Smart Structures and Systems, *Vol. 35, No. 5 (2025) 267-284*
https://doi.org/10.12989/sss.2025.35.5.267

Unsupervised deep learning method for concrete and asphalt crack segmentation using vision transformer and probability thresholding

Muhammad Tanveer [1a] and Soojin Cho*[1,2]

[1] *Department of Civil Engineering, University of Seoul, Dongdaemun-gu, Seoul 02504, South Korea*
[2] *Graduate School of Urban Bigdata Convergence, University of Seoul, Dongdaemun-gu, Seoul 02504, South Korea*

(*Received November 14, 2024, Revised June 18, 2025, Accepted June 23, 2025*)

Abstract. Crack detection is vital for maintenance of civil structures. Recently, deep learning-based semantic segmentation models have shown promise in accurately identifying cracks. However, these methods require laborious manual data labeling. To address this, an unsupervised learning-based crack segmentation method was proposed, using a self-supervised Vision Transformer (ViT) as a backbone network to learn crack patterns from unlabeled images. A diverse crack image dataset with various crack types and backgrounds was used to train the network without time-consuming labeling, and to test the model after constructing ground-truths. The model was optimized with unsupervised contrastive loss function parameters, and probability thresholding was applied to enhance detectability by eliminating low confidence pixels, reducing false positives. On 1,399 test images, the unsupervised model achieved a mean F1-score of 75.02% and a mean Intersection over Union (mIoU) of 63.01%, with mIoU improving to 66.14% after thresholding, which shows great detection performance of unsupervised model. The model's application to high-resolution real crack images using a sliding window technique further demonstrated its suitability for field use, offering an efficient solution for real-time structural monitoring. These findings highlight the potential of unsupervised deep learning for crack detection, significantly reducing the need for manual labeling while delivering strong performance.

Keywords: crack detection; probability thresholding; sliding window technique; structural monitoring; unsupervised learning

1. Introduction

Infrastructure, including buildings, bridges, roads, and tunnels, is prone to damage from aging, heavy loads, environmental changes, and natural hazards. Common damages like cracks, spalling, corrosion, and efflorescence can reduce the infrastructure's lifespan and pose disaster risks. To monitor infrastructure health, many countries employ structural health monitoring (SHM). Traditional methods, like visual inspection, are time-consuming and labor-intensive. Therefore, computer vision (CV) techniques are gaining popularity for automated, efficient, and cost-effective inspection. Among various types of damages, cracks are considerable challenging to detect due to their varying sizes and widths. For instance, cracks in asphalt driveways and roads commonly caused by infiltrating water and temperature fluctuations tend to be long and straight, whereas cracks in concrete structures exhibit curvier shapes. The cracks can progress to an alligator crack if left unrepaired. Traditional CV algorithm such as Gabor filtering, Otsu's method and morphological approaches (Abdel-Qader *et al*. 2003, Kabir *et al*. 2009), rely on handcrafted rules but can be unreliable when images contain noise, uneven lighting, or complex backgrounds due to their sensitivity to parameter settings. Recent research has made extensive use of the categorization capabilities of machine learning (ML) methods. ML techniques used handcrafted features fed to algorithms like support vector machines (SVMs), Random Forest (RF), and naïve Bayes classifiers (NBC) for improved accuracy (Nick *et al*. 2015, Prasanna *et al*. 2016, Wang *et al*. 2015). Since these handcrafted features quantify the characteristics of typical cracks, however, these approaches face limitations in changing environments, noisy images, or similar objects in the scene. Deep Convolutional Neural Networks (CNN) have emerged as a solution, autonomously extracting features and demonstrating promising results in detecting damage on concrete and asphalt structures (Dorafshan *et al*. 2018, Yang *et al*. 2018a, Yusof *et al*. 2019).

Deep Learning has revolutionized computer vision, yielding significant results in image categorization (Krizhevsky *et al*. 2017), object localization (Ren *et al*. 2017), and semantic image segmentation (Long *et al*. 2015). In the context of crack detection, various deep learning models offer distinct advantages. Studies have explored these models, each with its unique strengths. For instance, (Cha *et al*. 2017) compared the efficiency of CNN-based crack detection with traditional edge detection methods. (Zhang *et al*. 2019) adopted fully convolutional layers and varying dilation rates for feature extraction, while (Kim and Cho 2018) investigated the use of AlexNet for crack morphology detection. There have been numerous studies that have explored the use of deep learning models for crack

*Corresponding author, Ph.D., Professor,
E-mail: soojin@uos.ac.kr
[a] Ph.D. student, E-mail: tanveer715@uos.ac.kr

ISSN: 1738-1584 (Print), 1738-1991 (Online)

detection and localization (Lv *et al*. 2023, Ramli *et al*. 2021, Wang *et al*. 2020, Wang and Nishio 2024). Ali *et al*. (2021) built a custom CNN model for crack detection and localization outperforming other pre-trained models in accuracy. (Wang and Xiang 2021) compared the performance of Sobel and Canny algorithms with CNN model in railway sleeper crack detection. They proposed that CNN based model outperformed conventional Sobel and Canny algorithms in edge detection. (Solhmirzaei *et al*. 2024) proposed a deep learning-based computer vision method to detect cracks in ultra-high-performance concrete (UHPC) beams, overcoming traditional inspection limitations. Results showed enhanced crack detection efficiency, enabling autonomous infrastructure monitoring. However, as crack characteristics vary in shapes and locations, making image classification and object detection less precise regarding crack properties like length and shape (Kim and Cho 2019, Yang *et al*. 2018b). In recent years, deep learning has delivered impressive results in crack segmentation applications by automatically learning distinctive features from the training data (Minh *et al*. 2022, Zhou *et al*. 2023). Encoder-decoder networks like fully convolution networks (FCN) (Long *et al*. 2015), UNet (Ronneberger *et al*. 2015) and DeepLabV3+ (Chen *et al*. 2018) have enabled feature extraction in image segmentation. FCN excels in detecting large cracks but struggled with small ones (Dung 2019, Zhou and Song 2020). Liu and Wang (2022) proposed a UNet-based model that predicts thin and tiny cracks with improved accuracy, while Shen *et al*. (2023) optimized DeepLabV3+ model for crack detection that yielded 91.7% IoU with thin cracks in noisy background conditions. Tang *et al*. (2022) modified U-Net for pixel wise crack detection using high resolution images. Recently, Song (2024) introduced a DeepLabV3+ based deep learning approach for automated detection of post-earthquake structural damage, achieved high segmentation accuracy and strong generalization. The method demonstrated its potential for advancing automated structural health monitoring. Vision transformers (ViTs) (Dosovitskiy *et al*. 2020), have enhanced crack detection by integrating global contextual information via self-attention mechanisms. Self-attention mechanisms are the interactions between input sequences that assist the model in determining which region merits greater attention during training. Wang and Su (2022) proposed a hierarchically transformer-based encoder called SegCrack, outperforming other CNN-based segmentation models for concrete crack detection. Furthermore, Xiang *et al*. (2022) combined YOLOv5 and ViTs for pavement crack detection, successfully extracted contextual information around crack patches. Shamsabadi *et al*. (2022) proposed a novel transformer based framework concluded that transformer based model detect cracks better in the complex background conditions.

However, the studies are related to supervised learning approaches requiring huge amount of labeled dataset for training the models. For semantic segmentation, data should be labeled pixel-wise that is very time consuming and laborious. Manual labeling of damages in civil structures is a complex process that demands specialized expertise (Bang *et al*. 2019, Jha and Babiceanu 2023, Panella *et al*. 2022). Crack image, on average, require approximately 20 minutes for labeling, although certain complex alligator network crack images required up to 1 hour for precise labeling (Song *et al*. 2023). Structural damage, especially cracks and rebar exposure, often exhibits small sizes and intricate geometrical features that pose labeling challenges. Additionally, damages like spalling and efflorescence have amorphous shapes, further complicating the labeling task. Manual labeling introduces inconsistencies due to human subjectivity, with variations among experts due to differences in perception and interpretation of visual cues indicating damage. These subjective differences result in inconsistencies within labeled datasets, affecting the performance of supervised learning models. To address these limitations, recent efforts have introduced unsupervised learning approaches for pixel-wise and classification tasks, eliminating the need for costly and time-consuming labeling work.

Unsupervised learning models have been developed significantly over the past few years. Traditional unsupervised learning algorithms, such as K-means and hierarchical clustering, focus on clustering data points based on feature similarity. Then dimensionality reduction algorithms like Principal Component Analysis (PCA) and t-distributed Stochastic Neighbor Embedding (t-SNE) emerged, aiming to simplify complex data while preserving its inherent structure (Van Der Maaten and Hinton 2008). With the advent of deep learning, unsupervised learning approaches began to incorporate neural networks. Autoencoders were early adopters of this approach, excelling in efficient data representation, noise reduction, and feature extraction (Hinton and Salakhutdinov 2006). More advanced architectures, like Generative Adversarial Networks (GANs) introduced by Goodfellow *et al*. (2020), represented a significant leap forward in unsupervised learning, particularly in image generation tasks.. Furthermore, recently numerous investigations have strived to tackle the issue of semantic segmentation without supervision. These unsupervised states of the art methods were trained and tested on universal datasets like Cityscapes (Cordts *et al*. 2016), CocoStuff (Caesar *et al*. 2018), and they distinguished between different objects without using any true labels. These methodologies aimed to establish semantic correlations at the pixel level from self-supervised feature learning. Ji *et al*. (2019) proposed Invariant Information Clustering (IIC) technique that maximized mutual information of cluster assignments between the features of two differently augmented images, while Pixel-level feature Clustering using Invariance and Equivariance (PiCIE) proposed by Cho *et al*. (2021) enhances IIC results by leveraging photometric and geometric invariances. However, these techniques heavily rely on data augmentation and learning semantic consistency without prior knowledge remains challenging. Consequently, recent strategies (Hamilton *et al*. 2022, Ke *et al*. 2022, Yin *et al*. 2022, Zadaianchuk *et al*. 2022) have implemented the Vision Transformer (ViT) model, trained in a self-supervised way, i.e., Self-distillation with no labels (DINO) (Caron *et al*. 2021), as their core architecture within the

context of self-supervised learning, implementing self-distillation using an approach that involves exponential moving average updates. The authors of DINO demonstrate that its class-attention can generate object segmentations that are both localized and hold semantic significance. (Hamilton *et al*. 2022) presented Self-supervised Transformer with Energy-based Graph Optimization (STEGO) employing knowledge distillation, which learns the associations between features extracted from DINO. Among these methods, STEGO using ViTs achieved the best accuracy on datasets like Cityscapes and CocoStuff.

Though there is a notable progress in the unsupervised deep segmentation models for general tasks, but there have been limited studies exploring their application in the CV-based inspection of civil structures. At present, some of pavement crack detection methods utilize unsupervised machine learning (ML) or deep learning (DL) approaches. (Mubashshira *et al*. 2020) introduced an unsupervised model for pavement crack detection. The pavement region is first detected by color histogram analysis, and then the noise is thresholded by the Otsu's method. Cracks are segmented by K-means clustering with morphological closing operation to connect fragmented crack pixels. Zhang *et al*. (2020) utilized generative adversarial network (GAN) as self-supervised method for pixel-level crack detection. They introduced a novel discriminator that has a large field of view and focused on the crack patch as real to counteract the data imbalance problems in crack detection. Similarly, Duan *et al*. (2020) employed a GAN to transform crack images into binary form, introducing a cycle-consistent loss to enhance crack localization accuracy. They used convolutional neural networks (CNNs) for feature extraction and fully convolutional networks (FCNs) as discriminators. Yu *et al*. (2020) proposed Adversarial Image-to-Frequency Transform (AIFT) as an unsupervised method for detection of defects in road dataset. They used adversarial learning to optimize the model performance by image-to-frequency mapping function that outperformed traditional classification methods. Li *et al*. (2021) explored an approach that combine a deep CNN and K-means clustering in an unsupervised learning setting to eliminate the need for manually labeled ground truth images. Wu *et al*. (2021), introduced an unsupervised autoencoder model named Memory-Augment Autoencoder (MemAE) for crack classification, utilizing road images to capture additional features and employing self-supervised pre-training and post-processing. Furthermore, Wang *et al*. (2022) also adopted autoencoder as encoder-decoder module for unsupervised defect detection in pavement images. They enhanced the performance of the model in defect detection across various scenarios by adopting patch-aware mutual reasoning and adaptive soft gated anomaly measurement methods. Aung and Kumwilaisak (2023) proposed unsupervised method that based on graph neural network clustering for pavement crack segmentation. They combined Canny edge detection and morphological operations to eliminate noise while an unsupervised graph neural network clustering model was employed for pixel level crack segmentation. These unsupervised approaches in crack detection focus on using auto-encoders, GANs and CNN-based encoder-decoders for the tasks of classification, detection and segmentation. However, there is a notable absence of studies exploring the potential of vision transformers for unsupervised crack segmentation, therefore, this area of research remains open to research in this regard. Furthermore, most of the approaches were evaluated on limited datasets and did not provide experiments in order to validate their effectiveness on real-world scenarios. The proposed method in this paper aims to address these limitations by shifting the focus towards unsupervised crack segmentation using vision transformer, improving accuracy, reducing labor-intensive processes, and decreasing the demand for extensive labeled datasets and emphasizing the need for pixel-to-pixel crack analysis in civil structure assessment. In doing so, it seeks to provide a more robust and efficient solution for unsupervised civil structure inspection.

In this paper, an automated crack segmentation method was proposed by training an unsupervised semantic segmentation network using concrete and asphalt images without labeling and improving the result by applying probability thresholding method. Unlike traditional supervised and semi-supervised techniques, which require extensive labeled datasets, the proposed method employs the STEGO model to extract crack features from unlabeled concrete and asphalt images. This unsupervised approach is rare in the domain of civil infrastructure and represents a substantial advancement by eliminating the need for costly and labor-intensive labeling, thereby enhancing the method's applicability and scalability for real-world scenarios. Various crack image dataset sourced from Kaggle, which contains different crack types and background conditions, were used for training the network without time-consuming labeling, as well as for testing the trained network after constructing ground-truths. The parameters of contrastive loss function used in the model have been fine-tuned for the crack image datasets. Beyond the employment and tuning of the STEGO model, a thresholding technique on the probability map output without costly labeling work by the model was additionally proposed to further improve the segmentation performance with omitting false alarms. Experimental evaluation of the proposed method is performed on high resolution real crack images. While evaluating the performance, the sliding window technique was additionally proposed to analyze the high-resolution images in small sections and make predictions for each section separately. The results of this evaluation demonstrate the validity of the proposed method and its ability to accurately segment cracks in high resolution images, demonstrating its potential to contribute to the advancement of automated crack detection in civil engineering.

2. Unsupervised model architecture and post-processing techniques to improve performance

This section explains the concept and methods used in self-supervised learning that can be leveraged for the unsupervised semantic segmentation. Architecture of the

SEGO model is described, along with the employed unsupervised loss function in the model. Then, the post-processing techniques used to enhance the performance, such as probability thresholding and sliding window for high-resolution images, and the metrics used to evaluate performance are described here.

2.1 STEGO (*Self-supervised Transformer with Energy-based Graph Optimization*) model

STEGO model was proposed by Hamilton *et al*. (2022) that is based on unsupervised learning technique for semantic segmentation. This state-of-the-art model was trained on different datasets like Cityscapes, CocoStuff without using the ground truth labels (i.e., in unsupervised way). STEGO achieved better performances on the mentioned datasets as compared to other unsupervised or self-supervised models. Fig. 1 illustrates the architecture of STEGO model. The STEGO model employs a frozen Vision Transformer (ViT) backbone, pretrained in a self-supervised manner on ImageNet (Caron *et al*. 2021). This ViT backbone serves as an integral feature extractor, capturing essential visual information from input images, and contributes significantly to the model's ability to perform accurate semantic segmentation tasks. The feature extractor provided by the backbone network undergoes further modification and enhancement through the integration of a segmentation head. The segmentation head consists of two sequential convolution layers with a lightweight design, incorporating a nonlinear ReLU (Rectified Linear Unit) activation function between the layers. Within the model architecture, the backbone network distills the feature correspondences tensor, and the segmentation head refines these features, transforming them into a low-dimensional segmentation embedding.

Feature correspondence is the process of identifying and matching corresponding features between different images to establish relationships between the same objects in different images (Beymer 1996). It provides a high-quality learning signal for unsupervised segmentation. The backbone network N maps an image to a feature tensor with C channels while keeping the spatial resolution. The lightweight segmentation head S that converts the feature tensor into a code space of dimension K ($K < C$), and S aims to observe a nonlinear projection. Suppose $f \in R^{CHW}$ and $g \in R^{CIJ}$ are two feature tensors extracted from images $x \in R^{HW}$ and $y \in R^{IJ}$ as shown in Fig. 1, then the feature correspondence tensor F of these two feature tensors can be expressed as

$$F^{hwij} = \sum_{c} \frac{f_{chw}}{|f_{hw}|} \frac{g_{cij}}{|g_{ij}|} \tag{1}$$

where F^{hwij} describes the cosine similarity at spatial position between feature tensor $f(h, w)$ and $g(i, j)$. If these two feature tensors f and g equals, then the feature correspondence will compare the level of resemblance between two regions of the same image. By investigating segments of the correspondence tensor F at a given location (h, w), the relationship between two images according to the feature extractor can be visualized. This feature correspondence tensor demonstrates a strong positive correlation with the co-occurrence tensor representing true label relationships. This indicates that the identified features align closely with the actual labels, implying a robust and accurate mapping between the two data representations. And the corresponding segmentation features $s := S(f) \in R^{CHW}$ and $t := S(g) \in R^{CIJ}$ are converted from f and g, respectively, by the segmentation head S.

By element-wise multiplication of the feature correspondence tensor F and segmentation feature tensor S, the loss function to train the STEGO model is built as

$$L_{corr}(x, y, b) = -\sum_{hwij} \left(F^{SC}_{hwij} - b\right) \max\left(S_{hwij}, 0\right) \tag{2}$$

where x and y are pair of images in which features tensors F and S are calculated, and b is a hyper-parameter that incorporates uniform "negative pressure" to prevent collapse. SC means spatial centering as feature correspondences represent how different parts of an image or scene are related to each other based on their features. Spatial centering may involve reconfiguring these correspondences in such a way that the algorithm pays more attention to the spatial context or structure of the data. This can help improve the accuracy and balance of the optimization process, ensuring that the segmentation algorithm is not overly influenced by isolated or noisy data points but instead captures the broader spatial patterns and relationships among features. Minimizing L_{corr} relative to S encourages elements of S to be large when the elements of $F - b$ are positive, and small when the elements of $F - b$ are negative. More specifically, because the elements of F and S are cosine similarities, this imposes an attractive or repulsive force on pairs of segmentation features that is equal to the strength of their feature correspondences.

The STEGO model during training replicate the distilling feature corresponding tensors F and S across pairs of images as image and itself, image and K-nearest neighbors, and random other images respectively. These methods are known as self correspondence, KNN correspondence and random correspondence. Self correspondence typically involves comparing each element (e.g., pixel or region) within the input data to itself. It helps establish a baseline understanding of how similar or dissimilar the elements are within the same image or scene. This self-similarity information is valuable in distinguishing objects from the background and can serve as a reference point for identifying object boundaries. KNN correspondence extends the idea of self-correspondence by comparing each element not just to itself but also to its K nearest neighbors in the feature space. This approach leverages the similarity between an element and its neighbors to provide context and can assist in grouping similar elements together, potentially aiding in the identification of object clusters. Random correspondence introduces an element of randomness to the correspondence process. It serves as a form of background or noise representation, helping the segmentation algorithm

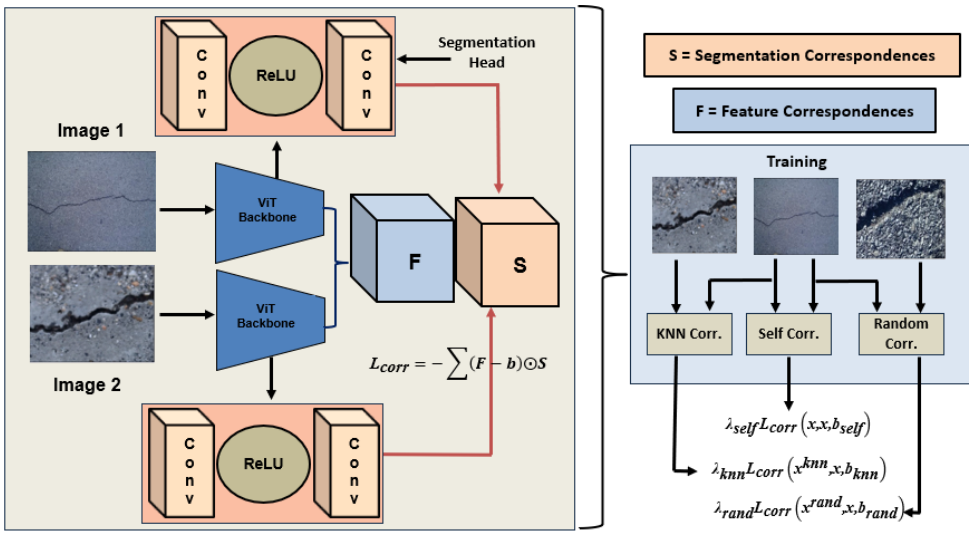


Fig. 1 Overview of STEGO model architecture

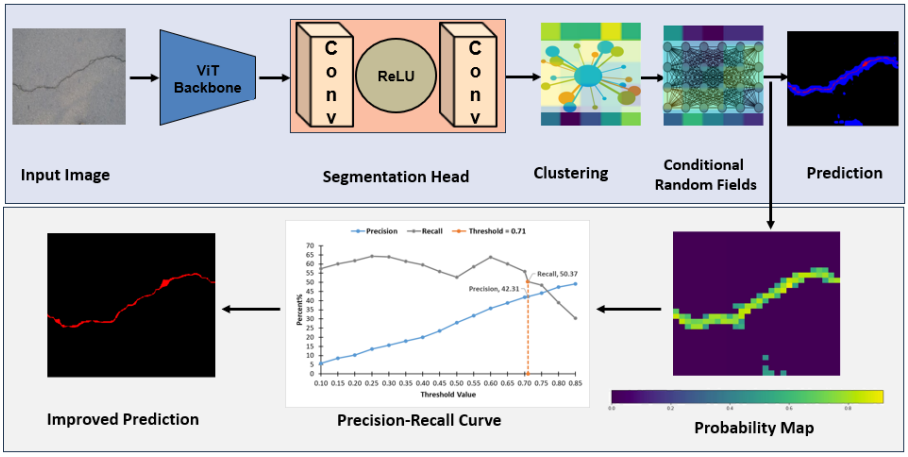


Fig. 2 Overview of model training and prediction process

differentiate between true object-related correspondences and random or unrelated correspondences. This is crucial in distinguishing objects from non-object elements. The self-correspondence and KNN correspondence give positive supervisory signals to the model and the random image correspondence provides negative signals to keep the balance during training. These correspondence methods collectively contribute to the STEGO contrastive full loss function as

$$L = \lambda_{self}L_{corr}(x,y,b_{self}) + \lambda_{knn}L_{corr}(x^{knn},x,b_{knn}) + \lambda_{rand}L_{corr}(x^{rand},x,b_{rand}) \quad (3)$$

where the $\lambda's$ and $b's$ are the hyper-parameters that regulate the positive-to-negative pressure ratio and the equilibrium of the learning signals, respectively. However, it is observed that keeping the ratio of λ_{self} and λ_{rand} twice the λ_{knn} during training, the model showed optimal performance (Hamilton *et al.* 2022). In general, the parameters $b's$ are dependent on both dataset and network architecture and tuned manually according to dataset. The optimization of parameters b for cracks dataset used in the proposed method are discussed in section 3.3 in details.

Fig. 2 illustrates the overall process of the proposed method, from training of the STEGO model to visualization of prediction results and prediction improvement. Initially, input images traverse through the pretrained self-supervised ViT model, followed by the lightweight segmentation head. This segmentation head operates by generating segmentation embeddings from the processed images. The resulted segmentation embeddings are clustered according to the cosine similarity. To enhance the quality of these clusters and refine the predictions, Conditional Random Fields (CRF) were employed. CRF is a classification error reduction technique that plays a crucial role in improving noisy or low-resolution class predictions. It achieves this by aligning predictions with edges and color-correlated regions in the image, thereby improving the accuracy and coherence of the final segmentation results. The detailed discussion about the process of prediction improvement was discussed below in section 2.2.

2.2 Probability thresholding

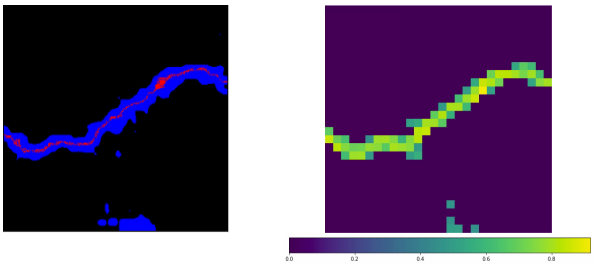
Though the STEGO model has the capability to segment fine objects, thin cracks are still hard to be segmented with accurate boundaries. The predicted cracks by the trained STEGO model were found to be thicker than the ground truth with more false positive detections as to be discussed in section 4.1. To mitigate this issue, a probability thresholding that a common method removing false positives in crack segmentation (Liu *et al*. 2021, Zou *et al*. 2012) is proposed in this study. It involves creating a probability map, which assigns a probability value to each pixel in an image, indicating the likelihood of that pixel belonging to a crack. This probability map can filter out false positive detections by removing pixels with low probability values and retaining only pixels with high probability values.

The STEGO model generates a raw output score, for each pixel in the image for each class like cracks and non-crack or background. Then model used CRF that refine the clusters predictions, contains a softmax function that applied to these scores to convert them into probability values. For a given pixel i, the softmax function for the 'crack' class can be calculated as

$$p_{i,c=1} = \frac{e^{z_{i,c=1}}}{e^{z_{i,c=0}} + e^{z_{i,c=1}}} \quad (4)$$

where $p_{i,c=1}$ is the estimated probability that pixel i is part of a crack, $z_{i,c=1}$ is the raw output score of pixel i for the 'crack' class, and $e^{z_{i,c=0}}$ and $e^{z_{i,c=1}}$ are the exponential values of the raw output scores of pixel i for the 'non-crack' and 'crack' classes, respectively. The sum of the probabilities for the 'non-crack' and 'crack' classes becomes one for each pixel. The estimated probability $p_{i,c=1}$ for all pixels can be used to generate a probability map to be used for the post-processing.

Fig. 3 is an example of visual representation of a prediction image by the STEGO model and its accompanying probability map. In the prediction, the model's identified cracks were observed to be wider than the actual cracks, leading to an increase in false positives painted in blue. To illustrate the confidence levels associated with these predictions, a color-coded probability map was generated. In this map, color intensity corresponds to the model's estimated probability that a given pixel is part of a crack: a higher intensity color signifies a higher probability. It was observed that the false positive detections generally exhibited less intense colors, indicating lower associated probability scores compared to true positives. These visual observations demonstrate the utility of thresholding in the post-processing step. By applying a threshold to the probability map, predictions associated with lower probability scores (less intense colors) can be removed, thereby eliminating many of the false positive detections. The optimal threshold value can be determined by considering the trade-off between precision and recall.



(a) Prediction (b) Probability map

Fig. 3 Example of prediction by a segmentation model in colors (red: true positive, green: false negative, blue: false positive) and its probability map

2.3 Sliding window technique for high-resolution images

A sliding window technique was employed for processing high-resolution real images in proposed method performance evaluation to mitigate potential computational expenses and avoid exceeding the model's input size constraints. The sliding window technique divides an original image with the size of $H \times W$ into smaller patches using a predetermined-size (e.g., $h \times w$), as shown in Fig. 4, and then process each patch individually using the segmentation model. Then, the prediction result of the original image can be reconstructed by stitching the predictions from the patches. In high-resolution image processing for proposed method performance evaluation, the adoption of a sliding window technique is driven by the need to strike a delicate balance between computational efficiency and evaluation accuracy. The inherent challenge lies in the computational cost and potential input size limitations of the model when handling large images.

The sliding window technique possesses several significant technical advantages. One of its key strengths lies in its capacity to optimize the utilization of computational resources. By processing smaller, manageable patches, ensure the efficient use of available computational resources, making it especially valuable in resource-constrained environments. Moreover, the technique lends itself well to parallel processing, where each patch can be processed independently. Additionally, the sliding window technique offers improved localization capabilities.

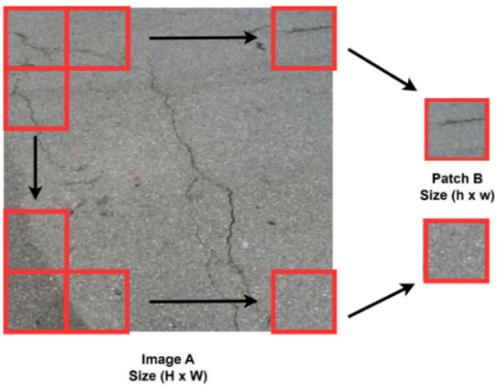


Fig. 4 Illustration of sliding window technique for high resolution image inference

Analyzing image patches individually enables to gain fine-grained insights into the model performance at various regions of the image. This not only aids in identifying potential weaknesses but also allows for a more detailed understanding of the model to localized features or patterns within the image. Thus, the sliding window technique serves as a valuable strategy for the precise processing of high-resolution images while adeptly navigating potential constraints associated with the model's input size limitations. Its resource optimization, parallel processing capabilities, and support for enhanced localization collectively contribute to its efficacy in the context of image analysis and evaluation.

2.4 Evaluation metrics

In the field of semantic segmentation, several metrics are commonly used to evaluate the performance of models. The proposed method was evaluated using mean precision (mPrecision), mean recall (mRecall), mean F1-score (mF1-score), and mean intersection over union (mIoU) for each class including crack and background. These metrics help to provide a comprehensive understanding of the model's performance by considering different aspects of the predictions, such as the ability to correctly identify pixels of interest, the balance between false positives and false negatives, and the similarity between the predicted and ground truth masks.

Precision is a measure of the model's ability to correctly identify the pixels that belong to the object of interest. It is calculated as the number of true positive pixels divided by the total number of true positive and false positive pixels. Meanwhile, Recall is a measure of a model's ability to correctly identify all relevant instances. It is calculated as the number of true positives divided by the number of true positives plus the number of false negatives. Precision and Recall can be formulated as

$$mPrecision = \frac{TPs}{TPs + FPs} \tag{5}$$

$$mRecall = \frac{TPs}{TPs + FPs} \tag{6}$$

where, true positives (TPs) are the pixels in the prediction that overlap with the ground truth, false positives (FPs) are the pixels in the prediction that do not overlap with the ground truth, and false negatives (FNs) are the pixels in the ground truth that do not overlap with the prediction or referred as missing cracks pixels.

F1-score is a measure that balances Precision and Recall; it is the harmonic mean of Precision and Recall, and a high F1-score indicates that the model has a good balance of Precision and Recall. F1-score is formulated as

$$mF1 - score = 2 \times \frac{Precision \times Recall}{Precision + FPs} \tag{7}$$

mIoU is a measure of the similarity between the predicted and ground truth segmentation masks. It is calculated as the area of the intersection of the predicted and ground truth masks divided by the area of the union of the predicted and ground truth masks as

$$mIoU = \frac{TPs}{TPs + FPs + FNs} \tag{8}$$

3. Training of unsupervised model for crack detection

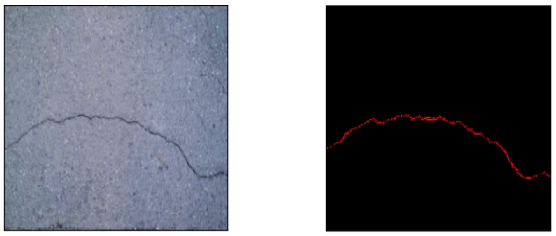
In this section, the details of the dataset that were used for testing and training the model are described, as well as the training details, which include the hardware specification, model hyperparameters and loss function parameters optimization for the dataset.

3.1 Dataset

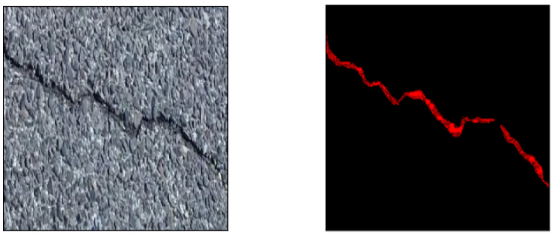
The proposed method adopted an unsupervised learning strategy for crack segmentation and was trained and evaluated using a composite dataset curated from multiple publicly available sources hosted on Kaggle. This dataset is constructed by aggregating eight widely used crack segmentation datasets: Crack Forest Dataset (CFD) (Shi *et al*. 2016), Crack-500 (Yang *et al*. 2020), Cracktree200 (Zou *et al*. 2012), DeepCrack (Liu *et al*. 2019), GAPS384 (Eisenbach *et al*. 2017), Eugen Muller (Ham *et al*. 2021), Rissbilder (Kulkarni *et al*. 2023), and Volker (Kulkarni *et al*. 2023). Fig. 5 depicts some images of these datasets with the corresponding ground truths. These datasets were specifically selected to ensure comprehensive representation of diverse crack characteristics and imaging conditions. The integrated dataset includes a broad range of crack types, such as longitudinal, transverse, diagonal, fine hairline cracks, and complex alligator cracking patterns. These cracks appear on both asphalt and concrete surfaces and vary in terms of width, continuity, orientation, and severity. In addition to geometric diversity, the dataset spans varied image acquisition environments, including differences in lighting, surface texture, background clutter, and image quality. The original image resolutions vary significantly across datasets, ranging from 440 × 360 pixels to 2000 × 1500 pixels. For consistency during training, all images were resized to 224 × 224 pixels. The combined dataset contains a total of 8,899 labeled images, partitioned into 7,500 training images and 1,399 testing images, which provides a sufficiently large and heterogeneous sample for robust unsupervised learning. This dataset diversity supports better generalization of the proposed method to unseen conditions. A detailed breakdown of each dataset, including structural material, resolution, and image count, is provided in Table 1.

3.2 Model training and hyperparameters

The unsupervised model was trained and evaluated using 8899 images presented in Table 1 without laborious labeling work. The training was performed on a workstation having four NVIDIA RTX Titan GPUs with a total memory of 24 GB each. The workstation has Ubuntu system with CUDA version 10 and cuDNN version of 10.2 using Python with PyTorch library. During the training and testing of the proposed model, a distributed training approach for faster

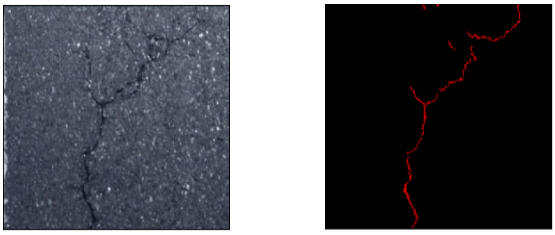


(a) Crack Forest Dataset (CFD) (Shi *et al*. 2016)

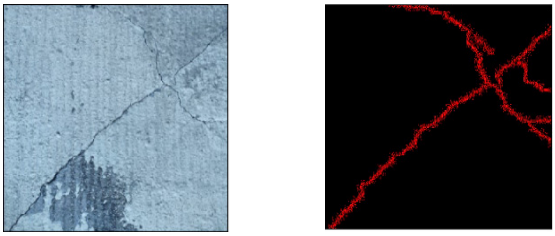


(b) Crack-500 (Yang *et al*. 2020)

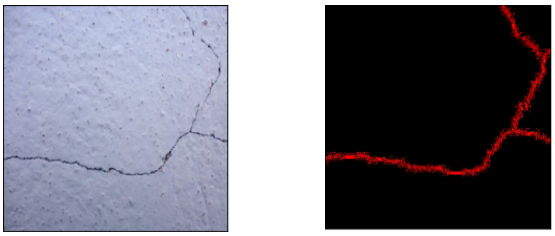
(c) Deep Crack (Liu *et al*. 2019)



(d) GAPS384 (Eisenbach *et al*. 2017)



(e) Rissbilder (Kulkarni *et al*. 2023)



(f) Volker (Kulkarni *et al*. 2023)

Fig. 5 Examples of images in Kaggle crack segmentation dataset and their ground truths for cracks

convergence and better performance was employed, as the model can process more data in parallel. The model

Table 1 Description of each dataset and its quantitative details

Datasets	Imaged structures	Resolution (Px × Px)	Training images	Testing images
GAPS384	Asphalt Pavement	1920 × 1080	433	76
CFD	Asphalt Road	480 × 320	100	18
Crack500	Asphalt Pavement	2000 × 1500	2858	505
Eugen Muller	Concrete Building	440 × 360	47	5
Deep Crack	Asphalt Pavement	544 × 384	443	78
Cracktree200	Asphalt Pavement	800 × 600	175	31
Volker	Concrete Building	512 × 512	842	148
Rissbilder	Concrete Building	512 × 512	2602	538
	Total		7500	1399

employed the ViT backbone of DINO, which was pre-trained on ImageNet (Deng *et al*. 2009). The model was trained using self-supervised learning without access to ground-truth labels. The self-supervised learning leverages the context of the input data to predict the context, facilitating the acquisition of generalizable feature representations.

During the training of the proposed model, two types of data augmentation techniques were applied to the dataset: geometric transforms and photometric transforms. Geometric transforms were applied using random horizontal flipping and random resized cropping with a scale of 0.8 to 1. These types of transforms are utilized to alter the spatial structure of an image, such as changing its size, rotation, or perspective. Photometric transforms, on the other hand, were applied to modify the visual appearance of the images using Color Jitter, random grayscale, and random Gaussian Blur. Color Jitter altered the brightness, contrast, saturation, and hue of the images, with each parameter set at factors of 0.3 for brightness, contrast, saturation, and 0.1 for hue. Random Grayscale converted the images to grayscale with a probability of 0.2. Random Gaussian Blur applied a Gaussian blur (with a 5 × 5 pixel kernel size) to the images, with an application probability of 50%. The application of these data augmentation techniques during training helped to increase the diversity of the training data.

During the training of the proposed model, all images were preprocessed by resizing them to a resolution of 224px × 224px. The decision to resize all images to a resolution of 224px × 224px is influenced by the relationship between image size, model complexity, and computational efficiency. Larger images increase model parameters and computational demands, potentially leading to overfitting and inefficient training. Conversely, a resolution of 224px × 224px reduces the computational load, making the model

training more efficient, while still maintaining sufficient detail for effective crack detection. The model was trained for a total of 5000 steps with a global batch size of 16, using an adaptive momentum optimizer known as ADAM, with a learning rate of 0.0005. For the evaluation of the model, a clustering-based approach known as the cluster probe was utilized. During the clustering step, cluster probe is used to group pixels or regions based on their visual characteristics, such as color, texture, or spatial proximity. Importantly, this step does not require ground truth supervised labels. After obtaining the clusters, the Hungarian matching algorithm (Kuhn 1955) is employed to align the predicted clusters with the ground truth labels. This alignment is done to evaluate the consistency between the predicted semantic segments and the ground truth annotations. The Hungarian matching algorithm determines the best assignment of predicted clusters to the corresponding ground truth labels by considering their similarities or consistencies. This approach allows to gauge the consistency of the predicted semantic segments with the actual labels, and it remains consistent regardless of any permutations in the predicted class labels. By combining unsupervised clustering with the Hungarian matching algorithm, the proposed approach provides a reliable means to evaluate and visualize semantic segmentation results without relying on ground truth labels during the clustering step.

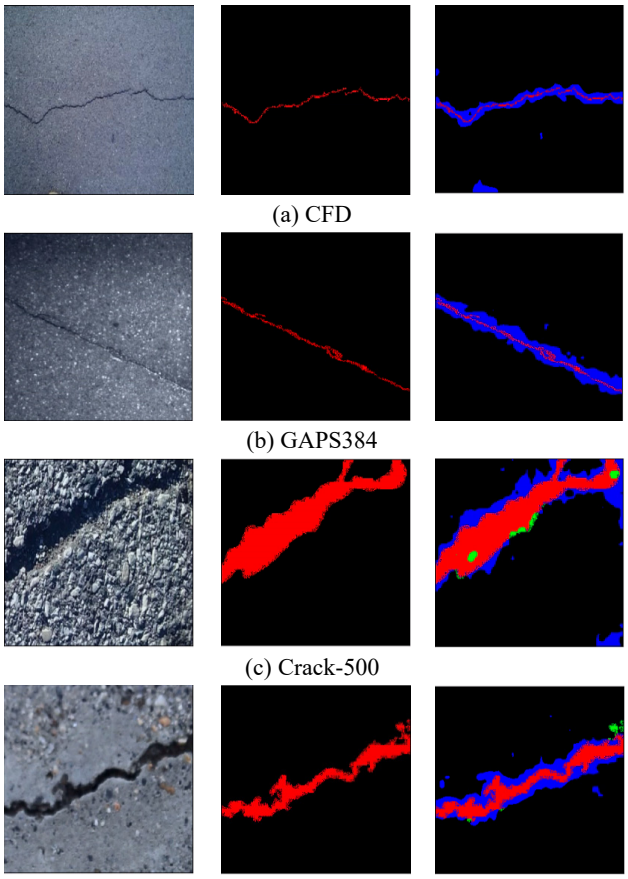
3.2 Loss function parametric optimization

To determine the optimized model based on loss function parameters, parametric study was performed on two ViTs backbones as the loss function parameters that highly depends on the dataset as well as pretrained model backbone. The loss function gives positive and negative signals to the model during training to differentiate between cracks and background pixels. In the Eq. (3) of loss function the parameters b's that belongs to three correspondences highly contribute to the model performance. In Table 2, the values of $b(self)$, $b(knn)$, $b(rand)$ represents the correspondences in the images itself, k-nearest neighbors, and random image respectively. However, in all experiments the loss function value of λ_{self} = 0.90, λ_{rand} = 0.90 kept twice than value of λ_{knn} = 0.45 during training. The evaluations metrics for each case including mPrecision, mRecall, mF1-score and mIoU values were also reported.

Table 2 shows the experiments using ViT small and ViT base backbone that have 21.9 million and 86.6 million parameters respectively. In the case of the ViT small backbone, it was observed that configurations with a higher "$b(self)$" value than "$b(knn)$" and "$b(rand)$" generally led to better model performance, optimized at 0.3. The configuration of $b(knn)$value was considerable selected lower than $b(self)$ and $b(rand)$ as increasing its values causes uncertainty between precision and recall scores as it is evident from case no 3 and case no 4. Similar trend was observed for the configurations of $b(rand)$ values and the difference between b(rand) values and $b(knn)$ values were kept lower as compared to $b(self)$ values. However, the impact of the $b(rand)$ parameter on the evaluation metrics was not as significant. Overall, these results suggest that the configuration with the $b(self)$ value (0.30) and the $b(knn)$ value (0.01) with $b(rand)$ value (0.05)

Table 2 Loss function parametric optimization using ViT Small and ViT base backbone

S.No	Backbone	$b(self)$	$b(knn)$	$b(rand)$	mPrecision%	mRecall%	mF1-Score%	mIoU%
1	ViT Small Parameters 21.9M	0.10	0.18	0.23	55.97	73.89	63.69	56.46
2		0.20	0.05	0.10	57.36	76.26	65.47	57.15
3		0.30	0.20	0.10	77.30	60.95	67.76	59.45
4		0.35	0.25	0.15	78.92	60.25	68.33	59.78
5		0.30	0.001	0.05	65.10	75.26	69.81	60.13
6		0.30	0.1	0.05	73.24	68.28	70.67	60.18
7		**0.30**	**0.01**	**0.05**	**67.82**	**78.01**	**72.55**	**61.43**
8		0.30	0.05	0.01	66.18	76.70	71.05	60.59
9		0.35	0.03	0.05	66.84	73.51	70.01	60.15
10		0.40	0.01	0.05	63.99	69.16	66.43	59.38
1	ViT Base Parameters 86.6M	0.75	0.22	0.20	73.21	60.30	66.13	59.29
2		0.50	0.15	0.05	57.93	75.70	65.63	57.18
3		0.50	0.20	0.10	76.33	67.72	71.76	61.13
4		0.50	0.22	0.15	78.31	68.28	72.95	61.80
5		0.50	0.22	0.20	79.38	64.78	71.34	61.11
6		**0.50**	**0.10**	**0.20**	**72.18**	**78.10**	**75.02**	**63.01**
7		0.60	0.10	0.20	65.85	77.28	71.10	60.63
8		0.65	0.05	0.25	64.23	76.58	69.89	60.10
9		0.70	0.05	0.15	62.13	68.50	65.15	58.97
10		0.73	0.10	0.23	73.65	61.86	67.24	59.52



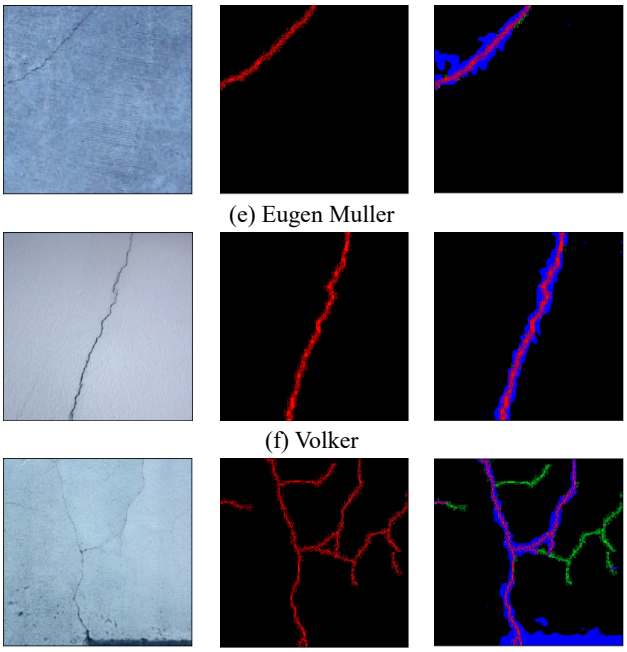
(a) CFD

(b) GAPS384

(c) Crack-500

(d) Deep Crack

Fig. 6 Continued



(e) Eugen Muller

(f) Volker

(g) Rissbilder

Fig. 6 Examples of (left) original images, (middle) their ground truths, and (right) predictions by the unsupervised model in colors (red: true positive, green: false negative, blue: false positive)

achieved the best performance, with a mPrecision of 67.82%, mRecall of 78.01%, mF1-score of 72.55%, mIoU of 61.43%. In the case of the ViT base pretrained backbone, it was found that higher "$b(self)$" values compared to "$b(knn)$" and "$b(rand)$" resulted in better model performance, with an optimized value of 0.50. Increasing or decreasing the $b(self)$ value from the optimized 0.50 affect the performance of the model. On the other hand, it was observed that lowering the $b(knn)$ values from 0.22 to 0.10 help the model in performance improvement. In contrast, the effect of $b(rand)$ value on the model performance was observed different as the balance between precision and recall improved by increasing the values from 0.05 to 0.20. The experiment results indicate that the optimal configuration was achieved with a $b(self)$ value of 0.50, $b(knn)$ value of 0.10, and $b(rand)$ value of 0.20. This configuration resulted in the highest performance of model, with a mPrecision of 72.18%, mRecall of 78.10%, mF1-score of 75.02%, and mIoU of 63.01%.

The results of the comparison between ViT base and ViT small in Table 2 indicated that for the same dataset, the ViT base backbone utilized larger values for the loss function parameters in comparison to the ViT small backbone. However, it is crucial to note that these parameter values are influenced by, and relative to, the specific architecture of each model. Furthermore, a more balanced precision and recall score was observed in the ViT base backbone, leading to an increased performance from ViT small. Despite this improvement, it should be noted that the parameters in ViT base are four times larger than those in ViT small, thus the decision between model parameters and accuracy should be carefully evaluated based on available computational resources. In conclusion, these results highlight the significant impact that the loss function parameters $b(self)$, $b(knn)$, and $b(rand)$ have on the performance of the model, as measured by the evaluation metrics.

4. Experimental validation

This section comprehensively elaborates on the validation process of the trained optimized model presented in section 3.3 using the crack dataset from Kaggle. The improvement of detectability, achieved through techniques such as probability mapping and thresholding, has been incorporated to enhance the model performance. Moreover, the section presents a detailed explanation of the application of the proposed method to real-world high-resolution images by using sliding window technique and comparison with other supervised and unsupervised methods. By incorporating these techniques and optimizations, the aim is to enhance the overall performance, accuracy, and utility of the unsupervised STEGO model in practical scenarios.

4.1 Experimental validation of trained model

Fig. 6 presents a qualitative evaluation of the proposed method based on the combination of different datasets in the test set. The test set consists of images with various background conditions and cracks. The first column lists the

original images, followed by ground truths, and the prediction results. It is evident from the visual examination of the model's predictions that it accurately identifies cracks in most images in the datasets. The background conditions of the CFD, and GAPS384 datasets are similar, as they were collected from asphalt pavement with narrow crack widths. As a result, the model predicts cracks with a thicker compared to the ground truth images, resulting in a higher number of false positive pixels being detected. As the model is unsupervised, it may also misjudge some textures that resemble cracks, such as stain regions. Crack500 and Deep crack datasets have cracks that are much wider than those in the other datasets. As a result, the model predicts cracks with fewer false positive pixels. However, some cracks in these images have fractures that are difficult to detect and are classified as false negative pixels. The images in the Volker, Eugen Muller, and Rissbilder datasets are sourced from concrete walls and thus have narrow crack widths. In the Eugen Muller and Volker images, the proposed method predicts cracks with a greater width. However, in the Rissbilder image, the cracks are of a different type, referred to as alligator cracks, which are more complex. The proposed method predicts some pixels of clear cracks with the same pattern as in other images with false positive pixels and misjudges dark stain regions that resemble the color and texture of cracks. However, cracks with narrow patterns are failed to detect and classified as false negatives in Rissbilder image.

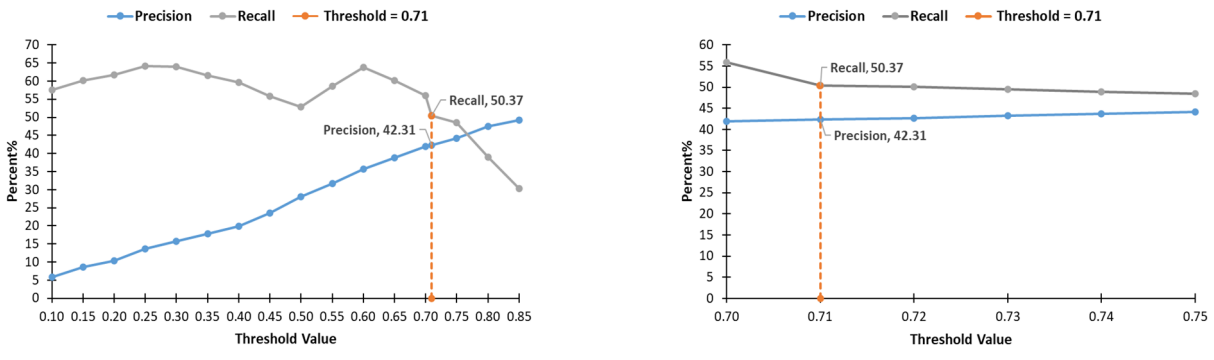
It has been demonstrated that the model accurately detects cracks in most images after analyzing the results of the model predictions across dataset. However, certain aspects have contributed to inconsistencies in its performance. Particularly, image resolution has distorted the typical expectations of crack width, with asphalt surfaces generally have wider cracks than concrete. However, the images used in the study depict cracks that appear to be narrower in asphalt images as shown in CFD and GAPS384 datasets. Datasets depicting more typical crack shapes, such as Volker, Eugen Muller, and Rissbilder had a greater number of false positives such as CFD, and GAPS384. These results imply that while the model is effective in general, it could misinterpret specific types of cracks or particular background conditions, resulting in higher false positives. Overall, the model predicts both longitudinal and transverse cracks and discontinuities in alligator cracks and unclear crack regions are observed. Despite some misjudgments, the proposed method demonstrates its robustness and generalizability, as it was trained on a diverse dataset.

It is observed that the proposed method predicted cracks with thicker than ground truths, result in higher false positives pixel. It can be improved with many pre and post-processing algorithms like subtraction (Kim and Cho 2019), dilation-erosion (Huang *et al*. 2022, Zhao *et al*. 2022), thresholding class activation mapping (Al-Huda *et al*. 2022, Konig *et al*. 2022, Zoubir *et al*. 2022) and probability mapping (Kim and Cho 2018, Shi *et al*. 2022). However, thresholding probability mapping technique was used in this study to improve the accuracy by removing false detections.

4.2 Detectability improvement using probability thresholding

Probability mapping leverages the output of the deep learning model, which typically provides a probability for each pixel belonging to a particular class - in this case, crack or non-crack. By applying a certain threshold to these probabilities, it is possible to filter out pixels with lower confidence, reducing false positives. In the context of the thicker detected cracks, probability mapping helps address this issue by providing granular, pixel-level control over the detected regions. Pixels within the detected crack regions but with lower probabilities could potentially represent the excessive thickness detected by the model. By excluding these lower probability pixels, the false positive rate is reduced, and the thickness of the detected cracks more accurately represents the ground truth.

The trade-off between precision and recall was explored to establish the optimal threshold for probability predictions. The threshold value is dependent on the dataset employed and the accuracy of the model's predictions. In the proposed model, most image predictions have false positive detections that are more frequent than false negatives, resulting in low precision scores and high recall scores. Fig. 7(a) provides the results of a parametric study of probabilities for threshold selection for Kaggle dataset. The precision and recall values in Fig. 7 specifically represent the detection performance for the crack class, as the primary focus was to determine the optimal threshold for improving crack detection accuracy. At the threshold

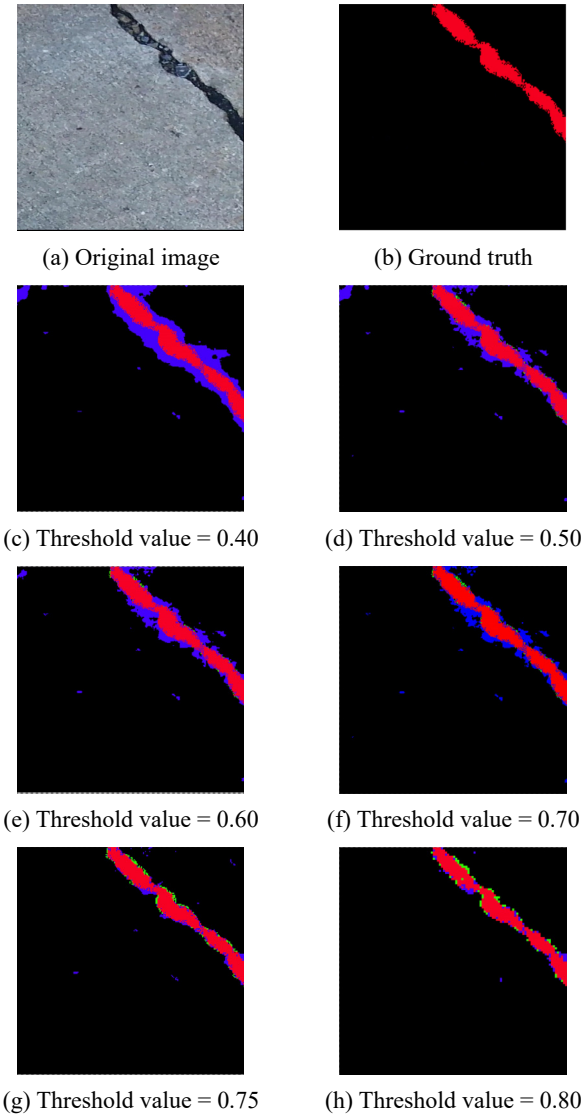


(a) Precision-Recall curve across thresholds

(b) Precision-Recall curve from 0.70 to 0.75 thresholds

Fig. 7 Probability map thresholding

lowered than 0.40, the difference between recall and precision is high, with a precision score is around 20%. As the threshold value is raised, precision and recall become increasingly balanced. At a threshold of 0.71, the model achieved improved precision and recall scores, with a precision of 42.31% and a recall of 50.37%. Fig. 7(b) depicted the detailed illustration of the precision and recall curve between the threshold values of 0.70 to 0.75. It can be observed that the precision score improved constantly in this range, however, recall score decreases after the threshold value exceeds 0.71. Therefore, further increasing in the threshold causes reduction in recall score which caused increasing in missing actual crack pixels. So, the optimal threshold for probabilities is 0.71 for this model and dataset, however it may be different according to the datasets and need. The probability threshold value of 0.71 in this study was determined through a global analysis of the precision-recall trade-off across the entire test set. This threshold was fixed and uniformly applied to all test images, rather than being dynamically adjusted on a per-image basis. Although adaptive thresholding could offer improved flexibility for individual cases, the use of a global threshold was prioritized to maintain consistency, ensure reproducibility of results, and simplify potential deployment in real-world systems. The diversity of the test dataset, with variations in crack morphology, surface material, lighting conditions, and background textures, the fixed threshold was found to generalize well across different scenarios, striking a balance between false positive suppression and crack detection sensitivity.



(a) Original image (b) Ground truth

(c) Threshold value = 0.40 (d) Threshold value = 0.50

(e) Threshold value = 0.60 (f) Threshold value = 0.70

(g) Threshold value = 0.75 (h) Threshold value = 0.80

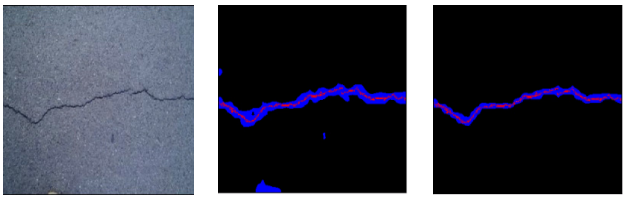
Fig. 8 Qualitative analysis of predictions after thresholding with varying threshold values

Fig. 8 provides a comprehensive qualitative analysis detailing the effect of threshold values on the prediction results. As previously discussed, the implementation of lower thresholds often leads to a higher frequency of false predictions, as evident at the threshold value of 0.40. As the threshold value incrementally increases up to 0.70, there is a noticeable reduction in the instances of false detection within the prediction image. This subsequently enhances the precision of the model crack prediction, contributing to an overall increase in accuracy. However, it is essential to note a critical trade-off when the threshold value transcends 0.70, extending up to 0.80. While it is observed that the false-positive rates decline, but there is a concurrent increase in false-negative pixels. This increase in false negatives adversely impacts the model accuracy, highlighting a complex balance between reducing false positives and avoiding an increase in false negatives. Fig. 9 compares the results of normal predictions to predictions made using a thresholded probability of cracks. Thresholding was found to effectively reduce false detections and enhance the representation of crack thickness. However, in some instances where cracks are thick, the use of a threshold may result in increased false negatives. For example, consider Fig. 9(c), which showcases an image with thick cracks. Due to the wide thickness of these cracks, a predetermined threshold might not accurately capture the full extent of the damage. Consequently, some areas of the cracks that are significant could be ignored or undetected, which increases the number of false negatives. This happens because the threshold is set with an expectation of narrower cracks, and it cannot fully account for situations where the crack thickness exceeds this expectation. Nevertheless, in the Kaggle dataset, images with thin cracks are more prevalent, making a higher threshold value acceptable.

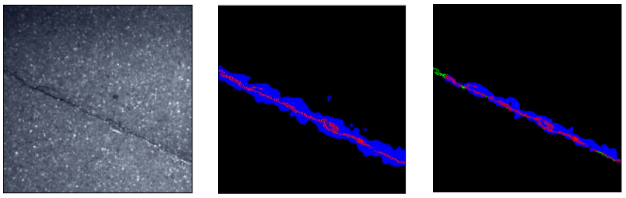
The impact of probability thresholding on the various models discussed in Section 3.3 for loss function parameters optimization was also investigated. Presented in Table 3 are the mIoU values for all experiments conducted before and after applying thresholding for model optimization. The results indicate varying improvements in mIoU across different cases. A consistent threshold value of 0.71 was applied to all cases, and notable enhancements were observed in each instance. In the context of the ViT small backbone, the mIoU showed a marked improvement, increasing from 61.43% to 63.52% after thresholding. This represented a notable gain of approximately 2% in mIoU. Conversely, for the ViT base backbone, the optimized model mIoU experienced an improvement from 63.01% to 66.14%, resulting in an increase of approximately 3% after thresholding. This analysis emphasizes the significant

Table 3 Comparison of the mIoUs before and after probability thresholding

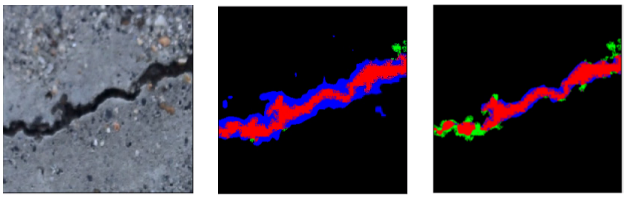
S.No	Backbone	$b(self)$	$b(knn)$	$b(rand)$	mIoU% before thresholding	mIoU% after thresholding
1	ViT Small Parameters 21.9M	0.10	0.18	0.23	56.46	57.57
2		0.20	0.05	0.10	57.15	59.10
3		0.30	0.20	0.10	59.45	59.60
4		0.35	0.25	0.15	59.78	60.70
5		0.30	0.001	0.05	60.13	62.41
6		0.30	0.1	0.05	60.18	61.30
7		**0.30**	**0.01**	**0.05**	**61.43**	**63.52**
8		0.30	0.05	0.01	60.59	62.36
9		0.35	0.03	0.05	60.15	61.67
10		0.40	0.01	0.05	59.38	61.41
1	ViT Base Parameters 86.6M	0.75	0.22	0.20	59.29	59.92
2		0.50	0.15	0.05	57.18	61.23
3		0.50	0.20	0.10	61.13	62.85
4		0.50	0.22	0.15	61.80	63.35
5		0.50	0.22	0.20	61.11	62.15
6		**0.50**	**0.10**	**0.20**	**63.01**	**66.14**
7		0.60	0.10	0.20	60.63	63.31
8		0.65	0.05	0.25	60.10	62.81
9		0.70	0.05	0.15	58.97	61.52
10		0.73	0.10	0.23	59.52	62.13



(a) CFD



(b) GAPS384



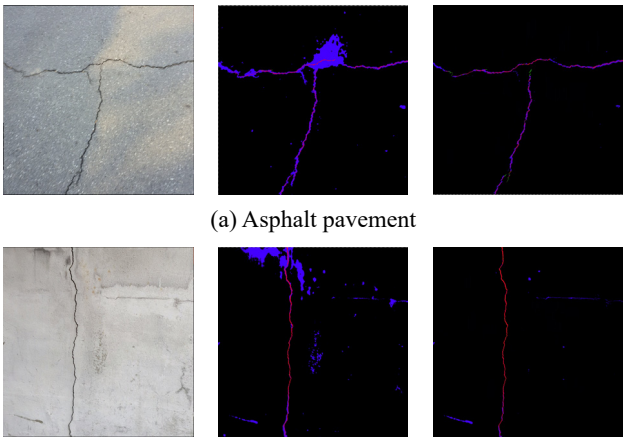
(c) Deep crack

Fig. 9 Comparison of predictions before and after thresholding: (left) original images, (middle) predictions before thresholding, and (right) predictions after thresholding

impact of thresholding probability maps on the effectiveness of models. Understanding the impact of the threshold value on the results and making appropriate adjustments can lead to improvements in the model prediction. This is achieved by minimizing the instances of both false negatives and false positives, thereby refining the overall accuracy of the crack prediction. The proposed method achieved a 66.14% mIoU on a diverse dataset that contained various crack texture and background conditions, surpassing (Aung and Kumwilaisak 2023), which reported mIoU values between 61% and 65% optimized only on GAPS-384 and Crack-500 datasets.

4.3 Sliding window evaluation on high resolution real images

To validate the performance of the proposed unsupervised semantic segmentation model, it was evaluated on high resolution real crack images obtained from both asphalt pavement and concrete building walls. The images were captured by cell phone camera with high resolution and labeled for performance evaluation. With an image resolution of 2048 × 2048, it was not feasible to perform inference on the entire image due to the computational demands involved. To address this challenge, the sliding window technique was employed as discussed in section 2.3 for the evaluation. To provide a thorough comparison of the model's performance, the evaluation was conducted using two techniques: the sliding window technique and direct evaluation by resized image.



(a) Asphalt pavement

(b) Concrete wall

Fig. 10 Examples of predictions on high-resolution images: (left) original images, (middle) predictions without and (right) with applying sliding window

For the evaluation using sliding window technique, a window size of 256 × 256 was selected, and the optimized model with a VIT base backbone was chosen, as described in section 4.2. An important part of the evaluation was setting the optimal threshold for probability mapping, which was optimized at 0.71 for both techniques. This threshold value was determined to provide the best balance between detecting true positives and minimizing false positives. Fig. 10 illustrates the images and corresponding crack segmentation results, which are displayed in the form of a confusion matrix. In the first image, which is asphalt concrete, the background presents different lighting conditions and contrasts. However, the model effectively detects almost all pixels of the cracks using sliding window technique, even in the presence of noise in the background, as evidenced by the low number of false negative pixels. The number of false positive pixels is higher, however, as the cracks are quite thin. In the concrete wall, the cracks are detected with a similar pattern, and only a small number of false positive pixels are detected outside of the crack region in the background noise with a texture similar to the cracks using sliding window technique. In contrast to the sliding window technique, the direct evaluation approach involved resizing the original high-resolution images from 2048px to 880px, a modification necessitated by computational resource constraints. The results of this direct prediction method, as depicted in Fig. 10, exhibit an increased number of false positives in both asphalt and concrete images as compared to the prediction of sliding window technique indicating that it incorrectly identifies certain non-crack regions as cracks. Furthermore, the downsizing process might blur some finer details, leading to some crack features being undetected or wrongly identified, thus contributing to a higher number of false positives. Moreover, as the image resolution is reduced, the pixel-level detail is also decreased, meaning that more distinct features in the original high-resolution images are now averaged and condensed. This can make it more challenging for the model to distinguish between actual cracks and textures or noise that might resemble cracks in lower resolution.

For a more precise evaluation of the proposed method, Table 4 presents the quantitative prediction results for real crack images of asphalt pavement and concrete walls using both techniques. The metrics assessed include mPrecision, mRecall, mF1-score, and mIoU for each image. The trend observed in the precision and recall values for both techniques is noteworthy. In the case of recall, the prediction without the sliding window technique marginally outperforms the sliding window technique. However, the precision values are significantly more favorable when employing the sliding window technique. This discrepancy highlights a notable difference in the rate of false alarms between the two approaches. In the context of asphalt pavement, the prediction without the sliding window technique yields an mF1-score of approximately 65.84% and an mIoU of 55.34% concerning crack detection. Conversely, the sliding window technique exhibits remarkable improvements, with mF1-score of 72.16% and an mIoU of 64.01%. A similar pattern was observed in the concrete wall image, with the sliding window technique achieving a better balance between precision and recall, resulting in mF1-score of 78.10% and an mIoU value of 73.98%. In contrast, the prediction without sliding window technique experiences a decline in accuracy due to an increased number of false alarms, yielding an mF1-score of 69.13% and an mIoU of approximately 60.98%. The results of the sliding window evaluation demonstrate the benefits of this approach. The sliding window approach provides more detailed and granular insights into the performance of the model, as it allows for the evaluation of specific regions of the image, that helps to improve the prediction results. Furthermore, these results demonstrated that the application of sliding window technique for the evaluation of proposed method significantly enhanced detectability, achieving an mIoU up to 73%.

4.4 Comparison with other supervised and unsupervised methods

Table 5 presents a comparative analysis of the proposed

Table 4 Quantitative results of high-resolution images with and without sliding window technique

Structures	Sliding window	mPrecision %	mRecall %	mF1-Score%	mIoU%
Asphalt pavement	X	56.70	**78.52**	65.84	55.34
	O	**68.61**	76.10	**72.16**	**64.01**
Concrete wall	X	62.28	**77.69**	69.13	60.98
	O	**79.10**	77.13	**78.10**	**73.98**

Table 5 Comparison with other supervised and unsupervised methods

Methods	Models	Dataset	mIoU%
Supervised	FCN	Kaggle crack dataset	69.38
	U-Net	Kaggle crack dataset	71.06
	U-Net++	Kaggle crack dataset	73.71
	DeepLabV3+	Kaggle crack dataset	76.60
	SegFormer-B5	Kaggle crack dataset	77.80
Unsupervised	CrackDiffusion	CFD, Crack-500, Cracktree200	40.00-61.00
	Ours	**Kaggle crack dataset**	**66.14**

method against several established supervised and unsupervised crack segmentation models. All supervised models were evaluated on the same composite dataset used in this study, referred to as the Kaggle crack dataset, which is constructed from eight publicly available sources, encompassing multiple crack types, materials, and imaging conditions. The supervised models considered, Fully Convolutional Network (FCN), U-Net, and U-Net++ from earlier convolutional architectures, along with DeepLabV3+ and SegFormer-B5 representing more recent advances, have been previously evaluated in related studies (Cao *et al*. 2023, Li *et al*. 2024), providing benchmark results for comparison in this work. These models were trained using fully annotated dataset. The highest performance among them was achieved by SegFormer-B5, with a mIoU of 77.80%, followed by DeepLabV3+ (76.60%), and U-Net++ (73.71%). In contrast, the proposed method, trained in a fully unsupervised manner without using annotated data, achieved an mIoU of 66.14%. This corresponds to approximately 95% of the performance of FCN and 85% of SegFormer-B5, highlighting the efficiency of the proposed approach despite the absence of manual labeling. For unsupervised methods, CrackDiffusion (Han *et al*. 2024) is the only comparable baseline reported in the literature that has been tested individually on CFD, Crack-500, and Cracktree200, where it achieved mIoU values ranging from 40.00% to 61.00%. In comparison, the proposed method outperformed CrackDiffusion by achieving a higher mIoU on a more diverse, combined dataset, demonstrating improved generalization and segmentation capability under unsupervised settings.

5. Limitations and future work

The proposed unsupervised crack segmentation method demonstrates strong performance across multiple datasets; however, some limitations must be acknowledged. The model produced false positives by misclassifying background elements such as stains, shadows, or construction joints as cracks. This is particularly evident in images with complex textures or noise, where visual similarities can confuse the model. Additionally, the segmentation output often exaggerates crack thickness, especially for fine or intricate patterns, resulting in over-segmented regions compared to the ground truth. Such behavior may reduce the reliability of crack width analysis for structural assessments. Although training utilized diverse datasets covering various crack types, materials, and surface conditions, the model's generalization to unseen environments, such as those with different lighting, textures, or surface materials, remains a challenge. Furthermore, while the sliding window technique enables accurate segmentation of high-resolution images, it introduces computational overhead that limits the model's real-time applicability in practical scenarios. Future work will explore domain adaptation and generalization techniques to improve robustness under varying conditions and improving boundary precision through advanced post-processing techniques. In addition, the development of lightweight architectures will be pursued to enable efficient deployment in real-time or edge-computing environments.

6. Conclusions

This study primarily focused on developing an unsupervised learning approach for detecting cracks on various surfaces, including concrete and asphalt, without relying on labeled data. The proposed unsupervised method provides significant advantages for rapid structural monitoring by eliminating the need for labor-intensive data labeling. This approach provides a practical solution for real-world applications where timely assessments are essential, enabling efficient, scalable, and cost-effective monitoring of civil infrastructure.

The model's generalization and robustness were enhanced through training on a diverse dataset featuring multiple crack types and varying background conditions. Model was trained without access to ground truths and ViT was used as backbone network that was pretrained on ImageNet dataset. Two types of ViT backbone were used including ViT small and ViT base classified based on its training parameters. Parametric study of contrastive loss function was performed on both backbone networks to get the optimal performance of the models. The model was evaluated on 1399 testing images which include different types of cracks based on its geometry and texture. To validate the proposed model performance for field applications, the performance was evaluated on high resolution (2048px × 2048px) real crack images by sliding window technique. Following are the conclusions made based on the evaluation experiments:

- Qualitatively the model detected cracks irrespective of any background noise. However, some missing crack pixels were observed in alligator cracks and thin cracks, Furthermore, model predicted cracks thicker than the ground truths that resulted in high false positives pixels.
- Probability maps was used to improve the crack prediction by omitting the false positives and thresholded as 0.71 by balancing the precision and recall curve.
- Based on loss function parametric optimization, ViT small that contains 29.8M training parameters was optimized to achieve 72.55% of mF1-score, 61.43% of mIoU and thresholded mIoU of 63.52% respectively. However, ViT base that have 86.6M trainable parameters have achieved 75.02%, 63.01% and 66.14% of mF1-score, mIoU and thresholded mIoU respectively.
- The optimized model also predicted cracks pixels of high-resolution crack images using sliding window technique, that showed the generalization of the proposed model and the application for prompt evaluation of damages in the real concrete structure or asphalt roads.

Overall, this study demonstrated the potential of unsupervised learning applications for structural inspection and useful to avoid the data annotation which is time consuming.

Acknowledgments

The research described in this paper was financially supported by the 2022 Research Fund of the University of Seoul.

References

Abdel-Qader, I., Abudayyeh, O. and Kelly, M.E. (2003), "Analysis of edge-detection techniques for crack identification in bridges", *J. Comput. Civil Eng.*, **17**(4), 255-263. https://doi.org/10.1061/(asce)0887-3801(2003)17:4(255)

Al-Huda, Z., Peng, B., Algburi, R.N.A., Alfasly, S. and Li, T. (2022), "Weakly supervised pavement crack semantic segmentation based on multi-scale object localization and incremental annotation refinement", *Appl. Intell.*, **53**(11), 14527-14546. https://doi.org/10.1007/s10489-022-04212-w

Ali, L., Alnajjar, F., Jassmi, H.A., Gocho, M., Khan, W. and Serhani, M.A. (2021), "Performance evaluation of deep CNN-based crack detection and localization techniques for concrete structures", *Sensors*, **21**(5), 1-22. https://doi.org/10.3390/s21051688

Aung, H.T. and Kumwilaisak, W. (2023), "Unsupervised crack segmentation with candidate crack region identification and graph neural network clustering", *Proceedings of the 13th International Conference on Advances in Information Technology*. https://doi.org/10.1145/3628454.3631581

Bang, S., Park, S., Kim, H. and Kim, H. (2019), "Encoder–decoder network for pixel-level road crack detection in black-box images", *Comput. Civil Infrastr. Eng.*, **34**(8), 713-727. https://doi.org/10.1111/mice.12440

Beymer, D. (1996), "Feature correspondence by interleaving shape and texture computations", *Proceedings CVPR IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 921-928.

Caesar, H., Uijlings, J. and Ferrari, V. (2018), "Coco-stuff thing and stuff classes in context", *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1209-1218.

Cao, H., Gao, Y., Cai, W., Xu, Z. and Li, L. (2023), "Segmentation detection method for complex road cracks collected by uav based on HC-Unet++", *Drones*, **7**(3), p. 189. https://doi.org/10.3390/drones7030189

Caron, M., Touvron, H., Misra, I., Jégou, H., Mairal, J., Bojanowski, P. and Joulin, A. (2021), "Emerging properties in self-supervised vision transformers", *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9630-9640. ttps://doi.org/10.1109/ICCV48922.2021.00951

Cha, Y.J., Choi, W. and Büyüköztürk, O. (2017), "Deep learning-based crack damage detection using convolutional neural networks", *Comput. Civil Infrastr. Eng.*, **32**(5), 361-378. https://doi.org/10.1111/mice.12263

Chen, L.C., Zhu, Y., Papandreou, G., Schroff, F. and Adam, H. (2018), "Encoder-decoder with atrous separable convolution for semantic image segmentation", *Proceedings of the European Conference on Computer Vision* (*ECCV*), pp. 833-851.

Cho, J.H., Mall, U., Bala, K. and Hariharan, B. (2021), "Picie: Unsupervised semantic segmentation using invariance and equivariance in clustering", *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16789-16799. https://doi.org/10.1109/CVPR46437.2021.01652

Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S. and Schiele, B. (2016), "The cityscapes dataset for semantic urban scene understanding", *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3213-3223. https://doi.org/10.1109/CVPR.2016.350

Deng, J., Dong, W., Socher, R., Li, L.J., Li, K. and Fei-Fei, L. (2009), "ImageNet: A large-scale hierarchical image database", *2009 IEEE Conf. Comput. Vis. Pattern Recognit.*, pp. 248-255.

Dorafshan, S., Thomas, R.J. and Maguire, M. (2018), "Comparison of deep convolutional neural networks and edge detectors for image-based crack detection in concrete", *Constr. Build. Mater.*, **186**, 1031-1045. https://doi.org/10.1016/j.conbuildmat.2018.08.011

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S. and Uszkoreit, J. (2020), "An image is worth 16x16 words: transformers for image recognition at scale."

Duan, L., Geng, H., Pang, J. and Zeng, J. (2020), "Unsupervised pixel-level crack detection based on generative adversarial network", *Proceedings of the 2020 5th International Conference on Multimedia Systems and Signal Processing*, pp. 6-10.

Dung, C.V. (2019), "Autonomous concrete crack detection using deep fully convolutional neural network", *Autom. Constr.*, **99**, 52-58. https://doi.org/10.1016/j.autcon.2018.11.028

Eisenbach, M., Stricker, R., Seichter, D., Amende, K., Debes, K., Sesselmann, M., Ebersbach, D., Stoeckert, U. and Gross, H.M. (2017), "How to get pavement distress detection ready for deep learning? A systematic approach", In: *2017 International Joint Conference on Neural Networks* (*IJCNN*), pp. 2039-2047.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. and Bengio, Y. (2020), "Generative adversarial networks", *Commun. ACM*, **63**(11), 139-144. https://doi.org/10.1145/3422622

Ham, S., Bae, S., Kim, H., Lee, I., Lee, G.P. and Kim, D. (2021), "Training a semantic segmentation model for cracks in the concrete lining of tunnel", *J. Korean Tunnell. Undergr. Space*

Assoc., **23**(6), 549-558. https://doi.org/https://doi.org/10.9711/KTAJ.2021.23.6.549

Hamilton, M., Zhang, Z., Hariharan, B., Snavely, N. and Freeman, W.T. (2022), "Unsupervised semantic segmentation by distilling feature correspondences", pp. 1-26.

Han, C., Yang, H., Ma, T., Wang, S., Zhao, C. and Yang, Y. (2024), "CrackDiffusion: A two-stage semantic segmentation framework for pavement crack combining unsupervised and supervised processes", *Autom*. *Constr*., **160**, p. 105332. https://doi.org/10.1016/j.autcon.2024.105332

Hinton, G.E. and Salakhutdinov, R.R. (2006), "Reducing the dimensionality of data with neural networks", *Science*, **313**(5786), 504-507. https://doi.org/10.1126/science.1127647

Huang, H., Zhao, S., Zhang, D. and Chen, J. (2022), "Deep learning-based instance segmentation of cracks from shield tunnel lining images", *Struct*. *Infrastruct*. *Eng*., **18**(2), 183-196. https://doi.org/10.1080/15732479.2020.1838559

Jha, S.B. and Babiceanu, R.F. (2023), "Deep CNN-based visual defect detection: Survey of current literature", *Comput*. *Ind*., **148**, p. 103911. https://doi.org/10.1016/j.compind.2023.103911

Ji, X., Henriques, J.F. and Vedaldi, A. (2019), "Invariant information clustering for unsupervised image classification and segmentation", *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9864-9873. https://doi.org/10.1109/ICCV.2019.00996

Kabir, S., Rivard, P., He, D.C. and Thivierge, P. (2009), "Damage assessment for concrete structure using image processing techniques on acoustic borehole imagery", *Constr*. *Build*. *Mater*., **23**(10), 3166-3174. https://doi.org/10.1016/j.conbuildmat.2009.06.013

Ke, T.W., Hwang, J.J., Guo, Y., Wang, X. and Yu, S.X. (2022), "Unsupervised hierarchical semantic segmentation with multiview cosegmentation and clustering transformers", *Proc*. *IEEE Comput*. *Soc*. *Conf*. *Comput*. *Vis*. *Pattern Recognit*., 2561-2571. https://doi.org/10.1109/CVPR52688.2022.00260

Kim, B. and Cho, S. (2018), "Automated vision-based detection of cracks on concrete surfaces using a deep learning technique", *Sensors*, **18**(10). https://doi.org/10.3390/s18103452

Kim, B. and Cho, S. (2019), "Image-based concrete crack assessment using mask and region-based convolutional neural network", *Struct*. *Control Heal*. *Monitor*., p. e2381. https://doi.org/10.1002/stc.2381

König, J., Jenkins, M.D., Mannion, M., Barrie, P. and Morison, G. (2022), "Weakly-supervised surface crack segmentation by generating pseudo-labels using localization with a classifier and thresholding", *IEEE Trans*. *Intell*. *Transp*. *Syst*., **23**(12), 24083-24094. https://doi.org/10.1109/TITS.2022.3204853

Krizhevsky, A., Sutskever, I. and Hinton, G.E. (2017), "ImageNet classification with deep convolutional neural networks", *Commun*. *ACM*, **60**(6), 84-90. https://doi.org/10.1145/3065386

Kuhn, H.W. (1955), "The hungarian method for the assignment problem", *Naval Research Logistics Quarterly*, **2**(1-2), 83-97. https://doi.org/10.1002/nav.3800020109

Kulkarni, S., Singh, S., Balakrishnan, D., Sharma, S., Devunuri, S. and Korlapati, S.C.R. (2023), "CrackSeg9k: A collection and benchmark for crack segmentation datasets and frameworks", In: *European Conference on Computer Vision*, pp. 179-195. https://doi.org/10.1007/978-3-031-25082-8_12

Li, W., Huyan, J., Gao, R., Hao, X., Hu, Y. and Zhang, Y. (2021), "Unsupervised deep learning for road crack classification by fusing convolutional neural network and k_means clustering", *J*. *Transp*. *Eng*. *Part B Pavements*, **147**(4), p. 04021066. https://doi.org/10.1061/JPEODX.0000322

Li, H., Zhang, H., Zhu, H., Gao, K., Liang, H. and Yang, J. (2024), "Automatic crack detection on concrete and asphalt surfaces using semantic segmentation network with hierarchical Transformer", *Eng*. *Struct*., **307**, p. 117903. https://doi.org/10.1016/j.engstruct.2024.117903

Liu, F. and Wang, L. (2022), "UNet-based model for crack detection integrating visual explanations", *Constr*. *Build*. *Mater*., **322**, p. 126265. https://doi.org/10.1016/j.conbuildmat.2021.126265

Liu, Y., Yao, J., Lu, X., Xie, R. and Li, L. (2019), "DeepCrack: A deep hierarchical feature learning architecture for crack segmentation", *Neurocomputing*, **338**, 139-153. https://doi.org/10.1016/j.neucom.2019.01.036

Liu, K., Yan, H., Meng, K., Chen, H. and Sajid, H. (2021), "Iterating tensor voting: A perceptual grouping approach for crack detection on EL images", *IEEE Trans*. *Autom*. *Sci*. *Eng*., **18**(2), 831-839. https://doi.org/10.1109/TASE.2020.2988314

Long, J., Shelhamer, E. and Darrell, T. (2015), "Fully convolutional networks for semantic segmentation", *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3431-3440.

Lv, Z., Tian, J., Zhu, Y. and Li, Y. (2023), "Automatic crack detection of dam concrete structures based on deep learning", *Comput*. *Concrete*, *Int*. *J*., **32**(6), 615-623. https://doi.org/10.12989/cac.2023.32.6.615

Minh, D., Wang, H.X., Li, Y.F. and Nguyen, T.N. (2022), "Explainable artificial intelligence: a comprehensive review", *Artif*. *Intell*. *Rev*., **55**(5), 3503-3568. https://doi.org/10.1007/s10462-021-10088-y

Mubashshira, S., Azam, M.M. and Ahsan, S.M.M. (2020), "An unsupervised approach for road surface crack detection", In: *2020 IEEE Region 10 Symposium*, pp. 1596-1599.

Nick, W., Asamene, K., Bullock, G., Esterline, A. and Sundaresan, M. (2015), "A study of machine learning techniques for detecting and classifying structural damage", *Int*. *J*. *Mach*. *Learn*. *Comput*., **5**(4), 313-318. https://doi.org/10.7763/ijmlc.2015.v5.526

Panella, F., Lipani, A. and Boehm, J. (2022), "Semantic segmentation of cracks: Data challenges and architecture", *Autom*. *Constr*., **135**, p. 104110. https://doi.org/10.1016/j.autcon.2021.104110

Prasanna, P., Dana, K.J., Gucunski, N., Basily, B.B., La, H.M., Lim, R.S. and Parvardeh, H. (2016), "Automated crack detection on concrete bridges", *IEEE Trans*. *Autom*. *Sci*. *Eng*., **13**(2), 591-599. https://doi.org/10.1109/TASE.2014.2354314

Ramli, J., Coulson, J., Martin, J., Nagaratnam, B., Pologanathan, K. and Cheung, W.M. (2021), "Crack detection and localisation in steel-fibre-reinforced self-compacting concrete using triaxial accelerometers", *Sensors*, **21**(6), 1-20. https://doi.org/10.3390/s21062044

Ren, S., He, K., Girshick, R. and Sun, J. (2017), "Faster R-CNN: Towards real-time object detection with region proposal networks", *IEEE Trans*. *Pattern Anal*. *Mach*. *Intell*., **39**(6), 1137-1149. https://doi.org/10.1109/TPAMI.2016.2577031

Ronneberger, O., Fischer, P. and Brox, T. (2015), "U-Net: Convolutional networks for biomedical image segmentation", *Proceedings of Medical image computing and computer-assisted intervention–MICCAI 2015*: *18th International Conference*, October, Munich, Germany, pp. 234-241.

Shamsabadi, E.A., Xu, C. and Dias-da-Costa, D. (2022), "Robust crack detection in masonry structures with Transformers", *Measur*., **200**, p. 111590. https://doi.org/10.1016/j.measurement.2022.111590

Shen, Y., Yu, Z., Li, C., Zhao, C. and Sun, Z. (2023), "Automated detection for concrete surface cracks based on Deeplabv3+ BDF", *Buildings*, **13**(1), p. 118. https://doi.org/10.3390/buildings13010118

Shi, Y., Cui, L., Qi, Z., Meng, F. and Chen, Z. (2016), "Automatic road crack detection using random structured forests", *IEEE Trans*. *Intell*. *Transp*. *Syst*., **17**(12), 3434-3445. https://doi.org/10.1109/TITS.2016.2552248

Shi, P., Zhu, F., Xin, Y. and Shao, S. (2022), "U^2 CrackNet: A deeper architecture with two-level nested U-structure for pavement crack detection", *Struct. Healtth. Monitor.*, **22**(4), 2910-2921. https://doi.org/10.1177/14759217221140976

Solhmirzaei, R., Salehi, H. and Kodur, V. (2024), "A computer vision-based approach for crack detection in ultra high performance concrete beams", *Comput. Concrete*, *Int. J.*, **33**(4), 341-348. https://doi.org/10.12989/cac.2024.33.4.341

Song, C. (2024), "Research on damage detection and assessment of civil engineering structures based on DeepLabV3+ deep learning model", *Struct. Eng. Mech.*, *Int. J.*, **91**(5), 443-457. https://doi.org/10.12989/sem.2024.91.5.443

Song, W., Zhang, Z., Zhang, B., Jia, G., Zhu, H. and Zhang, J. (2023), "ISTD-PDS7: A benchmark dataset for multi-type pavement distress segmentation from ccd images in complex scenarios", *Remote Sens.*, **15**(7), p. 1750. https://doi.org/10.3390/rs15071750

Tang, W., Wu, R.T. and Jahanshahi, M.R. (2022), "Crack segmentation in high-resolution images using cascaded deep convolutional neural networks and Bayesian data fusion", *Smart Struct. Syst.*, *Int. J.*, **29**(1), 221-235. https://doi.org/10.12989/sss.2022.29.1.221

Van der Maaten, L. and Hinton, G. (2008), "Visualizing data using t-SNE", *J. Mach. Learn. Res.*, **9**, 2579-2625.

Wang, S. and Nishio, M. (2024), "Review for vision-based structural damage evaluation in disasters focusing on nonlinearity", *Smart Struct. Syst.*, *Int. J.*, **33**(4), 263-279. https://doi.org/10.12989/sss.2024.33.4.263

Wang, W. and Su, C. (2022), "Automatic concrete crack segmentation model based on transformer", *Autom. Constr.*, **139**, p. 104275. https://doi.org/10.1016/j.autcon.2022.104275

Wang, G. and Xiang, J. (2021), "Railway sleeper crack recognition based on edge detection and CNN", *Smart Struct. Syst.*, *Int. J.*, **28**(6), 779-789. https://doi.org/10.12989/sss.2021.28.6.779

Wang, Y., Xiong, W., Cheng, J., Chia, S.C., Chen, W., Huang, W. and Zhou, J. (2015), "Vision based hole crack detection", In: *2015 IEEE 10th Conference on Industrial Electronics and Applications* (*ICIEA*), pp. 1932-1936. https://doi.org/10.1109/ICIEA.2015.7334428

Wang, L., Kawaguchi, K.I. and Wang, P. (2020), "Damaged ceiling detection and localization in large-span structures using convolutional neural networks", *Autom. Constr.*, **116**, p. 103230. https://doi.org/10.1016/j.autcon.2020.103230

Wang, Y., Song, K., Niu, M., Bao, Y., Dong, H. and Yan, Y. (2022), "Unsupervised defect detection with patch-aware mutual reasoning network in image data", *Autom. Constr.*, **142**, p. 104472. https://doi.org/10.1016/j.autcon.2022.104472

Wu, T., Zhang, H., Liu, J., Chen, Y., Yi, J. and Zhang, Z. (2021), "Memory-augment convolutional Autoencoder for unsupervised pavement crack classification", In: *2021 China Automation Congress*, pp. 2952-2956.

Xiang, X., Wang, Z. and Qiao, Y. (2022), "An improved YOLOv5 crack detection method combined with transformer", *IEEE Sens. J.*, **22**(14), 14328-14335. https://doi.org/10.1109/JSEN.2022.3181003

Yang, X., Li, H., Yu, Y., Luo, X., Huang, T. and Yang, X. (2018a), "Automatic pixel-level crack detection and measurement using fully convolutional network", *Comput. Civil Infrastruct. Eng.*, **33**(12), 1090-1109. https://doi.org/10.1111/mice.12412

Yang, X., Li, H., Yu, Y., Luo, X., Huang, T. and Yang, X. (2018b), "Automatic pixel-level crack detection and measurement using fully convolutional network", *Comput. Civil Infrastruct. Eng.*, **33**(12), 1090-1109. https://doi.org/10.1111/mice.12412.

Yang, F., Zhang, L., Yu, S., Prokhorov, D., Mei, X. and Ling, H. (2020), "Feature pyramid and hierarchical boosting network for pavement crack detection", *IEEE Trans. Intell. Transp. Syst.*, **21**(4), 1525-1535. https://doi.org/10.1109/TITS.2019.2910595

Yin, Z., Wang, P., Wang, F., Xu, X., Zhang, H., Li, H. and Jin, R. (2022), "TransFGU: A top-down approach to fine-grained unsupervised semantic segmentation", In: *European Conference on Computer Vision*, pp. 73-89.

Yu, J., Kim, D.Y., Lee, Y. and Jeon, M. (2020), "Unsupervised pixel-level road defect detection via adversarial image-to-frequency transform", In: *2020 IEEE Intelligent Vehicles Symposium* (*IV*), pp. 1708-1713. https://doi.org/10.1109/IV47402.2020.9304843

Yusof, N.A.M., Osman, M.K., Hussain, Z., Noor, M.H.M., Ibrahim, A., Tahir, N.M. and Abidin, N.Z. (2019), "Automated asphalt pavement crack detection and classification using deep convolution neural network", In: *2019 9th IEEE International Conference on Control System*, *Computing and Engineering* (*ICCSCE*, pp. 215-220. https://doi.org/10.1109/ICCSCE47578.2019.9068551

Zadaianchuk, A., Kleindessner, M., Zhu, Y., Locatello, F. and Brox, T. (2022), "Unsupervised semantic segmentation with self-supervised object-centric representations", pp. 1-23.

Zhang, J., Lu, C., Wang, J., Wang, L. and Yue, X.G. (2019), "Concrete cracks detection based on FCN with dilated convolution", *Appl. Sci.*, **9**(13), p. 2686. https://doi.org/10.3390/app9132686

Zhang, K., Zhang, Y. and Cheng, H.D. (2020), "Self-supervised structure learning for crack detection based on cycle-consistent generative adversarial networks", *J. Comput. Civil Eng.*, **34**(3). https://doi.org/10.1061/(ASCE)CP.1943-5487.0000883

Zhao, F., Chao, Y., Liu, X. and Li, L. (2022), "A novel crack segmentation method based on morphological-processing network", In: *2022 15th International Congress on Image and Signal Processing*, *BioMedical Engineering and Informatics* (*CISP-BMEI*), pp. 1-6.

Zhou, S. and Song, W. (2020), "Concrete roadway crack segmentation using encoder-decoder networks with range images", *Autom. Constr.*, **120**,p. 103403. https://doi.org/10.1016/j.autcon.2020.103403

Zhou, S., Canchila, C. and Song, W. (2023), "Deep learning-based crack segmentation for civil infrastructure: data types, architectures, and benchmarked performance", *Autom. Constr.*, **146**, p. 104678. https://doi.org/10.1016/j.autcon.2022.104678

Zou, Q., Cao, Y., Li, Q., Mao, Q. and Wang, S. (2012), "CrackTree: Automatic crack detection from pavement images", *Pattern Recognit. Lett.*, **33**(3), 227-238. https://doi.org/10.1016/j.patrec.2011.11.004

Zoubir, H., Rguig, M., El Aroussi, M., Chehri, A., Saadane, R. and Jeon, G. (2022), "Concrete bridge defects identification and localization based on classification deep convolutional neural networks and transfer learning", *Remote Sens.*, **14**(19), p. 4882. https://doi.org/10.3390/rs14194882