



**WRT 1025:**

# **ARCHITECTING FOR DIGITAL TWINS AND MCE WITH AI/ML PART II**

Principal Investigator:

Dr. Mark Blackburn, Stevens Institute of Technology

Co-Principal Investigator:

Dr. Mark Austin, University of Maryland

Final Technical Report SERC-2021-TR-007

April 21, 2021



**SYSTEMS**  
**ENGINEERING**  
RESEARCH CENTER

The Networked National Resource to further systems research  
and its impact on issues of national and global significance



## **Architecting for Digital Twins and MCE\_with\_Part\_II**

**A013 Final Technical Report SERC-2021-TR-007**

**April 21, 2021**

### **Principal Investigator**

Dr. Mark Blackburn, Stevens Institute of Technology

### **Co-Principal Investigator**

Dr. Mark Austin, University of Maryland

### **Research Team**

**University of Maryland:** Maria Coelho

### **Sponsors**

OUSD (R&E)



Castle Point on Hudson, Hoboken, NJ 07030

Copyright © 2021 Stevens Institute of Technology, Systems Engineering Research Center

This material is based upon work supported, in whole or in part, by the U.S. Department of Defense through the Systems Engineering Research Center (SERC) under Contract HQ0034-19-D-003 (Task Order 0201, WRT-1025). SERC is a federally funded University Affiliated Research Center managed by Stevens Institute of Technology

Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Department of Defense.

#### NO WARRANTY

THIS STEVENS INSTITUTE OF TECHNOLOGY AND SYSTEMS ENGINEERING RESEARCH CENTER MATERIAL IS FURNISHED ON AN "AS-IS" BASIS. STEVENS INSTITUTE OF TECHNOLOGY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. STEVENS INSTITUTE OF TECHNOLOGY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.

This material has been approved for public release and unlimited distribution.

# Table of Contents

<b>1</b>	<b>Introduction.....</b>	<b>8</b>
1.1	Objectives.....	10
1.2	Approach and Case Study .....	11
1.3	Scope .....	14
1.4	Summary Status of Accomplishments .....	14
1.5	Organization of Document .....	15
<b>2</b>	<b>Background and Context.....</b>	<b>16</b>
<b>3</b>	<b>Research Overview .....</b>	<b>17</b>
3.1	Historical Progression .....	18
3.2	Graphs Representations and Analytics .....	19
3.3	Controlled Behaviors on Spatial-Temporal Graphs .....	21
<b>4</b>	<b>Semantic Modeling and Reasoning with Graphs .....</b>	<b>23</b>
4.1	Graph Types and Draft Graph Ontology. ....	24
4.2	Graph Analysis and Semantic Rules for Graphs .....	26
<b>5</b>	<b>Teaching Machines to Understand Graphs .....</b>	<b>27</b>
5.1	Machine Learning of Large-Scale Graphs .....	28
5.2	Graph Auto Encoders (GAE) .....	29
5.3	Teaching Machines to Understand Graphs with Graph Autoencoder .....	30
5.4	Convergence of the Adam Optimizer .....	33
5.5	Encoder Architecture.....	33
5.6	Neural Networks and Graph Autoencoder.....	34
5.6.1	Line Topology .....	35
5.6.2	Ring Topology.....	36
5.6.3	Fully Connected (FC) Topology .....	37
5.6.4	Star Topology .....	38
5.6.5	Mesh Topology.....	40
5.6.6	Tree Topology .....	41
5.7	Urban Topology .....	43
5.8	GAE Architecture and the Optimization Algorithm .....	45
5.9	The Hessian and the Optimization Algorithm.....	46
5.10	Neural Network Architecture for Classification .....	46
<b>6</b>	<b>Deliverables and Events .....</b>	<b>50</b>
<b>7</b>	<b>Summary .....</b>	<b>50</b>
7.1	Next Steps .....	51
<b>8</b>	<b>Acronyms and Abbreviation .....</b>	<b>53</b>
<b>9</b>	<b>Trademarks.....</b>	<b>53</b>
<b>10</b>	<b>References .....</b>	<b>55</b>

## Figures

Figure 1. Digital twin (cyber physical) working alongside of a physical twin .....	11
Figure 2. Architectural template for the construction of digital twins. Box 1: Semantic modeling, Box 2: machine learning/data mining, Box 3: Machine learning/network modeling. ....	12
Figure 3. Schematic of a digital thread framework supporting flows of data and integration of viewpoints across the system lifecycle.....	13
Figure 4. Emergence of Digital Twin Era, a replacement for MBSE with SysML .....	16
Figure 5. Annotated traffic intersection at the entrance to UMD.....	17
Figure 6. Neural Network Autoencoder Use to Understand Structure and Features of Graph...	18
Figure 7. Schematic of traditional (adjacency matrix) and machine learning (graph embedding) approaches to modeling graphs. ....	20
Figure 8. Auto-encoder design for link prediction and deep graphs.....	21
Figure 9. Spatio-temporal modeling of a vehicle making a left-hand turn at a traffic intersection .....	22
Figure 10. Synthesis of graph data models from pathways and controls in Open Street Map....	23
Figure 11. Framework for Multi-Domain Semantic Modeling.....	24
Figure 12. Classification of graph types + draft of graph ontology mirroring architecture of JGraphT software .....	25
Figure 13. Shortest path analysis in a directed weighted graph. Top: Minimum weighted cost pathway for A -> L. Bottom: Revised pathway analysis when edges in graph are obstructed by a weather event. ....	27
Figure 14. Traditional encoder-decoder approach.....	29
Figure 15. Process flowchart for training and executing machine .....	30
Figure 16. Flowchart of GAE computations applied to Case Study. ....	31
Figure 17. Effects of learning rate on optimization convergence (“Python Lessons”).....	33
Figure 18. Common system topologies. ....	34
Figure 19. Six node line topology with node attributes.....	35
Figure 20. Learning curve for line topology with one hidden layer and one neuron architecture .....	35
Figure 21. Learning curve for line topology with one hidden layer and two neurons architecture. ....	35
Figure 22. Six-node ring topology with node attributes.....	36
Figure 23. Learning curve for ring topology with one hidden layer with one neuron architecture. ....	36

Figure 24. Learning curve for ring topology with one hidden layer with two neurons architecture.....	37
Figure 25. Six node FC topology with node attributes. ....	37
Figure 26. Learning curve for FC topology with one hidden layer with one neuron architecture. ....	37
Figure 27. Six node star topology with node attributes. ....	38
Figure 28. Learning curve for star topology with one hidden layer with one neuron architecture. ....	38
Figure 29. Learning curve for star topology with one hidden layer with two neurons architecture.....	38
Figure 30. Learning curve for star topology with one hidden layer with three neurons architecture.....	39
Figure 31. Learning curve for star topology with one hidden layer with four neurons architecture.....	39
Figure 32. Learning curve for star topology with one hidden layer with five neurons architecture.....	40
Figure 33. Six-node mesh topology with node attributes. ....	40
Figure 34. Learning curve for mesh topology with one hidden layer with one neuron architecture.....	40
Figure 35. Learning curve for mesh topology with one hidden layer with two neurons architecture.....	41
Figure 36. Learning curve for mesh topology with one hidden layer with three neurons architecture.....	41
Figure 37. Six-node tree topology with node attributes. ....	41
Figure 38. Learning curve for tree topology with one hidden layer with one neuron architecture. ....	42
Figure 39. Learning curve for tree topology with one hidden layer with two neurons architecture.....	42
Figure 40. Learning curve for tree topology with one hidden layer with three neurons architecture.....	43
Figure 41. Urban topology with node attributes.....	44
Figure 42. Learning curve for urban topology graph and one hidden layer containing 74 nodes. ....	45
Figure 43. An Introduction to Computing with Neural Nets .....	47
Figure 44. Two Class Classification using XOR Example.....	48

Figure 45. Decision Planes .....	48
Figure 46. Representation of Neural Network Architecture .....	49
Figure 47. Representation of Classification for Adjacency Matrix .....	49

## Tables

No table of figures entries found.



## Executive Summary

This is the final technical report of the Systems Engineering Research Center (SERC) research task WRT-1025. This research investigated digital twin design architectures that support Artificial Intelligence (AI) and Machine Learning (ML) formalisms working side-by-side as a team. The unique aspects of this research are the use semantic technologies that can leverage AI and ML providing complementary and supportive roles in the collection, formalizing representations and processing of data, identification and correlation of events, in evolving spatial contexts and automated decision making throughout the system lifecycle. The research developed graph embedding procedures with ML tasks, which together can enhance digital twin design and decision making to factor in evolving temporal and spatial information, such as those encountered in urban settings.

This research builds upon our previous work on teaching machines to understand urban networks with graph analytics techniques. These advances are due in part, to advances in computer, communications and sensing technologies that evolved over the past three decades in large-scale urban systems and 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 create challenges now that we have opportunities to better integrate mission engineering, system design, analysis and integration of multi-disciplinary concerns. We have made progress against these challenges by teaching machines to understand graphs that represent urban networks. This report discusses research efforts and accomplishments for using a recently developed graph autoencoding approach to encode the structure and associated network attributes as low-dimensional vectors. We successfully demonstrated the approach on a problem involving identification of leaks in urban water distribution systems.

We have made unique progress and have been able to share work in SERC events such as the workshop on AI4SE and SE4AI. However, there is still more research needed. We would propose to investigate ways for automating cluster identification, and test the scalability of our approach (i.e., test larger graphs). In addition, several objectives from previous reports are still pending such as exploring composition (i.e., learning graph topology in parts), decoder architectures, and linking models of graph topology to models of system behavior and identification of events, among others. Therefore, needed investigations should consider the following set of basic questions, such as does graph composition help alleviate the challenges posed by larger graph sizes and complexity?

## 1 INTRODUCTION

---

The increased use of models such as descriptive models for mission and systems engineering is providing additional rigor to characterize digital representations of mission and systems that link to cross-domain discipline-specific models. Semantically rich representations are needed for all of these type of model elements in characterizing a digital twin. Knowledge representation plays a key role in applying Artificial Intelligence (AI). Ontologies and associated semantic technologies provide a means to domain modeling and reasoning that are needed across the domains of a digital thread instantiated in digital system models (DSM). These DSMs evolve over time as digital twins, which co-evolve with physical instantiations of a DSM. Our prior research has used ontologies and semantic technologies to formalize knowledge with interoperable ontologies enabling reason about systems engineering across domains [5][7][13][17][19][20][21] [33][50].

There is also a wealth of data enabled by an explosion of sensors (e.g., Internet-of-Things), real-time monitoring and synchronization of data associated with events of interacting capabilities within evolving environments to understand physical systems throughout their lifecycle that can be factored back into digital twins. Machine learning (ML) techniques are used for classification, clustering, and identification of association relationships. AI and ML technologies will be deeply embedded in new methods and tools for model-centric engineering (MCE), as well as new digital twin operating system environments for observation, reasoning and physical systems control. Realization of this opportunity is complicated by the reality that within the world of MCE, present-day use of AI and ML technologies is fragmented. However, we are at a crossroads for leveraging AI and ML enabling technologies in the broader context of MCE.

This research started to develop rules associated with ontologies for knowledge representation, and consideration of the ways in which ontologies and rules can work together to respond to sequences of events about interacting objects and agents in evolving spatial contexts (e.g., environments) to support decision making. Formalization of MCE models provides a foundation for extending with ontologies and rules that can integrate cross-domain information; this type of AI provides for logic and reasoning [16]. We also need complementary ML techniques for classification, clustering, and identification of association relationships, remembering the details of data streams, and finding anomalies in behavior enabled by rich data sets. Recent advances have created ML algorithms to learn the structure of large-scale graphs and their attributes. An objective is to overcome limits in ML techniques that struggle to explain the rationale for decision making, by integrating multi-domain semantic modeling with rule-based reasoning and providing a means for representations of sequences of objects and events over time. Our brains do this fairly well (in addition to dealing with uncertainty), but we want to computationally enable these types of capability for MCE by characterizing AI/ML design patterns that support architecting effective digital twins by bringing AI and ML together in a new way [5].

This report 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 [2] on teaching machines to understand urban networks with graph analytics techniques.

A digital twin is a digital representation of a system that mirrors its implementation in the physical world through simulation and real-time monitoring and synchronization of data associated with events [6]. The associated software and algorithms work to provide superior levels of attainable performance in system development and operation. The digital twin concept dates back to the 2000-2010 era; it was initially proposed as a way to support the design and operation of air vehicles for NASA. Since then, the range of potential applications has expanded to include automotive components, manufacturing processes, personalized medicine and smart cities, among others. Within the world of model-centric engineering, there is strong need for support throughout the entire systems lifecycle. If successful, AI and ML technologies will be deeply embedded in new methods and tools for model-centric engineering, as well as new digital twin operating system environments for observation, reasoning and physical systems control [5].

Realization of this opportunity is complicated by the reality that, within the world of model-centric engineering, present-day use of AI and ML technologies is fragmented and at a crossroads [5]. During that past decade, systems engineering researchers in AI have tended to focus on:

- The comprehensive development of ontologies for a domain, (e.g., human genome, satellites)
- System development activity (e.g., requirements, behavior modeling)
- Extension of development activities from common core ontologies (e.g., extensions for geospatial, time, actors, events,)
- Higher-level basic formal ontologies [4]

Far less attention has been given to the development of rules associated with ontologies, and consideration of the ways in which ontologies and rules can work together to respond to events and support decision making. At the same time, machine learning techniques provide comprehensive support for the classification, clustering, and identification of association relationships and anomalies in streams of real-world data. Remarkable advances in machine learning algorithms (2016-2019) include the ability of a machine to learn the structure of large-scale graphs and their attributes. The consequences of this recent capability for the model-based systems engineering community would appear to be enormous [16]. And yet, machine learning techniques struggle to explain the rationale for decision making, which is a task that multi-domain semantic modeling and rule-based reasoning can complete with ease.

The key research challenge is how to design the digital twin elements and their interactions so that collectively they can support a wide variety of systems engineering methods and processes. Previous University of Maryland research focused on semantic foundations and reasoning for two application areas, energy-efficient buildings and brain cancer profiles [24]. In both cases, ML techniques for classification/clustering of data provided useful feedback on the structuring of ontologies for semantic reasoning. While this research showed promise, our current methods are far from what seems possible. Additional research is needed to understand the range of possibilities for which machine learning of large-scale graphs and their attributes can support activities in model-centric engineering. This knowledge will be used to guide the architectural development of future digital twins enabled by AI-ML technology. Fundamental questions include: Can we develop design methods to enable machine learning to improve semantic

modeling? What design methods can enable semantic modeling to improve machine learning? How do we employ these methods in the context of model-centric engineering and digital twins?

This research provides progress to help us understand how to design the digital twin elements and their interactions so that collectively they can support two purposes:

- Development of methods and tools for model centric engineering, and
- Development of digital twin operating system environments for observation, reasoning and system control.

We believe that the knowledge gained provides some steps to be used to guide the architectural development of future digital twins and threads enabled by AI-ML technology.

---

## 1.1 OBJECTIVES

At the start of this research task, the primary near- and long-term objectives were to:

- Develop basic mechanisms for semantic / machine learning interaction. Investigate opportunities for semantics to broaden the scope of understanding an ML processor will have of a domain. Conversely, investigate methods for ML to assist the semantic modeling through recording of past scenarios and pathways of decision making, and increases in efficiency of decision making.
- Teach machines to understand small graphs having static graph topologies. Develop new approaches for graph modeling and analysis that combine semantic models with graph structure in ways that are computationally efficient.
- Develop methods for auto-encoding designs in system graph representations. Develop encoder and decoder approaches that preserve the overall system graph.
- Identify events via time-series anomaly detection using dynamic attribute network embedding (DANE) or other approaches. Show how these approaches would be integrated with dynamic simulation.
- Integrate simulation and machine learning approaches. Operational digital twins are mobile and need to make decisions to deal with events in the right place and the right time. The research shall investigate machine learning capability for reasoning with events, space and time, and linking these capabilities to simulations of military interest.
- Develop a set of heuristics and design principles to use AI-ML approaches to develop digital twins and model-centric engineering.
- Develop a demonstration showcasing application of the new AI-ML design methods. The demonstration should involve at least two digital twins being used in different environments. If possible, the demonstration should be based on information relevant to one of the SERC's Service sponsors for research in model-centric engineering.
- Develop a set of future research challenges based on knowledge accumulated of the course of the previous year's research.

## 1.2 APPROACH AND CASE STUDY

This one-year project focused on the needs for the design of digital twins that work as operating systems, with AI and ML formalisms working side-by-side as a team providing complementary and supportive roles in the collection of data, identification (or prediction) of events, and support for automated decision making throughout the system lifecycle. We specifically looked to understand how digital twin elements and their interactions should be designed so that collectively they can support two purposes: (1) Development of methods and tools for model-centric engineering, and (2) Development of digital twin operating system environments for observation, reasoning and systems control.

Figure 1 shows the essential details of a digital twin architecture for the proposed approach. We envision digital twins working as data- and event-driven operating systems with semantic models responsible for knowledge representation and reasoning, and machine learning responsible for prediction of likely demands on the system, learning the structure and sequencing for various representations, identification of objects / events, and remembering scenarios from past activities.

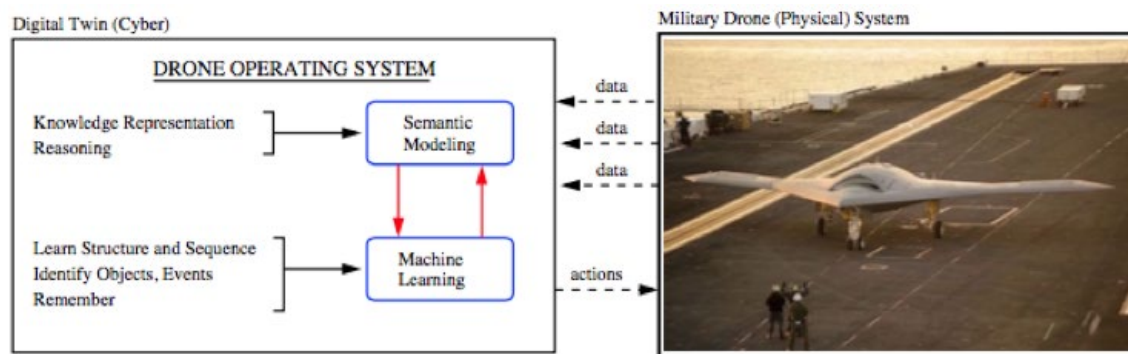
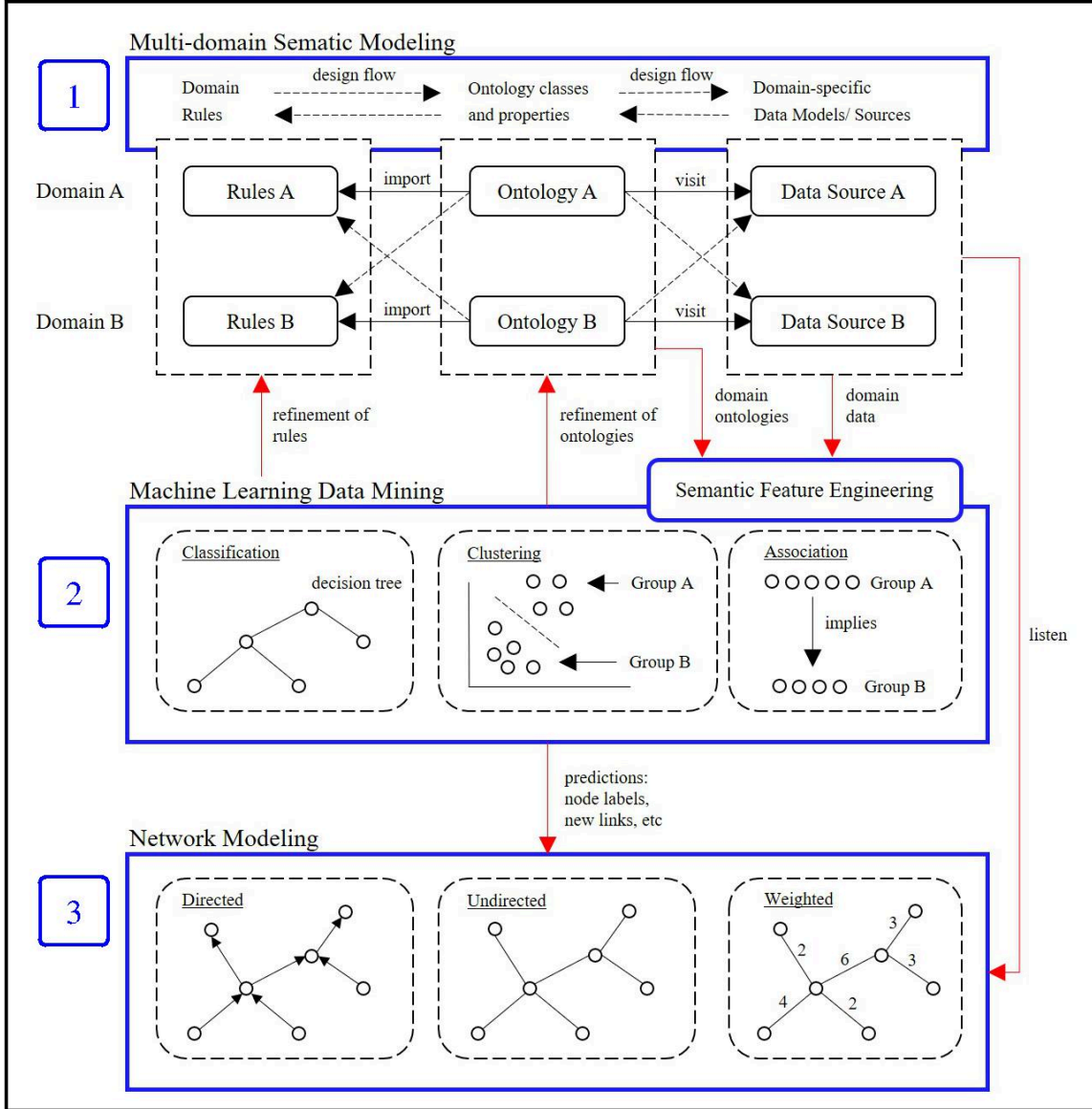


Figure 1. Digital twin (cyber physical) working alongside of a physical twin

Figure 2 is an extension of Figure 1 and shows the proposed architectural template for a combined multi-domain semantic modeling and machine learning approach to the implementation of digital twin applications.



**Figure 2. Architectural template for the construction of digital twins. Box 1: Semantic modeling, Box 2: machine learning/data mining, Box 3: Machine learning/network modeling.**

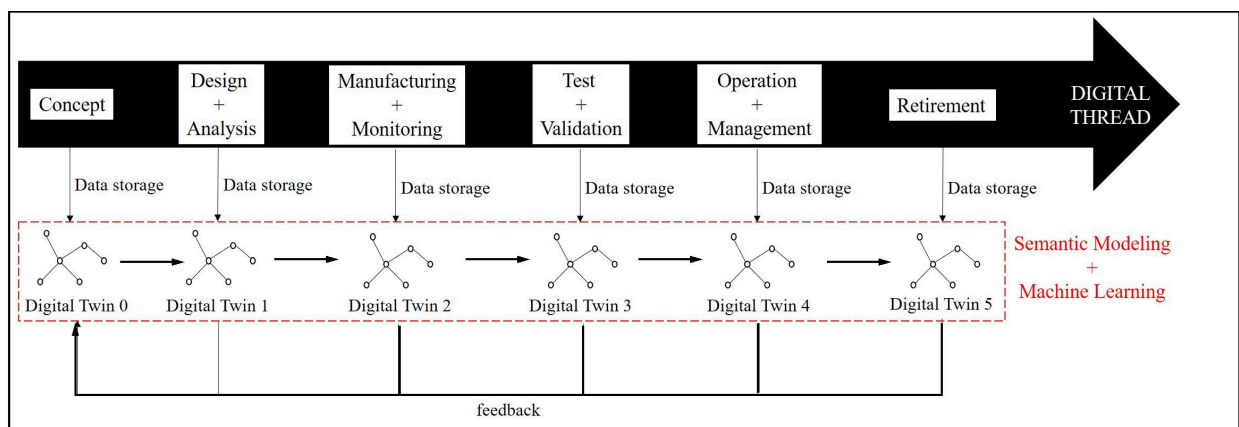
Semantic modeling is concerned with the development of knowledge representations (ontologies) for multi-domain applications, population of semantic graphs with data (so-called individuals), and development of rules to transform semantic graphs in response to events and to ensure consistency and completeness of provided data [33]. Semantic graph models persist throughout the system lifecycle and, thus, are central to digital system models (DSMs) and digital threads. Instead of creating a small number of all-encompassing ontologies and associated rules, our goal [19][20] is to put the development of data, ontologies and rules on an equal footing (see Box 1 of Figure 2), and create architectural templates for a specific domain or design concern (a convenient name is the data-ontology-rule footing). Machine learning techniques are concerned with learning the structure and sequence of various representations, identifying objects and

events, remembering the details of data streams, and finding anomalies in behavior enabled by rich data sets.

Our proposed architectural template (see Box 2 of Figure 2), employs data mining/machine learning techniques for three styles of learning – tree-based classification, clustering algorithms, association algorithms – to gain insight into the data [67]. The final part of the proposed template (see Box 3 of Figure 2) is concerned with machines learning the structure of large-scale graphs and their attributes [19].

Our research objective focused on finding ways in which Boxes 1 through 3 can work together as a team to solve problems in a way that takes advantage of present-day machine learning techniques, while also allowing for explanation of rationale in decision making. Our brains do this fairly well (in addition to dealing with uncertainty), but we want to computationally enable these types of capability for model-centric engineering by characterizing AI/ML design patterns that support architecting effective digital twins by bringing AI and ML together in a new way.

Ultimately, a goal, not accomplished in this research is to address a closely related, but more difficult, problem is one of using combinations of AI/ML for the modeling of digital threads throughout the systems lifecycle as reflected in Figure 3 [5]. A digital thread is a communication framework that allows connected data flow and an integrated view of the system’s data throughout its lifecycle, across viewpoints that are isolated functional perspectives. A fully implemented thread enables anticipation and effective communication bi-directionally up and down streams of dependency in the lifecycle. This framework ensures all participants (stakeholders) have access to and can utilize the most current data and can react quickly to changes in the system objectives or in response to new insight. For our purposes, a simple way of distinguishing twins from threads is as follows: twins represent the state of a system; threads capture the evolution of systems through a sequence of states and transformations [5]. And since systems engineers focus on different things and different stages of the lifecycle – concept, design, manufacturing, test, operations, retirement – ease of systems adaptability and interoperability is a major challenge in getting digital threads to work.



**Figure 3. Schematic of a digital thread framework supporting flows of data and integration of viewpoints across the system lifecycle.**

---

### 1.3 SCOPE

Our program of investigation was motivated by the needs of two case study problem domains. This work planned to take initial steps toward studying their behavior as event- and data-driven behavior on spatial-temporal graphs. The demonstration prototypes would be designed to highlight the ways AI-ML can work together to manage these concerns.

**Case Study Problems.** The two digital twin domains are as follows:

1. **Skyzer UAV Search and Rescue.** This case study will leverage, refine and/or extend an evolving surrogate pilot developed for a Skyzer UAV operating in the context of Search and Rescue mission scenarios, in particular, landing scenarios in the context of ship-based operations, command and control, and flexible autonomy [16]. The problem domain provides a rich environment for temporal and spatial interactions, and also brings in lifecycle considerations of the digital twin caused by repeated missions in different environmental conditions.
2. **Vehicle Traversal of a Traffic Intersection.** The second case study involves a digital twin vehicle that has to traverse a busy traffic intersection safely and without causing an accident. Again, this case study also provides a rich environment for temporal and spatial interactions, and also brings in lifecycle considerations of the digital twin in evolving environmental conditions.

These case study problem areas share: (1) the presence of multiple domains, (2) event-driven behaviors, (3) multiple streams of heterogeneous data, and (4) scenarios that are dynamic and time critical. Looking ahead, the demonstration prototypes will be extrapolated to DoD vehicle operational scenarios as our objectives are to be able to transition the research in the future to other SERC-sponsor services.

The efforts planned to use two case studies, with one that includes the Skyzer System model discussed in this report to provide a means for demonstrating and explaining AI/ML for MCE in the context of mission, system and discipline-specific models and scenarios already understood by SERC research task sponsors. However, as discussed starting in Section 3, the efforts to consistently reconstruct a graph consumed most of the effort. While these efforts resulted in positive results related to teaching machine to understand graphs, we were not able to directly apply these results to these use cases.

---

### 1.4 SUMMARY STATUS OF ACCOMPLISHMENTS

The following provides a summary of our research that has been accomplished on this task. Additional details are provided starting Section 3. Our investigations made progress toward the objective in two areas: (1) semantic modeling and reasoning with graphs, and (2) teaching machines to understand graphs, which has been the most important part of this research. The semantic modeling and reasoning side address the basic questions such as: how to represent various types of graph and create hooks to backend computational support for graph analysis.

To summarize the efforts and accomplishment, we have:



- Analyzed different graph embedding approaches.
- Decided to proceed with the Graph Autoencoder (GAE) proposed by Kipf et al. [43]
- Determined that guarantees of correct reconstruction of the graph topology and attributes using the GAE framework depend on the convergence of the optimization algorithm, the encoder architecture, and the decoder architecture.
- Investigated how learning curves can help diagnose underfit, overfit and optimization convergence issues, and assist in the design of the GAE architecture.
- Investigated requirements for convergence of the optimization algorithm.

We presented our research at the SERC Sponsor Review on November 18, 2020 [6] and the AI4SE/SE4AI Research Workshop in October [15]. The presentations were well received and some expert attendees (e.g., keynote speakers) at the AI for SE & SE for AI (AI4SE/SE4AI) Workshop noted that combining semantics with ML is a unique approach and contribution.

There is still research needed to devise a formula that allows us to look at the input graph and determine the neural network architecture that is required to reconstruct it precisely with a high degree of certainty.

---

## 1.5 ORGANIZATION OF DOCUMENT

**Section 1** provides an overview of the context for the needed research, objectives, approach, planned case study, brief summary of accomplishments and organization of this report.

**Section 2** provides some information on the background associated with the incubator projects to set context for the needed research.

**Section 3** provides an overview of the research that is discussed in more detail in Sections 4 and 5, in addition to providing some underlying information related to graphs.

**Section 4** describes considerations for using semantic modeling with graphs, by describing different types of graphs, approaches to graph analysis and semantic rules for reasoning about graphs.

**Section 5** describes an area of the research that consumed the most amount of time for this research, where we accomplished a key objective to teach machines to understand graphs. This section discusses the details that advanced throughout the research.

**Section 6** summarizes the deliverables and events.

**Section 7** provides a summary of the results as well as describing a few next steps related to things that emerged as additional needs out of the research.

**Section 8** lists acronyms and abbreviations used through the report.

**Section 9** lists trademarks to different tools and technologies.

**Section 10** provides relevant references.

## 2 BACKGROUND AND CONTEXT

This section provides some background and context described in the SERC WRT-1011 Final Technical Report of the incubator projects that resulted in this research and report [5].

**Digital Twins: What's the problem? How is it done today? Who cares? Challenges?** A digital twin is a cyber (or digital) representation of a system that mirrors its implementation in the physical world through real-time monitoring and synchronization of data associated with events. The associated software and algorithms work to provide superior levels of attainable performance in system development and operation. The digital twin concept dates back to the 2000-2010 era; it was initially proposed as a way to support the design and operation of air vehicles for NASA. Since then, the range of potential applications has expanded to include automotive components, manufacturing processes, personalized medicine and smart cities, among others.

Within the world of model-centric engineering, there is strong need for support throughout the entire systems lifecycle, and not just the frontend. As a result, as shown in Figure 4, tool vendors such as Siemens and IBM now anticipate that digital twin capabilities will be the likely successor to model-based systems engineering (e.g., with SysML). Present-day trends in technology development at companies such as Google, Microsoft, Facebook and Apple indicate that AI and machine learning (ML) technologies will be deeply embedded in **new methods and tools for model-centric engineering**, as well as new **digital twin operating system environments** for observation, reasoning and physical systems control.

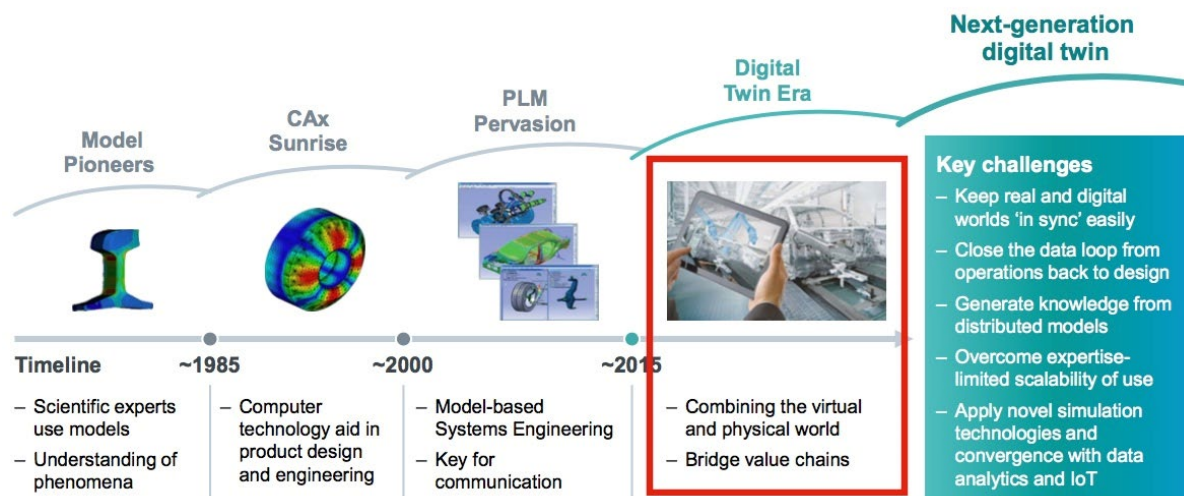


Figure 4. Emergence of Digital Twin Era, a replacement for MBSE with SysML

**Challenges?** Realization of this opportunity is complicated by the reality that within the world of model-centric engineering, present-day use of AI and machine learning technologies is fragmented and at a crossroads. During that past decade systems engineering researchers in AI (i.e., knowledge representation and reasoning) have tended to focus on the comprehensive development of ontologies for a domain (e.g., satellites) or system development activity (e.g., requirements, system mission, behavior modeling) and their extension from common core

ontologies [23] (e.g., for geospatial, time, actors, events) and higher-level basic formal ontologies [4]. Far less attention has been given to the development of rules associated with ontologies, and consideration of the ways in which ontologies and rules can work together to respond to events and support decision making. At the same time, machine learning (i.e., modern neural networks, data mining) techniques provide comprehensive support for the classification, clustering, and identification of association relationships, remembering the details of data streams, and finding anomalies in behavior. Remarkable advances in machine learning algorithms include the ability of a machine to learn the structure of large-scale graphs and their attributes. Looking forward, the consequences of this recent capability for the model-based systems engineering community would appear to be enormous. And yet, machine learning techniques struggle to explain the rationale for decision making, a task that multi-domain semantic modeling and rule-based reasoning can complete with ease.

A second source of difficulty stems from enhanced expectations for digital twins enabled by AI/ML technology. Digital twins deployed in real-world situations will be required to produce superior levels of system performance, agility and economy, across multiple scales of problem size and spatial and temporal extent. Consider, for example, a scenario where a digital twin vehicle has to traverse a busy traffic intersection safely and without causing an accident. As illustrated in Figure 5, challenges include the presence of multiple domains, multiple streams of heterogeneous data, event-driven behaviors, scenarios that are dynamic and time critical. Together with appropriate sensing technologies, AI and ML will be required to observe (monitor) the surrounding environment, evaluate options and take actions in a timely manner. Larger scale systems, such as a city or fleet of aircraft or ships, will be defined by collections of digital twins. From an AI/ML perspective, we expect that individual digital twins will belong to communities, and benefit from AI/ML software common to the community needs.

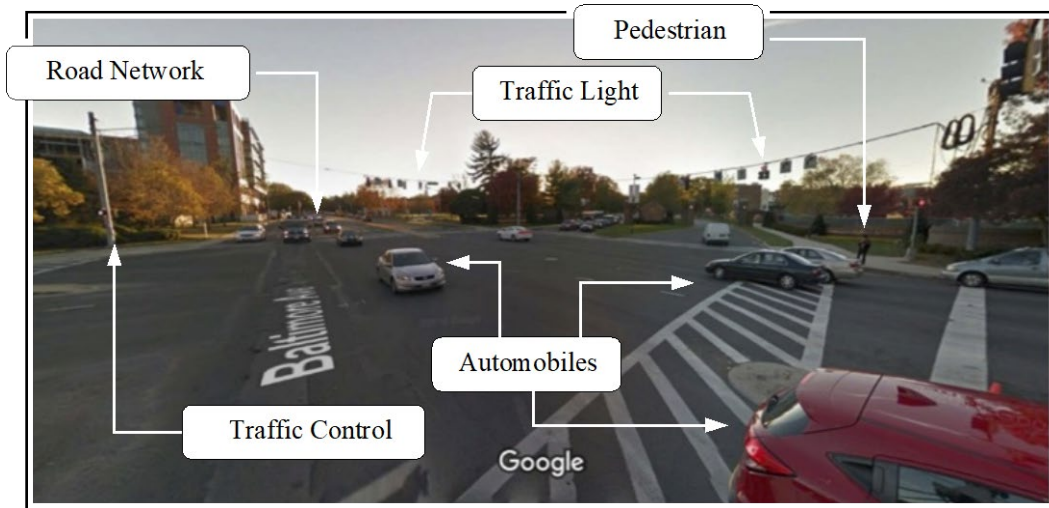


Figure 5. Annotated traffic intersection at the entrance to UMD.

### 3 RESEARCH OVERVIEW

This section discusses the thrust of the research given the context discussed in Section 2. The areas of advancements are reflected in the lower right of Figure 6, which extends Figure 4. Figure

6 reflects on the autoencoder contribution to the overarching objective to understand graphs. One of our research objectives is to understand the range of possibilities for which machine learning of large-scale graphs and their attributes can support urban digital twins. This section discusses what we learned in order to defined experiments where we have positive progress towards the first few objectives for this research.

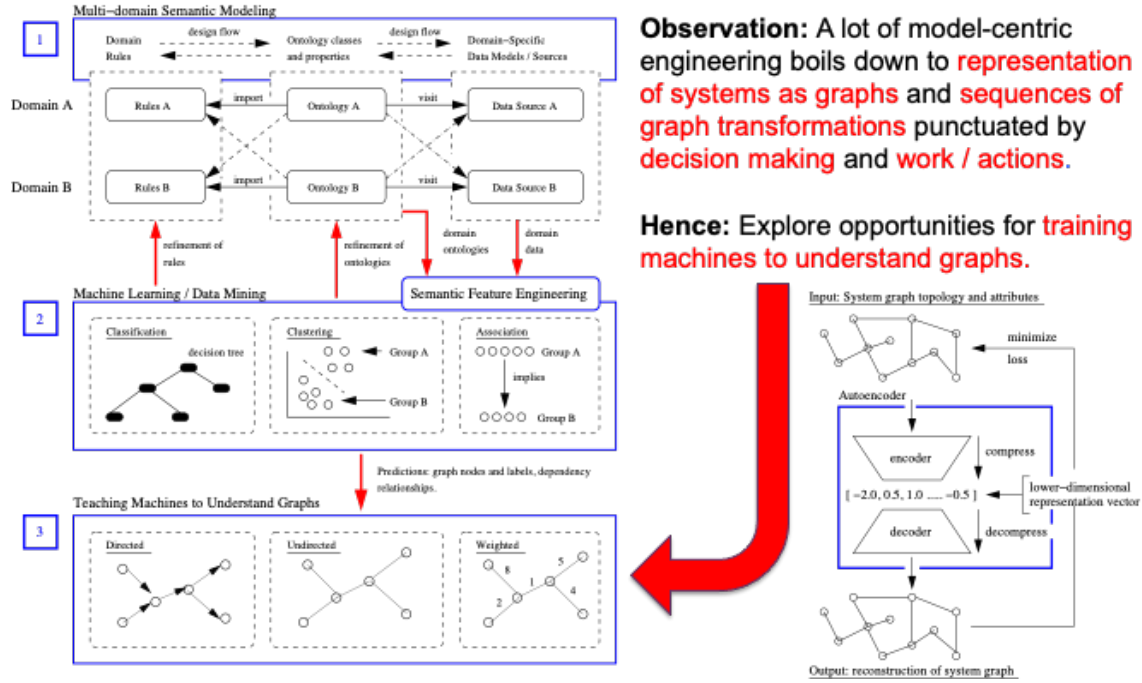


Figure 6. Neural Network Autoencoder Use to Understand Structure and Features of Graph

### 3.1 HISTORICAL PROGRESSION

We start with a historical perspective on the research steps that we covered in attempt to determine the necessary steps for teaching machines to understand graphs as discussed in Section 3.2. We first examined different graph embedding approaches developed in recent years and concluded that the Graph Autoencoder (GAE) approach proposed by Kipf [43] and co-workers in 2016 was the most suitable for our purposes, as it is able to encode both structural and attribute information while ensuring the generated embedding is an accurate representation of the information contained in the graph. We also understood some of the limitations of the GAE framework and concluded that guarantees of correct reconstruction of the graph topology depend on: 1) the convergence of the Adam optimization [42], 2) the encoder architecture, and 3) the decoder architecture. Early work also included experiments with JGraphT [41] and shortest path (and all directed paths) analysis. On the machine learning side, we did experiments with graph auto-encoder (GAE) embedding vectors and considered opportunities for AI-ML cooperation and started investigations of semantic modeling and reasoning about graphs prior to our explanation of the key details about the GAE Architecture and Optimization Algorithm.

Our research next focused on investigating how learning curves can help us adapt the GAE architecture to avoid underfit, overfit and convergence issues. Sample case studies from common system topologies were reconstructed by the GAE framework, and learning curves were used to analyze the effects of model architecture to performance. Those experiments showed that certain architectures successfully reconstruct the input graphs, while others lead to convergence issues. Hence, our next goal was to devise a mathematical formula that allows us to look at the input graph and determine what architecture is required to perfectly reconstruct the graph. As a first step towards this objective, we studied the relationship between the input graph and the convergence of the optimization algorithm. More details on this aspect of the research are provided in Section 5.

We then moved on to devise a formula that allows us to look at the input graph and determine what architecture is required to reconstruct it precisely with a high degree of certainty. These efforts are summarized in Section 5.10, but build on the examples and prior efforts summarized in Section 5.3. A GAE is neural network-based algorithm, and as a first step towards achieving our objective is to understand how neural networks learn graph topology. We divided this objective into two sub-tasks: (1) understanding the role of neurons in the learning, and (2) understanding the role of layers in the learning.

We were later able to brief our progress to some well-informed participants at the US-attendance only AI4SE/SE4AI Research Workshop and later at the SERC Sponsor Review; this provided some level of validation about our approach, which was well received based on the comments and questions. We emphasized that what may be considered most unique about our research is that we are “integrating” semantic graphs based on ontologies and reasoners (this would be the deductive side of AI) with graphs representing temporal and spatial events of a cyber physical system based on ML. The two most important questions coming from the audience that reflect most highly on the significance of our research were related to what we believe to be unsolved (especially based on those questions); that is, how to relate information gained from ML back into the semantic graphs, and vice-versa, as reflected by the red lines in Figure 1 and Figure 2. Ultimately, this type of information would continue to feed forward during different phases of the lifecycle as reflected by Figure 3.

---

## 3.2 GRAPHS REPRESENTATIONS AND ANALYTICS

Figure 7 shows a simplified representation for traditional and machine learning approaches to graph representation. Traditional approaches to network/graph modeling employ adjacency matrices (or a simplified representation of network adjacencies) to model the topology of graphs. If one were to build a semantic model of a graph (see Box 1 in Figure 2) the corresponding network ontology would contain classes for the nodes, edges and attributes in a graph and result in a structure along these lines. Mathematical algorithms to determine the properties (e.g., existence of cycles, connectedness, minimum paths) of a graph are well established. This is **graph analysis**.

However, for high-dimensional problems that are data sparse, such approaches can quickly become computationally prohibitive. A second problem is that traditional approaches to graph

representation do not capture the semantics of the network. The lower half of Figure 7 shows that simple approaches to machine learning of graphs come at the problem from an entirely different perspective. Instead of focusing on the graph topology (connectivity relations), the graph nodes and their attributes (semantics in domain applications) are mapped (or encoded) to a low-dimensional embedding space, with the goal of preserving local linkage structure (not global structure). Embedding structures are derived from random walk-based procedures; three such approaches are LINE [60], DeepWalk [49] and Node2Vec [32]. Decoders are designed to extract views of the graph representation from the low-dimensional embedding. Because information can be lost in the encoder-embedding-decoder transformation process, the output of machine learning for graphs is statistical in nature and, as such, should be interpreted as **graph analytics** (not graph analysis). Graph analytics can support a variety of decision-making tasks, including: node classification, node clustering, prediction of anomalies in data streams, link prediction, and recommendations for association relationships.

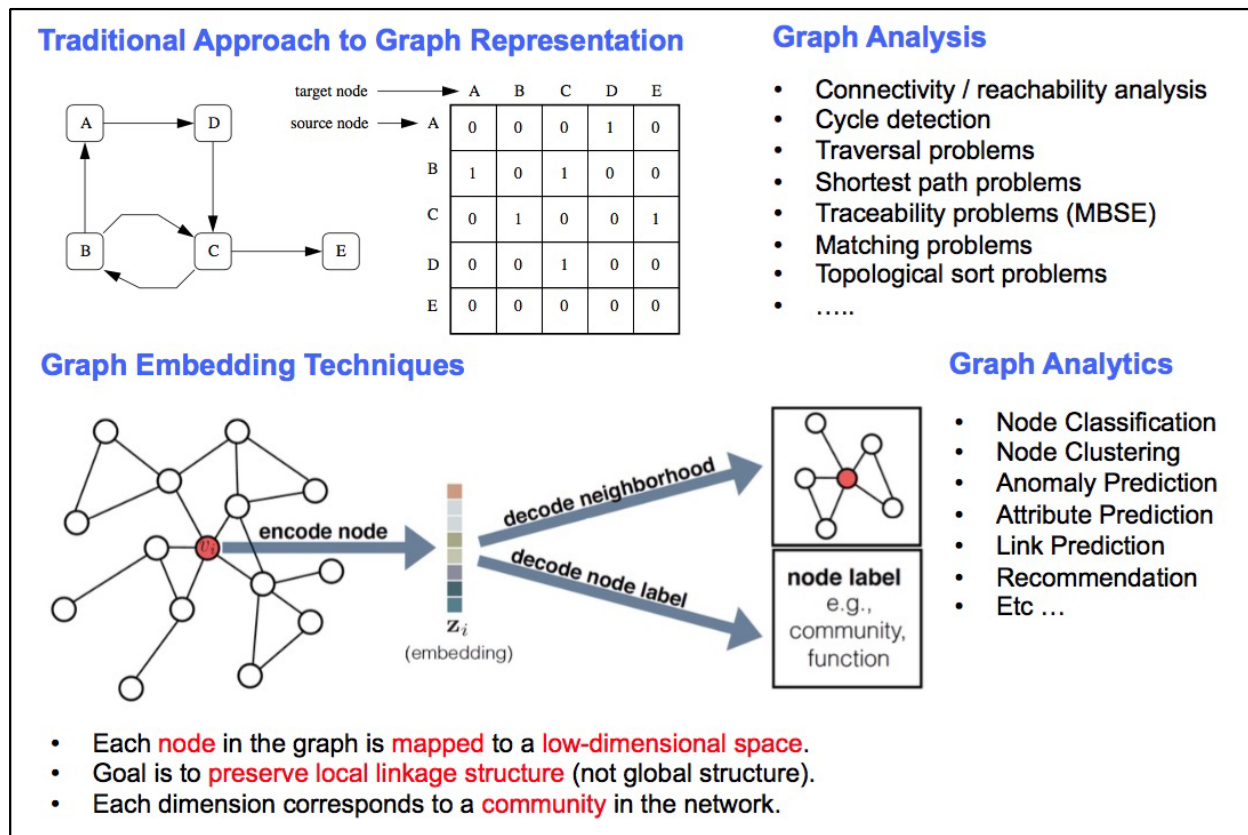


Figure 7. Schematic of traditional (adjacency matrix) and machine learning (graph embedding) approaches to modeling graphs.

We believe that with some imagination, these features could be incorporated into new types of tools for model-centric engineering. For example, the upper half of Figure 8 shows an auto-encoder design of link prediction – one can imagine a “missing feature” recommendation service working in parallel with semantics models and rules for system validation and verification. The lower half of Figure 8 shows extensions of the basic auto-encoder design to deep graphs. Notice that unlike the simplified representation of (localized) graph topology shown in Figure 7, the



embedding vector for deep graph learning extracts an embedding vector for proximity to all of the other nodes. Thus, one can expect that deep graph representations will contain guarantees on arbitrarily high – first, second, third, fourth – levels of nodal proximity. This would appear to be a necessary feature for machine learning to participate in operations for requirements traceability.

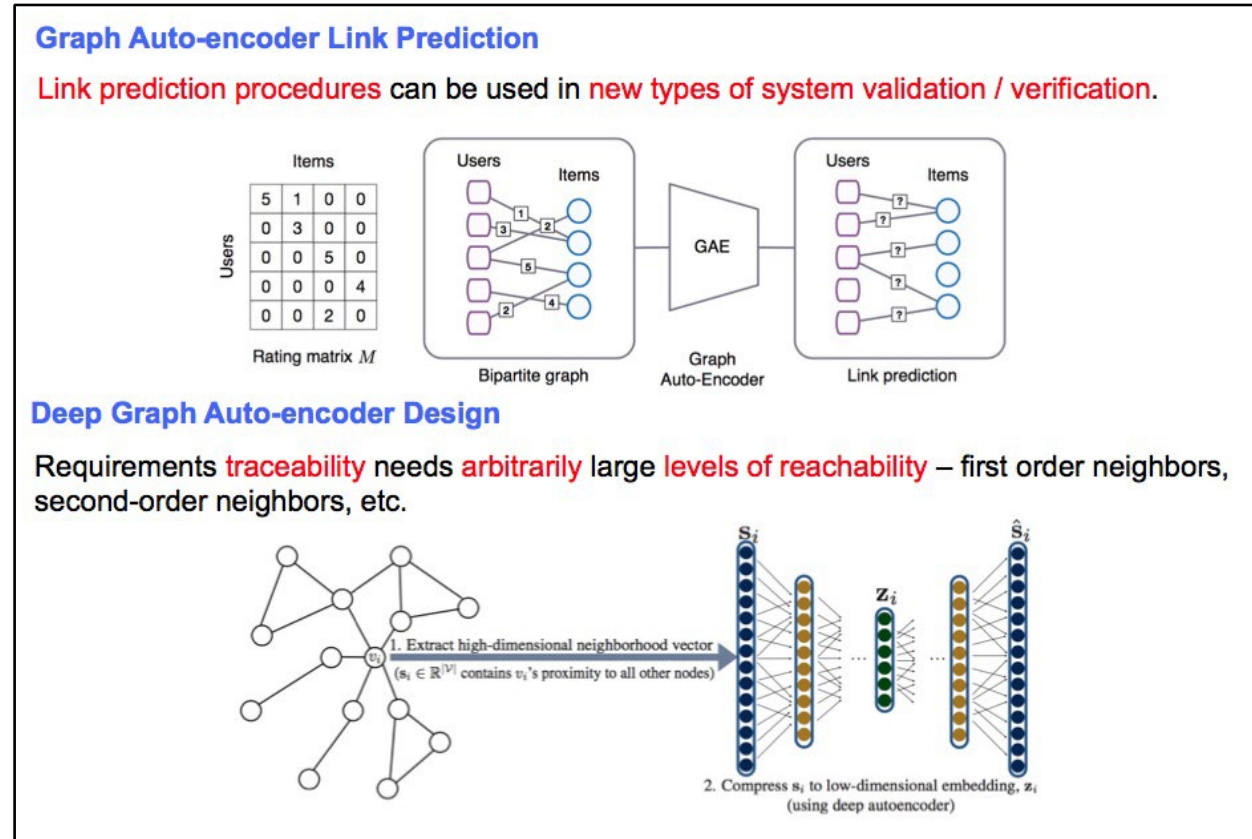
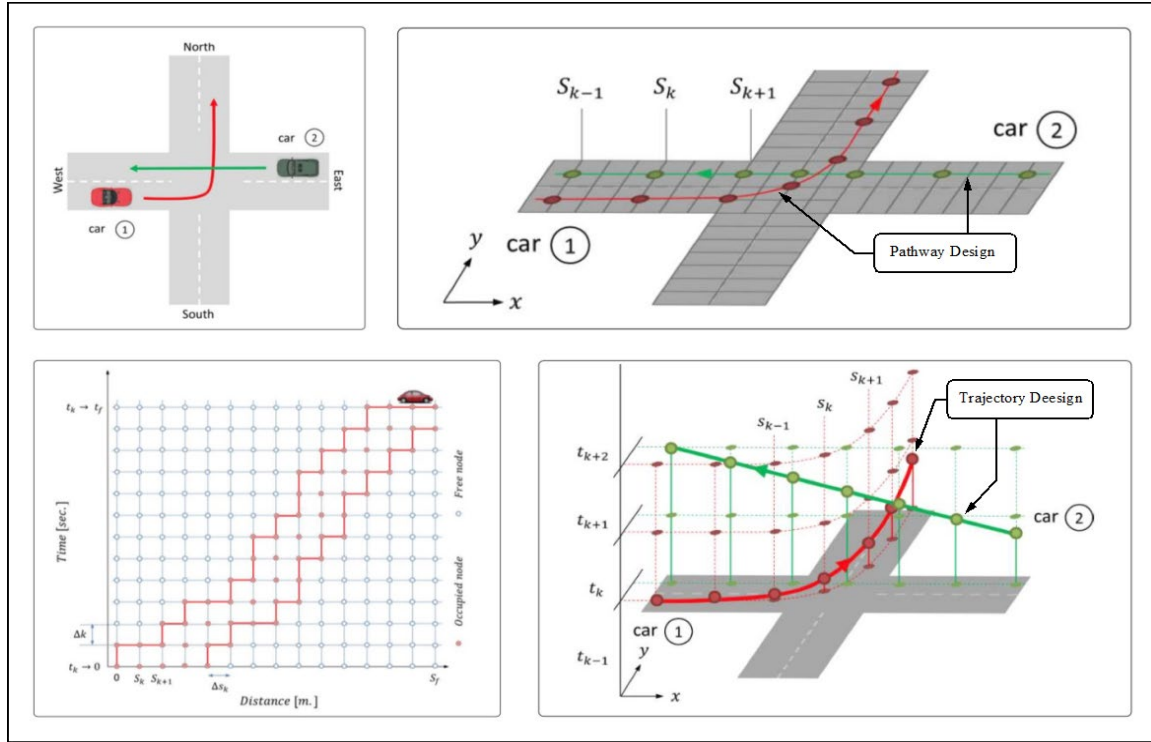


Figure 8. Auto-encoder design for link prediction and deep graphs.

### 3.3 CONTROLLED BEHAVIORS ON SPATIAL-TEMPORAL GRAPHS

This section moves away from the underlying graph representations and relates the graphing technologies to a use case scenario. To understand what the underlying modeling and mathematical frameworks might look like, Figure 9 shows four views of required behavior when a vehicle needs to make a left-hand turn at a simple traffic intersection. This problem is very simple: two cars, one traffic intersection, one left-hand turn and no provision for handling of unexpected events. The goals of the individual vehicles can be expressed as pathways (e.g., starting from position A, move to B, then C). Vehicle trajectories describe the position of the vehicle as a function of time (see bottom left-hand schematic of Figure 9) in so-called space-time terrain. When multiple vehicles have pathways that cross the intersection space, the key to preventing accidents is to ensure that the space-time trajectories for individual vehicles are well spaced (see bottom right-hand corner of Figure 9). Note that this use case can apply to other types of vehicles such as tanks and/or air vehicles on a runway on land or on a ship.



**Figure 9. Spatio-temporal modeling of a vehicle making a left-hand turn at a traffic intersection**

To see what issues and opportunities arise when a problem domain is scaled up, Figure 10 is a plan view of runways, taxiways, and holding positions (control points) for plane operations at Baltimore Washington International (BWI) airport. Collectively, runway, taxiway and holding position entities connect into a graph structure; plane movements can be viewed as controlled behavior of multiple entities on a spatial-temporal graph. This problem setup is considerably more complicated than the previous example because the graph structure contains many points requiring time management of spatial resources and satisfaction of safety concerns. In a commercial setting, issues of operational performance and fairness also need to be taken into account. When a plane taxis from a departure gate to the end of a runway, the chosen pathway can be viewed as a sequence of taxiway segments connected to holding positions. Holding positions are control points that require a plane to stop until permission is given to proceed onto the next segment of the pathway. The corresponding spatial-temporal trace will be continuous, but unlike the previous example not necessarily smooth. This operation repeats many times on a daily basis. One role for machine learning is to provide predictions of congestion and associated delays, and factor these estimates into the selection of an appropriate pathway.



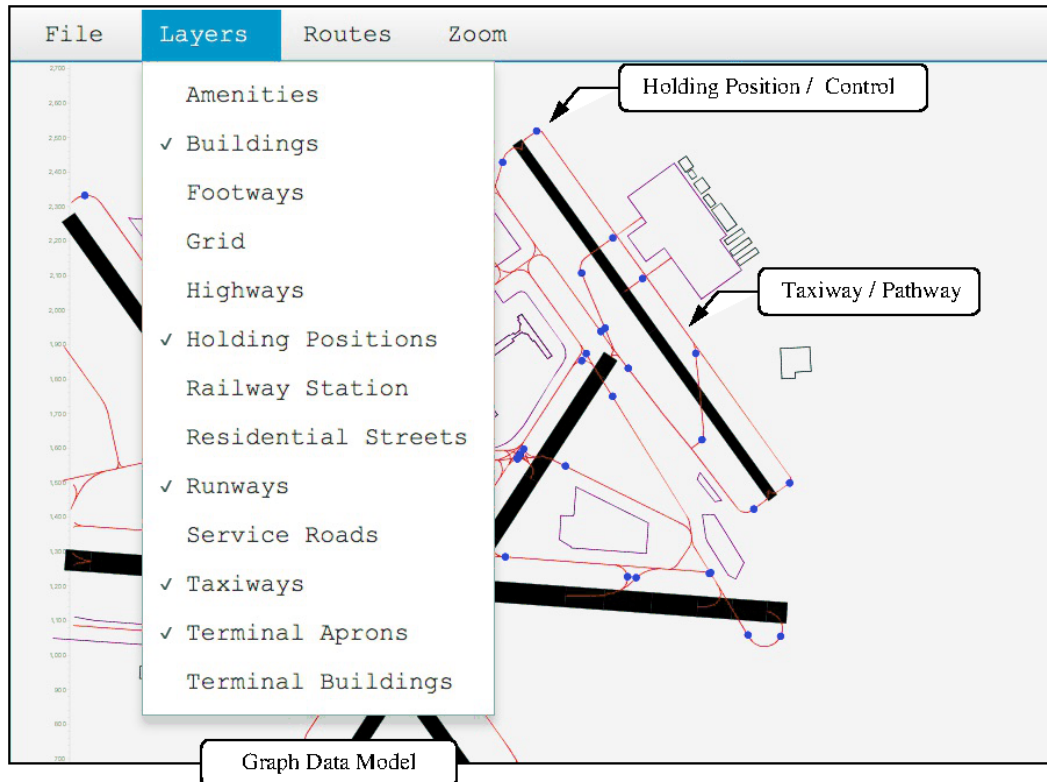


Figure 10. Synthesis of graph data models from pathways and controls in Open Street Map

## 4 SEMANTIC MODELING AND REASONING WITH GRAPHS

The semantic side of the problem as reflected by Box 1 of Figure 2, is where we developed and are evolving software to support: (1) the semantic representation of graphs, and (2) the formal representation of graphs, plus access to data structures and algorithms for graph analysis with JGraphT [41].

This work fits into a general framework for multi-domain semantic modeling and reasoning, as illustrated in Figure 11. To support the graph data structures and algorithms for graph analysis, we envision graph ontologies and graph rules operating at the multi-domain level. The shadow below “graph rules” represents links to backend software for graph analysis. Further details on how this can work are provided toward the end of this section (also see the 2017 paper of Delgoshaei and Austin [24]). In addition to graph structures, we also need support for representation and analysis of pathways and trajectories. Pathways indicate where something will go; trajectories describe the course of a measured variable (e.g., position of vehicle or drone) over time. To ensure the space-time trajectories of vehicles and drones are adequately separated, we need backend support for spatial and temporal reasoning, plus strategies for controlling movement. In Figure 11, this is indicated by the shadows adjacent to time rules and spatial rules. Networks are simply specialized graphs.

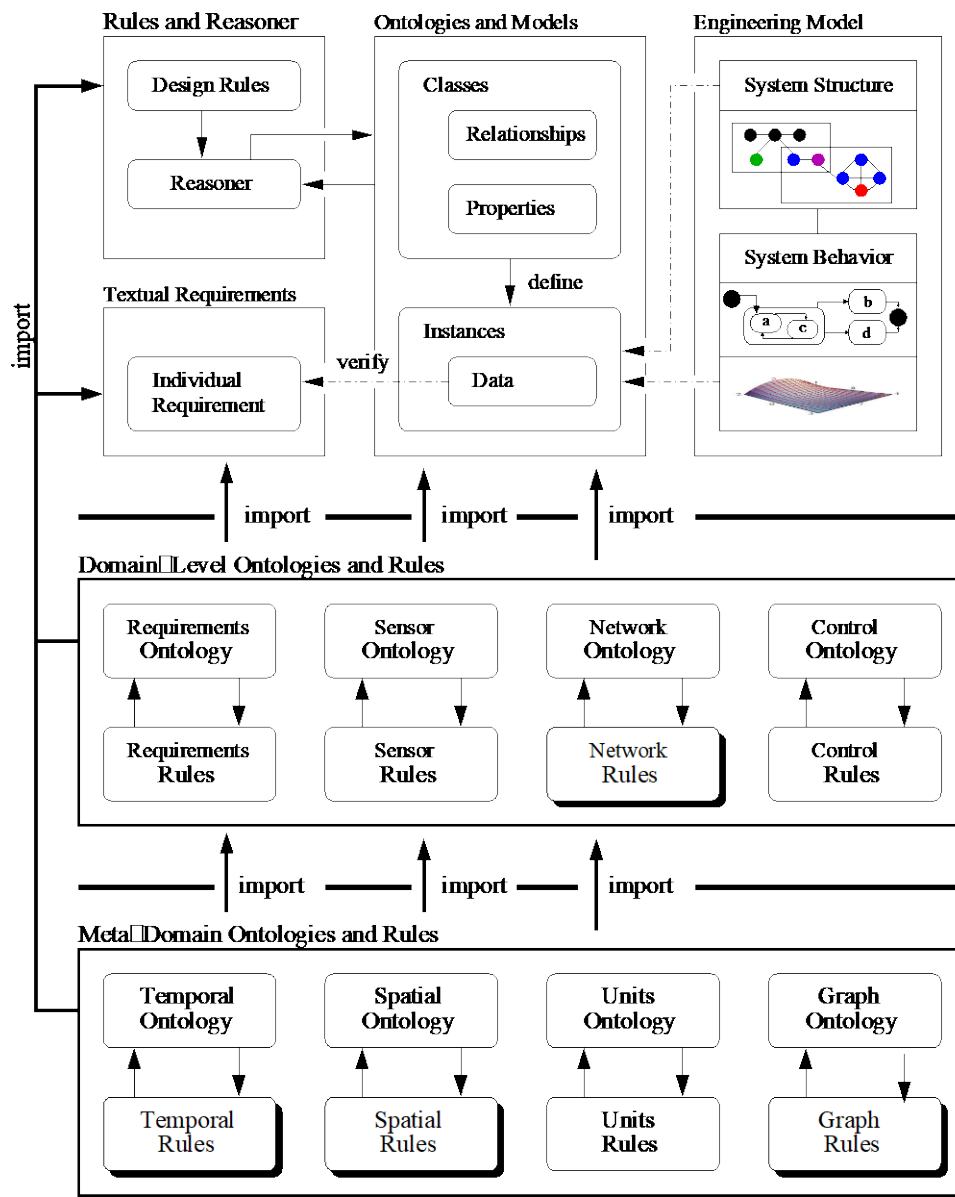


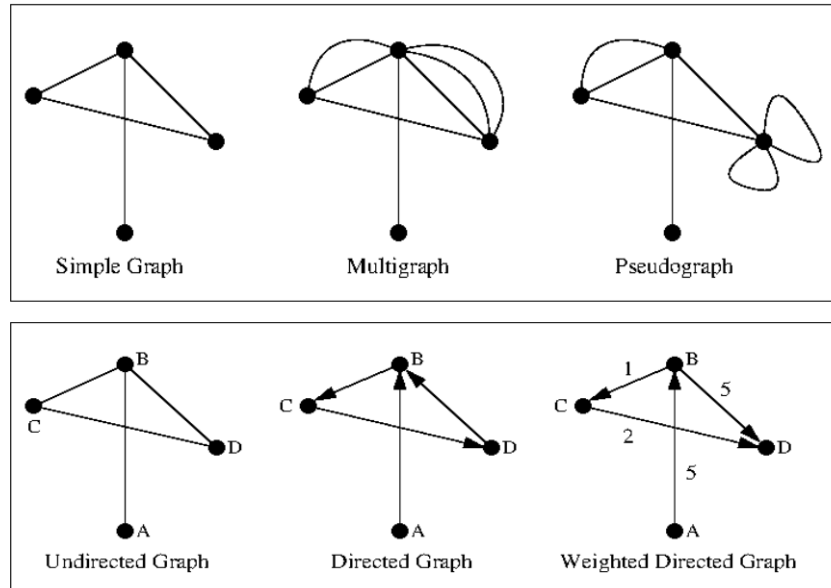
Figure 11. Framework for Multi-Domain Semantic Modeling

#### 4.1 GRAPH TYPES AND DRAFT GRAPH ONTOLOGY.

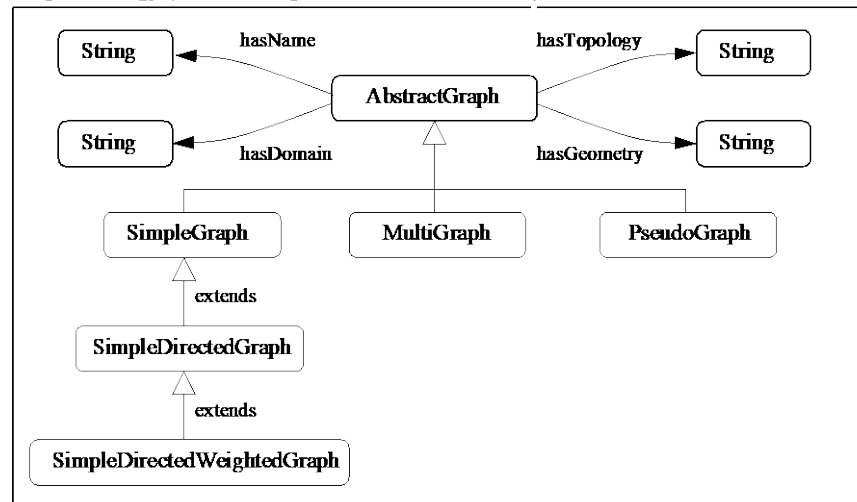
The upper half of Figure 12 shows the range of graph types supported by JGraphT. We have sets of nodes and edges connected in a variety of ways. Simple graphs have edges that connect nodes. Edges are unique and undirected. Multigraphs extend this formalism by allowing support for multiple edges between two nodes. Pseudographs add the possibility of edges that self-loop to a single node. Edges may also be directed and accompanied by labels and weights. The lower half of Figure 12 shows a new graph ontology (work in progress) that we anticipate will act as a bridge between the graph / system model data and rules for validating and analyzing graphs. In 2017 we employed a similar approach for the development of a spatial ontology for two-

dimensional geometry, with backend support for computation of geometric relationships (e.g., polygon set operations) with the Java Topology Suite (JTS) [39]. This worked well, so we are hopeful a similar approach will work well for graphs.

#### Classification of Graph Types



#### Graph Ontology (Mirrors JGraphT Software Architecture)



**Figure 12. Classification of graph types + draft of graph ontology mirroring architecture of JGraphT software**

Graph types are specialization of an abstract graph containing references to string representations of the graph topology (i.e., nodes, edges, and their connectivity). Since many of our graphs will be embedded in spatial terrain it also seems reasonable that the graph will also have an abstract geometry specification, which in turn will link to the JTS. This latter feature will allow for a system to answer questions like: which parts of a graph are within a certain geographical region? Or perhaps, what types of graph are within a certain geographical region?

---

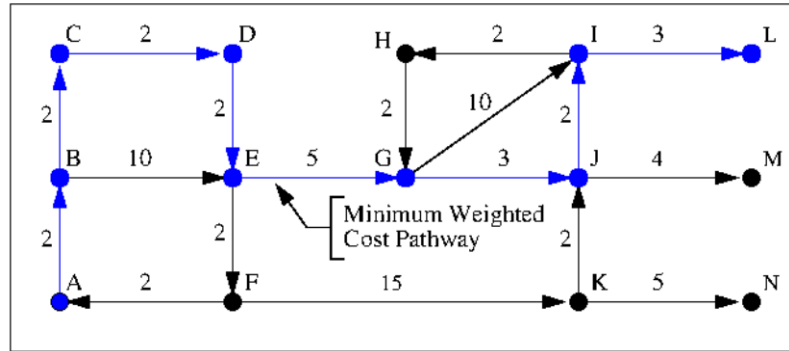
## 4.2 GRAPH ANALYSIS AND SEMANTIC RULES FOR GRAPHS

In order to support demonstrations, we are particularly interested in directed weighted graphs. Graph analysis algorithms of interest include: analysis of edge and vertex connectivity; minimum spanning trees; shortest path analysis; identification of sub-graphs having strong connectivity and all paths analysis. From a software standpoint, listeners can be attached to the nodes and edges of a JGraphT model. Thus, when the value of an element (e.g., a weight on an edge) changes, or a vertex or edge is added / removed from the model, an observer will be notified. Apache Jena uses this mechanism to fire the execution of rules in Jena [3].

We used the following scenario to see how JGraphT might be used for static and dynamic mission planning, the upper half of Figure 13 shows a small directed weighted graph. The blue contour shows the minimum cost pathway from nodes A to L. The shortest path algorithm circumvents the two edges having weight = 10. Now suppose that an environmental event occurs in the proximity of the graph, causing delays (increased edge weight) in select edges. The scope of our analysis is also extended to include sets of sources (i.e., nodes A, B and C) and sets of targets (i.e., nodes L, M and N). The “all paths” problem solves the following problem: given that you are located in one of the source nodes, what is the minimum weighted path to one of the nodes in the target set? The environmental event causes the weight of edge B-E to increase from 5 to 20. The blue contour in the lower half of Figure 13 shows the minimum cost pathway from node A to any of the nodes in the target set, in this case, it’s node N. From a demonstration standpoint, we can think of this modification as trajectory adjustment in a dynamic environment.

The graphs shown in Figure 13 are small and have been manually assembled. A next logical step is to synthesize graphs from Open Street Map data [53] and connect graphs to graph ontologies and rules and backend functions. Figure 10 is a multi-domain view of BWI airport, including its taxiways, runways and holding positions. One might think that the ways (i.e., <way> ... </way> tag pairs) would provide holding-position-to-holding-position connectivity. This turns out not to be true. Hence, we will need to work a little harder to read in the OSM data (i.e., nodes and ways) and organize it into a format suitable for analysis with JGraphT. A second common problem with OSM is missing data. We should be able to use the JGraphT analysis capabilities to ensure the graph is fully connected and, thus, suitable for semantic analysis.

### Static Mission Pathway Planning [ A $\rightarrow$ L ]



### Pathway / Destination / Trajectory Adjustment in a Dynamic Environment [ A $\rightarrow$ N ]

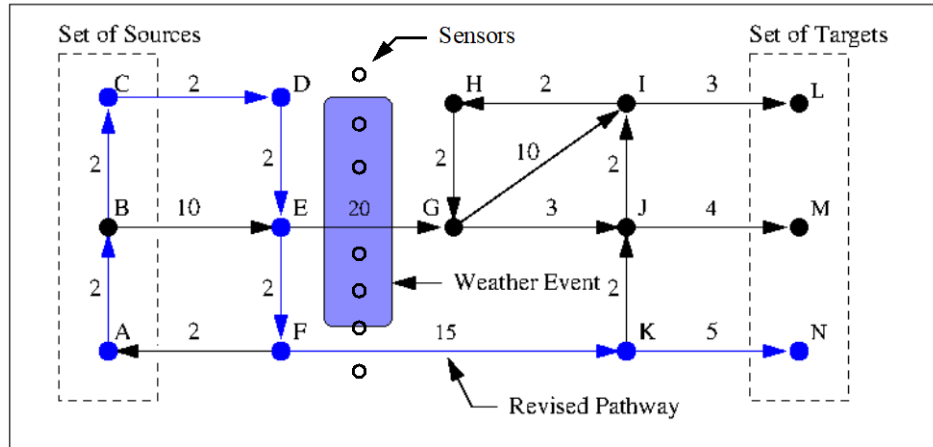


Figure 13. Shortest path analysis in a directed weighted graph. Top: Minimum weighted cost pathway for A  $\rightarrow$  L. Bottom: Revised pathway analysis when edges in graph are obstructed by a weather event.

The lower half of Figure 13 shows the general framework for modeling and reasoning with graphs in Apache Jena and Jena Rules. A key benefit in working with string representations of graphs is that they provide a means for graph analysis via the built-in functions. Thus, in this setup graph analysis algorithms will work along semantic rules for graphs – the details for exactly how this should work still need to be determined.

## 5 TEACHING MACHINES TO UNDERSTAND GRAPHS

A key requirement for creating a successful digital twin for model-centric engineering is achieving the ability to identify anomalies (faults) in system performance, and to model the behavior of processes and interactions among the different domains within a system. Our incubator final report proposed Figure 2 as an architectural template for the construction of digital twins [5]. Machine Learning (ML) formalisms and Semantic Model representations will work side-by-side as a team, providing supportive roles for the collection and processing of data, identification of events, and real-time management of operations.

This section turns the focus on teaching machines to understand graphs (see Box 3 of Figure 2). Remarkable advances in ML algorithms (2016-2019) include the ability of a machine to learn the structure of a graph and its attributes. 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.

---

## 5.1 MACHINE LEARNING OF LARGE-SCALE GRAPHS

In aligning with our research goal, we needed to understand the range of possibilities for which machine learning of large-scale graphs and their attributes can support urban digital twins. In recent years, many graph embedding approaches have been developed. Goyal and Ferrara categorized embedding methods into three broad categories: 1) Factorization based, 2) Random Walk based, and 3) Deep Learning based [31].

Factorization based methods represent the connections between graph nodes as a matrix, and then factorize this matrix to obtain the graph embedding. There are different types of matrices that can be used to represent the connections, including node adjacency matrix, Laplacian matrix, node transition probability matrix, and Katz similarity matrix, among others. Deciding how to factorize the representative matrix will depend on the matrix properties. For instance, when the problem description matrix is positive semidefinite (e.g., the Laplacian matrix) eigenvalue decomposition can be used to perform the factorization; if the obtained matrix is unstructured a gradient descent method can be used instead.

In a random walk-based approach, the embeddings are generated based on stochastic measures of similarity instead of deterministic measures. These stochastic similarities are determined by performing random walks on the graph. The idea is to have similar embedding for the nodes that frequently co-occur on these random walks. DeepWalk [49] was the first graph embedding algorithm that adopted this approach, other algorithms include Node2vec [32] and LINE [60]. Random walk-based approaches to machine learning of graphs are generally less computationally expensive than the factorization-based methods, and are especially useful when the graph can only be partially observed, or the graph is too large to measure in its entirety.

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; this leads to computational inefficiency since the number of trainable parameters grows linearly with the number of nodes in the graph. Second, these approaches also cannot generalize, they are only able to generate embedding vectors for the nodes that were present during the training phase and not for any unseen nodes. Third, 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. Therefore, other approaches that address some or all of the issues mentioned above have emerged recently and are referred to as deep learning-based methods.

Deep learning-based approaches use a deep neural network architecture, generally referred to as Graph Neural Networks (GNNs), to generate embedding vectors which depend both on the structure and the attributes of the graph. Wu et al. categorize GNNs into four groups: recurrent graph neural networks (RecGNNs), convolutional graph neural networks (ConvGNNs), spatial-temporal graph neural networks (STGNNs), and graph autoencoders (GAEs) [68].

RecGNNs utilize a recurrent layer to generate node embeddings for the graph. This approach assumes that a node in a graph exchanges information with its neighboring nodes until a stable equilibrium is reached. Although RecGNNs are computationally expensive, they are conceptually important and inspired later research on ConvGNNs.

Unlike RecGNNs, ConvGNNs use different graph convolutional layers to generate node embeddings. The main idea is to generate a node representation by aggregating its own features and its neighbors' features. Because graph convolutions are more computationally efficient and convenient to combine with other types of neural networks, the popularity of ConvGNNs has been rapidly growing in recent years.

Many alternative deep learning-based methods have been developed in recent years, including graph autoencoders (GAEs) and spatial-temporal graph neural networks (STGNNs). These frameworks can be built on RecGNNs, ConvGNNs, or other neural network architectures for generating graph embeddings. STGNNs aim to model dynamic node inputs, while assuming spatial and temporal interdependencies between connected nodes.

## 5.2 GRAPH AUTO ENCODERS (GAE)

GAEs are deep neural architectures that are trained to reconstruct their original input. Figure 14 shows a high-level architecture for GAE. An encoder takes a graph as its input and systematically compresses it into a low-dimensional (embedding) vector. The decoder then takes the vector representation and attempts to generate a reconstruction of some user-defined graph analysis tool (e.g., 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.

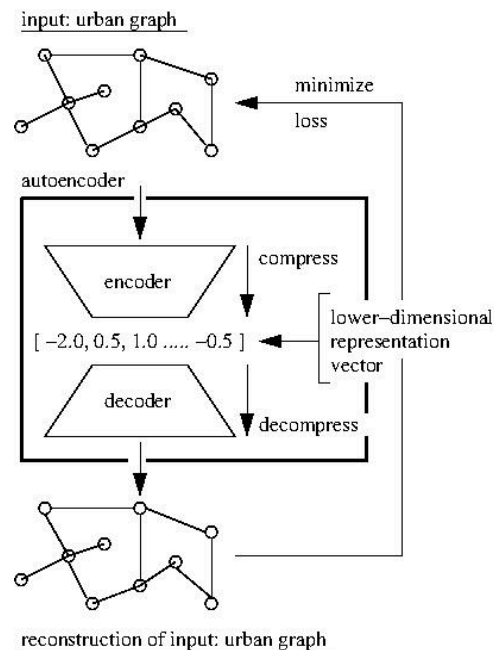


Figure 14. Traditional encoder-decoder approach.

Looking ahead, Figure 15 is a schematic for how the embedding process might be integrated in a digital twin. We start by extracting a graph representation of the system and determining its initial parameters. 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, recovery procedures will be triggered. Therefore, embeddings will be an 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. With this goal in mind, it is clear that GAE based approaches are the most suitable for our purposes.

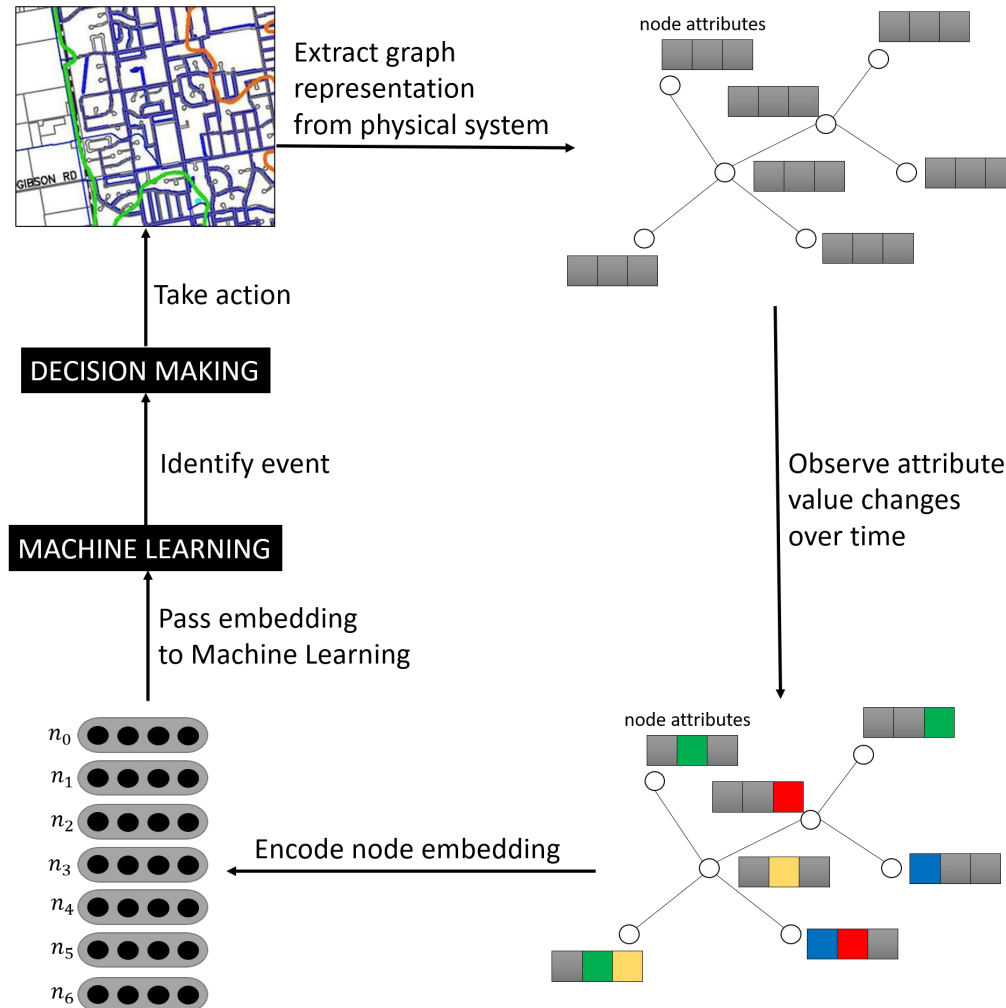


Figure 15. Process flowchart for training and executing machine

### 5.3 TEACHING MACHINES TO UNDERSTAND GRAPHS WITH GRAPH AUTOENCODER

Figure 16 shows the architecture of the GAE framework applied to this case study. Details of the following case study are being assembled into a journal paper to support our co-author, Maria's



PhD research. We will discuss the details at a higher-level, and if a draft of the journal paper is needed, we can provide in the future.

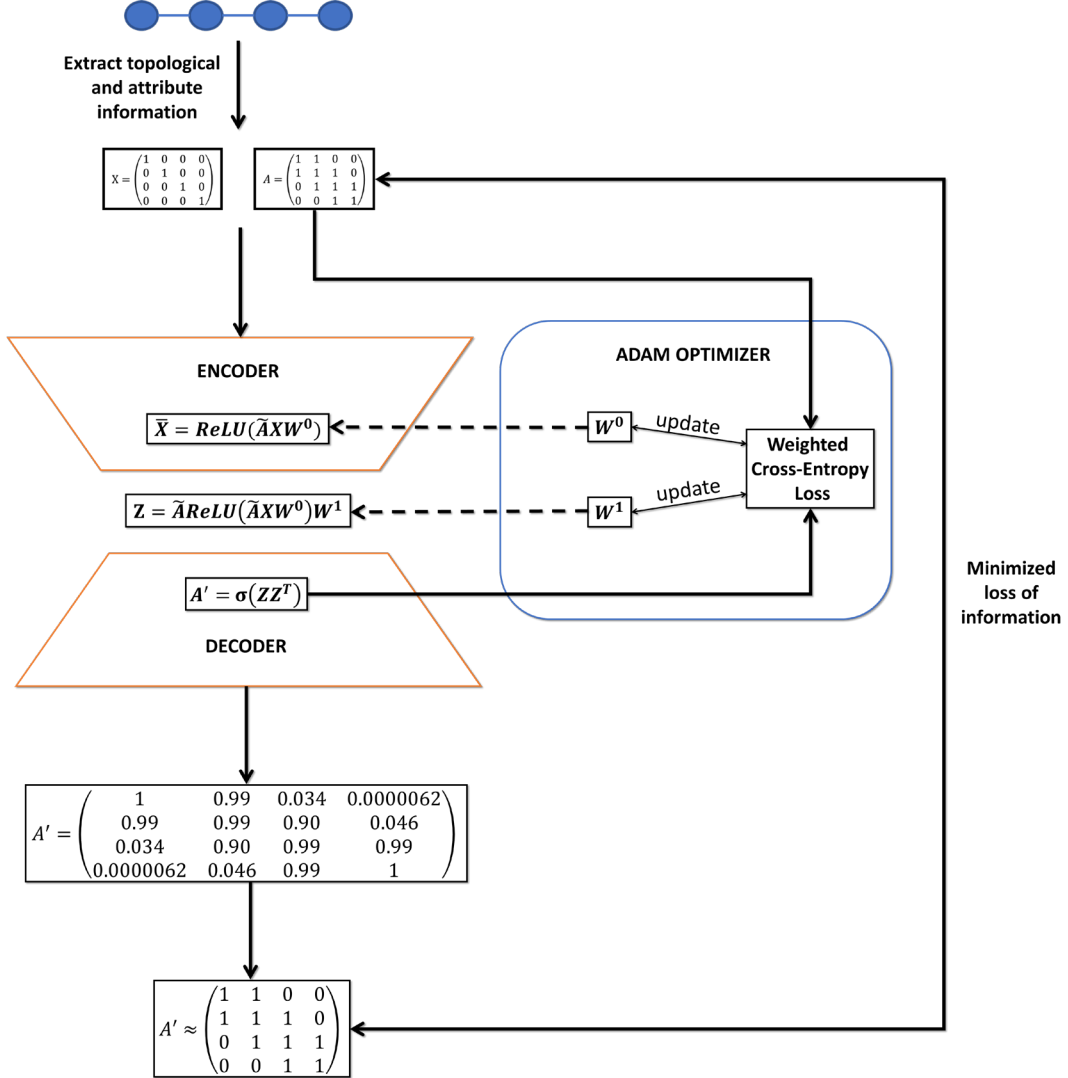


Figure 16. Flowchart of GAE computations applied to Case Study.

We start by encoding a four (4)-node graph and its node attributes. 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.

We next generate the first convolutional layer generates a lower-dimensional feature matrix defined as (data not shown):

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

where  $W_0$  is a trainable parameter matrix initialized to a small random value. In this case study,  $W_0$  was initialized (data not shown). The second convolutional layer takes as input the output of the previous layer and generates the node embeddings (data not shown):

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

where  $W_1$  is also a trainable parameter matrix initialized to a small random value. In this case study,  $W_1$  was initialized (data not shown).

The task 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, resulting in the reconstructed adjacency matrix (data not shown):

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

In order to arrive at the optimal embedding matrix  $Z$ , the  $W_0$  and  $W_1$  parameters are systematically updated through an Adam optimization [42] of the weighted cross-entropy loss between the adjacency matrix  $A$  and the soft reconstruction  $A'$ . The weighted cross-entropy loss is calculated where  $N_{pos}$  is the number of edges in the graph, and  $w_{pos} = \frac{N^2 - N_{pos}}{N_{pos}}$ . Multiplying the positive labels term by  $w_{pos}$  balances the quantity of edges and the non-edge counterparts. With the initial values of the weight matrices set above for this case study,  $N_{pos}$ ,  $w_{pos}$ , and loss  $L$  obtain (data not shown):

In this first iteration the weight values were set randomly; however, the goal is to find the set of weight values that minimize loss. Hence, through Adam optimization we find the direction to move the weight values in order to get a lower loss on the next iteration (i.e., gradient). Knowing which direction is downhill, we are able to update the weight values.

After 200 iterations we obtain the node embeddings (data not shown) to reconstruct the final graph. Comparing the reconstructed adjacency matrix  $A'$  to the original adjacency matrix  $A$ , we can conclude that for this simple case study the GAE framework was able to perfectly reconstruct graph topology.

We now understand the inner architecture of GAE and what limitations it may encounter, as well as the ways in which it can be changed to adapt to specific problems. For instance, it is clear that in order to obtain a good reconstruction of the input graph, the algorithm depends on the:

1. Convergence of the Adam optimization.
2. Encoder architecture
3. Decoder architecture

---

## 5.4 CONVERGENCE OF THE ADAM OPTIMIZER

One of the key hyperparameters to set in order to construct an Adam optimizer is the learning rate [42]. 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, training will progress very slowly as we are making very small updates to the weights values. However, if one's learning rate is set too high, 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). These cases can be visualized in Figure 17.

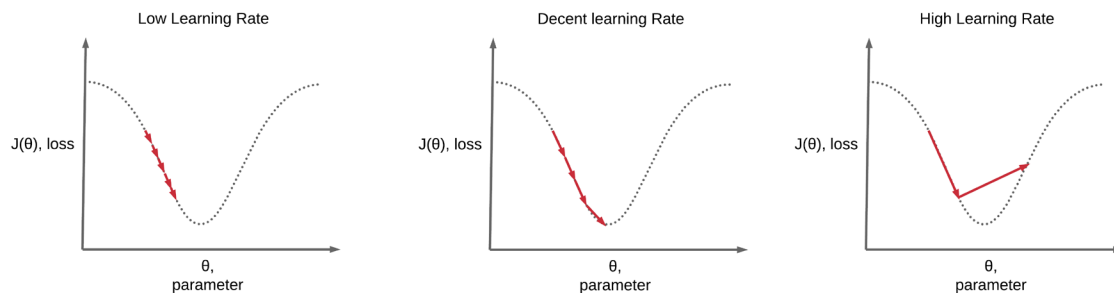


Figure 17. Effects of learning rate on optimization convergence ("Python Lessons").

---

## 5.5 ENCODER ARCHITECTURE

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 [43], 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: 1) how many hidden layers to actually have in the neural network and 2) how many neurons will be in each of these layers. Problems that require more than two hidden layers were rare prior to deep learning. Two or fewer layers will often suffice with simple data sets. However, with complex datasets additional layers can be helpful. According to Heaton [35] 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, 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 a network with more than two layers can learn complex representations (i.e. feature engineering) for layer layers [35]. For our purposes, **we are interested in investigating whether adding more layers to GAE algorithm can bring benefits** (i.e., faster convergence, lower loss, 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 [35]. 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 [35]. Additionally, a large number of neurons in the hidden layers can increase the time it takes to train the network to a point where it is impossible to adequately train the neural network [35]. 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 (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 issues described above.**

---

## 5.6 NEURAL NETWORKS AND GRAPH AUTOENCODER

The selection of an architecture for our neural network will come down to avoiding underfit, overfit and convergence issues. Learning curves are a widely used tool in machine learning for algorithms that learn (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 and answer the following research question: **Can learning curves help us find the optimal number of neurons and layers?** In this report we use simple case studies from common system topologies (see Figure 18), apply GAE framework to them, and use learning curves to analyze the effects of model architecture to performance.

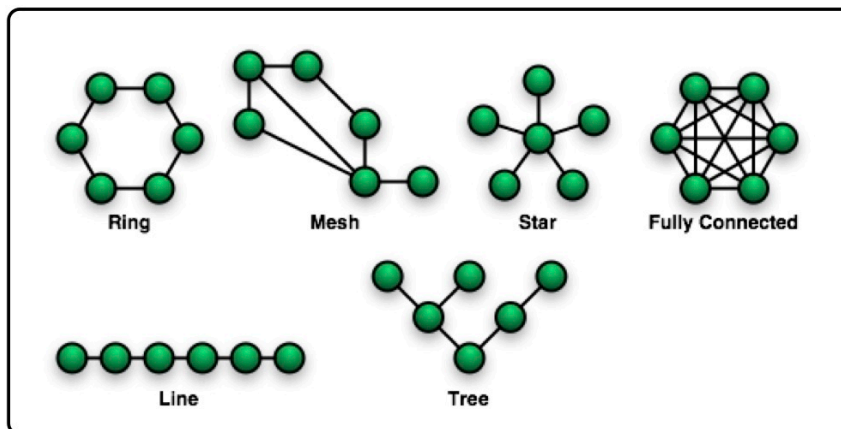


Figure 18. Common system topologies.

### 5.6.1 LINE TOPOLOGY

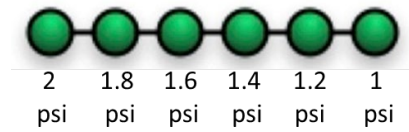


Figure 19. Six node line topology with node attributes.

In order to attempt reconstructing the line topology shown in Figure 19, we start with the simple architectural design consisting of one hidden layer, with one neuron, and a learning rate of 0.01 (see Figure 20). With this configuration the algorithm arrived at a minimum cross-entropy loss value of 0.30086634 and was not able to obtain an isomorphic graph topology reconstruction (i.e., graph edit distance [GED] of 3). We then added one more neuron to the hidden layer (see Figure 21) and obtained a minimum cross-entropy loss value of 0.002684557 and an isomorphic graph reconstruction (i.e., graph edit distance of 0; essentially identical).

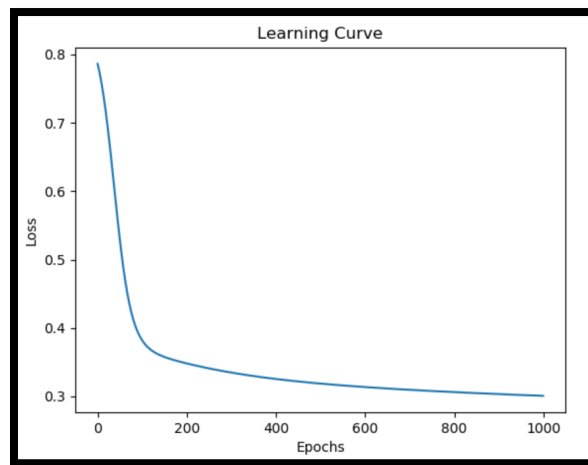


Figure 20. Learning curve for line topology with one hidden layer and one neuron architecture

(Layer size = 1 neuron, Min Loss = 0.30086634, Min GED = 3)

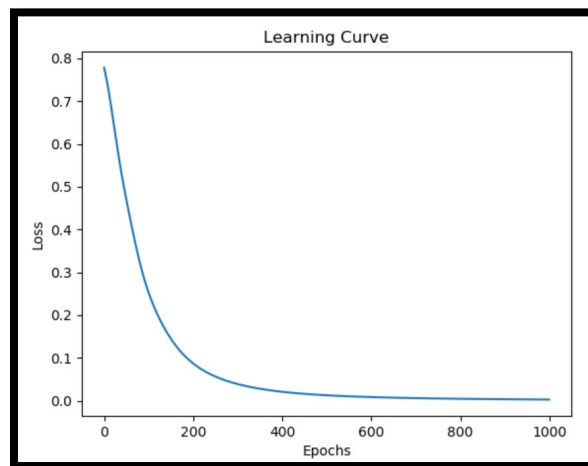


Figure 21. Learning curve for line topology with one hidden layer and two neurons architecture.

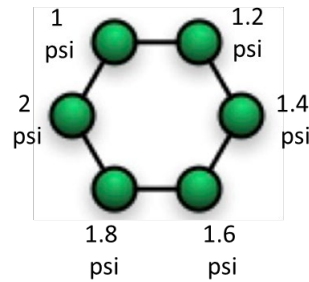
(Layer size = 2 neurons, Min Loss = 0.002684557, Min GED = 0)

Hence, using the learning curve we can conclude that the six-node line topology requires a minimum of one hidden layer with a minimum of two neurons in the hidden layer. Following the same approach used for the line topology we incrementally added neurons to the hidden layer until an isomorphic graph reconstruction was obtained. Results are shown in the sections to follow for other types of graphs.

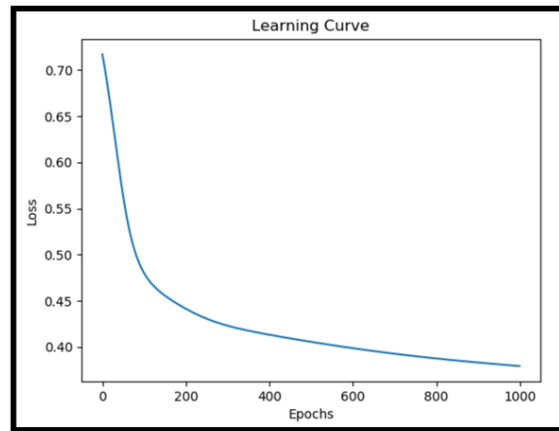
---

### 5.6.2 RING TOPOLOGY

For the ring topology shown in Figure 22, isomorphic reconstruction was obtained at a minimum cross-entropy loss of 0.0012228565 with a one layer, two neurons architecture.



**Figure 22. Six-node ring topology with node attributes.**



**Figure 23. Learning curve for ring topology with one hidden layer with one neuron architecture.**

(Layer size = 1 neuron, Min Loss = 0.37929922, Min GED = 4)

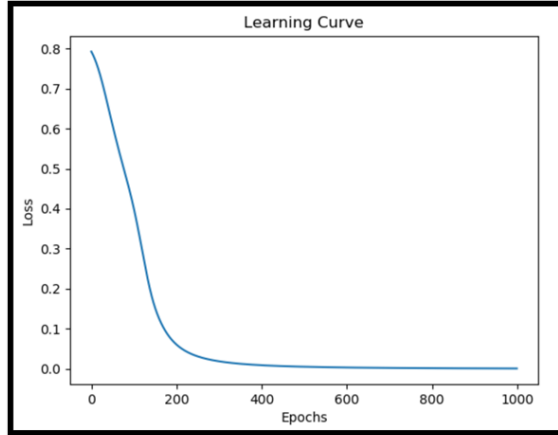


Figure 24. Learning curve for ring topology with one hidden layer with two neurons architecture.

(Layer size = 2 neurons, Min Loss = 0.0012228565, Min GED = 0)

### 5.6.3 FULLY CONNECTED (FC) TOPOLOGY

For the FC topology shown in Figure 25, isomorphic reconstruction was obtained at a minimum cross-entropy loss of 0.0002543262 with a one layer, one neuron architecture.

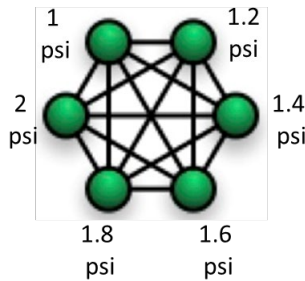


Figure 25. Six node FC topology with node attributes.

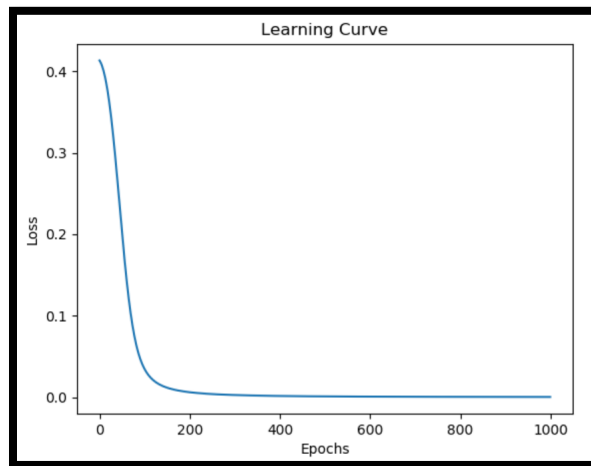


Figure 26. Learning curve for FC topology with one hidden layer with one neuron architecture.

(Layer size = 1 neuron, Min Loss = 0.0002543262, Min GED = 0)

#### 5.6.4 STAR TOPOLOGY

For the star topology shown in Figure 27, isomorphic reconstruction was obtained at a minimum cross-entropy loss of 0.0017013812 with a one layer, five neurons architecture.

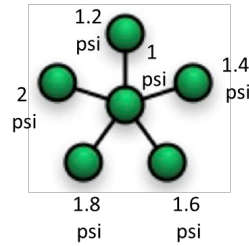


Figure 27. Six node star topology with node attributes.

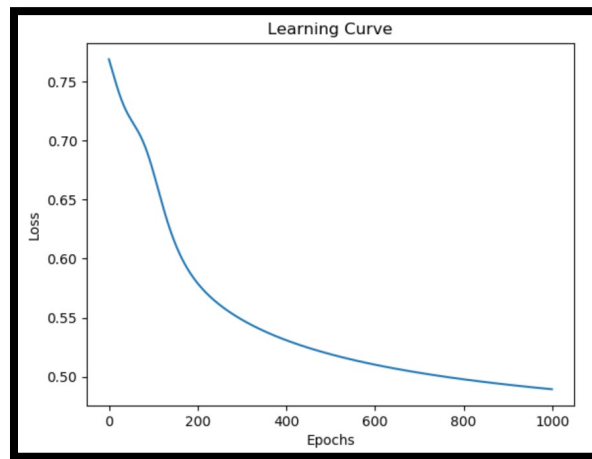


Figure 28. Learning curve for star topology with one hidden layer with one neuron architecture.

(Layer size = 1 neuron, Min Loss = 0.4893911, Min GED = 10)

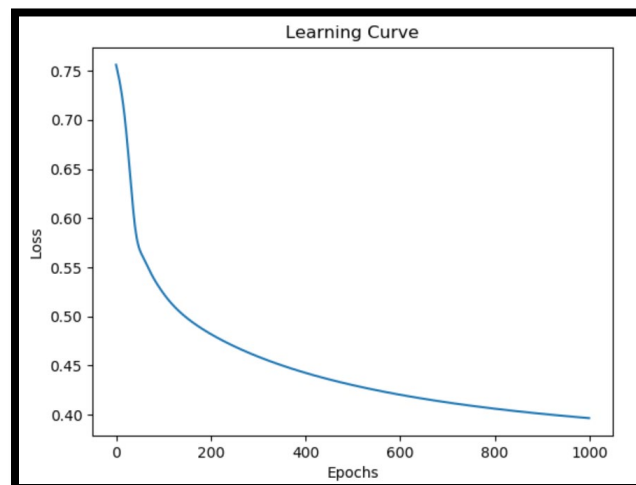
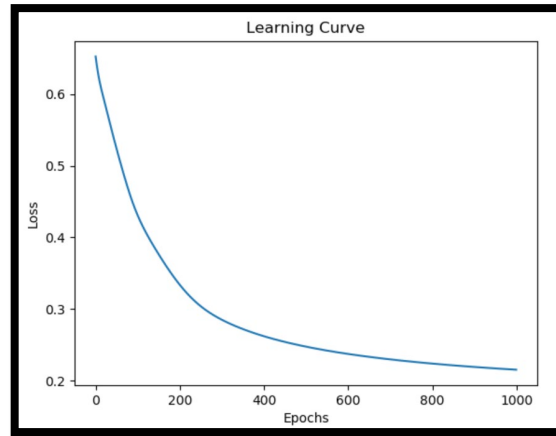


Figure 29. Learning curve for star topology with one hidden layer with two neurons architecture.

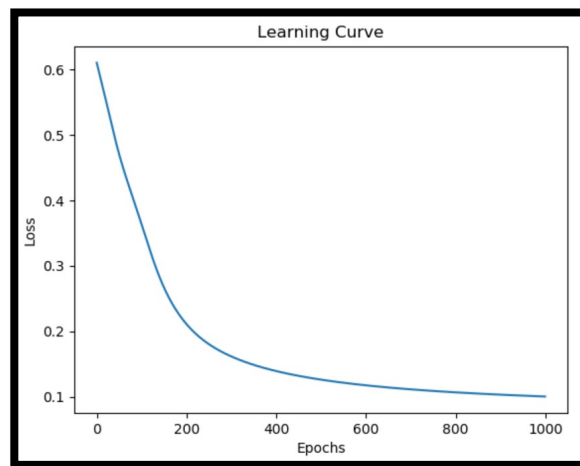


(Layer size = 2 neurons, Min Loss = 0.3965266, Min GED = 4)



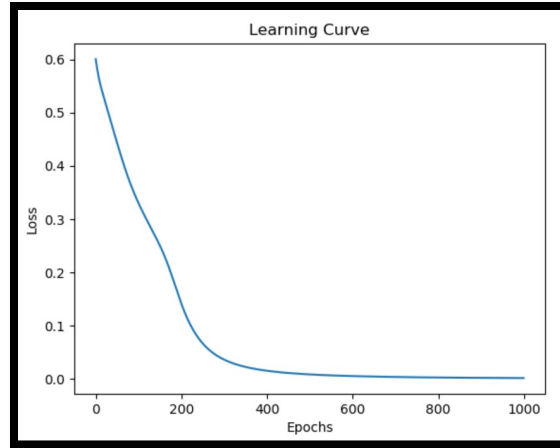
**Figure 30. Learning curve for star topology with one hidden layer with three neurons architecture.**

(Layer size = 3 neurons, Min Loss = 0.21524647, Min GED = 3)



**Figure 31. Learning curve for star topology with one hidden layer with four neurons architecture.**

(Layer size = 4 neurons, Min Loss = 0.1004491, Min GED = 1)

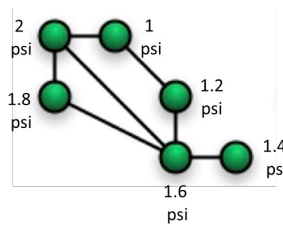


**Figure 32. Learning curve for star topology with one hidden layer with five neurons architecture.**

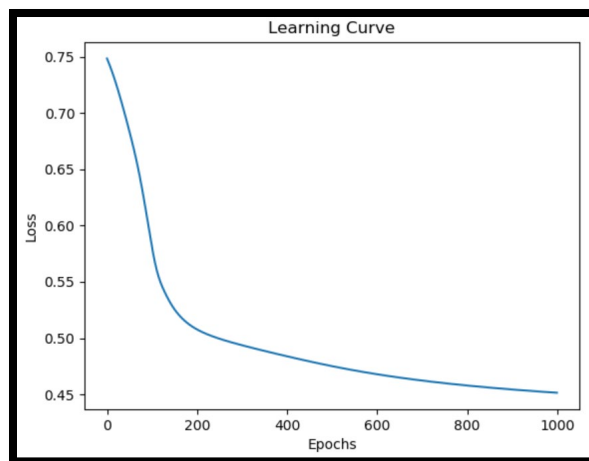
(Layer size = 5 neurons, Min Loss = 0.0017013812, Min GED = 0)

### 5.6.5 MESH TOPOLOGY

For the mesh topology shown in Figure 33, isomorphic reconstruction was obtained at a minimum cross-entropy loss of 0.0047862437 with a one layer, three neurons architecture.

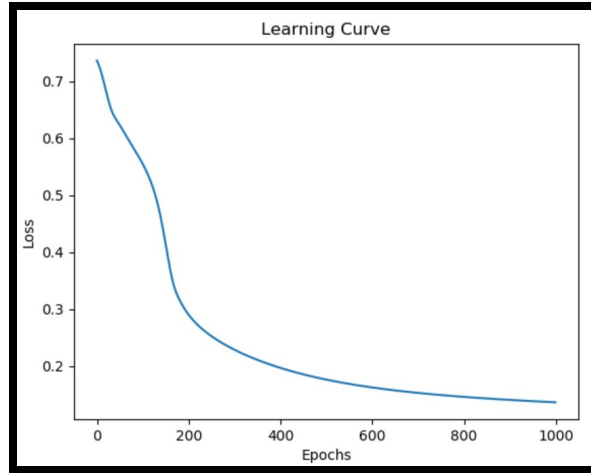


**Figure 33. Six-node mesh topology with node attributes.**



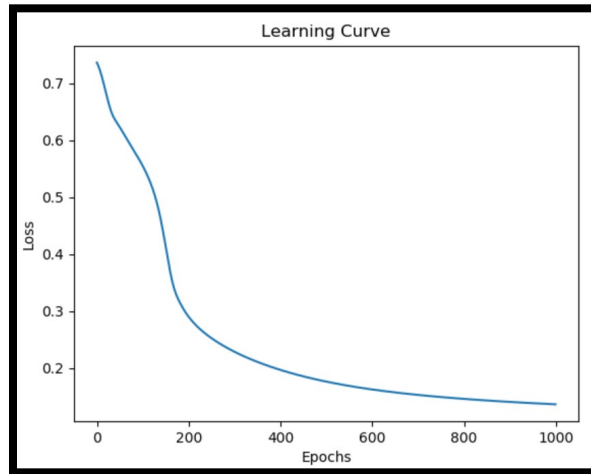
**Figure 34. Learning curve for mesh topology with one hidden layer with one neuron architecture.**

(Layer size = 1 neuron, Min Loss = 0.45157686, Min GED = 4)



**Figure 35. Learning curve for mesh topology with one hidden layer with two neurons architecture.**

(Layer size = 2 neurons, Min Loss = 0.13661459, Min GED = 1)

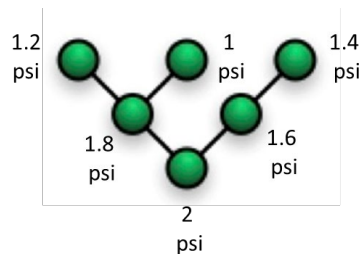


**Figure 36. Learning curve for mesh topology with one hidden layer with three neurons architecture.**

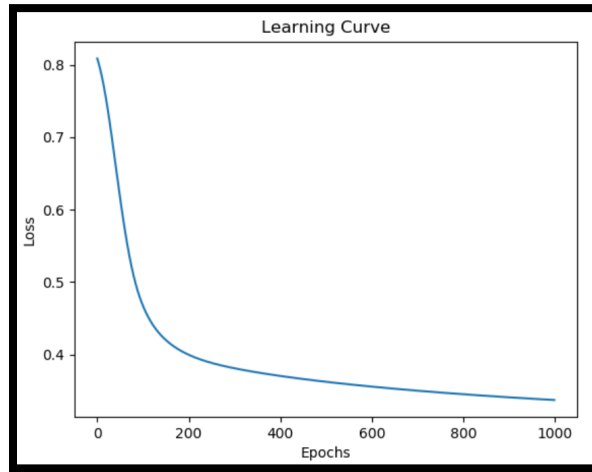
(Layer size = 3 neurons, Min Loss = 0.0047862437, Min GED = 0)

### 5.6.6 TREE TOPOLOGY

For the tree topology shown in Figure 37, isomorphic reconstruction was obtained at a minimum cross-entropy loss of 0.0025835969 with a one layer, three neurons architecture.

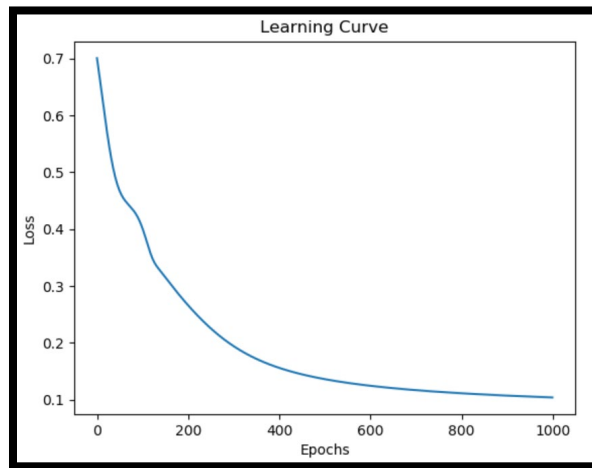


**Figure 37. Six-node tree topology with node attributes.**



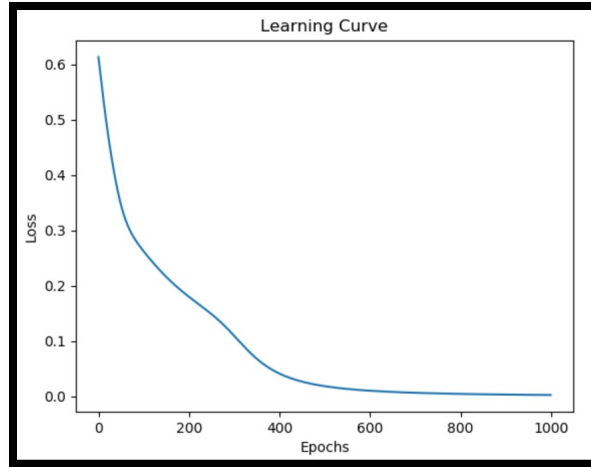
**Figure 38. Learning curve for tree topology with one hidden layer with one neuron architecture.**

(Layer size = 1 neuron, Min Loss = 0.33715013, Min GED = 4)



**Figure 39. Learning curve for tree topology with one hidden layer with two neurons architecture.**

(Layer size = 2 neurons, Min Loss = 0.10415384, Min GED = 1)



**Figure 40. Learning curve for tree topology with one hidden layer with three neurons architecture.**

(Layer size = 3 neurons, Min Loss = 0.0025835969, Min GED = 0)

---

## 5.7 URBAN TOPOLOGY

We now have learned through experimenting with different type of graph topologies, and we look to apply this knowledge to a larger problem that relates back to our case study. In order to access the scalability of our model we use a water network topology with pressure values as node attributes. The topology was obtained from the hydraulic simulation software EPANET, contains 36 nodes, and is shown in Figure 41. An isomorphic reconstruction was obtained at a minimum cross-entropy loss of 0.00196 with a one layer, 74 neurons architecture. As shown in the results in Figure 42, the approach shows that we can scale to larger graphs. The key question is: can we determine the number of neurons needed in advance based on the input?

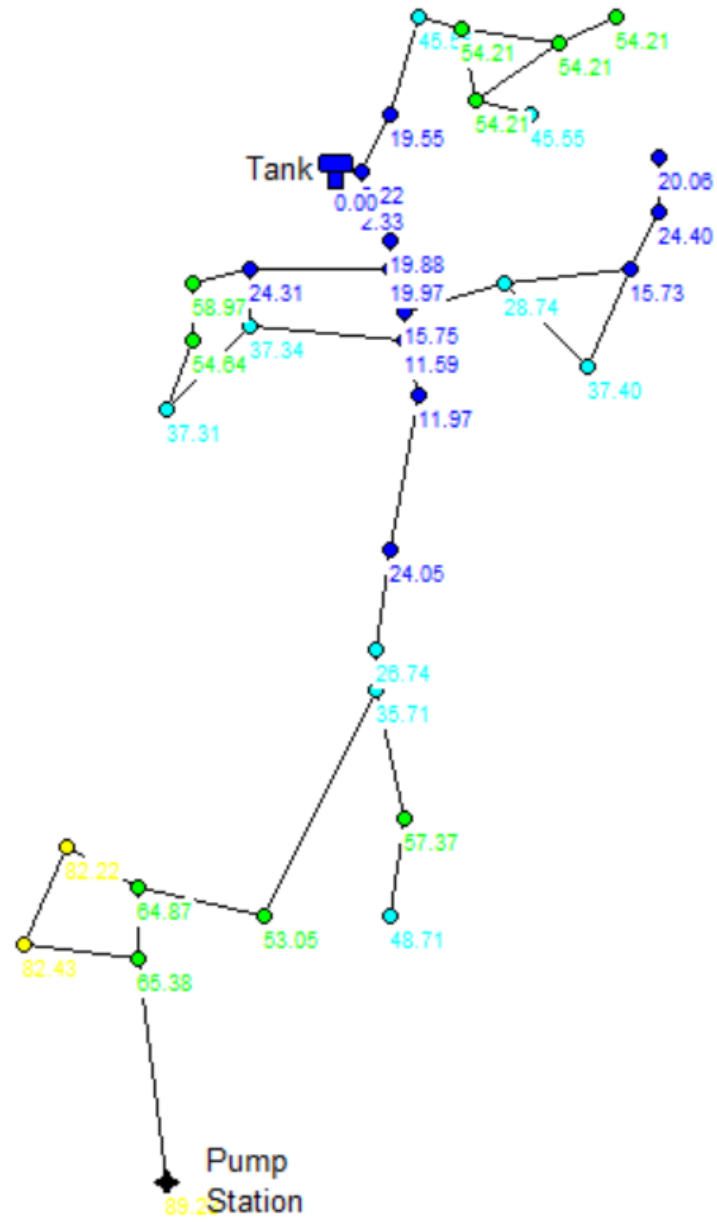


Figure 41. Urban topology with node attributes.

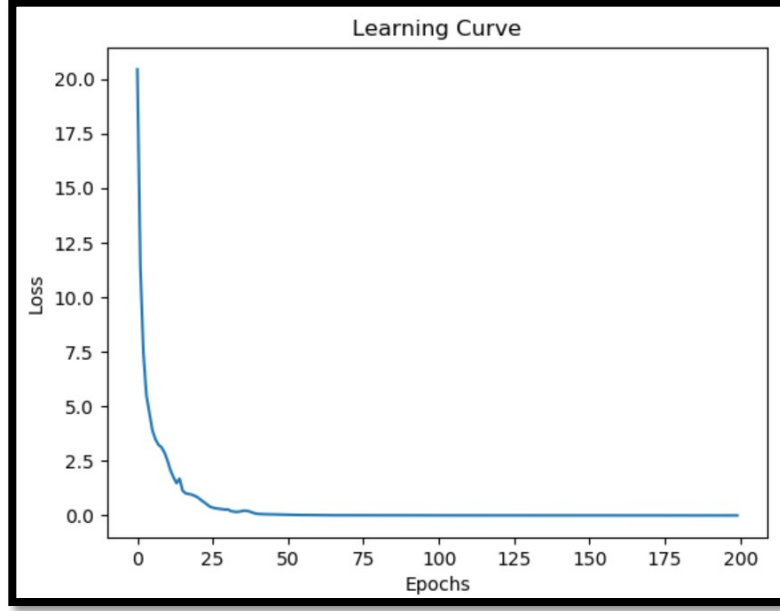


Figure 42. Learning curve for urban topology graph and one hidden layer containing 74 nodes.

## 5.8 GAE ARCHITECTURE AND THE OPTIMIZATION ALGORITHM

For most of the use cases used in this report, the GAE algorithm was able to reconstruct the input graph with one convolutional layer and no activation functions as discussed in the prior subsections. Hence, this will be the general encoder architecture used moving forward. With this set up, the encoder consists of one graph convolutional layer and a simple inner product decoder, which aims to decode node relational information from generated embeddings by reconstructing the graph adjacency matrix. The convolutional layer takes as input the graph's node feature matrix  $X$  and the symmetrically normalized adjacency matrix  $\tilde{A} = D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$ , where  $A$  is the adjacency matrix with added self-connections and  $D$  is the diagonal node matrix of  $A$ . This convolutional layer generates a lower dimensional node embedding defined as  $Z = \tilde{A}XW$ , where  $W$  is a trainable parameter matrix.

The purpose of the decoder is to reconstruct the adjacency matrix  $A$  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  $\sigma$  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  $W$  parameters are systematically updated through an Adam optimization of the weighted cross-entropy loss  $L$  between the adjacency matrix  $A$  and the soft reconstruction  $A'$ .

NOTE: The weighted cross-entropy loss  $L$  equations have been removed until this research has been published in a journal to support our co-author, Maria Coelho's PhD dissertation. The Adam optimization uses a combination of extensions to stochastic gradient descent (i.e., adaptive

gradient estimation and root mean square propagation) to estimate gradients and their moments as a moving average. These gradient estimates allow us to find the direction in which the weight parameters  $W$  should be adjusted in order to reduce the weighted cross-entropy loss.

---

## 5.9 THE HESSIAN AND THE OPTIMIZATION ALGORITHM

Our ongoing research is considering the “shape” of the weighted cross-entropy loss function (i.e. whether it is convex or non-convex) to identify if the optimization has reached a minimum or a saddle point. In 1-variable calculus, we can look at the second derivative at a point and tell what is happening with the concavity of a function (positive implies concave up, negative implies concave down). In multivariate calculus the Hessian matrix is the equivalent to the second derivative, as it contains all the second partial derivatives of a function. The eigenvalues of the Hessian give information about the second partial derivatives of the underlying function along specific directions:

- If the sign of the eigenvalue is positive, the function is convex (i.e., multivariable equivalent to “concave up”) along that direction.



- If the sign of the eigenvalue is negative, the function is concave (i.e., multivariable equivalent to “concave down”) along that direction.



- If the eigenvalues are mixed (some positive, some negative), you have a saddle point.



We have found an expression for the Hessian matrix of the weighted cross-entropy loss function. The Hessian matrix can reveal information about the shape of the function and ensure the optimization algorithm has reached a minimum instead of a saddle point. We can provide additional details on the Weighted Cross-Entropy Loss Hessian equation that support the analysis the shape of the cross-entropy cost function after our research is published.

---


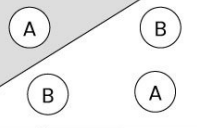
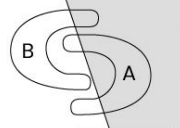


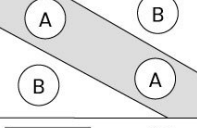
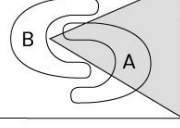
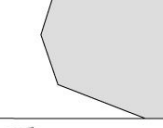

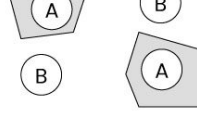
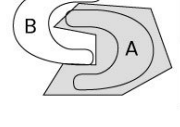
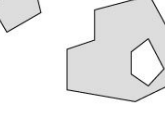
## 5.10 NEURAL NETWORK ARCHITECTURE FOR CLASSIFICATION

As GAE is a neural network-based algorithm. A first step towards achieving our objective is to understand how neural networks learn graph topology. We can divide this task into two sub-tasks: (1) understanding the role of neurons in the learning, and (2) understanding the role of layers in the learning.



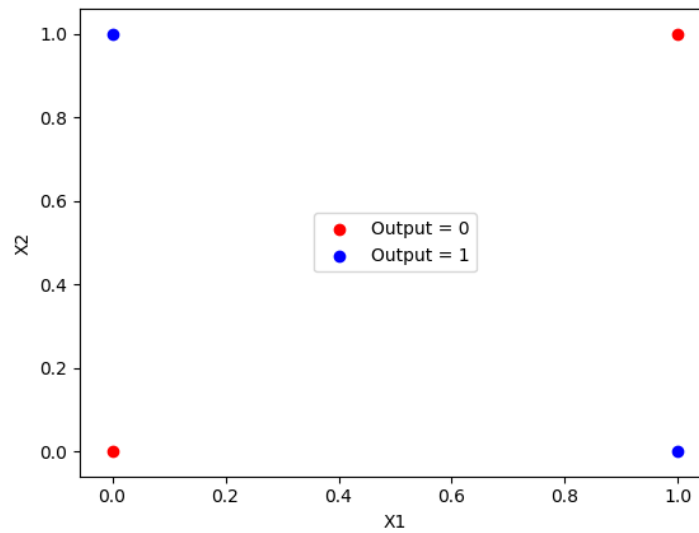
It was shown in Figure 43 by Lippmann in his work “*An Introduction to Computing with Neural Nets*” presented in 1987 [46] that:

- Neural networks with no hidden layers are capable of solving only linearly separable problems, correctly classifying data sets where the classes can be separated by one decision plane.
- Neural networks with one hidden layer and two hidden layers can approximate any desired bounded continuous function. The neurons in the first hidden layer generate decision planes to divide the input space. Neurons in the second hidden layer form regions as intersections of these decision planes.
- Output neurons form unions of the regions.

	Types of Decision Regions	Exclusive-OR Problem	Classes with Meshed Regions	Most General Region Shapes
<b>Single-Layer</b> 	Half Plane Bounded by Hyperplane			
<b>Two-Layer</b> 	Convex Open or Closed Regions			
<b>Three-Layer</b> 	Arbitrary (Complexity Limited by No. of Nodes)			

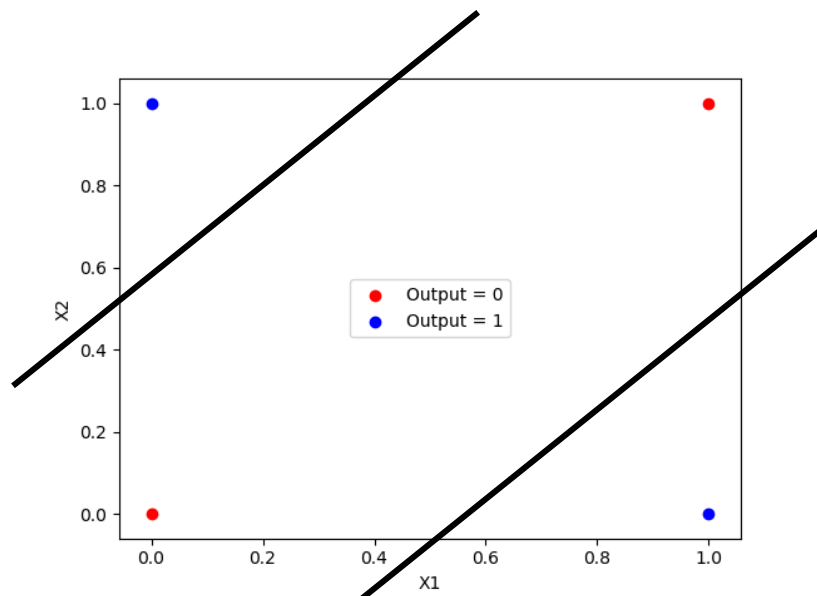
**Figure 43. An Introduction to Computing with Neural Nets**

The topology learning problem can be seen as a two-class classification problem, we expect the system to output 1 for existing links or 0 for non-existing links. A famous two class classification problem in the domain of AI is the classical XOR problem. It has two input values (0 or 1), and outputs a value of 0 or 1 depending on the combination of the inputs. It can be depicted graphically as follows:



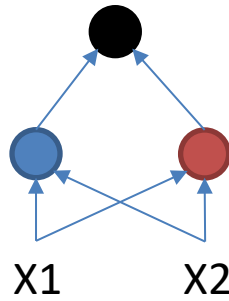
**Figure 44. Two Class Classification using XOR Example**

The neural network needs to produce two different decision planes to separate the input data based on the output classification as reflected in Figure 45.



**Figure 45. Decision Planes**

Therefore, the XOR problem can be solved with one hidden layer with two nodes (since two decision planes are needed); and one output layer with one node. The neural network architecture is shown in Figure 46:



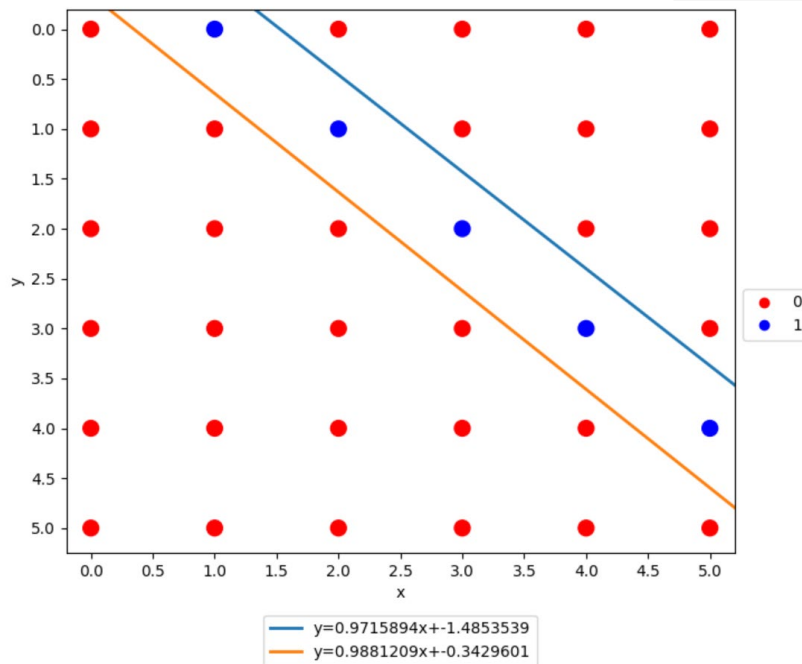
**Figure 46. Representation of Neural Network Architecture**

Learning a network topology can be framed similarly to the XOR problem. Imagine a directed line network with six nodes:



Its topology can be represented by an adjacency matrix:  $A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$

The adjacency matrix can be graphically represented as shown in Figure 47:



**Figure 47. Representation of Classification for Adjacency Matrix**

The two decision planes were derived through the same neural network architecture as the XOR problem. Hence the number of neurons in a neural network hidden layer is determined by the

number of decision planes required to separate the classes. This is the basic approach we are applying to other graphs. We started using this approach on the examples discussed in Section 5.3. For some types of graphs the classification results was not as clean as that shown in Figure 47, and we investigated other algorithms for matrix reordering, which have positive results.

## **6 DELIVERABLES AND EVENTS**

---

We have provided all of the required deliverables, which include:

- A008 Status report (bi-monthly)
- A009 Technical and Management Workplan
- A013 Final Technical Report (this report) – Report will include recommendations on next steps for research towards potential future deployment
- Kick off meeting – required before work starts (Done)

In addition, our journal papers are cited in this report, and can be provided to our sponsor on request. We anticipate that Maria Coelho's PhD dissertation work will result in two more journal papers beyond the conclusion of this one-year study.

We have presented our research at the SERC Sponsor Review on November 18, 2020. These presentations were expanded from the AI4SE/SE4AI Research Workshop in October 2020. A copy of Maria Coelho's PhD Dissertation (currently scheduled for completion, May 2022), journal and conference papers, demonstrations of twin capability, and computer-based models and software tools will be provided to the sponsor on request. We hope that these two events provided the necessary characterizations to show case our research for AI-ML design methods.

## **7 SUMMARY**

---

Our long-term research objective is design of digital twins that work as operating systems, with AI and ML formalisms working side-by-side as a team providing complementary and supportive roles in the collection of data, identification (or prediction) of events, and support for automated decision making throughout the system lifecycle. The current project has been motivated by the common characteristics and needs of two case study applications: (1) Skyzer UAV search and rescue, and (2) vehicle traversal at a traffic intersection. These characteristics and needs are: (1) presence of multiple domains, (2) event-driven behaviors, (3) streams of heterogeneous data and (4) scenarios that are dynamic and time critical. To fully address these challenges, contributions are needed on AI and ML formalisms to deal with: (1) system structure, and (2) event- and data-driven system behaviors. Our concerns for this project have been on finding ways to represent and work with the system structure of graphs and networks.

On the semantic modeling and reasoning side, basic questions include addressed how to represent various types of graph and create hooks to backend computational support for graph analysis. We have proposed a simple graph ontology that maps directly to JGraphT, a powerful software for graph analysis, and demonstrated that graph models and path planning algorithms can respond to changing environmental conditions.

Machine learning is the most interesting and challenging part of the proposed framework. As a starting point, this project has focused on the challenge of learning the structure of networks and graphs. It is important to note that because the motivating case study applications are safety and life critical, if a machine is going to learn the structure of a graph, then that representation needs to be accurate and complete. This requirement sets our work apart from graph learning procedures for social networks, which can tolerate approximations.

To summarize the efforts and accomplishment, we have:

- Analyzed different graph embedding approaches and explored the role that Graph Autoencoders (GAE) can play in the representation of network structures.
- Found that guarantees of correct reconstruction of the graph topology and attributes using the GAE framework depend on the convergence of the optimization algorithm, the encoder architecture, and the decoder architecture.
- Investigated how learning curves can help diagnose underfit, overfit and optimization convergence issues, and assist in the design of the GAE architecture.
- Investigated requirements for convergence of the optimization algorithm.
- Devised an initial formula that allows us to look at the input graph and determine the neural network architecture that is required to reconstruct it precisely with a high degree of certainty.
- Discovered that from a numerical analysis perspective, some of the early efforts in machine learning for graphs structures are, in fact, too conservative. This observation stems from the well-known XOR / XNOR problem, and analogies that can be drawn to binary matrix representations of connectivity relations in graphs.

We have made unique progress and have been able to share work in SERC events such as the workshop on AI4SE and SE4AI. Our preliminary results have already been published in the International Journal of Advances in Network and Services [19].

---

## 7.1 NEXT STEPS

One of the hallmarks of good research is solving problems, which, in turn, open the door to even more interesting and challenging problems. To that end, we certainly have a much better understanding of the opportunities and challenges ahead than a year ago.

While our graphs of interest may not be of the order of billions of nodes (e.g., think Facebook), graphs and network models in model-based engineering and urban setting could still be very large. Graph representations need to be efficient, scalable, and easily amenable to extension. The latter suggests a strong need for compositional approaches to the learning and assembly of network models. Future work should include development of a formula that characterizes the neural network size (no of neurons and no of layers) needed to learn a graph structure. We already know that a minimal neural network architecture depends on the ordering of the vertices, so an associated problem is development of algorithms to bundle the adjacency relations into clusters (or islands). A first cut at this problem would minimize the number and shapes of the islands. Knowing the number of clusters present in the reordered adjacency is

important because it will determine the number of decision planes required to classify the data, and hence determine the number of neurons/layers required to build a good neural network.

To create fully operational digital twin demonstrations, future work also needs to address ways in which AI and ML can work together to support event- and data-driven decision making. Given that the semantic side of the formulation is event driven (i.e., semantic graphs respond to events), the pivotal role for ML is in sensing incoming data streams and identifying either objects or anomalies in behavior that require attention. These objects could be in an image, sensed from audio (e.g., a vehicle at an intersection hears an ambulance) or inferred from a sensed data stream. Anomalies in behavior can be modeled with recurrent neural networks coupled with statistical algorithms for the identification of outlier measurements.

Finally, work is needed to connect the ML and AI modules (i.e., create the red arrows in Figure 2 and Figure 3) and determine a protocol for transmission of events. We are hopeful that if a machine has an understanding of the network structure is spatially and temporally aware, then it will also have the ability to send events that are more precise than would otherwise be possible.

## 8 ACRONYMS AND ABBREVIATION

---

This section provides a list of some of the terms used throughout the paper. The model lexicon should have all of these terms and many others.

2D	Two dimensions
3D	Three dimensions
AI	Artificial Intelligence
AI4SE	Artificial Intelligence for Systems Engineering
ConvGNN	convolutional graph neural networks
DANE	Dynamic Attribute Network Embedding
DSM	Digital System Model
DE	Digital Engineering
DoD	Department of Defense
GAE	Graph Autoencoder
GNN	Graph Neural Networks
IEEE	Institute of Electrical and Electronics Engineers
JTS	Java Topology Suite
MBE	Model Based Engineering
MCE	Model Centric engineering
MBSE	Model Based System Engineering
MCE	Model-Centric Engineering
ML	Machine Learning
NDIA	National Defense Industrial Association
OSM	Open Street Map
OWL	Web Ontology Language
RecGNN	recurrent graph neural networks
RDF	Resource Description Framework
RT	Research Task
SE	System Engineering
SERC	Systems Engineering Research Center
SE	System Engineering
SE4AI	Systems Engineering for Artificial Intelligence
SPARQL	SPARQL Protocol and RDF Query Language
STGNN	spatial-temporal graph neural networks
SW	Software
SWT	Semantic Web Technology
UAV	Unmanned Aerial Vehicle
WRT	Washington Research Task

## 9 TRADEMARKS

---

Cameo Simulation Toolkit is a registered trademark of No Magic, Inc.

CREO is a registered trademark of PTC Corporation.

Java™ and J2EE™ are trademark of SUN Microsystems

Java is trademarked by Sun Microsystems, Inc.

XML™ is a trademark of W3C

All other trademarks belong to their respective organizations.



## 10 REFERENCES

---

- [1] Abdulali, Arsen, Seokhee Jeon. Data-Driven Modeling of Anisotropic Haptic Textures: Data Segmentation and Interpolation. In *Haptics: Perception, Devices, Control, and Applications: 10th International Conference, EuroHaptics 2016, London, UK*, pp. 228–239. Springer International Publishing, 2016.
- [2] Agatsuma, Shotaro, Junya Kurogi, Satoshi Saga, Simona Vasilache, and Shin Takahashi. Simple Generative Adversarial Network to Generate Three-axis Time-series Data for Vibrotactile Displays. In *Proceedings of International Conference on Advances in Computer-Human Interactions, ACHI 2020*, pp. 19–24, March 2020.
- [3] Apache Jena. Java Framework for Building Semantic Web and Linked Data Applications. See: <http://jena.apache.org> (Accessed, July 6, 2020).
- [4] Arp, Robert, Barry Smith, and Andrew D. Spear. 2015. Building Ontologies with Basic Formal Ontology Mit Press.
- [5] Austin, M., M. Coelho, M. Blackburn, Architecting Digital Twins for Model-Centric Engineering: A Combined Semantic Modeling and Machine Learning Approach, SERC, WRT-1011, December 2019.
- [6] Austin, M., M. Coelho, M. Blackburn, Using AI/ML Design Patterns for Digital Twins and Model-Centric Engineering, 12th Annual SERC Sponsor Research Review, November 18, 2020.
- [7] Austin M.A., Coelho M.C., and Blackburn M.R., Semantic Modeling and Event-Driven Execution of Multi-Domain Systems and System of Systems with the Data-Ontology-Rule Footing, System Engineering, Submitted June 2018.
- [8] Baker, A., K. Pepe, N. Hutchinson, M. Blackburn, R. Khan, R. Peak, J. Wade, C. Whitcomb, Enabling the Digital Transformation of the Workforce: A Digital Engineering Competency Framework, 2021 IEEE International Systems Conference (SysCon), 2021.
- [9] Behrisch, M., B. Bach, N. H. Riche, T. Schreck, and J.-D. Fekete, “Matrix Reordering Methods for Table and Network Visualization,” *Computer Graphics Forum*, vol. 35, no. 3, pp. 693–716, 2016.
- [10] Blackburn, M., Roadmap Concept with OpenMBEE and Cyber Ontology Example, ACM / IEEE 23rd International Conference on Model Driven Engineering Languages and Systems (MODELS), October 2020.
- [11] Blackburn, M., B. Kruse, W. Stock, M. Ballard, Digital Engineering Modeling Methods for Digital Signoffs, NDIA Systems and Mission Engineering Conference, November 2020.
- [12] Blackburn, M., D. Verma, Digital Engineering: Linking Mission Level Behavior Modeling with Component Level Performance Modeling, Design Reasoning across Multiple Domains and Disciplines – and Multiple Abstraction Levels, Navy SERC Tech Talk, June 10, 2020.
- [13] Blackburn, M., D. Verma, Ontologies for Engineering: A Pragmatic Perspective, Navy SERC Tech Talk, October 21, 2020.
- [14] Blackburn, M., D. Verma, Digital Engineering: Acquisition Source Selection and Design Reviews, Navy - SERC Tech Talk, July 15, 2020.
- [15] Blackburn, M., Austin, M., M. Coelho, Using AI/ML Design Patterns for Digital Twins and Model-Centric Engineering, AI4SE/SE4AI Workshop, October 2020.
- [16] Blackburn, M., R., M. A. Bone, J. Dzielski, B. Kruse, R. Peak, S. Edwards, A. Baker, M. Ballard, M. Austin, M. Coelho, D. Rhodes, B. Smith, Transforming Systems Engineering through Model-Centric Engineering, Final Technical Report SERC-2019-TR-103, RT-195 (NAVAIR), May 28, 2019.
- [17] Blackburn, M., P. Denno, Using Semantic Web Technologies for Integrating Domain Specific Modeling and Analytical Tools, Complex Adaptive Systems Conference, Nov. 2015.
- [18] Cai H., Zheng V.W., and Chang K.C., “A Comprehensive Survey of Graph Embedding: Problems, Techniques and Applications,” *IEEE Transactions on Knowledge and Data Processing*, vol. 30, no. 9, 2018.
- [19] Coelho M., Austin, M.A., Blackburn, M.R., Teaching Machines to Understand Urban Networks: A Graph Autoencoder Approach, *International Journal on Advances in Networks and Services* Volume 13, Number 3 & 4, December 2020.
- [20] Coelho M., Austin M.A., and Blackburn M.R., The Data-Ontology-Rule Footing: A Building Block for Knowledge-Based Development and Event-Driven Execution of Multi-Domain Systems, *Systems Engineering in Context - Proceedings of the 16th Annual Conference on Systems (CSER 2018)*, Springer, Charlottesville, VA, May 8-9 2018.

- [21] Coelho M., Austin M.A., and Blackburn, M.R. Distributed System Behavior Modeling of Urban Systems with Ontologies, Rules and Many-to-Many Association Relationships. The 12th International Conference on Systems (ICONS 2017), pages 10–15, April 23-27 2017.
- [22] Coley C.W., Jin W., Rogers L., Jamison T.F., Jaakkola T.S., Green W.H., Barzilay R., and Jenson K.F., A Graph-Convolution Neural Network Model for the Prediction of Chemical Reactivity, *Chemical Science*, Vol. 10, 2019, pp. 370-377.
- [23] Common Core Ontologies <http://www.ontologyrepository.com>
- [24] Delgoshaei P. and Austin M.A. , Framework for Knowledge-Based Fault Detection and Diagnostics in Multi-Domain Systems: Application to Heating Ventilation and Air Conditioning System, *International Journal on Advances in Intelligent Systems*, Vol. 10, No 3 and 4, December 2017, pp. 393-409.
- [25] Delgoshaei P., Heidarinejad M., and Austin M.A., Combined Ontology-Driven and Machine Learning Approach to Management of Building Energy Consumption," 2018 Building Performance Analysis Conference and SimBuild, Chicago, September 26-28, 2018.
- [26] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations*, 2015.
- [27] Engel, Jesse, Kumar Krishna Agrawal, Shuo Chen, Ishaan Gulrajani, Chris Donahue, and Adam Roberts. GANSynth: Adversarial neural audio synthesis. In *International Conference on Learning Representations*, 2019.
- [28] Glorot, Xavier, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pp. 315–323, 2011.
- [29] Goodfellow, Ian, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pp. 2672–2680, 2014.
- [30] Google Inc. Tensorflow. <https://tensorflow.org/> (accessed on 2020.11.20).
- [31] Goyal, P., & Ferrara, E. (2018). Graph embedding techniques, applications, and performance: A survey. *Knowledge-Based Systems*, 151, 78–94. <https://doi.org/10.1016/j.knosys.2018.03.022>
- [32] Grover, Aditya, Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In KDD. ACM, 855–864.
- [33] Hagedorn, T., M. Bone, B. Kruse, I. Grosse, M. Blackburn, Knowledge Representation with Ontologies and Semantic Web Technologies to Promote Augmented and Artificial Intelligence in Systems Engineering, Special Article in INCOSE INSIGHT, March 2020.
- [34] Haghighi I., Bartocci E., Jones A., Grosu R., Kong Z., and Belta C., SpaTeL: A Novel Spatial-Temporal Logic and Its Applications to Networked Systems, HSCC'15, Seattle, Washington, April 14-16, 2015.
- [35] Heaton, Jeff. "Artificial Intelligence For Humans". Heaton Research, Inc, 2015.
- [36] Herzig R., Levi E., Xu H., Gao H., Brosh E., Wang X., Globerson A., and Darrell T., Spatio-Temporal Action Graph Networks, IEEE International Conference on Computer Vision, 2019.
- [37] Geng X., Li Y., Wang L., Zhang L., Yang Q., Ye J., and Liu Y., Spatio-Temporal Multi-graph Convolution Network for Ride-Hailing Demand Forecasting, Association for Advancement in Artificial Intelligence, 2019.
- [38] Ioffe, Sergey and Christian Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *International Conference on Machine Learning*, pp. 448–456, 2015.
- [39] Java Topology Suite (JTS): A Java Library that creates an Object Model for Planar Linear Geometry.
- [40] Jenkins, J. S., N. Rouquette, Semantically-Rigorous systems engineering modeling using SysML and OWL, 5th International Workshop on Systems & Concurrent Engineering for Space Applications, Lisbon, Portugal, October 17-19, 2012.
- [41] JGraphT. Open-Source Java Library of Graph Theory Data Structures and Algorithms. See: <http:jgrapht.org> (Accessed, July 6, 2020).
- [42] Kingma, D. J. Ba, "Adam: A Method for Stochastic Optimization," International Conference on Learning Representations, December 2014
- [43] Kipf, T. N., & Welling, M. (2016). Variational Graph Auto-Encoders. arXiv preprint arXiv:1611.07308.
- [44] Ledig, Christian, Lucas Theis, Ferenc Huszar, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. In *Proceedings of the IEEE Conference on Computer*

- Vision and Pattern Recognition*, pp. 4681–4690, 2017.
- [45] Lee, Kathryn A., Gregory Hicks, and German Nino-Murcia. Validity and reliability of a scale to assess fatigue. *Psychiatry Research*, Vol. 36, No. 3, pp. 291–298, 1991.
  - [46] Lippmann, R., "An introduction to computing with neural nets," in IEEE ASSP Magazine, vol. 4, no. 2, pp. 4–22, Apr 1987, doi: 10.1109/MASSP.1987.1165576.
  - [47] McDermott, T., E. Van Aken, N. Hutchison, M. Blackburn, M. Clifford, Y. Zhongyuan, N. Chen, A. Salado, K. Henderson, Digital Engineering Metrics, Technical Report SERC-2020-SR-002, May 22, 2020.
  - [48] McDermott, T., D. DeLaurentis, P. Beling, M. Blackburn, M. Bone, AI4SE and SE4AI: A Research Roadmap, Special Issue of INCOSE Insight (to be published 2020).
  - [49] Perozzi, Bryan, Rami Al-Rfou, and Steven Skiena. 2014. DeepWalk: Online learning of social representations. In KDD. ACM, 701–710.
  - [50] Petnga, L., M. Austin, "An ontological framework for knowledge modeling and decision support in cyber-physical systems," *Advanced Engineering Informatics*, vol. 30, no. 1, pp. 77–94, Jan. 2016.
  - [51] Mono Wireless Inc. TWE-Lite-2525A. <https://mono-wireless.com/jp/products/TWE-Lite-2525A/index.html> (accessed on 2020.11.20).
  - [52] Mirza, Mehdi, Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.
  - [53] Open Street Map (OSM). See <http://www.openstreetmap.org> (Accessed, July 6, 2020).
  - [54] "Python Lessons". Pylessons.Com, 2020, <https://pylessons.com/YOLOv3-TF2-mnist/>.
  - [55] Saga, Satoshi, Ramesh Raskar. Simultaneous geometry and texture display based on lateral force for touchscreen. In *Proceedings of IEEE World Haptics 2013*, pp. 437–442, Apr. 2013.
  - [56] Shotaro Agatsuma, Shinji Nakagawa, Tomoyoshi Ono, Satoshi Saga, Simona Vasilache, and Shin Takahashi. Classification method of rubbing haptic information using convolutional neural network. In *Proceedings of International Conference, HCI International 2018*, pp. 159–167, July 2018.
  - [57] Simonyan, Karen and Andrew Zisserman. Very Deep Convolutional Net-works for Large-Scale Image Recognition. In *International Conference on Learning Representations*, 2015.
  - [58] Srivastava, Nitish, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, Vol. 15, No. 1, pp. 1929–1958, 2014.
  - [59] Strese, Matti, Yannik Boeck, and Eckehard Steinbach. Content-based surface material retrieval. In *World Haptics Conference (WHC), 2017 IEEE*, pp. 352–357. IEEE, 2017.
  - [60] Tang, Jian, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. LINE: Large-scale information network embedding. In WWW. 1067–1077.
  - [61] Ujitoko, Yusuke and Yuki Ban. Vibrotactile signal generation from texture images or attributes using generative adversarial network. In *International Conference on Human Haptic Sensing and Touch Enabled Computer Applications*, pp. 25–36. Springer, 2018.
  - [62] Wikipedia, Ontology, [http://en.wikipedia.org/wiki/Ontology\\_\(information\\_science\)](http://en.wikipedia.org/wiki/Ontology_(information_science)), 2014.
  - [63] World Wide Web Consortium. OWL 2 Web Ontology Language Document Overview. 2009. Available from: <http://www.w3.org/TR/2009/REC-owl2-overview-20091027/>.
  - [64] World Wide Web Consortium. RDF Vocabulary Description Language 1.1: RDF Schema, February 2014 <https://www.w3.org/TR/rdf-schema/>.
  - [65] World Wide Web Consortium. SPARQL 1.1 Overview, March 2013, <http://www.w3.org/TR/sparql11-overview/>.
  - [66] World Wide Web Consortium. Turtle - Terse RDF Triple Language, 28 March 2011, <http://www.w3.org/TeamSubmission/turtle/>.
  - [67] Witten I.H., Frank E., Hall M.A., and Christopher J.P. Data Mining: Practical Machine Learning Tools and Techniques. Morgan Kauffmann, 2017.
  - [68] Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., & Yu, P. S. (2020). A Comprehensive Survey on Graph Neural Networks. *IEEE Transactions on Neural Networks and Learning Systems*, 1–21. <https://doi.org/10.1109/tnnls.2020.2978386>
  - [69] Yu B., Yin H., and Zhu Z., Spatiotemporal Graph Convolutional Networks: A Deep Learning Framework for Traffic Forecasting, *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence (IJCAI-18)*, 2018.