Graph Transfer Learning-Based Attack Detection in Cyber-Physical Water Distribution Systems

Md Rakibul Ahasan*, Faaiz Joad*, Rachad Atat†, Coleman Thompson*, Erchin Serpedin‡, and Abdulrahman Takiddin*

*Department of Electrical and Computer Engineering, Florida State University, Tallahassee, FL, USA
†Computer Science and Mathematics Department, Lebanese American University, Beirut, Lebanon
‡Department of Electrical and Computer Engineering, Texas A&M University, College Station, TX, USA

***Abstract*—Water distribution systems (WDSs) are critical infrastructures that rely on digital monitoring and control through a cyber layer, which makes them vulnerable to cybersecurity threats. Cyberattacks, such as denial of service and replay attacks, can manipulate sensor readings and disrupt normal operations, leading to system failures. Machine learning-based solutions, including graph neural networks (GNNs), have been adopted to detect cyberattacks in WDSs. However, GNNs are computationally expensive to train due to the large size of the input graphs. In this paper, we propose a transfer learning (TL) technique based on a spatio-temporal graph transformer (STGT) model (TL-STGT) for attack detection that significantly reduces training time for larger WDS graphs by $66\% - 89\%$ compared to benchmark models. The proposed TL-STGT model improves cyberattack detection performance by up to 57%, 45%, and 20% in F1-score compared to shallow, deep, and graph-based models, respectively.**

***Index Terms*—Cyberattacks, graph neural network, machine learning, transfer learning, transformer, water distribution systems.**

I. Introduction

Water distribution systems (WDSs) are critical infrastructures that ensure clean water availability [1]. Supervisory control and data acquisition (SCADA) systems monitor WDSs by collecting real-time data from pressure sensors, flow meters, and valves [2]. However, digitization introduces cybersecurity risks, including denial of service (DoS) and replay attacks, which manipulate sensor data and disrupt communications [3]. Such attacks lead to incorrect control decisions, causing overflows, inefficiencies, or failures. WDSs, as cyber-physical systems [4], depend on continuous sensor data exchange, making communication crucial for functionality and security. In 2021, a hacker remotely accessed a water treatment plant control system in Florida, USA and manipulated the sodium hydroxide to toxic levels [5]. Hence, the complexity and reliance of WDSs on digital communication create vulnerabilities, reflecting the need for robust attack detectors.

A. Related Works

In the context of WDS, various machine learning approaches have been proposed for cyberattack detection, including shallow, deep, and graph-based techniques as follows.

Shallow models like light gradient-boosting machine (LGBM), linear discriminant analysis, and support vector machine (SVM) achieved relatively low F1-scores of 10%, 57%, and 63%, respectively [6], [7]. Deep learning models outperformed shallow ones. For instance, a deep autoencoder (AE) offered an F1-score of 72% [7]. Deep recurrent models like, gated recurrent unit (GRU), long short term memory (LSTM), and basic recurrent neural network yielded F1-scores of 60%, 72%, and 77%, respectively, against the battle of the attack detection algorithms (BATADAL) dataset [8].

Both shallow and deep learning models struggle to capture the complex patterns and spatial aspects necessary for effective detection of cyberattacks, resulting in sub-optimal performance. Therefore, graph neural network (GNN)-based detectors have emerged as a promising alternative. For example, temporal graph convolutional network (TGCN) and graph deviation network-based approaches achieved F1-scores of 75% and 81%, respectively [9] [10]. Despite their superior performance, GNNs face high computational complexity in large-scale systems [11].

Spatio-temporal features combine spatial information (i.e., system physical layout and connectivity) with temporal data (i.e., sensor readings over time). Shallow models struggle with spatio-temporal complexity, while deep models demand high computational resources. GNNs capture spatial relationships, but become computationally expensive for large WDSs [12]. To address such challenges, enhancing efficiency is crucial, which will also improve scalability [13], enabling deployment on larger graphs without performance trade-offs.

B. Contributions

To address the aforementioned limitations, we propose a transfer learning (TL)-based spatio-temporal graph transformer (STGT) called (TL-STGT) for attack detection. While enhancing cyberattack detection performance, our proposed TL-STGT-based approach efficiently models spatio-temporal patterns while reducing computational complexity by transferring learned knowledge from smaller to larger graph representations, achieving faster training without compromising detection performance. Our key contributions include:

- We propose a TL-based STGT (TL-STGT) model for cyberattack detection in WDSs, achieving F1-score improvements of $35-57\%$, $29-45\%$, and $5-20\%$ over shallow, deep, and graph-based benchmarks.
- We introduce a TL approach leveraging betweenness centrality to abstract key nodes. This method accelerates training across various WDS graph representation scales,

ISBN: 978-9-46-459362-4

reducing training time by $66\%-89\%$ compared to benchmarks.
- We exhibit scalability in the proposed TL-STGT model as its F1-score improves by $15-25\%$ when detecting attacks in larger WDSs compared to smaller ones.

The paper is structured as follows. Section II covers the data preparation. Section III describes the STGT block and Section IV details the TL approach of the proposed TL-STGT model. Section V presents our results. Section VI concludes the paper.

II. Data Preparation

For realistic results, we create a spatio-temporal WDS dataset with benign and attack samples. Specifically, we adopt the C-Town WDS benchmark dataset [14] from BATADAL [15], which is widely used for evaluating cyber-attack detection in WDSs. Our dataset consists of spatial and temporal aspects as follows.

A. Spatial Aspect

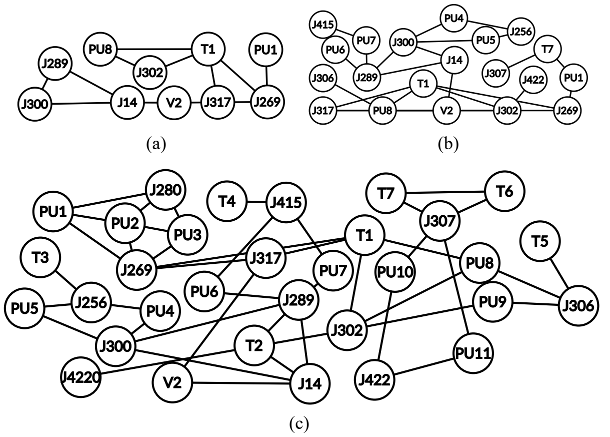
The spatial aspects represents nodes (i.e., WDS components including tanks, pumps, valves, and junctions) and edges (i.e., connectivity of nodes via water pipes). We model our WDS using three graphs, with 10, 20, and 31 nodes. Next, we discuss the node selection and graph modeling processes.

1) Node Selection: As part of our TL approach for cyber-attack detection in WDS (see Section IV), we start with the original BATADAL 31-node WDS as our initial system with the largest size. Then, we apply the node selection algorithm to reduce the input graph size and mitigate the computational cost of training large graphs, where GNNs require $O(N^2)$ matrix multiplications [16]. Graph reduction is based on betweenness centrality, which quantifies node importance by measuring the number of shortest paths passing through a node [17]. The betweenness centrality $C_B(v)$ of node v is given by:

$$C_B(v) = \sum_{i \neq v \neq j} \frac{\eta_{ij}(v)}{\eta_{ij}} \tag{1}$$

where $\eta_{ij}(v)$ is the number of shortest paths from node i to j passing through v, and η_{ij} is the total shortest paths between nodes i and j. Using $C_B(v)$, we construct small (10-node) and medium (20-node) graphs systems out of initial large (31-node) graph. Then, for training, we first train the model on the constructed 10-node graph, save the learned weights, and then fine-tune models on larger graphs, enabling faster training without compromising detection performance. These variations assess computational efficiency and detection performance as graph complexity increases. Fig. 1 illustrates the reduced graphs where T, PU, V, and J denote tanks, pumps, valves, and junctions, respectively.

2) WDS Graph Modeling: We model each of the 10, 20, and 31-node systems as a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where nodes $\mathcal{V}$ represent WDS components and edges $\mathcal{E}$ represent connection pipes. The dataset is structured as a time-series matrix $\mathbf{X} \in \mathbb{R}^{|T| \times |\mathcal{V}|}$, where $\mathbf{X}_{t,i}$ denotes a reading at node i at time t, classified as either benign $\mathbf{X}^b_{t,i}$ or malicious $\mathbf{X}^m_{t,i}$. The spatial structure is captured by the adjacency matrix $\mathbf{A} \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}|}$, where $\mathbf{A}_{ij} = 1$ if nodes i and j are directly connected, and 0 otherwise.



(a)

(b)

(c)

Fig. 1. Reduced C-Town WDS graphs with (a) 10, (b) 20, and (c) 31 nodes.

B. Temporal Aspect

The temporal aspect represents sensor readings (i.e., tank water level, flow rates, and pressure level) from the three WDSs. We generate such readings using EpanetCPA [18] with $1,400$ hours of data evenly split between normal and attack operations, including replay, DoS, and data manipulation attacks for each of the three WDSs. The datasets are split into training, validation, and testing sets using an $80\%-10\%-10\%$ ratio with equal number of samples per class in each set.

1) Benign Samples: During normal operation, sensor readings are recorded over time without intervention. Such readings represent benign samples denoted as $\mathbf{X}^b \in \mathbf{X}$, where $\mathbf{X}^b_{t,i}$ represents a benign sample at time step t and node i.

2) Attack Samples: A malicious sample $\mathbf{X}^m_{t,i}$ is generated by modifying benign samples using attack functions as follows.

a) Replay Attack: This attack replaces a current reading $\mathbf{X}^b_{t,i}$ with a past value from a previous time step $t - \Delta t$, i.e.,

$$\mathbf{X}^m_{t,i} = \mathbf{X}^b_{t-\Delta t,i}, \tag{2}$$

where Δt is the time difference between the current and replayed time step.

b) DoS Attack: This attack disrupts communication between sensors and SCADA, freezing updates. The last valid reading before disruption freezes during the attack, i.e.,

$$\mathbf{X}^m_{t,i} = \mathbf{X}^b_{t-1,i}, \tag{3}$$

where $t-1$ is the last valid time step before the attack.

c) Data Manipulation Attack: This attack alters sensor readings via a perturbation value of $-5 \leq \delta_{t,i} \leq 5$ with 0.2 increments, selected based on experimental tuning, resulting in

$$\mathbf{X}^m_{t,i} = \mathbf{X}^b_{t,i} + \delta_{t,i}. \tag{4}$$

III. Model Architecture

The STGT block of the proposed TL-STGT model integrates TGCNs, GRUs, and transformers to capture spatio-temporal dependencies effectively. A high-level overview is shown in Fig. 2. Next, we present the STGT components.

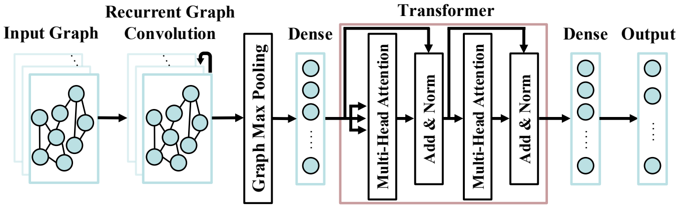


Fig. 2. Architecture of the proposed STGT model.

A. Graph Convolution

To capture spatial aspects, we apply graph convolution on graph $\mathcal{G}$ for node i as:

$$h_i^{l+1} = \tanh\left(\sum_{j\in\mathcal{N}(i)} \frac{1}{\sqrt{d_i d_j}} \mathbf{A}_{ij} \mathbf{W}^{l_g} h_j^l\right), \tag{5}$$

where h_i^l is the hidden state at layer l, $\mathcal{N}(i)$ denotes neighbors, $\mathbf{A}_{ij}$ is the adjacency matrix, d_i, d_j are node degrees, and $\mathbf{W}^{l_g}$ is a trainable weight matrix.

B. Gated Recurrent Unit

We add a GRU layer as it models temporal dependencies more efficiently compared to LSTMs [19]. The output of the graph convolution h_i^{l+1} is then passed to the GRU layer where:

$$r_t = \sigma(\mathbf{W}_r \cdot [h_{t-1}, x_t]) \tag{6}$$

$$u_t = \sigma(\mathbf{W}_u \cdot [h_{t-1}, x_t]) \tag{7}$$

$$c_t = \tanh(\mathbf{W}_c \cdot [r_t * h_{t-1}, x_t]) \tag{8}$$

$$h_t = u_t * h_{t-1} + (1 - u_t) * c_t \tag{9}$$

where r_t, u_t, and c_t are the reset, update, and candidate gates, respectively. $\mathbf{W}_r$, $\mathbf{W}_u$, and $\mathbf{W}_c$ are the trainable weight matrices. x_t and h_t denote the input features and hidden state, respectively, at time t. $*$ denote the Hadamard multiplication.

C. Transformer

After the GRU step, we obtain the hidden states h_t that contain the spatio-temporal features. We then use transformers to capture the longer range dependencies that may not be captured just from the graph convolution or GRU layers alone [20]. The hidden state h_t from the GRU output is then passed through a global max pooling layer and a dense layer to create a fixed sized input into the transformer. The core of the transformer is multi-head attention, defined as follows:

$$\text{Attention}(Q, P, F) = \text{softmax}\left(\frac{QP^\intercal}{\sqrt{\hat{d}_p}}\right) F, \tag{10}$$

$$\text{MultiHead}(Q, P, F) = \text{Concat}(\text{head}_1, \ldots, \text{head}_n)\mathbf{W}_o, \tag{11}$$

where $Q, P,$ and F are the query, key, and value matricies respectively. $\hat{d}_p$ is the key vector dimension, and $\mathbf{W}_o$ is the output weight matrix. Multi-head attention allows the model to focus on different parts of the input sequence simultaneously to learn complex patterns and capture relationships efficiently through parallel attention mechanisms [21]. The output is then passed through a final dense layer, representing the reconstructed vector of the next time step.

IV. Transfer Learning

We propose a progressive TL approach (illustrated in Fig. 3) that learns spatio-temporal features on the smallest 10-node graph, leveraging its trained weights for subsequent larger graphs, ensuring faster convergence and improved efficiency.

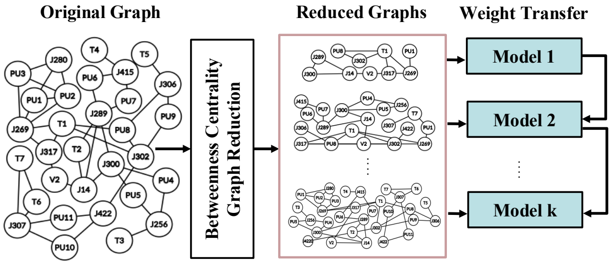


Fig. 3. Overview of the proposed TL approach.

The training of the proposed model is carried out in a supervised manner using the mean squared error loss function:

$$\mathcal{L}(Y, \hat{Y}) = \frac{1}{Z}\sum_z \left\| Y[z] - \hat{Y}[z] \right\|^2, \tag{12}$$

where Z is the total number of batches. Y and $\hat{Y}$ correspond to the actual and predicted sequences, respectively. Training is performed end-to-end, allowing the model to learn the temporal and spatial dependencies directly from the input sequences. The model training is based on the Adam optimizer. The training dataset is divided into mini-batches, which are fed into the model over several epochs to minimize the loss function. The input data consists of the time-series dataset $\mathbf{X} \in \mathbb{R}^{Z\times S\times|\mathcal{V}|}$ where S is the batch size, and $|\mathcal{V}|$ is the number of nodes. The model also has the spatial representation as the adjacency matrix $\mathbf{A} \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$. The model inference output is a prediction of all node features for a given timestamp. After training, we save the trained weights and use them to train another model with the next largest graph input. We continue this iterative process to train models for the larger graphs.

As shown in Algorithm 1, $\boldsymbol{\mathcal{G}} = \{\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_k\}$ represents the set of all graphs where $\mathcal{G}_k$ denotes the last graph. $\boldsymbol{\mathcal{X}} = \{\mathbf{X}_{\mathcal{G}_1}, \mathbf{X}_{\mathcal{G}_2}, \ldots, \mathbf{X}_{\mathcal{G}_k}\}$ represents the training data where $\mathbf{X}_{\mathcal{G}_k}$ denotes the training data of the last graph $\mathcal{G}_k$. $\mathcal{W}_0$ denotes the initial random model weights, $\mathcal{W}_{\mathcal{G}_g}$ denotes the set of model parameters (weights) after training on a graph $\mathcal{G}_g$ with corresponding training data $\mathbf{X}_{\mathcal{G}_g}$. The model is trained on the input data $\mathbf{X}_{\mathcal{G}_g}$ associated with graph $\mathcal{G}_g$, while updating the parameters through optimizing $\mathcal{W}_{\mathcal{G}_g} = \text{argmin}_{\mathcal{W}_{\mathcal{G}_{g-1}}} \mathcal{L}(\mathbf{X}_\mathcal{G}, \mathcal{W}_{\mathcal{G}_{g-1}})$, which is the loss function that quantifies the reconstruction error. After training, the optimized parameters $\mathcal{W}_{\mathcal{G}_g}$ are saved and and used to initialize the parameters for the next model, which is on the next largest graph $\mathcal{G}_{g+1}$ with its corresponding input data $\mathbf{X}_{\mathcal{G}_{g+1}}$. This process is repeated iteratively: $\mathcal{W}_{\mathcal{G}_{g+1}} \leftarrow \mathcal{W}_{\mathcal{G}_g}$. Thus, for each subsequent graph $\mathcal{G}_{g+1}$, the model is initialized with the weights $\mathcal{W}_{\mathcal{G}_g}$ from the previously trained model on graph $\mathcal{G}_g$, and the training continues.

```
Algorithm 1: TL for Progressive Graph Sizes
Input: $\mathcal{G}$ and $\mathcal{X}$ with $\mathcal{W}_0$ random initial weights
Output: Final model $M_{\mathcal{G}_k}$ with parameters $\mathcal{W}_{\mathcal{G}_k}$
for each graph $\mathcal{G}_g$ from $\mathcal{G}_1$ to $\mathcal{G}_k$ do
    Initialize the model $M_{\mathcal{G}_g}$ with parameters
      $\mathcal{W}_{\mathcal{G}_g} = \mathcal{W}_0$ if $g = 1$, otherwise $\mathcal{W}_{\mathcal{G}_g} = \mathcal{W}_{\mathcal{G}_{g-1}}$;
    Compute $\hat{Y}_{\mathcal{G}_g}$ from $\mathbf{X}_{\mathcal{G}_g}$
    Calculate the loss: $\mathcal{L}(Y_{\mathcal{G}_g}, \hat{Y}_{\mathcal{G}_g})$
    Update parameters $\mathcal{W}_{\mathcal{G}_g}$ using the Adam optimizer
    if $g < k$ then
        Transfer the learned parameters $\mathcal{W}_{\mathcal{G}_g}$ to the
          next model $M_{\mathcal{G}_{g+1}}$ by setting: $\mathcal{W}_{\mathcal{G}_{g+1}} \leftarrow \mathcal{W}_{\mathcal{G}_g}$;
        Freeze the weights in earlier layers to preserve
          the learned features: $\frac{\partial \mathcal{W}^{(\text{frozen})}_{\mathcal{G}_{g+1}}}{\partial t} = 0$;
    end
end
return Trained model $M_{\mathcal{G}_k}$ with parameters $\mathcal{W}_{\mathcal{G}_k}$
```

V. Experimental Results

We herein present the experimental results in terms of attack detection and model efficiency.

A. Model Evaluation

The deep and proposed models are evaluated by comparing its inferences $\hat{Y}$ with ground truth labels Y. We compute the Mahalanobis distance for each error vector $\mathbf{e}$ in error matrix $\mathcal{E} = Y - \hat{Y}$. The Mahalanobis distance is given by: $\Xi = \sqrt{(\mathbf{e}-\boldsymbol{\mu})^T \varphi^{-1}(\mathbf{e}-\boldsymbol{\mu})}$ where $\boldsymbol{\mu}$ is the global mean error, φ is the covariance matrix of $\mathcal{E}$. This distance metric helps identify outliers, which indicate attacks. By examining the squared Mahalanobis distance over consecutive batches of predictions, we calculate the mean squared Mahalanobis distance for each batch: $\overline{\Xi}^2 = \frac{1}{S}\sum_{i=1}^{S} \xi_i^2$, where S is the batch size and ξ is an error vector of Ξ. If the mean squared distance exceeds a predefined threshold, the batch is flagged as an attack. The threshold is determined based on the model's performance using the validation set. Detection performance is then quantified using F1-score (F1 $= \frac{2\cdot \text{TP}}{2\cdot \text{TP}+\text{FP}+\text{FN}}$), accuracy (ACC $= \frac{\text{TP}+\text{TN}}{\text{TP}+\text{TN}+\text{FP}+\text{FN}}$), and detection rate (DR $= \frac{\text{TP}}{\text{TP}+\text{FN}}$), where TP, TN, FP, and FN denote true positive, true negative, false positive, and false negative samples, respectively. We also measure model complexity in terms of the time (in seconds) taken to train the model.

B. Model Hyperparameters

We compare the performance of the proposed models to benchmarks. The hyperparameters of all models are tuned using sequential grid-search [22] with the following outcomes.

1) Shallow Models: SVM has gamma and regularization values of 0.07 and 10, respectively. Random forest (RF) utilizes 10 estimators with entropy criterion. LGBM employs a learning rate of 0.5, max depth of 2, and 100 estimators.

TABLE I
Detection performance on cyberattacks (%).

Model	Metric	Input Graph Size (Number of Nodes)		
		10-nodes	20-nodes	31-nodes
SVM	**F1**	6.1	13.2	26.2
	ACC	80.7	78.4	77.7
	DR	3.2	9.3	20.1
RF	**F1**	17.5	22.7	28.9
	ACC	82.4	80.1	76.1
	DR	9.5	12.4	30.1
LGBM	**F1**	19.8	20.1	22.3
	ACC	82.5	81.3	81.9
	DR	11.1	11.3	13.3
FFNN	**F1**	20.9	24.1	36.1
	ACC	78.6	80.6	83.3
	DR	14.5	22.7	24.1
LSTM	**F1**	26.0	27.5	34.5
	ACC	77.0	70.3	64.7
	DR	20.6	25.9	47.4
AE	**F1**	21.5	31.4	38.1
	ACC	67.2	80.1	83.4
	DR	23.1	24.3	26.1
TGCN	**F1**	49.6	59.3	59.8
	ACC	79.3	79.6	80.4
	DR	51.8	53.4	54.8
Proposed STGT	**F1**	52.3	59.4	76.3
	ACC	81.5	83.7	84.8
	DR	54.2	55.6	74.7
Proposed TL-STGT	**F1**	54.8	65.1	79.7
	ACC	82.6	86.1	87.1
	DR	53.6	59.6	76.8

2) Deep Models: Feed forward neural network (FFNN) has 5 layers, 500 neurons, and $\tanh$ activation. LSTM has 3 layers, 100 neurons, and $\tanh$ activation. AE has 4 layers, and ReLU activation. All deep models use Adam optimizer.

3) Graph Models: TGCN has 3 layers, 64 hidden units, 0.001 learning rate, and Adam optimizer. The proposed models use 4 layers, 64 GRU hidden units, 8 transformer heads, 128 dense layer neurons, ReLU activation, 0.001 learning rate, and Adam optimizer.

C. Simulation Results

This section presents the simulation results in terms of attack detection performance and efficiency (i.e., training time).

1) Attack Detection Performance: Table I presents the attack detection performance of the benchmark and proposed models. The following overall observation are made. First, the deep benchmark models outperform shallow ones by $1.1 - 19.9\%$ in F1-score due to their ability to capture complex patterns that shallow models might miss. Second, the graph benchmark model (TGCN) outperforms shallow and deep ones by $29.8 - 46.1\%$ and $21.7 - 35.2\%$ in F1-score, respectively, due to its ability to capture spatial aspects of the data. Third, the proposed STGT model outperforms shallow, deep, and graph benchmark models by $32.5\% - 54\%$, $26.3\% - 41.8\%$, and $0.1 - 16.5\%$ in F1-score, respectively, due to its ability to capture spatio-temporal aspects of the data with longer range dependencies as it implements a Transformer. Fourth, the proposed TL-STGT model outperforms benchmark shallow, deep, and graph models by $35\% - 57.4\%$, $28.8\% - 45.2\%$, and $5.2 - 19.9\%$ in F1-score, respectively, offering enhanced F1-score by $2.5 - 5.7\%$ compared to the proposed STGT. Besides capturing spatio-temporal aspects, the superior performance of the TL-STGT model is due to efficiently capturing dependen-

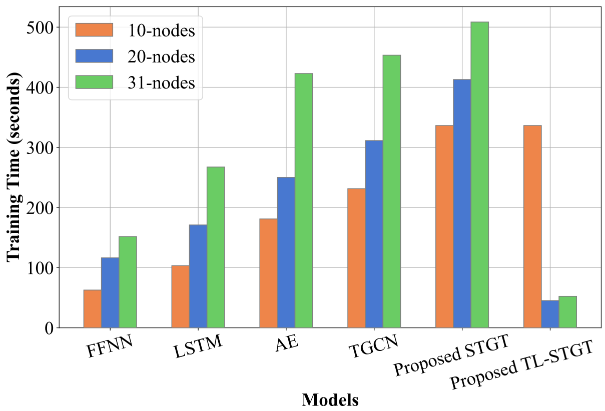


Fig. 4. Training time with early stopping (seconds).

cies in graph-based structures using its TL ability that transfers weights from smaller graphs to bigger ones while offering enhanced computational complexity as described next.

2) Proposed Model Efficiency and Scalability: Figure 4 illustrates the training efficiency of the benchmark models compared to the proposed TL-STGT approach in terms of training time (in seconds). For the 10-node graph, the proposed TL-STGT model requires a higher training time due to the initial training phase, during which all features are learned, which is expected given its graph structure, leading to higher training time during the first step. However, for the 20-node and 31-node graphs, the training process leverages the weights from the previously trained models, significantly reducing the training time. In general, larger graph structures require more computations due to the increased number of nodes and edges, resulting in longer training times. For the proposed TL-STGT approach, TL substantially reduces computation time, saving between 99.5 and 401.6 seconds on the larger graphs. This represents a reduction of $65.5-88.5\%$ in training time compared to benchmark models. The proposed model's improved efficiency is attributed to TL, where several layers are frozen and retain their pre-trained weights from smaller models. Consequently, the model only needs to learn new weights for the graph convolution and GRU steps, reducing the computational burden and enabling faster training on larger graphs. As a result, the proposed TL-STGT offers high scalability with improved F1-score by $14.6-24.9\%$ when detecting attacks in larger WDSs (31-node graph) compared to smaller ones (20 and 10-node graphs).

VI. Conclusions

In this paper, we proposed a transfer learning-based technique (TL-STGT) to detect cyberattacks in WDSs. The proposed TL-STGT model utilizes graph convolution to capture spatial features from WDS graph representations, while GRU and transformer blocks are employed to capture long-range temporal dependencies. TL demonstrates that larger graph structures can be trained with reduced computational complexity, enhancing efficiency. The proposed model offered improvements over shallow, deep, and graph models by $35-57\%$, $29-45\%$, and $5-20\%$, respectively, against replay, DoS, and data manipulation attacks. The proposed model reduces the training time by $66\%-89\%$ compared to benchmark models, offering high scalability with F1-score improvements of $15-25\%$ when detecting attacks in larger WDSs compared to smaller ones. Future work will focus on the detection and localization of more attack types.

References

[1] L. Palma *et al.*, "Contaminations in water distribution systems: a critical review of detection and response methods," *AQUA—Water Infrastructure, Ecosys. and Soc.*, vol. 73, no. 6, pp. 1285–1302, June 2024.

[2] H. H. Addeen *et al.*, "A survey of cyber-physical attacks and detection methods in smart water distribution systems," *IEEE Access*, vol. 9, pp. 99 905–99 921, July 2021.

[3] U. Parajuli and S. Shin, "Identifying failure types in cyber-physical water distribution networks using machine learning models," *AQUA—Water Infra., Ecosys. and Soc.*, vol. 73, no. 3, pp. 504–519, Mar. 2024.

[4] A. Takiddin *et al.*, "Resilience of data-driven cyberattack detection systems in smart power grids," in *32nd European Signal Processing Conf. (EUSIPCO)*. Lyon, France, 26-30 Aug. 2024, pp. 1992–1996.

[5] Stateline, "Florida Hack Exposes Danger to Water Systems," https://tinyurl.com/2emfebs2, [Online: accessed Mar. 2024].

[6] D. T. Ramotsoela *et al.*, "Attack detection in water distribution systems using machine learning," *Human-centric Computing and Information Sciences*, vol. 9, pp. 1–22, April 2019.

[7] R. Taormina *et al.*, "Deep-learning approach to the detection and localization of cyber-physical attacks on water distribution systems," *J. of Water Res. Pln. and Mang.*, vol. 144, no. 10, Oct. 2018.

[8] T. D. Ramotsoela *et al.*, "Behavioural intrusion detection in water distribution systems using neural networks," *IEEE Access*, vol. 8, pp. 190 403–190 416, Oct. 2020.

[9] M. N. K. Sikder *et al.*, "Deep h2o: Cyber attacks detection in water distribution systems using deep learning," *J. of Water Process Eng.*, vol. 52, p. 103568, April 2023.

[10] A. Deng and B. Hooi, "Graph neural network-based anomaly detection in multivariate time series," in *AAAI Conf. on artificial intelligence*, vol. 35, no. 5. Vancouver, Canada, 2–9 Feb. 2021, pp. 4027–4035.

[11] R. Xue *et al.*, "Large-scale graph neural networks: The past and new frontiers," in *SIGKDD Conf. on Knowledge Discovery and Data Mining*. Long Beach, CA, USA, 06-10 Aug. 2023, pp. 5835 – 5836.

[12] W. Hamilton *et al.*, "Inductive representation learning on large graphs," *Advances in neural information processing systems*, vol. 30, pp. 1025–1035, Dec. 2017.

[13] A. Takiddin *et al.*, "Generalized graph neural network-based detection of false data injection attacks in smart grids," *IEEE Trans. on Emerging Topics in Computational Intelligence*, vol. 7, no. 3, pp. 618–630, June 2023.

[14] A. Ostfeld, Salomons *et al.*, "Battle of the water calibration networks," *J. of Water Res. Pln. and Mang.*, vol. 138, no. 5, pp. 523–532, Sept. 2012.

[15] R. Taormina *et al.*, "Battle of the attack detection algorithms: Disclosing cyber attacks on water distribution networks," *J. of Water Res. Pln. and Mang.*, vol. 144, no. 8, p. 04018048, Aug. 2018.

[16] J. Zhou *et al.*, "Graph neural networks: A review of methods and applications," *AI Open*, vol. 1, pp. 57–81, Apr. 2020.

[17] R. Saxena *et al.*, "An efficient influence maximization technique based on betweenness centrality measure and clustering coefficient," in *Conf. on Comp. and Aut. Eng.* Sydney, Australia, 03–05 Mar. 2023, pp. 565–569.

[18] R. Taormina *et al.*, "A toolbox for assessing the impacts of cyber-physical attacks on water distribution systems," *Environmental modelling & software*, vol. 112, pp. 46–51, Feb. 2019.

[19] A. Takiddin *et al.*, "Robust electricity theft detection against data poisoning attacks in smart grids," *IEEE Trans. on Smart Grid*, vol. 12, no. 3, pp. 2675–2684, May 2021.

[20] J. Xu *et al.*, "Anomaly transformer: Time series anomaly detection with association discrepancy," in *Int. Conf. on Learn. Rep.* Virtual, 25–29 April 2022.

[21] A. Vaswani *et al.*, "Attention is all you need," in *Conf. on Neural Info. Proc. Sys.* Long Beach, CA, USA, 4 – 9 Dec. 2017.

[22] A. Takiddin *et al.*, "Spatio-temporal graph-based generation and detection of adversarial false data injection evasion attacks in smart grids," *IEEE Trans. on Artificial Intelligence*, vol. 5, no. 12, pp. 6601–6616, Dec. 2024.