How Do Label Errors Affect Thin Crack Detection by DNNs

Liang Xu[1] Han Zou[1,2] Takayuki Okatani[1,2]
[1]Graduate School of Information Sciences, Tohoku University [2]RIKEN Center for AIP
{xu, hzou, okatani}@vision.is.tohoku.ac.jp

Abstract

Numerous studies have been conducted on detecting cracks in images of roads, surfaces of concrete and other materials, and so on. Accurate annotation of cracks is crucial for supervised learning, but identifying cracks in images, particularly thin cracks, can be challenging, making accurate annotation difficult. However, little is known about how annotation errors in training data affect the accuracy of detectors trained on them. This study attempts to address this gap by synthesizing annotation errors and analyzing their effects, which, to the authors' knowledge, has not been done before in the literature. This is made possible by employing an annotation method that labels cracks as curves with a single-pixel width alongside appropriate training and evaluation methods. We synthesized various types of annotation errors, including under and over-annotation errors caused by crack-like image structures and polyline approximation of crack curves, to reduce annotation costs. The experimental results reveal several important findings, such as that under-annotation is more harmful than over-annotation and that polyline approximation has a modest impact on detection accuracy.

1. Introduction

Detecting cracks emerging on the surface of an object from its images is an important problem of computer vision, as it has a wide range of applications. The standard approach employed in recent studies [22,46,50,51] is to formulate it as a semantic segmentation problem and train deep neural networks (DNNs) using a specific dataset created for the problem; this approach is reported to produce satisfactory results. Cracks can emerge on the surface of different materials, such as metal, wood, concrete, etc. As their image textures are diverse, it is necessary to create training data for each target, which is usually done manually.

However, Annotating cracks is a task that can be prone to errors. Even for human annotators, accurately identifying thin cracks is not always straightforward. Depending on the image acquisition quality, such as image resolution, sharpness, motion blur, illumination condition, etc., annotators often fail to correctly identify cracks, leading to annotation errors. When these errors are present in the crack annotations, they can serve as noisy labels for machine learning models including DNNs. Nevertheless, researchers have not paid much attention to how such annotation errors affect crack detection performance.

In this paper, we investigate the impact of annotation errors on crack detection by synthesizing these errors and training DNN models using the data with the synthesized errors. Subsequently, we test the models on test data with error-free annotations. Currently, crack detection is formulated as a binary segmentation task where the training data is labeled with pixel-wise annotations of crack and non-crack regions. However, modeling and synthesizing annotation errors that occur in practice using this method can be challenging. To address this, we use an alternative annotation method where only the center lines of cracks are labeled, making it easier to model annotation errors. Despite the change in annotation method, existing DNN models developed for the segmentation formulation can still be used when appropriate training and evaluation methods are employed.

To analyze the behavior of the model in detail, we synthesize three types of errors: feature-independent errors, feature-dependent errors, and errors resulting from approximating crack curves with polylines. Feature-dependent errors are modeled using image blur to simulate the effect of images captured from distant positions. Through experiments, we investigate the impact of these errors and their combinations on the accuracy of crack detection. Our findings offer best practices for annotating cracks to achieve more accurate detection. Notably, while our primary focus was on identifying cracks on concrete surfaces of bridges, the discussions and analyses presented in this paper can be applied to cracks emerging on other objects and materials.

2. Related Work

2.1. Crack Detection

The detection of cracks in images of roads, concrete surfaces, and other materials has been the subject of extensive research. Early studies relied on image processing techniques [18, 27], which required strong assumptions regarding the appearance of cracks, thereby limiting detection accuracy. Subsequent studies focused on developing learning-based methods [29, 35], which improved accuracy. However, these methods still fall short of the accuracy needed for practical applications.

Deep learning has significantly improved the performance of crack detection, as it has for many other visual recognition tasks. Several studies [7, 8, 20, 32, 39, 41, 48, 51] extract features from a patch and determine whether its center pixel is crack or not. Some studies [33, 47] have utilized transfer learning to classify pavement images into cracks and sealed-cracks, while others [28] have proposed a two-step approach that first segments the target area from the input image and then detects cracks within that area. Standard formulations of semantic segmentation, such as fully convolutional networks, have been adopted in other studies [4, 19]. Recent works following this approach include DeepCrack [22] and CrackFormer [21]. The accuracy of this approach largely depends on the availability of good datasets with accurate annotations, such as CrackTree260 [50], CrackLS315 [51], and Stone331 [16].

2.2. Dealing with Noisy Annotation

Training data for supervised learning of computer vision tasks are usually created by human annotators. It is inevitable that the data contain erroneous annotation. Some studies have focused on examining the impact of noisy annotation on model performance on image classification. One way to mitigate noisy annotation is by learning from label noise, as explored by [9]. Theoretical studies [1, 5, 10] have primarily assumed the presence of random classification noise, where labels are subject to random errors. In practice, a common approach is to identify potentially mislabeled examples [6] and then attempt to correct them. To address noise annotation, several studies have dealt with noisy annotations independently [23, 37, 43], employed loss functions [11, 49], weighted training samples [24, 30], discarded noisy samples [34], filtered out noisy samples [25], and used co-teaching [12]. Other approaches include joint optimization (i.e., inference) of network parameters, data augmentation [26], and label correction [38]. Furthermore, label noise has been studied from the perspective of analyzing when deep neural networks generalize to unseen inputs and when they memorize training samples [2, 45].

3. Methodology

3.1. Problem Formulation

3.1.1 Real-world Demands for Crack Detection

Previous studies formulate the problem of crack detection as binary image segmentation. Then, a machine-learning model performs binary classification on each image pixel, i.e., whether each pixel is crack or not. To train the model, a pixel-wise binary map with the same resolution as the input image is provided. Most existing datasets of crack detection provide such pixel-wise binary maps, created by pixel-wise annotation.

However, while it is overlooked in the computer vision community, real-world applications barely require predicting crack regions in pixel-level accuracy. Reflecting the fact that cracks have a linear structure (e.g., a curve or line segments) rather than a region, the first priority is *being able to predict the existence of cracks as curves, which may have an at most few-pixel width.* While it aligns with the formulation as a segmentation task, the conventional region-based annotation does not match the practical demands. It complicates the annotation work, leading to an increase in cost and errors. Hence, there is some gap between the practical requirement and the standard problem formulation of crack detection.

3.1.2 Curve-based Annotation

To address the gap mentioned earlier, we choose to annotate cracks as independent curves. This decision is critical to achieve the goal of this study, which is to analyze the impact of erroneous annotation. Using curve-based annotation allows us to create realistic annotation errors and analyze their effects on crack detection.

Despite the differences in annotation methods, we treat crack detection as a segmentation task, similar to existing studies. To examine the impact of erroneous annotation on detection performance, we use the current state-of-the-art crack detection methods and standard semantic segmentation methods.

Although it may appear to conflict with the problem formulation, curve-based annotation can provide excellent segmentation supervision when appropriate training methods are employed. For instance, we diffuse the ground-truth crack labels to occupy specific areas of the image rather than a single-pixel width. Additionally, we use appropriate metrics for evaluation. This approach is similar to human pose estimation, where key points, such as human body joints, are represented as single image points, and DNNs are trained to predict their likelihood maps.

3.1.3 Evaluation

Given the use of a new annotation method, standard pixel-wise IoU metrics, as used in previous studies to evaluate predicted crack "regions," are inadequate. The trained DNNs output a two-dimensional map of crack likelihoods, which, upon thresholding, yields a binary map consisting of predicted crack regions rather than curves. In contrast, the ground truth map represents cracks as one-pixel-width curves representing their center.

To address this gap, we use a morphological operation. Here, we need to classify all pixels in the predicted map as true positive (TP), true negative (TN), false positive (FP), or false negative (FN). To identify TPs and FPs, we dilate the ground-truth crack labels using a small disk. Predicted crack pixels inside the dilated region are counted as TPs, while pixels outside are counted as FPs.

This evaluation method prevents the penalization of predicted crack pixels that do not align precisely with the true crack. It also absorbs slight positional errors of predicted cracks; they are totally allowable in practice, as mentioned earlier. The radius of the dilating disk should desirably be the smallest that achieves these effects, to make it possible to distinguish two isolated cracks that are close to each other. (We set the radius to three pixels in our experiments.) We use the original one-pixel-width label when counting TNs and FNs.

3.1.4 Dataset

We employ the ThinCrack2019 dataset [40], a dataset providing curve-based annotation in our experiments; we also create additional annotation for our experiments; see Sec. 4.1.1 for more details.

Following existing studies [21, 51], we split the images in the dataset into smaller patches (512×512 pixels). From all the patches, we select those containing at least a single pixel with crack label and use only the selected patches to train and test DNN models. For brevity, we call them *positive patches* and all the remaining patches *negative patches*. So, we discard all the negative patches in experiments, similarly to previous studies [21,51]. This procedure is essential to address the severe class imbalance in the dataset, as only a small fraction of pixels are labeled as positive (i.e., crack), while the rest are labeled negative. This step is necessary to perform model evaluation computationally efficiently. Although models are required to classify all pixels belonging to negative patches as non-crack, we can accurately evaluate their ability using only positive patches, which mostly consist of non-crack pixels.

3.2. Types of Annotation Errors

There are several different causes for annotation errors. To investigate their impacts on detection accuracy, we classify possible annotation errors into three types: feature-independent, feature-dependent, and polyline annotation errors. We experimentally synthesize these types of errors and train DNN models on the training data with the synthetic errors. By testing the trained models on error-free test data, we experimentally examine the impact of these errors.

Feature-independent annotation error. This is the type of annotation errors that emerge purely randomly and are independent of image brightness. To simulate this type of errors, for a given crack label map, we eliminate some part of labeled cracks or add crack labels in a purely random fashion, where we do not use a paired image. Thus, such errors are easy to synthesize but may not represent real-world annotation errors accurately.

Feature-dependent annotation error. This is the type of errors that are caused by some image structures. That is, they will occur either when annotators fail to label a crack as crack because it does not look like a crack (e.g., too thin), or they erroneously label a non-crack image structure as crack because it looks like a crack, e.g., scratches, stains, etc. Such errors will represent most real-world errors more accurately.

Polyline annotation error. This type of errors occur when annotators do not provide the precise trace of a crack in images; for instance, they annotate cracks only roughly. We simulate such errors by approximate the track of a crack with a polyline, i.e., connected line segments.

Independently of the above categorization of errors, there is another property of annotation errors, which is under- and over-annotation. One is **over-annotation** if non-crack pixels are labeled as crack and the other is **under-annotation** if crack pixels are labeled as non-crack. Errors of the first two types, i.e., feature-independent and dependent, each contain over- and under-annotation.

3.3. Synthesizing Feature-independent Errors

To analyze and evaluate their effects, we synthetically generate feature-independent annotation errors in a 'patch-wise' manner, as follows. As explained above, we split images into patches, whose size is 512×512 pixel size. Each of them is named positive if it contains a crack pixel and negative if it contains no crack pixel.

We then create under-annotation errors by choosing patches randomly from the positive patches and removing all the crack labels from each of them; see Fig. 1(c). As a result, these patches contain cracks and are nevertheless treated negative patches. We create over-annotation errors by choosing pairs of patches randomly from the positive patches and copy-paste the crack labels in the first patch to the second one; see Fig. 1(d). We keep the original crack labels in both patches in each pair.

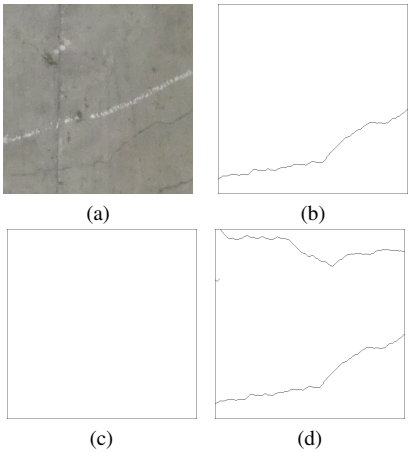


Figure 1. Examples of input patches and feature-independent annotation errors. (a) Input patch. (b) Ground-truth annotation. (c) Under-annotation error. (d) Over-annotation error.

3.4. Synthesizing Feature-dependent Errors

We define feature-dependent errors as annotation errors that occur when either crack-like image structures are incorrectly labeled as cracks (false positive), or actual cracks are not identified as such (false negative). In this study, we specifically focus on errors that arise from cracks with widths too narrow to be discernible to the human eye. To synthesize such errors, we intentionally blur the images, using a combination of down-sampling and up-sampling with identical factors. We choose this method as it offers a simple yet effective simulation of image acquisition from a greater distance than that of the original input image. Further details are provided below.

Let $I_{1\times}$ and $\overline{y}_{1\times}$ be an image patch and its correct label map, respectively; see Fig. 2(a). We down-sample $I_{1\times}$ by the factor of 4 and then up-sample the down-sampled image by the same factor using Lanczos interpolation. We denote the resulting image by $I_{4\times}$, which has the same resolution as $I_{1\times}$; see Fig. 2(b). Due to the series of down- and up-sampling operation, $I_{4\times}$ has blurry appearance compared with $I_{1\times}$. We carefully annotate the crack pixels of $I_{4\times}$, yielding its label map $\overline{y}_{4\times}$. There is often fluctuation in crack widths; thicker cracks survive the down-/up-sampling operation, whereas thinner cracks do not. The down/up-sampling make some parts of cracks seen in $I_{1\times}$ disappear in $I_{4\times}$. We apply similar down- and up-sampling operation but by the factor of 8 to $I_{1\times}$, obtaining more blurry image $I_{8\times}$, for which we carefully annotate its crack pixels to obtain $\overline{y}_{8\times}$

Finally, we choose $I_{4\times}$ as an input and use $\overline{y}_{4\times}$ as its correct crack annotation. We use $\overline{y}_{1\times}$ as over-annotation and $\overline{y}_{8\times}$ as under-annotation.

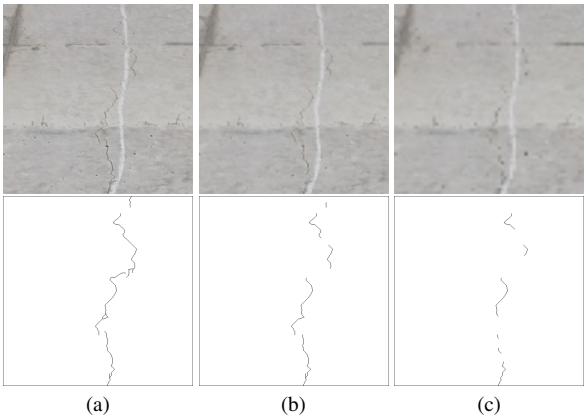


Figure 2. Example of generated feature-dependent annotation errors. (a) Original patch $I_{1\times}$ and its manual annotation $\overline{y}_{1\times}$. (b) Blurred patch $I_{4\times}$ by applying down-sampling and up-sampling to $I_{1\times}$ in turn, with the same factor of 4, along with its manual annotation $\overline{y}_{4\times}$. (c) Further blurred patch $I_{8\times}$ with the down-/up-sampling factor of 8 and its manual annotation $\overline{y}_{8\times}$. We use $I_{4\times}$ as an input and $\overline{y}_{1\times}$, $\overline{y}_{4\times}$, and $\overline{y}_{8\times}$ as under-, correct, and over-annotations for it, respectively.

3.5. Polyline Annotation

A polyline is defined as a collection of straight lines that connect a series of ordered points. Using polylines to annotate cracks can reduce the cost of annotation compared to precisely tracing their centers. Assuming that accurate crack labels are provided, we generate polyline annotations by approximating the original labels with polylines. To achieve this, we use the Douglas-Peucker algorithm [14], which has a threshold that allows us to adjust the level of approximation. This threshold controls the degree of "roughness" of the polyline annotation. Examples of approximated polylines are shown in Fig. 3. The compression rates are the ratio of polyline points to the number of pixels in the original crack labels.

4. Experiments

4.1. Experimental Configuration

4.1.1 Dataset

The goal is to investigate the effects of imperfect annotation on crack detection using synthetic erroneous annotation of several types described above. Then, we first need data with a sufficiently accurate annotation of cracks. We use the ThinCrack2019 dataset [40], which contains 352 images of concrete surfaces of various bridges captured by hand-held cameras. Their sizes are in the range from $4,000 \times 3,000$ to $5,000 \times 4,000$ pixels, and each image is annotated by a semi-automatic method with careful manual supervision, which produces accurate crack labels. A

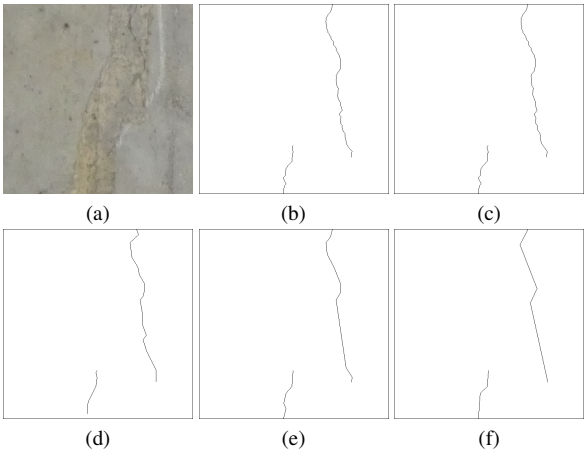


Figure 3. Examples of polyline approximation with different parameters. (a) Original patch. (b) Ground-truth annotation. (c) Polyline approximation with compress rate $= 25\%$. (d) 5%. (d) 3%. (e) 1%. Best viewed on a PC monitor.

unique feature of this dataset is that cracks are annotated with curves with one-pixel width that trace the center of cracks, which matches the purpose of this study.

As explained above, we split each image into 512×512 pixel patches, yielding positive patches containing at least a single pixel annotated as crack and negative patches without crack pixel annotation. We have 2,341 and 479 positive patches for training and testing, respectively.

We generate noisy annotation using the methods explained in Sec. 3.3 and Sec. 3.4. For each type of annotation errors, we synthesize errors with different error rates $\widetilde{\mathcal{R}} \in \{10\%, 20\%, \cdots, 50\%\}$ in a patch-wise fashion. Specifically, we randomly choose positive patches from the training set with a probability $\widetilde{\mathcal{R}}$ and apply each error-synthesis method to them, replacing the original patches with the new ones. We then use the updated training set to train DNN models. To obtain feature-dependent annotation errors, we apply the smoothing by the combination of down- and up-sampling with the factor of four and eight pixels.

We then evaluate the detection accuracy of a DNN by training it with the updated training sets with a different type and level of annotation errors. For each experiment, we iterate the procedure from the updating the training set to training a model for three times, reporting the average accuracy.

4.1.2 Evaluated Methods

To examine if different DNNs are affected differently by annotation errors, we tested four models for crack detection. Two are models for semantic segmentation and the other two are the state-of-the-art methods for crack detection.

UNet [31] is a classical encoder-decoder CNN and is employed as a baseline models in our experiments. We use a variant of the standard UNet model [31] with 1/4 channels in each convolutional layer and a BN layer after every convolutional layer, before activation. We confirmed through preliminary experiments that this light-weight UNet works even better than the original one despite its computational efficiency.

HRNet-W18-C [42] is a lightweight version of HRNetV2 [36, 44] designed for semantic segmentation. These are transformer-based networks proven to attain a good trade-off between segmentation accuracy and computational cost.

DeepCrack [22] is a method developed for crack detection that uses a fully convolutional network (FCN). Aiming at learning multi-scale and multi-level features, direct supervision is applied to each stage in the hierarchy at training time. To improve detection accuracy, it applies guided filtering [13] and conditional random fields (CRFs) [17] to the network's output.

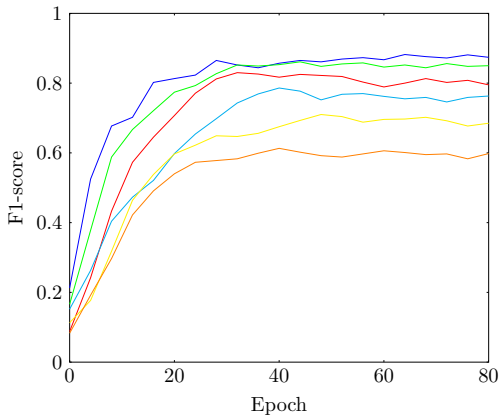
CrackFormer [21] is a network designed for crack detection and has a transformer-based encoder–decoder structure. The design is similar to SegNet [3], which proposes a self-attention block and scaling-attention block for fine-grained crack detection.

4.1.3 Training and Evaluation

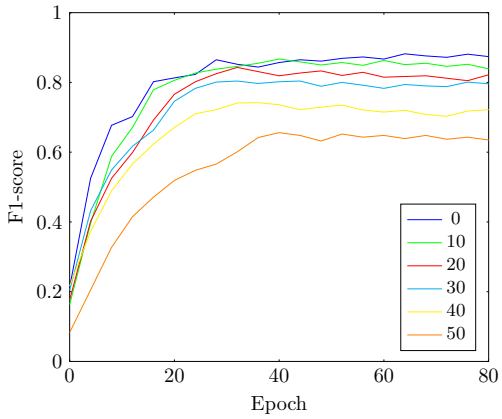
We employ the same training procedure for all the methods (i.e., DNNs) to equalize possible impacts of differences in the procedure[1]. The crack labels are given as binary images with the same sizes as input images. Cracks have the shape of curves with one-pixel width in the maps. We apply Gaussian blur with the sigma $= 3$ pixels to the map to spatially smooth the labels. We use the mean squared error (MSE) loss for all experiments, i.e., the average of the squared difference between the ground-truth value and its prediction, both in the range $[0, 1]$, at pixels, as we confirmed this works better than a cross-entropy loss in our preliminary experiments. We apply data augmentation of random 360-degree rotation and random flipping to the training patches. Specifically, we train each model for 80 epochs using the Adam optimizer [15]; We set the initial learning rate to 10^{-3}, and drop it to 10^{-4} after 30 epochs, then further drop it to 10^{-5} after 50 epochs. Due to the limitation of memory size, mini-batch size is set to 16 for UNet, 8 for CrackFormer, and 4 for HRNet and DeepCrack, respectively.

To evalute the methods' performance, we utilize the method described in Sec. 3.1.3. Initially, we applied a

[1]This means the procedure partially differs from that employed in the original paper of each method, leading to possibly worse performance than reported in the paper. However, it is not so important here since our primary objective is to measure the impact of annotation errors on each individual method.



(a) Under-annotation



(b) Over-annotation

Figure 4. Crack detection accuracy vs. training epochs for **feature-independent** annotation errors. UNet is trained on data containing feature-independent under-/over-annotation errors with different rates $\widetilde{\mathcal{R}} = 0, \ldots, 50[\%]$ and evaluated on error-free test data.

threshold of 0.5 to the predicted map generated by each method, which contained pixel-wise probabilities of cracks. This converts it into a binary map. To calculate true positives (TPs) and false positives (FPs) of the predicted cracks, we dilated the ground truth crack labels using a small disk. We set the radius of dilation to three pixels, taking into account the minimum distance between two independent cracks. The primary metric used was the F1-score.

4.2. Effects of Feature-independent Errors

We first evaluate how feature-independent annotation errors affect detection accuracy. After training the UNet model on the training data with synthetic errors, we evaluate its performance on error-free images from the test set.

Figure 4 shows the results, i.e., how F1-score changes with training epochs for different levels of errors ranging from $\widetilde{\mathcal{R}} = 0\%$ (error-free) to 50%. It is observed that the detection performance deteriorates with increasing levels of errors, but the impact is modest and tolerable with low error levels. Comparing under- and over-annotation errors, we can state that the impact is smaller for over-annotation; the impact of 30% error for over-annotation (Fig. 4(b)) is mostly the same as that of 20% error for under-annotation (Fig. 4(a)).

Table 1. F1-scores on error-free test images of different DNN models trained with training data having different levels of annotation errors.

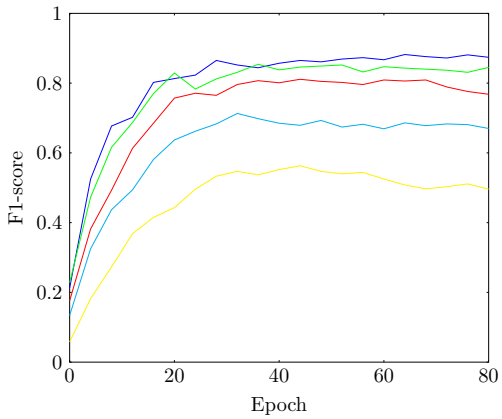
Error	Clean	Under-annotation			
level	0	10	20	30	40
HRNet	0.950	0.937	0.921	0.894	0.859
CrackFormer	0.929	0.914	0.894	0.861	0.817
DeepCrack	0.903	0.885	0.859	0.821	0.762
UNet	0.882	0.861	0.830	0.786	0.710
Error	Clean	Over-annotation			
level	0	10	20	30	40
HRNet	0.950	0.943	0.932	0.915	0.884
CrackFormer	0.929	0.921	0.907	0.885	0.846
DeepCrack	0.903	0.891	0.872	0.841	0.789
UNet	0.882	0.867	0.843	0.804	0.742

We also report the F1-scores of compared models in Table 1. Several observations can be made. Firstly, HRNet achieves relatively better performance than other models. Specifically, when training with errror-free labels, HRNet achieves 0.950, whereas CrackFormer, DeepCrack, and UNet yield lower scores by 2.2%, 4.9%, and 7.2%, respectively. Secondly, better models seem more robust to annotation errors. For instance, 40% under-annotation errors result in 9.6%, 12.1%, 15.6%, and 19.5% decreases of F1-score for HRNet, CrackFromer, DeepCrack, and UNet, respectively, compared to the error-free cases. Additionally, 40% over-annotation errors result in 6.9%, 8.9%, 12.6%, and 15.8% decreases for HRNet, CrackFromer, DeepCrack, and UNet, respectively, compared to the error-free cases. Finally, the aforementioned tendency that over-annotation has smaller impact on detection accuracy than under-annotation hold for these models as well.

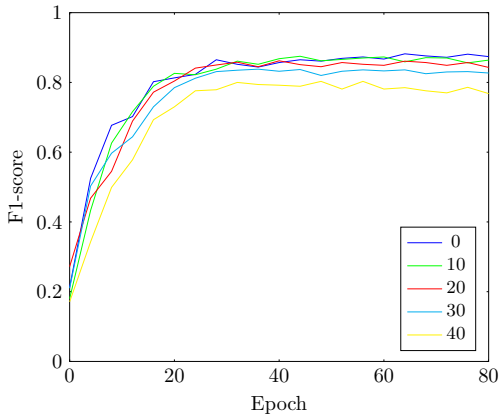
4.3. Effects of Feature-dependent Errors

We then examine how feature-dependent annotation errors affect detection accuracy. Similar to the above, we train models usimg training data with synthetic errors and test them on error-free test data.

Figure 5 shows the results. It is first observed that under-annotation has a greater negative impact than over-annotation. This is consistent with the feature-independent case, but the difference is more pronounced. For instance, if there are 20% under-annotation errors, their impact is similar to that of 40% over-annotation errors. Table 2 shows the results of the four compared methods. Our above observation holds true for the other three models as



(a) Under-annotation



(b) Over-annotation

Figure 5. Crack detection accuracy vs. training epochs for **feature-dependent** annotation erorrs. UNet is trained on data containing feature-dependent under-/over-annotation errors with different rates $\widetilde{\mathcal{R}} = 0, \ldots, 50[\%]$ and evaluated on error-free test data.

well. Specifically, when compared to error-free cases, 40% under-annotation errors lead to a considerable reduction in F1-scores for HRNet, CrackFormer, DeepCrack, and UNet, with a reduction of 17.3%, 21.5%, 29.5%, and 36.2%, respectively. In contrast, 40% over-annotation errors only result in a minor decrease of 3.9%, 5.1%, 7.3%, and 9.0%, respectively.

When comparing models, the same observations as those made for feature-independent errors can be made. Specifically, models that demonstrate higher accuracy with error-free training data exhibit better robustness with the increase in annotation errors. For instance, in our experimental setting, HRNet trained with 40% over-annotation data outperforms DeepCrack trained with error-free data.

Table 2. F1-scores achieved by diverse DNN models trained with various types of feature-dependent noisy labels under different error levels

Error level	Clean	Under-annotation			
	0	10	20	30	40
HRNet	0.950	0.935	0.889	0.848	0.785
CrackFormer	0.929	0.911	0.884	0.832	0.729
DeepCrack	0.903	0.879	0.843	0.764	0.637
UNet	0.882	0.854	0.811	0.713	0.563
Error level	Clean	Over-annotation			
	0	10	20	30	40
HRNet	0.950	0.948	0.943	0.931	0.914
CrackFormer	0.929	0.924	0.916	0.901	0.882
DeepCrack	0.903	0.897	0.886	0.864	0.837
UNet	0.882	0.875	0.862	0.836	0.803

4.4. Effects of Mix of Under- and Over-annotation

So far, we have only considered input images with either under-annotation or over-annotation errors. However, in practical scenarios, an image will have both types of errors. Therefore, we investigate the impact of mixed errors involving under- and over-annotation. We maintain a fixed total error level $\widetilde{\mathcal{R}}$ of 30% or 40%, and vary the ratio of under- and over-annotation errors, such as under/over $30/0$, $20/10$, $15/15$, $10/20$, and $0/30$. We evaluate the performance of our baseline UNet separately for feature-dependent and feature-independent error types.

Tables 3 and 4 present the results of feature-dependent and feature-independent errors, respectively, at total error levels of 30% and 40%. Several observations can be drawn from these results. Firstly, in the case of feature-independent errors, the ratio of under- and over-annotation has no significant impact on detection accuracy, while the total error level is the main factor. However, in the case of feature-dependent errors, the ratio of under- and over-annotation has a significant impact. Combining both types of errors leads to better performance than using only one type. For example, at a total error level of 30%, the ratio 10/20 achieves better performance (0.854) than 0/30 (0.836) and significantly outperforms 30/0 (0.713). This finding suggests that a combination of under- and over-annotation errors is likely to improve accuracy, which is good news as it is more common to encounter a mixture of the two types of errors rather than just one type. Additionally, it is crucial for annotators to ensure that under-annotation errors occur less frequently than over-annotation errors, which aligns with the aforementioned observation.

4.5. Effects of Polyline Annotation

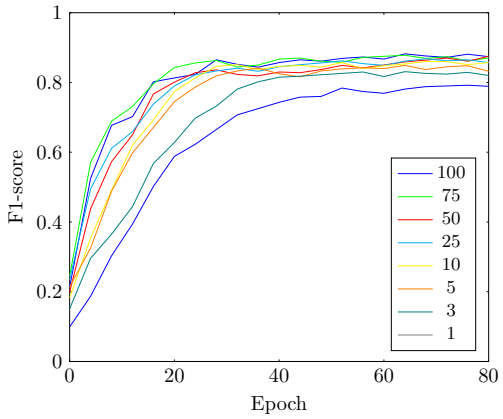
We synthesized polyline annotations using the method described in Sec. 3.5, and subsequently trained four DNN models using the synthesized training data. Figure 6 shows the F1-score of the UNet model trained on the polyline data with different compression rates and then tested on error-

Table 3. F1-scores of the UNet model trained with data having different ratios of under- and over-annotation errors, $\widetilde{\mathcal{R}}_u/\widetilde{\mathcal{R}}_o$, whose total error level is 30%.

Combination($\widetilde{\mathcal{R}}_u/\widetilde{\mathcal{R}}_o$)	30/0	20/10	15/15	10/20	0/30
Independent	0.786	0.793	0.797	0.800	0.804
Dependent	0.713	0.786	0.842	0.854	0.836

Table 4. F1-scores of the UNet model trained with data having different ratios of under- and over-annotation errors, $\widetilde{\mathcal{R}}_u/\widetilde{\mathcal{R}}_o$, whose total error level is 40%.

Combination(Ru/Ro)	40/0	30/10	20/20	10/30	0/40
Independent	0.710	0.721	0.728	0.734	0.742
Dependent	0.563	0.759	0.810	0.829	0.803



(a) Polyline annotation (Douglas-Peucker).

Figure 6. Crack detection accuracy vs. training epochs for **polyline annotation**. UNet is trained on data annotated as polylines with different compression rates, $1, \ldots, 100[\%]$, and evaluated on error-free test data.

free test data. A compression rate is the ratio of polyline points to the number of pixels in the original crack labels. We varied the parameter in the Douglas-Peucker algorithm to change the rate in the range from 1% to 100%. Table 5 reports the F1-scores of all four models.

It is observed from the results that the accuracy of the models decreases proportionally to the compression rate; their ranking remains unchanged regardless of the compression rate. It should be noted that even at a compression rate of 25%, all the models show only a modest decrease in accuracy. This indicates that annotating cracks with polylines is an effective way to reduce annotation cost while maintaining the accuracy of the models' crack detection.

5. Summary and Conclusions

In this study, we have considered the accuracy of deep learning methods for detecting cracks in images of various objects/materials. The accuracy of supervised learning methods largely depends on the quality of the training data. However, annotating cracks accurately, especially thin ones, can be challenging, leading to some degree of annotation errors. The potential impact of these errors on training data is not well understood.

Table 5. F1-scores achieved by the four methods trained on polyline annotation with different compression rates.

Comp. rate(%)	HRNet	CrackFormer	DeepCrack	UNet
100	0.950	0.929	0.903	0.882
75	0.944	0.925	0.898	0.878
50	0.938	0.919	0.892	0.875
25	0.931	0.912	0.884	0.868
10	0.920	0.903	0.877	0.860
5	0.908	0.895	0.868	0.849
3	0.897	0.883	0.854	0.831
1	0.883	0.872	0.826	0.792

To gain a better understanding, we synthesized annotation errors and assessed the impact of these errors on the accuracy of deep neural networks trained on the data. Crack detection is currently formulated as a binary segmentation task, with the training data labeled in pixel-wise labels of crack/non-crack. However, modeling and synthesizing annotation errors that occur in practice using this annotation method can be challenging. Therefore, we used an alternative annotation method where only the center lines of cracks were labeled, making it easier to model annotation errors.

We assessed three types of errors that affect crack detection accuracy: feature-independent errors, feature-dependent errors, and errors resulting from approximating crack curves with polylines. To model feature-dependent errors, we simulated the effect of image blur, which can occur when images are captured from a distance. Our experiments aimed to investigate how these errors, individually and in combination, impact the accuracy of crack detection.

Our findings from the experiments can be summarized as follows. Firstly, we found that under-annotation has a more significant negative impact than over-annotation, regardless of feature-dependent or feature-independent errors, as well as differences in DNN models. This implies that when annotating cracks, it is crucial for annotators to ensure that there are fewer under-annotation errors than over-annotation errors. Secondly, a combination of under- and over-annotation errors tend to improve accuracy. Specifically, for the same amount of error, the accuracy of the result is better when both types of errors are present rather than just one. Thirdly, models that demonstrate higher accuracy are also more robust to annotation errors. Finally, we found that polyline annotation is an effective way to reduce annotation costs while maintaining model detection accuracy.

Acknowledgments: This work was partly supported by JSPS KAKENHI Grant Number 20H05952 and 19H01110.

References

[1] Dana Angluin and Philip Laird. Learning from noisy examples. *Machine Learning*, 2:343–370, 1988. 2

[2] Devansh Arpit, Stanisław Jastrzębski, Nicolas Ballas, David Krueger, Emmanuel Bengio, Maxinder S. Kanwal, Tegan Maharaj, Asja Fischer, Aaron Courville, Yoshua Bengio, and Simon Lacoste-Julien. A closer look at memorization in deep networks. In *Proceedings of the 34th International Conference on Machine Learning*, 2017. 2

[3] V. Badrinarayanan, A. Kendall, and R. Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(12):2481–2495, 2017. 5

[4] Seongdeok Bang, Somin Park, Hongjo Kim, and Hyoungkwan Kim. Encoder-decoder network for pixel-level road crack detection in black-box images. *Computer-Aided Civil and Infrastructure Engineering*, 02 2019. 2

[5] Avrim Blum, Adam Kalai, and Hal Wasserman. Noise-tolerant learning, the parity problem, and the statistical query model. *Journal of the ACM (JACM)*, 50(4):506–519, 2003. 2

[6] Carla E Brodley and Mark A Friedl. Identifying mislabeled training data. *Journal of artificial intelligence research*, 11:131–167, 1999. 2

[7] Young-Jin Cha, Wooram Choi, and Oral Buyukozturk. Deep learning-based crack damage detection using convolutional neural networks. *Computer-Aided Civil and Infrastructure Engineering*, 32:361–378, 03 2017. 2

[8] Zhun Fan, Yuming Wu, Jiewei Lu, and Wenji Li. Automatic pavement crack detection based on structured prediction with the convolutional neural network. *SoICT 2018: Proceedings of the Ninth International Symposium on Information and Communication Technology*, pages 251–256, 11 2018. 2

[9] Benoît Frénay and Michel Verleysen. Classification in the presence of label noise: a survey. *IEEE transactions on neural networks and learning systems*, 25(5):845–869, 2013. 2

[10] Wei Gao, Lu Wang, Zhi-Hua Zhou, et al. Risk minimization in the presence of label noise. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30, 2016. 2

[11] Aritra Ghosh, Himanshu Kumar, and P. Sastry. Robust loss functions under label noise for deep neural networks. *AAAI*, 2017. 2

[12] Bo Han, Quanming Yao, Xingrui Yu, Gang Niu, Miao Xu, Weihua Hu, Ivor Tsang, and Masashi Sugiyama. Co-teaching: Robust training of deep neural networks with extremely noisy labels. In *NeurIPS*, 2018. 2

[13] Kaiming He, Jian Sun, and Xiaoou Tang. Guided image filtering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(6):1397–1409, 2013. 5

[14] John Hershberger and Jack Snoeyink. Speeding up the douglas-peucker line-simplification algorithm. In *Proc. 5th Intl. Symp. on Spatial Data Handling*, pages 134–143, 1992. 4

[15] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations (ICLR)*, 2015. 5

[16] Jacob König, Mark David Jenkins, Mike Mannion, Peter Barrie, and Gordon Morison. Optimized deep encoder-decoder methods for crack segmentation. *Digital Signal Processing*, 108:102907, 2021. 2

[17] John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 282–289. Morgan Kaufmann Publishers Inc., 2001. 5

[18] H. Li, D. Song, Y. Liu, and B. Li. Automatic pavement crack detection by multi-scale image fusion. *IEEE Transactions on Intelligent Transportation Systems*, 20(6):2025–2036, 2019. 2

[19] Qing-tong Li and Dong-ming Zhang. Deep learning based image recognition for crack and leakage defects of metro shield tunnel. *Tunnelling and Underground Space Technology*, 77:166–176, 07 2018. 2

[20] Yundong Li, Hongguang Li, and Hongren Wang. Pixel-wise crack detection using deep local pattern predictor for robot application. *Sensors*, 18:3042, 09 2018. 2

[21] Huajun Liu, Xiangyu Miao, Christoph Mertz, Chengzhong Xu, and Hui Kong. Crackformer: Transformer network for fine-grained crack detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3783–3792, 2021. 2, 3, 5

[22] Yahui Liu, Jian Yao, X. Lu, Renping Xie, and L. Li. Deepcrack: A deep hierarchical feature learning architecture for crack segmentation. *Neurocomputing*, 2019. 1, 2, 5

[23] Z. Lu, Z. Fu, T. Xiang, P. Han, L. Wang, and X. Gao. Learning from weak and noisy labels for semantic segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017. 2

[24] Zahra Mirikharaji, Yiqi Yan, and Ghassan Hamarneh. Learning to segment skin lesions from noisy annotations. In Qian Wang, Fausto Milletari, Hien V. Nguyen, Shadi Albarqouni, M. Jorge Cardoso, Nicola Rieke, Ziyue Xu, Konstantinos Kamnitsas, Vishal Patel, Badri Roysam, Steve Jiang, Kevin Zhou, Khoa Luu, and Ngan Le, editors, *Domain Adaptation and Representation Transfer and Medical Image Learning with Less Labels and Imperfect Data*. Springer International Publishing, 2019. 2

[25] Duc Tam Nguyen, Chaithanya Kumar Mummadi, Thi Phuong Nhung Ngo, Thi Hoai Phuong Nguyen, Laura Beggel, and Thomas Brox. Self: Learning to filter noisy labels with self-ensembling. In *International Conference on Learning Representations*, 2020. 2

[26] Kento Nishi, Yi Ding, Alex Rich, and Tobias Hollerer. Augmentation strategies for learning with noisy labels. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8022–8031, 2021. 2

[27] Henrique Oliveira and Paulo Correia. Crackit-an image processing toolbox for crack detection and characterization. *IEEE ICIP*, pages 798–802, 01 2014. 2

[28] Somin Park, Seongdeok Bang, Hongjo Kam, and Hyoungkwan Kim. Patch-based crack detection in black box images using convolutional neural networks. *Journal of Computing in Civil Engineering*, 33(3), May 2019. 2

[29] P. Prasanna, K. J. Dana, N. Gucunski, B. B. Basily, H. M. La, R. S. Lim, and H. Parvardeh. Automated crack detection on concrete bridges. *IEEE Transactions on Automation Science and Engineering*, 13(2):591–599, 2016. 2

[30] Mengye Ren, Wenyuan Zeng, Bin Yang, and Raquel Urtasun. Learning to reweight examples for robust deep learning. In *Proceedings of the 35th International Conference on Machine Learning*, 2018. 2

[31] O. Ronneberger, P.Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, 2015. 5

[32] S. J. Schmugge, L. Rice, N. R. Nguyen, J. Lindberg, R. Grizzi, C. Joffe, and M. C. Shin. Detection of cracks in nuclear power plant using spatial-temporal grouping of local patches. In *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1–7, 2016. 2

[33] S. J. Schmugge, L. Rice, N. R. Nguyen, J. Lindberg, R. Grizzi, C. Joffe, and M. C. Shin. Detection of cracks in nuclear power plant using spatial-temporal grouping of local patches. In *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1–7, 2016. 2

[34] Yanyao Shen and Sujay Sanghavi. Learning with bad training data via iterative trimmed loss minimization. In *Proceedings of the 36th International Conference on Machine Learning*, 2019. 2

[35] Y. Shi, L. Cui, Z. Qi, F. Meng, and Z. Chen. Automatic road crack detection using random structured forests. *IEEE Transactions on Intelligent Transportation Systems*, 17(12):3434–3445, 2016. 2

[36] Ke Sun, Bin Xiao, Dong Liu, and Jingdong Wang. Deep high-resolution representation learning for human pose estimation. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5686–5696, 2019. 5

[37] Nima Tajbakhsh, Laura Jeyaseelan, Qian Li, Jeffrey N. Chiang, Zhihao Wu, and Xiaowei Ding. Embracing imperfect datasets: A review of deep learning solutions for medical image segmentation. *Medical Image Anal.*, 2020. 2

[38] Daiki Tanaka, Daiki Ikami, Toshihiko Yamasaki, and Kiyoharu Aizawa. Joint optimization framework for learning with noisy labels. In *CVPR*, 2018. 2

[39] Nguyen Tien Sy, Stephane Begot, Florent Duculty, and Manuel Avila. Free-form anisotropy: A new method for crack detection on pavement surface images. In *International Conference on Image Processing, ICIP*, pages 1069–1072, 09 2011. 2

[40] Liang Xu, Taro Hatsutani, Xing Liu, Engkarat Techapanurak, Han Zou, and Takayuki Okatani. Pushing the envelope of thin crack detection, 2021. 3, 4

[41] Xincong Yang, Heng Li, Y. Yu, Xiaochun Luo, Ting Huang, and X. Yang. Automatic pixel-level crack detection and measurement using fully convolutional network. *Comput. Aided Civ. Infrastructure Eng.*, 33:1090–1109, 2018. 2

[42] Changqian Yu, Bin Xiao, Changxin Gao, Lu Yuan, Lei Zhang, Nong Sang, and Jingdong Wang. Lite-hrnet: A lightweight high-resolution network. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10440–10450, 2021. 5

[43] Xingrui Yu, Bo Han, Jiangchao Yao, Gang Niu, Ivor Tsang, and Masashi Sugiyama. How does disagreement help generalization against label corruption? In *Proceedings of the 36th International Conference on Machine Learning*. PMLR, 2019. 2

[44] Yuhui Yuan, Xilin Chen, and Jingdong Wang. Object-contextual representations for semantic segmentation. In *The European Conference on Computer Vision (ECCV)*, 2020. 5

[45] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. In *5th International Conference on Learning Representations, (ICLR)*, 2017. 2

[46] Dejin Zhang, Qingquan Li, Ying Chen, Min Cao, Li He, and Bailing Zhang. An efficient and reliable coarse-to-fine approach for asphalt pavement crack detection. *Image and Vision Computing*, 2016. 1

[47] Kaige Zhang, H.D. Cheng, and Boyu Zhang. Unified approach to pavement crack and sealed crack detection using preclassification based on transfer learning. *Journal of Computing in Civil Engineering*, 32, 03 2018. 2

[48] L. Zhang, F. Yang, Y. Daniel Zhang, and Y. J. Zhu. Road crack detection using deep convolutional neural network. In *2016 IEEE International Conference on Image Processing (ICIP)*, 2016. 2

[49] Zhilu Zhang and Mert Sabuncu. Generalized cross entropy loss for training deep neural networks with noisy labels. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2018. 2

[50] Qin Zou, Yu Cao, Qingquan Li, Qingzhou Mao, and Song Wang. Cracktree: Automatic crack detection from pavement images. *Pattern Recognition Letters*, 2012. 1, 2

[51] Qin Zou, Zheng Zhang, Qingquan Li, Xianbiao Qi, Qian Wang, and Song Wang. Deepcrack: Learning hierarchical convolutional features for crack detection. *IEEE Transactions on Image Processing*, 2018. 1, 2, 3