Graph Neural Network for Accurate and Low-complexity SAR ATR

Bingyi Zhang*, Sasindu Wijeratne*, Rajgopal Kannan†, Viktor Prasanna*, Carl Busart†
*University of Southern California †DEVCOM US Army Research Lab
e-mail: *{bingyizh, kangaram, prasanna}@usc.edu †{rajgopal.kannan.civ, carl.e.busart.civ}@army.mil

***Abstract*—Synthetic Aperture Radar (SAR) Automatic Target Recognition (ATR) is the key technique for remote sensing image recognition. The state-of-the-art works exploit the deep convolutional neural networks (CNNs) for SAR ATR, leading to high computation costs. These deep CNN models are unsuitable to be deployed on resource-limited platforms. In this work, we propose a graph neural network (GNN) model to achieve accurate and low-latency SAR ATR. We transform the input SAR image into the graph representation. The proposed GNN model consists of a stack of GNN layers that operates on the input graph to perform target classification. Unlike the state-of-the-art CNNs, which need heavy convolution operations, the proposed GNN model has low computation complexity and achieves comparable high accuracy. The GNN-based approach enables our proposed *input pruning* strategy. By filtering out the irrelevant vertices in the input graph, we can reduce the computation complexity. Moreover, we propose the *model pruning* strategy to sparsify the model weight matrices which further reduces the computation complexity. We evaluate the proposed GNN model on the MSTAR dataset and ship discrimination dataset. The evaluation results show that the proposed GNN model achieves 99.38% and 99.7% classification accuracy on the above two datasets, respectively. The proposed pruning strategies can prune 98.6% input vertices and 97% weight entries with negligible accuracy loss. Compared with the state-of-the-art CNNs, the proposed GNN model has only 1/3000 computation cost and 1/80 model size.**

Keywords*—Synthetic aperture radar, automatic target recognition, graph neural network, low computation complexity, model pruning

I. Introduction

Synthetic aperture radar (SAR) is capable of high-resolution remote sensing and independent of weather conditions to observe the targets on the earth ground. SAR automatic target recognition (ATR) is the crucial technique to classify the target in the SAR images and has been used in many real-world applications, such as agriculture [1] [2], civilization [3] [4], etc. SAR devices are typically mounted on moving platforms, such as aircraft, spacecraft, and small/micro satellites [5]–[9]. These moving platforms usually have limited computation resources and power budgets (e.g., 80-180W [10]). The state-of-the-art works [11]–[15] develop complex convolutional neural networks (CNNs) for SAR ATR to achieve high classification accuracy. However, complex CNNs suffer from high computation costs and large memory footprints, making them unsuitable to be deployed on resource-limited platforms. For example, to achieve real-time image classification using CNNs, GPU is widely used. The power consumption of a state-of-the-art GPU device (e.g., NVIDIA RTX3090 has a power consumption of 450W) can exceed the power budget of the small/micro satellites.

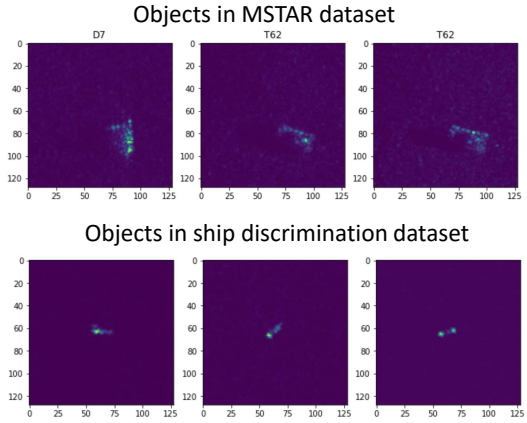

Figure 1. The objects in the SAR images

We identify that CNNs have high computation costs due to (1) heavy convolution operations and (2) CNNs do not exploit the data sparsity in SAR images because CNNs need to use the whole image as input. As shown in Figure 1, an object in a SAR image usually has a small number of pixels, and most pixels are irrelevant for classification. Recently, Graph Neural Networks (GNNs) are proposed to operate on graph data structure and have been successfully applied to many graph classification tasks [16]–[18], such as point cloud classification. [19] has proven that GNN can classify a graph based on its graph structural information and vertex features. Motivated by that, we propose to use GNN for SAR ATR. First, we extract the image pixels of the target object. We use these pixels to build a graph by constructing the edge connections among the pixels. We exploit GNN to operate on the input graph for target classifying. The proposed GNN-based approach achieves significantly less computation cost and comparable accuracy compared with state-of-the-art CNNs. Moreover, we propose attention mechanisms, including vertex attention and feature attention, to improve the model's accuracy. Our main contributions are:

- We propose a novel GNN model for SAR ATR with attention mechanisms, including vertex attention and feature attention, to achieve high accuracy with low computation complexity.
- We propose the input pruning strategy and the weight pruning strategy to further reduce the computation complexity with negligible accuracy loss.
- We perform detailed ablation studies to evaluate (1) various connectivity for constructing the input graph, (2) various types of GNN layers, (3) the effect of the attention

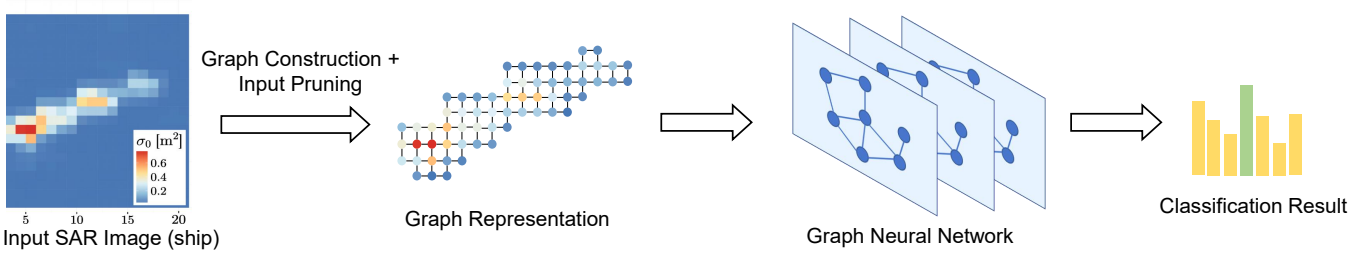

Figure 2. Overview of the proposed approach

TABLE I. NOTATIONS

Notation	Description	Notation	Description
$\mathcal{G}(\mathcal{V},\mathcal{E},\boldsymbol{X^0})$	input graph	v_i	i^{th} vertex
$\mathcal{V}$	set of vertices	e_{ij}	edge from v_i to v_j
$\mathcal{E}$	set of edges	L	number of GNN layers
$\boldsymbol{h}_i^l$	feature vector of v_i at layer l	$\mathcal{N}(i)$	neighbors of v_i

mechanism, and (4) the impact of the proposed pruning strategies.

- We evaluate the proposed approach on MSTAR and ship discrimination datasets. The evaluation results show that the proposed GNN model achieves 99.38% and 99.7% classification accuracy on the above two datasets, respectively. Compared with the state-of-the-art CNNs, the proposed GNN model has only 1/3000 computation cost and 1/80 model size.

The rest of the paper is organized as follows: Section II presents the proposed GNN model for SAR ATR; Section III describes the proposed pruning strategies for reducing computation complexity; Section IV demonstrates the evaluation results.

II. Proposed Model

Figure 2 depicts the overview of the proposed approach. In Section II-A, we introduce the basics of the graph neural network. In Section II-B, we cover the proposed graph representation for the SAR images. In Section II-C, we introduce the proposed GNN model architecture.

A. Graph Neural Network

We define GNN notations in Table I. Graph Neural Networks (GNNs) [20]–[22] are proposed for representation learning on graph $\mathcal{G}(\mathcal{V},\mathcal{E},\boldsymbol{X}^0)$. GNNs can learn from the structural information and vertex features and embed this information into low-dimension vector representation/graph embedding (For example, $\boldsymbol{h}_i^L$ is the embedding of vertex v_i). The vector representation can be used for many downstream tasks, such as node classification [21] [20], link prediction [23], graph classification [24], etc. As shown in Figure 3, GNNs follow the message-passing paradigm that vertices recursively aggregate information from the neighbors.

Input: Graph: $\mathcal{G}(\mathcal{V},\mathcal{E})$; vertex features: $\left\{\boldsymbol{h}_1^0,\boldsymbol{h}_2^0,...,\boldsymbol{h}_{|\mathcal{V}|}^0\right\}$;
Output: Output vertex features $\left\{\boldsymbol{h}_1^L,\boldsymbol{h}_2^L,...,\boldsymbol{h}_{|\mathcal{V}|}^L\right\}$;
1: **for** $l = 1...L$ **do**
2: **for** each vertex $v \in \mathcal{V}$ **do**
3: $\boldsymbol{a}_v^l = \text{Aggregate}(\boldsymbol{h}_u^{l-1} : u \in \mathcal{N}(v))$
4: $\boldsymbol{z}_v^l = \text{Update}(\boldsymbol{a}_v^l, \boldsymbol{W}^l)$, $\boldsymbol{h}_v^l = \sigma(\boldsymbol{z}_v^l)$

Figure 3. GNN Computation Abstraction

B. Graph Representation

We transform the input SAR image into a graph representation $\mathcal{G}(\mathcal{V},\mathcal{E},\boldsymbol{X^0})$, where each pixel in the SAR image is mapped to a vertex $v \in \mathcal{V}$ in the graph. The SAR signal value of the pixel becomes the feature of the vertex. Each pixel is connected to its neighbors as the edge connections $\mathcal{E}$. As shown in Figure 4, we propose the following two ways of connecting a pixel to its neighbors and evaluate them in the experiments:

- **4-connectivity**: Each pixel is connected to the four neighbors: up ($p2$), down ($p8$), left ($p4$), and right ($p6$).
- **8-connectivity**: Each pixel is connected to the eight neighbors: $p1$, $p2$, $p3$, $p4$, $p6$, $p7$, $p8$, $p9$.

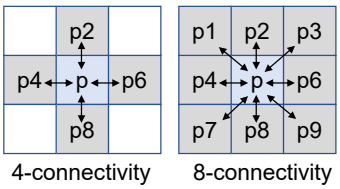

Figure 4. Two types of connectivity for constructing input graph

C. Model Architecture

The proposed model architecture is shown in Figure 5, which consists of a stack of layers, including Graph Neural Network layers, graph pooling layers, and attention layers. The final Multi-layer Perceptron (MLP) generates the classification result. For simplicity, $v_{i,j}$ denotes the vertex/pixel that locates at i^{th} row and j^{th} column in original SAR image. The input to layer l $(1 \leqslant l \leqslant L)$ is the vertex feature vectors $\{\boldsymbol{h}_{i,j}^{l-1} : v_{i,j} \in \mathcal{V}_{l-1}\}$ and edges $\{e : e \in \mathcal{E}_{l-1}\}$ that defines

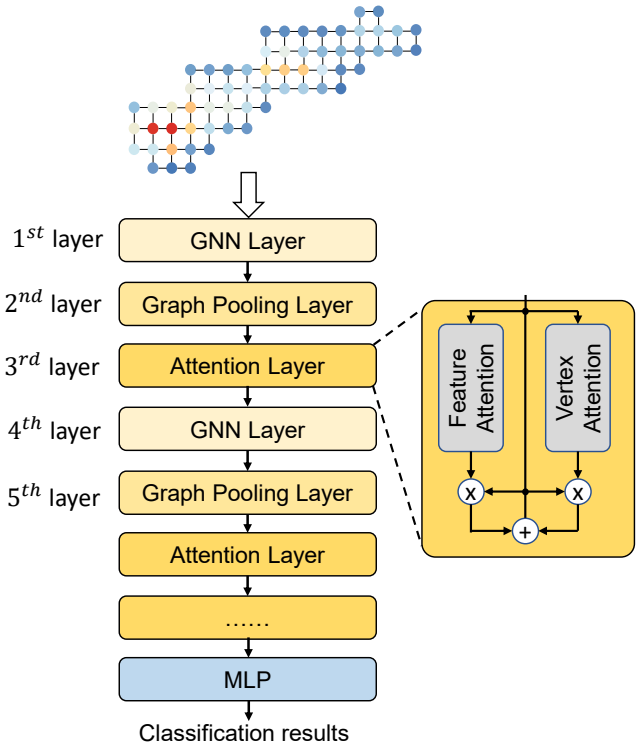

Figure 5. Diagram of model architecture

the connectivity of the vertices in $\mathcal{V}_{l-1}$. The output of layer l is the vertex feature vectors $\{\boldsymbol{h}_{i,j}^{l} : v_{i,j} \in \mathcal{V}_l\}$.

Graph neural network (GNN) layer: A GNN layer follows the *Aggregate-Update* paradigm as shown in Algorithm 3. Using the Aggregate() function, each vertex aggregates the feature vectors from the neighbors (line 3 of Algorithm 3). Then, each feature vector is updated by the Update() function to generate the updated feature vector (line 4 of Algorithm 3). There are some representative Graph Neural Network layers, such as GCN [20], GraphSAGE [21], GIN [19], and SGC [25].

Graph pooling layer: It downscales the input graph $\mathcal{V}_{l-1}$ into a smaller output graph $\mathcal{V}_l$. The pooling operaton is similar to the pooling in the 2-D images:

$$\boldsymbol{h}_{i,j}^{l} = \max(\boldsymbol{h}_{2i,2j}^{l-1}, \boldsymbol{h}_{2i+1,2j}^{l-1}, \boldsymbol{h}_{2i,2j+1}^{l-1}, \boldsymbol{h}_{2i+1,2j+1}^{l-1}) \quad (1)$$

where $v_{i,j}^{l} \in \mathcal{V}_l$, and $v_{2i,2j}^{l-1}, v_{2i+1,2j}^{l-1}, v_{2i,2j+1}^{l-1}, v_{2i+1,2j+1}^{l-1} \in \mathcal{V}_{l-1}$.

Attention layer: We exploit the attention mechanism to improve the accuracy. The attention layer consists of *feature attention* that calculates the attention scores for each vertex feature, and *vertex attention* that calculates the attention scores for each vertex. The feature attention is calculated by:

$$\begin{aligned}\boldsymbol{F}_{\text{fa}} = \text{sigmoid}(\text{mean}(\{\boldsymbol{h}_{i,j} : v_{i,j} \in \mathcal{V}\})\boldsymbol{W}_{\text{fa}}^{\text{mean}} + \\ \text{sum}(\{\boldsymbol{h}_{i,j} : v_{i,j} \in \mathcal{V}\})\boldsymbol{W}_{\text{fa}}^{\text{sum}})\end{aligned} \quad (2)$$

where $\boldsymbol{h}_{i,j}, \boldsymbol{F}_{\text{fa}} \in \mathbb{R}^c$, $\boldsymbol{W}_{\text{fa}}^{\text{mean}}, \boldsymbol{W}_{\text{fa}}^{\text{sum}} \in \mathbb{R}^{c \times c}$, and c denotes the length of feature vector. $\text{fa}[i]$ is the attention score for i^{th} feature. The vertex attention score is calculated using a GNN layer:

$$\{\alpha_{i,j} : v_{i,j} \in \mathcal{V}_l\} = \text{sigmoid}(\text{GNNL}(\{\boldsymbol{h}_{i,j} : v_{i,j} \in \mathcal{V}_{l-1}\})), \quad (3)$$

Where $\alpha_{i,j}$ is the attention score for vertex $v_{i,j}$. Then, the output of the attention layer is calculated by:

$$\{\boldsymbol{h}_{i,j}^{\text{out}} : \boldsymbol{h}_{i,j}^{\text{out}} = (1 + \alpha_{i,j})\boldsymbol{h}_{i,j}^{\text{in}} + \boldsymbol{h}_{i,j}^{\text{in}} \otimes \boldsymbol{F}_{\text{fa}}\} \quad (4)$$

where $\otimes$ is element-wise multiplication.

Multi-layer Perceptron (MLP): After a sequence of layers, all the feature vectors are flattened into a single vector, which is sent to the MLP for classification. MLP has a stack of fully connected (FC) layers.

III. Pruning

This section covers the proposed pruning techniques, including, input pruning (Section III-A), and weight pruning (Section III-B).

A. Input Pruning

The key benefit of using GNN is that GNN is flexible in accepting any graph structure as the input. Thereby, we are able to exploit input pruning to reduce the computation complexity. Theoretically, in a SAR image (See Figure 1), the pixels not in the target do not affect the classification results. As studied in [26], by properly setting up a constant threshold I_v, we can filter out most irrelevant pixels since the pixels that do not belong to the target usually have negligible SAR signal magnitude. After constructing the input graph from the SAR image, we prune the vertices that have a magnitude smaller than I_v. The magnitude of a vertex is calculated by $\sqrt{x_1^2 + x_2^2 + ... + x_{np}^2}$ where np denotes the number of polarization of the SAR signal. For example, a quad-polarization system has four kinds of polarization – horizontal-horizontal (HH), vertical-vertical (VV), horizontal-vertical (HV), and vertical-horizontal (VH). After pruning the vertices, all the edges connected to the pruned vertices are also pruned. Due to the input pruning, the graph pooling operation (Equation 1) is slightly modified:

$$\begin{aligned}\boldsymbol{h}_{i,j}^{l} = \max(\mathbb{1}_{2i,2j}^{l-1} \cdot \boldsymbol{h}_{2i,2j}^{l-1}, \mathbb{1}_{2i+1,2j}^{l-1} \cdot \boldsymbol{h}_{2i+1,2j}^{l-1}, \\ \mathbb{1}_{2i,2j+1}^{l-1} \cdot \boldsymbol{h}_{2i,2j+1}^{l-1}, \mathbb{1}_{2i+1,2j+1}^{l-1} \cdot \boldsymbol{h}_{2i+1,2j+1}^{l-1}),\end{aligned} \quad (5)$$

where $\mathbb{1}_{i,j} \in \{0, 1\}$ is the indicator that indicates the existence of vertex $v_{i,j}$. After input pruning, we can skip the computation for the pruned vertices, which greatly reduces the total computation complexity.

B. Weight Pruning

As analyzed in [27], [28], the weight matrices in GNNs have redundancy, and some weight entries can be pruned without affecting the classification accuracy. Therefore, to reduce the total computation complexity, we perform weight pruning by training the model using lasso regression [29]. We add the L1 penalty to the loss function:

$$\text{loss} = l(y, y') + \lambda \sum_{w}^{W} |w| \quad (6)$$

where $l(y, y')$ is the classification loss, and $\lambda \sum_w^W |w|$ is the L1 penalty term parameterized by λ. The L1 penalty leads to weight shrinkage during training. Thereby, some model weights become zeros and can be eliminated from the model. After training, we set a threshold I_w, and the model weights with absolute values smaller than I_w are pruned.

IV. Evaluation

We evaluate our approach on two widely used datasets:

- **MSTAR**: The setting of the MSTAR dataset follows the state-of-the-art work [11] [14] [15] [12]. MSTAR contains the SAR images of ten classes of ground vehicles, with 2747 images in the training set and 2427 images in the testing set.
- **Ship discrimination [30]:** For the ship discrimination dataset, we follow the setting in [31], which is a binary classification task that identifies if a given SAR image has a ship or not. The dataset contains 1596 positive image samples and 1596 negative image samples.

A. Evaluation on MSTAR Dataset

1) Experimental Setting: For the MSTAR dataset, we use the following setting. The proposed model consists of 12 layers. We develop the proposed model using Pytorch Geometric. We use the cross-entropy loss as the classification loss (Equation 6). We train the model using the Adam optimization algorithm. The training batch size is set as 20, and the initial learning rate is 0.02. λ (for lasso regression) is set as 0.002. The L2 weight decay is set as 0.08. We train the model for 150 epochs, and the learning rate is multiplied by 0.5 for every 10 epoch. We use the 8-connectivity to build the input graph. We evaluate the three widely used GNN layers in the proposed model – GCN layer [20], GraphSAGE layer [21], and GAT [22]. We train the proposed model using one NVIDIA RTX A6000 GPU.

Performance metrics: We evaluate the proposed approach using the following metrics: *classification accuracy*, *computation complexity*, and *number of parameters*.

TABLE II. THE ACCURACY ON MSTAR DATASET

GNN Layer Type	Connectivity	Training Accuracy	Testing Accuracy	Training Time
GCN	4	99.16%	90.06%	3.0 hours
	8	95.44%	83.82%	4.0 hours
GAT	4	99.53%	92.21%	1.8 hours
	8	82.71%	71.33%	1.9 hours
GraphSAGE	4	100.00%	97.81%	52 min
	8	100.00%	99.38%	55 min

2) Classification Accuracy: The accuracy of the proposed model (under various GNN layer types and connectivity) is shown in Table II. We observe that using the GraphSAGE layer as the GNN layer leads to the highest training/testing accuracy. Using the GraphSAGE layer also leads to the lowest training time. For the GraphSAGE layer, using 8-connectivity to build the input graph can result in higher accuracy but slightly higher training time than 4-connectivity. Table III shows that the proposed GNN model achieves higher accuracy compared with the state-of-the-art CNNs [11], [12], [14], [15] with negligible computation complexity for inference.

TABLE III. COMPARISON WITH THE STATE-OF-THE-ART CNNS ON MSTAR DATASET

	Type	Accuracy	# of FLOPs	# of Para.
[11]	CNN	92.3%	$\frac{1}{12}\times$	0.5×10^6
[14]	CNN	97.97%	$\frac{1}{10}\times$	0.65×10^6
[15]	CNN	98.52%	$\frac{1}{3}\times$	2.1×10^6
[12]	CNN	99.3%	$1\times$ (6.94 GFLOPs)	2.5×10^6
This work [after pruning] (GraphSAGE layer, 8-connectivity)	GNN	99.1%	$\frac{1}{3000}\times$	0.03×10^6

TABLE IV. THE IMPACT OF THE ATTENTION MECHANISM (USING GRAPHSAGE LAYER AND 8-CONNECTIVITY)

Vertex Attention	Feature Attention	Training Accuracy	Testing Accuracy	Training Time
✗	✗	99.67%	93.77%	31 min
✗	✓	100.0%	98.51%	40 min
✓	✗	100.0%	99.26%	41 min
✓	✓	100.0%	99.38%	55 min

3) Ablation Study: We perform an ablation study to evaluate the impact of the attention mechanism (using GraphSAGE layer and 8-connectivity). The result is shown in Table IV. Without vertex and feature attention, the model achieves only 93.77% accuracy. With only vertex attention, the model achieves 99.26% accuracy. With only feature attention, the model achieves 98.51% accuracy. With both vertex and feature attention, the model achieves 99.38% accuracy. The evaluation result demonstrates that the attention mechanism can improve classification accuracy without significantly increasing computation complexity.

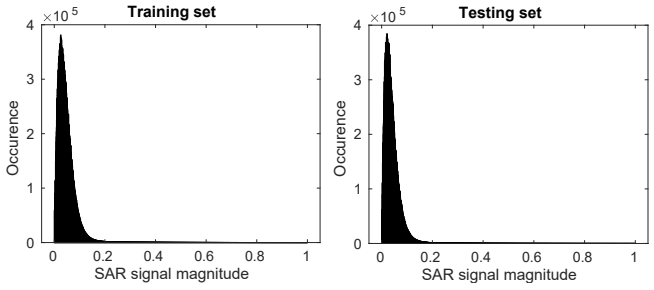

Figure 6. The distribution of the SAR signal magnitude in the training/testing set of MASTAR

4) Evaluation on the Pruning Strategy: We evaluate the proposed input pruning and weight pruning strategies. We use GraphSAGE layer and 8-connectivity as the setting of the model.

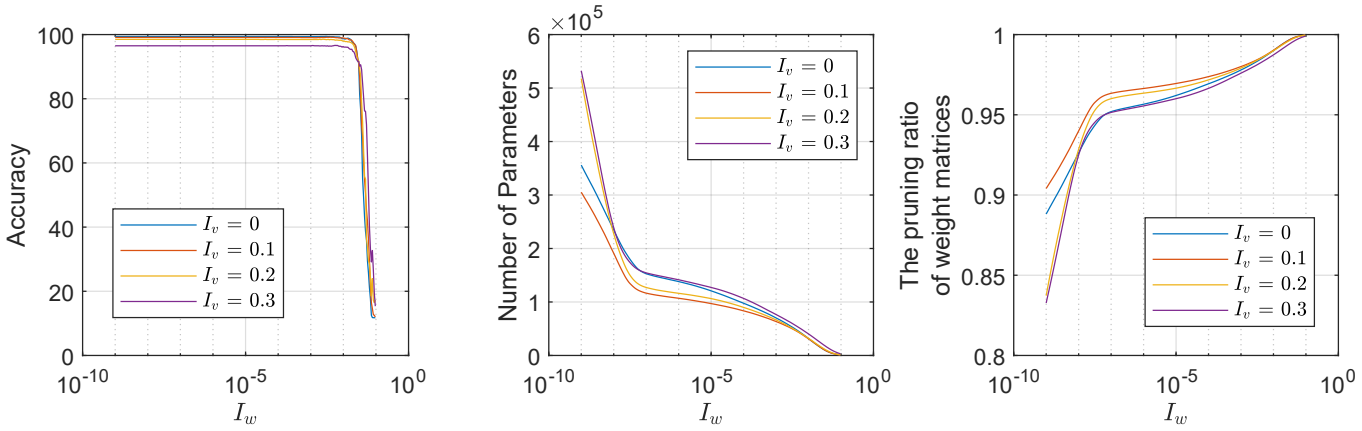

Figure 7. Evaluation of proposed pruning strategy

Input Pruning: Figure 6 shows the data distribution of the SAR signal magnitude of the image pixels in the training/testing set. The SAR signal magnitude ranges from 0 to 16. Since most pixels have a magnitude between 0-1, Figure 6 only shows the range 0-1. For experiment, we set the pruning threshold I_v (See Section III-A) to be 0, 0.1, 0.2, 0.3 respectively. The image pixels that have a magnitude small than I_v are pruned.

Weight Pruning: The weights in weight matrices can be either negative or positive. We set the threshold I_w for weight pruning (See Section III-B). The weights that have an absolute value that is smaller than I_w are pruned. In the experiment, we set I_w to be between 1×10^9 and 1×10^1.

The evaluation results for the pruning strategy are shown in Figure 7. We have the following observations:

- Without weight pruning, when I_v = 0.1, 93.4% input vertices/pixels are pruned, the accuracy is dropped to 99.1%; when I_v = 0.2, 98.6% input vertices/pixels are pruned, the accuracy is dropped to 98.5%; when I_v = 0.3, 99.1% input vertices/pixels are pruned, the accuracy is dropped to 96.5%.
- When weight pruning threshold $I_w < 10^7$, the accuracy does not change w.r.t. to I_w. When $I_w = 10^7$, more than 95% weights are pruned. Therefore, most entries in the weight matrices are redundant.

Therefore, by setting proper threshold I_v, I_w for input pruning and weight pruning, most input pixels and weights can be pruned without significantly dropping the accuracy. Figure 7 shows the evaluation results for the pruning strategy, 97% weight entries are pruned, and the accuracy is 99.1%. By skipping the computation for the pruned vertices and weights, we can dramatically reduce the total computation complexity.

5) Experimental Setting: For ship discrimination dataset, we follow the setting of [31] to conduct experiment for few-shot learning. Since the ship discrimination is a binary class task, the few-shot learning task can be formed as a 2-way-K-shot-classification problem, where $K = \{1, 2, .., 10\}$ denotes the number of labeled training images for each class. We train the model using the Adam optimization algorithm. The training batch size is set as $\frac{K}{2}$, and the learning rate is set as $0.001 * K$. The L2 weight decay is set as 0.08.

6) Classification Accuracy: As shown in Figure 8, we compare our accuracy with [31] (baseline) for the few-shot learning on the ship discrimination dataset. Note that the baseline [31] uses a convolutional neural network (CNN), and the authors pretrained their CNN using the ship discrimination dataset on the Electro-Optical (EO) domain. We do not pretrain our network on any dataset. For various K, the proposed model outperforms the baseline [31], which is a pretrained deep CNN model.

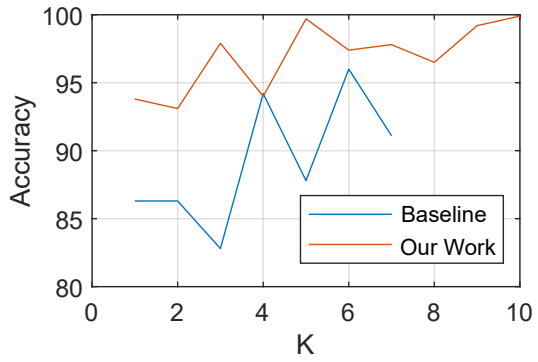

Figure 8. The accuracy on the ship discrimination dataset

TABLE V. COMPARISON OF ACCURACY (%)

K	1	2	3	4	5	6	7
Baseline [31]	86.3	86.3	82.8	94.2	87.8	96.0	91.1
Our work	93.8	93.1	97.9	94.0	99.7	97.4	97.8

V. Conclusion and Future Work

In this paper, we proposed a novel GNN-based approach for SAR automatic target recognition. The proposed approach uses the GNN layer as the backbone and uses the attention mechanism to improve classification accuracy. We proposed

pruning strategies, including input pruning and weight pruning, to reduce the computation complexity. The evaluation results on the MSTAR and ship discrimination datasets show that the proposed model outperforms the state-of-the-art CNNs in classification accuracy and computation complexity. In [32], we designed a hardware accelerator for the proposed GNN model. In the future, we plan to extend the proposed GNN model to more SAR-related tasks, such as object detection.

ACKNOWLEDGMENT

This work is supported by the National Science Foundation (NSF) under grants CCF-1919289 and OAC-2209563, and the DEVCOM Army Research Lab (ARL) under grant W911NF2220159.

REFERENCES

[1] L. Landuyt, A. Van Wesemael *et al.*, "Flood mapping based on synthetic aperture radar: An assessment of established approaches," *IEEE Transactions on Geoscience and Remote Sensing*, 2018.

[2] P. Zhan, W. Zhu, and N. Li, "An automated rice mapping method based on flooding signals in synthetic aperture radar time series," *Remote Sensing of Environment*, vol. 252, p. 112112, 2021.

[3] N. Li, Z. Guo *et al.*, "Characterizing ancient channel of the yellow river from spaceborne sar: Case study of chinese gaofen-3 satellite," *IEEE Geoscience and Remote Sensing Letters*, vol. 19, pp. 1–5, 2021.

[4] T. Zhang, X. Zhang *et al.*, "Hyperli-net: A hyper-light deep learning network for high-accurate and high-speed ship detection from synthetic aperture radar imagery," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 167, pp. 123–153, 2020.

[5] F. Bardi, W. Frodella *et al.*, "Integration between ground based and satellite sar data in landslide mapping: The san fratello case study," *Geomorphology*, vol. 223, pp. 45–60, 2014.

[6] H. Septanto and O. Sudjana, "Simulation-based energy balance analysis of sar micro-satellite," in *Sixth International Symposium on LAPAN-IPB Satellite*, vol. 11372. International Society for Optics and Photonics, 2019, p. 113721W.

[7] Y. Yokota, Y. Okada *et al.*, "Newly developed x-band sar system onboard japanese small satellite "asnaro-2"," in *Conference Proceedings of 2013 Asia-Pacific Conference on Synthetic Aperture Radar (APSAR)*. IEEE, 2013, pp. 81–83.

[8] P. R. Akbar, H. Saito *et al.*, "Parallel-plate slot array antenna for deployable sar antenna onboard small satellite," *IEEE Transactions on Antennas and Propagation*, 2016.

[9] K. Tanaka, H. Saito *et al.*, "Development of 1kw high power x-band sar installed on small satellite for on-demand observation," in *Proceedings of the International Astronautical Congress, IAC*, 2018.

[10] M. Tsamsakizoglou, H. Löfgren, and M. Gunnarsson, "Microsatellite power control and distribution unit for the innosat platform," in *E3S Web of Conferences*, vol. 16. EDP Sciences, 2017, p. 18007.

[11] M. Zhang, J. An *et al.*, "Convolutional neural network with attention mechanism for sar automatic target recognition," *IEEE Geoscience and Remote Sensing Letters*, 2020.

[12] D. A. Morgan, "Deep convolutional neural networks for atr from sar imagery," in *Algorithms for Synthetic Aperture Radar Imagery XXII*, vol. 9475. SPIE, 2015, pp. 116–128.

[13] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 7132–7141.

[14] J. Pei, Y. Huang *et al.*, "Sar automatic target recognition based on multiview deep learning framework," *IEEE Transactions on Geoscience and Remote Sensing*, 2017.

[15] Z. Ying, C. Xuan *et al.*, "Tai-sarnet: Deep transferred atrous-inception cnn for small samples sar atr," *Sensors*, vol. 20, no. 6, p. 1724, 2020.

[16] H. Zhu, X. Du, and Y. Yao, "Convsppis: identifying protein-protein interaction sites by an ensemble convolutional neural network with feature graph," *Current Bioinformatics*, 2020.

[17] T. Zhao, Y. Hu *et al.*, "Identifying drug–target interactions based on graph convolutional network and deep neural network," *Briefings in bioinformatics*, 2021.

[18] C. R. Qi, L. Yi *et al.*, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," *Advances in neural information processing systems*, vol. 30, 2017.

[19] K. Xu, W. Hu *et al.*, "How powerful are graph neural networks?" *arXiv preprint arXiv:1810.00826*, 2018.

[20] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[21] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2017.

[22] P. Veličković, G. Cucurull *et al.*, "Graph attention networks," *arXiv preprint arXiv:1710.10903*, 2017.

[23] M. Zhang and Y. Chen, "Link prediction based on graph neural networks," *Advances in Neural Information Processing Systems*, 2018.

[24] R. Ying, J. You *et al.*, "Hierarchical graph representation learning with differentiable pooling," *arXiv preprint arXiv:1806.08804*, 2018.

[25] F. Wu and A. Souza, "Simplifying graph convolutional networks," in *International conference on machine learning*. PMLR, 2019.

[26] H. Zhu, N. Lin *et al.*, "Target classification from sar imagery based on the pixel grayscale decline by graph convolutional neural network," *IEEE Sensors Letters*, vol. 4, no. 6, pp. 1–4, 2020.

[27] M. Rahman, A. Azad *et al.*, "Triple sparsification of graph convolutional networks without sacrificing the accuracy," *arXiv preprint arXiv:2208.03559*, 2022.

[28] H. Zhou, A. Srivastava *et al.*, "Accelerating large scale real-time gnn inference using channel pruning," *arXiv preprint arXiv:2105.04528*, 2021.

[29] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 58, no. 1, pp. 267–288, 1996.

[30] C. P. Schwegmann, W. Kleynhans *et al.*, "Very deep learning for ship discrimination in synthetic aperture radar imagery," in *2016 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*. IEEE, 2016, pp. 104–107.

[31] M. Rostami, S. Kolouri *et al.*, "Deep transfer learning for few-shot sar image classification," *Remote Sensing*, vol. 11, no. 11, p. 1374, 2019.

[32] B. Zhang, R. Kannan *et al.*, "Accurate, low-latency, efficient sar automatic target recognition on fpga," in *2022 32nd International Conference on Field-Programmable Logic and Applications (FPL)*, 2022, pp. 1–8.