ISSN: 2472-3703 DOI: 10.6911/WSRJ.202405_10(5).0007

Road Crack Detection Network Based on Stage Feature Reuse and Adaptive Threshold

Wenmao Hu[1, a], Shibao Sun[1, b, *]

[1]School of Information Engineering, Henan University of Science and Technology, Luoyang, China

[a]huwenmao123@126.com, [b]sunshibao@haust.edu.cn

Abstract

An important factor affecting road safety is road cracks. If road cracks are not discovered and repaired in time, safety hazards will increase over time, and eventually traffic accidents will occur, causing irreversible losses. The detection of cracks often requires on-site inspection by engineers with rich experience, which has the problems of low efficiency, high cost, and many false detections due to fatigue. With the success of deep learning in various fields of computer vision, deep learning is naturally introduced into the crack detection task. However, in the crack detection task, convolutional neural networks with many parameters can easily cause overfitting. To solve this problem, we propose stage feature reuse convolutional neural network. In addition, deep learning is a data-driven task. In the manually labeled crack detection data set, the proportion of positive samples in the samples is small, and the proportion of positive samples in each image in the data set is different. It is difficult for the model to learn to capture ability to achieve small targets, so we propose a adaptive threshold method for predicting each crack prediction map to alleviate this problem.

Keywords

Crack detection; Semantic segmentation; Deep learning; Convolutional neural network.

1. INTRODUCTION

Road cracks refer to linear or surface fissures or gaps that appear on the surface of roads, typically caused by factors such as traffic loads, temperature changes, water penetration, and material aging. These cracks pose serious hazards to the safety and smoothness of roads. When cracks exist, they can lead to a decrease in the stability of the road surface, hindering vehicle travel, thereby affecting driving safety and efficiency. Additionally, cracks may also result in water infiltration into the bottom of the road surface, accelerating the softening and degradation of the roadbed material, leading to road subsidence, potholes, and cracking, significantly impacting the service life of the road and increasing maintenance costs. Therefore, timely repair and prevention of road cracks are crucial to ensuring smooth and safe passage on the roads.

With the rise of deep learning, convolutional neural networks (CNNs) have been widely applied to various computer vision tasks such as image classification, segmentation, and object detection, achieving significant success in these fields. In order to address the issues of low efficiency and high cost associated with manual detection of road cracks, utilizing CNNs for automated detection of road cracks has become a popular approach.

However, there still exist numerous issues and challenges in applying convolutional neural networks to crack detection tasks. Firstly, in crack detection tasks, the images of the roads to be

ISSN: 2472-3703 DOI: 10.6911/WSRJ.202405_10(5).0007

inspected are composed of a large amount of uniform texture and color. This differs from other computer vision tasks such as classification or segmentation, where the training data is more diverse, encompassing not only texture and color but also various object and semantic information. This means that crack detection tasks are prone to overfitting when using convolutional neural networks, leading to significant redundancy in computation and an increase in deployment costs due to the excessive number of parameters. Furthermore, crack detection, similar to segmentation and other semantic segmentation tasks, requires setting a threshold typically around 0.5 to binarize the probability map of cracks predicted by the neural network. However, in crack detection tasks, cracks vary in shape and size, leading to significant non-uniformity in foreground and background distributions. Consequently, there is a considerable deviation in the confidence distribution of predictions between samples, making it unreasonable to uniformly set 0.5 as the threshold.

To tackle the aforementioned issues, we propose a Stage-wise Feature Reuse network, SFR-Crack, which incorporates and utilizes features from all preceding stages in the convolutional neural network modeling process to alleviate overfitting. Additionally, we introduce an adaptive thresholding method where a dynamic threshold is predicted by the neural network to enhance the stability of prediction results.

2. RELATED WORK

Road crack detection tasks are equivalent to binary semantic segmentation tasks in computer vision. A typical and widespread application scenario for binary semantic segmentation tasks is medical image segmentation. U-Net [1] is a classic model for medical image segmentation. Due to its simple architecture and versatility, it has been applied in numerous other domains with considerable success. Its unique "U" shape comprises a contracting path for context extraction and an expansive path for precise localization. Skip connections facilitate information flow between these paths, enabling the retention of fine details during upsampling. Another noteworthy development is the emergence of Swin-Unet, a U-Net model based purely on Transformers, following the success of Transformers in the field of computer vision. UNet combines the self-attention mechanism of Transformers, introducing new possibilities for image segmentation tasks. Swin-UNet [2] inherits the U-shaped structure of UNet but replaces traditional convolutional layers with Swin Transformer [3] modules based on Transformers. This adaptation allows the model to better capture long-range dependencies when dealing with large-scale images. By leveraging the successful experiences of Transformers in sequence modeling, this integration brings higher performance and efficiency to image segmentation tasks.

Since the vast majority of road cracks are linear in nature, the detection of road cracks can also be considered a form of edge detection. In edge detection tasks, one of the more classical and influential methods is HED (Holistically-Nested Edge Detection) [4]. Inspired by Fully Convolutional Networks (FCN) [5] ,HED employs a pruned VGG16 as its backbone to extract multi-level features from input images. It comprises two main branches: the side branch, responsible for merging feature maps and computing side loss, and the fusion branch, averaging prediction scores from the side branch and computing fusion loss. This multi-branch structure is a key feature. Feature fusion involves convolution, bilinear interpolation, and weighted feature map fusion, enabling effective fusion of features from different levels. Following HED, RCF (Richer Convolutional Features for edge detection) [6] was proposed to enhance HED. RCF model focuses on utilizing features extracted at each convolutional layer to prevent loss of important edge details with increasing model depth. It aggregates features progressively from shallow to deep layers at different downsampling stages, capturing information at various scales and semantics. After aggregation, RCF upsamples the features to the input image

ISSN: 2472-3703 DOI: 10.6911/WSRJ.202405_10(5).0007

resolution, ensuring detailed information preservation. Finally, it combines high-resolution feature maps to generate the edge prediction map. BDCN(Bi-directional cascade network) [7] addresses whether shallow networks can match or exceed the performance of deep networks in edge detection tasks. It introduces a Scale Enhancement Module (SEM) to leverage dilated convolutions with different dilation rates for multi-scale feature fusion. Additionally, it employs a bidirectional cascade structure with top-down and bottom-up paths to integrate detailed and semantic features effectively. By using VGG16 as the backbone, BDCN outperforms RCF, which relies on a deeper ResNet-101 [8] backbone.

Convolutional neural networks specifically designed for crack detection are mostly based on popular segmentation frameworks that are pruned or modified. For example, FPHBN(Feature pyramid and hierarchical boosting network) [9] introduces feature pyramids and hierarchical boosting networks to the framework of HED. DeepCrack [10] combines the encoder and decoder outputs of each stage of Segnet and utilizes features of different scales from these stages for crack detection.

3. METHOD

3.1. STDC block

Since AlexNet [11] achieved great success in the computer vision ImageNet classification task, convolution operations have been widely used in image processing tasks. However, standard convolutional operations apply convolutional kernel parameters to all channels, resulting in a large number of parameters and redundant computations. This could potentially lead to overfitting issues in crack detection tasks. Therefore, we choose to use the STDC (short-term dense concatenate) [12] module to replace the standard convolution module. In the standard convolution module, a convolution operation is followed by a batch normalization layer and a ReLU activation function layer. But in the STDC block, the number of channels in the input features is first mapped to a hidden space with half the previous number of channels through a linear layer (i.e., a convolution operation with a 1×1 convolution kernel). This operation reduces computational and parameter overhead by reducing the number of feature channels. The number of feature channels is halved to create a feature map, which is both retained for future use and passed to the next 3×3 convolutional layer. This 3×3 convolutional layer not only performs convolutional operations on the feature map but also reduces the number of feature channels back to half of their original count. Repeat the above steps, where the i-th convolutional layer adjusts the number of feature channels in the convolutional operation to $N/2^i$, where N is the number of input feature maps. The final convolutional layer differs from the previous ones in that it no longer adjusts the number of feature channels; instead, it maintains the same number of feature channels as the output of the preceding convolutional layer. Here, we set i to be 4. Finally, concatenate the feature maps preserved from each convolutional layer to generate the final output. The overall structure of the STDC block is shown in Figure 1.

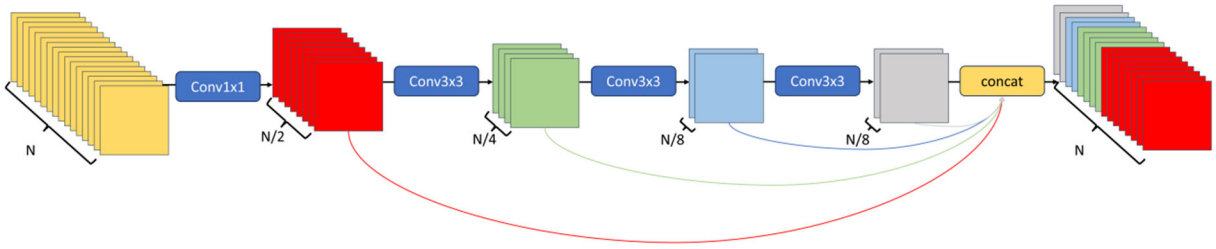

Figure 1. Overall structure of STDC block.

ISSN: 2472-3703 DOI: 10.6911/WSRJ.202405_10(5).0007

3.2. Stage feature reuse network

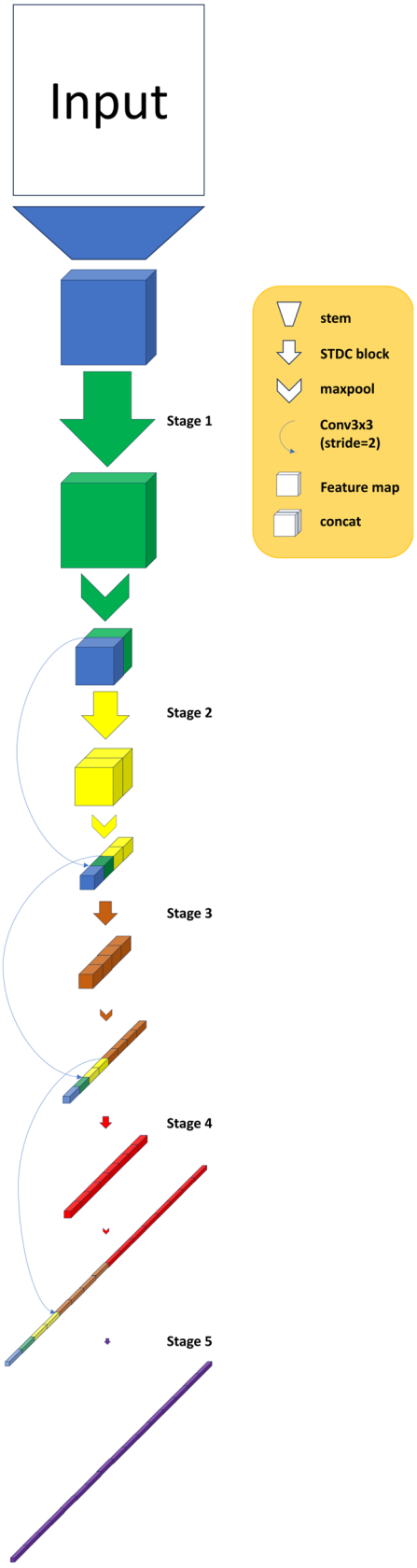

Figure 2. Overall structure of the stage feature reuse network

ISSN: 2472-3703 DOI: 10.6911/WSRJ.202405_10(5).0007

Since ResNet introduced residual structures to alleviate the vanishing gradient problem caused by the deep depth of convolutional neural networks, this design approach with skip connections has garnered widespread attention from researchers. Skip connections can be viewed as a form of feature reuse. Inspired by this, DenseNet [13] was proposed. DenseNet takes the output of each convolutional layer as the input for all subsequent convolutional layers, resulting in dense connections, which are a characteristic feature of DenseNet. Inspire by DenseNet, We consider each stage as a unit and use the outputs of these units as inputs for the next stage. The overall structure of the network is shown in Figure 2.

DenseNet consists of dense connection layers and transition layers. Our network differs from it because even with the constraints of the growth rate in dense connection layers, the number of feature channels gradually increases with the number of layers. This is not conducive to minimizing computational and parameter overhead, so we replace dense connection layers with STDC blocks. Most importantly, DenseNet halves the output of dense connection layers at the end of each stage using transition layers, which results in the loss of information from some previous stage feature maps. In our network, the feature maps grown after downsampling are downsampled from feature maps of all previous stages without any reduction, maximizing the utilization of information from each stage.

Our decoder module is similar to U-net, but instead of using concatenation for skip connections, we directly perform element-wise addition. Blue arrows represent convolutional layers with 3×3 kernel size, while orange arrows represent transposed convolutional layers with 4×4 kernel size for 2× upsampling. Before upsampling, the signal goes through an STDC block for smoothing. The overall decoder structure is shown in Figure 3.

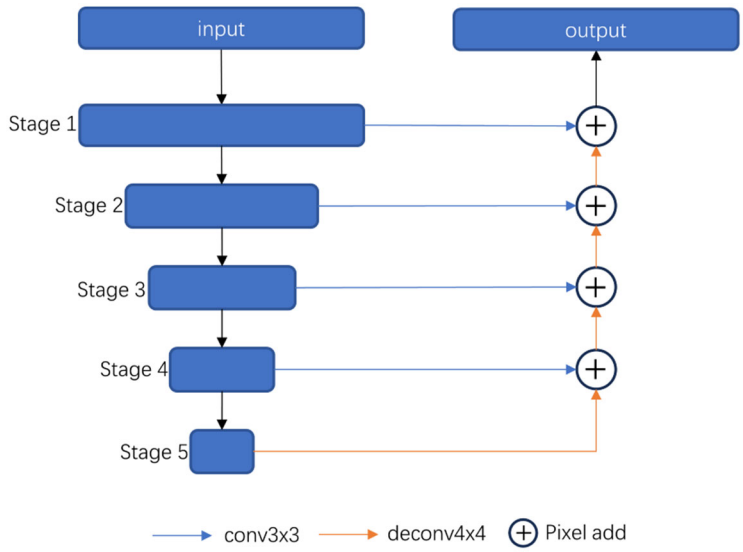

Figure 3. Overall decoder structure

3.3. Adaptive threshold

Cracks or fissures, often appear in a narrow and elongated manner, resulting in small areas but long lengths. However, cracks or fissures are not merely lines, any defect that significantly impacts the integrity of the structure can be considered as a crack or fissure. Therefore, model needs to capture not only lines but also areas. Due to variations in the sizes of the gaps, some may have large areas, while others may be small but elongated, resulting in a severe class imbalance issue, posing a challenge for deep learning models. In the usual workflow, after feeding each input image into the model to obtain prediction results, a threshold of 0.5 is used: values with prediction confidence greater than 0.5 are regarded as positive samples, while values with confidence less than or equal to 0.5 are regarded as negative samples, which is

ISSN: 2472-3703 DOI: 10.6911/WSRJ.202405_10(5).0007

disadvantageous for predicting and segmenting small targets. Therefore, we hope the model can autonomously evaluate its predictions for each input image. For results with higher certainty, the threshold of confidence is lowered, so that probabilities above the threshold become the prediction results. For ambiguous results, the threshold of confidence is raised to ensure that high threshold outputs as many correct predictions as possible.

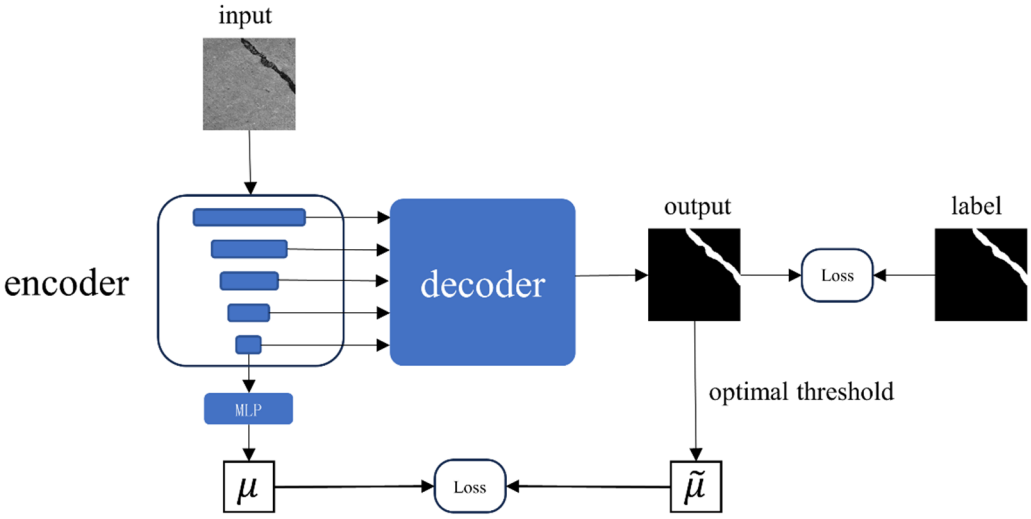

Figure 4. Schematic diagram of the training stage for adaptive thresholding method

As a result, we designed the method as follows, starting with the training stage shown in Figure 4. The input image is first fed into the encoder, and the features extracted at various stages of the encoder are passed to the decoder module, which outputs images. These images are then used to compute the loss with the labels, supervising the output of the crack detection model. Confidence scores from the output are iterated as thresholds from 0.1 to 0.99 to find the threshold that maximizes the F1 score, which is then used as the label. On the other hand, after the input is encoded by the encoder, the last layer connects to a multi-layer perceptron (MLP), which predicts a threshold. This threshold is used for loss computation alongside the best threshold label obtained from the output result iteration. And in the inference stage as shown in Figure 5, the thresholds predicted by the multi-layer perceptron directly binarize the predictions output by the crack detection model to form the final output.

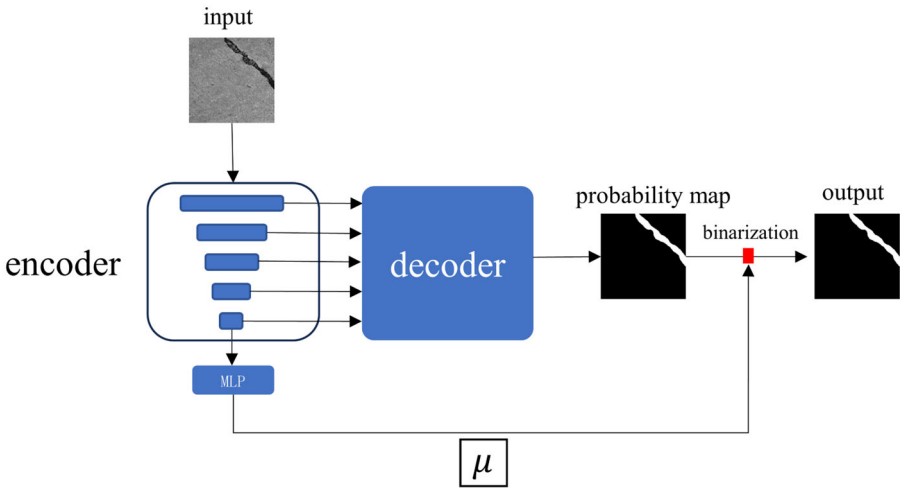

Figure 5. Schematic diagram of the inference stage for adaptive thresholding method

ISSN: 2472-3703 DOI: 10.6911/WSRJ.202405_10(5).0007

4. EXPERIMENT AND RESULTS

4.1. Loss function

Due to the focus of the BCE loss on a larger quantity of negative samples, namely the background, while neglecting the smaller proportion of positive samples, namely the foreground, this can lead to poorer predictions for minority samples. Therefore, we use the DICE loss function. The DICE loss function can be represented as:

$$DiceLoss = 1 - \frac{2|X \cap Y|}{|X| + |Y|} \tag{1}$$

where X represents the predicted matrix, and Y represents the label matrix.

The loss function for thresholding adopts the Smooth L1 function, which can be represented as:

$$L(\mu, \tilde{\mu}) = smooth_{L1}(\tilde{\mu} - \mu) \tag{2}$$

$$smooth_{L1}(x) = \begin{cases} 0.5x^2 & \text{if } |x| < 1, \\ |x| - 0.5 & \text{otherwise,} \end{cases} \tag{3}$$

Where μ represents the predicted threshold, and $\tilde{\mu}$ represents the true threshold.

4.2. Dataset

Our experimental dataset adopts the Crack500 dataset established by Yang et al. [9, 14] for crack detection. Due to the large size of the original images, which is not conducive to training, the images in the dataset are divided into several smaller images of size . After division, there are 1896 images in the training set, 348 images in the validation set, and 1124 images in the test set. We performed data augmentation on the training set by rotating the images 90°, 180°, and 270°, as well as horizontally flipping them, to combat overfitting.

4.3. Model deployment details

Our experiments were conducted on the PyTorch platform, with all experiments based on a single NVIDIA RTX 3090 GPU. In the Crack500 dataset, the batch size for all models was set to 8, and they were trained for 80,000 iterations. We chose the AdamW optimizer with a learning rate of 0.00001 and weight decay of 0.0002. The learning rate was scheduled using a polynomial decay strategy with a momentum of 0.9. We resized all images to a uniform resolution of 224×224.

4.4. Evaluation metrics

We use the following four metrics to measure the performance of the model: precision, recall, F1 score, and Intersection over Union (IoU). Precision refers to the proportion of correctly predicted positive samples among all samples predicted as positive. It can be expressed by the following formula:

$$\text{Precision} = \frac{TP}{TP + FP} \tag{4}$$

ISSN: 2472-3703 DOI: 10.6911/WSRJ.202405_10(5).0007

Recall, also known as sensitivity or true positive rate, measures the proportion of correctly predicted positive samples among all actual positive samples. The formula for recall is as follows:

$$\text{Recall} = \frac{TP}{TP + FN} \tag{5}$$

The F1 score is used to balance the trade-off between precision and recall by computing their harmonic mean. The formula for the F1 score is as follows:

$$\text{F1 Score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision+Recall}} \tag{6}$$

The Intersection over Union (IoU) represents the rate of overlap between the predicted results and the ground truth labels. The formula for IoU is as follows:

$$\text{IoU} = \frac{TP}{TP + FP + FN} \tag{7}$$

4.5. Comparison and analysis of experimental results

Firstly, we trained our model SFR-net on the Crack500 dataset as mentioned above. Subsequently, we employed the method of adaptive thresholding under identical conditions to deploy and train, aiming to observe the effectiveness of this approach. The experimental results are shown in Table 4.1.

Table 1. Comparison experiment between SFR-net without adaptive methods and SFR-net with adaptive methods

Method	Precision	Recall	F1 Score	IoU
SFR-net	**0.686**	0.749	0.692	0.551
SFR-net-AT	0.684	**0.754**	**0.695**	**0.552**

From the table, it can be observed that the recall rate increased by 0.5%, the F1 score improved by 0.3%, and the intersection over union (IoU) increased by 0.1%. Due to the trade-off between precision and recall, the increase in recall resulting from the adaptive thresholding method led to a decrease in precision. Overall, with the application of the adaptive thresholding method, both the F1 score and IoU showed improvement, indicating that appropriate thresholds can enhance the stability of prediction results.

We compared our SFR-net model and the SFR-net model with the added adaptive thresholding method against outstanding segmentation models including Unet, Swin-Unet, HED, RCF, BDCN, and DeepCrack. The experimental results are shown in Table 2. The penultimate and last columns in the table show the crack prediction results for our SFR-net model without the addition versus with the addition of the adaptive thresholding method.

ISSN: 2472-3703 DOI: 10.6911/WSRJ.202405_10(5).0007

Table 2. Comparison experiment results of various methods

Method	Precision	Recall	F1 Score	IoU
Unet	0.675	0.749	0.687	0.546
Swin-Unet	0.620	0.716	0.634	0.487
HED	0.648	0.752	0.668	0.526
RCF	0.664	0.748	0.679	0.537
BDCN	0.648	**0.766**	0.678	0.535
DeepCrack	0.635	0.765	0.669	0.527
SFR-net	**0.686**	0.749	0.692	0.551
SFR-net-AT	0.684	0.754	**0.695**	**0.552**

From the table, it can be observed that our method, SFR-net, along with SFR-net-AT, achieved the best performance in terms of F1 score, and intersection over union (IoU). In the recall metric, BDCN performed the best. However, the higher recall rate implies that BDCN misclassifies many backgrounds as cracks. We display the results in Figure 5.

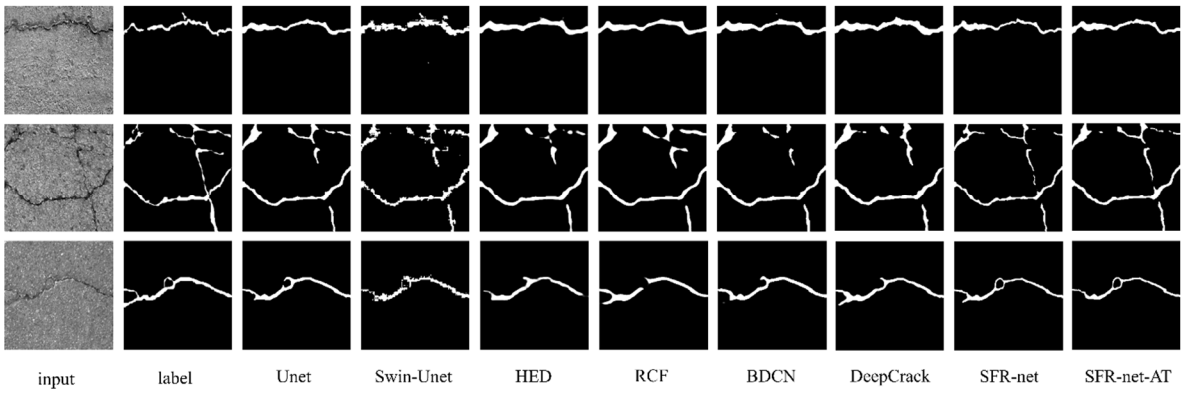

Figure 5. Comparison experiment results of various methods

From the figure 5, we can observe that although Unet segmentation results are continuous, there are many cases where the background is misclassified as foreground. Swin-Unet performs the worst among other models, possibly due to both its encoder and decoder being based on Transformer and the relatively small dataset, leading to training difficulties. The segmentation results of HED are rougher compared to Unet, with discontinuous detection of cracks. RCF and BDCN have similar segmentation results, lacking in the ability to detect details in crack areas. Our SFR-net model generates cracks that are sharper and more accurate compared to other methods. However, after incorporating the adaptive thresholding method, our SFR-net model appears to be more conservative in predicting smaller targets while capturing relatively accurate fine cracks.

5. CONCLUTION

To address the relatively uniform data pattern in the crack detection dataset, we employed a method of dimension expansion and modeling by utilizing features from all previous stages after downsampling at each stage, thereby avoiding overfitting issues. At each stage, we used STDC blocks to reduce redundant computational overhead caused by standard convolutional layers. Furthermore, to tackle the instability issue commonly encountered with fixed threshold binarization methods in segmentation image predictions, we proposed an adaptive thresholding approach to enhance the stability of the model's predictions.

ISSN: 2472-3703 DOI: 10.6911/WSRJ.202405_10(5).0007

ACKNOWLEDGEMENTS

This paper was supported by the Longmen laboratory for Exploratory Research Project. [No.MQYTSKT034]

REFERENCES

[1] RONNEBERGER O, FISCHER P, BROX T: U-net: Convolutional networks for biomedical image segmentation, Medical image computing and computer-assisted intervention, p.234-41.

[2] CAO H, WANG Y, CHEN J, et al: Swin-unet: Unet-like pure transformer for medical image segmentation, European conference on computer vision, p.205-18.

[3] LIU Z, LIN Y, CAO Y, et al: Swin transformer: Hierarchical vision transformer using shifted windows, Proceedings of the IEEE/CVF international conference on computer vision, p.10012-22.

[4] XIE S, TU Z: Holistically-nested edge detection, Proceedings of the IEEE international conference on computer vision, p.1395-403.

[5] LONG J, SHELHAMER E, DARRELL T: Fully convolutional networks for semantic segmentation, Proceedings of the IEEE conference on computer vision and pattern recognition, p.3431-40.

[6] LIU Y, CHENG M-M, HU X, et al: Richer convolutional features for edge detection, Proceedings of the IEEE conference on computer vision and pattern recognition, p.3000-9.

[7] HE J, ZHANG S, YANG M, et al: Bi-directional cascade network for perceptual edge detection, Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, p.3828-37.

[8] HE K, ZHANG X, REN S, et al: Deep residual learning for image recognition, Proceedings of the IEEE conference on computer vision and pattern recognition, p.770-8.

[9] YANG F, ZHANG L, YU S, et al: Feature pyramid and hierarchical boosting network for pavement crack detection. IEEE Transactions on Intelligent Transportation Systems, Vol.21(2019) No.4, p. 1525-35.

[10] LIU Y, YAO J, LU X, et al: DeepCrack: A deep hierarchical feature learning architecture for crack segmentation. Neurocomputing, Vol.338(2019), p. 139-53.

[11] KRIZHEVSKY A, SUTSKEVER I, HINTON G E: Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems, Vol.25(2012).

[12] FAN M, LAI S, HUANG J, et al: Rethinking bisenet for real-time semantic segmentation, Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, p.9716-25.

[13] HUANG G, LIU Z, VAN DER MAATEN L, et al: Densely connected convolutional networks, Proceedings of the IEEE conference on computer vision and pattern recognition, p.4700-8.

[14] ZHANG L, YANG F, ZHANG Y D, et al: Road crack detection using deep convolutional neural network, 2016 IEEE international conference on image processing (ICIP), p.3708-12.