A Deep Learning based Framework for Detecting and Visualizing Online Malicious Advertisement

by

Xichen Zhang

Master of Chemical Engineering, China National Pulp and Paper Research Institute, Beijing, China, 2013

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

Master of Computer Science

In the Graduate Academic Unit of Faculty of Computer Science

Supervisor:	Ali A. Ghorbani, Ph.D., Faculty of Computer Science
Examining Board:	Huajie Zhang, Ph.D., Faculty of Computer Science, Chair
	Rongxing Lu, Ph.D., Faculty of Computer Science
	Hsin-Chen Lin, Ph.D., Faculty of Business Administration

This thesis is accepted

Dean of Graduate Studies

THE UNIVERSITY OF NEW BRUNSWICK

January, 2018

Abstract

Online advertisement detection has attracted increasing attention in recent years in both academia and industry. The existing advertising blocking systems are vulnerable to the evolution of new attacks and can cause time latency issues by querying remote servers. In this thesis, we propose a lightweight detection system depending only on lexical-based features. The statistical analysis of different URL sources shows that lexical-based features can provide sufficient information for online advertisement representation. Deep learning algorithms are used for online advertising classification and visualization. After optimizing the architecture of the deep neural network, the deep learning based approach can achieve satisfactory detection results with false negative rate as low as 1.31%. In URL visualization, with the implementation of AutoEncoder for feature preprocessing and *t-SNE* for visualization, our approach outperforms other dimensionality reduction algorithms and can generate clear clusters for different URL families. Finally, a sequence-2-sequence model is built for fake URL generation, which can create distinguishing characteristics for fake malicious URLs.

Dedication

To my parents Zhijun Zhang and Lijuan Wang, my grandmother Guilan Wang, for all the love you give to me during my life.

To my wife Jiejie Wang, for the love, encourage and support during my study.

To my son Luke Zhang, for the happiness and hope you give to me during this journey.

Acknowledgements

I would like to express the deepest appreciation to my supervisor, Dr. Ali A. Ghorbani, for the continuous help and guidance during my master study. Your scientific attitude and creative thinking give me a more open view of doing research. Your caring and patience give me an excellent atmosphere for my study. Without your persistent support, this dissertation would not have been possible.

I would like to thank Dr. Arash Habibi Lashkari, who as a dedicated project leader, an excellent mentor, and a good friend, is always willing to help and give your best suggestions. It is really an unforgettable experience working with you and I could not have imagined having a better advisor.

My sincere thanks also goes to Dr. Rongxing Lu, for your time, encouragement and insightful comments.

I would like to thank all the research colleagues, students and staff at Canadian Institute for Cybersecurity for the regular discussions and exciting projects we work on together. I really learn a lot from you and I hope we can work together again in the future.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

t-SNE: t-distributed stochastic neighbor embedding

CNN: Convolutional Neural Network

RNN: Recurrent Neural Network

MLP: Multiple Layers Perceptron

NLP: Natural Language Processing

DBN: Deep Belief Network

RBM: Restricted Boltzmann Machine

DNN: Feedforward Deep Neural Network

PCA: Principle Component Analysis

LLE: Local Linear Embedding

LSTM: Long Short-Term Memory

Chapter 1

Introduction

1.1 Problem Statement

The explosive development of the Internet has recently witnessed the tremendous growth in the number of online advertisement. Online advertising has become the major strategy for business propaganda and has had a deep impact on Internet activities, such as e-commerce, online banking, e-mail and social networking.

Unfortunately, because of the increasing popularity of online advertising, a massive amount of meaningless contents are displayed or downloaded during Web surfing, which greatly degrade the user experience. Furthermore, being extensively used for illegal purposes, online advertising becomes a big platform for delivering malicious activities, such as malware, scamming, spam, and frauds [50] [48]. By inserting malicious codes into legitimate websites,

attackers can automatically redirect victims to phishing pages or compromise their computers by installing malware. In addition, obfuscation techniques are always involved with the evolution of criminal enterprises, which makes identifying malicious online advertisements even more difficult [66].

Many latest online URL detection systems, such as *PhishTank* [63], *AdBlock* [56] and *Easylist* [24] are based on blacklist filters or URL-pattern databases. Large human involvement is necessary for maintaining such systems, and time latency is the main issue that prevents end users from visiting potentially dangerous webpages. In order to improve the generality of existing systems, machine learning techniques are explored for malicious URL detection [50] [48] [70] [98] [82]. Features used for URL classification can be categorized as lexical-based features and external features [41]. External features such as WHOIS property are extremely dependent on the availability of remote servers and can cause latency issues by remote query requesting. So a lightweight detection system is highly desired in order to efficiently identify online advertisement URLs.

1.2 Motivation and Objective

As the rapid growth of online advertising, a massive amount of unnecessary and intrusive information is downloaded during Web surfing [82]. Users cannot get access to the useful messages on the Internet efficiently. And a majority of the customers consider online advertisements as annoying, ob-

trusive, distracting and all over the places [2].

Furthermore, as online advertising develops as the predominant model for marketing and promotion today, malicious advertisements have become the cornerstone for fraudulent activities, such as propagating malware, scamming, click frauds, etc. [48]. Currently, there are potential interests in building systems to prevent customers from visiting these websites [50].

Globally during 2010 to 2015, the number of desktop computer users using ad-blocking softwares grow by 41% year over year. By June 2015, the number of monthly active ad-blocking users is 198 million [60]. Attackers and hackers consider online advertising as low-cost and highly effective means to conduct fraudulent activities. Consequently, hundreds of top ranking websites have been infected by malicious advertisements [48].

However, today the existing ad-blocking systems are always based on laboriously hand-coded rules or webpage content analysis. Tremendous human involvements are needed. Also, new types of malicious trend and new obfuscation techniques are introduced every day to evade detection. So these systems are vulnerable to the evolution of new malicious advertisements [82].

In addition, time latency is one of the biggest issues in online advertising identification. Any manual review of the blacklist or the analysis of the web pages content may take a considerable amount of time. According to *PhishTank*'s own statistic, the average verification time of a single URL ranges from tens to fifty hours[94]. As a result, many users may be exposed to the danger of malicious URLs during the update of the blacklist. All the reasons

mentioned above motivate us to design a lightweight system for malicious advertising detection with high performance.

1.3 Significance of the Study

The study of online advertisement detection is significant for the following reasons.

First, nowadays online advertisement has already grown into a billion-dollar business [58]. The estimated loss of global revenues due to adverting blocking is nearly $22 billion in 2015 [60]. So there is really a huge industrial demand for generating effective online advertisement blocking system.

Second, compared to traditional media such as television or newspaper, more interactions are involved between users and online advertisers. For instance, one could easily sign up an account, fill up some sensitive information or even do online shopping in an online advertisement webpage. As a result, hackers have found that online advertising to be an effective way to distribute malicious information and conduct fraudulent activities. More new trends of obfuscation techniques or malicious tactics have been developed for evading detection. So the establishment of online advertisement detection system can describe a clear landscape of the nature of the suspicious attacks. And this thesis can give a comprehensive guideline for detecting malicious advertising using lexical-based features.

Third, the finding of this study can be applied to build a more effective

and efficient blocking system for online advertisements. To mitigate the issues in previous research, lightweight lexical-based features are used to represent malicious entities. Online advertising datasets are selected and generated to mimic the real-world cases as much as possible. The deep learning algorithm is applied in order to perform a better detection output and make our system be applicable to more scenarios. This research outlines the foundation for establishing a deep learning based framework for online advertisement detection. It can provide useful solutions on characterizing online advertisement for both industry and academia. And we believe that in the future the application of our system can be used extensively among other cybersecurity fields such as phishing, spam or malware detection.

1.4 Summary of Contributions

The main purpose of this thesis is to design a deep learning based framework for online advertisement detection and visualization. And the contributions of this thesis can be summarized as follows: we propose a deep learning based approach for online advertisement classification and visualization. The statistical analysis of different URLs demonstrates that lexical-based features can help to distinguish different URL families and are good candidates for the lightweight online advertisement detection. We select the best architecture for training deep neural network by optimizing the number of hidden layers and the number of neurons in each hidden layer. The classification results

show that deep learning based approaches show better performance than machine learning techniques, with false negative rate as low as 1.31%. Also, we design a novel deep learning based method for data visualization. AutoEncoder plus *t-SNE* outperforms other dimensionality reduction algorithms by creating better clusters for different URL instances. Finally, with the implement of sequence-2-sequence model, we can generate fake advertising URLs with distinguishing characteristics.

1.5 Thesis Organization

The rest of the thesis is organized as follows:

- Chapter 2 demonstrates the background of online advertisements. It gives different scopes and varieties of normal online advertisements. This chapter also introduces some popular malicious advertising techniques and discusses why malicious advertisement detection becomes more complicated and difficult.

- Chapter 3 reviews the related works of literature on URL detection and the application of deep learning algorithms in cybersecurity. It concludes the machine learning techniques, datasets, and features used in the existing advertising blocking systems. It also addresses the limits and issues in earlier studies.

- Chapter 4 presents the proposed framework used in this thesis. First,

we describe in detail how lexical-based features are extracted from URL sources. Each URL is divided into 7 components (domain, port, path, filename, query, reference) and finally 141 lexical-based features are selected and used in our experiment. Then we carry out a comprehensively lexical analysis for different URL sources. This can help us find the most influential features for distinguishing advertisement URLs with normal URLs. In the next, we conduct a topology study for building an optimal deep neural network. Deep learning algorithms are applied for advertisement URL detection and visualization.

- Chapter 5 introduces the implementation of the experiment and the evaluation of the results. It discusses the dataset used in the experiment and then illustrates the deep learning algorithms applied for URL detection and visualization. In the evaluation section, first we show the lexical analysis for influential features, then we compare the detection results between deep learning algorithms and conventional machine learning techniques, after that, we design a novel unsupervised method for URL visualization, and finally, we implement a sequence-2-sequence model for fake URL generation.

- Chapter 6 recaps the conclusions of this thesis and provides an outlook for the future work.

Chapter 2

Background

2.1 Normal Advertisement

To clearly understand the scope and variety of online advertising, some important high-level concepts for classifying online advertisements are listed in the following [39] [6] :

- **Media and content**: The advertisement may use various types of media, e.g., text, image, video, and sound. And in the advertisement, the webpage content can be changed or new links can be added.
- **Intrusiveness**: The advertisement compels users to view it by blocking the webpage content. Users need to take an explicit action to dismiss the advertisement.
- **Interactivity**: The advertisement involves some interactions with the

user.

- **Privacy**: The advertiser can trace user behaviors and generate user profiles. The information can be stored in the logs and be kept for a long time.

Nowadays online advertisements tend to be much more interactive, intrusive, and creative with respect to the way they are designed. And it is not always easy to construct complete standardization for all the possible formats. Following are some popular and fundamental types of online advertisements.

- **Banner Advertisement**: Banner advertisements are typically rectangular advertisements that integrated into a webpage, displaying banners, images or multi-media objects (such as sounds and videos) [7] [39]. Most banner advertisements are clickable and the traffic linked to the advertiser will be generated after a user clicks on them [6]. Banner advertisements can interact with users by expanding, shrinking or downward-pushing the content they present.

- **Pop-up Advertisement**: Pop-up advertisements [7] are strongly intrusive and non-integrated advertisements, which appear in a separate window in front of the webpage content. Explicit action, such as closing the pop-up window, is needed to dismiss pop-up. Pop-under advertisement is a variation of pop-up, which is slightly less intrusive by displaying the window under the main content.

- **Floating Advertisement**: Floating advertisements [83] are intrusive advertisements with multi-media displays, which appear over the webpage content, and disappear or become unobtrusive after a specific time period without user interaction. Tear-back and peel-back advertisements are variations of floating advertisement, usually with a teaser to trigger click action from users.

- **In-text Advertisement**: In-text advertisements [6] can change the display size by adding links to the keywords in the content. Typically the keywords are visually distinct and can trigger the appearance of new content.

- **Transition Advertisement**: Transition advertisements [93] are interstitial advertisements that either being inserted between two pages of content or open in a new window intrusively. After displaying animated Flash or static messages for a certain period, transition advertisements will disappear and the user-requested webpage will appear.

- **Video Advertisement**: Video advertisements can display online videos on websites, and can be divided into two main categories: linear and non-linear [6] [22]. Linear advertisements appear before, during, or after a break in the video content, whereas non-linear advertisements appear along with the video content. Integrated into the webpage, video advertisements can also be displayed like a banner advertisement.

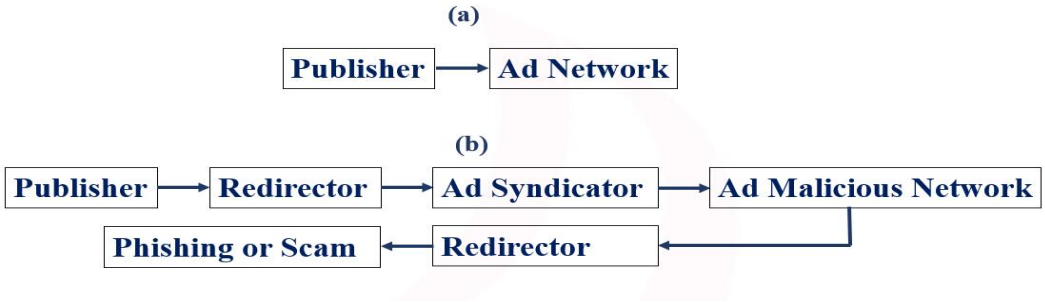

Figure 2.1: Advertisement delivery process: (a) A normal delivery chain (b) A delivery chain of advertising syndication

2.2 Malicious Advertisement

By visiting compromised websites, end users are at the high risk of infection. Various types of obfuscation and code packing techniques are implemented in order to evade detection [48], and the following categories are summarized as the most frequently used techniques by the attackers.

- **Phishing Advertising**: By letting end users visit such advertisement webpages, the attackers attempt to obtain private and sensitive information such as username, account name, password or financial messages. Usually, the infected webpages are distinguished as trustworthy entities in an electronic communication. Also, we should note that although the URL of a trusty webpage appears to be legitimate, the hyperlink would actually be pointed to a phishing page [96].

- **Drive-by Download**: When an end-user visits a malicious advertisement webpage, any download of a computer virus, spyware, or malware

may happen without the user's knowledge [74]. And this type of attack exploits the vulnerabilities of browsers or plugins using dynamic content in JavaScript or Flash [48].

- **Advertisement Syndication**: Advertising Syndication is highlighted in [48] and [101] as another popular malicious advertising technique and is shown in Figure 2.1. This method obfuscates the suspicious URLs among the advertising delivery chain, resulting in increased difficulty of detection and characterization. Instead of a normal delivery, the user is usually involved in multi-redirecting delivery steps. The delivery process can begin with a big and legitimate publisher, going through some normal advertising networks or redirections, and finally landing on a malicious advertising webpage.

- **Others**: Other malicious techniques include using HTML iframes to spread malware [78], involving third-party advertisers or applications to distribute malicious activities [67]. In addition, many obfuscation techniques are manipulated by attackers for evading detection and analysis. For example, just like advertising syndication, multiple entities (such as advertisement network, redirectors or fake advertising host) are always working together in one advertisement delivery. Different entities are controlled by different malicious parties, which makes analyzing the malicious URLs more complicated[48].

Chapter 3

Related Work

In this chapter, we discuss some related studies on URL classification, and the application of deep learning based techniques in cybersecurity.

3.1 URL Classification

Some related works on the detection of online malicious webpages are reviewed in this section. After evaluating the features used in previous studies, we identify a collection of commonly used features which could yield high detection accuracy. Table 3.1 shows 36 features extracted from the related works that concern malicious webpages detection along with a short description.

Justin Ma *et al.* [50] conduct a URL-based approach for malicious websites detection. Similar to our study, their objective is to design and build a

Table 3.1: Lists of detection features fetched from previous works

#	Category	Feature Name	Description	Objective	Reference
F1	Textual-based Feature	Textual length of URL	The total length of the URL	Detect ads or malicious ads	[82] [?] [50] [?]
F2	URL-based Feature	URL component presence	If query component or userinfo component present	Detect ads	[82] [?]
F3	Textual-based Feature	Token occurrences	The occurrences of each word in URL	Detect ads or web content	[82] [?] [?] [?]
F4	URL-based Feature	Token occurrences by URL component	The count of token occurrences for each component	Detect ads	[82] [?]
F5	Textual-based Feature	Sequential n-gram	URL first split into tokens, then derive sequence of n letters from them	Detect ads	[?] [82] [?] [?] [?]
F6	Textual-based Feature	Full token n-gram	The count of occurrences of tokens with regard to succession relation	Detect ads	[82]
F7	URL-based Feature	Token count (total and per URL component)	Ads are likely to have many parameters in query component	Detect ads	[82]
F8	Textual-based Feature	Numeric tokens count (total and per URL component)	The count of occurrences of numeric values in URL	Detect ads	[82]
F9	Textual-based Feature	Ad-related keywords	Such as 'ad', 'advert', 'popup','banner', 'sponsor', 'iframe', 'googlead', 'adsys' and 'adser'	Detect ads or phishing URL	[9] [39] [64]
F10	Textual-based Feature	Suspicious symbol presence	If the URL contain semicolons to separate parameters? If the URL contains valid query? If any suspicious symbol just like @ present?	Detect ads or phishing URL	[9] [64]
F11	URL-based Feature	Original page related	If the base domain name is present in the query of the URL? If the requested URL is on the same domain?	Detect ads	[9]
F12	URL-based Feature	Size of Ads in URL	Indicating the size of the ads it going to display	Detect ads	[9]
F13	URL-based Feature	Dimensions of the URL	Indicating the dimensions of the screen or browser	Detect ads	[9] [?] [39]
F14	URL-based Feature	Iframe container	Indicating if the URL is requested from within an iframe either in the context of the page or in the context of nested iframes	Detect ads	[9]
F15	URL-based Feature	Proportion of external requested resources	The proportions of external iframe, script and resource requests	Detect ads	[9]
F16	Textual-based Feature	Length of hostname	The length of hostname	Detect malicious ads	[50] [?]
F17	Textual-based Feature	Dots occurrences	The number of dots in the URL	Detect malicious ads	[50] [?] [64]
F18	Host-based Feature	IP address	Is the IP address in the blacklist?	Detect malicious ads	[50] [?]
F19	Host-based Feature	WHOIS properties	What is the date of registration, update and expiration? Who is the registrar? Who is the registrant? Is the WHOIS entry blocked?	Detect malicious ads	[50] [?] [64] [48]
F20	Host-based Feature	Domain name properties	What is the TTL for DNS records associated with the hostname?	Detect malicious ads	[50] [?]
F21	Host-based Feature	Geographic properties	In which continent/country/city does the IP address belong?	Detect malicious ads	[50] [?]
F22	JavaScript Feature	JavaScript source code	Including code obfuscation, dynamic code and URL generation, code structure, function call distribution, event handling, script origin, presence of keywords	Detect ads	[?] [?]
F23	URL-based Feature	URL tree	The left-most item (http:) becomes the root node of the tree, successive tokens in the URL become the children of the previous token	Detect ads or web content	[?]
F24	Host-based Feature	Blacklist membership	Is the IP address in a blacklist?	Detect malicious URL	[?]
F25	Host-based Feature	Connection speed	What is the speed of the uplink connection?	Detect malicious URL	[?]
F26	Textual-based Feature	Presence of IP address	IP is not included in a normal URL	Detect phishing URL	[64]
F27	Textual-based Feature	Unknown noun presence	Domain names are not created by using some random letters	Detect phishing URL	[64]
F28	URL-based Feature	Misplaced top level domain	If the domain name is present in the path section?	Detect phishing URL	[64]
F29	Network-based Feature	URL redirection	If the URL has been used to redirect to many other pages?	Detect phishing URL or ads	[64] [?] [39] [48]
F30	Network-based Feature	Traffic received	The amount of web traffic that each website gets	Detect phishing URL	[64]
F31	Network-based Feature	HTML table tree	The visual/physical placement of links on a page	Detect ads or web page content	[?]
F32	Textual-based Feature	Precedence Bigram	The left-to-right precedence of two tokens in the URL	Detect ads	[?]
F33	Network-based Feature	URL redirect path	The redirection chain of a set of URLs	Detect malicious ads	[48]
F34	Network-based Feature	Domain redirect path	The redirection chain of a set of domains	Detect malicious ads	[48]
F35	Host-based Feature	Domain frequency	The number of publishers that associated with the domain each day	Detect malicious ads	[48]
F36	Host-based Feature	Domain-pair frequency	The frequency of two neighboring URLs/domains	Detect malicious ads	[48]
F37	Textual-based Feature	Dash count in hostname	The number of dash present in hostname	Detect suspicious URLs	[?]

lightweight URL classification system without webpage analysis. In their work, *DMOZ Open Directory* and *Yahoo Directory* are used for benign URL

sources, and *PhishTank* and *Spamscatter* are used for malicious URL sources. The features they extracted can be categorized as lexical-based features and host-based features. Different from our approach, most of their features are generated by the "*bag of words*" representation, and machine learning techniques such as Logistic Regression and Support Vector Machine are used in their experiment. By comparing classification results between full feature set and select feature set, they find that a large feature set significantly improves classification accuracy.

Zhou Li *et al.* [48] propose a topology study of online advertisement by analyzing ad-related Web traces and entities. Though similar in research scope (detecting malicious advertisements) to our study, they focus on the interactions and relationships between different entities in an ad-delivery chain. *EasyList*, *EasyPriacy* and *Google SafeBrowsing* are used for data labeling after crawling 90,000 URLs from Alexa top websites. According to their evaluation, malicious advertising always has a longer ad-delivery chain and a shorter existing period. Finally, a detection system is built for malicious online advertisement. With the implementation of Decision Tree algorithm, they achieve a classification accuracy of 94% with a low false positive rate.

Anh Le *et. al* [41] design a lightweight method for phishing URL detection using only lexical-based features. They collect 6000 phishing URLs from *PhishTank* and *MalwarePatrol*, and 8000 benign URLs from *Yahoo Directory* and *DMOZ Open Directory*. The experiment results show that using

lexical-based features at the client side, they can still develop a phishing detection framework with comparable classification accuracy. Their system is lightweight in terms of both computation and memory requirements. However, the error rate is the only evaluation metric used in their work, and they do not assess false positive rate and false negative rate in a reasonable depth. Our work initially builds on this paper, but the main differences from [41] is that our focus is online advertisement URLs, and we give a comprehensive lexical analysis for the URLs from different sources.

In a very similar study, Piotr L. Szczepański [82] *et al.* discuss an automated system for online advertisement detection. Several classic machine learning techniques such as Decision Tree and Random Forest are implemented in their experiment. With the application of "*n-gram*" algorithm, 330,000 lexical-based features are extracted, and then reduced to 64,000 by feature selection approaches. In our opinion, the high dimensionality of the feature set poses certain challenges for the classification task. In the evaluation, the authors emphasize that online advertisement detection is a cost-sensitive task, the misclassification of legitimate webpages cost differently than the misclassification of advertising websites. This idea is also highlighted in our paper. However, since the authors do not describe how their URL sources are selected, it is difficult to repeat their work and evaluate their performance.

In some other related works, Li Xu *et al.* [98] propose a cross-layer malicious website detection system, with the combination of network-layer and application-layer analysis; Kurt Thomas *et al.* [85] design a real-time URL

spam filtering system, using lexical-based features, IP-based features and HTML-based features; Colin Whittaker *et al.* [94] perform how to maintain *Google SafeBrowsing* blacklist automatically by examining the URL and the content of a webpage; Hyunsang Choi *et al.* [12] identify multiple types of malicious attacks by URL lexical-based analysis, and Sangho Lee *et al.* [45] detect suspicious URLs in Twitter streams. In conclusion, all previous studies explore either lexical-based features alone or a combination of lexical-based features and external features for URL classification. Based on their work, we implement the first comprehensive study of URL lexical analysis for online advertisement detection. And also, by introducing a novel deep learning based approach, our thesis presents the first study on lexical-based visualization for advertisement URLs.

3.1.1 Summary of Limits of Previous Studies

The limits of previous research are mainly discussed in this section. And also our new solutions are proposed to tackle the previous issues.

- **Issues about the dataset**: Different data sources can provide different feature distributions for distinguishing malicious and benign URLs. For example, *PhishTank* has more undefined WHOIS registration dates, more IP addresses in the hostname, and more URLs in blacklists. So certain typical features are introduced in classification if only certain data sources are chosen. In order to tackle this issue, we build a crawler

to manually select a dataset which contains over 5000 non-advertising URLs before we start our experiment.

And also the distributions of positive instances and negative instances are still important. For example, average two-third (67%) of Alexa top websites display some forms of advertising [39], and more than 1% of Alexa top 90,000 domain have been infected by malicious contents [48]. In previous papers, the distribution of the ad-related URLs and non-ad URLs does not follow the percentage above, which makes their detection system not be applicable to real-world cases. In our experiment, we select three data sources as our URLs input, and try to mimic the circumstances in the real world, more information please see Section 5.1.

- **Issues about the features:** As we mentioned in the Chapter 1, a lightweight detection system is significant for online advertisement detection task, since any delay of classification may cause unexpected results. Although certain type of features, such as WHOIS properties, can provide useful information about the URLs, they may cause the time latency issue by sending requests to remote servers. This issue become more severe when the size of dataset is large. So in our study, we try to avoid any type of features that can trigger this kind of problems. Also some reputation based features are used in the previous studies, such as *PageRanking* value and *Gmail reputation* value [94].

However, the quality and reliability of these kinds of features mainly depend on the third-party, we cannot control the data and services provided by them. That is why these types of features are not used in our experiment neither. In our study, we propose a lightweight online advertisement detection system by using lexical-based features only. More details please see Section 4.2.

3.2 A Brief Overview of Deep Learning Algorithms

A brief overview of the mainstream deep learning algorithms is introduced in this section. Three popular and well established deep neural networks are emphasized, along with their primary applications. The three deep neural networks are Convolutional Neural Networks, Recurrent Neural Networks, and Deep Belief Networks.

3.2.1 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are biologically inspired variants of Multiple Layers Perceptron (MLP) since the connectivity pattern between their neurons is inspired by the organization of the animal visual cortex. CNNs are particularly designed for multi-dimensional data, such as images and videos [5], and they can process data that come in the form of multiple

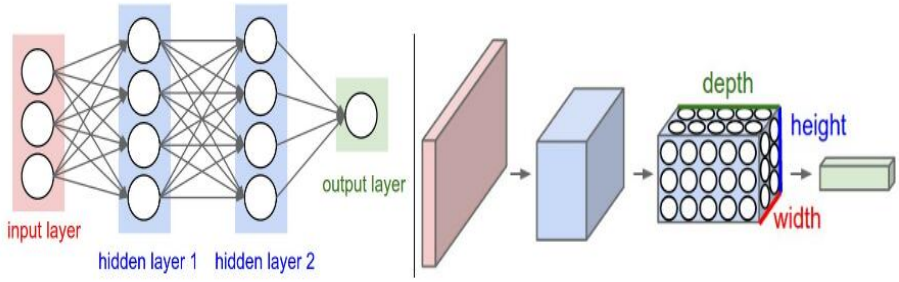

Figure 3.1: Comparison of the architecture between a regular 3-layer Neural Network and a typical CNN

arrays [43]. A typical CNN architecture is designed with the following key points: local connections, shared weights, pooling and the use of many layers [43] [42]. Two types of layers are need for structuring the architecture of CNNs, convolutional layers and pooling layers [59]. The role of the convolutional layer is to detect local conjunctions of features from the previous layer, and the role of the pooling layer is to merge semantically similar features into one [43].

Unlike a regular Neural Network, the layers of a CNN have neurons which can be arranged in 3 dimensions. Every layer of a CNN transforms the 3-dimension input volume to a 3-dimension output volume of neuron activations. This provides the natural advantage of CNNs over other Neural Networks in the application of image recognition (see in Figure 3.1 [16]). Also CNNs have been successfully applied to many other recognition tasks, such as digit recognition [76], object recognition [44] and natural language processing [14].

3.2.2 Recurrent Neural Networks

RNNs are a family of artificial neural networks that their neurons can form fully connected or partially connected cycles [97] [54]. RNNs can process an input sequence one element at a time, maintaining in their hidden units a 'state vector' that implicitly contains information about the historical record of all the past elements of the sequence [43]. The computational neurons can get inputs from other neurons at previous time steps, and then map an input sequence into an output sequence (see in Figure 3.2 [43]), so RNNs can create an internal state of the network which allows it to exhibit dynamic temporal behavior. This training architecture make RNN a powerful application for sequential data, such as handwriting [29], speech [72] and language [43].
Over the past few years, RNNs have shown their big potentials for Natural Language Processing (NLP). And many RNNs architectures have emerged for grammatical inference [13] [31], natural language understanding and translation[36] [79], and a variety of other fields such as phonology [32], morphology [69] and role assignment [36]. Also as the improvement of learning strategies, RNNs have been highlighted as better solutions for whole document understanding [43].

3.2.3 Deep Belief Networks

DBN can be viewed as a composition of simple, unsupervised networks such as Restricted Boltzmann Machines (RBMs) [33] or AutoEncoders [8] (see in

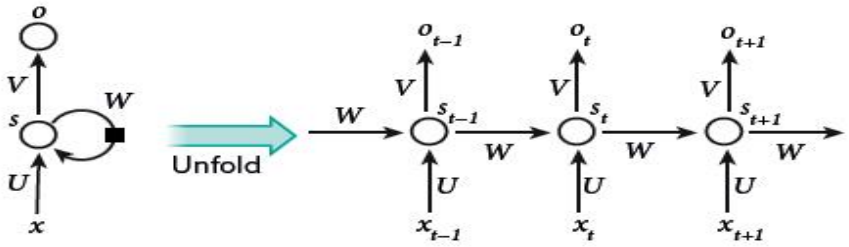

Figure 3.2: An example of a typical RNN with its unfolding explanations in time

Figure 3.3 [55]), where each subnetwork's hidden layer serves as the visible layer for the next. An RBM has a bottom layer of "visible" units, and a top layer of "hidden" units, which are fully and bidirectionally connected with symmetric weights [15]. These networks are "restricted" to a single visible layer and single hidden layer, where connections are formed between the layers, but not between units within each layer [5] [33].

The generative properties make DBN an impressive application on a board range of classification tasks. Performance outputs demonstrate that DBN is a good solution for handwritten character recognition with the introduction of feedforward networks [5]. Similar results are presented in [8] to solidify that the use of DBN on unsupervised tasks is as well as continuous valued inputs. In addition, recently DBNs show their applicable potentials in various problems such as image classification [77], speech recognition [71], audio classification [55], sentiment analysis [102] and language understanding problems [73].

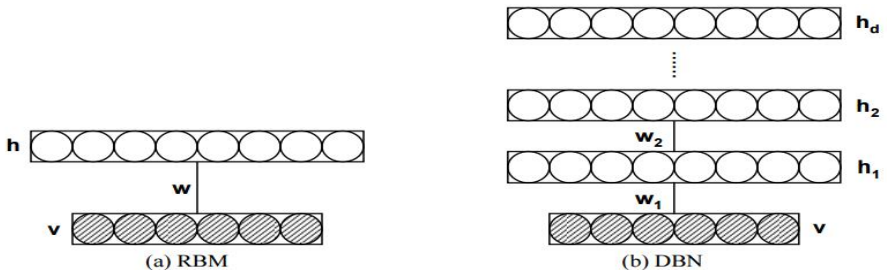

Figure 3.3: The architecture of RBM (a) and DBN (b). The DBN is composited by several RBMs

3.3 Deep Learning in Cybersecurity

Different with conventional machine learning techniques, deep learning can build computational models with multiple processing layers [43]. With the multiple levels of layers, deep learning algorithms can be fed with raw data to discover the representations of the input object. Intricate functions can be learned by using the backpropagation algorithm through the multi-level structure. Nowadays deep learning algorithms have been widely used in object recognition and detection tasks. CNNs and DBNs have shown their potentials in processing images, videos, speech and audio, whereas RNNs are emphasized on sequential data such as language and speech understanding. As the attention for security and privacy issues grows dramatically, deep learning algorithms have also brought breakthroughs in the area of cybersecurity. Many studies are working on automatic malware detection and signature generation using deep learning.

Yuancheng Li *et al.* [47] propose a hybrid malicious code detection scheme,

using AutoEncoder for dimension reduction and DBN for classification; Omid E. David *et al.* [18] design a deep learning based method for automatic malware signature generation. With the implementation of DBN, they can generate invariant compact representations for malware behaviors; [62] [21] [86] [92] and [53] focus on deep learning based malware detection and classification.

Different from previous studies, our study is mainly working on online advertising URL detection using deep learning. Two types of deep learning models are selected in our experiment for the classification task, RNN and Feedforward deep neural network (DNN). Over the past few years, RNN has shown a great performance in the area of NLP, such as grammatical inference [13], natural language understanding and translation[36] [79], and document understanding [43]. DNN is also widely used in various classification tasks, such as speech recognition [20] and NLP [14]. Lexical-based URL understanding has much in common with NLP because the analysis of the divided URL can be considered as a form of sequential text modeling. We believe that an RNN or DNN based architecture can be an effective method for lexical-based URL classification.

AutoEncoder is an unsupervised neural network which was proposed in [34]. With the introduction of the "*bottleneck*" hidden layer, AtutoEncoder is capable of creating sparse representation of the input data and can, therefore, be used for dimension reduction. After classification, we design a hybrid deep learning based method for unsupervised feature visualization by combining

AutoEncoder with *t-SNE* algorithm. More details please see Section 5.

Finally, we implement an RNN based framework called sequence-2-sequence model for fake URL generation. This model is widely used in the areas of language translation, lyrics generation, text summarization and conversational response. In our experiment, we can generate distinguishing fake URLs for different URL families. More details please see Section 5.

Chapter 4

The Proposed Framework

In this chapter, we give a brief overview of the proposed model used in the experiment, then we describe how we select and extract lexical-based features for URL sources, followed by the discussion of URL classification, finally, we present a novel hybrid method for unsupervised data visualization and fake URL generation.

4.1 The Proposed Model

Figure 4.1 illustrates the proposed model used in the experiment. Our model contains four principle sections: Feature Extraction, Anatomy study, Classification and Visualization. At first, we select Malicious-ad ULRs, Benign-ad URLs and Non-ad URLs as data sources for URL analysis. 141 lexical-based features are extracted to represent each input URL.

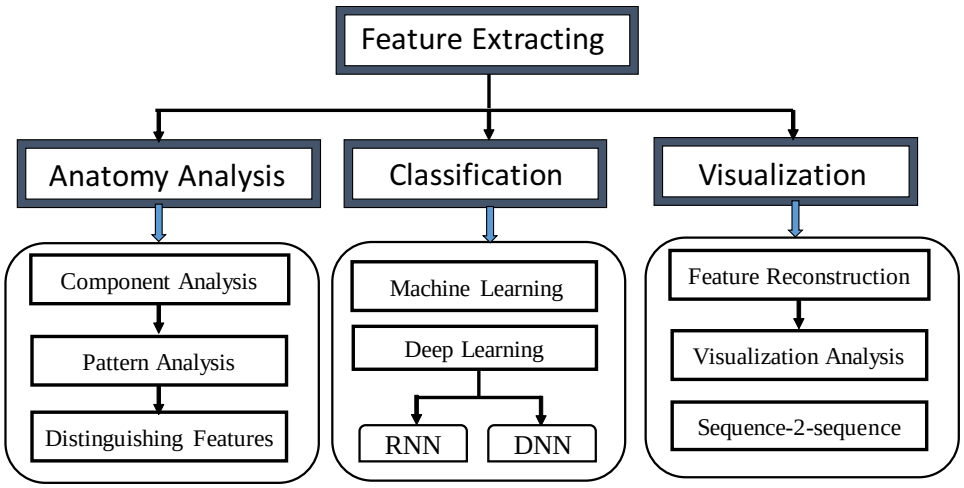

Figure 4.1: Illustration of the proposed approaches used in the experiment

In anatomy study section, we conduct a comprehensive lexical analysis for different URL sources, including URL component analysis and URL pattern analysis. By analyzing the length distribution, symbols, ad-related keywords and structural patterns in different URLs, we find that lexical-based features can provide useful information for online advertising URL detection. And also we find the most influential features for distinguishing different URL families.

In classification section, we implement deep learning based approaches and conventional machine learning algorithms for advertisement URL identification. We optimize the neural networks by selecting the best architectures (e.g., the number of hidden layers and the number of neurons in each hidden layer) and activation functions. After that, we evaluate the performance of different algorithms depending on precision, recall, accuracy, false positive

rate and false negative rate.

In the final section, we design and develop a new method for URL visualization. AutoEncoder, a deep learning based dimensionality reduction algorithm is applied first for feature preprocessing, then *t-SNE* is implemented for visualizing different URL families. This method is totally unsupervised and can be used to evaluate the quality of the extracted features. After that, we implement an RNN based framework called sequence-2-sequence model for fake URL generation. This model can remember the patterns of different data sources and create fake URLs with distinguishing characteristics.

4.2 Features

Lexical-based features have been used in URL classification and are good candidates for building a lightweight detection system. In this thesis, we extract 141 lexical-based features for URL identification, then give a comprehensive anatomy analysis for online advertisement URLs.

The structure of a URL example can be expressed by the following:

`http://www.url.com:80/path/to/my/file/filename.html?query#reference`

Generally a URL contains the following main components: Protocol (*http://*), domain name (*www.url.com*), port number (*80*), path (*/path/to/my/file/*), filename (*filename.html*), query (*?query*) and fragment or known as reference (*#reference*). Not all of the components are mandatory, but all of them should be organized and positioned in the right way in order to build

Table 4.1: The final lexical-based feature set used in the experiment

	URL	Host	Port	Path	Filename	Query	Reference	Size
Component present?		✓	✓	✓	✓	✓	✓	6
Numeric token present?	✓	✓		✓	✓	✓	✓	6
Ad-related keyword present?	✓	✓		✓	✓	✓	✓	6
Malicious ad-related keyword?	✓	✓		✓	✓	✓	✓	6
Symbol present?	✓	✓		✓	✓	✓	✓	6
Executable file present?	✓							1
IP address present?	✓							1
Length	✓	✓		✓	✓	✓	✓	6
Length of longest token	✓	✓		✓	✓	✓	✓	6
Length of average token	✓	✓		✓	✓	✓	✓	6
Length ratio: domain/URL	*	*						1
Length ratio: path/URL	*			*				1
Length ratio: filename/URL	*				*			1
Length ratio: query/URL	*					*		1
Length ratio: path/domain		*		*				1
Length ratio: filename/domain		*			*			1
Length ratio: query/domain		*				*		1
Length ratio: ref./path				*			*	1
Length ratio: ref./query						*	*	1
Max token length	✓							3
Dot count	✓	✓		✓	✓	✓	✓	6
Dash count	✓	✓		✓	✓	✓	✓	6
Delimiter count	✓	✓		✓	✓	✓	✓	6
Letter count	✓	✓		✓	✓	✓	✓	6
Digit count	✓	✓		✓	✓	✓	✓	6
Symbol count	✓	✓		✓	✓	✓	✓	6
Digit-letter-digit	✓	✓		✓	✓	✓	✓	6
Letter-digit-letter	✓	✓		✓	✓	✓	✓	6
Number rate	✓	✓		✓	✓	✓	✓	6
Character continuity rate	✓							1
2-gram of malicious-ad	✓	✓		✓	✓	✓	✓	6
2-gram of benign-ad	✓	✓		✓	✓	✓	✓	6
3-gram of malicious-ad	✓	✓		✓	✓	✓	✓	6
3-gram of benign-ad	✓	✓		✓	✓	✓	✓	6
Total								141

a working URL.

Lexical-based features can provide sufficient information about the URL [41] and are the most readily available features. The URL of a suspicious webpage may look different than normal URLs. For example, malicious webpages usually use IP address as their domain name [49] [50], the abnormal length ratio of different URL components may help to find remarkable URLs [52], and ad-related keywords are also good indicators for identifying advertisement

websites [9] [64].

Based on available features mentioned in previous studies, we propose an integrated feature set for URL lexical analysis. We split each single URL into 6 main components (*domain name*, *port number*, *path*, *filename*, *query* and *reference*), all the lexical-based features are extracted from these components in order to construct the final feature set. Table 4.1 shows the lexical-based features we extracted and used in our experiment, the dimensions of each lexical-based feature and the final feature set. All the features can be categorized into the following groups:

- **Presence**: In this group of features, we check if all the components are present in the URL; if there is numeric token present; if there is ad-related keyword (*ad, advert, banner, popup, sponsor, iframe, adsys, adser*) or malicious keyword (*account, accounting, bank, banking, login, signin, websrc, sec*) present; if *exe.* or IP address is present in the URL.

- **Length based feature**: This group of features includes the length of the whole URL and each component; length ratio of different components; the maximum letter length, the maximum digit length, and the maximum symbol length.

- **Count based feature**: We record the number of dots, dashes, delimiters, letters, digits and symbols in the whole URL and in each component.

- **Pattern based feature**: This group of features describe specific lexical-based patterns in the URLs.
  - **Letter-digit-letter**: If there is a digit between two letters in the given URL.
  - **Digit-letter-digit**: If there is a letter between two digits in the given URL.
  - **Continuity rate**: We categorize the character type in the URL as letter, digit and symbol. Then we record the longest continuous length for each type. For example, the continuity rate for URL "*go0gle12*@*" is (3 + 2 + 2) / 10 = 0.7.
  - **Number rate**: The percentage of the number of digits in the given URL.
- **n-gram feature**: We also record the "*n-gram*" information for both malicious advertising URLs and benign advertising URLs. We extract 2-gram and 3-gram tokens from the above data sources to construct the "*bag of words*". Then we split each input URL into 2 or 3-token sequences, and check if the target token is in the bag.

4.3 Classification and Visualization

In this section, we outline how URL classification and visualization are conducted in our experiment.

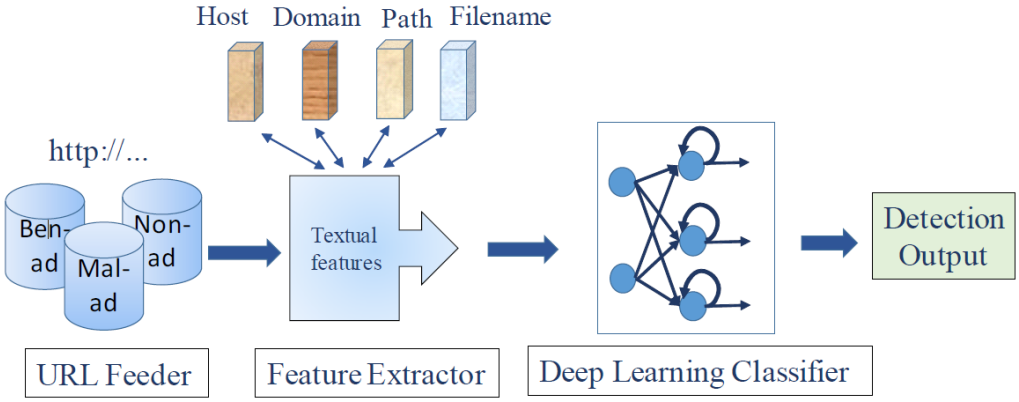

Figure 4.2: The proposed flowchart for URL classification

- **Model 1**: URL Classification

  The first model is built for URL classification task. The overall design for model sees in Figure 4.2. Three major components are included in model 1, URL Feeder, Feature Extractor, and Deep Learning Classifier.

  - **URL Feeder**: In URL Feeder, malicious-ad URLs, benign-ad URLs, and non-ad URLs are being fed into the system.
  - **Feature Extractor**: A comprehensive lexical feature set mention in Section 4.2 is extracted in Feature Extractor, and used in our experiment.

    Depending on the existing detection system, lexical-based features are good indicators for representing online advertisement URLs. So in our proposed model, we analyze the lexical characteristics of each component in the URLs, and then combine them as a

comprehensive lexical feature set. This feature set contains both real number and binary number, and will be the data feed of our deep learning system.

  - **Deep Learning Classifier**: Deep learning based architecture is used as the classifier. Based on the discussion in Section 3.2, DNN and RNN have proven very successful in processing sequential data. Based on the analysis of textual-based features, DNN and RNN have their natural advantages over other deep neural networks, and are appropriate solutions for our detection system.

- **Model 2**: URL Visualization

  The second model is built for URL visualization. The overall design for model sees in Figure 4.3. Three major components are included in model 2, URL Feeder, Feature Extractor, and Feature Preprocessing.

  - **URL Feeder**: In URL Feeder, malicious-ad URLs, benign-ad URLs, and non-ad URLs are being fed into the system.
  - **Feature Extractor**: The comprehensive lexical feature set mentioned in Model 1 will also be used in Model 2.
  - **Feature Preprocessing**: In this thesis, we design and develop a novel method for lexical-based URL visualization. AutoEncoder is applied for feature preprocessing, and then a dimensionality reduction algorithm called *t-SNE* is used to visualize different URL

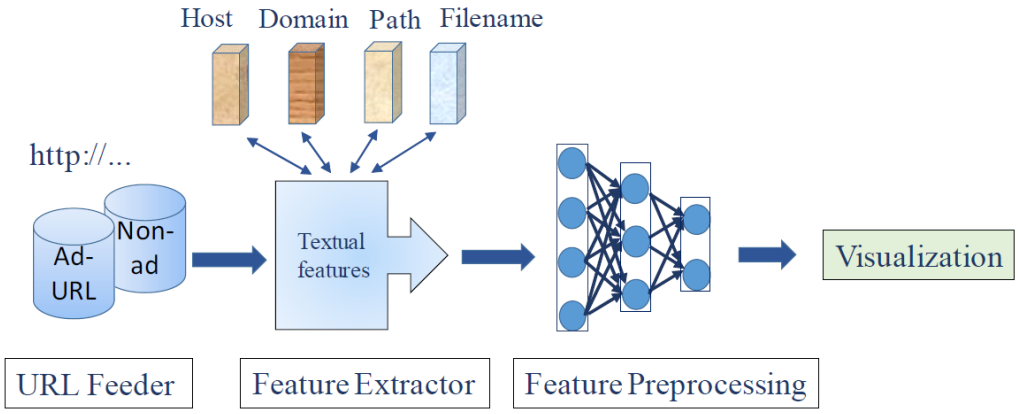

Figure 4.3: The proposed flowchart for URL visualization

sources. Also some other popular dimensionality reduction algorithms such as *Principle component analysis* (PCA), *Local linear embedding* (LLE), *Isomap* and *Sammon mapping* are also introduced and applied for comparison in evaluation. More details please see Section 5.

Chapter 5

Experiment and Evaluation

5.1 Experiment

5.1.1 Dataset

Three types of URL sources are collected in order to conduct the classification task.

Source I (Malicious-ad URLs): We obtained examples of malicious advertising URLs from a URL blacklist service website, *www.urlblacklist.com*. The data sources are all about malicious advert servers and banned URLs, the latest modification date of the dataset is July 28th, 2016.

Source II (Non-ad URLs): We built a crawler to fetch all the *non-advertising URLs* from Alexa Top 5000. For each Alexa domain (such as *google.com*), the crawler visits at most 20 pages which are originally linked with Alexa top page, from August 28th, 2016 to September 10th, 2016. Af-

ter that, 5000 *non-advertising URLs* are selected manually from the fetched URLs mentioned above.

Source III (Benign-ad URLs): We collected the benign advertising URLs from an advertising dataset, *www.code.google.com/archive/p/open-advertising-dataset/*. This website is created by the University College London as an independent computational advertising dataset from the publicly available sources.

Figure 5.1 shows the snapshot of 10 examples for each URL source. With the data sources mentioned above, we build our URL detection system by creating four scenarios. Each scenario contains two URL data sources for binary classification. Table 5.1 shows four scenarios based on these three types of URL sources.

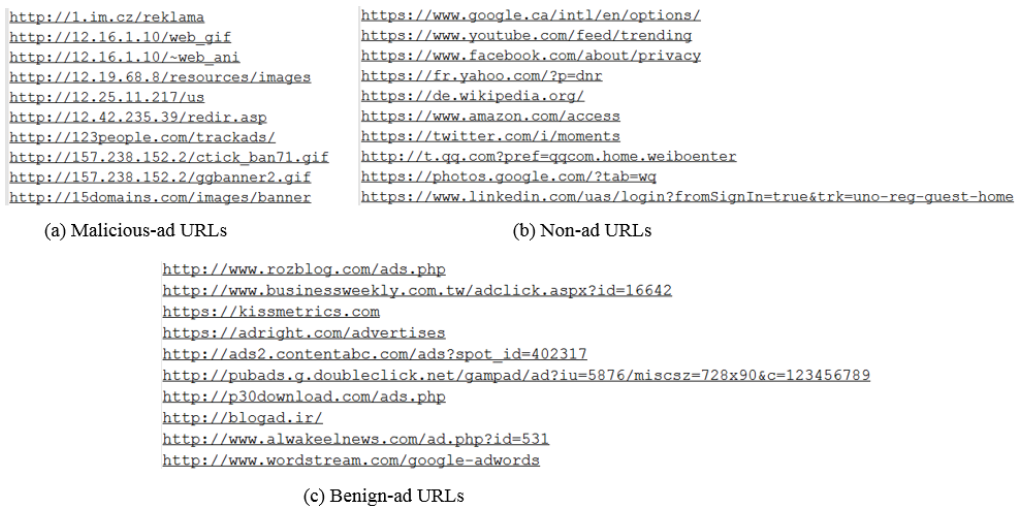

Figure 5.1: The snapshot of three different URL sources

Table 5.1: Summary of the dataset used in the experiment

Scen.	Data Source
A	Ads (4115) *vs* Non Ads (5000)
B	Benign Ads (3000) *vs* Malicious Ads (1115)
C	Benign Ads (3000) *vs* Non Ads (5000)
D	Malicious Ads (1115) *vs* Non Ads (5000)

5.1.2 Classification with Deep Learning

5.1.2.1 Algorithm Description

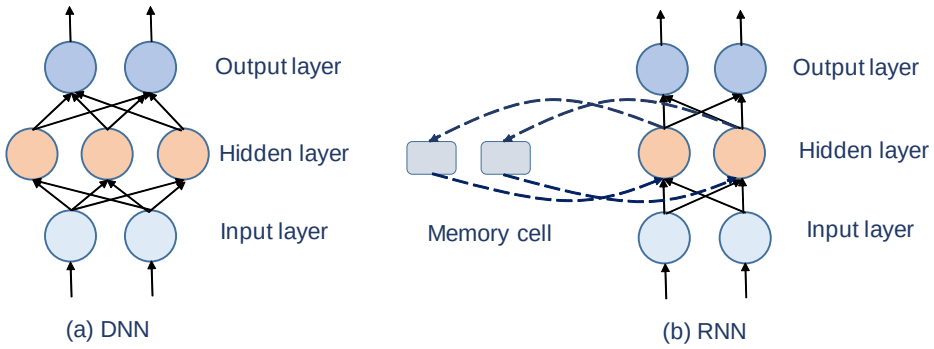

Figure 5.2: The architecture of DNN (a) and RNN (b)

In DNN, all the information flows in one direction, from the input layer to the output layer (see Figure 5.2 (a)). Different from DNN, RNN can process the input sequence one element at a time, and store the information of the input element into a memory cell (see Figure 5.2 (b)). Then for the next time step, RNN gets inputs from both the stored information in the memory cell and the next sequence element. This means RNN can remember the information of past elements of the sequence. However, this memory is short

because backpropagated gradients either grow or shrink tremendously after many time steps [43], typically making the training weights either explode or vanish.

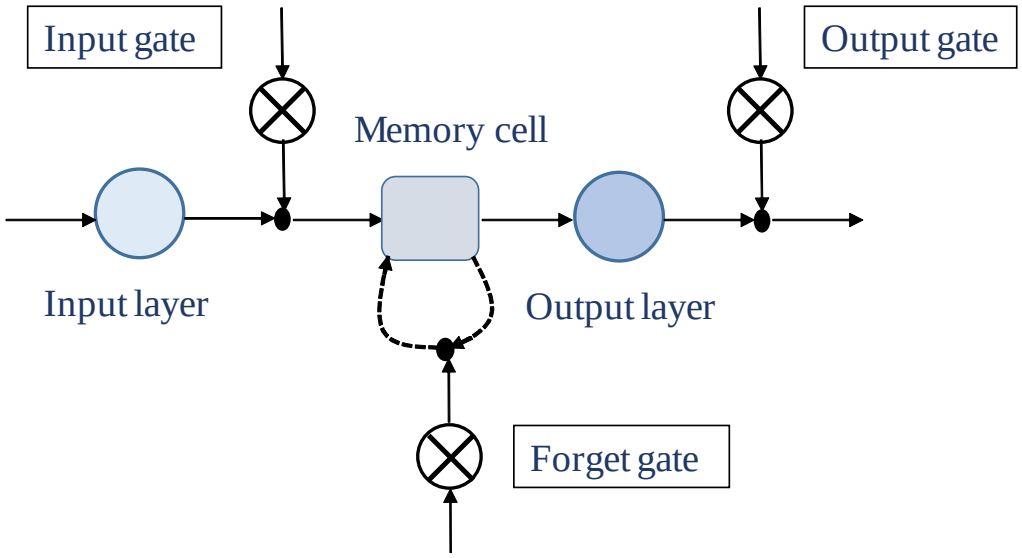

Figure 5.3: The architecture of a memory block in LSTM

In order to avoid the long-term dependency problem, another RNN called Long Short-Term Memory (LSTM) RNN was proposed in [35]. Three "gates" are installed in the memory block in order to control the flow of information into or out of the memory cell (see in Figure 5.3). The outputs of the three gates range from 0 to 1. 0 means the gate is totally closed, and 1 means the gate is completely open. Especially, an input gate controls the extent to which a new sequential element flows into the memory cell. Similarly, an output gate controls how much information in the memory cell can be used to calculate the output activation of the block. A forget gate controls how

Table 5.2: Formulas used to calculate the number of neurons in the hidden layer

Formula	Expressioin	Reference
F1	$N = C \times \sqrt{N_t}$	[46]
F2	$N = C \times log_2 N_t$	[90]
F3	$N = C \times \sqrt{N_i \times N_o}$	[38]
F4	$N = C \times \frac{N_t}{\sqrt{N_i \times log_2 N_t}}$	[99]
F5	$N = \frac{N_i + \sqrt{N_t}}{L}$	[75]
F6	$N = \frac{N_t}{C \times (N_i + N_o)}$	[88]

much information can still be kept in the memory cell in each time step.

5.1.2.2 Optimizing the Architecture for Neural Network

In our classification task, the lexical-based features mentioned in Section 4.2 is used as the inputs for our neural network. Every URL instance is represented as a 141-dimensional vector plus 1 label, so the size of the input layer is 141. Noted previously in Section 5.1.1, all the scenarios in our experiment are binary-classification, so the size of the output layer is 2.

In order to optimize the DNN's architecture, we test the following numbers as the number of hidden layers in the network: {*1, 2, 3, 4*}. And in each model, formulas {*F1, F2, F3, F4, F5, F6*} in Table 5.2 are used to calculate the number of neurons in each hidden layer.

In Table 5.2, N is the number of hidden nodes; N_t is the number of training samples; N_i and N_o are the number of nodes in the input layer and output layer respectively; L is the number of the hidden layers in the neural network, and C is the scaling coefficient usually from 2 to 10.

For DNN, $\{F3\}$ with $\{3\}$ hidden layers are selected and used in building our deep network model, because this architecture can achieve a high precision/recall and a lower false negative rate. Comparison results of different architectures will be discussed in Section 5.2. The final architecture is 141 – 90 – 60 – 16 – 2, and this architecture is also used in training AutoEncoder for data processing.

Rectified linear units (ReLU) is extensively applied when training deep neural networks since it can accelerate the convergence process and diminish the gradient vanishing issue [27]. In this section, ReLU is used as the activation function of the input layer and hidden layers for DNN, and can be expressed as: $f(x) = max(x, 0)$, where x is the input of the neuron. By returning the maximum value between x and 0, ReLU can remove the negative parts of the input information.

Computational complexity is a big issue for RNN, because a fully connected RNN with a multiple-layer architecture would be very expensive in training[80] [61] [57]. In order to avoid the large and intractable time complexity problem, a simple architecture proposed in [3] is used in our experiment. The final RNN contains one LSTM hidden layer with 128 neurons, and the architecture is 141 – 128 – 2.

5.1.2.3 Optimizing the Activation Function for Neural Network

In the deep neural network, the activation function defines the output of a node given a set of inputs. Mathematically speaking, the activation function

calculates a "weighted sum" of the input values, adds a bias and then decides whether the result should be active or to what extent the result should be active. The activation function is really important for training deep neural network, because it introduces the nonlinear mappings between input data and output data, and it also defines how the neural network learns and represents the input signals.

There is a number of common activation functions in training deep neural network. In this section, we choose 6 of them and evaluate the performance of different activation functions in the deep neural network for URL classification. The activation functions we used in our experiment are: *ReLU, ELU , Softplus, Softsign, Sigmoid,* and *Tanh.*

Table 5.3: Activation functions evaluated in our experiment [95]

Name	**Equation**	**Range**	**Plot**
ReLU	$f(x) = max(x, 0)$	(0, ∞)	
Softplus	$f(x) = \ln(1 + e^x)$	(0, ∞)	
Softsign	$f(x) = \frac{x}{1+\|x\|}$	(-1, 1)	
Sigmoid	$f(x) = \frac{1}{1+e^{-x}}$	(0, 1)	
Tanh	$f(x) = \frac{2}{1+e^{-2x}} - 1$	(-1, 1)	
ELU	$f(x) = \begin{cases} \alpha(e^x - 1) & x < 0 \\ x & x \geq 0 \end{cases}$	(-α, ∞)	

Table 5.3 shows the activation functions evaluated in our experiment, including the name, the equation expression, range and the plot of each activation function.

5.1.2.4 Training the Neural Network

In this section, we discuss the activation function, loss function and the algorithm we used for training the neural network.

Rectified linear units (ReLU) is extensively applied when training deep neural networks since it can accelerate the convergence process and diminish the gradient vanishing issue [27]. In our experiment, ReLU is used as the activation function of the input layer and hidden layers for DNN, and can be expressed as: $f(x) = max(x, 0)$, where x is the input of the neuron. By returning the maximum value between x and 0, ReLU can remove the negative parts of the input information.

Sigmoid function is commonly used in training LSTM RNN [11], since it can control how much information flow into or out of the LSTM cell by outputing a value from 0 to 1. In our experiment, *Sigmoid* function is used as the activation function for RNN and can be expressed as: $f(x) = \frac{1}{1+e^{-x}}$, where x is the input data of the neural network. *Softmax* function is used in the final output layer as activation function for DNN and RNN. For a deep neural network with K neurons in the output layer, x is the input of the neuron. The classification result of each neuron j can be expressed by *Softmax* function as follows:

$$f(x)_j = \frac{e_j^x}{\sum_{k=1}^{K} e_k^x}$$

Softmax function is a generalization of logistic function, that can squash the K-dimensional vector in the output layer to a K-dimensional vector in the

range of 0 to 1 [4]. It is the most commonly used activation function for output layer in training deep neural network, since the results of softmax function can represent the probability distribution of K different neurons (in our case K = 2).

In the training process, our objective is to minimize the *cross entropy* loss function, which can be expressed as follows:

$$E(Y, Y') = -\frac{1}{n}\sum_{1}^{n}[y_i \times ln(y_i^{'}) + (1 - y_i) \times ln(1 - y_i^{'})]$$

where Y = y_1, y_2, ... y_n is the set of labels for corresponding input instance; $Y^{'} = y_1^{'}, y_2^{'}, ... y_n^{'}$ represents the set of output results for each input instance calculated by *Softmax* function. Each y_n is either 0 or 1 indicating that the URL belongs to one classification or the other.

Gradient descent is one of the most common algorithms to optimize deep neural network. In our experiment online *AdaGrad* is used to minimize our loss function. *AdaGrad* can greatly improve the robustness of *stochastic gradient descent* (SGD) by adapting different learning rates to different parameters, and it is well-suited for dealing with sparse data [23] [19]. *AdaGrad* can be expressed as follows:

$$\theta_{t+1,i} = \theta_{t,i} - \frac{\eta}{\sqrt{\sum_{t'=1}^{t} \bigtriangledown E(\theta_i)^2 + \varepsilon}} \times \bigtriangledown E(\theta_i)$$

where $\theta_{t,i}$ is the i *th* parameter at time t; η is the initial learning rate; ε is a smoothing term that avoid division by 0, and $\bigtriangledown E(\theta_i)$ is the gradient of the objective function E (calculated by *cross entropy* function) at parameter θ_i.

5.1.3 Visualization with Deep Learning

Visualizing high dimensional data is significant for understanding the complex relationships between different variables. By using dimensionality reduction algorithms, we can preserve as much of the significant information of the high-level data as possible in a more compact pattern. Also, visualization facilitates the classification task, because after converting the high dimensional dataset into two or three dimensions, we can achieve a better understanding of the dataset and evaluate the quality of the extracted features.

Traditional dimensionality reduction algorithms include PCA, LLE, *Isomap*, *Sammon mapping* and *t-SNE*.

PCA is one of the most popular linear dimensionality reduction techniques. It can convert the original data into its principal directions with the maximal variance, which is also called principal component. This transformation is defined in such a way that the number of principal components is usually less than the number of original data in order to reduce dimensionality [89] [91].

LLE is an unsupervised learning algorithm that computes low-dimensional, neighborhood-preserving embeddings of high-dimensional inputs. LLE can map the inputs into a single global coordinate space with lower dimensionality, and its optimizations do not involve local minima. LLE is also able to learn the global structure of nonlinear manifolds, such as those generated by images of faces or documents of text [68].

Isomap is one of the popular nonlinear low-dimensional embedding methods.

And it can be used to compute a quasi-isometric, low-dimensional embedding from a set of a high-dimensional data set. *Isomap* provides an easy way for calculating the intrinsic geometry of a data manifold based on the geodesic distances between all pairs of points on the manifold [84] [1].

Sammon mapping is an algorithm that maps spaces between high-dimensionality and low-dimensionality by trying to preserve the structure of inter-point distances in the lower-dimension projection. More specifically, *Sammon mapping* is designed to minimize the differences between corresponding inter-point distances in the two spaces. Unlike PCA, *Sammon mapping* simply provides a way to construct a new lower-dimensional dataset, which has the structure as similar to the original dataset as possible [17] [87].

t-SNE is a nonlinear dimensionality reduction technique that visualizes high-dimensional data by giving each data point a location in a two or three-dimensional map. Specifically, it models each high-dimensional object by a 2 or 3-dimensional point in such a way that similar objects are modeled by nearby points and dissimilar objects are modeled by distant points. And *t-SNE* is better than existing techniques for creating a single map that reveals structure at many different scales [51] [40].

In our experiment, a novel visualization approach is designed based on the deep learning algorithm. AutoEncoder is implemented for data preprocessing, then *t-SNE* is applied for final visualization. Other representative dimensionality algorithms mentioned above are also introduced and compared in the evaluation section.

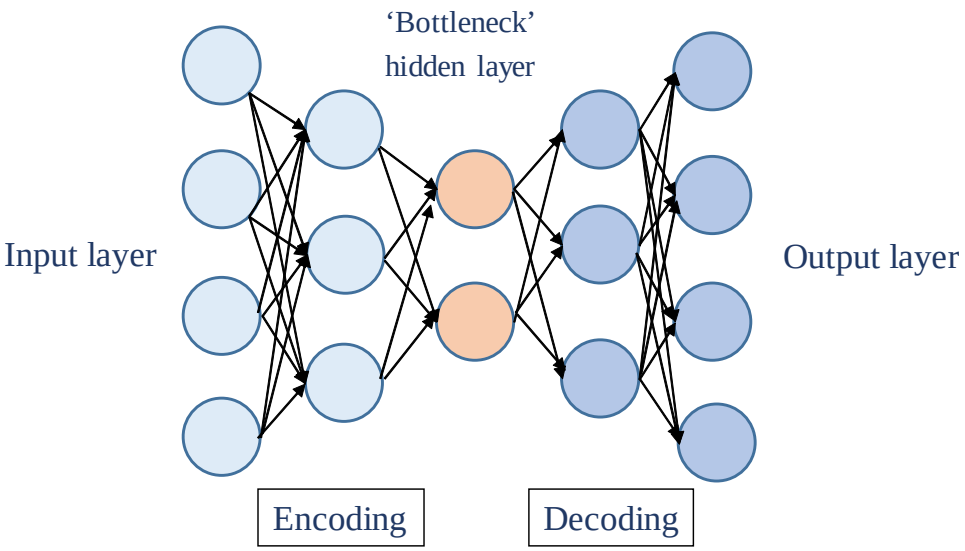

Figure 5.4: The architecture of AutoEncoder

Figure 5.4 illustrates the architecture of a typical AutoEncoder. In AutoEncoder, the number of neurons in the input layer and the output layer are the same. Instead of predicting the target value given by the input data, the purpose of AutoEncoder is to reconstruct the input data in an unsupervised way. AutoEncoder usually consists of an encoding process and a decoding process, and the "*bottleneck*" hidden layer in the middle of the network is always extracted to represent the input data in a more compressed format. In our experiment, the AutoEncoder is developed with the following architecture: 141 – 90 – 60 – 16 – 60 – 90 – 141. The 'bottleneck' hidden layer with 16 neurons is extracted from the encoding process for further visualization. *Sigmoid* function is the most commonly used activation function for AutoEncoder [30] [37], and is used in our experiment for training AutoEncoder.

In our experiment, *Sigmoid* function is used for both encoding process and decoding process.

Sum of squared errors (SSE) function is used in building AutoEncoder as loss function, and can be expressed as follows:

$$E(X, X^{'}) = \frac{1}{2} \times \sum_{1}^{n} (x - x^{'})^2$$

where X = x_1, x_2, ... x_n is the set of input data; $X^{'} = x_1^{'}$, $x_2^{'}$, ... $x_n^{'}$ represents the set of output data of the decoding process in AutoEncoder. Similar to RNN and DNN, *AdaGrad* is used to minimize the loss function.

Table 5.4: Two groups of models in our experiment for visualization

Baseline group (141 – 2)	**AutoEncoder based group (141 – 16) + (16 –2)**
PCA	AutoEncoder + PCA
LLE	AutoEncoder + LLE
Sammon mapping	AutoEncoder + *Sammon*
Isomap	AutoEncoder + *Isomap*
t-SNE	AutoEncoder + *t-SNE*

Table 5.4 shows two groups of visualization models established in our experiment. The first group is a baseline group, the five dimensionality reduction algorithms (PCA, LLE, *Sammon mapping, Isomap, t-SNE*) are applied. The size of data input is 141, and the size of the data output is 2 for visualization. The second group is the AutoEncoder based group. First, we implement AutoEncoder to compress the original data from dimension 141 to 16, then we explore the dimensionality reduction algorithms on top of AutoEncoder for final visualization. The dimension of input data is 16 and the dimension of

output data is 2.

5.1.4 Sequence-2-sequence Model for URL Generation

Sequence-2-sequence deep learning framework is proposed in [81] for sequential data modeling.This framework is based on RNNs and has great potential in the area of Natural Language Processing (NLP) such as language translation [28], lyrics generation [10], text summarization [26], conversational response [65] [25] and so on. In this section, we discuss briefly how sequence-2-sequence model works, and give the objectives of why this model is used in our thesis for fake URL generation. Finally, we describe how the sequence-2-sequence model is trained for URL generation.

5.1.4.1 Introduction to Sequence-2-sequence Model

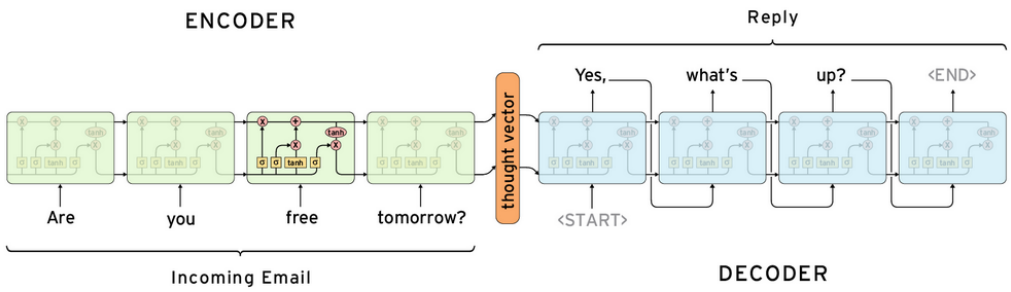

Figure 5.5: The architecture of a simple sequence-2-sequence model [25]

A sequence-2-sequence model contains two major components: an encoder and a decoder. The encoder can accept a sequence of data input one by one and then convert the input data into a fixed sized feature vector. This final

vector is comprised of all the sequential information from the input data. The decoder can decode the target data from the final feature vector for training purpose.

Figure 5.5 shows the architecture of a simple sequence-2-sequence model for automating email reply. In this example, the input sequence is "*Are you free tomorrow*?", and the corresponding output sequence is "Yes, what is up?". The RNN cells in the encoder first accept the input words one by one, and then convert the sequential information into a fixed sized vector. In the decoding process, at the first time step, word "*Yes*" is fed into the decoder as a "<START>". The output of "*Yes*" is then fed back and becomes the input of the decoder at the second time step (as seen in Figure 5.5), and actually this is the reason why sequence-2-sequence model can be used for predicting next word/character. The decoder repeats this process until all words in the target data have been processed. After training all the input instances, the sequence-2-sequence model store the entire information for testing purpose.

5.1.4.2 The Objectives of Building Sequence-2-sequence Model

In our experiment, we implement the sequence-2-sequence model for fake URL simulation. There are two objectives in this section. First, we want to evaluate the feasibility of implementing deep learning based model for online advertisement URL generation. Sequence-2-sequence model is powerful for modeling sequential data. And in this thesis, we describe how to use this model for fake URL generation. The URLs generated by our proposed ap-

proach can be used to test the vulnerability of existing advertising blocking system. In the future, sequence-2-sequence model can also be used in other similar cybersecurity domains such as phishing email generation. Second, the fake URL generation and simulation can be considered as the final step for online advertisement URL visualization. In this thesis, we conduct a comprehensive analysis for online advertisement detection and visualization using deep learning. We collect lexical-based features for advertising URL representation, and exploit a statistical learning on the distinguishing features. Finally, based on our three data sources, we generate fake non-ad URLs, fake benign-ad URLs and fake malicious-ad URLs. The fake malicious-ad URLs is distinct in appearance from the other types of fake URLs. The results and discussions are shown in Section 5.2. So in this thesis, we design and build a deep learning based model for online advertisement URL detection and visualization both lexically and visually.

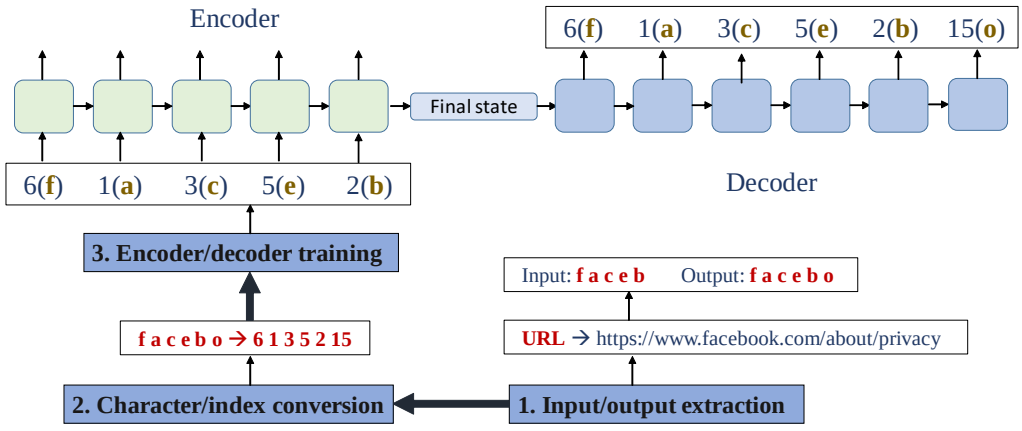

Figure 5.6: Training the sequence-2-sequence model

5.1.4.3 Training the Sequence-2-sequence Model

In this section, we discuss how sequence-2-sequence model is trained for fake URL generation. The training process contains three major components, input/output extraction, character/index conversion and encoder/decoder training. Figure 5.6 shows the overall illustration of training the sequence-2-sequence model.

We take the following URL: `https://www.facebook.com/about/privacy` as an example. First of all, we need to extract the input sequence and output sequence from the original data set for fake URL generation. Ignoring the protocol "*https://*" part and the "*www.*" part, the first extraction begins at character $\{f\}$. Then we define the number of characters in the input sequence is 5, and the number of characters in the output sequence is 6. This means the first 5 characters $\{f\ a\ c\ e\ b\}$ is extracted as the input sequence, and the first 6 $\{f\ a\ c\ e\ b\ o\}$ characters is extracted as the corresponding output sequence. By doing this, our sequence-2-sequence model can gradually learn how to predict the next character in URL generation. For the next input extraction, the beginning character becomes $\{a\}$. So in this case, the input sequence becomes $\{a\ c\ e\ b\ o\}$ and the output sequence becomes $\{a\ c\ e\ b\ o\ o\}$. We repeat the same extraction procedure until every character in the URL has been selected.

Before all the input sequence and output sequence are fed into the model, we need to convert the data sequence into indexes so that the neural network can read and learn. In character/index conversion, a dictionary is built for

mapping characters with indexes, in which the key is the character, and the value is the index. For example, if the dictionary is generated by alphabetical order, then the input characters $\{f\ a\ c\ e\ b\}$ can be represented as $\{6\ 1\ 3\ 5\ 2\}$, the output characters $\{f\ a\ c\ e\ b\ o\}$ can be represented as $\{6\ 1\ 3\ 5\ 2\ 15\}$. Until now, everything is ready for training the sequence-2-sequence model. The encoder and decoder are both multiple-layer LSTM RNNs. Each of them has 3 hidden layers, and each hidden layer contains 100 hidden neurons. The architecture, activation functions, and important parameters are previously introduced in Section 5.1.2. The fake URLs generated by the sequence-2-sequence model are presented and discussed in Section 5.2.7

5.2 Evaluation

In this section, first we give a comprehensive anatomical analysis for online advertisement URLs, then we discuss which architecture that used for classification, and finally we demonstrate the comparison of two groups of models for visualization.

5.2.1 Anatomy Analysis for Advertisement URLs

In this section, we give a complete statistical analysis for the three URL sources based on lexical-based features extracted in our experiment.

First, we want to know the length distribution for different URL sources and different components in each URL. Figure 5.7 shows that benign advertise-

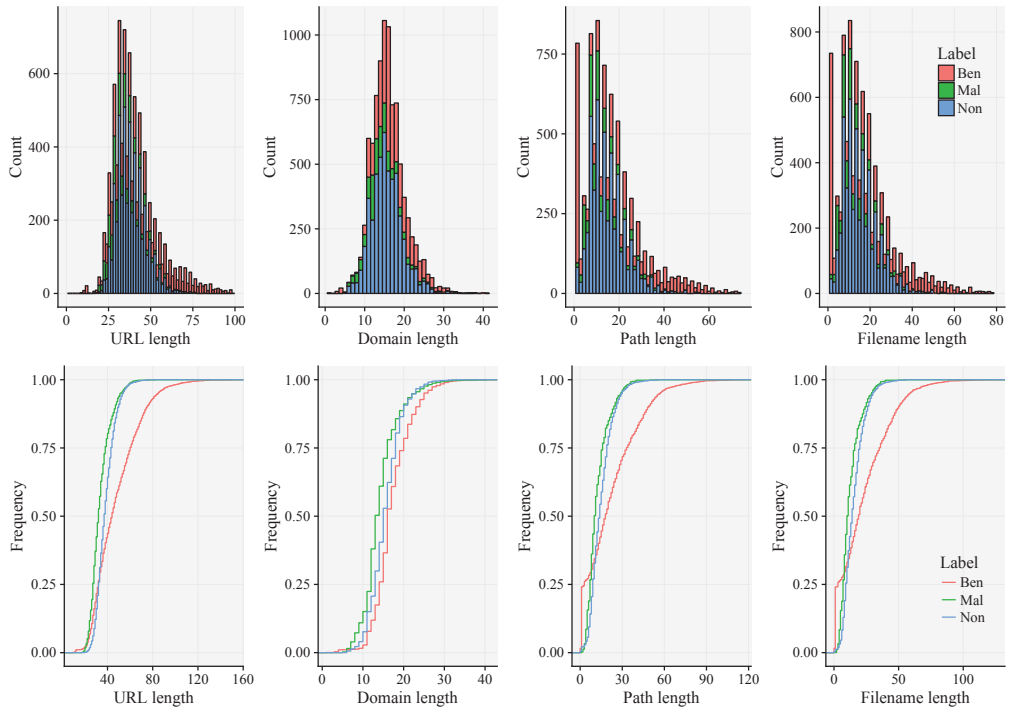

Figure 5.7: Length analysis for different components in URL

ments tend to have longer URL, longer path, and longer filename. Nearly 25% of benign ad-URLs are longer than 60 characters, whereas the number for malicious ad-URLs and non-ad URLs are only about 3%. Based on the previous studies [49], malicious URLs usually have longer domain names because attackers can obfuscate the host with a large hostname or another domain. However, in our experiment, the length distributions for the domain name in different URL sources are close to each other. Domain name length in malicious ad-URLs does not show too much difference from benign ad-URLs and non-ad URLs.

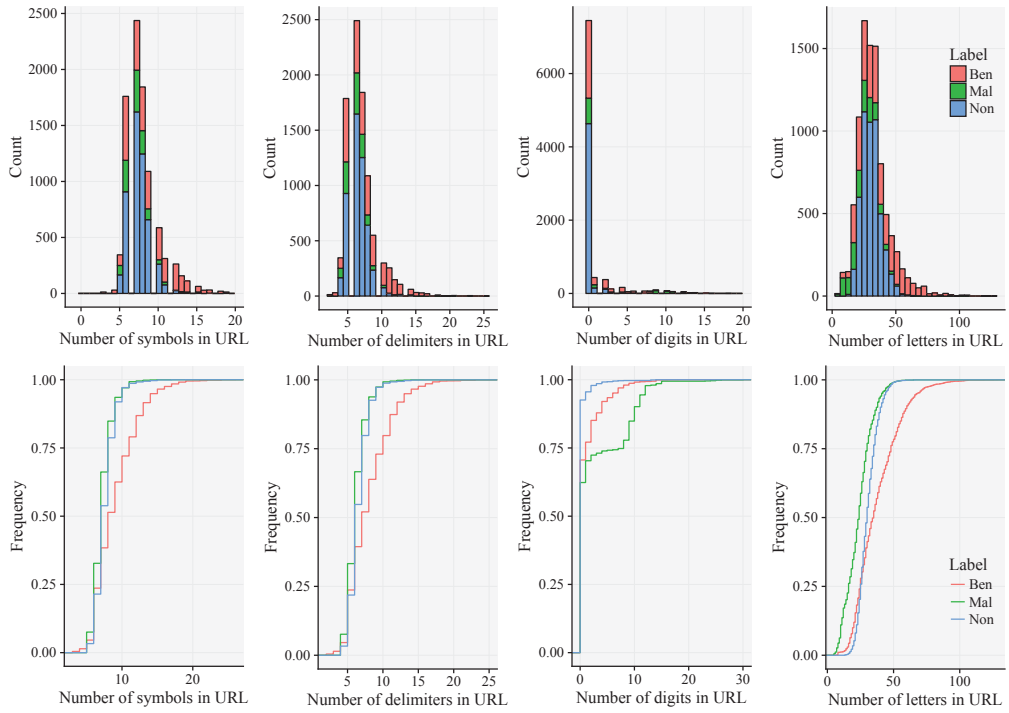

Figure 5.8: Numbers of symbols, delimiters, digits and letters in different URLs

Then, we record the number of symbols (*';', '@', '+', '?', '=', '-', '!', '%'* etc.), the number of delimiters (*'.', '/', '∼' , '&', '?' '#', '\'*), the number of digits, and the number of letters in each URL source. Again this time, different URL sources show different characteristics. In Figure 5.8, nearly 25% of benign ad-URLs have at least 10 symbols and 10 delimiters in their URL, but the number for malicious ad-URLs and non-URLs is less than 5%. Only a small portion of non-ad data sources (around 5%) have digits in the URLs. However, 35% of malicious ad-URLs contain digit tokens, and this feature can help to distinguish malicious ad-URLs from non-ad webpages.

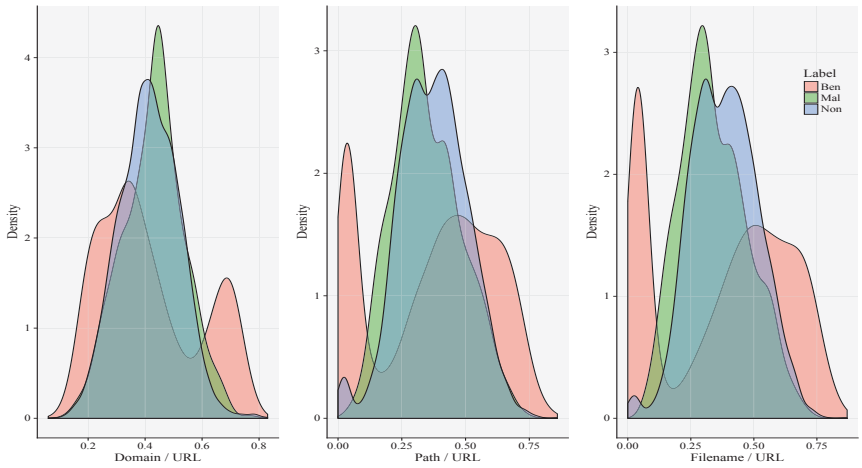

Figure 5.9: Density distribution of different length ratio features

In the next stage, we analyze the value of length ratio feature in our feature set. We compare *Domain/URL, Path/URL and Filename/URL*. In Figure 5.21, the distribution of these three features in non-ad URLs looks like a normal distribution, whereas two distinct peaks (local maxima) appear in benign ad-URLs, which make benign ad-URLs different from non-ad URLs.

Table 5.5: Statistical summary of two features for different URLs

URL sources	**Ad-related keywords**	**IP as domain**
Malicious Ads	45.40%	25.00%
Benign Ads	11.63%	0.00%
Non-Ads	5.81%	0.00%

According to Table 5.5, *Ad-related keywords* and *IP as domain* are two influential features for malicious advertising detection. In our experiment, *'ad', 'advert', 'banner', 'popup', 'sponsor', 'iframe', 'googlead', 'adsys', 'adser', 'account', 'login', 'singin', 'websrc', 'sec'* are extracted as *Ad-related key-*

words feature. 45.4% of malicious ad-URLs and 11.63% of benign ad-URLs contain those tokens, indicating that ad-related keywords are more common in advertising URLs than non-ad URLs. Confirming the findings of [94] [52], *IP as domain* is far more popular in malicious ad-URLs. Table 5.5 shows that, 25% of malicious advertising URLs are hosted by IP address, but none of the other two data sources use IP as their domain name.

5.2.2 Evaluation of Different Architectures

In this section, we optimize the parameters for building the deep neural network by comparing different architectures mentioned in Section 5.1.2.2. For the architectures evaluated in our experiment, the number of hidden layers and the number of neurons in each hidden layer are listed in Table 5.6.

Table 5.6: Different architectures evaluated in our experiment

Number of neurons in each hidden layer	**Number of hidden layers**				**Activation function**	**Loss function**
	1	***2***	***3***	***4***		
F1	141–96–2	141–96–76–2	141–96–76–47–2	141–96–76–47–20–2	*ReLU* for input layer and hidden layers, *Softmax* for output layer	*Cross entropy* as loss function, *AdaGrad* for minimizing loss function
F2	141–13–2	141–26–13–2	141–65–26–13–2	141–104–65–26–13–2		
F3	141–16–2	141–60–16–2	**141–90–60–16–2**	141–112–90–60–16–2		
F4	141–5–2	141–10–5–2	141–25–10–5–2	141–40–25–10–5–2		
F5	141–80–2	141–80–59–2	141–80–59–47–2	141–80–59–47–39–2		
F6	141–63–2	141–63–31–2	141–63–31–12–2	141–63–31–12–8–2		

According to [82], advertisement URL detection is a cost-sensitive classification task. The undetected malicious online websites (false negative) lead to worse results than the misclassified legitimate URLs (false positive). In a URL detection system, some false alarms are more tolerable than the misclassification of malicious webpages, since the undiscovered malicious URLs can result in irreversible harm to the customers. Therefore, False Negative

Table 5.7: Metrics used for evaluating the performance of different algorithms

Metrics	Expression
Precision	$Pr = \frac{TP}{TP+FP}$
Recall	$Rc = \frac{TP}{TP+FN}$
False positive rate	$FPR = \frac{FP}{FP+TN}$
False negative rate	$FNR = \frac{FN}{TP+FN}$
F-score	$F-score = \frac{2\times Pr\times Re}{Pr+Re}$
Accuracy	$Acc = \frac{TP+TN}{TP+TN+FP+FN}$

Rate (FNR) is considered as a significant metric for performance evaluation in our experiments.

In our experiments, Precision (Pr), Recall (Re), FNR, False Positive Rate (FPR), F-score and Accuracy (Acc) are used as evaluation metrics. The formulas are expressed in Table 5.7, and TP, TN, FP, FN are true positive (detected ad-URLs), true negative (detected legitimate URLs), false positive (misclassified legitimate URLs) and false negative (undetected ad-URLs) respectively.

From Table 5.8, we can see that different architectures really have dramatic effects on the performance of deep neural networks. Formula $\{F3\}$ with $\{3\}$ hidden layers shows better performance than other architectures, since it can achieve high F-score and low FNR. Also in agreement with the argument in [100], we find that "*deeper is not always better*" in our model. When we increase the number of hidden layers from 3 to 4, most of the models suffer a degradation in both F-score and FNR. The advantage of multiple layers in deep neural network is that they can learn features at different levels of

Table 5.8: F-score and False Negative Rate for different architectures in our experiment

Scenario A	**F1**		**F2**		**F3**		**F4**		**F5**		**F6**	
Hidden layer	**F-score**	**FNR**	**F-score**	**FNR**	**F-score**	**FNR**	**F-score**	**FNR**	**F-score**	**FNR**	**F-score**	**FNR**
1	81.22%	26.83%	87.56%	8.50%	81.86%	26.82%	91.83%	7.97%	87.10%	20.0%	86.07%	20.78%
2	92.82%	7.70%	94.34%	5.66%	92.83%	6.22%	94.01%	5.83%	91.56%	10.60%	92.16%	9.70%
3	95.26	5.69%	94.12%	6.80%	**95.64%**	**3.30%**	92.88%	7.12%	92.68%	8.18%	91.63%	5.95%
4	91.72%	8.87%	92.29%	8.46%	91.02%	4.79%	92.83%	7.96%	91.46%	10.23%	91.67%	6.67%
Scenario B	**F1**		**F2**		**F3**		**F4**		**F5**		**F6**	
Hidden layer	**F-score**	**FNR**	**F-score**	**FNR**	**F-score**	**FNR**	**F-score**	**FNR**	**F-score**	**FNR**	**F-score**	**FNR**
1	93.92%	5.31%	93.29%	7.06%	96.30%	2.83%	93.70%	6.46%	93.03%	11.5%	88.58%	11.42%
2	94.47%	5.69%	93.27%	7.13%	96.28%	2.81%	93.88%	5.69%	91.36%	10.62%	90.08%	9.89%
3	94.79%	5.06%	92.36%	8.86%	**97.98%**	**2.02%**	94.05%	5.55%	91.76%	10.02%	90.03%	9.03%
4	91.70%	6.65%	91.22%	9.57%	95.78%	3.88%	91.46%	9.72%	88.98%	12.33%	88.21%	11.25%
Scenario C	**F1**		**F2**		**F3**		**F4**		**F5**		**F6**	
Hidden layer	**F-score**	**FNR**	**F-score**	**FNR**	**F-score**	**FNR**	**F-score**	**FNR**	**F-score**	**FNR**	**F-score**	**FNR**
1	89.67%	11.23%	89.38%	11.20%	92.58%	10.04%	91.20%	8.32%	86.97%	12.37%	87.82%	13.30%
2	90.89%	10.52%	89.69%	11.23%	92.60%	10.06%	92.60%	8.53%	88.96%	12.36%	90.75%	10.57%
3	91.47%	10.06%	90.73%	10.50%	**95.41%**	**6.02%**	95.42%	6.79%	90.49%	10.57%	91.73%	9.69%
4	88.87%	12.36%	90.31%	10.46%	92.53%	7.62%	92.53%	8.32%	93.83%	9.62%	89.21%	12.57%
Scenario D	**F1**		**F2**		**F3**		**F4**		**F5**		**F6**	
Hidden layer	**F-score**	**FNR**	**F-score**	**FNR**	**F-score**	**FNR**	**F-score**	**FNR**	**F-score**	**FNR**	**F-score**	**FNR**
1	90.52%	10.57%	93.31%	5.65%	96.67%	3.35%	93.61%	5.73%	92.96%	4.57%	92.17%	6.69%
2	93.18%	6.63%	93.93%	4.66%	97.25%	2.30%	94.49%	4.66%	92.23%	5.55%	93.00%	6.03%
3	94.10%	6.59%	95.00%	3.05%	**98.71%**	**1.31%**	95.43%	3.67%	92.43%	5.19%	93.41%	6.59%
4	92.19%	5.65%	94.17%	2.01%	97.55%	1.56%	94.45%	5.50%	92.85%	5.47%	91.43%	9.68%

abstraction. But, it is not always necessary to build a complex network. As the size of neurons and hidden layers increase, the neural network need learn and train more parameters, which will heighten the chances of overfitting. When overfitting happens, the neural network just memorizes the training data and fails to generalize new data. In addition, it is computationally expensive to train a deep neural network. So the more complicated the architecture, the longer it takes to train the neural network. This works against our original intention to build a lightweight detection system.

Finally the architecture of 141–90–60–16–2 is used for training DNN and AutoEncoder in our experiment.

5.2.3 Evaluation of Different Activation Functions

In this section, the performance of different activation functions mentioned in Section 5.1.2.3 is evaluated for URL detection and classification.

Table 5.9: Evaluation of different activation functions in the experiment

Activation Function	Scenario A					Scenario B				
	Pr	**Rc**	**F-score**	**FPR**	**FNR**	**Pr**	**Rc**	**F-score**	**FPR**	**FNR**
ReLU	94.60%	96.70%	95.63%	1.00%	3.30%	98.27%	97.98%	98.13%	7.00%	2.02%
ELU	96.93%	96.15%	96.54%	4.89%	3.85%	98.78%	97.39%	98.08%	5.31%	2.61%
Softplus	96.55%	90.59%	93.47%	3.64%	9.41%	96.16%	97.33%	96.74%	3.20%	2.67%
Softsign	98.06%	95.60%	96.82%	2.97%	4.40%	96.98%	95.17%	96.01%	4.20%	4.83%
Sigmoid	98.13%	93.28%	95.67%	3.06%	6.72%	98.97%	96.57%	97.76%	4.07%	3.43%
Tanh	95.12%	94.93%	95.03%	7.82%	5.07%	97.73%	95.36%	96.53%	9.48%	4.64%

The activation functions are *ReLU, ELU, Softplus, Softsign, Sigmoid* and *Tanh*. Based on the results in Section 5.2.2, the architecture of 140–90–60–16–2 is used for building the deep neural network, and each activation function is applied in the input layer and hidden layer, *Softmax* activation function is used in the output layer for binary classification task.

Table 5.9 shows Pr, Re, FNR, FPR and F-score for different activation functions in both Scenario A and Scenario B. We can see that there are not too many differences between different activation function in URL detection and classification, *ReLU* shows good performance in Re and FNR, whereas *Softsign* and *Sigmoid* show good performance in Pr.

Compare Table 5.9 and Table 5.8, we can make a conclusion that the architecture of a deep neural network plays a more important role than the activation function in URL identification. In our experiment, the deep neural network can always achieve a satisfactory performance, as long as the

architecture of the network is reasonable and the number of iterations for training the neural network is big enough.

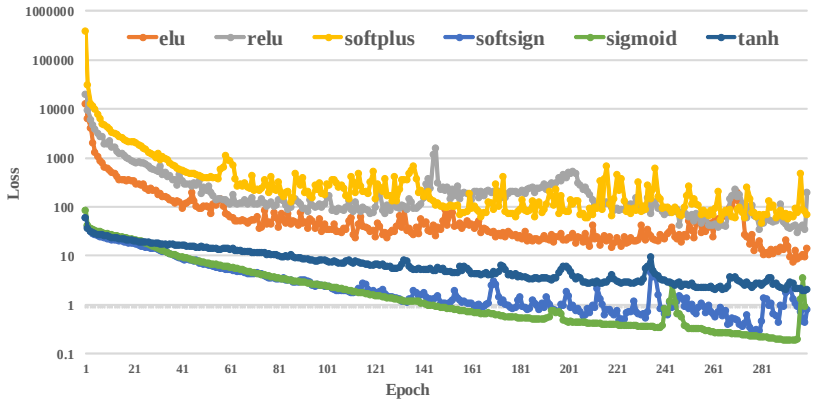

Figure 5.10: The variation trend of entropy loss in training DNN

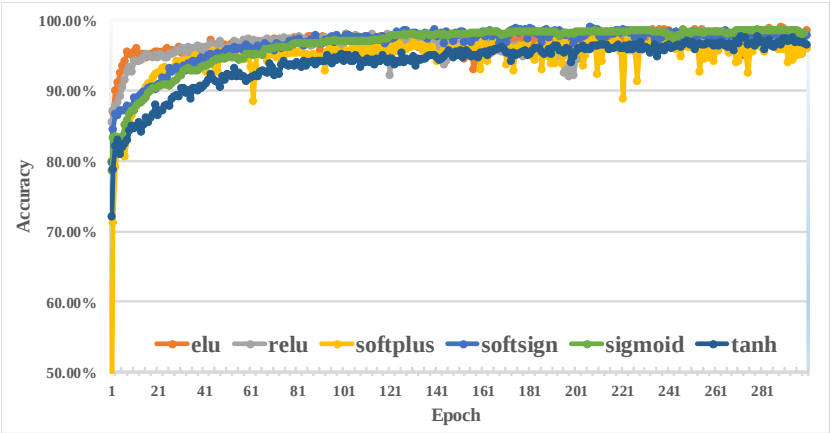

Figure 5.11: The variation trend of accuracy in training DNN

Epoch is a terminology used in training deep neural network. It refers to one forward pass and one backward pass of training the entire dataset. Epoch defines the number of iterations the dataset has been trained before testing

process. In our experiment, the epoch is chosen as 300, which means that the entire dataset is trained for 300 times. Figure 5.10 and Figure 5.11 show the variation trend of the entropy loss and the accuracy along the 300 epochs. We can see that as the epoch number increases, the entropy loss decreases and the accuracy increases. Although each activation function has a different starting point in entropy loss and accuracy, the final results of each activation function tend to be the same after 300 epochs. And this may also help to explain why activation function is not so important in building and training a deep neural network.

5.2.4 Advertising URL Classification

We implement DNN and RNN for online advertising classification, and then evaluate the performance of deep learning based algorithms with conventional machine learning techniques. The machine learning techniques used in our experiment are Decision Tree (C4.5), Random Forest (RF), K-nearest neighbor (KNN), Logistic Regression, Naive Bayes (NB) and Bayesian Network (B-network). We perform 10-fold cross-validation on both deep learning based approaches and machine learning techniques for evaluation and record the overall performance.

Figure 5.12 to Figure 5.15 show the comparison between deep learning techniques and conventional machine learning techniques in different scenarios, and the values of FNR in each scenario are highlighted in the figure. Our proposed approach gives better performance compared with the machine learn-

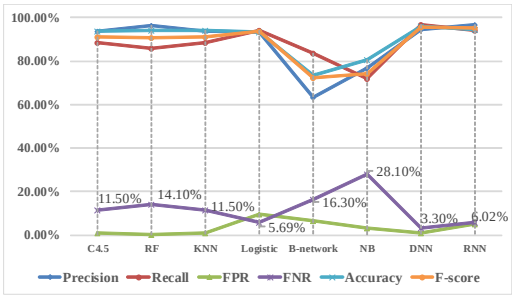

Figure 5.12: Comparison of deep learning techniques and machine learning techniques in Scenario A

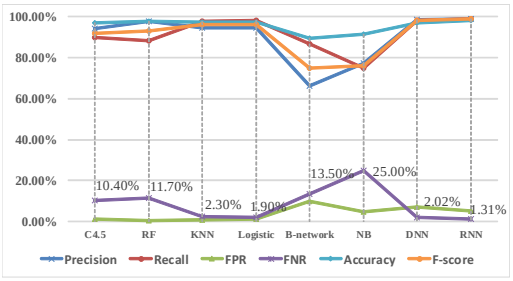

Figure 5.13: Comparison of deep learning techniques and machine learning techniques in Scenario B

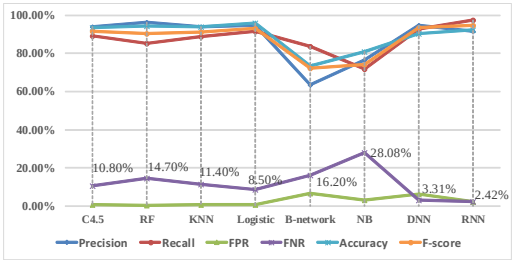

Figure 5.14: Comparison of deep learning techniques and machine learning techniques in Scenario C

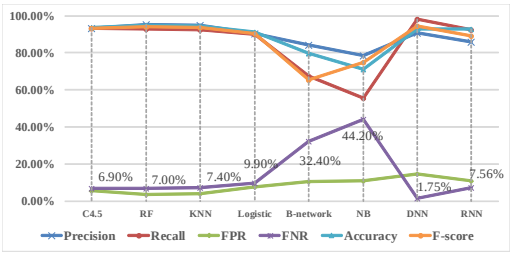

Figure 5.15: Comparison of deep learning techniques and machine learning techniques in Scenario D

ing models. In previous works, [50] conducts a suspicious URL detection system using lexical-based and host-based features, and they obtain a FNR of 7.6%; [98] proposes a cross-layer system for malicious URL detection and the final FNR is 6.09%; [85], [94] and [45] focus on spam, phishing and malicious URL detection and their FNRs are 7.5%, 5.0% and 10.69% respectively. With the implementation of DNN and RNN, the FPR in our model is as low as 1.31%, indicating that our proposed framework shows a reasonable improvement over previous studies. By introducing deep learning algorithm in

the URL classification task, our approach can enhance the existing system with a lower FNR and high F-score value.

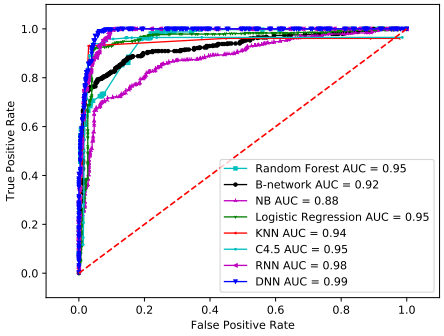

Figure 5.16: Receiver Operating Characteristic curve for Scenario A

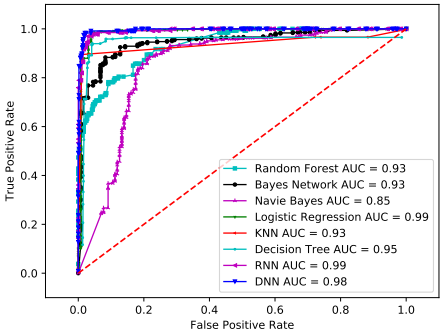

Figure 5.17: Receiver Operating Characteristic curve for Scenario B

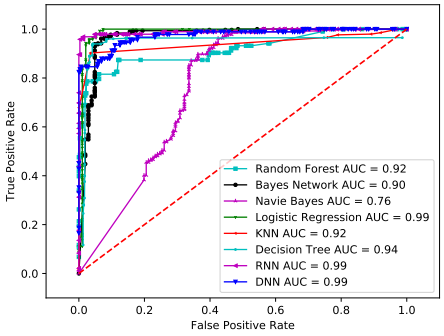

Figure 5.18: Receiver Operating Characteristic curve for Scenario C

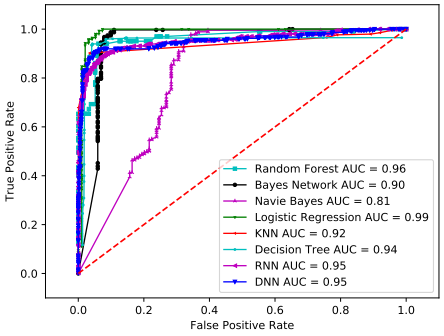

Figure 5.19: Receiver Operating Characteristic curve for Scenario D

Finally, we construct Receiver Operating Characteristic (ROC) curve to visualize the performance of the binary classifiers used in our experiment. Generally speaking any increase in FPR will be accompanied by a decrease in TPR, and the ROC curve can display the trade-off between FPR and true positive

rate (TPR). Also, we can quantitatively compare different algorithms using Area Under the ROC Curve (AUC), because a larger area under the ROC curve implies a higher accuracy for the binary classification.

From Figure 5.16 to Figure 5.19 we can see , deep learning based algorithms show excellence performance in both scenarios. The ROC curves for DNN and RNN are closer to the top left corner, and the AUCs for RNN and DNN are nearly 1. In addition, we notice that conventional machine learning techniques such as Logistic Regression, C4.5 and RF also perform well in terms of Accuracy, F-score, and AUC value. This may indicate that the lexical-based features we extract in the study are useful for distinguishing different URL sources, and are good candidates for online advertising identification.

5.2.5 Execution Time Analysis

The objective of this thesis is to build a lightweight online advertising detection system. And in this section, we compare the execution time for different algorithms.

A lightweight system is significant for online advertisement detection task, since any delay of classification can cause unexpected results for customers. Although features like WHOIS properties contain useful information about the webpages, and are widely used in previous studies such as [50] [64] and [48], they can cause time latency issues by sending requests to remote servers. And these issues become more severe when the size of data is large. [82] demonstrated a classification system for benign-ad URLs with lexical-based

analysis. However, since the introduction of "*n-gram*" and "*bag of word*" techniques, the size of their lexical-based features is large. This leads to a problem that the training time for their classification system is long, and the time delay will still degrade the user experience.

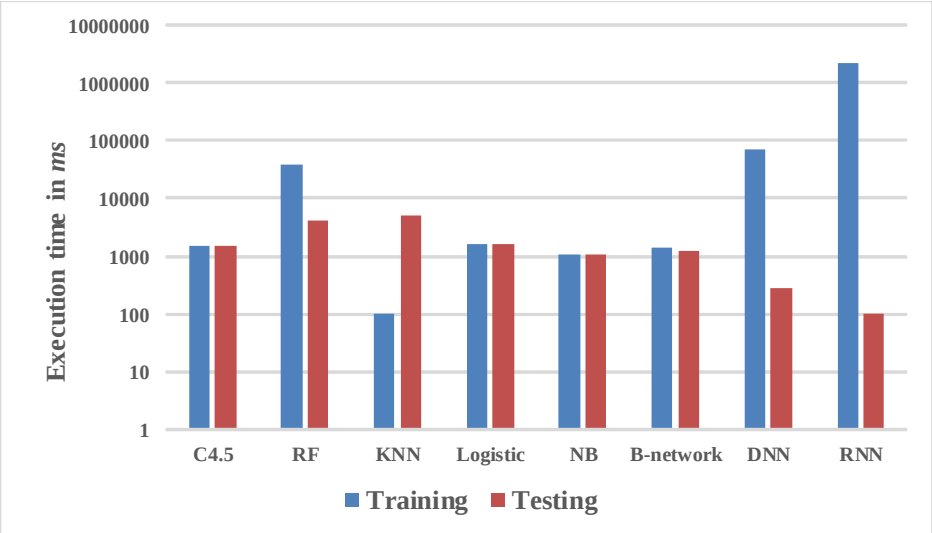

Figure 5.20: The execution time (in *ms*) of different classifiers

In Figure 5.20, we compare the training time and testing time of different algorithms. We can see that both machine learning techniques and deep learning algorithms have low execution times. The maximum execution time for machine learning algorithms is *40.0s* with Random Forest. Although it takes longer to train deep learning models (*69.3s* for DNN and *2189.6s* for RNN), it only takes *0.3s* and *0.09s* for testing new instances using DNN and RNN. According to [94], online blacklist system will averagely take 1 hour to 10 hours to identify a single malicious URL. So the results in Fig-

ure 5.20 indicate that our lightweight detection system shows a considerable improvement over the existing blacklist approaches.

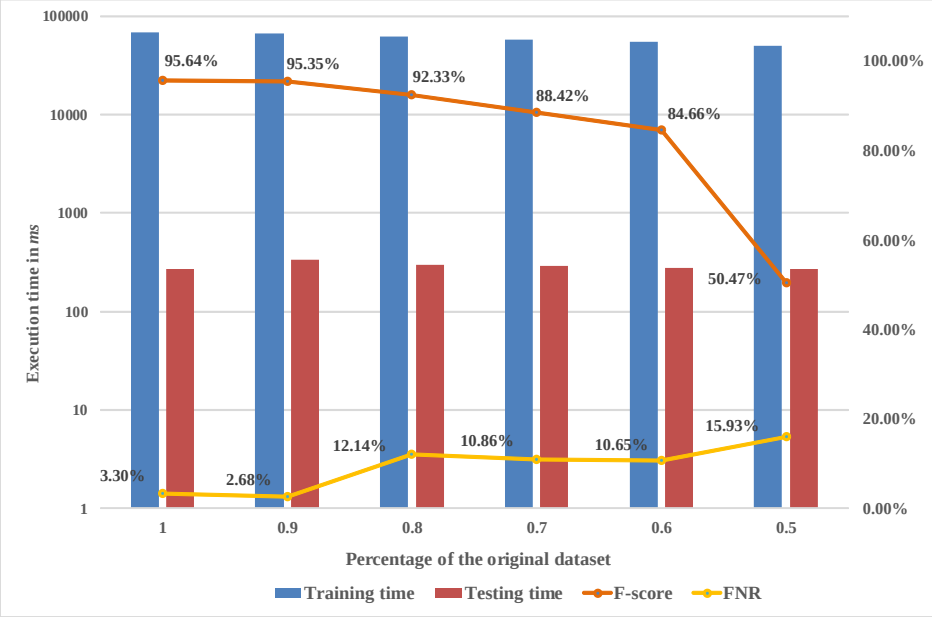

Figure 5.21: The F-score, FNR and execution time of DNN at different sizes of dataset

In the next, we want to evaluate the trade-off between the performance and the execution time of the proposed DNN model. So we implement the following experiment by compare F-score, FNR and execution time at different input data sizes. We set the percentage ratio to 1, 0.9, 0.8, 0.7, 0.6, and 0.5, which means only a certain percentage of data is feed into the DNN model. In Figure 5.21, we can see that the decrease of data size ratio from 1 to 0.5 does not influence the execution time too much. However, the performance of the proposed model degrades a lot. When we use only 50% dataset for training

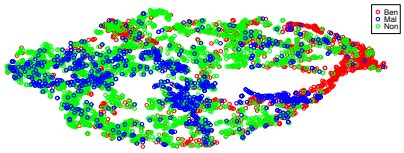

Figure 5.22: Visualizing different URLs: t-SNE

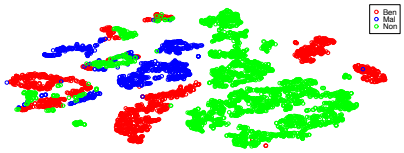

Figure 5.23: Visualizing different URLs: AutoEncoder + t-SNE

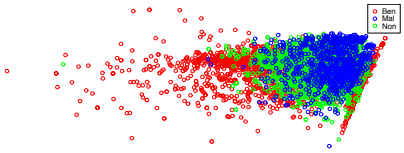

Figure 5.24: Visualizing different URLs: PCA

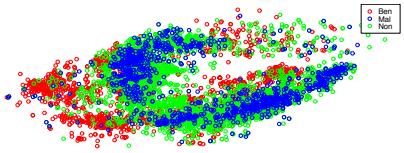

Figure 5.25: Visualizing different URLs: AutoEncoder + PCA

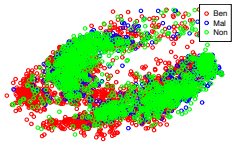

Figure 5.26: Visualizing different URLs: Sammon

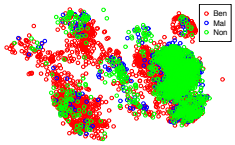

Figure 5.27: Visualizing different URLs: AutoEncoder + Sammon

the DNN model, the F-score is only 50.47%, and the FNR is 15.93%, which is not acceptable for a classification system. So training on a big dataset is beneficial to build a reliable deep learning model with high performance.

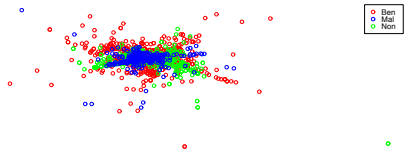

Figure 5.28: Visualizing different URLs: LLE

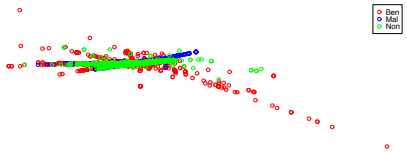

Figure 5.29: Visualizing different URLs: AutoEncoder + LLE

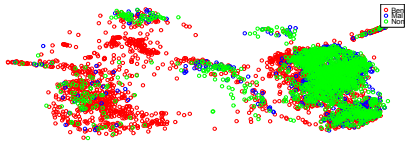

Figure 5.30: Visualizing different URLs: Isomap

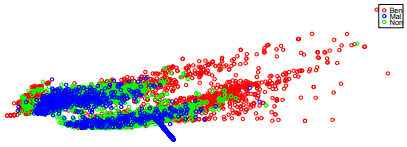

Figure 5.31: Visualizing different: AutoEncoder + Isomap

5.2.6 Advertising URL Visualization

We propose a novel approach for visualizing online advertising URLs. First an AutoEncoder with architecture of 141–90–60–16 is used for data preprocessing, then dimensionality reduction algorithm *t-SNE* is applied to convert data dimension from 16 to 2 for final visualization. Several other popular dimensional reduction algorithms (PCA, LLE, *Sammon mapping, Isomap*) are also implemented for comparison in our experiment and are listed in Table 5.4. Figure 5.22 to Figure 5.31 show the visualization results for two groups of models: the baseline based model and the AutoEncoder based model.

We should notice that the whole visualization process is completely unsupervised, and the labels are only for coloring different URL sources. The model

AutoEncoder plus *t-SNE* in Figure 5.23 clearly outperforms all the other models. In AutoEncoder plus *t-SNE*, same URL family is mostly grouped in the same clustering space, which demonstrates that the lexical-based features extracted in our experiment are good representations of online advertising URLs.

5.2.7 Fake Advertising URL Generation

In this section, we discuss the fake URLs that produced by the sequence-2-sequence model. Three groups of URLs are generated based on the three data sources, the fake non-ad URLs, the fake benign-ad URLs and the fake malicious-ad URLs, and they are shown as follows.

- Fake non-ad URLs:

  `.ons/frene&=hicomer`

  `qqint/ninge`

  `t.hewee.wiy/?/c//`

  `w.qqcom.yogol/pri/om`

  `www.koccmins/boIna`

  `twittf.k.coogopkoc/rohetttgohoim.com/?th`

  `t.hewfebe.com/?p=d2//`

  `3/i///p22/p/`

  `intenteehs//n.sing/en/zaed/e.:irogtohetxtenc`

www.limau/u/ymins/meanre

wwwthiogs/maahq:outpuhs/irs/menarentsehtkinge

ominrephep=qj/qom/cc/en/ins/rlen/en/a123.comimacgrttggohoo-.:

intimepamimacx

www.lou/pnu/ehttcen/a.jinominretetIn-/omecepg/minr

- Fake benign-ad URLs:

carbonite.com/

www.pcls.hww.ce.wikipem/em/

www.vbuce.wi.wig.cww.wigentervi/

www.rerackup.com/

www.aginacle.com/producs.com/

www.wiles.com/fackup/gttluran2

ehoww.ag/sotftoaracku/Aloe.wikipemfe.com/propchrown/pripsoces/

ag.ikisoce.wikices/a/igcomtnipvopaararacv.wikonirunt

ww.ce.wikipem/

www.vbraccos/

yko.ceky

ww.ivep.com

www.ptomacku/obbackups.cww.w.ie_glaren.com/P

w.wibedlwplicelice.cww.wige_gorbadlyplycelewplicarbwi-gllwletlicebar.
com/ptlobebyplhaarls.nenorv

- Fake malicious-ad URLs:

  238.153.23.78/~cem/an9.com/anys

  12.193.69.165.245.152/cgimp

  193.69.165.245/annonser

  193.69.165.139.152.2/cmrdjsp

  192/imagesp

  178/ads/

  190.224.215.205.192/imt

  123.230/reasimimpimpimpimag

  238.157.24.71.4wp

  192.22.93.150/mmadour

  14.38.cofer

  14.38.com/ader

  193.98.152.2/gomdsimpipimpimpimag

  238.1.im5.3dno

  s.179/reampan9.cofeb_gitesitesr

  193.158.23.67.2232/w1r/r

```
192.192/impan9.cofeb_giterr.imp

193.69.165.2355.39//ctimpan9.uo2.69.152.2/ggbannerr/imp

193.68.165.205.19.161/resr
```

After we iterate all the original data sources and feed them into the sequence-2-sequence network for training, the proposed model can gradually remember the patterns of different datasets, and can be used to generate fake URLs by a given "<START>". In our experiment, we assign the string "https://" to the "<START>" as the beginning of a fake URL, then our trained model can generate the first character that follows the "<START>", then generate the second character that follows the first character, and so on. Based on the fake URL listed above, we can clearly discover that different types of URLs have different patterns and appearances. The fake malicious-ad URLs are distinct from other fake URLs, since they always contain numeric characters at the beginning of the URLs. This pattern is consistent with the findings discussed in Section 5.2.1 that malicious adverting URLs always use IP address as their domain name. This consistency also demonstrates that the proposed sequence-2-sequence model could learn the patterns of different data sources and create fake URLs with distinguishing features.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

With the rapid development of the Internet since the mid-1990s, the online advertisement has become one of the most important platforms for commercial propaganda. As a result, a large number of fraudulent activities such as spam, phishing, and malware are distributed through online advertising. Existing advertising blocking systems are based on blacklists or hand-coded rules, which are easy to be defeated by attackers. In this paper, we design a lightweight online advertising visualization system using lexical-based URL features.

In this thesis, we conduct a comprehensive statistical analysis of lexical-based URLs for different data sources. Benign ad-URLs always have a longer path and a longer filename in their URLs, whereas malicious ad-URLs more com-

monly contain suspicious ad-related keywords and they are often hosted by IP address. Then a deep learning based approach is developed for classifying and visualizing advertising URLs. In recent years deep learning algorithms have gained great success in image recognition and language modeling. In our paper, we illustrate that DNN and RNN are also good solutions for online advertising analysis. After architecture optimizing, deep learning based techniques perform better than conventional machine learning methods and our approach enhances the existing URL detection system with FNR as low as 1.31%. After that, we design a novel deep learning based approach for data visualization. AutoEncoder is applied for data preprocessing, and then *t-SNE* is implemented for final visualization. This approach can create clear clusters for different URL data sources and can be used to evaluate the quality of extracted features. Finally, we develop a RNN based deep learning framework called sequence-2-sequence model for fake URL generation. The fake malicious-ad URLs always begin with numeric characters, and are distinct from the other two types of URLs. This is consistent with the original malicious-ad URLs, which often use IP address as their domain. So our proposed model can learn the influential patterns of different URLs and can generate fake URLs with distinguishing characteristics.

6.2 Future Work

In the future, the following work can be performed to extend the work of this thesis.

- By scaling our data set to millions of URLs, we can address the issue of how to apply our deep learning based approach to real-world cases. We can evaluate the performance of deep learning algorithms on online advertisement URL detection with large data set. Furthermore, we can design and develop an online system for URL identification, which can be a feasible complementary for the existing blacklist systems.

- In this thesis, we implement deep learning based methods for advertising URL classification and visualization. In the experiment, DNN and RNN are used as classifiers, AutoEncoder is used for feature preprocessing, sequence-2-sequence model is used for fake URL generation. In the future, there is a big opportunity that this deep learning based framework can be used, extended or improved as other functionalities. For example, AutoEncoder can be used for feature compression and extraction. Sequence-2-sequence model can be used for phishing URL and spam email generation, it can also be used for malware signature generation.

- In this thesis, we present a novel method for data visualization by combining AutoEncoder and *t-SNE*. Our proposed method shows better performance in visualizing online advertising URLs than traditional

machine learning techniques. In the future, we should assess the stability of this method even further by visualizing other data sources.

- Sequence-2-sequence model has shown big potentials in sequential data modeling. In our experiment, this model could generate different types of fake URLs with distinguishing patterns. And in the future, these fake URLs can be used to evaluate the performance and vulnerability of the existing advertising blocking systems.

Bibliography

[1] J. B. Tenenbaum, V. de Silva and J. C. Langford , *Isomap*, `http://web.mit.edu/cocosci/isomap/isomap.html`.

[2] Adobe, *Click Here: The State of Online Advertising*, `https://www.adobe.com/aboutadobe/pressroom/pdfs/Adobe_State_of_Online_Advertising.pdf`, 2013.

[3] Yoshihiro Ando, Hidehito Gomi, and Hidehiko Tanaka, *Detecting fraudulent behavior using recurrent neural networks*, (2016).

[4] Yuichiro Anzai, *Pattern recognition and machine learning*, Elsevier, 2012.

[5] Itamar Arel, Derek C Rose, and Thomas P Karnowski, *Deep machine learning-a new frontier in artificial intelligence research [research frontier]*, IEEE Computational Intelligence Magazine **5** (2010), no. 4, 13–18.

[6] John Aycock, *Spyware and adware*, vol. 50, Springer Science & Business Media, 2010.

[7] BannerSnack, *Banner Ads*, `https://www.bannersnack.com/blog/banner-advertising-basics/`, 2008, (Accessed date September 2016).

[8] Yoshua Bengio, Pascal Lamblin, Dan Popovici, Hugo Larochelle, et al., *Greedy layer-wise training of deep networks*, Advances in neural information processing systems **19** (2007), 153.

[9] Sruti Bhagavatula, Christopher Dunn, Chris Kanich, Minaxi Gupta, and Brian Ziebart, *Leveraging machine learning to improve unwanted resource filtering*, Proceedings of the 2014 Workshop on Artificial Intelligent and Security Workshop, ACM, 2014, pp. 95–102.

[10] Lei's Blog, *Tensorflow lyrics generation* , `http://leix.me/2016/11/28/tensorflow-lyrics-generation/`.

[11] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio, *Learning phrase representations using rnn encoder-decoder for statistical machine translation*, arXiv preprint arXiv:1406.1078 (2014).

[12] Hyunsang Choi, Bin B Zhu, and Heejo Lee, *Detecting malicious web links and identifying their attack types.*, WebApps **11** (2011), 11–11.

[13] Axel Cleeremans, David Servan-Schreiber, and James L McClelland, *Finite state automata and simple recurrent networks*, Neural computation **1** (1989), no. 3, 372–381.

[14] Ronan Collobert and Jason Weston, *A unified architecture for natural language processing: Deep neural networks with multitask learning*, Proceedings of the 25th international conference on Machine learning, ACM, 2008, pp. 160–167.

[15] Peter Connor, Daniel Neil, Shih-Chii Liu, Tobi Delbruck, and Michael Pfeiffer, *Real-time classification and sensor fusion with a spiking deep belief network*, Neuromorphic Engineering Systems and Applications (2015), 61.

[16] cs.stanford.edu, *Cnns for visual recognition*, `https://downloads.pagefair.com/wp-content/uploads/2016/05/2015_report-the_cost_of_ad_blocking.pdf`, 2015.

[17] Daniel McNeela, *Sammon mapping*, `http://retina.readthedocs.io/en/latest/nldr/sammon/`.

[18] Omid E David and Nathan S Netanyahu, *Deepsign: Deep learning for automatic malware signature generation and classification*, Neural Networks (IJCNN), 2015 International Joint Conference on, IEEE, 2015, pp. 1–8.

[19] Jeffrey Dean, Greg Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Mark Mao, Andrew Senior, Paul Tucker, Ke Yang, Quoc V Le, et al., *Large scale distributed deep networks*, Advances in neural information processing systems, 2012, pp. 1223–1231.

[20] Li Deng, Geoffrey Hinton, and Brian Kingsbury, *New types of deep neural network learning for speech recognition and related applications: An overview*, Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on, IEEE, 2013, pp. 8599–8603.

[21] Yuxin Ding, Sheng Chen, and Jun Xu, *Application of deep belief networks for opcode based malware detection*, Neural Networks (IJCNN), 2016 International Joint Conference on, IEEE, 2016, pp. 3901–3908.

[22] DoubleClick, *Video advertising overview*, `https://support.google.com/dfp_premium/answer/1711021?hl=en`, 2016, (Accessed date September 2016).

[23] John Duchi, Elad Hazan, and Yoram Singer, *Adaptive subgradient methods for online learning and stochastic optimization*, Journal of Machine Learning Research **12** (2011), no. Jul, 2121–2159.

[24] easylist, *EasyList filter lists*, `https://easylist.to/`, Accessed : 2017-06-08.

[25] gitbooks, *Sequence to Sequence mapping with RNN*, https://tilneyyang.gitbooks.io/conversationalmodel/content/seq2seq.html.

[26] GitHub, *Tensorflow Seq2seq Text Summarization*, https://github.com/thunlp/TensorFlow-Summarization.

[27] Xavier Glorot, Antoine Bordes, and Yoshua Bengio, *Deep sparse rectifier neural networks.*, Aistats, vol. 15, 2011, p. 275.

[28] Googole, *Sequence-to-sequence models*, https://www.tensorflow.org/tutorials/seq2seq.

[29] Alex Graves, Marcus Liwicki, Santiago Fernández, Roman Bertolami, Horst Bunke, and Jürgen Schmidhuber, *A novel connectionist system for unconstrained handwriting recognition*, IEEE transactions on pattern analysis and machine intelligence **31** (2009), no. 5, 855–868.

[30] Yanming Guo, Yu Liu, Ard Oerlemans, Songyang Lao, Song Wu, and Michael S Lew, *Deep learning for visual understanding: A review*, Neurocomputing **187** (2016), 27–48.

[31] Mary Hare, *The role of similarity in hungarian vowel harmony: a connectionist account*, Connectionist natural language processing, Springer, 1992, pp. 295–322.

[32] Mary Hare, David Corina, and Garrison Cottrell, *A connectionist perspective on prosodic structure*, Annual Meeting of the Berkeley Linguistics Society, vol. 15, 1989, pp. 114–125.

[33] Geoffrey E Hinton, *Deep belief networks*, Scholarpedia **4** (2009), no. 5, 5947.

[34] Geoffrey E Hinton and Ruslan R Salakhutdinov, *Reducing the dimensionality of data with neural networks*, science **313** (2006), no. 5786, 504–507.

[35] Sepp Hochreiter and Jürgen Schmidhuber, *Long short-term memory*, Neural computation **9** (1997), no. 8, 1735–1780.

[36] Mark F St John and James L McClelland, *Learning and applying contextual constraints in sentence comprehension*, Artificial Intelligence **46** (1990), no. 1-2, 217–257.

[37] Hanna Kamyshanska and Roland Memisevic, *The potential energy of an autoencoder*, IEEE transactions on pattern analysis and machine intelligence **37** (2015), no. 6, 1261–1273.

[38] Jinchuan Ke and Xinzhe Liu, *Empirical analysis of optimal hidden neurons in neural network modeling for stock prediction*, Computational Intelligence and Industrial Application, 2008. PACIIA'08. Pacific-Asia Workshop on, vol. 2, IEEE, 2008, pp. 828–832.

[39] Viktor Krammer, *An effective defense against intrusive web advertising*, Sixth Annual Conference on Privacy, Security and Trust, 2008, IEEE, 2008, pp. 3–14.

[40] Laurens van der Maaten, *t-sne*, `https://lvdmaaten.github.io/tsne/`.

[41] Anh Le, Athina Markopoulou, and Michalis Faloutsos, *Phishdef: Url names say it all*, INFOCOM, 2011 Proceedings IEEE, IEEE, 2011, pp. 191–195.

[42] Yann LeCun and Yoshua Bengio, *Convolutional networks for images, speech, and time series*, The handbook of brain theory and neural networks **3361** (1995), no. 10, 1995.

[43] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton, *Deep learning*, Nature **521** (2015), no. 7553, 436–444.

[44] Yann LeCun, Fu Jie Huang, and Leon Bottou, *Learning methods for generic object recognition with invariance to pose and lighting*, Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on, vol. 2, IEEE, 2004, pp. II–97.

[45] Sangho Lee and Jong Kim, *Warningbird: Detecting suspicious urls in twitter stream.*, NDSS, vol. 12, 2012, pp. 1–13.

[46] Jin-Yan Li, Tommy WS Chow, and Ying-Lin Yu, *The estimation theory and optimization algorithm for the number of hidden units in the higher-order feedforward neural network*, Neural Networks, 1995. Proceedings., IEEE International Conference on, vol. 3, IEEE, 1995, pp. 1229–1233.

[47] Yuancheng Li, Rong Ma, and Runhai Jiao, *A hybrid malicious code detection method based on deep learning*, methods **9** (2015), no. 5.

[48] Zhou Li, Kehuan Zhang, Yinglian Xie, Fang Yu, and XiaoFeng Wang, *Knowing your enemy: understanding and detecting malicious web advertising*, Proceedings of the 2012 ACM conference on Computer and communications security, ACM, 2012, pp. 674–686.

[49] Min-Sheng Lin, Chien-Yi Chiu, Yuh-Jye Lee, and Hsing-Kuo Pao, *Malicious url filtering: A big data application*, big data, 2013 IEEE international conference on, IEEE, 2013, pp. 589–596.

[50] Justin Ma, Lawrence K Saul, Stefan Savage, and Geoffrey M Voelker, *Beyond blacklists: learning to detect malicious web sites from suspicious urls*, Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, 2009, pp. 1245–1254.

[51] Laurens van der Maaten and Geoffrey Hinton, *Visualizing data using t-sne*, Journal of Machine Learning Research **9** (2008), no. Nov, 2579–2605.

[52] Mohammad Saiful Islam Mamun, Mohammad Ahmad Rathore, Arash Habibi Lashkari, Natalia Stakhanova, and Ali A Ghorbani, *Detecting malicious urls using lexical analysis*, International Conference on Network and System Security, Springer, 2016, pp. 467–482.

[53] Niall McLaughlin, Jesus Martinez del Rincon, BooJoong Kang, Suleiman Yerima, Paul Miller, Sakir Sezer, Yeganeh Safaei, Erik Trickel, Ziming Zhao, Adam Doupé, et al., *Deep android malware detection*, Proceedings of the Seventh ACM on Conference on Data and Application Security and Privacy, ACM, 2017, pp. 301–308.

[54] Larry Medsker and Lakhmi C Jain, *Recurrent neural networks: design and applications*, CRC press, 1999.

[55] Abdel-rahman Mohamed, George Dahl, and Geoffrey Hinton, *Deep belief networks for phone recognition*, Nips workshop on deep learning for speech recognition and related applications, vol. 1, 2009, p. 39.

[56] mozdev.org, *the AdBlock project*, `http://adblock.mozdev.org./`, Accessed : 2016-04-20.

[57] Daniel Neil, Michael Pfeiffer, and Shih-Chii Liu, *Phased lstm: Accelerating recurrent network training for long or event-based sequences*,

Advances in Neural Information Processing Systems, 2016, pp. 3882–3890.

[58] Mediatel Newsline, *Global ad expenditure to return to pre-recession peak level this year*.

[59] Jiquan Ngiam, Zhenghao Chen, Daniel Chia, Pang W Koh, Quoc V Le, and Andrew Y Ng, *Tiled convolutional neural networks*, Advances in neural information processing systems, 2010, pp. 1279–1287.

[60] PageFair, *The cost of ad blocking. pagefair and adobe 2015 ad blocking report*, `http://cs231n.github.io/convolutional-networks/`, 2015.

[61] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio, *On the difficulty of training recurrent neural networks*, International Conference on Machine Learning, 2013, pp. 1310–1318.

[62] Razvan Pascanu, Jack W Stokes, Hermineh Sanossian, Mady Marinescu, and Anil Thomas, *Malware classification with recurrent networks*, Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on, IEEE, 2015, pp. 1916–1920.

[63] phishtank, *PhishTank: out of the net, into the tank, join the fight against phishing*, `https://www.phishtank.com/`, Accessed : 2017-06-08.

[64] KV Pradeepthi and A Kannan, *Performance study of classification techniques for phishing url detection*, Advanced Computing (ICoAC), 2014 Sixth International Conference on, IEEE, 2014, pp. 135–139.

[65] Suriyadeepan Ram, *Chatbots with Seq2Seq*, `http://suriyadeepan.github.io/2016-06-28-easy-seq2seq/`.

[66] Dhanalakshmi Ranganayakulu and C Chellappan, *Detecting malicious urls in e-mail–an implementation*, AASRI Procedia **4** (2013), 125–131.

[67] READWRITE, *Report: The 3 Biggest Enterprise Website Malware Vulnerabilities*, `http://readwrite.com/2010/07/26/3-biggest-web-vulnerabilities/`, 2016.

[68] Sam T Roweis and Lawrence K Saul, *Nonlinear dimensionality reduction by locally linear embedding*, science **290** (2000), no. 5500, 2323–2326.

[69] David E Rumelhart and James L McClelland, *On learning the past tenses of english verbs.*, Tech. report, DTIC Document, 1985.

[70] Doyen Sahoo, Chenghao Liu, and Steven CH Hoi, *Malicious url detection using machine learning: A survey*, arXiv preprint arXiv:1701.07179 (2017).

[71] Tara N Sainath, Brian Kingsbury, Bhuvana Ramabhadran, Petr Fousek, Petr Novak, and Abdel-rahman Mohamed, *Making deep be-*

lief networks effective for large vocabulary continuous speech recognition, Automatic Speech Recognition and Understanding (ASRU), 2011 IEEE Workshop on, IEEE, 2011, pp. 30–35.

[72] Hasim Sak, Andrew W Senior, and Françoise Beaufays, *Long short-term memory recurrent neural network architectures for large scale acoustic modeling.*, INTERSPEECH, 2014, pp. 338–342.

[73] Ruhi Sarikaya, Geoffrey E Hinton, and Anoop Deoras, *Application of deep belief networks for natural language understanding*, IEEE/ACM Transactions on Audio, Speech, and Language Processing **22** (2014), no. 4, 778–784.

[74] The H Security, *Scam and Phishing*, `http://www.h-online.com/security/news/item.html`, 2011.

[75] Katsunari Shibata and Yusuke Ikeda, *Effect of number of hidden neurons on learning in large-scale layered neural networks*, ICCAS-SICE, 2009, IEEE, 2009, pp. 5008–5013.

[76] Patrice Y Simard, David Steinkraus, and John C Platt, *Best practices for convolutional neural networks applied to visual document analysis.*, ICDAR, vol. 3, 2003, pp. 958–962.

[77] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman, *Deep inside convolutional networks: Visualising image classification models and saliency maps*, arXiv preprint arXiv:1312.6034 (2013).

[78] Aditya K Sood and Richard J Enbody, *Malvertising–exploiting web advertising*, Computer Fraud & Security **2011** (2011), no. 4, 11–16.

[79] Andreas Stolcke, *Learning feature-based semantics with simple recurrent networks*, Citeseer, 1990.

[80] John Sum, *Comparative study on various pruning algorithms for rnn. i. complexity analysis*, Machine Learning and Cybernetics, 2002. Proceedings. 2002 International Conference on, vol. 4, IEEE, 2002, pp. 2225–2230.

[81] Ilya Sutskever, Oriol Vinyals, and Quoc V Le, *Sequence to sequence learning with neural networks*, Advances in neural information processing systems, 2014, pp. 3104–3112.

[82] Piotr L Szczepański, Adrian Wiśniewski, and Tomasz Gerszberg, *An automated framework with application to study url based online advertisements detection*, Journal of Applied Mathematics, Statistics and Informatics **9** (2013), no. 1, 47–60.

[83] TechTarget, *Floating Ads*, `http://searchcrm.techtarget.com/definition/floating-ad`, 2007, (Accessed date September 2016).

[84] Joshua B Tenenbaum, Vin De Silva, and John C Langford, *A global geometric framework for nonlinear dimensionality reduction*, science **290** (2000), no. 5500, 2319–2323.

[85] Kurt Thomas, Chris Grier, Justin Ma, Vern Paxson, and Dawn Song, *Design and evaluation of a real-time url spam filtering service*, Security and Privacy (SP), 2011 IEEE Symposium on, IEEE, 2011, pp. 447–462.

[86] Shun Tobiyama, Yukiko Yamaguchi, Hajime Shimada, Tomonori Ikuse, and Takeshi Yagi, *Malware detection with deep neural network using process behavior*, Computer Software and Applications Conference (COMPSAC), 2016 IEEE 40th Annual, vol. 2, IEEE, 2016, pp. 577–582.

[87] Petri Törönen, Mikko Kolehmainen, Garry Wong, and Eero Castren, *Analysis of gene expression data using self-organizing maps*, FEBS letters **451** (1999), no. 2, 142–146.

[88] Cross Validated, *How to choose the number of hidden layers and nodes in a feedforward neural network?*, `https://stats.stackexchange.com/questions/181/1097`, 2017.

[89] Victor Powell, *Principal component analysis*, `http://setosa.io/ev/principal-component-analysis/`.

[90] Nayer Wanas, Gasser Auda, Mohamed S Kamel, and FAKF Karray, *On the optimal number of hidden nodes in a neural network*, Electrical and Computer Engineering, 1998. IEEE Canadian Conference on, vol. 2, IEEE, 1998, pp. 918–921.

[91] Wei Wang, Yan Huang, Yizhou Wang, and Liang Wang, *Generalized autoencoder: A neural network framework for dimensionality reduction*, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2014, pp. 490–497.

[92] Zi Wang, Juecong Cai, Sihua Cheng, and Wenjia Li, *Droiddeeplearner: Identifying android malware using deep learning*, Sarnoff Symposium, 2016 IEEE 37th, IEEE, 2016, pp. 160–165.

[93] whatIs.com, *Transition Ads*, `http://whatis.techtarget.com/definition/transition-ad`, 2006, (Accessed date September 2016).

[94] Colin Whittaker, Brian Ryner, and Marria Nazif, *Large-scale automatic classification of phishing pages.*, NDSS, vol. 10, 2010, p. 2010.

[95] Wikipedia, *Activation function*, `https://en.wikipedia.org/wiki/Activation_function`.

[96] Wikipedia, *Phishing*, `https://en.wikipedia.org/wiki/Phishing`, 2016.

[97] Wikipedia, *Recurrent neural networks (rnns)*, `https://en.wikipedia.org/wiki/Recurrent_neural_network`, 2016.

[98] Li Xu, Zhenxin Zhan, Shouhuai Xu, and Keying Ye, *Cross-layer detection of malicious websites*, Proceedings of the third ACM conference on Data and application security and privacy, ACM, 2013, pp. 141–152.

[99] Shuxiang Xu and Ling Chen, *A novel approach for determining the optimal number of hidden layer neurons for fnns and its application in data mining*, (2008).

[100] Yanfang Ye, Lingwei Chen, Shifu Hou, William Hardy, and Xin Li, *Deepam: a heterogeneous deep learning framework for intelligent malware detection*, Knowledge and Information Systems (2017), 1–21.

[101] Tiliang Zhang, Hua Zhang, and Fei Gao, *A malicious advertising detection scheme based on the depth of url strategy*, Computational Intelligence and Design (ISCID), 2013 Sixth International Symposium on, vol. 2, IEEE, 2013, pp. 57–60.

[102] Shusen Zhou, Qingcai Chen, and Xiaolong Wang, *Active deep networks for semi-supervised sentiment classification*, Proceedings of the 23rd International Conference on Computational Linguistics: Posters, Association for Computational Linguistics, 2010, pp. 1515–1523.

Vita

Candidate's full name: Xichen Zhang

Master of Chemical Engineering, China National Pulp and Paper Research Institute, Beijing, China, 2013.
Bachelor of Chemical Engineering, Changsha University of Science and Technology, Changsha, China, 2010.

Publications:

- Xichen Zhang, Arash Habibi Lashkari, Ali A. Ghorbani. "A Lightweight Online Advertising Classification System Using Lexical-based Features". SECRYPT17, Madrid, Spain. July 25th, 2017.
- Xichen Zhang, Arash Habibi Lashkari, Ali A. Ghorbani. "Visualizing Advertisement URLs Using Deep Learning". Under review.