Runet: Convolutional Networks for Crack Detection

Lin Hu¹, Fuping Yang¹
¹Chongqing University of Posts and Telecommunications, China
E-mail: s190231137@stu.cqupt.edu.cn

Abstract. In daily life, crack is a common phenomenon. Such as road cracks, stone cracks. These cracks are usually not easy to be quickly and intuitively identified, because these cracks are characterized by poor continuity and low contrast. Among the existing crack detection methods based on deep learning, the model is too large to be directly used in life. We design an end-to-end neural network, its name is Runet, which greatly reduces the parameters of the model by depthwise separable convolution, increases the receptive field of the network by hole convolution. At the same time, we introduce the trainingtime and inferencetime architecture to increase the running speed and precision.

1. Introduction

In work and life, if some cracks are found in time, it can avoid large economic losses and protect people’s life safety. Up to now, the existing defect detection methods can not be widely used in our life. If we can directly use mobile phones to detect cracks, the prospect of large scale application of deep learning for crack detection can be expected.

With the development of technology, deep learning method has been used in crack detection. Among the common semantic segmentation methods, there are FCN[1], SegNet[2], U-Net[3], Attention-Unet[4], Deeplabv2[5], Deeplabv3[6], Deeplabv3+[7], PspNet[8] and so on. The common ground of these methods is the encoder-decoder structure. At present, some network structure in image recognition is also widely used to obtain extract advanced semantics information in encoder path, such as VggNet[9], ResNet[10], DenseNet[11], InceptinoNetV3[12], MobileNetV2[13], Mobil- eNetV3[14] and so on.

Inspired by the methods of semantic segmentation and image classification, we put forward the idea of adding depthwise separable convolution into encoder network, and apply the hole convolution to increase the receptive field, and we are inspired by the RepVgg[15] network, we add the trainingtime and inferencetime architecture to our network, then we construct a new end-to-end neural network for crack detection. Our main contributions are as follows:

– The network makes full use of the information which obtained from the encoder-decoder structure network, and establishes a trainingtime and inferencetime architecture by a structural reparameterization technique.

– Compared with the existing crack detection neural network, the parameters of our network model are greatly reduced.

– On the basis of greatly reducing the number of parameters, our network model can still guarantee good recall and precision.
2. Neural Network

2.1. Network Architecture

The network is based on the U-Net network. We call it Runet. As is shown in the Fig.1, it consist two path, the contracting path and expansive path, which can be called as encoder path and decoder path. We add the principle of RepVGG, which consists of the repeated basic convolution module, the basic module is consists of one $3 \times 3$ convolution and one $1 \times 1$ convolution and one batch normalize. Each convolution followed by a batch normalize. In the contracting path, every two basic module followed by a $2 \times 2$ max pooling operation with stride 2 for downsampling. Every step in the expansive path consists of an upsampling operation. After this operation, there is a concatenation with the correspondingly feature map from the contracting path. Then followed two basic module. At the network, the last convolution operation, produces a 1-channel feature map, which 1 represent the cracks.

As we know, we use upsampling operation at the expansive path, but we can not ignore one thing that the upsampling operation cause a loss of spatial resolution, the result of this is that it will make the bias of boundaries. In order to avoid the lack of the information of edge, we use bilinear interpolation in the expansive path. Simultaneously, we think of every pixel as squares rather than dots, the feature map tensors are aligned through the center of their corner pixels, thus preserving the values at the corner pixels. Through the above measures, we can get sparse feature maps, which can provide more precise location message of region boundaries.

The contracting path is designed to get the features of the input image, in order to get more detailed information, we can learn from receptive filed, which represent a point on the feature map corresponds to the region on the input image. The larger the receptive field, the larger the range of the original image it can touch, which means that it may contain more global and higher semantic features. The smaller the receptive field is, the more local and detailed the features are. For semantic segmentation task, the larger the receptive field, the more important information will not be ignored. So, we used dilated convolution in contracting path to increase the receptive field. At the same time, In order to reduce the parameters of the whole model, we learn from the idea of depthwise separable convolution, when we use it to backbone network, the parameters of the whole model down by $\frac{1}{N} + \frac{1}{D}$ (the $N$ represent the channels of output feature map, and the $D$ represent the size of convolution kernel).

As is shown in the Fig.1, the basic module Rep is consists of two convolution branch and a BN branch, but the module will change when the Runet is used for inference, the change process is shown in Fig.2, we call this process as Re-param, the purpose of Re-param is blend the parameters of $3 \times 3$ convolution and $1 \times 1$ convolution and BN branch, we use $\mu^1$, $\sigma^1$, $\gamma^1$, $\beta^1$ as the accumulated mean, standard deviation and learned scaling factor and bias of the BN branch, $\mu^2$, $\sigma^2$, $\gamma^2$, $\beta^2$ for the BN layer following $3 \times 3$ conv, $\mu^3$, $\sigma^3$, $\gamma^3$, $\beta^3$ for the BN layer following $1 \times 1$ conv. We use $w_i$, $b_i$ as the parameters of a $i \times i$ conv and the bias of each branch. The important step is fuse every BN and its preceding conv layer, the formula is

$$w' = \frac{\gamma_i}{\sigma_i}w, \quad b' = \beta_i - \frac{\mu_i \gamma_i}{\sigma_i}$$

(1)

In BN branch, the formula (1) can also be used because it can be viewed as a $1 \times 1$ conv with an identity matrix as the kernel. By doing this change, there will have one $3 \times 3$ conv, two $1 \times 1$ conv, we use zero to pad $1 \times 1$ conv to $3 \times 3$ conv, then there have three $3 \times 3$ conv, after this, we use formula (1) to fuse parameters of every branch, after doing this, every branch’s parameters can be shown as $\{w', b'\}$, the last step is the $3 \times 3$ conv addition of the three branches.

The reason for separating the training and inference process is that training a multi-branch model avoids the gradient vanishing problem. But the multi-branch architecture are good at training and not good at inference. In order to solve the problem, we can convert the architecture from one to another
via transforming its parameters. By doing this, the body of inference-time Runet only involves simple operation: $3 \times 3$ convolution, ReLu, down-sampling, up-sampling, concat., which make Runet fast on generic computing devices.

**Fig. 1** Runet architecture

**Fig. 2** The changing process of Runet

### 3. Experiments And Results

#### 3.1. Experimental Settings

We build Runet by using Pytorch. In contracting path and expansive path, batch normalization is used after every convolutional layer. The upsampling operation in decoder network is conducted by bilinear interpolation. The optimizer is RMSPro algorithm, which can eliminate the swing during gradient...
descent to accelerate gradient descent. The initial global learning rate is 0.0001, and the parameter of weight-decay is $10^{-8}$ in RMSPro, the parameter of momentum is 0.9 in RMSPro. The size of the input image is $512 \times 512$. The batchsize is 16. The neural network is trained 1000 epochs in total, and we choose the best model throughout the process. Considering that the number of images in the datasets is too small, we adopt data augmentation to expand the datasets. All the experiments are carried out by using a single GeForce Tesla V100, which memory is 32G.

We use two datasets to verify the accuracy and efficiency of the neural network. Among the crack datasets, 70% for training, 10% for val, and the rest for testing.

In order to evaluate the advantages and disadvantages of our neural network, we choose precision and Recall curve, which can be calculated by comparing the detected cracks against the ground truth image. By comparing the area enclosed by the P-R curve and the coordinate axis, we can know which model is better, if the area enclosed is not easy to compare, we can compare the point where the P-R curve intersects $y = x$ curve.

Because the crack have width, so a predict crack pixel is still regarded as true positive if it is no more than 1 pixels away from ground truth image.

In order to fully demonstrate the superiority of our network model in reasoning accuracy and model parameters, we compare the effect with U-Net, Attention U-Net, PspNet and SegNet in the same datasets.

### 3.2. Experimental Comparison

In this part, we compare the result of different networks in two datasets. By comparing the result of the same datasets in different networks, we can know which is the best network. The result is shown in Fig.3 and Fig 4. In this pictures, The abscissa is recall, The ordinate is precision.

![Fig. 3](image_url) the result in the datasets of CrackTree260
As we know, the model size is connected to the speed of network. The smaller the model size is, the faster the model runs, so we can compare the parameters of the four network. The first column is the network model, and the second column is the model size, the input size of all network is $512 \times 512$. as is shown in the Table 1.

From Figure 3 and Figure 4, It can be seen that the effect of Runet is slightly lower than U-Net in cracktree260 datasets and slightly higher than U-Net in stone331 datasets. In these two datasets, U-Net and Runet have the leading effect than other models. At the same time, in terms of model size, it can be seen from the Table 1 that Runet is only about half of unet and about one third of SegNet, so Runet’s model size advantage is far ahead than others. Therefore, Runet achieves the best balance between crack identification effect and model size.

### Tab. 1 model size comparison

| Network          | Model Size |
|------------------|------------|
| SegNet           | 112MB      |
| U-Net            | 65.9MB     |
| Attention U-Net  | 67.2MB     |
| PspNet           | 96MB       |
| Runet            | 33.8MB     |

### 4. Conclusion

In this paper, we propose a new network to detect cracks. We call this new network as Runet, we add the hole convolution, depthwise separable convolution to Runet, we also introduce the training time and inferencetime architecture to Runet. by doing this, we can see the result from the pictures and table, it shows the advantages of Runet.

### References

[1] Long J, Shelhamer E and Darrell T 2015 Fully convolutional networks for semantic
segmentation Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition pp 3431-3440

[2] Badrinarayanan V, Kendall A and Cipolla R 2017 Segnet: A deep convolutional encoder-decoder architecture for image segmentation IEEE Transactions on Pattern Analysis and Machine Intelligence 39(12) pp 2481-2495

[3] Ronneberger O, Fischer P and Brox T 2015 U-net: Convolutional networks for biomedical image segmentation International Conference on Medical Image Computing and Computer-assisted Intervention Springer Cham pp 234-241

[4] Oktay O, Schlemper J, Folgoc L L, Lee M, Heinrich M, Misawa K and Rueckert D 2018 Attention u-net: Learning where to look for the pancreas arXiv preprint arXiv:1804.03999

[5] Chen L C, Papandreou G, Kokkinos I, Murphy K and Yuille A L 2017 Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs IEEE transactions on Pattern Analysis and Machine Intelligence 40(4) pp 834-848

[6] Chen L C, Papandreou G, Schroff F & Adam H 2017 Rethinking atrous convolution for semantic image segmentation arXiv preprint arXiv: 1706.05587

[7] Chen L C, Zhu Y, Papandreou G, Schroff F and Adam H 2018 Encoder-decoder with atrous separable convolution for semantic image segmentation Proceedings of the European Conference on Computer Vision (ECCV) pp 801-818

[8] Zhao H, Shi J, Qi X, Wang X and Jia J 2017 Pyramid scene parsing network Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition pp 2881-2890

[9] Simonyan K & Zisserman A 2014 Very deep convolutional networks for large-scale image recognition arXiv preprint arXiv: 1409.1556

[10] He K, Zhang X, Ren S and Sun J 2016 Deep residual learning for image recognition Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition pp 770-778

[11] Huang G, Liu Z, Van Der Maaten L and Weinberger K Q 2017 Densely connected convolutional networks Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition pp 4700-4708

[12] Szegedy C, Vanhoucke V, Ioffe S, Shlens J and Wojna Z 2016 Rethinking the inception architecture for computer vision Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition pp 2818-2826

[13] Sandler M, Howard A, Zhu M, Zhmoginov A and Chen L C 2018 Mobilenetv2: Inverted residuals and linear bottlenecks Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition pp 4510-4520

[14] Howard A, Sandler M, Chu G, Chen L C, Chen B, Tan M and Adam H 2019 Searching for mobilenetv3 Proceedings of the IEEE/CVF International Conference on Computer Vision pp 1314-1324

[15] Ding X, Zhang X, Ma N, Han J, Ding G and Sun J 2021 Repvgg: Making vgg-style convnets great again Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition pp 13733-13742