Research Article

An Improved Nondestructive Semantic Segmentation Method for Concrete Dam Surface Crack Images with High Resolution

Jun Zhang and Jia Zhang

1State Grid Hunan Electric Power Company Limited Research Institute, Changsha, Hunan, China
2Changsha Jinfengtan High-Tech Company Limited, Changsha, Hunan, China

Correspondence should be addressed to Jun Zhang; sghnzj1988@163.com

Received 8 June 2020; Revised 16 September 2020; Accepted 28 October 2020; Published 17 November 2020

Academic Editor: Yu-Sheng Shen

To nondestructive semantic segment the crack pixels in the image with high resolution, previous methods often use sliding window and the crack patches to train the FCNs, and then use the trained FCNs for crack recognition. However, the FCNs will produce a higher proportion of false crack predictions with messy distributions in the high-resolution image. A CNN-to-FCN method is proposed to solve this problem. The CNN is trained by all the patches for large-scale crack and background recognition, and the screened crack predictions are then segmented by the FCN. A real-world concrete dam surface crack image database is firstly established to verify the improved method. The results indicated that (1) the improved method can extremely avoid the higher proportion of false crack predictions and their messy distributions in the high-resolution image through the full utilization of background patches and large-scale background recognition; (2) the ResNetv2 backbone and DeepLabv3 architecture recommended by the improved method can be further modified by reducing the bottleneck channels and adding a DUC module to achieve better performance; (3) the improved method can also reduce the prediction time when the image has low proportion of crack patches, which becomes more practicable for the engineering applications.

1. Introduction

Dam safety monitoring is one of the most important tasks in reservoir project management. The “monitoring” includes not only the instrument observation and analysis of the fixed measuring points [1, 2] but also the visual inspection and instrument exploration on the dam. Through monitoring, the abnormal state of the dam can be discovered and dealt with in time to avoid serious consequences.

Crack detection plays a vital role in dam safety monitoring. The crack occurs on the concrete dam since the construction stage. Some cracks are beneficial, such as the artificial crack which is used to prevent the hydration heat temperature crack. Some cracks are harmful, which can decline the dam concrete strength or format leakage passage. Manual inspection is a traditional way to detect cracks. However, the vast space of the dam will limit the inspection scope and lead to time-consuming and inefficient.

Moreover, the crack detail and developing process are difficult to track and evaluate. Automated crack detection with computer vision method is an effective way to replace manual inspection. In the past, researchers often adopted image processing, edge detection, and morphological operations to detect crack [3–9]. For dam concrete, Fan Xinnan et al. [10] uses local-global clustering analysis and the image processing method to identify the visual detection of underwater dam surface cracks. These methods have been proved to be useful in some situations, such as bright characteristics, high contrast, less noise, and strong continuity of crack. However, they may not be applicable to the dam concrete surface with diverse background texture, various types of noise, and irregular crack distribution (Figure 1(b)).

In recent years, a more powerful image recognition and detection technology based on the convolutional neural network (CNN) has been proposed by Lecun et al. [11], and
Figure 1: Continued.
then shows excellent potential in dealing with these complex scenarios above. To further classify the image in pixel level, Long et al. [12] proposed a specific CNN named the fully convolutional network (FCN). The basic FCN model is then developed by some optimized backbones and algorithms, including decoder variants and integrating context knowledge, such as the SG-net, U-net, and DeepLab [13]. Meanwhile, the crack detection techniques based on CNN or FCN have already applied in various civil structures such as the bridges, roads, buildings, highways, and tunnel, and proved to be a more efficient way than the previous methods.

For classification tasks, Dorafshan et al. [14] compare the performance of common edge detectors and deep convolutional neural networks (AlexNet DCNN) for image-based crack detection in concrete panels. Hongyan Xu et al. [15] use the techniques of atrous convolution, atrous spatial pyramid pooling (ASPP) module, and depthwise separable convolution to improve the traditional CNN, and the bridge

![Figure 1: The crack patches from high-resolution images of dam surface: (a) simple cracks; (b) intricate cracks.](image-url)
crack dataset is established to verify the proposed model. Umme Hafla Billah et al. [16] propose the ResNet to detect cracks of various roads, highways, and bridge decks at different times of the day and for different light orientation conditions. With the ResNet-based classifier, Chen Feng et al. [17] use AL to reduce the number of civil infrastructure surface images required for annotation and thus reduce the effort and cost of annotation by domain experts. Meanwhile, the backbones of VGG and GoogLeNet are also used to detect concrete crack [18–20].

For pixel-level classification tasks, Jianming Zhang et al. [21] use FCN and dilated convolution to detect cracks from various campus buildings. Zhenqing Liu et al. [22] propose U-Net to detect the concrete cracks of campus buildings. Allen Zhang et al. [23] developed a CNN architecture without pooling layers named CrackNet for automated pavement crack detection on 3D asphalt surfaces. Weidong Song et al. [24] used multisidedilated convolutional module for automated pavement crack detection. Yupeng Ren et al. [25] used techniques of dilated convolution, spatial pyramid pooling, skip connections, and an optimized loss function for concrete crack detection in tunnels. For backbones, the VGG, DenseNet, and ResNet are frequently used to detect pixel-wise concrete crack [26–28].

In this study, we establish a crack image database of dam concrete surface for the first time. A high-resolution camera acquires these images. However, the high-resolution image is inappropriate as the FCN input directly due to the limited calculating conditions such as GPU memory. To pursue the speed, a destructive way is to shrink the image size, but it will reduce the accuracy, especially for the small pixel width of cracks, as shown in Figure 1(b). A nondestructive solution is to apply the sliding window to divide the high-resolution image into small patches to fit the input size of FCN. However, previous FCNs only adopt the crack patches to alleviate the category imbalance and extensive computation, which has a small percentage of all the patches. Frustratingly, it will produce a higher proportion of false crack predictions with messy distributions in the high-resolution image prediction.

Therefore, we provide a CNN-to-FCN nondestructive semantic segmentation method to solve the problem above. The method firstly adopts sliding window and CNN to locate the crack approximately and then uses FCN to segment the crack predictions. CNN and FCN are trained separately by different datasets. The CNN trains the crack and background patches to ensure the background information is fully utilized, while the FCN trains only the crack patches.

The contributions of this paper mainly include five aspects, as follows:

1. Propose a CNN-to-FCN nondestructive semantic segmentation method for dam crack detection
2. Apply the ResNetv2 backbone and the DeepLabv3 architecture for dam crack semantic segmentation
3. Crop the ResNetv2 backbone bottleneck channels to accelerate networks and improve their performance
4. Propose the dense upsampling convolution (DUC) in the upsampling stage of DeepLabv3
5. Establish a concrete dam crack dataset (Dam-CrackDataset) for the first time

The content of this paper is described as follows: Section 2 demonstrates the basic CNN and FCN, the mainstream backbone ResNet and the DeepLab architecture, and the DUC module. Section 3 introduces our improved nondestructive semantic segmentation method and improved model for concrete dam crack detection. The Dam-CrackDataset is established to verify the performance of different methods and models in Section 4. Finally, Section 5 concludes this article.

2. CNN and FCN

2.1. Basic CNN and FCN. The basic CNN consists of three parts: the input layer, convolution layers, and pooling layers, and a fully connected multilayer perceptron classifier. Convolution and pooling operations can greatly simplify model complexity and reduce model parameters. The basic CNN is a feature extractor with the advantage that human engineers do not need to design multiple layers of features. Compared to the standard feedforward neural networks, CNN has better learning and adaptive ability due to its unique designs.

The main function of the convolution layer is to convolve the convolution kernel with the input data of the upper layer, in which the convolution kernel in the convolutional layer is applied to feature extraction. It can also reduce the connection between different layers to prevent overfitting and too many parameters. The next layer output can be expressed as follows:

$$y(i, j) = \sum_{m=1}^{M} \sum_{n=1}^{N} x(i + m - 1, j + n - 1)k(m, n),$$

where $y(i, j)$ is the element of the output matrix $y$ with order $p \times q$; $x(i + m - 1, j + n - 1)$ is the element of the input matrix $x$ with order $p \times q$; and $k(m, n)$ is the element of the convolution kernel $k$ with order $M \times N$.

The pooling layer divides the input data into multiple nonoverlapping regions, and take the maximum value (maximum pooling) or average value (average pooling) of each region. It can eliminate noncritical feature samples, thus improving the training efficiency and estimation accuracy, and the pooling formula is

$$y' = \max_{R} (y) \text{ or } \text{mean}_{R}(y),$$

where $R$ is the pooling region.

The fully connected layer expands the two-dimensional data passing through the convolution layer and pooling layer into one-dimensional data. It can be expressed as follows:

$$o = f(w^Tc + b),$$
where \( \mathbf{w} \) is the connection weight vector; \( \mathbf{c} \) is the expanded one-dimensional data; \( \mathbf{b} \) is the bias; \( \mathbf{o} \) is the output; and \( f \) is an activation function which can enhance the network’s nonlinearity. For example, rectified linear unit (ReLU) is a popular activation function which can activate the neurons of the neural network sparsely and can be expressed as follows:

\[
\text{ReLU}(y(i, j)) = \begin{cases} 
  y(i, j), & y(i, j) > 0, \\
  0, & y(i, j) \leq 0.
\end{cases}
\]

The difference between FCN and CNN is that FCN converts the fully connected layer of CNN to the convolution layer. The converted convolution layer adopts upsampling to make the output to be the same size as the input image, which can achieve pixel-level prediction and retaining the original input space information of images.

2.2. ResNet Backbone. The ResNet backbone \([29, 30]\) is put forward to solve the problem that deeper networks are more challenging to train. This kind of network is equivalent to adding a new channel of the input so that the input can reach the output directly. Then, the optimization objective changes from the original output \( H(x) \) to the residual between the \( H(x) \) and input \( x \). The ResNet backbone shows excellent characteristics in precision and convergence by using an extremely deep network. The ResNet series have two versions named ResNetv1 and ResNetv2 (Figure 2). ResNetv2 adopts the identity after-addition activation to make information propagation smoother. The asymmetric after-addition activation is equivalent to constructing a preactivation residual unit. In this study, the ResNetv2 model is adopted.

2.3. DeepLab Architecture. DeepLab series \([31–34]\) are semantic segmentation deep learning models developed from FCN model. There are two technical hurdles in the application of basic FCN model: downsampling and spatial invariance. The downsampling method will reduce the resolution, especially at the high-level layers. The spatial invariance means that obtaining object-centric decisions from a classifier requires invariance to spatial transformations, inherently limiting the spatial accuracy. DeepLabv1 employs the atrous convolution algorithm and conditional random field (CRF) to address the downsampling and the spatial invariance, respectively. DeepLabv2 uses atrous spatial pyramid pooling (ASPP) to robustly segment objects at multiple scales. To encode multiscale information, DeepLabv3 proposes a cascaded module and an improved ASPP module. The cascaded module gradually doubles the atrous rates, and the improved ASPP module augmented with image-level features probes the features with filters at multiple sampling rates and effective field-of-views. DeepLabv3+ extends DeepLabv3 by adding a simple yet effective decoder module to refine the segmentation results, especially along object boundaries. For backbone, DeepLabv1 is constructed by VGG-16. DeepLabv2 and DeepLabv3 use the ResNet. DeepLabv3+ adopts the ResNet and Xception.

3. Improved Method and Model

3.1. CNN-to-FCN Method. The flowchart of our CNN-to-FCN nondestructive semantic segmentation method is shown in Figure 3 and demonstrated as follows:

(i) Step 1: get high-resolution images of dam concrete surface. The images are then manual labelled by pixel level.

(ii) Step 2: use the sliding window to get the image patches and their annotations. Meanwhile, the patches can be augmented, but the annotations should be nondestructive for FCN. For CNN, annotation of each patch should be further used to generate a new global crack dichotomy annotation. For FCN, the pixel-level annotations are used directly.

(iii) Step 3: split the patches into training, validation, and test patches. The training patches are applied to train the model, while the validation patches are to choose the best parameters. The test patches are used to estimate the performance of the model.

(iv) Step 4: train and optimize CNN using the training and validation patches, including crack and background patches.

(v) Step 5: train and optimize FCN using the training and validation patches, including only the crack patches.

(vi) Step 6: predict the test patches and judge whether each patch is crack by using the trained CNN.

(vii) Step 7: classify the crack predictions according to Step 6 on pixel level by using the trained FCN.

Significantly, previous methods only train and test the crack patches, while the improved method trains the crack and background patches to fully utilize the background information. The large-scale background recognition by the improved method can make us focus on the fewer crack patches of the high-resolution image, thus avoiding the messy distributions of false crack predictions.

3.2. Improved Model. In this study, we use the ResNetv2-50 backbone and integrate a dense upsampling convolution (DUC Figure 4) module proposed by Wang et al. \([35]\) into DeepLabv3 architecture (Figure 5) because the DUC operation can be naturally integrated into the DeepLabv3, makes the whole encoding and decoding process end-to-end trainable, and only increases the FLOPs by a small amount.

At the stage of upsampling, the DeepLab series models often used the bilinear interpolation method. As most of the width pixels of cracks are fewer than 4 pixels in our DamCrackDatabase, the bilinear interpolation will more likely miss the fine-detailed information of crack when the downsampling rate is less than 1/4. DUC is more effective than the bilinear interpolation method, and the deconvolution method used by the basic FCN model. The key idea of DUC is to transform the whole label map into a smaller label map with multiple channels. At the upsampling stage, DUC
only needs to reshape the output feature map into the whole label map.

ResNet is invented to deal with the large datasets, such as the ImageNet [36]. However, our DamCrackDatabase is a relatively smaller dataset which may not need many parameters to learn. Therefore, we cut the bottleneck channels of ResNetv2-50 in half (named ResNetv2-50s) to reduce the learning parameters and accelerate the computing speed. As shown in Table 1, the FLOPs of ResNetv2-50 and DeepLabv3-DUC are shortened about three times as

Figure 2: ResNetv1 and ResNetv2 units. The units consist of the same components, while the orders are different. (a) ResNetv1. (b) ResNetv2.

Figure 3: The CNN-to-FCN method for dam crack detection.
Figure 4: Dense upsampling convolution (DUC) module.

Figure 5: Illustration of the DeepLabv3 with the DUC layer. DUC is applied on top of the network for decoding purpose.

Table 1: Different backbones and architectures and their FLOPs.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>conv1</td>
<td>112 × 112</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>conv2_x</td>
<td>56 × 56</td>
<td>1 × 1, 64</td>
<td>3 × 3, 64</td>
<td>1 × 1, 128</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 × 3, 64</td>
<td>3 × 3, 128</td>
<td>3 × 3, 256</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 × 1, 128</td>
<td>1 × 1, 128</td>
<td>1 × 1, 256</td>
<td></td>
</tr>
<tr>
<td>conv3_x</td>
<td>28 × 28</td>
<td>1 × 1, 128</td>
<td>3 × 3, 128</td>
<td>1 × 1, 256</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 × 3, 128</td>
<td>3 × 3, 256</td>
<td>3 × 3, 128</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 × 1, 512</td>
<td>1 × 1, 128</td>
<td>1 × 1, 256</td>
<td></td>
</tr>
<tr>
<td>conv4_x</td>
<td>14 × 14</td>
<td>1 × 1, 256</td>
<td>3 × 3, 128</td>
<td>1 × 1, 256</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 × 3, 256</td>
<td>3 × 3, 256</td>
<td>3 × 3, 128</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 × 1, 1024</td>
<td>1 × 1, 128</td>
<td>1 × 1, 256</td>
<td></td>
</tr>
<tr>
<td>conv5_x</td>
<td>7 × 7</td>
<td>1 × 1, 512</td>
<td>3 × 3, 128</td>
<td>1 × 1, 256</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 × 3, 256</td>
<td>3 × 3, 128</td>
<td>1 × 1, 256</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 × 1, 2048</td>
<td>1 × 1, 128</td>
<td>1 × 1, 256</td>
<td></td>
</tr>
<tr>
<td>conv5_d</td>
<td>14 × 14</td>
<td>1 × 1, 512</td>
<td>3 × 3, 128</td>
<td>1 × 1, 256</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 × 3, 256</td>
<td>3 × 3, 256</td>
<td>1 × 1, 256</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 × 1, 2048</td>
<td>1 × 1, 1024</td>
<td>1 × 1, 256</td>
<td></td>
</tr>
<tr>
<td>conv6_d</td>
<td>14 × 14</td>
<td>[1 × 1, 256]</td>
<td>[3 × 3, 256]</td>
<td>[1 × 1, 256]</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>[3 × 3, 256]</td>
<td>[3 × 3, 256]</td>
<td>[1 × 1, 256]</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>[1 × 1, 2048]</td>
<td>[1 × 1, 1024]</td>
<td>[1 × 1, 256]</td>
<td></td>
</tr>
</tbody>
</table>
before when using the ResNetv2-50s backbone. Furthermore, our experiments will demonstrate that the improved backbone and architecture can achieve better performance.

We have also considered the U-Net and DeepLabv3+ architectures. The U-Net is proved to be an effective way to detect concrete crack [22]. However, its training set is less than $1 \times 10^5$, and the VGG-16 backbone has the FLOPs of $1.54 \times 10^9$, which is about 15 times more than our proposed model’s FLOPs ($1.04 \times 10^9$) by adopting the same input size. That is time-consuming and unacceptable to train our data over $5 \times 10^4$. The DeepLabv3+ with ResNetv2-50s backbone is used for comparison in Table 2.

### 4. Experiments and Results

We collected 344 images of cracks from a concrete arch dam surface at different elevations. These images with the resolution of $3456 \times 4608$ are labelled on pixel level by domain experts and then cut into $224 \times 224$ resolution patches using the sliding window with a stride of 112. We assign each patch a crack label if its centering $200 \times 200$ region contains at least one pixel. Otherwise, patches are labelled as background. Meanwhile, the crack patches are augmented by rotating the angle to 90, 180, and 270 degrees. Finally, we get 484092 patches including 402796 background patches and 81296 crack patches with the ratio about 5:1. For CNN, all the patches are divided into training, validation, and test data with the ratio of 4:1:1, respectively. For FCN, we only use the crack patches in the training, validation, and test data, and these data are also set as the ratio of 4:1:1, respectively.

For the two networks, the initial learning rates are both set as $0.001$ and the Adam updater which designs an independent adaptive learning rate for different parameters by calculating the first-order moment estimation and the second-order moment estimation of the gradient is used. It is very robust and usually converges quickly and gives pretty good performance and the hyperparameters $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\varepsilon = 10e−8$ [37]. The weight decay to use for regularizing the model is set $10e−5$ to prevent overfitting. The batch normalization (BN) parameter is set $\varepsilon = 10e−6$ [38], which is a small constant to prevent division by zero when normalizing activations by their variance in BN.

All these algorithms are implemented using TensorFlow (v1.14.1), and used in their original version and performed via a laptop (CPU: Intel i9 9900K @ 3.6GHz, RAM: 32GB, GPU: Nvidia GeForce RTX 2080). The class-balanced cross-entropy loss function is used in equation (5) to alleviate the category imbalance:

$$
\text{Loss} = -\alpha y \log y' - (1 - \alpha) (1 - y) \log (1 - y'),
$$

where $\alpha = (Y_c/Y)$ and $1 - \alpha = (Y_b/Y)$; $Y_c$ and $Y_b$ denote the crack and background label sets, respectively; $y$ denotes the positive and negative label; and $y'$ denotes the output probability.

The performance indexes are as follows:

$$
\text{Precision} = \frac{TP}{TP + FP},
\text{Recall} = \frac{TP}{TP + FN},
\text{F-score} = \frac{(1 + \beta^2) \cdot \text{Recall} \cdot \text{Precision}}{\text{Recall} + \beta^2 \cdot \text{Precision}},
$$

where $TP$ denotes the true positive predictions; $FP$ means the false positive predictions; $FN$ denotes the false negative predictions; $F$ score measures the weighted harmonic mean of Precision and Recall; and $\beta$ is a weighted factor.

When focusing on the Recall, the $\beta$ is set larger than 1, while focusing on the Precision, the $\beta$ should be smaller than 1. We increase the Recall to obtain the TP predictions from the true positive labels as many as possible at the CNN stage. So, the $\beta$ is set as 2. For CNN, we use the indexes to value all the patches. For FCN, we use the above indexes to estimate each patch, and their mean indexes to value all the patches. If each patch’s Precision or Recall is

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>conv7_d</td>
<td>14×14</td>
<td>[1×1, 256]</td>
<td>[1×1, 512]</td>
<td>[1×1, 256]</td>
<td>[1×1, 512]</td>
</tr>
<tr>
<td>DUC</td>
<td>224×224</td>
<td>Reshape, 14×14×512 to 224×224×2, softmax</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FLOPs</td>
<td>3.87×10^9</td>
<td>1.04×10^9</td>
<td>8.72×10^9</td>
<td>3.08×10^9</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Model name</th>
<th>FCN architecture</th>
<th>Backbone</th>
<th>Bath size</th>
<th>Max mIoU</th>
<th>Total epoch</th>
<th>Relative/absolute training time</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>DeepLabv3</td>
<td>ResNetv2-50s</td>
<td>30</td>
<td>0.27</td>
<td>400</td>
<td>1.00x/59h</td>
</tr>
<tr>
<td>B</td>
<td>DeepLabv3-DUC</td>
<td>ResNetv2-50s</td>
<td>30</td>
<td>0.57 + 111.1%</td>
<td>400</td>
<td>1.04x/61h</td>
</tr>
<tr>
<td>C</td>
<td>DeepLabv3+</td>
<td>ResNetv2-50s</td>
<td>30</td>
<td>0.56 + 107.4%</td>
<td>400</td>
<td>1.73x/102h</td>
</tr>
</tbody>
</table>

Table 1: Continued.

Table 2: mIoU index of FCNs on validation crack set.
Table 3: $F$ index of different CNN backbones on validation set.

<table>
<thead>
<tr>
<th>Model name</th>
<th>CNN backbone</th>
<th>Bath size</th>
<th>Max $F$</th>
<th>Total epoch</th>
<th>Relative/absolute training time</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>ResNetv2-50</td>
<td>50</td>
<td>0.87</td>
<td>60</td>
<td>1.00x/49 h</td>
</tr>
<tr>
<td>II</td>
<td>ResNetv2-50s</td>
<td>50</td>
<td>0.88 + 1.1%</td>
<td>60</td>
<td>0.43x/21 h</td>
</tr>
<tr>
<td>III</td>
<td>ResNetv2-50s</td>
<td>50</td>
<td>0.86 – 1.1%</td>
<td>60</td>
<td>0.29x/14 h</td>
</tr>
</tbody>
</table>

Figure 6: CNN loss and $F$ during training and validation: (a) loss; (b) $F$.

Figure 7: FCN loss and mIoU during training and validation: (a) loss; (b) mIoU.
Figure 8: Different models’ prediction results on test crack set: (a) original patch, (b) labelled patch, (c) model A, (d) model B, and (e) model C.

Table 4: Indexes of ONLY-FCN and CNN-FCN by different thresholds on test set.

<table>
<thead>
<tr>
<th>Method name</th>
<th>Model II threshold</th>
<th>Model B threshold</th>
<th>TP</th>
<th>FP</th>
<th>TN</th>
<th>FN</th>
<th>mR</th>
<th>mP</th>
<th>mIoU</th>
<th>Relative/absolute prediction time per patch</th>
</tr>
</thead>
<tbody>
<tr>
<td>ONLY-FCN</td>
<td>0.999</td>
<td>—</td>
<td>29227</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>0.188</td>
<td>0.865</td>
<td>0.182</td>
<td>1.00x/7.9 ms</td>
</tr>
<tr>
<td></td>
<td>0.9</td>
<td>—</td>
<td>40719</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>0.408</td>
<td>0.806</td>
<td>0.370</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.9</td>
<td>0.99</td>
<td>54502</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>0.645</td>
<td>0.729</td>
<td>0.514</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.5</td>
<td>0.9</td>
<td>69316</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>0.795</td>
<td>0.648</td>
<td>0.547</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.001</td>
<td>0.5</td>
<td>13496</td>
<td>56779</td>
<td>10364</td>
<td>43</td>
<td>0.795</td>
<td>0.648</td>
<td>0.547</td>
<td>1.42x/11.2 ms</td>
</tr>
<tr>
<td></td>
<td>0.01</td>
<td>0.5</td>
<td>13410</td>
<td>41299</td>
<td>25844</td>
<td>129</td>
<td>0.795</td>
<td>0.648</td>
<td>0.548</td>
<td>1.22x/9.6 ms</td>
</tr>
<tr>
<td></td>
<td>0.1</td>
<td>0.5</td>
<td>12885</td>
<td>10988</td>
<td>56155</td>
<td>654</td>
<td>0.798</td>
<td>0.649</td>
<td>0.550</td>
<td>0.84x/6.6 ms</td>
</tr>
<tr>
<td></td>
<td>0.5</td>
<td>0.5</td>
<td>12151</td>
<td>2851</td>
<td>64292</td>
<td>1388</td>
<td>0.801</td>
<td>0.650</td>
<td>0.552</td>
<td>0.73x/5.8 ms</td>
</tr>
<tr>
<td>CNN</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>0.53x/4.2 ms</td>
</tr>
</tbody>
</table>

TN denotes the true background predictions.
Figure 9: $R_{\text{time}}$ and $D_{\text{FP}}$ of all the image: (a) $R_{\text{time}}$; (b) $D_{\text{FP}}$.

Figure 10: Continued.
incomputable, then the patch is not taken into account, respectively.

For FCN, we also add the common index as follows:

$$mIoU = \frac{1}{M} \sum_{m=1}^{M} \frac{TP}{TP + FP + FN}$$

where $M$ denotes the number of patches which are computable.

The CNNs are trained for 60 epochs. Observing the curves in Figure 6(a), the loss decreases gradually on the training set, while the validation set converges after about the 20th epoch. As shown in Table 3 and Figure 6(b), model II changes the backbone of model I to ResNetv2-50 s, and obtain the best $F$ and saves more than half of the training time. Model III cut the bottleneck channels of ResNetv2-50 to a quarter (named ResNetv2-50s) and obtain the worst $F$, which shows that the bottleneck channels cannot be cut too much. Therefore, we finally choose the ResNetv2-50s as the backbone for the next FCNs.

The FCNs are trained for 400 epochs because we found that model B and model C need more time to obtain a better $mIoU$ (Table 2 and Figure 7(b)). As can be seen from Figure 7, the loss and $mIoU$ of model A converge the most quickly than model B and model C on both the training and validation sets. Model B substantial increases the $mIoU$ more than 110% by adding the DUC module, and shows the best prediction performance on the test crack set (Table 2 and Figure 8). Model C is the most time-consuming over 100 hours, and its $mIoU$ is worse than model B.

Significantly, the loss in model B has the poorest convergence performance than model A and model C after about the 250th epoch, which shows that the loss function may not be the best training way to achieve a better $mIoU$.

Thresholds are vital parameters to tune the models’ indexes. To compare the distributions of false crack (FP) predictions by using the methods ONLY-FCN and CNN-FCN (our improved method), we assign each patch predicted by model B and model II FP if its region contains at least one FP pixel. As shown in Table 4, the threshold of model B can be applied to reduce the FP and increase the mean Precision (mP) for method ONLY-FCN. However, it will reduce the mean Recall (mR) and $mIoU$ at the same time. The threshold of model II can be applied to reduce the FP more efficiently than the threshold of model B. As the FN growth rate is much less than the rate of FP decline, and the mR and $mIoU$ rise by the increase of the model II threshold without loss of mean Precision (mP). Therefore, we can coordinate the FP and FN by tuning the threshold of model II to adapt to different requirement on recognition. Meanwhile, we use the index as follows to evaluate model II FP distribution:

$$DFP = \frac{FP}{TP + FP + TN + FN}$$

For prediction time-consuming, we can conclude the function as follows:
\[ T_{\text{FCN}} = t_{\text{FCN}}N, \]
\[ T_{\text{CNN-FCN}} = t_{\text{CNN}} + t_{\text{FCN}}N_{cp}, \]
\[ R_{\text{time}} = \frac{T_{\text{CNN-FCN}}}{T_{\text{FCN}}} = \frac{t_{\text{CNN}} + \frac{N_{cp}}{N}}{t_{\text{FCN}}}, \]

where \( T_{\text{CNN-FCN}} \) and \( T_{\text{FCN}} \) denote the time-consuming on all the patches using the methods CNN-FCN and ONLY-FCN, respectively; \( t_{\text{CNN}} \) and \( t_{\text{FCN}} \) denote the time-consuming on each patch using CNN and FCN, respectively; \( N \) represents the number of all the patches; \( N_{cp} \) represents the number of crack predictions from model II; and \( R_{\text{time}} \) denotes the ratio of \( T_{\text{CNN-FCN}} \) to \( T_{\text{FCN}} \).

For our proposed model, the \( \frac{t_{\text{CNN}}}{t_{\text{FCN}}} \) is 0.53 (Table 4), so if the \( \frac{N_{cp}}{N} \) is smaller than 0.47, our CNN-FCN method can reduce the prediction time-consuming. Note that the test set has a low proportion of crack patches (about 15%) and the crack predictions (TP + FP) can be controlled in a close range. As we have augmented the crack patches, for the real-world dam surface, the crack patches will have a lower proportion. Therefore, our improved method will reduce the prediction time, which becomes more practicable for the engineering applications.

The \( R_{\text{time}} \) and \( D_{sp} \) are used to estimate all the 344 images. The thresholds of model II and model B are set 0.1 and 0.5, respectively. As shown in Figures 9 and 10, our proposed method CNN-FCN can reduce the prediction time to 76% as before and extremely eliminate the messy distribution of FP from 84% to 15%.

5. Conclusions

Image with high resolution can acquire more information and ensure clarity when the camera frame is enlarged. It is beneficial for the dam with wide spaces and small width of cracks which are hard to discover.

To nondestructive segment the dam surface crack pixels of the image with high resolution, this study proposed a CNN-to-FCN method. Comparing with the previous FCNs, our method can extremely avoid the higher proportion of false crack predictions and their messy distributions in the high-resolution image through the large-scale background information recognition by CNN. The improved method also makes the computer easy to get the primary information of dam surface crack and achieve better mIoU than previous FCNs. A few true crack patches are missed at the CNN prediction stage, which should be controlled in a reasonable range by tuning the threshold. Meanwhile, the improved method can also reduce the prediction time when the image has a low proportion of crack patches, which becomes more practicable for the engineering applications.

To obtain better results, we also modify the ResNetv2 backbone and DeepLabv3 architecture by reducing the bottleneck channels and adding a DUC module. The results indicated that the modified model is efficient and dramatically increases the mIoU on dam crack recognition.

Note that our proposed method is universal. However, for a deep learning model, both data and model selection affect the final performance. We have augmented the data from the 344 images to improve the deep learning model’s generalization ability. Meanwhile, we proposed an improved model for nondestructive semantic segmentation. However, more data from different dam scenarios will be more representative. We need to collect more representative data to verify and further improve our proposed model in the future.

Finally, the DamCrackDataset is established for the first time and can be found in https://figshare.com/articles/DamCrackDataset_crack_encrypted_zip/12362159 for further study on concrete dam surface crack detection.

Data Availability

The DamCrackDataset used to support the findings of this study is currently under embargo while the research findings are commercialized. Requests for data, 6 months after publication of this article, will be considered by the corresponding author.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the State Grid Hunan Electric Power Company Limited Science Project (no. 5216A518000N). The authors would like to thank Yan Zhaohui, Chen Shiqiao, and Zhang Zhichao of the State Grid Hunan Hydropower Company for providing the verification data.

References

14 Mathematical Problems in Engineering


