locate the damage more accurately. It should also be mentioned that the bridge damage dataset contains many small damages and complex background information, such as the vegetation, water stain, debris on surface, and different types of bridge elements. These inherent features could make the damage detection task more difficult. In this case, a larger dataset covering a broader and more comprehensive background and damage information could help to further improve the model's performance.

5.3 | Comparative study

Finally, the detection accuracy and speed of the improved YOLOv3 are compared with the two-stage detector Faster R-CNN. To have a fair comparison, the improvements mentioned earlier were implemented in the Faster R-CNN with ResNet-101 as well. The three improvements were sequentially applied, and the same training and testing procedures described in Section 3.3 were employed. Table 5 summarizes the testing results of Faster R-CNN at both IoU = 0.5 and IoU = 0.75. With the TL-B, the testing accuracy mAP_{50} and

FIGURE 14 Failure samples from the improved YOLOv3 model

TABLE 5 Testing results of the Faster R-CNN with improvements

Model	mAP_{50}	mAP_{75}
Faster R-CNN	0.691	0.367
Faster R-CNN/TL-B	0.721	0.414
Faster R-CNN/TL-B & BR	0.738	0.425
Faster R-CNN/TL-B & BR & FL	0.744	0.434

mAP_{75} increased by 3.0% and 4.7%, respectively, which is less than that observed in the original YOLOv3. This is mainly due to the fact that Faster R-CNN has fewer layers initialized without pretrained weights compared with the original YOLOv3. Also, the Faster R-CNN uses the single-scale feature map (i.e., last layer of the "conv4" block) for prediction task, which may result in a smaller domain-shift problem compared with the multiscale prediction of YOLOv3.

It should be noted that the training method used as in Huang et al. (2017) froze the BN parameters to be those in pretrained ResNet-101 weights. This is because the Faster R-CNN was trained with a batch size of 1 due to the varying input image size and large GPU memory footprint, and training BN layers with such extreme batch size works poorly (Huang et al., 2017; Wu & He, 2018). When replacing the frozen BN with BR and fine-tuning the BR layers with a batch size of 1, the testing accuracy mAP_{50} and mAP_{75} was increased marginally by 1.7% and 1.1%, respectively. In this case, the BR becomes an affine transformation of the instance normalization (Ulyanov et al., 2016), which normalizes the input vector along each channel for each sample. The small gain on accuracy from the BR is due to the benefit of normalization as the frozen BN case turns normalization layer into a constant transformation layer. Moreover, the FL showed negligible effect on the testing accuracy of the Faster R-CNN. This is because the RPN stage filters out most background samples and the Fast R-CNN stage applies a sampling procedure to maintain the balance between foreground and background samples.

Figure 15 and Table A1 present the comparison of testing accuracy between the YOLOv3 and the Faster R-CNN. The improved YOLOv3 has the highest accuracy among all cases, and the accuracy of the Faster R-CNN is in between the original and the improved YOLOv3. The mAP_{50} and mAP_{75} of the Faster R-CNN with improvements (TL-B, BR, and FL) are about 5.5% and 3.8% lower than that of the improved YOLOv3 at input image size of 512. Moreover, the Faster R-CNN shows a higher recall than precision and the difference between recall and precision is larger than that in YOLOv3 (Table A1), which indicates the Faster R-CNN has more false positives. This could be due to the fact that the

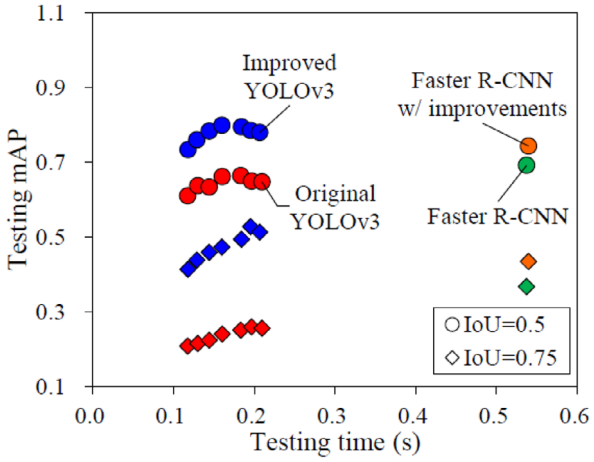


FIGURE 15 Comparison of testing results between Faster R-CNN and YOLOv3

Faster R-CNN assigns multiple anchors and proposed regions to each ground-truth box during training. Such multiple-to-one matching principle generates more training positives containing a certain portion of background. In terms of detection speed, when tested on the same platform, both the original and improved YOLOv3 are three to five times faster than the Faster R-CNN.

6 | CONCLUSIONS

In this study, a vision-based approach for detecting multiple surface damages in concrete highway bridges was proposed based on the single-stage detector, YOLOv3. The backbone of YOLOv3 was Darknet-53, which performed on par with the state-of-the-art classifiers but with a lower computational cost. The head subnet of YOLOv3 adopted the FPN to improve multiscale predictions. To apply YOLOv3 for concrete bridge inspection, a database comprising 2,206 field inspection images was collected. These images were labeled for four types of concrete damages: concrete crack, pop-out, spalling, and exposed rebar. During the training process, multiscale training with seven image sizes and a systematic data augmentation procedure were utilized. Testing results showed that the original YOLOv3 model could achieve a detection accuracy of $mAP_{50} = 66.2\%$ and $mAP_{75} = 24.0\%$ for two IoU metrics of 0.5 and 0.75, respectively, which are 2.9% and 12.7% lower than that of the Faster R-CNN with ResNet-101. In terms of detection speed, the original YOLOv3 is about three to five times faster than that of the Faster R-CNN when tested on the same platform. However, the threefold gap between mAP_{50} and mAP_{75} for the original YOLOv3 indicates a high localization error. This could be an obstacle for subsequent tasks of damage assessment like segmentation inside the detected regions.

Three modifications were then proposed to improve the original YOLOv3. A novel TL method was proposed to fine-tune the network with fully pretrained weights from a larger dataset having the closest distribution of geometric attributes (object scales and aspect ratios) to the damage dataset. It is demonstrated that pretraining with such a properly selected dataset offers a better TL performance especially for localization accuracy compared with those performance gained by simply fine-tuning off-the-shelf models. Cross-validation results showed that applying this new TL method could increase the validation accuracy mAP_{50} and mAP_{75} by about 7.6% and 16.6%, respectively. Next, to resolve the problem of low performance when training on a small batch size, BN in the original version was replaced by BR. This modification could increase the mAP_{50} and mAP_{75} by 3.7% and 2.3%, respectively. Finally, the FL was introduced to mitigate the imbalance in confidence and class predictions by down-weighting easy samples and performing training on a sparse set of hard samples. The validation results showed that the FL could further increase the mAP_{50} and mAP_{75} by 1.9% and 3.2%, respectively.

The testing results showed that the improved YOLOv3 model after the three modifications could achieve a detection accuracy of $mAP_{50} = 79.9\%$ and $mAP_{75} = 47.2\%$. Meanwhile, when the three aforementioned improvements were applied on the Faster R-CNN with ResNet-101, the testing accuracy mAP_{50} and mAP_{75} increased by 5.3% and 6.7%, but they were still 5.5% and 3.8% lower than the improved YOLOv3. In this study, the bridge damage dataset contains many small damages and a complex background information, which could inhibit the model's capacity and generalization. In the future, more inspection images with a broader background and damage information will be added to the database to further improve the model's performance. Also, because the results of the improved YOLOv3 are based on the given dataset, another area for future research involves more validation against standard benchmark datasets to further demonstrate the general applicability of the proposed three modifications.

REFERENCES

Azizpour, H., Razavian, A. S., Sullivan, J., Maki, A., & Carlsson, S. (2016). Factors of transferability for a generic convnet representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *38*(9), 1790–1802.

Ba, J. L., Kiros, J. R., & Hinton, G. E. (2016). *Layer normalization*. Retrieved from https://arxiv.org/abs/1607.06450/

Bishop, C. M. (2006). *Pattern recognition and machine learning (information science and statistics)*. New York: Springer.

Brownlee, J. (2017). *What is the difference between test and validation datasets. Machine learning process*. Retrieved from https://machinelearningmastery.com/difference-test-validation-datasets/

Brownlee, J. (2018). *A gentle introduction to K-fold cross-validation. Statistical methods*. Retrieved from https://machinelearningmastery.com/k-fold-cross-validation/

Cha, Y. J., Choi, W., & Büyüköztürk, O. (2017). Deep learning-based crack damage detection using convolutional neural networks. *Computer-Aided Civil and Infrastructure Engineering*, *32*(5), 361–378.

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.

Cheng, J. C., & Wang, M. (2018). Automated detection of sewer pipe defects in closed-circuit television images using deep learning techniques. *Automation in Construction*, *95*, 155–171.

Cui, Y., Song, Y., Sun, C., Howard, A., & Belongie, S. (2018). Large scale fine-grained categorization and domain-specific transfer learning. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Salt Lake City, UT, 4109–4118.

Dai, H., & Cao, Z. (2017). A wavelet support vector machine-based neural network metamodel for structural reliability assessment. *Computer-Aided Civil and Infrastructure Engineering*, *32*(4), 344–357.

Dawood, T., Zhu, Z., & Zayed, T. (2017). Machine vision-based model for spalling detection and quantification in subway networks. *Automation in Construction*, *81*, 149–160.

Deng, J., Dong, W., Socher, R., Li, L., Li, K., & Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Miami, FL, 248–255.

Ellenberg, A., Kontsos, A., Bartoli, I., & Pradhan, A. (2014). Masonry crack detection application of an unmanned aerial vehicle. In *Proceedings of the International Conference on Computing in Civil and Building Engineering*, 1788–1795.

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.

Gao, Y., & Mosalam, K. M. (2018). Deep transfer learning for image-based structural damage recognition. *Computer-Aided Civil and Infrastructure Engineering*, *33*(9), 748–768.

German, S., Brilakis, I., & DesRoches, R. (2012). Rapid entropy-based detection and properties measurement of concrete spalling with machine vision for post-earthquake safety assessments. *Advanced Engineering Informatics*, *26*(4), 846–858.

Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Columbus, OH, 580–587.

Glorot, X., & Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. *Journal of Machine Learning Research*, *9*, 249–256.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. Cambridge, MA: MIT Press.

Gopalakrishnan, K., Khaitan, S. K., Choudhary, A., & Agrawal, A. (2017). Deep convolutional neural networks with transfer learning for computer vision-based data-driven pavement distress detection. *Construction and Building Materials*, *157*, 322–330.

Hallermann, N., & Morgenthal, G. (2014). Visual inspection strategies for large bridges using unmanned aerial vehicles (UAV). *Proceedings of 7th IABMAS, International Conference on Bridge Maintenance, Safety and Management*, Shanghai, 661–667.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Las Vegas, NV, 770–778.

Huang, J., Rathod, V., Sun, C., Zhu, M., Korattikara, A., Fathi, A., … Murphy, K. (2017). Speed/accuracy trade-offs for modern convolutional object detectors. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Honolulu, HI.

Huh, M., Agrawal, P., & Efros, A. A. (2016). *What makes imagenet good for transfer learning?* Retrieved from https://arxiv.org/abs/1608.08614/

Ioffe, S. (2017). Batch-renormalization: Towards reducing minibatch dependence in batch-normalized models. *Proceedings of Advances in Neural Information Processing Systems*, Long Beach, CA, 1945–1953.

Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *Proceedings of the 32nd International Conference on Machine Learning*, Lille, France, 448–456.

Kim, H., Sim, S., & Cho, S. (2015). Unmanned aerial vehicle (UAV)-powered concrete crack detection based on digital image processing. *Proceedings of 6th International Conference on Advances in Experimental Structural Engineering, 11th International Workshop on Advanced Smart Materials and Smart Structures Technology*, University of Illinois, Chicago, IL.

Kim, I., Jeon, H., Baek, S., Hong, W., & Jung, H. (2018). Application of crack identification techniques for an aging concrete bridge inspection using an unmanned aerial vehicle. *Sensors*, *18*(6), 1881.

Kingma, D. P., & Ba, J. (2014). *Adam: A method for stochastic optimization*. Retrieved from https://arxiv.org/abs/1412.6980/

Koch, C., Georgieva, K., Kasireddy, V., Akinci, B., & Fieguth, P. (2015). A review on computer vision based defect detection and condition assessment of concrete and asphalt civil infrastructure. *Advanced Engineering Informatics*, *29*(2), 196–210.

Koziarski, M., & Cyganek, B. (2017). Image recognition with deep neural networks in presence of noise—Dealing with and taking advantage of distortions. *Integrated Computer-Aided Engineering*, *24*(4), 337–349.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Proceedings of Advances in Neural Information Processing Systems*, Lake Tahoe, NV, 1097–1105.

Kucuksubasi, F., & Sorguc, A. (2018). *Transfer learning-based crack detection by autonomous UAVs*. Retrieved from https://arxiv.org/abs/1807.11785/

Lattanzi, D., & Miller, G. (2017). Review of robotic infrastructure inspection systems. *Journal of Infrastructure Systems*, *23*(3), 04017004. https://doi.org/10.1061/(ASCE)IS.1943-555X.0000353

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436–444.

LeCun, Y. A., Bottou, L., Orr, G. B., & Müller, K. R. (2012). Efficient backprop. In G. Montavon, G. B. Orr, & K. R. Müller (Eds.), *Neural networks: Tricks of the trade* (Vol. 7700, pp. 9–48). Lecture Notes in Computer Science. Berlin: Springer.

Li, R., Yuan, Y., Zhang, W., & Yuan, Y. (2018). Unified vision-based methodology for simultaneous concrete defect detection and

geolocalization. *Computer-Aided Civil and Infrastructure Engineering*, *33*(7), 527–544.

Liang, X. (2018). 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.

Lim, J. J., Salakhutdinov, R., & Torralba, A. (2011). Transfer learning by borrowing examples for multiclass object detection. *Proceedings of Advances in Neural Information Processing Systems*, Granada, Spain, 118–126.

Lin, T. (2015). *LabelImg. Git code*. Retrieved from https://github.com/tzutalin/labelImg

Lin, T. Y., Dollár, P., Girshick, R., He, K., Hariharan, B., & Belongie, S. (2017). Feature pyramid networks for object detection. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Honolulu, HI, 936–944.

Lin, T. Y., Goyal, P., Girshick, R., He, K., & Dollár, P. (2018). Focal loss for dense object detection. *Proceedings of the IEEE International Conference on Computer Vision*, Los Alamitos, CA, 2999–3007.

Lin, T. Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., … Zitnick, C. L. (2014). Microsoft COCO: Common objects in context. *Proceedings of the European Conference on Computer Vision*, Springer, Berlin, 740–755.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y., & Berg, A. C. (2016). SSD: Single shot multibox detector. *Proceedings of European Conference on Computer Vision*, Springer, Berlin, 21–37.

Luo, P., Ren, J., Peng, Z., Zhang, R., & Li, J. (2018). *Differentiable learning-to-normalize via switchable normalization*. Retrieved from https://arxiv.org/abs/1806.10779/

Luo, W., Li, Y., Urtasun, R., & Zemel, R. (2016). Understanding the effective receptive field in deep convolutional neural networks. *Proceedings of Advances in Neural Information Processing Systems*, Barcelona, Spain, 4898–4906.

Maas, A. L., Hannun, A. Y., & Ng, A. Y. (2013). Rectifier nonlinearities improve neural network acoustic models. *Proceedings of the 30th International Conference on Machine Learning*, Atlanta, GA, 1–6.

Maeda, H., Sekimoto, Y., Seto, T., Kashiyama, T., & Omata, H. (2018). Road damage detection and classification using deep neural networks with smartphone images. *Computer-Aided Civil and Infrastructure Engineering*, *33*(12), 1127–1141.

Moghaddam, M. E., & Jamzad, M. (2006). Linear motion blur parameter estimation in noisy images using fuzzy sets and power spectrum. *EURASIP Journal on Advances in Signal Processing*, *2007*(1), 068985. https://doi.org/10.1155/2007/68985

Molina-Cabello, M. A., Luque-Baena, R. M., López-Rubio, E., & Thurnhofer-Hemsi, K. (2018). Vehicle type detection by ensembles of convolutional neural networks operating on super resolved images. *Integrated Computer-Aided Engineering*, *25*(4), 321–333.

Nishikawa, T., Yoshida, J., Sugiyama, T., & Fujino, Y. (2012). Concrete crack detection by multiple sequential image filtering. *Computer-Aided Civil and Infrastructure Engineering*, *27*(1), 29–47.

O'Byrne, M., Schoefs, F., Ghosh, B., & Pakrashi, V. (2013). Texture analysis based damage detection of ageing infrastructural elements. *Computer-Aided Civil and Infrastructure Engineering*, *28*(3), 162–177.

Oquab, M., Bottou, L., Laptev, I., & Sivic, J. (2014). Learning and transferring mid-level image representations using convolutional neural networks. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Columbus, OH, 1717–1724.

Ortega-Zamorano, F., Jerez, J. M., Gómez, I., & Franco, L. (2017). Layer multiplexing FPGA implementation for deep back-propagation learning. *Integrated Computer-Aided Engineering*, *24*(2), 171–185.

Prasanna, P., Dana, K., Gucunski, N., & Basily, B. (2012). Computer-vision based crack detection and analysis. *Proceedings of the SPIE, Sensors and Smart Structures Technologies for Civil, Mechanical, and Aerospace Systems*, San Diego, CA.

Rafiei, M. H., & Adeli, H. (2017). A novel machine learning-based algorithm to detect damage in high-rise building structures. *Structural Design of Tall and Special Buildings*, *26*(18). https://doi.org/10.1002/tal.1400

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. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Las Vegas, NV, 779–788.

Redmon, J., & Farhadi, A. (2017). YOLO9000: Better, faster, stronger. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Honolulu, HI.

Redmon, J., & Farhadi, A. (2018). *YOLOv3: An incremental improvement*. Retrieved from https://arxiv.org/abs/1804.02767/

Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster R-CNN: Towards real-time object detection with region proposal networks. *Proceedings of the Advances in Neural Information Processing Systems*, Montreal, 91–99.

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L. (2018). MobileNetV2: Inverted residuals and linear bottlenecks. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Salt Lake City, UT, 4510–4520.

Shrivastava, A., Gupta, A., & Girshick, R. (2016). Training region-based object detectors with online hard example mining. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Las Vegas, NV, 761–769.

Simonyan, K., & Zisserman, A. (2014). *Very deep convolutional networks for large-scale image recognition. Proceedings of the International Conference on Learning Representations*, San Diego, CA.

Singh, B., & Davis, L. S. (2018). An analysis of scale invariance in object detection—SNIP. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Salt Lake City, UT, 3578–3587.

Sun, C., Shrivastava, A., Singh, S., & Gupta, A. (2017). Revisiting unreasonable effectiveness of data in deep learning era. *Proceedings of the IEEE International Conference on Computer Vision*, Venice, Italy, 843–852.

Torres, J. F., Galicia, A., Troncoso, A., & Martínez-Álvarez, F. (2018). A scalable approach based on deep learning for big data time series forecasting. *Integrated Computer-Aided Engineering*, *25*(4), 335–348.

Uijlings, J. R., Van De Sande, K. E., Gevers, T., & Smeulders, A. W. (2013). Selective search for object recognition. *International Journal of Computer Vision*, *104*(2), 154–171.

Ulyanov, D., Vedaldi, A., & Lempitsky, V. (2016). *Instance normalization: The missing ingredient for fast stylization*. Retrieved from https://arxiv.org/abs/1607.08022/

Wang, G., Forsyth, D., & Hoiem, D. (2013). Improved object categorization and detection using comparative object similarity. *IEEE*

Transactions on Pattern Analysis and Machine Intelligence, *35*(10), 2442–2453.

Wang, L., Lu, Y., Wang, H., Zheng, Y., Ye, H., & Xue, X. (2017). Evolving boxes for fast vehicle detection. *Proceedings of the IEEE International Conference on Multimedia and Expo*, Hong Kong, 1135–1140.

Wang, P., & Bai, X. (2018). Regional parallel structure based CNN for thermal infrared face identification. *Integrated Computer-Aided Engineering*, *25*(3), 247–260.

Wu, Y., & He, K. (2018). Group normalization. *Proceedings of the European Conference on Computer Vision*, Munich, Germany, 3–19.

Yang, L., Li, B., Li, W., Liu, Z., Yang, G., & Xiao, J. (2017). Deep concrete inspection using unmanned aerial vehicle towards CSSC database. *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Vancouver, BC, 24–28.

Yang, X., Li, H., Yu, Y., Luo, X., Huang, T., & Yang, X. (2018). Automatic pixel-level crack detection and measurement using fully convolutional network. *Computer-Aided Civil and Infrastructure Engineering*, *33*(12), 1090–1109.

Yang, X., Sun, H., Sun, X., Yan, M., Guo, Z., & Fu, K. (2018). Position detection and direction prediction for arbitrary-oriented ships via multitask rotation region convolutional neural network. *IEEE Access*, *6*, 50839–55049.

Yosinski, J., Clune, J., Bengio, Y., & Lipson, H. (2014). How transferable are features in deep neural networks? *Proceedings of the Advances in Neural Information Processing Systems*, Montreal, 3320–3328.

Zalama, E., Gómez-García-Bermejo, J., Medina, R., & Llamas, J. (2014). Road crack detection using visual features extracted by Gabor filters. *Computer-Aided Civil and Infrastructure Engineering*, *29*(5), 342–358.

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. (2019). Surface damage detection for concrete bridges using single-stage convolutional neural networks. *Proceedings of the SPIE, Health Monitoring of Structural and Biological Systems XIII*, Denver, CO.

How to cite this article: Zhang C, Chang C-C, Jamshidi M. Concrete bridge surface damage detection using a single-stage detector. *Comput Aided Civ Inf*. 2019;1–21. https://doi.org/10.1111/mice.12500

APPENDIX

TABLE A1 Summary of detection results on testing dataset

			AP_{50}							AP_{75}						
Model	**Test image size**	**Testing time (s)**	**Crack**	**Pop-out**	**Spall**	**Rebar**	***Rec.***	***Prec.***	***mAP***	**Crack**	**Pop-out**	**Spall**	**Rebar**	***Rec.***	***Prec.***	***mAP***
Faster R-CNN	Shorter side = 600	0.537	0.708	0.668	0.627	0.762	0.764	0.586	0.691	0.313	0.373	0.364	0.419	0.502	0.406	0.367
Faster R-CNN w/improvements		0.539	0.765	0.718	0.681	0.810	0.815	0.656	0.744	0.392	0.437	0.436	0.472	0.575	0.482	0.434
Original YOLOv3	416	0.117	0.577	0.606	0.596	0.661	0.685	0.804	0.610	0.185	0.229	0.209	0.210	0.352	0.405	0.208
	448	0.130	0.599	0.644	0.606	0.700	0.691	0.810	0.637	0.189	0.233	0.214	0.223	0.366	0.431	0.215
	480	0.144	0.593	0.654	0.608	0.678	0.696	0.788	0.633	0.207	0.234	0.224	0.230	0.375	0.424	0.224
	512	0.160	0.631	0.679	0.624	0.713	0.724	0.786	0.662	0.219	0.255	0.238	0.248	0.399	0.436	0.240
	544	0.183	0.636	0.660	0.639	0.722	0.741	0.805	0.664	0.233	0.256	0.249	0.265	0.413	0.449	0.251
	576	0.196	0.633	0.641	0.613	0.709	0.724	0.770	0.649	0.229	0.262	0.257	0.290	0.415	0.446	0.260
	608	0.209	0.628	0.652	0.613	0.698	0.720	0.762	0.648	0.221	0.264	0.258	0.279	0.415	0.450	0.256
Improved YOLOv3	416	0.118	0.706	0.761	0.671	0.796	0.779	0.749	0.734	0.386	0.440	0.383	0.444	0.541	0.515	0.413
	448	0.129	0.730	0.785	0.721	0.804	0.806	0.764	0.760	0.400	0.477	0.410	0.464	0.572	0.539	0.438
	480	0.144	0.743	0.826	0.729	0.834	0.836	0.757	0.783	0.445	0.495	0.402	0.493	0.580	0.522	0.459
	512	0.160	0.765	0.827	0.739	0.866	0.833	0.772	0.799	0.456	0.494	0.432	0.508	0.592	0.545	0.472
	544	0.184	0.771	0.802	0.719	0.886	0.840	0.755	0.795	0.468	0.526	0.467	0.510	0.610	0.548	0.493
	576	0.195	0.746	0.804	0.710	0.879	0.828	0.730	0.785	0.493	0.542	0.518	0.556	0.644	0.565	0.527
	608	0.207	0.729	0.790	0.717	0.882	0.834	0.731	0.779	0.481	0.528	0.504	0.539	0.643	0.562	0.513