Author/s:
Rao, AS;Nguyen, T;Le, ST;Palaniswami, M;Ngo, T

Title:
Attention recurrent residual U-Net for predicting pixel-level crack widths in concrete surfaces

Date:
2022-11-01

Citation:
Rao, A. S., Nguyen, T., Le, S. T., Palaniswami, M. & Ngo, T. (2022). Attention recurrent residual U-Net for predicting pixel-level crack widths in concrete surfaces. Structural Health Monitoring, 21 (6), pp.2732-2749. https://doi.org/10.1177/14759217211068859.

Persistent Link:
https://hdl.handle.net/11343/302061

Attention Recurrent Residual U-Net for Predicting Pixel-level Crack Widths in Concrete Surfaces

Journal Title
XX(X):1–12

DOI: 10.1177/ToBeAssigned
www.sagepub.com/
SAGE

Aravinda S. Rao[1], Tuan Nguyen[2], Son Tay Le [2], Marimuthu Palaniswami[1] and Tuan Ngo[2]

Abstract

Cracks in concrete structures are one of the most important indicators of structural damage, and it is a necessity to detect and measure cracks for ensuring safety and integrity of concrete structures. The widely practised approach in inspecting the structures is by performing visual inspections followed by manual estimation of crack widths. This approach is not only time-consuming, laborious and time-intensive but also prone to subjective errors and inefficient. To address these issues, we propose a novel deep learning framework for detecting cracks and then estimating crack widths in concrete surface images. Our framework handles both small- and large-sized images and provides a prediction of crack width at locations specified by the user. The proposed framework uses Attention Recurrent Residual U-Net (Attention R2U-Net) with Random Forest regressor to predict crack width with the mean prediction error of ± 0.31 mm for crack widths varying from 0 to 8.95 mm and produces the lowest absolute maximum error of 1.3 mm. Our model has a coefficient of determination (R^2) of 0.91, showing a non-linear mapping function with low prediction errors. We compare our model with a combination of four other segmentation models and regression models. Our proposed model has superior performance compared to other models, and one can easily adopt our framework to a variety of Structural Health Monitoring (SHM) applications using Internet of Things (IoT) sensors.

Keywords

Structural Health Monitoring, Automated Assessment, Crack Width Prediction, Deep Learning, Segmentation Models

1 Introduction

Concrete is the world's most used man-made material and is being used to build bridges, builds, dams, roads, pipelines and many other critical infrastructures[1]. The occurrence of concrete cracks shows underlying structural damage and requires continuous monitoring and measurement. The development of these cracks in terms of width and length may severely affect the structural safety, strength and maintenance of these structures. Early detection of cracks, localizing them and quantifying the crack widths play an important role in ensuring sound structural health and durability. For example, as per the current inspection manual regulated by the local road and bridge management agency, the manual specifies that either single or multiple cracks with a width greater than 0.3 mm could affect the structural integrity of concrete structures, and require continuous measuring and monitoring[2]. In particular, the inspection manual[2] also specifies requiring location, extent (*i.e.*, length), width of cracks, and changes in these defects (compared to previous inspections) as key indicators of potential changes in the structural capacity and performance. The location, length and width of these defects also inform condition rating, overall maintenance, and repair information for inspected structures[2]. This work focuses on detecting millimeter-level accuracy of crack widths from concrete structure images. This level of accuracy and measurement of cracks are crucial for the inspection and maintenance of concrete structures. For example, in the inspection and maintenance manual of the local infrastructure agency[2], it is highlighted that those crack locations and crack widths are the key indicators of the performance and capacity of concrete structures.

[1]Department of Electrical and Electronic Engineering, The University of Melbourne, Melbourne, Australia
[2]Department of Infrastructure Engineering, The University of Melbourne, Melbourne, Australia

Corresponding author:
Tuan Ngo, Department of Infrastructure Engineering, The University of Melbourne, Melbourne, Australia.
Email: dtngo@unimelb.edu.au

Traditional structural engineering relies on visual inspection by measuring the crack widths using gauges and then transferring these measurements to the drawings. They often measure the crack widths at pre-determined locations as set by the end-users or decided based on the structure. However, this process of manual measurements is not only labor intensive but also time-consuming and prone to subjective errors[3]. In addition, manual inspection is infeasible at many hazardous locations where the inspection team cannot access the structures[4]. Sensors such as strain gauges[5], vibrating wire strain gauges[6], and distributed fiber optic sensors[7] have been used to assess crack widths. People have also explored used Non-Destructive Testing (NDT) technique such as the ultrasonic Coda Wave Interferometry (CWI) to detect cracks[8]. However, automating the detection of cracks is not workable with these approaches and would require setting up of sensors and devices regularly to measure crack widths. Therefore, it is critically important to use a robust, automated crack-width measuring techniques to provide accurate and less-expensive solutions for assessing infrastructure damages.

1.1 Image Processing Methods

To enable automated detection of cracks and widths, engineers and researchers explored imaging systems to detect structural defects since '90s. Imaging systems allow to capture the structures as images and then analyze individual images to detect defects. For instance, image processing is used to detect spalling and transverse cracks[9] and concrete cracks[10–13]. Researchers have designed Computer Vision-based intelligent algorithms to automate such tasks where analyzing individual image is laborious and time-intensive. For example, vision-based algorithms can quickly detect cracks in large structures, such as in dams[14,15] and bridges[16,17] with high accuracy. Likewise, we find vision-based algorithms being used in many engineering projects with applications to detect cracks in pavements[18,19], road surfaces[20,21], concrete surfaces[22,23], concrete bridges[24] and pothole detection[25,26]. Over the years, computer vision algorithms have been playing an important role in reducing cost, saving time and also providing much higher detection accuracy (matching or exceeding human performance).

To detect damage and assessment loss of reinforced concrete elements, Rivera *et al.*[27] proposed an automated procedure involving digital image processing called "I-Crack" to detect crack width from nine reinforced concrete shear of walls comprising different aspect ratios and reinforcement ratios. They use a range of image processing techniques (including morphological operations, segmentation) and pattern recognition to estimate width and compare with *I-Crack*. The results show a reasonably good accurate estimation, and the estimation varies with different reinforced concretes.

Lins and Givigi[28] propose crack recognition and crack measurement algorithms to estimate crack width. The crack recognition algorithm uses color histogram and particle filtering to detect cracks in four categories: complex, diagonal, vertical, or horizontal. Then the crack measurement algorithm uses the least-squares first degree polynomial approach to approximate the width. This approach has an error rate of 7.51% in estimating the crack widths.

Cho *et al.*[29] propose an edge-based crack-width transformation algorithm to detect concrete structural cracks. The technique comprises five steps: transforming crack widths, filtering aspect ratios, searching crack regions, filling holes, and finally, the threshold operation. They detect cracks in the images, but do not provide the width of the cracks. Also, the approach is more heuristic because of multiple variables used in detecting the cracks, and hence, may not always generalize well to different datasets.

Nguyen *et al.*[30] propose automatic crack-width measurement framework in 2D multiple-phase images. First, they enhance the cracks in images through filters; next, they use B-spline level set model to detect the crack locations, and in the last step they use Savitzky-Golay filter to measure crack widths. Although the results are good, they showed the results only in a few images, making it difficult to validate on larger datasets.

1.2 Deep Learning Methods

Since 2012, the deep learning field has gained significant attention for their ability to identify objects automatically in images, localize the detected objects, segment the objects from background, classify textual documents and recognize voices[31]. Convolutional Neural Networks (CNNs) form a major part of deep learning architectures to achieve state-of-the-art performances. This superior attribute of CNNs is because of their ability to learn mapping functions automatically from input to output, especially from spatial data (such as images). Before the revival of deep learning field for various vision-based automated tasks, researchers relied on handcrafted features. Example of such handcrafted features for detecting cracks in images include Hough Transform (features) and Support Vector Machine[32], texture features and neural networks[33], edge features and Adaptive Boosting (AdaBoost)[33], and Wavelet filters (as features) and energy functionals[34].

We can broadly classify CNN-based crack detection into two categories[35]: (i) Region-based classification methods and (ii) Semantic segmentation methods. Region-based CNN methods focus on identifying and classifying whether particular regions in an image have cracks. Semantic segmentation methods try to classify each pixel as belonging to one of the pre-defined class labels. For example, we can label the pixels associated with cracks to one class and associate the remaining pixels can to another class. Thus, the semantic segmentation approach aims to provide pixel-level classification of pixels. This is much more suited for predicting crack width, especially when we want to measure the crack widths automatically and when we need a much finer information about the cracks.

Region-based Classification Methods

Region-based methods consider certain image regions to detect cracks. In[36], Beckman *et al.* use the Faster R-CNN[37] approach to detect concrete spalling. The CrackDN[38] approach by Huyan *et al.* uses a modified version of the Faster R-CNN[37] to detect sealed and unsealed cracks. In[39], Zhang *et al.* combine convolution features along the time axis to produce favorable results in detecting and localizing cracks. Rao *et al.*[35] show how different region-based CNN methods perform with very high accuracy, sensitivity and specificity. In addition, Rao *et al.*[35] show that by creating patches from the larger images, they can detect cracks with near perfect accuracy and compare models for real-time inferences. The drawback of region-based classification methods is that they do not provide information about the exact width of the cracks—they inform us of the region (in which a crack is present) but not the exact location of the crack. Therefore, we cannot directly use region-based classification methods for crack-width quantification and measurements.

Semantic Segmentation Methods

Semantic segmentation methods aim to classify each pixel into one of the pre-determined classes ("crack" or "no crack"). In this direction, Zhang *et al.*[40] proposed *CrackNet* framework, which is a 5-layer CNN architecture, that detects cracks in 3D asphalt surfaces with 90.13% precision and 87.63% recall. On the other hand, Fully Convolutional Network (FCN)[41] employing encoder-decoder network has also shown excellent performance in classifying pixels and identifying cracks[35]. FCN employs encoder-decoder network to encode features (extract from input images with one of the backbone architectures from region-based methods to classify, but without the final output layer) and decode features (de-convolve and up-sample layers to reconstruct segmented images) to produce pixel-level classification. Li *et al.*[42] proposed FCN with DenseNet-121 (as encoder) to achieve a pixel accuracy of 98.61%. Yang *et al.*[43] tried FCN with VGG-19, but achieved the maximum accuracy of only 81.73%. In *DeepCrack*[44], Liu *et al.* extend FCN by combining FCN and Deeply Supervised Nets (DSN), whereas Alipour *et al.*[45] introduced *CrackPix* framework using FCN to detect cracks. Their framework achieved 92% accuracy in detecting crack pixels and 99% in detecting non-crack pixels. Bang *et al.*[46] employ FCN with ResNet-152 encoder network to detect cracks in road surface images with an F-measure of 0.74. In[47], Zhang *et al.* propose *SegNet* to detect pixel-level cracks. One of the limitation of these approaches is that after detecting the crack pixels, these methods do not estimate crack width from the segmented output.

Recently, Liu *et al.*[48] proposed U-Net architecture[49] to detect concrete cracks. Like FCN, the U-Net[49] uses encoder-decoder network but with some changes to the network architecture (please refer to Rao *et al.*[35] for more detailed information). In[48], Liu *et al.* use Focal Loss to improve the detection of crack pixels, but do not estimate crack width.

The performance of the handcrafted (manual) features often falls behind the deep learning methods. As a result, vision-based crack detection methods that use deep learning methods often outperform traditional approaches. In this paper, our proposed approach adopts a semantic segmentation deep learning framework. Our approach incorporates an encoder-decoder network with attention mechanism, time information (Recurrent Residual) and Random Forest regressor on top to predict crack width from user-specified locations.

2 Methodology

In this section, we present *Attention Recurrent Residual U-Net* (*Attention R2U-Net*) framework for detecting concrete cracks at pixel-level and predicting crack widths in RGB images. The Attention Recurrent Residual U-Net (Attention R2U-Net) model is a combination of *Attention U-Net*[50] and *Recurrent Residual U-Net*[51], designed for detecting pixel-level cracks in color images containing concrete surfaces. The proposed Attention R2U-Net model segments cracks at pixel level, and we use Random Forest (RF) regressor for learning the mapping function to estimate crack width from segmented pixels. From the literature, we see that the patch-based detection of cracks do not provide several

pixels belonging to crack regions[35]—they only identify the region where there is a crack which sometime is inefficient if we want to know the width of the cracks (say in millimeter or number of pixels). To address these challenges, we introduce a new pixel-wise segmentation architecture for generating crack segmentation map. This is a two-step approach including: (1) dense prediction map of each pixel from input image to segmented image using Attention R2U-Net and (2) predicting crack widths using a Random Forest regressor. The following subsections describe each of these steps. The current section describes the *Attention U-Net* architecture, training methodology and predicting crack width. Figure 1 provides three flowcharts: Figure 1(a) training the segmentation model, Figure 1(b) the regression (perdition) model and Figure 1(c) testing the complete (segmentation + regression) model. The flowcharts also highlight the steps connected to different sections of this article.

2.1 Attention R2U-Net Architecture for Crack Segmentation Map

Figure 2 shows the Attention R2U-Net architecture used for crack segmentation. It has two main paths: contracting and expansive. In the contracting path, we do a series of 3×3 convolutions, Rectified Linear Unit (ReLU) activations and 2×2 max-pooling operations with a stride (step) size of 2 (i.e., we are "down-sampling"). For each down-sampling operation, we double the number of channels. In contrast, the expansive path does the reverse of contraction path *i.e.*, contracting path incorporates "up-sampling" operation followed by 2×2 up-convolution (reduces feature channels by half), concatenating respective cropped feature maps from contracting path, and two 3×3 convolution operations each succeeded by ReLU activation. Here, the crop operations compensate for losing border pixels during convolution operations. At the end of the network, we have 1×1 convolution that maps 64-channel features into two-class ("crack" and "non-crack") pixel-level segmentation map. In the up-sampling and down-sampling operations, we use Recurrent Residual (R2) convolutional unit as shown in Figure 2, where we pass a recurrent convolution network through a residual unit. In addition, we have Attention Gates (AG) that take gate signal from the layer below and concatenate feature maps through skip connections to enhance features at coarser and finer levels. We have designed our approach to use Attention R2U-Net on 256×256 for predicting pixel-level crack segmentation map. Therefore, a $1024 \times 1024 \times 3$ size input image will have 16 number of $256 \times 256 \times 3$ image patches. We feed each patch to the Attention R2U-Net to produce a 256×256 segmentation map and then combine these segmentation maps to form a segmentation map whose size equals the original image size. We show this procedure in Figure 2 and suggest readers to refer to[50,51] for further information about R2U-Net and Attention U-Net.

2.2 Training Attention R2U-Net

We train the input images along with their ground truth segmentation maps using Stochastic Gradient Descent (SGD) optimization algorithm to learn the Attention R2U-Net weight parameters. For training the network, we combine the soft-max function with binary cross entropy (BCE) loss function at the last layer. The soft-max function provides the probability of each pixel either belonging to crack or non-crack, given by

$$p_k(\mathbf{x}) = \frac{\exp(a_k(\mathbf{x}))}{\sum_j^K \exp(a_j(\mathbf{x}))} \tag{1}$$

where $a_k(\mathbf{x})$ is the k^{th}-channel activation function at pixel position $x \in \mathbb{Z}^2$, K denotes the number of segmentation classes (two in our case), and $p_k(x)$ is the maximum probability of pixel at position x at k^{th}-channel; $p_k(x) \approx 1$ for k which achieves maximum activation $a_k(\mathbf{x})$; $p_k(x) \approx 0$ for rest of the k.

We can express the binary cross entropy (BCE) loss as

$$l(\mathbf{y}, \hat{\mathbf{y}}) = -\frac{1}{N} \sum_{i=1}^{n} w_1 \times y_i \log(\hat{y}_i) + w_0 \times (1 - y_i) \log(1 - \hat{y}_i), \tag{2}$$

where y is the actual label of the pixel in the mask provided by annotation, $\hat{y} = p_k(\mathbf{x})$ is the predicted label class for the same pixel by the model and N is the number of samples, w_k is the weight function applied to add more importance to a particular class k of pixels during training to compensate pixel distribution. BCE loss function is a special case of cross entropy function that measures the distance between two probability distributions $\mathbf{y}$ and $\hat{\mathbf{y}}$, wherein $\mathbf{y}$ is the distribution of pixels given as input to the model and $\hat{\mathbf{y}}$ is the distribution predicted by the model. The entropy function measures the randomness in an event (measured via information bits) and the BCE loss suggests the additional bits required to represent the target class because of randomness in the data in place of true pixel distribution. The term $y_i \log(\hat{y}_i)$ is the cross entropy of "crack" class and the second term $(1 - y_i) \log(1 - \hat{y}_i)$ is

the cross entropy of "non-crack" class[35], w_1 is the weights applied to "crack" class pixels, and w_0 is the weights applied to "non-crack" class pixels. In (2), the BCE loss function penalizes the model when it deviates from true pixel distribution for each pixel.

2.3 Crack Width Prediction: Attention R2U-Net + Random Forest

Figure 3 shows the overview of our approach to predict crack width from Attention R2U-Net segmentation map and Random Forest regressor. To predict the width of the cracks, we extract feature vectors (1×100 pixels) containing segmentation map of cracks from Attention R2U-Net at two locations. During training, these feature vectors are used to map from segmentation output from Attention R2U-Net to crack width. In the testing phase, the feature vectors from test images provide the predicted crack width. The Random Forest Regressor[52] maps the segmentation map using feature vectors to generate crack widths. In our case, we use Random Forests to extract a random combination of features from $n \times 100$ training features to generate n decision trees. Our approach constructs a decision tree based on the feature values and decides how best to predict the crack width. An individual decision tree alone is not sufficient, as it may lead to under- or over-training. Hence, Random Forests use a combination of decision trees to make several weak predictions. Then, the aggregated (averaged) weak predictions from individual trees predict a single continuous value, which in our case is the crack width.

3 Experiment and Evaluation

3.1 Model training and configuration

We implemented all the models using Python 3.6 and PyTorch 1.4 comprising Nvidia V100 Graphics Processing Unit (GPU) card with 16 GB graphics memory on Ubuntu 16.04 to train the models. To test the performance of the trained models, we used a laptop with 64-bit Ubuntu Operating System (OS), 16 GB RAM and 8$^{\text{th}}$ generation Intel® Core™ i7-8550 CPU with a clock speed of 1.80 GHz.

3.2 Dataset and Annotation

The dataset comprises 4253 training images (3828 train + 425 validation) of size 256×256 and 150 test images of size 1024×1024. The images have 24-bit depth (three channels). We created this data by combining crack 2333 images from[35] with 150 new images of 1024×1024 acquired in our lab from 3 concrete beams (50 from each specimen). From 150 images of 1024×1024, we used 120 images for training/validation and the remaining 30 for testing. We divided the 120 images into 256×256 images resulting in $120 \times 16 = 1920$ patches. Table 2 provides a detailed information of the dataset used in this work. We combined 2333 crack images from Rao *et al.*[35] and 1920 images of size 256×256 from three concrete beams (C1,C2 and C3 obtained in our lab) to improve the model robustness by providing diverse textured concretes and different variety of cracks. The concrete beam data was collected through a video camera with a resolution of 1920×1088 at the frame rate of 24 frames per second and later center-cropped to images of size 1024×1024. The three samples (C1, C2, and C3) were from the same standard mix for concrete structures (using general-purpose cement, class F fly ash, fine aggregate, coarse aggregate, and water). In this study, to demonstrate the performance of the proposed method by detecting cracks and predicting crack width, the three beam specimens (100 mm × 100 mm × 450 mm) were subjected to the three-point bending test setup in the laboratory. The pixel density for the three concrete beams (C1, C2, and C3) are 10.78803, 10.65113, and 10.60151 pixels/mm, respectively.

We annotated each pixel in the image as "crack" or "no crack" and other researchers can access via this link*.

3.3 Handling Data Imbalance

Table 3 shows the percentages of crack and non-crack pixels used in training and testing the models. From the table, we see the percentage of crack pixels in only 1.5%, whereas the percentage of non-crack pixels is about 98.5%, highlighting significant challenges the vision algorithms face in differentiating crack and non-crack pixels. This makes the training dataset skewed and as a result adds bias to the models (to learn only majority class while discarding other classes) using this dataset as it is. This problem is known as "data imbalance," which occurs commonly is most of the practical datasets. If we do not manage this data imbalance, we are likely to have incorrect pixel-level classification results on the test samples. To ensure that the models have right discriminating representations of both classes (0 as background and 1 as crack), we provide a weight tensor during training of $[1, 200]$ based on the pixel distribution available in Table 3. We arrived at this weight vector by using the inverse relationship between classes and pixel density and scaling (this is an approximation based on expert domain knowledge for balancing pixel distribution). During training,

*https://www.dropbox.com/sh/li52jux68xz5zkq/AABJTdJW9dCcUQQObaj9g5jwa?dl=0

this weight vector instructs the learning model to learn the threshold boundary between two classes in feature space such that the model can balance the majority class by adding a weight (of 200) to class 1 ("crack") pixels and reduce the weight (by a factor of 200) of class 0 ("no crack") pixels. By adding a weight vector, we ensure that the trained model is balanced in the sample space and provides an unbiased prediction in the test cases.

3.4 Evaluation Metrics

The most widely used metrics for evaluation of segmentation maps are: (i) pixel accuracy (PA), (ii) mean pixel accuracy (mPA), (iii) mean intersection over union (mIoU), and (iv) frequency weighted intersection over union (FwIoU)[42]. We use these metrics to quantify our results. In addition, Boundary F1 score (BF) provides an efficient metric to measure each individual segmentation structures within an image[53]. However, we are not using BF score to evaluate our results as it is more appropriate for contours with shapes, whereas it is not suitable to measure crack widths. In this work, we adopt the definitions of these metrics defined in[41]

If we represent the number of classes as n_{cl}, the number of pixels n_{ij} of class i predicted to be of class j, and $t_i = \sum_j n_{ij}$ being the total number of pixels of class i, then we can compute:

- Pixel accuracy (PA) as the number of correctly identified pixel for each class:

$$\text{PA} = \frac{\sum_i n_{ii}}{\sum_i t_i} \tag{3}$$

- Mean pixel accuracy (mPA) as the average accuracy over all classes:

$$\text{mPA} = \frac{1}{n_{cl}} \sum_i \frac{n_{ii}}{t_i} \tag{4}$$

- Intersection over Union (IoU) measuring the amount of predicted overlapping pixels with respect to ground truth pixels:

$$\text{IoU} = \frac{\text{Overlapping Area}}{\text{Area of Union}} = \frac{|P \cap G|}{|P \cup G|} \tag{5}$$

where P is the predicted pixels, G is ground truth pixels and $|\cdot|$ is the number of pixels in the set.

- Mean IoU (mIoU) as the average IoU over all classes:

$$\text{mIoU} = \frac{1}{n_{cl}} \sum_i \frac{n_{ii}}{t_i} + \sum_j n_{ji} - n_{ii} \tag{6}$$

- Frequency weighted intersection over union (FwIoU) extending mIoU with weights assigned based on frequency of each class:

$$\text{FwIoU} = \frac{1}{\sum_k t_k} \frac{\sum_i t_i n_{ii}}{t_i} + \sum_j n_{ji} - n_{ii} \tag{7}$$

3.5 Training and Validation of Pixel-level Segmentation Models

In this work, we use the BCE loss function in (2) as the primary training optimization objective and allow ± 2 pixel tolerance for validating the models. We use Stochastic Gradient Descent (SGD) with step learning decay of 25 epochs, and learning rate decay (γ) of 0.1. During training, all the layers in the model were allowed to update gradients, and we have used 50 epochs with a mini-batch size of 4, 8 16 or 32 (depending on how much GPU memory is available for each model before starting the training). In this work, we consider the best models based on how each model performs on validation dataset.

3.6 Training and Validation of Regression Models Prediction

To predict crack widths from Attention R2U-Net output (*i.e.*, from segmented crack maps), we use feature vectors of size 1×100 pixels from two locations and train the Random Forest model. We chose two locations because we annotated cracks at two locations in the three concrete specimen (C1, C2 and C3) for evaluation. Hence, we are using two locations in our crack width prediction algorithms. We chose the width of 100 pixels because our maximum crack width is about 8.95 mm (*i.e.*, 8.95 x pixel density of C1 = $8.95 \times 10.78803 \simeq 97$ pixels' width). Table 1 provides a statistical summary of widths of the cracks present at two locations in the three concrete specimens (C1, C2 and C3). We have 50 images of 1024×1024 size from each concrete type (total 150 images from C1, C2 and C3) of which, 40 each used for training and 10 each for testing the regression models. As shown in Figure 3, feature vectors from two locations will result in 80 training samples (feature vectors) and 20 testing samples for each concrete specimen. From Table 1, we notice that the maximum widths of the cracks in the training set to be 8.95 mm for C1, 4.37 mm from C2 and 3.80 mm for C3. In the testing images, we can see similar maximum crack widths (8.28 mm, 3.89 mm and 3.15 mm for C1, C2 and C3, respectively). Figure 4 provides a visual spread of crack widths for training and testing sets for C1, C2 and C3. From Figure 4, we notice that the test sets have

similar distributions as training sets and we also see some dominant crack widths (such as 0 mm to 3 mm) appearing common across C1, C2 and C3. For training our Random Forest model, we use the grid search approach to identify the hyper-parameters from the training set. In our case, our optimal hyper-parameters were (1) number of decision trees = 100, (2) criteria to split each node as "MAE", (3) max depth of trees = 90, (4) subset of features when splitting uses "log2" function, and (5) minimum samples at leaf nodes = 1. We use grid search to find the hyper-parameters for all the regression models, and the details of these settings are available in the code.

3.7 Comparison of Segmentation Results

Figures 5, 6 and 7 show the sample segmentation output of U-Net, Attention R2U-Net and other segmentation models for the three concrete surfaces (C1, C2 and C3). We notice that the three concrete specimen (C1, C2 and C3) have different textures and hence the prediction results will be different. This provides us with an opportunity to generalize our models for crack detection. We compare our Attention R2U-Net results with other state-of-the-art segmentation architectures, such as Attention U-Net[50], Recurrent Residual U-Net[51], FCN-ResNet101[41] and DeepLab-v3-Plus[54].

Table 4 provides a quantitative comparison of crack segmentation output based on four evaluation (PA, mPA, mIoU and FwIoU) metrics. The U-Net model uses encoder and decoder architecture for pixel-level prediction, whereas the Attention U-Net[50] uses attention gate (AG) mechanism to focus on target shapes and structures (*i.e.*, the cracks). The proposed Attention R2U-Net[51] combines attention mechanism along with time component to aggregate features from the last iteration, providing a better feature representation of segmented cracks. We use the time parameter ($t = 2$, *i.e.*, our model uses a recurrent convolutional operation including a single convolution layer followed by two recurrent convolutional layers) in the Attention R2U-Net for crack-width prediction. Attention R2U-Net provides better pixel accuracy (PA) and FwIoU, as seen in Table 4. On the other hand, DeepLab-v3-Plus[54] uses special operator called atrous spatial pyramid pooling (ASPP) at the end of encoder to handle segmentation at multiple scales and hence, have better mPA (0.9613) and mIoU (0.8176). Though these models perform well in quantitative metrics, they also introduce noise in the segmentation output. In addition, models such as DeepLab-v3-Plus[54] and FCN-ResNet101[41] are complex and deep, which need large datasets and memory, limiting their usage in certain practical use cases. As a result, there is also a tendency for these models to overfit on the training data. Therefore, we use Attention R2U-Net, which is relatively simple, generates very minimal noise in the output and also requires less GPU memory and training time.

3.8 Comparison of Crack Width Prediction

In Table 5, we compare the five segmentation models (FCN-ResNet101, DeepLab-v3-Plus, U-Net, Attention U-Net and Attention R2U-Net) with five regression models for predicting crack width. To provide a fair comparison of model performances, we have trained all the segmentation models on the same dataset and then tabulated their performance in Table 5. The five regression models include Random Forest (RF), Neural Network (NN), Support Vector Regression (SVR), Gradient Boosting Regression (GBR), and Extreme Gradient Boosting Regression (XGBR). From Table 5, we see that Attention R2U-Net with Random Forest has the lowest absolute maximum error (AME) of 1.30 mm, with an RMSE of 0.45 mm, MAE of 0.31 mm, and coefficient of determination $R^2 = 0.91$. We note that although Attention R2U-Net with NN provides better RMSE (0.38 mm), MAE (0.25 mm) and $R^2 = 0.94$, it has larger AME of 1.44 mm when compared with RF. AME is the absolute maximum prediction error we get from the regression models on the test samples. We prefer AME as low as a possible so that there is an upper bound on the error for practical uses by the end users. Attention R2U-Net with SVR, GBR and XGBR regression models also provide good prediction results in terms of RMSE, MAE and R^2 metrics. From Table 5, we can notice that lower the RMSE and MAE errors, higher will be the R^2 score; however, AME need not always be lower with higher R^2.

From Table 5 overview, we notice that Attention R2U-Net provides the highest R^2 (with lower RMSE and MAE) when compared with FCN-ResNet101, DeepLab-v3-Plus, U-Net and Attention U-Net. Along the same lines, one can notice that RF regression model performs better than other regression models. We prefer RF over NN in predicting crack-width as RF has inherent advantages in the way it creates decision trees and randomizes the data during training, reducing over-fitting and generalizing well for new, unknown data points. For example, Attention R2U-Net with NN produces $R^2 = 0.94$ and AME of 1.44 mm, whereas the Attention R2U-Net with RF has a slightly lower $R^2 = 0.91$ but better AME of 1.3 mm. We can attribute this property to NN model's higher variance (trying to fit all points in the model during training), whereas RF tries to reduce variances by using random subsets of training samples and subsets of features (*i.e.*, subsets of feature vector columns)

during training to learn the mapping function from crack segmentation to predict crack width.

Figure 8 shows the regression results of Attention R2U-Net with RF, NN, SVR and GBR for three concrete beams (C1, C2 and C3) used in this research. There were 60 crack-width testing samples (refer to Table 1 for more information dataset) containing varying crack widths from 0 mm to approximately 8.95 mm. C2 and C3 have crack widths ranging between 0 and approximately 4 mm, whereas C1 from 0 to ~ 8.95 mm. From Figure 8, we see that all regression models are very close in predicting the crack widths; however, RF and NN predict better compared to SVR and GBR. As mentioned earlier, our aim is to have models with low variance. RF has lower variance and we can visualize this in Figure 8, where it tries to predict the crack width by balancing R^2 and AME. In contrast, NN tries to predict most of the points, but it cannot predict correctly if some of the testing samples have more deviations. Figure 9 shows similar results but highlights two crack locations from three concrete beams. Location 2 has a much wider crack width and as a result, we can expect higher prediction error because of wider Location 2 crack width in all models. We chose two locations because we annotated cracks in the three concrete specimen at two locations for evaluation. Hence, we are choosing two locations in our crack-width prediction algorithms. The width of 100 pixels was chosen because our maximum crack width is about 8.95 mm (*i.e.*, 8.95 x pixel density of C1 = $8.95 \times 10.78803 \simeq 97$ pixels' width). The crack widths of C2 and C3 are lesser than C1 and hence we extracted patterns of 100 pixels from two locations to compare and evaluate our approach. We can also choose 1×1024 pixels, which would result in the same result but with more computation. The result from 1×100 pixels and 1×1024 pixels (*i.e.*, the one complete row of the input image) would be same because we did not have multiple crack vertically and our approach will identify all the pixels belonging to the crack pixels as "1"s and hence, performing a regression on 1×100 pixels and 1×1024 pixels would yield the same results.

Table 6 provides a comparison of the number of model parameters and the prediction error. Figure 10 provides a comparison of (a) prediction error (in mm) for each segmentation model combined with regressors and (b) the inference time (for each patch of size 256×256) of each segmentation model. We see that the inference time for the basic U-Net is about 0.2 seconds. In contrast, the inference times increase as the model's complexity increases, especially with the inclusion of the Attention mechanism. The proposed Attention R2U-Net requires about 0.8 seconds to infer crack width from a patch of size 256×256. There is a trade-off between achieving the lowest AME: for example, the proposed Attention R2U-Net, which provides the lowest AME, has higher model complexity (in terms of architecture) and requires higher inference time.

On the other hand, U-Net has the lowest model complexity (and the number of trainable parameters) and hence requires only about 0.2 seconds per patch but has higher AME (of 2.32 with RF as a regressor, refer to Table 5). The remaining segmentation models have in-between complexities, and hence, their inference times also lie in-between. We also note from Table 6 that the model with more trainable parameters does not necessarily result in slower inference. For example, both FCN-ResNet101 and DeepLab-v3-Plus have 51 million and 58 million parameters, respectively. In contrast, Attention R2U-Net has 39 million parameters but requires a higher inference time. Attention R2U-Net with RF provides the lowest AME compared to other segmentation methods and has a higher R-squared measure. This work outlines the model complexities and inference times, and depending on the nature of crack width prediction application, one can choose a suitable model. Please note that the model parameters listed in Table 6 do not include parameters from regression models. The number of regression model parameters are significantly less compared to the segmentation model parameters.

The proposed (segmentation) approach has two main advantages to predict crack width when compared to region-based methods. First, when we compare the inference times of the proposed Attention R2U-Net with the region-based crack detection approaches[35], the VGG-16 (with the highest accuracy) requires 0.22 seconds to infer the presence of a crack on the patch size of 64×64. If we extrapolate the same inference time of VGG-16 to a patch size of 256×256 to compare with the Attention R2U-Net, we see that the VGG-16 requires $\sim 0.22 \times 16 = 3.52$ seconds, which is much higher than the inference time of Attention R2U-Net. Second, the region-based methods such as the VGG-16 detect only the crack regions. In contrast, Attention R2U-Net (a segmentation approach) provides the width of the crack with millimeter accuracy.

We note that the patch-based detection of cracks does not provide several pixels belonging to crack regions as they only identify the crack regions, but not the width of the cracks[35]. In other words, we want to detect millimeter-level accuracy of crack widths for automated detection of cracks. As presented in this work, the accurate detection and measurement of cracks are crucial in the inspection and maintenance of concrete structures. For example,

in the inspection and maintenance manual of the local infrastructure agency[2], it is highlighted that those crack locations and crack widths are the key indicators of the performance and capacity of concrete structures. The trained model developed in this work can be deployed on drones to inspect bridges by detecting and quantifying cracks on concrete beams, columns. In addition, the developed model can be used to assist the measurement of cracks on the laboratory testing of concrete structures (e.g., concrete walls under seismic tests in[27]).

3.9 Comparison of Detected Crack Length

Besides detecting crack width, it is also important to extract crack length for reporting and inspection purposes. In our framework, we have included an additional image processing module to process the segmented images from segmentation models and extract the crack lengths automatically from the images. The module does a series of image morphological operations (such as noise removal, finding the top and bottom pixels of the cracks) to extract the length. Table 7 provides a comparison of detected crack length against the actual length for three different concrete specimen (C1, C2 and C3). From Table 7 we see that, overall, the Attention U-Net model is performing better in terms of error metrics. Most models are performing better in extracting the crack lengths on C1 and C2 specimen when compared with C3 specimen. Our proposed *Attention R2U-Net* framework also does comparatively well in terms of MAE and R-squared metrics. (Note: we want RMSE and MAE to be lower, whereas R-squared to be close to 1). We provided this section to show how the proposed approach is useful for detecting crack lengths in addition to crack width. However, detecting length accurately needs further work, which we have outlined in the following section.

3.10 Future Work

In this research, we proposed Attention R2U-Net with RF for predicting crack widths in concrete images, which provides the lowest absolute maximum error (of 1.3 mm) and $R^2 = 0.91$. We also note some limitations in our work and try to address these in our future research. First, we collected data from three concrete surfaces in our lab. As we had limited data, we combined this data with another dataset to train the models. The crack widths in the two datasets vary and can introduce bias in the models we train. Second, acquiring large datasets and then annotating them at pixel level is a time-consuming and labour-intensive process. The pixel-level annotations depend on the domain expertise and precision with which experts can annotate, and the provision provided by the software. In our case, we use open source annotation tool (Sefexa [†]) to segment pixels and this had limited capability in segmenting pixels, especially when the cracks were tiny and this might have inadvertently considered the surrounding pixels as belonging to cracks. Nonetheless, we have trained the models to produce robust results for a variety of crack widths with good accuracy and confidence. In addition, we have designed an image processing algorithm for automatically extracting the crack length for inspection and reporting purposes. In future work, one could also look into building regression models to map extracted length to match correct lengths. As our primary goal was to detect crack widths, we have not pursued this direction and instead we have shown that we could extract lengths that approximately match the correct length with no regression analysis.

Overall, the proposed work provides a promising solution for estimating and/or predicting crack width in concrete images. In our work, Attention Gates and Recurrent Residual convolutional layers in the Attention R2U-net improved the efficiency of detecting cracks and is therefore, performing well when compared to other segmentation models. This work will contribute towards automated detection of crack widths during building inspections, bride monitoring, and other SHM applications. We can also easily adopt the models to SHM using drones and Internet of Things (IoT), and integrate with other big data models that predict structural health from other IoT sensors.

4 Conclusion

This work focused on detecting millimeter-level accuracy of crack widths from images of concrete structures. This level of accuracy and measurement of cracks are crucial for the inspection and maintenance of concrete structures. This work proposed *Attention Recurrent Residual U-Net (Attention R2U-Net)* combined with *Random Forest* to predict pixel-level crack widths in concrete surface images. Patch-based crack detection approach cannot provide the crack width (or the number of pixels belonging to the crack), and hence they can only identify the region where there is a crack. In contrast, our work provides a deep learning framework for pixel-level crack prediction, with the mean prediction error of ± 0.31 mm for crack widths varying from 0 to ~ 8.95 mm. Our contributions in this work include:

[†] http://www.fexovi.com/sefexa.html

1. Our proposed *Attention R2U-Net* with Random Forest is a novel framework for estimating crack width. To the best of our knowledge, we are the first to propose this framework for predicting crack width in concrete crack images.

2. Our framework uses local prediction and feature aggregation to handle larger-sized images (1024×1024 pixels or more). We achieve this by slicing images into smaller-sized patches (256×256 pixels) and combining them to have view cracks globally. This approach provides flexibility in analyzing larger images. It ensures that we can quickly train the models on IoT sensor devices with limited memory and processing capabilities, such as drones.

3. As our framework provides a direct estimate of crack widths in millimeters from input images, it enables real-time crack width prediction in building inspection, bridge monitoring and other SHM applications.

Our work also provides a way forward for processing large-sized images commonly acquired using High Definition(HD) cameras. We have compared our proposed framework with a combination of four other pixel segmentation models and regression models. Our proposed model produces the lowest absolute maximum error (1.3 mm). Further, one can quickly adapt our framework to various SHM applications using drones and IoT sensors.

Acknowledgements

This work was supported by the CRC-P for Advanced Manufacturing of High Performance Building Envelope project, funded by the CRC-P program of the Department of Industry, Innovation and Science, Australia, and the Asia Pacific Research Network for Resilient and Affordable Housing (APRAH) grant, funded by the Australian Academy of Science, Australia.

References

1. Biernacki JJ, Bullard JW, Sant G et al. Cements in the 21st century: challenges, perspectives, and opportunities. *Journal of the American Ceramic Society* 2017; 100(7): 2746–2773.
2. VicRoads. Road structures inspection manual, 2018.
3. Kim H, Ahn E, Cho S et al. Comparative analysis of image binarization methods for crack identification in concrete structures. *Cement and Concrete Research* 2017; 99: 53–61.
4. Peter C, Alison F and Liu S. Review paper: health monitoring of civil infrastructure. *Structural health monitoring* 2003; 2(3): 0257–267.
5. Tikka J, Hedman R and Silijander A. Strain gauge capabilities in crack detection. In *4th International Workshop on Structural Health Monitoring*. pp. 15–17.
6. Neild S, Williams M and McFadden P. Development of a vibrating wire strain gauge for measuring small strains in concrete beams. *Strain* 2005; 41(1): 3–9.
7. Tan X and Bao Y. Measuring crack width using a distributed fiber optic sensor based on optical frequency domain reflectometry. *Measurement* 2020; : 108945.
8. Wang X, Chakraborty J, Bassil A et al. Detection of multiple cracks in four-point bending tests using the coda wave interferometry method. *Sensors* 2020; 20(7): 1986.
9. Tsao S, Kehtarnavaz N, Chan P et al. Image-based expert-system approach to distress detection on crc pavement. *Journal of Transportation Engineering* 1994; 120(1): 52–64.
10. Abdel-Qader I, Abudayyeh O and Kelly ME. Analysis of edge-detection techniques for crack identification in bridges. *Journal of Computing in Civil Engineering* 2003; 17(4): 255–263.
11. Yamaguchi T and Hashimoto S. Image processing based on percolation model. *IEICE transactions on information and systems* 2006; 89(7): 2044–2052.
12. Yamaguchi T and Hashimoto S. Automated crack detection for concrete surface image using percolation model and edge information. In *IECON 2006 - 32nd Annual Conference on IEEE Industrial Electronics*. ISBN 1553-572X, pp. 3355–3360.
13. Yamaguchi T and Hashimoto S. Improved percolation-based method for crack detection in concrete surface images. In *2008 19th International Conference on Pattern Recognition*. ISBN 1051-4651, pp. 1–4.
14. Chen Cp, Wang J, Zou L et al. A novel crack detection algorithm of underwater dam image. In *2012 International Conference on Systems and Informatics (ICSAI2012)*. IEEE, pp. 1825–1828.
15. Shi P, Fan X, Ni J et al. A detection and classification approach for underwater dam cracks. *Structural Health Monitoring* 2016; 15(5): 541–554.
16. Yeum CM and Dyke SJ. Vision-based automated crack detection for bridge inspection. *Computer-Aided Civil and Infrastructure Engineering* 2015; 30(10): 759–770.
17. Oh JK, Jang G, Oh S et al. Bridge inspection robot system with machine vision. *Automation in Construction* 2009; 18(7): 929–941.
18. Kaseko MS and Ritchie SG. A neural network-based methodology for pavement crack detection and classification. *Transportation Research Part C: Emerging Technologies* 1993; 1(4): 275–291.

19. Huang Y and Xu B. Automatic inspection of pavement cracking distress. *Journal of Electronic Imaging* 2006; 15(1): 013017.
20. Sy N, Avila M, Begot S et al. Detection of defects in road surface by a vision system. In *MELECON 2008-The 14th IEEE Mediterranean Electrotechnical Conference*. IEEE, pp. 847–851.
21. Gavilán M, Balcones D, Marcos O et al. Adaptive road crack detection system by pavement classification. *Sensors* 2011; 11(10): 9628–9657.
22. Koch C, Georgieva K, Kasireddy V et al. A review on computer vision based defect detection and condition assessment of concrete and asphalt civil infrastructure. *Advanced Engineering Informatics* 2015; 29(2): 196–210.
23. Yu SN, Jang JH and Han CS. Auto inspection system using a mobile robot for detecting concrete cracks in a tunnel. *Automation in Construction* 2007; 16(3): 255–261.
24. Prasanna P, Dana KJ, Gucunski N et al. Automated crack detection on concrete bridges. *IEEE Transactions on automation science and engineering* 2014; 13(2): 591–599.
25. Koch C and Brilakis I. Pothole detection in asphalt pavement images. *Advanced Engineering Informatics* 2011; 25(3): 507–515.
26. Rao AS, Gubbi J, Palaniswami M et al. A vision-based system to detect potholes and uneven surfaces for assisting blind people. In *2016 IEEE International Conference on Communications (ICC)*. IEEE, pp. 1–6.
27. Rivera JP, Josipovic G, Lejeune E et al. Automated detection and measurement of cracks in reinforced concrete components. *ACI Structural Journal* 2015; 112(3).
28. Lins RG and Givigi SN. Automatic crack detection and measurement based on image analysis. *IEEE Transactions on Instrumentation and Measurement* 2016; 65(3): 583–590.
29. Cho H, Yoon HJ and Jung JY. Image-based crack detection using crack width transform (cwt) algorithm. *IEEE Access* 2018; 6: 60100–60114.
30. Nguyen HN, Nguyen TY and Pham DL. Automatic measurement of concrete crack width in 2d multiple-phase images for building safety evaluation. Intelligent Information and Database Systems, Springer International Publishing. ISBN 978-3-319-75420-8, pp. 638–648.
31. LeCun Y, Bengio Y and Hinton G. Deep learning. *nature* 2015; 521(7553): 436.
32. Hu H, Gu Q and Zhou J. Htf: a novel feature for general crack detection. In *2010 IEEE International Conference on Image Processing*. ISBN 2381-8549, pp. 1633–1636.
33. Chen Z, Derakhshani R, Halmen C et al. A texture-based method for classifying cracked concrete surfaces from digital images using neural networks. In *The 2011 International Joint Conference on Neural Networks*. IEEE. ISBN 1424496373, pp. 2632–2637.
34. Li G, He S, Ju Y et al. Long-distance precision inspection method for bridge cracks with image processing. *Automation in Construction* 2014; 41: 83–95.
35. Rao AS, Nguyen T, Palaniswami M et al. Vision-based automated crack detection using convolutional neural networks for condition assessment of infrastructure. *Structural Health Monitoring* 2020; : 1–19DOI:10.1177/1475921720965445.
36. Beckman GH, Polyzois D and Cha YJ. Deep learning-based automatic volumetric damage quantification using depth camera. *Automation in Construction* 2019; 99: 114–124.
37. Girshick R, Donahue J, Darrell T et al. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. pp. 580–587.
38. Huyan J, Li W, Tighe S et al. Detection of sealed and unsealed cracks with complex backgrounds using deep convolutional neural network. *Automation in Construction* 2019; 107: 102946.
39. Zhang Q, Barri K, Babanajad SK et al. Real-time detection of cracks on concrete bridge decks using deep learning in the frequency domain. *Engineering* 2020; .
40. Zhang A, Wang KC, Li B et al. Automated pixel-level pavement crack detection on 3d asphalt surfaces using a deep-learning network. *Computer-Aided Civil and Infrastructure Engineering* 2017; 32(10): 805–819.
41. Long J, Shelhamer E and Darrell T. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. pp. 3431–3440.
42. Li S, Zhao X and Zhou G. Automatic pixel-level multiple damage detection of concrete structure using fully convolutional network. *Computer-Aided Civil and Infrastructure Engineering* 2019; 34(7): 616–634.
43. Yang X, Li H, Yu Y et al. Automatic pixel-level crack detection and measurement using fully convolutional network. *Computer-Aided Civil and Infrastructure Engineering* 2018; 33(12): 1090–1109.
44. Liu Y, Yao J, Lu X et al. Deepcrack: A deep hierarchical feature learning architecture for crack segmentation. *Neurocomputing* 2019; 338: 139–153.
45. Alipour M, Harris DK and Miller GR. Robust pixel-level crack detection using deep fully convolutional neural networks. *Journal of Computing in Civil Engineering* 2019; 33(6): 04019040. DOI:doi:10.1061/(ASCE)CP.1943-5487.0000854.
46. Bang S, Park S, Kim H et al. Encoder–decoder network for pixel-level road crack detection in black-box images. *Computer-Aided Civil and Infrastructure Engineering* 2019;

34(8): 713–727. DOI:10.1111/mice.12440.

47. Zhang X, Rajan D and Story B. Concrete crack detection using context-aware deep semantic segmentation network. *Computer-Aided Civil and Infrastructure Engineering* 2019; 34(11): 951–971.
48. Liu Z, Cao Y, Wang Y et al. Computer vision-based concrete crack detection using u-net fully convolutional networks. *Automation in Construction* 2019; 104: 129–139.
49. Ronneberger O, Fischer P and Brox T. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*. Springer, pp. 234–241.
50. Oktay O, Schlemper J, Folgoc LL et al. Attention u-net: Learning where to look for the pancreas. In *1st Conference on Medical Imaging with Deep Learning*. pp. 1–10.
51. Alom MZ, Yakopcic C, Hasan M et al. Recurrent residual u-net for medical image segmentation. *Journal of Medical Imaging* 2019; 6(1): 014006.
52. Breiman L. Random forests. *Machine learning* 2001; 45(1): 5–32.
53. Csurka G, Larlus D, Perronnin F et al. What is a good evaluation measure for semantic segmentation?. In *BMVC*, volume 27. p. 2013.
54. Chen LC, Zhu Y, Papandreou G et al. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European conference on computer vision (ECCV)*. pp. 801–818.

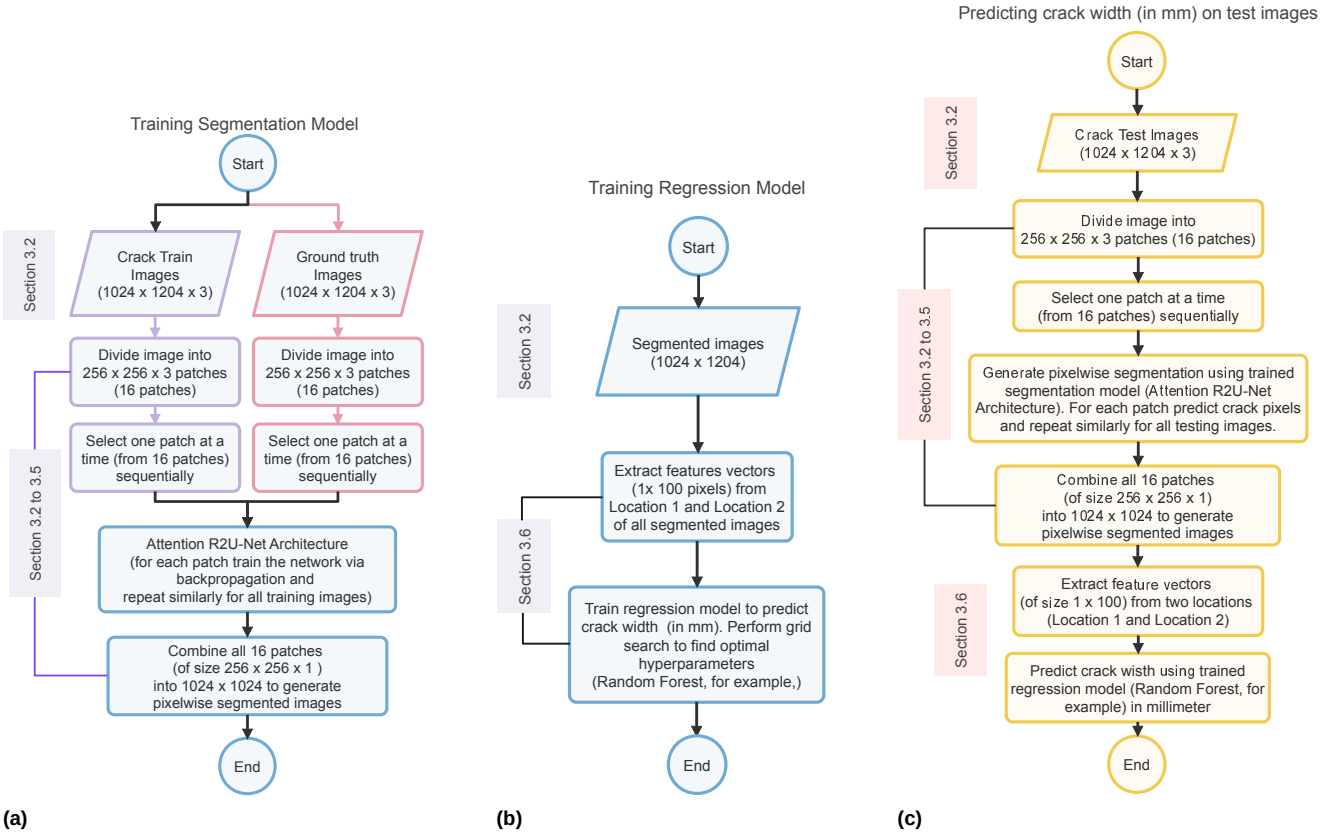

Figure 1. Crack width prediction flowchart: (a) training segmentation model, (b) training regression model, and (c) predict crack width by combining segmentation and regression models. *Note:* we have shown Attention R2U-Net segmentation model and Random Forest as the regression model as an example here. However, when comparing the model performances, we change the segmentation and regression models.

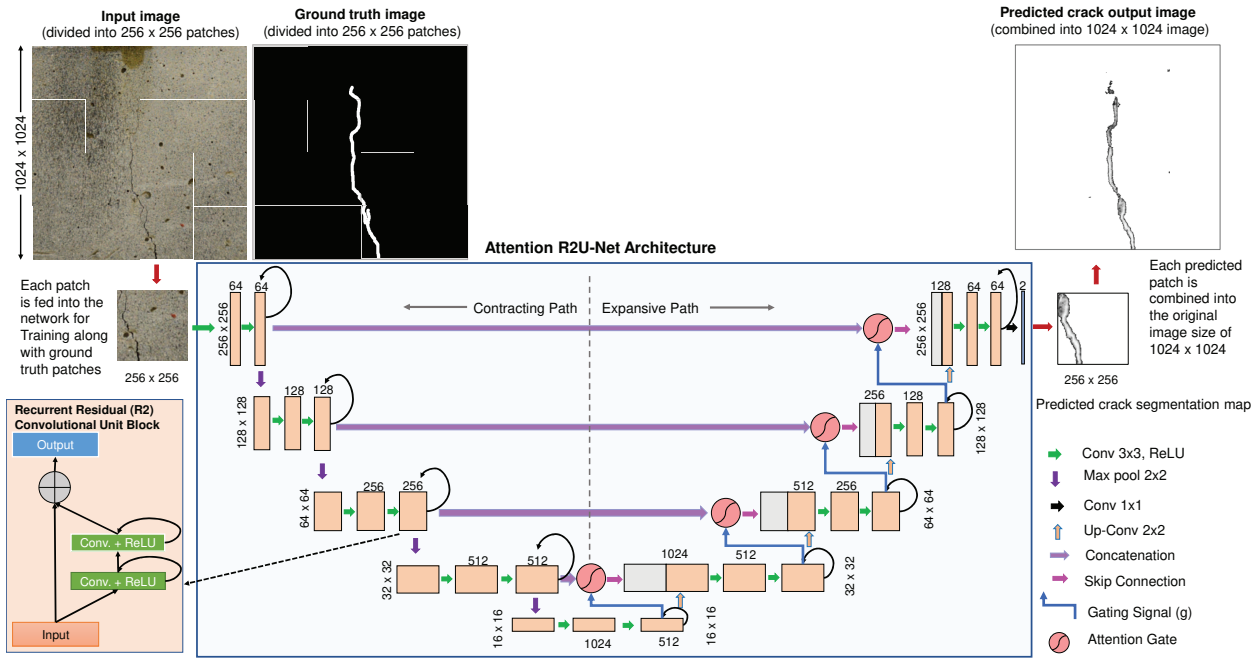

Figure 2. Pixel-level crack detection based on Attention R2U-Net architecture in RGB images. Input images of size 1024×1024 are divided into 256×256 patches to create 16 such patches. Each patch is then used as input to Attention R2U-Net for training and testing to produce crack segmentation map. These 16 segmented patches are then combined into a single 1024×1024 image to produce the final crack segmentation map of the original image.

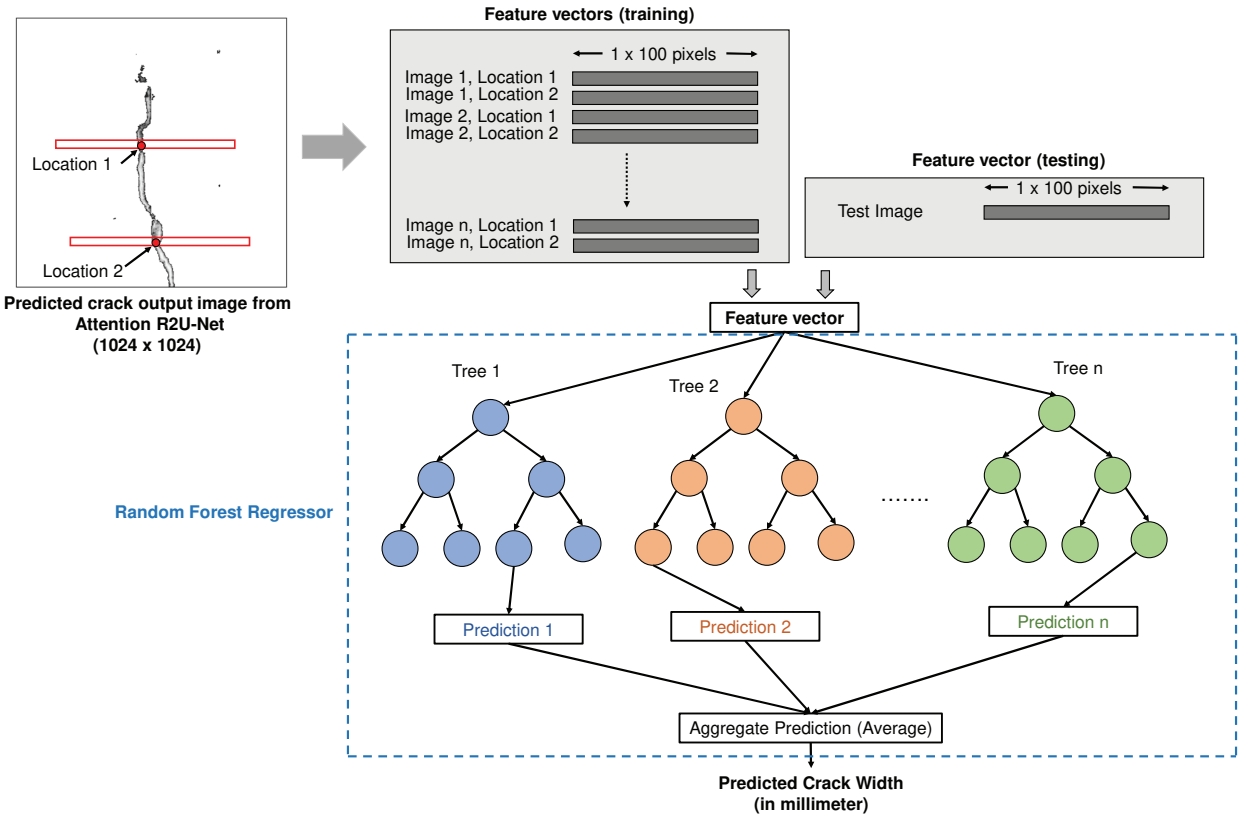

Figure 3. Crack width prediction (in millimeters) from crack segmentation map. The crack segmentation map produced by the Attention R2U-Net architecture is combined into 1024×1024 size, then 1×100 pixel values are extracted from two locations to be given as input to a Random Forest regressor for predicting the crack width. During training, 1×100 pixels from two locations from each training image are extracted and given as input with the actual (manually) measured width measured. In the testing phase, 1×100 pixels from any location can be given as input and the network will predict the crack width in millimetres.

Table 1. Statistical summary of crack width distribution (in mm) used for training and testing the four regression models on three different concrete beams (C1, C2 and C3). Sample numbers include numbers from both locations 1 and 2 for each concrete specimen.

	Concrete	Location	Samples	Mean	Standard Deviation	Minimum	Median	Maximum
Train	C1	1, 2	80	2.44	2.78	0	1.58	8.95
	C2	1, 2	80	1.24	1.12	0	0.91	4.37
	C3	1, 2	80	1.23	0.99	0	1.12	3.80
Test	C1	1, 2	20	1.78	2.13	0	1.33	8.28
	C2	1, 2	20	1.20	1.35	0	0.81	3.89
	C3	1, 2	20	1.13	0.98	0	0.96	3.15

Table 2. Details of the dataset used for training, validation and testing the segmentation models. Details include the number of images and image sizes used from three different concrete beams (C1, C2 and C3) for training, validation and testing.

	Size	Total Images	C1	C2	C3
Train	256×256	$4253 = (2333^{35} + 1920)$	640	640	640
Validation	256×256	425			
Test	1024×1024	30	10	10	10

Table 3. Percentages of crack and non-crack pixels in training and test data.

	Crack pixels (%)	Non-crack pixels (%)
Train	1.40	98.60
Test	1.52	98.48

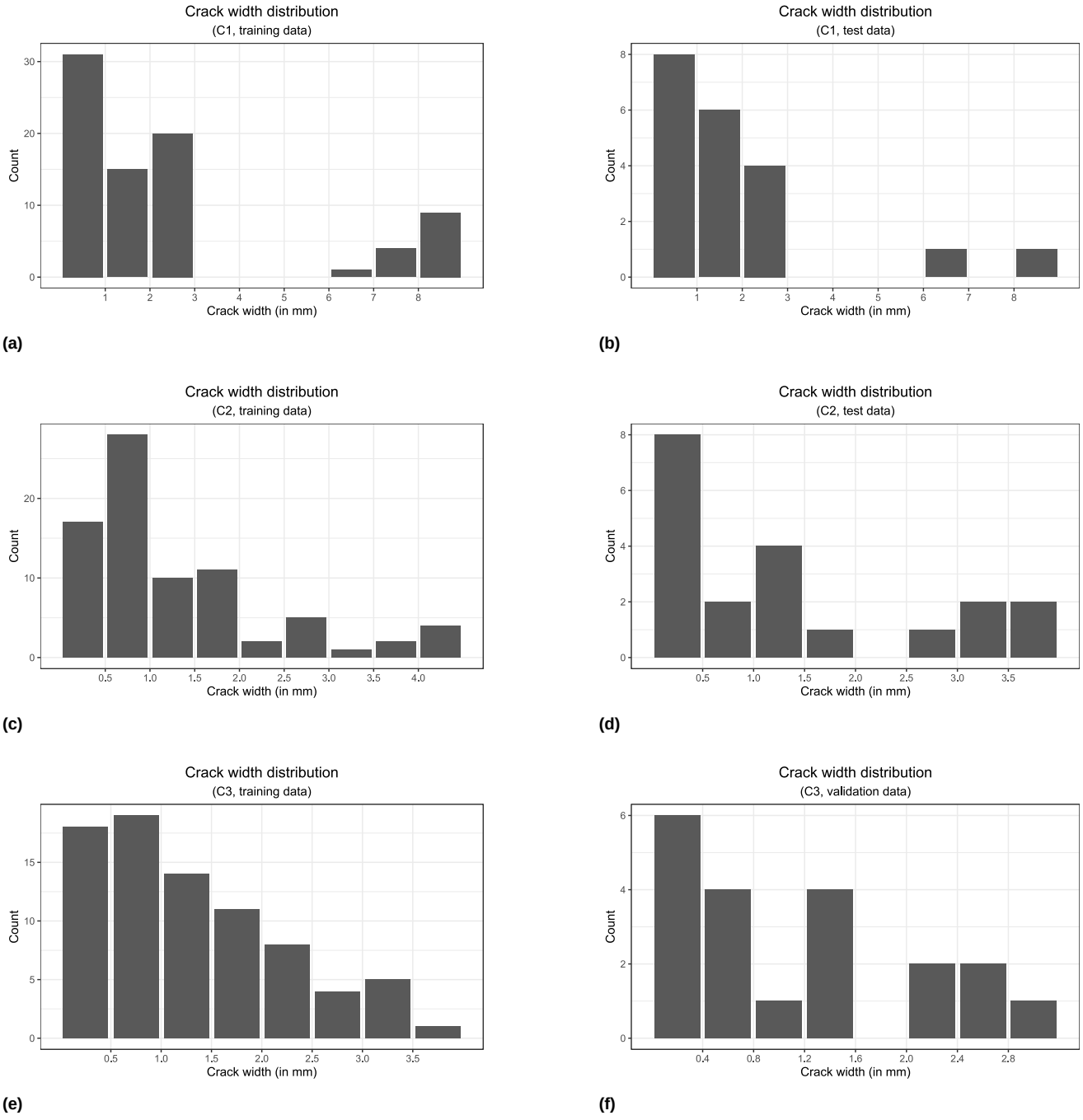

Figure 4. Crack width distribution used for training and validating regression models to predict crack width for three concrete surfaces (C1, C2 and C3).

Table 4. Performance comparison of the proposed Attention R2U-Net pixel-level crack detection with other models on test dataset. (R2: Recurrent Residual; PA: Pixel Accuracy; mPA: mean Pixel Accuracy; mIoU: mean Intersection over Union; FwIoU: Frequency-weighted IoU.)

Method	PA	mPA	mIoU	FwIoU
FCN-ResNet101 [41]	0.9366	0.9473	0.7924	0.8936
DeepLab-v3-Plus [54]	0.9461	**0.9613**	**0.8176**	0.9079
U-Net [49]	0.9366	0.9380	0.7900	0.8932
Attention U-Net [50]	0.9362	0.9212	0.7844	0.8917
Attention R2U-Net (proposed)	**0.9771**	0.9467	0.7299	**0.9638**

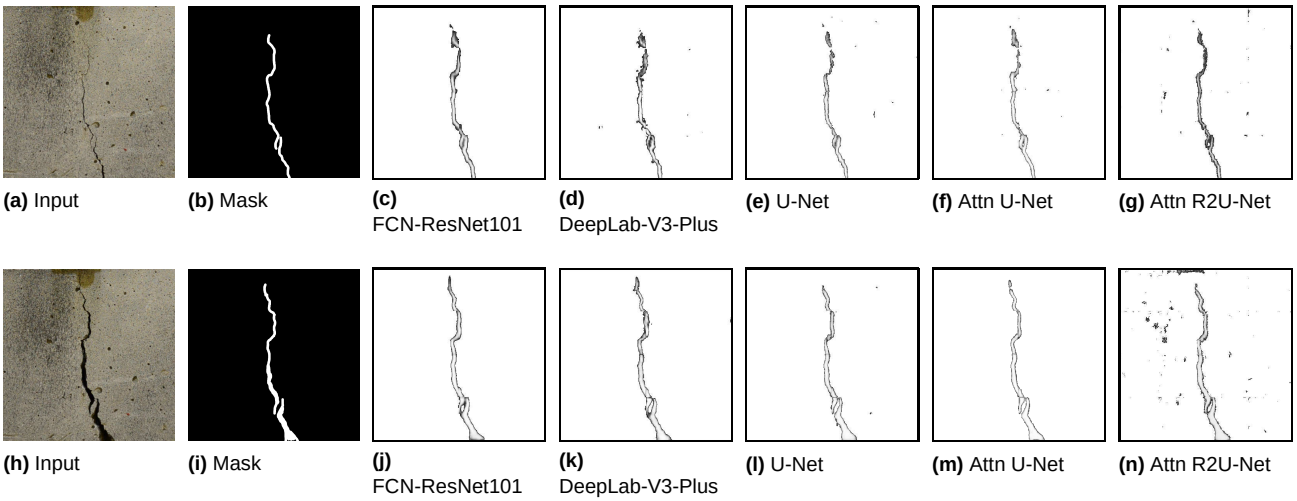

Figure 5. Comparison of pixel-level segmentation results for concrete specimen C1 for two different test images [(a): IMG_0008 and (h): IMG_0035]. Mask in the ground truth. Attention R2U-Net also detects small holes present in the images.

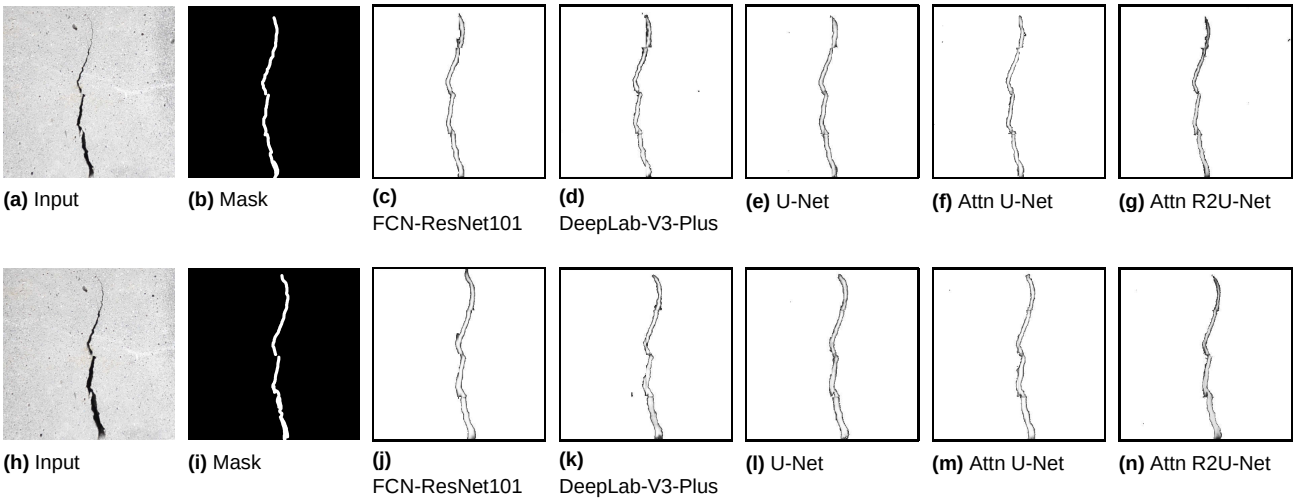

Figure 6. Comparison of pixel-level segmentation results for concrete specimen C2 for two different test images [(a): IMG_0026 and (h) IMG_0037]. Mask in the ground truth.

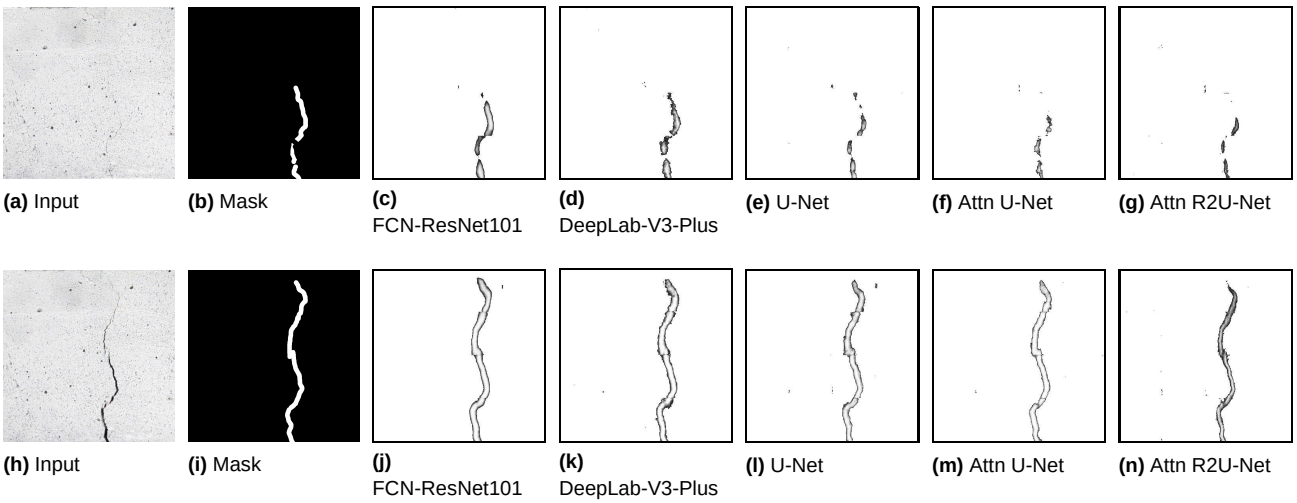

Figure 7. Comparison of pixel-level segmentation results for concrete specimen C3 for two different test images [(a): IMG_0001 and (h): IMG_0017]. Mask in the ground truth.

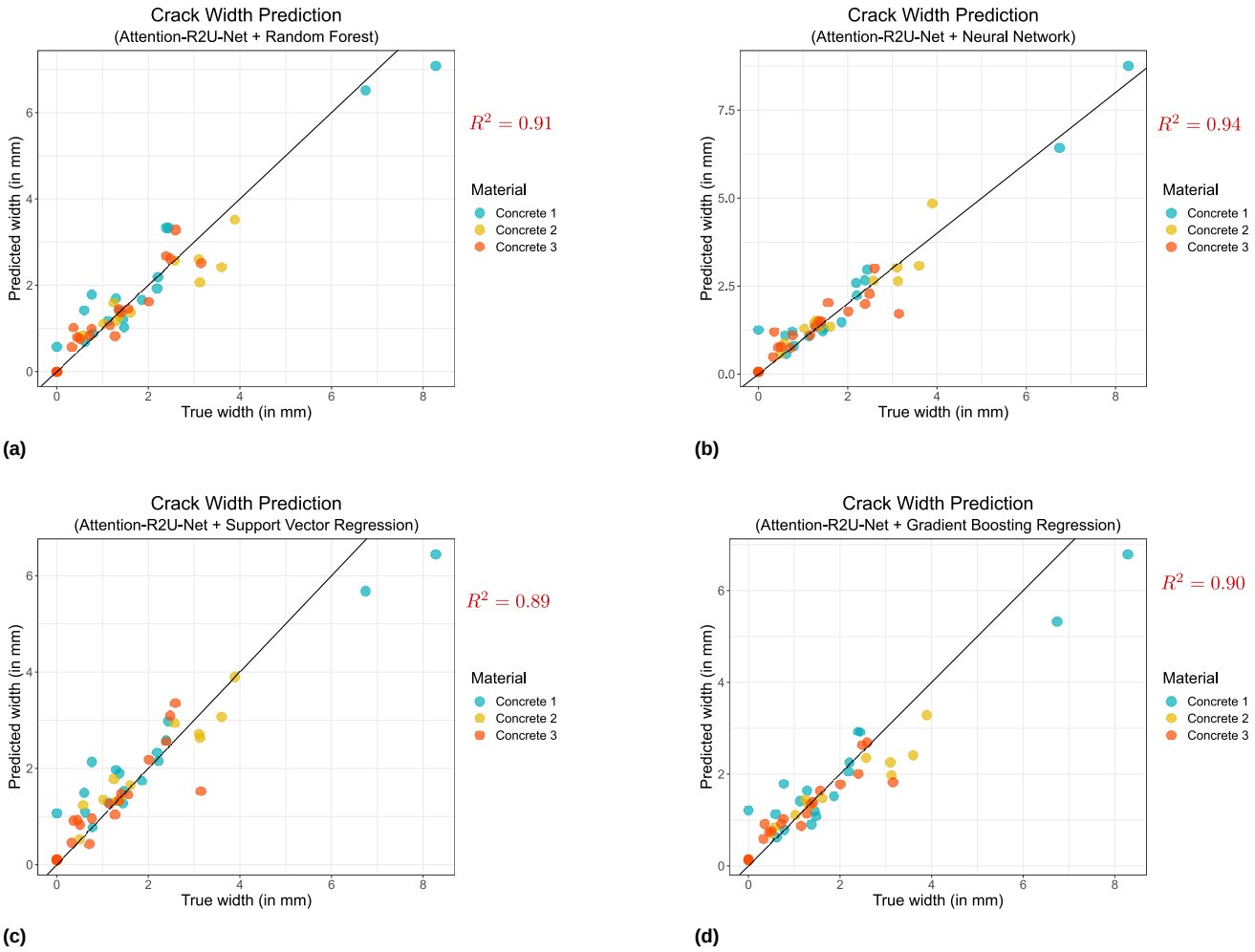

Figure 8. Crack width prediction grouped by concrete specimen type (C1, C2 and C3) for four regression models with Attention R2U-Net: (a) Attention R2U-Net + Random Forest ($R^2 = 0.91$), (b) Attention R2U-Net + Neural Network ($R^2 = 0.94$), (c) Attention R2U-Net + Support Vector Regression ($R^2 = 0.89$), and (d) Attention R2U-Net + Gradient Boosting Regression ($R^2 = 0.90$).

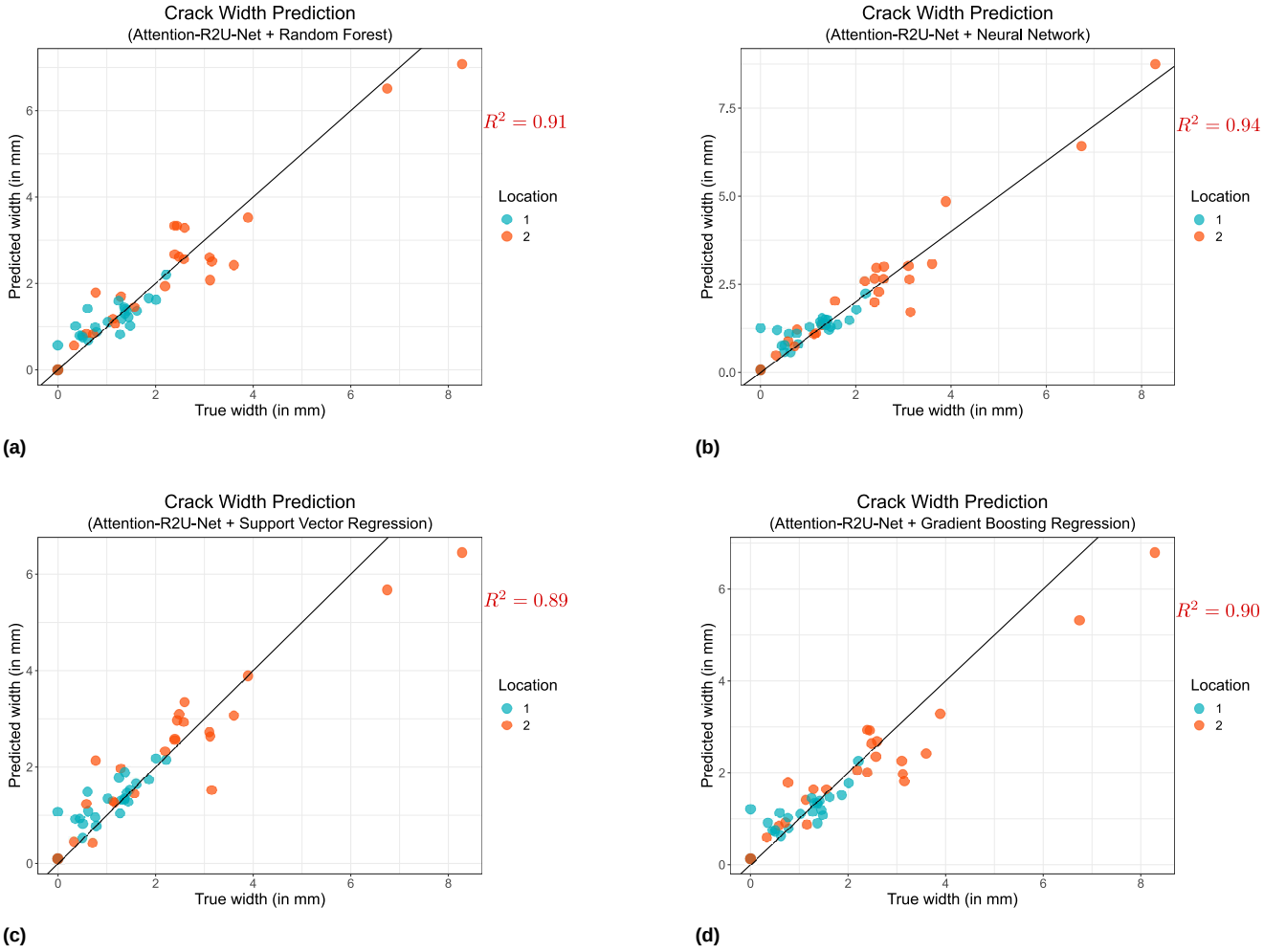

Figure 9. Crack width prediction grouped by two locations (1 and 2) for four regression models with Attention R2U-Net: (a) Attention R2U-Net + Random Forest ($R^2 = 0.91$), (b) Attention R2U-Net + Neural Network ($R^2 = 0.94$), (c) Attention R2U-Net + Support Vector Regression ($R^2 = 0.89$), and (d) Attention R2U-Net + Gradient Boosting Regression ($R^2 = 0.90$).

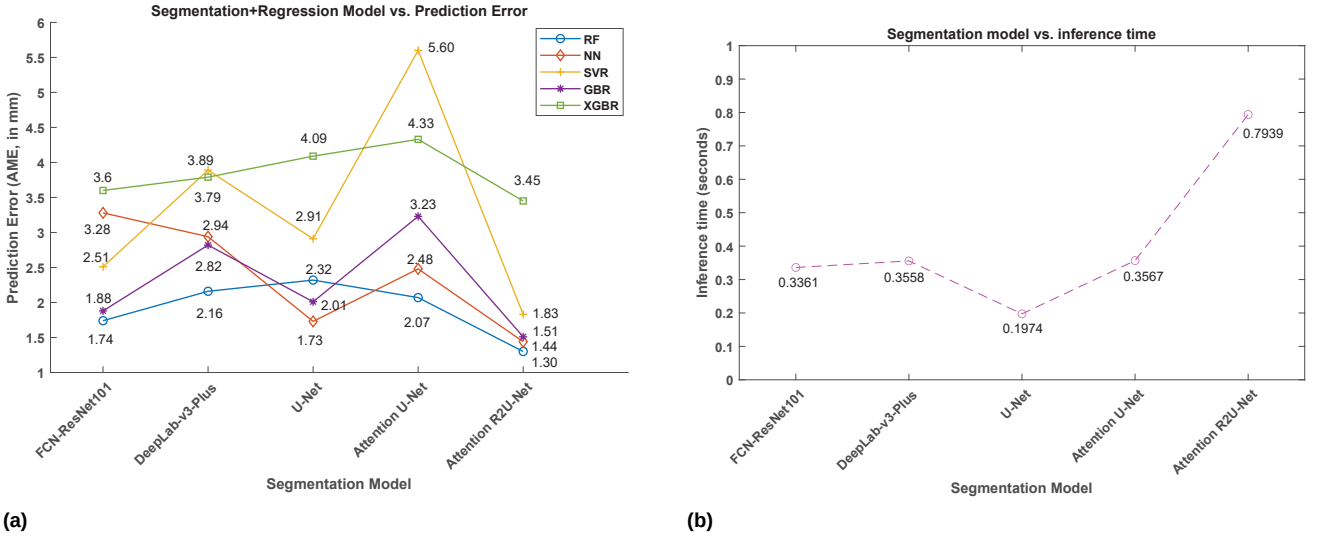

Figure 10. Comparison of (a) prediction error (AME, in mm) and (b) inference time (in seconds) per patch of size 256×256. The plot is based on average inference time calculated on test images using the trained segmentation models. There is a trade-off between achieving the lowest absolute maximum error (Attention R2U-Net) and the inference time.

Table 5. Comparison of crack width prediction models (segmentation combined with regression). Comparison include five segmentation models in combination with four regression models. We use Mean-Squared Error (MSE), Root Mean Squared Error (RMSE), R-Squared (R^2, coefficient of determination) and Absolute Maximum Error (AME) to evaluate the models. Attention R2U-Net segmentation with Neural Network regression model provides the lowest absolute maximum error. (RF: Random Forest; NN: Neural Network; SVR: Support Vector Regression; GBR: Gradient Boosting Regression; XGBR: Extreme Gradient Boosting Regression.)

Segmentation Model	**Regression Model**	**RMSE** (in mm)	**MAE** (in mm)	**R-squared**	**Absolute Maximum Error** (in mm)
FCN-ResNet101	RF	0.60	0.46	0.85	1.74
	NN	0.84	0.53	0.71	3.28
	SVR	0.69	0.46	0.80	2.51
	GBR	0.51	0.33	0.89	1.88
	XGBR	0.84	0.58	0.71	3.60
DeepLab-v3-Plus	RF	0.74	0.47	0.77	2.16
	NN	0.57	0.31	0.87	2.94
	SVR	0.90	0.52	0.66	3.89
	GBR	0.76	0.53	0.76	2.82
	XGBR	0.98	0.61	0.60	3.79
U-Net	RF	0.64	0.45	0.83	2.32
	NN	0.65	0.44	0.82	1.73
	SVR	0.82	0.53	0.72	2.91
	GBR	0.57	0.39	0.86	2.01
	XGBR	0.95	0.61	0.62	4.09
Attention U-Net	RF	0.60	0.39	0.85	2.07
	NN	0.79	0.51	0.74	2.48
	SVR	1.05	0.56	0.55	5.60
	GBR	0.72	0.45	0.78	3.23
	XGBR	1.02	0.66	0.57	4.33
Attention R2U-Net	RF	0.45	0.31	0.91	**1.30**
	NN	0.38	0.25	0.94	1.44
	SVR	0.51	0.33	0.89	1.83
	GBR	0.49	0.33	0.90	1.51
	XGBR	0.85	0.59	0.70	3.45

Table 6. Comparison of trainable segmentation model parameters. Regression models producing the lowest prediction error (AME, in mm) are included here from Table 5 for comparison. (AME: Absolute Maximum Error; RF: Random Forest; NN: Neural Network).

Model	**Parameters**	**Regression Model**	**Prediction error (in mm)**
FCN-ResNet101	51,938,881	RF	1.74
DeepLab-v3-Plus	58,748,833	RF	2.16
U-Net	13,395,329	NN	1.73
Attention U-Net	34,878,573	RF	2.07
Attention R2U-Net	39,442,925	RF	1.30

Table 7. Comparison of crack length for three different concrete specimen (C1, C2 and C3) against the ground truth. The extracted lengths using five segmentation models are measured using RMSE, MAE and R-squared error (which explains the total variation).

	RMSE (in mm)			**MAE** (in mm)			**R-squared**		
Model	**C1**	**C2**	**C3**	**C1**	**C2**	**C3**	**C1**	**C2**	**C3**
FCN-ResNet101	4.378	3.133	15.192	2.423	1.655	7.348	0.984	0.993	0.839
DeepLab-v3-Plus	5.416	7.907	9.440	2.813	3.297	4.908	0.977	0.955	0.935
U-Net	7.108	10.158	14.329	4.368	5.54	7.988	0.960	0.930	0.861
Attention U-Net	6.509	8.285	4.353	3.584	4.003	2.155	0.965	0.95	0.986
Attention R2U-Net	7.930	8.163	14.509	4.227	4.115	6.443	0.953	0.949	0.849