Teaching Machines to Understand Urban Networks: A Graph Autoencoder Approach

Maria Coelho*, Mark A. Austin†, Shivam Mishra‡, and Mark Blackburn §
*†‡ University of Maryland, College Park, MD 20742
*† Department of Civil and Environmental Engineering
†‡ Institute for Systems Research
E-mail: mecoelho@terpmail.umd.edu; austin@umd.edu; smishra8@umd.edu
§ Stevens Institute of Technology, Hoboken, NJ, USA
E-mail: mblackbu@stevens.edu

***Abstract*—Due to remarkable advances in computer, communications and sensing technologies over the past three decades, large-scale urban systems are now far more heterogeneous and automated than their predecessors. They may, in fact, be connected to other types of systems in completely new ways. These characteristics make the tasks of system design, analysis and integration of multi-disciplinary concerns much more difficult than in the past. We believe these challenges can be addressed by teaching machines to understand urban networks. This paper explores opportunities for using a recently developed graph autoencoding approach to encode the structure and associated network attributes as low-dimensional vectors. We exercise the proposed approach on a problem involving identification of leaks in urban water distribution systems.**

Keywords-Systems Engineering; Machine Learning; Graph Embeddings; Graph Autoencoders; Digital Twins; Water Distribution Systems.

I. Introduction

This paper is concerned with the integration of recently developed graph embedding procedures with machine learning tasks, which together can enhance digital twin design and decision making in urban settings. It builds upon our previous work [1] on teaching machines to understand urban networks with graph analytics techniques.

A. Problem Statement

The concept of creating digital replicas to serve as tools to improve decision-making has long been used in engineering. During the past three decades, however, remarkable advances in technology – including Internet of Things, artificial intelligence, and augmented and virtual reality (AR/VR) – have created opportunities to develop "digital twins," a cyber representation of an object, process or place that mirrors its implementation in the physical world through real-time monitoring, and synchronization of data with events. To this end, sensor and actuator systems, and algorithms and software are provided for observation, reasoning and physical systems control. NASA initially proposed the digital twin concept in the late 90s as a way to support the design and operation of air vehicles [2]. More recently, cities around the world have entertained use of the digital twin concept as a way to transform processes for day-to-day management and long-term urban planning and design.

The design of strategies to achieve superior levels of urban operation, even when available resources are limited, is complicated by the distributed, concurrent, and multi-disciplinary nature of large-scale urban systems. It is well known that when a disruption to urban operations requires the instantiation of recovery procedures, it is essential that the participating domains share information at key points in the system operation to enhance common knowledge, and their individual ability to make decisions appropriate to their understanding of the system state, its goals and objectives.

Despite the abundance of available urban data, current urban systems' potential for reaching enhanced capabilities in the decision-making and management of city infrastructure is hampered by lack of systematic knowledge exchange. The digital twins concept can overcome this barrier through its ability to integrate domains into the real-time knowledge discovery process from heterogeneous urban data. At this time, however, many challenges remain in digital twin design and implementation. First, there is a lack of unified models or a generic digital twin architecture in the literature, with no consensus on how to build a digital twin system [3]. Second, urban environments create further design difficulties, mainly related to the complexity of city-wide modeling and the lack of standards supporting cross-disciplinary data exchange. Research is needed to understand how to design the digital twin elements and their interactions so that collectively they can overcome these challenges.

B. Architecting Urban Digital Twins

Requirements for architecting urban digital twins include the need for systems that can identify anomalies (faults) in system performance, and model the behavior of processes and interactions among the different domains within a city. Since a generic (or unifying) digital twin architecture does not exist at this time, we believe that the best pathway forward for digital twin design is with architectures that combine Machine Learning (ML) formalisms and Semantic Model representations that work side-by-side as a team, providing supportive roles for the collection and processing of data, identification of events,

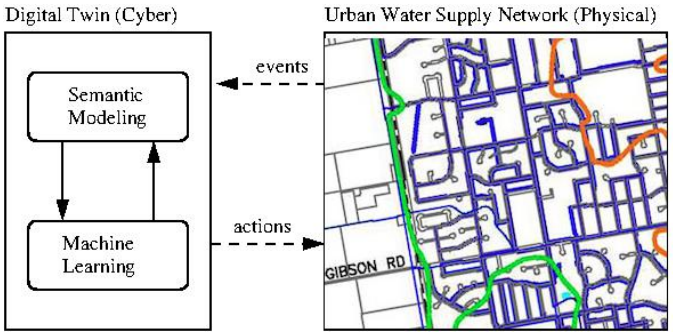

Figure 1. High-level representation for an urban water supply network digital twin (cyber) working alongside a physical urban water supply network.

and real-time management of city operations. Figure 1 shows the application of this concept to a small urban water supply system.

Figure 2 shows the proposed architecture, building upon our recent work in semantic modeling for (multi-domain) system of systems [4]–[6] and exploration of a combined semantic and ML framework exercised in energy-efficient buildings and brain cancer profiles [7], [8]. The proposed architecture is comprised of three sections: (1) multi-domain semantic modeling, (2) data mining, and (3) machine learning of graphs.

Box 1: Multi-domain Semantic Modeling. By their very nature, urban systems are a composition of many different types of domains and their interactions. These domains are geographically dispersed and intertwined, and have behaviors that are distributed and concurrent. Box 1 of Figure 2 shows that instead of modeling the dynamic behavior of systems with a centralized control and one large catch-all network, we explore opportunities for modeling systems as collections of domain-specific networks that dynamically evolve in response to events. Individual urban domains will operate as concurrent processes, each having their own thread of execution, and will respond to incoming data from external domains. Each domain will have a graph that evolves according to a set of domain-specific rules, and subject to satisfaction of constraints. Domains will interact when they need to in order to achieve collective objectives. If goals are in conflict, or resources are insufficient, then negotiation will need to take place. The model uses Semantic Web technologies for the implementation of ontologies, rules checking, and message passing mechanisms. The distinguishing feature of this framework is the concurrent development of ontologies, rules and data models, which are placed on an equal footing. When used in an urban context as part of the digital twin architecture, this approach to semantic modeling forces cross-domain data exchange that is more homogeneous than it would otherwise be, and establishes common knowledge among domains. Rule-based reasoning procedures can also be developed to impose fairness in domain operations and prevent deadlocks.

Box 2: Data Mining. Recent developments in the field of computational intelligence are sometimes termed machine learning (ML). Machine learning techniques deviate from traditional models of computation in their ability to perform data analytics by learning patterns and hidden insights in data. Box 2 of Figure 2 shows ML for three classes – classification, clustering and association – of data mining. For the most part, these three data mining techniques were developed in the 1980s and 90s, and so the associated algorithms and software [9] are now quite mature. Data mining techniques also include use of recurrent neural network architectures to represent temporal sequences, and algorithms to detect anomalies in expected temporal behavior. For our purposes, these anomalies are events that can trigger the activation of urban recovery procedures modeled in Box 1.

Box 3: Machine Learning of Graphs. Remarkable advances in ML algorithms (2016-2019) include the ability of a machine to learn the structure of a graph and its attributes. The so called graph embedding methods learn a continuous vector space for the graph, assigning each node (and/or edge) in the graph to a specific position in the vector space. These embeddings can be later used to advance various learning tasks, such as node classification, node clustering, node recommendation, link prediction, and so forth [10].

C. Scope and Objectives

During 2018, our studies [7], [8], [11] focused on semantic foundations and data mining techniques working together as a team. Data mining/ML techniques for the classification and clustering of data provided useful feedback on the structuring of ontologies. In 2019, our studies started to explore graph embedding techniques for learning urban networks (i.e., see Box 3 of Figure 2) [1]. The Dynamic Attributed Network Embedding (DANE) [12] was used to generate low-dimensional vectors for a water distribution network. The embeddings were then fed to a Random Forest (RF) algorithm trained to identify water leaks. Although these initial studies showed successful results in identifying water leaks, the DANE framework did not incorporate the capability to verify that the embedding input to the RF Classifier was an accurate representation of the physical network topology and node attribute information. This lack of insight suggests that a better approach would replace hand generation of features with learning models trained to identify features encoded within embedding vectors, and then validate that the trained machine models faithfully replicate the physical graph topology. Research is also needed to understand the effects of graph size on learning performance. The latter is key for scalability of the proposed approach.

In a step toward resolving these problems, and extending our previous work, the objectives of this work are three fold: (1) identify an embedding framework that better fits our need for minimizing the loss of information during the network embedding process; (2) exercise the new embedding framework on the problem of leaks identification in an urban water distribution system; (3) explore the effects of network size on the learning performance. The remainder of this paper proceeds as follows: Related work in traditional and ML approaches for graph modeling is covered in Section II. An overview of different graph embedding techniques is provided in Section III. We exercise graph embedding and ML classification procedures in a water distribution system

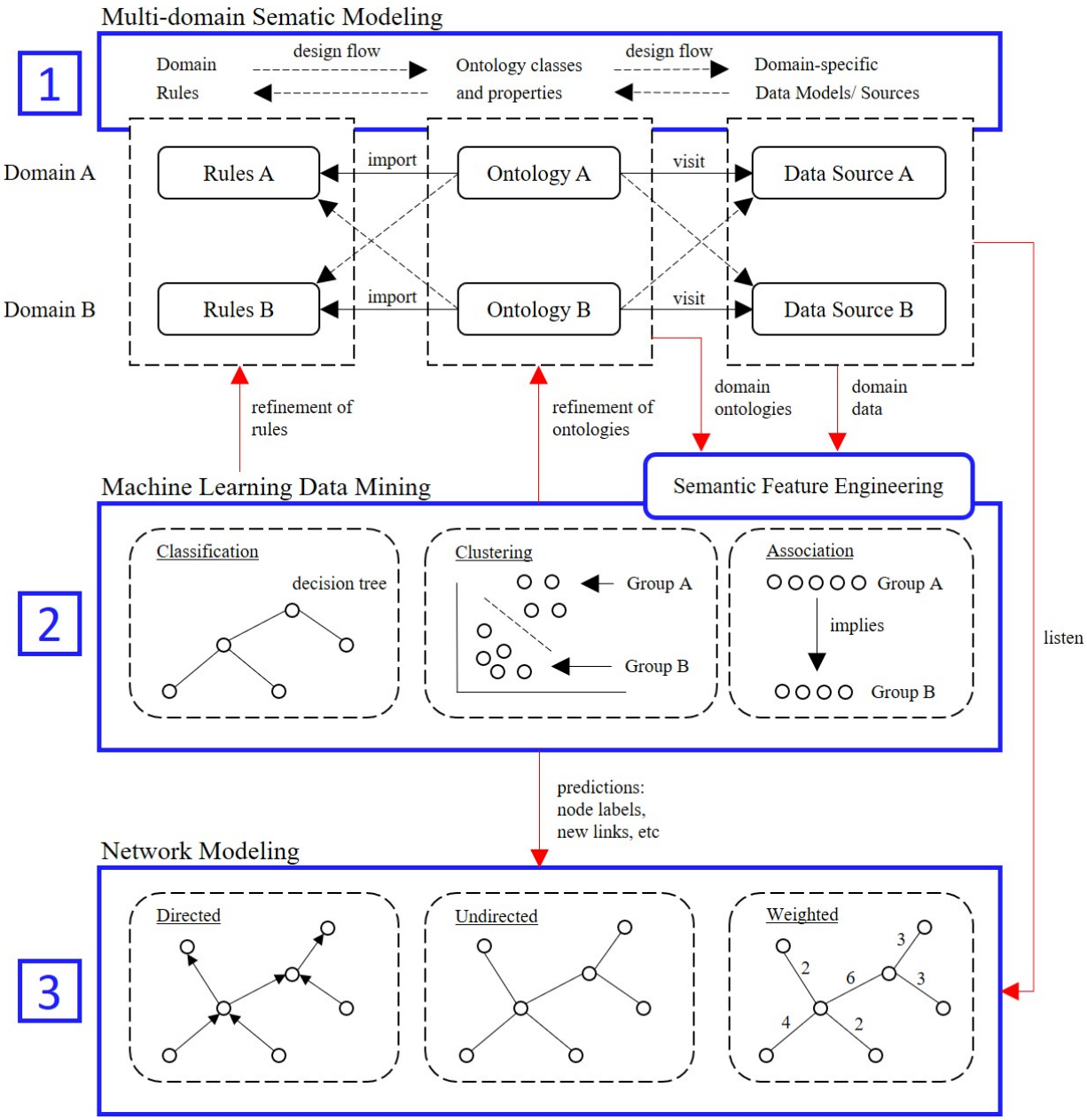

Figure 2. Digital twin architecture.

application involving identification of leaks in Section IV. We discuss ideas for scaling up the simulations in Section V. The conclusions and directions for future work are located in Section VI.

II. Related Work

This section covers related work in traditional and ML approaches to graph modeling.

A. Graph Theory

In mathematical terms, a graph G = (V, E), where V is a set of vertices (also called nodes or points), E = set of edges (also called links or lines), and each edge is formed from pair of distinct vertices in V. V and E are usually taken to be finite. Graph theory is the study mathematical structures used to model pairwise relations between node and edge objects. It plays a central role in understanding how solutions to problems on graphs (e.g., traversal of nodes; reachability) are affected by the fundamental characteristics of the graph.

The study of urban systems as networks, and networks of interacting networks, draws upon many aspects of graph theory. Urban networks may be homogeneous, heterogeneous, and carry auxiliary information modeled as attributes. The information to be preserved in the network is strongly affected by the underlying characteristics of the system and, unfortu-

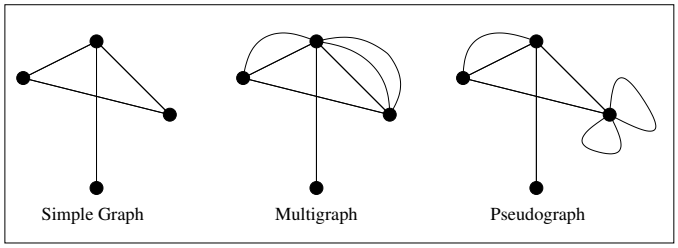

Figure 3. Structure of simple graphs, multigraphs and pseudographs.

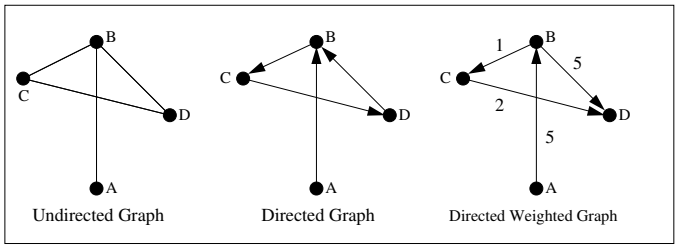

Figure 4. Graphs with undirected, directed, and directed weighted edges.

nately, in urban settings this is complicated by a wide range of possibilities. Figures 3 and 4 show, for example, that graph structures include simple, multigraphs and pseudographs. Their edges may be undirected, directed and/or weighted.

Kong and Simonovic [13] describe a few examples of how urban systems can be modeled as graphs. A street network can be represented as a graph composed of street junctions, end points and street segments. Generally, the edges are undirected, homogeneous, and the street network is fully connected. A water distribution network can be represented as graph where water works, storage facilities and pump stations are nodes with different attributes, and the water distribution pipes are directed edges, where the direction indicates the flow of water. A power grid can be represented by a graph where power plants, distribution and transmission substations are nodes with different attributes, and power lines are directed edges for the flow of electricity transmission. Information infrastructure can be represented as a graph where the Internet Service providers are nodes and cable connections are undirected edges, since these networks provide a bidirectional flow of information.

Regardless of the type of graph and its purpose in an urban setting, we observe that data mining techniques can be employed to access the information stored in graphs. This information can then be used to improve the design and operation of urban systems. Techniques for graph data mining can be divided into graph analysis and graph analytics.

B. Graph Analysis

Graph analysis tasks are mainly focused on exploring the graph data to gain insights about its topological properties. Network topology (see Figure 3) nearly always affects function; therefore, it is important to characterize it. For instance, the topology of social networks affects the spread of information and disease, and the topology of the power grid affects the robustness and stability of power transmission [14]. Traditional approaches to network/graph modeling employ adjacency matrices (or a simplified representation of network adjacency) to model the topology of graphs. The topological properties can then be explored through typical graph analysis tasks including connectivity analysis, traceability analysis, cycle detection, and shortest path identification.

C. Graph Analytics

For high-dimensional problems that are data sparse, traditional approaches to graph representation and analysis can quickly become computationally prohibitive. A second problem is these traditional approaches do not capture the semantics of the network (i.e., node attributes). In recent years, there has been a surge in ML approaches that automatically learn to encode graph topology and attributes into low-dimensional embeddings.

Figure 5 shows simplified representations for traditional and machine learning approaches to graph representation. In a significant departure from traditional approaches to representing and studying the the properties of a graph, graph embedding methods learn a continuous vector space for the graph, assigning each node (and/or edge) in the graph to a specific position in the vector space, with the goal of preserving local linkage structure (not global structure) and/or network semantics. First the encoder maps the node to a low-dimensional vector embedding, based on the node's position in the graph and/or its attributes. Next, the decoder extracts information from the embedding vector (i.e., node's local graph neighborhood, or a classification label associated with the node). By jointly optimizing the encoder and decoder, graph embedding methods learn to compress information about graph structure and semantics into the low-dimensional embedding space. Because information can be lost in the embedding transformation process – it can be viewed as dimensionality reduction – the output embeddings are statistical in nature and, as such, should be interpreted as graph analytics (not graph analysis). Graph analytics can extract unseen or difficult to obtain properties of the graph, either directly or by feeding the learned representations to a downstream inference pipeline, such as node classification, node clustering, and link prediction.

III. Teaching Machines to Understand Graphs

In recent years, many graph embedding approaches have been developed. This section introduces these different approaches, their advantages and limitations.

A. Graph Embedding Methods

Goyal and Ferrara [15] have organized embedding methods into three broad categories: factorization based, random walk based, and deep learning based. Both the factorization and random walk based approaches train unique embedding vectors for each node independently, which results in several limitations. First, there is an absence of parameter sharing between the nodes, which leads to computational inefficiency

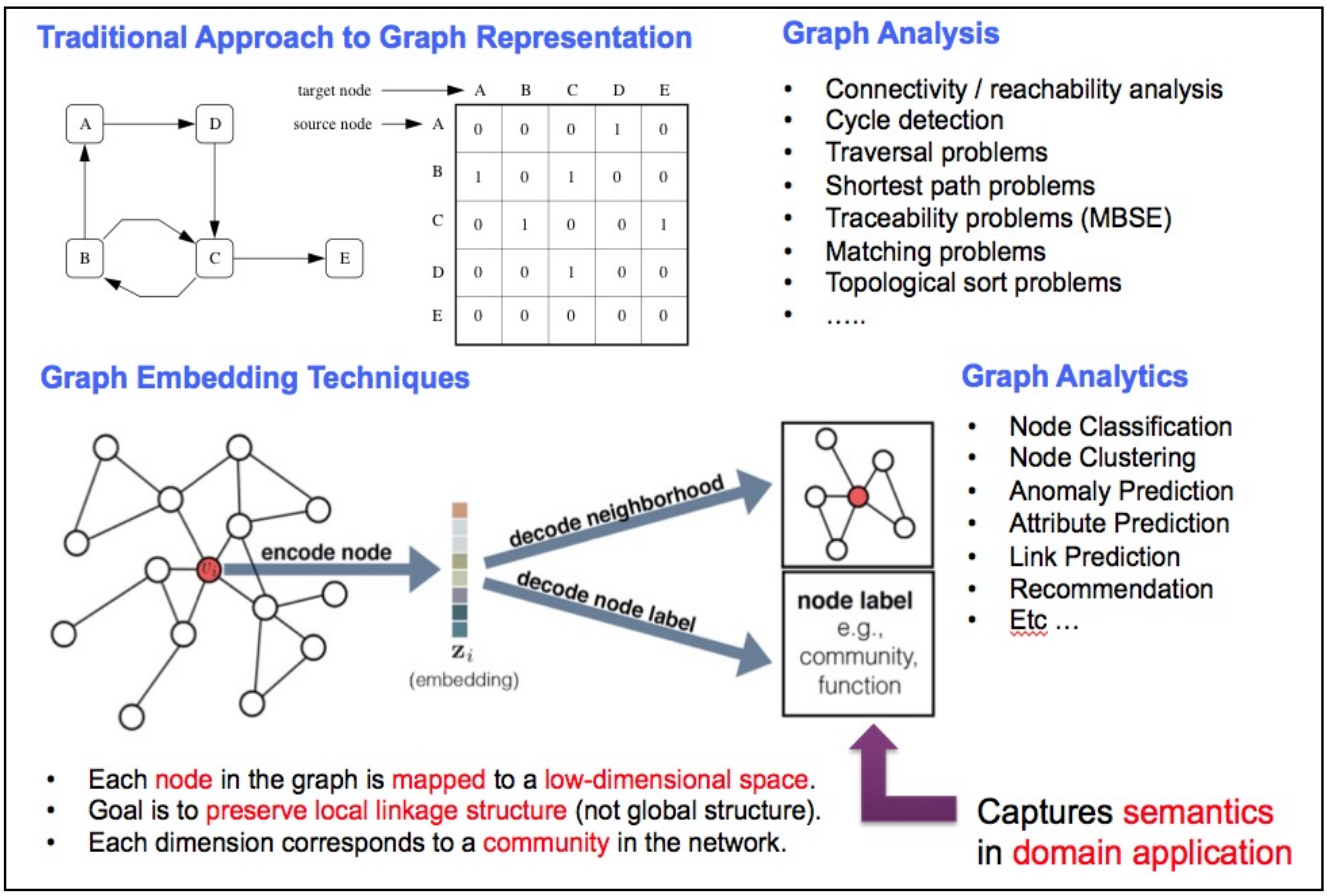

Figure 5. Graph analysis vs graph analytics.

since the number of trainable parameters grows linearly with the number of nodes in the graph. Second, these approaches are not extensible and are only able to generate embedding vectors for the nodes that were present during the training phase and not for any unseen nodes. A third problem is that these approaches lack an ability to incorporate node attributes during embedding generation, when node attributes can be highly informative about the node's position and role in the network.

In a concerted effort to mitigate these concerns, recent research efforts have led to the emergence of deep learning based methods. Deep learning based approaches use a deep neural network architecture, generally referred to as Graph Neural Networks (GNNs), to generate embeddings which depend both on the structure and the attributes of the graph. Wu et al. identified many types of GNNs among which are graph autoencoders (GAEs) [16].

B. Graph Autoencoders

GAEs are deep neural network architectures that are trained with the objective of reconstructing their original graph input. Figure 6 shows a high-level GAE architecture. First, an encoder takes a graph as its input and systematically compresses it into a low-dimensional vector. The decoder then takes the vector representation and attempts to generate a reconstruction of some user-defined graph analysis tool (e.g., the adjacency matrix) of the original graph input. Encoder-decoder pairs are designed to minimize the loss of information between the input graph and the output (i.e., reconstructed) graph. These frameworks may be deterministic or probabilistic [17].

C. Network Embedding in an Urban Context

Figure 7 is a schematic for how the embedding process might be integrated in an urban network digital twin. We start by extracting a graph representation of the urban infrastructure and determining the initial parameters of the system. As time progresses, the digital twin will monitor changes in the system. Embeddings will be generated, and machines will be trained to understand a number of salient features of acceptable and unacceptable behavior. When an unacceptable behavior is identified, urban recovery procedures will be triggered. Since embeddings are a central and essential part of the learning process, there is a need to ensure the embedding input to the machine is an accurate representation of the information contained in the graph.

IV. Case Study Problems

In this section, we exercise a graph autoencoding procedure that can encode the structure and network attributes

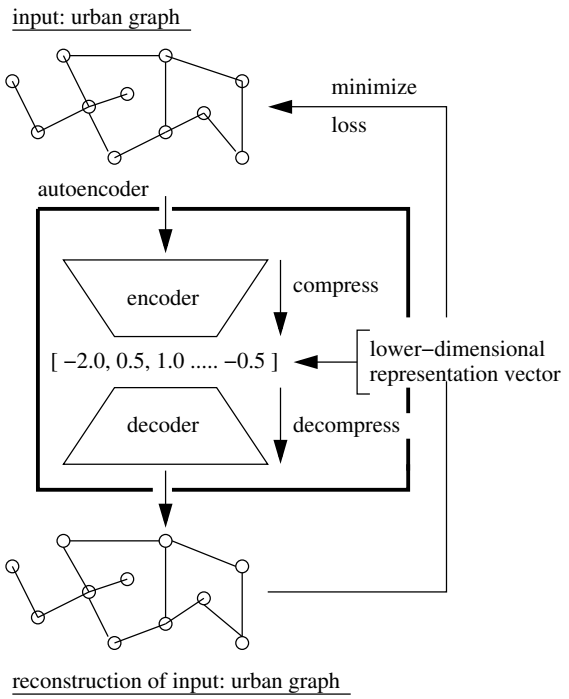

Figure 6. Traditional encoder-decoder approach.

on problems involving the identification of leaks in an urban water distribution system.

Topic 1. Case Study Applications

The two case study applications presented in this work aim to explore the use of graph autoencoding (GAE) procedures and ML classification techniques for the detection and localization of leakages in water distribution systems (WDSs). We start by extracting a graph representation of the system at hand and determining the initial node attributes of the system. Topics 2 through 5 follow the analysis procedure and describe: (2) the encoding of the network topology and node attribute information as node embeddings by a GAE framework; (3) synthetic generation of hydraulic (node pressure) data by the hydraulic simulation software EPANET [18]; (4) training and testing of a Random Forest (RF) algorithm [19] with the node embeddings to infer leak location; and (5) performance of the proposed framework.

Topic 2. Node Embedding

The selection of an appropriate GAE is not a trivial task. Water distribution networks are associated with a rich set of node attributes that are in constant flux due to day-to-day temporal variations in demand and, longer term, network expansion and urban growth. These variations cause the emergence of new content patterns and the fading of old content patterns. Despite these complications, it has been widely studied and reported that there exists a strong correlation among the attributes of linked nodes [20]. These node correlations and changing characteristics motivate us to seek an effective embedding representation to capture network structure and attribute evolving patterns, which is of fundamental importance to learning in a dynamic environment. Early GAE algorithms such as Deep Neural Network for Graph Representations (DNGR) [21] and Structural Deep Network Embedding (SDNE) [22] only consider node structural information (i.e., connectivity between pairs of nodes), and ignore node attribute information. In 2016 Kipf et al. introduced Graph Autoencoder (GAE*) [23], an algorithm that leverages a ConvGNN [24] to encode node structural information and node feature information at the same time. We employ this algorithm in the two case studies of this work.

Although GAE* is the graph embedding approach most suitable for our purposes, a key research question remains: Are the procedures for reconstruction of the graph topology guaranteed to give the correct answer only most of the time or all of the time? An essential prerequisite for understanding the robustness of GAE* reconstruction procedures is to first understand the procedure itself (i.e., graph encoding, embedding optimization and reconstruction). Hence, the objective of our first case study is to walk step by step through the application of the GAE* framework to urban graphs from input to output. We deliberately keep the network layout and node attributes very simple, and begin with encoding for a 4 node graph and its node attributes.

Figure 8 shows the architecture of the GAE* framework applied to this use case. The encoder consists of two graph convolutional layers and a simple inner product decoder, which aims to decode node relational information from generated embeddings by reconstructing the graph adjacency matrix. The first convolutional layer takes as input the graph's node feature matrix X and the symmetrically normalized adjacency matrix $\tilde{A} = D^{-1/2}AD^{-1/2}$, where A is the adjacency matrix with added self connections and D is the diagonal node matrix of A. For the purposes of keeping the case study as simple as possible, all of the node features are simply assigned a value of 1. Note, however, in the later case study, different system nodes will have different feature values that are naturally changing.

The first convolutional layer generates a lower-dimensional feature matrix defined as:

$$\bar{X} = ReLu(\tilde{A}XW^0)$$

where W^0 is a trainable parameter matrix. The second convolutional layer takes as input the output of the previous layer and generates the node embeddings:

$$Z = \tilde{A}ReLu(\tilde{A}XW^0)W^1$$

where W^1 is also a trainable parameter matrix. The purpose of the decoder is to reconstruct the adjacency matrix A (with added self-connections) from Z. By applying the inner product on the latent variable Z and Z^T, the algorithm learns the similarity of each node inside Z. By applying the sigmoid function $\sigma(\cdot)$ the algorithm computes the probability of edges existing between the range of 0 and 1. Therefore, the reconstructed adjacency matrix is defined as:

$$A' = \sigma(ZZ^T)$$

In order to arrive at the optimal embedding matrix Z, the W^0 and W^1 parameters are systematically updated through

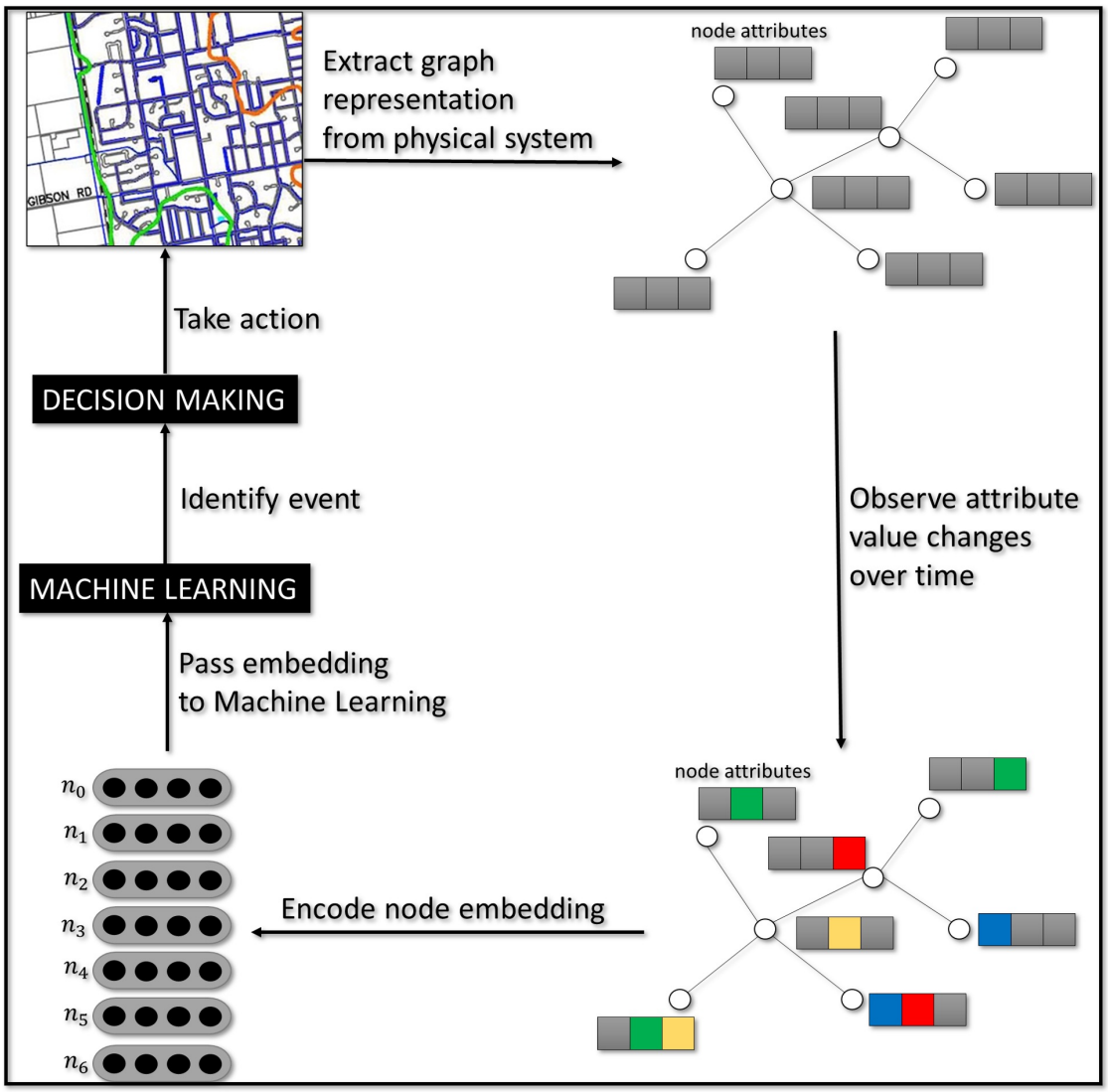

Figure 7. Process flowchart for training and executing machine.

an Adam optimization [25] of the weighted cross-entropy loss between the adjacency matrix A and the soft reconstruction A'. Adam optimization uses a combination of extensions (i.e., adaptive gradient estimation and root mean square propagation) to stochastic gradient descent to estimate gradients and their moments as a moving average. These gradient estimates allow us to find the direction in which the weights should be adjusted in order to reduce the weighted cross-entropy loss.

By applying the GAE* architecture to the layout of Case Study 1, we obtain successful reconstruction of the test network topology. However, GAE* does not have the ability to reconstruct graph attribute information. Thus, an important research question that remains open is whether the embedding obtained using GAE* is an accurate representation of node attribute information needed for leak detection in WDS. In the following case study, we aim to answer this question by applying GAE* architecture to a leaking WDS, and use the generated embedding as input for a RF classifier trained to identify leaks.

Topic 3. Data Generation

Automatic water leakage detection can be performed with ML classification algorithms. These algorithms require training data involving normal operation conditions and abnormal operation conditions. In the case of a WDS, training data should involve hydraulic parameters at different locations, and pertain to normal operations and leaks from the past. Ideally, one would like to train a network using data measurements from a real-world network. For security reasons, however, WDS data (i.e., geographical layout of pipes, tanks, and demands) are kept confidential by the water utility companies and are not readily available to the public. A second option is to use simulators to synthetically generate data for machine learning. This study employs the simulation tool EPANET [18] to generate the training data for the WDS under consideration [26]. EPANET is a computerized model produced by the Environmental Protection Agency of the USA that simulates the dynamic hydraulic and water quality behavior within a WDS operating over an extended period of time. WDSs are

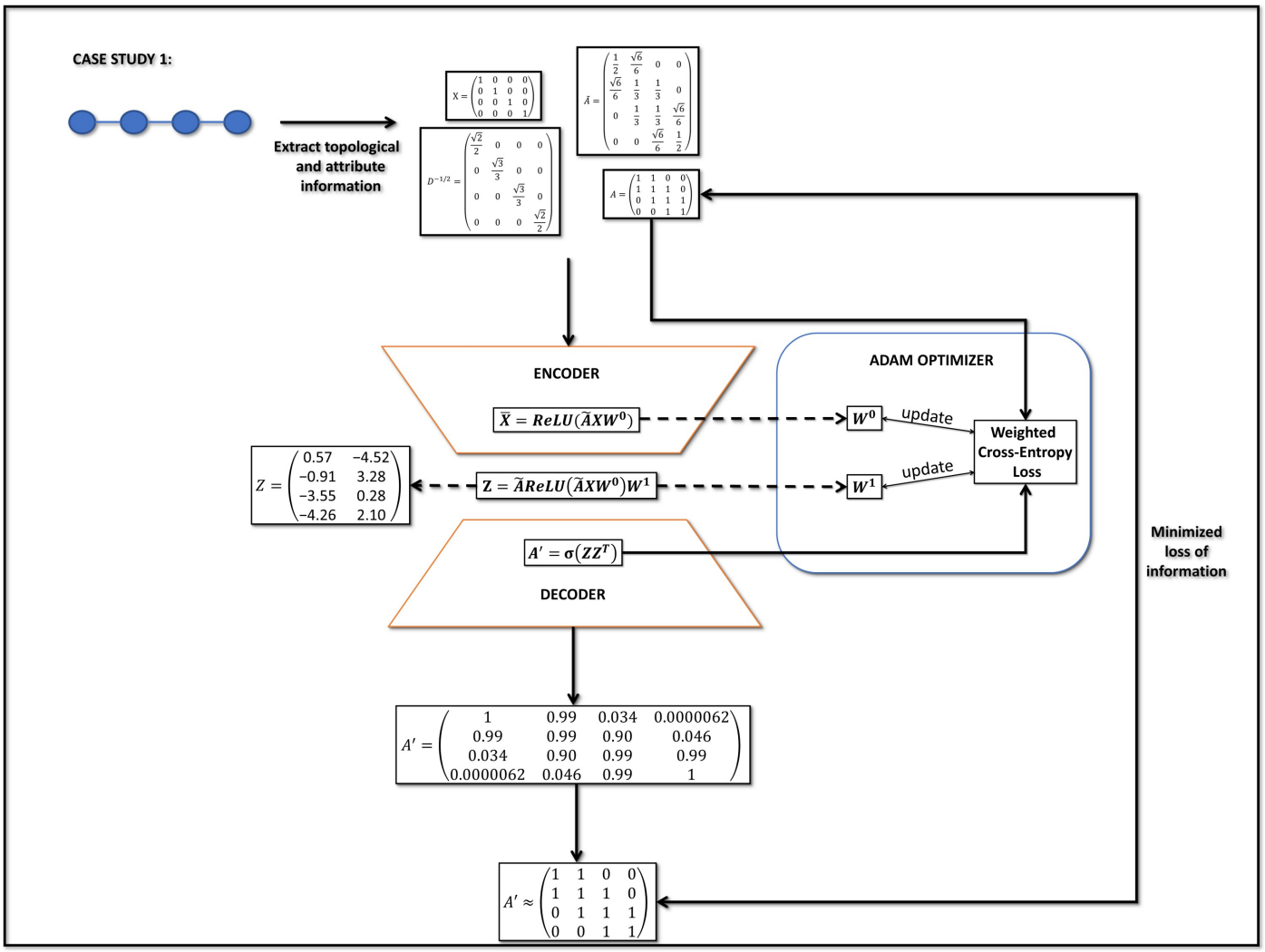

Figure 8. Flowchart of GAE* computations applied to Case Study 1.

composed by pipes (edges), nodes (junctions), pumps, valves, and storage tanks or reservoirs. During a simulation period, EPANET can track the flow of water in each pipe, the pressure at each node, the height of the water in each tank, the type of chemical concentration throughout the network, the age of the water, and source tracing. Various characteristics of a network element can be edited and a simulation can be performed to observe its effect on the overall system. For simplicity, we will limit the hydraulic parameter of interest in this study to node pressure. We obtain the pressure data by making two additional assumptions: (1) The data obtained through simulation does not involve any noise in it (i.e., the sensors are ideal), and (2) Water leakage is assumed to occur only at the junction nodes.

While EPANET does not explicitly support the modeling of leakages, its computational engine is demand-driven and, thus, leaks can be modeled as additional demand, and in a way that is independent of the pressure in a consumption node. The water output data at each node is defined as the base demand, and the demand can be increased at different times during the simulation. The network layout used to perform the hydraulic simulations in this work is shown in Figure 9, which from here on forth we will refer to as Case Study 2. The network contains 4 junctions, 4 links, a pump station, a water source, and a tank. The same network configuration was also used in our previous work [1].

ML training sets require data for a large number of positive (i.e., leaking) and negative (i.e., non leaking) cases, meaning that EPANET simulations also need to be performed for a large number of cases. A practical way of automating this process is with the EPANET-Python Toolkit, an open-source scripting software that interfaces with the latest version of EPANET.

With data from the hydraulic network simulations in hand, the next step is to perform the graph embedding. For this case study we are interested in obtaining a low-dimensional node vector representation for each node in the network. The hope is that the learned embeddings will advance various learning tasks, particularly leak detection by node classification. Applying GAE* to the pressure data outputted from EPANET simulation, yields 2-dimensional node embedding vectors for each node of Case Study 2. The next challenge to

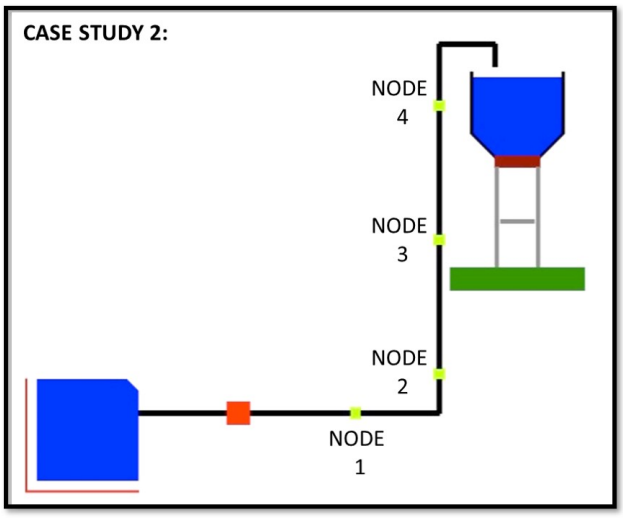

Figure 9. Elevation view of urban water distribution networks and junction (node) numbers used by EPANET simulation.

address is how to determine the optimal number of embedding dimensions? Since answers to this question are still an open research problem, we chose a set up for which the best topology reconstruction results were observed.

Topic 4. Node Classification

With the node embeddings obtained from GAE*, leakage detection can be performed. To be specific, we wish to identify the node or nodes where leakage has occurred during hydraulic simulation. To this end, the target function assigns a value of 1 to leaking nodes and a value of 0 to the non-leaking nodes. The embedding and target data are passed to a Random Forest classification algorithm. There are a number of good reasons to choose this pathway of analysis. First, the random forest classifier is composed of a number of decision trees. This problem solving strategy is well known to be accurate and robust. In addition, it does not suffer from the overfitting problem often encountered in other ML algorithms, since it takes the average of all the predictions of decision trees, and cancel out the biases. Random forests can also handle missing values, by using median values to replace continuous variables, or computing the proximity-weighted average of missing values. Finally, random forest provides a measurement of relative feature importance, which helps in the selection of contributing features for the classifier [19].

In this case study, we select a training data set that maximizes the expected variation in the target and environment, by building a hydraulic simulations where all of the nodes are leaking for half of the simulation duration, and for the other half of the duration none of the nodes are leaking. Since the simulation was set to last 24 hours, with pressure readings at every hour, the training set contains 24 cases for each node. The top portion of Figure 10 shows the plots for each node in Case Study 2, where the first of the embedding dimensions is plotted against time. Note that when a leak occurs at a specific time step, the embedding value for that dimension will also change; thus, this problem can be framed as anomaly detection.

In order to test the trained RF classifier, we generate a test set from a hydraulic simulation where none of the nodes are leaking for half of the simulation duration, and for the other half of the simulation duration only one of the nodes is leaking. In this use case we chose the leaking node in the testing set to be Node 3. Similar to the training set, the test set also contains 24 cases for each node. The bottom of Figure 10 shows the first of the embedding dimensions plotted against time. Notice that the embedding values change slightly compared to the previous scenario where all the nodes where leaking; therefore, the goal of the ML process is not only to detect the anomalies, but also identify which anomalies are actual leaks and which ones are just a propagation of the leak effects. Moreover, since the initial objective of this work is identification of leak location (and not timing of incipient leakage), leak durations are kept constant through all simulations. We do acknowledge, however, that the hour at which a leak occurs is relevant and future work will need to address variations in leakages in both space and time.

Topic 5. Results

As noted in the previous topic, the Random Forest classifier was trained on both leak and non leak data for each node. We were then able to test whether the classifier is able to detect a leak in the system. The test procedure involved feeding the node embeddings for a scenario where initially none of the nodes were leaking, and later only one node was leaking (node 3). As classification problems are perhaps the most common type of ML problem, there are a myriad of metrics that can be used to evaluate outputted classifications, the most common being classification accuracy (i.e., the ratio of number of correct predictions to the total number of input samples). After training and testing an RF classifier, we obtained a classification accuracy of 100 percent (i.e., the classifier correctly predicted the location of test leaks in the network). We recognize such a high performance may be due to several simplifications made in this experiment. First the network maintained a constant base demand for each node through the hydraulic simulation, when in reality WDSs experience changes in water demand depending on time of day. Second, the decoder component of GAE* framework successfully reconstructed network topologies in case study 1 and 2, and consequently generated accurate embeddings representations of the networks, in part because the topologies were relatively simple and static. An unresolved question from these initial experiments is if the same success can be obtained for larger or dynamic topologies. Section V discusses the effects of size on GAE* performance. Third, we assumed sensor nodes were deployed in each junction of the network; however, in our resource limited world, placing as many sensors as there are junction nodes to monitor all of the nodes in real time is extremely infeasible. In Section VI we discuss ideas for addressing these simplification issues in future work.

V. Discussion

The automated identification of leaks in WDSs depends critically on accurate representations of network topology and

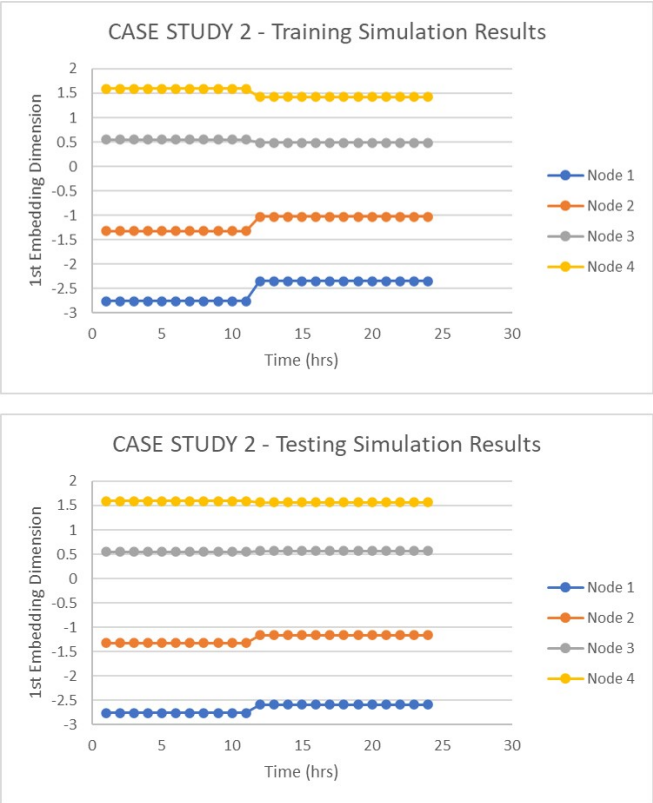

Figure 10. Case study 2 node embeddings (1st dimension) obtained for train and test sets plotted against time.

node attribute information. The embedding obtained by using GAE* framework is fed to downstream classifiers that will determine presence of leaks in the system. If the embedding is not accurate enough, errors could cascade across the fault detection mechanism and lead to false positive or false negative leakage identifications. Therefore, it is important to understand the limitations of GAE* in generating a representation of the network at hand. This section discusses these limitations and potential solutions.

A. GAE* Limitations

Our results have indicated that when network topologies are simple, GAE* is capable of providing network embeddings that can be decoded to perfectly reconstruct the input network. One important research question that remains open is the effect of network size on GAE* learning performance. In order to address this concern, we have attempted to reconstruct three attributed WDSs of varying sizes using GAE*. The goal of this experimental study was to learn for what ranges of embedding dimensions the algorithm was able to reconstruct the input network accurately. In order to measure the accuracy of the reconstruction, the graph edit distance (GED) between the input graph and the reconstructed graph was used. GED is a graph similarity measure defined as the minimum cost of a sequence of node and edge edit operations transforming the reconstructed graph into the input graph. A GED value of zero shows that the reconstructed graph is isomorphic to the input graph. The higher the GED value, the higher the dissimilarity between the reconstructed graph and the input graph. In order to maintain a sense of GED importance for different graph sizes, the GED values obtained in our experiments were then normalized by the size of the input graph.

Figure 11 summarizes the results of the numerical experiments, and shows that if the neural network used to train the GAE* has an insufficient number of neurons, then the reconstruction performance is poor. This makes perfect sense. But when the same network is modeled with too many neurons, the reconstruction performance is also poor. The results suggest that there is a window of convergence where good reconstruction can be achieved. For the networks with 6 and 11 nodes the algorithm is able to perfectly reconstruct (i.e., GED is zero) certain network configurations, but for the network with 36 nodes the algorithm is not able to reach perfect reconstruction. From Case Study 1, we now understand the inner architecture of GAE* and what limitations it may encounter, as well as the ways in which it might be changed to adapt to specific problems. To obtain a good reconstruction of the input graph, the underlying algorithm needs to work. In turn, the latter depends on: (1) convergence of the Adam optimization, (2) the encoder architecture design, and (3) the decoder architecture design. We suspect that the phenomena observed in Figure 11 may be related to limitations in these three GAE* components.

A. Convergence of the Adam Optimizer

One of the key hyper-parameters to set in order to construct an Adam optimizer is the learning rate. This parameter scales the magnitude of our weight updates in order to minimize the network's loss function. If the learning rate is set too low, then training will progress very slowly as we are making very small updates to the weights values. However, if your learning rate is set too high, then it can cause undesirable divergent behavior in the loss function (i.e., the gradient of the weight oscillates back and forth, and it is difficult to make the loss reach the global minimum).

B. Encoder Architecture Design

Although Kipf et al. used 2 hidden layers, 32 neurons in the first hidden layer, and 16 neurons in the second hidden layer when presenting the GAE* architecture [24], it is possible to modify these characteristics of the framework in order to adapt to specific needs. When considering the structure of GAE*, there are really two decisions that must be made: how many hidden layers to actually have in the neural network, and how many neurons will be in each of these layers? Prior to deep learning, problems that required more than two hidden layers were rare. Two or fewer layers will often suffice with simple data sets. However, with complex datasets additional layers can be helpful. According to Heaton a neural network with no hidden layers is only capable of representing linear separable functions or decisions, a network with one hidden layer can approximate any function that contains a continuous mapping from one finite space to another, and a network with two hidden layers can represent an arbitrary decision boundary to arbitrary accuracy with rational activation functions and can approximate any smooth mapping to any accuracy. And

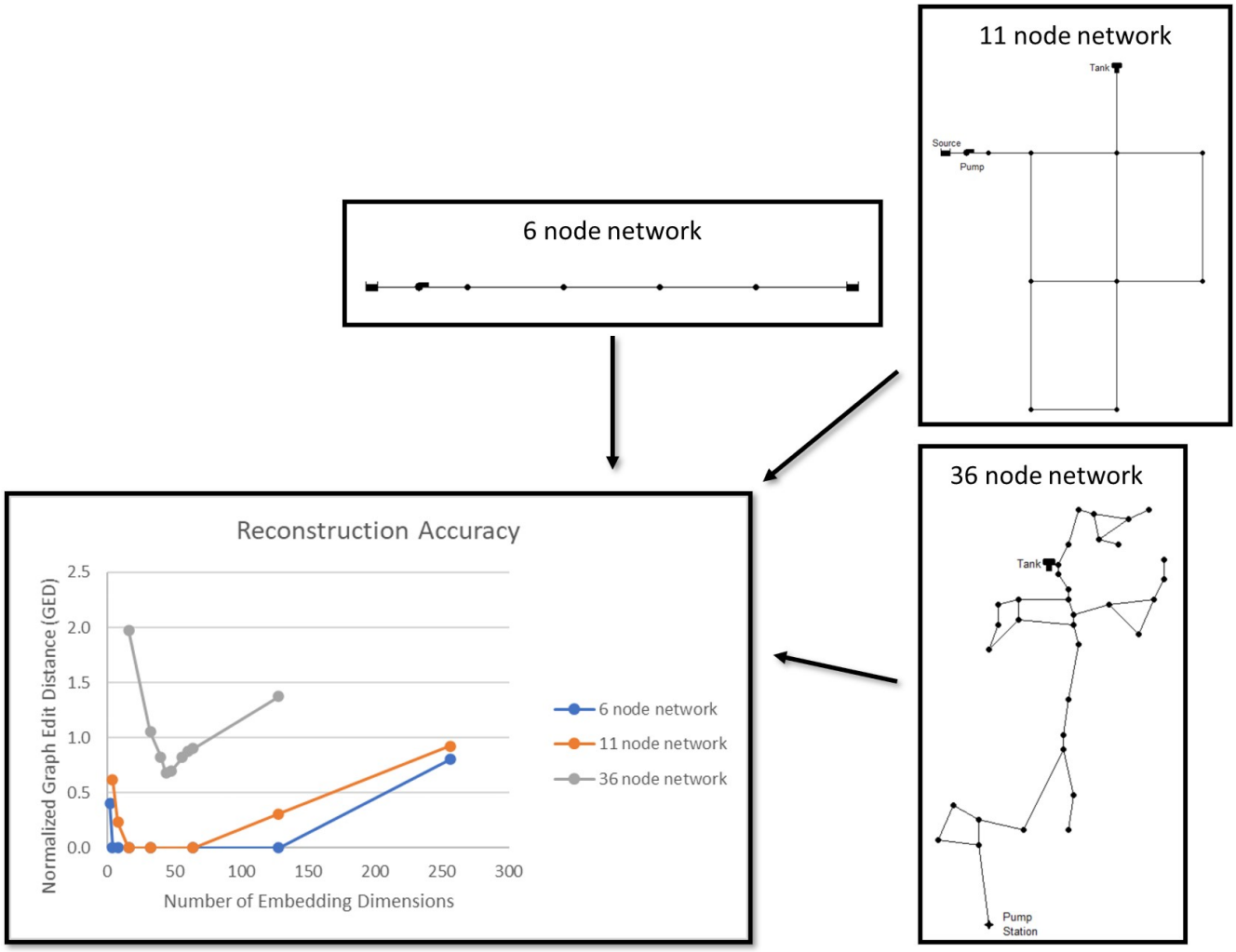

Figure 11. GAE* reconstruction accuracy for networks of varying sizes.

a network with more than two layers can learn complex representations (i.e., feature engineering) for layered layers [27]. For our purposes, we are interested in investigating whether adding more layers to GAE* algorithm can bring benefits (i.e., faster convergence, lower loss, fewer nodes, etc.). Deciding the number of hidden layers is only a small part of the problem, we must also determine how many neurons will be in each of these hidden layers. Using too few neurons in the hidden layers can result in underfitting. Underfitting occurs when there are too few neurons in the hidden layers to adequately process information in a complicated data set. Too many neurons on the other hand can result in overfitting. Overfitting occurs when the neural network has so much information processing capacity that the limited amount of information contained in the training set is not enough to train all of the neurons in the hidden layers [27]. Computational requirements also need to be taken into account. Generally speaking, a large number of neurons in the hidden layers will increase the time it takes to train the network, possibly to a point where it is computationally infeasible to adequately train the neural network [27]. Obviously, some compromise must be reached between too many and too few neurons in the hidden layers. Ultimately, the selection of an architecture for our neural network will come down to avoiding underfit, overfit and convergence issues. Learning curves are widely used in machine learning for algorithms that learn (i.e., optimize their internal parameters) incrementally over time. The shape and dynamics of a learning curve can be used to diagnose the behavior of a machine learning model and, in turn, suggest changes that may be made to improve learning and/or performance. For our purposes, we are interested in investigating how learning curves can help us adapt the GAE* architecture to avoid the abovementioned issues.

C. Decoder Architecture Design

Much of the node embedding research work performed in recent years has focused on refining and improving the encoder architecture. Most methods still rely on basic pairwise decoders (e.g., inner-product decoders), which predict pairwise relations between nodes and ignore higher-order graph structures involving more than two nodes [17]. Higher-order structural sub-graphs are essential to the structure and function of complex networks. Hence, testing the performance of decoding algorithms other than inner-product decoders will

be an important step in unveiling GAE* limitations.

VI. Conclusions and Future Work

The long term objective of this research is to understand how ML and semantic-modeling can work hand-in-hand to build digital twins architectures that enhance the collection of data, identify events, and manage operations of systems in an urban context. In this paper, we started by exploring the recently developed GAE* framework in Case Study 1, where a simple graph was encoded to a low-dimensional vector while minimizing the loss information in the encoding process. Then, we investigated the possibility of using a GAE* generated embedding as input for a RF classier trained to identify leaks in WDSs in Case Study 2. Finally, we examined the effects of network size on learning performance by generating embeddings for larger size networks. Although procedures for encoding network information (i.e., structure and attributes) with the GAE* framework, and using RF classifier to identify water leaks have shown to be successful, the work reported in this paper takes an initial step toward exploring potential applications of ML to the identification of leaks in urban networks. Many questions and issues remain either unanswered or unresolved. Opportunities for future work include varying base node demands according to time of day, experimenting with more complex topologies and topologies that are dynamic (i.e., edges are removed or created), investigating how many sensor nodes are needed and where to place them in the network, exploring the use of learning curves in finding the optimal number of neurons, layers, and learning rate values for optimizing learning performance, and exploring different decoder architectures with the aim of improving reconstruction results for more complex graphs.

VII. Acknowledgment

The first and second authors were supported by a grant from the Systems Engineering Research Center/Office of the Under Secretary of Defense for Research and Engineering.

References

[1] M. Coelho and M.A. Austin, "Teaching Machines to Understand Urban Networks," The Fifteenth International Conference on Systems (ICONS 2020), February 23-27 2020, pp. 37–42.

[2] E.H. Glaessgen and D.S. Stargel, "The Digital Twin Paradigm for Future NASA and U. S. Air Force Vehicles," in 53rd AIAA/ASME/ASCE/AHS/ASC Struct. Struct. Dyn. Mater. Conf., 2012.

[3] A. Fuller, Z. Fan, C. Day, and C. Barlow, "Digital Twin: Enabling Technologies, Challenges and Open Research," IEEE Access, vol. 8, June 2020, pp. 108 952–108 971.

[4] M. Coelho, M.A. Austin, and M.R. Blackburn, "Distributed System Behavior Modeling of Urban Systems with Ontologies, Rules and Many-to-Many Association Relationships," The Twelth International Conference on Systems (ICONS 2017), April 23-27 2017, pp. 10–15.

[5] ——, "Semantic Behavior Modeling and Event-Driven Reasoning for Urban System of Systems," International Journal on Advances in Intelligent Systems, vol. 10, no. 3 and 4, December 2017, pp. 365–382.

[6] ——, The Data-Ontology-Rule Footing: A Building Block for Knowledge-Based Development and Event-Driven Execution of Multi-Domain Systems. Proceedings of the 16th Annual Conference on Systems Engineering Research, Systems Engineering in Context, Chapter 21, Springer, 2019, pp. 255–266.

[7] P. Delgoshaei, M. Heidarinejad, and M.A. Austin, "Combined Ontology-Driven and Machine Learning Approach to Monitoring of Building Energy Consumption," in 2018 Building Performance Modeling Conference and SimBuild, Chicago, IL, September 26-28 2018.

[8] J. Abraham, "Semantic Foundations for Formalizing Brain Cancer Profiles," MS Thesis, Master of Science in Systems Engineering Program, University of Maryland, College Park, MD 20742, 2019.

[9] I.H. Witten, E. Frank, M.A. Hall, and J.P. Christopher, Data Mining: Practical Machine Learning Tools and Techniques. Morgan Kauffmann, 2017.

[10] H. Cai, V. W. Zheng, and K. C. Chang, "A Comprehensive Survey of Graph Embedding: Problems, Techniques and Applications," IEEE Transactions on Knowledge and Data Processing, vol. 30, no. 9, 2018, pp. 1616–1637.

[11] M.A. Austin, P. Delgoshaei, M. Coelho, and M. Heidarinejad, "Architecting Smart City Digital Twins: A Combined Semantic Model and Machine Learning Approach," Journal of Management in Engineering, ASCE, vol. 36, no. 4, 2020.

[12] J. Li et al., "Attributed Network Embedding for Learning in a Dynamic Environment," in Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, ser. CIKM '17. New York, NY, USA: ACM, 2017, pp. 387–396. [Online]. Available: http://doi.acm.org/10.1145/3132847.3132919

[13] J. Kong, S. Simonovic, and Z. Chao, "Resilience Assessment of Interdependent Infrastructure Systems: A Case Study Based on Different Response Strategies," Sustainability, vol. 11, November 2019, p. 6552.

[14] S. Strogatz, "Exploring Complex Networks," http://www.nature.com/nature/journal/v410/n6825/pdf/410268a0.pdf, vol. 410, March 2001.

[15] P. Goyal and E. Ferrara, "Graph Embedding Techniques, Applications, and Performance: A Survey," CoRR, vol. abs/1705.02801, 2017. [Online]. Available: http://arxiv.org/abs/1705.02801

[16] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P.S. Yu, "A Comprehensive Survey on Graph Neural Networks," CoRR, vol. abs/1901.00596, 2019.

[17] W. L. Hamilton, R. Ying, and J. Leskovec, "Representation Learning on Graphs: Methods and Applications," CoRR, vol. abs/1709.05584, 2017. [Online]. Available: http://arxiv.org/abs/1709.05584

[18] L.A. Rossman L.A., "EPANET 2: Users Manual," Water Supply and Water Resources Division National Risk Management Research Laboratory Cincinnati, OH 45268, Tech. Rep. EPA/600/R-00/057, September 2000.

[19] L. Breiman, "Random Forests," in Machine Learning, vol. 45, no. 1. Norwell, MA, USA: Kluwer Academic Publishers, October 2001, pp. 5–32.

[20] J.J. Pfeiffer, S. Moreno, T. La Fond, J. Neville, and B. Gallagher, "Attributed Graph Models: Modeling Network Structure with Correlated Attributes," in Proceedings of the 23rd International Conference on World Wide Web, ser. WWW '14. New York, NY, USA: ACM, 2014, pp. 831–842. [Online]. Available: http://doi.acm.org/10.1145/2566486.2567993

[21] S. Cao, W. Lu, and Q. Xu, "Deep Neural Networks for Learning Graph Representations," 2016, In AAAI.

[22] D. Wang, P. Cui, and W. Zhu, "Structural Deep Network Embedding," in Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ser. KDD '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 1225–1234.

[23] T.N. Kipf and M. Welling, "Semi-Supervised Classification with Graph Convolutional Networks," CoRR, vol. abs/1609.02907, 2016.

[24] ——, "Variational Graph Auto-Encoders," ArXiv, vol. abs/1611.07308.

[25] D. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," International Conference on Learning Representations, December 2014.

[26] J. Mashford, D. Silva, S. Burn, and D. Marney, "Leak Detection in Simulated Water Pipe Networks using SVM," Applied Artificial Intelligence, vol. 26, May 2012, pp. 429–444.

[27] J. Heaton, Artificial Intelligence for Humans, Volume 3: Deep Learning and Neural Networks, ser. Artificial Intelligence for Humans. Createspace Independent Publishing Platform, 2015. [Online]. Available: https://books.google.com/books?id=q9mijgEACAAJ