Deep-learning-based road crack detection frameworks for dashcam-captured images under different illumination conditions

Da-Ren Chen1 • Wei-Min Chiu1

Accepted: 1 June 2023 / Published online: 16 June 2023
© The Author(s), under exclusive licence to Springer-Verlag GmbH Germany, part of Springer Nature 2023

Abstract
Machine learning techniques have been used to increase detection accuracy of cracks in road surfaces. However, most studies failed to consider variable illumination conditions on the target of interest (ToI), and only focused on detecting the presence or absence of road cracks. This paper proposes a new road crack/defect detection method, IlumiCrack, which integrates Gaussian mixture models (GMM) and object detection CNN models. This work presents several contributions: firstly, a large-scale road crack/defect dataset was prepared using a dashcam with a variety of illumination scenarios. Secondly, experimental evaluations were conducted on 2–4 levels of brightness using GMM for optimal classification. Thirdly, the IlumiCrack framework integrates deep learning-based object detection framework, the YOLO and SSD, to classify road crack and defect images into eight types with high accuracy. In the model training phase, the localization loss was modified to Focal-EIoU, obtaining higher-quality anchor box. Comprehensive model precision and geometric mean (G-mean) achieve 79.1% and 77.1, respectively. Compared to YOLOv3 and SSD, IlumiCrack improves classification accuracy by at least 15.6% on two levels of brightness.

Keywords Convolution neural network • Gaussian mixture models • Road crack detection

1 Introduction
Climate factors and heavy-load transportation requirements have left road surfaces in Taiwan vulnerable to damage from natural disasters and daily wear and tear. Road cracks are usually scattered over the road surface, making it difficult to predict potential threats to road safety and vehicles. Traditional road inspection techniques are time and labor-intensive, and many local authorities have begun to outsource this work to the general driving public through telephone or online reporting (Citizen Service Hotline 2023). Drivers with smartphones or dashcams report the damage location and upload images or video of the damage to a central website. Local authority staffs then evaluate the images and schedule a road maintenance work.

In recent years, dashcams have become popular with drivers, with some models automatically uploading images and video to cloud storage, making them a promising source information for automatic road crack detection. In the development of traditional image detection techniques for road cracks, researchers empirically tuned a series of filters to differentiate cracks from background noise (Fujita and Hamamoto 2011; Protopapadakis et al. 2016; Sinha and Fieguth 2006; Turkan et al. 2018). Notwithstanding significant improvements in image processing-based crack detection methods, such approaches may not work without manually feature adjustment for varying crack patterns. In actual road conditions, these approaches have difficulty adapting to all scenarios due to complex illumination conditions (Saha et al. 2016; Taha et al. 2016) and varying crack shapes or textures (He et al. 2017). In recent years, researchers have sought to develop solutions by applying machine learning methods.

Deep learning (Elman 1990; Hinton et al. 2006; LeCun et al. 1989), a branch of machine learning, outperforms traditional image processing and other machine learning techniques (Citizen Service Hotline 2023; Zalama et al.
First, it simplifies the workflow of machine learning because it learns all features automatically without hand-crafted feature selections. Second, the incremental representations of its intermediate levels are learned jointly such that the feedback and parameters for each layer are dedicated to global optimization. Third, it applies multiple nonlinear layers of neural networks to better represent data, leading to better results or models.

In recent years, significant improvements have been made in road crack identification using novel deep learning frameworks such as Region with CNN (R-CNN) (2016a), You Only Look Once (YOLO) (Redmon et al. 2016; Zalama et al. 2014), Single Shot multibox Detection (SSD) (Maeda et al. 2018), etc. Zang et al. (2016a) photographed road cracks using a smart phone, and increased image sample diversity by using random shooting angles. They applied R-CNN to a preprocessed dataset to identify road cracks. Zalama et al. (2014) photographed road cracks using a car-based high-speed camera and trained the dataset using AdaBoost (Schwenk and Bengio 1997). Shi et al. (2016) identified road cracks using random structured forests in which the dataset are preprocessed jointly with diverse image channels. Maeda et al. (2018) collected road cracks through a car-based smart phone and proposed a classification method for the dataset. They applied the object detection framework SSD to train the dataset, obtaining considerable improvements to identification accuracy. Wan et al. (2022) improve the accuracy of YOLO detection model by employing Focal-EIoU loss during model training. They propose a new backbone network ShuffleECA-Net with BiFPN to improve the model weights and increase the speed of detection. Du et al. (2021) constructed a large-scale road crack dataset which composed of 45,788 images captured with a high-resolution industrial camera. Their model is also applicable to a variety of weather and illuminance conditions, but does not consider nighttime scenarios. Recent studies on road crack detection/identification are compared in Table 1.

Table 1 indicates that most studies do not consider the variability of illumination conditions. Moreover, apart from the study conducted by Maeda et al. (2018), other studies only focus on the patterns of road cracks and do not consider other road markings such as crosswalks and line dividers. In this paper, we first collect additional road datasets during different daylight periods through a low-cost dashcam so as to ensure the dataset better represents a wide variety of authentic road use conditions. Second, we analyzed the collected datasets to evaluate the impact of pixel variation on the performance of YOLO and SSD, which are state-of-the-art object detection frameworks. Third, we conducted a study to determine the optimal number of pixels HSV (Hue, Saturation, Value) categories that result in the highest detection accuracy. Based on Gaussian Mixture Model (GMM), we propose an object detection framework that integrates YOLO, SSD and neural networks to classify dataset images according to HSV features and to identify the location and class of individual road cracks. The performance of the framework is evaluated to determine the optimal number of brightness groups; it also compared to the ordinary YOLO and SSD to understand its outperformance on road crack detection.

The remainder of this paper is organized as follows. Section 2 describes the methodology and state-of-the-art object detection frameworks. Section 3 presents the proposed framework and algorithms. Section 4 describes the experimental setup and results. Finally, Sect. 5 draws concluding remarks.

## 2 Methodology

### 2.1 Gaussian mixture model (GMM)

Stauffer and Grimson (1999) applied GMM to create a background model that enhances the differentiation between background and foreground in an image. Different from Gaussian distribution, GMM considers the effect of prior probability to derive the probability of its Gaussian distributions and can be denoted as

$$
p(x|\theta) = \sum_{k=1}^{K} z_k N(x|\mu_k, \Sigma_k)
$$

where

$$
\sum_{k=1}^{K} z_k = 1, 0 \leq z_k \leq 1
$$

where $\Sigma_k$ denotes the covariance matrix, and $\mu_k$ denotes the average of distribution $k$ in a GMM model. Notation $z_k$ denotes the probability of distribution $k$ in the model. In most cases, the parameters are unknown to the GMM model, thus requiring an iterative algorithm to find the best parameter combinations. The expectation–maximization (EM) algorithm (Dempster et al. 1977) (Algorithm 1) finds the maximum likelihood or maximum posterior estimate of parameters, depending on unobserved latent variables in the model.
Algorithm 1 GMM-EM

Input : dataset $X$
Output : parameters of GMM
Step 1. Initialize $\{a_k, \mu_k, \Sigma_k\} | 1 \leq k \leq K$
Step 2. Derive the likelihood of sample $x_i$ in group $k$ using
\[
\gamma(i, k) = \frac{a_k^{(t)} n(x_i | \mu_k^{(t)}, \Sigma_k^{(t)})}{\sum_{j=1}^{K} n(x_i | \mu_j^{(t)}, \Sigma_j^{(t)})}
\]
Step 3. Re-estimate $a_k^{(t+1)}, \mu_k^{(t+1)}, \Sigma_k^{(t+1)}$ using $\frac{1}{n} \sum_{i} x_i$ and $\Sigma^{2} = \frac{1}{n} \sum_{i} (x_i - \mu)^2$
Step 4. Repeat Stage 2 and Stage 3 until the error between iterations $t+1$ and $t$ is less than 0.001

Table 1 Recent studies on road crack detection

|Strengths| Weakness|
|---|---|
|Zhang et al. (2022)| Low cost, smart phone-based data collection| Considers only crack patterns|
|Zalama et al. (2014)| Integrates multiple classifications| Considers only crack patterns|
|Shi et al. (2016)| Uses joint diverse image channels for preprocessing| Considers only daylight conditions|
|Maeda et al. (2018)| Proposes new classification for road cracks| Considers only crack patterns|
|Du et al. (2021)| Uses large-scale dataset| Considers only daylight conditions|
|Wan et al. (2022)| Uses Focal-EIOU to get higher-quality anchor box| Considers only daylight conditions|

Fig. 1 Convolution Neural Network (CNN) structure
GMM not only provides a smooth overall distribution fit, its components can clearly detail a multimodal density (Huang et al. 2005). Gao et al. (2000) have concluded that GMM outperforms single modal Gaussian distribution in background and pixel classification. Zeng et al. (2016) proposed a GMM-based color reduction algorithm that classifies images with a pixel color quantization representation. In this paper, the motivation to apply GMM to the pixel identification problem is that the GMM has been shown to be a powerful approach to text-independent speech verification and identification which bears a strongly similarity to the task of illumination and texture classification for images.

2.2 Convolution Neural Network (CNN)

A Convolutional Neural Network (CNN) is a deep learning algorithm first proposed by LeCun et al. (1989). CNN is widely used in image processing, where the regional features are convolved with the kernel to generate new outputs. Unlike traditional image processing methods where the kernel parameters are predefined, the parameters in convolutional layers are determined during the training process. CNN identifies the obvious features in the images using a convolution layer, reduces the image size in a pooling layer, and resolves linear classification through a fully connected layer with activation functions.

Using sliding window and convolution operations, the convolution layer shown in Fig. 1 assigns levels of importance through a predefined filter to various aspects or objects in the image and differentiates them from each other. These operations extract the high-level features, such as edges. From the input image, analogous to the connectivity pattern of neurons in the human brain. Conventionally, the first layer captures the low-level features such as edges, color, gradient orientation, etc. With additional layers, CNN adapts to the high-level features as well, giving us a network, which has a holistic understanding of the dataset images, similar to how the human brain processes images. The pooling layer is used to decrease computational time required to process the data through.
Fig. 4 Comparison of object detections

(a) Single feature map

(b) Pyramidal feature hierarchy

(c) Feature Pyramidal Network

Fig. 5 Overview of SSD framework

Fig. 6 Example of non-maximum suppression algorithm

Non-Maximum Suppression Algorithm

| Run 1 | Run 2 |
|-------|-------|
| Sort  | Sort  |
| 0.9   | 0.89  |
| 0.75  | 0.75  |
| 0.7   | 0.7   |

Confidence score

NMS

Selected objects

When all score = 0, then NMS stop.
dimension reduction. It also extracts dominant features which are positional and rotational invariant, thus ensuring effective model training. Max pooling returns the maximum value from a part of the image covered by the kernel. It mostly removes the noisy activations and also performs de-noising along with dimensionality reduction. Average pooling returns the average of all the values from a part of the image covered by the kernel. It performs dimensionality reduction merely as a noise suppressing mechanism.

After converting the input image into a fitting form for the multi-level perceptron, the image will be flattened into a column vector through the flatten layer. The flattened output is then fed to a feed-forward neural network with backpropagation applied to each training iteration. The fully connected layer shown in the right-hand side of Fig. 1, is used to learn nonlinear combinations of the high-level features as represented by the output of the convolutional layer. Over a series of epochs, the model can distinguish between dominating and certain low-level features in images and classify them using a Soft-max activation function. Soft-max, known as the normalized exponential function, is a nonlinear function which accelerates the training process and features complex models required in real-life (Carlile et al. 2017). In particular, it normalizes the multiple outputs of neurons into an interval between 0 and 1, which can be regarded as probability values over multiple classes. For example, output vector [9, 6, 3, 1, 0] from neurons correspond to the values [0.5294, 0.3157, 0.1578, 0.0526, 0.01342] of the soft-max function.

### 2.3 CNN-based object detection frameworks

Girshich et al. (2014) proposed an R-CNN framework with processing stages. In the first stage, selective search (Felzenszwalb and Huttenlocher 2004) is applied to identify and crop 2000 regions of equal size for each of the input images. The second stage performs CNN on those regions and outputs their features in vector form. Finally, a linear SVM is applied to classify those features. However, R-CNN could be time consuming because it requires more crops, leading to considerable redundant computation and memory space consumption from overlapping crops. Fast R-CNN (Girshick 2015) resolves this time-consuming and space redundancy problem by using a feature extractor for the entire image so that crops share the computation load of feature extraction. Compared with R-CNN, the framework of Fast R-CNN applies an ROI pooling layer analogous to Max pooling that compresses feature crops into $7 \times 7$ images which accelerates the processing of the full-connected layer. In addition, Fast R-CNN uses two different fully connected layers. The first is a soft-max regression
that predicts the object classification, and the second layer uses regression to predict the location of each object. The selective search is independent of Fast R-CNN and is also time consuming. Faster R-CNN proposed by Ren et al. (2015), remedies this performance issue by replacing selective search with a feature extractor called the Region Proposal Network (RPN) technique. However, Faster R-CNN cannot crop proposals directly from the image and re-runs crops through the feature extractor, leading to redundant computations.

YOLO is an object detection framework that can predict the region and class of objects with a single CNN. YOLO achieves considerable mean average precision as well as processing speed because background information and regression are involved. YOLO algorithm produces the locations of the bounding box of the object candidate and the confidence of the inference. YOLO has published several major versions. YOLO v1, announced in 2016 (Redmon et al. 2016), reduces the input images to $448 \times 448$ pixels inputs to the neural network, producing the probabilities and locations of object bounding boxes, filtering the duplicated bounding boxes using Non-Maximum Suppression (NMS), and identifying their final locations. YOLO achieves object detection speeds of up to 45fps on Titan X GPUs. In 2017, YOLO v2 (Redmon and Farhadi 2017) improved on the first version by replacing its fully connected layers with convolution layers and introduced the notion of an anchor box, inspired by Faster R-CNN. In Fig. 2, YOLO v3, announced in 2018 (Redmon and Farhadi 2018), is composed of Darknet-53 (Redmon 2013; Redmon et al. 2016) and Feature Pyramid Network (FPN) (Lin et al. 2017), and detects the shrunken input images with scales of 1/32, 1/16 and 1/8. Darknet-53 is applied for feature extraction and improves gradient descent using ResNet (He et al. 2016), proposed by He et al., because a deeper neural network has been shown to not necessarily provide improved precision. Instead, a deeper neural network may incur a vanishing gradient or gradient exploding problem. Therefore, YOLO v3 modifies the VGG19 network with the notion of a “shortcut” shown in Fig. 3. Figure 3a describes an ordinary neural network while Fig. 3b shows a neural network with an output residual block denoted as $H(x) = F(x) + x$. When a residual $F(x)$ approximates 0 during a reverse-propagation stage, YOLO v3 skips this layer using the shortcut. FPN has been applied to road crack detection (Du et al. 2021; Lin et al. 2017; Wan et al. 2022). Figure 4a shows how an image is detected using single scale applied to Fast R-CNN and Faster R-CNN. Prediction rates fall when input image sizes are non-uniform. In Fig. 4b, the SSD object detection framework detects images with different scales. Figure 4c presents the principle of FPN which extracts the features from different image scales in a bottom-up fashion, while enhancing these image scale features in a top-down fashion.

Proposed by Liu et al. (2016), SSD adopts the feature extractor VGG-16 (Simonyan and Zisserman 2014). Similar to YOLO, SSD runs a CNN on the input image only once to learn the representations. A series of $3 \times 3$ convolution filters are performed on these representations to predict the bounding boxes and class probability. The key feature of SSD is its use of pyramid feature hierarchy to predict objects with different scales after using multiple convolutional layers. Figure 5 presents an overview of the SSD framework, where fully connected layers fc6 and fc7 are replaced with convolution layers conv6 and conv7 such that the SSD expands the coverage of convolution detection without additional parameters or complex models (Chen et al. 2017). Due to the multi-scale property, SSD
| Class ID | Class name                                      | Sample image                                                                 |
|----------|------------------------------------------------|-------------------------------------------------------------------------------|
| D00      | *Longitudinal crack*                          | ![Image](image-url)                                                           |
|          | parallel to the pavement centerline or laydown direction |                                                                              |
| D01      | *Longitudinal construction joint (crack)*     | ![Image](image-url)                                                           |
|          | is created when concrete is placed in stages or pours, particularly in large concrete structures such as bridges and highways |                                                                              |
| D10      | *Lateral crack*                                | ![Image](image-url)                                                           |
|          | single cracks perpendicular to the pavement’s centerline or laydown direction |                                                                              |
| D11      | *Lateral construction joint (crack)*          | ![Image](image-url)                                                           |
|          | a type of joint used in construction to connect two adjacent sections of roads or concrete |                                                                              |
| D20      | *Alligator crack*                              | ![Image](image-url)                                                           |
|          | fatigue cracking on pavement caused by repeated traffic loading, aging, or poor maintenance |                                                                              |
| D40      | *Pothole (defect)*                             | ![Image](image-url)                                                           |
|          | the result of moisture infiltration and usually the end result of untreated alligator cracking, along with vertical sharp edge near the top of the hole |                                                                              |
| D43      | *Faded crosswalk (defect)*                     | ![Image](image-url)                                                           |
|          | a worn-out crosswalk is less visible to drivers and put pedestrians at risk |                                                                              |
| D44      | *Blurred lane divider (defect)*                | ![Image](image-url)                                                           |
|          | a worn-out lane separator less visible to drivers |                                                                              |
produces 8732 default boxes for each class according to the convolution layers with different aspect ratios.

Using the non-maximum suppression (NMS) technique (Neubeck and Gool 2006), SSD filter out default boxes with low probabilities. The example shown in Fig. 6 illustrates two objects with four default boxes on the top-left corner. Each default box is scored and sorted in descending order. The default box with the highest score is computed with other default boxes using the Intersection over Union (IoU) operation. When the IoU of those boxes exceeds a certain threshold, the boxes with lower confidence scores are set to 0, otherwise the box is not relevant to the highest score box. The default box with the highest score is labeled with a select object. The NMS algorithm is performed until all default boxes score less than 0. In Eq. (3), IoU loss function calculates the intersection union ratio of the prediction boundary box and the ground truth box.

\[
L_{\text{IoU}} = 1 - \frac{B \cap B_G}{B \cup B_G}
\]  

where \( B \cap B_G \) is the intersection of the prediction boundary box and the ground truth box, and \( B \cup B_G \) is the union of the prediction boundary box and the ground truth box. YOLO v3 uses the traditional CIoU loss function, which considers the distance between the center points of the predicted box and the center points of the ground truth bounding box, as shown in Eq. (4).

\[
L_{\text{CIoU}} = 1 - \text{IoU} + \frac{\rho^2(b, b_G)}{c^2} + \alpha v
\] 

where \( \rho^2(b \cap b_G) \) is the distance between the center points of the predicted box and the ground truth bounding box. \( c \) is the diagonal length of the smallest enclosing box covering the predicted box and the ground truth bounding box, and \( \alpha v \) denotes the aspect ratio between the predicted box and the ground truth bounding box. Accurately predicting the ground truth bounding box using CIoU loss can be challenging due to the variable size and diverse types of road damage (Wan et al. 2022). To address this issue, EIoU in Eq. (4) utilizes CIoU loss to calculate the overlap loss and center distance loss, while also incorporating the minimum difference between the width and height of the predicted box and the ground truth bounding box. This approach enables the model to converge faster and achieve higher accuracy.

\[
L_{\text{EIoU}} = 1 - \text{IoU} + \frac{\rho^2(b, b_G)}{c^2} + \frac{\rho^2(w, w_G)}{C_w^2} + \frac{\rho^2(h, h_G)}{C_h^2}
\] 

where \( C_w \) and \( C_h \) denote the width and height of the smallest enclosing box covering the predicted box and the ground truth bounding box, respectively. Using EIoU may create an issue of unbalanced data samples in the road...
crack dataset, as it may result in a smaller number of high-quality anchor boxes with small regression errors in comparison to a larger number of low-quality samples with large regression errors. Furthermore, training with poor quality samples can cause large gradients and affect the overall training process. The proposed method uses Focal-EIoU loss (Wan et al. 2022; Zhang et al. 2022), as shown in Eq. (6), to address imbalanced datasets and improve accuracy loss:

\[
L_{\text{Focal-EIoU}} = \text{IoU} \gamma L_{\text{EIoU}}
\]

where \( \gamma \) controls the degree of inhibition of outliers, thus enabling better accuracy in the detection process.

Table 5 List of clusters

| Qty. of clusters (K) | Cluster ID list       |
|----------------------|-----------------------|
| 2                    | PD2-1, PD2-2          |
| 3                    | PD3-1, PD3-2, PD3-3   |
| 4                    | PD4-1, PD4-2, PD4-3, PD4-4 |

One-stage detectors such as YOLO and SSD are known as their simplicity and efficiency, but they may suffer from a higher number of false positives and an imbalance between foreground and background examples. To address this challenge, SSD adopts hard negative mining (Zhang et al. 2022), which only keeps a small set of hard background examples for training. The background and foreground examples are re-weighted by focal loss such that the hard examples are assigned with large weights. Moreover, in bounding box regression (BBR), the imbalance problem still exists because most anchor boxes have small overlaps with target boxes. While only a small number of boxes are informative for object localization, the most irrelevant boxes with small IOUs can produce excessively large gradients that are inefficient for training regression models. Therefore, the CIoU is replaced with the focal and
efficient IoU (Focal-EIoU) (Zhang et al. 2022) loss function, which solves the CIoU problem and improve the quality of anchor boxes.

3 Proposed method

The proposed framework shown in Fig. 7 is composed of classification and training phases. In the classification phase, a large number of road crack images are gathered over a single day. Those images are formulated as vectors according to HSV (hue, saturation, value) representations and processed with dimensionality reduction. Before the training phase, the dataset is classified in terms of pixel HSV and image features. Sections 3.2 and 3.3, respectively, discuss the effectiveness of pixel and image classification, along with how many groups of pixels/images would contribute to the best precision rate.

3.1 Dataset dimensionality reduction

Luminance information plays a critical role in recognizing road crack classes and affects color perception (Saha et al. 2016; Taha et al. 2016). To better analyze color and brightness features, using the HSV color space is preferable over the RGB color space. In order to improve the proposed classifier’s accuracy, we explore the relationship between road crack types, pixel hue, brightness, or lightness and group the datasets into \( k \) clusters. These clusters are then used to train the proposed model and improve its precision rate.

HSV has been applied to classify an image as daytime or nighttime images (Saha et al. 2016). Hue and value thresholds are set to group images into daytime and nighttime images. Taha et al. (2016) applied this technique to characterize cracks as being portrayed in daytime or nighttime images, and to isolate them from noise. Assume \( X \) denotes the dataset of road images and can be defined in Eq. (7)

\[
X = \begin{bmatrix}
x_{11} & x_{12} & \cdots & x_{1m} \\
x_{21} & x_{22} & \cdots & x_{2m} \\
\vdots & \vdots & \ddots & \vdots \\
x_{d1} & x_{d2} & \cdots & x_{dm}
\end{bmatrix} \in \mathbb{R}^{d \times m}, x_{ij}
\]

\[
= (H_{ij}, S_{ij}, V_{ij})^T,
\]

where \( d \) denotes the number of images and \( m \) denotes the number of pixels. Variable \( x_{ij} \) presents the HSV value of the \( j \)th pixel in \( i \)th image where \( H \), \( S \) and \( V \), respectively, denote the pixel’s hue, saturation and value. To consolidate the image and pixel dimensions while preserving their

### Table 7  Mean AP (mAP) of the target methods (%)

| Algorithms          | D00 | D01 | D10 | D11 | D20 | D40 | D43 | D44 |
|---------------------|-----|-----|-----|-----|-----|-----|-----|-----|
| SSD                 | 48.1 | 61.0 | 44.6 | 47.5 | 66.7 | 80.7 | 83.9 | 71.3 |
| YOLO                | 46.6 | 69.0 | 53.0 | 56.3 | 56.4 | 48.5 | 77.4 | 72.7 |
| \( K = 2 \) IlumiCrack(PD2-1) + SSD | 65.4 | 71.9 | 58.5 | 53.4 | 81.8 | 60.2 | 77.9 | 77.3 |
| \( K = 2 \) IlumiCrack(PD2-2) + SSD | 68.9 | 64.3 | 61.8 | 63.0 | 82.2 | 55.6 | 82.6 | 85.1 |
| \( K = 2 \) IlumiCrack(PD2-1) + YOLO | 43.4 | 72.1 | 52.4 | 57.1 | 63.6 | 52.6 | 82.2 | 75.5 |
| \( K = 2 \) IlumiCrack(PD2-2) + YOLO | 53.7 | 69.8 | 57.5 | 59.5 | 53.2 | 48.4 | 76.6 | 73.9 |
| \( K = 3 \) IlumiCrack(PD3-1) + SSD | 38.3 | 51.3 | 34.0 | 23.9 | 51.7 | 32.0 | 54.9 | 67.1 |
| \( K = 3 \) IlumiCrack(PD3-2) + SSD | 40.1 | 63.8 | 33.1 | 26.1 | 64.0 | 34.6 | 77.4 | 78.4 |
| \( K = 3 \) IlumiCrack(PD3-3) + SSD | 46.9 | 69.9 | 28.2 | 25.4 | 66.4 | 34.5 | 81.0 | 67.2 |
| \( K = 3 \) IlumiCrack(PD3-1) + YOLO | 27.1 | 68.6 | 30.8 | 20.5 | 45.3 | 30.2 | 64.4 | 59.8 |
| \( K = 3 \) IlumiCrack(PD3-2) + YOLO | 41.9 | 67.5 | 20.5 | 30.1 | 50.4 | 30.6 | 77.9 | 72.6 |
| \( K = 3 \) IlumiCrack(PD3-3) + YOLO | 41.5 | 64.1 | 19.2 | 9.5 | 61.4 | 31.1 | 84.1 | 68.7 |
| \( K = 4 \) IlumiCrack(PD4-1) + SSD | 43.4 | 63.0 | 19.4 | 11.5 | 66.5 | 23.1 | 76.9 | 72.3 |
| \( K = 4 \) IlumiCrack(PD4-2) + SSD | 32.2 | 58.6 | 22.5 | 13.7 | 55.1 | 19.6 | 76.5 | 66.5 |
| \( K = 4 \) IlumiCrack(PD4-3) + SSD | 36.8 | 17.3 | 21.5 | 21.6 | 35.5 | 18.5 | 55.5 | 54.4 |
| \( K = 4 \) IlumiCrack(PD4-4) + SSD | 39.0 | 38.1 | 23.9 | 15.3 | 72.8 | 63.5 | 67.1 | 74.9 |
| \( K = 4 \) IlumiCrack(PD4-1) + YOLO | 38.5 | 64.3 | 18.5 | 13.7 | 67.7 | 0.3 | 74.5 | 68.5 |
| \( K = 4 \) IlumiCrack(PD4-2) + YOLO | 34.4 | 55.8 | 7.5 | 17.9 | 49.4 | 13.7 | 79.2 | 62.1 |
| \( K = 4 \) IlumiCrack(PD4-3) + YOLO | 28.2 | 71.9 | 32.2 | 26.2 | 47.0 | 3.1 | 60.8 | 61.0 |
| \( K = 4 \) IlumiCrack(PD4-4) + YOLO | 27.8 | 54.8 | 38.8 | 52.4 | 65.8 | 91.0 | 62.1 | 71.3 |
HSV (Chen et al. 2017; Saha et al. 2016), Eq. (7) can be rewritten as

\[ X = \left[ x_1 \ x_2 \ \cdots \ x_{(d \times m)} \right] \in \mathbb{R}^{(d \times m)}, \quad x_i = (H_i, S_i, V_i)^T. \]  

(8)

3.2 Pixels classification

To understand how many groups of pixels substantially improves object detection, the number of groups of pixels HSV are given as \( P = \{2, 3, 4\} \). Based on HSV values, the image pixels can be denoted as a Gaussian distribution \( N(x | \mu_p, \sum_p) \) shown in Eq. (9),

\[ p(x | \theta) = \sum_{p=1}^{P} \pi_{p}N(x | \mu_p, \sum_p), \]  

(9)

where \( \theta \) denotes the parameter. When the dataset is divided into \( P \) groups, \( x_p \) denotes the probability of the individual pixel group, \( \mu_p \) is the reference HSV, and \( \sum_p \) is the pixel distribution. The sum of probability of the pixel groups is 1.

\[ \sum_{p=1}^{P} x_p = 1, \quad 0 \leq x_p \leq 1, \]  

(10)

To find satisfactory parameters in GMM, the EM algorithm is applied with initialized parameters shown in Algorithm Pixel Classification. In Step 2, the pixel values \( x_p \) are substituted into Eq. (11), deriving the probability of pixel \( x_j \) in each group. In Step 3, set \( x_i = \gamma(i, k) \), we derive new values of \( x_p, \mu_p, \) and \( \sum_p \) by substituting \( \mu \) and \( \sigma^2 \) derived from Eqs. (12) and (13) for Eq. (11). The process will repeat until error improvement falls below a predefined threshold.

\[ \gamma(j, p) = \frac{x_p \mathcal{N}(x_j | \mu_p, \sum_p)}{\sum_{b=1}^{P} x_b \mathcal{N}(x_j | \mu_b, \sum_b)} \]  

(11)

\[ \mu = \frac{1}{n} \sum_j x_j \]  

(12)

\[ \sigma^2 = \frac{1}{n} \sum_j (x_j - \mu)^2 \]  

(13)
Algorithm Pixel Classification

Input:  \( X = [x_1 \ x_2 \ \cdots \ x_{(d\times m)}] \in \mathbb{R}^{(d\times m)} \),  \( x_i = (H_i, S_i, V_i)^T \)

Output: GMM classification model

Step 1. Initialize parameters  \( \{\{\alpha_p, \mu_p, \Sigma_p\} \ | \ 1 \leq p \leq P\} \),  \( P = \{2,3,4\} \)

Step 2. Compute probability \( \gamma(i, k) \) of  \( x_i \) being classified to group  \( k \)

Step 3. Let \( x_i = \gamma(i, k) \), re-estimate the values of  \( \alpha_p, \mu_p \) and \( \Sigma_p \)

Step 4. Repeat Step 2 and Step 3, until the difference between the errors of iteration \( t+1 \) and \( t \) is less than 0.001

### 3.3 Image classification

After classifying the pixels in accordance with HVS values into 2, 3, and 4 categories, images can be classified in terms of different proportions of pixel categories. To perform image classification, an image’s pixel proportions should be referred to by pixel properties. By using one-hot-encoding, the pixel proportions of an image can be formulated by

\[
\text{Image}_\text{ratio} = \sum_{\text{pixel}} \text{oneHotEncoding}(\text{GMM Pixel}(\text{pixel}))
\]

\[
p(x|\theta) = \sum_{i=1}^{I} c_{i} N(x|\mu_{i}, \Sigma_{i})
\]

\[
\sum_{i=1}^{I} c_{i} = 1, \quad 0 \leq c_{i} \leq 1.
\]

In addition to discussing the effectiveness of the number of clusters on the CNN object detection framework, the number of groups in pixel HSV are predetermined in \( I = \{2, 3, 4\} \). As shown in Eq. (15), an image is composed of the pixels of (2, 3, or 4) groups, where  \( \theta \) denotes the parameters of GMM,  \( x_i \) defined in Eq. (16) is the probability of individual distribution,  \( \mu_i \) is the average of group \( i \), and  \( \Sigma_i \) is the variance of group \( i \). To find the  \( \theta \) for image classification, the EM algorithm is applied to Algorithm Image Classification with input images and a predetermined number of categories in \( K \). Step 2 derives the value of \( \gamma(i, k) \) denoting the probabilities of image \( x_i \) belonging to pixel proportion \( k (k < K) \) by substituting parameters  \( \alpha_k, \mu_k \) and  \( \Sigma_k \) in Eq. (16). In the Step 3, set  \( \gamma(i, k) = x_i \), the new values of  \( \alpha_k, \mu_k \) and  \( \Sigma_k \) are derived from Eq. (11) by substituting  \( \mu \) and  \( \sigma^2 \) from Eqs. (12) and (13). Repeat Steps 2 and 3 until the difference between  \( t \) and  \( t + 1 \) error converges within 0.001.

Algorithm Image Classification

Input:  \( X = [x_1 \ x_2 \ \cdots \ x_d] \in \mathbb{R}^d \),  \( x_i = \text{Image}_{\text{bright ratio}} \)

Output: GMM classification model

Step 1. Initialize parameters  \( \{\{\alpha_k, \mu_k, \Sigma_k\} \ | \ 1 \leq k \leq K\} \),  \( K = \{2,3,4\} \)

Step 2. Compute probability  \( \gamma(i, k) \) of image \( x_i \) belonging to group  \( k \) of image proportion

Step 3. Set  \( \gamma(i, k) = x_i \), re-estimate the values of  \( \alpha_p, \mu_p \) and  \( \Sigma_p \)

Step 4. Repeat Step 2 and Step 3, until the difference between the errors of iteration \( t+1 \) and \( t \) is less than 0.001
3.4 Framework setting

YOLO applies an input size of 416 × 416 with momentum of 0.9 to reduce the possibility of local optimization. The learning rate is initialized to 0.001 and is decayed 1/10 when it performs over 400,000 steps, and decayed 1/10 again when it reaches 450,000 steps. Higher learning rates benefits training speed, and decreasing learning rates over time increases detection precision. Due to the limited GPU memory size, the batch size is set to 8, i.e. obtaining eight random samples once from the training set.

Figure 8 illustrates the architecture darknet-53 used by YOLO v3, where Convolutional and Residual, respectively, denote the convolutional and residual layers. A 3 × 3 convolutional filter benefits network depth and decreases the number of parameters while increasing valuable features (Simonyan and Zisserman 2014). Instead of a pooling layer, YOLO v3 uses a 3 × 3 convolution layer with a stride of 2 to improve feature loss. A 1 × 1 convolutional layer aims to synthesize features instead of convolutional computation. The activation function \( f(x) = \max(ax, x) \) behind a convolution layer is Leaky ReLU so that the weights can be still updated when \( x < 0 \) during the reverse propagation phase.

The input to the SSD network has a size of 300 × 300 with momentum of 0.9, learning rate of 0.001 and batch size of 8 at random. Figure 9 shows the architecture VGG consists of a majority of 3 × 3 fully connected layers. Unlike YOLO v3, we scales SSD pictures down by applying maxpool with a filter size of 2 and a stride of 2 (Liu et al. 2016). In addition to the ordinary SSD shown in Fig. 5, we modified the SSD to include convolution layers after the last maxpool layer. Without changing the output size, the filter and padding size of those convolution layers are, respectively, set to 3 and 1 to further reduce information redundancy. The convolutional layer next to the last maxpool layer has a padding of 6 to produce various image scales for the subsequent convolutional layers.

### 4 Experiments

#### 4.1 Data preparation

In this section, we provide an overview of the dataset, including how it was acquired, organized, and labeled. In most previous studies on road crack detection, images are usually simplified or emphasized the objects of interest.
Table 10 Overall criteria of the target methods (%)  

| Algorithms                  | Algorithms                  | mAP  | Precision | Recall | F1 score | G-mean |
|-----------------------------|-----------------------------|------|-----------|--------|----------|--------|
| SSD                         | 51.8                        | 61.6 |           | 57.7   | 63.7     | 64.7   |
| YOLO                        | 52.9                        | 61.3 |           | 59.4   | 61.4     | 66.7   |
| $K = 2$                     |                             |      |           |        |          |        |
| IlumiCrack (PD2-1 & SSD)    | 79.5                        | 80.3 |           | 79.9   | 77.4     | 79.3   |
| IlumiCrack (PD2-2 & SSD)    | 81.7                        | 79.1 |           | 79.6   | 79.6     | 77.1   |
| IlumiCrack (PD2-1 & YOLO)   | 71.1                        | 80.0 |           | 72.8   | 71.1     | 72.0   |
| IlumiCrack (PD2-2 & YOLO)   | 67.7                        | 70.9 |           | 73.7   | 70.3     | 76.3   |
| $K = 3$                     |                             |      |           |        |          |        |
| IlumiCrack (PD3-1 & SSD)    | 55.8                        | 53.9 |           | 60.8   | 56.1     | 64.4   |
| IlumiCrack (PD3-2 & SSD)    | 56.8                        | 62.0 |           | 58.6   | 59.1     | 61.1   |
| IlumiCrack (PD3-3 & SSD)    | 55.9                        | 58.3 |           | 65.2   | 59.9     | 59.3   |
| IlumiCrack (PD3-1 & YOLO)   | 58.2                        | 61.8 |           | 63.7   | 61.7     | 58.3   |
| IlumiCrack (PD3-2 & YOLO)   | 55.2                        | 50.9 |           | 54.0   | 51.2     | 67.4   |
| IlumiCrack (PD3-3 & YOLO)   | 57.1                        | 62.2 |           | 54.9   | 55.9     | 55.2   |
| $K = 4$                     |                             |      |           |        |          |        |
| IlumiCrack (PD4-1 & SSD)    | 47.9                        | 50.6 |           | 48.1   | 50.1     | 56.0   |
| IlumiCrack (PD4-2 & SSD)    | 48.7                        | 48.8 |           | 55.1   | 56.8     | 51.1   |
| IlumiCrack (PD4-3 & SSD)    | 48.3                        | 46.6 |           | 50.9   | 51.3     | 54.7   |
| IlumiCrack (PD4-4 & SSD)    | 47.4                        | 49.9 |           | 48.6   | 49.9     | 48.9   |
| IlumiCrack (PD4-1 & YOLO)   | 49.8                        | 50.5 |           | 51.2   | 53.5     | 51.8   |
| IlumiCrack (PD4-2 & YOLO)   | 50.0                        | 55.8 |           | 50.7   | 55.9     | 53.4   |
| IlumiCrack (PD4-3 & YOLO)   | 51.1                        | 60.0 |           | 52.9   | 58.0     | 49.7   |
| IlumiCrack (PD4-4 & YOLO)   | 50.7                        | 51.5 |           | 53.9   | 59.6     | 52.4   |

Fig. 14 Object detection results

from the local-trimmed pictures. However, these images are challenging to obtain and may not be representative of real-life scenarios. To address this issue, we collected authentic road images captured by vehicle-based cameras, such as dashcams, to train models that can be used in practical situations. The widespread use of dashcams or smartphones in cars allows any driver to easily upload captured images to models running in cloud computing environments, or even running natively on stand-alone mobile devices. Similar to the categories proposed in Maeda et al. (2018), the damage types are divided into eight categories described in Table 2, which are further classified into two main groups: road cracks and road defects. Within the crack group, there are linear crack/joint and alligator cracks, and the defect groups include not only potholes, but also blurred road markings. Notably, categories D00 and D10 both pertain to irregular cracks resulting from natural or human-induced damage to the road surface, while categories D01 and D11 do not road damages, but rather represent the regular joints formed...
after road construction. While some categories, such as D10 and D11, may appear less representative, there are still notable differences in their actual characteristics and their impact on road users and maintenance resort.

The images used in this paper were obtained from two sources. Firstly, a review of 163,664 smartphone-captured road images from Maeda et al. (2018) was conducted, of which 8404 contained cracks. These 8404 images were manually tagged with class labels, as per the definitions presented in Table 2. Secondly, given that the images obtained from Maeda et al. (2018) were exclusively taken under daytime lighting conditions, a dashcam (SJCAM SJ5000) was installed on the dashboard of a motorcycle (see Fig. 10), which recorded video as the motorcycle was...
driven around Taiwan city streets. Subsequently still images measuring 600 × 600 pixels were extracted from the video files. The images from both sources were combined to create a dataset, as listed in Table 3, and were manually tagged with their respective class labels. The proportions of each category are shown in Fig. 11. Furthermore, the number of daytime and nighttime images were proportionately assigned to each image class of training and testing datasets. The fuzzy transitive images between daytime and nighttime usually appear during dusk and dawn. The HSV-based day/night detectors proposed in Liu et al. (2016) and He et al. (2016) were applied to these images, using the hue-histogram and value-histogram on the top half of the images to threshold them, accounting for the influences of the misclassified images. Some of the images misclassified by the detectors were classified as misclassified nighttime and misclassified daytime images. The number of daytime, nighttime, misclassified nighttime and misclassified daytime images are summarized in the Table 4.

It is worth noting that most of the misclassified daytime images were captured during dawn, within an hour after streetlights were turned off. Some of the misclassified nighttime images were captured during dusk, within an hour before streetlights were turned on. Other misclassified nighttime images were taken in urban downtown scenes with turned-on street lamps, banner spotlights, and display/screen for information or advertisements. In the training process, these misclassified images occupied 1/12 proportionally in the daytime and nighttime images in the datasets before the training process, respectively. Despite various illumination scenarios, these misclassified nighttime images achieved a high precision rate comparable to the performance on daytime images.

During image preprocessing, pictures were transformed to the data type of floating point. To make full use of the dataset and to improve the generalization ability of the model, the images containing various types of cracks were preferentially used. After careful selection, feature extraction was conducted using Scikit-Image package for Python 3.7. to collect the feature values of Contrast, Energy Entropy, and HSV of the images. The feature values of the collected images were standardized to equalize the scale before model training. The values were normalized using Eq. (17) and scaled into interval [0, 1].

\[
\hat{z}_i = \frac{x_i - \min(x)}{\max(x) - \min(x)} \tag{17}
\]

According to the needs of the experiment, the images were randomly divided into a training set, and a test set, according to 9:1. To address overfitting, we applied repeat n-fold cross validation (CV) to our experiment, where n = 10 is a common value in applied machine learning. This traditional approach produces a nearly unbiased but highly variable estimator. In n-fold CV, a model is trained on n − 1 folds of the data, and validated on the remaining fold. However, a single run of n-fold CV can be noisy, leading to different distribution of performance scores for different splits of dataset (Kuhn and Johnson 2013). To mitigate this, we repeated the tenfold CV procedure five times to obtain a more accurate estimate of model performance. This involved fitting and evaluating 50 different models. By collaborating CV with bootstrap approaches in the road crack classification, we achieved a model with a balance trade-off between bias and variance errors.

During the model training phase, Focal-EIoU loss function was employed in lieu of the conventional IoU loss function to address the issue of imbalanced samples and to enhance the precision of the bounding box. Moreover, a bootstrap technique was employed to reduce the variability of the estimator for small sample sizes. This method is well-known to deliver superior performance in situations where the sample size is small, owing to its reduced variance. To further mitigate imbalanced data quantity of
Fig. 19 Training and test performance of datasets using SSD

(a) overall
(b) PD2-1
(c) PD2-2
(d) PD3-1
(e) PD3-2
(f) PD3-3

a without category, b in group 1 with two categories, c in group 2 with two categories, d in group 1 with three categories, e in group 2 with three categories and f in group 3 with three categories
various image types and prevent under-fitting during model training, it is often necessary to supplement the amount of data. More specifically, the original images were duplicated and modified by the image data generator in *imgaug* library for image augmentation methods such as distortion, expansion, trimming, rotation and zooming on specific parts. An example is presented in Fig. 12. Eventually, those pictures were resized to 300 × 300. For this purpose, an application that can process road images was developed and is publicly available on our website. All experiments were performed using an Intel Core i5-7400 3.00 GHz CPU with 16 GB DDR4 RAM and NVidia GeForce GTX 1080 GPU. The proposed framework was constructed using Python 3.7 with a Pytorch neuro network.

4.2 Pixel difference with HSV value

As mentioned Sect. 3.2, one of the most important goals of this paper is to discuss the effectiveness of illumination on a picture in terms of a pixel’s HSV values. We also wonder how many categories of HSV are required for optimal results. Following Algorithm Image Classification, the number of categories \( k \in K = \{2, 3, 4\} \) are investigated. Notation \( PD_k-n \) denotes the Pixel Difference in \( k \) classes where \( n \) denotes the \( n \)th class shown in Table 5. The clustering results are shown in Fig. 13.

We employed *imgaug* to augment the datasets, which were subsequently transformed to the HSV color space by applying thresholding before conducting GMM-EM clustering. Figure 13a–c, respectively, shows examples of clustering with the pixel difference dividing the dataset into two, three and four categories. Different colors in each figure represent a GMM model and each point denotes a pixel with an \( X-Y-Z \) coordinate corresponding to respective values of Hue, Saturation and Value. Abundant cluster point overlaps (e.g., the overlap between the two lower clusters in Fig. 13b) may interfere with image brightness assessment. When a cluster blurred, it becomes an insignificant factor for object detection.

---

Fig. 20 Training and test performance of datasets in a group 1, b group 2, c group 3 and d group 4 with four categories
Fig. 21 Training and test performance of datasets using YOLO a in group 1 with two categories, b in group 2 with two categories, c in group 1 with three categories, d in group 2 with three categories and e in group 3 with three categories
4.3 Performance evaluations

In the experiments, we utilized three commonly used metrics, namely Accuracy, F1 score and G-mean, derived from precision and recall rates, to objectively evaluate the classifier’s performance with an IoU threshold of 0.5 and a confidence threshold of 0.4. Accuracy represents the percentage of data that is correctly classified, F1 score is a harmonic mean of precision and recall, while G-mean, or geometric mean, is derived from the square root of precision and recall. These metrics are derived from the confusion matrix shown in Table 6, where True Positive (TP) and True Negative (TN), respectively, indicate the number of samples in the positive and negative classes that are correctly classified, while False Negative (FN) and False Positive (FP), respectively, indicate the number of samples in the positive and negative classes that are incorrectly classified. Accuracy, F1 score and G-mean are defined in Eqs. (18)–(20),

\[
\text{Accuracy} = \frac{TP + TN}{TP + FN + FP + TN} \quad (18)
\]

\[
\text{F1 score} = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \quad (19)
\]

\[
\text{G-mean} = \sqrt{\text{precision} \times \text{recall}} \quad (20)
\]

where precision is defined as TP/(TP + FP) and recall is defined as TP/(TP + FN). It should be noted that G-mean is a reliable metric for evaluating classifier accuracy on both the majority and minority classes when the datasets described in Sect. 4.1 are proportionally increased or decreased. This metric aims to optimize accuracies on all classes while ensuring balance among them. In the case of a multi-class problem, G-mean is the higher root of the product of sensitivities for each class. A low G-mean value indicates poor performance in classifying positive cases, regardless of the correct classification of negative cases. G-mean helps prevent overfitting of the negative class and underfitting of the positive class. The F1-score (Okran et al. 2022) is a useful metric for evaluating binary classification.

Fig. 22 Training and test performance of datasets using YOLO in a group 1, b group 2, c group 3 and d group 4 with four categories.
models where there is an imbalance between positive and negative samples. It can also be used for multi-class classification problems by computing the F1 score for each class separately and then taking the average. Average precision (AP), a popular metric for measuring the accuracy of object detectors, was first introduced by Everingham et al. (Everingham et al. 2009) in the 2010 PASCAL Visual Object Classes (VOC) competition. AP computes the average precision value for recall value over 0–1 on the basis of Eqs. (21) and (22). Precision measures prediction accuracy, i.e., the percentage of correct predictions, and the Recall rate finds the percentage of possible positive cases in our top N predictions.

\[
AP = \frac{1}{N} \sum_{r \in \{0.0, 1, \ldots, 1\}} p_{\text{interp}}(r) \quad (21)
\]

\[
p_{\text{interp}}(r) = \max_{\frac{r_1}{r_2} \geq r} p(r) \quad (22)
\]

The performance of the proposed IlumiCrack framework, which utilizes pixel and image categorization based on HSV and illuminations, is compared with the SSD model improved in Maeda et al. (2018) and YOLOv3 under eight types of road crack datasets in at most four categories of brightness given in Fig. 11 and Table 4. In Tables 7, 8, 9, and 10, the boldface highlights the significant outperformance of the proposed framework using SSD, particularly when K = 2. The experimental results presented in Table 7 demonstrate that IlumiCrack outperforms SSD and YOLO in terms of mAP for all classes of road crack images when k = 2. Despite the increased difficulty in distinguishing between D11 and D10, and D00 and D01, the proposed IlumiCrack (PD2) + SSD still outperforms the SSD and YOLO models without HVS and brightness categorization. Furthermore, the proposed approach with k = 2 outperforms SSD and YOLO models with k > 2, indicating the superiority of using two categories of HSV for achieving the best precision rate. We also compared their F1 score and G-mean in terms of different number of brightness categories, respectively, shown in Tables 8 and 9. The average F1 score of SSD and YOLO was found to be 63.7% and 61.4%, respectively, while the average F1 score of IlumiCrack(PD2) + SSD and IlumiCrack(PD2) + YOLO was 71.8% and 69.4%, respectively. Table 9 shows that the average G-mean of SSD and YOLO was 64.7% and 66.7%, respectively, while the average G-mean of IlumiCrack(PD2) + SSD and IlumiCrack(PD2) + YOLO was 71.5% and 72.1%, respectively. Table 9 also reveals that IlumiCrack(PD2) + YOLO outperforms IlumiCrack(PD2) + SSD addressing imbalanced datasets. Similar to the model precision shown in Table 7, all models are able to detect alligator cracks and potholes (D20 and D40) at significant accuracy compared to all other road crack types. The accuracy of detecting road defects (D43 and D44) is much higher compared to that of detecting road cracks.

Table 10 summarizes the performance metrics, including mAP, precision, recall, F1 score, and G-mean, of the target methods across various brightness clustering configurations. The results show that the proposed method achieves superior performance with only two brightness clusters (i.e. k = 2), outperforming the methods with three or more clusters.

### 4.4 Implementation results

The experimental results shown in Figs. 14, 15, 16, 17 and 18 are derived from the IlumiCrack(PD2-1) + SSD and IlumiCrack(PD2-2) + SSD models. In Figs. 14 and 15, IlumiCrack(PD2) + SSD can detect damages with high reliability and accuracy. On the basis of GMM model with HSV values, it is not affected by uneven light intensity and has strong resistance to the external environment. In Figs. 16, 17, and 18, in the case of low exposure and partial shadow occlusion, IlumiCrack(PD2) + SSD provides considerable detection confidence and can detect pits of small objects in the different illuminations. This shows that Focal-EIoU loss’s treatment of sample imbalance enhances the detection ability of the model for small objects to a certain extent. By optimizing the sample imbalance processing method, the sensitivity of small object recognition can be improved. Therefore, the missed detection rate of targets with unclear features can be reduced, and better road defect detection performance can be achieved.

### 4.5 Training performance

This section presents the performance of models under training and test datasets. Figures 19 and 20 present that the training loss and test loss decrease rapidly before 20 epochs, based on the SSD framework. In Fig. 19a, the model without categories exhibits higher fluctuation in training loss than the models with two and three categories (see Fig. 19b–f). This indicates that brightness categories can benefit crack/defects detection models in reducing training loss. However, as shown in Fig. 20, categorizing datasets in four groups leads to considerably higher training loss. Therefore, an over-categorized framework may not benefit crack detection due to the limited image patterns or features. Overall, the proposed framework, with or without categories, exhibits low overfitting as the test losses decrease smoothly after 10 epochs, indicating good performance on the test dataset.

Figures 21 and 22 illustrate the training and test performance of YOLO. The training loss of PD2 exhibits significant fluctuations in a wide spectrum of epochs,
whereas PD3 and PD4 show relatively stable training loss. The presence of overfitting is evident in both figures, where the test loss increases rapidly after 50 epochs, indicating that the YOLO framework performs worse than the SSD framework for road crack/defect detection. Despite the inclusion of various illumination conditions, road surface images are relatively uniform compared to other applications. Therefore, the dataset with two categories achieves the best performance for images containing road surfaces in the background.

5 Conclusions

Different from previous studies on road crack detection, we focus on detecting cracks/defects under various illumination conditions. To achieve this goal, we have constructed a new extensive dataset of images captured in poorly lit conditions, including nighttime scenarios. These images were combined with existing datasets of road images obtained in Maeda et al. (2018, and manually verified and classified those images that depict road cracks or defects into eight distinct categories. The proposed framework initially employs GMM to classify HVS pixels into two, three and four groups. The dataset images are then categorized as brightness based on the proportions of these groups, resulting in two, three and four classes. We found that a two-class brightness achieves the best precision and G-mean for feature recognition using the proposed HVS categories. To address imbalanced datasets and improve bounding box accuracy loss, we also employ Focal-EIoU approach. The proposed framework outperforms ordinary YOLO v3 and SSD under various brightness conditions. Experiment results indicate that pixel classification with HSV values improves automatic road crack/defect detection in environments with varying illumination conditions, whereas the use of 3 or 4 brightness levels does not improve model accuracy. In future research, we intend to incorporate the proposed method into cloud-based systems, allowing road maintenance agencies to automate and outsource crack detection and identification to the public.

Acknowledgments We would like to express our gratitude to all colleagues and students who made valuable contributions to this study. I am also grateful to my student Yu-Hao Chen, who helped in data collection. We also acknowledge the editor and reviewers for their constructive criticisms of a previous version of this paper.

Author contributions D-RC have substantial contributions to the conception and design of the work. He also planned and performed the acquisition, analysis, and interpretation of data for the work. The drafting and revising work were also completed by him. W-MC performed the data acquisition, and assisted D-RC in analyzing and organizing materials for the work.

Data availability Enquiries about data availability should be directed to the authors.

Declarations

Conflict of interest I and another author declare that we have no conflict of interest and Competing interests.

References

Carlile B, Delamarter G, Kinney P, Marti A (2017) Improving deep learning by inverse square root linear units (ISRLUs). arXiv preprint 2017
Chen LC, Papandreou G, Kokkinos I, Murphy K, Yuille AL (2017) Deeplab: semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. IEEE Trans Pattern Anal Mach Intell 40(4):834–848
Citizen Service Hotline. Construction bureau of Taichung City government, 2023 (online). Available: https://itunes.apple.com/tw/app/%E8%87%BA%E4%B8%AD%E5%AE%95%E5%A5%BD%E8%A1%8C/id932232922m?mt=8
Dempster AP, Laird NM, Rubin DB (1977) Maximum likelihood from incomplete data via the EM algorithm. J Roy Stat Soc: Ser B (methodol) 39(1):1–22
Du Y, Pan N, Xu Z, Deng F, Shen Y, Kang H (2021) Pavement distress detection and classification based on YOLO network. Int J Pavement Eng 22(13):1659–1672
Elman JL (1990) Finding structure in time. Cogn Sci 14(2):179–211
Everingham M, Van Gool L, Williams CK, Winn J, Zisserman A (2009) The pascal visual object classes (VOC) challenge. Int J Comput Vision 88:303–308
Felzenszwalb PF, Huttenlocher DP (2004) Efficient graph-based image segmentation. Int J Comput Vis 59:167–181
Fujita Y, Hamamoto Y (2011) A robust automatic crack detection method from noisy concrete surfaces. Mach vis Appl 22:245–254
Gao X, Boult TE, Coetzee F, Ramesh V (2000, June) Error analysis of background adaption. In: Proceedings IEEE conference on computer vision and pattern recognition. CVPR 2000 (Cat. No. PR00662), 2000
Girshick R (2015) Fast r-cnn. In: Proceedings of the IEEE international conference on computer vision
Girshick R, Donahue J, Darrell T, Malik J (2014) Rich feature hierarchies for accurate object detection and semantic segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition
He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition
He K, Girshick R (2017) Mask r-CNN. In: Proceedings of the IEEE international conference on computer vision
Hinton GE, Osindero S, Teh YW (2006) A fast learning algorithm for deep belief nets. Neural Comput 18(7):1527–1554
Huang Y, Englehart KB, Hodgins B, Chan AD (2005) A Gaussian mixture model based classification scheme for myoelectric control of powered upper limb prostheses. IEEE Trans Biomed Eng 52(11):1801–1811
Kuhn M, Johnson K (2013) Applied predictive modeling, vol 26. Springer, New York, p 13
LeCun Y, Boser B, Denker JS, Henderson D, Howard RE, Hubbard W, Jackel LD (1989) Backpropagation applied to handwritten zip code recognition. Neural Comput 1(4):541–551
Lin TY, Dollár P, Girshick R, He K, Hariharan B, Belongie S (2017) Feature pyramid networks for object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition
Liu W, Anguelov D, Erhan D, Szegedy C, Reed S, Fu CY, Berg AC (2016) Ssd: single shot multibox detector. In: Computer vision—ECCV 2016: 14th european conference, Amsterdam, The Netherlands, 2016 October 11–14
Maeda H, Sekimoto Y, Seto T, Kashiyama T, Omata H (2018) Road damage detection using deep neural networks with images captured through a smartphone. arXiv preprint
Neubeck A, Van Gool L (2006) Efficient non-maximum suppression. In 18th international conference on pattern recognition (ICPR’06) August
Okran AM, Abdel-Nasser M, Rashwan HA, Puig D (2022) Effective deep learning-based ensemble model for road crack detection. In: 2022 IEEE international conference on big data (Big Data), 2022 December
Protopapadakis E, Stentoumis C, Doulamis N, Doulamis A, Loupos K, Makantasis K, Amidis A (2016) Autonomous robotic inspection in tunnels. ISPRS Ann Photogramm Remote Sens Spatial Inf Sci 3:167–174
Redmon J (2013) Darknet: open source neural networks in C (online). Available: http://pjreddie.com/darknet/
Redmon J, Farhadi A (2017) YOLO9000: better, faster, stronger. In: Proceedings of the IEEE conference on computer vision and pattern recognition
Redmon J, Farhadi A (2018) Yolov3: an incremental improvement. arXiv preprint arXiv:1804.02767
Redmon J, Divvala S, Girshick R, Farhadi A (2016) You only look once: unified, real-time object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition
Ren S, He K, Girshick R, Sun J (2015) Faster r-cnn: towards real-time object detection with region proposal networks. In: Advances in neural information processing systems, vol 28
Saha B, Davies D, Raghavan A (2016) Day night classification of images using thresholding on HSV histogram. U.S. Patent and Trademark Office Patent U.S., Washington DC, Patent No. 9530556
Schwenk H, Bengio Y (1997) Training methods for adaptive boosting of neural networks. In: Advances in neural information processing systems, vol 10
Shi Y, Cui L, Qi Z, Meng F, Chen Z (2016) Automatic road crack detection using random structured forests. IEEE Trans Intell Transp Syst 17(12):3434–3445
Simonyan K, Zisserman A (2014) Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556
Sinha SK, Fieguth PW (2006) Automated detection of cracks in buried concrete pipe images. Autom Constr 15(1):58–72
Stauffer C, Grimson WEL (1999) Adaptive background mixture models for real-time tracking. In: Proceedings. 1999 IEEE computer society conference on computer vision and pattern recognition (Cat. No PR00149), 1999 June
Taha M, Zayed HH, Nazmy T, Khalifa M (2016) Day/night detector for vehicle tracking in traffic monitoring systems. Int J Comput Inf Eng 10(1):98–104
Turkan Y, Hong J, Laflamme S, Puri N (2018) Adaptive wavelet neural network for terrestrial laser scanner-based crack detection. Autom Constr 94:191–202
Wan F, Sun C, He H, Lei G, Xu L, Xiao T (2022) YOLO-LRDD: a lightweight method for road damage detection based on improved YOLOv5s. EURASIP J Adv Signal Process (1), 98
Zalama E, Gómez-García-Bermejo J, Medina R, Llamas J (2014) Road crack detection using visual features extracted by Gabor filters. Comput Aided Civ Infrastruct Eng 29(5):342–358
Zeng S, Huang R, Wang H, Kang Z (2016) Image retrieval using spatiograms of colors quantized by gaussian mixture models. Neurocomputing 171:673–684
Zhang YF, Ren W, Zhang Z, Jia Z, Wang L, Tan T (2022) Focal and efficient IOU loss for accurate bounding box regression. Neurocomputing 506:146–157
Zhang L, Yang F, Zhang YD, Zhu YJ (2016a) Road crack detection using deep convolutional neural network. In: 2016a IEEE international conference on image processing (ICIP), 2016a, September

Publisher’s Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.