Visual Abstract

Automatic detection of concrete cracks from images using Adam-SqueezeNet deep learning model

Lin Wang
Hefei University, Hefei, Anhui Province, 230061, China
linw@smu.edu, http://orcid.org/0000-0002-6481-2409

ABSTRACT. Cracks in the concrete surfaces are typically clear warning signs of a potential threat to the integrity and serviceability of the structure. The techniques based on image processing can effectively detect cracks in digital images. These techniques, however, are generally susceptible to user-driven heuristic thresholds and irrelevant distractors. Inspired by the recent success of artificial intelligence, a deep learning-based automated crack detection system named CrackSN was presented. This proposed deep learning model, built on the Adam-SqueezeNet architecture, automatically learns the discriminative feature directly from the labeled and augmented patches. An image dataset of concrete surfaces is collected by smartphone and carefully prepared in order to develop and train the CrackSN system. The hyper-parameters of SqueezeNet are tuned with Adam optimization through the training and validation procedures. The fine-tuned CrackSN system outperforms state-of-the-art models in recent literature by correctly classifying 97.3% of the cracked patches in the image dataset. The success of CrackSN, demonstrated by its light network design and outstanding performance, provides a crucial step toward automated damage inspection and health evaluation for infrastructure.

KEYWORDS. Concrete crack, Automated damage inspection, SqueezeNet, Adam optimization, Deep learning.

Citation: Wang, L., Automatic detection of concrete crack from images using Adam-SqueezeNet deep learning model, Frattura ed Integrità Strutturale, 65 (2023) 289-299.

Received: 07.04.2023
Accepted: 18.06.2023
Online first: 19.06.2023
Published: 01.07.2023

INTRODUCTION

Concreted civil infrastructure such as buildings, bridges, and highways sustain various external loads throughout their service period. Earthquake vibration, wind, temperature, or vehicle-made excitation initiate structural damage during their lifetime and, subsequently, trigger catastrophic failure [1, 2]. Regular inspection and maintenance are vital to ensuring the integrity of concrete infrastructure for public safety, economic running, and national security. Cracks on concrete surfaces are essential indicators of the potential damage attached to the infrastructure [1-5]. In this regard,

detecting cracks in structure surfaces in the form of digital images captured by unmanned aerial vehicles (UAVs) [6, 7], smartphones [8, 9], or ground robots [10] has recently gained significant traction.

The first and foremost challenge for a structure inspection system is to develop a computer model that can automatically process the image data of concrete surfaces and identify the signs of structural damage and distress [2–5]. Therefore, researchers developed several defect detection models focusing on structure crack detection. These models design a variety of gradient features or operators for each image pixel and determine whether an image pixel contains a crack with a binary classifier. However, these models highly rely on expert-driving heuristic thresholds or hand-crafted features and are also not discriminative enough to differentiate the crack from complex image variations. Convolutional neural networks (CNNs)-based deep learning becomes an alternate method for automatic crack detection, inspired by successful applications in medical diagnosis. Instead of the hand-crafted features, the CNNs can automatically learn the discriminative features from the digital image to decide whether a crack exists. Zhang et al. developed a CNN model and trained it using image patches taken from pavement [11]. A notable increase in prediction accuracy was achieved compared to support vector machines and boosting methods. Cha et al. established an image dataset consisting of a total of 2,366 sub-images cropped from 300 annotated images of two bridges and a building [12]. He subsequently trained a faster region-based CNN architecture on this image dataset to identify damage with a reported average precision of 88%. Alipour et al. developed a full convolutional network FCN2s for pixel-level defect detection in concrete infrastructure systems. Sensitive analysis revealed that their model could correctly detect over 92% of crack pixels [4]. Chen and He present a novel U-shaped encoder-decoder network with an attention mechanism to achieve enhanced accuracy in pixel-level crack detection of concrete roads compared with other advanced networks [13]. Zhu et al. compared the performance of the three CNN algorithms-Fast R-CNN, YOLOv3, and YOLOv4-for distress detection based on UAV-captured pavement images [14]. Shang et al. presented a multi-fusion U-Net network to automatically detect the pixel-level pavement crack with superior performance compared to seven other state-of-the-art models [15]. Recently, Ha et al. develop a novel system integrating SqueezeNet, U-Net and mobilenet-SSD models together for detection, classification, and severity assessment of five types road cracks [16]. This combined system is capable of crack severity assessment and crack type classification with a reported accuracy of 91.2%. The recent success of deep learning methods for crack detection applications demonstrates the feasibility of fine-tuning CNNs within an ample parameter space [2-5, 11-16]. For instance, the CNN models, such as ConvNet, U-Net, and YOLO adopted in [11, 14, 15, 16], have more than 60 million parameters. The FCN2s network for pixel-level crack identification has 144 million parameters [4]. Iterative optimization of such a vast parameter space to determine the best network model is time-consuming and computationally costly. It is eager to develop a deep learning-based model for automated crack detection with a compact network structure for easy system embedding while maintaining equivalent or even better performance.

To this end, a deep learning-based system called CrackSN is implemented for automatic crack detection in concrete infrastructure, integrating the SqueezeNet network and the Adam optimization algorithm. The proposed deep learning model, which enables fast and stable inspection of concrete cracks, has a lightweight structure architecture compared to other CNNs and is, therefore, easy to implement for embedded and mobile systems. The proposed deep learning Adam-SqueezeNet based system, and components, i.e., image data preparation, SqueezeNet architecture, and Adam optimization, are described in detail. The experimental results of the presented model demonstrate the capability of the CrackSN system for crack detection. The presented system's limitations are discussed, followed by a brief conclusion.

METHODS

Due to their strong self-learning abilities, CNN models can classify large-scale image datasets with human-like accuracies [17]. A convolutional neural network is a chain of collaborative convolution layers, activation functions, pooling, and batch normalization operations. Sequential convolution layers transform the 2D or 3D image inputs into a high-level feature map with specified kernels, significantly reducing computation complexity and enhancing generalization ability [18]. In this work, we propose a system, namely CrackSN, integrated Adams optimization, and SqueezeNet architecture, for the diagnostic of concrete cracks from digital images is proposed in this work. The proposed framework intends to provide a discriminative and rapid system for automatically detecting distress in concrete surfaces from digital images, i.e., captured by smartphones or UAVs. This system can also facilitate quick status evaluation and decision-making in the maintenance of concrete infrastructure. The following sections describe the proposed framework, image data preparation, SqueezeNet architecture, and Adam optimization in detail.

Proposed system: Adam-SqueezeNet based CrackSN

Fig. 1 describes the overall architecture of the Adam-SqueezeNet based system CrackSN. The proposed system consists of three main stages: augmentation of the image dataset, training and validation of the Adam-SqueezeNet model, and the testing stage for decision-making. This system aims to classify the digital image data captured by smartphones or UAVs. In the first stage, the offline augmentation operation is performed on the captured images to overcome the possible imbalance problem. The augmented dataset is split into the train, validation, and test subsets without overlap. The training dataset comprises the image patch of the concrete surface and is set as the input of the CrackSN system for parameter optimization of the SqueezeNet network. The validation set contains patches different from the train set to evaluate the trained SqueezeNet model. The hyperparameters of SqueezeNet will be tuned based on the performance of validation. The test set is taken as the testing input for the optimized CrackSN system to provide an unbiased final evaluation of the trained SqueezeNet model. The SqueezeNet convolution network is adopted in the presented system. It utilizes the fire module, incorporating squeeze and expand layers, so a smaller and more effective CNN architecture is constructed. Adams optimization is utilized in the CNN network in order to obtain the best decision-making model. The best SqueezeNet model is later used for the decision-making process with the test set. The optimized network model classifies the images in the test dataset, and the classification performances are determined.

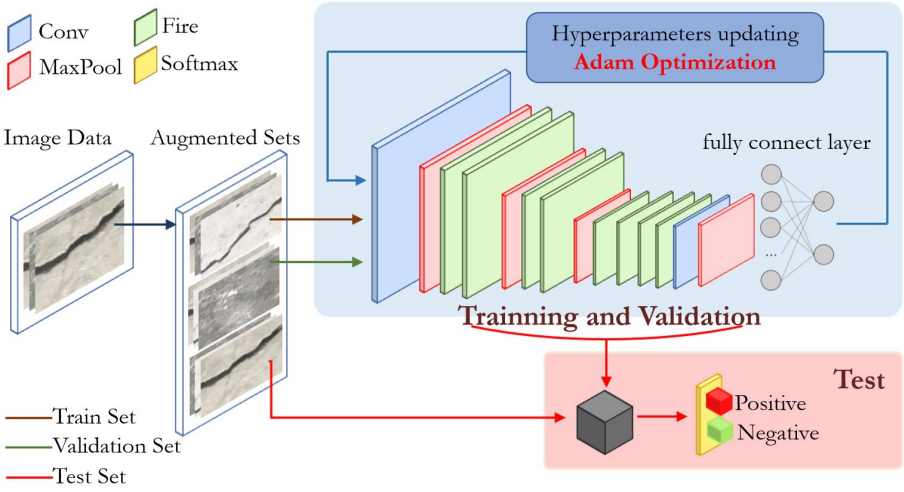

Figure 1: Architecture of the deep learning crack detection system.

Image data preparation

An image dataset of concrete crack images for classification was obtained from a total of 284 images (1706×1280 pixels, 8 bits, RGB channels) taken on the campus of Hefei University by a smartphone. Following the same methodology outlined in [11], the full-size, high-resolution images were cropped, resized into spatially sized square patches, and labeled into two classes. A sub-image was labeled as a cracked (i.e., Positive) patch if it was centered 10 pixels from the crack centroid; otherwise, it was classified as a normal (i.e., Negative) patch. To reduce the patch similarity in the image dataset, the overlap of two cracked or positive patches $P^{(1)}$ and $P^{(2)}$ is defined as $O=\text{area}(P^{(1)} \cap P^{(2)})/\text{area}(P^{(1)} \cup P^{(2)})$ should be maintained at a low level. The distance between the centers of two adjacent patches was set to 0.7 times the patch width.

The patches were resized to a fixed size of 227×227 pixels to accommodate the required input image size of the SqueezeNet model. Since crack features comprise only a small portion of the collected images, the sub-image was rotated around its center by a random angle $\gamma \in$ [0, 360] to boost the total number of cracked patches. The inadequacy of the image data has a detrimental effect on classification performance [19]. In order to overcome the imbalance problem of the image dataset, the sub-images were augmented by flipping them along their vertical axis and randomly shifting them up to 30 pixels in either direction or both directions. The augmentation operation also prevents the network from overfitting and memorizing the exact details of the image data during the training process. Fig. 2 shows the sample patches of the concrete surface with labels, which will be used for network training and validation. The obtained patches of concrete surfaces demonstrate variety in terms of texture, surface finish, illumination, and focus. The obtained dataset contains 6,000 images, half of which are cracked or positive patches. By random selection, the images after offline augmentation was split into the train, validation, and test image datasets. The number of images in the three corresponding datasets is 4000, 1000, and 1000, respectively.

Given the selection's randomness and the data's abundance, the normal and cracked images are split almost equally among the three image datasets.

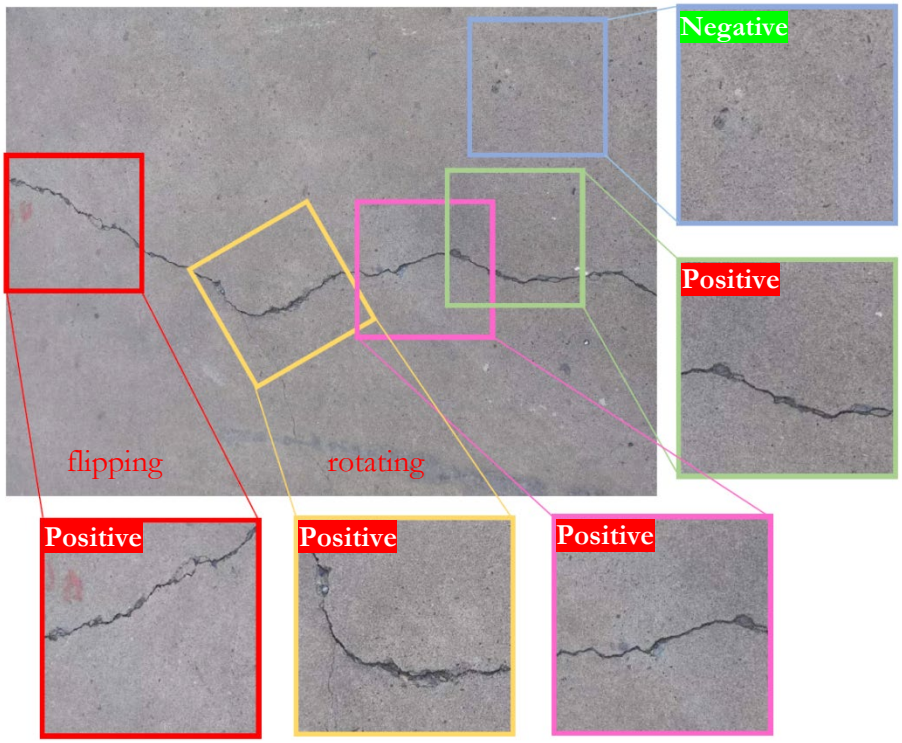

Figure 2: Concrete image with cracks and image patches obtained through cropping, flipping, and rotating augmentation.

SqueezeNet architecture

A deep learning algorithm based on SqueezeNet CrackSN is implemented for automatic crack detection. The SqueezeNet is a CNN-based architecture with only 1/50 parameters while maintaining the competitive accuracy of the AlexNet [20]. In contrast to CNNs, SqueezeNet replaces the filter or kernel size from 3×3 to 1×1 and reduces the number of input channels to 3×3 filters. These strategies bring significant parameter size reductions to CNNs while maintaining the prediction accuracy of the algorithm. To optimize its accuracy with a limited parameter size, SqueezeNet executes down-sampling later in the network so that convolutional layers have large activation maps.

As illustrated in Fig. 1, the augmented image datasets were used as input for network training and optimization. The SqueezeNet starts with a standalone convolution layer (Conv1), followed by eight fire modules (Fire2 to Fire9), and ends with a final convolution layer (Conv10). The entire layer configuration of the SqueezeNet-based architecture is presented in Tab. 1. A convolution layer is used to transform an input image into a feature map, as demonstrated in Fig. 3. The filter converts the input image $\boldsymbol{x}$ into a subsequent filtered image $\boldsymbol{y}$ or feature map, by

$$y_{<m,n>}=(x * w)_{i,j}=\sum_{t}\sum_{r} w_{t,r}x_{i-t,j-r} \quad \text{with } t, r=0,1,2. \tag{1}$$

The output dimension of a convolution operation is determined by two parameters, namely the step size of stride s and the padding length f.

The SqueezeNet model uniquely defines the fire module as a squeeze convolution layer (1×1 filter) that feeds into an expanded layer with a mix of 1×1 and 3×3 convolution filters. Both the squeeze convolution and expanded layers are connected to the rectified linear unit ReLU activation function. Compared to other functions in traditional neural models, ReLU brings in non-linearity and effectively speeds up the training and evaluation phases of the network [20]. The concatenate operation then stacks the expanded outputs up in the depth dimension as the input tensor of the subsequent convolution layer. Max-pooling with a stride step size of 2 was performed after the Conv1, Fire3, and Fire5 layers to summarize the feature response across neighboring pixels. The max-pooling process with a pooling filter size of 3x3 in the presented model further reduces the feature map $\boldsymbol{y}$,

$$z_{i,j}=\max(y_{i+t,j+r}) \quad \text{with } t, r=0,1,2. \tag{2}$$

These processes allow the implemented algorithm to learn spatially invariant features.

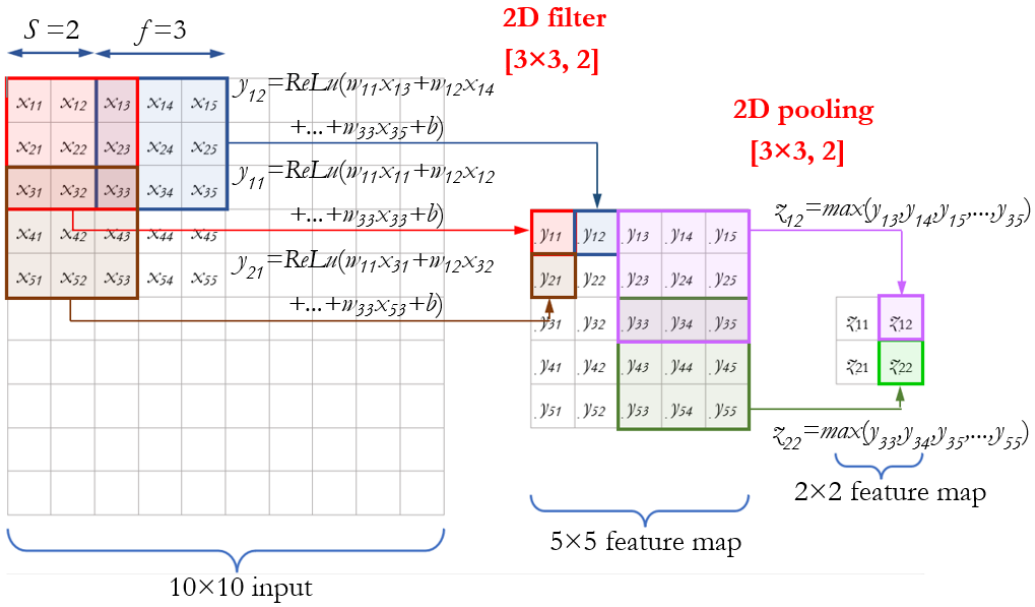

Figure 3: Schematic illustration of the 2D 3 × 3 filter and a stride of 2, and a max pooling with size 3 × 3 and a stride of 2.

The network training aims to increase the variation extracted from the input image data while preventing overfitting. A dropout action with a probability of 0.5 prevents complex co-adaptations after the Fire9 module, reducing the possibility of overfitting. The global average pooling operation converts the class feature maps into a single value later. A full-connected 2-class layer was implemented to replace the original 1000-class layer of SqueezeNet [21]. At the end of the network, the softmax activation function calculates the probability of negative or positive classes given an input image for the final classification decision-making. Given an image dataset $S=\{P^{(i)}, L^{(i)}\}$ of m image patches, where $P^{(i)}$ is the i-th patch and $L^{(i)} \in \{0,1\}$ is the associated class label. If $P^{(i)}$ is a positive patch, $L^{(i)}$ is assigned as 1; otherwise, $L^{(i)}$ is set to 0. Let $z_j^{(i)}$ be the output of unit j in the softmax layer for image $P^{(i)}$, then the probability that the label $L^{(i)}$ of $P^{(i)}$ is j can be calculated by the formula

$$p(L^{(i)}=j \mid z_j^{(i)})=\frac{e^{z_j^{(i)}}}{\sum_{l=1}^{k} e^{z_j^{(i)}}} \tag{3}$$

The literature [19] is referred to interested readers for algorithm details. During training, the first small batch images of concrete surface were input into the SqueezeNet with initial parameter settings. In the SqueezeNet network, max pool layers execute a down-sampling operation in spatial dimensions. The Fire moduli sequentially transform the image inputs into high-level feature maps through the squeeze and expand operations. The network outputs are a weighted combination of the feature maps on tensors. The feature map of classes is later converted into one value by the global average pool operation, which gives the multiclass probability distribution by the Softmax activation function at the end of the network. The loss function in the cross entropy loss is calculated to compare the classification results to image labels. The parameters, weighting, and functionality are then adjusted, as needed, for network training on the following batch images iteratively. In parallel, a validation process is performed to estimate the accuracy of the developing network. The validation process aims to assess the function and performance in crack detection of the SqueezeNet under training against unbiased and independent inputs. The hyperparameters controlling the overall process are updated if the validation error is unsatisfactory. After various iterations of resampling and fine-tuning in train and validation stages, the best SqueezeNet model is obtained once the desired performance metrics are achieved. The functionality of the best model obtained is verified on the test or

holdout image set for practical application on unseen inputs. In the test procedure, the model works as a black box with images passed through it to diagnose if a crack exists in the concrete surface or not. The hyperparameters of the model will not be adjusted in the testing stage.

Layer	Type	Kernel Size	Stride	Padding	Output Size
Data	-	-	-	-	227×227×3
Conv1	{Conv+ReLU}	3×3	2	0	113×113×64
Pool1	Max Pool	3×3	2	0	56×56×64
Fire2/Fire3	{Squeeze+ReLU}	1×1	1	0	56×56×16
	{Expand+ReLU}	1×1	1	0	56×56×64
	{Expand+ReLU}	3×3	1	1	56×56×64
	Concat				56×56×128
Pool3	Max Pool	3×3	2	0	28×28×128
Fire4/Fire5	{Squeeze+ReLU}	1×1	1	0	28×28×32
	{Expand+ReLU}	1×1	1	0	28×28×128
	{Expand+ReLU}	3×3	1	1	28×28×128
	Concat				28×28×256
Pool5	Max Pool	3×3	2	0	14×14×256
Fire6/Fire7	{Squeeze+ReLU}	1×1	1	0	14×14×48
	{Expand+ReLU}	1×1	1	0	14×14×192
	{Expand+ReLU}	3×3	1	1	14×14×192
	Concat				14×14×384
Fire8/Fire9	{Squeeze+ReLU}	1×1	1	0	14×14×64
	{Expand+ReLU}	1×1	1	0	14×14×256
	{Expand+ReLU}	3×3	1	1	14×14×256
	Concat				14×14×512
Conv10	Conv+ReLU	1	1	0	14×14×1000
Pool10	Global Average Pool	-	-	-	1×1×1000
fc_add	Fully Connect	-	-	-	1×1×2
Output	Softmax	-	-	-	1×1×2

Table 1: Detailed layer configuration of the implemented network.

Adam Optimization

During training, the parameters of the SqueezeNet model were fine-tuned through the optimization iteration updated by the validation error. The hyperparameters in the deep learning model play a crucial role in deep learning algorithms, as these parameters tightly control the actions of the training algorithms and significantly affect the performance of the models. The parameters in the SqueezeNet model are synergistically optimized by minimizing the misclassification error during training. The Adam optimization algorithm is used to fine-tune the parameters of the SqueezeNet-based model. The Adam algorithm improves network training by using learning rates that differ by parameter and can automatically adapt to the loss function being optimized. It maintains an element-wise moving average of both the parameter gradients and their squared values,

$$m_l=\beta_1 m_{l-1}+(1-\beta_1)\nabla E(\theta_l) \tag{4}$$

$$v_l=\beta_2 v_{l-1}+(1-\beta_2)[\nabla E(\theta_l)]^2 \tag{5}$$

where ℓ is the iteration number, β_1 and β_2 are the decay rates, θ is the parameter vector, and $E(\theta)$ is the loss function. The Adam algorithm updates the network parameters using the moving averages as

$$\theta_{l+1}=\theta_l-\frac{\alpha m_l}{\sqrt{v_l}+\varepsilon} \tag{6}$$

where $a>0$ is the learning rate, and ε is a small positive value. If the gradients are similar over many iterations, a moving average of the gradient allows the parameter updates to pick up momentum in a particular direction. If the gradients contain mostly noise, then the moving average of the gradient becomes smaller, and so the parameter updates become smaller too. The full Adam update also includes a mechanism to correct for any bias that appears at the beginning of training. The trained model with optimized parameters was later applied to the test dataset for crack detection decision-making.

RESULTS AND DISCUSSION

Automatic crack detection system CrackSN was implemented and tested in the MATLAB R2022b platform on a desktop with an Intel(R) Core(TM) i7-13700KF CPU running at 3.40GHz, 64GB of RAM, and an NVIDIA RTX3090 GPU of 24GB. The network was trained using the Adam optimizer with a batch size of 100 examples and an initial learning rate of 0.0002. The performance of the implemented system was evaluated using three metrics. Recall (REC), or accuracy, is the ratio of correct crack predictions to the number of crack patches. Precision (PRE) is the ratio of correct crack precisions to total crack predictions. The F1 score is the harmonic mean of precision and recall. The cracked and uncracked patches were designated as positive and negative, accordingly.

$$\mathrm{REC}=\frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}} \tag{7}$$

$$\mathrm{PRE}=\frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}} \tag{8}$$

$$\mathrm{F1}=\frac{2\cdot\mathrm{PRE}\cdot\mathrm{REC}}{\mathrm{PRE}+\mathrm{REC}} \tag{9}$$

where, in turn, TP, TN, FP, and FN stand for the corresponding true positive, true negative, false positive, and false negative. Fig. 4 depicts the changes in loss function and crack accuracy of the SqueezNet-based model during its training and validation processes. The loss function achieves its minimum after ten epochs on the training set, indicating the convergence of the weights. The detailed performance of the implemented crack detection system, CrackSN, on the training and validation image datasets, is presented in Tab. 2. The implemented SqueezeNet-based model was compared with state-of-the-art models such as CrackPix FCN2s [4], classification VGG16 [4], and ConvNet [11]. It should be noted that every model was trained on different image datasets. In addition, the CrackPix FCN2s model provides crack identification at the pixel level, as opposed to the patch level of the other models. Compared to the reported results of the models mentioned above in the literature, our model provides an encouraging performance in detecting the concrete crack at the patch level. The values of REC, PRE, and F1 metrics demonstrate the effectiveness of our model CrackSN for automatic crack detection from concrete surface images. Meanwhile, the presented system, CrackSN, based on the Adam-SqueezeNet deep learning architecture, has only 1.24 million parameters and a model size (implemented in MATLAB R2022b) of around 5MB. Although a direct efficiency comparison of the ConvNet, the classification VGG16, and the CrackPix FCN2s models is not available for testing on the same image dataset, the training expense of these models can be indirectly evaluated from their backbone networks. The ConvNet model built on the CNN network has more than 60 million parameters and a code size of 227 MB. The classification VGG16 and the CrackPix FCN2s adopt the VGG16 and VGG19 backbones, which have about 140 million parameters and more than 500 MB in code implementation. The deep learning structure of the proposed system in this work is much more compact than the other models listed in Tab. 2. When integrated with cloud storage and an unmanned aerial vehicle for image acquisition, the Adam-SqueezeNet-based system CrackSN can potentially be used for real-time and remote health diagnosis of concrete infrastructure. The advantage in computational efficiency and lightweight structure of the presented system enables the work to be mobile, appealing to civil engineers for structure damage evaluation with satisfied performance.

Fig. 5 shows the test image and the calculated probability map of correct classification by the proposed crack detection system, CrackSN. The probability map of the pixel in red indicates high confidence in classifying the image as cracked or not. For instance, normal images without crack characteristics typically have a large, connected area with high confidence. In contrast to normal or uncracked images, the probability maps of cracked images are highly localized to the cracked pixels. These results demonstrate the discriminative capability of the presented deep learning model. For further evaluation, the ground truth segmentation of concrete cracks at the pixel level was obtained from the identified positive or cracked patches and shown in Fig. 6. Crack size is an essential reference when assessing the in-service condition of concrete buildings [3-5, 14]. The crack size information, such as the type, width, length, and location of the visible crack, is crucial to evaluating the severity of cracking development and the evolution of damage in the structure. Decisions on subsequent maintenance could be made by tracking and comparing the cracking condition of a concrete infrastructure at each inspection to ensure the serviceability and integrity of the structure.

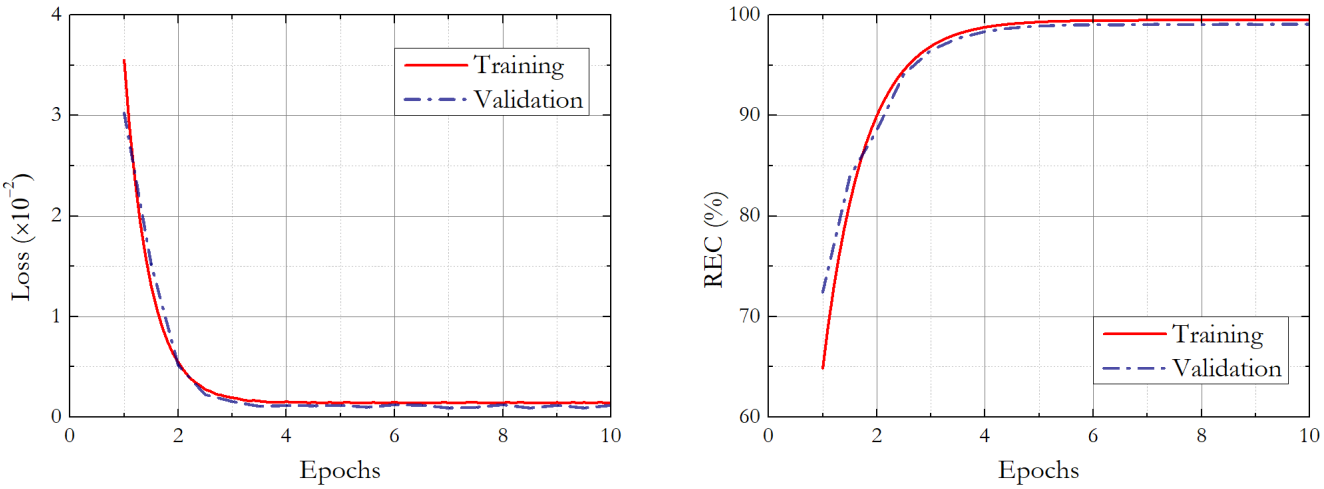

Figure 4: Development of loss and crack accuracy REC during training of the CrackSN model.

Model	REC	PRE	F1
CrackSN	**0.973**	**0.986**	**0.962**
ConvNet[11]	0.870	0.925	0.897
VGG16[4]	0.938	0.919	0.928
CrackPix FCN2s[4]	0.922	0.912	0.917

Table 2: Performance of patch-level and pixel-level models.

The experimental results of the presented system prove a successful first step in automatically detecting concrete cracks from captured images. The limitations of the CrackSN system are worth mentioning. First, our model is a patch-level crack detection system to identify whether there is a crack or not within a given image. The details of the crack isolation at the pixel level are obtained through the following segmentation operation. Therefore, an implementation of up-sampling or deconvolution layers would be embedded later in the present model to establish a pipeline flow for pixel-level classification and crack information determination [3, 4]. Secondly, the SqueezeNet model can only operate on a specific image size of 227×227, so our model is highly image size dependent. Such dependence severely constrains flexibility by taking images of arbitrary sizes from various imaging devices, such as smartphones, cameras, and UAVs. A fully convolutional network is considered an ideal model to bypass the size dependency, enabling the pixel-level segmentation of cracks [4, 5, 14, 22]. Last but not least, the training dataset does not account for the potential crack patterns, background material, texture, and color appearance, which can cause a significant amount of variance in the captured images of the building surface that are recorded [14]. As a result, it will be necessary to build a versatile image dataset that can achieve realistic scenarios, including various crack patterns, crack-like distractions, and background characteristics.

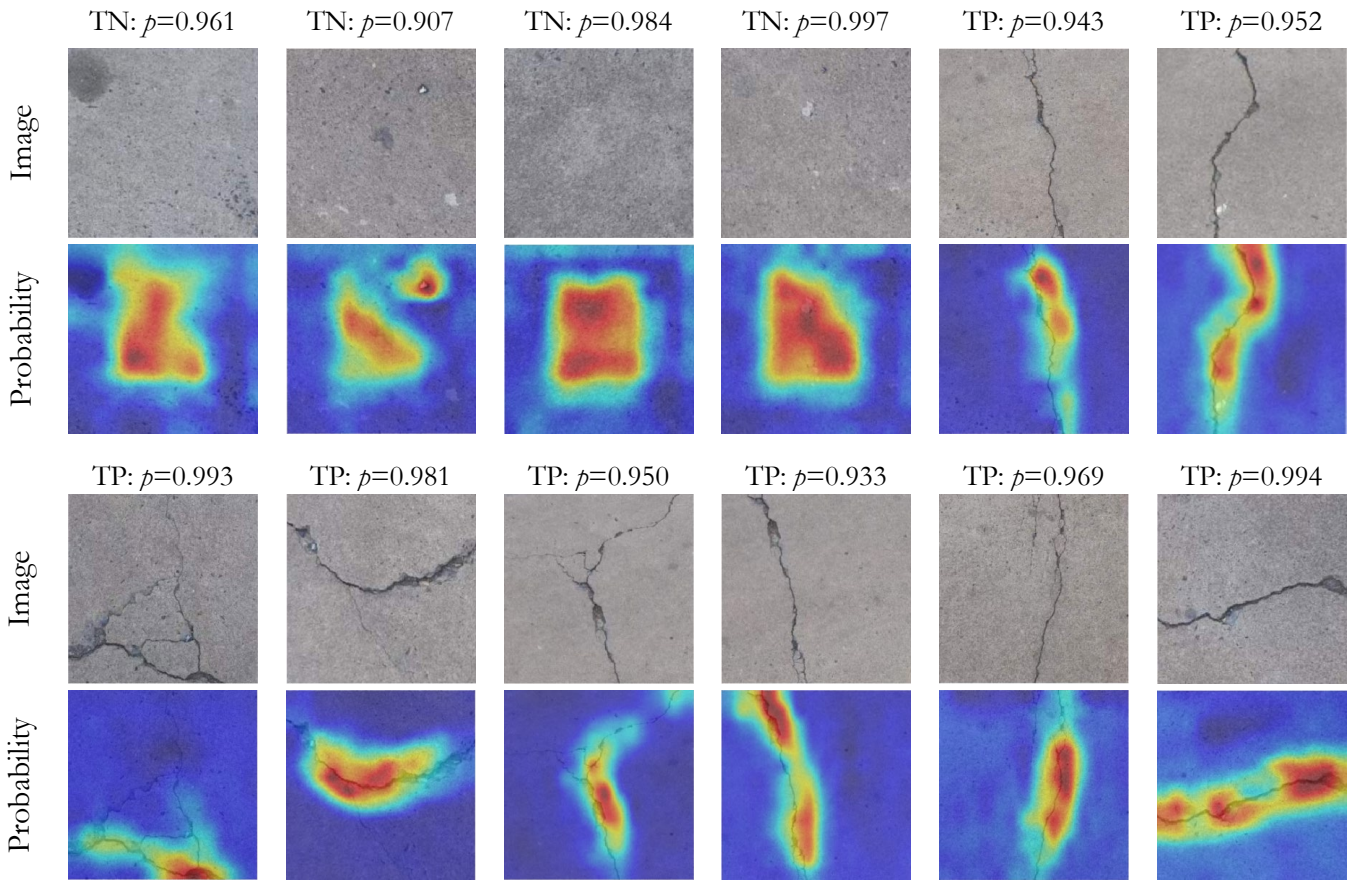

Figure 5: Detection of concrete images and corresponding probability maps.

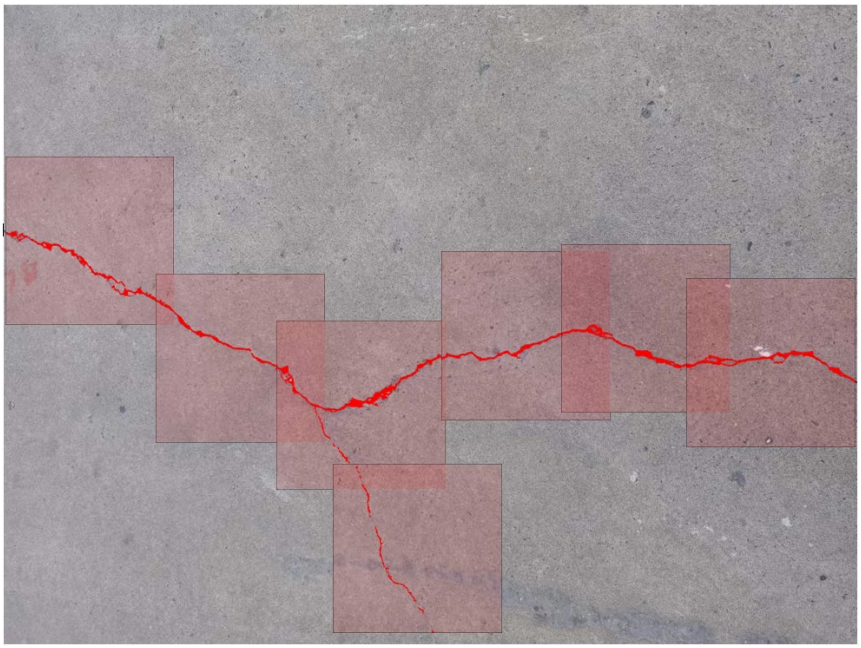

Figure 6: Segmentation of concrete crack.

Conclusion

Cracks in concrete infrastructure are crucial indicators of structural damage. They severely affect structural integrity and functionality. In this research, we proposed an automatic decision-making system CrackSN based on deep learning to inspect cracks in captured digital images. The experimental results of our model provide the following conclusions:

1. Through the training and validation process, the CrackSN system automatically learns the features from manually annotated image dataset and successfully predicts 97.3% of the cracked patches in the test datasets. The presented Adam-SqueezeNet deep learning based CrackSN system exhibits superior performance in crack detection compared with state-of-the-art models.
2. The deep Adam-SqueezeNet model with a small structural size and superior performance is preferred in embedded applications for real-time crack or damage detection in concrete infrastructure for fast damage detection and decision-making of maintenance.
3. The performance of our model can be further improved to pixel-level accuracy by incorporating up-sampling layers or other deep learning algorithm. Training the presented model on a wider variety of image datasets with realistic scenarios is also necessary for practical engineering applications.

Acknowledgment

The financial support from the University Nature Sciences Research Program of Anhui Province (KJ2021A1001) and the Academic Visit Program of Anhui Province (gxgnfx2022054) is greatly appreciated.

Nomenclature

Symbol	Meaning	Symbol	Meaning
$\boldsymbol{x}$	matrix of input image	$\boldsymbol{y}$	feature map
$\boldsymbol{w}$	weight of convolution filter	$\boldsymbol{z}$	max pool
s	step size of stride	f	padding length
S	image dataset	P	image patch
L	class label of image patch	q	output of softmax
m	parameter gradient	v	squared parameter gradient
β_1, β_2	decay rates	l	iteration number
θ	parameter vector	$E(\theta)$	loss function
γ	rotational angle of image patch	a	learning rate

Reference

[1] Sony, S., Laventure, S., Sadhu, A., (2019). A literature review of next-generation smart sensing technology in structural health monitoring. Struct. Control Health Monit., 26(3), e2321. DOI: 10.1002/stc.2321

[2] Ye, X. W., Jin, T., Yun, C. B., (2019). A review on deep learning-based structural health monitoring of civil infrastructures. Smart Struct. Syst., 24(5), pp. 567-585. DOI: 10.12989/sss.2019.24.5.567.

[3] Fei, Y., Wang, K. C., Zhang, A., Chen, C., Li, J. Q., Liu, Y., Li, B., (2019). Pixel-level cracking detection on 3D asphalt pavement images through deep-learning-based CrackNet-V. IEEE trans. Intell. Transp. Syst., 21(1), pp. 273-284. DOI: 10.1109/TITS.2019.2891167.

[4] Alipour, M., Harris, D. K., and Miller, G. R., (2019). Robust pixel-level crack detection using deep fully convolutional neural networks. J. Comput. Civ. Eng., 33(6), 04019040. DOI: 10.1061/(ASCE)CP.1943-5487.0000854.

[5] Ni, F., Zhang, J., Chen, Z., (2019). Pixel-level crack delineation in images with convolutional feature fusion. Struct. Control Health Monit., 26(1), e2286. DOI: 10.1002/stc.2286.

[6] Zhao, S., Kang, F., Li, J., Ma, C., (2021). Structural health monitoring and inspection of dams based on UAV photogrammetry with image 3D reconstruction. Autom. Constr., 130, 103832. DOI: 10.1016/j.autcon.2021.103832.
[7] Yan, Y., Mao, Z., Wu, J., Padir, T., Hajjar, J. F., (2021). Towards automated detection and quantification of concrete cracks using integrated images and lidar data from unmanned aerial vehicles. Struct. Control Health Monit., 28(8), e2757. DOI: 10.1002/stc.2757.
[8] Ellenberg, A., Kontsos, A., Moon, F., & Bartoli, I. (2016). Bridge related damage quantification using unmanned aerial vehicle imagery. Struct. Control Health Monit., 23(9), pp. 1168-1179. DOI: 10.1002/stc.1831.
[9] Dorafshan, S., Thomas, R. J., Maguire, M., (2018). Comparison of deep convolutional neural networks and edge detectors for image-based crack detection in concrete. Constr. Build Mater., 186, pp. 1031-1045. DOI: 10.1016/ j.conbuildmat.2018.08.011.
[10] Gurunsi, N., B. Basily, J. Kim, J. Yi, T. Duong, K. Dinh, S.-H. Kee, and A. Maher., (2017). RABIT: Implementation, performance validation and integration with other robotic platforms for improved management of bridge decks. Int. J. Intell. Rob. Appl., 1(3), pp. 271–286. DOI: 10.1007/s41315-017-0027-5.
[11] Zhang, L., Yang, F., Zhang, Y. D., Zhu, Y. J., (2016). Road crack detection using deep convolutional neural network. In 2016 IEEE international conference on image processing (ICIP), pp. 3708-3712. IEEE. DOI: 10.1109/ ICIP.2016.7533052.
[12] Cha, Y. J., W. Choi, O. Büyüköztürk., (2017). Deep learning based crack damage detection using convolutional neural networks. Comput.-Aided Civ. Infrastruct. Eng., 32(5), pp. 361–378. DOI: 10.1111/mice.12263.
[13] Chen, J., He, Y., (2022). A novel U-shaped encoder–decoder network with attention mechanism for detection and evaluation of road cracks at pixel level. Comput.-AidedCiv. Infrastruct. Eng., 37(13), pp. 1721-1736. DOI: 10.1111/ mice.12826.
[14] Zhu, J., Zhong, J., Ma, T., Huang, X., Zhang, W., Zhou, Y., (2022). Pavement distress detection using convolutional neural networks with images captured via UAV. Autom. Constr., 133, 103991. DOI: 10.1016/j.autcon. 2021.103991.
[15] Shang, J., Xu, J., Zhang, A. A., Liu, Y., Wang, K. C., Ren, D., He, A., (2023). Automatic pixel-level pavement sealed crack detection using multi-fusion u-net network. Measurement, 112475. DOI: 10.1016/j.measurement.2023.112475.
[16] Ha, J., Kim, D., Kim, M., (2022). Assessing severity of road cracks using deep learning-based segmentation and detection. J. Supercomput., 78(16), pp. 17721-17735. DOI: 10.1007/s11227-022-04560-x.
[17] Ucar, F., Korkmaz, D., (2020). COVIDiagnosis-Net: Deep Bayes-SqueezeNet based diagnosis of the coronavirus disease 2019 (COVID-19) from X-ray images. Med. hypotheses, 140, 109761. DOI: 10.1016/j.mehy.2020.109761.
[18] Raghu, S., Sriraam, N., Temel, Y., Rao, S. V., Kubben, P. L., (2020). EEG based multi-class seizure type classification using convolutional neural network and transfer learning. Neural Netw., 124, 202-212. DOI: 10.1016/j. neunet. 2020.01.017.
[19] Buda, M., Maki, A., Mazurowski, M. A., (2018). A systematic study of the class imbalance problem in convolutional neural networks. Neural Netw., 106, pp. 249-259. DOI: 10.1016/j.neunet.2018.07.011.
[20] Iandola, F. N., Han, S., Moskewicz, M. W., Ashraf, K., Dally, W. J., Keutzer, K., (2016). SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and< 0.5 MB model size. arXiv preprint arXiv:1602.07360. DOI: 10.48550/arXiv.1602.07360.
[21] Kenta, (2023). Classify crack image using deep learning and explain "WHY". https://github.com /KentaItakura/releases/tag/v1.1
[22] Long, J., Shelhamer, E., Darrell, T., (2015). Fully convolutional networks for semantic segmentation. In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 3431-3440. DOI: 10.48550/arXiv.1411.4038.