applied sciences

MDPI

Article

Robust Airport Surface Object Detection Based on Graph Neural Network

Wenyi Tang [1] and Hongjue Li [2,*]

[1] State Key Laboratory of Air Traffic Management System, Nanjing 210000, China
[2] School of Astronautics, Beihang University, Beijing 100191, China
* Correspondence: lihongjue@buaa.edu.cn

Abstract: Accurate and robust object detection is of critical importance in airport surface surveillance to ensure the security of air transportation systems. Owing to the constraints imposed by a relatively fixed receptive field, existing airport surface detection methods have not yet achieved substantial advancements in accuracy. Furthermore, these methods are vulnerable to adversarial attacks with carefully crafted adversarial inputs. To address these challenges, we propose the Vision GNN-Edge (ViGE) block, an enhanced block derived from the Vision GNN (ViG). ViGE introduces the receptive field in pixel space and represents the spatial relation between pixels directly. Moreover, we implement an adversarial training strategy with augmented training samples generated by adversarial perturbation. Empirical evaluations on the public remote sensing dataset LEVIR and a manually collected airport surface dataset show that: 1. our proposed method surpasses the original model in precision and robustness; 2. defining the receptive field in pixel space performs better than that on representation space.

Keywords: airport surface; object detection; graph neural network; adversarial training

Citation: Tang, W.; Li, H. Robust Airport Surface Object Detection Based on Graph Neural Network. *Appl. Sci.* **2024**, *14*, 3555. https://doi.org/10.3390/app14093555

Academic Editors: Andrea Prati and Antonio Fernández-Caballero

Received: 26 January 2024
Revised: 10 March 2024
Accepted: 11 April 2024
Published: 23 April 2024

1. Introduction

Airport surface object detection is a crucial technology for maintaining the security and efficiency of airport operations by analyzing real-time data streams, including video and radar. Specifically, airport surface video plays an important role in detecting and tracking planes automatically and localizing ground crews and vehicles. Leveraging airport surface object detection technologies, an airport administrator is capable of surveilling the airport efficiently, reducing the reliance on manual monitoring [1].

With the development of artificial intelligence techniques, a number of deep learning-based methods are proposed to address the object detection problem with advancements in accuracy and speed. Existing objection detection algorithms can be mainly divided into two categories: one-stage and two-stage approaches. The one-stage method such as SSD [2], YOLO [3–6] series offers higher detection speed. On the other hand, the two-stage method such as Faster RCNN [7] and R-FCN [8] provides greater accuracy at the cost of reduced speed, which is more suitable in real-time detection and is also the focus of this work.

Additionally, from the view of the neural network architecture, these objection detection methods can be divided into convolution-based [7] and transformer-based [9] frameworks. The convolution-based method extracts image features by leveraging convolutional operations with learnable convolutional kernels. Due to the square-like kernel shape, this method is more suitable to integrate information from a square-like receptive field (neighborhood). Not limited to the square-like receptive field, Dai et al. proposed deformable convolution [10] and deformable kernel [11] that explored different shapes of the receptive field. However, these deformable methods are not capable of representing spatial relations between two pixels (nodes). The transformer-based method [9] collects information from the entire feature map through the self-attention mechanism, i.e., a patch's

neighborhood is designed to contain all patches, resulting in a receptive field covering the entire feature map. Though they can reach great performance, these methods suffer from substantial computational costs.

Apart from these two methods, object detection based on graph neural networks (GNN) has been proposed nowadays. GNN is traditionally used for non-Euclidean data with inherent graphical structures, such as a molecular graph with its atoms and chemical bonds corresponding to nodes and edges, respectively. GNN was first introduced in [12,13], and the graph convolutional network [14] was proposed based on the spectral graph theory [15]. In recent years, a series of GNNs were introduced to deal with graph data efficiently, such as mixture model network [16], message passing neural network [17], and graph attention network [18].

The usages of GNN for computer vision mainly involve scene graph construction [19] and point cloud classification [20] because these two issues contain naturally constructed graphs. When GNN is introduced to image feature extraction, the first and most important problem is how to construct a graph for an image or a feature map. For a pixel in the feature map, Vision GNN (ViG) [21] defines its neighborhood (receptive field) as k-nearest-neighbor in representation space, which achieves great performance. However, as there is not a naturally constructed graph structure among all pixels, it is of great significance to explore which is a better way to construct a graph on pixels when introducing GNN into image feature extraction.

Recently, many researchers began to study how to utilize object detection in airport scenes. For airport surface object detection, Zhu et al. [1] proposed an attentional multiscale feature fusion enhancement network based on CNN. Li et al. [22] applied Ting-YOLO to airport aircraft object recognition and proposed a simplified Ting-YOLO framework to improve the detection speed. Guo et al. applied the YOLO v3 [23] algorithm to the field of aircraft detection on the airport surface and proposed several improvements. Han [24] introduced a unified and effective moving object detection architecture, which combined appearance and motion cues. As far as we know, there is no research exploring GNN-based method potential in this domain.

For airport surface object detection, the adversarial attack is considered a big challenge for model robustness and has been studied by more and more researchers. Adversarial attacks, whether arbitrarily caused or artificially crafted, are mainly performed by adding perturbation to the input image. Arbitrary perturbation results from weather changes, transmission failure, camera failure, and so on. This kind of perturbation can be detected by human eyes and solved immediately. On the contrary, artificial perturbation is implemented by malicious hackers who create imperceptible perturbation on an image. The wrong prediction in airport surveillance caused by artificial perturbation may lead to a serious safety problem. So, it is necessary to improve model robustness against adversarial attacks in our scenario.

Gradient-based adversarial attack methods, e.g., FGSM [25], PGD [26], are well studied these years as well as black box attack methods [27]. Meanwhile, researchers proposed several methods to tackle the problem of model robustness. Adversarial training methods, such as AdvProp [28] for classification and Det-AdvProp [29] for detection, create adversarial samples and add them to the training process to improve the model's robustness. RobustDet [30] modifies the model framework to defend against attack.

In this paper, we propose a Vision GNN-Edge (ViGE) block that is integrated into a one-stage detector's (i.e., YOLOv5s) backbone for airport surface surveillance. We explore different receptive field definitions and implement adversarial training. Our method aims to improve the detection accuracy and robustness of the GNN-based backbone, which can be applied in various backbones and other areas.

Our contributions are mainly summarized in three aspects:

- We propose a novel GNN-based block named ViGE and plug it into a one-stage detector's (i.e., YOLOv5s) backbone to improve the performance of object detection.

- We explore several receptive field (neighbor) definition approaches and find that defining the receptive field in pixel space is better than that in representation space empirically.
- We apply an adversarial training framework to improve model robustness against adversarial attacks for airport surface object detection.

2. Materials and Methods

In this section, we will first introduce and analyze the original Vision GNN block briefly. Subsequently, we will focus on the improvements of Vision GNN-Edge in the second part, including the graph construction and the message-passing neural network architecture. Finally, we present the adversarial training strategy in the third part.

2.1. ViG Block

Vision GNN (ViG) [21] is a pioneering work that first introduces GNN into image feature extraction. Several ViG blocks are stacked as a backbone and each ViG block takes a feature map $\boldsymbol{F} \in \mathbb{R}^{H\times W\times C}$ as input and outputs an updated feature map $\boldsymbol{F'} \in \mathbb{R}^{H\times W\times C}$. There are three basic modules in the ViG block: two convolutional neural networks (CNN) and a message-passing neural network (MPNN) (Figure 1). Firstly, in the CNN module, the ViG block implements convolutional operation on the feature map to capture local feature information. Then, in the MPNN module, the ViG block applies a convolutional operation on the feature map before the message-passing layer. The feature map is treated as a graph in message-passing layer where it implements message passing between nodes. After that, the feature map goes through the following CNN modules. Before output, the ViG block adds the original feature map $\boldsymbol{F}$ as a residual connection on the updated feature map $\boldsymbol{F'}$ to overcome gradient vanishing and over-smoothness caused by the graph neural network. We will describe the details of the message-passing layer in the second part.

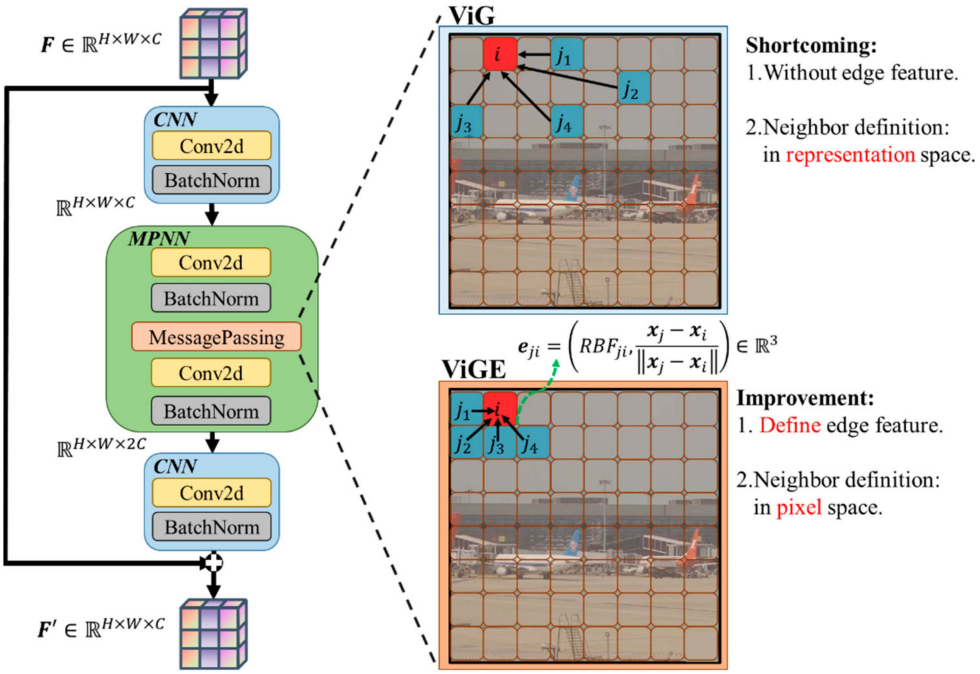


Figure 1. Illustration of ViG and ViGE block (number of neighbors is set to 4).

As there is not a naturally defined graph structure for a feature map, we can define any graph structure with inductive bias. Specifically, ViG defines graph structure in the representation space and achieves competitive performance. After graph construction, GNN facilitates message passing among graph nodes, aggregating information from the receptive field, which is also the same as neighboring nodes. Despite the innovative

approach introduced by ViG, it has yet to fully leverage the potential of GNN, such as in defining edge features or investigating alternative approaches to neighbor definition.

2.2. ViGE Block2

Our ViGE block is constructed based on the ViG's framework with three principal enhancements: graph construction, message passing, and model usage. Initially, in the graph construction phase, while ViG defines the k-nearest neighbors based on the feature proximity in the representation space and leverages position embedding to represent the position information, it overlooks the edge feature. In contrast, our ViGE defines neighbors in the pixel space and employs carefully designed edge features to represent two nodes' spatial relations more directly. After graph construction, we design a novel message-passing operation that leverages edge features to gather information from neighbors. Different from ViG, which stacks several ViG blocks as a backbone, our approach integrates a single ViGE block within the backbone of a one-stage detector, such as YOLOv5, to assess its compatibility with mainstream CNN architecture. The following sections will explain these differences in detail.

Graph construction. Considering a feature map $\boldsymbol{F}$ with dimensions $H \times W \times C$, where H, W, and C represent height, width, and channel, respectively, we segment it into $H \times W$ nodes, each representing a pixel and transforming $\boldsymbol{F}$ into a node set $\boldsymbol{\mathcal{V}} = [v_1, v_2, \cdots, v_{HW}]$ with each node feature $v_i \in \mathbb{R}^C$. The proximity between any two distinct nodes is determined by their Euclidean distance in the pixel space, representing the spatial separation of corresponding pixels in the feature map. The closer two nodes are in pixel space, the more likely they are neighbors in a graph. Based on the pixel Euclidean distance between nodes, we identify each node v_i, with its k-nearest neighbors $\mathcal{N}(v_i)$ and add an edge $e_{ji} \in \boldsymbol{\mathcal{E}}$ from node v_j to v_i, where $v_j \in \mathcal{N}(v_i)$. Furthermore, we employ the dilation number to integrate information from different distances. Based on the pairwise distances between nodes, a node is selected as a neighbor for every increment of "dilation number" in the sequence, thereby ensuring a systematic and sparse selection of neighboring nodes based on spatial proximity. A diagram of neighbor definition in pixel space is shown in the following Figure 2, where k is finally set to 12 and the dilation number is set to 3.

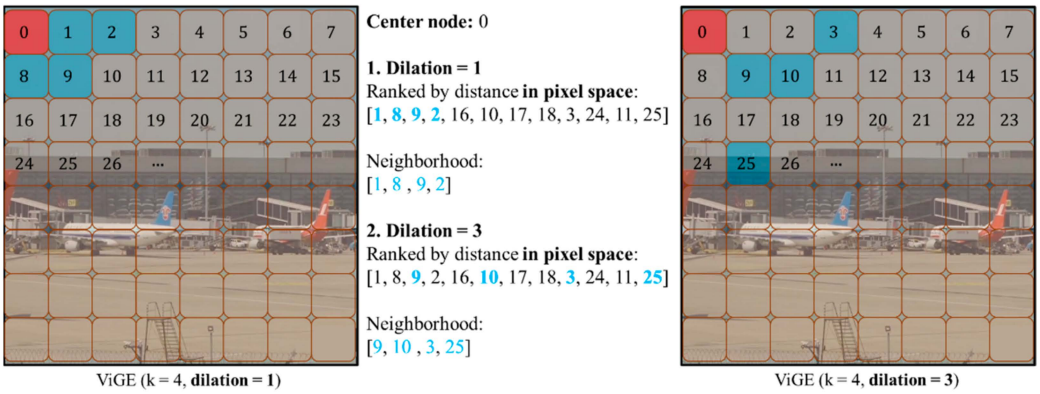


Figure 2. Neighbor definition in pixel space. (Blue number denotes the corresponding neighbor of the center node).

After defining neighbors, we construct the edge feature according to the distance and direction between two nodes:

$$\boldsymbol{e_{ji}} = Concat\left(RBF_{ji}, \frac{\boldsymbol{x_j} - \boldsymbol{x_i}}{\|\boldsymbol{x_j} - \boldsymbol{x_i}\|_2}\right) \in \mathbb{R}^3$$

where $\boldsymbol{x_i} \in \mathbb{R}^2$ denotes the Cartesian coordinate vector of node $\boldsymbol{v_i}$ in feature map $\boldsymbol{F}$; $\|\cdot\|_2$ denotes L-2 norm, which is usually used to calculate Euclidean distance; $Concat(\cdot,\cdot)$ denotes

concatenation operation; $RBF_{ji} \in \mathbb{R}$ is a radius basis function scalar between node v_j and v_i which is transformed from the distance in pixel space:

$$RBF_{ji} = \exp\left(-\frac{\left\|x_j - x_i\right\|_2}{\sqrt{H^2 + W^2}}\right) \in \mathbb{R}$$

By transforming the feature map $\boldsymbol{F}$ into a graph $\mathcal{G}(\boldsymbol{F}) = (\boldsymbol{\mathcal{V}}, \boldsymbol{\mathcal{E}})$, we enable the application of Graph Neural Network (GNN) methods for effective feature extraction. It is important to note that the vanilla ViG block does not define edge features as described previously but integrates position embedding to each node feature v_i to encode positional information. The original position embedding is sinusoidal and recurrent, which does not directly represent the spatial relationship between two nodes' spatial relation.

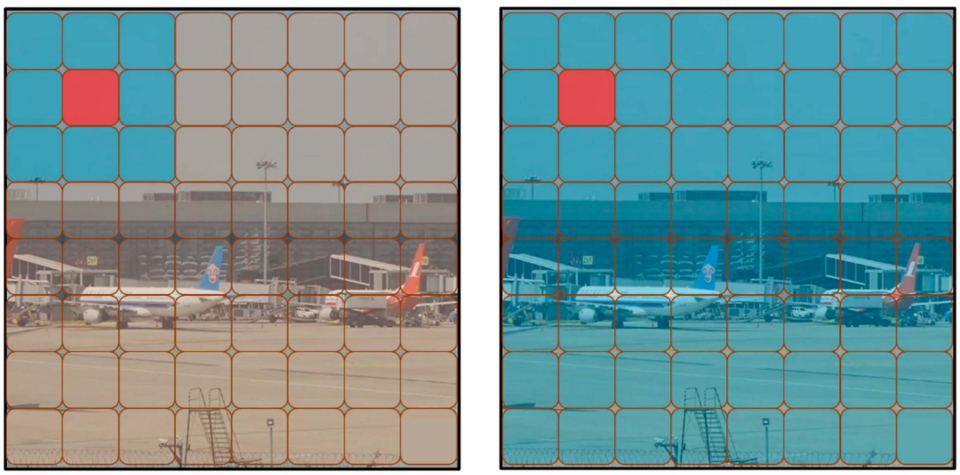
Message passing. With defined a graph with nodes, edges, and corresponding features, we can operate message passing to update node representation. Given a set of node feature $\boldsymbol{\mathcal{V}} = [v_1, v_2, \cdots, v_{HW}] \in \mathbb{R}^{HW \times C}$, on the l^{th} message passing layer, for a node v_i, we have:

$$agg_j = Conv_1\left(Concat\left(v_i^{(l)}, v_j^{(l)}, e_{ji}\right)\right) \in \mathbb{R}^{2C+3},\ v_j \in \mathcal{N}(v_i)\ and\ l = 0, 1, \cdots, L-1$$

$$v_i^{(l+1)} = Conv_2\left(Concat\left(v_i^{(l)}, \frac{1}{k}\sum_{j=1}^{k} agg_j\right)\right) \in \mathbb{R}^C$$

where agg_i denotes aggregated information from neighbor node $v_j \in \mathcal{N}(v_i)$ to center node v_i; $v_i^{(l)}$ denotes node feature v_i on the l^{th} layer; and e_{ji} represents the edge feature between two nodes, which $v_i^{(0)}$ is equal to original feature v_i before message passing; $Conv_1(\cdot)$ and $Conv_2(\cdot)$ denote convolutional operations with different parameters. To stabilize the training process, the parameters of two convolution layers are shared between different message-passing layers, respectively. After L message-passing layers, we obtain the updated node feature $\boldsymbol{\mathcal{V}}^{(L)} = \left[v_1^{(L)}, v_2^{(L)}, \cdots, v_{HW}^{(L)}\right] \in \mathbb{R}^{HW \times C}$.

Different from convolution and transformer operations, the ViGE block aggregates information from each node's k-nearest neighbors defined in pixel space. The convolution method aggregates information from each node's surrounding neighbors in pixel space and the transformer method aggregates information from all nodes by deploying the self-attention mechanism. Treating the feature map as a graph, we illustrate neighbor definitions of convolution and transformer in Figure 3, respectively, where the red patch denotes the center node and the blue patch denotes the neighbor node.



a. Convolution's neighbor definition (kernel size = 3×3) b. Transformer's neighbor definition

Figure 3. Neighbor definitions of convolution and transformer.

After building ViGE block, we add one into YOLOv5s' backbone to gather information from defined neighbors (Figure 4). YOLOv5 is typically separated into three parts: backbone, neck, and detection head. We follow vanilla YOLOv5 and implement the backbone network with CSPDarknet [4], which employs Cross-Stage Partial (CSP) and CSP Bottleneck with a Lite head (CBL) for information integration. For simplicity, CSP1_X represents X residue unit in a single CSP block. The single ViGE block is integrated after the second CSP block to gather information for the feature map generated by previous multiple convolution layers.

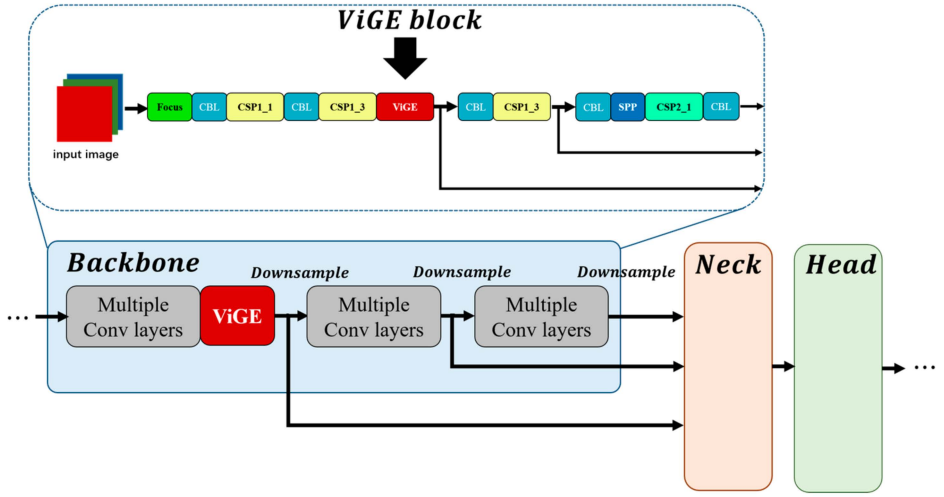


Figure 4. Framework of YOLOv5s with a ViGE block.

2.3. Adversarial Training

Adversarial sample generation. Given a clean image I and object detector D_θ, we generate adversarial perturbation based on the gradient derived from the loss function [29]:

$$\hat{I} = \mathbb{B}(I + \epsilon \cdot sign(\nabla_I \mathcal{L}(D_\theta(I), GT))) \quad (1)$$

where $\hat{I}$ denotes the adversarial sample derived from the clean image I; $\mathbb{B}(\cdot)$ is a boundary function that restricts the amplitude of perturbation, s.t. $\left\{\hat{I} \middle| \left\|\hat{I} - I\right\|_\infty \leq \epsilon\right\}$; ϵ is the step size of adversarial perturbation; $sign(\cdot)$ is a signum function, $sign(x) = 1$ when $x \geq 0$ and vice versa; $\mathcal{L}(\cdot)$ is a loss function of the object detector in the training phase, e.g., classification loss, location loss, and so on; $D_\theta(I)$ is the prediction of the object detector according to the clean image I; θ denotes the detector's parameters; GT denotes ground truth of image I, i.e., classification label, ground truth bounding box.

According to Equation (1) above, we first apply the model D_θ to detect clean image I and calculate its loss with ground truth GT. Then, we calculate the gradient vector of loss $\mathcal{L}$ w.r.t the clean image I and implement a step of gradient ascent on image I with step size ϵ. Adding this adversarial perturbation may cause the model to obtain the wrong detection results. If step size ϵ is small enough, the perturbation cannot be perceived by human eyes. What is more, we can iterate Equation (1) several times to obtain an adversarial sample by different adversarial iterations.

Contrary to adversarial sample generation, updating the model is based on gradient descent:

$$\theta' = \mathbb{B}(\theta - \epsilon \cdot \nabla_\theta \mathcal{L}(D_\theta(I), GT)) \quad (2)$$

where θ' denotes the updated parameters.

Equation (2) calculates gradient w.r.t model parameters θ. It aims to modulate model parameters to decrease the loss value. Meanwhile, Equation (1) aims to add some imperceptible perturbation on the image to increase the loss value, i.e., causing the wrong detection.

Adversarial training. For a clean image I, we can generate amounts of adversarial samples by attacking different losses or attacking several times. After calculating the loss

of clean image I, we implement an adversarial attack on it and obtain adversarial samples. Then, we apply the same model to detect these adversarial images and calculate loss with the same ground truth as I's to form the total training objective:

$$\mathcal{L} = \mathcal{L}_{OD} + \lambda_{adv} \cdot \mathcal{L}_{adv}$$

where λ_{adv} is the weight that balances the gradients calculated from the clean image and adversarial samples and is set as 0.2. The pseudo-code of adversarial training is in the following Algorithm 1.

Algorithm 1: Pseudo code of Adversarial training

Input: Dataset $\mathcal{D}$

Output: Updated network parameters θ

for *each training epoch* **do**

  Sample a batch $\{I, GT\} \in \mathcal{D}$;

  Compute object detection loss $\mathcal{L}_{\mathrm{OD}}(\mathbf{D}_\theta(I), GT)$;

  Generate adversarial sample;

  $\hat{I} = B(I + \epsilon \cdot sign(\nabla_I \mathcal{L}_{\mathrm{od}}(\mathbf{D}_\theta(\boldsymbol{I}), GT)))$, which $\mathcal{L}_{\mathrm{od}}$ can be part of $\mathcal{L}_{\mathrm{OD}}$;

  Compute adversarial loss $\mathcal{L}_{\mathrm{adv}} = \mathcal{L}_{\mathrm{OD}}(\mathbf{D}_\theta(\hat{I}), GT)$;

  Compute total loss $\mathcal{L} = \mathcal{L}_{\mathrm{OD}} + \lambda_{adv} \cdot \mathcal{L}_{\mathrm{adv}}$;

  Perform a step of gradient descent w.r.t. θ: min $\mathcal{L}$

end

3. Results

We will introduce the datasets and experiment settings in the following sections, along with three experiments: model comparison on LEVIR, model comparison on airport surface dataset, and model robustness under adversarial attack on LEVIR and airport surface dataset. Due to some copyright reasons, we do not have access to a well-annotated airport surface dataset. Without loss of generality, we evaluate our method on an open remote sensing dataset LEVIR and a manually collected airport surface dataset.

3.1. Dataset and Set Up

Dataset. The datasets we use include LEVIR [31] and the airport surface dataset. LEVIR is a popular remote sensing target detection dataset derived from Google Earth, which is one order of magnitude larger than any other datasets in this field. This dataset contains 3.8 k labeled images with three kinds of labels: plane, ship, and tank (i.e., oil pot). The instance numbers of plane, ship, and tank are 5030, 3197, and 2851, respectively. Each image in LEVIR has a size of 800 × 600 pixels and a resolution of 0.2~1.0 m/pixels. Building upon the methodology of prior research [32], we adopt a randomized dataset division approach, apportioning the images into training, validation, and testing sets in a ratio of 8:1:1, respectively.

To meet the demand for airport surveillance, we collect several airport surface videos from public resources on the Internet. Since the difference between adjacent frames is too slight, we select an image every five frames and annotate it manually. Finally, we build a small-scale dataset with only 143 images of 1920 × 1080 pixels and two label categories: airplane and vehicle. The instance numbers of airplane and vehicle are 187 and 56, respectively. We employed a random partitioning of the dataset according to distinct videos three times, maintaining a training, validation, and testing split ratio of 8:1:1. We ensured that no frame from any single video was concurrently represented in the training, validation, and testing subsets.

Experiment Set up. All experiments in this paper are performed on the same server with CPU of Intel Xeon Gold 6226R 2.90 GHz, 4 GPUs of NVIDIA RTX 3090, the operating system of Ubuntu 18.04, and PyTorch version 1.10.1. Each minibatch comprises 32 images with the resolution of 512 × 512 pixels. We use SGD optimizer to implement gradient descent with an initial learning rate of 0.1, momentum of 0.937, and weight decay of 0.0005. Image preprocessing is the same as YOLOv5, which keeps the width–height ratio, reshapes to a given size, and uses gray padding. To improve the performance on small objects, we follow YOLOv5 to employ mosaic data augmentation, which joins four images into one single image where objects are at a smaller scale. We train each model no more than 300 epochs with patience value of 100. It means that if the loss does not decrease lasting for 100 epochs, we treat the training loss as convergent, stop training, and save the best and the last models' parameters. It is noted that the best model performs the best on the evaluation set. With the best and the last models' parameters, we test both of them on the test set and record the one that reaches higher accuracy.

The models we compare in this paper includes vanilla YOLOv5s, YOLOv5m, YOLOv5s -ViGE, YOLOv5s -ViG, YOLOv5s -A, YOLOv5s -ViGE -A, and YOLOX-s [5]. The vanilla YOLOv5 series is derived from Ultralytics' source code on GitHub, including small-size YOLOv5s and middle-size YOLOv5m mentioned above; '-ViGE' denotes a base model added a ViGE block in the backbone, which defines 12-nearest-neighbor in pixel space with dilation number =3; '-ViG' denotes a base model added an original ViG [21] block in the backbone, which defines 12-nearest-neighbor in pixel space with dilation number =3; '-A' denotes implementing adversarial training by attacking location loss for one iteration. Considering the computational cost, we only set one message-passing layer in each ViGE block. YOLOX-s is another outstanding model of YOLO series after YOLOv5. All the models we used are trained from scratch.

3.2. Result and Analysis

Result on LEVIR. We train and evaluate several models on the LEVIR dataset; the results are in Table 1.

Table 1. Model performance on LEVIR.

Models	P ↑	R ↑	mAP.5 ↑	mAP.5:.95 ↑
YOLOv5s	0.926	0.883	0.937	0.684
YOLOv5m	0.939	0.910	0.957	0.725
YOLOv5s-ViG	0.933	**0.913**	0.926	0.681
YOLOX-s	--	--	**0.960**	0.716
YOLOv5s-ViGE (ours)	**0.945**	0.912	0.958	**0.737**

Note: The ↑ symbol means the larger the better; the best performance is marked in bold text. The -- symbol means that metrics are not provided in the result.

We illustrate the detection result on an image in Figure 5. According to the experiment above, we can conclude that:

- Compared with vanilla YOLOv5s, ViGE block brings a significant improvement of about 2% for mAP.5. This proves that introducing a new and proper receptive field for YOLOv5s' backbone is able to extract more effective features.
- Compared with the original ViG block, the ViGE block performs better with about 3% mAP.5 increase. The ViG block defines neighbor in representation space with positional embedding, while the ViGE block defines neighbor in pixel space with a well-designed edge feature. The reason why ViGE outperforms the original ViG may come from two aspects. One is that defining neighbor in pixel space does better than that in representation space. This phenomenon confirms the previous success of CNN, which gathers information from neighbors in pixel space. The other is that our edge feature can better capture spatial relations than positional embedding.

- YOLOv5s-ViGE shows competence against other popular models, i.e., YOLOv5m and YOLOX-s. It is noticed that YOLOX adopts several tricks on the basis of YOLOv5, e.g., decouple head, SimOTA, and so on. We only add a ViGE block on the basis of YOLOv5s and obtain an approximate accuracy with YOLOX-s, which proves the effectiveness of the ViGE block in improving detection accuracy.

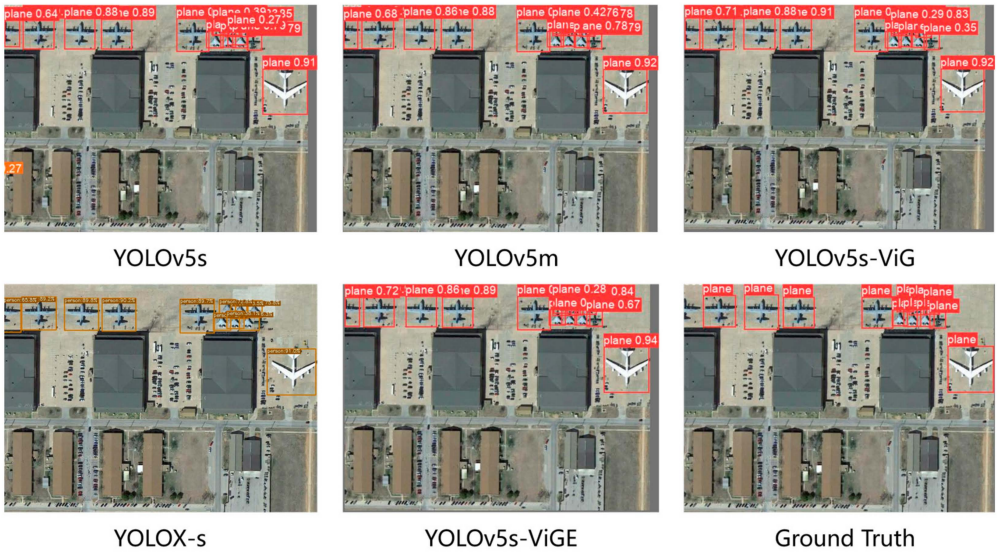


Figure 5. Detection results for LEVIR dataset (a sample).

Result on airport surface dataset. We show different models' test results with their average value and standard deviation on the airport surface dataset in Table 2 as follows.

Table 2. Model performance on airport surface dataset.

Models	P ↑	R ↑	mAP.5 ↑	mAP.5:.95 ↑
YOLOv5s	**0.851 ± 0.020**	0.667 ± 0.018	0.768 ± 0.024	0.378 ± 0.003
YOLOv5m	0.768 ± 0.016	0.773 ± 0.005	0.823 ± 0.011	0.390 ± 0.009
YOLOv5s-ViG	0.805 ± 0.012	**0.797 ± 0.004**	0.822 ± 0.007	**0.386 ± 0.009**
YOLOX-s	--	--	0.756 ± 0.005	0.367 ± 0.006
YOLOv5s-ViGE (ours)	0.814 ± 0.013	0.741 ± 0.003	**0.837 ± 0.005**	0.381 ± 0.007

Note: The ↑ symbol means the larger the better; the best performance is marked in bold text. The -- symbol means that metrics are not provided in the result.

We illustrate the detection result on an image in Figure 6:

Figure 6. Detection results for airport surface dataset (a sample).

According to the experiment above, we obtain the conclusion that:

- Our YOLOv5s-ViGE model gets about 7% mAP.5 improvement compared to vanilla YOLOv5s, which verifies the ViGE block's potential on feature extraction. What is more, the original ViG block brings accuracy improvement to YOLOv5s as well. These phenomena demonstrate that GNN can also perform well in image feature extraction.
- Though YOLOX-s performs better on LEVIR, its accuracy is lower than YOLOv5s-ViGE on the airport surface dataset. This result indicates that YOLOv5s-ViGE obtains a better convergence on small-scale datasets from another side.
- In our experiments, we have explored several neighbor definition approaches with different setups, including k-nearest-neighbor in pixel space, k-nearest-neighbor in representation space, and k-farthest-neighbor in representation space. The worst definition is k-farthest-neighbor in representation space, whose accuracy always stays at 0 during training. Obviously, gathering information from irrelevant parts does not help to improve model performance. Though the original ViG is amazing, defining neighbors in pixel space surpasses that in representation space verified by our experiments. As for parameter setups, i.e., radius k and dilation number, they need to be adjusted carefully to define a proper receptive field. Due to space limitations, we do not show this part's experiment.

Result of robustness under adversarial attack. When applying detection models for airport surveillance, it is necessary to improve their robustness against potential attacks. Considering the importance of ensuring perturbations remain imperceptible within images, we focus on the gradient-based artificial perturbation methods. These methods are well-studied these years and constitute the primary challenge that adversarial training approaches aim to mitigate [28,29]. For each salient sample, we attack the model's location loss for five iterations to generate a corresponding adversarial sample based on the PGD algorithm [26]. Then, adversarial samples are utilized to test the robustness of models. The result is in Table 3 as follows.

Table 3. Model robustness under adversarial attack.

Models	LEVIR				Airports			
	Without Attack		Under Attack		Without Attack		Under Attack	
	mAP.5	mAP.5:.95	mAP.5	mAP.5:.95	mAP.5	mAP.5:.95	mAP.5	mAP.5:.95
YOLOv5s	0.937	0.684	0.492	0.211	0.786	0.377	0.315	0.076
YOLOv5m	0.957	0.725	0.445	0.197	0.827	0.396	0.314	0.073
YOLOv5-ViG	0.926	0.681	0.517	0.200	0.825	0.399	0.273	0.065
YOLOv5s-ViGE	**0.958**	**0.737**	0.486	0.171	0.832	0.383	0.203	0.052
YOLOv5s-A	0.943	0.718	0.760	**0.406**	0.833	0.384	0.457	0.125
YOLOv5s-ViGE-A	0.945	0.717	**0.773**	**0.406**	**0.836**	**0.431**	**0.508**	**0.155**

Note: The best performance is marked in bold text.

According to the experiment above, we come to the conclusion that:

- On the LEVIR and airport surface dataset, the adversarial attack does cause a significant loss in object detection accuracy for all models above. Especially for detectors with GNN block, they are more vulnerable than purely convolution-based detectors (i.e., YOLOv5s and YOLOv5m) against attack. In detail, detectors with GNN block obtain about 38% and 67% mAP.5 decreases while vanilla YOLOv5s models obtain about 47% and 60% on two datasets, respectively. When using a GNN-based detector in safety supervision, its poor robustness must be taken into consideration.
- Adversarial training does help to improve model robustness against attack. Although models through adversarial training still suffer from accuracy loss towards adversarial attacks, this phenomenon has been alleviated obviously compared with those models without adversarial training.

- Adversarial training also helps to improve detection accuracy. This framework takes the use of adversarial samples as auxiliary samples to train the model, which can be seen as a data augmentation approach. It is known that data augmentation is able to improve the model's performance, especially for small-scale datasets.
- Model performance under various settings of adversarial attack is provided in Table 4. As the number of attack iterations rises to 10, there is a significant drop in the mAP.5 of the GNN-based model without adversarial training. However, the application of adversarial training mitigates these adverse effects under different adversarial settings.

Table 4. Model performance under various settings of adversarial attack.

Models	Without Attack		5 Iterations Attack		10 Iterations Attack	
	mAP.5	mAP.5:.95	mAP.5	mAP.5:.95	mAP.5	mAP.5:.95
YOLOv5s-ViGE	**0.958**	**0.737**	0.486	0.171	0.320	0.156
YOLOv5s-ViGE–A-0.1	0.943	0.719	0.764	0.397	0.667	**0.340**
YOLOv5s-ViGE–A-0.2	0.945	0.717	**0.773**	**0.406**	**0.683**	0.322

Note: The best performance is marked in bold text.

4. Conclusions

In this paper, we design a GNN-based feature extraction block, called ViGE, and implement adversarial training for the popular object detector YOLOv5s. Different from CNN which aggregates local messages and transformer which gathers global information through self-attention, ViGE block aggregates information from each node's k-nearest-neighbors in pixel (or representation) space. Compared to the original ViG block, our ViGE block defines edge features that can represent spatial relations more directly. Considering potential perturbation in airport surveillance, it is necessary to enhance the model's robustness, especially towards adversarial attacks. We perform sufficient experiments to prove our methods' effectiveness in detection accuracy and robustness improvement.

When introducing GNN into image representation learning, how to construct a graph for an image (or a feature map) is the first problem we encounter. Though we have explored several neighbor definitions approaches in this article (i.e., in different spaces with various parameters), there still are other promising approaches we have not discovered. What is more, the noisy node trick [33] may help to improve GNN's robustness. Furthermore, the performance of the GNN on datasets with a broader range of object classes, such as the DOTA dataset [34], remains to be researched. Like transformers, we do anticipate that GNN can also obtain great success in the computer vision domain. However, there is a long way to go.

Author Contributions: Conceptualization, W.T. and H.L.; methodology, W.T.; software, W.T.; validation, W.T. and H.L.; formal analysis, H.L.; investigation, W.T.; resources, W.T.; data curation, W.T.; writing—original draft preparation, W.T.; writing—review and editing, H.L.; visualization, W.T.; supervision, H.L.; project administration, H.L.; funding acquisition, H.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Guangdong Basic and Applied Basic Research Foundation, grant number 2022A1515110837. This research was also funded by the State Key Laboratory of Air Traffic Management System Funding Project, grant number SKLATM202110.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The open remote sensing dataset LEVIR can be accessed from [31].

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Zhu, X.; Liang, B.; Fu, D.; Huang, G.; Yang, F.; Li, W. Airport small object detection based on feature enhancement. *IET Image Process.* **2022**, *16*, 2863–2874. [CrossRef]
2. Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C.Y.; Berg, A.C. Ssd: Single shot multibox detector. In Proceedings of the European Conference on Computer Vision, Amsterdam, The Netherlands, 8–16 October 2016; Springer: Cham, Switzerland, 2016; pp. 21–37.
3. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You only look once: Unified, real-time object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 779–788.
4. Bochkovskiy, A.; Wang, C.Y.; Liao, H.Y.M. Yolov4: Optimal speed and accuracy of object detection. *arXiv* **2020**, arXiv:2004.10934.
5. Ge, Z.; Liu, S.; Wang, F.; Li, Z.; Sun, J. Yolox: Exceeding yolo series in 2021. *arXiv* **2021**, arXiv:2107.08430.
6. Li, C.; Li, L.; Jiang, H.; Weng, K.; Geng, Y.; Li, L.; Ke, Z.; Li, Q.; Cheng, M.; Nie, W.; et al. YOLOv6: A Single-Stage Object Detection Framework for Industrial Applications. *arXiv* **2022**, arXiv:2209.02976.
7. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster R-CNN: Towards real-time object detection with region proposal networks. *IEEE Trans. Pattern Anal. Mach. Intell.* **2016**, *39*, 1137–1149. [CrossRef] [PubMed]
8. Dai, J.; Li, Y.; He, K.; Sun, J. R-fcn: Object detection via region-based fully convolutional networks. In Proceedings of the 30th International Conference on Neural Information Processing Systems, Barcelona, Spain, 5–10 December 2016; p. 29.
9. Carion, N.; Massa, F.; Synnaeve, G.; Usunier, N.; Kirillov, A.; Zagoruyko, S. End-to-end object detection with transformers. In Proceedings of the European Conference on Computer Vision, Glasgow, UK, 23–28 August 2020; Springer: Cham, Switzerland, 2020; pp. 213–229.
10. Dai, J.; Qi, H.; Xiong, Y.; Li, Y.; Zhang, G.; Hu, H.; Wei, Y. Deformable convolutional networks. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 764–773.
11. Gao, H.; Zhu, X.; Lin, S.; Dai, J. Deformable kernels: Adapting effective receptive fields for object deformation. *arXiv* **2019**, arXiv:1910.02940.
12. Gori, M.; Monfardini, G.; Scarselli, F. A new model for learning in graph domains. In Proceedings of the 2005 IEEE International Joint Conference on Neural Networks, Montreal, QC, Canada, 31 July–4 August 2005; Volume 2, pp. 729–734.
13. Scarselli, F.; Gori, M.; Tsoi, A.C.; Hagenbuchner, M.; Monfardini, G. The graph neural network model. *IEEE Trans. Neural Netw.* **2008**, *20*, 61–80. [CrossRef] [PubMed]
14. Kipf, T.N.; Welling, M. Semi-supervised classification with graph convolutional networks. *arXiv* **2016**, arXiv:1609.02907.
15. Chung, F.R. *Spectral Graph Theory*; American Mathematical Society: Providence, RI, USA, 1997; Volume 92.
16. Monti, F.; Boscaini, D.; Masci, J.; Rodola, E.; Svoboda, J.; Bronstein, M.M. Geometric deep learning on graphs and manifolds using mixture model cnns. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 5115–5124.
17. Gilmer, J.; Schoenholz, S.S.; Riley, P.F.; Vinyals, O.; Dahl, G.E. Neural message passing for quantum chemistry. In Proceedings of the International Conference on Machine Learning, Sydney, NSW, Australia, 6–11 August 2017; pp. 1263–1272.
18. Velickovic, P.; Cucurull, G.; Casanova, A.; Romero, A.; Lio, P.; Bengio, Y. Graph attention networks. *arXiv* **2017**, arXiv:1710.10903.
19. Yang, J.; Lu, J.; Lee, S.; Batra, D.; Parikh, D. Graph r-cnn for scene graph generation. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 670–685.
20. Landrieu, L.; Simonovsky, M. Large-scale point cloud semantic segmentation with superpoint graphs. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 4558–4567.
21. Han, K.; Wang, Y.; Guo, J.; Tang, Y.; Wu, E. Vision GNN: An Image is Worth Graph of Nodes. *arXiv* **2022**, arXiv:2206.00272.
22. Guo, J.; Liu, L.; Xu, F.; Zheng, B. Airport Scene Aircraft Detection Method Based on YOLO v3. *Laser Optoelectron. Prog.* **2019**, *56*, 191003.
23. Redmon, J.; Farhadi, A. Yolov3: An incremental improvement. *arXiv* **2018**, arXiv:1804.02767.
24. Han, S.C.; Zhang, B.H.; Li, W.; Zhan, Z.H. Moving Object Detection for Airport Scene Using Patterns of Motion and Appearance. *J. Aerosp. Inf. Syst.* **2021**, *18*, 852–859. [CrossRef]
25. Gowal, S.; Rebuffi, S.A.; Wiles, O.; Stimberg, F.; Calian, D.A.; Mann, T.A. Improving robustness using generated data. *Adv. Neural Inf. Process. Syst.* **2021**, *34*, 4218–4233.
26. Mądry, A.; Makelov, A.; Schmidt, L.; Tsipras, D.; Vladu, A. Towards deep learning models resistant to adversarial attacks. *arXiv* **2017**, arXiv:1706.06083.
27. Yang, J.; Jiang, Y.; Huang, X.; Ni, B.; Zhao, C. Learning black-box attackers with transferable priors and query feedback. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 12288–12299.
28. Xie, C.; Tan, M.; Gong, B.; Wang, J.; Yuille, A.L.; Le, Q.V. Adversarial examples improve image recognition. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 13–19 June 2020; pp. 819–828.
29. Chen, X.; Xie, C.; Tan, M.; Zhang, L.; Hsieh, C.J.; Gong, B. Robust and accurate object detection via adversarial learning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Nashville, TN, USA, 20–25 June 2021; pp. 16622–16631.
30. Dong, Z.; Wei, P.; Lin, L. Adversarially-Aware Robust Object Detector. *arXiv* **2022**, arXiv:2207.06202.
31. Zou, Z.; Shi, Z. Random access memories: A new paradigm for target detection in high resolution aerial remote sensing images. *IEEE Trans. Image Process.* **2017**, *27*, 1100–1111. [CrossRef] [PubMed]

32. Shamsolmoali, P.; Chanussot, J.; Zareapoor, M.; Zhou, H.; Yang, J. Multipatch feature pyramid network for weakly supervised object detection in optical remote sensing images. *IEEE Trans. Geosci. Remote Sens.* **2021**, *60*, 5610113. [CrossRef]
33. Wang, J.; Li, Z.; Long, Q.; Zhang, W.; Song, G.; Shi, C. Learning node representations from noisy graph structures. In Proceedings of the 2020 IEEE International Conference on Data Mining (ICDM), Sorrento, Italy, 17–20 November 2020; IEEE: Piscataway, NJ, USA, 2020; pp. 1310–1315.
34. Xia, G.S.; Bai, X.; Ding, J.; Zhu, Z.; Belongie, S.; Luo, J.; Datcu, M.; Pelillo, M.; Zhang, L. DOTA: A large-scale dataset for object detection in aerial images. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–26 June 2018; pp. 3974–3983.