Geometry-Aware Guided Loss for Deep Crack Recognition

Zhuangzhuang Chen Jin Zhang Zhuonan Lai Jie Chen Zun Liu Jianqiang Li [*]
Shenzhen University, Shenzhen, China
chenzhuangzhuang2016@email.szu.edu.cn, {jin.zhang, chenjie, zunliu, lijq}@szu.edu.cn

Abstract

Despite the substantial progress of deep models for crack recognition, due to the inconsistent cracks in varying sizes, shapes, and noisy background textures, there still lacks the discriminative power of the deeply learned features when supervised by the cross-entropy loss. In this paper, we propose the geometry-aware guided loss (GAGL) that enhances the discrimination ability and is only applied in the training stage without extra computation and memory during inference. The GAGL consists of the feature-based geometry-aware projected gradient descent method (FGA-PGD) that approximates the geometric distances of the features to the class boundaries, and the geometry-aware update rule that learns an anchor of each class as the approximation of the feature expected to have the largest geometric distance to the corresponding class boundary. Then the discriminative power can be enhanced by minimizing the distances between the features and their corresponding class anchors in the feature space. To address the limited availability of related benchmarks, we collect a fully annotated dataset, namely, NPP2021, which involves inconsistent cracks and noisy backgrounds in real-world nuclear power plants. Our proposed GAGL outperforms the state of the arts on various benchmark datasets including CRACK2019, SDNET2018, and our NPP2021.

1. Introduction

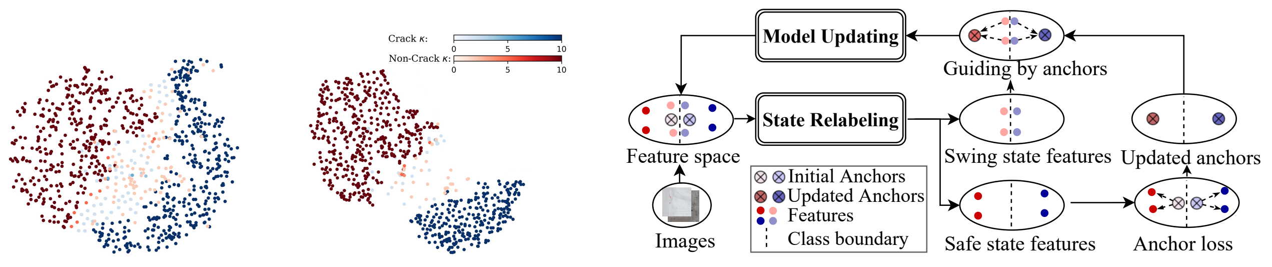
Concrete structure health monitoring is of crucial importance in various industries [2, 5, 7, 38, 39], of which crack damage classification is the first and critical stage. There is a great need for automated testing methods to recognize and repair cracks before the onset of serious deterioration so that the heavy maintenance costs could be reduced. However, it is extremely challenging to realize crack recognition in a real-world industrial environment. The cracks are commonly presented as dark curves on walls in inconsistent sizes and irregular shapes [13], moreover, mixed with varying surrounding backgrounds, resulting in different signal-to-noise ratios [44] [2].

Recently, the application of deep learning technology on crack recognition tasks has achieved great success [4, 14, 39]. These methods [14, 43] commonly train the convolution neural network (CNN) on sub-images, then apply the trained model on the high-resolution images with a sliding window [6] to scan and classify the existence of cracks in each sub-image. However, the traditional CNN-based methods for crack recognition have a crucial limitation: *the inability of extracting discriminative features of cracks in a realistic environment*. That is, due to the inconsistent cracks in varying sizes and shapes [43], the noisy texture patterns of the surrounding background, and the limited discriminative ability of the cross-entropy (CE) loss [37], it is challenging for deep models to extract the discriminate features that can be successfully classified. To verify this, Fig. 1(a) visualizes the crack and non-crack images in the feature space. It demonstrates that each class of images projected in the feature space spread in a wide area, and crack/non-crack features are mixed together and close to the class boundary. Therefore, the existing approaches are inappropriate for crack recognition in a realistic environment.

In this paper we propose a novel geometry-aware guided loss (GAGL) to enhance the discriminative power of the deeply learned features with the joint supervision of CE loss. GAGL enlarges the inter-class features separation and reduces the intra-class features variation by pulling the deep features of the same class to the feature that has the largest geometric distance to the class boundary. There are several challenges to realize such a loss as discussed in the below.

Due to the irregularity of the class boundaries [24], it is hard to detect the class boundaries and estimate the geometric distances of features to them. To solve this issue, we propose a feature-based geometry-aware projected gradient descent method (FGA-PGD) to approximate the geometric distances of the features to class boundaries. Intuitively, features which are close to the decision boundaries are vunerable to attacks. As shown in Fig. 1, to find the misclassified adversarial variants of input features, the features near the boundary require less rounds (i.e., lighter colors) of

[*]Corresponding Author



(a) w/o GAGL (b) w/ GAGL (c) The illustration of GAGL

Figure 1. T-SNE feature visualizations of the crack and non-crack classes from the CRACK2019 dataset. The color gradient of the feature is calculated based on the least attack number κ that PGD needs to find its misclassified adversarial variant. The features that need a large κ (darker red and blue) are farther away from the decision boundary. Meanwhile, the features that need a small κ (lighter red and blue) are closer to the decision boundary. (a) CE supervision, (b) CE and GAGL supervision, (c) The illustration of GAGL. In each iteration, GAGL first relabels the features with a larger κ than their corresponding class anchors as the safe state features. Otherwise, the features are relabeled as the swing state features. Then, GAGL iteratively updates the anchor of each class as the alternative for the safest state feature by exploiting the anchor loss. Meanwhile, the object function of GAGL is defined to enhance the discriminative power of the deeply learned features by making the swing state features close to their corresponding class anchors. (Best viewed in color)

adversarial attacks than those faraway. In contrast, the features far away from class boundaries require a larger number of attacks. By exploiting this effect, FGA-PGD generates adversarial samples based on feature embeddings instead of images such that the geometric distances of crack and non-crack features could be approximated in an efficient manner (see Sec. 3.2). Herein, FGA-PGD exploits the κ-step Projected Gradient Descent (PGD-κ) method [21,22] as the adversarial attack method.

Since the entire training set is very large and contains tens of thousands of training samples, it is inefficient and even impractical to select the feature that has the largest geometric distance to the class boundary in each iteration. To address this problem, we exploit the aforementioned FGA-PGD method to measure the geometric distance and develop a geometry-aware update rule to learn an anchor (a vector with the same dimension as the feature) of each class as the approximation of the feature expected to have the largest geometric distance to the corresponding class boundary (see Sec. 3.3).

Then, the objective function of GAGL can be defined by optimizing the distances between the anchors and the features that are closer to the class boundaries than the corresponding class anchors.(see Sec. 3.4). The optimization process of GAGL can be viewed as guiding the model to pull feature embeddings away from class boundaries and gather intra-class features together, which effectively help to separate crack and non-crack images in the feature space.

Hence, the goal of the GAGL can be realized by simultaneously updating the anchors by the geometry-aware update rule and iteratively optimizing the features by the objective function of GAGL. The main contributions of this paper can be summarized as:

(1) Propose the feature-based geometry-aware projected gradient descent (FGA-PGD) method to approximate the geometric distances of features to the class boundaries. Moreover, FGA-PGD is highly efficient as the adversarial attacks performed on the features instead of the images.

(2) Propose the geometry-aware guided loss (GAGL) to enhance the discriminative power of the deeply learned features by penalizing the distances between the features and their corresponding class anchors in the feature space, where the anchor of each class is learned by the proposed geometry-aware update rule as the approximation of the feature expected to have the largest geometric distance to the corresponding class boundary.

(3) A labeled dataset has been constructed, namely, NPP2021, which contains accurate annotations, to facilitate the research about crack recognization tasks for the noisy backgrounds and the inconsistent cracks in varying size problems.

2. Related work

In this section, we briefly review existing researches on related topics.

2.1. Crack recognition

Following the research [11], the image-based crack recognition via classification approaches in the literature can be clustered into two groups: traditional approaches and deep learning approaches. The methods of the first group usually contain two stages [19]. At the first stage, HOG [18] and LBP [28, 34] are used to extract the image features by a local descriptor. Then, at the second stage, a well-trained classifier is introduced to recognize potential crack patches. The classifier is usually chosen from the follows: Gaussian process [25], SVM [18, 28, 34], and Neural Network [40].

Most of the deep learning approaches re-train a deep CNN model for crack image classification [11]. Variants of CNN are applied on classifying crack images on sub-images of 120×120 pixels [8], 200×200 pixels [42], and 256×256 pixels [6, 10]. Ali et al [3] modify the architecture of the VGG16 model and use the CE loss to train this model. Unlike the previous works, we develop a novel loss function to enhance the discriminative power of the deeply learned features for the crack classification tasks, instead of designing a customized model.

2.2. Adversarial attacks

The general setting of the adversarial attacks is two-fold: (1) black-box setting, the attacker can only get the substitute model, (2) white-box setting, the attacker must have full access to the attacked models. Compared with the methods in the black-box setting, gradient-based attack methods under the white-box setting are more practical and powerful on real tasks [45]. Specifically, for the gradient-based attack methods, FGSM [12] generates adversarial examples to fool the model by directly increasing the loss of the model. Then, κ-step projected gradient descent method [21, 22] (PGD-κ) performs the iterative attack, which can be viewed as an iterative version of FGSM. Typically, the stronger adversarial examples [22] can be generated by taking more attack iterations (larger κ). However, each attack iteration needs to compute the gradient on the input image by the back-propagation, which causes a large computational overhead. In this paper, we investigate the relationship between the least number of iterations required for features to be successfully attacked by the PGD-κ method and the geometric distances of the features to the class boundaries. Compared with the image-based attacks, our feature-based attacks can achieve higher efficiency, e.g., for the ResNet50 [16] model, feature-based attacks take only 1ms for one iteration, while image-based attacks take 82ms.

2.3. Robust loss function

Recently, several robust loss functions are proposed to extract discriminative features. For example, in the work [32], CE loss and the contrastive loss are combined to learn more discriminative features. Focal loss [20] is designed to prevent a large number of easy samples from dominating the training procedure by assigning a large weight to the hard examples. Furthermore, a cosine margin term is introduced to further maximize the decision margin in the angular space [36]. To increase interpretability, the geometric interpretation has been added on a hypersphere [9] to boost the performance of the face recognition tasks. From a different perspective, center loss [37] simultaneously learns a center for deep features of each class and penalizes the distances between the samples and their corresponding class centers in the feature space. However, center loss suffers from the sacrifice of inter-class separability, that is, some of the features, which are farther away from the class boundary than the corresponding class centers, will be pulled close to the class boundary when supervised by center loss.

Our proposed GAGL shares the same purpose to extract discriminative features with the previous loss functions. However, ours is distinct from these loss functions in two-fold: (1) We approximate the geometric distance of feature to the class boundary by the feature-based adversarial attacks, (2) Unlike the center loss simply learning the centers of features, GAGL learns the anchor of each class as the approximation of the feature expected to have the largest geometric distance to the corresponding class boundary.

3. Geometry-aware guided loss

In this section, we introduce the proposed GAGL. We start with the motivation of the GAGL in Sec. 3.1, and provide the descriptions of the feature-based geometry-aware projected gradient descent method (FGA-PGD) in Sec. 3.2, the geometry-aware update rule in Sec. 3.3, and the learning objective of GAGL in Sec. 3.4.

3.1. Motivation of GAGL

GAGL is based on an intuitive idea: the geometric distance of the feature from the class boundary is closely related to the least number of iterations required for this feature to be successfully attacked.

To verify this intuition, we conduct an experiment on the CRACK2019 dataset. Specifically, we first train a ResNet50 [16] model on CRACK2019 dataset, and extract the features of the images from this dataset by ResNet50, then we perform the PGD attacks on these features to get the least number of iterations κ. After that, we use t-SNE [33] to visualize the features with the κ denoted by the color gradient in Fig. 1(a). It can be observed that features involving small/large κ are more likely close to/far from class boundaries. Thus, the geometric distances between features and class boundaries can be approximated by the least number of attacks.

In Fig. 1(a), analogy to U.S. elections [1], the blue and red states are used to denote the features with the crack and non-crack label, respectively. Then, we use the swing state features to refer to the features that their prediction labels are easily changed by adversarial attacks with a small κ, these features are close to the class boundaries. Otherwise, if the features are hard to be attacked, then these features are referred to the safe state features.

Our GAGL aims to simultaneously learn the anchor of each class (see Sec. 3.3) as the alternative for the safest state feature in the same class and enhance the discriminative power of the deeply learned features by making the swing state features close to their corresponding class anchors (see

[1]https://edition.cnn.com/election

Sec. 3.4). We train the ResNet50 with the supervision of CE and GAGL, and show the features learned by the model in Fig. 1(b). We can observe that there are less swing state features in Fig. 1(b) compared with Fig. 1(a). This further verifies that the geometric distance of the features can be approximated by the least number of the attacks, and some of the swing state features can be pulled away from the class boundaries by GAGL.

3.2. Realization of FGA-PGD

As prior discussed, the geometric distance of the feature to the class boundary can be approximated as the least number of iterations required to generate its misclassified adversarial variant by the adversarial attack. Herein, we propose an efficient feature-based geometry-aware projected gradient descent method (FGA-PGD), that returns the geometric value of the feature to the class boundary. Compared to the existing method PGD-κ, FGA-PGD is more efficient as it directly attacks the features, and does not require back-propagation of the feature extractor $\mathbb{M}^e$. The corresponding pseudocode for feature geometric distance approximation is presented in Algorithm 1.

The adversarial attacks in FGA-PGD can be interpreted as a multi-step scheme for maximizing the CE loss function ℓ under the feature $\mathcal{F}_x$ of the image x with the ground truth label y. $\hat{y}$ denotes the prediction label of x. $\mathcal{F}_x$ can be obtained by $\mathbb{M}^e$ as follows:

$$\mathcal{F}_x = \mathbb{M}^e(x). \tag{1}$$

$\hat{y}$ can be obtained by the classifier $\mathbb{M}^c$ with respect to $\mathcal{F}_x$ as follows :

$$\hat{y} = \underset{i}{argmax}\, \mathbb{M}^c(\mathcal{F}_x). \tag{2}$$

Then, the $(t+1)^{th}$ adversarial variant feature $\mathcal{F}_x^{(t+1)}$ is updated along the gradient $\nabla_{\mathcal{F}_x^{(t)}}\ell(\cdot)$ of ℓ with respect to $\mathcal{F}_x^{(t)}$. As a consequence, $\mathcal{F}_x^{(t)}$ can be obtained by the follows:

$$\mathcal{F}_x^{(t+1)} = \mathcal{F}_x^{(t)} + \Pi_{\mathcal{B}[\mathcal{F}_x^{(t)},\epsilon]}\left(\gamma\,\mathrm{sign}\left(\nabla_{\mathcal{F}_x^{(t)}}\ell\left(\mathbb{M}_\theta^c\left(\mathcal{F}_x^{(t)}\right),y\right)\right)\right), \tag{3}$$

where 0^{th} adversarial variant feature $\mathcal{F}_x^{(0)}$ is initialized by $\mathcal{F}_x$. The step size γ is used to control the magnitude of adversarial variant features change, and the project function $\Pi_{\mathcal{B}[\mathcal{F}_x^{(t)},\epsilon]}$ is used to project $\mathcal{F}_x^{(t)}$ back into the center of $\mathcal{F}_x^{(t)}$, where the metric ϵ-ball controls its perturbation bound.

After that, the least number of adversarial attacks needed to find misclassified features can be obtained by starting from $\mathcal{F}_x^{(0)}$ until t^{th} adversarial variants can fool the current network. In Algorithm 1, the certain stopping criterion is that the prediction label $\hat{y}$ of the $\mathcal{F}_x^{(t)}$ is unequal to the ground truth label y, or t reaches the maximum PGD attack number K. Finally, the geometric distance $\kappa(\mathcal{F}_x, y)$ of $\mathcal{F}_x$ with the label y can be approximated by the t.

Algorithm 1 Feature-based geometry-aware projected gradient descent method (FGA-PGD)

Input: image x, label y, feature extractor $\mathbb{M}^e$, classifier $\mathbb{M}^c$, CE loss ℓ, maximum attack number K, step size γ and perturbation bound ϵ

```
1:  Extract the feature $\mathcal{F}_x$ of $x$ : $\mathcal{F}_x = \mathbb{M}^e(x)$
2:  Adversarial feature initialize: $\mathcal{F}_x^{(0)} \leftarrow \mathcal{F}_x$
3:  Adversarial attack number initialize : $t \leftarrow 0$
4:  while K >0 do
5:    if $\arg\max_i \mathbb{M}^c(\mathcal{F}_x^{(t)}) = y$ then
6:      $t \leftarrow t+1$
7:    else
8:      $\kappa(\mathcal{F}_x, y) \leftarrow t$, $K = 0$
9:    end if
10:   $\nabla = \Pi_{\mathcal{B}[\mathcal{F}_x^{(t)},\epsilon]}\left(\gamma\,\mathrm{sign}\left(\nabla_{\mathcal{F}_x^{(t)}}\ell\left(\mathbb{M}_\theta^c\left(\mathcal{F}_x^{(t)}\right),y\right)\right)\right)$
11:   $F_x^{(t+1)} = \mathcal{F}_x^{(t)} + \nabla$
12:   $K \leftarrow K-1$
13: end while
    Output: Feature geometric distance $k(\mathcal{F}_x, y)$
```

3.3. Geometry-aware update rule

Taking the entire training set into account and selecting the safest state feature of each class that has the largest geometry distance to the corresponding class boundary in each iteration are inefficient even impractical. We develop the geometry-aware update rule to address this problem to update the anchor of each class as the alternative for the safest state feature in the same class. The intuition behind this mini-batch-based rule is two-fold: (1) In each mini-batch, assuming that some features have larger geometric distances than their corresponding class anchors, then the anchors can be learned to increase their geometric distances by minimizing their distances from these features with the same class. The learned anchors will not be updated in the case that no other features have a larger geometric distance than the current anchor. (2) The features with a large geometric distance should be given a large importance weight so that the geometric distances of the learned anchors can be significantly increased by making the anchors close to these features.

Base on the above intuition, we decompose the geometry aware update rule into three steps: (1) Feature state relabeling, in each mini-batch, we relabel the features that have a larger geometric distance than their corresponding class anchors as the safe state features, otherwise as the swing state features. (2) Dynamic re-weighting function, we use this function to assign the different weights to the safe state features considering their importance to the anchors. (3) Anchor update rule, we develop this rule to update the anchors. Specifically, we first design the anchor loss that minimizes the re-weighting distances between the anchors and the safe state features by considering their importance, then the update rule can be formulated by this loss.

Feature state re-labeling. We provide an in-detail description of how to re-label the feature state. Specifically, as we have mentioned before, the feature geometric distance $\kappa(\mathcal{F}_x, y)$ of the feature $\mathcal{F}_x$ with the label y can be approximated by the least number of attacks. Meanwhile, the geometric distance $\kappa(\boldsymbol{A_y}, y)$ of the anchor $\boldsymbol{A}_y$ for the class y can also be approximated by Algorithm 1. Then, we use the label $\mathcal{S}$ to denote the safe state features that have larger geometric distances than their corresponding class anchors. Otherwise, when the geometric distances of features are smaller than or equal to the anchors, these features are referred to the swing state features labeled as $\mathcal{W}$. Our feature state re-labeling function is shown as follows:

$$s(\mathcal{F}_x, y) = \left\{ \begin{array}{ll} \mathcal{S}, & \text{case } \kappa(\mathcal{F}_x, y) > \kappa(\boldsymbol{A_y}, y) \\ \mathcal{W}, & \text{case } \kappa(\mathcal{F}_x, y) \leq \kappa(\boldsymbol{A_y}, y) \end{array} \right. , \quad (4)$$

where $s(\mathcal{F}_x, y)$ denotes the state of $\mathcal{F}_x$ with respect to its geometric distance.

Dynamic re-weighting function. Since the safe state features have unequal importance for the anchors, the dynamic re-weighting function with respect to $\kappa(\mathcal{F}_x, y)$ is designed to pay more attention to the feature that has a larger geometric distance by assigning a larger weight. This function is heuristically defined as the non-decreasing function shown as follows:

$$\omega(\mathcal{F}_x, y) = \frac{(1{+}\tanh(\beta + 5 \times (2 \times \kappa(\mathcal{F}_x, y)/K - 1)))}{2}, \quad (5)$$

where the β is used to control the magnitude of weight changes, if $\beta = +\infty$, all the safe state features will be assigned to the same weight for the anchor update. Here we heuristically give one example, and more examples are discussed in Sec. 4.2.

Anchor update rule. Consider the $\mathcal{T}^{th}$ training iteration on the mini-batch $S_m = \{(x_i, y_i)\}_{i=1}^{m}$, where x_i denotes the i^{th} image sample with class y_i, we use $\boldsymbol{A}_j^{\mathcal{T}}$ to denote the anchor with the class $j, j \in \{1, ..., C\}$ at iteration $\mathcal{T}$, where C denotes the number of class. Since $\boldsymbol{A}_j^{\mathcal{T}}$ is updated by making it close to the safe state features with unequal weights, the anchor loss $\ell_{\boldsymbol{A}_j^{\mathcal{T}}}$ with respect to the $\boldsymbol{A}_j^{\mathcal{T}}$ can be defined by the follows:

$$\ell_{\boldsymbol{A}_j^{\mathcal{T}}} {=} \frac{1}{2} \sum_{i=1}^{m} \delta(y_i{=}j) \delta\left(s\left(\mathcal{F}_{x_i}, y_i\right){=}\mathcal{S}\right) \omega\left(\mathcal{F}_{x_i}, y_i\right) \left(\mathcal{F}_{x_i} {-} \boldsymbol{A}_j^{\mathcal{T}}\right)^2, \quad (6)$$

where $\delta(condition){=}1$ if the condition is satisfied, otherwise, $\delta(condition){=}0$. $\mathcal{F}_{x_i}$ denotes the extracted feature of x_i. Then, we use $\nabla_{\boldsymbol{A}_j^{\mathcal{T}}}$ to denote the gradient of $\ell_{\boldsymbol{A}_j^{\mathcal{T}}}$ with respect to $\boldsymbol{A}_j^{\mathcal{T}}$, and the formula is as follows:

$$\nabla_{\boldsymbol{A}_j^{\mathcal{T}}} = \sum_{i=1}^{m} \delta(y_i{=}j) \delta(s(\mathcal{F}_{x_i}, y_i){=}\mathcal{S}) \omega(\mathcal{F}_{x_i}, y_i) \left(\boldsymbol{A}_j^{\mathcal{T}} {-} \mathcal{F}_{x_i} \right). \quad (7)$$

Finally, $\boldsymbol{A}_j^{\mathcal{T}}$ can be updated along the negative gradient $\nabla_{\boldsymbol{A}_{y_i}^{\mathcal{T}}}$ to make the anchor close to the safe state features. The corresponding formula is as follows:

$$\boldsymbol{A}_j^{\mathcal{T}+1} = \boldsymbol{A}_j^{\mathcal{T}} - \alpha \nabla_{\boldsymbol{A}_j^{\mathcal{T}}}, \quad (8)$$

where $\boldsymbol{A}_j^{\mathcal{T}+1}$ represents the anchor with the class j at the $(\mathcal{T}+1)^{th}$ training iteration. The α is used to control the magnitude of $\boldsymbol{A}_j^{\mathcal{T}}$ update, and $\boldsymbol{A}_j^{0}$ is randomly initialized.

3.4. Learning objective of GAGL

With the prior rule, we can learn the anchor of each class as the alternative for the safest state feature in the same class. However, those swing state features are still more likely close to the boundary and mixed together in the feature space. To address this problem, the learning objective of GAGL is defined by minimizing the distances between the swing state features and their corresponding class anchors. In addition, the swing state features, which have smaller geometric distances to the class boundary, are more easily misclassified. Hence, we pay more attention to these features by assigning a larger weight to them. Unlike the non-decreasing dynamic re-weighting function $\omega(\mathcal{F}_{x_i}, y_i)$ in Sec. 3.3, we use the non-increasing dynamic re-weighting function $1 - \omega(\mathcal{F}_{x_i}, y_i)$ to assign different weights on the swing state features with respect to their geometric distances. Then, the objective function of GAGL is defined as follows:

$$\ell_G = \sum_{i=1}^{m} \frac{1}{2} \delta(s(\mathcal{F}_{x_i}, y_i){=}\mathcal{W})(1 - \omega(\mathcal{F}_{x_i}, y_i)) \left(\mathcal{F}_{x_i} - \boldsymbol{A}_{y_i}\right)^2. \quad (9)$$

As the FGA-PGD is performed by CE loss, hence, we adopt the CE loss and GAGL as the joint supervision loss ℓ_{js} to train the $\mathbb{M}^e$ and $\mathbb{M}^c$ for discriminative feature learning. The formulation is given as follows:

$$\begin{aligned} \ell_{js} &= \ell_{CE} + \lambda \ell_G \\ &= -\sum_{i=1}^{m} y_i \cdot \log\left(\text{softmax}\left(\mathbb{M}^c\left(\mathbb{M}^e(x_i)\right)\right)\right) + \lambda \ell_G, \end{aligned} \quad (10)$$

where the scalar λ is used to balance the two loss functions.

By exploiting the rule in Sec. 3.3 and the objective of GAGL, we are able to simultaneously learn the anchors and the discriminative features. The corresponding pseudocode for the model under the joint supervision loss of CE and GAGL is presented in Algorithm 2.

4. Experiment

The essential experimental setup is described in Sec. 4.1. Then, we discuss different re-weighting functions ω in Sec. 4.2. To find the most suitable parameters α, β, and λ in GAGL, we carry out some experiments on different parameters as shown in Sec. 4.3. Furthermore, to verify the effectiveness of our proposed method, we not only compared

Algorithm 2 Geometry-aware feature learning algorithm

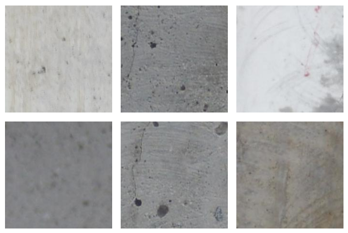
```
Input: Training dataset $S = \{(x_i, y_i)\}_{i=1}^n$, feature extractor $\mathbb{M}^e$, classifier $\mathbb{M}^c$, hyperparameters $\alpha$, $\beta$, and $\lambda$, the iteration number $\mathcal{T}$ initialized to 0, mini-batch size $m$ and the random initial anchors $\boldsymbol{A}_j^0, j \in \{1, ..., C\}$, where $C$ denotes the number of class
1: while not converge do
2:     $\mathcal{T} \leftarrow \mathcal{T} + 1$
3:     For each mini-batch $S_m = \{(x_i, y_i)\}_{i=1}^m$ of the training set $S$, $\mathcal{F}_{x_i} = \mathbb{M}^e(x_i), i \in \{1, ..., m\}$
4:     Calculate the geometric distances of the features and the anchors by Algorithm 1
5:     Calculate the anchor loss by Eq. 4 to Eq. 6
6:     Update the anchor $\boldsymbol{A}_j^{\mathcal{T}}$ for each class $j$, $j \in \{1, ..., C\}$ by Eq. 7 to Eq. 8
7:     Calculate the joint loss $\ell_{js}$ by Eq. 9 to Eq. 10
8:     Update weights in $\mathbb{M}^c$ and $\mathbb{M}^e$ according to the gradient of the joint loss
9: end while
   Output: The feature extractor $\mathbb{M}^e$ and classifier $\mathbb{M}^c$
```

it with other crack classification approaches in Sec. 4.4, but also compared it with other state of the art loss functions in Sec. 4.5. Moreover, we compare the learned anchors with the centers learned by center loss in Sec. 4.6.

4.1. Experimental setup

In this paper, we carry out experiments based on the following crack classification datasets and implementation details.

NPP2021. This dataset is collected by unmanned aerial vehicles at the nuclear power plants, comprising 300 images with the size of 7000 × 6000. The images of this dataset include cracks as narrow as 0.05 mm and as wide as 10 mm. Due to the complex environment, the collected images contain a lot of noise. To augment the dataset without compromising the resolution, the images are sliced into images of 224 × 224 pixels, composing a final dataset with 13372 samples, which are carefully manually labeled in three classes: w/o cracks, w/ cracks and w/ scratches. The numbers of pictures in each category are 5317, 4254, and 3801, respectively. This dataset is divided into training set, validation set, and test set at the ratio of 3 : 1 : 1.



CRACK2019 [26, 43]**.** This dataset is partially from crack500 [43]. The data is collected from Temple University and various METU Campus Buildings. The dataset contains 40000 total images with 227 × 227 pixels with RGB channels, divided into non-crack class and crack class for the crack classification task, and each class has 20000 images. As this dataset is generated from 458 high-resolution images (4032 × 3024 pixels), which have variance in terms of surface finish and illumination condition, it is difficult to successfully classify the images. In our experiments, we divide it into training set, validation set, and test set at the ratio of 3 : 1 : 1.

SDNET2018 [1]**.** This dataset contains over 56000 images of crack and non-crack concrete bridge decks, walls, and pavements. The dataset includes cracks as narrow as 0.06 mm and as wide as 25 mm, and images with a variety of disturbances, including shadow, surface roughness, scaling, edge, and hole. In our experiments, we divide it into training set, validation set, and test set at the ratio of 3 : 1 : 1.

Implementation details. To verify the effectiveness of our method, our experiments select the ResNet50 [16] and VGG16 [30] as the feature extractor. Although the input resolutions of CRACK2019, SDNET2018, and NPP2021 are 227 × 227, 256 × 256, and 224 × 224 respectively, we resize all the images into the size of 224 × 224 and perform data augmentation via random horizontal flip and synthesis [17, 35]. For these three datasets, the models are trained with 150 epochs in total. The initial learning rate is set to 0.01 and reduced by a factor of 10 after 65, 95, 125 epochs. We use mini-batch SGD [15] as the optimizer and set the mini-batch size to 64. According to the setting in the paper [41], the maximum PGD step K is set to 10. During the initial period of the training epochs, the geometric distance is less informative when the classifier is not properly learned. Thus, the initial 30 epochs are burn-in period, in which we do not update the anchors and the model is only supervised by CE loss. For a fair comparison, all methods are implemented with the same training configuration when it is possible. Finally, we use two metrics to evaluate these methods (i) Error rate, denoted as ER. (ii) Error rate on adversarial features generated by PGD-20 [27], denoted as PGD-ER.

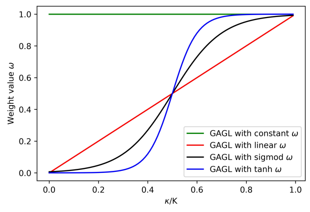
4.2. Experiments on re-weighting functions

Based on Sec. 3.3, the re-weighting functions should be the non-decreasing function with respect to the geometric distance κ. In Fig. 2, we compare the tanh-type Eq. 5 (Blue line) with three other types of re-weighting functions: (i) constant function ω, which means the function $\omega(\mathcal{F}, y) \equiv 1$ as shown by the green line. When GAGL takes constant $\omega(\mathcal{F}, y) \equiv 1$ over the training epochs, the anchors will be updated to the center of safe state features, and make other swing state features close to their corresponding class anchors; (ii) a linear increasing function $\omega(\mathcal{F}, y) = \frac{\kappa(\mathcal{F}, y)}{K}$ is shown by the red line; (iii) the sigmoid-type increasing function $\omega(\mathcal{F}, y) = \sigma(\beta + 5 \times (2\kappa(\mathcal{F}, y)/K - 1))$ is de-

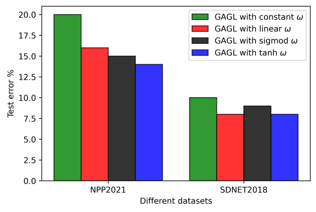
Methods \ Dataset	NPP2021		SDNET2018		CRACK2019	
	ER (%)	PGD-ER (%)	ER (%)	PGD-ER (%)	ER (%)	PGD-ER (%)
ConvNet [43]	21.65 ± 0.49	54.21 ± 0.38	13.87 ± 0.41	58.19 ± 0.23	2.31 ± 0.19	49.89 ± 0.21
Crack-CNN [8]	18.23 ± 0.23	51.03 ± 0.31	10.47 ± 0.27	53.85 ± 0.29	0.96 ± 0.23	46.01 ± 0.42
SDNET [1]	19.71 ± 0.32	51.34 ± 0.29	11.23 ± 0.23	55.6 ± 0.37	1.07 ± 0.27	46.31 ± 0.37
AliNet [3]	17.07 ± 0.31	48.15 ± 0.29	9.47 ± 0.26	52.3 ± 0.29	0.92 ± 0.18	44.67 ± 0.29
SilvaNet [29]	15.09 ± 0.23	47.76 ± 0.33	8.85 ± 0.28	43.9 ± 0.31	0.68 ± 0.21	41.54 ± 0.31
Res-GAGL	13.24 ± 0.18	44.15 ± 0.24	8.24 ± 0.32	40.37 ± 0.21	0.52 ± 0.17	39.21 ± 0.21
VGG-GAGL	**11.37 ± 0.19**	**42.76 ± 0.21**	**7.47 ± 0.25**	**38.2 ± 0.19**	**0.41 ± 0.21**	**37.12 ± 0.23**

Table 1. Comparison with the state-of-the-art crack classification methods on various datasets. We report the mean test error (ER), mean error rate on adversarial features generated by PGD-20 (PGD-ER), and their standard deviations by three independent experiments.

noted by the black line, where $\sigma(x) = \frac{1}{1+e^{(-x)}}$.



(a) Re-weighting functions



(b) Results with different functions

Figure 2. Comparison of GAGL with different re-weighting functions on NPP2021 dataset.

In Fig. 2, we can observe that compared with constant function, GAGL with different weight assignment functions have similar degradation of test error on the test dataset under $\beta = 0$, but GAGL with the tanh-type has the better test error. Thus, we use the tanh-type function in the next experiments, and further explore the Eq. 5 with different β in Sec. 4.3.

4.3. Sensitivity study on hyper-parameters

Three hyper-parameters have been introduced in this paper. α is used to control the magnitude of anchors update in Eq. 8, β is used to control the re-weighting function ω of GAGL, and λ is exploited to balanced the CE loss and GAGL. The hyper-parameter sensitivity study on the CRACK2019 dataset with ResNet50 as the feature extractor is introduced in Fig. 3. It is observed the three hyper-parameters α, β, and λ across a wide range only have 0.4%, 0.2%, 0.2% test error increase compared with the lowest, respectively, which demonstrates the satisfying stability and robustness of our proposed methods applied in real-world crack datasets. Based on these observations, we set $\alpha = 0.1$, $\beta = 4$ and $\lambda = 0.01$ in our next experiments.

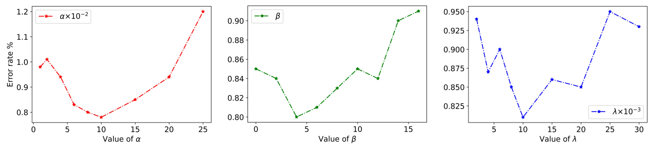


Figure 3. Hyper-parameter sensitivity study of α, β, λ with ResNet50 as feature extractor on CRACK2019 dataset.

4.4. Comparison on different crack classification approaches

In this section, our method is compared to several baselines including state-of-the-art robust crack classification methods: ConvNet [43] designs a supervised deep convolutional neural network to classify each image patch of the collected images, Crack-CNN [8] proposes a deep learning framework to deal with the noisy background in the image, SDNET [1] uses the AlexNet architecture to detect the crack, SilvaNet [29] uses the VGG16 architecture, AliNet [3] proposes a customized ResNet50 architecture. Since the previous works used the ResNet50 and VGG as the backbone, we also use these models as our backbone for the fairness. The Res-GAGL and VGG-GAGL denote ResNet50 [16] and VGG16 [30] under the supervision of the joint loss in Sec. 3.4, respectively. Both Res-GAGL and VGG-GAGL are compared with other approaches in terms of ER and PGD-ER metrics.

ER: In Tab. 1, it can be observed that our methods compare favorably to other competitive crack classification approaches. For example, compared with the AliNet [3], the test errors of Res-GAGL and VGG-GAGL are 13.24% and 11.37% while the AliNet only achieves 17.07% on NPP2021. For the SDNET2018, Res-GAGL and VGG-GAGL reduce test errors by about 0.61% and 1.38% compared with SilvaNet. Also for CRACK2019, the test errors of Res-GAGL and VGG-GAGL are 0.52% and 0.41% while the AliNet achieves 0.92%. Our methods show consistent improvements on these datasets. This indicates that GAGL can enhance the discriminative power of the deeply learned features by enlarging the intra-class compactness and inter-class separability, so that these features can be successfully classified to crack or non-crack class.

PGD-ER: As GAGL aims to turn the swing state features to safe state features, the learned features should be hard to be attacked. To verify this, we use the PGD-ER metrics to evaluate our methods compared with other approaches. In Tab. 1, compared with all existing methods, VGG-GAGL gives the best performance on all datasets. For example, the error rate on adversarial features generated by PGD-20 is 42.76% for the NPP2021 dataset. The reason for this effect is that our GAGL can guide the model to pull feature embeddings away from class boundaries, and the learned features are hard to be attacked.

4.5. Comparison on the different loss functions

To further validate our method, we conduct a series of experiments on SDNET2018 dataset and report quantitative results with ER metric and PGD-ER metric to verify the effectiveness of GAGL. We use VGG16 as our feature extractor in this section. As the intuition of GAGL is to enhance the model ability at extracting discriminate features, we compare it with other loss functions which have the same intuition, including CE loss [23], focal loss [20], center loss [37], cosface loss [36], arcface loss [9], and circle loss [31].

Methods	ER (%)	PGD-ER (%)
CE [23]	9.77 ± 0.23	54.3 ± 0.43
Focal loss [20]	9.43 ± 0.24	48.8 ± 0.78
Center loss [37]	9.28 ± 0.31	49.1 ± 0.24
Cosface [36]	8.92 ± 0.26	49.3 ± 0.32
Arcface [9]	8.53 ± 0.25	47.6 ± 0.19
Circleloss [31]	8.46 ± 0.33	40.5 ± 0.20
GAGL	$\mathbf{7.47 \pm 0.25}$	$\mathbf{38.2 \pm 0.19}$

Table 2. We report mean values and standard deviations by three independent experiments on the SDNET2018 dataset.

Tab. 2 shows the results compared with the existing loss functions. We can observe that, first, our GAGL outperforms the center loss with a large margin. This benefits from that the learned anchors have a larger geometric distance to the class boundaries than the centers learned by center loss. Then, the learned anchors can better guide the model to pull feature embeddings away from class boundaries and gather intra-class features together, which effectively helps to separate the crack and non-crack images in the feature space. Second, our GAGL performs well on the PGD-ER metric and achieves 38.2% test error on the adversarial features. The reason for this effect is that the learned features are optimized to get close to the learned anchors, and then the geometric distances of these features will be increased, as a result, these features are more difficult to be attacked.

4.6. Comparison between anchors and centers

To further validate the learned anchors by GAGL, we compare them with the centers learned by center loss using the average geometric distance of all classes as the metric, which is denoted as avg-κ. We train VGG16 and ResNet50 models with 150 epochs under the center loss and GAGL, respectively. We can observe from Fig. 4 that the anchors achieve a higher average geometric distance than the centers when the training epoch is over 50 under VGG16 and ResNet50 models. This demonstrates that our geometry-aware update rule helps learn the anchors far from the class boundaries.

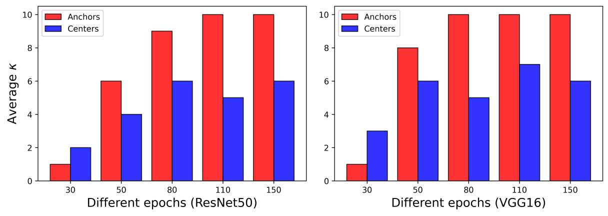


Figure 4. Comparison with centers by the average geometric distance avg-κ at different epochs on ResNet50 and VGG16.

5. Conclusion

In this paper, we propose the geometry-aware guided loss (GAGL) that enhances the discrimination ability of the deeply learned features for the crack classification task. The GAGL consists of the feature-based geometry-aware projected gradient descent method (FGA-PGD) that approximates the geometric distances of the features to the class boundaries, and the geometry-aware update rule that learns an anchor of each class as the approximation of the feature expected to have the largest geometric distance to the corresponding class boundary. Then, the goal of GAGL can be realized through simultaneously updating the anchors by the geometry-aware update rule and iteratively optimizing the features by penalizing the distances between the features and their corresponding class anchors in the feature space. The experiments on crack classification tasks demonstrate that our method outperforms state-of-the-art crack classification approaches and loss functions. Since our method can learn the discriminate features, we plan to further apply our method to other tasks, such as fine-grained image classification tasks and face recognition tasks.

Acknowledgements: This work was supported by the National Nature Science Foundation of China under Grants U2013201, 62073225, 62072315, 61836005 and 62006157, the Natural Science Foundation of Guangdong Province-Outstanding Youth Program under Grant 2019B151502018, the Guangdong "Pearl River Talent Recruitment Program" under Grant 2019ZT08X603.

References

[1] Sdnet2018: An annotated image dataset for non-contact concrete crack detection using deep convolutional neural networks. *Data in Brief*, 21:1664–1668, 2018. 6, 7

[2] Gyumin Lee a, Seung Jun Lee b, and Changyong Lee c. A convolutional neural network model for abnormality diagnosis in a nuclear power plant. *Applied Soft Computing*, 2020. 1

[3] Luqman Ali, Fady Alnajjar, Hamad Al Jassmi, Munkhjargal Gochoo, Wasif Khan, and M Adel Serhani. Performance evaluation of deep cnn-based crack detection and localization techniques for concrete structures. *Sensors*, 21(5):1688, 2021. 3, 7

[4] Seongdeok Bang, Somin Park, Hongjo Kim, Hyoungkwan Kim, et al. A deep residual network with transfer learning for pixel-level road crack detection. In *ISARC*, volume 35, pages 1–4, 2018. 1

[5] Yingchun Cai and Yamin Zhang. Research on pavement crack recognition methods based on image processing. In *ICIP*, 2011. 1

[6] Young-Jin Cha, Wooram Choi, and Oral Büyüköztürk. Deep learning-based crack damage detection using convolutional neural networks. *Computer-Aided Civil and Infrastructure Engineering*, 32(5):361–378, 2017. 1, 3

[7] Chady, Tomasz, Enokizono, Masato, Sikora, Ryszard, Todaka, Takashi, Tsuchida, and Yuji. Natural crack recognition using inverse neural model and multi-frequency eddy current method. *IEEE Transactions on Magnetics*, 2001. 1

[8] Fu-Chen Chen and Mohammad R Jahanshahi. Nb-cnn: Deep learning-based crack detection using convolutional neural network and naïve bayes data fusion. *IEEE Transactions on Industrial Electronics*, 65(5):4392–4400, 2017. 3, 7

[9] Jiankang Deng, Jia Guo, Niannan Xue, and Stefanos Zafeiriou. Arcface: Additive angular margin loss for deep face recognition. In *CVPR*, pages 4690–4699, 2019. 3, 8

[10] Sattar Dorafshan, Robert J Thomas, and Marc Maguire. Sdnet2018: An annotated image dataset for non-contact concrete crack detection using deep convolutional neural networks. *Data in brief*, 21:1664–1668, 2018. 3

[11] Fen Fang, Liyuan Li, Ying Gu, Hongyuan Zhu, and Joo-Hwee Lim. A novel hybrid approach for crack detection. *PR*, 107:107474, 2020. 2, 3

[12] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014. 3

[13] Kasthurirangan Gopalakrishnan. Deep learning in data-driven pavement image analysis and automated distress detection: A review. *Data*, 3(3):28, 2018. 1

[14] Kasthurirangan Gopalakrishnan, Siddhartha K Khaitan, Alok Choudhary, and Ankit Agrawal. Deep convolutional neural networks with transfer learning for computer vision-based data-driven pavement distress detection. *Construction and Building Materials*, 157:322–330, 2017. 1

[15] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017. 6

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *ECCV*, pages 630–645, 2016. 3, 6, 7

[17] Yifan Jiang, Shiyu Chang, and Zhangyang Wang. Transgan: Two pure transformers can make one strong gan, and that can scale up. In *NeurIPS*, 2021. 6

[18] Rafał Kapela, Paweł Śniatała, and Turkot. Asphalt surfaced pavement cracks detection based on histograms of oriented gradients. In *MIXDES*, pages 579–584, 2015. 2

[19] Christian Koch, Kristina Georgieva, Varun Kasireddy, Burcu Akinci, and Paul Fieguth. A review on computer vision based defect detection and condition assessment of concrete and asphalt civil infrastructure. *Advanced Engineering Informatics*, 29(2):196–210, 2015. 2

[20] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *ICCV*, pages 2980–2988, 2017. 3, 8

[21] Zhengping Luo, Shangqing Zhao, Zhuo Lu, Yalin E Sagduyu, and Jie Xu. Adversarial machine learning based partial-model attack in iot. In *Proceedings of the 2nd ACM Workshop on Wireless Security and Machine Learning*, pages 13–18, 2020. 2, 3

[22] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *ICLR*, 2018. 2, 3

[23] Shie Mannor, Dori Peleg, and Reuven Rubinstein. The cross entropy method for classification. In *ICML*, pages 561–568, 2005. 8

[24] Konstantinos Nikolaidis, John Yannis Goulermas, and QH Wu. A class boundary preserving algorithm for data condensation. *PR*, 44(3):704–715, 2011. 1

[25] Henrique Oliveira and Paulo Lobato Correia. Automatic road crack detection and characterization. *IEEE Transactions on Intelligent Transportation Systems*, 14(1):155–168, 2012. 2

[26] Ç F Özgenel and A Gönenç Sorguç. Performance comparison of pretrained convolutional neural networks on crack detection in buildings. In *ISARC*, volume 35, pages 1–8, 2018. 6

[27] Tianyu Pang, Xiao Yang, Yinpeng Dong, Kun Xu, Jun Zhu, and Hang Su. Boosting adversarial training with hypersphere embedding. *arXiv preprint arXiv:2002.08619*, 2020. 6

[28] Marcos Quintana, Juan Torres, and José Manuel Menéndez. A simplified computer vision system for road surface inspection and maintenance. *IEEE Transactions on Intelligent Transportation Systems*, 17(3):608–619, 2015. 2

[29] Wilson Ricardo Leal da Silva and Diogo Schwerz de Lucena. Concrete cracks detection based on deep learning image classification. In *Multidisciplinary digital publishing institute proceedings*, volume 2, page 489, 2018. 7

[30] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 6, 7

[31] Yifan Sun, Changmao Cheng, Yuhan Zhang, Chi Zhang, Liang Zheng, Zhongdao Wang, and Yichen Wei. Circle loss: A unified perspective of pair similarity optimization. In *CVPR*, pages 6398–6407, 2020. 8

[32] Yi Sun, Xiaogang Wang, and Xiaoou Tang. Deep learning face representation by joint identification-verification. *arXiv preprint arXiv:1406.4773*, 2014. 3

[33] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008. 3

[34] Srivatsan Varadharajan, Sobhagya Jose, Karan Sharma, Lars Wander, and Christoph Mertz. Vision for road inspection. pages 115–122, 2014. 2

[35] Chenglong Wang and Zhifeng Xiao. Lychee surface defect detection based on deep convolutional neural networks with gan-based data augmentation. *Agronomy*, 11(8):1500, 2021. 6

[36] Hao Wang, Yitong Wang, Zheng Zhou, Xing Ji, Dihong Gong, Jingchao Zhou, Zhifeng Li, and Wei Liu. Cosface: Large margin cosine loss for deep face recognition. In *CVPR*, pages 5265–5274, 2018. 3, 8

[37] Yandong Wen, Kaipeng Zhang, Zhifeng Li, and Yu Qiao. A discriminative feature learning approach for deep face recognition. In *ECCV*, pages 499–515, 2016. 1, 3, 8

[38] Z. Xu, X. Zhao, H. Song, T. Lei, and N. Wei. Asphalt pavement crack recognition algorithm based on histogram estimation and shape analysis. *Chinese Journal of Scientific Instrument*, 31(10):2260–2266, 2010. 1

[39] H. Zakeri, Fereidoon Moghadas Nejad, and Ahmad Fahimifar. Image based techniques for crack detection, classification and quantification in asphalt pavement: A review. *Archives of Computational Methods in Engineering*, 2016. 1

[40] Hamzeh Zakeri, F Moghadas Nejad, Ahmad Fahimifar, A Doostparast Torshizi, and MH Fazel Zarandi. A multi-stage expert system for classification of pavement cracking. In *Joint IFSA World Congress and NAFIPS Annual Meeting*, pages 1125–1130, 2013. 2

[41] Jingfeng Zhang, Jianing Zhu, Gang Niu, Bo Han, Masashi Sugiyama, and Mohan Kankanhalli. Geometry-aware instance-reweighted adversarial training. In *ICLR*, 2020. 6

[42] Kaige Zhang, HD Cheng, and Boyu Zhang. Unified approach to pavement crack and sealed crack detection using preclassification based on transfer learning. *Journal of Computing in Civil Engineering*, 32(2):04018001, 2018. 3

[43] Lei Zhang, Fan Yang, Yimin Daniel Zhang, and Ying Julie Zhu. Road crack detection using deep convolutional neural network. In *ICIP*, pages 3708–3712, 2016. 1, 6, 7

[44] Wang Zhao, Chunrong Hua, Dawei Dong, and Huajiang Ouyang. A novel method for identifying crack and shaft misalignment faults in rotor systems under noisy environments based on cnn. *Sensors*, 19(23):5158, 2019. 1

[45] Mingyi Zhou, Jing Wu, Yipeng Liu, Shuaicheng Liu, and Ce Zhu. Dast: Data-free substitute training for adversarial attacks. In *CVPR*, pages 234–243, 2020. 3