Swin Transformer Based Crack Detection

Neelesh Verma, Mengyang Pu, Mei Zhang, Danil Prokhorov, Jie Wei, and Haibin Ling

***Abstract*—Crack detection in various infrastructures is paramount to ensure safety, maintenance, and longevity. The recognition task is made complex by the structural inhomogeneity of cracks. Deep learning approaches have largely improved the performance of crack detection, but challenging cases still remain to be solved such as imprecision in crack details, partly due to the low spatial resolution in state-of-the-art solutions (*e.g.*, CrackFormer). To address these issues, this paper introduces a novel approach to crack detection, named CrackSwinT, using the Shifted window Transformer (Swin-T) architecture by taking advantage of its powerful multi-scale representation. Specifically, CrackSwinT extends previous CrackFormer in several important ways. First, to leverage the efficiency of Swin-T attention over normal attention, we use Swin attention blocks instead of normal attention blocks. Second, we introduce skip connections within encoders and decoders for more effective information flow thus stabilizing the training. Moreover, the decoder is fed with input from the previous decoder block and the corresponding encoder block with a 1×1 convolution for efficient learning of decoders to generate the crack feature maps. Finally, CrackSwinT uses focal loss to handle the imbalance between crack and non-crack pixel samples. Extensive experiments are conducted on three crack detection benchmarks, including CFD, Crack200, and Crack500$_{\text{fix}}$, where Crack500$_{\text{fix}}$ is enhanced from Crack500 by our careful investigations and revisions. On all three datasets, CrackSwinT outperforms previous state-of-the-art methods by significant margins, *e.g.*, nearly 5% in terms of optimal dataset scale (ODS) and optimal image scale (OIS) scores on Crack500$_{\text{fix}}$.**

***Index Terms*—Pavement crack detection, deep learning, convolutional neural network, Transformer, Swin Transformer.**

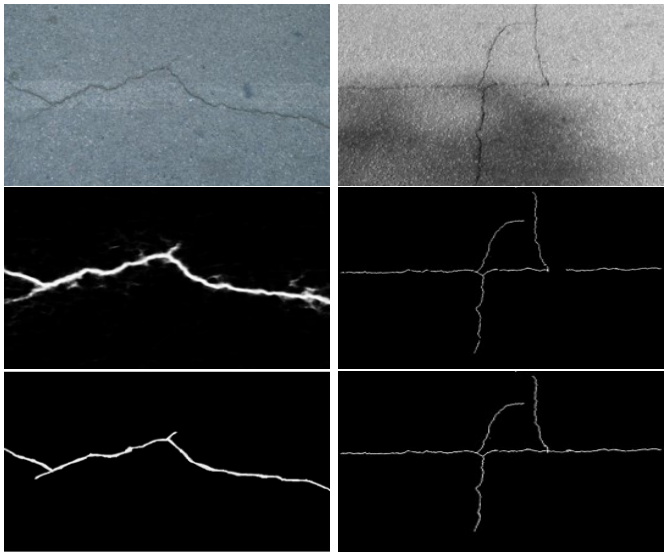


Fig. 1: Top to bottom: original input images, results from a state-of-the-art crack detector (CrackFormer-II [10]), and CrackSwinT results. One can observe that previous results have some noises near the cracks (middle left), and the detected crack is discontinuous (middle right). CrackSwinT (bottom) results improve the results with less noise and continuous cracks.

I. Introduction

CRACK in infrastructures, such as buildings, bridges, roads, and pipelines, pose serious threats to safety, stability, and durability [1]–[6]. Detecting and monitoring these cracks is of utmost importance to ensure early intervention, prevent catastrophic failures, and optimize maintenance efforts. With the rapid advancements in deep learning and computer vision, automated crack detection systems have emerged as a cost-effective solution to assess the state of infrastructure.

Deep learning (DL) models have revolutionized various computer vision tasks, such as object detection, image segmentation, and classification, by automatically learning relevant features from the data. Convolutional Neural Networks (CNNs) [7] have been the backbone of many successful image-related applications. However, CNNs suffer from limitations when it comes to capturing long-range dependencies and effectively processing large images. The CNN-based methods use encoder-decoder architectures. The most common among them is SegNet [8] and U-Net [9]. In such methods, the encoder learns the semantic representations using convolutional and pooling layers. However, the cracks have structural inhomogeneity, such as low contrast between the crack and background, thickness of the cracks, and continuity of the cracks. These methods fail to achieve pixel-level segmentation and hence result in coarse crack segmentation.

There have been multiple architectures suggested for the task of crack detection [5], [6], [11]–[14]. With deep neural networks, their receptive field keeps increasing with the depth and ultimately suppressing the local details which are important to capture in the crack detection task. To overcome the suppressing issue, Transformer-based architectures have been used in capturing long-range dependencies.

Recently, Transformer-based architectures have gained immense popularity in natural language processing tasks due to their ability to capture long-range dependencies in text sequences [15]–[17]. This success has inspired researchers to explore the application of Transformers in computer vision tasks as well [18]–[20]. The Transformers have been utilized in crack detection very recently in the CrackFormer net-

N. Verma, M. Pu and H. Ling are with the Department of Computer Science, Stony Brook University, Stony Brook, NY, 11794 USA. (e-mail: {neverma,hling}@cs.stonybrook.edu; mengyangpu.pmy@gmail.com)

M. Zhang is with the Department of Accounting and Finance, Rowan University, Glassboro, NJ 08028, USA (e-mail: zhangm@rowan.edu).

D. Prokhorov is with the Toyota Research Institute North America, Ann Arbor, MI 48105 USA. (e-mail: danil.prokhorov@toyota.com)

J. Wei is with the Department of Computer Science, City College of New York, New York, NY 10031 USA. (e-mail: jwei@ccny.cuny.edu)

works [10], [21]. The idea was to utilize its merits to capture long-range dependencies and global contextual information in the image. CrackFormer outperformed existing state-of-the-art methods [5], [6] with significant improvement. Despite the achievement, challenges remain to be resolved. One issue with these methods is their generalization across cracks of different widths. CrackFormer-I [21] focuses on thin cracks and have troubles when dealing with thick ones. CrackFormer-II [10] improves upon this issue and performs better on wide cracks, but oftentimes leaves very crisp boundaries. To address these issues, our main motivation in this work is to have a network that generalizes well across various types of cracks – coarse cracks, fine cracks, etc. As an illustration, Figure 1 shows an example of a very fine crack (right side) and a moderately thin one (left side). CrackFormer-I performs well on the thin crack but cannot maintain the continuity of the crack. On the thicker crack, the detected boundaries have a lot of noise in their periphery.

Even though multi-head self-attention of the Transformer can be made at least as expressive as any convolutional layer [22], normal attention still uses a fixed receptive field size, which might not capture both fine-grained and global information in the image effectively. Swin Transformer [23] overcomes the fixed-size issue by using hierarchical attention windows, allowing the model to attend to different parts of the image at different levels of granularity, improving its ability to capture both local and global features. This improvement over normal self-attention motivated us to explore the Swin Transformer (Swin-T) design in the crack segmentation task.

Inspired by Swin-T, we propose a new crack detection algorithm, named CrackSwinT, that effectively addresses the issues discussed. In particular, we build our CrackSwin on top of the CrackFormer framework with important improvements in various aspects. First, CrackSwinT upgrades the self-attention modules in CrackFormer with Swin-T-based self-attention to allow more efficient multi-scale inference. Second, we use concatenated skip connections, like in other computer vision tasks [24]–[26], to facilitate the information flow from earlier layers to later ones while preserving spatial information. CrackSwinT uses these concatenated skip connections in adjacent encoder blocks and decoder blocks separately. Third, we introduce skip connections between the encoder and the corresponding decoder for parameter efficiency and feature propagation. Lastly, we employ the focal loss to address the sample imbalance issue, which occurs naturally due to the sparsity of crack pixels.

The main contributions of this paper can be summarized as follows:

- Developed a novel Swin-based self-attention block to capture long and short-range contextual information using position embeddings and shifted windows.
- Devised the concatenated skip connections between adjacent encoder and decoder blocks for information flow from initial layers to later layers.
- Facilitated encoder decoder feature propagation, bypassing the feature map from the encoder layer to corresponding decoder layers by performing a 1×1 convolution with the feature maps from the previous layer of the decoder.
- $\text{Crack500}_{\text{fix}}$ dataset enhances the Crack500 dataset [6] with better data distribution and annotation.

The contributions together bring clear benefits to the proposed CrackSwinT method. In the extensive experiments conducted on three crack detection benchmarks, including CFD, Crack200, and $\text{Crack500}_{\text{fix}}$, CrackSwinT clearly outperforms previous state-of-the-art methods by significant margins, *e.g.*, nearly 5% in terms of optimal dataset scale (ODS) and optimal image scale (OIS) scores on $\text{Crack500}_{\text{fix}}$.

The rest of the paper is organized in the following way: Section II reviews related works; Section III describes the crack detection with Swin Transformers; Section IV presents the Crack500 dataset and some problems and resolutions; Section V provides experiments on different datasets with results and an ablation study for different components of the network, and Section VI concludes the paper.

II. Related works

In this section, we will briefly review non-deep learning and deep learning-based methods for crack segmentation and then discuss modern Transformer-based methods.

A. Traditional Methods

Crack detection has been studied in different fields on conventional image processing methods. Here, we summarize some of these methods directly related to traditional learning-based solutions.

In one of the early studies on crack detection, Iyer *et al.* [27] used morphological operations and cross-curvature evolution to detect cracks in noisy environments in pipe images. Building on this approach, Sinha *et al.* [28] proposed a two-step approach where local crack features are extracted using statistical properties, and then global cleaning and linking operations are performed to merge segments to form cracks. Huang *et al.* [29] and Peng *et al.* [30] also proposed a similar two-way approach with more emphasis on background removal. Xu *et al.* [31] used saliency and statistical features to identify cracks, which includes generating a saliency map, measuring the spatial continuity of potential crack pixels, updating the map with a Bayesian model, and ultimately extracting cracks through adaptive binarization.

There have also been studies about the frequency domain using Wavelet transforms. Subirats *et al.* [32] located the singularity of wavelet transform to detect the cracks. Zhou *et al.* [33] decomposed the image into different-frequency subbands by the wavelet transform. In the high-frequency subbands, distresses were transformed into high-amplitude wavelet coefficients and noise into low ones. Later, Subirats *et al.* [34] proposed a 2D continuous wavelet transform (CWT) method where a complex coefficient map was computed from CWT in two directions and the cracks were detected using these coefficients.

Similar to wavelets and saliency maps, many other features have also been used to detect and localize cracks. Hu *et al.* [35] utilized local binary patterns (LBP) features, and Kapela *et al.* [36] used histograms of oriented gradients (HoGs) for crack detection. Salman *et al.* [37] used multiple-oriented Gabor

filters. Another popular approach, *CrackTree* [4], constructed a crack probability map using tensor voting, modeled it as a graph, and derived minimum spanning trees from this graph. CrackForest [1] utilized hand-crafted features from multiple levels to detect cracks.

B. Deep Learning Methods

Deep neural networks (DNNs) have achieved extraordinary results in various vision tasks. One of the earliest methods was proposed by Zhang *et al.* [38]. The authors used a simple four convolutional layers and two fully connected layers. Later, deeper networks were used [39]–[42] and performance kept on improving.

With developments in deep learning (DL), many new architectures were developed. Crack localization can be viewed as a subtask of segmentation. Using pyramid-based networks and hierarchical features has been a common technique in segmentation. DeepCrack [5] utilized multi-scale and multi-level features from a pyramid-based network and Conditional Random Fields (CRFs) to refine the final prediction results. Yang *et al.* [6] proposed a feature pyramid and hierarchical boosting network that utilized the pyramid-based network to extract multi-scale features and integrate contextual information from different scales as well as the hierarchical boosting module to balance the contribution of easy and hard samples, this method performs well on both types of datasets - coarse and fine, but does not produce very sharp crack boundaries. Zhang *et al.* [43] proposed the CrackNet for crack detection purposes which specifically uses invariant spatial size through all layers such that supervised learning can be conducted at the pixel level. Fei *et al.* [44] built on this architecture to propose CrackNet-V with a deeper architecture and fewer parameters.

Crack detection heavily relies on learning contextual information, topological structures, and various degrees of damage. Encoder-decoder architectures have been tried to resolve some of these issues. Liu *et al.* proposed FPCNet [45], which uses a multi-dilation module to extract features of cracks with different widths and topologies and a Squeeze-and-Excitation (SE) module to improve the discriminative power of the learned features. A generative adversarial network (GAN) based method CrackGAN was proposed by Zhang *et al.* [46], which achieved high accuracy and robustness to partially accurate ground truths in detecting cracks on pavement surfaces.

With the advent of Transformers [17], the attention mechanism has been employed in various vision and natural language processing tasks. The CrackFormer network [21] proposes a novel Transformer-based network for fine-grained crack detection. The network consists of a SegNet-like encoder-decoder architecture with novel self-attention and scaling-attention modules. The self-attention modules capture long-range dependencies between pixels in the input image and the scaling-attention modules are used to combine outputs from the corresponding encoder and decoder blocks. However, the network focused on fine-grained crack detection and performed poorly with images having coarse cracks. To overcome the limitations of CrackFormer, the authors developed a modified version [10] that uses an upgraded Transformer Encoder block consisting of a local self-attention layer and a local feed-forward layer with skip connections between them. This change in the Encoder block helped the network learn the local context and structure information inside the patches. However, the paper does not perform extensive evaluations of coarse/thick crack datasets like Crack500 [6].

C. Other Related Work

Aside from the above-mentioned studies that directly work on crack detection, our work is also related to, or inspired by, other studies, especially in DL-based edge detection. The crack detection task has a lot of similarities with edge detection. Edge detection can be followed by segmentation techniques to separate the detected cracks from the background.

An early work applying DL to edge detection can be found in [47]. Recently, Transformer-based networks have been used in edge-detection tasks as well. Pu *et al.* [48] proposed a novel edge detection method based on the Transformer architecture which consists of two stages: a global Transformer encoder to capture long-range global context and a local Transformer encoder to excavate short-range local cues. The global Transformer encoder uses a self-attention mechanism to learn long-range dependencies between pixels in the input image. The local Transformer encoder uses a self-attention mechanism to learn short-range dependencies between pixels in the input image. CrackSwinT method draws architectural motivation from this line of work.

In the field of Transformers, several developments have been made. We use the Swin Transformer [23], which employs a hierarchical architecture with shifted windows to improve efficiency and performance, which has been shown to achieve better performance than Vision Transformers.

III. CRACK DETECTION WITH SWIN TRANSFORMERS

A. Overview

The CrackSwinT architecture uses a CrackFormer-like network as a backbone. The self-attention layers have been replaced by Swin-T-based self-attention. Concatenated skip connections are implemented between adjacent encoder and decoder blocks. Moreover, encoder feature maps are bypassed to the corresponding decoder feature map after performing a 1×1 convolution. The overall architecture is shown in Figure 2.

The encoder consists of 13 layers (the first 13 layers of VGG16 [49] as in SegNet). We want feature extraction at different layers and information flow between features to capture both long-range contextual information, so we deployed the encoder in 5 stages, which have respectively 2, 2, 3, 3, and 3 layers. There is also a concatenated skip connection from one stage to another as explained in Section III-C. The features from the encoder and decoder are then fused together using a FusionBlock to produce an attention coefficient mask. These masks from all 5 stages are then concatenated and fused to generate the final mask. Also, to have information flow between encoder and decoder features, CrackSwinT bypasses the feature map from the encoder layer to its corresponding

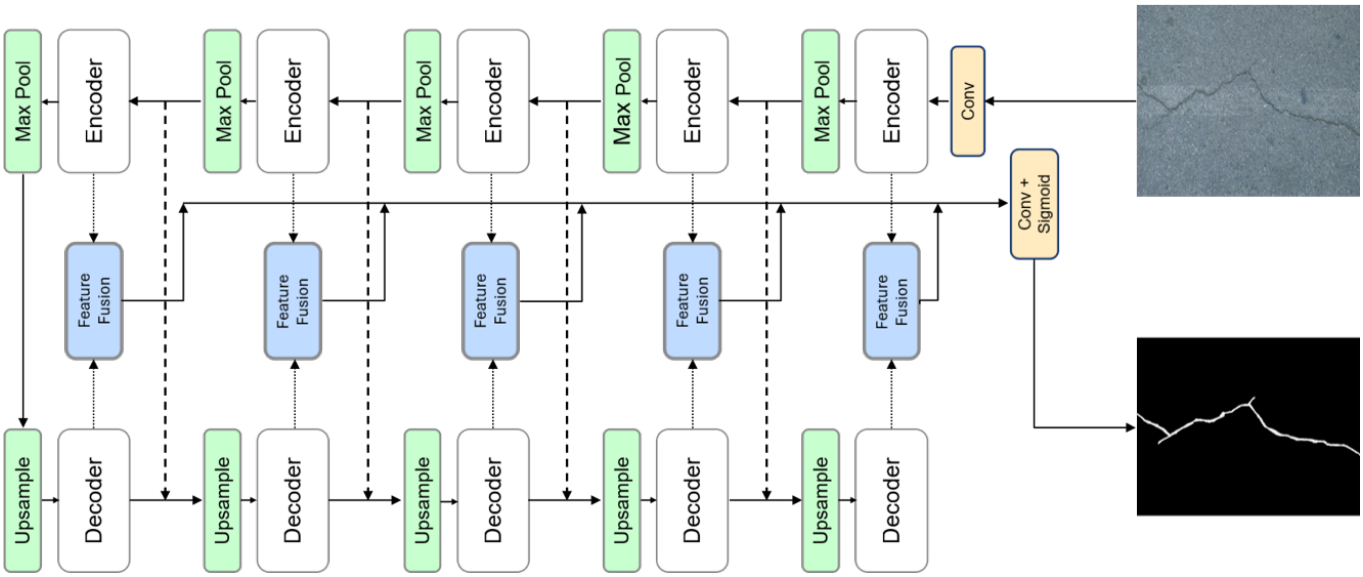


Fig. 2: The CrackSwinT network uses CrackFormer-like architecture as a backbone. It consists of 5 encoder blocks having respectively 2, 2, 3, 3, and 3 Swin-T-based self-attention layers and 5 corresponding decoders with 3, 3, 3, 2 and 2 self-attention layers. Concatenated skip connections between adjacent encoder blocks and adjacent decoder blocks are for information flow. Skip connections across the encoder block to the decoder block are for feature propagation.

decoder layer and performs a 1×1 convolution with the feature maps from the previous layer of the decoder. In the following sections, we will describe each of the modules and operations in detail.

B. Swin Transformer Block

The Swin Transformer [23] builds on Vision Transformer (ViT) [50] by introducing two key concepts: hierarchical architecture and shifted windows. The hierarchical architecture divides the image into a sequence of smaller patches and then learns features at different levels of granularity. Hierarchical design allows the Swin Transformer to learn more complex features than the original ViT. The shifted windows concept limits the self-attention computation to non-overlapping local windows. Window shift makes the Swin Transformer more efficient than ViT, which computes self-attention over the entire image.

We took the architectural motivation from the CrackFormer paper [21]. However, the CrackFormer uses the original self-attention layer (as proposed in the paper [17]), while we use the Swin Transformer. Instead of processing the entire image at once, it divides the image into non-overlapping patches called windows. These windows are shifted in a grid pattern to capture information from different parts of the image. The CrackSwinT network replaces the original self-attention block with shifted window-based self-attention. Shifted Window-based Self-Attention focuses on capturing local dependencies within a sequence by dividing it into non-overlapping segments or windows. Each segment is processed separately, and attention is applied within each window, thus making it beneficial for tasks where the context matters such as in crack detection.

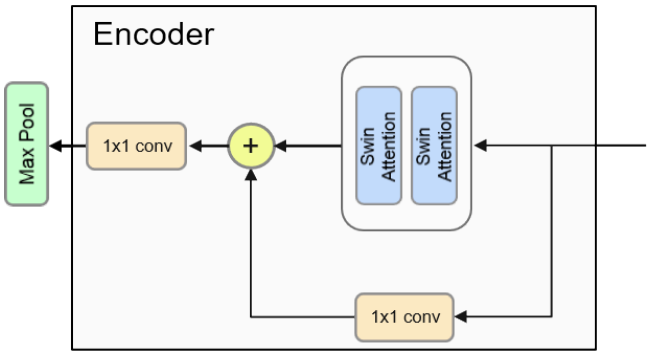


Fig. 3: Concatenated Skip Connection across an Encoder block

C. Concatenated Skip Connections

The encoder and decoder are laid out in 5 stages. Each encoder and decoder contains 2 or 3 Swin blocks. The receptive field keeps on increasing as we move deeper into the layers. Large receptive fields help in capturing large-range dependencies and contextual information. However, in the case of crack segmentation, we also want continuous cracks (continuous thick lines that are not broken). So we also wish to preserve short-range contexts in the deeper layers. To achieve context awareness, we have used skip connections between the stages as shown in Figure 3. Skip connections allow for a direct flow of information from one stage to another while preserving spatial information. Also, it brings more stability to the training because of better gradient flow.

D. Encoder-Decoder Feature Propagation

For the task of crack segmentation, CrackSwinT needs to understand both global context and local details. Apart from the information flow inside encoders and decoders, we also

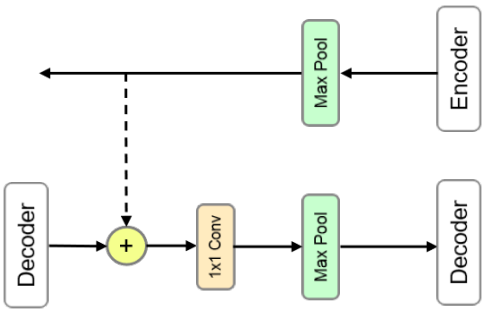


Fig. 4: Feature propagation from encoder to decoder

pass feature maps from encoder layers to the corresponding decoder layers as shown in Figure 4. Using the feature map helps the decoder better reconstruct and generate fine-grained details in the output, which is crucial for our task. It also enables the model to leverage both high-level abstractions and low-level details simultaneously.

E. Feature Fusion Module

Feature fusion module simply takes the features from each of the layers from the encoder and corresponding decoder block, and performs concatenation, convolution, and then upsampling operation. As suggested in U-Net [9], learned features in specific decoder blocks can boost the segmentation performance. Basically, the encoder features are passed through a sigmoid activation layer, which amplifies some features while suppressing others. These sigmoid features act as masks for decoder features and activate some decoder features while suppressing other irrelevant ones.

We follow the CrackFormer [21] to design the fusion module. First, the encoder features within an encoder block are concatenated and passed through a convolution, batch-norm, and sigmoid layer. The result is a mask-like feature. Similarly, the decoder features are also concatenated, convoluted, and passed through batch-norm and ReLu activation. Then this final decoder feature is multiplied by the mask from the encoder feature. The useful decoder features are preserved while irrelevant ones are attenuated at this point.

The feature fusion step is performed at each of the encoder and decoder blocks. This way, we have five maps, which are then passed through the convolution and sigmoid layers to get our final segmented image.

F. Handling Imbalanced Dataset

CrackSwinT predicts a binary map for each image with crack pixels having a value of 1 and non-crack pixels as 0. It is clear that in any image, non-crack pixels will be much more than the number of crack pixels; resulting in class imbalance since we use precision and recall-based metrics. To address this class imbalance, we use the Focal Loss [51]. The key idea behind Focal Loss is to reduce the loss assigned to well-classified examples (pixels in our case) and focus more on the hard or misclassified examples. It does this by introducing two main components:

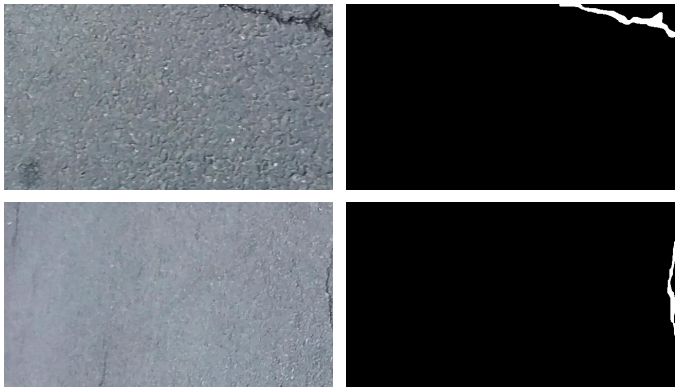


Fig. 5: Some inappropriate samples from Crack500 dataset. It can be noticed in these images that the cracks are lying towards the very corner hampering the training.

- Modulation Factor (Focusing Parameter) - Focal Loss introduces a hyperparameter called the "focusing parameter" (often denoted as gamma, γ). The "gamma" parameter controls the degree of down-weighting for well-classified examples. A higher value of gamma gives more emphasis to hard examples, making the loss more focused on correcting misclassification.
- Focal Loss Function - The Focal Loss function itself is defined as a modification of the standard cross-entropy loss. It is computed as follows:

$$\text{FL}(p_t) = -\alpha_t(1-p_t)^{\gamma}\log(p_t)$$

where α_t is a dynamic balancing factor that helps to address the class imbalance and is used to assign a higher weight to underrepresented classes, p_t represents the predicted probability of the correct class, and γ determines the degree of focusing on hard examples.

IV. CRACK500$_{\text{fix}}$: ENHANCED CRACK500

We need a dataset having both coarse and fine cracks. Most of the current datasets [4], [5], [52] generally, have only one of these attributes. An exception is the Crack500 dataset [6], on which we choose to focus our study. It comprises 500 images (2560×1440) of pavement cracks taken on Temple University's main campus using cell phones. Each image has a detailed binary map indicating the location of cracks at the pixel level. This dataset is currently the largest publicly available pavement crack dataset with fine-grained annotations. It is divided into 250 training images, 50 validation images, and 200 test images.

Because of the limited number of images and their large size, the authors divided each image into 16 non-overlapping regions. So new images are 640×360. Only regions with more than 1,000 pixels of cracks were retained, resulting in 1,896 training images, 348 validation images, and 1,124 test images. The validation data is used for model selection during training to prevent overfitting. Once the model is chosen, it is tested on the test data and other datasets to assess its performance and generalizability.

(a) Original Image (b) Ground Truth Map (c) Original Image (d) Ground Truth Map

Fig. 6: (a,c) Input crack images. (b) The ground truth binary map misses some crack pixels. (d) The crack is continuous in the original image but the ground truth binary map does not capture it.

TABLE I: Statistics of the new dataset Crack500$_{\text{fix}}$

Dataset	Total Images	Cracks in corner	Removed Images	Final Images
Training	1896	136	8	1888
Validation	348	22	4	344
Test	1120	55	4	1116

A. Issue with Crack500

Despite its desired diversity in crack patterns, we find some issues in the images or annotations in the original Crack500. Two sample images from the dataset are shown in Figure 5. We can observe that some of the images had cracks located in the extreme corners. This can be problematic during training because these corner cracks may not provide useful information or could introduce noise into the training process.

Additionally, while manually going through the dataset, we also find some frequent imprecision in the binary crack annotation: (1) the binary maps for some cracks are missing, and (2) in the case of fine cracks, the continuity of the crack sometimes is not well maintained in the binary map. Figure 6 illustrates both the issues.

B. Enhancing Crack500

To address the issues mentioned above, we went through the Crack500 dataset manually and carefully. If a crack appears at a corner, then first we will pull out the original image (before cropping into 16 non-overlapping regions). From this original image, we will simply shift the cropping window towards the crack by 150×150 pixels as shown in Figure 7. We finalized this shifting amount by manually using different shifting and looking at the resulting images. After performing shifting, some images still had cracks in the corners. Since these were very few, we simply removed them from the dataset. The dataset info for this new modified dataset is shown in Table I.

The second issue in the dataset was with the binary maps, in which, the binary map did not capture all the crack pixels correctly (found with thin cracks), as shown in Figure 6. To overcome this issue, we ran the two already existing algorithms - FPHBN [6] and CrackFormer II [10]. After passing the faulty images with these two algorithms, we took the intersection of the binary map produced by both of these methods. The results are shown in Figure 8. Since CrackFormer II works well on thin cracks and FPHBN produces coarse binary maps, this approach should be able to resolve this anomaly. We call this new dataset as *Crack500*$_{\text{fix}}$.

V. EXPERIMENTS

A. Datasets

We picked 3 datasets to validate our method.

1) Crack500$_{\text{fix}}$*:* As already mentioned, this dataset is a modified version of the Crack500 dataset [6]. The modification procedure is explained in the section IV.

2) Cracktree200: The Cracktree200 dataset was presented by Zhou *et al.* [4]. It consists of 206 pavement images of dimensions 800×600 with various kinds of cracks as shown in Figure 9. It comes with some challenges for testing and evaluating the proposed method, including issues such as shadows, occlusions, low contrast, noise, etc. These challenges mimic real-world conditions. The annotation of this dataset is done manually on a pixel-wise basis.

3) CFD: This dataset was proposed by Shi *et al.* [1] to evaluate their methodology. The dataset contains 118 images of urban road surfaces with dimensions 480×320, shown in Figure 9. The annotation has been done manually. The images contain noises such as shadows, oil spots, and water stains. The images were taken by an iPhone5 with a focus of 4mm, an aperture of f/2.4, and an exposure time of 1/134s.

B. Implementation Details

1) Hyper-parameters Setting: All the images have been resized to 360×640. We set Adam optimizer with learning rate = $1e^{-3}$, weight decay = $2e^{-4}$, and training iteration 5000.

2) Loss Function: As discussed in Section III, we use Focal Loss as our loss function. The implementation is already available in the PyTorch library. We used the default parameters - γ as 2, and α as 0.25. We have outputs from all 5 encoder-decoder pairs and a final output from our FusionModule. Our final loss is simply a summation of focal loss (FL) over all these outputs.

$$\text{Loss} = \sum_{i=1}^{N} \left(\left(\sum_{j=1}^{5} \text{FL}_j^{\text{side}} \right) + \text{FL}^{\text{fusion}} \right)$$

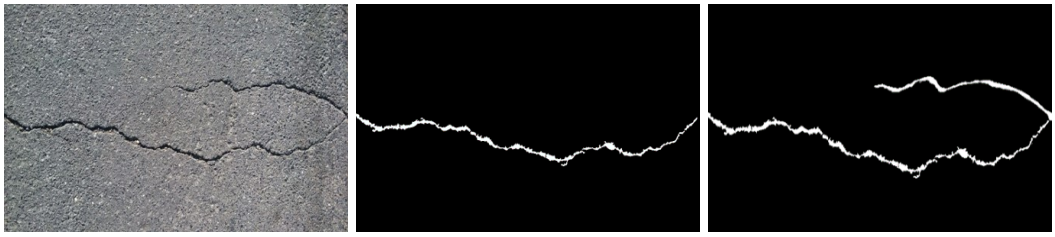
where $\text{FL}_j^{\text{side}}$ represents the focal loss from the j-th encoder-decoder pair output and $\text{FL}^{\text{fusion}}$ represents the focal loss from the fusion layer.

C. Performance Metrics

The crack detection can be thought of as a binary classification task on a per-pixel level. Therefore, we can leverage usual

(a) Original image in Crack500 (b) Ground Truth (c) Shifting image by 150×150 (d) Shifted Ground Truth

Fig. 7: Dealing with Cracks in the corner by shifting the cropped window towards the crack.



(a) Original image (b) Ground truth in Crack500 (c) Ground truth fixed

Fig. 8: Crack pixels are missing in the ground truth of Crack500 (b). We fixed the ground truth in (c).

Fig. 9: Examples from the CFD (top) and Cracktree200 (bottom) datasets.

metrics like Precision and Recall. Precision can be calculated as $P = \frac{TP}{TP+FP}$ and Recall as $R = \frac{TP}{TP+FN}$ for true/false positive (TP/FP) and false negative (FN). So, for each image, we can compute the F1 score ($2\frac{P*R}{P+R}$).

Depending on the way of thresholding, we can have two evaluation metrics -

1) ODS: If we fix the threshold for the dataset and compute the F1 score at different thresholds, we get the Optimal Dataset Scale (ODS).

$$\text{ODS} = \max_{t \in \mathcal{T}} \left(2 \frac{p_t r_t}{p_t + r_t} \right)$$

where p_t and r_t represents the precision and recall at threshold t, and $\mathcal{T} = \{0.01, 0.02, \ldots\}$ is the set of sampled thresholds.

2) OIS: Instead of fixing the threshold across the dataset, if we choose the best F1 score for each image, we get the Optimal Image Scale (OIS).

$$\text{OIS} = \frac{1}{N} \sum_{i=1}^{N} \max_{t \in \mathcal{T}} \left(2 \frac{p_t^i r_t^i}{p_t^i + r_t^i} \right)$$

where N is the total number of images.

D. Comparison with SOTA methods

We have compared our results with three State-of-the-art methods - FPHBN [6] CrackFormer I [21] and CrackFormer II [10].

1) FPHBN: We trained FPHBN from scratch on the Crack500$_{\text{fix}}$ dataset (since the original results were on Crack500). We used the same parameters as given in the original paper [6].

2) CrackFormer I: This is the first paper in the CrackFormer series. The paper [21] does not perform the evaluation on the Crack500 dataset. So we retrained the network on the Crack500$_{\text{fix}}$ dataset for comparison.

3) CrackFormer II: Similar to its predecessor, this paper [10] also does not perform the evaluation on the Crack500 dataset. So we had to train it again on our Crack500$_{\text{fix}}$ dataset. All the hyper-parameters are set the same.

E. Experimental Results

1) Results on Crack500$_{\text{fix}}$: This dataset contains both thin and thick cracks. The evaluation metric scores are also listed in Table II. CrackSwinT method outperforms all the existing ones. CrackSwinT performs 5.2% better on ODS and 4.5% better on OIS as compared to CrackFormer II. Primarily, CrackSwinT performs well on thick cracks which was not the case with CrackFormer II and CrackSwinT also produces crisp

TABLE II: The ODS, and OIS of different methods on $\text{Crack500}_{\text{fix}}$ test dataset.

Methods	ODS	OIS
CrackForest [1]	0.199	0.199
EDTER [48]	0.660	0.661
DeepCrack [5]	0.640	0.668
FPHBN [6]	0.604	0.635
CrackFormer I [21]	0.661	0.683
CrackFormer II [10]	0.672	0.693
CrackSwinT	**0.724**	**0.738**

TABLE III: The ODS, and OIS of different methods on CrackTree200 test dataset.

Methods	ODS	OIS
CrackForest [1]	0.080	0.080
EDTER [48]	0.481	0.530
DeepCrack [5]	0.379	0.513
FPHBN [6]	0.517	0.579
CrackFormer I [21]	0.840	0.859
CrackFormer II [10]	0.851	0.864
CrackSwinT	**0.879**	**0.893**

boundaries which was a drawback of FPHBN. Some results are also shown in Figure 10 depicting the improvement.

2) Results on CrackTree200: CrackTree200 is a thin crack dataset that has been used in all the methods from which we are comparing. The results are shown in the Table III. We can see from Table III that CrackFormers were already performing well on the thin cracks and CrackSwinT outperforms them. We gained 2.8% on ODS and 2.9% on OIS. Some of the results are shown visually in Figure 10. It can be observed that CrackSwinT produces thin and crisp boundaries.

3) Results on CFD: The CrackFormer series has no results on the CFD dataset. So we evaluated both the networks on this dataset. From Table IV, it can be seen that CrackSwinT achieves better scores than FPHBN and CrackFormer I and II (2.3 % gain on ODS and 3.3 % gain on OIS). Some results are also illustrated in Figure 10.

F. Ablation Study

In CrackSwinT, there are three main modules - Swin Transformer, Skip Connections within encoders and decoders, and Encoder-Decoder Information Flow. We will show in subsequent subsections, how each of these modules plays an important in improving the evaluation score. Lastly, we also analyzed Focal Loss as opposed to Binary Cross Entropy (BCE) Loss. In the following subsections, whenever we are removing a particular module, the rest of the modules and architecture remain the same.

1) Swin Transformer vs Conventional Transformer Block: Traditional transformers use fixed-size windows for self-attention, which can be problematic when dealing with images due to their grid-like structure. Swin Transformer addresses this issue by using shifted windows, enabling better capture of local and global information in images. This is also captured in Table V. We can conclude that there is a significant improvement over the conventional Transformer with the Swin.

TABLE IV: The ODS and OIS of different methods on CFD dataset.

Methods	ODS	OIS
CrackForest [1]	0.104	0.104
EDTER [48]	0.671	0.692
DeepCrack [5]	0.669	0.681
FPHBN [6]	0.683	0.705
CrackFormer I [21]	0.708	0.724
CrackFormer II [10]	0.748	0.760
CrackSwinT	**0.771**	**0.793**

TABLE V: The ODS, and OIS of Swin vs Conventional Transformer Block on $\text{Crack500}_{\text{fix}}$ test dataset.

Module	ODS	OIS
Conventional Transformer	0.670	0.685
Swin Transformer	0.724	0.738

TABLE VI: Ablation study for different modules on $\text{Crack500}_{\text{fix}}$ test dataset.

Module	ODS	OIS
With all modules	0.724	0.738
Without Concatenated Skip Connections	0.697	0.711
Without Information Flow	0.708	0.722

TABLE VII: The ODS, OIS scores of BCE vs Focal Loss

Module	ODS	OIS
BCE Loss	0.701	0.725
Focal Loss	0.724	0.738

2) Concatenated Skip Connections: It has been shown in various vision tasks that concatenated skip connections, in general, improve performance and stability in training. Self-attention mechanism itself has skip connections. In CrackSwinT, we introduced the skip-connections between the encoders and the decoders. Table VI shows the evaluation scores with and without skip connections.

3) Encoder-Decoder Feature Propagation: The decoder receives the features not just from the previous decoder block but also from the corresponding encoder block. We simply perform a 1×1 convolution to reduce the dimensions and pass it to the decoder block. The encoder provides valuable contextual information for segmentation. Since different levels of features captured by the encoder represent information at various spatial resolutions, by combining features from both the encoder and previous decoder blocks, the model can leverage multi-scale information. We performed the ablation study for this by removing this information flow and found out that the results suffer, as shown in Table VI.

4) Focal Loss vs BCE Loss: The loss functions used in FPHBN [6], CrackFormers [10], [21], and DeepCrack [5] are cross-entropy losses with slight modifications. We have used Focal Loss since it addresses class imbalance. To quantify its importance, we trained our network with Focal Loss and BCE Loss. The results are summarized in Table VII, where we can deduce that focal loss indeed improves the overall performance.

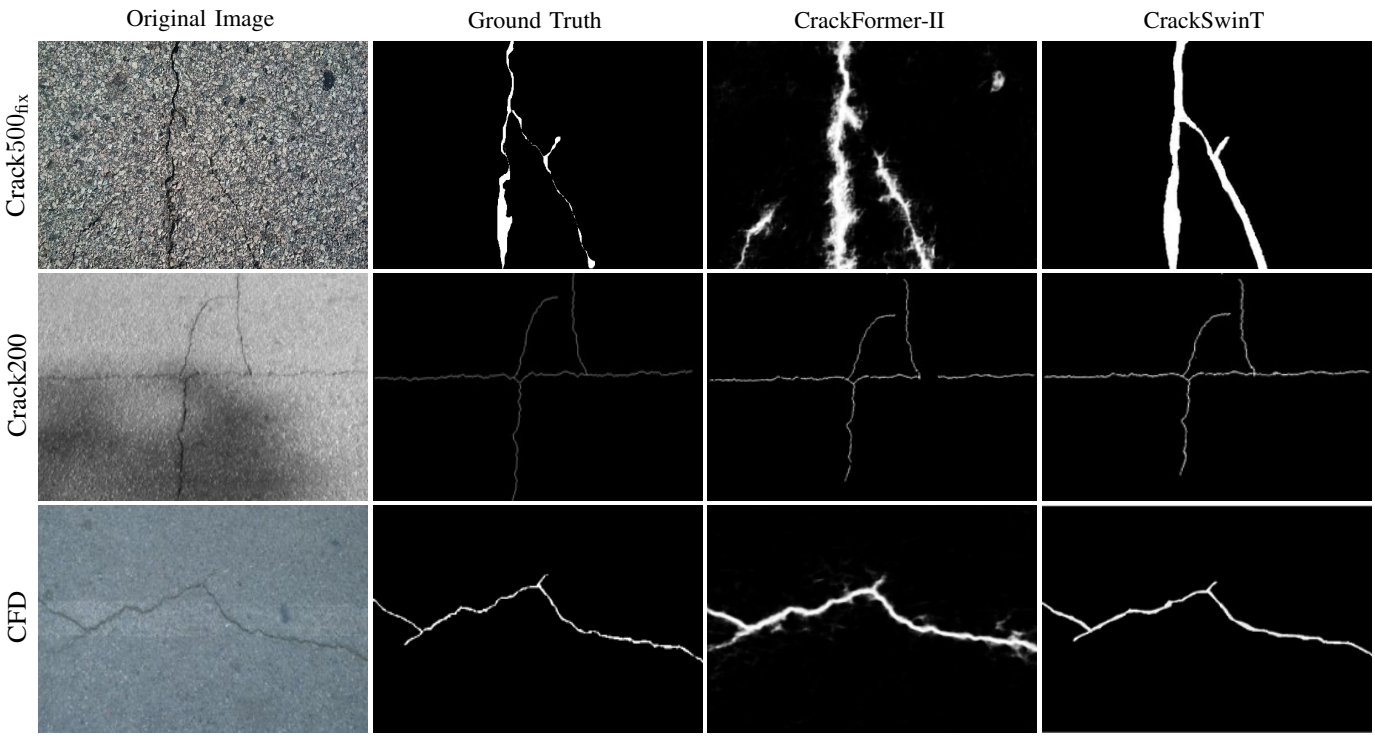


Fig. 10: Comparison of our method with CrackFormer-II on different datasets. The first row is from $\text{Crack500}_{\text{fix}}$ (a thick crack). The second row is a thin crack of Crack200 and the last row is a medium-sized crack from the CFD dataset. CrackSwinT consistently performs well across different widths of crack.

VI. Conclusion

In this paper, we proposed a Swin Transformer-based architecture, named CrackSwinT, for crack detection using CrackFormer as a backbone. The self-attention layer of the Swin Transformer is used to capture both short and large ranges of contextual information because of its flexible and large receptive field. The learned features from encoder layers are passed to the decoder with a 1×1 convolution which helps the decoder to regenerate those learned features efficiently. We also introduced concatenated skip connections between adjacent stages of the encoder and decoder for information flow and stability in training. Moreover, focal loss is employed to address the data imbalance issues in crack pixels. There are three datasets on which we evaluated CrackSwinT. Out of these three, Crack500 has a good mix of different sizes of cracks but suffers from some imprecision annotation and inappropriate crack locations. We fixed these issues and proposed an enhanced version named $\text{Crack500}_{\text{fix}}$. Results demonstrated that CrackSwinT outperformed other methods on all datasets. Future work includes enhancing the method for multi-modal crack detection, applying for corrosion detection, and implementing it in a human-machine inspection device.

Acknowledgement

The work was supported in part by US National Science Foundation Grants 2006665, 2128350, and 2324052. This work is also supported in part by the Air Force Office of Scientific Research FA 9550-23-2-0002 and FA9550-21-1-0082. The support of these agencies is gratefully acknowledged. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation or the United States Air Force. M. Pu conducted the work during her stay with Stony Brook University. We would also like to thank Dr. Erik Blasch for the concept discussion and paper revision.

References

[1] 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*, vol. 17, no. 12, pp. 3434–3445, 2016.

[2] H. Oliveira and P. L. Correia, "Automatic road crack detection and characterization," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 1, pp. 155–168, 2013.

[3] M. Quintana, J. Torres, and J. M. Menéndez, "A simplified computer vision system for road surface inspection and maintenance," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 3, pp. 608–619, 2016.

[4] Q. Zou, Y. Cao, Q. Li, Q. Mao, and S. Wang, "Cracktree: Automatic crack detection from pavement images," *Pattern Recognition Letters*, vol. 33, pp. 227–238, 2012.

[5] Q. Zou, Z. Zhang, Q. Li, X. Qi, Q. Wang, and S. Wang, "Deepcrack: Learning hierarchical convolutional features for crack detection," *IEEE Transactions on Image Processing*, vol. 28, no. 3, pp. 1498–1512, 2019.

[6] F. Yang, L. Zhang, S. Yu, D. Prokhorov, X. Mei, and H. Ling, "Feature pyramid and hierarchical boosting network for pavement crack detection," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 4, pp. 1525–1535, 2019.

[7] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[8] 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*, vol. 39, no. 12, pp. 2481–2495, 2017.

[9] 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.

[10] H. Liu, J. Yang, X. Miao, C. Mertz, and H. Kong, "Crackformer network for pavement crack segmentation," *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 9, pp. 9240–9252, 2023.

[11] Y.-J. Cha, W. Choi, and O. Buyukozturk, "Deep learning-based crack damage detection using convolutional neural networks," *Computer-Aided Civil and Infrastructure Engineering*, vol. 32, pp. 361–378, 03 2017.

[12] A. Alfarrarjeh, D. Trivedi, S. H. Kim, and C. Shahabi, "A deep learning approach for road damage detection from smartphone images," in *IEEE International Conference on Big Data (Big Data)*, 2018, pp. 5201–5204.

[13] Y. Fei, K. C. P. Wang, A. Zhang, C. Chen, J. Q. Li, Y. Liu, G. Yang, and B. Li, "Pixel-level cracking detection on 3d asphalt pavement images through deep-learning- based cracknet-v," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 1, pp. 273–284, 2020.

[14] A. Zhang, K. Wang, Y. Fei, Y. Liu, C. Chen, G. Yang, J. Li, E. Yang, and S. Qiu, "Automated pixel-level pavement crack detection on 3d asphalt surfaces with a recurrent neural network: Automated pixel-level pavement crack detection on 3d asphalt surfaces using cracknet-r," *Computer-Aided Civil and Infrastructure Engineering*, vol. 34, 08 2018.

[15] J. D. M.-W. C. Kenton and L. K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *North American Chapter of the Association for Computational Linguistics (NAACL)*, 2019.

[16] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, "Albert: A lite bert for self-supervised learning of language representations," in *Eighth International Conference on Learning Representations (ICLR)*, 2020.

[17] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in Neural Information Processing Systems (NIPS)*, vol. 30, 2017.

[18] P. Zhang, X. Dai, J. Yang, B. Xiao, L. Yuan, L. Zhang, and J. Gao, "Multi-scale vision longformer: A new vision transformer for high-resolution image encoding," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021, pp. 2998–3008.

[19] S. Zheng, J. Lu, H. Zhao, X. Zhu, Z. Luo, Y. Wang, Y. Fu, J. Feng, T. Xiang, P. H. Torr *et al.*, "Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 6881–6890.

[20] A. Vaswani, P. Ramachandran, A. Srinivas, N. Parmar, B. Hechtman, and J. Shlens, "Scaling local self-attention for parameter efficient visual backbones," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 12 894–12 904.

[21] H. Liu, X. Miao, C. Mertz, C. Xu, and H. Kong, "Crackformer: Transformer network for fine-grained crack detection," in *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021, pp. 3763–3772.

[22] J.-B. Cordonnier, A. Loukas, and M. Jaggi, "On the relationship between self-attention and convolutional layers," in *Eighth International Conference on Learning Representations (ICLR)*, 2020.

[23] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021, pp. 10 012–10 022.

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

[25] T. Tong, G. Li, X. Liu, and Q. Gao, "Image super-resolution using dense skip connections," in *IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 4809–4817.

[26] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 1492–1500.

[27] S. Iyer and S. K. Sinha, "A robust approach for automatic detection and segmentation of cracks in underground pipeline images," *Image and Vision Computing*, vol. 23, no. 10, pp. 921–933, 2005.

[28] S. K. Sinha and P. W. Fieguth, "Automated detection of cracks in buried concrete pipe images," *Automation in Construction*, vol. 15, no. 1, pp. 58–72, 2006.

[29] W. Huang and N. Zhang, "A novel road crack detection and identification method using digital image processing techniques," in *2012 7th International Conference on Computing and Convergence Technology (ICCCT)*, 2012, pp. 397–400.

[30] L. Peng, W. Chao, L. Shuangmiao, and F. Baocai, "Research on crack detection method of airport runway based on twice-threshold segmentation," in *Fifth International Conference on Instrumentation and Measurement, Computer, Communication and Control (IMCCC)*, 2015, pp. 1716–1720.

[31] W. Xu, Z. Tang, J. Zhou, and J. Ding, "Pavement crack detection based on saliency and statistical features," in *IEEE International Conference on Image Processing*, 2013, pp. 4093–4097.

[32] P. Subirats, O. Fabre, J. Dumoulin, V. Legeay, and D. Barba, "A combined wavelet-based image processing method for emergent crack detection on pavement surface images," in *12th European Signal Processing Conference*, 2004, pp. 257–260.

[33] J. Zhou, P. S. Huang, and F.-P. Chiang, "Wavelet-based pavement distress detection and evaluation," in *Wavelets: Applications in Signal and Image Processing X*, vol. 5207, 2003, pp. 728 – 739.

[34] P. Subirats, J. Dumoulin, V. Legeay, and D. Barba, "Automation of pavement surface crack detection using the continuous wavelet transform," in *International Conference on Image Processing*, 2006, pp. 3037–3040.

[35] Y. Hu and C.-x. Zhao, "A novel lbp based methods for pavement crack detection," *Journal of Pattern Recognition Research*, vol. 5, pp. 140–147, 01 2010.

[36] R. Kapela, P. Śniatała, A. Turkot, A. Rybarczyk, A. Pożarycki, P. Rydzewski, M. Wyczałek, and A. Błoch, "Asphalt surfaced pavement cracks detection based on histograms of oriented gradients," in *22nd International Conference Mixed Design of Integrated Circuits Systems (MIXDES)*, 2015, pp. 579–584.

[37] M. Salman, S. Mathavan, K. Kamal, and M. Rahman, "Pavement crack detection using the gabor filter," in *16th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, 2013, pp. 2039–2044.

[38] L. Zhang, F. Yang, Y. Daniel Zhang, and Y. J. Zhu, "Road crack detection using deep convolutional neural network," in *IEEE International Conference on Image Processing (ICIP)*, 2016, pp. 3708–3712.

[39] A. Zhang, K. Wang, Y. Fei, Y. Liu, C. Chen, G. Yang, J. Li, E. Yang, and S. Qiu, "Automated pixel-level pavement crack detection on 3d asphalt surfaces with a recurrent neural network: Automated pixel-level pavement crack detection on 3d asphalt surfaces using cracknet-r," *Computer-Aided Civil and Infrastructure Engineering*, vol. 34, 08 2018.

[40] L. Pauly, H. Peel, S. Luo, D. Hogg, and R. Fuentes, "Deeper networks for pavement crack detection," in *Proceedings of the 34th International Symposium on Automation and Robotics in Construction (ISARC)*. Taipei, Taiwan: Tribun EU, s.r.o., Brno, July 2017, pp. 479–485.

[41] X. Long, S. Zhao, C. Jiang, W. Li, and C. Liu, "Deep learning-based planar crack damage evaluation using convolutional neural networks," *Engineering Fracture Mechanics*, vol. 246, p. 107604, 2021.

[42] A. Alfarrarjeh, D. Trivedi, S. H. Kim, and C. Shahabi, "A deep learning approach for road damage detection from smartphone images," in *IEEE International Conference on Big Data (Big Data)*, 2018, pp. 5201–5204.

[43] A. Zhang, K. C. P. Wang, B. Li, E. Yang, X. Dai, Y. Peng, Y. Fei, Y. Liu, J. Q. Li, and C. Chen, "Automated pixel-level pavement crack detection on 3d asphalt surfaces using a deep-learning network," *Computer-Aided Civil and Infrastructure Engineering*, vol. 32, no. 10, pp. 805–819, 2017.

[44] Y. Fei, K. C. P. Wang, A. Zhang, C. Chen, J. Q. Li, Y. Liu, G. Yang, and B. Li, "Pixel-level cracking detection on 3d asphalt pavement images through deep-learning- based cracknet-v," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 1, pp. 273–284, 2020.

[45] W. Liu, Y. Huang, Y. Li, and Q. Chen, "Fpcnet: Fast pavement crack detection network based on encoder-decoder architecture," *arXiv preprint arXiv:1907.02248*, 2019.

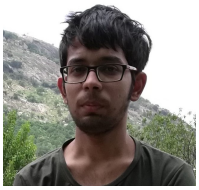
[46] K. Zhang, Y. Zhang, and H.-D. Cheng, "Crackgan: Pavement crack detection using partially accurate ground truths based on generative adversarial learning," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 2, pp. 1306–1319, 2020.

[47] S. Xie and Z. Tu, "Holistically-nested edge detection," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 1395–1403.

[48] M. Pu, Y. Huang, Y. Liu, Q. Guan, and H. Ling, "Edter: Edge detection with transformer," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022, pp. 1402–1412.

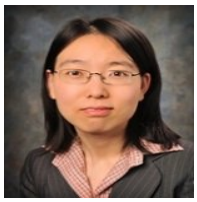
[49] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Third International Conference on Learning Representations (ICLR)*, 2015.

[50] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.
[51] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 2980–2988.
[52] J. König, M. D. Jenkins, M. Mannion, P. Barrie, and G. Morison, "Optimized deep encoder-decoder methods for crack segmentation," *Digital Signal Processing*, vol. 108, p. 102907, 2021.



Neelesh Verma received the B.S. degree from the Department of Computer Science and Engineering, Indian Institute of Technology Bombay, India, in 2020. He is currently a Master's student at the Department of Computer Science, Stony Brook University, working under Prof. Haibin Ling. His research interests include Computer Vision, Machine Learning, Robustness, and Security of Machine Learning systems.

Mengyang Pu received the Ph.D. degree from the School of Computer and Information Technology, Beijing Jiaotong University, China, in 2022. In 2020, she was a visiting Ph.D. student in Computer Science at Stony Brook University. She is currently a lecturer at the School of Control and Computer Engineering, at North China Electric Power University. Her research interests include computer vision, image processing, and deep learning.



Mei Zhang received the B.S. and M.S. degrees from Tsinghua University in 1998 and 2001, respectively, and the Ph.D. degree from the University of Maryland, College Park in 2008. After that, she joined Montclair State University as an assistant professor in the fall of 2008. Then, in the fall of 2009, she moved to Rowan University where she is now an associate professor in the Department of Accounting and Finance. Dr. Zhang's research interests include financial reporting, valuation, auditing issues, data analysis, and transportation logistics.

Danil Prokhorov was a Research Engineer with the St. Petersburg Institute for Informatics and Automation, Russian Academy of Sciences, Saint Petersburg, Russia. He has been involved in automotive research since 1995. He was an Intern with the Scientific Research Laboratory, Ford Motor Company, Dearborn, MI, USA, in 1995. In 1997, he became a Research Staff Member with Ford Motor Company, where he was involved in application-driven research on neural networks and other methods. Since 2005, he has been with Toyota Technical Center, Ann Arbor, MI, USA. He is currently Deputy CRO of TRINA, Ann Arbor, MI.

Jie Wei received his B.S, M.S, and Ph.D. from the University of Science and Technology of China in 1989; Institute of Software in Chinese Academy of Sciences in 1992; and Simon Fraser University in 1999, all in computer science. Since 1999 he has been on the faculty of the Dept. of Computer Science, City College and the Graduate Center, the City University of New York, where he is now a professor. His research interests are multi-modal computing, computer vision, medical imaging, and machine learning. His research has been supported by NIH, NSF, AFOSR, AFRL, ARO, and ONR.

Haibin Ling received the B.S. and M.S. degrees from Peking University in 1997 and 2000, respectively, and the Ph.D. degree from the University of Maryland, College Park, in 2006. From 2000 to 2001, he was an assistant researcher at Microsoft Research Asia. From 2006 to 2007, he worked as a postdoctoral scientist at the University of California Los Angeles. In 2007, he joined Siemens Corporate Research as a research scientist; then, from 2008 to 2019, he worked as an Assistant Professor and then Associate Professor at Temple University. In the fall of 2019, he joined Stony Brook University as a SUNY Empire Innovation Professor in the Department of Computer Science. His research interests include computer vision, augmented reality, medical image analysis, machine learning, and human-computer interaction. He received the Best Student Paper Award at ACM UIST (2003), the Best Journal Paper Award at IEEE VR (2021), the NSF CAREER Award (2014), the Yahoo Faculty Research and Engagement Award (2019), and the Amazon Machine Learning Research Award (2019). He serves or served as Associate Editors for IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI), Pattern Recognition (PR), and Computer Vision and Image Understanding (CVIU). He has served as Area Chair various times for CVPR and ECCV. He is a Fellow of IEEE.