# APPLICATION OF GRAPH NEURAL NETWORKS ON SOFTWARE MODELING

**A Thesis Submitted to**
**the Graduate School of**
**İzmir Institute of Technology**
**in Partial Fulfillment of the Requirements for the Degree of**

**MASTER OF SCIENCE**

**in Computer Engineering**

**by**
**Onur Yusuf LEBLEBİCİ**

# ACKNOWLEDGEMENTS

First, I would like to thank my supervisors, Assoc. Prof. Dr. Tuğkan Tuğlular and Prof. Dr. Fevzi Belli, who advised, motivated and shared their experiences with me. Their guidance helped me in all the time of research and writing of this thesis.

I am grateful to my sister-in-law Assoc. Prof. Dr. Gonca Özçelik Kayseri for her support.

Finally, I would like to thank my wife Sinem Leblebici for her unconditional and constant support in this process, and my dear daughters Bade and Öykü, whom I deprived of the time we spent together in that period, for their patience and support. This thesis is dedicated to my beloved family

# ABSTRACT

APPLICATION OF GRAPH NEURAL NETWORKS ON SOFTWARE MODELING

Deficiencies and inconsistencies introduced during the modeling of software systems can cause undesirable consequences that may result in high costs and negatively affect the quality of all developments made using these models. Therefore, creating better models will help the software engineers to build better software systems that meet expectations. One of the software modelling methods used for analysis of graphical user interfaces is Event Sequence Graphs (ESG). The goal of this thesis is to propose a method that predicts missing or forgotten links between events defined in an ESG via Graph Neural Networks (GNN). A five-step process consisting of the following steps is proposed: (i) data collection from ESG model, (ii) dataset transformation, (iii) GNN model training, (iv) validation of trained model and (v) testing the model on unseen data. Three performance metrics, namely cross entropy loss, area under curve and accuracy, were used to measure the performance of the GNN models. Examining the results of the experiments performed on different datasets and different variations of GNN, shows that even with relatively small datasets prepared from ESG models, predicts missing or forgotten links between events defined in an ESG can be achieved.

# ÖZET

## GRAFİK YAPAY SİNİR AĞLARININ YAZILIM MODELLEMESİNE UYGULANMASI

Yazılım sistemlerinin modellemeleri sırasında yapılan eksiklikler ve oluşan tutarsızlıklar, bu modeller kullanılarak yapılan tüm geliştirmelerde de yüksek maliyetlerle sonuçlanan istenmeyen sonuçlara sebep olabilmektedir. Yazılım modellemesi sırasında yazılım mühendislerine verilebilecek öneriler ile daha iyi modeller oluşturulabilir ve bu sayede kullanıcı beklentilerini daha iyi karşılayan sistemler oluşturulabilir. Yazılım modellemede kullanılan yöntemlerden bir taneside, grafik kullanıcı arayüzlerinin analizinde kullanılan olay akış grafikleridir. Bu tezin hedefi olay akış grafikleri üzerinde yer alan bileşenler arasında unutulmuş veya eksik bağlantıları grafik yapay sinir ağları kullanarak tahminleyecek bir yöntem öndermektir. Bu yöntem beş basamaktan oluşan bir süreçten oluşmaktadır: (i) veri toplama, (ii) grafik yapay sinir ağ modelini eğitmek, (iii) eğitilen modeli doğrulamak ve (v) modeli daha önce görmediği veriler ile test etmek. Eğitilen grafik yapay sinir ağ modellerinin performansını ölçmek için çapraş entropi kaybı, eğri altında kalan alan ve doğruluk performans metrikleri kullanılmıştır. Farklı veri kümeleri ve farklı grafik yapay sinir ağı varyasyonları ile yapılan deneylerin incelenmesi sonucunda, nispeten küçük ölçekli veri kümelerinde dahi başarı elde edilebildiği gözlemlenmiştir.

# TABLE OF CONTENTS

# LIST OF FIGURES

**Figure** **Page**

**Figure** **Page**

**Figure** **Page**

# LIST OF TABLES

**Table** **Page**

# CHAPTER 1
# INTRODUCTION

Software applications are generally sophisticated. Those systems may require many sub-systems and components in it. In order to design a software system, details of the system are required to be understood at varying levels. Typical software modeling approaches may not be able to reduce this complexity for engineers. Predicting and giving recommendations on the connections between software components such as classes, events, UI elements, user interactions etc. has great importance in modelling. From the software engineering perspective, deciding and connecting components with each other require significant effort. It is also error prone. Instead of putting all the workload on the software engineer, providing some recommendation can help engineers with modeling the composition and interaction of components in a software system. There are many different models and tools used in software modeling. Most of these models are graph-based models and there is a well-established theory of graph transformations [1], which has a number of system modelling and software engineering applications. The graph-based modeling technique taken under consideration in this thesis is Event Sequence Graphs (ESGs) [2].

GUIs can be modeled as sequences of events of the objects defined in GUI. The operations on components of the GUI, such as buttons, lists, and checkboxes, are controlled and/or observed by input/output devices. Thus, an event can be a user input or a system response; both of them are elements of event set V and lead to a sequence of user inputs and expected desirable system outputs. An ESG is a tuple (V, E, $\Xi$, $\Gamma$), where $V \neq \emptyset$ is a finite set of nodes (vertices or events) and $E \subseteq V \times V$ is a finite set of arcs (edges), and $\Xi, \Gamma \subseteq V$ finite sets of distinguished vertices with $\xi \in \Xi$, $\gamma \in \Gamma$, called entry nodes (start events) and exit nodes (finish events), respectively [2]. The semantics of an ESG is as follows. Any $v \in V$ represents an event. For two events $v_1, v_2 \in V$, the event $v_2$ must be enabled after the execution of $v_1$ if and only if $(v_1, v_2) \in E$ [3]. ESG is chosen in this thesis as modeling technique because of its simple semantics.

There are many applications of graph-structured data, such as finding friends on social network [4], [5], molecule interactions in medicine [6], highlighting the product which a customer is interested in e-commerce [7], relation learning for knowledge graphs [8], extrapolating paths [9], metabolic network reconstructions [10]. Graph-structured

data are different from linear-structured data. Graph-structured data cannot be given to neural networks by traditional methods as in linear-structured data. The applications of neural networks on graph-structured data, is getting attraction in recent years. In 2009, Scarselli et al. proposed a method [11], which makes it possible to work neural networks on graphs. After this proposal, the studies published in this field has come infrequent until 2016, and then the studies started to gain momentum again. With many researches, graph classification, node classification, edge classification studies have been carried out using different types of neural network architectures [12][13][14][15]. Some of the architectures are as follows, recurrent neural network (RNN), convolutional neural network (CNN), autoencoders (AE), attention mechanism.

Another important application of graph neural network is link prediction. Zhang and Chen proposed a next generation approach for link prediction called Weisfeiler-Lehman Neural Machine [16], which learns the patterns on graphs to predict links. That was a game changing approach. They used fully connected neural network for learning the link existence patterns on graphs. After that Zhang and Chen improved their method and proposed deep graph convolutional neural network (DGCNN) for link prediction [17]. They extracted local enclosing subgraphs for each vertex and applied DGCNN to predict edges. Another approach is based on the representation of connections between nodes is due to the relationship between the features of the nodes, these relationships can be learned through graph neural networks (GNN). Gu et al. proposed a method [18] based on that idea an used graph attention network [15] (GAT) as GNN variation.

One of the methods used for link prediction is heuristic methods. This method, which is based on assumptions such as having common neighbors, can be used to suggest friends in social networks, but does not show any success in predicting molecule connections[6] or extrapolating paths [9]. ESGs are considered analogous to molecules, where events are like atoms and their different combination means a new model.

In this thesis, an application of graph neural network (GNN), which predicts missing or unnecessary links between events defined in an ESG, is introduced. For an ESG, a link means a transition between two events. Experiments were performed on four different datasets with two different GNN variations to predict links that have not been seen before. The following steps were followed in the experiments: data collection, data transformation, model training, validation of the trained model, and measurement of the performance.

The motivation in this thesis is to help software engineers during system modeling. With the developed approach, errors that may arise from models will be prevented or reduced and quality will be increased. Since these models are used for coding, testing and design in the software development processes later, and any deficiencies and errors may occur in this process can cause very high costs. Modeling quality directly affects the quality of the system.

The outline of the thesis is as follows. Chapter 2 provides fundamental information about terms and terminologies used in this thesis. Chapter 3 gives an overview of preliminary research on link prediction using GNN. Chapter 4 introduces steps of the developed approach with, Chapter 5 presenting the evaluations over different datasets and different GNN models using proposed approach. The last Chapter provides conclusions and possible future works.

# CHAPTER 2
# FUNDAMENTALS

## 2.1. Neural Networks

Machine learning is used to find patterns in data represented by numbers. Neural networks are models that learn the nonlinear relationship between input and output. The basic methods used to train neural networks can be listed as follows.

**Supervised learning:** It is a learning method in which the expected output based on the given input is predefined and these definitions are used to train the model. In other words, this method is used to generate a function that produces desired outputs with respect to the given inputs.

**Unsupervised learning:** This learning method tries to learn groups according to the characteristics of the given data. There is no labeling process for data, only data. They are used to solve problems such as clustering, dimensionality reduction.

**Semi-Supervised learning:** This learning method is used in cases where only a certain amount of data is labeled in the existing data set. Labeled data used through the supervised learning, while the un-labeled data used through the unsupervised learning. The purpose here is to guess the labels of un-labeled data.

**Reinforcement learning:** The main purpose in this learning method is to win the game. A method called policy is used, in which the agent reacts according to the environment and receives feedback from the environment. The agent tries different actions each time and must learn to make the best choice according to the reward-penalty system.

Problem domains handled by neural networks can be listed as follows.

**Classification:** The classification problem occurs when one or more labels needed to be generated as output. Neural network model makes predictions based on previous observations. The purpose of a neural network model specialized for classification is to approximate the function that describes the discrete outputs based on the given input values. Predicting the plant species can be given as an example of such problems.

**Regression:** The regression problem occurs when continuous value needed to be generated as output. Neural network model makes predictions based on previous

observations. The purpose of a neural network model specialized for regression is to approximate the function that describes the continues numeric outputs based on the given input values. Calculating the risk ratio for insurance can be given as an example for such problems.

**Clustering:** It is the grouping of data with similar characteristics in a data set. There are many similarities among clusters created by unsupervised learning, but less similarities between different clusters.

## 2.2. Neural Network Models

**Feed Forward Neural Networks (FFNN) and Multilayer Perceptron** [19] [20]**:** This kind of neural network models consist of layers and these layers are named as input, hidden and output. Models consist of 1 input, 1 output layer and 0 or more hidden layers. Generally, each layer fully connected to the next. They feed information from input to output. The simplest model is logic gate and it has two input cells and one output cell.

**Convolutional Neural Networks (CNN)** [21]**:** They are different from other artificial neural networks. Although they are generally used for image classification purposes, there are many different usage areas. CNN processes a given input by passing it through the following steps; there are convolution layer, non-linear function and pooling layer those are arranged one after the other. After passing through those layers, data pass through the final fully connected layer and generates numbers on output layer that explain the possibility of being a certain class. The Convolution layer is the basic building component of CNNs where the most calculations are made. In the conversion of the convolution layer, each learnable filter with size $n \times n \times k$ (where n is height-width and k is depth) is slide (with a given factor) on the input data. The filter and the input data segment at the current position and size of the filter are subjected to matrix operation and an activation map is created as output. The point is that learnable filters are activated when they see certain patterns. ReLU is widely used as a non-linear function. In the pooling layer, the method of max-pooling is also widely used.

**Recurrent Neural Networks (RNN)** [22] [23]**:** They are neural network models that uses outputs of the previous layers as the input of the next layers, recurrently. Although RNN models are generally used in natural language processing, there are many other application areas. RNNs have different types according to the number of inputs and

outputs. For example, one-to-many (one input-multiple output) source code generation, many-to-one (multiple input-one output) sentiment classification, many-to-many (multiple input-multiple output) language translation. RNNs are difficult to train due to vanishing and exploding gradient problems. The difficulties arising from these problems have been reduced with the Long Short Term Memory (LSTM) [24].

**Autoencoders (AE)** [25]**:** Autoencoders represents compressed knowledge of the original input data and used for representation learning tasks. They learn the hidden representation of the input data in an unsupervised manner. The architecture of autoencoders consists of hidden layers that are symmetrically structured between input and output layers.

**Generative Adversarial Networks (GAN)** [26]**:** These are models in which two different networks, one generative and the other discriminative, work together. The generative network in the model generates data, the discriminative network consumes this generated data. Training process of this models as follows, while the purpose of the generative network is to fool the discriminative network, the purpose of the discriminative network is to ensure if the given input real or fake. The problem encountered in this model is that the tuning process for both networks is handled separately. Because of that the performance of one of the two models is lower than the other, affects the overall performance of the system.

**Graph Neural Networks (GNN)** [11]**:** Graph structured data are different from data in matrix or vector structures. When each data kept in matrix or vector structure considered as a cell, changing the order of these cells breaks the integrity of the data. In contrast, graph structured data is isomorphic. The nodes defined on the graph and the links between these nodes contain very valuable data. GNNs are neural networks specialized for learning over graph structed data.

## 2.3. Neural Network Hyper Parameters

**Epoch:** Giving the entire training set to the neural network model once is called one epoch. The number of epochs expresses how many times the training set will pass through to the neural network model.

**Batch Size:** It represents the amount of data that will be used in each iteration in neural network trainings. If batch size is equal to all dataset, it is called batch-mode, if it

is greater than one and less than the whole dataset, it is called mini-batch, and if it is one, it is called stochastic-mode.

**Dropout:** It is the random removal of the connections of neurons in MLP layers at a given rate. The main purpose of the parameter is to prevent the overfitting (model learns the training data but cannot makes predictions on unseen data).

**Number of Hidden Units:** It expresses, how many neurons will be defined in a hidden layer.

**Learning Rate:** On neural networks, parameter values are updated via backpropagation process. Parameter value modification of a perceptron is performed by finding the difference by taking derivatives and multiplying the difference with the given learning rate. This parameter is a tuning parameter of optimization algorithms.

## 2.4. Metrics to Evaluate Neural Networks

One of the most crucial part working with neural network is evaluation of the model performance. Most of the time accuracy metric is used to measure the performance of the model, but it is not enough just using one metric. Some of the metrics used to evaluate neural network models can be listed as follows.

**Confusion Matrix:** It is a table that shows the total number of inputs, actual outputs and predicted inputs to visualize complete performance. An example is given in Table 2.1. True positive (45), true negative (45), false positive (55) and false negative (55) values can be seen in the given table below. this metric visualizes prediction and expected value made by model.

Table 2.1. A Sample Confusion Matrix

| # Input: 100 | Predicted # Class A | Predicted # Class B |
|---|---|---|
| Actual # Class A | 30 | 45 |
| Actual # Class B | 10 | 15 |

**Classification Accuracy:** It is the ratio of the number of inputs given for estimation to the neural network and the number of outputs correctly predicted by the model. In order to provide reliable results, an equal or close to the number of inputs for

each class should be given. In this way, how accurately the trained model can predict is measured. Since GNNs have not been used to predict a software model before, random baseline is used for comparison.

$$Accuracy = \frac{\text{Number Of Correct Predictions}}{\text{Total Number Of Input}}$$

**F1 Score:** When the distribution of classes in dataset is not balanced, classification accuracy is not a good performance measurement choice. To overcome this issue there are two sub metrics used in F1 score, one is recall which is the ratio of the sum of true positive and false negative to true positive, and the other one is precision which is the ratio of the sum of true positive and false positive to true positive. F1 Score tries to find a balance between precision and recall and combines them into a single metric.

$$Recall = \frac{\text{True Possitive}}{\text{True Possitive} + \text{False Negative}}$$

$$Precision = \frac{\text{True Possitive}}{\text{True Possitive} + \text{False Possitive}}$$

$$F1\ Score = 2 \text{ x } \frac{\text{Pression x Recall}}{\text{Pression} + \text{Recall}}$$

**Area Under Curve (AUC):** To understand AUC, it is necessary to explain the ROC Curve first. Neural networks assign a certain probability to each class in their outputs, while making class selections, either the class with the highest probability is selected or the classes above a predetermined threshold value are specified as output. As the threshold value changes (increases or decreases) the outputs of the model to be used as predicted classes will naturally change. ROC shows the ratio between true positive rate

and false positive rate for different threshold values as shown in Figure 2.1. Since GNNs have not been used to predict a software model before, random baseline is used for comparison.

$$True\ Positive\ Rate = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}}$$

$$False\ Positive\ Rate = \frac{\text{False Positive}}{\text{False Positive} + \text{True Negative}}$$

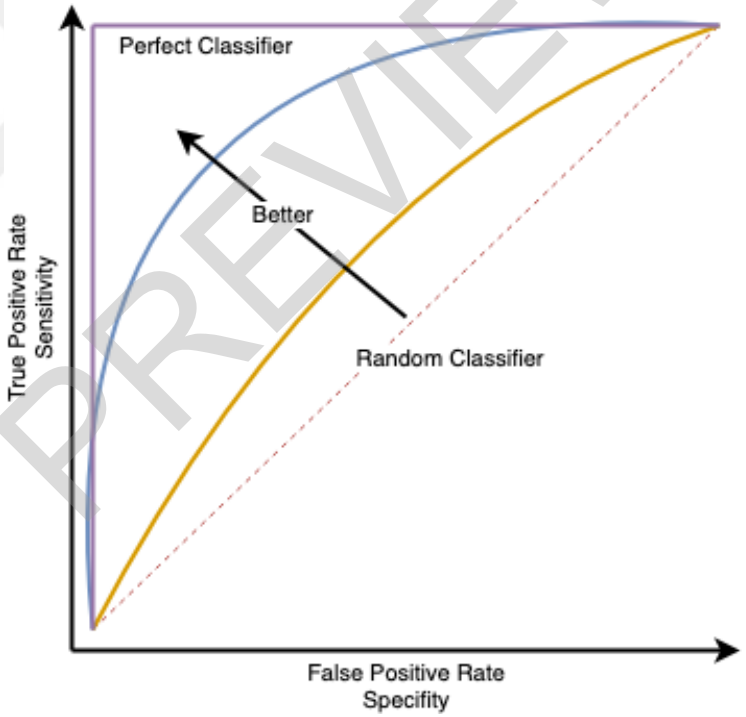


Figure 2.1. ROC Curve

AUC is a metric of performance of a binary classifier on any threshold values, so the metric does not change by threshold. AUC specifies the area under ROC Curve shown in Figure 2.2. The value of AUC is between 0 and 1, and higher values are better.

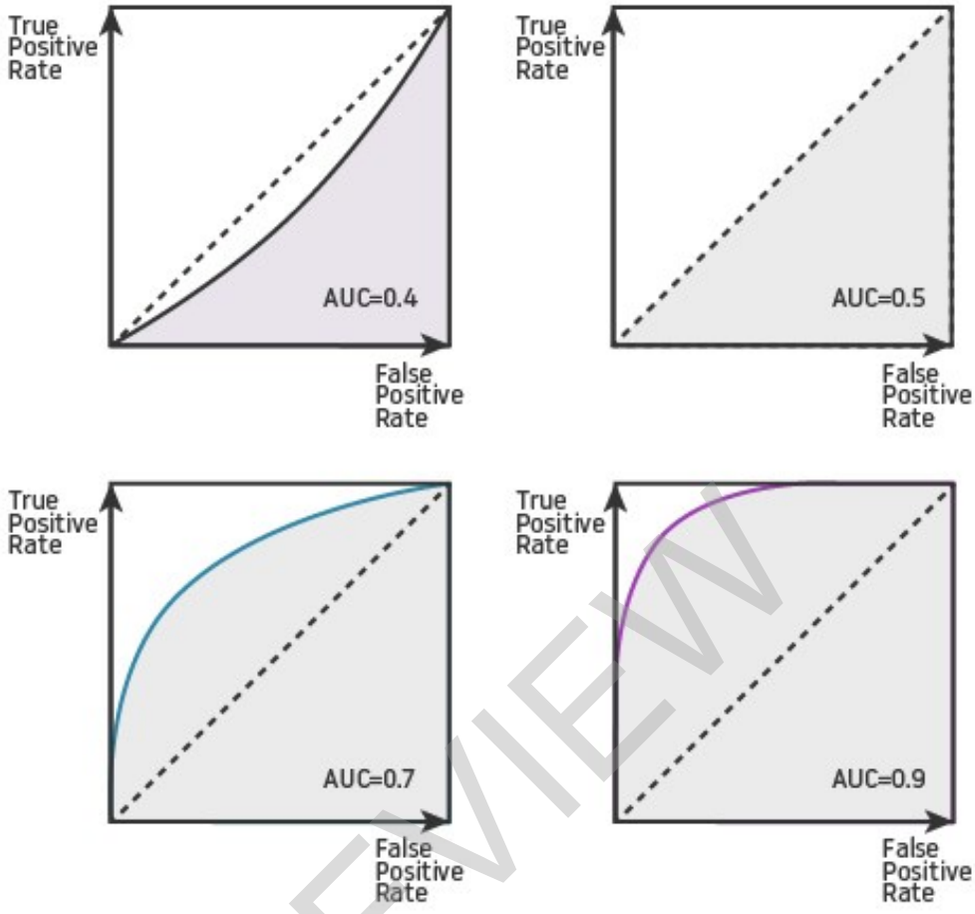


Figure 2.2. Area Under Curve (AUC)



**Cross Entropy Loss (Log Loss):** It is the measurement of the distance between the values (which are between 0 and 1) produced by the output layer of a neural network, from the expected values. For example, suppose that a neural network model has two units in the output layer, the expected output for this network is 1 for the first unit and 0 for the second unit. If the predicted output is 0.01 for the first unit, which is said to be a high loss value. For the best results, the loss should converge to zero.

**Mean Absolute and Squared Error:** It is the average distance between predictions and actual outputs only difference between absolute and squared one is, mean squared error takes the square of the distance. This metrics are used to measure accuracy for continuous variables. However, it does not talk about whether the predictions are overestimating or underestimating.