Measuring the robustness of network controllability

Ashish Dhiman

TU Delft

NAS

Measuring the robustness of network controllability

by

Ashish Kumar Dhiman

in partial fulfilment of the requirement for the degree of

Master of Science
in Electrical Engineering
Track Telecommunications and Sensing Systems

at the Delft University of Technology,
to be defended publicly on Tuesday August 04, 2020 at 10:00 AM.

Student number:	4949366	
Project duration:	November 12, 2019 – August 04, 2020	
Thesis committee:	Prof. dr. ir. Robert Kooij,	TU Delft, supervisor
	Dr. Maksim Kitsak,	TU Delft
	Dr. Johan Dubbeldam,	TU Delft
	Peng Sun,	TU Delft, daily supervisor

An electronic version of this thesis is available at
http://repository.tudelft.nl/.

Preface

With this thesis, “Measuring the robustness of network controllability”, I complete my Masters of Science degree in Electrical Engineering in the track of Telecommunications and Sensing Systems at the Delft University of Technology (TU Delft). This work was realized at the Network Architecture and Services (NAS) group, within the faculty of Electrical Engineering, Mathematics, and Computer Science (EEMCS) at TU Delft.

First of all, I would like to express my sincere gratitude to my supervisor Robert Kooij for giving me the opportunity to carry out this master thesis at the NAS group. I am thankful to you for your invaluable guidance and support throughout the entire duration of this project. Moreover, your constructive feedback and help with the presentation and report writing guided me in significantly improving my work. I would like to thank Peng Sun, my daily supervisor for his kind help whenever I needed and for the valuable brainstorming sessions. Furthermore, I would like to thank both my supervisors for all the weekly discussions and later on our Skype meetings, which were indispensable for the successful completion of this project, even in this difficult COVID-19 situation. I would like to thank Dr. Maksim Kitsak and Dr. Johan Dubbeldam for being a part of my thesis committee. Additionally, I would also like to thank all the members of the NAS group for the exciting and enjoyable discussions over Friday afternoon sessions.

Last but not the least, I am grateful to my parents and friends for always believing in me, giving me confidence and encouraging me during this exciting time.

Ashish Dhiman
Delft, August 2020

Abstract

Networks are all around us, telecommunication networks, road transportation networks, and the Internet are a few examples of networks that we encounter every day. The entities in a network are represented by nodes and the interconnections between them are represented by links. For example, in a telecommunication network, a node could be an end point for data transmission, a redistribution point or in physical terms, an entity that is capable of receiving, transmitting or redistributing information and a link could be a wired or a wireless connection between the nodes. It is of prime importance that the networks perform their functions properly. To ensure the effective operation of such networks, we need to control them by applying external inputs on some nodes which are known as driver nodes. We say that a network is controllable if it can be driven from any arbitrary state to any desired state in finite time under the control of driver nodes which are attached to the external inputs. Networks are often confronted with perturbations in the form of targeted and random attacks to disrupt their operation. Such perturbations make these networks less controllable. Thus, more driver nodes are needed to gain the full controllability of these networks. As a result, the robustness of network controllability decreases. In this study, the minimum number of driver nodes which gain full controllability after failures or attacks is chosen as the robustness metric.

Existing closed-form analytical approximations estimate the normalized minimum number of driver nodes as a function of the fraction of removed links in both targeted and random attacks. However, the approximations sometimes do not fit with the simulations and the errors between the approximations and simulations are large. The main objectives of this study are to improve the analytical approximations using machine learning methods for both targeted and random attacks and additionally, derive a suitable analytical approximation for the out-in degree-based attack. Specifically, we use Linear Regression, Random Forest, and Artificial Neural Networks. To evaluate the performance of our machine learning models, we compare them with analytical approximations and simulations. In addition to this, we also study the attack based variability in estimating the minimum number of driver nodes using robustness envelopes.

Based on the performance evaluations, we found that the approximations using machine learning models significantly outperform the existing closed-form analytical approximations for the minimum number of driver nodes, both in real-world and synthetic networks. Furthermore, we also assess the performance of our analytical approximations for the out-in degree-based attacks by comparing them with simulations.

Contents

List of Figures

List of Tables

1

Introduction

We are surrounded by a plethora of networks around us such as the Internet, the air-transportation network, and telecommunication networks. We depend very highly on them and entrust ourselves to their proper functioning. Naturally, for the desired operation and proper maintenance of such networks, it is important to control them. So, the ability to control systems plays a vital role in our daily lives. We represent the entities of a network by nodes and the interconnections between these nodes are represented by links. For example, in an air-transportation network, the nodes represent the cities and the links represent the flight routes that connect these cities. Many real-world networks consist of thousands of nodes and links and network science offers a way to analyze such networks by modeling them as complex networks. We control the networks by applying some inputs on certain nodes which are known as driver nodes [30]. The detailed explanation of driver nodes and controllability of networks will be discussed in Section 2.3.

By controlling the driver nodes, we can control the entire network and steer it to a desired state. For example, we can control the amount of traffic that passes through a node. Furthermore, a good question to ask here would be, how many nodes need to be controlled so that we can control the entire system? We will address this question in Section 2.3 in detail and also present relevant previous work, as it forms the basis of this research.

Networks are often confronted with perturbations which could be random perturbations or targeted attacks. Random perturbations occur as a result of the aging of network components, natural disasters, or failures. Targeted attacks are launched on the networks by people with malicious intent to disrupt the networks with a purpose to maximize the damage. In such scenarios, the number of driver nodes required to control the network changes as we will see in Section 2.4.

1.1. Challenges

There has been a lot of research to study the effects of various attacks on the network controllability but most of the works are based on the simulations which consume a lot of time and recently, Sun *et al.* [30] proposed closed-form approximations for the minimum number of driver nodes. Although, we now have the analytical approximations, the errors between the approximations and simulations are large. This leads us to the objective of this thesis, to improve these analytical approximations using various machine learning algorithms. We will assess the performance of the new machine learning based approximations by comparing them with simulations and the original analytical approximations on various real-world and synthetic networks.

1.2. Objectives

Based on the challenges mentioned in the previous section, the objectives of our study are as following:

1. Improve the analytical approximations by means of machine learning models for both random and targeted attacks and study the robustness of network controllability using envelopes.
2. Derive suitable analytical approximations for out-in degree-based attacks and evaluate their performance against the simulations and also compare the analytical approximations with machine learning based approximations.

1.3. Contribution

In this thesis, we use various machine learning algorithms to develop models for estimating the minimum number of driver nodes under various attacks. Specifically, the main contributions of this thesis are:

1. Development of multi-linear regression, Random Forest, and Artificial Neural Network models to quantify the minimum number of driver nodes under random and targeted attacks.
2. Evaluation of our machine learning models by comparing them with the analytical approximations and simulations based on relative errors. We evaluate our models on 40 real-world networks and two types of synthetic networks (Erdős-Rényi and Barabási-Albert networks).
3. Derivation of analytical approximations for out-in degree-based attacks and performance evaluation by comparisons with the simulations.
4. Measurement of the robustness of network controllability in case of targeted and random attacks using envelopes.

1.4. Approach

A step by step approach to realize the objectives mentioned in Section 1.2 is described below:

1. Carry out a literature study related to network controllability to understand the current developments, shortcomings of existing approaches, and identify the scope of possible improvements.
2. Study various machine learning algorithms such as multi-linear regression, Random Forest, and Artificial Neural Networks.
3. Obtain real-world network data to be used as training and testing and process it to a suitable form before it can be used in the algorithm.
4. Generate data for synthetic networks (Erdős-Rényi and Barabási-Albert networks) to be used in training, validation, and testing for both targeted and random attacks.
5. Develop the multi-linear regression model and evaluate its performance by comparing it with the analytical approximations and simulations.
6. Develop the Random Forest model and compare its results with the previous step.
7. Develop the Artificial Neural Network models for both targeted and random attacks and compare the performance with the previous step.
8. Derive analytical approximations for the out-in degree-based attack and similar to step 7, compare its performance with simulations. Furthermore, also compare the results with machine learning models.
9. Study the robustness of network controllability using envelopes.
10. Present the conclusions, limitations, and scope for future work.

1.5. Thesis Outline

This thesis is divided into six chapters:

- Chapter 2 presents an introduction to network controllability and the previous research work related to it along with its analysis. Furthermore, it also discusses network robustness and various types of attacks that affect network controllability.
- Chapter 3 describes the necessary background on various machine learning algorithms such as multi-linear regression, Random Forest, and Artificial Neural Networks. Training and testing data generation for synthetic networks and the tools required to develop the machine learning models are also discussed.

- Chapter 4 illustrates the steps used in selecting and developing suitable machine learning models.
- Chapter 5 presents a performance assessment by comparing the machine learning approximations with analytical approximation and simulations. Analytical approximations for the out-in degree-based attacks are also derived here. Furthermore, the robustness of network controllability using envelopes is also studied.
- Chapter 6 presents the conclusions, limitations, and scope for future work on measuring the robustness of network controllability.

2

Background and Related work

In this chapter, we explain in detail, the important concepts, necessary background and relevant previous work related to the robustness of network controllability.

Networks are all around us, the human brain, transportation networks, and telecommunication networks [32] are a few such examples. To better understand the features of various networks, graph modeling is used to represent the networks and such networks are known as complex networks. Network science deals with the study of complex networks using graph theory. A network consists of *nodes* that represent the individual entities and the interconnections between the nodes are represented by the *links*. This section introduces graph theory along with various network models and the metrics to understand them.

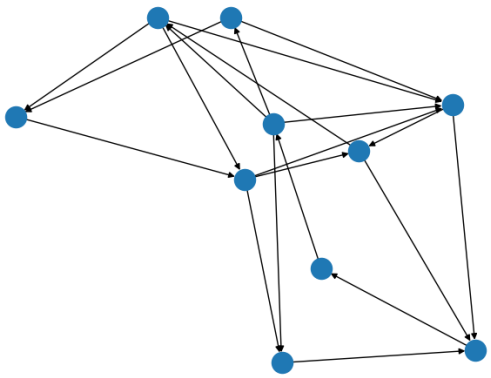

Figure 2.1: A directed Erdős-Rényi network with 10 nodes and 20 links.

2.1. An introduction to Graph Theory

A graph is a representation of a network that describes its structure and interconnections. It consists of a set of N nodes which are connected by a set of L links as shown in Figure 2.1. The interconnections between different nodes are represented by an $N \times N$ adjacency matrix [11]. For undirected graphs, if there is a link between node i and j, then the element $a_{ij} = 1$, otherwise $a_{ij} = 0$. For directed graphs, the element a_{ij} represents a directed link that starts from node i and ends at node j and it also has a weight associated with it. For example, if $a_{ij} = 0$, then there is no directed link from node i to node j and if $a_{ij} = 1.2$, then it represents a directed link from node i to node j with 1.2 as its associated weight.

2.1.1. Graph Metrics

Graph metrics quantify the properties of a graph. For example, the centrality metrics of graphs indicate how important or central a node or a link is. There is an extensive list of different graph metrics and it is important to identify the metrics that are important for the considered application. In this work, we have identified various metrics that are relevant to studying and analyzing the network controllability.

- **Average Degree**: The degree of a node measures the number of direct neighbors and the average degree of a graph measures how much degree on average a node has. The basic law of degree is given as

$$\sum_{j=1}^{N} d_j = 2L, \tag{2.1}$$

  where, d_j is the degree of the j^{th} node and L is the number of links. The average degree is then calculated as

$$E[D] = \frac{1}{N} \sum_{j=1}^{N} d_j = \frac{2L}{N}. \tag{2.2}$$

  For directed networks, the sum of degrees of all nodes is simply L and hence the average degree,

$$E[D] = \frac{1}{N} \sum_{j=1}^{N} d_j = \frac{L}{N}. \tag{2.3}$$

- **Clustering Coefficient**: When we want to know how our surrounding is connected around a node, we talk about clustering coefficient. For a node v, the clustering coefficient $c_G(v)$ is given as

$$c_G(v) = \frac{2y}{d_v(d_v - 1)}, \tag{2.4}$$

where, y denotes the number of links between the neighbors. The clustering coefficient of a graph is then,

$$c_G = \frac{1}{N}\sum_{v=1}^{N} c_G(v). \tag{2.5}$$

- **Eccentricity** The eccentricity of i^{th} node in a network denoted by ϵ_i is defined as the hopcount of the longest shortest path between the i^{th} node and any other j^{th} node,

$$\max_{j \epsilon \mathcal{N}}(H(\mathcal{P}_{i \to j})), \tag{2.6}$$

where, $\mathcal{P}_{i \to j}$ is the shortest path between A and B.

- **Diameter** The diameter denoted by ρ is the longest shortest path in a network. It gives us an idea about how much a graph is extended. Diameter is the maximum eccentricity over all the nodes,

$$\rho = \max_{i \epsilon \mathcal{N}}(\epsilon_i). \tag{2.7}$$

2.1.2. Complex Graph models

Apart from the simple graph topologies such as a complete graph, star graph, and a ring, there is another class of graphs known as random graphs [34]. These graphs are constructed based on the probabilistic rules and have many realizations. Each realization belongs to a particular class of random graphs. In many real-world networks, the topology keeps on changing with time and each such network can be represented by a particular realization of the random graph. Random graphs also provide means to analyze the network performance as it is important to model the networks before analyzing them. In this way, random complex graph models play a significant role in our understanding of complex networks. In this work, we will test our models on various real-world and synthetic networks and before we proceed, it is important to understand the basics of synthetic networks. The following sections describe the random complex graph models that are used in this thesis.

1. **Erdős-Rényi (ER) Random Graph**: Random graphs were introduced by Erdős and Rényi [9] during the 1950s. The Erdős-Rényi random graph denoted by $G_p(N)$ or $G(N, L)$ consists of N nodes and p represents the probability of a link in the former class and N nodes and L links in the latter. As the random graphs are constructed based on the probability of a link, we have different realizations. So, the class $G_p(N)$ consists of all such realizations with N nodes where each node pair is independently connected with probability p. The Erdős-Rényi random graph has a binomial degree distribution and $L = \frac{N(N-1)}{2}p$, gives the average number of links. In a directed Erdős-Rényi network $G(N, p)$, p specifies the probability of an outbound link from every node to the other nodes.

2. **Barabási-Albert (BA) Scale-Free graph**: The Barabási-Albert Scale-Free model [1] $G(N, M)$ is based on the procedure of preferential attachment in which the new nodes are more likely to attach to higher degree nodes in the original network. N denotes the number of nodes in the network and M is the number of links with which a new node adds itself to the current network. Many real-world networks such as the Internet and social networks show scale-free characteristics [3]. A directed Barabási-Albert is generated with N nodes and M indicates the number of outbound links from each newly added node to the existing network. Initially, the network has a complete digraph of M_O nodes such that M_O equals M and the new nodes are added to the network one at a time. With a probability that is proportional to the number of links that the existing nodes already have, each new node is added to the existing M_O nodes.

2.2. Network Robustness

The robustness of a network is its ability to cope with errors and failures. There are numerous examples of failures that occur in various real-world networks such as power failure in electrical networks, failures due to natural disasters, and it is important to deal with such failures by making networks robust. Naturally, a question arises, what is a robust network? A house that can withstand heavy winds, rain, and provide suitable living conditions can be considered as a robust house because it serves its purpose. Following this example, it is obvious to understand that a generalized definition of robustness is not possible because it depends on the type of network considered and its purpose and as long as the purpose is satisfied, it can be considered as robust. In this thesis, we assess the robustness of networks in terms of its controllability i.e. by quantifying the increase in the minimum number of driver nodes (explained in Section 2.3.2) under various perturbations discussed in the following section.

2.2.1. Types of Attacks

We consider various types of attacks in this work that lead to link removals. Removal of nodes is not considered.

1. Random Attacks: These are the unintentional failures, for example, failures due to manufacturing defects, natural disasters, and exhausted mechanical parts [6]. Because these failures occur randomly, hence, they are classified as random attacks.

2. Targeted Attacks: These types of attacks are aimed at maximizing the damage to disrupt the normal operation of a network and are carried out by people with malicious intent [5][7][14][16]. In targeted attacks, an attacker knows the network topology, functions, and vulnerabilities. The attacker tries to exploit the vulnerabilities to maximize the damage. In this thesis, we have considered two types of

targeted attacks, critical link attack, discussed in Section 2.5.2 and out-in degree-based attack, discussed in Section 5.3.1.

2.2.2. Network Robustness under perturbations

A perturbation is the removal or addition of link(s) or node(s) in a network [35]. In this work, as explained in Section 2.2.1, we only consider link removals. The robustness of networks under challenges or perturbations has been extensively studied. The following section describes a few notable works related to it.

- Socievole *et al.* [28] studied the robustness of networks in epidemics specifically Susceptible-Infected-Susceptible(SIS) spreads. They considered N-Intertwined Mean-Field Approximation(NIMFA) epidemic threshold as the robustness metric and found out that spectral radius, which is inversely related to epidemic threshold, is sufficient to study the robustness of the networks that belong to the same class. They also considered viral conductance as it provides information related to the behavior of the network and showed that viral conductance and spectral radius are highly correlated.

- Trajanovski *et al.* [33] considered robustness of networks in case of node removals for both targeted and random attacks. Not only they depicted the average case performance but the worst and the best case realizations as well using robustness envelopes based on probability density functions. For evaluating the robustness, two metrics were used, the size of the giant component and efficiency. Furthermore, it is mentioned that increasing the assortativity by a moderate amount provides more robustness in case of targeted attacks and decreasing it provides more robustness in case of random attacks.

- Wang *et al.* [36] investigated the effective graph resistance as a robustness metric in both synthetic and real-world networks. It is shown that link addition or removal significantly affects network robustness. They also derived the upper and lower bounds for the effective graph resistance under perturbations that consider link addition in such a way that it decreases the effective graph resistance maximally and link removal where the effective graph resistance decreases.

Koc *et al.* [20] studied the robustness of electrical power grids during the blackouts that result from cascading failures. However, in this thesis, we consider only link removals and study the robustness in case of targeted and random attacks and we use network controllability as the robustness metric. Cascading failures are not considered in this work. The following sections explain in detail the concept of network controllability and mention recent works related to it.

2.3. Network Controllability

The concept of network controllability has been extensively studied in the last ten years and has become an exciting research area. Network controllability is the ability to drive a system from an initial state to any other state in a limited amount of time by applying certain inputs on some nodes [17]. Controlling the amount of traffic that passes through a node, controlling a vehicle are some of its examples. The following sections describe in detail the concept of network controllability, starting with its classification as state controllability and structural controllability.

2.3.1. State Controllability

In the 1960s, Kalman introduced the concept of state controllability, also known as complete controllability [17]. Although, most of the real-world systems are dictated by nonlinearities, studying the linearized counterpart of such systems offers a way to understand the behavior of such systems [22]. A linear time-invariant (LTI) systems can be described by

$$\frac{dx(t)}{dt} = Ax(t) + Bu(t), \tag{2.8}$$

where $x(t) = [x_1(t), x_2(t), ..., x_N(t)]^T$ represents the state vector of the system at time t and the state $x_i(t)$ could be the number of packets passing through the i^{th} node in a telecommunication network or i^{th} person's view in a social network. A is the $N \times N$ adjacency matrix that represents the network's interconnections and B is the $N \times M$ $(M \leq N)$ input matrix that identifies the nodes that are directly controlled. $u(t) = [u_1(t), u_2(t), ..., u_M(t)]^T$ is the input vector consisting of input signals.

Kalman's Controllability Condition

The system described in Eq.(2.8) is said to be controllable if the controllability matrix

$$C = (B, AB, A^2B, ..., A^{N-1}B) \tag{2.9}$$

has full rank i.e. $Rank(C) = N$. The state controllability using Kalman's rank condition has a few drawbacks.

- The rank condition is not computationally efficient for real-world networks which consist of thousands of nodes, for smaller networks with small N values, it is easier to compute.
- The condition requires the exact values of parameters (the link weights) of matrices A and B but in reality, the parameter values are often unknown and we only have the information about whether there is a link or not. For example, in case of a gene regulatory network, it is difficult to estimate the link weights.
- The rank condition also ignores a suitable method to appropriately find B.

Due to the above-mentioned drawbacks, the concept of structural controllability was introduced.

2.3.2. Structural Controllability

In 1974, Lin introduced the concept of structural controllability [21]. It takes into account the network's underlying structure to calculate the state controllability. The matrices A and B are parameterized here but the system's structure is preserved. An LTI system described in Eq.(2.8) is said to be structured if the values in A and B are either fixed zeroes or independent non-zero parameters that can be varied. Such matrices are known as structured matrices in which the underlying structure of the network is maintained by the fixed zeros that cannot be changed but the link weights can vary.

An LTI system is said to be structurally controllable if we can set some value to the non-zero parameters in A and B such that it is controllable in Kalman's controllability condition i.e. $Rank(C) = N$. The rank of the controllability matrix gives an idea about the dimension of a system's controllable subspace. We have to appropriately choose B consisting of driver nodes to ensure the full rank condition of the controllability matrix C. The importance of structural controllability lies in the fact that we can determine the controllability of a network even when the exact link weights of a few or all the links are not known. It offers a framework to study the controllability of real-world networks which often lack complete knowledge about the weights or exhibit various uncertainties.

Method to achieve Structural Controllability

To achieve the structural controllability of directed networks, Liu *et al.* [22] developed the minimum input theory that determines the minimum number of nodes known as driver nodes, on which the external input needs to be applied. The minimum input theory is based on the concept of maximum matching, in which the unmatched nodes need to be driven by external inputs to exhibit full control. Cowan *et al.* [8] pointed out that the results of Liu *et al.* [22] are based on the assumption that there are no self-links, which means that the state of a node can only be changed by interacting with the other nodes. According to Liu *et al.* [22], the minimum number of driver nodes required to control the network depends on the degree distribution of the network. Furthermore, the driver nodes have a tendency to abstain from the nodes with high degrees. Along with this, they also showed that it is difficult to control the sparse and heterogeneous networks in contrast to the dense and homogeneous networks, that are easier to control in the sense that they require only a few driver nodes. Before we proceed further, it is important to understand the concept of maximum matching which is explained below.

In Figure 2.2, there are four nodes and four links. To determine the minimum number of driver nodes required to fully control this network, we follow the steps discussed below.

- First, we determine the matching links, the links which do not share common start or end nodes, denoted by green in Figure 2.2.

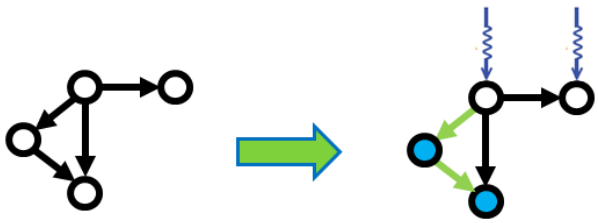

Figure 2.2: Determining matching links, matched nodes and driver nodes. The green links are the matching links that do not share common start or end nodes. The blue nodes are the matched nodes where the matching links end. The remaining white nodes are the driver nodes to which external inputs are applied.

- We determine the matched nodes i.e. the nodes where the matching links end, depicted by the blue nodes. The remaining nodes are known as unmatched nodes.
- The unmatched nodes are the driver nodes to which we will apply external inputs, depicted by the blue arrows, and in this way, we can control this network. Matched nodes do not need to be applied with external control input as they are already controlled by their superiors.
- We denote the number of driver nodes by N_D, here $N_D = 2$.

A maximum-sized matching is known as maximum matching. There could be multiple maximum matchings in a network as shown in Figure 2.3. Although the maximum matchings are not unique, the minimum number of driver nodes remains the same.

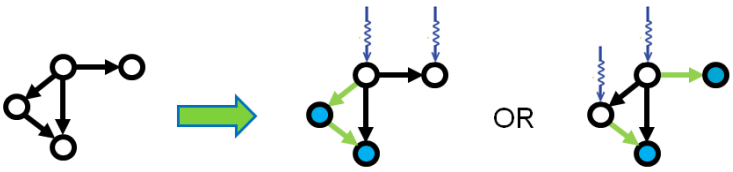

Figure 2.3: A representation of multiple maximum matchings in the same network.

To determine the maximum matchings in a directed network G with N nodes and L links, for calculating the minimum number of driver nodes N_D, a bipartite graph $B_{N,N}$ is constructed with $2N$ nodes and L links as shown in Figure 2.4. To generate a bipartite graph, each node in the original network is expressed as a source node V_+ and a target node V_- and the links between the source and target nodes are the directed links in the original network. The Hopcroft-Karp algorithm [15] provides an efficient method of calculating the maximum matching of a bipartite representation of G with a computation efficiency $O(\sqrt{N}L)$.

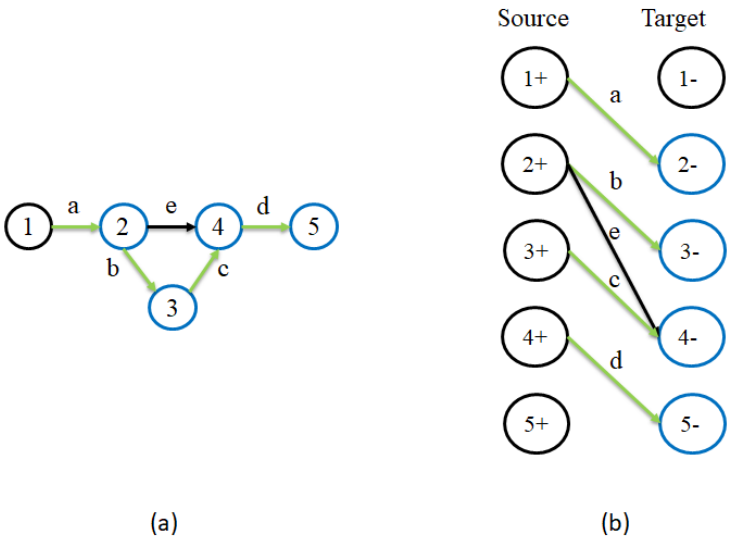

Figure 2.4: A bipartite representation of a directed graph (a) shown in (b) [30]. The green links a, b, c and d are the matching links that do not share source and target nodes. The target nodes of these matching links i.e. nodes 2, 3, 4 and 5 represented in blue, are the matched nodes and the remaining node 1 in black, is the unmatched or driver node.

2.4. Robustness of network controllability under perturbations

In this section, we discuss the relevant previous work related to the robustness of network controllability under perturbations. Figure 2.5 shows the effect of removal of a link on the number of driver nodes. Comparing Figure 2.5 with Figure 2.2 shows that N_D increases to three when a link is removed from the graph depicted in Figure 2.2, which initially had two driver nodes. In other words, when a network is attacked and links are removed, it becomes less robust as the number of driver nodes increases or we can say that there is a decrease in network controllability.

- For a given attack strategy, Nie *et al.* [25] compared the robustness of Erdős-Rényi and Barabási-Albert networks and found that a modest power-law exponent Barabási-Albert network is more robust than a modest average degree Erdős-Rényi network.

- Pu *et al.* [26] investigated the controllability of directed synthetic networks, Erdős-Rényi and Barabási-Albert networks under single node attacks and cascading failures. It is shown that the degree based attacks affect the network controllability in a more efficient way than random attacks. Furthermore, the greater the number of links a network has, the greater the robustness against cascading failures.

Sun *et al.* [30] studied the robustness of network controllability under two types of attacks, considering only link removals for both random attacks and targeted attacks. They

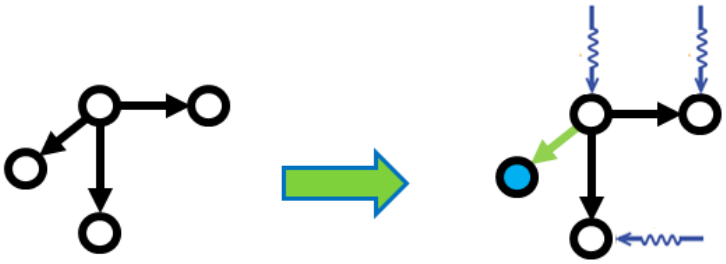

Figure 2.5: Change in the number of driver nodes upon link removal.

quantified the increase in the minimum number of driver nodes with respect to failures and derived analytical approximations, discussed in detail in Section 2.5. Although the results are promising, still there is room for improvement, as we will in see in the next section. Next we discuss the approximations of Sun *et al.* [30] and point out their drawbacks and room for improvement. The main aim of this thesis is to improve these approximations using machine learning. Our improved models will be compared with both the approximations suggested by Sun *et al.* [30] and simulations, in Section 5.1 and Section 5.2.

2.5. Analytical Approximations by Sun *et al.*

All the previous works, mentioned in Section 2.4 are mostly based on simulations. We need analytical approximations because simulations are often slow and consume a lot of time. In this section, we discuss the analytical approximations derived in [30]. Before we delve deeper into the analytical approximations, it is worth to understand the concept of critical links as it forms the basis of how the links are removed.

- Critical Link: A link is considered as a critical link, if its removal increases the number of driver nodes needed to fully control the network.
- Redundant Link: A link is said to be redundant, if it does not belong to a maximum matching. In other words, if we can remove a link and it does not change the current number of driver nodes, then it is a redundant link.
- Ordinary Link: If a link is neither critical nor redundant, it is an ordinary link.

2.5.1. Number of driver nodes under random attacks

Consider a network G with N nodes and L links. The initial number of driver nodes N_{DO} required to control the network without any attack can be calculated by applying the Hopcroft-Karp algorithm [15]. To determine the number of critical links L_C, we first apply the Hopcroft-Karp algorithm to get the initial number of drivers N_{DO}. After that, we

remove each link and apply the Hopcroft-Karp algorithm again. If the number of driver nodes increases, then it means that the removed link is a critical link. The algorithm is applied L times so that all the links are covered. Let the number of removed links be m, so the fraction of removed links l equals $\frac{m}{L}$ and the fraction of critical links l_C equals $\frac{L_C}{L}$.

Case 1: $l \leq l_C$

Consider the case in which $l \leq l_C$ so that we uniformly remove m links at random and $m \leq l_C$. Furthermore, i links out of m are critical and the remaining $m - i$ are not critical. For each removed critical link, the number of driver nodes increases by one [22]. Hence, for each i, the contribution in increase of the driver nodes N_D can be expressed as $i\binom{L_C}{i}\binom{L-L_C}{m-i}$ and its expectation is given by

$$N_D^* = \frac{\sum_{i=1}^{m} i\binom{L_C}{i}\binom{L-L_C}{m-i}}{\binom{L}{m}}. \tag{2.10}$$

The numerator term $\sum_{i=1}^{m} i\binom{L_C}{i}\binom{L-L_C}{m-i}$ can be further expressed as,

$$\sum_{i=1}^{m} i\binom{L_C}{i}\binom{L-L_C}{m-i} = \sum_{i=1}^{m} \frac{L_C!}{(i-1)!(L_C-i)!}\binom{L-L_C}{m-i}$$

$$= L_C \sum_{i=1}^{m} \binom{L_C-1}{i-1}\binom{L-L_C}{m-i}$$

$$= L_C \sum_{i=0}^{m-1} \binom{L_C-1}{i}\binom{L-L_C}{m-i-1}.$$

Applying the Vandermonde's formula $\sum_{j=0}^{k}\binom{a}{j}\binom{b}{k-j} = \binom{a+b}{k}$, $L_C \sum_{i=0}^{m-1}\binom{L_C-1}{i}\binom{L-L_C}{m-i-1} = L_C\binom{L-1}{m-1}$, the expectation of the increase in the number of driver nodes can be expressed as,

$$N_D^* = lL_C. \tag{2.11}$$

Hence, when the fraction of removed links is less than or equal to the fraction of critical links ($l \leq l_C$), the number of driver nodes is given by,

$$N_D = N_{DO} + lL_C. \tag{2.12}$$

Also,

$$n_{D,rand} = \frac{N_{DO} + lL_C}{N}, \tag{2.13}$$

where $n_{D,rand}$ represents the normalized value of the minimum number of driver nodes in case of random attacks.

Case 2: $l \geq l_C$

For the case in which $l \geq l_C$, the normalized minimum number of driver nodes is expressed by a parabolic approximation given by,

$$n_D = al^2 + bl + c, \tag{2.14}$$

where, a, b and c are derived from the following boundary conditions.

- At $l = l_C$, Eq.(2.14) has the same value and derivative as Eq.(2.13). Hence, we get $N_{DO} + l_C L_C = N(al_C^2 + bl_C + c)$ and $L_C = N(2al_C + b)$.
- When all the links are removed, we need to control all the nodes i.e. we need to apply inputs to all the nodes. Hence, at $l = 1$, $n_D = 1$ which gives us the third condition, $a + b + c = 1$.

Using the above boundary conditions, we get the values of the parameters a, b and c where,

$$a = \frac{N - N_{DO} - L_C}{N(l_C - 1)^2},$$

$$b = \frac{L_C}{N} - 2al_C,$$

$$c = 1 - \frac{L_C}{N} + a(2l_C - 1).$$

Finally, for random attacks, the normalized minimum number of driver nodes can be expressed as,

$$n_{D,rand} = \begin{cases} \frac{N_{DO} + lL_C}{N}, & l \leq l_C \\ al^2 + bl + c, & l \geq l_C \end{cases} \tag{2.15}$$

Sun *et al.* [30] evaluated the performance of their approximation for Erdős-Rényi networks, Barabási-Albert networks and eight real-world networks. Figure 2.6 shows a comparison between the approximation and simulation for an Erdős-Rényi network.

Approximation by Liu *et al.*

There is an another approximation for ER networks under random attacks by Liu *et al.* [22] based on the average out-degree of the network. If k denotes the average out-degree of an ER network then,

$$k = p(N - 1). \tag{2.16}$$

According to Liu *et al.* [22], if l denotes the fraction of randomly removed links from the ER network, then the normalized minimum number of driver nodes is given by,

$$n_D = w_1 - w_2 + k(1 - l)w_1(1 - w2), \tag{2.17}$$

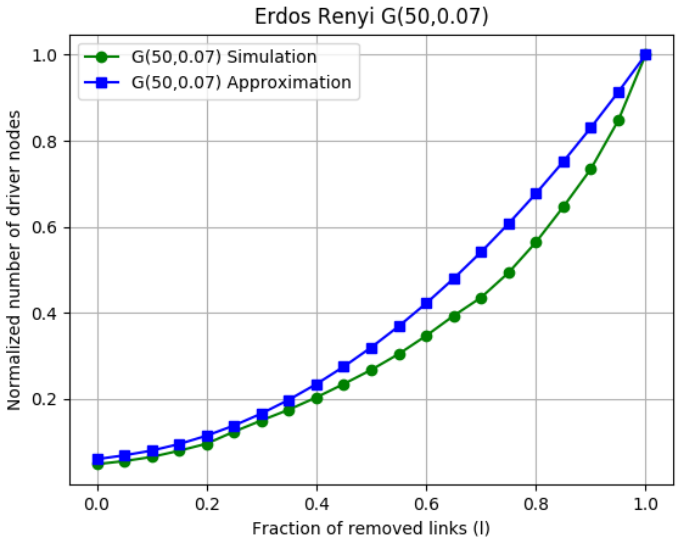

Figure 2.6: Performance comparison of the approximation Eq.(2.15) for the normalized minimum number of driver nodes n_D as a function of the fraction of removed links l in ER network under random attack.

where, solving the following implicit equation gives us the value of w_1,

$$w_1 = e^{-k(1-l)e^{-k(1-l)w_1}}, \tag{2.18}$$

and w_2 is given by,

$$w_2 = 1 - e^{-k(1-l)w_1}. \tag{2.19}$$

In Section 5.2, we will compare the approximations given by Eq.(2.15) and Eq.(2.17) with machine learning based approximation to find out which one of these approximations performs the best.

2.5.2. Number of driver nodes under targeted attacks

In case of targeted attacks, the links are first identified as critical and non-critical and it is assumed that the attacker has this information. So, at first, the critical links are removed one by one at random and once all the critical links are removed, the remaining links are removed at random. Similar to this, Mengiste *et al.* [23] also suggested targeted attacks based on the concept of removal of critical links, but only considering the simulations.

Case 1: $l \leq l_C$

For the case where the fraction of removed links is less than or equal to the fraction of critical links i.e. $l \leq l_C$, with the removal of each critical link at random, the number of driver nodes increases by one as also explained in Section 2.5.1. Hence, with each

subsequent removal, the number of driver nodes increases linearly with the fraction of removed links. Hence, the normalized minimum number of driver nodes for $l \leq l_C$ can be expressed as,

$$n_{D,crit} = \frac{N_{DO} + lL}{N}. \tag{2.20}$$

Case 2: $l \geq l_C$

Similar to the previous case of random attacks, here, the normalized minimum number of driver nodes is approximated by a parabolic equation of the form $n_D = dl^2 + el + f$, when the fractions of removed links are greater than or equal to the fraction of critical links. The values of the parameters are again calculated using the boundary conditions listed below.

- When the fraction of the removed links equals one, then all the nodes need to be controlled, hence n_D=1 or we can say that the parabola passes through the point $(1, 1)$.
- At $l = l_C$, we have the value of n_D as $\frac{N_{DO}+l_C L}{N}$.
- At l_C, the derivative of the parabola is zero.

Using these three boundary conditions, the values of the parameters d, e and f are calculated where,

$$d = \frac{N - N_{DO} - l_C L}{N(l_C - 1)^2},$$
$$e = -2dl_C,$$
$$f = 1 + d(2l_C - 1).$$

Finally, for targeted attacks, the normalized minimum number of driver nodes can be expressed as,

$$n_{D,crit} = \begin{cases} \frac{N_{DO}+lL}{N}, & l \leq l_C \\ dl^2 + el + f, & l \geq l_C \end{cases} \tag{2.21}$$

Figures 2.6, 2.7 show the performance of the analytical approximations compared to the simulations for Erdős-Rényi networks. It can be seen that the approximations perform well, in particular for small fractions of removed links, but still deviate significantly for larger fractions of removed links. In other words, for random attacks, the approximation fits well when the fraction of removed links l is smaller than or equal to the fraction of critical links l_C. For targeted critical link attack, the approximation is accurate when the fraction of removed links is sufficiently small. A better approximation would be as close as possible to the simulations and we will show in Chapter 5 that machine learning models can help to construct better models. It is also worth noting that in case of targeted attacks, the approximation always overestimates the value of driver nodes and hence, can be considered as a worst-case approximation.

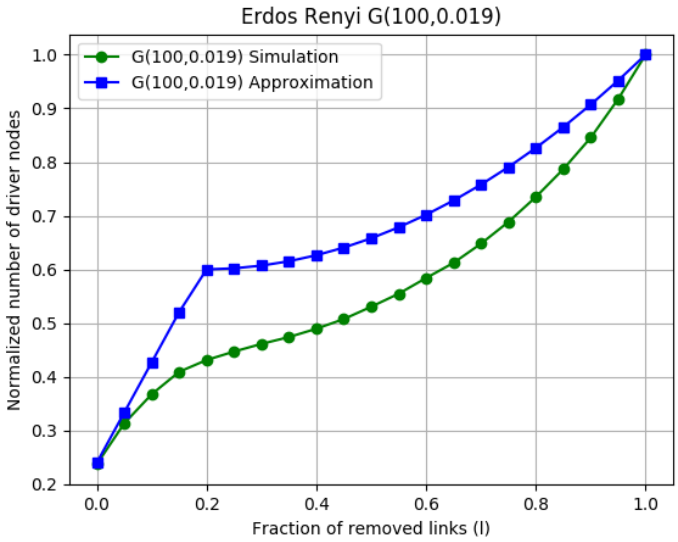

Figure 2.7: Performance comparison of the approximation Eq.(2.21) for the normalized minimum number of driver nodes n_D as a function of the fraction of removed links l in ER network under targeted attack.

We use machine learning techniques to improve the approximations and before we delve into the models, it is important to understand the theory and algorithms behind a few machine learning techniques, which are discussed in the next chapter.

3

Introduction to Machine Learning

In this chapter, we discuss in detail various machine learning algorithms based on which we build our models. The steps involved in building the models will be discussed in Chapter 4.

3.1. Introduction

Machine learning, in simple terms, is a technique to predict the outcome of a certain event with the help of data to learn from it by finding the hidden patterns. There is an extensive list of use cases of machine learning these days ranging from e-Commerce to predict the customer's buying habits, meteorology for weather-forecasts to predict the pressure and temperature variations in our atmosphere and Virtual Personal Assistants such as Siri, Alexa, and Google Now. Machine learning involves collecting data that is used to train the algorithm. The data could be already available from some sensors, data companies, or it can be generated by proper simulations. The quality of data also plays an important role in learning algorithms to develop better models, the lesser the number of missing values, the better the data and in turn, the better the model.

In broader terms, machine learning can be divided into supervised learning, unsupervised learning, and reinforcement learning. In supervised learning, we have labeled data with one or more features (inputs) and also the desired outputs. A mathematical model is then constructed based on this data. The model is trained until a desired level of accuracy or performance is achieved. In the case of unsupervised learning, the data is unlabelled and we also do not have desired outputs labels. Reinforcement learning is another class of machine learning algorithms in which the algorithm performs a certain task by interacting with a dynamic environment in which feedback is provided that symbolizes a reward that needs to be maximized. Furthermore, supervised machine learning is used for solving two types of problems, Classification, and Regression.

- Classification: A classification algorithm maps the features to some classes or it categorizes the input features into discrete classes. For example, predicting a

disease as infectious or non-infectious is a classification problem.

- Regression: In the case of regression problems, the algorithm maps one or more features to continuous-valued outputs. For example, rent prediction of the houses based on several features such as location, age, and size. Another such example, which is the goal of this thesis, is to predict the normalized minimum number of driver nodes given various network metrics as features in case of both synthetic and real-world networks under targeted and random attacks.

In this thesis, we consider three different supervised learning algorithms- Linear-Regression, Random Forest, and Artificial Neural Networks. The following section describes in detail the theory behind these algorithms. The implementation of these algorithms will be discussed in Chapter 4.

3.2. Linear Regression

Linear Regression (LR) is a supervised machine learning method in which a linear relationship is established between an input feature and output. A simple linear regression model can be described by,

$$y = b_0 + b_1 x_1, \tag{3.1}$$

where y is dependent on the independent variable x_1. The slope of the linear line is b_1 and b_0 is the intercept. Figure 3.1 shows an example of the input and output relationship between the actual data and the linear regression line.

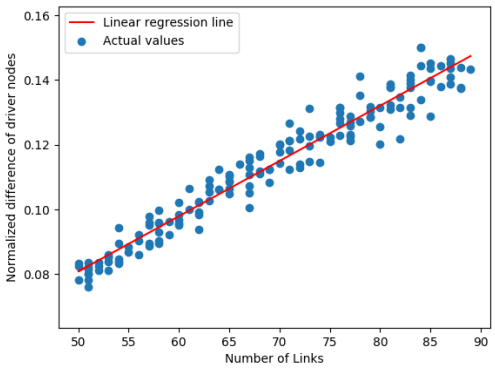

Figure 3.1: An example of a linear regression model.

So far, we have only considered a single feature, but the algorithm can also be extended to more than one feature, which is known as multi-linear regression. In this thesis, we consider multi-linear regression for our predictions as we use various input features to build our model. Similar to simple-linear regression, a multi-linear regression can be expressed as

$$y = b_0 + b_1 x_1 + b_2 x_2 + b_3 x_3 + ... + b_n x_n, \tag{3.2}$$

where $x_1, x_2, ..., x_n$ are the independent features with $b_1, b_2, ..., b_n$ as their respective coefficients. One of the disadvantages of linear regression is that, it does not perfrom well when the relationship between the features and the output is not linear. In that case, we use non-linear machine learning techniques and one of the widely used technique is Random Forest, which is discussed in the next section.

3.3. Random Forest

Random Forest (RF) is another supervised learning algorithm that is used for both classification and regression problems. It is based on the concept of ensemble learning in which we combine different algorithms or use the same algorithm many times. Random Forest specifically uses multiple decision trees to build a forest. A decision tree utilizes a tree structure to divide the dataset into various subsets and predicts the output as shown in Figure 3.2. For example, if the value of the feature is less than 2, then the output is 1.5. If it lies between 2 and 10, then the output is 0.5 and lastly, if the value of the feature is greater than 10, then the output is 2.5.

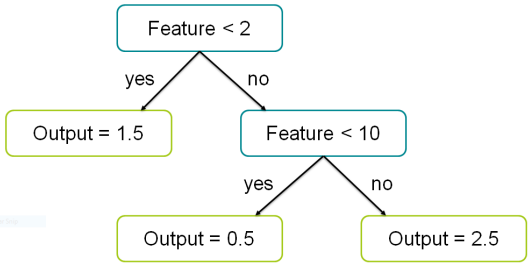

Figure 3.2: A representation of a decision tree to predict the output based on the feature values.

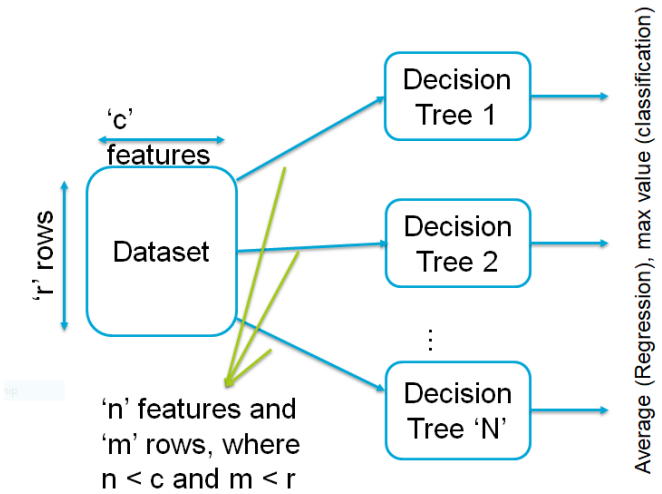

Figure 3.3: A random forest model consisting of multiple decision trees.

Random Forest is based on the bootstrap aggregation technique which is also known as bagging. In bagging, we divide the dataset and train individual decision trees based on the separate divisions of data. In other words, a random subset of features with replacement are selected and given to the individual decision trees for training. The accuracy of the random forest can be improved by tuning the number of trees as we will show in Chapter 4. Figure 3.3 shows how a random subset of features and rows is provided to the individual decision trees to predict the output. It is to be noted that for regression problems, the output is the average of the outputs of all the decision trees.

3.4. Artificial Neural Network

Artificial Neural Network (ANN), as the name suggests, is a supervised learning algorithm based on the biological neurons in our brain. ANN consists of many neurons that learn in a way similar to how the human brain learns. As in the case of any supervised learning algorithm, ANN is provided with a labelled dataset consisting of labelled inputs and outputs, from which it learns before it can predict the output. ANNs are used for both regression and classification problems, but in this thesis, we use ANN for regression problems only.

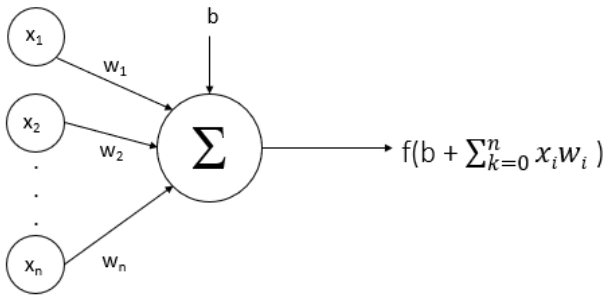

Figure 3.4: A single neuron consisting of weighted inputs to which an activation function is applied to predict the output.

A typical neuron is shown in Figure 3.4. It consists of numerical inputs and outputs and a weight is also associated with each connection. Sometimes a bias b is also added. ANN consists of many such neurons connected in many layers which are subdivided into an input layer, one or more hidden layers, and an output layer. The weighted sum of inputs are given to the next layer neurons and an activation function is applied to the weighted sum and an output is generated based on the activation function [10]. Sigmoid and Rectified Linear Unit (ReLU) are the two extensively used activation functions which are discussed next.

- Sigmoid activation function: It is also known as the logistic function in which the

output is restricted to a value between 0 and 1. It is expressed as,

$$f(x) = \frac{1}{1 + e^{-x}}, \tag{3.3}$$

where, x is an input value. Figure 3.5 shows the input-output characteristics of the Sigmoid activation function. One of the drawbacks of using the Sigmoid activation function is that it leads to the vanishing gradients problem. It means that when we calculate the gradients to backpropagate the errors and to update the weights, the gradients of the Sigmoid activation function diminish. To avoid this problem, we use the ReLU activation function.

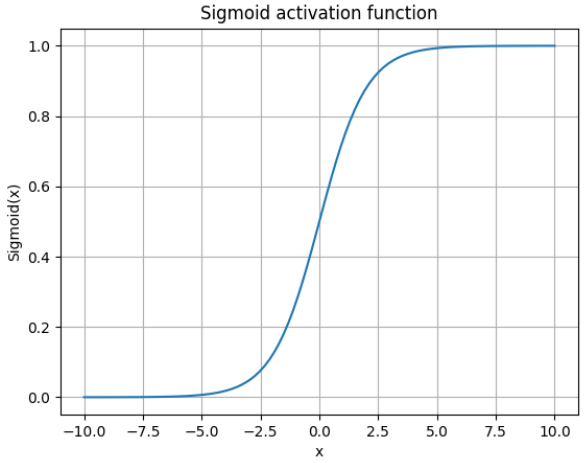

Figure 3.5: Sigmoid activation function.

- Rectified Linear Unit (ReLU) activation function: In ReLU activation function, the output value equals the input value when the input is greater than or equal to zero. For input values less than or equal to zero, the output is zero. Mathematically, it is expressed as,

$$f(x) = max(0, x). \tag{3.4}$$

Figure 3.6 shows the input-output characteristics of the ReLU activation function. One of the advantages of using the ReLU activation function is that it does not have the issue of vanishing gradients. For backpropagation of the errors, the derivative of the activation function is calculated. Its derivative is one for the input values greater than zero and it is zero for the input values less than or equal to zero. Another advantage of using the ReLU activation function is that it offers fast training.

We use a feed-forward ANN specifically the Multi-layer Perceptron (MLP) architecture as shown in Figure 3.7. In the feed-forward type of ANN, there are no cycles, the

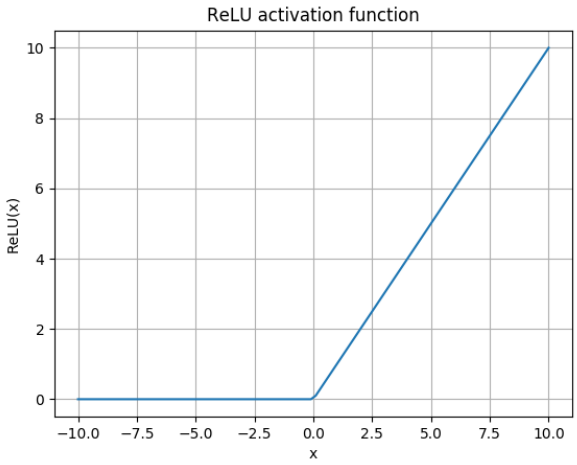

Figure 3.6: ReLU activation function.

connections are established only from the input side to the output side. A simple example of a feed-forward ANN is a Single-layer Perceptron in which the inputs are directly connected to the outputs. A more advanced and extensively used complex architecture is a Multi-layer Perceptron. In this work, we use MLP to develop our ANN models.

MLP consists of at least one hidden layer along with the input and output layers. Each neuron in the subsequent layer is connected to every other neuron of the previous layer. The number of layers can be varied depending on the requirement. For training the ANN, it is provided with a training dataset that consists of one or more input features and the desired outputs. The output of the ANN is then compared against the known outputs and based on that the errors are calculated, which are propagated backward to adjust the weights. The process continues until we have the errors under some desired range.

Hyper-parameter tuning

The performance of a neural network can be influenced by varying its hyper-parameters until a required performance level is reached. The hyper-parameters include the number of hidden layers, the number of neurons in each layer, the learning rate, etc. The larger the number of neurons and layers in a network, the larger time it takes to train the network. The selection of hyper-parameters will be discussed in Chapter 4.

Remove Overfitting

Initially, when we train a neural network, there are chances that our model would overfit and hence does not perform well. This can be visualized by the difference of errors in validation and test dataset. If the difference in errors is significantly large then it

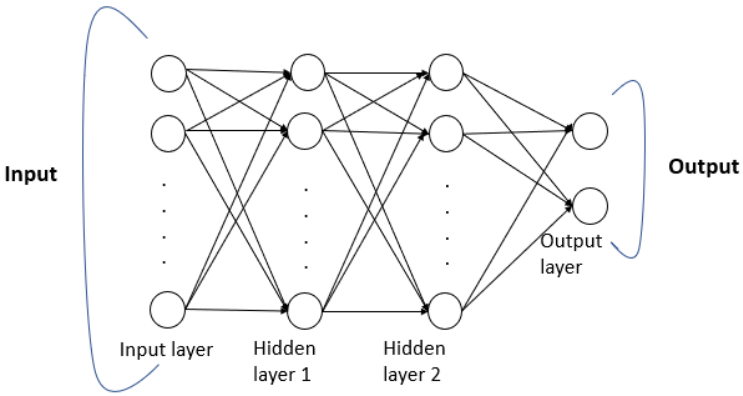

Figure 3.7: A Multi-layer Perceptron architecture.

means the model has memorized the validation dataset and it performs well only on that, whereas it fails to perform well on the test dataset. There are various ways to reduce overfitting.

L1 or L2 regularization and dropout are the two extensively used regularization techniques to reduce the overfitting [13][29]. Reducing overfitting is important because it helps in making the ANN model more generalized so that it performs better on the test data.

Early Stopping

Another extensively used technique to improve the ANN model is early stopping. If early stopping is not used, then the model assumes the weights of the last epoch and those might not be the best weights that have a minimum value of validation errors. So, we try to stop our training earlier than finishing all the epochs. This is done based on a patience value and generally, a patience value of 50 is used which means that if the value of validation errors does not decrease further for consecutive 50 epochs, then the training stops and then we can restore the best weights.

3.5. Tools and Datasets

Before building machine learning models for calculating the number of driver nodes under various types of attacks, we discuss the tools that we use, the datasets that we consider for real-world networks, and the data that we generate for synthetic networks.

3.5.1. Tools used

This section describes the important python libraries that are used to develop our machine learning models.

Tensor-flow, developed by Google, is a widely used open-source machine learning library for developing ANNs [31]. Another open-source library, Keras, helps in fast and easier prototyping of neural networks [18]. It provides a user-friendly application programming interface (API) for developing ANN models. We have used these software to develop our ANN models that will be discussed in Chapter 4. Along with these, we also use scikit-learn to develop Linear Regression and Random Forest models [27]. Additionally, for generating complex networks and studying their properties, we use python's NetworkX library [24].

3.5.2. Dataset

To develop machine learning models, we divide a dataset into various sets: a training set, a validation set, and a testing set. The training set, as the name suggests, is used to train the model. A validation set is used to fine-tune the model and to keep track of the improvements until a desired amount of accuracy is achieved. Finally, the model is tested on the test dataset.

For the real-world networks, we use a publicly accessible dataset available at The Internet Topology Zoo [19]. It consists of a collection of publicly available data from various network operators. It is not a fixed dataset in the sense that it is an ongoing project and often the network information is updated as the networks grow. Also, new networks are added. The data is interpreted from the maps that network providers make accessible. However, in some networks, because of the insufficient knowledge, the interpretations are not accurate. As a result, there are some ambiguities in the dataset. The dataset is available in two formats, Graph Markup Language (GML) [12] format and GraphML [4] format. In this thesis, we use the dataset that is available in the GraphML format as it is simpler to parse using NetworkX python library. We filter the data such that there are no disconnected networks. Furthermore, we also do not consider multigraphs i.e. the graphs in which there are multiple edges between a pair of nodes. Finally, we have 232 real-world networks. Out of these 232 networks, we use 192 networks for training and we test our models on the remaining 40 networks. It is to be noted that these networks are not directed. However, GraphML format has two attributes, edge source and target. We use this information to generate a list of links that consists of source and target nodes. For example, (a, b) represents a directed link from source node a to target node b.

Furthermore, the average degrees of these networks are small. The number of links is of the order of the number of nodes. Arpanet196912 network is the smallest network in this dataset with $N = 4$ nodes and $L = 4$ links. Cogentco network has the largest number of nodes and links with $N = 197$ and $L = 243$. Moreover, there are some networks in which there are no critical links. Based on the above analyses, we

conclude that the dataset consists of networks that vary a lot. The machine learning models will have difficulties in learning from such a varying dataset because for training the machine learning models, the data should be consistent in some sense. Table 3.1 shows the properties of some of the test real-world networks. The information related to the remaining networks is available at the NAS website [2].

For synthetic networks, we generate data through simulations. We generate ER networks with different N and p values. For each value of N and p, we generate 100 corresponding networks and calculate the average number of links, the average number of critical links and other graph metrics such as clustering coefficient and average degree. We follow the same process for BA networks as well for different N and M values. For targeted critical link attacks, as we already mentioned, first the critical links are removed at random and then the remaining links also at random. As the links are removed randomly, we use 10,000 realizations of attacks to get the average values of the normalized minimum number of driver nodes. Similarly, for the random attacks, we use 10,000 realizations as all the links are removed uniformly at random. In the next chapter, we will discuss the methods that we use to develop machine learning models using these datasets.

Table 3.1: Properties of 10 real-world networks used for testing our models.

Network	**N**	**L**	$\mathbf{L_C}$	$\mathbf{N_{DO}}$
Colt	153	177	38	81
Surfnet	50	68	23	15
EliBackbone	20	30	12	5
Garr200912	54	68	9	30
GtsPoland	33	37	12	14
Ibm	18	24	6	6
Arpanet19706	9	10	6	2
GtsHungary	30	31	8	18
BellCanada	48	64	17	16
Uninet	69	96	19	4

4

Measuring the robustness of network controllability using machine learning

In this chapter, we will develop multi-linear regression, Random Forest, and Artificial Neural Network models to study the robustness of network controllability. In each of the following sections, we first develop a particular model and then tune it to three different types of attacks by modifying the parameters used as each attack is different. In other words, a parameter or a feature that is important for a targeted attack might not be important for a random attack and hence, we have developed attack based models.

4.1. Multi-linear regression model

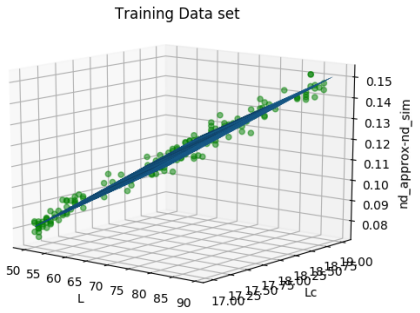

Figure 4.1: A training dataset and its linear regression model.

Based on the theory mentioned in Chapter 3, a multi-linear regression model is developed that takes various features such as the number of nodes, number of links, number of critical links, clustering coefficient, diameter as inputs and minimizes the least

square errors to build a linear model and predict the output. In our case, the output is the difference in the values of the normalized minimum number of driver nodes at $l = l_C$ between the approximation Eq.(2.21) and simulation. The reason for choosing the difference at $l = l_C$ and the resulting new approximation will be discussed in detail in the next chapter.

Figure 4.1 shows an example of the training data and a linear regression line for Erdős-Rényi networks using two features, L and L_C to predict the difference $n_{D\_approx} - n_{D\_sim}$ in the values of the normalized minimum number of driver nodes at $l = l_C$ between the approximation and simulation. As we already mentioned in Chapter 3, we use the number of nodes, links, critical links, clustering-coefficient, average degree and diameter as the input features. Additionally, we use k-fold cross validation technique to validate our LR model so that it does not overfit. In k-fold cross validation, we split the dataset in k subsets such that we train the model on k-1 subsets and leave one for the testing. We repeat this process k times and finally take the average. A parameter to consider here, is the value of k. In this work, we have selected $k = 10$. Figure 4.2 shows a comparison of predictions with the actual data. As we can see that the predictions vary significantly from the actual data, hence, we will shift on to more complex non-linear models such as Random Forest. A comprehensive analysis of the performance of the LR model for both synthetic and real-world networks under targeted and random attacks will be presented in Chapter 5.

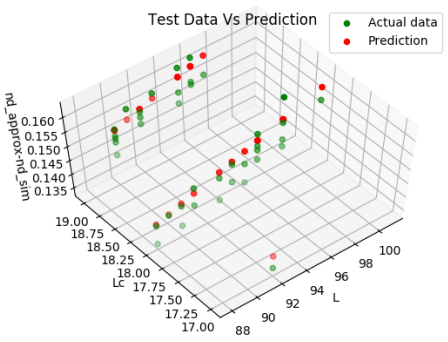

Figure 4.2: A comparison of predictions with the actual data using linear regression for ER networks under targeted attacks.

4.2. Random Forest model

In this section, we develop our Random Forest model. The important parameter to consider while developing a random forest model is the number of trees or number of estimators. To select the number of trees, we calculate the Root Mean Square Errors (RMSE). Table 4.1 shows the number of trees along with their respective errors in case of targeted attacks for 40 test real-world networks. As it is evident from the table, selecting

number of estimators as 50 gives the least value of error. Hence, we select the number of trees as 50.

Table 4.1: Selection of the number of trees based on RMSE in the case of 40 test real-world networks for targeted attacks.

Number of trees	Root Mean Squared Error
10	0.0644
20	0.0630
25	0.0629
50	0.0619
75	0.0623
100	0.0628
125	0.0632
150	0.0638

Based on the similar reasoning, Table 4.2 shows the selection of the number of trees for all three attacks and different networks. Furthermore, not all the features contribute equally to predicting the output, some of the features have more weight as compared to others and as a result contribute relatively more. In Random Forest, we achieve this using relative feature importance. Figure 4.3 shows the relative importance scores of various features. The more the score is, the more it contributes relatively to predicting the output.

Table 4.2: Selection of number of trees for different attacks and networks.

Networks	Number of trees		
	Real-world	Erdős-Rényi	Barabási-Albert
Targeted critical link attack	50	20	20
Random attack	50	20	20
Out-in degree-based attack	50	20	20

Figure 4.3 shows feature importance scores for real-world networks in case of targeted attacks. Similarly, the importance scores are calculated for Erdős-Rényi and Barabási-Albert networks in case of random and out-in degree-based attacks. In addition to this, similar to the LR model, the RF model is also validated using the k-fold cross validation to check for overfitting.

4.3. Artificial Neural Network Model

As discussed in Chapter 3, we use a feed-forward neural network with Multi-layer Perceptron architecture in which all the neurons in the previous layer are connected to all the neurons of the next layer. To develop our model, we start with a model with only one hidden layer with 128 neurons and evaluate it based on RMSE and subsequently,

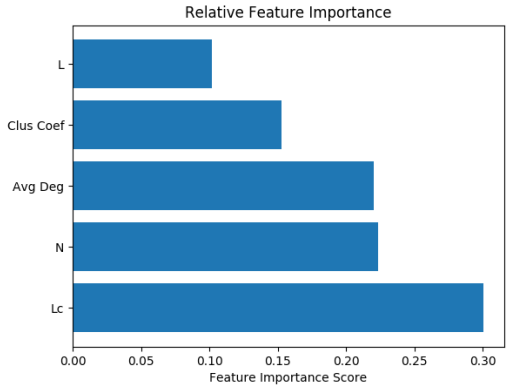

Figure 4.3: Relative feature importance scores for the real-world networks under targeted attacks.

we add more neurons to the same layer and then also add a few more layers. Figure 4.4 shows the variation of errors with ANN of different sizes and we select the model which has the least value of errors. Figure 4.4 is specifically for real-world networks in case of targeted attacks. The remaining cases are presented in Table 4.3.

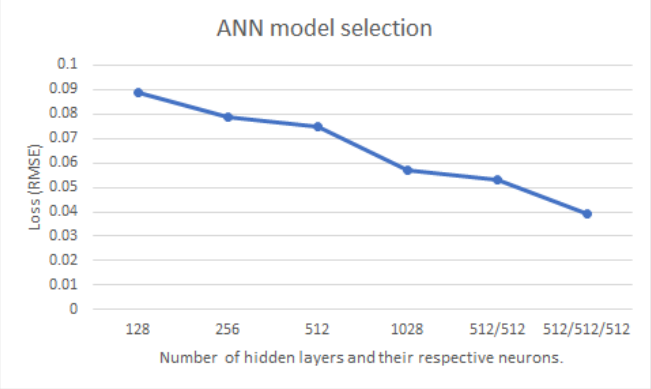

Figure 4.4: ANN model selection based on the RMSE values for the real-world networks.

After selecting the number of hidden layers and the neurons in each hidden layer, we start training the neural network. As pointed out in Section 3.4, the neural network might begin to overfit by memorizing the data and hence, would not perform well on the test data. So, we use a dropout method to reduce overfitting by intentionally skipping a few neurons so that the model becomes more generalized. We have chosen a dropout rate of 0.2. In other words, a dropout of 0.2 means that we randomly ignore 20 % of neurons for training.

Another important hyper-parameter to consider is early stopping. Figure 4.5 shows the value of validation errors with respect to the number of epochs. It is evident from the visual inspection that in this case, when the number of epochs reaches 220, we have

Table 4.3: Selection of ANN size for different networks under targeted, random and out-in degree-based attacks.

Networks	Number of hidden layers		
	Real-world	**Erdős-Rényi**	**Barabási-Albert**
Targeted critical link attack	512/512/512	128	512/512/512
Random attack	512/512/512	128/512/512/512	128/512/512/512
Out-in degree based attack	512/512/512	128	512/512/512

the least value of mean-squared errors (MSE)[1]. But the model continues to learn and updates weights till the last epoch, which is 250 in this case. A consequence of this is that the model would then assign the weights of the last epoch and those might not be the best weights. To eliminate this and restore the best weights, we use early stopping. We can stop the training process if it no longer leads to improvement in validation errors for a particular number of epochs. We do that by assigning a patience value. A patience value of 50 is suitable in this case. In other words, if the validation errors do not improve for 50 consecutive epochs then the training stops and we restore the weights of the last epoch with the least value of validation errors.

Batch size is another important hyper-parameter to consider while developing an ANN model. It refers to the number of training samples to consider before updating the weights. A commonly used value of batch size is 32 samples. Moreover, an epoch consists of one or more batches. For example, if a dataset consists of 320 training samples, then a batch size of 32 would give us 10 batches. After each batch, the weights are updated based on MSE. In this case, one epoch consists of 10 batches.

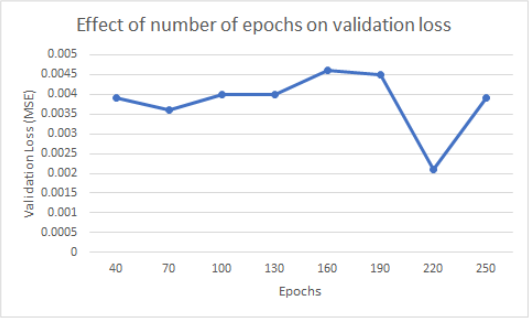

Figure 4.5: Variation of MSE with the number of epochs.

[1]Keras [18] has various loss functions for regression problems and one of the commonly used loss functions is mean-squared error (MSE). It is the error that is minimized during the training process. So, for the training process using ANN, we will consider MSE. RMSE is not available as a loss function in Keras.

To summarize, a complete list of hyper-parameters that we use to develop our ANN model is shown in Table 4.4.

Table 4.4: ANN hyper-parameters selection.

Hyper-parameters	**Selection**
Hidden Layers	Network and attack dependent
Activation Function	ReLU
Loss Function	MSE
Dropout rate	0.2
Early Stopping	Yes
Patience	50
Restore best weights	Yes
Epochs	300
Batch size	32

5

Results and Performance Comparisons

In this chapter, we present our results and assess the performance of our machine learning models by comparing them with analytical approximations (Section 2.5) and simulations in case of targeted critical link attacks, random attacks, and out-in degree-based attacks. We use 10,000 realizations of attacks for simulations. Here, we also derive the analytical approximation for out-in degree-based attacks. Furthermore, we test our machine learning models on three types of networks, Erdős-Rényi, Barabási-Albert, and various real-world networks. In the final section of this chapter, we also take into account the variability of attacks due to the random removal of links and study the robustness of network controllability with the help of envelopes.

5.1. Targeted Critical Link Attack

In this section, we assess the performance of machine learning models for three types of networks under targeted critical link attack starting with Erdős-Rényi networks.

5.1.1. Assessment on synthetic networks

We trained our model on various values of N, staring from a smaller network with $N = 50$ and gradually increasing it to a larger network with $N = 500$. Along with that, we also vary p values such that the generated ER networks are sparse in the sense that the number of links is around two, three, or four times the number of nodes, not the order of tens or hundreds of nodes. The reason for such a selection is that the data that we have for real-world networks consists of the links that are of the order of the number of nodes as we mentioned in Section 3.5.2 and we will compare the performance of our machine learning based approximations on both real-world and synthetic networks. So, only for the purpose of comparisons, we have considered such networks.

To build our machine learning based approximation for the targeted critical link attack, we will predict the difference in the normalized minimum number of driver nodes n_D between the analytical approximation Eq.(2.21) and simulation at l_C so that the new n_D moves closer to the simulation at l_C. The reason we choose to predict the difference at l_C is because of the fact that the original analytical approximation Eq.(2.21) fits well with the simulation when the fraction of removed links l is less than or equal to the fraction of critical links l_C. Additionally, the difference at $l = l_C$ is significant as we already saw in Section 2.5.2. The idea is to minimize this difference using machine learning so that the new machine learning based approximation moves closer to the simulation. We will subtract the predicted difference from the analytical approximation Eq.(2.21) to get a new value of n_D at l_C. Let this new value be n_{DX}. Now we build our machine learning based approximation in the following manner.

Similar to the analytical approximation in Section 2.5.2, we assume a linear relationship for the normalized minimum number of driver nodes n_D as a function of the fraction of removed links l for $l \leq l_C$. At $l = 0$, the normalized minimum number of driver nodes is n_{DO} and at $l = l_C$, the value is n_{DX}. Using these two points we get,

$$n_{D,crit,ML} = n_{DO} + \frac{n_{DX} - n_{DO}}{l_C} l, \tag{5.1}$$

where, $n_{D,crit,ML}$ represents the new machine learning based approximation for the normalized minimum number of driver nodes as a function of the fraction of removed links for $l \leq l_C$. Similar to Eq.(2.21), when the fraction of removed links is greater than or equal to the fraction of critical links i.e. for $l \geq l_C$, the normalized minimum number of driver nodes is approximated by a parabolic equation of the form,

$$n_{D,crit,ML} = d_{ML} l^2 + e_{ML} l + f_{ML}, \tag{5.2}$$

where d_{ML}, e_{ML} and f_{ML} are derived from the following boundary conditions.

- At $l = l_C$, $n_{D,crit,ML}$ equals n_{DX}.
- When the fraction of removed links equals one, then all the nodes need to be controlled. Hence, n_D equals one.
- At $l = l_C$, the derivative of the parabola is zero.

Using these three boundary conditions, we get the values of the parameters d_{ML}, e_{ML} and f_{ML} where,

$$d_{ML} = \frac{1 - n_{DX}}{l_C^2 - 2l_C + 1},$$
$$e_{ML} = -2 d_{ML} l_C,$$
$$f_{ML} = 1 + d_{ML}(2l_C - 1).$$

Finally, for the targeted critical link attack, the machine learning based approximation for the normalized minimum number of driver nodes is expressed as,

$$n_{D,crit\_ML} = \begin{cases} n_{DO} + \frac{n_{DX} - n_{DO}}{l_C} l, & l \leq l_C \\ d_{ML} l^2 + e_{ML} l + f_{ML}, & l \geq l_C \end{cases} \tag{5.3}$$

In Figure 5.1, we compare three machine learning algorithms with simulation and analytical approximation Eq.(2.21) for $N = 100, 200, 300$ and 400. It is evident from the visual inspection that our machine learning based approximations outperforms the analytical approximations Eq.(2.21) and fits well with the simulations. In the case of ER networks, all three ML algorithms perform well. ANN model slightly performs better than Random Forest and Linear Regression models. This can be explained based on the non-linear relationship between the input features and the output difference that we estimate. Hence, the LR model performs the least among the three machine learning algorithms, nevertheless, it still outperforms the analytical approximations Eq.(2.21). To quantify this improvement, we use mean relative errors[1] as depicted in Table 5.1. For example, in $ER(100, 0.019)$ the mean relative error decreases from 19 % to 2.1 % when we use ANN. We also notice that ANN is only slightly better than RF but takes more time to train. For example, for a dataset consisting of 100 rows, RF took around 2-3 seconds for training, while ANN took 3-4 seconds. So, depending on the dataset, ANN takes more time to train. However, testing time is not an issue for any of the models.

Table 5.1: Performance indicators for synthetic networks under targeted attacks.

Network	**Mean Absolute Error**		**Mean Relative Error**	
	Approximation	**ANN**	**Approximation**	**ANN**
ER(50, 0.04)	0.1005	0.0133	0.1990	0.0235
ER(100, 0.019)	0.1000	0.0124	0.1907	0.0213
ER(200, 0.0063)	0.0663	0.0115	0.1008	0.0175
ER(300, 0.004)	0.0596	0.0066	0.0879	0.0101
ER(400, 0.0026)	0.0472	0.0046	0.0659	0.0071
BA(50, 2)	0.0590	0.426	0.0821	0.0582
BA(100, 2)	0.051	0.0351	0.0704	0.0467

We now compare our machine learning based approximation with the simulation and the analytical approximation Eq.(2.21) in Barabási-Albert networks. Figure 5.2 shows that the machine learning models perform better than the analytical approximations as the ML curves fit better with the simulation. Although, the machine learning curves are closer to the simulation curve but still do not completely fit the simulation. When comparing Barabási-Albert networks with Erdős-Rényi networks, from Table 5.1, we see the relative errors in case of BA networks are larger than the ER networks. This can be explained based on the network properties, as in BA networks, there are a few nodes with a high degree and the new nodes tend to attach themselves to those nodes.

[1]We use Mean Relative Error (MRE) for evaluating the goodness of fit of the approximations. However, for building the models, we use MSE as the loss function because MRE is not available as a loss function in Keras.

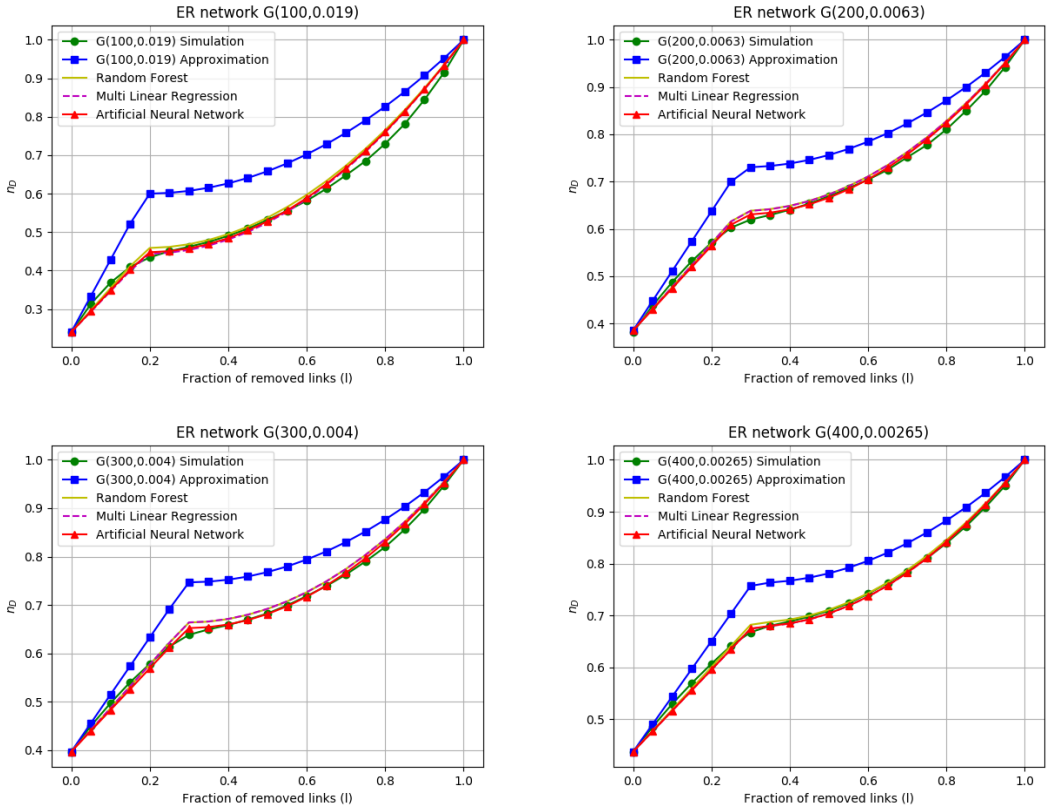

Figure 5.1: Performance comparison of the machine learning approximations for the normalized minimum number of driver nodes n_D as a function of the fraction of removed links l in Erdős-Rényi networks under targeted attacks. Simulations are based on 10,000 realizations of attacks.

So, after all the critical links are removed, the number of driver nodes do not change significantly. On the contrary, an ER network has a fixed probability p of an outbound link from every node to the other nodes and hence, the curve is relatively steeper after the fraction of removed links is larger than l_C.

5.1.2. Assessment on real-world networks

In this section, we compare the machine learning based approximation Eq.(5.3) with analytical approximation Eq.(2.21) and simulation for real-world networks. We train our model on 192 networks and test on 40 networks. As the number of real-world networks for training is less, we expect our model to not perform well on all 40 test networks as we will also see in the latter part of this section. Furthermore, the variability of networks is greater in real-world networks, there are some networks with $N = 5$ and others with $N = 197$. Moreover, it is not practical to show all the 40 plots here, so we will show a few plots and analyze the results.

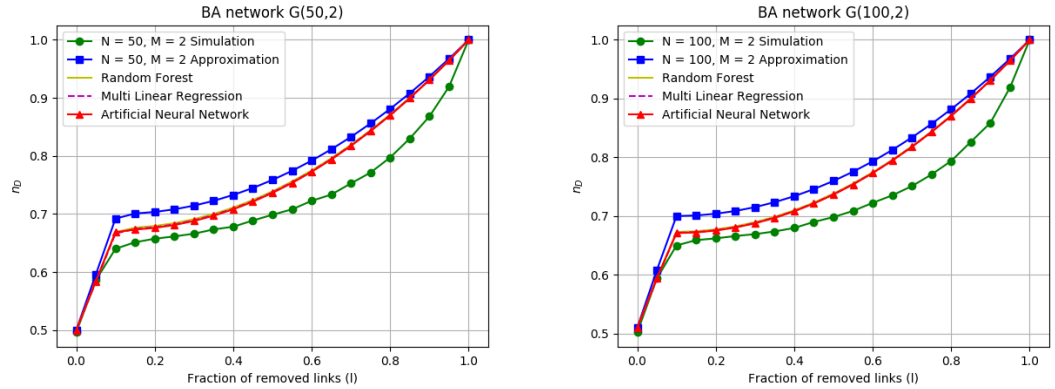

Figure 5.2: Performance comparison of the machine learning based approximations for the normalized minimum number of driver nodes n_D as a function of the fraction of removed links l in Barabási-Albert networks under targeted attacks. Simulations are based on 10,000 realizations of attacks.

Table 5.2: A comparison of different ML algorithms for 40 test real-world networks based on RMSE.

ML algorithm	**RMSE**
Multi-linear regression	0.0723
Random Forest	0.0550
Artificial Neural Network	0.0430

Figure 5.3 shows that our machine learning curves fit well with the simulation. We also compare LR, RF, and ANN based on Root Mean Square Error (RMSE)[2]. Table 5.2 compares all three ML algorithms. ANN has the least value of RMSE and hence, in this case, performs the best. Table 5.3 shows the performance of the ANN model for 10 real-world networks. We see that the model performs the best for the Colt network with a mean relative error of 1.46 %, while performing the least for the Ibm network with a mean relative error of 8.3 %. Additionally, in 9 out of 10 networks the mean relative error is less than 5 %. To analyze the results of all 40 networks, Figure 5.4 shows the number of real-world networks in which the ML based models perform better than analytical approximation Eq.(2.21) for all three ML algorithms.

Among the 40 test networks, we also show in Figure 5.4 that machine learning algorithms perform better than analytical approximation Eq.(2.21) in 38 networks. But this does not account for their individual performance, so we also show the performance distribution of these algorithms. Among these 38 cases, ANN performs the best in 19 cases, RF in 16 cases, and LR in only 3 cases. So, again ANN and RF perform almost similar. If we only consider the ANN case, Figure 5.5 shows that ANN performs better than analytical approximation Eq.(2.21) in 30 out of 40 networks. This tells us about the individual performance of the ANN model, this also includes the cases in which RF

[2]We use RMSE to compare the predictions using different machine learning models. We do not use MRE because here we are not evaluating the approximation curves with the simulations.

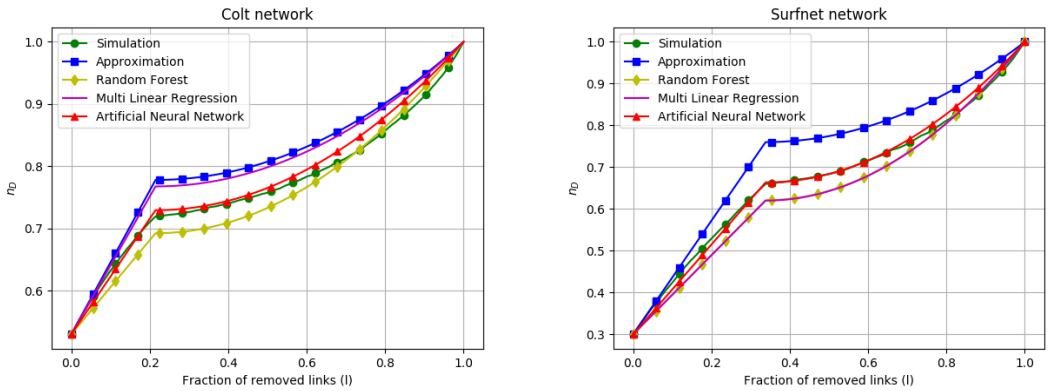

Figure 5.3: Performance comparison of the machine learning based approximations for the normalized minimum number of driver nodes n_D as a function of the fraction of removed links l in real-world networks under targeted attacks. Simulations are based on 10,000 realizations of attacks.

performs better than ANN but overall ANN has a slight edge over RF in terms of the number of better cases. So, considering only the ANN model, we can conclude that our ANN model has an accuracy of 75 % in the case of real-world networks under targeted attacks. Furthermore, for the 10 cases in which the machine learning based approximation performs worse than the analytical approximation, the mean relative errors are only a little bit better for the analytical approximation with the difference of less than 2 %. We also observe that the mean relative error never exceeds 10 % using machine learning based approximations while, for the analytical approximation, it exceeds 10 % in 3 cases. Additionally, for the networks in which the machine learning based approximation performs worse than the analytical approximation, the mean relative errors are less than 5 % in 8 out of 10 cases. The results for the 30 remaining real-world networks are also available at the NAS website [2].

On comparing the real-world networks to the synthetic networks, we also conclude that the ANN based approximation performs better on synthetic networks as the data that we consider is similar and its variability is less, on the contrary, we have limited training data for real-world networks and also, the networks vary a lot in the number of nodes, links and other parameters as also mentioned in [19].

5.2. Random Attack

This section presents the performance comparison of our machine learning models with the analytical approximation Eq.(2.15) and simulation for both synthetic and real-world networks under random attacks. Similar to the targeted critical link attack, we compare our ANN and RF models based on the RMSE values. In the case of real-world networks, the RMSE comes out to be 0.0192 for the RF model and 0.0165 for the ANN model. It is to be noted that the RMSE values of real-world networks are based on the 40 real-world networks. We see that ANN performs better than RF based on the RMSE. So, for the

Table 5.3: Performance indicators for real-world networks under targeted attacks.

Network	N	L	L_C	N_{DO}	Mean Absolute Error		Mean Relative Error	
					Approx-imation	ANN	Approx-imation	ANN
Colt	153	177	38	81	0.0393	0.0116	0.0512	0.0146
Surfnet	50	68	23	15	0.0597	0.0095	0.0866	0.0151
EliBackbone	20	30	12	5	0.1468	0.0201	0.2471	0.0376
Garr200912	54	68	9	30	0.0223	0.0202	0.0277	0.0251
GtsPoland	33	37	12	14	0.0266	0.0171	0.0335	0.0235
Ibm	18	24	6	6	0.0595	0.0519	0.0956	0.0832
Arpanet19706	9	10	6	2	0.0440	0.0255	0.0588	0.0434
GtsHungary	30	31	8	18	0.0269	0.0321	0.0311	0.0373
BellCanada	48	64	17	16	0.0502	0.0135	0.0757	0.0230
Uninet	69	96	19	4	0.1195	0.0309	0.184	0.0485

Figure 5.4: Performance comparison of different ML algorithms.

remainder of this section, we will use only the ANN model.

Before we proceed with the performance assessment, it is to be noted that for random attacks, we use machine learning to predict the normalized minimum number of driver nodes n_D for the entire range of the fraction of removed links l. Specifically, we predict n_D for 21 points starting from $l = 0$ to $l = 1$ in steps of 0.05. For example, for each value of N and p in ER networks, we generate 21 data points for the training using simulations. We follow the same process for BA networks for each N and M value. The reason we predict n_D in such a way is because there is not much difference between the approximation value of n_D and the simulation value at $l = l_C$ as we saw in Section 2.5.1. So, even if we predict the difference correctly, the new machine learning based curve will follow the analytical approximation curve.

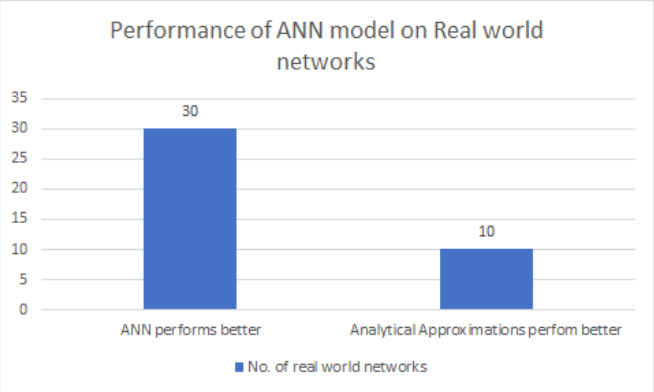

Figure 5.5: Performance of ANN model on real-world networks.

5.2.1. Assessment on synthetic networks

Figure 5.6 shows the normalized minimum number of driver nodes n_D as a function of fraction of removed links l based on simulation, analytical approximation Eq.(2.15) and ANN for synthetic networks. It can be seen that the ANN curve fits the simulation curve better than the analytical approximation Eq.(2.15) for both Erdős-Rényi and Barabási-Albert networks. It is also evident from Table 5.4 based on the relative and absolute errors that ANN outperforms the analytical approximation Eq.(2.15). For example, there is a significant improvement in the mean relative error from 30 % to 6 % in ER network with $N = 50$ and $p = 0.082$. Similarly, we also see an improvement from 4 % to 0.4 % in case of BA networks with $N = 100$ and $M = 2$.

Table 5.4: Performance indicators for synthetic networks under random attacks.

Network	**Mean Absolute Error**		**Mean Relative Error**	
	Approximation	**ANN**	**Approximation**	**ANN**
ER(50, 0.082)	0.0712	0.0105	0.3080	0.0675
ER(100, 0.016)	0.0085	0.0024	0.0137	0.0044
BA(50, 2)	0.035	0.0032	0.0517	0.0051
BA(100, 2)	0.032	0.0030	0.0455	0.0049

Furthermore, we now compare Sun's approximation Eq.(2.15) and Liu's approximation Eq.(2.17) with the ANN based approximation. Figure 5.7 shows the estimation of the normalized minimum number of driver nodes n_D as a function of the fraction of removed links l in $ER(200, 0.006)$ network under random attack using all three approximations. We notice that Liu's approximation Eq.(2.17) and the ANN approximation overlaps, but Liu's approximation Eq.(2.17) slightly performs better than ANN based on the mean relative errors. For Liu's approximation Eq.(2.17), the mean relative error comes out to be 0.18 % and for ANN, it is 0.59 %. Both of these approximations perform better than Sun's approximation Eq.(2.15), which has a mean relative error of 1.1 %. Additionally,

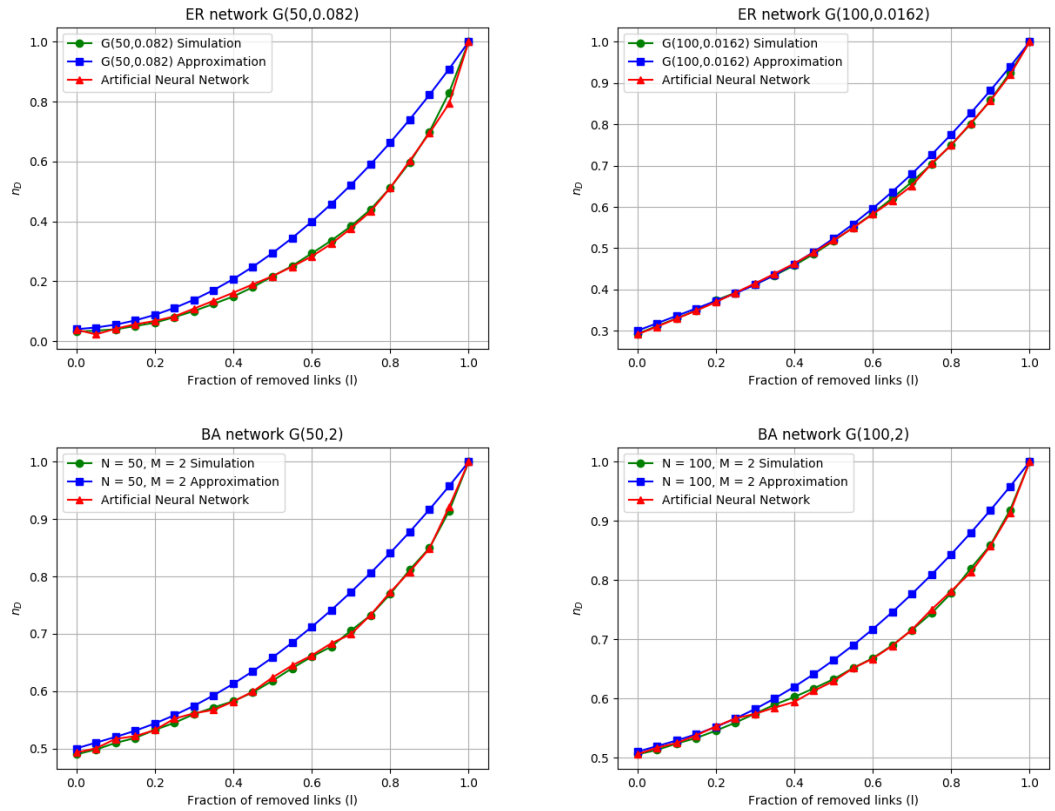

Figure 5.6: Performance comparison of the machine learning based approximations for the normalized minimum number of driver nodes n_D as a function of the fraction of removed links l in synthetic networks under random attacks. Simulations are based on 10,000 realizations of attacks.

we also do a performance analysis of all three approximations for a few more ER networks as shown in Table 5.5. We observe that although ANN fits well with the simulation, Liu's approximation Eq.(2.17) always fits the best among these three approximations. An ANN model that is trained on a larger dataset consisting of thousands of datapoints could match the performance of Liu's approximation Eq.(2.17) but that will take a lot of time to train the neural network, nevertheless, it could be one of the future research.

5.2.2. Assessment on real-world networks

For the real-world networks, prediction of n_D for the entire range of the fraction of removed links l is not possible with such a small dataset as the predictions always performed worse than the analytical approximations. So, we followed a different approach for real-world networks under random attacks. As we already explained in the previous section, there is not much difference between the approximation value of n_D and the simulation value at $l = l_c$ as the original approximation is already good at $l = l_c$.

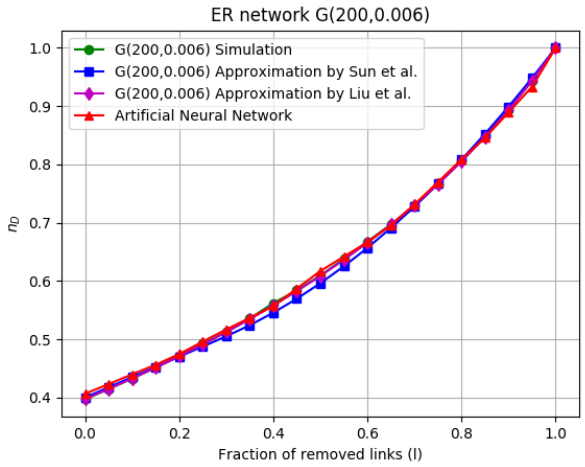

Figure 5.7: Performance comparison of the machine learning based approximation, Sun's approximation Eq.(2.15) and Liu's approximation Eq.(2.17) for the normalized minimum number of driver nodes n_D as a function of the fraction of removed links l in synthetic networks under random attacks. Simulation is based on 10,000 realizations of attacks.

Table 5.5: Performance indicators for all three approximations for ER networks under random attacks.

Network	**Mean Relative Error**		
	Approximation by Sun *et al.* Eq.(2.15)	**ANN**	**Approximation by Liu *et al.* Eq.(2.17)**
ER(100, 0.015)	0.0162	0.0084	0.0045
ER(100, 0.017)	0.0156	0.0097	0.0020
ER(200, 0.006)	0.0117	0.0059	0.0018

So, we choose a different value ($l = 0.4$) to predict the difference as the difference between the approximation and simulation is significant for larger values of the fraction of removed links l. We subtract the predicted difference from the approximation value of n_D at $l = 0.4$ to get a new value of n_D that is closer to the simulation. Now we build our machine learning based approximation. Let the value of $l = 0.4$ be l_X. At $l = 0$, the normalized minimum number of driver nodes is n_{DO} and at $l = l_X$, the value is n_{DX}. Using these two points we get,

$$n_{D,rand,ML} = n_{DO} + \frac{n_{DX} - n_{DO}}{l_X} l, \tag{5.4}$$

where, $n_{D,rand,ML}$ represents the new machine learning based approximation for the normalized minimum number of driver nodes as a function of the fraction of removed links for $l \leq l_X$. Similar to Eq.(2.15), for $l \geq l_X$, the normalized minimum number of driver nodes is approximated by a parabolic equation of the form,

$$n_{D,rand,ML} = a_{ML} l^2 + b_{ML} l + c_{ML}, \tag{5.5}$$

where a_{ML}, b_{ML} and c_{ML} are derived from the following boundary conditions.

- At $l = l_X$, Eq.(5.5) has the same value and derivative as Eq.(5.4). Hence, we get $a_{ML}l_X^2 + b_{ML}l_X + c_{ML} = n_{DX}$ and $2a_{ML}l_X + b_{ML} = \frac{n_{DX}-n_{DO}}{l_X}$.
- When all the links are removed, we need to control all the nodes. Hence, at $l = 1$, n_D equals one so that, $a_{ML} + b_{ML} + c_{ML} = 1$.

Using the above boundary conditions, we get the values of the parameters a_{ML}, b_{ML} and c_{ML} where,

$$a_{ML} = \frac{n_{DO} - 1 + \frac{n_{DX}-n_{DO}}{l_X}}{-l_X^2 + 2l_X - 1},$$

$$b_{ML} = \frac{n_{DX} - n_{DO}}{l_X} - 2a_{ML}l_X,$$

$$c_{ML} = 1 + a_{ML}(2l_X - 1) - \frac{n_{DX} - n_{DO}}{l_X}.$$

Finally, for the random attack, the machine learning based approximation for the normalized minimum number of driver nodes is expressed as,

$$n_{D,rand,ML} = \begin{cases} n_{DO} + \frac{n_{DX}-n_{DO}}{l_X}l, & l \leq l_X \\ a_{ML}l^2 + b_{ML}l + c_{ML}, & l \geq l_X \end{cases} \tag{5.6}$$

Now we will assess the performance of the machine learning based approximation in estimating the normalized minimum number of driver nodes n_D as a function of the fraction of removed links l for real-world networks under random attacks. Figure 5.8 shows the performance comparison of two real-world networks. It can be noticed that the ANN based approximation fits well with the simulation. To quantify the improvement in performance, Table 5.6 depicts the mean relative errors and mean absolute errors for 10 real-world networks. The ANN model performs the best in the Colt network with a mean relative error of 0.58 % and the least in the Uninet network with a mean relative error of 2.75 %. It should also be observed that in Ibm and Arpanet19706 network, the model does not perform well. The reason for this can be explained based on the availability of a limited number of training real-world networks and hence, the model does not fit well on all the real-world networks. As already pointed out, we test our model on 40 real-world networks, and based on the mean relative errors, our model performs better than the analytical approximations in 28 real-world networks. In other words, ANN based approximation outperforms the analytical approximation Eq.(2.15) in 70 % of the cases. Additionally, we observe that the mean relative errors never exceed 3 % for 29 out of 30 remaining cases using machine learning based approximation. Also in the cases where the analytical approximation fits better than the machine learning based approximation, the difference between the two is always less than 3 %. A complete list of results for the remaining 30 cases is also available at our NAS website [2].

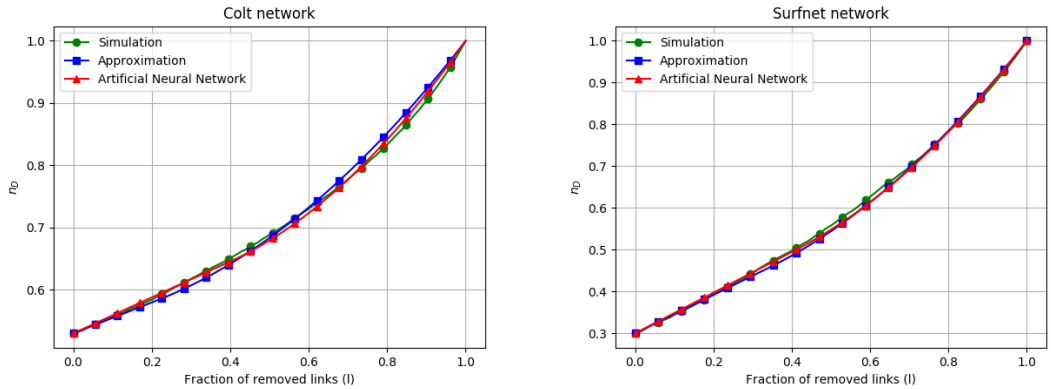

Figure 5.8: Performance comparison of the machine learning based approximations for the normalized minimum number of driver nodes n_D as a function of the fraction of removed links l in real-world networks under random attacks. Simulations are based on 10,000 realizations of attacks.

Table 5.6: Performance indicators for real-world networks under random attacks.

Network	**N**	**L**	**L_C**	**N_{DO}**	**Mean Absolute Error**		**Mean Relative Error**	
					Approx-imation	**ANN**	**Approx-imation**	**ANN**
Colt	153	177	38	81	0.0079	0.0043	0.0106	0.0058
Surfnet	50	68	23	15	0.0072	0.0052	0.0128	0.0090
EliBackbone	20	30	12	5	0.0256	0.0160	0.0454	0.0274
Garr200912	54	68	9	30	0.0121	0.0094	0.0156	0.0130
GtsPoland	33	37	12	14	0.0081	0.0046	0.0127	0.0068
Ibm	18	24	6	6	0.0072	0.0086	0.012	0.015
Arpanet19706	9	10	6	2	0.0046	0.0062	0.0073	0.0123
GtsHungary	30	31	8	18	0.0082	0.0072	0.0098	0.0088
BellCanada	48	64	17	16	0.0105	0.0071	0.0197	0.0122
Uninet	69	96	19	4	0.0207	0.0166	0.0338	0.0275

5.3. Out-in degree-based Attack

In this section, we will first derive the analytical approximation for out-in degree-based attacks and in the latter part, we will evaluate its performance. Out-in degree of a link is the sum of the out-degree of a source node and the in-degree of a target node.

5.3.1. Analytical Approximation: Out-in degree-based attacks

In this section, we derive an analytical approximation for the normalized minimum number of driver nodes n_D as a function of the fraction of removed links l for out-in degree-based attacks. Before we proceed, we will show by simulations various types of out-in degree-based attacks and then choose the most efficient one. In other words, we will attack the network based on, first, the increasing order of out-in degrees, second, if the out-in degrees are the same, we remove the links with out degrees in increasing order and third, we remove links in decreasing order of out-in degrees. Figure 5.9 shows that the third option, decreasing order of out-in degrees is the least efficient one, so, we will not consider this case. Also, the first two cases almost overlap with each other, so we consider only the first case in which we remove the links based on the increasing order of out-in degrees. It should be noted that in out-in degree-based attack, we remove the links one by one based on the increasing order of out-in degrees and after each removal we re-calculate the out-in degrees.

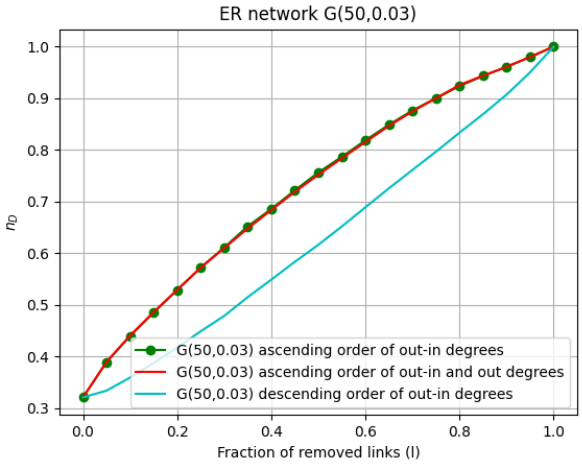

Figure 5.9: Comparison of different out-in degree-based attack strategies to select the most efficient one.

Case 1: $l \leq l_c$

When the fraction of removed links l is less than or equal to the fraction of critical links i.e. $l \leq l_c$, we assume a linear relationship similar to targeted critical link attack such

that the minimum number of driver nodes increases linearly with the fraction of removed links. The normalized number of driver nodes can be expressed as,

$$n_{D,out\_in} = \frac{N_{DO} + lL}{N}. \tag{5.7}$$

Case 2: $l \geq l_C$

When the fraction of removed links is greater than or equal to the fraction of critical links i.e. $l \geq l_C$, then for the normalized minimum number of driver nodes, we use a quadratic relationship of the form,

$$f(l) = n_D = min(gl^2 + hl + i, 1), \tag{5.8}$$

where g, h and i are derived from the boundary conditions presented below. Moreover, we use $min(gl^2+hl+i, 1)$ as the approximation might exceed the maximum value of 1 for some fraction of removed links l.

- At $l = l_C$, using Eq.(5.7), n_D equals $\frac{N_{DO}+l_C L}{N}$.
- At $l = 1$, all the links are removed and we need to control all the nodes, hence $n_D = 1$.
- At $l = 1$, the derivative equals zero or $f'(1) = 0$.

Using these three boundary conditions, we get the values of the parameters g, h and i where,

$$g = \frac{x - 1}{l_C^2 - 2l_C + 1},$$

$$x = \frac{N_{DO} + l_C L}{N},$$

$$h = -2g,$$

$$i = 1 - g - h.$$

Finally, for out-in degree-based attacks, the normalized minimum number of driver nodes can be expressed as,

$$n_{D,out\_in} = \begin{cases} \frac{N_{DO}+lL}{N}, & l \leq l_C \\ min(gl^2 + hl + i, 1), & l \geq l_C \end{cases} \tag{5.9}$$

5.3.2. Performance analysis of the approximation

In this section, we evaluate the performance of Eq.(5.9) for the normalized minimum number of driver nodes n_D as a function of the fraction of removed links l. First, we will test this approximation on synthetic networks. Figure 5.10 shows the analytical approximation Eq.(5.9) in comparison to simulation for Erdős-Rényi and Barabási-Albert networks. We notice that the approximation is better for BA networks than for ER networks which is also evident from the relative errors depicted in Table 5.7. The mean relative errors for ER networks are greater than 10 % and for BA networks, the errors are less than 3 %.

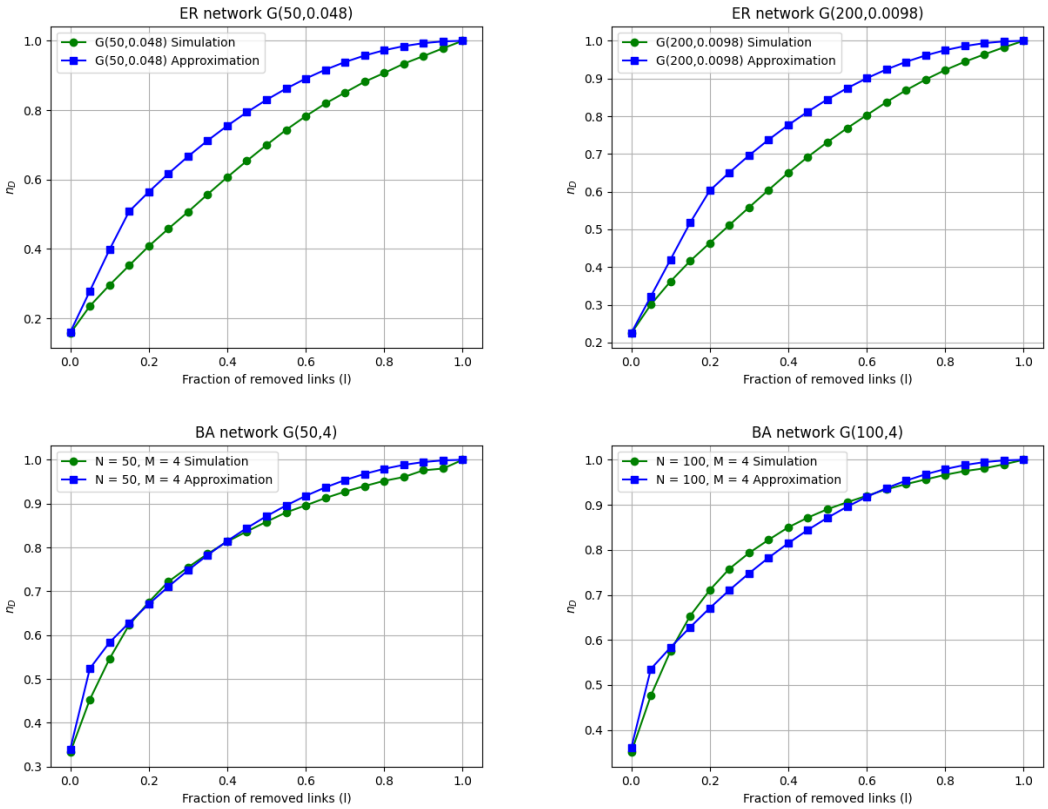

Figure 5.10: Performance comparison of the analytical approximations for the normalized minimum number of driver nodes n_D as a function of the fraction of removed links l for ER and BA networks under out-in degree-based attacks.

Now, we will test the approximation Eq.(5.9) on real-world networks. Figure 5.11 shows the comparison of the approximation with simulation for Colt and Surfnet networks. It should be noted that the approximation curves do not show the average values as the real-world networks have a fixed number of nodes and links and we attack the networks based on fixed ascending order of out-in degrees. Hence, these curves de-

Table 5.7: Performance indicators for synthetic networks under out-in degree-based attacks.

Network	Mean Absolute Error (Approximation)	Mean Relative Error (Approximation)
ER(50, 0.048)	0.0959	0.1786
ER(100, 0.02)	0.0828	0.1380
ER(200, 0.0098)	0.0782	0.1271
BA(50, 2)	0.0117	0.0142
BA(50, 4)	0.0193	0.0278
BA(100, 2)	0.0113	0.0143
BA(100, 4)	0.0201	0.0276

pict the actual values rather than averages. Additionally, we evaluate the performance of the approximation Eq.(5.9) based on the mean relative errors for 8 more real-world networks as shown in Table 5.8. We observe that in 8 out of 10 networks, the mean relative errors are less than 10 %. The approximation fits the best in the GtsHungary network with a mean relative error of 1.53 % and the least in the Uninet network with a mean relative error of 13.61 %.

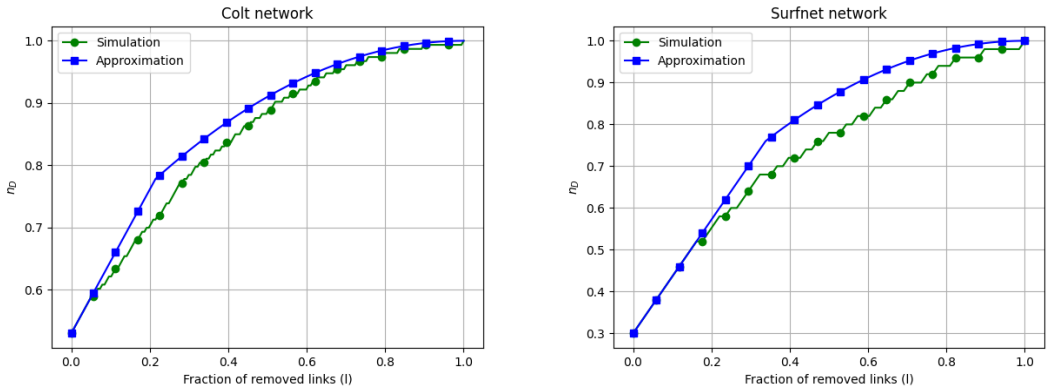

Figure 5.11: Performance of the approximations for the normalized minimum number of driver nodes n_D as a function of the fraction of removed links l in real-world networks under out-in degree-based attacks.

Now, we will use ANN to further improve the approximation Eq.(5.9). We will predict the difference in the normalized minimum number of driver nodes n_D between the approximation Eq.(5.9) and simulation at l_C so that the approximation value of n_D moves closer to the simulation at l_C. We use the predicted difference to build our machine learning based approximation. We subtract the predicted difference from the approximation Eq.(5.9) at l_C to get a new value of n_D. Let this new value be n_{DX}. Next we build our machine learning based approximation in the following manner.

Similar to the analytical approximation derived in Section 5.3.1, for the fraction of removed links less than or equal to the fraction of critical links i.e. for $l \leq l_C$, we assume a linear relationship for the normalized minimum number of driver nodes n_D as a function

Table 5.8: Performance indicators for real-world networks under out-in degree-based attacks.

Network	N	L	L_C	N_{DO}	Mean Absolute Error (Approximation)	Mean Relative Error (Approximation)
Colt	153	177	38	81	0.0210	0.0267
Surfnet	50	68	23	15	0.0469	0.0609
Elibackbone	20	30	12	5	0.0846	0.1188
Garr200912	54	68	9	30	0.0229	0.0262
GtsPoland	33	37	12	14	0.0256	0.0309
Ibm	18	24	6	6	0.0665	0.0922
Arpanet19706	9	10	6	2	0.0416	0.0522
GtsHungary	30	31	8	18	0.0140	0.0153
BellCanada	48	64	17	16	0.0546	0.0742
Uninet	69	96	19	4	0.0956	0.1361

of the fraction of removed links l. At $l = 0$, the normalized minimum number of driver nodes is n_{DO} and at $l = l_C$, the value is n_{DX}. Using these two points we get,

$$n_{D,out\_in,ML} = n_{DO} + \frac{n_{DX} - n_{DO}}{l_C} l, \tag{5.10}$$

where $n_{D,out\_in,ML}$ represents the new machine learning based approximation for the normalized minimum number of driver nodes. For the fraction of removed links greater than or equal to the fraction of critical links i.e. for $l \geq l_C$, we assume a quadratic relationship similar to Eq.(5.9) such that,

$$f_{ML}(l) = n_{D,out\_in,ML} = min(g_{ML} l^2 + h_{ML} l + i_{ML}, 1), \tag{5.11}$$

where g_{ML}, h_{ML} and i_{ML} are derived from the following boundary conditions.

- At $l = l_C$, $n_{D,out\_in,ML}$ equals n_{DX}.
- At $l = 1$, all the links are removed and we need to control all the nodes, hence $n_{D,out\_in,ML} = 1$.
- At $l = 1$, the derivative equals zero or $f'_{ML}(1) = 0$.

Using these three boundary conditions, we get the values of the parameters g_{ML}, h_{ML} and i_{ML} where,

$$g_{ML} = \frac{n_{DX} - 1}{l_C^2 - 2l_C + 1},$$
$$h_{ML} = -2g_{ML},$$
$$i_{ML} = 1 - g_{ML} - h_{ML}.$$

Finally, for out-in degree-based attacks, the machine learning based approximation for the normalized minimum number of driver nodes is expressed as,

$$n_{D,out\_in,ML} = \begin{cases} n_{DO} + \frac{n_{DX} - n_{DO}}{l_C} l, & l \leq l_C \\ min(g_{ML} l^2 + h_{ML} l + i_{ML}, 1), & l \geq l_C \end{cases} \tag{5.12}$$

Assessment on synthetic networks

In this section, we will compare the performance of the machine learning based approximation Eq.(5.12) and the analytical approximation Eq.(5.9) with the simulations on synthetic networks.

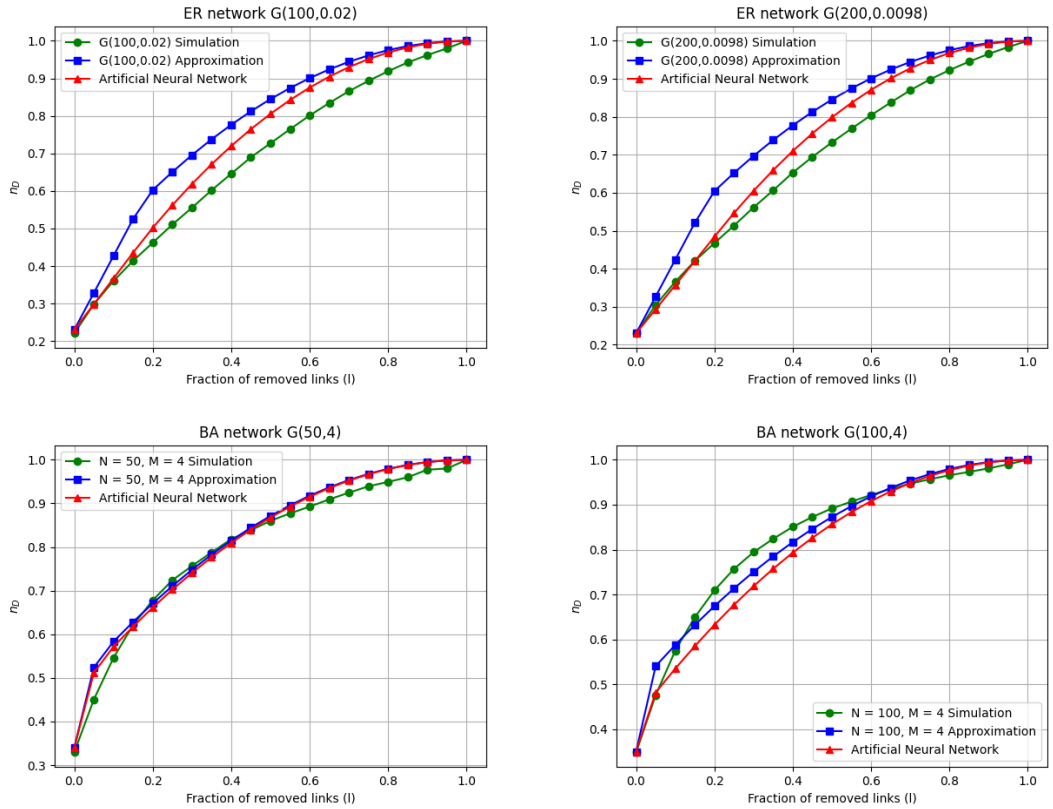

Figure 5.12: Performance comparison of the machine learning based approximations for the normalized minimum number of driver nodes n_D as a function of the fraction of removed links l in synthetic networks under out-in degree-based attacks.

Figure 5.12 shows that the machine learning based approximation fits better with simulation for Erdős-Rényi networks. For Barabási-Albert networks, the analytical approximation already fits well with the simulation and we do not see any significant improvement with machine learning based approximation. The same is also evident from

Table 5.9 where, for ER networks, we see that the mean relative errors for machine learning based approximation Eq.(5.12) are less than 10 % while, the errors are greater than 10 % for the analytical approximation Eq.(5.9). For BA networks, the mean relative errors are already less than 3 %. So, the machine learning based approximation does not significantly improve the analytical approximation. For $BA(100, 4)$, we also see an increase in the mean relative error from 2.76 % to 4 % using machine learning based approximation. The overall conclusion of this section is that the analytical approximation Eq.(5.9) fits well with simulation for BA networks. The analytical approximation Eq.(5.9) is also decent for ER networks but the machine learning based approximation Eq.(5.12) fits better with the simulations.

Table 5.9: Performance indicators for synthetic networks under out-in degree-based attacks.

Network	**Mean Absolute Error**		**Mean Relative Error**	
	Approximation	**ANN**	**Approximation**	**ANN**
ER(50, 0.0381)	0.0872	0.0435	0.1421	0.0618
ER(50, 0.048)	0.0959	0.0568	0.1786	0.0924
ER(100, 0.02)	0.0828	0.0463	0.1380	0.0680
ER(200, 0.0098)	0.0782	0.0372	0.1271	0.0521
BA(50, 4)	0.0193	0.0189	0.0278	0.0266
BA(100, 4)	0.0201	0.0308	0.0276	0.0400

Assessment on real-world networks

In this section, we will compare the performance of the machine learning based approximation Eq.(5.12) and the analytical approximation Eq.(5.9) with the simulations on real-world networks.

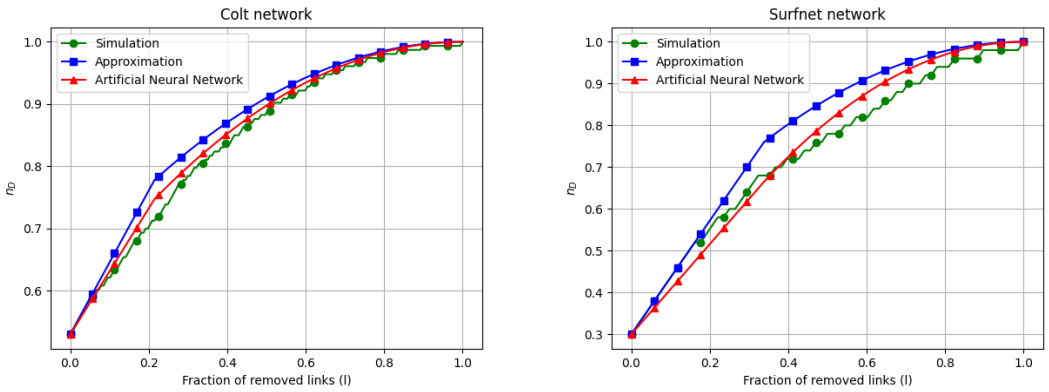

Figure 5.13: Performance of the approximations for the normalized minimum number of driver nodes n_D as a function of the fraction of removed links l in real-world networks under out-in degree-based attacks.

Figure 5.13 shows that the machine learning based approximation Eq.(5.12) fits bet-

ter with simulation than the analytical approximation Eq.(5.9). In Table 5.10 we compare the performance of these approximations and we see that the machine learning based approximation performs better than the analytical approximation in 7 out of 10 real-world networks based on the mean relative errors. We also notice that the mean relative errors for all the 10 real-world networks are less than 10 % using machine learning based approximation. In the remaining 24 out of 30 networks, the mean relative errors are less than 6 % for machine learning based approximation.

Table 5.10: Performance indicators for real-world networks under out-in degree-based attacks.

Network	**N**	**L**	$\mathbf{L_C}$	$\mathbf{N_{DO}}$	**Mean Absolute Error**		**Mean Relative Error**	
					Approx-imation	**ANN**	**Approx-imation**	**ANN**
Colt	153	177	38	81	0.0210	0.0102	0.0267	0.0129
Surfnet	50	68	23	15	0.0469	0.0280	0.0609	0.0395
EliBackbone	20	30	12	5	0.0846	0.0373	0.1188	0.0539
Garr200912	54	68	9	30	0.0229	0.0213	0.0262	0.0242
GtsPoland	33	37	12	14	0.0256	0.0357	0.0309	0.0447
Ibm	18	24	6	6	0.0665	0.0682	0.0922	0.0951
Arpanet19706	9	10	6	2	0.0416	0.0340	0.0522	0.0519
GtsHungary	30	31	8	18	0.0140	0.0135	0.0153	0.0148
BellCanada	48	64	17	16	0.0546	0.0657	0.0742	0.0917
Uninet	69	96	19	4	0.0956	0.0586	0.1361	0.0829

5.4. Robustness Envelopes

So far, we have only considered the average case scenarios in all three types of attacks. It is also important to analyze the variability of simulations over different realizations of attacks. For example, in case of targeted critical link attack, we first remove all the critical links uniformly at random and then the remaining links also at random. As we remove links randomly, different realizations of attacks lead to different values of minimum number of driver nodes and hence, it becomes essential to also take into account such variability of attacks.

In this section, we show how different realizations of attacks lead to different values of the normalized minimum number driver nodes as a function of the number of challenges and study the robustness of network controllability with the help of envelopes. The number of challenges corresponds to the fraction of removed links l. For example, when the number of challenges equals 20, then it means 20 % of the links are removed or the fraction of removed links equals 0.2. It should be noted that we will only show the robustness envelopes for targeted critical link attack and random attack because out-in degree-based attacks do not have attack based variability as the links are removed

using a fixed order, in our case, based on the increasing order of out-in degrees. So, there is no attack based variability in case of out-in degree-based attacks.

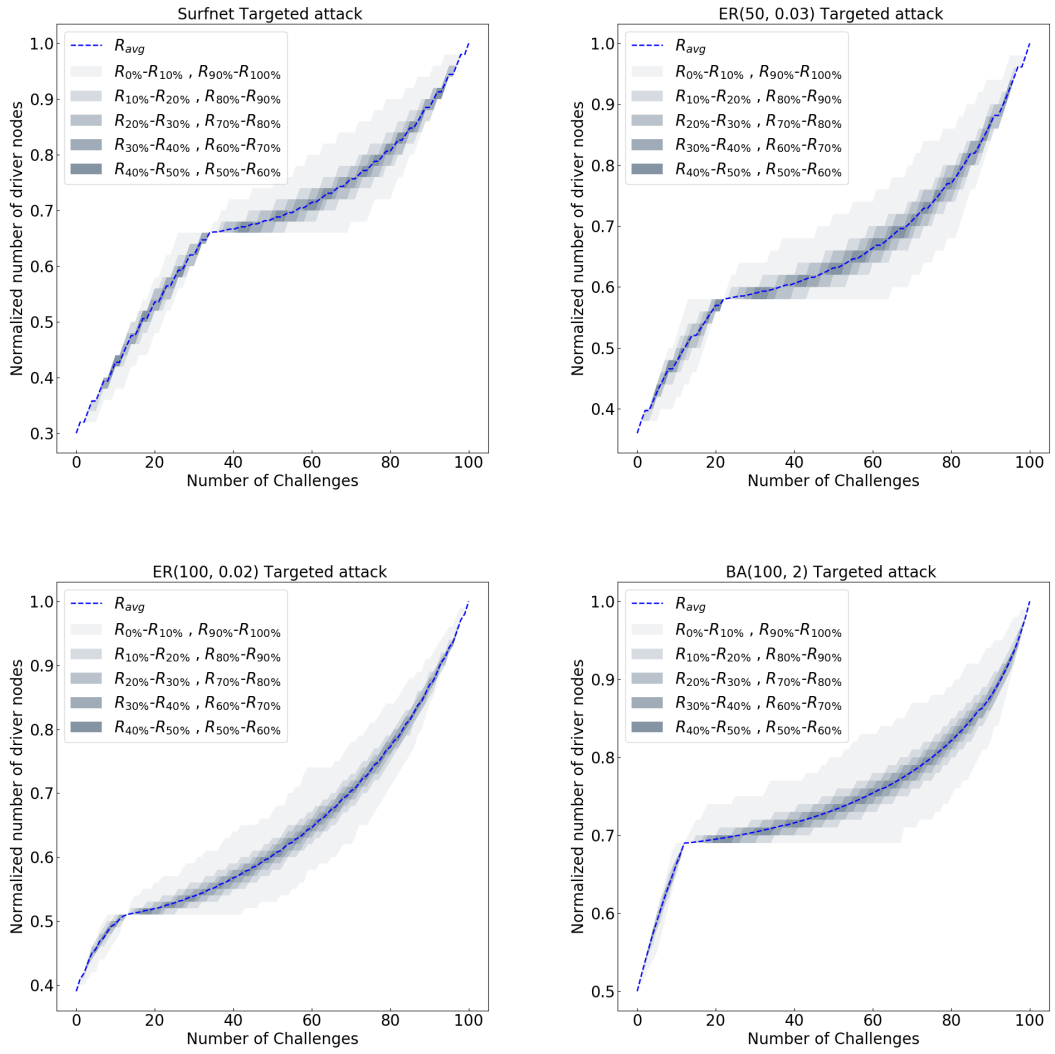

Figure 5.14: Robustness envelopes for different networks under targeted critical link attack based on 10,000 realizations of attacks.

Figure 5.14 shows the robustness envelopes for targeted critical link attack on both real-world and synthetic networks. As we already pointed out, we use network controllability as the robustness metric. It can be noticed that when the fraction of removed links is less than or equal to the fraction of critical links, then the variability is less and we can say that the average case gives a good indication of robustness of network controllability. For $l \geq l_c$, the envelope is wider and the number of driver nodes varies a lot. Furthermore, we also notice that as the network becomes larger, the variability de-

creases. In Figure 5.15, we study the envelopes for random attacks on both real-world and synthetic networks. We observe a behavior similar to targeted attacks here as well when we increase the size of the network. As the number of nodes and links increases, the envelopes become narrow and network controllability shows less variations with each fraction of removed links.

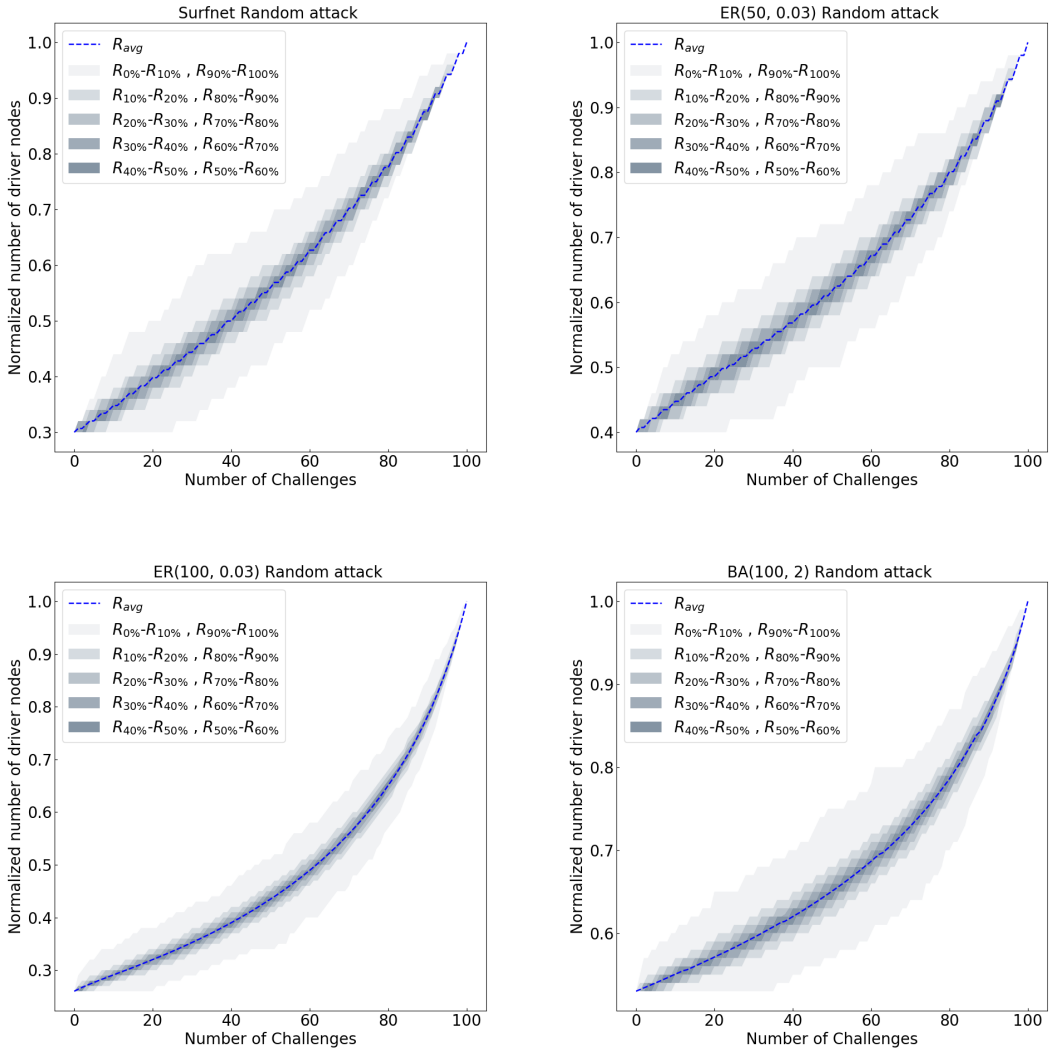

Figure 5.15: Robustness envelopes for different networks under random attack based on 10,000 realizations of attacks.

6

Conclusions and Future Work

In this chapter, we will summarize our work and also propose suggestions for future research. We start with the main conclusions of this thesis in Section 6.1 followed by some recommendations for future research in Section 6.2.

6.1. Conclusions

In this thesis, we have used three machine learning algorithms, multi-linear regression, Random Forest, and Artificial Neural Network to estimate the normalized minimum number of driver nodes n_D as a function of the fraction of removed links l. We used these machine learning models to study the robustness of network controllability of both synthetic and real-world networks under targeted critical link attack, random attack, and out-in degree-based attack. Simulations are often slow and the existing analytical approximations for targeted and random attacks perform fairly well but the relative errors are large. Then, we use machine learning to improve the analytical approximations. Furthermore, we also derive analytical approximations for out-in degree-based attack. In addition to this, we also study the variability of attacks using envelopes.

We first evaluate the performance of machine learning models and conclude that ANN performs the best but takes more time to train as compared to Random Forest. For targeted critical link attacks, our ANN model outperforms the analytical approximation for all the synthetic test networks, while the performance is 75 % for real-world networks. Also, for random attacks, the normalized minimum number of driver nodes as a function of the fraction of removed links using the ANN model outperforms the analytical approximations for all the synthetic test networks, while the performance is 70 % for real-world networks. We conclude that this is due to the availability of a limited number of real-world networks for training. On the other hand, for synthetic networks, we generate sufficient data through simulations and hence, the model relatively performs better on synthetic networks. In addition to this, the approximation based on implicit equations by Liu *et al.* Eq.(2.17) for ER networks under random attacks performs better

than both machine learning based approximation and Sun's approximation Eq.(2.15).

Furthermore, we derive the analytical approximation for out-in degree-based attack and assess its performance on both synthetic and real-world networks. Based on the results presented in Section 5.3.2, we conclude that the approximation fits well on both synthetic and real-world networks. The approximation fits the best in BA networks where the mean relative errors are less than 3 % while, for ER networks, the errors are greater than 10 %. Furthermore, in 8 out of 10 considered real-world networks, the mean relative errors are less than 10 %. In addition to this, we also improve the approximation using ANN for both synthetic and real-world networks and conclude that for BA networks, the analytical approximation already fits well with the mean relative errors of less than 3 % and the ANN based approximation does not lead to any significant further improvements. However, for ER networks, the mean relative errors reduce to less than 10 % using machine learning based approximation. Furthermore, the mean relative errors also reduce to less than 6 % in 7 out of 10 considered real-world networks.

Finally, we also study the variability of attacks as we uniformly remove links at random. For targeted attacks, we first remove all the critical links randomly, and then the remaining links also at random. In the case of random attacks, we remove all the links uniformly at random. Due to this, we experience attack based variability and hence, we use the envelope method to account for such variability. For out-in degree-based attacks, we remove links in a fixed ascending order of out-in degrees, and hence, there is no attack based variability.

6.2. Recommendations for future research

Based on the results and conclusions, we recommend applying ANN on a larger dataset consisting of thousands of real-world networks. A larger dataset would also allow us to train the machine learning models to predict the normalized minimum number of driver nodes n_D for the entire range of the fraction of removed links l for both targeted and random attacks. We have seen that the ANN based approximations are better than the analytical approximations, another interesting area would be to derive new analytical approximations for both targeted critical link and random attacks. Furthermore, our approximation for out-in degree-based attack performs well on Barabási-Albert networks but the errors are still large for Erdős-Rényi and real-world networks. A better analytical approximation for the out-in degree-based attacks would also be an interesting research.

Bibliography

[1] Réka Albert and Albert-László Barabási. Statistical mechanics of complex networks. *Reviews of Modern Physics*, 74(1):47, 2002.

[2] Network Architecture and Services. Delft university of technology. https://www.nas.ewi.tudelft.nl/.

[3] Albert-László Barabási, Erzsébet Ravasz, and Tamas Vicsek. Deterministic scale-free networks. *Physica A: Statistical Mechanics and its Applications*, 299(3-4): 559–564, 2001.

[4] Ulrik Brandes, Markus Eiglsperger, Ivan Herman, Michael Himsolt, and M Scott Marshall. Graphml progress report structural layer proposal. In *International Symposium on Graph Drawing*, pages 501–512. Springer, 2001.

[5] Hale Cetinay, Karel Devriendt, and Piet Van Mieghem. Nodal vulnerability to targeted attacks in power grids. *Applied Network Science*, 3(1):34, 2018.

[6] Reuven Cohen, Keren Erez, Daniel Ben-Avraham, and Shlomo Havlin. Resilience of the internet to random breakdowns. *Physical Review Letters*, 85(21):4626, 2000.

[7] Reuven Cohen, Keren Erez, Daniel Ben-Avraham, and Shlomo Havlin. Breakdown of the internet under intentional attack. *Physical Review Letters*, 86(16): 3682, 2001.

[8] Noah J. Cowan, Erick J. Chastain, Daril A. Vilhena, James S. Freudenberg, and Carl T. Bergstrom. Nodal dynamics, not degree distributions, determine the structural controllability of complex networks. *PLoS ONE*, 7(6), 2012. ISSN 19326203. doi: 10.1371/journal.pone.0038398.

[9] Paul Erdős and Alfréd Rényi. On the evolution of random graphs. *Publ. Math. Inst. Hung. Acad. Sci*, 5(1):17–60, 1960.

[10] Aurélien Géron. *Hands-On Machine Learning with Scikit-Learn and TensorFlow*. O'Reilly Media. ISBN 9781491962299.

[11] Frank Harary. The determinant of the adjacency matrix of a graph. *Siam Review*, 4(3):202–210, 1962.

[12] Michael Himsolt. Gml: A portable graph file format. Technical report, Technical report, Universitat Passau, 1997.

[13] Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, abs/1207.0580, 2012. URL http://arxiv.org/abs/1207.0580.

[14] Petter Holme, Beom Jun Kim, Chang No Yoon, and Seung Kee Han. Attack vulnerability of complex networks. *Physical Review E*, 65(5):056109, 2002.

[15] John E Hopcroft and Richard M Karp. An n^5/2 algorithm for maximum matchings in bipartite graphs. *SIAM Journal on Computing*, 2(4):225–231, 1973.

[16] Xuqing Huang, Jianxi Gao, Sergey V Buldyrev, Shlomo Havlin, and H Eugene Stanley. Robustness of interdependent networks under targeted attack. *Physical Review E*, 83(6):065101, 2011.

[17] Rudolf Emil Kalman. Mathematical description of linear dynamical systems. *Journal of the Society for Industrial and Applied Mathematics, Series A: Control*, 1(2): 152–192, 1963.

[18] Keras. Landing page. https://keras.io/, Last accessed: June 27, 2020.

[19] Simon Knight, Hung X Nguyen, Nickolas Falkner, Rhys Bowden, and Matthew Roughan. The internet topology zoo. *IEEE Journal on Selected Areas in Communications*, 29(9):1765–1775, 2011.

[20] Yakup Koç, Martijn Warnier, Piet Van Mieghem, Robert E. Kooij, and Frances M.T. Brazier. The impact of the topology on cascading failures in a power grid model. *Physica A: Statistical Mechanics and its Applications*, 402:169–179, 2014. ISSN 03784371. doi: 10.1016/j.physa.2014.01.056. URL http://dx.doi.org/10.1016/j.physa.2014.01.056.

[21] Ching-Tai Lin. Structural controllability. *IEEE Transactions on Automatic Control*, 19(3):201–208, 1974.

[22] Yang Yu Liu, Jean Jacques Slotine, and Albert László Barabási. Controllability of complex networks. *Nature*, 473(7346):167–173, 2011. ISSN 00280836. doi: 10.1038/nature10011.

[23] Simachew Abebe Mengiste, Ad Aertsen, and Arvind Kumar. Effect of edge pruning on structural controllability and observability of complex networks. *Scientific Reports*, 5(December):1–14, 2015. ISSN 20452322. doi: 10.1038/srep18145.

[24] NetworkX. Network analysis in python. https://networkx.github.io/, Last accessed: July 19, 2020.

[25] Sen Nie, Xuwen Wang, Haifeng Zhang, Qilang Li, and Binghong Wang. Robustness of controllability for networks based on edge-attack. *PLoS ONE*, 9(2), 2014. ISSN 19326203. doi: 10.1371/journal.pone.0089066.

[26] Cun Lai Pu, Wen Jiang Pei, and Andrew Michaelson. Robustness analysis of network controllability. *Physica A: Statistical Mechanics and its Applications*, 391 (18):4420–4425, 2012. ISSN 03784371. doi: 10.1016/j.physa.2012.04.019. URL http://dx.doi.org/10.1016/j.physa.2012.04.019.

[27] Scikit-learn. Supervised learning. https://scikit-learn.org/stable/supervised_learning.html#supervised-learning, Last accessed: June 24, 2020.

[28] A. Socievole, F. De Rango, C. Scoglio, and P. Van Mieghem. Assessing network robustness under SIS epidemics: The relationship between epidemic threshold and viral conductance. *Computer Networks*, 103:196–206, 2016. ISSN 13891286. doi: 10.1016/j.comnet.2016.04.016.

[29] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014. URL http://jmlr.org/papers/v15/srivastava14a.html.

[30] Peng Sun, Robert E. Kooij, Zhidong He, and Piet Van Mieghem. Quantifying the Robustness of Network Controllability. *2019 4th International Conference on System Reliability and Safety, ICSRS 2019*, pages 66–76, 2019. doi: 10.1109/ICSRS48664.2019.8987628.

[31] TensorFlow. Landing page. https://www.tensorflow.org/, Last accessed: June 27, 2020.

[32] Thomas M Tirpak. Telecommunication network resource management based on social network characteristics, November 11 2010. US Patent App. 12/463,445.

[33] Stojan Trajanovski, Javier Martín-Hernández, Wynand Winterbach, and Piet Van Mieghem. Robustness envelopes of networks. *Journal of Complex Networks*, 1 (1):44–62, 2013. ISSN 20511329. doi: 10.1093/comnet/cnt004.

[34] Remco van der Hofstad. Random graphs models for complex networks, and the brain. *Complexity Science: An Introduction*, page 199, 2019.

[35] P Van Mieghem, C Doerr, H Wang, J Martin Hernandez, D Hutchison, M Karaliopoulos, and RE Kooij. A framework for computing topological network robustness. *Delft University of Technology, Report20101218*, 2010.

[36] Xiangrong Wang, Evangelos Pournaras, Robert E. Kooij, and Piet Van Mieghem. Improving robustness of complex networks via the effective graph resistance. *European Physical Journal B*, 87(9):1–12, 2014. ISSN 14346036. doi: 10.1140/epjb/e2014-50276-0.