Front. Struct. Civ. Eng. 2024, 18(4): 516–535
https://doi.org/10.1007/s11709-024-1040-z

RESEARCH ARTICLE

Investigation of crack segmentation and fast evaluation of crack propagation, based on deep learning

Than V. TRAN[a], H. NGUYEN-XUAN[b*], Xiaoying ZHUANG[a,c*]

[a] *Institute of Photonics, Leibniz University Hannover, Hannover 30167, Germany*
[b] *CIRTech Institute, HUTECH University, Ho Chi Minh City 700000, Vietnam*
[c] *College of Civil Engineering, Tongji University, Shanghai 200092, China*
[*] *Corresponding authors. E-mails: ngx.hung@hutech.edu.vn; zhuang@iop.uni-hannover.de*

ABSTRACT Identifying crack and predicting crack propagation are critical processes for the risk assessment of engineering structures. Most traditional approaches to crack modeling are faced with issues of high computational costs and excessive computing time. To address this issue, we explore the potential of deep learning (DL) to increase the efficiency of crack detection and forecasting crack growth. However, there is no single algorithm that can fit all data sets well or can apply in all cases since specific tasks vary. In the paper, we present DL models for identifying cracks, especially on concrete surface images, and for predicting crack propagation. Firstly, SegNet and U-Net networks are used to identify concrete cracks. Stochastic gradient descent (SGD) and adaptive moment estimation (Adam) algorithms are applied to minimize loss function during iterations. Secondly, time series algorithms including gated recurrent unit (GRU) and long short-term memory (LSTM) are used to predict crack propagation. The experimental findings indicate that the U-Net is more robust and efficient than the SegNet for identifying crack segmentation and achieves the most outstanding results. For evaluation of crack propagation, GRU and LSTM are used as DL models and results show good agreement with the experimental data.

KEYWORDS deep learning, crack segmentation, crack propagation, encoder−decoder, recurrent neural network

1 Introduction

In engineering, most structural components present material defects due to local damage during manufacturing, operation, and maintenance. Cracks are one of the common defects and can be found on the surface of various structural components; they represent important information about the durability and safety of an entire structure. Under different load conditions, these defects can coalesce to form larger cracks, and this is one of the main causes of structural collapse [1]. Therefore, the detection and evaluation of the growth of cracks become very important and have great significance for repair and maintenance. One of the most common methods of detecting defects is human-based visual inspection, due to its simplicity and shortage of alternatives [2]. However, due to the variety and large number of defects, the defect detection task is not only time-consuming but also involves significant costs. In addition, the detection ability and experience of the inspectors significantly affect the quality of the process. Therefore, reliable alternative methods are needed to improve the accuracy and efficiency of damage detection.

To overcome the drawbacks of manual inspection, various methods of detecting and evaluating the propagation of cracks have been proposed and widely studied by many researchers. For example, several advances in the extended finite element formulation combined with the genetic algorithm to detect any type of defect (cracks or holes) of any shape in structures have been reported in Ref. [3]. Vu-Bac et al. [4] proposed the node-based smoothed extended finite element method to analyze fracture problems related to two-dimensional (2D) elasticity. A fully-automated method named CrackTree, to detect cracks from images, was studied in Ref. [5]. Butcher et al. [6] used random neural architectures, a

Article history: Received Apr 23, 2023; Accepted Jun 27, 2023

non-invasive technique, to detect defects in reinforced concrete. Yagi et al. [7] used the tetrahedral finite element model to evaluate the crack propagation behaviors in a T-shaped tubular joint. The virtual element method was used to analyze crack propagation in 2D [8]. Wang et al. [9] used the effective notch stress method to evaluate the crack propagation and fatigue strength of rib-deck welds. Several other approaches have been employed such as Refs. [10–16]. The aforementioned methods exhibited impressive results. However, their main drawback is their large computational and time cost.

Deep learning (DL) has in recent years gained significant traction in numerous domains for performing complex tasks, including computer vision, and speech recognition, natural language processing, and machine translation. It has outperformed traditional methods in some applications such as image classification [17–20], object detection [21–23], image segmentation [24,25], and time series forecasting [26–28]. DL is a data-driven approach that uses mathematical functions to map the examples of the input to that of the output without any manual intervention. The model building process consists of three main steps that fix a suitable network structure, a loss function for the model to learn, and an optimizer to update the model's parameters. In the field of crack segmentation and prediction of crack propagation, an example of a project investigates buildings or bridges uses a controlled drone that flies around buildings or bridges to take pictures of different surfaces. (This is especially meaningful in places that are difficult for humans to reach.) The images are then processed by a computer to identify potential areas on buildings or bridges that could be showing damage and assess the development of that damage. An accurate model reduces the human effort that is needed to process these photos, reducing or eliminating the time-consuming, costly and inefficient requirement for inspectors to check each photo. Currently, the crack segmentation and evaluation of crack propagation methods based on DL have been used with outstanding results [29–35]. Since there is a lot of data available and there has been a dramatic increase in hardware performance, DL-based methods outperform other methods. Herein, crack detection and crack propagation prediction using an automated DL show high accuracy with the highest computational efficiency, saving implementation time and cost. Furthermore, we can use the skeleton extraction algorithm to measure the characteristics of the crack, such as length, width, and area of the crack from the result of crack segmentation [36].

The DL algorithms search for different patterns and trends. There is no single algorithm that is best for all data sets or in all cases. To find the best solution, we need to test all the possibilities by conducting many observations, tuning their hyper-parameters, and evaluating the outputs to choose the one that gives the best result. This might not be a problem initially, but when dealing with large amounts of data, even a single epoch can take a considerable amount of time. Therefore, it is extremely important to choose a suitable algorithm.

To the best of the authors' knowledge, there have not been many studies that comprehensively explore the issue of cracking, especially cracks in concrete surfaces, with DL. Specifically, in this study, five common loss functions (namely binary cross-entropy (BCE) loss, dice loss, Tversky loss, focal Tversky loss, and Lovasz-Softmax loss) in general segmentation problems are investigated in concrete crack segmentation problem. In addition, gated recurrent unit (GRU) and long short-term memory (LSTM) are investigated in evaluating crack propagation. Notably, GRU has been less explored than LSTM in the literature. Considering the huge potential of DL algorithms, this study aims to propose DL networks to identify cracks in concrete surface images and predict crack propagation in structures. We use SegNet and U-Net networks to automatically segment cracks on images. Then, GRU and LSTM networks are employed to predict crack propagation on experimental examples.

To sum up, the main contributions of this work are as follows.

1) The frameworks and basic characteristics of SegNet, U-Net, GRU, and LSTM are described. In addition, various loss functions (which are used for imbalanced data and small region of interest (ROI) segmentation) are explored, as well as optimizers and model evaluation metrics are presented.

2) The approach for the concrete crack segmentation task was investigated using SegNet and U-Net models (which are automatic crack segmentation methods based on convolutional neural networks (CNNs)). Both models are explored by experimenting with various loss functions and two different optimizers to evaluate and compare the performance of each model (based on the intersection over union (IoU) metric). The model with the most outstanding prediction results is given, and is ready for practical application.

3) By experimental investigation, case studies are carried out to evaluate and demonstrate the effectiveness of GRU and LSTM models in the task of predicting crack propagation, and are implemented on crack public data sets such as fatigue analysis of a fuselage panel, ADB610 steel specimen, and L-shape concrete specimen.

The remaining sections of the paper are structured as follows. Section 2 represents deep neural network architectures for crack segmentation (SegNet and U-Net architectures) and crack propagation evaluation (GRU and LSTM architectures). In addition, the loss functions, optimization algorithms and model evaluation metrics used in this study are also presented in Section 2. Section 3 provides a brief description of the data sets used, and

the performance of the proposed models is demonstrated and compared in detail with experimental results. In addition, some discussions regarding our approach are also provided in this section. The conclusions of the paper and suggestions for further research are given in Section 4.

2 Methodology

2.1 Encoder–decoder network

In DL, the CNNs are a class of artificial neural network (ANN) which is most commonly applied to analyze and process visual imagery. In the DL community, particularly for computer vision tasks, it is one of the most extensively utilized architectures. CNN was first introduced by Fukushima et al. [37] in their paper on the "Neocognitron" for a visual pattern recognition mechanism. Subsequently, Waibel et al. [38] introduced CNN for phoneme recognition. In 1998, LeCun et al. [39] proposed a CNN architecture for document recognition. Convolutional, pooling and fully connected (FC) layers are the three types of layers that commonly make up a CNN. Features from the input images are extracted using the convolutional layer (Fig. 1). Herein, the convolution operation is carried out between the input image and a filter with a specific size. The filter moves across the input image, the dot product is obtained between parts of the input image and the filter according to the size of the filter. The pooling layer (Fig. 2) is often followed by convolution layers that aims to scale down the feature map's size in order to reduce calculation expenses. Max pooling and average pooling are the two most commonly used kinds of pooling operations. In some models, convolutional layer with stride > 1 is used instead of a pooling layer to reduce data size. The FC layer includes neurons along with the weights and biases that are used to link neurons between the two distinct layers. These layers are frequently positioned before the output layer, which

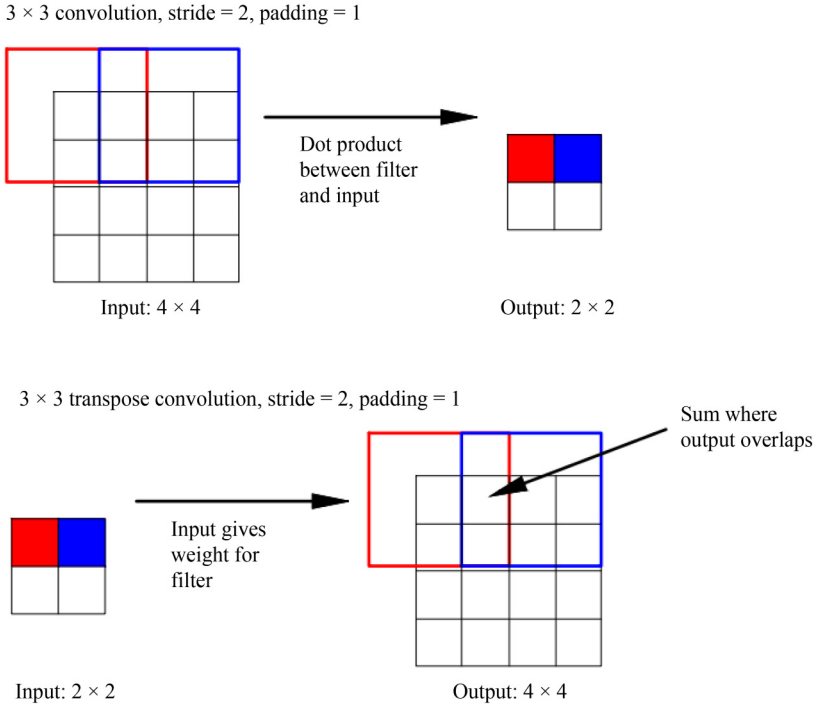


Fig. 1 Convolution and transposed convolution layers example. Both operations use a 3 × 3 kernel, stride of two and padding of one.

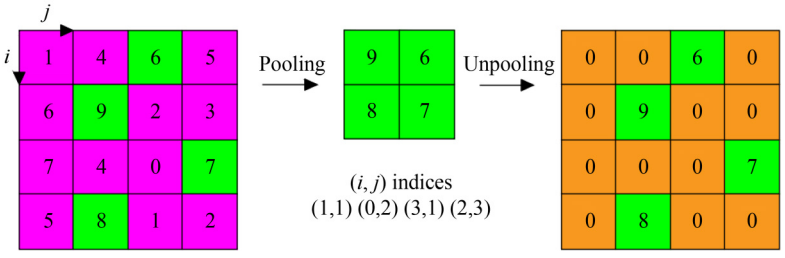


Fig. 2 Max-pooling and max-unpooling layers example. To upsample the feature maps, the max-pooling indices are kept up and reused.

makes up the final few layers of the CNN architectures.

Encoder−decoder architectures [40] are commonly used in segmentation models based on DL. Such models learn how to map data from the input domain to the output domain through a two-stage (encoder and decoder) network as shown in Fig. 3. Convolutional layers and pooling layers are used at the encoder stage to down-sample the input image. A latent space representation called z is created by implementing an encoding function called $z = f(x)$, which compresses the input x. In contrast, the decoder stage consists of convolutional layers and unpooling layers (or transposed convolution layers) to upsample the input images. It implements a decoding function $y = g(z)$ to predict the output y from the latent space representation z. The semantic information of the input is captured by the latent space representation z, which is helpful for predicting the outcome. The transpose convolution (Fig. 1) and unpooling (Fig. 2) are the reverse operations of the convolution and pooling operations, respectively.

2.1.1 SegNet architecture

SegNet is an image segmentation network with a fully convolutional encoder−decoder architecture that was introduced by Badrinarayanan et al. [41]. As depicted in Fig. 4, the SegNet architecture is composed of an encoder network, a corresponding decoder network, and finally a pixel-wise classifier. The 13 convolutional layers of the encoder network's architecture are exactly the same as the 13 convolutional layers in the VGG16 network [42], which was created for image classification. The number of parameters in the encoder network has been significantly reduced as a result of the removal of the FC layers in order to maintain higher resolution feature maps at the encoder output. There is a corresponding decoder layer for each encoder layer, resulting in a total of 13 convolutional layers in the architecture of the decoder network. A multiclass Softmax classifier receives the output of the final decoder and generates class probabilities for each individual pixel. In addition, the decoder conducts nonlinear upsampling using pooling indices implemented in the corresponding encoder's maxpooling phase. The decoder network's role is to convert full input resolution feature maps from low-resolution encoder feature maps so that pixel-wise classification may be performed.

2.1.2 U-Net architecture

U-Net is a U-shaped encoder−decoder architecture proposed by Ronneberger et al. [43] for segmenting biomedical images, as illustrated in Fig. 5. It is made to learn efficiently from fewer training samples, which is one of the methods most frequently employed in semantic segmentation tasks. The U-Net architecture is composed of an encoder network (contracting path) to collect context and a symmetric decoder network (expansive path) to enable exact localization; the two paths are connected via a bridge. The contracting path is designed according to the typical architecture of a CNN. It performs the role of a feature extractor by using a sequence of encoder blocks to learn an abstract representation of the input image. A semantic segmentation mask is produced from the abstract representation using the expanded path. By using a skip connection, the expanded path is joined with the matching feature map from the encoder blocks. Due to the depth of the network, these skip connections provide additional features from previous layers that might be lost during learning. A 1 × 1 convolution with a sigmoid activation function is

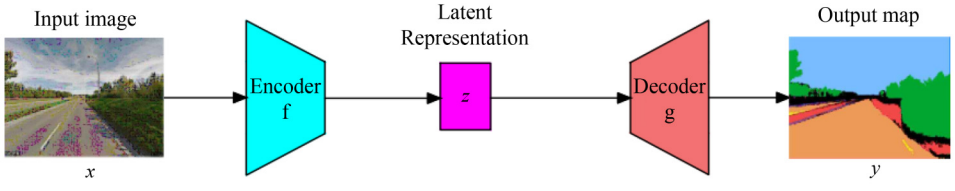


Fig. 3 The encoder−decoder network's architecture.

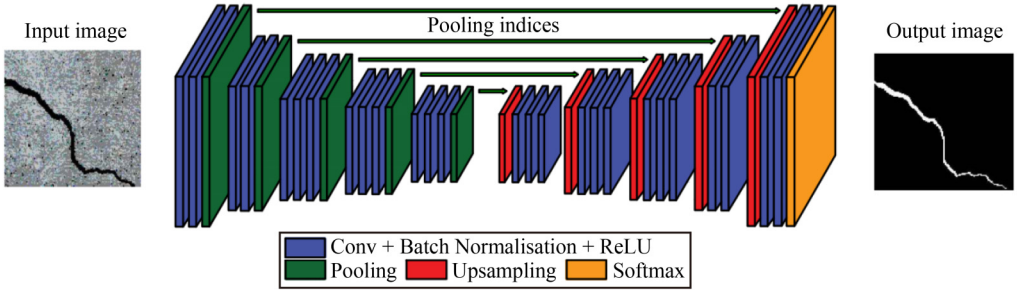


Fig. 4 The SegNet architecture. It is only convolutional layers.

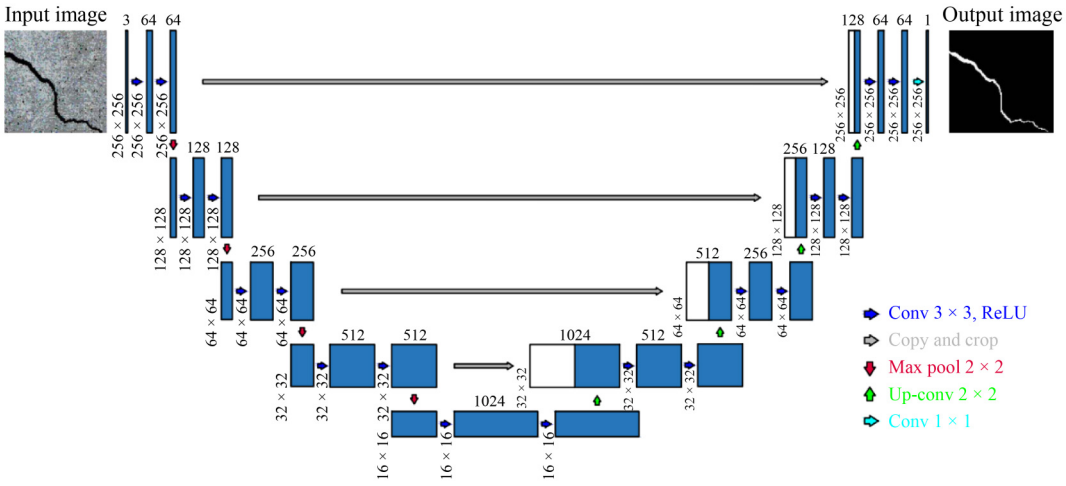


Fig. 5 The U-Net architecture. The different operations are denoted by arrows.

employed to process the last output of the expansive path. The sigmoid activation function produces a segmentation mask that represents pixel-wise categorization.

2.2 Recurrent neural network (RNN)

The RNNs [44] are a sort of ANN that is well-suited to processing time-series data and other sequential data. The current output of RNN is based on the previous information in the sequence. Here we briefly introduce RNN as a feedforward network extension that can handle variable-length sequences, as well as some of the most common recurrent architectures in use, namely LSTM [45] and GRU [46].

The basic architecture of the RNN unit is shown in Fig. 6. Its input includes the output of the previous phase (h_{t-1}) and the current input (x_t) passed through tanh activation function, with no gates present:

$$h_t = \tanh\left(W_{\mathrm{h}} \cdot [h_{t-1}, x_t] + b_{\mathrm{h}}\right), \tag{1}$$

where all weights W_{h} and biases b_{h} are parameters to be learned, tanh is an activation function.

Because of the issue with vanishing gradients, simple RNN is unable to learn long-term relationships from data; this is a major drawback. The vanishing gradient affects the RNN more than other neural network architectures as it processes more steps. To address this issue, the LSTM and GRU architectures were developed.

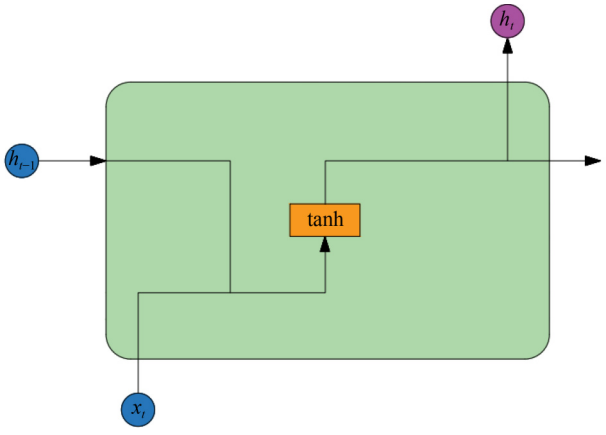


Fig. 6 The basic architecture of a RNN cell.

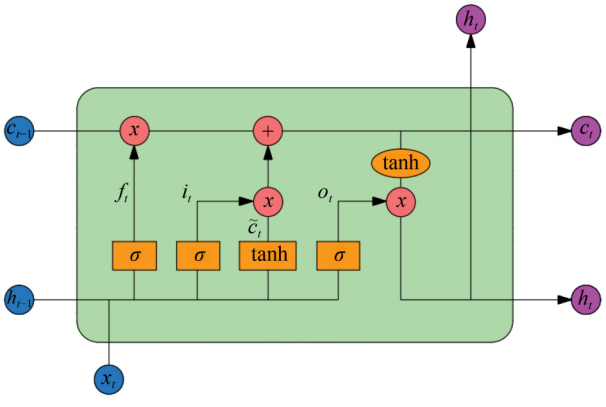


Fig. 7 The basic architecture of LSTM cell.

2.2.1 Long short-term memory

The LSTM was created to address the issue of vanishing gradient and since then it has become one of the most widely used RNN architectures. The workflow of LSTM is similar to that of RNN, with the exception of the operations performed within the LSTM unit. Various variants of LSTM exist [47], but the most common variant of LSTM is shown in Fig. 7 and described by:

$$f_t = \sigma\left(W_{\mathrm{f}} \cdot [h_{t-1}, x_t] + b_{\mathrm{f}}\right), \tag{2}$$

$$i_t = \sigma\left(W_{\mathrm{i}} \cdot [h_{t-1}, x_t] + b_{\mathrm{i}}\right), \tag{3}$$

$$o_t = \sigma\left(W_{\mathrm{o}} \cdot [h_{t-1}, x_t] + b_{\mathrm{o}}\right), \tag{4}$$

$$\tilde{c} = \tanh\left(W_{\mathrm{c}} \cdot [h_{t-1}, x_t] + b_{\mathrm{c}}\right), \tag{5}$$

$$c_t = f_t \odot c_{t-1} + \bar{t}_t \odot \tilde{c}_t, \tag{6}$$

$$h_t = o_t \odot \tanh(c_t), \tag{7}$$

where σ, tanh are logistic sigmoid function, hyperbolic tangent function, respectively; $\odot$ is denoted for point-wise multiplication of the two vectors.

2.2.2 Gated recurrent unit

The GRU was suggested as a simpler alternative to the LSTM (as it employs one fewer gate and does not distinguish between hidden states and memory cells) and has since gained popularity. It works by the same mechanism as the LSTM to alleviate the vanishing gradient problem. The most popular variant of GRU is shown in Fig. 8 and defined as follows:

$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t] + b_z), \tag{8}$$

$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t] + b_r), \tag{9}$$

$$\tilde{h} = \tanh(W_h \cdot [r_t \odot h_{t-1} + x_t] + b_h), \tag{10}$$

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t. \tag{11}$$

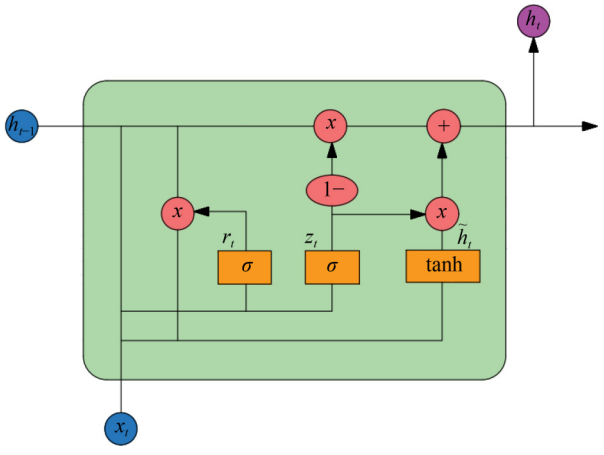


Fig. 8 The basic architecture of GRU cell.

2.3 Loss functions

One of the most important components of DL-based image segmentation is the loss function, sometimes referred to as the objective function, and its purpose is to evaluate how well the predicted segmentation compares with the ground truth. It is crucial to choose the correct loss function when designing a DL architecture, because it promotes the learning process of algorithms. To learn and optimize an objective function, DL algorithms use a stochastic gradient descent (SGD) method. We need to ensure a mathematical representation of our loss function, which can cover even edge cases if we want a model to learn more accurately and faster. The loss functions are derived from traditional machine learning, in which these loss functions measure the disparity between predicted and actual labels in a model. Categorical cross-entropy, for instance, is produced from the Multinoulli distribution, while BCE is derived from the Bernoulli distribution. In our study, we emphasize semantic segmentation rather than instance segmentation, so the number of classes at the pixel level is limited to two. Herein, we will look at five loss functions that are widely used in semantic segmentation.

2.3.1 Binary cross-entropy loss

Cross-entropy [48] is a measurement of the difference between two probability distributions for a set of events or a given random variable is called cross-entropy. It is frequently employed in classification tasks. Segmentation also performs well because it is classified at the pixel level.

In this problem, each pixel belongs to one of two classes that are either white or black (1 or 0). So here BCE is used. It can be described as follows:

$$L_{\mathrm{BCE}}(y, \hat{y}) = -(y \log \hat{y} + (1 - y) \log(1 - \hat{y})), \tag{12}$$

where $\hat{y}$ is the predicted value and y is the corresponding target value.

2.3.2 Dice loss

Among segmentation evaluation metrics, the Dice coefficient is frequently employed in the field of computer vision. Dice loss [49] directly optimizes the Dice coefficient with some adaptations, it can be defined as:

$$L_{\mathrm{Dice}}(y, \hat{p}) = 1 - \frac{2y\hat{p} + \varepsilon}{y + \hat{p} + \varepsilon}, \tag{13}$$

where ε is added to the numerator and denominator to make sure that the function is determined even when both y and $\hat{p}$ are zero.

2.3.3 Tversky loss

The Tversky index can alternatively be viewed as a generalization of the Dice coefficient. Tversky loss [50] reshapes Dice loss and emphasizes false negatives (FN) to achieve a better trade-off between precision and recall. Similarly to Dice loss, Tversky loss can be defined as follows:

$$L_{\mathrm{Tvershy}}(p, \hat{p}) = 1 - \frac{\varepsilon + \hat{p}}{\varepsilon + \hat{p} + \beta(1 - p)\hat{p} + (1 - \beta)p(1 - \hat{p})}, \tag{14}$$

where β is a hyper-parameter that controls the balance between FN and false positives (FP).

2.3.4 Focal Tversky loss

Focal Tversky loss [51] uses the concept of focal loss to attempt to learn hard examples with a low probability such as high imbalanced data and small ROI through the use of the γ coefficient. It can be defined as:

$$L_{\text{Focal Tvershy}} = (1 - L_{\text{Tversky}})^{\frac{1}{\gamma}}, \tag{15}$$

where γ varies in the range [1,3].

2.3.5 Lovasz-Softmax loss

Lovasz-Softmax loss [52] is a loss function for multi-class semantic segmentation that combines with the Softmax operation in the Lovasz extension. In the binary situation, the Lovasz hinge and Jaccard loss are employed, it can be defined as:

$$L_{\text{Lovasz-Softmax}}(F) = \overline{\Delta_{J_1}}(m(F)), \tag{16}$$

where

$$\bar{\Delta} : m \in \mathbb{R}^p \mapsto \sum_{i=1}^{p} m_i g_i(m),$$

with

$$g_i(m) = \Delta(\pi_1, \ldots, \pi_i) - \Delta(\pi_1, \ldots, \pi_{i-1}). \tag{17}$$

2.4 Optimization algorithms

To minimize the loss function in DL networks when mapping input data to output data, an optimization algorithm is used to update the parameters (weights) for each new iteration. The optimization algorithms have a great influence on the accuracy as well as the training speed of DL models. This increases the need to choose a suitable optimization algorithm for each specific task. In this study, we will go over two optimizers that are widely used in DL models, namely SGD and adaptive moment estimation (Adam).

2.4.1 Stochastic gradient descent optimizer

An iterative technique for optimizing a loss function is SGD [53]. Essentially, because SGD uses an estimate of the gradient instead of the real gradient, it is a random approximation of gradient descent optimization. First, to train a DL network using SGD, a loss function is used to compute the gradient estimation. Then, the Γ parameters are adjusted at the k iteration. Each minibatch of n examples from the training set $\{x^{(1)}, \ldots, x^{(n)}\}$with corresponding targets $y^{(j)}$ can be computed as:

$$\hat{g} \leftarrow \frac{1}{n} \nabla_\Gamma \sum_i (f(x^{(j)}; \Gamma), y^{(j)}), \tag{18}$$

$$\Gamma \leftarrow \Gamma - \eta_k \hat{g}, \tag{19}$$

where η_k is learning rate at the kth iteration.

The learning rate is a very important hyperparameter, it determines the magnitude of updating parameters. In practice, it is required to gradually lower the learning rate throughout the training process.

2.4.2 Adaptive moment estimation optimizer

Adam [54] is an extension of SGD, and is one of the most frequently utilized optimization algorithms in DL. It combines the advantages of the RMSprop and AdaGrad algorithms to handle sparse gradients in noise problems. From the evaluation of both the first-order moments and the second-order moments of the gradients, the Adam algorithm computes the adaptive learning rate of each individual for different parameters. The update of weights can be represented as follows:

$$n_t = \gamma_1 n_{t-1} + (1 - \gamma_1) g_t, \tag{20}$$

$$c_t = \gamma_2 c_{t-1} + (1 - \gamma_2) g_t^2, \tag{21}$$

where g_t is gradient vector at t timestep; γ_1 and γ_2 are the exponential decay rates; n_t and c_t are estimates of the first biased moment (mean) and the second biased moment (uncentered variance) of the gradients at t timestep, respectively.

$$\hat{n}_t = \frac{n_t}{1 - \gamma_1^t}, \tag{22}$$

$$\hat{c}_t = \frac{v_t}{1 - \gamma_2^t}, \tag{23}$$

where $\hat{n}_t$ and $\hat{c}_t$ are estimates of the first bias-corrected moment (mean) and the second bias-corrected moment (uncentered variance) of the gradients at t timestep, respectively.

$$\Gamma_{t+1} = \Gamma_t - \frac{\eta}{\sqrt{\hat{c}_t} + \varepsilon} \hat{n}_t, \tag{24}$$

where η is learning rate, Γ_t is model parameters at t timestep.

2.5 Evaluation metrics

2.5.1 Dice and intersection over union metrics

To be able to detect a crack, we need to distinguish it

from the background. Since this is a binary classification problem, all possible cases can be classified into four categories as shown in Table 1. Here, true positive (TP) indicates the number of pixels at which a true crack is actually identified as a crack, FP is the number of pixels at which a true non-crack is identified as a crack. In a similar way, FN indicates the number of pixels where a true crack is identified as a non-crack, and true negative (TN) is the number of pixels where true non-crack is correctly identified as a non-crack.

Table 1 All results of the actual case and the predicted case

Total	Predicted crack	Predicted non-crack
Actual crack	TP	FN
Actual non-crack	FP	TN

Intuitively, a successful prediction is one that maximizes the overlap between the predicted object and its corresponding actual object. In the problem of semantic segmentation, two related but different metrics, Dice and IoU (or Jaccard) [55], are commonly used for the evaluation of segmentation tasks. They can be defined as:

$$Dice(A,B)=\frac{2||A\cap B||}{||A||+||B||}, \tag{25}$$

$$IoU(A,B)=\frac{||A\cap B||}{||A\cup B||}, \tag{26}$$

where A, B are the target and prediction masks for a given class, and $//\ //$ is the norm.

Both Dice and IoU metrics are bounded between zero (when two masks have no overlap) and one (when two masks completely overlap). For model comparison, only one of them is enough. The IoU metric has a simple and intuitive expression, so it is more widely used.

Dice and IoU metrics can be rephrased in terms of TP, FP, and FN:

$$Dice=\frac{2TP}{2TP+FP+FN}, \tag{27}$$

$$IoU=\frac{TP}{TP+FP+FN}. \tag{28}$$

2.5.2 Mean absolute error (*MAE*), mean squared error (*MSE*) and root mean squared error (*RMSE*) metrics

Continuous data are dealt with in regression models and its predictions are in a continuous range. *MAE*, *MSE*, and *RMSE* [56] are three commonly used metrics to evaluate regression models. Because we want to minimize them, all of these are also loss functions.

The *MAE* is the mean of the absolute difference between the actual values and the predicted values. It can be given by:

$$MAE=\frac{1}{m}\sum_{i=1}^{m}|y_i-\hat{y}_i|, \tag{29}$$

where m is the total number of data, y_i is actual value, $\hat{y}_i$ is predicted value.

Because it is the mean error, the *MAE* is the easiest to understand. However, since it is an absolute value function, the direction of the error is not indicated.

The mean of the squared difference between the actual values and the predicted values is known as the *MSE*. It can be given by:

$$MSE=\frac{1}{m}\sum_{i=1}^{m}(y_i-\hat{y}_i)^2, \tag{30}$$

where m is the total number of data, y_i is actual value, $\hat{y}_i$ is predicted value.

The *MSE* is quite similar to the *MAE*. However, the *MSE* penalizes larger errors much more than smaller errors, so the *MSE* is more commonly used than the *MAE*.

The square root of the mean of the squared differences between the actual values and the predicted values is known as the *RMSE*. It can be given by:

$$RMSE=\sqrt{\frac{1}{m}\sum_{i=1}^{m}(y_i-\hat{y}_i)^2}, \tag{31}$$

where m denotes the total number of data, y_i is actual value, $\hat{y}_i$ is predicted value.

RMSE can interpret in the units of y, so it is even more prevalent than the *MSE*. Also, the *RMSE* is a good tool for estimating the standard deviation of a distribution of errors.

3 Results and discussion

3.1 Data set

For crack segmentation, we experiment with 100 concrete crack images with their corresponding mask, which is taken from the open-source data set in Ref. [57]. This data set consists of 537 color images of cracks (asphalt and concrete are the two major scenes) with their corresponding masks. All images have been fixed at a size of 544 × 384 pixels, where each image presented one or more cracks. Some representative images of concrete cracks and their corresponding masks are shown in Fig. 9.

A schematic of crack segmentation using the SegNet/U-Net consists of the following steps: preparing a crack data set, training the model, and testing the trained model, as shown in Fig. 10. For training purposes, all

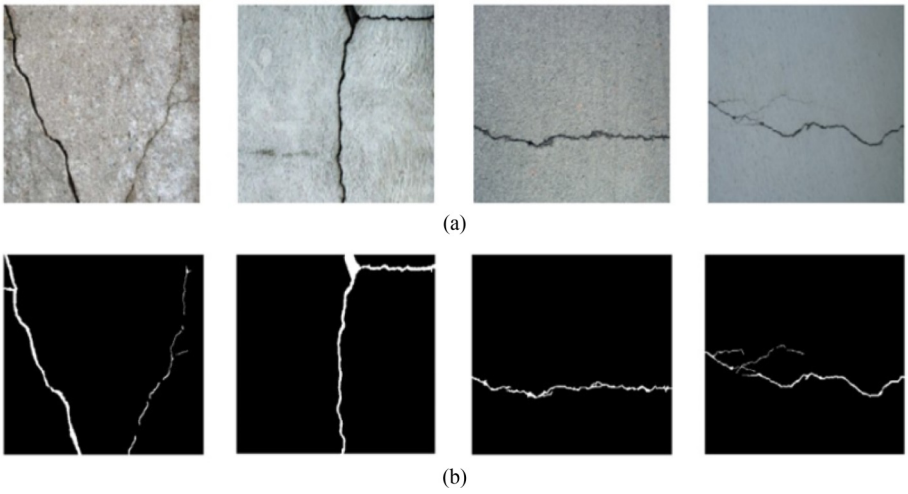


Fig. 9 Some representatives of the crack image database: (a) source images; (b) ground truth.

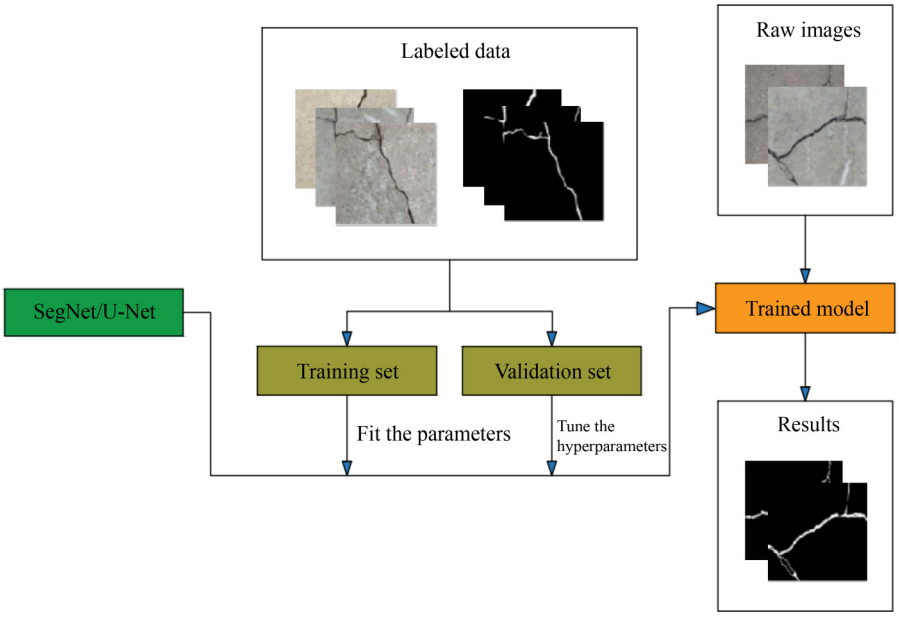


Fig. 10 The diagram of a SegNet/U-Net trained for concrete crack segmentation.

images are re-sampled to 256 × 256 pixels with a 70-30 train-validation split (70 images for the training set with 30 ones for the validation set). The size of each image is reduced before feeding into the network, which largely reduces the training parameters as well as the complexity in the network, leading to a significant reduction in training time without much impact on the results.

For evaluation of crack propagation, several examples are designed for crack propagation instances to illustrate and validate the capabilities of GRU and LSTM models. All data sets are gained from numerical results and experimental data (such as a fuselage panel [32], L-shaped concrete specimen [58], ADB610 steel sample [59]) that have been published in previous studies.

Figure 11 provides a schematic of the process of prediction of crack propagation using the GRU/LSTM, and shows the steps of preparing a crack data set, training the model, and evaluating the learned model. The data set in each example is split into three sets, namely training set, validation set, and testing set. The parameters of the model are fitted using the training set. The validation set is used to adjust the model's hyperparameters, which helps the model perform better during training. The testing set, which is distinct from both the training set and the validation set, is solely used to evaluate the model's final performance once it has been trained. When it comes to time-series data, the data are not randomly split into independent samples. For each point in time, historical data are used as features, while future data cannot be used. Hence, previous data are utilized to train the model and the later data on the testing set are utilized to evaluate the model.

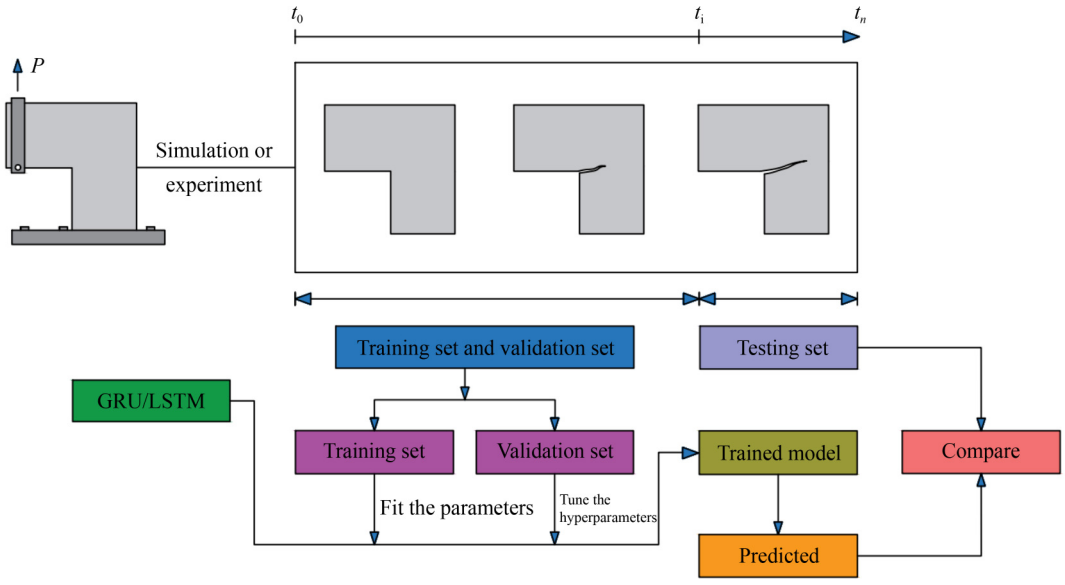


Fig. 11 The diagram of a GRU/LSTM trained for prediction of crack propagation.

3.2 Results on crack segmentation

In this subsection, we present the crack segmentation results from trained SegNet and U-Net networks and then discuss the outcome obtained. In each network, we employ the SGD and Adam optimizers with various loss functions in turn to train the models. The SGD optimizer is employed with a fixed learning rate of 0.01 and momentum of 0.9. The Adam optimizer is used with an initial learning rate of 0.0001; the default exponential decay rates for the first and second moment estimates are $\beta_1 = 0.9$, $\beta_2 = 0.999$, respectively, and $\epsilon = 10^{-8}$ is a small default number to maintain numerical stability during the training for 200 epochs. After each convolutional layer, batch normalization [60] is employed, which is a technique for speeding up the learning process of the network by normalizing the values of the weights in the preceding layer. In addition, all networks employ the ReLU activation function [61], which is critical in allowing the neural network model to learn the complicated nonlinear relationship.

3.2.1 Using SegNet

Figure 12 illustrates the change in loss values (on the training set) and IoU values (on the validation set) with respect to the epochs during the training process of the SegNet network that uses the SGD optimizer. It can be observed that for all five models, the training losses rapidly decrease at the beginning and slowly at the end; meanwhile, the IoU values gradually increase and reach around 76% after 200 epochs. In addition, the IoU curves demonstrate that the use of the dice loss leads to the best performance; in contrast, the BCE loss shows the lowest performance.

The loss curves on the training set, as well as the IoU curves on the validation set in relation to the epochs of the networks using the Adam optimizer, are depicted in Fig. 13. It can be seen that the training losses first decrease rapidly and begin to converge after about 75 epochs; meanwhile, the IoU values increase quickly and converge to about 77% after the 75th epoch for all five models. Also, the IoU curves reveal that the performances of all models are similar.

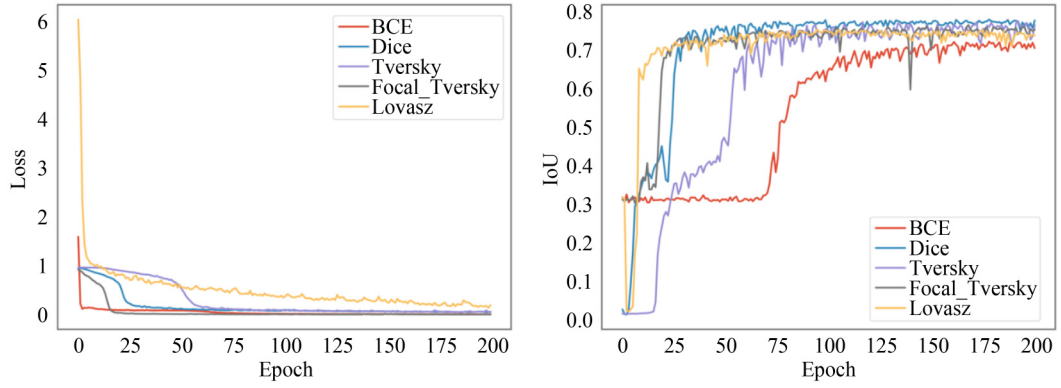


Fig. 12 Changes in the loss value and IoU value against the epoch during training of SegNet with SGD optimizer.

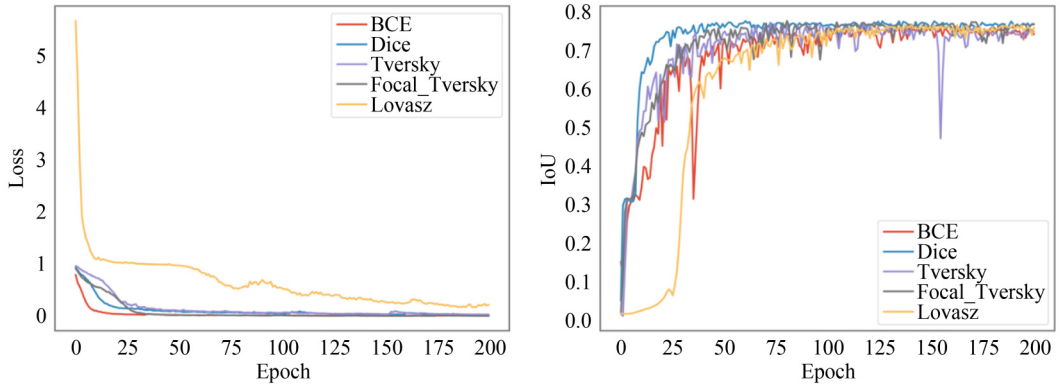


Fig. 13 Changes in the loss value and IoU value against the epoch during training of SegNet with Adam optimizer.

The performance of concrete crack segmentation models that are based on IoU value is summarized in Table 2. The result shows that the model using the SGD optimizer and Dice loss shows the highest IoU (78.11%); while the model using the SGD optimizer and the BCE loss shows a significantly lower IoU than those of other models, at only 72.40%.

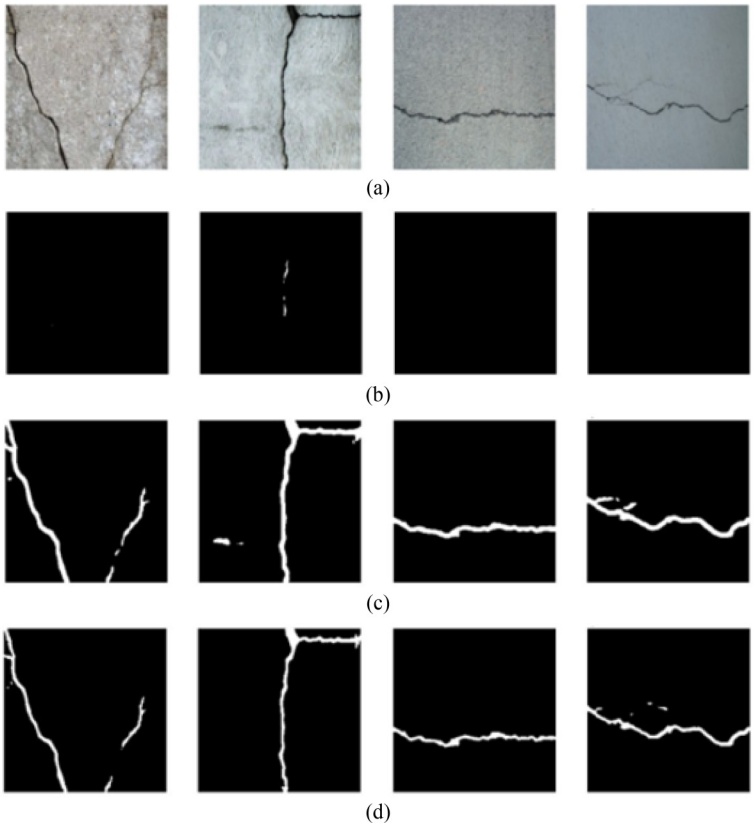
Table 2 Segmentation results (IoU) of the SegNet for different optimizers and loss functions. The metric is presented for the validation set

Optimizer	BCE	Dice	Tversky	Focal Tversky	Lovasz-Softmax
SGD	0.7240	0.7811	0.7768	0.7668	0.7557
Adam	0.7652	0.7789	0.7748	0.7794	0.7692

3.2.2 Using U-Net

The qualitative results of the U-Net model, which employs the Adam optimizer and the focal Tversky loss function, are depicted in Fig. 14. Herein, the source image, the ground truth, and the output of several samples on the training set at various epochs during training are shown. It is clear from Fig. 14 that the model is learning properly; the output at the 200th epoch is close to the ground truth, which the human eye can hardly distinguish.

In Fig. 15, the loss curves on the training set and the IoU curves on the validation set against the epochs of the



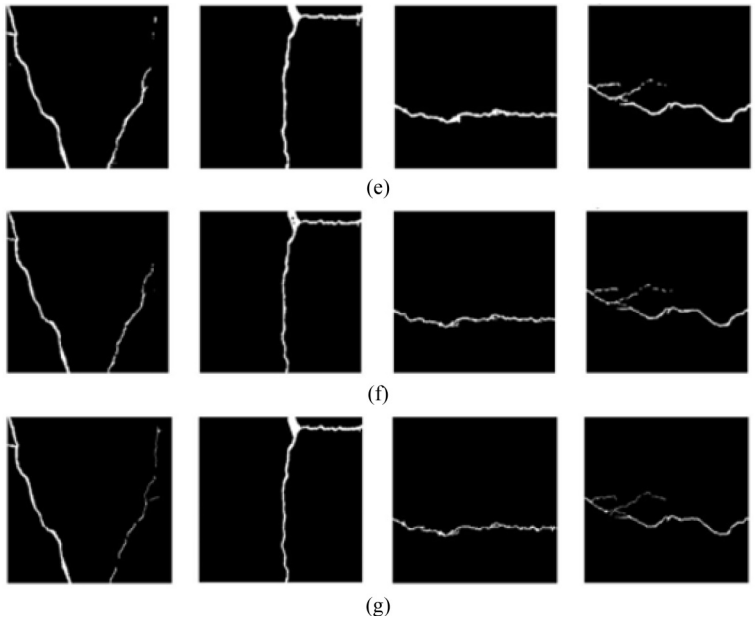


Fig. 14 Output on several samples at different epochs of U-Net model using Adam optimizer and the focal Tversky loss function: (a) source images; (b) 10th epoch; (c) 20th epoch; (d) 50th epoch; (e) 100th epoch; (f) 200th epoch; (g) ground truth.

networks using the SGD optimizer are shown. It is clear from Fig. 15 that the training losses reduce rapidly at first and begin to converge after around 75 epochs; meanwhile, the IoU values climb rapidly and converge to roughly 79% for all five models after the 75th epoch. Furthermore, the IoU curves show that the performance of all models is similar.

Figure 16 illustrates the change of the loss values (on

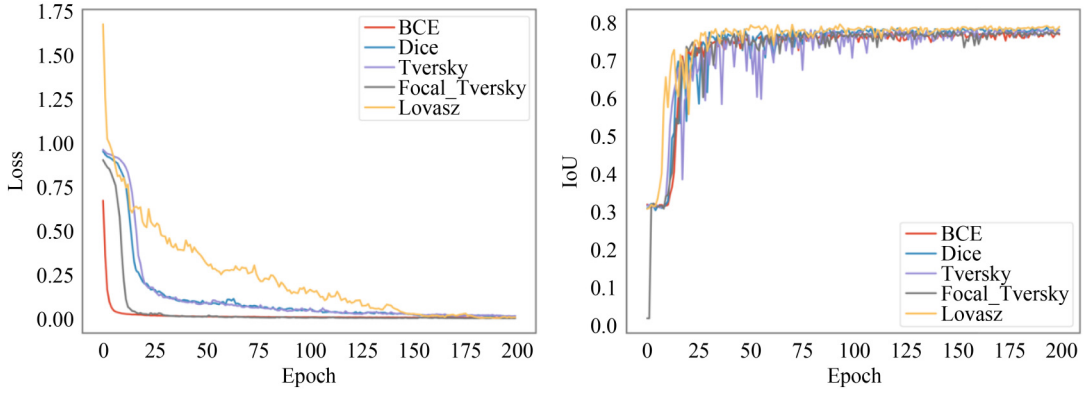


Fig. 15 Changes in the loss value and IoU value against the epoch during training of U-Net with SGD optimizer.

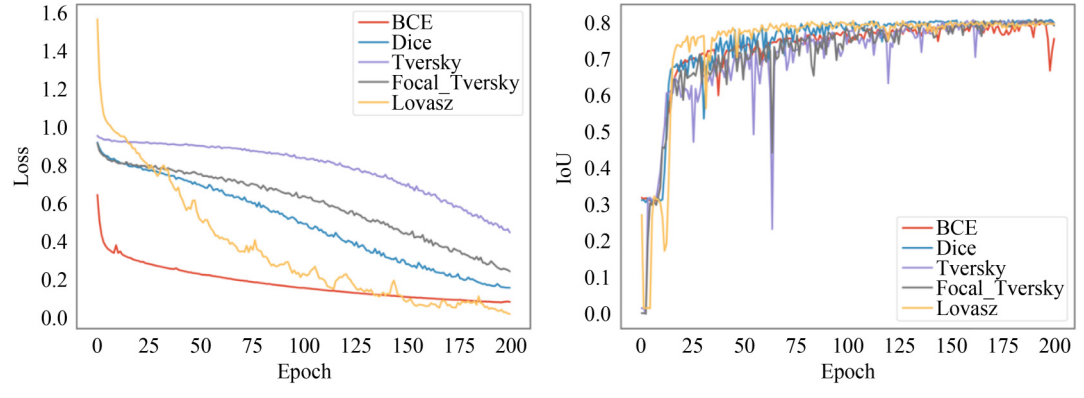


Fig. 16 Changes in the loss value and IoU value against the epoch during training of U-Net with Adam optimizer.

the training set) and the IoU values (on the validation set) with respect to the epochs during the training process of the U-Net network using the Adam optimizer. It can be seen from Fig. 16 that for all five models, the training loss gradually decreases, but there is still no clear sign of convergence after 200 epochs; meanwhile, the IoU values gradually increase and start to converge after about 75 epochs, eventually reaching an IoU value of about 81% at the end of the training process. Moreover, the IoU curves revealed that the performances of all models are relatively similar.

The obtained metrics from the trained models on the validation set are reported in Table 3. It can be observed from Table 3 that all the considered models achieve high IoU, that is 78% or more; the model using the Adam optimizer and focal Tversky loss outperforms the others, with an IoU value of 81.17%.

According to the aforementioned results, the U-Net model with Adam optimizer and focal Tversky loss performs the best in concrete crack segmentation. Some of the crack samples on the testing set with detections obtained by the U-Net model, which employs the Adam optimizer and the focal Tversky loss, are depicted in Fig. 17. The raw images, the predicted images, and the ground truth are displayed in order from top to bottom in Fig. 17. The segmented cracks are close to the ground truth, as may be observed. This demonstrates the efficacy of the crack segmentation approach used in this study.

Table 3 Segmentation results (IoU) of the U-Net for different optimizers and loss functions. The metric is presented for the validation set

Optimizer	BCE	Dice	Tversky	Focal Tversky	Lovasz-Softmax
SGD	0.7805	0.7919	0.7874	0.7825	0.7997
Adam	0.8044	0.8101	0.8093	0.8117	0.8077

3.3 Results for crack propagation

In this subsection, we present the results for crack propagation and discuss the outcome. Herein, three crack examples are looked at to verify the performance of the LSTM and GRU networks. All data sets used are obtained from the Paris model [62] or experiments in previous studies. The proportion of the data set used in the training process accounts for 84% (for the first two examples) and 62% (for the last example) and the rest of the data set is used for the testing set. In the training process of each experiment, the training data are further divided into two subsets: the training set has 80%, and the validation set accounts for 20%. For training purposes, the authors choose a sequence length of 30 timesteps. Input-output pairs are created where each input consists of the previous 30 timesteps, and the corresponding output is the next timestep.

To demonstrate the robustness and reliability of the approach, the architecture of the GRU and LSTM networks with four GRU/LSTM layers is utilized in all

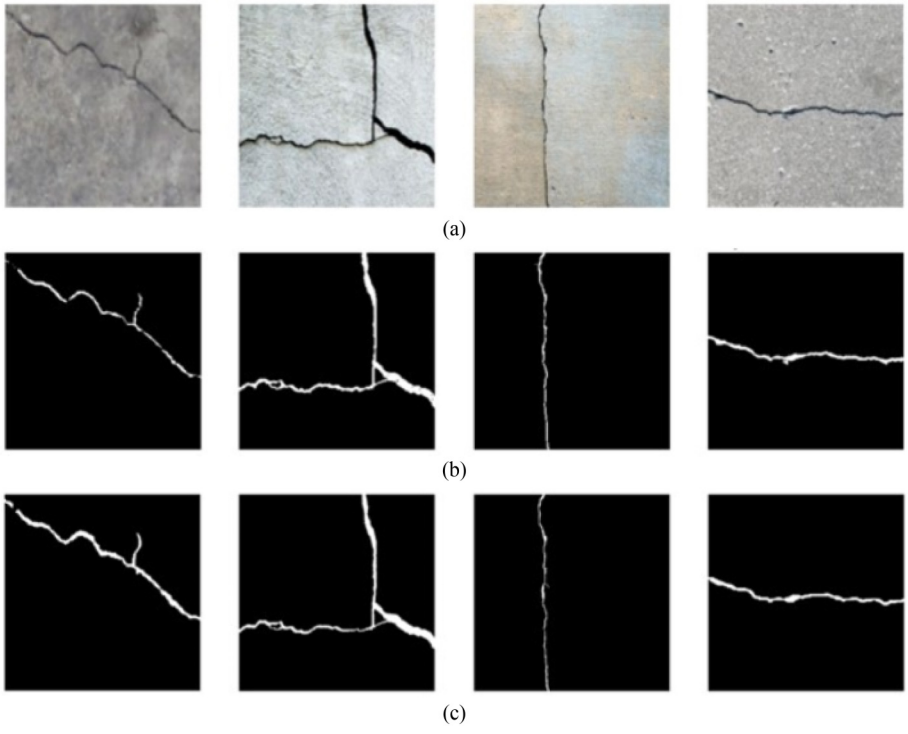


Fig. 17 Output on several samples on the testing set of U-Net model using Adam optimizer and the focal Tversky loss function: (a) raw images; (b) predicted images; (c) ground truth.

cases, as detailed in Fig. 18. The Adam optimizer is employed with default parameters and the *MSE* loss function to train the models. In addition to the tanh activation function and the sigmoid activation function (the sigmoid is used for the recurrent step), the "dropout" technique [63] with a dropout rate of 0.2 in each GRU/LSTM layer is also utilized to avoid the overfitting problem. Training was performed with 100 epochs for all models.

3.3.1 Example 1: A fuselage panel's fatigue analysis

An aluminum alloy fuselage panel [32], is as illustrated in Fig. 19. The panel has a 0.02 m long center crack on the wall, a 0.06 MPa applied pressure, a thickness of 0.00248 m, and a radius of 3.25 m. Young's modulus E = 70 GPa and Poisson's ratio ν = 0.33 are the material characteristics of aluminum alloy. Under constant amplitude mode I, this issue is considered to be treated as an endless plate with a through-the-thickness center crack. The constant $C = 1.5 \times 10^{-10}$ m/cycle and the exponent m = 3.8 of the Paris Law [62] are provided.

Figure 20 depicts the convergence history of the loss functions on the training and validation sets that are obtained during the training of the GRU and LSTM networks. It is clear from Fig. 20 that in only a few epochs, the loss values rapidly fall and converge to zero. This demonstrates that the networks are learning properly.

Figure 21 describes the crack growth trend predicted by the GRU and LSTM networks on the testing set, which is compared with the actual data. It can be seen that the prediction results obtained from the networks agree with the experimental data, and that the GRU network outperforms the LSTM network. The evaluation metrics obtained from the trained networks on the testing set are summarized in Table 4. The outcomes demonstrate that the coefficient of determination (R^2) value obtained from the GRU network is larger than that of the LSTM network with values 0.9858 and 0.9435, respectively.

3.3.2 Example 2: Fatigue life prediction

In this example, experimental data are used to forecast the growth of fatigue crack in the ADB610 steel sample [59] which confirms the applicability of the GRU/LSTM network. The specifications of this material with a stress ratio of 0.3 are shown in Table 5.

The convergence history of the loss functions on the training set and validation set produced during the training of GRU and LSTM networks is shown in Fig. 22. The loss values rapidly reduce and converge to zero after

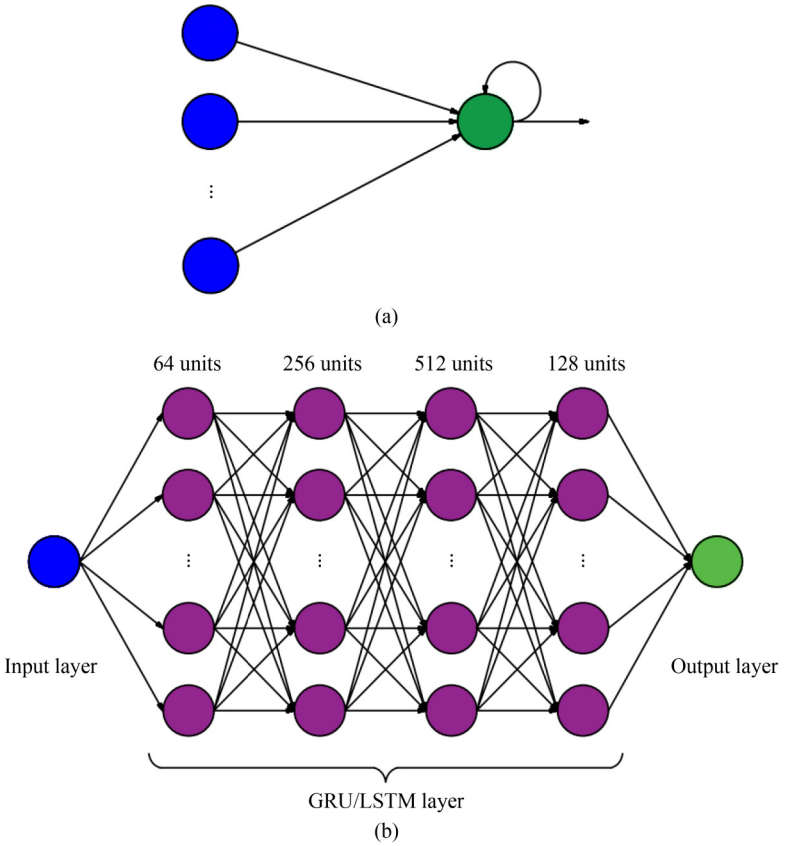


Fig. 18 The proposed deep GRU/LSTM network: (a) a single recurrent neural and (b) network with four GRU/LSTM layers.

only a few epochs, as seen in Fig. 22. This shows that the networks are appropriately learning.

The crack growth trend predicted by the GRU and LSTM networks on the testing set is depicted in Fig. 23, which is compared to the actual data. It can be shown that the network prediction results are consistent with the experimental data, with the GRU network outperforming the LSTM network. Table 6 summarizes the evaluation metrics acquired from the trained networks on the testing set. It can be seen that the R^2 value obtained from the GRU network is larger than the R^2 value obtained from the LSTM network, with values of 0.9588 and 0.9221, respectively.

3.3.3 Example 3: L-shape concrete specimen damage

As shown in Examples 1 and 2, both GRU and LSTM give good results (with $R^2 > 0.92$) for crack length prediction. However, the relationship between length and cycle is a monotonic function. To increase the complexity of the task, the authors further investigate Example 3 where the load−displacement curve is predicted instead of crack length to further explore the potential of both GRU and LSTM. This example uses experimental data from Ref. [58], which depicts the failure behavior of a basic L-shaped concrete sample. Figure 24 shows the geometry of a specimen with a thickness of 100 mm. The Young's modulus and the Poisson's ratio of this concrete material

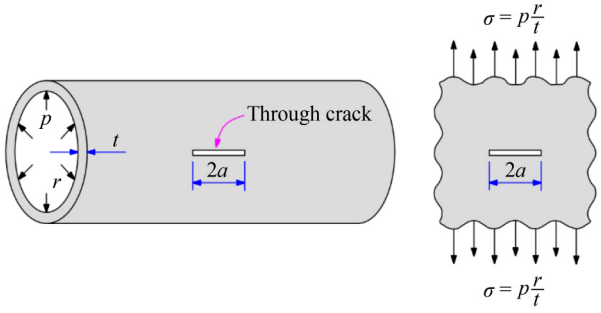


Fig. 19 The through-crack on a fuselage panel.

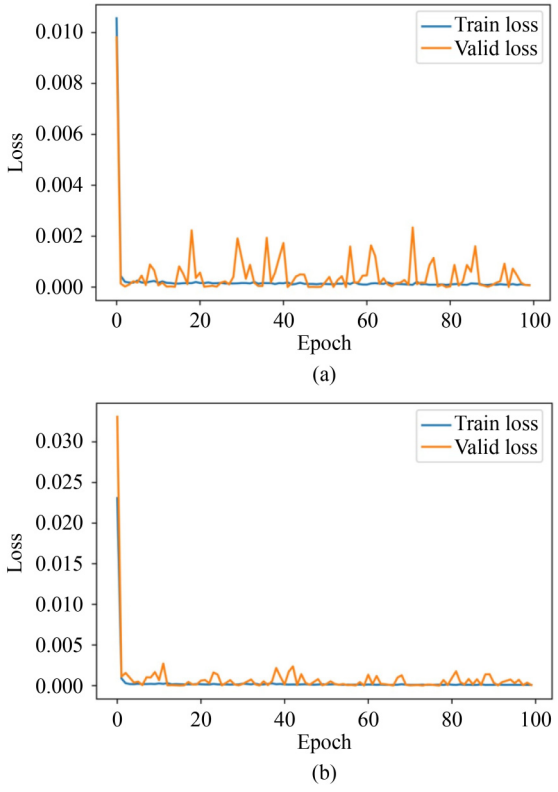


Fig. 20 The loss curve of (a) GRU and (b) LSTM on simulation data of the fuselage panel.

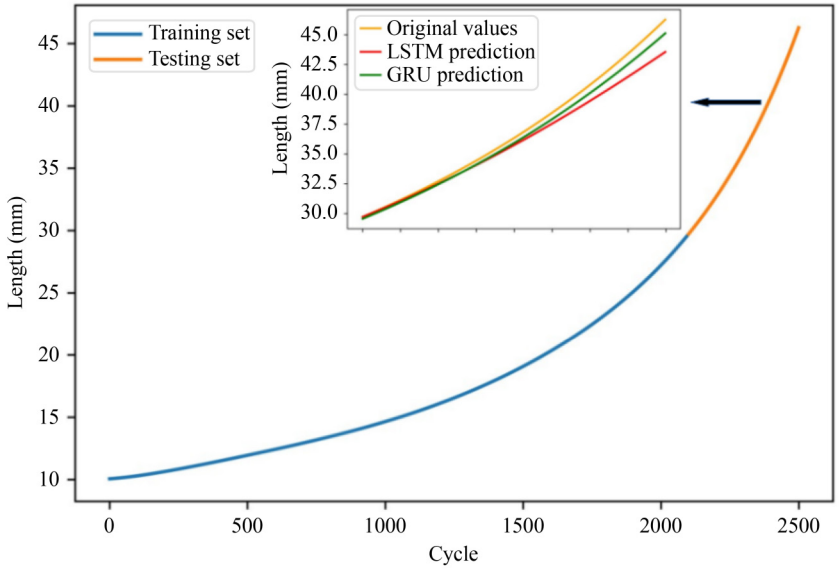


Fig. 21 The prediction results of crack growth on fuselage panel by GRU and LSTM.

are $E = 25.85$ GPa and $\nu = 0.18$, respectively.

The convergence history of the loss functions on the training and validation sets, during the training of the GRU and LSTM networks, is shown in Fig. 25. It can be observed that the training losses initially decrease quickly and begin to converge after only about the first 10 epochs. This shows that the neural networks are correctly learning.

On the testing set, the crack growth trends predicted by the GRU and LSTM networks are compared to the actual data in Fig. 26. It is clear that there is good agreement between the values predicted by the networks and the actual values, where the GRU network beats the LSTM network in terms of prediction outcomes. The errors and R^2 obtained from the trained networks on the testing set are presented in Table 7. It is clear from Table 7 that the R^2 values obtained from the networks are not much different; the R^2 value obtained by the GRU network is larger than that of the LSTM network, with values of 0.9823 and 0.9247, respectively.

Table 4 The errors of crack growth prediction on the fuselage panel using GRU and LSTM

Model	*MAE*	*MSE*	*RMSE*	R^2
GRU	0.4947	0.3197	0.5654	0.9858
LSTM	0.8414	1.2739	1.1287	0.9435

Note: The metrics are presented for the testing set.

Table 5 Information on ADB610 steel specimen

Information	Specimen information	Stress ratio	$\sigma_{min}/\sigma_{max}$
specimen type	compact tensile specimen	0.3	6/20
specimen size	60 mm × 50 mm × 12.5 mm (length × width × thickness)	–	–
initial crack length	15 mm	–	–
loading type	tension-tension, constant amplitude	–	–

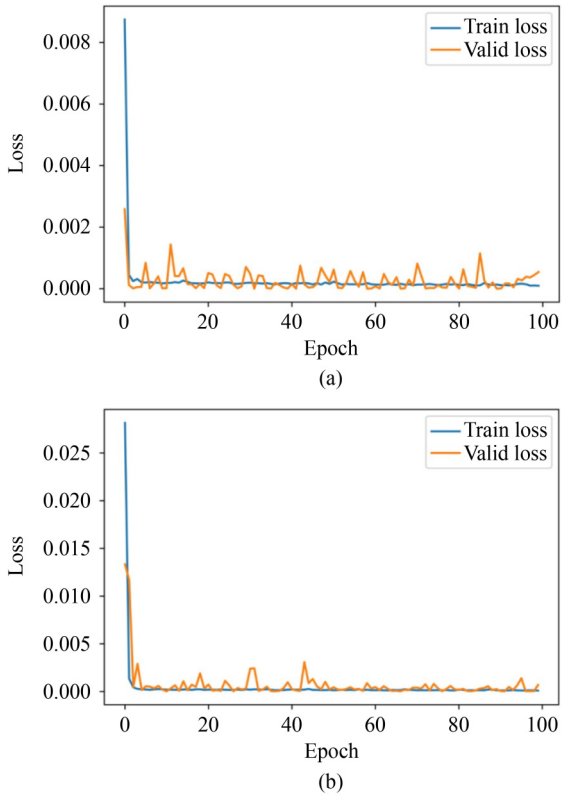


Fig. 22 The loss curve of (a) GRU and (b) LSTM on experimental data of ADB610 steel specimen with the stress ratio of 0.3.

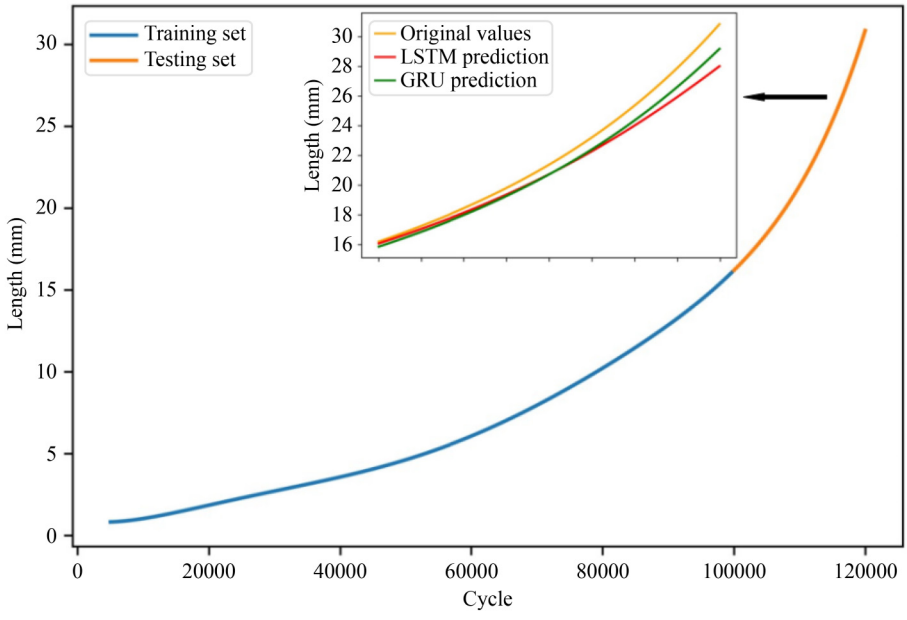


Fig. 23 The prediction results of crack growth on ADB610 steel specimen with the stress ratio of 0.3 by GRU and LSTM.

4 Conclusions

Pixel-level concrete crack detection and crack propagation prediction have great significance in structural engineering. In this study, DL networks are used to implement two main tasks, namely.

1) For pixel-level concrete crack detection: Unlike other object detection tasks, segmentation of crack regions implemented at the pixel level is much better than predicting bounding boxes. Our work contributes to the proposal of a DL model based on SegNet and U-Net networks to build a semantic segmentation model of concrete cracks automatically. The performance of SegNet and U-Net models, which use different optimizers and loss functions, is cross-compared. The findings of the experiment indicate that the U-Net model using the Adam

Table 6 The errors of crack growth prediction on ADB610 steel specimen using GRU and LSTM. The metrics are presented for the testing set

Model	*MAE*	*MSE*	*RMSE*	R^2
GRU	0.7636	0.7119	0.8437	0.9588
LSTM	0.9012	1.3454	1.1599	0.9221

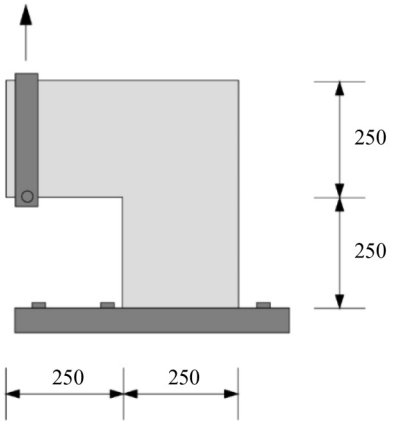


Fig. 24 The L-shaped concrete specimen's dimensions (unit: mm).

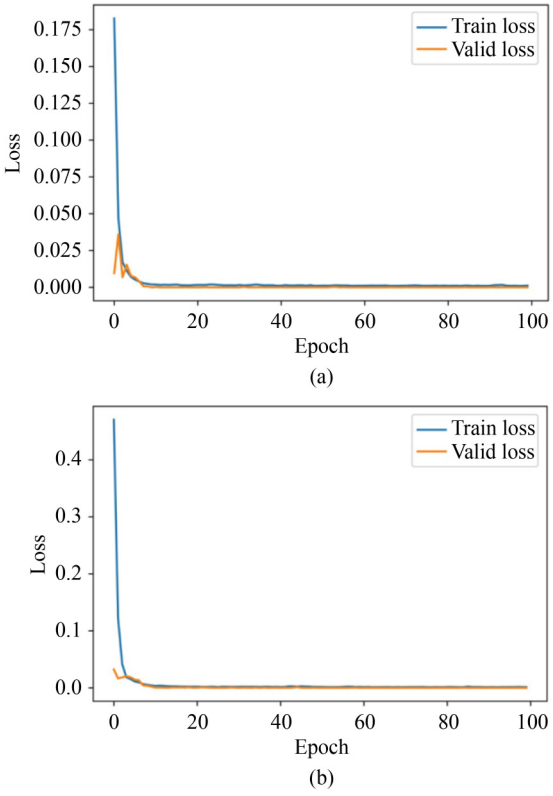


Fig. 25 The loss curves of (a) GRU and (b) LSTM on experimental data using L-shaped concrete specimens.

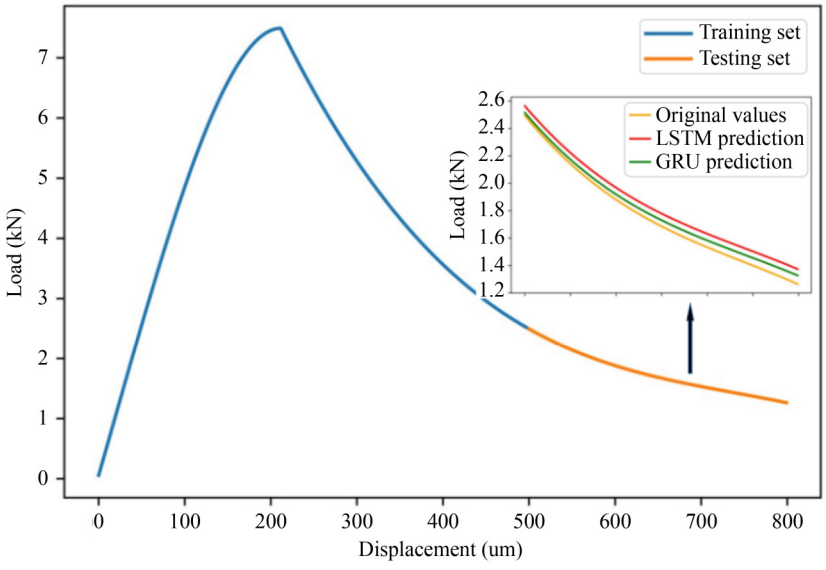


Fig. 26 The prediction results for crack growth in L-shaped concrete specimens, by GRU and LSTM.

Table 7 The errors of crack growth prediction for L-shaped concrete specimens using GRU and LSTM. The metrics are presented for the testing set

Model	*MAE*	*MSE*	*RMSE*	R^2
GRU	0.0435	0.0020	0.0449	0.9823
LSTM	0.0920	0.0086	0.0926	0.9247

optimizer and the focal Tversky loss with an IoU of 81.17% outperforms other models.

2) For the task of evaluating crack propagation: A DL approach uses time-series forecasting method to solve the problem of crack propagation; here GRU and LSTM networks are used. The crack propagation of some materials is studied through experimental examples. To demonstrate the superiority of DL algorithms, GRU and LSTM networks with the same network architecture are used in all experimental examples. The prediction results of both GRU and LSTM are consistent with the experimental data, where the GRU outperforms the LSTM in all experimental cases.

The network is verified based on the input data set. It can therefore be applied to other materials and databases by changing the input to the network. In future studies, which are beyond this study, building a larger data set to increase the robustness of the proposed method, and further comparative studies, will be carried out. To simultaneously identify and predict crack propagation, the problem of combining both networks (U-Net and GRU) into an end-to-end network will be studied. In addition, to optimize the hyper-parameters of the models, the Bayesian optimization method will be used. Finally, CycleGAN [64] is also of interest to our future work.

Acknowledgements The first author would like to thank European Commission H2020-MSCA-RISE BESTOFRAC project for research funding.

Funding note Open Access funding enabled and organized by Projekt DEAL.

Competing interests The authors declare that they have no competing interests.

References

1. Lemaitre J. A Course on Damage Mechanics. Berlin: Springer Science & Business Media, 2012
2. Graybeal B A, Phares B M, Rolander D D, Moore M, Washer G. Visual inspection of highway bridges. Journal of Nondestructive Evaluation, 2002, 21(3): 67−83
3. Chatzi E N, Hiriyur B, Waisman H, Smyth A W. Experimental application and enhancement of the XFEM–GA algorithm for the detection of flaws in structures. Computers & Structures, 2011, 89(7−8): 556−570
4. Vu-Bac N, Nguyen-Xuan H, Chen L, Bordas S, Kerfriden P, Simpson R, Liu G, Rabczuk T. A node-based smoothed extended finite element method (NS-XFEM) for fracture analysis. Computer Modeling in Engineering & Sciences, 2011, 73: 331−356
5. Zou Q, Cao Y, Li Q, Mao Q, Wang S. Cracktree: Automatic crack detection from pavement images. Pattern Recognition Letters, 2012, 33(3): 227−238
6. Butcher J B, Day C, Austin J, Haycock P, Verstraeten D, Schrauwen B. Defect detection in reinforced concrete using random neural architectures. Computer-Aided Civil and Infrastructure Engineering, 2014, 29(3): 191−207
7. Yagi K, Tanaka S, Kawahara T, Nihei K, Okada H, Osawa N. Evaluation of crack propagation behaviors in a T-shaped tubular joint employing tetrahedral FE modeling. International Journal of Fatigue, 2017, 96: 270−282
8. Nguyen-Thanh V M, Zhuang X, Nguyen-Xuan H, Rabczuk T, Wriggers P. A virtual element method for 2D linear elastic fracture analysis. Computer Methods in Applied Mechanics and Engineering, 2018, 340: 366−395
9. Wang Q, Ji B, Fu Z, Ye Z. Evaluation of crack propagation and fatigue strength of rib-to-deck welds based on effective notch stress method. Construction & Building Materials, 2019, 201: 51−61
10. Rabczuk T, Belytschko T. Cracking particles: A simplified meshfree method for arbitrary evolving cracks. International Journal for Numerical Methods in Engineering, 2004, 61(13): 2316−2343
11. Rabczuk T, Belytschko T. A three-dimensional large deformation meshfree method for arbitrary evolving cracks. Computer Methods in Applied Mechanics and Engineering, 2007, 196(29−30): 2777−2799
12. Oliveira H, Correia P L. Automatic road crack detection and characterization. IEEE Transactions on Intelligent Transportation Systems, 2013, 14(1): 155−168
13. Vu-Bac N, Nguyen-Xuan H, Chen L, Lee C K, Zi G, Zhuang X, Liu G R, Rabczuk T. A phantom-node method with edge-based strain smoothing for linear elastic fracture mechanics. Journal of Applied Mathematics, 2013, 2013: 978026
14. Adhikari R, Moselhi O, Bagchi A. Image-based retrieval of concrete crack properties for bridge inspection. Automation in Construction, 2014, 39: 180−194
15. Shi Y, Cui L, Qi Z, Meng F, Chen Z. Automatic road crack detection using random structured forests. IEEE Transactions on Intelligent Transportation Systems, 2016, 17(12): 3434−3445

16. Minh H L, Sang-To T, Abdel Wahab M, Cuong-Le T. A new metaheuristic optimization based on K-means clustering algorithm and its application for structural damage identification. Knowledge-Based Systems, 2022, 251: 109189
17. Ho T T, Kim T, Kim W J, Lee C H, Chae K J, Bak S H, Kwon S O, Jin G Y, Park E K, Choi S. A 3D-CNN model with CT-based parametric response mapping for classifying COPD subjects. Scientific Reports, 2021, 11(1): 34
18. Park S S, Tran V T, Doan N P, Hwang K B. Evaluation of damage level for ground settlement using the convolutional neural network. In: Proceedings of CIGOS 2021, Emerging Technologies and Applications for Green Infrastructure. Singapore: Springer Singapore, 2022, 1261–1268
19. Ho T T, Kim G T, Kim T, Choi S, Park E K. Classification of rotator cuff tears in ultrasound images using deep learning models. Medical & Biological Engineering & Computing, 2022, 60(5): 1269
20. Ogunjinmi P D, Park S S, Kim B, Lee D E. Rapid post-earthquake structural damage assessment using convolutional neural networks and transfer learning. Sensors, 2022, 22(9): 3471
21. Jeon H M, Nguyen V D, Jeon J W. Pedestrian detection based on deep learning. In: Proceedings of IECON 2019—45th Annual Conference of the IEEE Industrial Electronics Society. New York: IEEE, 2019, 144–151
22. Quang Dinh V, Munir F, Azam S, Yow K C, Jeon M. Transfer learning for vehicle detection using two cameras with different focal lengths. Information Sciences, 2020, 514: 71−87
23. Park S S, Tran V T, Lee D E. Application of various yolo models for computer vision-based real-time pothole detection. Applied Sciences, 2021, 11(23): 11229
24. Wang G, Li W, Zuluaga M A, Pratt R, Patel P A, Aertsen M, Doel T, David A L, Deprest J, Ourselin S, Vercauteren T. Interactive medical image segmentation using deep learning with image-specific fine tuning. IEEE Transactions on Medical Imaging, 2018, 37(7): 1562−1573
25. Minaee S, Boykov Y Y, Porikli F, Plaza A J, Kehtarnavaz N, Terzopoulos D. Image segmentation using deep learning: A survey. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2022, 44(7): 3523−3542
26. le Hien Nguyen D, Thi Thanh Do D, Lee J, Rabczuk T, Nguyen-Xuan H. Forecasting damage mechanics by deep learning. Computers, Materials & Continua, 2019, 61(3): 951−977
27. Schwarzer M, Rogan B, Ruan Y, Song Z, Lee D Y, Percus A G, Chau V T, Moore B A, Rougier E, Viswanathan H S, Srinivasan G. Learning to fail: Predicting fracture evolution in brittle material models using recurrent graph convolutional neural networks. Computational Materials Science, 2019, 162: 322−332
28. Wang J, Zheng Y, Luo R, Ma J, Peng Y, Aslam S, Jia W. Prediction method of three-dimensional crack propagation path based on deep learning application. Advanced Engineering Materials, 2021, 23(4): 2001043
29. Zou Q, Zhang Z, Li Q, Qi X, Wang Q, Wang S. DeepCrack: Learning hierarchical convolutional features for crack detection. IEEE Transactions on Image Processing, 2019, 28(3): 1498−1512
30. Liu Z, Cao Y, Wang Y, Wang W. Computer vision-based concrete crack detection using U-net fully convolutional networks. Automation in Construction, 2019, 104: 129−139
31. Anitescu C, Atroshchenko E, Alajlan N, Rabczuk T. Artificial neural network methods for the solution of second order boundary value problems. Computers, Materials & Continua, 2019, 59(1): 345
32. Thi Thanh Do D, Lee J, Nguyen-Xuan H. Fast evaluation of crack growth path using time series forecasting. Engineering Fracture Mechanics, 2019, 218: 106567
33. Hsu Y C, Yu C H, Buehler M J. Using deep learning to predict fracture patterns in crystalline solids. Matter, 2020, 3(1): 197−211
34. Goswami S, Anitescu C, Chakraborty S, Rabczuk T. Transfer learning enhanced physics informed neural network for phase-field modeling of fracture. Theoretical and Applied Fracture Mechanics, 2020, 106: 102447
35. Chakraborty A, Anitescu C, Zhuang X, Rabczuk T. Domain adaptation based transfer learning approach for solving PDEs on complex geometries. Engineering with Computers, 2022, 38(5): 4569
36. Yang X, Li H, Yu Y, Luo X, Huang T, Yang X. Automatic pixel-level crack detection and measurement using fully convolutional network. Computer-Aided Civil and Infrastructure Engineering, 2018, 33(12): 1090−1109
37. Fukushima K, Miyake S, Ito T. Neocognitron: A neural network model for a mechanism of visual pattern recognition. IEEE Transactions on Systems, Man, and Cybernetics, 1983, SMC-13(5): 826−834
38. Waibel A, Hanazawa T, Hinton G, Shikano K, Lang K J. Phoneme recognition using time-delay neural networks. IEEE Transactions on Acoustics, Speech, and Signal Processing, 1989, 37(3): 328−339
39. LeCun Y, Bottou L, Bengio Y, Haffner P. Gradient-based learning applied to document recognition. Proceedings of the IEEE, 1998, 86(11): 2278−2324
40. Goodfellow I, Bengio Y, Courville A. Deep Learning. Cambridge, MA: MIT Press, 2016
41. Badrinarayanan V, Kendall A, Cipolla R. SegNet: A deep convolutional encoder−decoder architecture for image segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2017, 39(12): 2481−2495
42. Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition. 2014, arXiv: 1409.1556
43. Ronneberger O, Fischer P, Brox T. U-Net: Convolutional networks for biomedical image segmentation. In: Proceedings of the 18th International Conference on Medical Image Computing and Computer-Assisted Intervention. Cham: Springer Cham, 2015, 234–241
44. DiPietro R, Hager G D. Deep learning: RNNs and LSTM. In: Proceedings of the Handbook of Medical Image Computing and Computer Assisted Intervention. Amsterdam: Elsevier, 2020, 503–519
45. Hochreiter S, Schmidhuber J. Long short-term memory. Neural Computation, 1997, 9(8): 1735−1780
46. Cho K, van Merriënboer B, Gulcehre C, Bahdanau D, Bougares F, Schwenk H, Bengio Y. Learning phrase representations using RNN encoder−decoder for statistical machine translation. 2014, arXiv: 1406.1078

47. Greff K, Srivastava R K, Koutník J, Steunebrink B R, Schmidhuber J. LSTM: A search space odyssey. IEEE Transactions on Neural Networks and Learning Systems, 2017, 28(10): 2222−2232
48. Ma Y D, Liu Q, Qian Z B. Automated image segmentation using improved PCNN model based on cross-entropy. In: Proceedings of the 2004 International Symposium on Intelligent Multimedia, Video and Speech Processing. New York: IEEE, 2004, 743–746
49. Sudre C H, Li W, Vercauteren T, Ourselin S, Jorge Cardoso M. Generalised dice overlap as a deep learning loss function for highly unbalanced segmentations. In: Proceedings of the Deep learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support. Cham: Springer Cham, 2017, 240–248
50. Salehi S S M, Erdogmus D, Gholipour A. Tversky loss function for image segmentation using 3D fully convolutional deep networks. In: Proceedings of the International Workshop on Machine Learning in Medical Imaging. Cham: Springer Cham, 2017, 379–387
51. Abraham N, Khan N M. A novel focal Tversky loss function with improved attention U-Net for lesion segmentation. In: Proceedings of the 2019 IEEE 16th International Symposium on Biomedical Imaging (ISBI 2019). New York: IEEE, 2019, 683–687
52. Berman M, Triki A R, Blaschko M B. The Lovász-Softmax loss: A tractable surrogate for the optimization of the intersection-over-union measure in neural networks. In: Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. New York: IEEE, 2018, 4413–4421
53. Robbins H, Monro S. A stochastic approximation method. Annals of Mathematical Statistics, 1951, 22(3): 400−407
54. Kingma D P, Ba J. Adam: A method for stochastic optimization. 2014, arXiv: 1412.6980
55. Bertels J, Eelbode T, Berman M, Vandermeulen D, Maes F, Bisschops R, Blaschko M B. Optimizing the dice score and Jaccard index for medical image segmentation: Theory and practice. In: Proceedings of the International Conference on Medical Image Computing and Computer-Assisted Intervention. Cham: Springer Cham, 2019, 92–100
56. Botchkarev A. Performance metrics (error measures) in machine learning regression, forecasting and prognostics: Properties and typology. 2018, arXiv: 1809.03006
57. Liu Y, Yao J, Lu X, Xie R, Li L. DeepCrack: A deep hierarchical feature learning architecture for crack segmentation. Neurocomputing, 2019, 338: 139−153
58. Winkler B, Hofstetter G, Niederwanger G. Experimental verification of a constitutive model for concrete cracking. Proceedings of the Institution of Mechanical Engineers, Part L: Journal of Materials: Design and Applications, 2001, 215(2): 75−86
59. Wang H, Zhang W, Sun F, Zhang W. A comparison study of machine learning based algorithms for fatigue crack growth calculation. Materials, 2017, 10(5): 543
60. Ioffe S, Szegedy C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. 2015, arXiv: 1502.03167
61. Agarap A F. Deep learning using rectified linear units (ReLU). 2018, arXiv: 1803.08375
62. Paris P, Erdogan F. A critical analysis of crack propagation laws. Journal of Basic Engineering, 1963, 85(4): 528−533
63. Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R. Dropout: A simple way to prevent neural networks from overfitting. Journal of Machine Learning Research, 2014, 15: 1929−1958
64. Zhu J Y, Park T, Isola P, Efros A A. Unpaired image-to-image translation using cycle-consistent adversarial networks. In: Proceedings of the 2017 IEEE International Conference on Computer Vision (ICCV). New York: IEEE, 2017, 2242–2251