THE APPLICATIONS OF MACHINE LEARNING TECHNIQUES IN NETWORKED SYSTEMS

by

SOHEIL JAMSHIDI

A DISSERTATION

Presented to the Department of Computer and Information Science
and the Graduate School of the University of Oregon
in partial fulfillment of the requirements
for the degree of
Doctor of Philosophy

September 2020

DISSERTATION APPROVAL PAGE

Student: Soheil Jamshidi

Title: The Applications of Machine Learning Techniques in Networked Systems

This dissertation has been accepted and approved in partial fulfillment of the requirements for the Doctor of Philosophy degree in the Department of Computer and Information Science by:

Prof. Reza Rejaie	Chair
Prof. Ramakrishnan Durairajan	Core Member
Prof. Jun Li	Core Member
Prof. Daniel Lowd	Core Member
Prof. Raghuveer Parthasarathy	Institutional Representative

and

Kate Mondloch	Interim Vice Provost and Dean of the Graduate School

Original approval signatures are on file with the University of Oregon Graduate School.

Degree awarded September 2020

© 2020 Soheil Jamshidi
This work is licensed under a Creative Commons
Attribution 4.0 License.

DISSERTATION ABSTRACT

Soheil Jamshidi

Doctor of Philosophy

Department of Computer and Information Science

September 2020

Title: The Applications of Machine Learning Techniques in Networked Systems

Many large networked systems ranging from the Internet to ones deployed atop the Internet (e.g., Amazon) play critical roles in our daily lives. In these systems, individual nodes (e.g., a computer) establish a physical or virtual connection/relationship to form a networked system and exchange data. An important task in these systems is the timely and accurate detection of security or management events, e.g. a denial of service attack on campus. Machine learning (ML) models offer a promising data-driven method to learn the "signature" of these events from the past instances and use that to detect future events. While ML models have been very successful in other domains (e.g., image processing), there are clear challenges in using them for event detection in networked systems including (i) limited availability of large scale labeled dataset, (ii) subtle and changing signature of target event, (iii) selecting and capturing proper traffic features for (re)training, (iv) "black-box" nature of ML models.

This dissertation presents three different applications of ML models for event detection based on exchanged messages in networked systems that tackle the above challenges. First, we develop an ML-based method to identify incentivized Amazon reviews. To this end, we present a heuristic-based signature to identify explicitly incentivized reviews (EIRs) and characterize related reviews, products, and reviewers. We use EIRs to train an ML model for detecting implicitly incentivized reviews. Second, we examine how casting and training strategies of unsupervised ML (and statistical) model affects their accuracy and overhead (and thus feasibility) for forecasting network data streams. In particular, we study the impact of the size, selection, and recency of the training data on accuracy and overhead. Third, we design and evaluate anomaly detection mechanisms based on an unsupervised ML-based method that takes input data streams from network traffic, end-system, and application load. Furthermore, we leverage model interpretation to identify the most important input data streams and deploy model extraction to infer the rules that represent model behavior. Overall, these three cases studies result in numerous insightful findings on a range of practical issues that arise in deploying ML models for event detection in networked systems.

CURRICULUM VITAE

NAME OF AUTHOR: Soheil Jamshidi

GRADUATE AND UNDERGRADUATE SCHOOLS ATTENDED:

University of Oregon, Eugene, OR, USA
University of Tehran, Tehran, Iran
Isfahan University of Technology, Isfahan, Iran

DEGREES AWARDED:

Doctor of Philosophy, Computer and Information Science, 2020, University of Oregon
Masters of Science, Information Technology Engineering, 2013, University of Tehran
Bachelor of Science, Information Technology Engineering, 2010, Isfahan University of Technology

AREAS OF SPECIAL INTEREST:

Machine Learning based Data Analytic
Timeseries Forecasting
Network Traffic Analysis

PROFESSIONAL EXPERIENCE:

Graduate Research Assistant, University of Oregon, Eugene, OR, USA, 2014-2020

Data Scientist (Intern): Cambia Health Solutions, Portland, OR, 2018

GRANTS, AWARDS AND HONORS:

The Conference on emerging Networking EXperiments and Technologies (CoNEXT) Travel Grant, 2019

Phillip Seeley Scholarship in Computer & Information Science University of Oregon, 2019

The Best Presentation Award, Graduate Forum and Security Day of Oregon, University of Oregon, 2017-18

The Best Graduate Teaching Fellow in Computer & Information Science University of Oregon, 2014

PUBLICATIONS:

Soheil Jamshidi, Zayd Hammoudeh, Ramakrishnan Durairajan, Daniel Lowd, Reza Rejaie, and Walter Willinger. "On the Practicality of Learning Modelsfor Network Telemetry." Network Traffic Measurement and Analysis Conference (TMA) (2020).

Reza Motamedi, Soheil Jamshidi, Reza Rejaie, and Walter Willinger. "Examining the evolution of the Twitter elite network." Social Network Analysis and Mining 10, no. 1, Springer, (2020).

Soheil Jamshidi, Reza Rejaie, and Jun Li. "Characterizing the dynamics and evolution of incentivized online reviews on Amazon." Social Network Analysis and Mining 9, no. 1, Springer, (2019).

Soheil Jamshidi, Reza Rejaie, and Jun Li. "Trojan Horses in Amazon's Castle: Understanding the Incentivized Online Reviews." International Conference on Advances in Social Networks Analysis and Mining (ASONAM), IEEE, (2018).

Soheil Jamshidi, Ali Torkamani, Jynelle Mellen, Malhar Jhaveri, Penny Pan, James Chung, and Hakan Kardes. "A hybrid health journey recommender system using electronic medical records." Workshop on Health Recommender Systems (HealthRecSys), (2018).

Soheil Jamshidi, and Mahmoud Reza Hashemi. "An efficient data enrichment scheme for fraud detection using social network analysis." International Symposium on Telecommunications (IST), IEEE, (2012).

ACKNOWLEDGEMENTS

I am thankful to my parents, my sister, and my wife for their limitless and unconditional love and support throughout my career. For taking on too many challenges selflessly to ensure I can grow and follow my dreams.

I thank my research advisor, collaborators, and committee members, and mentors, Prof. Reza Rejaie, Dr. Walter Willinger, Prof. Ramakrishnan Durairajan, Prof. Jun Li, Prof. Daniel Lowd, Prof. Raghuveer Parthasarathy, and Prof. Hank Childs, for their guidance and feedback in every step of my PhD. Finalizing my dissertation would not have been possible without each one of them.

I also gratefully acknowledge the support of Intel Corporation for giving access to the Intel AI DevCloud platform that was used to scale and speed up our analyses.

TABLE OF CONTENTS

Chapter Page

LIST OF FIGURES

Figure Page

Figure Page

Figure Page

Figure Page

LIST OF TABLES

Table Page

CHAPTER I

INTRODUCTION

During the past two decades, we have witnessed a significant increase in the scale and complexity of involved parties of networked systems, from online retail networks to computer networks. In computer networks, both design and analysis of the protocols increasingly demand capturing and understanding patterns in large scale multi-dimensional datasets. The increase in network access speed, the appearance of bandwidth-hungry applications (such as video streaming, and P2P file sharing), the ISPs' increased interest in precise user traffic profiling to offer tailored services, and a response to the enormous growth in the number of connected users and internet of things (IoT) devices are among the main reasons for such a demand.

Similarly, in the online retail stores, there has been a surge in the scale and complexity of the relationships between the involved parties including the customers and sellers. On one hand, we have the customers' dependence on the online ratings that guide them through the decision-making process, and on the other, we have the urge of sellers to improve their ranking in the retail store's search results to be among the top choices. This dynamic might not be constructive in all cases and therefore, demands scalable and automated approaches to identify misuse of the system that can affect customer's trust and experience.

The early generation of solutions in both cases has often relied on handcrafted, statistical techniques to identify desired patterns in different datasets solely based on known patterns. For example, port-based filtering was used to identify traffics related to a certain application in computer networks given that applications are associated with pre-defined port numbers, however, this was misleading for applications with a dynamic port assignment, such as P2P. Then, approaches evolve toward payload-based analysis and due to constraints of encrypted communications, flow-level characteristics of the traffic were explored. The scale of data and a higher level of the abstraction in the flow-level datasets compared to port- and payload-based methods set the stage for ML methods.

Recently, the prevalence of machine learning (ML) techniques with a proper fit for the mentioned challenges have emerged as a reasonable choice and led to growing deployments of these methods in design and evaluation of networked systems.

1.1 Dissertation Scope & Contributions

In this dissertation, we study the applications and challenges of utilizing machine learning techniques in a variety of networked systems. In a broad sense, this dissertation can be categorized into four main applications of ML techniques for (i) detection of incentivized online reviews, (ii) forecasting network data streams, (iii) anomaly detection based on modeling a network application behavior and explainability of utilized ML models and their impact in understanding the models and facilitating their practical usage. The following presents an overview of the main contributions of this dissertation.

1.1.1 Detecting Incentivized Online Reviews. During the past few years, online reviews have become the main source of decision making for online shoppers. Therefore, sellers have increasingly offered discounted or free products to selected reviewers of e-commerce platforms in exchange for their reviews. Such incentivized (and often very positive) reviews can improve the rating of a product which in turn sways other users' opinions about the product. Despite their importance, the prevalence, characteristics, and the influence of incentivized reviews in a major e-commerce platform have not been systematically and quantitatively studied. In this section, we examine the problem of detecting and characterizing incentivized reviews in two primary categories of Amazon products. We describe a new method to identify Explicitly Incentivized Reviews (EIRs) and then collect a few datasets to capture an extensive collection of EIRs along with their associated products and reviewers. We show that the key features of EIRs and normal reviews exhibit different characteristics. Furthermore, we illustrate how the prevalence of EIRs has evolved and been affected by Amazon's ban. Our examination of the temporal pattern of submitted reviews for sample products reveals promotional campaigns by the corresponding sellers and their effectiveness in attracting other users. We also demonstrate that a classifier that is trained by EIRs (without explicit keywords) and normal reviews can accurately detect other EIRs as well as implicitly incentivized reviews. Finally, we explore the current state of explicit reviews on Amazon. Overall, this analysis sheds insightful light on the impact of EIRs on Amazon products and users.

1.1.2 Forecasting Network Data Streams.

Today's data plane network telemetry systems enable network operators to capture fine-grained data streams of many different network traffic features (e.g., loss or flow arrival rate) at line rate. This capability facilitates data-driven approaches to network management and motivates leveraging either statistical or machine learning models (e.g., for forecasting network data streams) for automating various network management tasks. However, current studies on network automation-related problems are in general not concerned with issues that arise when deploying these models in practice (e.g., (re)training overhead).

In this part of the dissertation, we examine various training-related aspects that affect the accuracy and overhead (and thus feasibility) of both LSTM and SARIMA, two popular types of models used for forecasting real-world network data streams in telemetry systems. In particular, we study the impact of the size, choice, and recency of the training data on accuracy and overhead and explore using separate models for different segments of a data stream (e.g., per-hour models). Using two real-world data streams, we show that (i) per-hour LSTM models exhibit high accuracy after training with only 24 hours of data, (ii) the accuracy of LSTM models does not depend on the recency of the training data (i.e., no frequent (re)training is required), (iii) SARIMA models can have comparable or lower accuracy than LSTM models, and (iv) certain segments of the data streams are inherently more challenging to forecast than others. While the specific findings reported in this chapter depend on the considered data streams and specified models, we argue that irrespective of the data streams at hand, a similar examination of training-related aspects is needed before deploying any statistical or machine learning model in practice.

1.1.3 Modeling Network Application Behavior for Anomaly Detection .

There is a wide range of sources for anomalies in network traffic. Mis-configuration of network devices, hardware or software failure, and adversarial attempts. Among these sources, the distributed denial of service (DDoS) attacks have been widely a concern due to disproportionate damage they cause compared to the required resources to initiate the attack. As researchers attempt to detect earlier versions of the DDoS attacks such as ICMP, UDP, and SYN flooding, attackers adapt and utilize more sophisticated attacks in the application layer (Y. Xie & Yu, 2008). In the application layer, the web service is considered the most vulnerable application (Liao, Li, Kang, & Liu, 2015). Attacking web servers through abnormal type, rate, or sequence of requests is an example of such attacks (Jaafar, Abdullah, & Ismail, 2019). For example, Slowloris (*slowloris DDoS tool*, n.d.) overwhelms a web server by exhaustively starting new sessions and keeping them alive by sending sparse requests and therefore, while not sending too many requests, prevents the server from proper handling the incoming requests.

In this work, we consider 4 types of attack, three slow HTTP attacks (Shekyan, 2020) and a session flooding attack (*BoNeSi DDoS tool*, n.d.) against Apache web server given its popularity compared to other web servers (datanyze.com, 2020). Due to the differences in the mechanism of these attacks, they have a different footprint on the traffic attributes. We utilize three types of features including network, operation system, and application, to detect attacks in an unsupervised manner to have a practical and accurate system for anomaly detection. We further analyze the contribution of each of these feature sources through different model training strategies and model interpretation techniques.

1.2 Dissertation Outline

The remainder of this dissertation is organized as follows. We provide a background and overview of studies related to applications of machine learning in each of the above areas in Chapter II. Next, in Chapter III we characterize the incentivized online reviews and discuss how ML techniques can be utilized to differentiate incentivized reviews from normal ones in the light of Amazon's reaction to this phenomenon. Chapter IV presents our work on the application and challenges of forecasting network data streams using statistical and ML-based techniques, as well as the advantages and shortcomings of each of these techniques. We evaluate and characterize the performance of machine learning techniques to model a network application to detect anomaly in Chapter V. We conclude and summarize our contributions in Chapter VI.

CHAPTER II

RELATED WORK

In this chapter, we review a body of recent studies that leverage various ML techniques for the design and characterization of networked systems. Mainly, we group these studies by their target problem which serves as a common context and background for them. Within each group, we further categorize based on more specific themes when possible. For each cluster of studies, we focus on the formulation of the techniques for data analysis, related challenges, and opportunities. In particular, we focus on how domain knowledge has been used to customize their solution. We select the well-received, highly cited, and peer-reviewed papers in top tier venues.

We categorize the prior studies into 4 sub-domains as follows:

1. **Online Review Analysis**: We review studies that utilize machine learning models to process online reviews in online retail networks. Ranging from labeling the duplicate reviews as spam and using supervised learning techniques to detect spam reviews (Jindal & Liu, 2008), detection of behavioral abnormalities of reviewers (Lim, Nguyen, Jindal, Liu, & Lauw, 2010) and review quality and helpfulness (S.-M. Kim, Pantel, Chklovski, & Pennacchiotti, 2006; J. Liu, Cao, Lin, Huang, & Zhou, 2007; Mudambi, 2010).

2. **Network Traffic Prediction**: In this section, we review studies in two groups: we review studies that leverage time-series analysis on network traffic data to forecast the traffic in the next time slot(s) along with studies with new ideas to obtain improved results. In the first group, studies focus on traffic prediction in mobile networks given their limited resources and increasing demand using Neural network methods (Nikravesh, Ajila, Lung,

& Ding, 2016) and unsupervised learning (Zang, Ni, Feng, Cui, & Ding, 2015). In addition, this part also includes capturing spatio-temporal dynamics (X. Wang, Zhou, Yang, Liu, & Peng, 2017), using network traffic matrix (Azzouni & Pujolle, 2017), considering variations of LSTM (Vinayakumar, Soman, & Poornachandran, 2017), and predicting TCP output (Mirza, Sommers, Barford, & Zhu, 2010). In the second group, improving LSTM by employing a random connectivity trick (Hua et al., 2017), decomposing time-series (Dai, Fu, Lin, Li, & Wang, 2017), analyzing the prediction uncertainty factors (Zhu & Laptev, 2017), transforming the 1D traffic to 2D matrices and then applying the CNN (Z. Wang & Oates, 2015a, 2015b), encoding traffic to an image (Ma et al., 2017), and utilizing statistical methods (Hatami, Gavet, & Debayle, 2018; Z. Wang & Oates, 2015a, 2015b) are covered.

3. **Anomaly Detection**: In this section, we focus on studies that leverage machine learning-based anomaly detection for attacks in the application layer. Anomaly detection methods have a wide range of flavors, such as detection of anomaly without prior context (Carter, Lippmann, & Boyer, 2010), statistical methods to detect anomalies (Himura, Fukuda, Cho, & Esaki, 2009) scaleable anomaly detection in mobile networks (Casas & Vanerio, 2017), detection focusing on temporal traffic (Nevat et al., 2018), anomaly detection based on server profiling (Canini, Li, & Moore, 2009), and anomaly detection in the scale of wide-area networks (WAN) (Aqil et al., 2017), just to name a few.

4. **Model Explainability**: Sommer et al. (Sommer & Paxson, 2010) consider interpretability as one of the reasons that ML techniques are not widely utilized in practice in networking domain. Given the needs on that end, in addition to the above three sections, we also review the techniques for ML

model explainability by leveraging model interpretability and extraction techniques. We review what these techniques are and how they can be utilized to improve the quality and practicality of models that are used in the networked systems.

The rest of this chapter is organized as follows: We will have a separate section for each of the aforementioned groups and we conclude this chapter in §VI by summarizing the studies and the pros and cons of different approaches along with open research questions that can be explored by the community.

2.1 Online Reviews Analysis

As online reviews and opinions become the main source of information about the quality of service and products and shape the users' shopping decisions, legitimacy, and possible bias in the online reviews systems raised concerns.

Detection and analysis of spam reviews started in 2008 by labeling the (near) duplicate reviews as spam and using supervised learning techniques to detect spam reviews (Jindal & Liu, 2008). Since then, different aspects of online reviews have been investigated such as behavioral abnormalities of reviewers (Lim et al., 2010) and review quality and helpfulness (S.-M. Kim et al., 2006; J. Liu et al., 2007; Mudambi, 2010). As online reviews become more popular and sophisticated, other aspects of reviews have been examined such as detecting sarcastic sentences on Twitter and Amazon (Davidov, Tsur, & Rappoport, 2010). they focused on recognition of sarcastic sentences In Twitter and Amazon datasets which can directly affect the accuracy of sentiment detection techniques.

Studies on spam detection have deployed a diverse set of techniques. Early studies relied on unexpected class association rules (Jindal, Liu, & Lim, 2010) and standard word and part of speech n-gram features with supervised learning (Ott, Choi, Cardie, & Hancock, 2011) that are later improved by using a more diverse feature sets (F. Li, Huang, Yang, & Zhu, 2011). *FraudEagle* (Akoglu, Chandy, & Faloutsos, 2013) was proposed as a scalable and unsupervised framework that formulates opinion fraud as a network classification problem on a signed network of software product reviews of an app store. These studies also relied on different strategies, such as Amazon Mechanical Turk (Ott et al., 2011) or manual labeling (F. Li et al., 2011) to create a labeled dataset for their analysis.

The effect of incentives on reviewers and quality of reviews is studied by Qiao et al. (Qiao, Lee, Whinston, & Wei, 2017). They showed that external incentives might implicitly shift an individual's decision-making context from a pro-social environment to an incentive-based environment. This could also lead to a long-term change in the reviewer's behavior even after the incentives disappear. Also, a long-term change in reviewers' behavior was reported because as a reviewer is given some incentives, can face persistent mindset change that affects her future behaviors, even in the absence of the incentives. They concluded that the wide usage of incentives might be harmful to the platform in the long-term and might discourage reviewers' pro-social behaviors.

Wang et al. (J. Wang, Ghose, & Ipeirotis, 2012) modeled the impact of bonus rewards, sponsorship disclosure, and choice freedom on the quality of paid reviews. In a qualitative study, Petrescu et al. (Petrescu, O'Leary, Goldring, & Mrad, 2017) examined the motivations behind incentivized reviews as well as the relationship between incentivized reviews and satisfaction ratings assigned by consumers to a product. They showed that the level of user engagement depends on a cost-benefit analysis. Burtch et al. (Burtch, Hong, Bapna, & Griskevicius, 2017) focused on social norms instead of financial incentives. By informing individuals about the volume of reviews authored by peers, they test the impact of financial, social norms, and a combination of both incentives in motivating reviewers. The study by Xie (Z. Xie & Zhu, 2015) unveiled the underground market for app promotion and statistically analyzed the promotion incentives, characteristics of promoted apps and suspicious reviewers in multiple app review services.

As mentioned, incentivized reviews can add bias to the recommendation and online review systems. *Bias in recommendation systems* is studied from different perspectives. In (Shyong, Frankowski, & Riedl, 2006) the effect of biased reviews and recommendations on user behaviors and decision making is discussed, and related privacy and security concerns are outlined.

In addition to external factors such as biased reviews, internal systematic bias is also a source of concern. Australian Uber drivers accused the company of slowly decreasing their ratings to suspend drivers and then charge higher commissions to be reinstated (Businessinsider, 2017). The other algorithmic rating systems such as Yelp (Journal, 2017) and Fandango (FiveThrityEight, 2017) have faced similar criticisms. Systematic bias is investigated in different domains, such as racial advertisements (Sandvig, Hamilton, Karahalios, & Langbort, 2014), biases of online maps in representing international borders (Soeller, Karahalios, Sandvig, & Wilson, 2016), gender bias in online advertising (Datta, Tschantz, & Datta, 2015), understanding the Uber surge pricing algorithm by emulating Uber accounts (L. Chen, Mislove, & Wilson, 2015), discovering racial discrimination against Black users on Airbnb via creating multiple accounts (Edelman, Luca, & Svirsky, 2017), and in online hotel booking websites (Eslami, Vaccaro, Karahalios, & Hamilton, 2017).

To the best of our knowledge, none of the prior studies have systematically examined the prevalence of incentivized online reviews, their basic characteristics, and their influence on the level of interest among other users to a product based on large-scale quantitative measurements in a major e-commerce platform.

2.2 Network Traffic Prediction

In this section, we review studies that leverage time-series analysis on network traffic data to forecast the traffic in the next time slot(s). The time-series are defined as data points with temporal ordering. Such data points are available in a wide range of domains such as weather (Soares, Costa Jr, Costa, & Leite, 2018), bio-medical and bio-metrics (Ferenti, 2017), financial (stock and exchange rates) (Moliner & Epifanio, 2019), industrial sensors (Mandal, Santhi, Sridhar, Vinolia, & Swaminathan, 2019), and also video and music streaming (B. Kim, Chang, Heo, & Shin, 2020), for which the time-series analysis is applicable. Time-series analysis mainly includes prediction/forecasting, classification, and clustering techniques.

The traffic prediction can be done for two main purposes: resource management (e.g. scaling up hardware resources when a large number of requests are expected for example on the registration day in a campus) and monitoring (e.g. anomaly detection to detect abnormal behavior in the network). Several methods for traffic prediction are proposed that can be grouped into two categories: linear and non-linear methods. For the linear methods, the most adopted method is AutoRegression Integrated Moving Average (ARIMA) (B. Zhou, He, & Sun, 2006) and for non-linear models Support Vector Regression (SVR) (Castro-Neto, Jeong, Jeong, & Han, 2009) is widely used. While ARIMA is claimed to have limitations to capture the rapid variations due to its tendency on the mean values of past series data, SVR's main shortcoming is the lack of the structure to select the key parameters of the model. That is why a new trend on Neural Networks has been developed (Azzouni & Pujolle, 2017; Nikravesh et al., 2016).

2.2.1 Network Traffic Forecasting. Nikravesh et al. (Nikravesh et al., 2016) focus on the importance of resource management for the mobile network providers as the number of subscribers is increasing, and efficiency has become a must-have quality. They apply data analysis techniques to predict the future behavior of mobile network traffic and support network operators to maximize resource usage; preventing both under-provisioning and over-provisioning. They employ a real-life dataset from a commercial trial mobile network composed of a million rows and 27 features, each row representing aggregated (per hour) traffic of one specific cell in the network from 6K different wireless network cells. Although, for their experiments, they only focused on one network cell with the most data points and applied feature selection to exclude non-correlating features, ended up using only 168 data points and 24 features. The problem was formulated as a supervised regression. Their experiment has two folds: 1) to show if the values of attributes can be used to predict the value of a single unknown attribute (as target class), and 2) if prior values of each attribute can be used to predict its next step's value. They compare the accuracy of a fully-connected Neural network in the prediction of future behavior of mobile networks (number of active pieces of equipment in downlink), compare to SVM and Multi-Layer Perceptron with Weight Decay (MLPWD). They show that MLPWD with a sliding window is the best choice if the traffic data is uni-dimensional. Otherwise, SVM is a better option.

Nie et al. (Nie, Jiang, Yu, & Song, 2017) propose a deep belief network and Gaussian model-based traffic prediction. The proposed method first adopts a discrete wavelet transform to extract the low-pass component of network traffic, describing the long-range dependence of traffic. The low-pass component is modeled by learning a deep belief network. The rest high-pass component that expresses the fluctuations of network traffic, is modeled by a Gaussian model. The maximum likelihood method is used for the estimation of the parameters of the Gaussian model. Their dataset is not described in detail, but they use the first 2000 data points for training and 16 data points for tests. They compare their model with two older studies (principal component analysis (PCA) method (Soule et al., 2005) and the Tomogravity method (Y. Zhang, Roughan, Duffield, & Greenberg, 2003)), as well as the Sparsity Regularized Matrix Factorization method (SRMF) by Roughan et al. (Roughan, Zhang, Willinger, & Qiu, 2012). Their proposed prediction method outperforms three existing methods based on spatial and temporal relative errors (SRE, TRE).

Zang et al. (Zang et al., 2015) propose a framework for cellular traffic prediction by pre-processing the time-series using wavelet transformation. First, they apply K-means with the Euclidean distance between base stations as the distance metric, to identify geographically correlated base stations. Then, they apply the wavelet transformation to obtain the low and high-frequency components. They utilize Elman-NN to predict each of these four components and then using them, reconstruct the predicted frequency. Their goal is to predict the next hour's traffic value. Real-world traffic data measurement is used to test their framework. Having data from 358 BSs at a particular district in a metropolitan, that each of the BSs records the volumes of GPRS flows over time (hourly). After K-means clustering, the shape of traffic becomes a 50x168 matrix with 144 columns for training and the last 24 columns for prediction and testing. For evaluation, Normalized Mean Square Error (NMSE), Normalized Mean Absolute

Error (NMAE) and Mean Absolute Percentage Error (MAPE) was utilized. They discuss the superiority of wavelength transformation over traditional methods.

In another study on cellular traffic prediction, Wang et al. (X. Wang et al., 2017) capture the spatial-temporal dynamics of cellular traffic by in-cell and inter-cell decomposition using a graph-based deep learning approach. For example, in a transit station, the inter-cell traffic surges at a particular time while in other locations the in-cell traffic can easily dominate. They employ GNN toolkit to implement their experiments and compared the results with LSTM, ARIMA, and NAIVE (that predicts the traffic at time t, based on the traffic at time t of the last day). They study the characteristics of urban cellular traffic with large-scale cellular data usage dataset covering 1.5 million users and 5,929 cell towers in a major city of China. Their dataset has the flow-level data per user and covers the app-ID and device-ID for 14 days. They use the first 12 days data (aggregated every half an hour) for training and the last two days for testing. Evaluation is performed using the Mean Absolute Error (MAE) and Mean Absolute Relative Error (MARE). Based on their experiments, they conclude that spatial dependency and the interaction of spatial and temporal factors play an important role in accurate and robust prediction.

Cortez et al. (Cortez, Rio, Rocha, & Sousa, 2012) compare the time-series methods (Holt-Winters and ARIMA) and neural network methods for Internet traffic prediction. Their result shows that the NN methods achieve the best results for short-term (5 minutes and hourly) traffic prediction while the Holts-Winters methods were more accurate in predicting daily traffic.

Azzouni et al. (Azzouni & Pujolle, 2017) consider the Network Traffic Matrix (TM) prediction to estimate the future network traffic from the previous network traffic data using Long Short-Term Memory (LSTM) models. Considering the linear models such as ARMA (Autoregressive Moving Average), ARIMA (Autoregressive Integrated Moving Average), ARAR (Autoregressive Autoregressive) and HW (HoltWinters) algorithm, they compare the results with nonlinear time series prediction with neural networks. They incorporate real traffic data provided by GEANT[1] backbone networks, which is the pan-European research network. Using the 15 minutes intervals over three days, they have 309 matrices of network traffic. Using the traffic matrix of N nodes in the network (NxN) over time T (NxNxT), they concatenate rows from top to bottom to create a vector of size (NxN) per time t. A learning window is used to avoid high computational complexity (as the total number of time slots becomes too big over time). Mean Square Error (MSE) is their evaluation metric for the prediction accuracy. MSE is a scale-dependent metric that quantifies the difference between the forecasted values and the actual values. They discuss the MSE over a different number of hidden layers and hidden units.

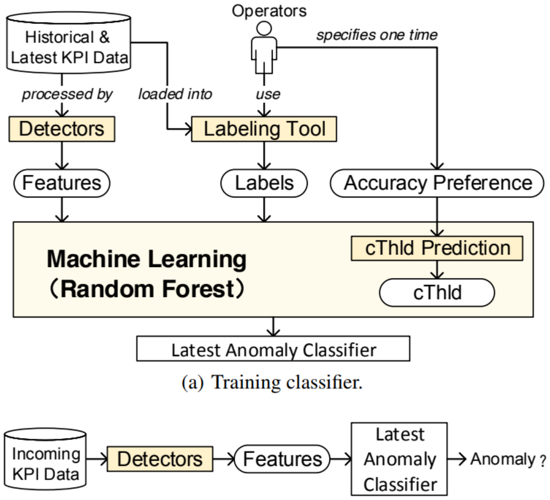
Given the wide usage of RNNs for traffic prediction, Vinayakumar et al. (Vinayakumar et al., 2017) analyze the variation of Recurrent Neural Networks (RNNs) to obtain the optimum flavor along with optimal parameters. Considering different layer numbers, hidden units, and learning rates, on multiple flavors of RNNs, they propose an architecture of traffic matrix with a sliding window on layers of stacked LSTM. Choosing the parameters to be five layers with 500 units and a learning rate of 0.1. They use the GEANT public dataset for their experiments.

[1] https://www.geant.org/Projects/GEANT_Project_GN4/

In a study on prediction of TCP throughput, Mizana et al. (Mirza et al., 2010) leverage Support Vector Regression (SVR) to predict the TCP throughput based on basic path characteristics including available bandwidth (AB), queuing delays (Q), and packet loss (L). They experiment with passive measurements with the multi-configuration testbed. Comparing the accuracy of SVR to the exponentially weighted moving average (EWMA) History-Based Predictor (HB) reveals that for bulk transfers in heavy traffic, TCP throughput is predicted within 10% of the actual value for 87% of the time. Overall, their results suggest approximately a 60% improvement over history-based methods with a much lower impact on end-to-end paths. They use the *relative prediction error* as evaluation metric, which is the predicted throughput (R1) minus actual throughput (R) over a minimum of R1 and R. They conclude that while *AB* feature is not necessary for accurate prediction, a combination of queuing delays (Q), and packet loss (L) is sufficient.

There are a large number of studies that leverage ML techniques or statistical approaches to provide monitoring and controlling services for network administrators. However, very small number of such methods are used in the real world scenarios. Liu et al. (D. Liu et al., 2015) explore the root causes and report that requiring parameter tuning in a manual and iterative way and configuration of thresholds are among the main reasons that offered monitoring services are not practical. This paper tackles this challenge by *Opprentice* (**Op**erators' ap**prentice**), that operators' only manual work is to periodically label the anomalies in the performance data with a convenient tool.

The Opprentice architecture is shown in Fig. 1. In this architecture, multiple existing detectors are applied to the performance data in parallel to extract



(a) Training classifier.

(b) Detecting anomaly.

Figure 1. Opprentice Architecture (D. Liu et al., 2015)

anomaly features. The features and the labels are used to train a random forest classifier to automatically select the appropriate detector-parameter combinations and the thresholds. The system works with only 2 parameters and is robust in existence of noise. Recall and precision are used as accuracy criteria and reported to by above 66% for three different service KPIs in a top global search engine including search page view (PV), slow responses of search data centers (#SR), which is an important performance metric of the data centers, and search response time (SRT) that has a measurable impact on the users' search experience.

In a detailed study (Sommer & Paxson, 2010) (find slides here[2]) the reasons behind the limited application of ML-based approaches for intrusion detection in large-scale and operational environments are discussed. However, such limitation in some cases can be applied to all of the discussed domains. They explain how the ML approaches are not suitable for the task of intrusion detection:

- lack of training data;

[2] Slides: HTTP://oakland10.cs.virginia.edu/slides/anomaly-Oakland.pdf

- a very high cost of errors;

- a semantic gap between results and their operational interpretation;

- enormous enormous variability in input data; and (v) fundamental difficulties for conducting the sound evaluation.

They have provided examples of why ML approaches work in some domains and why not for intrusion detection. For example, since the variation of attacks/anomalies are known in recommender systems or spamming, there will be enough training data on both (all) classes, therefore, ML approaches can be used effectively in the real world.

In terms of High cost of errors, they compare with Product recommendation systems, OCR technology, and Spam detection to make the point that in these cases the error of false positives/negatives are not as vital as it is in the intrusion detection domain. The semantic gap refers to the fact that in the intrusion detection domain, operational expectations are more than what a current ML technique can deliver. Instead of a large number of false positives that only mark unseen behaviors (that can be legitimate or attack), it is expected that the results be interpreted for the operators. The local and internal policies should also be considered in addressing the semantic gap. Addressing such policies as well as vague guidelines described by an imprecise legal language are the main issues. In terms of Diversity of Network Traffic, as network characteristics are variable over short time ranges (seconds to hours), it will be hard to predict them. Aggregating network traffic (time of the day, the day of the week) can be a better solution and work better.

The network traffic prediction is not considered as a trivial task, hence, various techniques (traffic matrix, spatial and temporal separation) and a wide range of models (CNN, LSTM, NN, ARIMA, HW, SVR) are applied on the network traffic data in order to predict the future behavior of the network accurately. However, the effectiveness of each of these methods and specially comparison of statistical methods and ML-based methods is not systematically explored. In addition, the practical implications and challenges of proposed methods are not widely investigated.

2.2.2 Ideas to Improve the Prediction.

In addition to traditional studies that address a challenge in the system by forecasting the network traffic and explain how ML methods can lead to more accurate models, there are studies that are inspired by tricks and concepts from other domains and their goal is to improve the forecasting performance. In this subsection, we review such ideas.

Inspired by the idea of Convolutional Neural Networks (CNN) with a sparse neural connection that shows comparable performance compared to the conventional CNNs, Hau et al. (Hua et al., 2017) propose Random Connectivity LSTM (RCLSTM) that contains fewer parameters (35% less neural connections) and show competitive performance. In their model, they initialize a random graph, where neurons connections are established independently in a stochastic manner with a probability of p. The connection is established if $p[i \rightarrow j] \geq T$ where threshold T indicates the sparsity of neural connectivity. Their experiments are based on GEANT backbone network traffic data from 75 workdays, normalized by average over standard deviation. They compare the effect of the number of training samples and length of input traffic (range of 10 to 55) on MSE and MAE accuracy measures, reporting accurate models on larger and longer input traffic.

De-trending is a well-known method to decompose the time series for statistical analysis. Recently, Dai et al. (Dai et al., 2017) propose that de-trending can improve the performance of LSTM models. Therefore, they decompose the traffic into a trend (capturing the fixed temporal pattern) and residual (used for prediction) series. They propose *DeepTrend* as a deep hierarchical neural network. Their model has two layers: a) a fully connected NN called extraction layer and b) an LSTM layer used to make flow prediction. To calculate the *trend*, they calculate the average of the periodic traffic flow time series collected in the same station and subtract it from the actual time-series. Their dataset is collected from 3.9K stations every 5 min in district 4 of freeway systems across California. Using the first 12 weeks for training and four weeks for testing. They consider stations with less than 1% loss and also normalized the data per station to be 0 mean and 1 standard deviation. Using mean square error (MSE) and mean absolute error (MAE). DeepTrend was able to model the time-series more accurate compared to LSTM models without detrending.

The prediction uncertainty is essential for assessing how much to trust the forecast produced by the model and has a profound impact on anomaly detection. Zhu and Laptev (Zhu & Laptev, 2017) focus on estimating the uncertainty in time-series predictions using neural networks on Uber data. They mention that focusing on Bayesian Neural Networks; the prediction uncertainty can be decomposed into three types: model uncertainty, inherent noise, and the model misspecification. For each, they have proposed a way for calculation for example: For the model uncertainty, given a new input x, they compute the neural network output with stochastic dropouts at each layer. That is, randomly dropout each hidden unit with a certain probability p. This stochastic feed-forward is repeated B times.

Then, the model uncertainty can be approximated by the sample variance. The model misspecification deals with the uncertainty when predicting unseen samples with very different patterns from the training dataset. To address that, they use an encoder-decoder. Using the encoder, they extract the representative features from a time series (decoder will, later on, reconstruct the time series from the encoded space). Then, at test time they measure how close is the encoding of the test samples to the training samples. Using the trip data from Uber on eight representative large US cities with three years for training and the following four months for validation and eight months for testing. They log-scales the data, removed the first-day value from the rest (to remove the trend), and fed the data to four models (including a two-layer stacked LSTM with 128 and 32 units and tanh activation). They use the sliding window with 1-day step size and the Symmetric Mean Absolute Percentage Error (SMAPE) as the performance metric.

Given the proven performance of Convolutional Neural Networks (CNNs) in the image classification domain, researchers considered different approaches to convert time series to images and benefit from CNNs. However, there are a small number of studies related to this approach in the network domain.

Using the 1D time-series signals as the input of a modified CNN architecture, transforming the 1D to 2D matrices and then applying the CNN (Z. Wang & Oates, 2015a, 2015b), and also using multiple CNNs and using a fully connected NN to leverage multiple features are among the most popular approaches.

Focusing on road traffic, Ma et al. (Ma et al., 2017) present a method of predicting network-wide traffic based on encoding a day of traffic as an image. This encoding maps time to the x-axis, sampled network locations to the y-axis, and represents the traffic at a given (location, time)-point as a single-dimensional “color" value. A convolutional neural network is then trained over previous images and used to predict future network behavior based on past network behavior. A comparison with other machine learning methods is performed using traffic data collected from taxis on two subnetworks of the Beijing road system demonstrating an improved accuracy of up to 43% within acceptable execution time.

Encoding of time-series signals to images using Gramian Angular Fields (GAF) and Markov Transition Fields (MTF) and then using a tiled CNN is used by Wang et al. on multiple datasets (Z. Wang & Oates, 2015a, 2015b). Hatami et al. (Hatami et al., 2018) propose to use the Recurrence Plots (RP) to transform the time-series into 2D texture images and then take advantage of a deep CNN classifier (with two hidden layers followed by a fully connected layer). Results suggest a boost in performance of Time-series classification rate.

Traffic prediction in mobile networks is necessary as the number of clients is increasing over time and resource management becomes more challenging. We reviewed the applications of ML methods on network traffic prediction as well as ideas on how to improve the accuracy of utilized methods. As mentioned, network traffic prediction is a complex question to be answered, hence, various techniques (traffic matrix, spatial and temporal separation) and a wide range of models (CNN, LSTM, NN, ARIMA, HW, SVR) applied on the network traffic data in order to predict the future behavior of the network accurately. To improve the accuracy, novel ideas from other domains, including randomized connection in LSTM, 2D data structure, and conversion to image and benefiting from CNN models are utilized.

It is worth noting that the prior studies have primarily focused on the "off-line" evaluation of forecasting method on network data streams with per-minute (or coarser) resolution due to the limited availability of representative streams with per-second granularity. Therefore, the characteristics of their data streams and assumptions were very different that what the model will be exposed to in real world. More importantly, to our knowledge, none of the prior studies have explored the effect of the following practical issues on incorporating forecasting model into telemetry systems using network data streams with different characteristics, (i) the variations in characteristics of different segments of the data stream that is often observed in network data streams, and (ii) the effect of volume, selection, and recency of training data on different models.

2.3 Anomaly Detection

The ML techniques have a wide variety of applications in the network security domain. In this section, the advantages of using ML methods in anomaly detection and security of network systems is discussed.

Deviation from normal behavior can be a strong signal for detecting attacks in networks. Different techniques have been proposed to capture the normal behavior of a system (Rasti, Magharei, Rejaie, & Willinger, 2010). Anomalies can also emerge as a result of routing issues or hardware failure. In all of these case, network providers are interested in the detection of anomalies in order to minimize the resulting damage.

Carter et al. (Carter et al., 2010) present a method of detecting anomalous network activities without providing any historical context, using hierarchical clustering. However, it is an O(N^2) operation, meaning that it will not purely scale to large N. This is why they operate their system on a per-service basis, focusing on port 80 in this study. They also used a decision tree to identify the common theme of the detected clusters. To this aim, they labeled data for each cluster as a positive label and all others as negative. DT was able to find the rule mostly based on byte ratio (byte per packet), IP ratio (flow/unique source IP), and outgoing bytes/packet. They experiment and report accurate results on a window of 5 minutes Netflow on port 80 where the server was known to be under SYN flood DDoS attack in that period.

As mentioned by Himura et al. (Himura et al., 2009) the modeling and characterization of Internet traffic are essential for simulations, traffic classification, anomaly detection, and QoS. Focusing on quantifying host-based applications, they use a statistical method (multi-scale gamma model) to label anomalies produced in the network in the presence of Internet worms. Using real network traffic traces, they show that applications show consistent behavior over time, however, vary as bandwidth changes. Applications can be characterized using statistical parameters and be distinguished from anomalous behaviors.

Casas and Vanerio (Casas & Vanerio, 2017) leverage the big data analytics and platforms to perform the automatic detection of anomalies in mobile networks. Using Big-DAMA, a big data analytics framework for network monitoring, they apply the Super-learner model (Van der Laan, Polley, & Hubbard, 2007) (a loss-based ensemble-learning method that finds the optimal combination of a collection of base predictors). They note and tackle two main challenges for the application of ML in anomaly detection: 1) nearly real-time expectation 2) model selection. The first issue is tackled using the Big-DAMA framework, which uses *Spark* for batch (pre-) processing, and *Cassandra* for storage. The benefit of using such a framework is that it is distributed, with no single point of failure, it will be fast and scalable, with the ability to handle unstructured data. To tackle the second issue, they overview the ensemble techniques (bagging, boosting, and stacking) and argue that ensembling can address the issue of selecting the best predictor as we can benefit from multiple predictors. They apply Super learner as a stacking learning algorithm and compare the results with individual ML methods such as decision tree, naive bayes, neural network, SVM, and k-NN, showing the superiority of ensembling models.

Nevat et al. (Nevat et al., 2018) consider the temporally correlated traffic and apply anomaly detection methods. Their dataset is 75-minute TCP traffic of 10th December 2014 from MAWI repository. A statistical method (Markov chain) was used for detection. Their goal is anomaly detection in temporally correlated traffic. They formulate the problem as the optimal statistical test, known as the Likelihood Ratio Test (LRT), using the Cross-Entropy (CE) method. As a result, not only it finds the anomaly but also finds the subset of flows causing it.

In another study, Marnerides et al. (Marnerides, Pezaros, Kim, & Hutchison, 2009) propose a measurement-based classification framework that exploits unsupervised learning to categorize network anomalies in specific classes accurately. They introduce the combinatorial use of two-class and multi-class unsupervised Support Vector Machines (SVM) to first distinguish normal from anomalous traffic and to further classify the latter category to individual groups depending on the nature of the anomaly, respectively. The features they have used are as follows:

- Initial flow classification is based on seven packet header characteristics; the IP source/destination addresses, the transport protocol, the transport source/destination ports, the mean packet inter-arrival time and the size of the first ten packets

– During the second stage, only four packet header features are used; the source/destination IP addresses and transport source/destination ports.

Anomaly detection and QoS customization require to know the underlying traffic types. Canini (Canini et al., 2009) uses flow features to create server profiles and then identifies potential proxies within the observed servers. Their methodology consists of four stages: 1) service identification to identify HTTP and HTTPS services, 2) server profiling - the features are based on the packets' inter-arrival times and payload sizes. A profile consists of the average and standard deviation of this features-, 3) proxy identification using K-means as an unsupervised technique, and 4) host cache management, which is a practical need of storing, updating, and deleting service profiles. In this visionary paper, they share preliminary results of experiments on two proxies: `guardster.com` and `anonymouse.org`, where they recorded the traffic to browse a dozen of popular websites using the direct connection and through the proxies, reaching a total of 81 servers. Results show that their method can profile and distinguish among these servers accurately.

Aqil et al. (Aqil et al., 2017) consider the network intrusion detection at the scale of a wide area network (WAN) using the ML methods. First, they focus on creating packet summaries that are concise, but sufficient to draw highly accurate inferences. Then, they transform traditional IDS rules to handle summaries instead of raw packets. Using the network traces from MAWI group, they inject five different network attacks such as SYN flooding and extensive port scans at a rate of less than 10% of benign traffic. Showing that Jaal reduces overheads by over 65% compared to sending raw packets while achieving a detection accuracy of over 98% in ISP scale.

One of the most important categories of anomalies in the network domain is the application layer DDoS attack. Detection of such attacks is considered to be more complex and challenging in the network security domain compare to other types of anomalies. Hoque et al. (Hoque, Bhattacharyya, & Kalita, 2015) discuss the followings as the main reasons: 1) *obscurity* as they can be performed using a combination of TCP and UDP protocols, 2) *efficiency* as a small number of connections is required to perform the attack, and 3) *lethality* since the attack can overwhelm a web server immediately, regardless of the type of hardware and its performance. Studies on the application layer and mainly HTTP DDoS attacks can be grouped into the followings categories based on their method as Liang et al. (Jaafar et al., 2019) suggest:

- Session/Request flooding attack. resources of a server become exhausted when session request rates get higher than the usual expected rate. Examples are HTTP GET/POST flooding attack that requires a botnet to be successfully initiated. in case of request flooding attack, the attacker initiates a large number of requests in one session. The number of requests per session is higher than the usual number of expected requests. The HTTP GET/POST session is an instance of attack in this category that takes advantage of the HTTP 1.1 feature, which allows more than one request within a single HTTP session.

- Slow Request/Response Attack. An attacker sends a high workload of requests to initiate an attack in the form of a session. The attacker partially sends HTTP requests that grow quickly and repeatedly. The server waits for each request to be properly completed but the attacker updates each request by additional requests slowly, and never closes the sessions.

Slow request/response attacks rely on the HTTP protocol. By design, HTTP requires requests to be completely received by the server before they are processed. Therefore, if an HTTP request is not complete (such as Slowloris or slow body attacks), or if the transfer rate is very low (slow read attack), then the server keeps its resources busy waiting for the rest of the data. In this case, the server's resources will be obtained quickly by the attacker without demanding too many requests to be sent, which creates a denial of service attack. There are three categories of such attacks:

Slowloris attack. that is an example of slow request attacks. The attacker opens many connections to the target web server and keeps the connections open by sending an incomplete sequence of requests to the server. Therefore, the server will keep these connections open, which will exhaust their maximum concurrent connection pool and eventually deny additional connection attempts from clients.

Slow POST (Slow Body) attack. in which the attacker sends legitimate HTTP POST headers to a Web server. In these headers, the sizes of the message body that will follow are correctly specified. However, the message body is sent at a painfully low speed. These speeds may be as slow as one byte every two minutes. Given that the message is in legitimate form, the targeted server will follow specified protocol rules. Therefore, the server will subsequently slow down the response rate. When the attacker launches hundreds of Slow POST attacks at the same time, server resources are rapidly consumed to the point that no legitimate connection can be served by the server.

Slow Read attack. that is very similar to Slowloris and slow POST, but instead of prolonging the request, it sends a legitimate HTTP request and reads the response slowly.

There is a range of tools available to generate each of these attacks. There are attack specific tools such as Slowloris tool(*slowloris DDoS tool*, n.d.) that only generates Slowloris attacks as well as frameworks to generate a range of attacks such as slow HTTP test tool (Shekyan, 2020) that can be used to generate all types of slow request and response attacks. The DDoS Botnet Simulator (BoNeSi (*BoNeSi DDoS tool*, n.d.)) is also a widely used tool that generates ICMP, UDP, and TCP (HTTP) flooding attacks.

While there is not a recent and publicly available dataset for different application layer DDoS attacks (Ndibwile, Govardhan, Okada, & Kadobayashi, 2015), there is a wide range of approaches for generating (Aborujilah & Musa, 2017; Bhatia, Benno, Esteban, Lakshman, & Grogan, 2019; Cao, Nejati, Balasubramanian, & Gandhi, 2019; Shiaeles & Papadaki, 2015), capturing (Celenk, Conley, Graham, & Willis, 2008; Ingham & Inoue, 2007), or using existing data sets (W. Lu & Ghorbani, 2008; Oza, Ross, Low, & Stamp, 2014) (such as DARPA dataset that is known in the security community for its flaws). The self-captured and -generated datasets utilized variety of tools available online, including: Web Record and replay (*Web Page Replay*, n.d.), LOIC (*Low Orbit Ion Cannon*, n.d.), Hulk (*Hulk DoS tool*, n.d.), RUDY (*R U Dead Yet (RUDY) DDoS tool*, n.d.), and BoNeSi (*BoNeSi DDoS tool*, n.d.), and slowloris (*slowloris DDoS tool*, n.d.). Publicly available web traces that can be used as normal background traffic is also used by several studies. Datasets including WorldCup98, KDD-CUP99 (*KDD CUP 99*, n.d.), DARPA 2009 (Lippmann, Haines, Fried, Korba, & Das, 2000), WorldCup98, ClarkNet, and NASA (Arlitt & Williamson, 1996), and CAIDA 2007 (*CAIDA DDoS 2007 Attack Dataset*, n.d.). Ring et al. (Ring, Wunderlich, Scheuring, Landes, & Hotho, 2019) provides an extensive summary of these datasets. However, none of these datasets or tools alone can provide the types of features or a wide range of attacks that were are looking for in our study.

In addition to common features such as number and size of packets, number and size of flows, and the number of unique sources and destinations, researchers have used a variety of other features including packet header (Hameed & Ali, 2018), popularity and repetition indices (Singh, Singh, & Kumar, 2018), packet counts per type and interval between two-page visit (Sreeram & Vuppala, 2019), packet rate (Hoque, Kashyap, & Bhattacharyya, 2017), HTTP GET count per connection or IP address (Johnson Singh, Thongam, & De, 2016), GeoIP, source MAC address, and a number of user agents (Shiaeles & Papadaki, 2015). However, none have used the combination of features from network, application, and resource usage to assess their contribution to the detection of different anomaly types.

A wide range of methods have been proposed to detect such attacks, Autoencoder models (Bhatia et al., 2019), reinforcement learning (Feng, Li, & Nguyen, 2020), PCA and ant-colony optimization methods (Fernandes Jr, Carvalho, Rodrigues, & Proença Jr, 2016), genetic Algorithm and fuzzy logic (Hamamoto, Carvalho, Sampaio, Abrão, & Proença Jr, 2018), ARIMA and Holt-Winter models (Jiang & Papavassiliou, 2006; Pena, de Assis, & Proença, 2013), wavelet approximation (W. Lu & Ghorbani, 2008), adaptive Wiener filtering and ARIMA Modeling(Celenk et al., 2008), to name a few.

These methods rely on differences in characteristics of normal and abnormal behavior and aimed to capture the deviation by PCA analysis (Fernandes Jr et al., 2016), relying on forecasting models and mark the deviation on the error as anomaly (Jiang & Papavassiliou, 2006) or classification of traffic based on a range of features into normal or attack (Feng et al., 2020). However, such models have practical challenges. Having an accurate forecasting model is an involved process and depends highly on the data and the model (Jamshidi et al., 2020a), classification relies on high quality labeled data that is not available for every network setting (Ring et al., 2019) and given the complexity of attacks some techniques such as PCA are known to be less effective in this domain (Brauckhoff, Salamatian, & May, 2009). Among the neural network-based unsupervised anomaly detection techniques, Autoencoders have received much more attention recently. As they have shown high performance in different applications such as noise reduction in audio (X. Lu, Tsao, Matsuda, & Hori, 2013) or images (Gondara, 2016) and collaborative filtering (S. Li, Kawale, & Fu, 2015), researchers in networking domain utilized the technique in IoT anomaly detection (Luo & Nagarajan, 2018), Web application anomaly detection (Mac et al., 2018; Vartouni, Teshnehlab, & Kashi, 2019; Yadav & Subramanian, 2016) and other network intrusion detection systems (Sakurada & Yairi, 2014; Xu et al., 2018; C. Zhou & Paffenroth, 2017).

However, none of the prior studies have considered the wide range of anomalies, all types of features, and assessment of feature importance on the detection rate that we are covering in our study.

2.4 Model Explainability

There is a large body of work on applications of machine learning models in networking, ranging from classification of traffic to forecasting and anomaly detection (Hoque et al., 2015; Jaafar et al., 2019,?; Y. Xie & Yu, 2008). However, the inner working logic, the effect and importance of the features for the model, and also contribution and relation of features are not discussed and examined in any of them. That will help to understand the models and use them in action. To that end, model interpretability and extraction techniques have been developed.

Sommer et al. (Sommer & Paxson, 2010) consider interpretability as one of the reasons that ML techniques are not widely used in practice in networking domain. Given the needs on that end, in addition to the above three sections, we also review the techniques for ML model explainability by leveraging model interpretability and extraction techniques. We review what are these techniques and how they can be used to improve the quality and practicality of models that are used in the networked systems.

2.4.1 Model Interpretation. While there is no widely accepted definition for model interpretability, many definitions have a common concept of understanding and predictability of the outcome (B. Kim, Khanna, & Koyejo, 2016; Miller, 2019). Miller (Miller, 2019) defines interpretability as the degree to which a human can understand the cause of a decision, similar to Kim et al. (B. Kim et al., 2016) that define it as the level of predicting a model's result consistently. The model interpretation has been the focus of different studies since quite a while ago. Jacobsson (Jacobsson, 2005) had the general model acquisition of a domain (understanding the patterns in a specific domain) and improving the model performance in mind as goals of applying model interpretation. However, the goals have not converged over time. Lipton et al. (Lipton, 2018) define the most important reasons models have to be interpretable as follows: 1) *Trust*; it is easier to trust a system that explains its decisions, 2) *Causality*; to ensure causal relationships are picked up by the model and it makes it easy for a human to relate to explanations, 3) *Transferability*; considering the effect of the model on the environment, and resiliency to adversarial manipulation, 4)*Informativeness*; the explanation for decisions through shedding light on model's inner workings, and 5)*Fair and Ethical Decision-Making*; to avoid discrimination, based on race or gender, using the model's decisions. Some of these angles were also used by Carvalho et al. (Carvalho, Pereira, & Cardoso, 2019) while they added *Privacy* to ensure that sensitive information in the data is protected.

There are two main groups of interpreters (X. Zhang, Wang, Ji, Shen, & Wang, 2018): back-propagation and perturbation based interpreters. The back-propagation guided interpretation techniques, compute the gradient of the model prediction with respect to a given input. These techniques use the magnitude of the gradient as a proxy for the relevance of the feature to the prediction (feature importance). Gradient saliency (Simonyan, Vedaldi, & Zisserman, 2013), DeepLIFT (Shrikumar, Greenside, & Kundaje, 2017), SmoothGrad (Smilkov, Thorat, Kim, Viégas, & Wattenberg, 2017) LRP (Bach et al., 2015) are considered as representative examples of this group. The perturbation-based interpretation methods rely on perturbing the input with minimum noise and observe the variations in the model output (prediction) by measuring the increase in the prediction error after permuting the feature's values. A feature is considered to be "important" if shuffling its values increases the model error because in this case, the model relied on the feature for the prediction. First proposed by (Breiman, 2001), and then other techniques such as Shapley Value sampling (Castro, Gómez, & Tejada, 2009), MASK (Fong & Vedaldi, 2017), Occlusion (Zeiler & Fergus, 2014), and more recently a model-agnostic version proposed by (Fisher, Rudin, & Dominici, 2018) in this group.

2.4.2 Model Extraction. While model interpretation methods can capture the importance of different features and facilitate visualizing the outcome, in such efforts the interaction between different features and their relative contribution towards the class labels is not explored. Many efforts have been done to overcome this limitation through model extraction.

Similar to model interpretation, model extraction is not a recent topic. Craven (Craven, 1996) in 1996, Krishnan et al. (Krishnan, Sivakumar, & Bhattacharya, 1999a, 1999b) in 1999 and Boz (Boz, 2002) in 2002 discussed different aspects of extracting a model's rules into a more understandable structure such as a decision tree. Jacobsson (Jacobsson, 2005) surveys a group of studies and explains how rule extraction (RE) can convert neural network models and specifically RNN models into finite state machines, therefore, one can mimic the model while having the advantage of being more transparent. The goals for model/Rule extraction can be considered as follows: 1)to acquire a generic model of the domain, as a tool in the acquisition process (data mining), 2)to provide an explanation of the model, 3)to allow verification or validation of the model with respect to some requirements (e.g. software testing) and thus make new, potentially safety-critical domains possible for the model, 4)to improve on current model architectures by pinpointing errors. In order to assess the quality of the ruleset, different aspects are considered. The *fidelity* which is the accuracy in approximation the original model as well as *rule accuracy* which shows the generalization ability of the ruleset, *rule consistency* as a result of extracting similar rules from models on the same task, and the *Rule comprehensibility* which is the readability of rules or the size of the ruleset, are the main quality measures of model extraction techniques.

There is a range of techniques proposed for model extraction. Zarate et al. (Zarate, Vimieiro, & Vieira, 2006) proposed a method to extract if-then type rules in order to further validate the RNN models by discussing the extracted rule by a domain expert. They test their method in the solar energy domain. Lakkaraju et al. (Lakkaraju, Bach, & Leskovec, 2016) propose a framework for extracting interpretable decision sets (sets of independent if-then rules) from models focusing on accuracy and conciseness. They compare their method with Bayesian Decision Lists (BDL) (Letham, Rudin, McCormick, & Madigan, 2015), CN2 (Clark & Niblett, 1989), and Classification Based on Associations (CBA) (B. Liu, Hsu, & Ma, 1998) methods. Sushil et al. (Sushil, Šuster, & Daelemans, 2018) propose a technique to induce sets of if-then-else rules that capture the relations among features and the labels. They achieve the goal by calculating the importance of the features in the trained network. Then weigh the original inputs with these feature importance scores, simplify the transformed input space, and finally fit a rule induction model (RIPPER-k (Cohen, 1995)) to explain the model predictions. They apply their method on a multi-label classifier. DTextract by Bastani et al. (Bastani, Kim, & Bastani, 2017b) proposed an algorithm to provide global explanations of complex, black-box models in the form of a decision tree (rule set) approximating the original model. They have applied the method on classification, regression, and reinforcement tasks (in health and student grade domains) and shown causal explanations that can be provided for the model's decision based on the calculated decision trees.

The inner working logic, the effect and importance of the features for the model, and also contribution and relation of features can be used in the network domain to illustrate the logic of trained models and gain trust for the system. That will facilitate the practical usage of such ML-based systems. Such analyses are not discussed and examined widely in the networking domain. Having such insights, will help to understand the models and use them in action.

CHAPTER III

DETECTION OF INCENTIVIZED ONLINE REVIEWS

Online retail stores are considered as a large scale network of sellers and shoppers. Shoppers, or in some systems both sides, can provide feedback about each other in a transaction in form of a numerical rating or textual review that forms the most important aspect of the rating system. The rating system can collaboratively guide the involved parties to make better decisions about who to initiate a transaction based on the quality of their prior experiences. However, not all opinions are provided with expected intentions. Given the large scale of such reviews and complexity of extracting intention, in this chapter, we focus on the application of machine learning models to detect and characterize reviews that are incentivized by the sellers‘ promotions in two primary categories of Amazon.com products. This work describes a new method to leverage the difference between reviews that are explicitly incentivized and normal reviews to detect incentivized reviews without any sign in the system. Furthermore, this chapter sheds light on how the prevalence of EIRs has evolved and been affected by Amazon's reaction to this phenomenon.

The content in this chapter is derived entirely from (Jamshidi, Rejaie, & Li, 2018, 2019) as a result of collaboration with co-authors listed in the manuscript. Soheil Jamshidi is the primary author of this work and responsible for conducting all the presented analyses.

3.1 Introduction

As the popularity of online shopping has rapidly grown during the past decade, the shoppers have increasingly relied on online reviews and rating provided by other users to make more informed purchases. In response to shoppers' behavior, product sellers have deployed various strategies to attract more positive reviews for their products as this could directly affect the popularity of these products among users and thus their ability to sell more products online. Several prior studies have examined different aspects of online reviews including fake or spam (Akoglu et al., 2013; Jindal & Liu, 2008; Jindal et al., 2010; F. Li et al., 2011; Lim et al., 2010; Ott et al., 2011) and also biased and paid reviews (Burtch et al., 2017; Petrescu et al., 2017; Shyong et al., 2006; J. Wang et al., 2012; Z. Xie & Zhu, 2015) in different online shopping platforms.

The importance of online reviews has also prompted major e-commerce sites (e.g. Amazon) to implement certain policies to ensure that the provided user reviews and ratings are legitimate and unbiased to maintain the trust of online shoppers. In response to these policies, seller's strategies for boosting their product rating have further evolved. In particular, in the past few years, some sellers have increasingly offered discounted or free products to selected online shoppers in exchange for their (presumably positive) reviews. We refer to these reviews as *incentivized reviews*. Major e-commerce sites such as Amazon require reviewers to disclose any financial or close personal connection to the brand or the seller of the reviewed products (Amazon's Community Guidelines, 2018). However, it is unlikely that average shoppers who solely rely on product ratings notice the biased nature of such reviews. Intuitively, the reviewers who provide incentivized reviews may behave differently than other reviewers for the following reasons: *(i)* they might feel obligated to post positive reviews as the products are provided for free or with a considerable discount, *(ii)* their expectations might be lower than other users as they do not pay the full price, and *(iii)* they do not often consider the long-term usage of the product (e.g. product return or customer service) in their reviews. The presence of such incentivized reviews in Amazon has been reported on (ReviewMeta.com, 2016), however, to our knowledge, the prevalence of incentivized reviews, their characteristics, and their impact on the ecosystem of a major e-commerce site have not been systematically and quantitatively studied. Although Amazon has officially banned submission of incentivized reviews on October of 2016 (aboutAmazon.com, 2016), it is important to study such reviews to be able to determine whether Amazon's new policy solved the issue or just forced reviewers to go undercover.

To tackle this important problem, this chapter focuses on capturing and characterizing several aspects of incentivized reviews in the Amazon.com environment. We leverage the hierarchical organization of Amazon products into categories and subcategories and collect all the information for top-20 best-seller products in all subcategories of two major categories. We then present a method to identify explicitly incentivized reviews (EIRs) on Amazon by identifying a number of textual patterns that indicate explicitly incentivized reviews. We carefully capture and fine-tune these textual patterns using a regular expression. We then use these patterns to identify a large number of EIRs along with their associated products and reviewers. We characterize the key features of EIRs and associated reviewers and products.

Our analysis demonstrates the effect of Amazon ban on the prevalence of EIRs as well as the difference between the features of EIRs and normal reviews. We also examine the temporal pattern of EIR, and non-EIR reviews that a product receives and a reviewer produces to address two questions: *(i)* how the arrival pattern of EIRs for a specific product affects the level of interest (i.e. rate of non-EIRs and their assigned rating) among other users, and *(ii)* how individual reviewers over time become engaged in providing EIRs. Given the apparent gap between the features of normal reviews and EIRs, we examine whether machine learning techniques can detect these differences to identify both explicitly or implicitly incentivized reviews. We show that such a technique can indeed detect other explicitly and implicitly incentivized reviews. Additionally, we investigate the current status of identified EIRs and the ability of sellers to solicit incentivized reviews in today's Amazon platform.

In the rest of this chapter, we describe our data collection technique and our datasets in §3.2. Next, §3.3 presents our method for detecting EIRs. We characterize several aspects of EIRs and their associated products and reviews in §3.4. In §3.5 we discuss the temporal patterns of EIRs and non-EIRs that are submitted for individual products or produced by individual reviewers. §3.6 presents our effort for automated detection of other explicitly or implicitly incentivized reviews using machine learning techniques. The current state of EIR reviews is explored in §3.7. Finally, §3.8 summarizes this chapter.

3.2 Data Collection and Datasets

This section summarizes some of the key challenges with data collection and then describes our methodology for collecting representative datasets that we capture and use for our analysis. Amazon web site organizes different products into categories that are further divided into smaller subcategories. Each product is associated with a specific seller. A user who writes one (or multiple) review(s) for any product is considered a reviewer of that product. For each entity (i.e. user, review or product), we crawled all the available attributes on Amazon as follows:

- Reviews' attributes: review id, reviewer id, product id, Amazon Verified Purchase (AVP) tag, date, rating, helpful votes, title, text, and link to images.
- Products attributes: product id, seller id, price, category, rating, and title.
- Reviewers' attributes: reviewer id, rank, total helpful votes, and publicly available profile information.

In particular, *AVP tag* of a review indicates whether the corresponding reviewer has purchased this product through Amazon and without deep discount or not, as is defined on (Amazon.com, 2018).

There are a few challenges for proper collection and parsing of this information from major online websites (Rejaie, Torkjazi, Valafar, & Willinger, 2010). First, there is a very large number of product categories where the format, available fields for products, and tendency of users to offer reviews widely vary across different categories. Furthermore, we need to comply with the ethical guidelines as well as the enforced rate limits by Amazon servers for crawlers which makes it impossible to collect the reviews for all products within a reasonable window of time. To cope with these challenges, collecting a representative sample can be used (Rasti et al., 2008). We collect three datasets where each one provides representative samples of products, reviews and reviewers. We developed Python pipelines using libraries such as Selenium. It took us more than 6 months to crawl the datasets given the fact that we limited our crawling rate.

Sample Products (DS1): We focus on two popular categories of products, namely *Electronics* and *Health & Personal Care* since they have a large number of sub-categories and products that receive many reviews. To make the data collection

Table 1. Basic Features of Our Datasets

	Products (DS1)	**EIRs (DS2)**	**Normal Reviews**	**Reviewers (DS3)**
Reviews	3,797,575	100,086	100,086	217,000
Reviewers	2,654,048	39,886	98,809	2,627
Products	8,383	1,850	1,641	184,124

manageable and given the skewed distribution of reviews across products, we only capture all the information for the top-20 [1]best seller products in each sub-category in the above two categories from *Amazon.com*. While these products represent a small fraction of all products in these two categories, the top-20 products receive most of the attention (#reviews) from users and enable us to study incentivized reviews. We refer to this product-centric dataset as *DS1*.

Sample EIRs (DS2): Using our technique for detecting Explicitly Incentivized Reviews (EIR) that is described in §3.3, we examine all the reviews associated with products in DS1 and identify any EIRs among them. We refer to this set of EIRs as DS2 dataset.

Normal Reviews: After excluding EIRs, we examine the remaining reviews for products in DS1 and consider each review as normal if it is not among EIRs and *(i)* associated with an Amazon Verified Purchase, *(ii)* submitted on the same set of products that received EIRs, and *(iii)* submitted by users who have not submitted any EIRs. We rely on this rather conservative definition of normal reviews to ensure that they are clearly not incentivized. We identified 1,214,893 normal reviews and then selected a random subset of them (the same number as

[1] https://www.amazon.com/gp/bestsellers/

EIRs). We refer to these selected reviews as our normal review dataset that serves as the baseline for comparison with EIRs in some of our analysis.

Incentivized Reviewers (DS3): To get a complete view of sample incentivized reviewers, we randomly select 10% of reviewers associated with the reviews in *DS2* dataset. For each selected reviewer with a public profile, we collect their profile information and all of their available reviews. Overall, we collect this information for 2,627 reviewers and only consider their reviews for our analysis.

The DS1, DS2, and Normal reviews datasets were collected in December 2016, and the Reviewers dataset (DS3) was collected in January 2018.

3.3 Detecting Explicit Incentivized Reviews

Automated identification (or labeling) of incentivized reviews requires a reliable indicator in such reviews. To this end, we first focus on reviews in which the reviewer *explicitly* indicates his/her intention for writing the review in exchange for a free or discounted product. Such an indication must be provided in the reviews since Amazon requires that reviewers disclose any incentive they might have received from the sellers as noted on (Amazon's Community Guidelines, 2018). Furthermore, these reviewers also include such incentives in their reviews to attract more sellers to offer them similar incentives in exchange for their reviews to promote their products. Our manual inspection of a large number of reviews revealed that many reviewers indeed explicitly state their incentive for writing their reviews. These reviews contain some variants of the following statements: *"I received this product at a discount in exchange for my honest/unbiased review/feedback."* To capture all variants of such statements, we select any review that matches the following regular expression in a single sentence of the review:

$$\begin{array}{|l|}\hline '(sent|receive|provide)[\hat{}\backslash.!?] \qquad * \\ (discount|free|in-trade|in-exchange)[\hat{}\backslash.!?]* \\ (unbiased|honest)[\hat{}\backslash.!?]* \\ (review|opinion|feedback|experience)' \\ \hline \end{array}$$

Among all the 3.79M reviews in the DS1 dataset, 100,086 reviews submitted by 39,886 users on 1,850 products match some variants of the above regular expression in one sentence. We consider these 100,086 reviews as EIRs and group them in our DS2 dataset.

We also considered a more relaxed setting where reviews could have the above regular expression across multiple sentences. This strategy tags 325,043 reviews from 210,198 users on 7,059 products as EIR. However, our careful inspection of many of the newly-identified EIRs by this more flexible strategy revealed that some of them are non-incentivized reviews that happen to match the regular expression. To avoid any such false-positives in our EIRs, we adopt a conservative strategy and only consider a review as EIR if the desired pattern is detected within a single sentence.

EIR-Aware Reviews: Our extensive manual inspection of the identified EIRs also revealed that in a tiny fraction (only 30 reviews) the reviewer simply refers to other EIRs to complain about them, indicate his/her awareness and inform other users of such incentivized reviews. However, these reviews are not incentivized themselves. To exclude these reviews, we manually checked random samples of reviews and found that these EIR-aware reviews contain one of the following terms (*who received/with the line “i received/which say they received/their so-called “honest"*).

We then exclude any identified EIR that matched these aware patterns. After extensive manual work in this step, we found only 30 aware reviews by 26 reviewers on 29 products that are excluded from DS2. Interestingly, all these aware reviews were collectively marked as helpful by 194 other users, indicating that many other reviewers felt the same way about the incentivized reviews. This illustrates how the presence of incentivized reviews could impact the trust of customers in the authenticity of Amazon reviews.

3.4 Basic Characterizations of Amazon Reviews

In this section, we examine a few basic characterizations of EIRs and their associated products and reviewers in order to shed some light on how these elements interact in Amazon.com.

3.4.1 Product Characteristics. One question is *what fraction of reviews for individual products are EIRs?* We use all products in dataset (*DS2*) to examine several characteristics of products that receive at least one EIR.

Fig. 2 and Fig. 3 present the summary distribution of the fraction of product reviews that are EIRs for different groups of products based on the total number of reviews in each category. The red lines (and red dots) show the median (mean) value for each box plot. The green diamonds on these figures show the fraction of all products (per category) that are in each group using the second Y-axis. These figures show that for products in Health and Personal Care category, typically 10-20% of reviews are EIR regardless of the total number of reviews for a product. However, for products in the Electronics category, the fraction of EIRs is generally smaller and rapidly drops as the number of product reviews increases. This suggests that the prevalence of EIRs could vary across different categories of Amazon products.

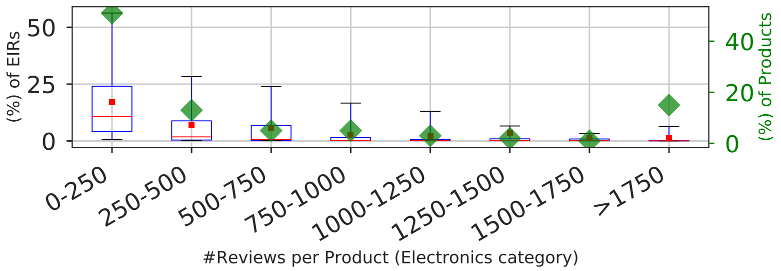


Figure 2. Distribution of Fraction of EIRs per Product in *Electronics* Category

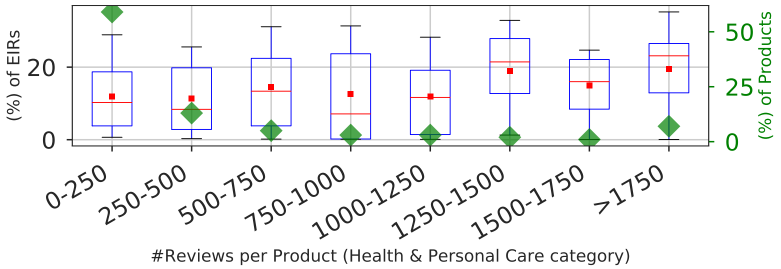


Figure 3. Distribution of Fraction of EIRs per Product in *Health* Category

Another important question is *how the total number of EIR reviews and associated products have changed over time?* Fig. 4 depicts the temporal evolution of the number of observed EIRs per day (with a red dot) as well as the cumulative number of unique products (with the dotted line using the right Y-axis) that received EIRs over time using our DS3 dataset. This figure reveals that while EIRs were present in Amazon at a very low daily rate since 2012, the number of EIRs and associated products have dramatically increased between the middle of 2015 and the middle of 2016. We can clearly observe that Amazon's policy for banning EIRs that was announced in October 2016 (aboutAmazon.com, 2016) have been very effective in rapidly reducing the daily rate of EIRs and the number of affected products within a couple of months. We note that the effect of this new policy on the implicitly incentivized reviews is unknown.

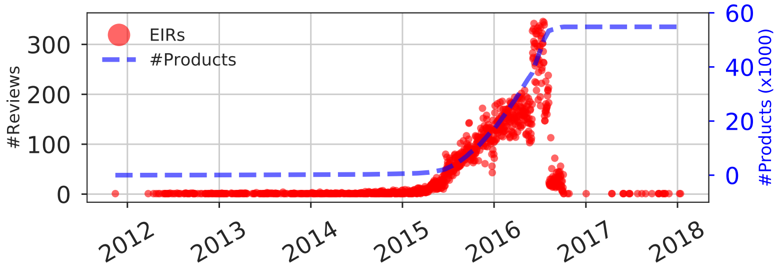


Figure 4. Evolution of the Daily Number of EIRs and the Total Related Products

Another issue is the price range for products that possibly motivate the reviewers to provide EIRs. We observe that 80% (95%) of these products cost less than \$25 (\$50). In essence, there is typically no significant financial gain in providing a small number of EIRs.

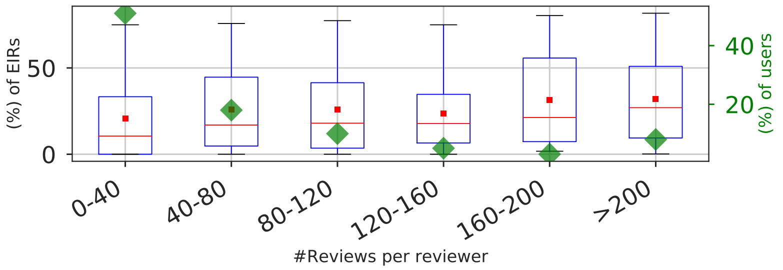


Figure 5. Distribution of the Fraction of Provided EIRs per Reviewer

3.4.2 Reviewer Characteristics. We now turn our attention to reviewers that provided at least one EIR (i.e. reviewers in DS3) to characterize several aspects of these reviewers. We first explore the question of *what fraction of reviews provided by individual reviewers are EIRs?* This illustrates to what extent a reviewer is engaged in writing EIRs.

Fig. 5 presents the summary distribution of the fraction of all reviews of individual users that are EIRs across different groups of users based on their total number of reviews. This figure also presents the number of reviewers in each group (green diamonds) using the second Y-axis. This result illustrates that the fraction of EIRs for most reviewers varies between 30-40% of all their reviews. Interestingly, as the reviewers become more active, EIRs make up a more significant fraction of their reviews. To get a better sense of the type (i.e. demography) of users who are likely to provide EIRs, we examined their public profile description and identified the following most common keywords (and their frequencies): "*love*" (1.0) , "*products*" (0.41), "*new*" (0.40), "*Review*" (0.39), "*home*" (0.38), and "*mom*" (0.34). Our manual inspection of these profiles confirms that around 18% of these reviewers are *moms staying at home that love to review new Amazon products.*

3.4.3 Review Characteristics.

We take a closer look at various features of EIRs in comparison with normal reviews as a reference group.

Helpfulness: An essential aspect of reviews is how helpful they are to other users. Amazon reports the total number of *helpful votes* (up-votes) per review. A slightly larger fraction of normal reviews (12.68%) receive up-votes compare to the EIRs (10.87%). Fig. 6 shows the Complementary Cumulative Distribution Function (CCDF) of the number of up-votes for EIRs and normal reviews. This figure reveals EIRs and normal reviews exhibit the same degree of helpfulness, but the extreme cases for normal reviews are much more helpful.

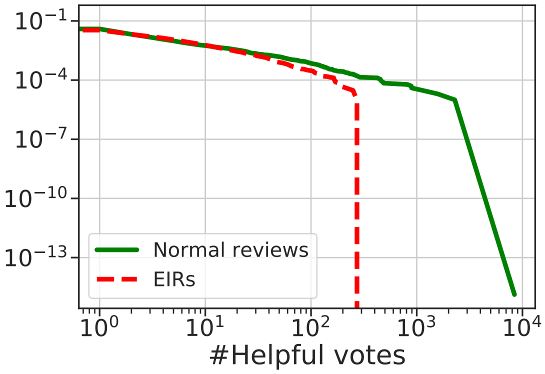


Figure 6. CCDF of Helpfulness

Review Content: We start by comparing several features of EIR content with normal reviews. First, we observe that 13% of EIRs attach at least one image to their reviews while this ratio is ten times smaller (1.3%) for normal reviews. We perform sentiment analysis on both content and title of reviews using *textblob* library. The sentiment is measured by a value within the range of [-1, 1] where 1 indicates positive, 0 neutral, and -1 a negative sentiment. Fig. 7 presents the distribution of sentiment for the content of EIRs and normal reviews. We observe that 9.5% (9,498) of normal reviews have negative sentiment, 9.1% are neutral (i.e. their sentiment measure is zero) and the rest are positive reviews that are spread across the whole range with some concentration around 0.5, 0.8, and 1. In contrast, the sentiment of nearly all EIRs are positive, but more than 80% of them are between 0 to 0.5. In essence, the sentiment of normal reviews is widespread across the entire range while sentiments for EIRs are mostly positive but more measured. Similarly, less than half of the normal reviews and three-quarter of EIRs have titles with positive sentiments.

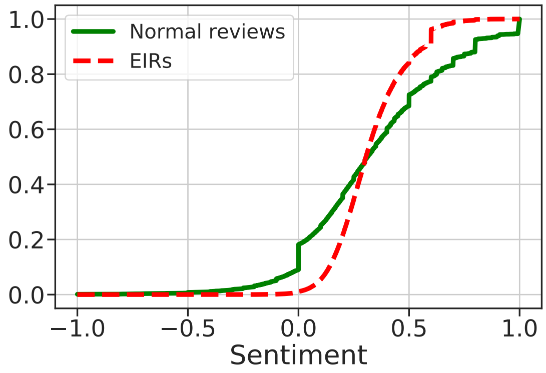


Figure 7. CDF of Review Sentiment

Using *TextBlob* library, we also analyzed the *Subjectivity* of reviews, which marks the presence of opinions and evaluations rather than using objective words to provide factual information. Fig. 8 depicts the CDF of the subjectivity across EIRs and normal review datasets. This figure reveals that the subjectivity for 83% of EIRs are between 0.4 and 0.8 while the subjectivity of normal reviews is widely spread across the whole range for normal reviews.

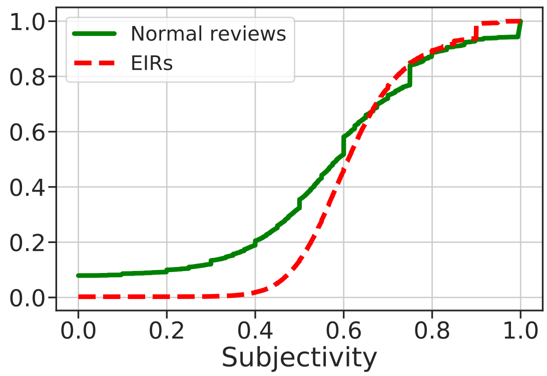


Figure 8. CDF of Review Subjectivity

We use the *Gunning Fog index* (Gunning, 1952) to measure the readability test for English writing in each group of reviews. This index estimates the number of years of formal education a person needs to understand the text on the first reading. For example, a Fog index of 12 requires the reading level of a U.S. high school senior. Fig. 9 shows the CDF of the Fog index across EIRs and normal reviews. This result illustrates that the readability of EIRs requires at least 4 years of education and is 1.5 years higher than normal reviews on average (7.5 vs. 6 years of education). Also, the index exhibits much smaller variations across EIRs. In short, the writing of EIRs is more elaborate.

Self-Similarity of User Reviews: Similarity of the content across submitted reviews by individual users reveals whether a reviewer merely repeat the same set of sentences across different reviews (and thus provides a generic review) or not. To this end, we assess the level of similarity in the text of all pairs of written reviews

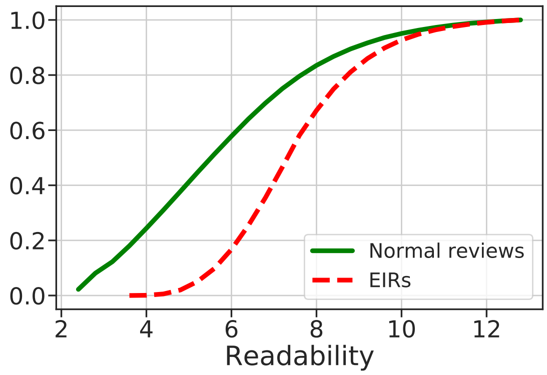


Figure 9. CDF of Review Readability

by a normal reviewer and all pairs of written EIRs by EIR. We use the *Jaccard index* on the uni-grams of reviews as a measure of similarity between the content of a pair of reviews. We consider all reviewers with at least 5 EIRs (1,004 users) from DS2 for this analysis and the same number of randomly selected normal reviewers with the same distribution of reviews as a reference.

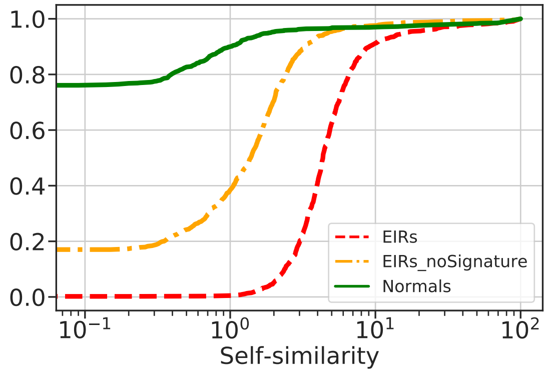


Figure 10. Pair-wise similarity of the content of submitted reviews by normal and EIR reviewers (with and without the text of disclaimer for their incentive) based on Jaccard index

Fig. 10 depicts the CDF of pairwise similarities between normal reviews and EIRs across normal and EIR reviewers using the log-scale for the x-axis. This figure demonstrates a measurably higher level of self-similarity between EIRs compare to normal reviews. In particular, 75% (97%) of pairs of normal reviews by individual normal reviewers exhibit zero (<5%) similarity. However, 90% (and 39%) of pairs of reviews submitted by EIR reviewers exhibit more than 1% (5%) similarity in their content. Interestingly, we identified 10 EIR and 14 normal reviewers who have submitted roughly between 6 to 10 identical reviews on different products. The presence of the regular expression in EIRs that explicitly indicates the reviewer's incentive could increase the level of similarity between reviews of an EIR reviewer. To ensure that our similarity measure is not significantly affected by the explicitly stated incentive in EIRs, the orange line (labeled EIRs-noSignature) in Fig. 10 also presents the level of similarity between pairs of EIRs per reviewer after removing the identified regular expression from all EIRs. Fig. 10 shows that the level of similarity between reviews of individuals with EIRs is still much larger than reviews of normal reviewers.

Review Submission Timeline: Another interesting question is whether the submission timeline for EIRs vs normal reviews for individual products is different? In particular, over which part of a product lifetime EIRs and normal reviews are submitted. In the absence of any explicit signal, we use the time between the first and last review of a product as an approximation of its lifetime [2] and assign a normalized submission time that we call recency of review for all reviews of products in DS1. For example, recency of 100% indicates that a review was submitted very recently whereas 0% implies a review that is submitted right after a product becomes available. Fig. 11 and 12 show the summary distribution of

[2] Amazon provides the date when a product becomes available for some categories of product. However, we frequently observe cases where a product has multiple versions in the same product page that have become available at different times but share the same pool of reviews. We use the time between the first and last reviews across all versions of a product to deal with this ambiguity in relating specific review to a particular version of a product.

recency of all reviews (left Y-axis) for different groups of products based on their age in terms of the number of months on the X-axis. Both figures show the fraction of products in each group (right Y-axis). The bottom part of Fig. 11 also presents the prevalence of EIRs in Amazon by showing the number of submitted EIRs in the same window of time. Fig. 12 clearly illustrates that the submission timeline of normal reviews is generally balanced across the life of corresponding products regardless of their age. However, the submission timeline for EIRs exhibits broadly two different patterns based on product age. For products that have become available during the most recent 18-month window when EIRs were prevalent in Amazon, a visibly larger fraction of EIRs was submitted during the first half of product lifetime. However, for older products, the EIRs were submitted during the more recent window since EIRs were not common on Amazon during the first half these products lifetime. In summary, the submission of EIRs in the early part of recent products' lifetime indicates sellers' effort to attract EIRs as they list a new product on Amazon whereas the late submission for older products is simply due to the relatively recent availability of EIRs over products' lifetime.

Length of Reviews: The overall length of a review and its title could be viewed as measures of its level of details. We observe that the typical (i.e. median) length of an EIR (599 characters) is more than three times longer than a normal review (179 characters). Interestingly, the longest normal review (14.8K character) is much longer than the longest EIR (11K character). We observe a similar pattern for the length of reviews based on word count. Furthermore, the title for EIRs are typically 6.6 words long which is two words longer than the title of normal reviews.

Star Rating: A critical aspect of a review is the star rating (in the range of 1 to 5 stars) that it assigns to a product. We observe that the assigned rating by EIRs

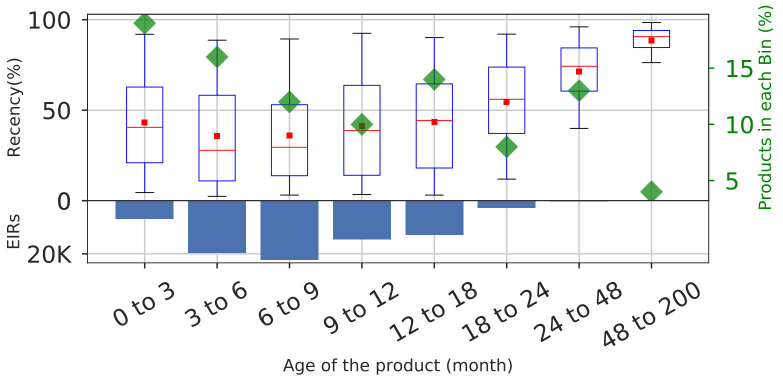


Figure 11. The submission timeline of EIRs across lifetime of products with different ages

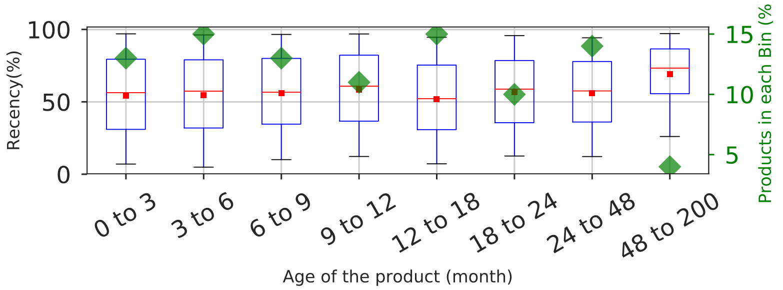


Figure 12. The submission timeline of normal reviews across lifetime of products with different ages

is frequently more positive than normal reviews. More specifically, 95% (75%) of EIRs associated the rating of at least 3 (5) stars while this number drops to 1 (4) for normal reviews.

Statistical Significance: We have shown that several features of EIRs - namely star rating, helpfulness, text and title length, readability, and sentiment - exhibit different distributions compare to normal reviews. This raises the question that whether the reported difference in these distributions are statistically significant. We perform statistical test to answer this question. We observe that none of these features follow a normal distribution as they did not pass the normal test (d'Agostino, 1971). Therefore, we rely on *Kruskal-Wallis* test (Kruskal & Wallis,

1952) to tackle this question. Kruskal-Wallis tests the null hypothesis that the population median of all the groups are equal. If we observe a large p-value (e.g. more than 0.01), then we cannot reject the null hypothesis. The observed that $p < 0.0001$ for the distributions of most of these characteristics and p=0.006 for review helpfulness. These results suggest hat the difference between the distribution of these features for EIRs and normal reviews are indeed statistically significant.

Reviewer-Review Mapping Per Product: A majority of reviewers (99.8%) in our EIR dataset (DS2) have written only one EIR for each product. We only found 73 users who have written multiple EIRs for at least one product. These reviews add up to the total of 151 EIRs for 32 unique products. None of the users in our user-centric dataset (DS1) writes multiple EIRs for a single product. Given the one-to-one relationship between the absolute majority of reviewer-review pairs per product, for the rest of our analysis, we assume each reviewer has only a single review per product and vice versa.

Association of Reviewers with Sellers: So far we have primarily focused on the relationship between reviewers and products through reviews. In practice, individual sellers often offer multiple products on Amazon. This raises two questions regarding the associations of EIR reviewers and sellers that we explore here: The first question is *whether an EIR reviewer is typically approached by single or multiple sellers to review their products?* Fig. 13 presents the CDF of the number of unique sellers that each EIR reviewer (in DS3) has submitted at least one EIR for their products. We observe that 75% (95%) of reviewers only submit EIR for at most 1 (3) products of each seller.

The second question is *how many products of a seller a reviewer submit an EIR for?* We examine the distribution of the number of products of a seller

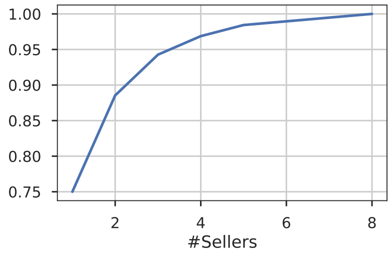


Figure 13. CDF of the number of sellers that each user submitted EIR for

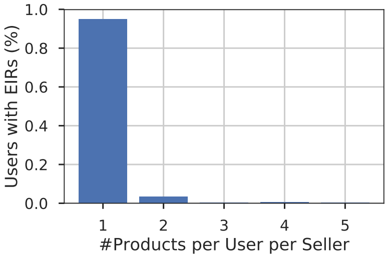


Figure 14. The distribution of number of products of a seller that each reviewer submits EIR for

that each reviewer in DS3 submits an EIR for. In Fig.14 we observe that the 94.5% (99%) of reviewer-seller relationships are through a single (two) reviewed product(s). *In summary, these results show that EIRs reviewers usually submit an EIR for a single product of one or two sellers.*

Crowdsource Agents: After closer examination of EIRs, we identified hundreds of reviews in DS1 where reviewers explicitly mentioned that they received the products from a specific agency (e.g. *BuzzAgent*, *Influenster*, and *AMZ Review Trader*) in exchange to share their reviews. Our investigations revealed that these websites are associated with crowdsourcing agents that promote a seller's products on different social media platforms. Once a user registers on these websites, she receives certain free products in exchange for her reviews on different social media platforms (e.g. Amazon, Twitter, YouTube). In essence, these websites manage some of the promotional campaigns for selected sellers' products (presumably) for a fee.

Using the name of 12 identified crowdsourcing agencies, we detected 2,124 incentivized reviews from 1,991 reviewers on 237 products among all reviews in DS1. 89.9% of these reviews assigned a strongly positive rating (4 and 5 stars) and 71% of them have a strong positive sentiment. Note that only a small fraction (116 out of 2,124) of these reviews were EIRs. This analysis demonstrates that there are other indirect ways for sellers to offer incentives to users for submitting positive reviews on Amazon and other more popular social media platforms.

3.5 Temporal Analysis of Amazon Reviews

All of our previous analysis have focused on the overall characteristics of reviews, reviewers, and products over their entire lifetime. Intuitively, product sellers offer various incentives to attract reviewers and obtain incentivized reviews for their specific product. Obtaining these incentivized reviews over time increases the available information and improves the overall image (e.g. rating) of the product. This, in turn, expands the level of interest among (ordinary) users who may consider to buy the product and provide their own review. Examining the temporal pattern of submitted reviews (by various reviewers) for a product or submitted reviews by a reviewer (for any product) sheds an insightful light in various dynamics among seller products, reviews, and reviewers.

In this section, we tackle two important issues: First, we inspect the *"review profile of sample products"* to study how the temporal pattern of obtained EIRs for a product affect the level of interest among other users. Second, we examine the *"review profile of sample reviewers"* to explore how reviewers get engaged in producing EIRs. To tackle these questions we have inspected temporal patterns for many products and reviewers, and only present a few sample cases to illustrate our key findings better.

In this analysis, we primarily focus on the number of EIRs, non-EIRs (i.e. reviews that are not tagged as EIR by our method) associated with a product (or a reviewer) per day and their (cumulative) average rating. [3] across EIRs or non-EIRs that a product receives or a reviewer assigns.

3.5.1 Product Reviews. We consider four different products to examine the temporal correlations between the daily number of EIRs and the level of interest among other users, namely the number of non-EIRs and their ratings, for each product.

Note that a product seller can (loosely) control the arrival rate of EIRs by offering incentives (or promotions) with a particular deadline to a specific set of reviewers. We refer to such an event as a *promotional campaign.* The goal of our analysis is to investigate whether and to what extent such a campaign affects the number of non-EIRs and their rating for individual products. Note that a product seller can (loosely) control the arrival rate of EIRs by offering incentives (or promotions) with a certain deadline to a specific set of reviewers. We refer to such an event as a *promotional campaign.* By specifying a deadline for the incentive or promotion, the seller can also force interested users to write their reviews within a specific window of time. We simply assume that any measurable, sudden increase in the number of daily EIRs for a product is triggered, by a promotional campaign that is initiated by its seller. The goal of our analysis is to investigate whether and to what extent such a campaign affects the number of non-EIRs and their rating for individual products. Each plot in Fig. 15 presents the daily number of EIRs (with a red X), the daily number of non-EIRs (with a green diamond), the cumulative average rating for all non-EIR (with dotted green) and EIR (with dotted red lines)

[3] Amazon appears to rely on some weighted averaging method (Bishop, 2015) to calculate the overall rating of a product based on factors such as the recency of a review, its helpfulness and whether it is associated with a verified purchase. Since the details of Amazon's rating method is unknown, we simply rely on a linear moving average of all ratings to determine the overall rating of each product or reviewer over time.

for a single product. Each plot also shows the cumulative rating of all reviews with a solid blue line. Three rating lines on each plot are based on the right Y-axis showing the star rating (1 to 5 scale).

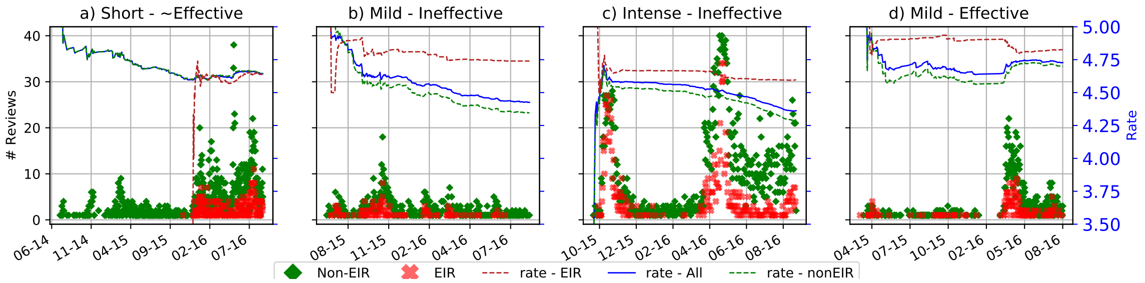


Figure 15. Temporal Patterns of Reviews for Individual Products

Short & Moderately Effective Campaigns: Fig. 15-a shows a product that has been consistently receiving a few daily non-EIR (and not a single EIR) reviews over a roughly two year period. Its average product rating rather consistently drops during 2015. A persistent daily rate of EIR suddenly starts in early 2016 and continues for a few months indicating a likely promotional campaign. The campaign triggers a significant increase in the number of non-EIRs. Interestingly, the average rating of EIR rapidly converges to the average rating of non-EIR (and the overall rating) and not only prevents further dropping but also rather improves the overall rating of this product. This appears to be a short-term (over a few months) and moderately effective promotional campaign by the seller.

Multiple Mild but Ineffective Campaigns : Fig.15 - b presents another product that consistently receives non-EIRs over a one year period. We can also observe ON and OFF periods of EIRs that did not seem to seriously engage other users with this product (i.e. no major increase in the daily rate of non-EIRs). The assigned rating by EIRs is relatively constant, and their gap with the rating of non-

EIRs (and overall rating) rapidly grows. Clearly, these multiple mild campaigns are not effective in raising the ratings of the product.

Multiple Intense but Ineffective Campaigns: Fig. 15- c shows a product that has been consistently receiving both EIR and non-EIRs over a year-long period. However, there are two (and possibly three) distinct windows of time (each one is a few weeks long) with pronounced peaks in the number of daily EIRs which suggests two intense campaigns. Interestingly, the first campaign only generates short-term interest among ordinary users (shown as a short-term increase in the daily number of non-EIRs) while the second campaign triggers more non-EIRs. The average rating of EIR is clearly above non-EIRs. However, the average rating of non-EIRs (and even EIRs) continues to drop over time despite the increased level of attention by other users after the second campaign. Therefore, these multiple intense campaigns were not able to improve the overall rating of this product.

Multiple Mild and Effective Campaigns: Fig. 15-d shows a product with a low and persistent daily EIR and non-EIR over a one-year period. We then observe a couple of months with absolutely no reviews that suggest the unavailability of the product. This is followed by a more active campaign of EIRs over a month that continues at a lower rate. This last campaign seems to significantly increase the level of interest among the regular users as well as their rating for this product. In particular, the average rating by non-EIRs was relatively stable and clearly below the rating by EIRs until the last campaign. Interestingly, the last campaign decreases the overall rating by EIRs while enhances the overall rating by non-EIRs. Therefore, we consider this as an effective campaign.

These examples collectively demonstrate that while a seller could loosely control the duration and intensity of its promotional campaign for a product, its impact on the level of engagement by other users could be affected by many other factors (e.g. quality of reviews, strategies of competitors, and quality of the product) and thus widely vary across different products.

3.5.2 User Reviews.

We now focus on the written EIRs and non-EIRs by individual users over time. Similar to the temporal patterns of product reviews, we show the number of daily EIRs (with a red X), non-EIRs (with a green circle). We also show average assigned rating by EIRs (with red dotted line) and non-EIRs (with green dotted line) of the reviewer over time. The three plots in Fig. 16 present the temporal pattern of all reviews (for any product) and their rating for three different reviewers.

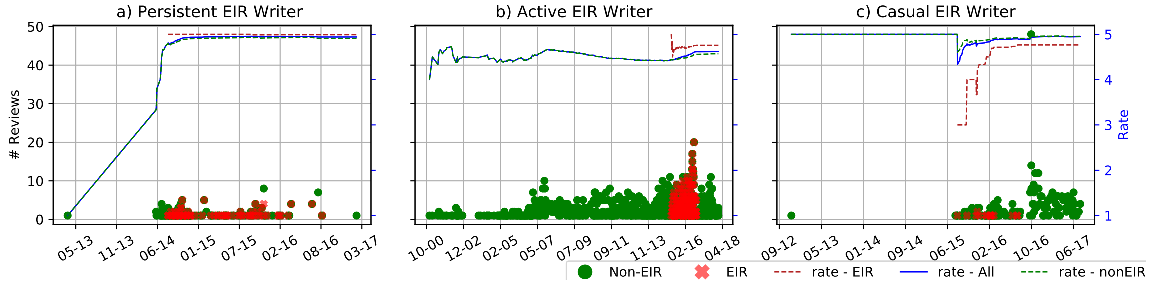


Figure 16. Temporal Patterns of Reviews for Individual Reviewers

Persistent EIR Writer: Fig. 16-a shows a user who provided a single review in 2013 and was inactive for more than a year. Starting in April 2014, she has been submitting a couple of EIRs and/or non-EIRs a day for 20 months, and then her activity significantly dropped. Her average rating for EIRs and non-EIRs are very similar. It appears that this reviewer has become active in Amazon mainly to provide EIRs. But it is rather surprising that she stopped submitting EIR when these types of reviews are very prevalent on Amazon (as we showed in Fig. 4).

Active EIR Writer: Fig. 16-b shows a user who has been actively writing non-EIRs over 16 years since 2001, and her level of activity has gradually increased.

Interestingly, she started posting EIRs from 2015 for two years and then stopped. These two years are perfectly aligned with the period in which EIRs have become rapidly popular in Amazon (as we showed in Fig. 4). Furthermore, the overall assigned rating by this reviewer in non-EIRs was relatively stable over time which was slightly lower than her assigned rating in EIR reviews. This reviewer is a perfect example of a serious Amazon reviewer who takes advantage of offered incentives by sellers for writing EIRs.

Casual EIR Writer: Fig. 16-c shows the temporal pattern of review submission by a user who has been in the system since 2013. However, he became moderately active in the middle of 2015 and provided some EIRs and mostly non-EIRs in the past two years. The number of his EIRs are limited and mostly written over a one year period. It is rather surprising that his rating in EIRs gradually grew over time and was always slightly lower than his ratings for non-EIRs. Far from normal behavior, he has written 49 non-EIRs in one day in 2016 (the green dot above the rating lines). Overall, he appears to be a moderate reviewer who casually writes EIRs.

In summary, our user-level temporal analysis of EIRs and non-EIRs indicates that: *Reviewers exhibit different temporal patterns in producing EIRs. However, users are more active while incentives are offered.*

3.6 Detecting Other Incentivized Reviews

So far in this chapter, we primarily focused on EIRs for our analysis since we can reliably detect and label them as incentivized reviews. However, in practice, there might exist a whole spectrum of explicitly or implicitly incentivized reviews besides EIRs. An intriguing question is *whether all these incentivized reviews (regardless of their implicit and explicit nature) share some common features that* *can be leveraged to detect them in an automated fashion?* To tackle this question, we consider a number of machine learning and neural network classification methods that are trained using a combination of basic and text features of the reviews.

Pre-processing Reviews: We use 100K random EIRs (from the DS3 dataset) and the same number of normal reviews as our labeled data. First, we remove the sentence that indicates the explicit incentive of a reviewer from each EIR before using the EIRs in this analysis so that these sentences do not serve as a prominent explicit feature. Second, we consider the following pre-processing of text of reviews to examine their exclusive or combined effect on the accuracy of various detection methods: *(i)* converting all characters to lower-case, *(ii)* using the stem of each word in the review (e.g. “wait" is the stem for words “waiting", “waits", “waited"). *(iii)* using only alphanumeric characters, and *(iv)* removing all the stop-words using the *NLTK* library in python. (v) converting all the frequent contractions such as *“'ve, 'd, I'm, 'll, n't"* to their formal form.

Classification Methods: We examine a number of classification methods including *Multi-Layer Perceptron (MLP)*, *SVM*, *GaussianProcess*, *DecisionTree*, *RandomForest*, *AdaBoost* Classifiers. Each classifier is trained and tested in three scenarios with a different combination of review features as follows: *(i) Basic Features*: Using nine basic features of reviews, length, sentiment, subjectivity, and readability of review text, star-rating, and helpfulness of reviews, as well as length, sentiment, and subjectivity of title, *(ii) Text Features*: Using extracted text features of the review including the word- or character-based {uni, bi, or Tri}-grams (limited to 1500 text features), *(iii) All Features*: Combination of all basic and text-based features. Individual methods are evaluated in 5 and 10-fold cross-validation as well as 70/30 test and training split manner. We only present the result for the 10-fold cross-validation of the MLP method using pre-processed reviews. The results for all other cases are available in our technical report (Soheil, Reza, & Jun, 2016-18).

We found MLP to be considerably better regarding memory usage, computation time, and accuracy on a 50-50% combination of EIR and normal reviews in the training set. We use 90% of data for training and testing and 10% of data for hyper-parameter tuning using the *grid-search* in SciKitLearn library. The MLP classifier is trained using default parameters, except for *alpha* (the L2 penalty regularization term) and *hidden_ layer_ size* that we set to 0.1 and (50,30,10), respectively. Table 2 presents the accuracy, recall, precision, F1-score, Precision-Recall Area Under Curve (P-R AUC), and the Receiver Operating Characteristic (ROC) AUC for MLP Classifier over all runs. These results indicate that even without the explicit acknowledgment sentence in EIRs, a classifier can accurately detect EIRs from normal reviews using basic or text feature. The accuracy further improves if we combine both sets of features.

Model Evaluation: We further evaluate the machine learning model by exploring the logic behind its decision-making process. This exercise demonstrates whether the model is trustworthy and exposes any potential problems in the model that should be addressed. First, we assess how certain our model is in making decisions. Fig. 17 shows the summary distribution of the prediction probability of all test records per class for the model that uses both basic and n-gram features along with the number of records per class (in blue). This result indicates that our model typically exhibits high confidence (92% and 93%) for predicting EIR

Table 2. The evaluation of MLP classifier in detecting EIRs.

	Accuracy	**Recall**	**Precision**	**F1-score**
Basic	0.85±0.03	0.83±0.01	0.8±0.03	0.83±0.01 (0.84,0.83)
Text	0.9±0.01	0.89±0.01	0.89±0.01	0.89±0.01 (0.89,0.89)
Basic+Text	**0.93**±0.02	**0.92**±0.01	**0.92**±0.03	**0.92**±0.01 (0.92,0.92)
C-Elect.	0.91±0.03	0.91±0.01	0.91±0.03	0.91±0.01 (0.91,0.91)
– on Health	0.8	0.87	0.76	0.81 (0.78,0.81)
C-Health	0.89±0.03	0.86±0.01	0.83±0.04	0.86±0.01 (0.86,0.86)
– on Elect.	0.85	0.89	0.83	0.86 (0.85,0.86)

	Pr-Re AUC	**ROC AUC**
Basic	0.88±0.01	0.83±0.01
Text	0.92±0.0	0.89±0.01
Basic+Text	**0.94**±0.01	**0.92**±0.01
C-Elect.	0.93±0.01	0.91±0.01
– on Health	0.85	0.8
C-Health	0.9±0.01	0.86±0.01
– on Elect.	0.89	0.85

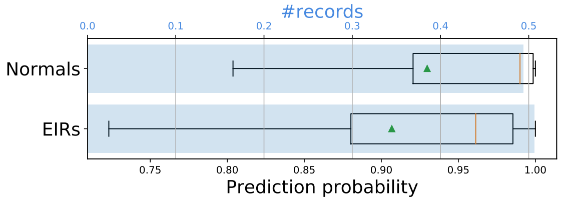


Figure 17. The classification probability for EIRs and normal reviews along with the number of records in each category

and normal reviews. However, its confidence has a rather wider variation for EIR records.

To explain how our model makes the decisions, we incorporate the LIME (Ribeiro, Singh, & Guestrin, 2016) framework to assess the feature importance for each of the labels. Fig. 18 and Fig. 19 depicts the summary distribution of feature importance for the top 10 features across all testing samples on predicting EIR and normal reviews, respectively. Features are sorted based on their prevalence (light blue bars). We observe that 8 out of 10 top features are the N-grams of the review content and the other two features are basic overall characteristics of reviews, namely their length and subjectivity. As expected, our model considers reviews with higher subjectivity values as a candidate for EIR and lower ones as normal, although makes this decision in combination with other features.

Category-specific classification: To asses how generally accurate our model can be and whether we need a category specific model, we examine the ability of a classifier for detecting EIRs in other categories. To this end, we divide EIRs and normal reviews into two groups based on the category of their corresponding product (i.e. Electronics and Health). We train two classifiers, called *C-Health* and *C-Elect.* where each one only uses EIRs and normal reviews (with a combination of basic and text features) associated with products in the corresponding category. Finally, we test each classifier on reviews from the same (the 4th and 6th rows of Table 2) as well as on reviews from the other category (the 5th and 7th rows of Table 2) to assess their accuracy in detecting EIR and normal reviews. The last four rows of Table 2 present the accuracy of MLPC for detecting EIRs within each category and between two categories. These results show that the accuracy of detection for EIRs within each category is around 90% and it remains above 80% for cross-category detection of EIRs. Interestingly, the classifier that is trained with Health reviews exhibits a higher accuracy in detecting Electronics reviews.

Next, we investigate the ability of our trained classifier using the basic and text-based features in detecting other incentivized reviews, namely implicitly incentivized reviews (IIRs) and other explicitly incentivized reviews that do not contain the identified regular expressions and thus they were not detected by our

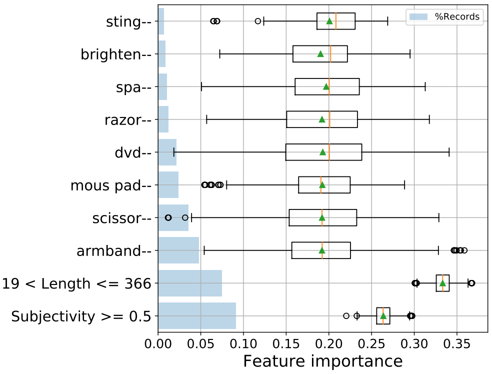


Figure 18. Importance of top 10 features for EIRs

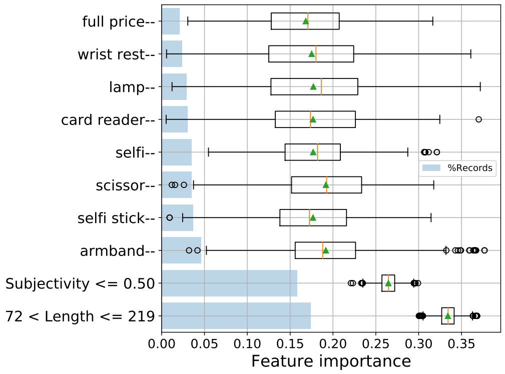


Figure 19. Importance of top 10 features for Normal Reviews

method. We randomly select 50,000 reviews (during 2016) from the DS1 dataset that are neither EIR nor normal reviews. After removing reviews with less than three words in the text, we kept 49,956 reviews. We use the trained classifier to determine whether any of these *unseen* reviews are classified as incentivized or normal reviews. The classifier flags 10,693 (21.4%) of these reviews as incentivized. Our manual inspection of the content of these reviews revealed that they can be broadly divided into two groups as follows:

Other Explicitly Incentivized Reviews: 2,154 (20.1%) of reviews labeled as incentivized contain a variety of different explicit patterns that was so sparse to be captured by our regex, e.g. "I had the opportunity to get it for my review", "received with a promotion rate".

Implicitly Incentivized Reviews (IIRs): We note that the absence of any explicit disclosure of incentives in the remaining reviews does not imply that they are not incentivized. We hypothesize that some of them are implicitly incentivized reviews (IIRs). However, as there is no strong signal to confirm whether these reviews are implicitly incentivized, we consider three different ways to verify our hypothesis as follows:

First, across all the remaining flagged reviews by our classifier, we consider the pairwise relationship between review-product and review-reviewer. We check each of these reviews against the following two conditions: *(i)* whether a review is associated with a product that had received at least one other EIR, or *(ii)* whether a review is provided by a user who has submitted at least one other EIR. We observe that 2,330 (21.8%) reviews are affiliated with both EIR reviewers and EIR products (i.e. meet both conditions) while 3,762 (35.2%) of them are only affiliated with EIR products and 534 reviews are only affiliated with EIR reviewers. Intuitively, meeting both conditions offers stronger evidence that a review could be IIR. Our manual inspection of reviews in these 3 groups confirmed this intuition. While reviews that met both conditions contain an indication of incentive (e.g. *for my honest result*, *promotional price*), reviews related only to products contained moderate hints (e.g. *I have to thank seller*).

Second, we compare the distribution of text and title length, word count, helpfulness, sentiment, subjectivity, readability, and star-rating of 38% of reviews that were labeled as incentivized but neither have explicit pattern nor pairwise relationship with other EIRs. Our analysis show that the distribution of these features for flagged reviews closely follow the corresponding distribution for EIRs, suggesting that these IIRs are flagged correctly. We also use Kruskal-Wallis test to verify the similarity in the distribution of these features for EIRs and flagged reviews. We observe that p is close to 0.0 for all features, except the title length where $p = 0.19$.

Third, since Amazon has warned to remove reviews that violate its guidelines, we examine whether the reviews tagged as IIRs have been removed from Amazon during the past two years. We noticed that 69% of these reviews have been removed from Amazon which indicate the problematic nature of these reviews. We emphasize the presence of the remaining 31% of the reviews does not imply that these are normal reviews.

3.7 The Current State of EIRs

On October of 2016, Amazon announced the “community guidelines to prohibit incentivized reviews". As we showed in Fig. 4, the daily number of submitted EIRs has significantly dropped after this announcement. In this section, our goal is to examine the status of previously submitted EIRs and related product as well as the ability of sellers to attract incentivized reviews.

We check the availability of more than half of randomly selected EIRs in DS2 in December of 2018, more than two years after we originally collected them. Our analysis showed that an absolute majority of these EIRs (98.5%) were removed from Amazon. Furthermore, when we tried to submit a review for a product that recently had a promotional campaign, we received the message shown in Fig 20 indicating the limitation to submit any reviews for this product due to unusual reviewing activity. This suggests that Amazon has actively limited the submission of new reviews for these problematic products.

We also inspected the content of the tiny fraction of remaining EIRs in the system and noticed that the content of roughly 20% of them was modified. Interestingly, half of these modified EIRs were shortened by removing the

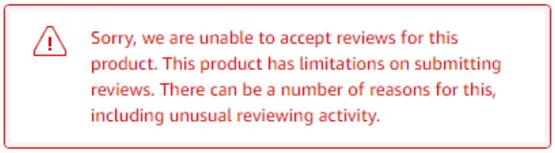

Sorry, we are unable to accept reviews for this product. This product has limitations on submitting reviews. There can be a number of reasons for this, including unusual reviewing activity.

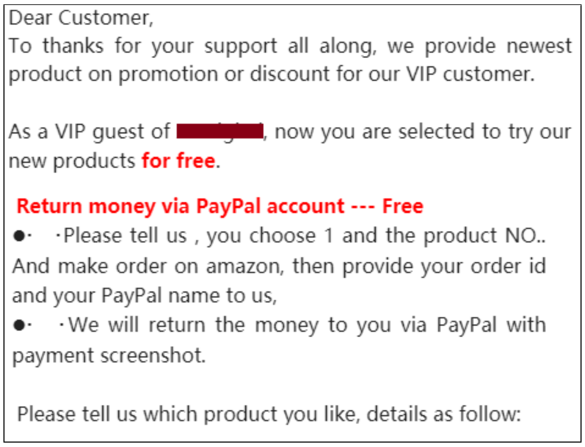
Figure 20. Amazon limits review submission for products with suspicious review activity

disclaimer of explicit incentive by the reviewer. This could be the reason that they were not removed. The other half of modified EIRs were short reviews that later added some personal experience (e.g. what is in the box, how to use the product, pros, and cons).

We performed the following experiment to verify whether and how Amazon sellers might attract incentivized reviews from users. To this end, we made a purchase on Amazon and received the product that included seller's instructions to contact them for receiving other products for a free or discounted price as a VIP customer. We emailed the customer service of this (and a few other) seller(s) indicating our interest to be a VIP customer and received a variant of the response showed in Fig.21. This response basically provides the instructions for users to buy a product on Amazon, submit their (incentivized) review, and then be reimbursed for their purchase through their Paypal account, i.e. receiving a product for free. Clearly, such an incentivized review does not need to contain any disclaimer and would not be detectable by Amazon. This is a clear indication that incentivized reviews still exist on Amazon but they are not explicit since exchanged information and money between sellers and reviewers are not visible to Amazon.

3.8 Conclusion

This chapter presented an application of machine learning techniques in a large scale networked system of sellers and shoppers to detect incentivized online reviews that can degrade the users‘ trust in the rating system. To that end, we detailed characterization of Explicitly Incentivized Reviews (EIRs) in two popular categories of Amazon products. A technique to detect EIRs was presented based on collected datasets from Amazon. We identified a large number of EIRs in Amazon along with their associated product and reviewer information.



Dear Customer,
To thanks for your support all along, we provide newest product on promotion or discount for our VIP customer.

As a VIP guest of [redacted], now you are selected to try our new products **for free**.

Return money via PayPal account --- Free

- ·Please tell us , you choose 1 and the product NO.. And make order on amazon, then provide your order id and your PayPal name to us,
- ·We will return the money to you via PayPal with payment screenshot.

Please tell us which product you like, details as follow:

Figure 21. Sample of a seller response to users who are interested in becoming a VIP customers that provides instructions for submitting incentivized reviews.

Using this information, we compared and contrasted various features of EIRs with reasonably normal reviews. We showed that EIRs exhibit different features compared to normal reviews and discussed the implications of these differences. Then, we zoomed into the temporal pattern of submitted EIR reviews for a few specific products and submitted reviews by a few specific reviewers. These temporal dynamics demonstrated whether/how promotional campaigns by a seller could affect the level of interest by other users and how reviewers could get engaged in providing EIRs. Finally, we illustrated that machine learning techniques can identify EIRs from normal reviews with a high level of accuracy. Moreover, such techniques can accurately identify other explicitly and implicitly incentivized reviews. We leverage affiliation of reviews with reviewers and products to infer their incentivized nature.

CHAPTER IV

FORECASTING NETWORK DATA STREAMS

A wide range of functionalities of a networked system can be observed as a data stream (time-series). Forecasting the future values of such data streams leads to more accurate resource scheduling and more effective monitoring, which improves the throughput of the system. However, due to the large amount of data that should be processed, domain-specific limitations and requirements, forecasting is not a trivial task. In this chapter, we focus on how machine learning models can be used to forecast real-time data streams in a wide-scale campus network and what are the potential practical challenges. Specifically, we examine various training-related aspects that affect the accuracy and overhead (and thus feasibility) of two popular types of models (SARIMA and LSTM models) used for forecasting real-world network data, streams in telemetry systems. In particular, we study the impact of the size, choice, and recency of the training data on accuracy and overhead and explore using separate models for different segments of a data stream (e.g., per-hour models). We discuss how these choices affect the accuracy of the models.

The content in this chapter is derived entirely from (Jamshidi et al., 2020b) as a result of collaboration with co-authors listed in the manuscript. Soheil Jamshidi is the primary author of this work and responsible for conducting all the presented analyses.

4.1 Introduction

Recent advances in the programmability of network data plane (e.g. Gupta et al. (2018)) enable network operators to monitor their desired traffic features at the line rate. For example, individual switches across a campus or data center network can emit a per-second data stream of loss rate (Y. Li, Miao, Kim, & Yu, 2016), flow arrival rate (Z. Liu, Manousis, Vorsanger, Sekar, & Braverman, 2016) or queue occupancy (X. Chen, Feibish, Koral, Rexford, & Rottenstreich, 2018) to a central collector as input to a telemetry task for detecting performance or security-related events (see § 4.2). The availability of network data streams coupled with the increasing complexity of today's networks motivates a data-driven approach for network management and security that can be usually cast as a prediction (Lazaris & Prasanna, 2019; Mirza et al., 2010; Yang, Sun, Li, Lin, & Tian, 2019) or a classification (Michael, Valla, Neggatu, & Moore, 2017; Yamansavascilar, Guvensan, Yavuz, & Karsligil, 2017; J. Zhang, Chen, Xiang, Zhou, & Wu, 2015) problem. In particular, forecasting techniques are used to predict the likely future values of a network data stream based on its past values. The impressive success of deep learning (e.g. recurrent neural network or RNN) techniques in other fields combined with their ability to learn short- and long-term dependencies in data streams make them a promising candidate for forecasting network data streams. This, in turn, has motivated several prior studies to rely on different neural networks (NNs) or statistical models to forecast various data streams in wireless or mobile networks, ranging from the throughput of individual TCP connections (Mei et al., 2019) and intensity of (per user and aggregate) traffic (Azari, Papapetrou, Denic, & Peters, 2019) to aggregate traffic (Nikravesh et al., 2016) or traffic at a base station (Zang et al., 2015). These studies typically consider a significant volume of past data for training which is time-consuming and typically requires significant amounts of computational resources. In addition, they evaluate the overall accuracy of a forecasting model in an off-line manner. To the best of our knowledge, prior studies on forecasting network data streams have not addressed the following important issues regarding the feasibility and deployability of the proposed models in the context of a network telemetry system (e.g. for anomaly detection) that operates in a streaming fashion: *(i)* How can the volume or selection of training data be adjusted to reduce training overhead without degrading forecasting accuracy? *(ii)* How often does a model have to be retrained to maintain a sufficient level of accuracy? *(iii)* Does the accuracy of forecasting models change across different segments of a data stream, and if so, in what manner? *(iv)* How do model selection (i.e. statistical vs. NN) and model casting (i.e. generic vs. per hour) affect the answers to the above questions?

In this chapter, we tackle the above issues regarding the feasibility and deployability of forecasting models. In particular, we adopt the following methodology to compare the accuracy of a type of deep learning model (i.e. long short term memory (LSTM) models) with the accuracy of a popular statistical forecasting technique (i.e. seasonal autoregressive integrated moving average (SARIMA) models). We consider two network data streams, namely a per-second flow arrival rate process for all incoming flows (RAF) and all incoming web flows (RWF) to a campus network, respectively. These data streams represent the type and resolution of data that is commonly captured by modern telemetry systems (Gupta et al., 2018; Santanna et al., 2015). We show in § 4.4 that these data streams exhibit very different characteristics and thus enable us to demonstrate the effect of these characteristics on the forecasting accuracy. We consider both generic and per-hour versions of both LSTM and SARIMA models for six evenly spaced hours to assess how different characteristics of data streams affect the accuracy of forecasting the next five seconds of data streams. For per-hour models, we explore how the volume, selection, and recency of the training data can affect the accuracy of the resulting model. This, in turn, reveals opportunities to reduce training overhead with minimal or no effect on the forecasting accuracy.

As the main contribution, we report in this chapter on a number of empirical findings from our analyses that offer valuable insights for deploying the considered models in practice. First, we observe that for per-hour models, increasing the volume of training data beyond 24 hours of a recent window or similar past instances (i.e. same hour, the same day of the week) does not improve the accuracy of the model but linearly increases the training overhead. Second, we find that per-hour models that are trained with a 24-hour data stream exhibit a comparable accuracy with a generic model that is trained with 30 days of training data. In practical terms, these findings show that we can significantly decrease training overhead without compromising the accuracy of LSTM models. Third, observing that changing the recency of the training data by a few weeks does not affect the accuracy of our per-hour forecasting models suggests that LSTM models do not require frequent (re)training. Fourth, we notice that for our RWF data stream, the prediction accuracy of all per-hour models is lower during the night hours and higher during day hours. This observation shows that certain segments of the data stream are inherently more difficult to forecast than others. Fifth, in the case of our RAF data stream, its more bursty behavior compared to the RWF data stream tends to result in higher accuracy for LSTM models in forecasting RWF compared to SARIMA models. Finally, all the models show wider variations in forecasting accuracy across different samples of our RAF data stream. It is important to note that while the reported findings depend specifically on our considered data streams, the described methodology is applicable to any data stream and should be part of an in-depth assessment of the practical issues that arise when deploying any statistical or learning models.

The rest of this chapter is organized as follows. In § 4.2, we present an example telemetry task to illustrate two requirements for a forecasting model in such a setting. § 4.3 provides some background on LSTM and SARIMA models. Our empirical approach is described in § 4.4. We assess the feasibility and accuracy of forecasting models in § 4.5. We summarize and outline future work in § 4.6.

4.2 Illustrative Example of Utilizing Forecasting Models

We present an overview of the anomaly detection (telemetry) task that incorporates a forecasting model to illustrate the implications of task requirements and the characteristics of network data streams on training and configuring the model. Consider a programmable switch (e.g. Tofino (2020)) that monitors a collection of desired traffic features at the line rate and emits a separate, fine-grained (e.g. per second) data stream for each feature to a remote collector. The availability of high resolution (per second or short timescale) data streams from the data plane telemetry systems enables forecasting models not only to capture finer variations in traffic but also to facilitate faster detection of anomalies. A forecasting model first requires using a history of these data streams for initial training. Then, a trained model can be deployed at the collector and uses the most recent n values of the data stream to forecast the next h values. Then, if the gap between the forecasted and actual h values is larger than the error in the model, this could be viewed as an indication of an anomaly. Subsequently, the telemetry task may trigger further examination of other traffic features and invoke proper actions on forwarding switch pipeline (e.g. dropping or re-routing the relevant packets) to mitigate the problem. Furthermore, depending on the characteristics of the captured data stream, the model may require periodic retraining such that its forecasted values remain sufficiently accurate.

This example illustrates two important requirements for practical deployment of learning models into a telemetry system:

- **R1**: Forecasting models are often trained using a large volume of past data (a few days to weeks or months) which typically takes a long time (hours to days). However, in telemetry systems, a long history of a data stream may not be available and a significant training overhead may not be feasible. To address this requirement, we explore how the training overhead can be reduced by limiting the volume (and selection) of training data without affecting the accuracy of the resulting model.

- **R2**: Since network data streams may evolve (over time), we need to periodically re-train a forecasting model to maintain sufficiently high accuracy. The duration of a re-training should be much shorter than the period for refreshing the model. We examine the effect of the recency of training data on the accuracy of forecasting models to shed light on the required frequency of retraining and its relationship with (re)training overhead.

4.3 Forecasting Methods: Background

This section provides a brief background on the forecasting models that we consider in this study.

LSTM. Long short-term memory (LSTM) is a recurrent neural network (RNN) architecture (Hochreiter & Schmidhuber, 1997) that has been widely used for forecasting in various domains, including networking (Adebiyi, Adewumi, & Ayo, 2014; Azzouni & Pujolle, 2017; Janardhanan & Barrett, 2017; X. Wang et al., 2017). The key feature of the LSTM model is its ability to capture potential long-range dependencies in the data stream. The LSTM model has several parameters related to its architecture, optimizer, and training approach that should be properly configured such as the number of stacked layers, number of hidden nodes, activation function, dropout, and the number of passes over the data during training (i.e. epochs). To train an LSTM model, the data stream X is divided into two separate sets: M consecutive values of ($X(t_0 - M)$ to $X(t_0 - 1)$) for the training set, and the immediate next N values ($X(t_0)$ to $X(t_0 + N)$) for the testing set. We consider non-overlapping training and testing sets, and further split each set into samples using a rolling window that has been shown to be an effective strategy (Mohamed et al., 2015; Ordóñez & Roggen, 2016; Radford, Apolonio, Trias, & Simpson, 2018; Saleh, Hossny, & Nahavandi, 2017; Yu, Li, Shahabi, Demiryurek, & Liu, 2017). Each window (of length $(A+B)$ consecutive values) is considered as a *sample.* In each sample, the first A values are used as *history* to forecast the next B values. The LSTM output size (S), is one of the LSTM's parameters. If the forecasting horizon H (i.e. the number of data points we expect to forecast) is larger than S, then we have to forecast by rolling the window $\frac{H}{S}$ times and using either the forecasted or actual values of the data stream as history for the next S values. All samples are divided into random and mutually exclusive batches of a certain size where each batch is used for a separate round of training until all samples are utilized. This training process can be repeated multiple times (epochs) to improve the accuracy of the model.

This description reveals several training parameters for an LSTM model: the relative size and selection of training and testing sets, window size, prediction size

(LSTM output), window overlap, batch size, and the number of epochs. We found optimal values for each during our tuning process.

(S)ARIMA. Auto-Regressive Integrated Moving Average (or ARIMA) is a popular statistical technique for forecasting stationary data streams (Ariyo, Adewumi, & Ayo, 2014; Contreras, Espinola, Nogales, & Conejo, 2003; Ediger & Akar, 2007; Gilbert, 2005). ARIMA has several configurable/tunable components: the auto-regressive component (p) that specifies the number of lags (past values) in the model, the integrated component (d) that represents the degree of differencing, a moving average component (q) that represents the error of the model as a combination of previous error terms. Subsequently, several variants of ARIMA were also proposed. For example, to model time series with periodic characteristics, Seasonal ARIMA (or SARIMA) model (Box & Jenkins, 1976) was proposed. SARIMA incorporates seasonal auto-regressive (P), differencing (D), and moving average (Q) components as well as a seasonal frequency (s). Given the inherently periodic (daily, weekly) characteristics of most networking data streams, we primarily focus on the SARIMA model in this study.

The process of training a SARIMA model is as follows: SARIMA models assume that the input data is stationary. The stationarity of the time series can be achieved via transformation (e.g. logarithms) to stabilize the time series *variance* and differencing to eliminate the *trend*. Besides, the decomposition can help to de-seasonalize the time series if necessary (Hyndman & Athanasopoulos, 2018). We use Augmented Dickey-Fuller (ADF) test (Enders, 2008) as a unit root test to check for deterministic trend stationarity as well as Kwiatkowski-Phillips-Schmidt-Shin (KPSS) test (Kwiatkowski, Phillips, Schmidt, & Shin, 1992) to complement the unit root test. On our data stream, the ADF test confirms the stationary nature of data with 99% confidence while KPSS test indicates that the timeseries are not stationary. In such cases, it is suggested to apply differencing. The KPSS test passes when applied on differenced data.

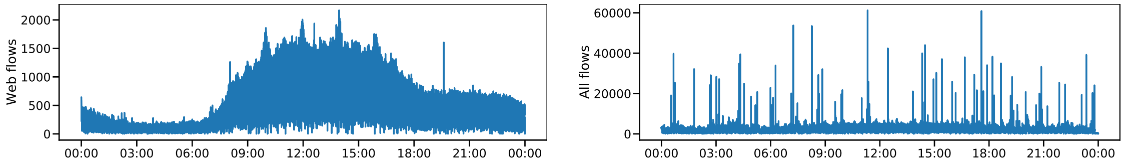
To identify the best combination of model parameters, we consider the range of values for p, d, and q obtained from the auto-correlation function (ACF) and the partial auto-correlation function (PACF) plots (not shown due to space constraints). Then, we train separate SARIMA models for a different combination of parameters in parallel and select the best model based on their performance on the validation set.

4.4 Methodology

In this section, we discuss our methodology for exploring our motivating questions on incorporating forecasting models into anomaly detection systems. We start by presenting the network data streams and the selection of forecasting models that we consider in this study as well as our training and testing strategies for these models.

4.4.1 Network Data Streams. We focus on flow arrival rate per second (i.e. number of unique incoming network flows that are observed in each second) as our target data streams since it is used as the input of telemetry tasks (e.g. Narayana et al. (2017); Santanna et al. (2015); Sutiene, Vilutis, and Sandonavicius (2011)). To this end, we use un-sampled NetFlow data for all the connections between the University of Oregon campus and the Internet to extract the rate of (incoming) web flows (RWF) and rate of (incoming) all flows (RAF) per second. Our NetFlow dataset covers a 10 month period from 1/5/2018 till 28/2/2019 where each daily segment of our dataset represents on average 8.8 TB of incoming traffic, associated with 200 million flows, from 3.6 million unique source IPs with 39k unique sources (Yeganeh, Rejaie, & Willinger, 2017).

Fig. 22a and 22b present the variations of our two data streams in a typical day (2018-09-26) and illustrate that these two data streams exhibit very different characteristics as follows. First, RWF shows significantly smaller variations that are dominated by a pronounced diurnal pattern compared to RAF. Therefore, forecasting these data streams is likely to present different challenges. Second, the RWF data stream exhibits a high, low, and moderate degree of variations during the night (0-8), day (8-16) and evening (16-24) hours, respectively. However, the degree of variations in the RAF data stream is very similar across all hours. This observation motivates us to consider *training a separate forecasting model for different hours of the day could lead to a higher accuracy.*



(a) RWF

(b) RAF

Figure 22. Sample daily variations of our two target flows arrival rates (per second).

To examine the (dis)similarity of both data streams across different days, we consider 13 scale-invariant attributes of each daily segment for both data streams (proposed by Kang, Hyndman, and Li (2019))[1], apply principal component analysis (PCA) to identify the top two principal components for each data stream. The blue dots and orange crosses in Figure 23 present the values of the top two principal components for RWF and RAF data streams in separate days, respectively. From

[1] These attributes include trend, spike, linearity, curvature, entropy, skewness, and ACF lags, compared to scale variant attributes such as mean, median, minimum, and maximum

this figure we make two key observations. First, while both data streams capture flow arrival rates, RAF data stream exhibits wide variations across different days while the characteristics of RWF data stream across different days are more consistent. Therefore these two data streams represent very different input network timeseries for our forecasting models. Second, the two pronounced clusters of blue dots in Figure 23 are related to weekdays (solid black line) and weekends (dotted red line). This evidence indicates that the RWF data streams have distinctly different characteristics on weekdays compared to weekends.

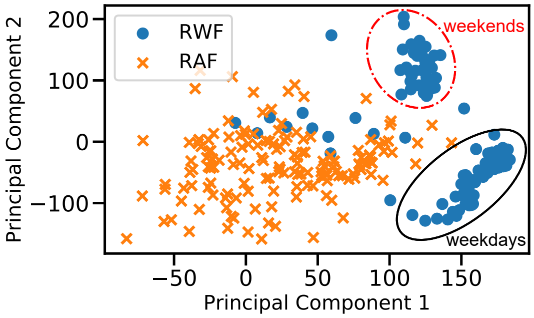


Figure 23. Comparison of per-second data streams using principal components.

4.4.2 Forecasting Models. To examine how the type (i.e. SARIMA vs. LSTM), volume, and selection of training data and other inputs of a model affect its forecasting accuracy, we consider the following four models. Using a short forecasting horizon improves model accuracy but limits the time to react to a detected anomaly and vice versa. To strike a balance between these two opposing requirements, we have examined 5, 10 and 20 seconds (5, 10, 20 values) forecasting horizons and selected 5-second forecasting horizon in all of our models.

LGW(w): This LSTM (L), generic (G) model forecasts any 5-second segment of a data stream by using the recent (w) hours window (W) of a data stream for training. A common practice in training an LSTM model is to use a large volume of a data stream for training and explore whether adding other features improve the accuracy of the models. LGW models represent this common approach for using LSTM models. We use a 30-day recent window of each data stream to train LGW models (i.e. w=30*24) [2].

LHW(x, w): This per-hour (H) LSTM (L) models forecast any 5-second segment of a data stream in a *specific target hour* x using the past w hours of the data stream for training. We train a separate model for six evenly spaced target hours (i.e. x is set to 3, 7, 11, 15, 19, 23) to explore the accuracy of our model for forecasting different parts of the data streams. By setting w to 1, 10, 24, and 48 hours, we also examine how the volume of training data affects the accuracy of these per-hour models. We also consider older windows of training data (from prior weeks) to explore the effect of data recency on the model accuracy.

LHI(x, i): This LSTM (L) model forecasts any 5-second segment in a specific target hour (H) x using the past i instances (I) of the target hour x in the data stream for training. An instance of a specific target hour x is defined by its hour-of-day and its day-of-week. For example, to train a model for 7am hour on a Monday, we use the 7-8am segment of the data stream from i prior Mondays for training. This training strategy is intuitively motivated by the repeating weekly pattern of some network data stream, such as RWF, which suggests that the most relevant training data is the past instances of the same hour.

SHW(x, w): This SARIMA (S) model forecasts any 5-second segment in a *specific target hour* (H) x using the past w hours of the data stream for training (similar to *LHW(x,w)*). By changing w, these models represent a statistical forecasting technique with a different volume of training data. Note that SARIMA model can

[2] We considered other additional input features such as day-of-week and time-of-day. However, they did not improve the accuracy of LGW models.

only use the most recent window of data for training while LSTM can rely on any past window of data for training.

This collection of models enables us to compare generic and per hour models while exploring the effect of the volume, recency, and selection of training data on the overall accuracy of forecasting models.

4.4.3 Tuning Models.

We take the following steps to properly tune each one of the selected models.

LSTM Models. We tune individual LSTM models by examining the accuracy of the model across thousands of different configurations and training parameters using random search on the validation data that is separate from the training and testing set. We then select the model that exhibits the highest validation accuracy. We leverage the sliding window approach to break both training and testing datasets into samples using the following parameters: window size=150[3], window overlap=135 and forecasting horizons=5 values. These parameters result in 233 samples in each hour of our data streams. The number of epochs is 300 for all models and batch size is 300 for LGW and 150 for other LSTM models. We use checkpoints to identify the model with the lowest validation loss. We utilize a Keras implementation of LSTM with Tensorflow backend. Table 3 summarizes the main hyperparameters including learning rate (LR), activation function (AF), drop rate (DR), number of hidden layers (LY), number of hidden nodes (NHN), and optimizer (OPT) for our tuned LSTM models.

We use mean square error ($MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$) as the loss function in our training process where n, y_i and $\hat{y}_i$ denote the forecasting horizon size, the actual and predicted values, respectively.

[3] Given the selected window size (150) for our samples, having more than 300 samples/hour results in increasing overlap between samples.

Table 3. Hyperparameters of our LSTM models.

		LR	AF	DR	LY	NHN	OPT
RWF	**LHW/I**	0.005	tanh	0.1	5	64-32(x4)	Adam
	LGW	0.003	tanh	0.2	3	256	Adam
RAF	**LHW/I**	0.005	ReLU	0.2	4	32-16(x3)	Adam
	LGW	0.005	ReLU	0.2	4	64	Adam

SARIMA Models. We train a model for 2,500 different combinations of parameters ($5p * 5q * 5P * 5Q * 2d * 2D \ = \ 2,500$) and perform grid search to identify the parameters of the best model. To make sure that models capture the patterns in the input data stream, we confirmed that the residuals follow the normal distribution and have no auto-correlation (using *Ljung-Box* test (Ljung & Box, 1978)). Table 4 presents the final configuration for the SHW models of different hours along with their associated training time.

Table 4. Configuration and training time of SARIMA (SHW$(x, 24)$) models for both data streams.

Target hour	**Data stream**	**(p,d,q)x(P,D,Q,s)**	**Train time (minute)**
03	RWF	(1, 0, 3)x(3, 1, 3, 15)	62.2
07	RWF	(3, 0, 2)x(2, 1, 3, 15)	129.52
11	RWF	(3, 1, 2)x(2, 1, 3, 15)	124.69
15	RWF	(1, 1, 3)x(2, 1, 3, 15)	156.89
19	RWF	(3, 0, 3)x(3, 0, 3, 15)	63.81
23	RWF	(3, 1, 1)x(2, 0, 3, 15)	18.27
03	RAF	(3, 1, 1)x(1, 1, 1, 6)	60.42
07	RAF	(3, 1, 1)x(1, 1, 2, 6)	48.33
11	RAF	(0, 1, 3)x(2, 0, 3, 6)	16.79
15	RAF	(0, 1, 3)x(2, 1, 3, 6)	107.24
19	RAF	(1, 0, 3)x(0, 1, 3, 6)	82.59
23	RAF	(0, 1, 3)x(2, 1, 3, 6)	122.37

4.4.4 Testing Models.

For testing each model, we consider 300 randomly selected points in each test hour and use the model to forecast the next immediate 5 seconds (i.e. our forecasting horizon) of the data stream. We use Root Mean Square Percentage Error (RMSPE) for evaluating the accuracy of our forecasting models across all samples as follows: $RMSPE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\frac{(y_i - \hat{y_i})}{y_i}^2}$ where n, y_i and $\hat{y}_i$ denote the forecasting horizon, the actual and predicted values, respectively. The normalized nature of RMSPE makes it scale-invariant and interpretable which is more appropriate for our purpose (Hyndman & Koehler, 2006). The overall accuracy of each model is presented with the summary distribution (box-and-whiskers plots where the box shows the quartiles while whiskers show 5^{th} and 95^{th} percentiles) of RMSPE across 300 random samples in each target hour. The variations of error for each model across different hours reveals the effect of temporal characteristics of data streams on the model accuracy.

For each hourly LSTM model LH*(x, w), we train seven separate models for each target hour in seven consecutive days (11/12/18 to 11/18/18)[4], test them on 300 samples in the target hour, and present the summary distribution of RMSPE for all (7*300) samples for each target hour. For LGW model, we train a single model but similarly test it on each target hour across 7 days to present the summary distribution of error for that hour. Therefore, our results are not biased towards a specific day of the week.

4.5 Assessing the Practicality of Forecasting Models

In this section, we assess the practicality of forecasting models in light of the two requirements described in § 4.2.

[4] This is a regular week that school was in session.

– Training a per-hour model with any size of the past segments of the data stream that is at least 24 hours long leads to good forecasting accuracy. The recency of a (sufficiently long) training dataset does not significantly affect the accuracy of the model.

– SARIMA exhibits a wide range of accuracies: it is comparable with LSTM *only* in certain times of the day.

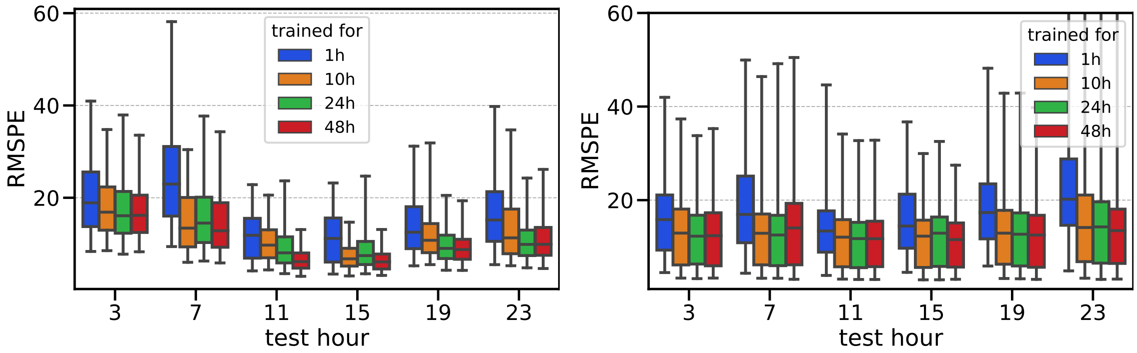
– One can use an LSTM model that is trained for short-term horizons (e.g. 5 seconds) to forecast long-term horizons (e.g. 80 seconds) *without* re-anchoring the model. In general, it works best for up to 75% of the samples; for the remaining samples, the errors are larger.

– LSTM models can closely track sudden changes in the data stream. SARIMA, on the other hand, cannot.

4.5.1 Impacts of Volume and Selection of Training Data.

We evaluate the effect of variations of the data stream as well as the volume, recency, and selection of training data on the accuracy of forecasting models. The goal is to shed light on requirement R1 mentioned in § 4.2.

Volume of Training Data. First, we explore the question of *whether the volume of the training dataset affects the accuracy of a per-hour model?* Figure 24a and 24b present the summary distribution of forecasting error for RWF and RAF data streams across different hours using LHW models. For each hour, we show the error for four models that are trained with 1, 10, 24, and 48-hours of most recent data stream. These two figures show the following points: First, for both data streams, increasing the amount of training data initially improves the accuracy of forecasting for up to 24 hours. However, increasing the training data beyond 24 hours has a diminishing return in the accuracy of the model for most hours (except 11 and 15 hours for RWF). Second, *the accuracy of the best-trained model for RWF varies across different hours* (Figure 24a). In particular, the forecasting error during the night hours is the highest and during the day hours is the lowest. In contrast, the accuracy of models for RAF is very similar across all hours.

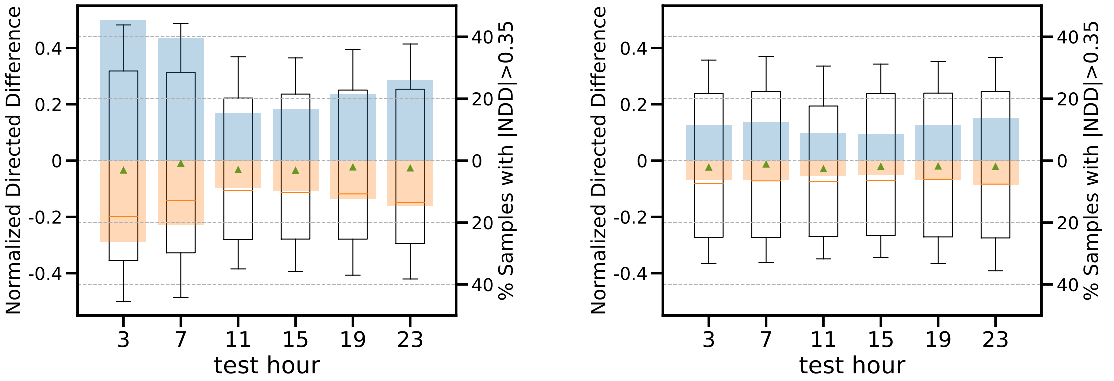


(a) RWF

(b) RAF

Figure 24. Effect of the volume of training data on the accuracy of LHW(x, w) model.

To explain the difference in the accuracy of models across different hours, we examine the variations of both data streams during different hours of the day. To this end, we measure the normalized directed difference (NDD) among (150) values in each test sample that is defined as follows $NDD = \frac{max(X)-min(X)}{max(X)} * sign(argmax(X) - argmin(X))$ where X is a test sample, and $argmin(argmax)$ denotes the index of the min and max values, indicating the order in which min and max values are observed (i.e. the positive/negative direction of major change between these values). The box plots in Figure 25 present the summary distribution of NDD values across all samples in each of 6 target hours over 7 days. To further focus on larger NDD values, we also show the fraction of samples with positive (negative) NDD values for each hour that are larger than 0.35 with a blue (orange) bar using the right y-axis. The plots in Figure 25 illustrate that the normalized changes across values of individual samples are larger in all hours of RWF data streams compared to RAF data stream. In particular, hour 3 and 7 of RWF exhibit the largest normalized variations. While it is not trivial to determine which specific aspects of a data stream affects the accuracy of a forecasting model, we believe that the larger variations in specific hours offer a plausible explanation for lower accuracy of our models for those hours.



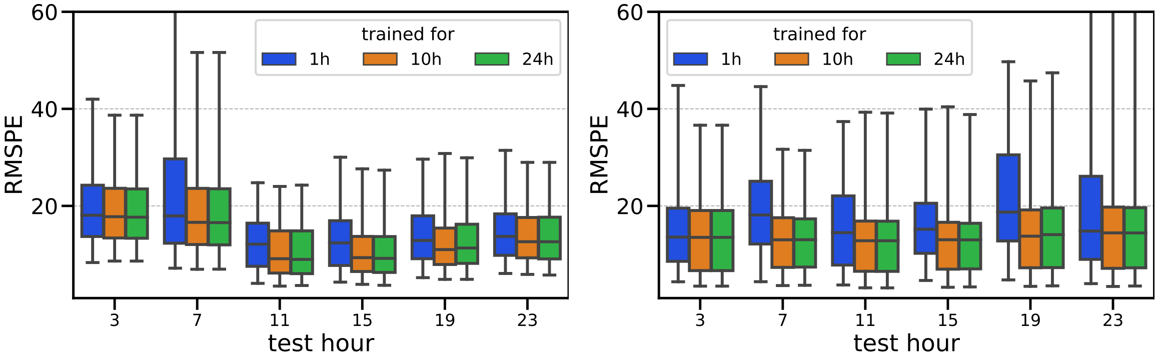
(a) RWF

(b) RAF

Figure 25. Distribution of normalized directed difference (NDD) of test samples.

For the rest of the analysis, we consider the LHW models that are trained with the recent 24-hour window of the data stream for these six target hours.

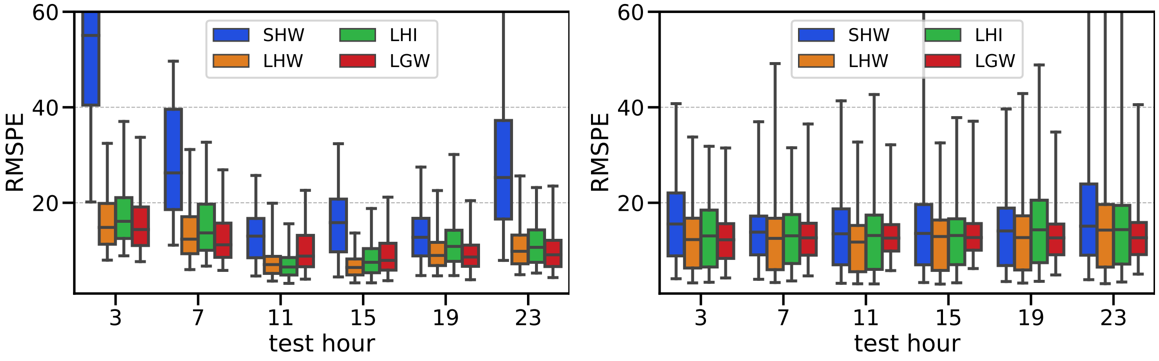
Selection of Training Data. Another important question is *whether the selection of training data affects the accuracy of a model?* We use LHI (x, i) models for each target hour that is trained with 1, 10, and 24 segments of data from prior instances of the same target hour. For example, an LHI (7,10) for a Monday uses the 7-8am segment of the data stream from 10 prior Mondays for training. Figures 26a and 26b present the accuracy of the LHI models for forecasting both data streams across all six target hours that are trained with 1, 10, 24 past instances of the target hour. These results show that *increasing the number of past instances of* *training segments from 1 to 10 improves the accuracy of both models but adding more training data has no measurable impact..*



(a) RWF (b) RAF

Figure 26. Effect of the selection of training data on the accuracy of LHI(x, i) model.

Head-to-Head Comparison of Different Models. We now compare the accuracy of all four models—LGW(30*24), LHW(x,24), LHI(x,24) and SHW(x,24)— for forecasting a 5-second horizon of different target hours of RWF and RAF data streams in Figures 27a and 27b, respectively. This figure illustrates a few important points: First, LHI (x,24), LHW (x,24), and LGW (24*30) exhibit a comparable accuracy across all hours of RWF data streams despite a significantly smaller amount of training data for LHI and LHW models. Note that the LGW model is simply a special case of the LHW model that uses 30 times more training data. Second, the accuracy of SHW models is lower particularly for hours that are difficult to forecast (i.e. 3, 7, and 23). Third, the relative pattern of changes in accuracy across different hours is very similar for RWF models – lowest accuracy in night hours, highest accuracy for day hours, and moderate accuracy in the evening. Fourth, on RAF data stream, SHW has only a slightly higher error compared to different LSTM models. All LSTM models have a very similar accuracy on RAF data stream but LGW exhibits much lower variations in error across different samples. Note that a commonly reported measure of accuracy (mean or median error) does not reveal this difference in the variations of accuracy.



(a) RWF

(b) RAF

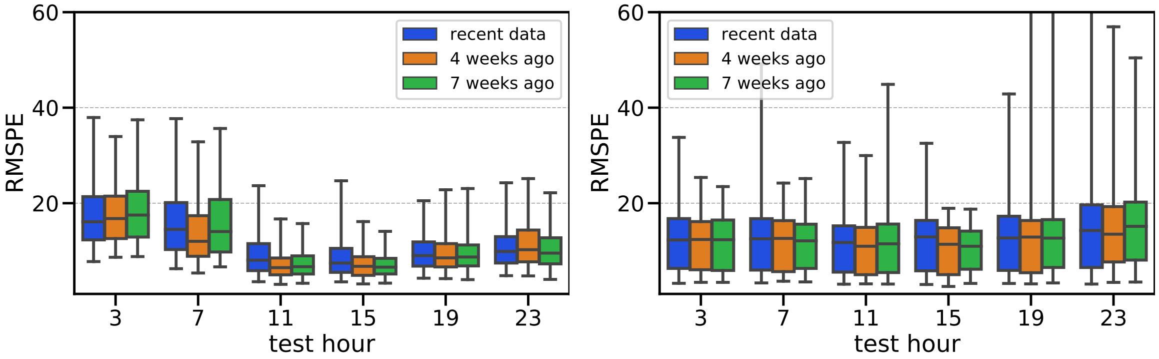
Figure 27. Comparison of four models on different hours of RWF and RAF data streams.

Fifth, comparing all models across both data streams show that LSTM models have a similar accuracy on both data streams during night (and early morning) hours (3, 7) but higher accuracy on forecasting RWF data stream in all other hours. Interestingly, while LSTM models show a similar accuracy for RWF data stream at night hours and RAF data stream at all hours, SHW models have much lower accuracy on RWF data stream at night hours than RAF data stream. This suggests that the LSTM models are more capable to forecast RWF data stream during the night hours (3 and 7) despite its larger variations (as we reported in Figure 25). Later in this section, our examination of the short term pattern of error offers more insight about this problem with the SARIMA models.

4.5.2 Impacts of Recency of Training Data and Training Overhead.

We next seek to evaluate the recency of training data and training overheads and how they affect the accuracy of forecasting models to shed light on R2 mentioned in § 4.2.

Recency of Training Data. We now explore the question of *whether the recency/freshness of training data affects the accuracy of the forecasting?* More specifically, does it make any difference if we train a LHW model with different 24-hour segments of the data stream? Figures 28a and 28b depict the accuracy of LHW (x,24) models for forecasting the six target hours of both data streams using three different training datasets for each model: *(i)* the most recent 24 hours of the data stream (labeled *recent data*), *(ii)* the same 24 hours of the data stream from *4 weeks ago*, and *(ii)* the same 24 hours of the data stream from *7 weeks ago*. Surprisingly, we observe that *the recency of a (sufficiently long) training dataset has a rather minor (or no) effect on the accuracy of the model for both data streams*. This finding suggests that *a LHW model that is trained with 24 hours of the data stream has observed a sufficiently rich set of variations and does not need to be retrained frequently in the absence of any major data drift.*



(a) RWF

(b) RAF

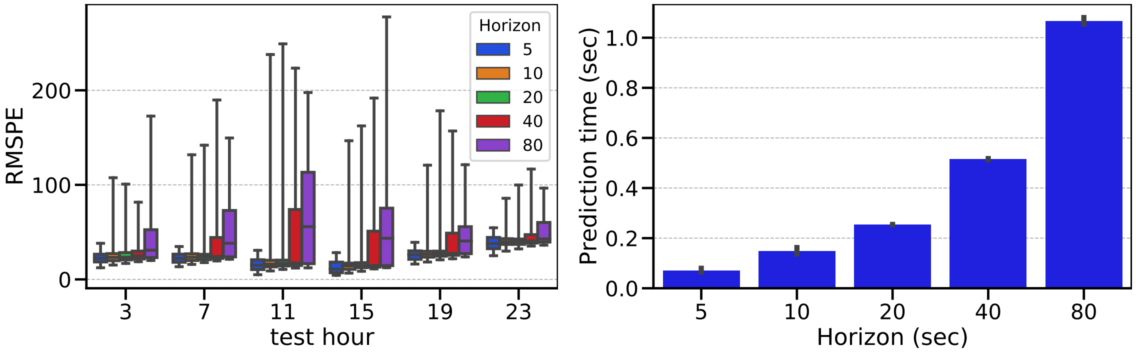
Figure 28. Effect of the recency of training data on the accuracy of LHW model for RAF data stream.

Training Overheads. Table 5 presents the total training time (with 300 epochs) for LGW and LHW/LHI (with different volume of training data) forecasting models of both data streams on both CPU (using two Intel Xeon Gold 5218) and GPU (using GeForce RTX 2080 Ti with 11GB GDDR6 memory). Table 5 shows that (i) training time linearly increases with the volume of training data and it is 7–26x faster on a GPU than a CPU, (ii) training a model for RWF takes 2–3x longer than RAF data stream, and (iii) it is feasible to retrain a new LHW/LHI model on a daily basis using GPU or CPU whereas LGW model can be retrained only on a daily (weekly) basis using GPU (CPU). The training times for hourly SARIMA models (i.e. SHW(x, 24)) on CPU are reported in Table 4. We observe that the training time for both data stream varies between 16-160min across different hours. This indicates that these model can be (re)trained on a daily basis.

Table 5. Total training time of LSTM models.

Model	**Volume of Training Data (hour)**	**RWF training time (minute)**		**RAF training time (minute)**	
		GPU	**CPU**	**GPU**	**CPU**
LHW/I	1	0.4	10	0.25	6.5
LHW/I	10	5	50	1.2	20
LHW/I	24	10	105	5	40
LHW/I	48	20	200	11	76
LGW	30*24	667	8,550	190	3,850

Longer Forecasting Horizon. The results presented so far are based on the forecasting horizon of 5-second horizons of the corresponding data stream by each model. If a telemetry task requires a longer forecasting horizon, we have two options: *(i)* “re-anchoring" the model every 5 seconds by using the past 150-second values of the data stream, or *(ii)* “rolling over" the model to forecast multiple 5-second intervals by feeding the forecasted values back to the model Figure 29a presents the accuracy of the latter option, namely rolling the model over, for forecasting longer horizons of RWF stream using the LHW model for different target hours. Note that the range of the Y-axis for this figure is much larger than our prior plots. Figure 29a shows that the typical accuracy of this roll-over strategy in all hours is clearly degraded as we increase the forecasting horizon. To complement this result, Figure 29b shows the effect of forecasting horizon on inference latency (prediction time). We can observe that the prediction time linearly grows with the forecasting horizon but remains generally low, e.g. roughly 1.1 seconds to predict the next 80 seconds of the data stream.



(a) On accuracy

(b) On prediction time

Figure 29. Effect of longer forecasting horizon (with roll over strategy) on model accuracy for RWF data stream

Short Term Pattern of Error. Our analyses have primarily considered the overall notion of error based on the RMSPE measure. In this subsection, we explore the temporal pattern of the forecasting error by SHW and LHW models to examine how closely it tracks sudden changes in the original data stream. Figure 30 depicts a 120-second segment of data stream along with the forecasted values by both LHW and SHW models (with 5-second forecasting horizon) using re-anchoring and roll-over strategies. This figure clearly demonstrates that the LHW model generally tracks the variations in the data stream, especially by the re-anchoring strategy. In particular, forecasting longer horizons (beyond 20 seconds in this example) even with the LSTM model can lead to a large error when the rolling strategy is used to extend the forecasting horizon. The forecasted values by the SARIMA model simply represent average values of the data stream and do not seem to track its

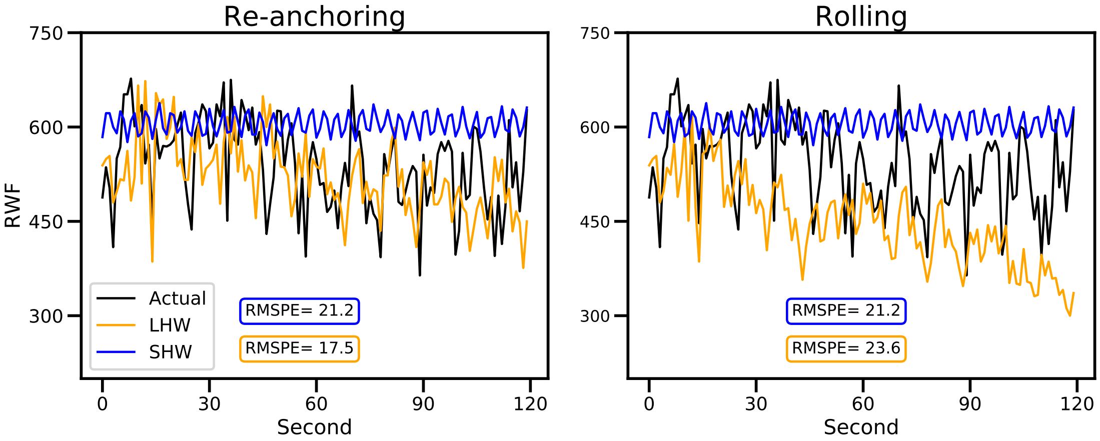
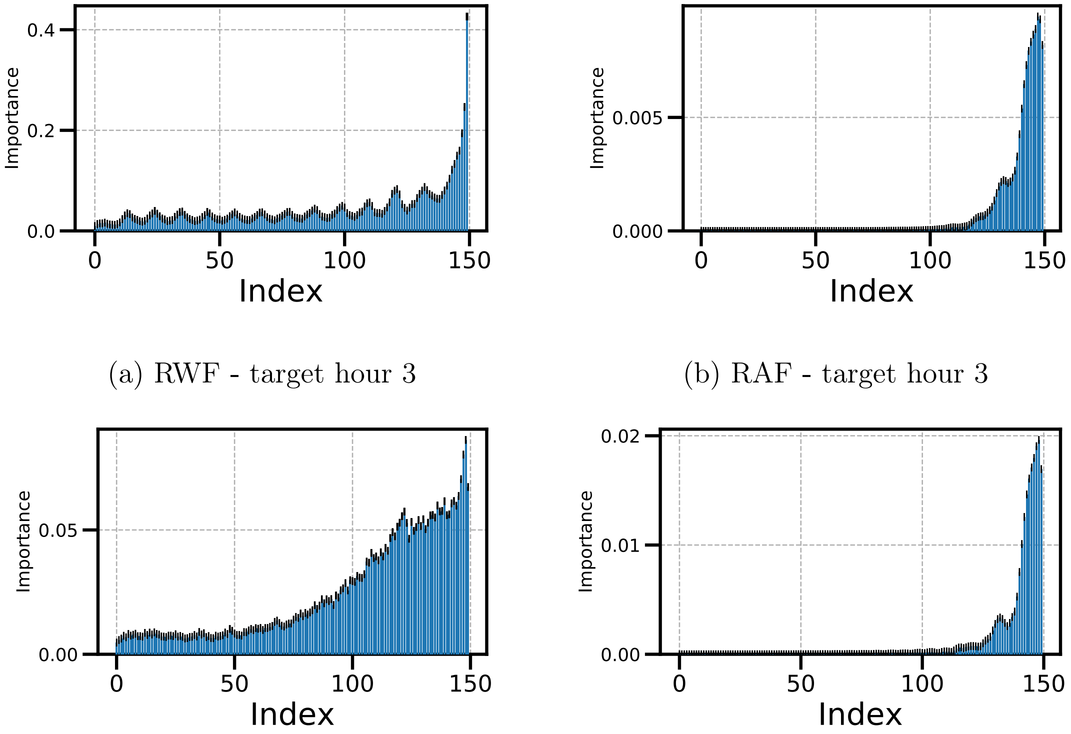


Figure 30. Effect of prediction strategy on accuracy.

variations. *In summary, the LSTM models coupled with re-anchoring strategy offers the most accurate forecasting results for our data streams.*

Relative importance of data points. In addition to gain insight on the importance of data points and if the weight of input data points is equal for the model or not, we leverage model interpretation methods, mainly the Saliency method (Simonyan et al., 2013) that is implemented in DeepExplain framework (Ancona, Ceolini, Öztireli, & Gross, 2017).

We use the test samples as input and the outcome is a score vector, one value per feature per test sample. Then, we can aggregate the scores across all cases by reporting the mean and standard deviation of importance score. This aggregated value can be used to rank the input features per data stream. Using this method, we calculate the importance of the past data points for the LHW models and the results are shown in Fig 31. The figure suggests that the immediate past 20 data points are among the most important historic data points for the model. For the RWF data stream, there is a difference between late night (3) in Fig 31a and evening hours (15) in Fig 31c while the importance stays similar across different hours for RAF (Fig 31b and 31d). Also there is a pronounce seasonality in importance of historic data points in case of RWF signal. For RAF, the importance of prior data points drops quickly beyond past 50 data points.



(a) RWF - target hour 3

(b) RAF - target hour 3

(c) RWF - target hour 15

(d) RAF - target hour 15

Figure 31. Importance of past data points for model prediction

4.6 Conclusion

The sheer volume of network data stream coupled with the increasing complexity of today's network and innovation in switch data planes motivate forecasting techniques as the key ingredient to automate network management and security tasks. While great progress has been made by prior efforts in applying AI/ML to network automation, the practicality of deploying ML models such as training strategies (e.g. the volume, selection and recency of training data; and having separate models for different hours of a data stream) have not received enough attention. To shed light on this issue, in this chapter we explore the forecasting per-second flow arrival rate for all incoming flows and incoming web flows using LSTM. Our results provide valuable insights into the ability of the forecasting models for short-term forecasting of the two data streams and elucidate the effects of training strategies, input features, among others, on the accuracy of models.

The main limitation of our study in this chapter is its focus on two specific network data streams from a particular campus network. While our methodology is certainly applicable to forecasting any data stream from any network, our trained models and findings cannot be generalized. More importantly, we argue that such a generalization of ML models—as it is done in other domains (e.g., image classification)—may not necessarily be feasible in networking. This observation is motivated by the fact that network data streams are likely to exhibit diverse (short term) characteristics across different networks. This, in turn, suggests that the training and deployment of ML models should be customized for a specific data stream from a particular network to ensure high accuracy. In short, any modeling study in networking is likely to be specific to its target setting.

CHAPTER V

NETWORK ANOMALY DETECTION USING APPLICATION BEHAVIOR MODELING

As a part of network monitoring efforts, it is important to understand and learn about the behavior of the system, especially when we are required to detect anomalous behaviors. There are a large number of potential sources for the anomaly in network operations ranging from distributed denial of service (DDoS) attacks to misconfiguration of the hardware and software components of the network. Given the large scale of unlabeled data in this domain and the need to perform such monitoring tasks in an automated and scalable manner, in this chapter we consider the application of machine learning methods in learning the normal behavior of the network applications and detecting anomalies. Understanding the inner working logic of such detection techniques and explaining the underlying reasons for decisions, can help the network admins to trust the models in the first place and then make informed decisions, however, due to the black-box nature of ML techniques additional effort is needed. To that end, the effect of feature selection and anomaly types are assessed and the benefit, accuracy, and challenges of using model interpretation and extraction methods for understanding the detected anomalies are discussed.

The content in this chapter is a result of collaboration with co-authors and is not published yet. Soheil Jamshidi is the primary author of this work and responsible for conducting all the presented analyses.

5.1 Introduction

Distributed Denial of Service (DDoS) attacks have been widely a concern due to disproportionate damage they cause compared to the required resources to initiate the attack. As researchers attempt to detect earlier versions of the DDoS attacks such as ICMP, UDP, and SYN flooding, attackers adapted and utilized more sophisticated attacks in the application layer (Y. Xie & Yu, 2008). In the application layer, the web service is considered the most vulnerable application (Liao et al., 2015). Attacking web servers through abnormal type, rate, or sequence of requests is an example of such attacks (Jaafar et al., 2019). More specifically, Slowloris (*slowloris DDoS tool*, n.d.) overwhelms a web server by exhaustively starting new sessions and keeping them alive by sending sparse requests and therefore while not sending too many requests, prevent server from proper handling the incoming requests.

There has been a wide range of techniques considered to tackle the application-layer DDoS attacks. Most of studies rely on aggregated network features such as the number of sources, ports, and packets while some use user behavior related attributes such as the interval between two-page visit (Sreeram & Vuppala, 2019), HTTP GET count per connection or IP address (Johnson Singh et al., 2016), GeoIP, source MAC address, and the number of user agents (Shiaeles & Papadaki, 2015). These attributes are used by different techniques to decide whether an incoming request is an anomaly or not. Techniques such as autoencoder models (Bhatia et al., 2019), reinforcement learning (Feng et al., 2020), PCA and ant-colony optimization methods (Fernandes Jr et al., 2016), genetic Algorithm and fuzzy logic (Hamamoto et al., 2018), ARIMA and Holt-Winter models (Jiang & Papavassiliou, 2006; Pena et al., 2013) are used. However, some of the application layer attacks, such as Slowloris, do not have an extensive footprint on the network side, which makes it challenging to detect them solely based on the network attributes while on the application side, they can have a unique footprint that allows us to distinguish them from previously seen patterns. In case of an asymmetric attack, for example where the client requests computation-intensive target pages, end-host resource usage information (e.g. CPU and memory usage) can be a better indicator of attack compared to the network signals. To the best of our knowledge, none of the prior studies have systematically compared the effect of sources of features used for modeling on detection accuracy. We hypothesize that the features from the network traffic (e.g. number of flows or size of packets), the end host resource usage (e.g. CPU and memory usage), and the network application (e.g. response status in case of a webserver) can contribute to the accuracy of detection of different types of attacks.

In this study, we consider 4 types of attacks: three slow HTTP attacks (Shekyan, 2020) and a session flooding attack (*BoNeSi DDoS tool*, n.d.) on Apache web server given its popularity compared to other web servers (datanyze.com, 2020). Due to the differences in the mechanism of these attacks, each has a different footprint on the traffic attributes. We utilize three types of features including network, operation system, and application, to detect attacks in an unsupervised manner and develop a practical and accurate system for anomaly detection. We further analyze the contribution of each of these feature sources through different model training strategies and model interpretation techniques.

Our analyses show that additional attributes from the application and operating system improved the accuracy of our model. The unsupervised neural network model is capable of differentiating among anomalies and normal behavior of the application. Furthermore, we illustrate how interpretation methods can facilitate the process of examining anomalies for network admins.

The rest of this chapter is organized as follows: we explain the system architecture that delivers the required attributes for the model in section 5.1, we discuss the detection system components in section 5.2, the characteristics of the data set is illustrated in section 5.5 and our model evaluation results are reported in section 5.6. Finally, we discuss the impact and benefits of model interpretation and extraction method in section 5.7.

5.2 System Overview

Detection of attacks requires the collaboration of many different entities within a network. In this section, we discuss the architecture of our proposed system. As shown in Fig 32, in our system the clients send requests to a network application which in our case is a web server. As they send the requests, three types of features are provided to a collector entity that determines whether the request is legitimate or not.

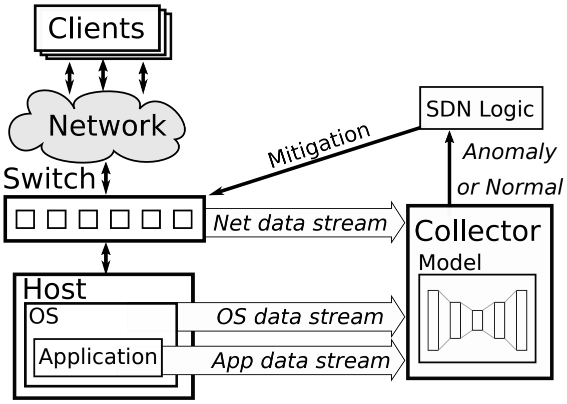


Figure 32. System architecture

The collector relies on three types of input features:

- The application provides logs about its own view. In our case, the application is a web server that provides information about the resources that have been requested by the clients and the response it provides back to the clients through log files that are known as “access log". We rely on Apache webserver given its dominance on the market compared to the other webservers with more than 8M web servers around the world and having more than 50% of the market share (datanyze.com, 2020). It is worth noting that other web servers also have similar types of logs available and our methodology is not limited to a specific web server.

- The operation system (OS) of the end host provides information about the resources that the application uses as it serves the client requests. In our case, the OS provides information about the CPU and memory usage of the web server.
- In addition, the switch connecting the end host to the rest of the network provides telemetry data about the incoming and outgoing network traffic. We use our Cedar switch-based telemetry system to gather per-flow reports of the listed traffic features and augment these features with aggregate features (e.g. number of sources) computed in software.

At the core, our proposed system has a collector component that learns the behavior of the system based on the provided signals mentioned above and detects deviations in the behavior. The collector receives these attributes from the above sources and performs pre-processing steps. Specifically, it needs to handle missing values (e.g. in the absence of any request, or if the host or web server is down), clean the data to have a unified set of values for different types of requests (e.g. to extract target pages from the request payload) and ensure that timestamps are synchronized across different entities. Note that each of these systems might have different timestamp sources for their provided information. After the pre-processing step, the collector converts the three input data vectors into a feature vector through the feature engineering step. We will discuss this step in detail in the next section. Then, the collector provides the feature vector to the machine learning model.

The granularity of the provided data determines how fast we can react to the anomalies. We expect to monitor the system in a per-second granularity. The coarser the granularity, the more aggregated the data will become (smoothing out some of the variations that lead to data loss) and the more time it takes to detect and respond to anomalies (for example in a per hour scale, after one hour we will realize if there was an anomaly in the past hour or not).

The model indicates if the provided information is related to a normal or anomaly case. The model outcome can be a binary label of normal/anomaly or it can further break it down to the type of anomaly. Since the latter requires labeled data for each type of an anomaly which is not feasible in many cases, we consider the former here, labeling incoming vectors as normal or anomaly and the model's decision can activate different mitigation plans that can range from dropping further requests form the client and blacklisting the source IP to activation of more sophisticated plans. Such plans are enforced in the network using the Software-defined networking (SDN)-based rules that will be defined by the network administrators based on the enterprise policies.

5.3 Threat Model

We assume that the anomaly in server's behavior can occur as a result of a spike in normal behavior (e.g. an event, a promotional campaign, seasonal effect) or as a result of an attack. In case of an attack, we assume that the goal of the attacker is to make the web server not respond to legitimate requests either with a huge footprint on network traffic (flooding attack) or without it (slow attacks). We assume that attacker does not have access to historic traffic attributes so can not initiate attacks based on the normal state of the server, and also we assume that attacks will impact at least one of the application, network, or the OS level attributes, therefore, the differences can be captured by a well trained ML model. On the defense side, we assume the availability of the attributes from the application, network, and operating system that based on anomaly detection output, a mitigation plan can be activated to bring the server back to a responsive state.

5.4 Detection System Components

In this section, we describe the components of our detection system. We generate HTTP attacks as our anomaly cases and use our proposed anomaly detection system to capture those cases. We explain our model selection, feature selection, and training strategy in the following subsections.

5.4.1 Model Selection. The model is at the core of the collector component. The type of the model (e.g. classifier or regressor, supervised or unsupervised) determines the type of data that is required to be provided to the collector. To address the related challenges that are mentioned earlier in chapter II section 2.3, in this study we rely on an unsupervised neural network model, an LSTM autoencoder. Autoencoders learn how to efficiently compress and encode data and then learn how to reconstruct the data back from the reduced encoded representation that is as close to the original input as possible. This approach, enables us to detect anomalies without relying on the availability of labeled data. Autoencoders have been shown to be accurate in similar settings (Luo & Nagarajan, 2018; Mac et al., 2018; Vartouni et al., 2019; Yadav & Subramanian,

2016) and can be trained and tuned with a lower overhead compared to forecasting techniques. A representation of such models is depicted in Fig 33. Autoencoders are made from four main parts:

- Encoder: the model learns to reduce the input (X) dimension and convert the input to compressed representation (z).
- Code (z): that has the compressed representation of the model
- Decoder: the model learns to reconstruct an approximation of the original input ($\hat{X}$) using the compressed representation (z)
- Reconstruction error: measuring how closely the model reconstructed the original inputs ($X - \hat{X}$).

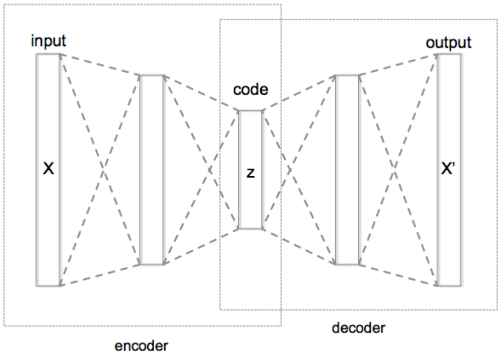


Figure 33. Autoencoder structure

A large reconstruction error shows deviation from normal behavior as the model is trained to have a low loss on normal behavior. Therefore, Reconstruction error can be used to determine whether the input is an anomaly or not. The challenge here is how to set the threshold for that purpose. One approach is to set the threshold automatically based on the model's performance on the validation set. Intuitively, if the model is able to reconstruct the normal data with reconstruction error in the range of (a, b), then any error beyond b can be considered as an anomaly. We discuss a sensitivity analysis for the threshold in the next section. We used Keras library implementation of LSTM with the TensorFlow backend for our model. After extensive hyperparameter tuning, we selected a model in a form of two hidden layers with 64 and 32 hidden nodes, L2 regularization on the first layer, ReLU activation function, and Adam optimizer, we used MAE as our loss function trained over 300 epochs.

5.4.2 Traffic Generation.

In order to detect the anomalies, we need to have a dataset representing the normal state of the system and a dataset representing the attacks and anomalies. As discussed in chapter II section 2.3, to the best of our knowledge there is no publicly available dataset that provides all sources of data (from OS, application, and network) together or covers different types of anomalies. Therefore, we collected our own dataset. To collect the normal data from the mentioned three sources (OS, Application, and network) that will be used for training our model, we replay the HTTP requests (sending them to our server) from one month's worth of all HTTP operations to the NASA Kennedy Space Center web server in Florida (Arlitt & Williamson, 1996) with the exact pace (relative timestamps) that the requests were logged. Then, we need to create our validation set for parameter tuning and selecting the threshold required for labeling anomalies. Given that each time step will be analyzed independently by our model, we set aside randomly selected 10 percent of our normal data (almost 2.4 hours) and use it as the validation set.

For the test dataset, we use publicly available tools to generate two types of attacks: the flooding, and slow HTTP attacks. We utilize BoNeSi (*BoNeSi DDoS* *tool*, n.d.) tool to generate TCP (HTTP) flooding attacks. For the slow HTTP attacks, we consider the Slow HTTP Test tool (Shekyan, 2020) and generate three types of slow HTTP attacks including slow body, Slowloris, and slow read. We used the default parameters provided by the tools to initiate the attacks. The primary parameter for the BoNeSi tool is the address of the target server and the type of flooding attack that we selected to be TCP (HTTP). For slow HTTP test tool, there are several parameters that should be selected to reflect a specific attack. We have listed the values for these parameters in Table 6. The parameters are as follows:

- c: the number of connections
- i: interval between follow up data in seconds, per connection
- n: interval between read operations from receive buffer
- p: timeout to wait for HTTP response on probe connection, after which server is considered inaccessible
- r: connection rate
- s: value of Content-Length header
- t: method
- w: start of range the advertised window size would be picked from
- x: max length of follow up data
- y: end of range the advertised window size would be picked from
- z: bytes to read from receive buffer with a single read operation.

Table 6. Parameters of our attack cases

	-c	-i	-n	-p	-r	-s	-t	-w	-x	-y	-z
SlowBody	300	110	-	3	200	8192	-	-	10	-	-
SlowLoris	300	10	-	3	200	-	GET	-	-	-	-
SlowRead	300	-	5	5	200	-	-	10	-	20	32

For our normal data, we replay and collect data from the NASA dataset logs for 24 hours. We selected a Monday as the server served the most requests on that day compared to other weekdays. For the test cases, we generate each attack for 3 minute per each variation while replaying normal background traffic, similar settings have been used in prior works (Hirakawa, Ogura, Bista, & Takata, 2016). With a 3 minute pause between each, we ensure that there is no overlap between attack variations. Given the per-second granularity of our data collection, 180 seconds will cover the range of changes in our feature set given the type of attacks. For example, it takes some time for the slow attacks to occupy all of the server's resources.

We have 4 attack types each with $3*60 = 180$ seconds length. In addition, to have normal test cases as well, we replayed normal HTTP requests for an additional day and then, we randomly selected 180 seconds to from our normal “test" data points. Now, for testing purposes, we assess how many of these 360 seconds (180 attacks + 180 normal) were labeled correctly as positive (attack) or negative (normal) by our model for each of the 4 attack types.

5.4.3 Features. We have three sources that provide information for the collector: the application (Apache webserver), network traffic, and system resource usage, as a set of features per second. From the webserver, we can obtain information about the HTTP request sent to the server, sender, server response status code (e.g. 200 for a successful response, 400 for not found target page) (*App_status_* *), and size of the response (*App_size_* *). Also the server provides the number of access request it received (*AppStat_AccessCount*), the number of requests it processed (*AppStat_processingRequests*), and available worker pool to serve new requests (*AppStat_IdleWorkers*)[1].

On the network side, we capture the number of packets, size (bytes), source and destination IP and ports, as well as protocol and TCP_flags (including *ACK*, *PSH*, *RST*, *SYN*, and *FIN*) per second for incoming and outgoing packets. On the system side, we capture the percentage of CPU and memory usage of the webserver process, as well as number of minor (major) memory faults made by the webserver process (*minflt* and *majflt*) using pidstat command[2] on Linux. All the above attributes are collected per second. We expand the above attributes by calculating their min, mean, 25, 50, 75 percentiles, and max values for numerical attributes (such as number or size of packets, CPU and memory usage, and faults), and the number of values per category for categorical attributes (such as server response status code), and the number of all and number of unique values for non-numerical and non-categorical values (such as source and destination). Altogether, we have a vector of 58 features [3] per second that is fed into the model for assessment. The full list of features is summarized in Table 7.

[1] For more information on the application features refer to the following link *https://www.elastic.co/guide/en/beats/metricbeat/current/exported-fields-apache.html*

[2] Using the following command: pidstat 1 -rud -C apache -h

[3] 16 OS, 15 application, and 27 network-related features

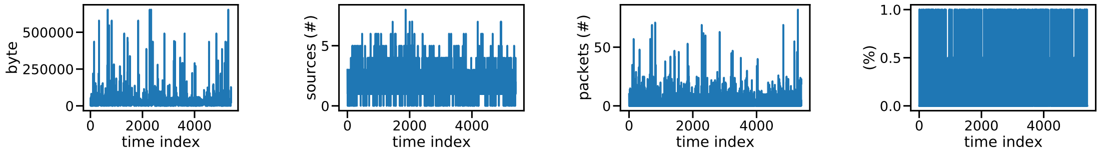
Table 7. List of features used for our modeling

OS	OS_process_count OS_%CPU_min OS_%CPU_mean OS_%CPU_25 OS_%CPU_50 OS_%CPU_75 OS_%CPU_max OS_%CPU_sum	OS_%MEM_sum OS_minflt/s_min OS_minflt/s_mean OS_minflt/s_25 OS_minflt/s_50 OS_minflt/s_75 OS_minflt/s_max OS_minflt/s_sum
Application	AppStat_AccessCount AppStat_processingRequests AppStat_idleWorkers App_size_25 App_size_50 App_size_75 App_size_max App_size_mean	App_size_min App_size_sum App_src_count App_src_nunique App_status_200_count App_status_400_count App_status_404_count
Network	NetIn_packets NetIn_bytes NetIn_sources NetIn_min_length NetIn_ave_length NetIn_max_length NetIn_min_TCPWindowSize NetIn_ave_TCPWindowSize NetIn_max_TCPWindowSize NetIn_ACK_count NetIn_PSH_count NetIn_SYN_count NetIn_FIN_count	NetOut_packets NetOut_bytes NetOut_sources NetOut_min_length NetOut_ave_length NetOut_max_length NetOut_min_TCPWindowSize NetOut_ave_TCPWindowSize NetOut_max_TCPWindowSize NetOut_ACK_count NetOut_PSH_count NetOut_SYN_count NetOut_FIN_count

5.5 Data Characteristics

As discussed earlier, we have a normal dataset used for training and generated a variety of attacks for the test dataset. In this section, we discuss their characteristics.

For the normal day dataset, we re-played 89,561 requests that were associated with 7,336 sources on 24 hours. Basic characteristics of the normal day data are shown in Fig. 34. As shown in Fig 34a the response size is up to 700KB per seconds, from up to 9 sources per seconds ((Fig 35d), 70 incoming packets per second (Fig 34c) and CPU usage up to 1% of the server's computation capacity (Fig 34d).



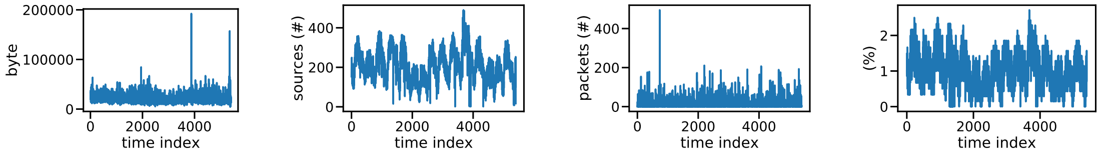
(a) Response size (mean per sec)

(b) #Unique sources per second

(c) #Incoming packets

(d) CPU usage

Figure 34. Characteristics of a normal day - Normal Flow



(a) Response size (mean per sec)

(b) #Unique sources per second

(c) #Incoming packets

(d) CPU usage

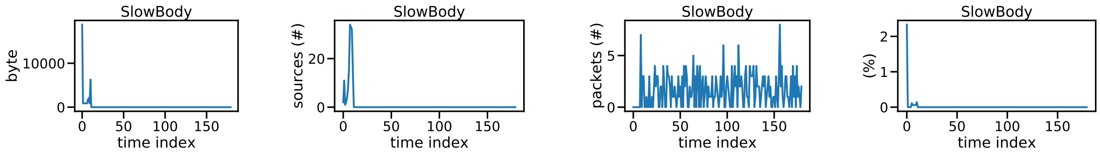
Figure 35. Characteristics of a normal day - Sending with no delay between requests

Sending the requests as logged on the dataset is one of the options. Additionally, we can send requests without any delay between them (noted as top speed). In this case, we will put the server under more pressure. Doing so results in having the features as shown in Fig 35. As shown, in this approach, the mean of the response sizes per second lowers to a maximum of 200KB per seconds (Fig 35a), with more than 400 unique sources per second (Fig 35b). The incoming packet rate is up to more than 400 pps (Fig 35c) and CPU usage varies between 0 and 2.5 percent (Fig 35d).

Our testing dataset contains 4 different attack cases[4] each for 3 minutes (180 seconds). As shown in Fig 36 to Fig 39, while the slow tests bring down the response size and sources to a small and steady number over the course of the attack (by their definition they are expected to do so), the DDoS TCP flooding

[4] 3 slow attack and one HTTP flooding

manages to keep the server busy without dramatically changing the features compare to normal ones seen in Fig 34.



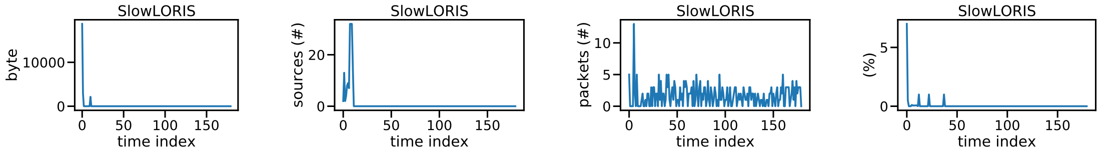
(a) Response size (mean per sec)

(b) #Unique sources per second

(c) #Incoming packets

(d) CPU usage

Figure 36. Characteristics of an Attack day - SlowBody



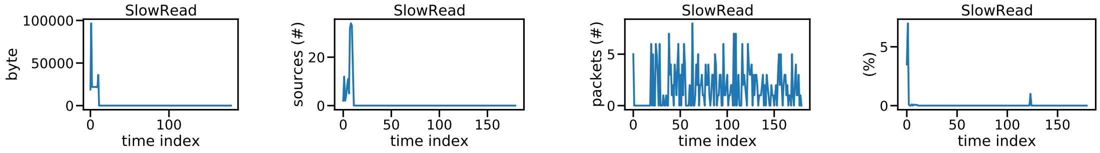
(a) Response size (mean per sec)

(b) #Unique sources per second

(c) #Incoming packets

(d) CPU usage

Figure 37. Characteristics of an Attack day - SlowLoris



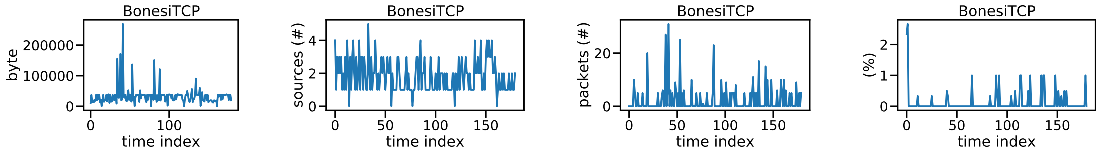
(a) Response size (mean per sec)

(b) #Unique sources per second

(c) #Incoming packets

(d) CPU usage

Figure 38. Characteristics of an Attack day - SlowRead



(a) Response size (mean per sec)

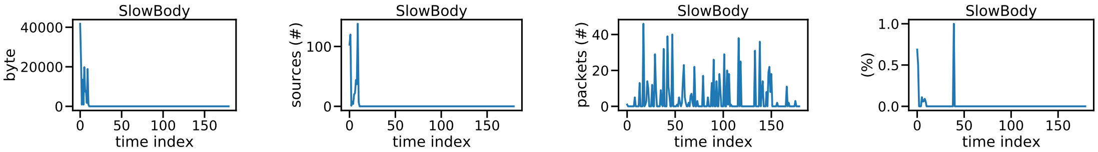
(b) #Unique sources per second

(c) #Incoming packets

(d) CPU usage

Figure 39. Characteristics of a Attack day - Bonesi TCP

Similar to normal data, here for the attacks, we have the option to replay the background traffic with no delay between the requests. As shown in Fig 40 to Fig 43, in this case, the number of sources and incoming packets changes compared to the previous option.



(a) Response size (mean per sec)

(b) #Unique sources per second

(c) #Incoming packets

(d) CPU usage

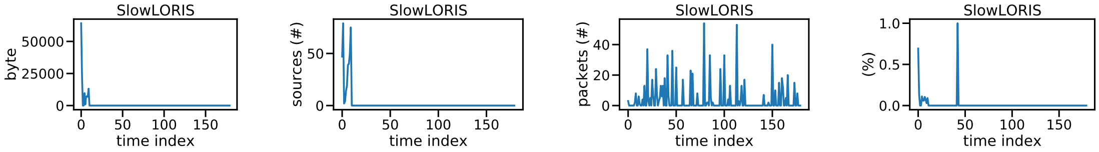
Figure 40. Characteristics of an Attack day - SlowBody - Sending background traffic with no delay

5.6 Evaluation

In this section, we perform sensitivity analyses on our parameters and discuss the accuracy of our model in different cases.

5.6.1 Effect of threshold on accuracy.

As discussed in Sec 5.4 to label data points as an anomaly, we should set a threshold on the reconstruction error. If the error is larger than the threshold, the label is an anomaly. We examine the effect of threshold choice on the model's accuracy in terms of true positive rate (Fig 44a) and false positive rate (Fig 44b). In these figures, we show the threshold in a range of 1 to 100 on the x-axis. The threshold indicates the percentile of reconstruction error on the validation data. Therefore, if we set the threshold to be 100, it means that we set the threshold to be the maximum of the validation error. We show a box plot per threshold value that is the summary distribution of values we see for each of the attack types (Slow and Flooding attacks). The average is shown with a green triangle for each box plot. The receiver operating characteristic curve, or ROC curve, based on true and false positive rates is shown in Fig 45



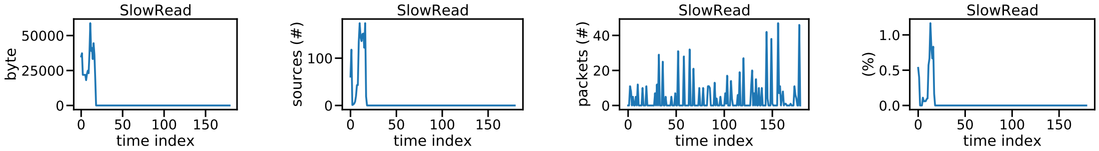
(a) Response size (mean per sec)

(b) #Unique sources per second

(c) #Incoming packets

(d) CPU usage

Figure 41. Characteristics of an Attack day - SlowLoris - Sending background traffic with no delay



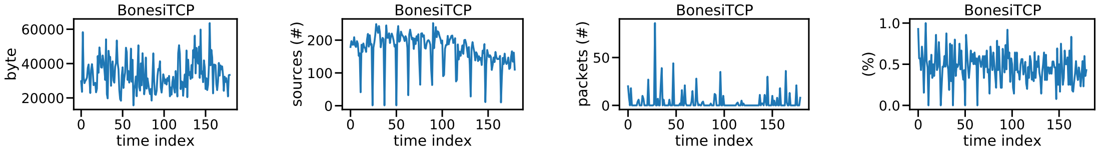
(a) Response size (mean per sec)

(b) #Unique sources per second

(c) #Incoming packets

(d) CPU usage

Figure 42. Characteristics of an Attack day - SlowRead - Sending background traffic with no delay



(a) Response size (mean per sec)

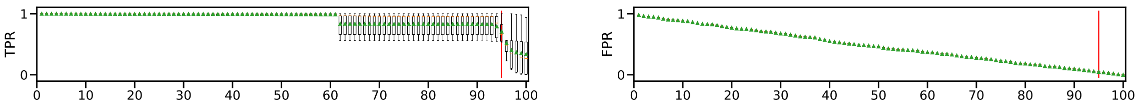
(b) #Unique sources per second

(c) #Incoming packets

(d) CPU usage

Figure 43. Characteristics of a Attack day - Bonesi TCP - Sending background traffic with no delay

where we have shown the 95 percentile on the plots using a vertical red line. The diagonal dashed red line indicates a random (no skill) detector. As shown, the lower the threshold, the larger number of data points will satisfy the condition $loss > threshold$ which leads to high false positive (labeling all data points as an attack). The larger the threshold, the fewer data points will be labeled as an attack and results in a higher false-negative rate (lower true positive), where in an extreme case it brings the false positive to zero while only detecting 50% of attack cases on average across all attack types.



(a) on TPR

(b) on FPR

Figure 44. The effect of reconstruction error threshold

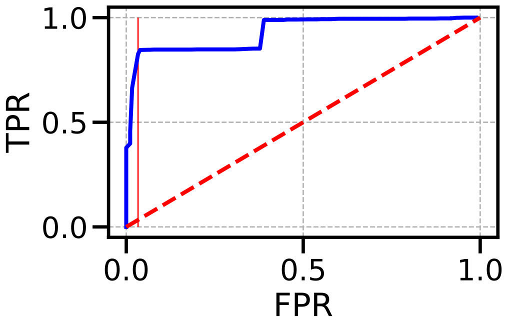


Figure 45. True positive and false positive rate ROC plot

We set the threshold to be the 95 percentile of reconstruction error on the validation set that is shown as a red line on both figures. It was shown to be a reasonable choice based on our experiments and inline with parameter selections in prior works (Sakurada & Yairi, 2014). Therefore, if our model is not able to accurately reconstruct the data points with an error that is more than 95% of the validation error, then we label it as an anomaly case.

5.6.2 Anomaly detection rate.

We measure how our model is able to catch different types of attacks out of 180 cases of anomaly and 180 cases of normal data points, how many were labeled correctly. In order to capture that, we report F1-score per anomaly type, where:

$$F_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

F1-score presents the harmonic mean of the precision and recall and reaches its best value at 1. We trained the model only using the network features, only application, using both, and also using all three types of features (net, app, and OS). Table 9 shows the result for when our normal dataset is collected with the normal rate (as indicated in the original logs). While Table 8 shows the results when the training data is collected by sending requests without any delay. The results suggest that our models had higher accuracy in the detection of anomalies for slow HTTP attacks when trained on the data that is captured with high speed rate replay. As the differences between these cases are more pronounced. However, when focusing only on net and app features, our model that is trained with the normal flow rate, has higher accuracy. Overall, considering the app and OS features have improved our detection capabilities.

Given the nature of the data in both cases and the effect of attacks on the features that we discussed in the previous section, both cases can be explained. First, slow HTTP attacks slow down the send and received rate, in the case of a server with low load, the changes are not pronounced and more challenging to be detected for the model that is trained on low load data. Training on a high load server data, however, leads to a stronger signal. For the flooding attacks, the scenario is on the contrary. The more load on the server in the normal state, the less detectable will be on the attack time as features will change relatively less compared to a low load server.

We have also considered one-class SVM (Amer, Goldstein, & Abdennadher, 2013; Erfani, Rajasegarar, Karunasekera, & Leckie, 2016; Perdisci, Gu, & Lee, 2006) and IsolationForest (Ding & Fei, 2013; S. Li, Zhang, Duan, & Kang, 2019; F. T. Liu, Ting, & Zhou, 2008; Puggini & McLoone, 2018) as alternative models for anomaly detection. We considered their Scikit-learn library implementation and we used its parameter-grid model selection for fine tuning these models. Table 10 and Table 11 show the results for these models, respectively. Comparing with Table 9, our auto-encoder model outperforms both models in its best performance using all three feature sets. However, the best case for these methods is not when the model used all three sources. For example for the one-class SVM model, the best result for flooding attack (BonesiTCP) is obtained when using only the network features or for isolation forest, using only the application features led to the highest accuracy across all attack types.

Table 8. Evaluation results - train on data with no inter-request delay

Name	F1_Net	F1_App	F1_NetApp	F1_AllThree
SlowBody	0.72	0.99	0.97	0.99
SlowLORIS	0.71	0.45	0.76	0.99
SlowRead	0.67	0.40	0.74	0.99
BonesiTCP	0.34	0.40	0.71	0.99

Table 9. Evaluation results - trained on actual rate data

Name	F1_Net	F1_App	F1_NetApp	F1_AllThree
SlowBody	0.75	0.99	0.97	0.97
SlowLORIS	0.73	0.45	0.78	0.94
SlowRead	0.72	0.40	0.77	0.94
BonesiTCP	0.34	0.40	0.76	0.94

5.6.3 Additional Challenging Cases.

Having a high detection accuracy is directly related to the elaborate footprint of the DDoS attacks on the application and OS features. Therefore, to make the test cases more challenging, we can consider the following scenario: what if the attacks lead to a signature close to the normal behavior of the application?

Table 10. One-class SVM evaluation results - trained on actual rate data

Name	F1_Net	F1_App	F1_NetApp	F1_AllThree
SlowBody	0.88	0.86	0.94	0.94
SlowLORIS	0.85	0.86	0.94	0.94
SlowRead	0.90	0.86	0.94	0.94
BonesiTCP	0.92	0.86	0.36	0.36

Table 11. IsolationForest evaluation results - trained on actual rate data

Name	F1_Net	F1_App	F1_NetApp	F1_AllThree
SlowBody	0.59	0.92	0.87	0.66
SlowLORIS	0.61	0.92	0.87	0.66
SlowRead	0.59	0.92	0.87	0.64
BonesiTCP	0.26	0.35	0.32	0.30

To perform this experiment, we considered the following cases: (i) Changing all features with a random ($0 \leq rnd \leq 1$) fraction of standard deviation (SD) of each feature. In this case the new feature values will be $Xi\_new = Xi + rnd * std(Xi)$. (ii) Changing all features, each with a separate random fraction of SD: $Xi\_new = Xi + (rnd\_i) * std(Xi)$, and (iii) Changing fraction of features with a probability of p ($0 \leq p \leq 1$) and therefore $Xi\_new = Xi + [rnd > p][(rnd\_i) * std(Xi)]$. We set p to be 0.1 and given the random nature of our definitions, we have repeated the experiment for 20 times while testing the model with only net, only app, app and net, and all features, similar to our report in the previous section. The results are shown in Table 12 suggest that application features are more sensitive to changes as the accuracy stays high when only focusing on these attributes. Also randomly changing the features (the most challenging case) had the most effect on our model's accuracy.

5.6.4 Spike in legitimate requests. In the earlier section, we showed that even if we change a fraction of attributes, our model is able to

	OnlyNet	OnlyApp	NetAndApp	AllThree
(i) AllFeaturesSameRatio	0.96±0.01	1.0±0.0	0.97±0.01	0.97±0.0
(ii) ratioPerFeature	0.98±0.0	1.0±0.0	0.98±0.0	0.98±0.0
(iii) randomFeat	0.84±0.02	0.87±0.01	0.85±0.0	0.84±0.01

Table 12. Our model's accuracy on more challenging cases

accurately detect the anomaly. The case of a spike in legitimate requests (e.g., sudden increased interest in the content of a web server) is an expanded version of that idea. To evaluate it, we use randomly selected 360 seconds of data from the normal day with no inter-request delay (top speed rate), we test if the model would detect these inputs as an anomaly or not. With the network-only features, only 33% were labeled as anomaly while with other combinations, all were labeled as anomaly (with threshold = 95%). So our model is marking these spikes as anomaly. However, the more important question is how can we differentiate the attacks from such normal events?

In such cases to determine the level of importance, we have a few options:

(i) Classification. We can train a classifier to determine the type of anomaly if the labeled data is available. However, as discussed earlier not only labeled data is not available in most cases, the type of attacks and normal spikes are not known ahead of time.

(ii) Feature importance. If labels are not available, then we can use feature importance to provide additional information for the network admin to make an informed decision when facing these anomaly cases.

In this case, we used the 360 cases that we used to test the model against the legitimate spike event, to calculate the feature importance. Fig 46 shows the top 10 important features for these cases. Interestingly, in these cases, the set of top important features has a small overlap with the top important features for the attacks that we earlier discussed. And uniquely, both the outgoing and incoming features show up among top important features, suggesting a symmetric effect on the attributes in this case. Also, our analysis shows that the number of idle workers of the web server was lower than normal cases yet 2 times more than of the attack cases that can be differentiating the attacks from normal spike cases. The important features, give the network admin areas to focus on, therefore, they can make informed decisions.

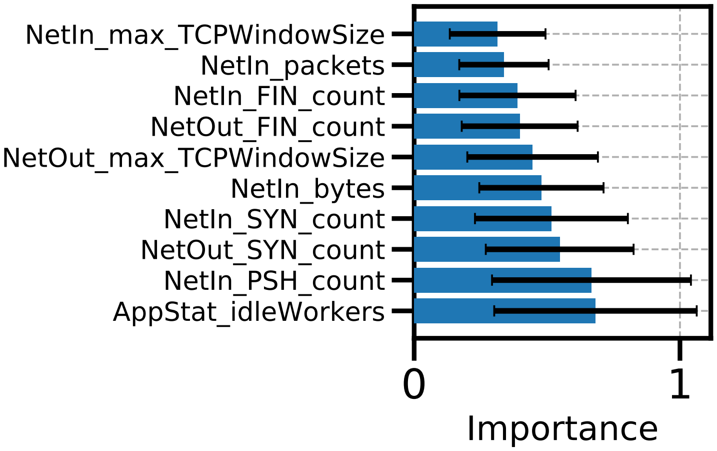


Figure 46. Feature importance for normal request spike cases

(iii) Ranking. An advantage of our chosen model is that we can use the reconstruction error to cast the problem as a ranking problem. Therefore, we can focus only on most important anomalies (presumably those with higher reconstruction error). To assess whether this approach can be considered or not, in Fig 47 we show the summary distribution of reconstruction error for the 3 more challenging cases that we defined in section 5.6.3, the normal spike in normal legitimate requests, as well as the 4 types of attacks that we evaluated in section

5.6.2. The green triangles represent the average value. Our analysis shows that attacks lead to higher reconstruction errors on average compared to the legitimate spikes and we can have a good distinction among different event types given the characteristics of their reconstruction error that might further benefit from dimension reduction techniques (such as PCA) or clustering methods.

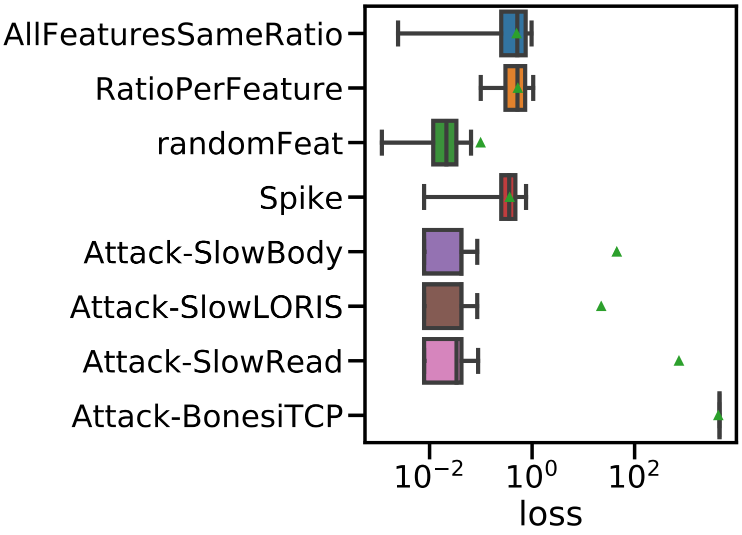


Figure 47. Reconstruction error for all cases

5.7 Model Explainability and Interpretation

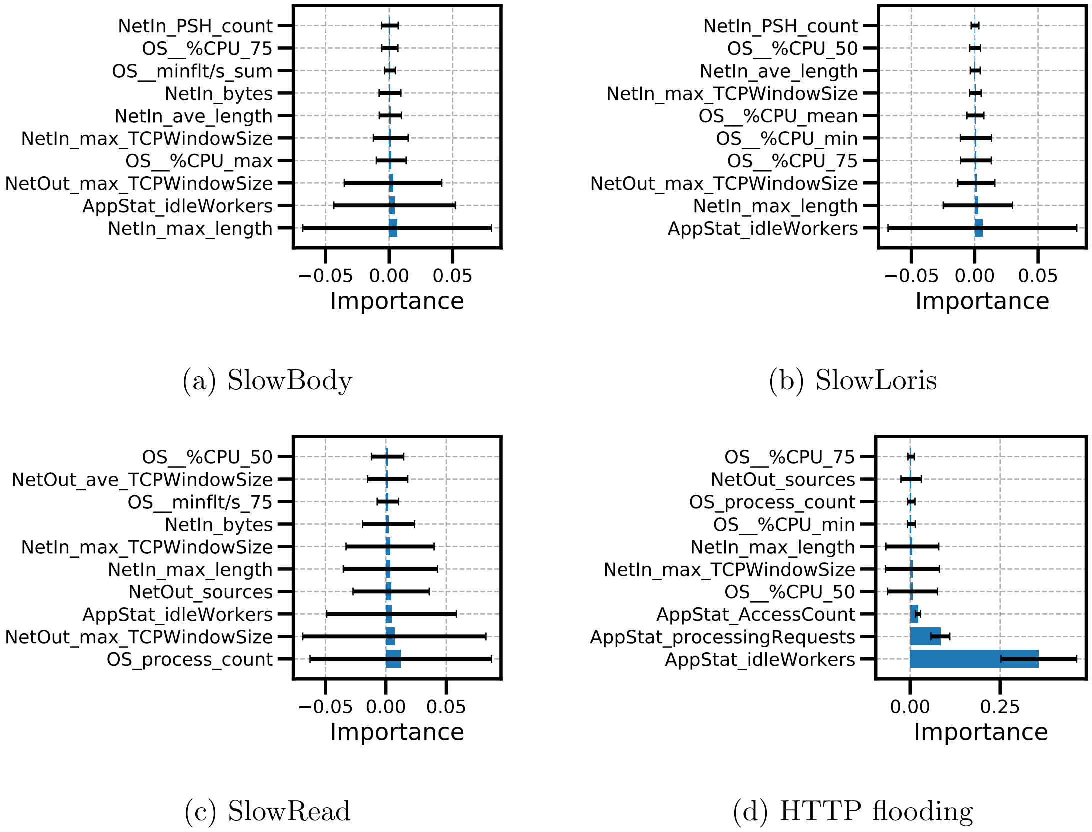
In this section, we focus on effectiveness of standard explanation techniques on our trained models. As if they are successful in interpreting the models, we can gain insights that help on practicality, understanding, and improvement of our trained models.

5.7.1 Model Interpretation Results. We utilize model interpretation techniques to focus on trust and informativeness aspects. The DeepExplain framework (Ancona et al., 2017) offers the implementation of a range of interpretability methods in a unified way including backpropagation and perturbation methods. We use the occlusion method (Zeiler & Fergus, 2014) as a perturbation-based attribution method to calculate the feature importance.

We leverage methods that are implemented in DeepExplain on the model that is trained in the earlier section. We used the test cases (each one is a vector of 58 features) per attack type (e.g. Slowloris or flooding) as input and the outcome is a score vector, one value per feature per test case. Then, we can aggregate the scores across all cases by reporting the mean and standard deviation of importance score. This aggregated value can be used to rank the features per attack type. These scores will indicate which features are the most important ones for our model when detecting specific attacks. Figure 48 shows the examples of feature importance outcome on different attack types. As shown, The top 3 features for slow attacks are almost the same while the TCP flooding attack has a different set. In addition, the flooding attack seems to rely more on app and OS features compared to the slow attacks.

Using the feature importance, we can confirm whether the model is relying on the right set of features or not. If not, then we can debug our training step with further investigation of our feature engineering, data, and model architecture and hyperparameters. However, in this case we note that the features relied on by the model correspond with our intuitions about these particular anomalies.

In addition, we can check whether training the model using a subset of the top important features can have relatively similar accuracy as the original model. The results are shown in Table 13 where we train our models using only the top $(1, 3, 5, 10, ..., 25)$ important features. Our analyses show that accuracy of such models depends on the attack type. For some of them, with even one of our features our models are able to obtain relatively accurate results. However, the



(a) SlowBody

(b) SlowLoris

(c) SlowRead

(d) HTTP flooding

Figure 48. Feature importance per attack type

effect of adding new features is not linear in all cases. Our analyses show that the aggregated feature importance is not an accurate way to specify how the model relies on different features and which subset of features can be used, although it helps to identify reasons for labeling individual cases.

5.7.2 Model Extraction Results.

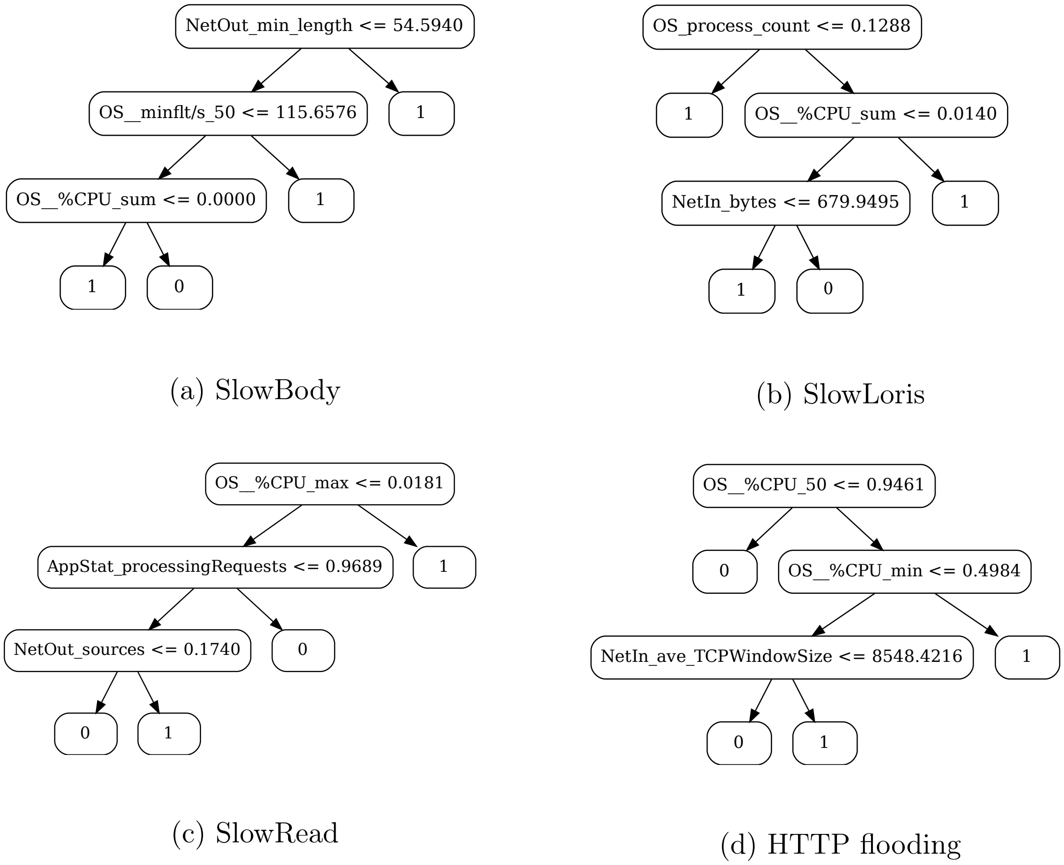
The interpretation methods do not capture the relation between the features and how they are contributing to the final model's outcome. To capture the relationships among them and gain insight on the inner working logic of our model we leverage the DTextract method (Bastani, Kim, & Bastani, 2017a).

Name	Top 1	Top 3	Top 5	Top 10	Top 15	Top 20	Top 25
SlowBody	0.34	0.35	0.41	0.41	0.40	0.41	0.49
SlowLORIS	0.35	0.76	0.36	0.74	0.73	0.75	0.73
SlowRead	0.33	0.35	0.38	0.41	0.34	0.37	0.37
BonesiTCP	0.97	0.96	0.94	0.42	0.33	0.42	0.42

Table 13. F1-score for models trained with top N important features

The output of DTextract method is a set of rules that approximate the bahavior of the original model. These rules can be visualized as a decision tree. The number of nodes in this tree is considered as a tuning parameter for the DTextract method and can affect how well the extracted ruleset can capture different cases, the larger the tree is the more specific (less general) the rules will become. As mentioned, the feature importance scores in Fig 48 depict which features are more important for the model. However, the interactions among these features are not captured. In order to capture the features' interactions, we visualize the extracted rules as decision trees in Fig 49. The decision trees are extracted based on test cases related to each of the attack types and anomaly is indicated by a positive label (label=1). While the tree with only 7 nodes is too simple to capture the whole logic, it shows how inner working logic of the model can be captured by a combination of network and OS features. For example, in Fig 49b we can consider a case as normal, only if it puts low pressure on CPU but sends large amount of data (relative to the attacks).

While not all selected features are among the top important features we found in earlier section, in case of flooding attack or SlowRead, 2 out of 3 features are among top 10 important features. These trees can help network admins to determine the underlying line of conditions that led to a decision (either anomaly or normal) and can help to determine and trigger the proper mitigation plan.



(a) SlowBody

(b) SlowLoris

(c) SlowRead

(d) HTTP flooding

Figure 49. Extracted Rules

One important factor for rulesets is fidelity. That is how closely the ruleset can approximate the original model. For that, we extract the decision rules with different numbers of nodes per tree for different attack types and report the relative accuracy of the resulted decision tree in labeling the normal and attack cases compared to the original model. We extracted the ruleset with a tree of size up to $2^i - 1$ for i in range 2 to 11. The fidelity results are shown in Table 14. Looking at all attacks together, the rules were not able to approximate the model even with the thousands of nodes. However, when extracting the ruleset per attack type, then the model gains higher accuracy with larger node sizes.

Challenges of model explanation. The extraction methods are expected to be capable of capturing the models' inner working logic, in the presence of a large number of feature that can correlate, the resulting ruleset can be large and therefore each rule can be relevant to a small number of cases. In such cases understanding the ruleset itself can be challenging. Also the extraction methods with off the shelf implementations require modifications to be applicable on different type of problems and in our case shown low fidelity.

In the case of model interpretation, given that the test cases can capture a wide and contradicting characteristics, feature importance methods are better to be used on each group of test cases (in our case we applied on each of our attack types) or even be used to explain the importance of features for individual test cases. In these cases the results can be more clear and reliable.

Table 14. F1-score of extracted rules with a tree of maximum N nodes

Name	3 Nodes	7 Nodes	15 Nodes	31 Nodes	63 Nodes
BonesiTCP	0.20	0.34	0.34	0.17	0.25
SlowBody	0.61	0.23	0.23	0.23	0.23
SlowLORIS	0.33	0.33	0.33	0.33	0.33
SlowRead	0.05	0.05	0.32	0.33	0.33
All together	0.36	0.36	0.36	0.36	0.36

Name	127 Nodes	255 Nodes	511 Nodes	1023 Nodes	2047 Nodes
BonesiTCP	0.25	0.20	0.34	0.25	0.34
SlowBody	0.23	0.61	0.23	0.23	0.61
SlowLORIS	0.33	0.33	0.33	0.33	0.33
SlowRead	0.33	0.33	0.33	0.33	0.51
All together	0.36	0.36	0.36	0.36	0.36

CHAPTER VI

CONCLUSIONS & FUTURE WORK

Large scale networked systems play a critical role in today's society e.g. Internet infrastructure, Facebook user relationships, and Amazon product-review relationships. As a result of information exchange or evolution of connectivity structure, certain types of events and patterns form in these systems that are required to be identified and managed properly. Examples of such events are fake posts, repetitive reviews, or denial of service (DoS) attacks. However, given the large scale of such networked systems and the dynamic nature of their evolution, the detection of such events becomes a challenging task. In many cases, even collecting the footprint or signature of such events is not possible as they behave differently in various settings. Such challenges open the need for machine learning (ML) techniques that offer promising approaches to learn the signature of an event and detect future instances. In this dissertation, we leveraged ML techniques to address the challenges of event detection in multiple networked systems and made the following conclusions:

- In order to detect incentivized online reviews on Amazon, heuristics-based labeling could be useful for training a model to identify reviews without any explicit sign.

- Unsupervised ML models can be used for event detection and they offer opportunities for efficient training and re-training.

- Model interpretation and extraction techniques can provide insights for the operators. However, interpretation of such insights can be challenging due to the complexity of resulting rules or feature dependencies.

- A trained model should be used in its specific setting (considering the input data streams and underlying network) and may not be useful in other similar settings due to the variations in event features. Therefore, the models might need re-tuning and re-training if are intended to be used in a new setting.

Specifically, we have made the following contributions in each chapter. In Chapter III we presented a detailed characterization of Explicitly Incentivized Reviews (EIRs) in two popular categories of Amazon products. We presented a technique to detect EIRs, collected a few datasets from Amazon, and identified a large number of EIRs in Amazon along with their associated product and reviewer information. Using this information, we compared and contrasted various features of EIRs with reasonably normal reviews. We showed that EIRs exhibit different features compared to normal reviews and discussed the implications of these differences.

Then, we zoomed into the temporal pattern of submitted EIR reviews for a few specific products and submitted reviews by a few specific reviewers. These temporal dynamics demonstrated whether/how promotional campaigns by a seller could affect the level of interest by other users and how reviewers could get engaged in providing EIRs. Finally, we illustrated that machine learning techniques can identify EIRs from normal reviews with a high level of accuracy.

Moreover, such techniques can accurately identify other explicitly and implicitly incentivized reviews. We leverage the affiliation of reviews with reviewers and products to infer their incentivized nature. Finally, we reported the current state of incentivized reviews on Amazon and how sellers are adapting to the new dynamic of the system.

In Chapter IV we shed light on the practicality of deploying ML models such as training strategies (e.g. the volume, selection, and recency of training data; and having separate models for different hours of a data stream) for forecasting network data streams. We explored the forecasting per-second flow arrival rate for all incoming flows and incoming web flows using LSTM. Our results provide valuable insights into the ability of the forecasting models for short-term forecasting of the two data streams and elucidate the effects of training strategies, input features, among others, on the accuracy of models.

The main limitation of our study in this chapter was its focus on two specific network data streams from a particular campus network. While our methodology is certainly applicable to forecasting any data stream from any network, our trained models and findings cannot be generalized. More importantly, we argue that such a generalization of ML models—as it is done in other domains (e.g., image classification)—may not necessarily be feasible in networking. This observation is motivated by the fact that network data streams are likely to exhibit diverse (short term) characteristics across different networks. This, in turn, suggests that the training and deployment of ML models should be customized for a specific data stream from a particular network to ensure high accuracy. In short, any modeling study in networking is likely to be specific to its target setting.

In Chapter V we used ML techniques to detect anomalous events in the behavior of a network application. We focused on a web server as our choice of network application since web servers are a popular example and yet one of the most vulnerable network applications. Therefore, they can serve as a representative example. We assessed the effect of using additional attributes from the network, operating system, and the application itself on the accuracy of our model. Using

LSTM auto-encoder models, we showed that our model is able to detect two types of anomalies accurately, the application layer attacks including slowbody, slowloris, slowread, and TCP flooding attacks as well as spikes in legitimate requests. Furthermore, we have shown improvement of up to 0.30 in F1-score when using all possible feature sources together.

Given that the accuracy of a model is not the only factor that affects the practical use of ML methods and understanding the inner working logic and rationale of decisions is also important, we used two types of model explainability techniques (interpretability and rule extraction techniques) to assess how effective the standard explanation techniques are on our models.

6.1 Future Work

In line with the presented dissertation topics, we present several possible directions for future work:

6.1.1 Online Reviews Analysis. Our efforts in analyzing the incentivized online reviews can be improved by incorporating other explicit patterns. Furthermore, we can deploy probabilistic techniques to infer the likelihood that a review is incentivized based on its affiliation with other products and reviewers while considering the fraction of EIRs for affiliated products or reviewers. We can also cast the problem in a ranking setting instead of binary classification.

6.1.2 Forecasting Network Data Streams. This project can be further extended along with the following directions. For one, the speed-accuracy trade-offs associated with automated inference determine how well a given telemetry task as a whole can be performed with no human operator in the loop. While the idea of exploiting the diversity in available compute and communication resources and programmability capabilities among the different hardware components to achieve the “best case” scenario is already being explored (Gupta et al., 2018), how to get the network to recognize such “best case” scenarios and then operate at such “sweet spots” remains an open problem. Moreover, other open problems include creating theoretical bounds on the time required and accuracy desired to perform a certain network telemetry task, and architectural designs that are needed to enable such “sweet spot-seeking” network telemetry at scale; that is, executing hundreds or thousands of highly diverse network telemetry tasks concurrently and as fast and accurately as possible despite the uncertainties in the environment (e.g. traffic load, application mix, failure scenarios). In addition, the effect of using “scheduled sampling" Bengio, Vinyals, Jaitly, and Shazeer (2015) as a technique to improve LSTM training can be investigated, and hopefully, it can inform those looking to build a more powerful technique based on this project’s idea.

6.1.3 Anomaly Detection using Application Behavior Modeling.

Deployment of our proposed system can help to gain insight into the performance and practical challenges of anomaly detection beyond the numeric accuracy of our model and will guide us to improve our model and system architecture. Our proposed methods can be further evaluated against similar methods from prior studies as the baseline. The effect of more courser time granularity (e.g. per minute or hour) can be assessed. Also, the effect of a wider range of attacks can be more informative. The idea of choosing the threshold based on the percentile of an error on the evaluation set can be further improved using a reinforcement learning model. Also, the model can be used as a part of multiple models trained on data from different parts of the network to provide input data for a meta learner model.

Furthermore, developing a terminal software can be useful to keep track of our model's detection and act as a dashboard for the network admins.

REFERENCES CITED

Aborujilah, A., & Musa, S. (2017). Cloud-based ddos http attack detection using covariance matrix approach. *Journal of Computer Networks and Communications*, *2017*.

aboutAmazon.com. (2016). *Update customer review.* `https://goo.gl/fiVa8j`.

Adebiyi, A. A., Adewumi, A. O., & Ayo, C. K. (2014). Comparison of ARIMA and artificial neural networks models for stock price prediction. *Journal of Applied Mathematics*.

Akoglu, L., Chandy, R., & Faloutsos, C. (2013). Opinion fraud detection in online reviews by network effects. *Proc. of the ICWSM*.

Amazon.com. (2018). *About amazon verified purchase reviews.* `https://goo.gl/aNcPCR`.

Amazon's Community Guidelines. (2018). *Community guidelines.* `https://www.amazon.com/gp/help/customer/display.html?nodeId=14279631`.

Amer, M., Goldstein, M., & Abdennadher, S. (2013). Enhancing one-class support vector machines for unsupervised anomaly detection. In *Proceedings of the acm sigkdd workshop on outlier detection and description* (pp. 8–15).

Ancona, M., Ceolini, E., Öztireli, C., & Gross, M. (2017). Towards better understanding of gradient-based attribution methods for deep neural networks. *arXiv preprint arXiv:1711.06104*.

Aqil, A., Khalil, K., Atya, A. O., Papalexakis, E. E., Krishnamurthy, S. V., Jaeger, T., . . . Swami, A. (2017). Jaal: Towards network intrusion detection at isp scale. In *Proceedings of the 13th international conference on emerging networking experiments and technologies* (pp. 134–146).

Ariyo, A. A., Adewumi, A. O., & Ayo, C. K. (2014). Stock price prediction using the ARIMA model. In *Uksim-amss international conference on computer modelling and simulation* (pp. 106–112).

Arlitt, M. F., & Williamson, C. L. (1996). *Web server workload characterization: The search for invariants (extended version)* (Tech. Rep.). Citeseer.

Azari, A., Papapetrou, P., Denic, S., & Peters, G. (2019). Cellular traffic prediction and classification: a comparative evaluation of LSTM and ARIMA. *arXiv preprint arXiv:1906.00939*.

Azzouni, A., & Pujolle, G. (2017). A long short-term memory recurrent neural network framework for network traffic matrix prediction. *arXiv preprint arXiv:1705.05690*.

Bach, S., Binder, A., Montavon, G., Klauschen, F., Müller, K.-R., & Samek, W. (2015). On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS one*, *10*(7).

Bastani, O., Kim, C., & Bastani, H. (2017a). Interpretability via model extraction. *arXiv preprint arXiv:1706.09773*.

Bastani, O., Kim, C., & Bastani, H. (2017b). Interpreting blackbox models via model extraction. *arXiv preprint arXiv:1705.08504*.

Bengio, S., Vinyals, O., Jaitly, N., & Shazeer, N. (2015). Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in neural information processing systems* (pp. 1171–1179).

Bhatia, R., Benno, S., Esteban, J., Lakshman, T., & Grogan, J. (2019). Unsupervised machine learning for network-centric anomaly detection in iot. In *Proceedings of the 3rd acm conext workshop on big data, machine learning and artificial intelligence for data communication networks* (pp. 42–48).

Bishop, T. (2015). *Product rating calculation.* `https://www.geekwire.com/2015/amazon-changes-its-influential-formula-for-calculating-product-ratings/`.

Bonesi ddos tool. (n.d.). Retrieved from `https://github.com/Markus-Go/bonesi`

Box, G., & Jenkins, G. (1976). Time series analysis: forecasting and control rev. *Oakland California Holden-Day.*.

Boz, O. (2002). Extracting decision trees from trained neural networks. In *Proceedings of the eighth acm sigkdd international conference on knowledge discovery and data mining* (pp. 456–461).

Brauckhoff, D., Salamatian, K., & May, M. (2009). Applying pca for traffic anomaly detection: Problems and solutions. In *Ieee infocom 2009* (pp. 2866–2870).

Breiman, L. (2001). Random forests. *Machine learning*, *45*(1), 5–32.

Burtch, G., Hong, Y., Bapna, R., & Griskevicius, V. (2017). Stimulating online reviews by combining financial incentives and social norms. *Management Science*.

Businessinsider. (2017). *Australian uber drivers say the company is manipulating their ratings to boost its fees.* `https://www.businessinsider.com.au/australian-uber-drivers-say-the-company-is-manipulating-their-ratings-to-boost-the-companys-fees-2016-5`.

Caida ddos 2007 attack dataset. (n.d.). Retrieved from `https://www.impactcybertrust.org/dataset_view?idDataset=117`

Canini, M., Li, W., & Moore, A. W. (2009). Toward the identification of anonymous web proxies. In *Pam-international conference on passive and active network measurement.*

Cao, Y., Nejati, J., Balasubramanian, A., & Gandhi, A. (2019). Econ: Modeling the network to improve application performance. In *Proceedings of the internet measurement conference* (pp. 365–378).

Carter, K. M., Lippmann, R. P., & Boyer, S. W. (2010). Temporally oblivious anomaly detection on large networks using functional peers. In *Proceedings of the conference on internet measurement conference* (pp. 465–471). doi: 10.1145/1879141.1879201

Carvalho, D. V., Pereira, E. M., & Cardoso, J. S. (2019). Machine learning interpretability: A survey on methods and metrics. *Electronics*, *8*(8), 832.

Casas, P., & Vanerio, J. (2017). Super learning for anomaly detection in cellular networks. In *Wireless and mobile computing, networking and communications (wimob),* (pp. 1–8).

Castro, J., Gómez, D., & Tejada, J. (2009). Polynomial calculation of the shapley value based on sampling. *Computers & Operations Research*, *36*(5), 1726–1730.

Castro-Neto, M., Jeong, Y.-S., Jeong, M.-K., & Han, L. D. (2009). Online-svr for short-term traffic flow prediction under typical and atypical traffic conditions. *Expert systems with applications*, *36*(3), 6164–6173.

Celenk, M., Conley, T., Graham, J., & Willis, J. (2008). Anomaly prediction in network traffic using adaptive wiener filtering and arma modeling. In *2008 ieee international conference on systems, man and cybernetics* (pp. 3548–3553).

Chen, L., Mislove, A., & Wilson, C. (2015). Peeking beneath the hood of uber. In *Proc. of the 2015 acm conference on internet measurement conference* (pp. 495–508).

Chen, X., Feibish, S. L., Koral, Y., Rexford, J., & Rottenstreich, O. (2018). Catching the microburst culprits with snappy. In *Afternoon workshop on self-driving networks* (pp. 22–28).

Clark, P., & Niblett, T. (1989). The cn2 induction algorithm. *Machine learning*, *3*(4), 261–283.

Cohen, W. W. (1995). Fast effective rule induction. In *Machine learning proceedings 1995* (pp. 115–123). Elsevier.

Contreras, J., Espinola, R., Nogales, F. J., & Conejo, A. J. (2003). ARIMA models to predict next-day electricity prices. *IEEE transactions on power systems*, *18*(3), 1014–1020.

Cortez, P., Rio, M., Rocha, M., & Sousa, P. (2012). Multi-scale internet traffic forecasting using neural networks and time series methods. *Expert Systems*, *29*(2), 143–155.

Craven, M. W. (1996). *Extracting comprehensible models from trained neural networks* (Tech. Rep.). University of Wisconsin-Madison Department of Computer Sciences.

d'Agostino, R. B. (1971). An omnibus test of normality for moderate and large size samples. *Biometrika*, *58*(2), 341–348.

Dai, X., Fu, R., Lin, Y., Li, L., & Wang, F.-Y. (2017). Deeptrend: A deep hierarchical neural network for traffic flow prediction. *arXiv preprint arXiv:1707.03213*.

datanyze.com. (2020). *Webservers market share.* `https://www.datanyze.com/market-share/web-and-application-servers--425/apache-http-server-market-share`. ([Online; accessed 19-June-2020])

Datta, A., Tschantz, M. C., & Datta, A. (2015). Automated experiments on ad privacy settings. *Proc. on Privacy Enhancing Technologies*, *2015*(1), 92–112.

Davidov, D., Tsur, O., & Rappoport, A. (2010). Semi-supervised recognition of sarcastic sentences in twitter and amazon. In *Proc. of the 14th conference on cnll.*

Ding, Z., & Fei, M. (2013). An anomaly detection approach based on isolation forest algorithm for streaming data using sliding window. *IFAC Proceedings Volumes*, *46*(20), 12–17.

Edelman, B., Luca, M., & Svirsky, D. (2017). Racial discrimination in the sharing economy: Evidence from a field experiment. *American Economic Journal: Applied Economics*, *9*(2), 1–22.

Ediger, V. Ş., & Akar, S. (2007). ARIMA forecasting of primary energy demand by fuel in Turkey. *Energy policy*, *35*(3), 1701–1708.

Enders, W. (2008). *Applied econometric time series.* John Wiley & Sons.

Erfani, S. M., Rajasegarar, S., Karunasekera, S., & Leckie, C. (2016). High-dimensional and large-scale anomaly detection using a linear one-class svm with deep learning. *Pattern Recognition*, *58*, 121–134.

Eslami, M., Vaccaro, K., Karahalios, K., & Hamilton, K. (2017). " be careful; things can be worse than they appear": Understanding biased algorithms and users' behavior around them in rating platforms. In *Proc. of the icwsm.*

Feng, Y., Li, J., & Nguyen, T. (2020). Towards intelligent defense against application-layer ddos with reinforcement learning. *IEEE/ACM International Symposium on Quality of Service (IWQoS).*

Ferenti, T. (2017). Biomedical applications of time series analysis. In *2017 ieee 30th neumann colloquium (nc)* (pp. 000083–000084).

Fernandes Jr, G., Carvalho, L. F., Rodrigues, J. J., & Proença Jr, M. L. (2016). Network anomaly detection using ip flows with principal component analysis and ant colony optimization. *Journal of Network and Computer Applications*, *64*, 1–11.

Fisher, A., Rudin, C., & Dominici, F. (2018). Model class reliance: Variable importance measures for any machine learning model class, from the" rashomon" perspective. *arXiv preprint arXiv:1801.01489*, *68*.

FiveThrityEight. (2017). *Be suspicious of online movie ratings, especially fandangos.* `https://fivethirtyeight.com/features/fandango-movies-ratings/`.

Fong, R. C., & Vedaldi, A. (2017). Interpretable explanations of black boxes by meaningful perturbation. In *Proceedings of the ieee international conference on computer vision* (pp. 3429–3437).

Gilbert, K. (2005). An ARIMA supply chain model. *Management Science*, *51*(2), 305–310.

Gondara, L. (2016). Medical image denoising using convolutional denoising autoencoders. In *2016 ieee 16th international conference on data mining workshops (icdmw)* (pp. 241–246).

Gunning, R. (1952). The technique of clear writing. *McGraw-Hill, NY*.

Gupta, A., Harrison, R., Canini, M., Feamster, N., Rexford, J., & Willinger, W. (2018). Sonata: Query-driven streaming network telemetry. In *Acm special interest group on data communication* (pp. 357–371).

Hamamoto, A. H., Carvalho, L. F., Sampaio, L. D. H., Abrão, T., & Proença Jr, M. L. (2018). Network anomaly detection system using genetic algorithm and fuzzy logic. *Expert Systems with Applications*, *92*, 390–402.

Hameed, S., & Ali, U. (2018). Hadec: Hadoop-based live ddos detection framework. *EURASIP Journal on Information Security*, *2018*(1), 1–19.

Hatami, N., Gavet, Y., & Debayle, J. (2018). Classification of time-series images using deep convolutional neural networks. In *Tenth international conference on machine vision (icmv 2017)* (Vol. 10696, p. 106960Y).

Himura, Y., Fukuda, K., Cho, K., & Esaki, H. (2009). Quantifying host-based application traffic with multi-scale gamma model. In *Pam-international conference on passive and active network measurement* (pp. 2–3).

Hirakawa, T., Ogura, K., Bista, B. B., & Takata, T. (2016). A defense method against distributed slow http dos attack. In *2016 19th international conference on network-based information systems (nbis)* (pp. 152–158).

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, *9*(8), 1735–1780.

Hoque, N., Bhattacharyya, D. K., & Kalita, J. K. (2015). Botnet in ddos attacks: trends and challenges. *IEEE Communications Surveys & Tutorials*, *17*(4), 2242–2270.

Hoque, N., Kashyap, H., & Bhattacharyya, D. K. (2017). Real-time ddos attack detection using fpga. *Computer Communications*, *110*, 48–58.

Hua, Y., Zhao, Z., Li, R., Chen, X., Liu, Z., & Zhang, H. (2017). Traffic prediction based on random connectivity in deep learning with long short-term memory. *arXiv preprint arXiv:1711.02833*.

Hulk dos tool. (n.d.). Retrieved from `https://github.com/Hyperclaw79/HULK-v3`

Hyndman, R. J., & Athanasopoulos, G. (2018). *Forecasting: principles and practice.* OTexts.

Hyndman, R. J., & Koehler, A. B. (2006). Another look at measures of forecast accuracy. *International journal of forecasting*, *22*(4), 679–688.

Ingham, K. L., & Inoue, H. (2007). Comparing anomaly detection techniques for http. In *International workshop on recent advances in intrusion detection* (pp. 42–62).

Jaafar, G. A., Abdullah, S. M., & Ismail, S. (2019). Review of recent detection methods for http ddos attack. *Journal of Computer Networks and Communications*, *2019*.

Jacobsson, H. (2005). Rule extraction from recurrent neural networks: A taxonomy and review. *Neural Computation*, *17*(6), 1223–1263.

Jamshidi, S., Hammoudeh, Z., Durairajan, R., Lowd, D., Rejaie, R., & Willinger, W. (2020a). On the practicality of learning modelsfor network telemetry. In *Network traffic measurement and analysis conference (tma).*

Jamshidi, S., Hammoudeh, Z., Durairajan, R., Lowd, D., Rejaie, R., & Willinger, W. (2020b). On the practicality of learning modelsfor network telemetry. In *Network traffic measurement and analysis conference (tma).*

Jamshidi, S., Rejaie, R., & Li, J. (2018). Trojan horses in amazon's castle: Understanding the incentivized online reviews. In *International conference on advances in social networks analysis and mining (asonam)* (pp. 335–342).

Jamshidi, S., Rejaie, R., & Li, J. (2019). Characterizing the dynamics and evolution of incentivized online reviews on amazon. *Social Network Analysis and Mining*, *9*(1), 22.

Janardhanan, D., & Barrett, E. (2017). Cpu workload forecasting of machines in data centers using LSTM recurrent neural networks and ARIMA models. In *International conference for internet technology and secured transactions* (pp. 55–60).

Jiang, J., & Papavassiliou, S. (2006). Enhancing network traffic prediction and anomaly detection via statistical network traffic separation and combination strategies. *Computer communications*, *29*(10), 1627–1638.

Jindal, N., & Liu, B. (2008). Opinion spam and analysis. In *Acm international conference on web search and data mining.*

Jindal, N., Liu, B., & Lim, E.-P. (2010). Finding unusual review patterns using unexpected rules. In *Proc. of the acm international conference on information and knowledge management.*

Johnson Singh, K., Thongam, K., & De, T. (2016). Entropy-based application layer ddos attack detection using artificial neural networks. *Entropy*, *18*(10), 350.

Journal, T. W. S. (2017). *Judge dismisses suit against yelp.* https://www.wsj.com/articles/SB10001424052970204505304577002170423750412.

Kang, Y., Hyndman, R. J., & Li, F. (2019). Gratis: Generating time series with diverse and controllable characteristics. *arXiv preprint arXiv:1903.02787*.

Kdd cup 99. (n.d.). Retrieved from http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

Kim, B., Chang, D., Heo, J., & Shin, S. Y. (2020). Time series analysis for enhancing the recognition of license plate number in video stream of iot camera. In *Proceedings of the 35th annual acm symposium on applied computing* (pp. 1901–1905).

Kim, B., Khanna, R., & Koyejo, O. O. (2016). Examples are not enough, learn to criticize! criticism for interpretability. In *Advances in neural information processing systems* (pp. 2280–2288).

Kim, S.-M., Pantel, P., Chklovski, T., & Pennacchiotti, M. (2006). Automatically assessing review helpfulness. In *Proc. of the acl conference on empirical methods in natural language processing.*

Krishnan, R., Sivakumar, G., & Bhattacharya, P. (1999a). Extracting decision trees from trained neural networks. *Pattern recognition*, *32*(12).

Krishnan, R., Sivakumar, G., & Bhattacharya, P. (1999b). A search technique for rule extraction from trained neural networks. *Pattern Recognition Letters*, *20*(3), 273–280.

Kruskal, W. H., & Wallis, W. A. (1952). Use of ranks in one-criterion variance analysis. *Journal of the American statistical Association*, *47*(260), 583–621.

Kwiatkowski, D., Phillips, P. C., Schmidt, P., & Shin, Y. (1992). Testing the null hypothesis of stationarity against the alternative of a unit root: How sure are we that economic time series have a unit root? *Journal of econometrics*, *54*(1-3), 159–178.

Lakkaraju, H., Bach, S. H., & Leskovec, J. (2016). Interpretable decision sets: A joint framework for description and prediction. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining* (pp. 1675–1684).

Lazaris, A., & Prasanna, V. K. (2019). An LSTM framework for modeling network traffic. In *Ifip/ieee symposium on integrated network and service management* (pp. 19–24).

Letham, B., Rudin, C., McCormick, T. H., & Madigan, D. (2015). Interpretable classifiers using rules and bayesian analysis: Building a better stroke prediction model. *The Annals of Applied Statistics*, *9*(3), 1350–1371.

Li, F., Huang, M., Yang, Y., & Zhu, X. (2011). Learning to identify review spam. In *Proc. of ijcai.*

Li, S., Kawale, J., & Fu, Y. (2015). Deep collaborative filtering via marginalized denoising auto-encoder. In *Proceedings of the 24th acm international on conference on information and knowledge management* (pp. 811–820).

Li, S., Zhang, K., Duan, P., & Kang, X. (2019). Hyperspectral anomaly detection with kernel isolation forest. *IEEE Transactions on Geoscience and Remote Sensing*, *58*(1), 319–329.

Li, Y., Miao, R., Kim, C., & Yu, M. (2016). Flowradar: A better netflow for data centers. In *USENIX symposium on networked systems design and implementation NSDI* (pp. 311–324).

Liao, Q., Li, H., Kang, S., & Liu, C. (2015). Application layer ddos attack detection using cluster with label based on sparse vector decomposition and rhythm matching. *Security and Communication Networks*, *8*(17), 3111–3120.

Lim, P., Nguyen, V., Jindal, N., Liu, B., & Lauw, H. (2010). Detecting product review spammers using rating behaviors. In *Proc. of acm international conference on info. and knowledge management.*

Lippmann, R., Haines, J. W., Fried, D. J., Korba, J., & Das, K. (2000). The 1999 darpa off-line intrusion detection evaluation. *Computer networks*, *34*(4), 579–595.

Lipton, Z. C. (2018). The mythos of model interpretability. *Queue*, *16*(3), 31–57.

Liu, B., Hsu, W., & Ma, Y. (1998). Integrating classification and association rule mining. In *Kdd* (Vol. 98, pp. 80–86).

Liu, D., Zhao, Y., Xu, H., Sun, Y., Pei, D., Luo, J., … Feng, M. (2015). Opprentice: Towards practical and automatic anomaly detection through machine learning. In *Proceedings of the internet measurement conference* (pp. 211–224). doi: 10.1145/2815675.2815679

Liu, F. T., Ting, K. M., & Zhou, Z.-H. (2008). Isolation forest. In *2008 eighth ieee international conference on data mining* (pp. 413–422).

Liu, J., Cao, Y., Lin, C.-Y., Huang, Y., & Zhou, M. (2007). Low-quality product review detection in opinion summarization. In *Proc. of the joint conference on emnlp-conll.*

Liu, Z., Manousis, A., Vorsanger, G., Sekar, V., & Braverman, V. (2016). One sketch to rule them all: Rethinking network flow monitoring with UnivMon. In *Acm sigcomm conference* (pp. 101–114).

Ljung, G. M., & Box, G. E. (1978). On a measure of lack of fit in time series models. *Biometrika*, *65*(2), 297–303.

Low orbit ion cannon. (n.d.). Retrieved from `https://github.com/NewEraCracker/LOIC`

Lu, W., & Ghorbani, A. A. (2008). Network anomaly detection based on wavelet analysis. *EURASIP Journal on Advances in Signal Processing*, *2009*, 1–16.

Lu, X., Tsao, Y., Matsuda, S., & Hori, C. (2013). Speech enhancement based on deep denoising autoencoder. In *Interspeech* (Vol. 2013, pp. 436–440).

Luo, T., & Nagarajan, S. G. (2018). Distributed anomaly detection using autoencoder neural networks in wsn for iot. In *2018 ieee international conference on communications (icc)* (pp. 1–6).

Ma, X., Dai, Z., He, Z., Ma, J., Wang, Y., & Wang, Y. (2017). Learning traffic as images: a deep convolutional neural network for large-scale transportation network speed prediction. *Sensors*, *17*(4), 818.

Mac, H., Truong, D., Nguyen, L., Nguyen, H., Tran, H. A., & Tran, D. (2018). Detecting attacks on web applications using autoencoder. In *Proceedings of the ninth international symposium on information and communication technology* (pp. 416–421).

Mandal, S., Santhi, B., Sridhar, S., Vinolia, K., & Swaminathan, P. (2019). Minor fault detection of thermocouple sensor in nuclear power plants using time series analysis. *Annals of Nuclear Energy*, *134*, 383–389.

Marnerides, A. K., Pezaros, D. P., Kim, H.-c., & Hutchison, D. (2009). Unsupervised two-class and multi-class support vector machines for abnormal traffic characterization. In *Pam-student workshop on passive and active network measurement.*

Mei, L., Hu, R., Cao, H., Liu, Y., Han, Z., Li, F., & Li, J. (2019). Realtime mobile bandwidth prediction using lstm neural network. In *International conference on passive and active network measurement* (pp. 34–47).

Michael, A. K. J., Valla, E., Neggatu, N. S., & Moore, A. W. (2017). *Network traffic classification via neural networks* (Tech. Rep.). University of Cambridge, Computer Laboratory.

Miller, T. (2019). Explanation in artificial intelligence: Insights from the social sciences. *Artificial Intelligence*, *267*, 1–38.

Mirza, M., Sommers, J., Barford, P., & Zhu, X. (2010). A machine learning approach to TCP throughput prediction. *IEEE/ACM Transactions on Networking (TON)*, *18*(4), 1026–1039.

Mohamed, A.-r., Seide, F., Yu, D., Droppo, J., Stoicke, A., Zweig, G., & Penn, G. (2015). Deep bi-directional recurrent networks over spectral windows. In *Ieee workshop on automatic speech recognition and understanding (asru)* (pp. 78–83).

Moliner, J., & Epifanio, I. (2019). Robust multivariate and functional archetypal analysis with application to financial time series analysis. *Physica A: Statistical Mechanics and its Applications*, *519*, 195–208.

Mudambi, S. (2010). What makes a helpful online review? a study of customer reviews on amazon.com. *MIS Quarterly*, *34*, 185–200.

Narayana, S., Sivaraman, A., Nathan, V., Goyal, P., Arun, V., Alizadeh, M., … Kim, C. (2017). Language-directed hardware design for network performance monitoring. In *Acm special interest group on data communication* (pp. 85–98).

Ndibwile, J. D., Govardhan, A., Okada, K., & Kadobayashi, Y. (2015). Web server protection against application layer ddos attacks using machine learning and traffic authentication. In *2015 ieee 39th annual computer software and applications conference* (Vol. 3, pp. 261–267).

Nevat, I., Divakaran, D. M., Nagarajan, S. G., Zhang, P., Su, L., Ko, L. L., & Thing, V. L. L. (2018). Anomaly detection and attribution in networks with temporally correlated traffic. *IEEE/ACM Transactions on Networking (ToN)*, *26*(1), 131–144.

Nie, L., Jiang, D., Yu, S., & Song, H. (2017). Network traffic prediction based on deep belief network in wireless mesh backbone networks. In *Ieee wireless communications and networking conference (wcnc)* (pp. 1–5).

Nikravesh, A. Y., Ajila, S. A., Lung, C.-H., & Ding, W. (2016). Mobile network traffic prediction using MLP, MLPWD, and SVM. In *Big data (bigdata congress)* (pp. 402–409).

Ordóñez, F., & Roggen, D. (2016). Deep convolutional and lstm recurrent neural networks for multimodal wearable activity recognition. *Sensors*, *16*(1), 115.

Ott, M., Choi, Y., Cardie, C., & Hancock, J. T. (2011). Finding deceptive opinion spam by any stretch of the imagination. In *Proc. of the acl human language technologies.*

Oza, A., Ross, K., Low, R. M., & Stamp, M. (2014). Http attack detection using n-gram analysis. *Computers & Security*, *45*, 242–254.

Pena, E. H., de Assis, M. V., & Proença, M. L. (2013). Anomaly detection using forecasting methods arima and hwds. In *2013 32nd international conference of the chilean computer science society (sccc)* (pp. 63–66).

Perdisci, R., Gu, G., & Lee, W. (2006). Using an ensemble of one-class svm classifiers to harden payload-based anomaly detection systems. In *Sixth international conference on data mining (icdm'06)* (pp. 488–498).

Petrescu, M., O'Leary, K., Goldring, D., & Mrad, S. B. (2017). Incentivized reviews: Promising the moon for a few stars. *Journal of Retailing and Consumer Services*.

Puggini, L., & McLoone, S. (2018). An enhanced variable selection and isolation forest based methodology for anomaly detection with oes data. *Engineering Applications of Artificial Intelligence*, *67*, 126–135.

Qiao, D., Lee, S.-Y., Whinston, A., & Wei, Q. (2017). Incentive provision and pro-social behaviors. In *Proc. of the hawaii international conference on system sciences.*

Radford, B. J., Apolonio, L. M., Trias, A. J., & Simpson, J. A. (2018). Network traffic anomaly detection using recurrent neural networks. *arXiv preprint arXiv:1803.10769*.

Rasti, A. H., Magharei, N., Rejaie, R., & Willinger, W. (2010). Eyeball ases: from geography to connectivity. In *Proceedings of the 10th acm sigcomm conference on internet measurement* (pp. 192–198).

Rasti, A. H., Torkjazi, M., Rejaie, R., Stutzbach, D., Duffield, N., & Willinger, W. (2008). Evaluating sampling techniques for large dynamic graphs. *Univ. Oregon, Tech. Rep. CIS-TR-08*, *1*.

Rejaie, R., Torkjazi, M., Valafar, M., & Willinger, W. (2010). Sizing up online social networks. *IEEE network*, *24*(5), 32–37.

ReviewMeta.com. (2016). *Analysis of 7 million amazon reviews.* `https://goo.gl/CPzHpB`.

Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). Why should i trust you?: Explaining the predictions of any classifier. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining* (pp. 1135–1144).

Ring, M., Wunderlich, S., Scheuring, D., Landes, D., & Hotho, A. (2019). A survey of network-based intrusion detection data sets. *Computers & Security*.

Roughan, M., Zhang, Y., Willinger, W., & Qiu, L. (2012). Spatio-temporal compressive sensing and internet traffic matrices. *IEEE/ACM Transactions on Networking (ToN)*, *20*(3), 662–676.

R u dead yet (rudy) ddos tool. (n.d.). Retrieved from `https://github.com/nosperantos/RUDY`

Sakurada, M., & Yairi, T. (2014). Anomaly detection using autoencoders with nonlinear dimensionality reduction. In *Proceedings of the mlsda 2014 2nd workshop on machine learning for sensory data analysis* (pp. 4–11).

Saleh, K., Hossny, M., & Nahavandi, S. (2017). Driving behavior classification based on sensor data fusion using LSTM recurrent neural networks. In *International conference on intelligent transportation systems (itsc)* (pp. 1–6).

Sandvig, C., Hamilton, K., Karahalios, K., & Langbort, C. (2014). Auditing algorithms: Research methods for detecting discrimination on internet platforms. *Data and discrimination: converting critical concerns into productive inquiry*.

Santanna, J. J., van Rijswijk-Deij, R., Hofstede, R., Sperotto, A., Wierbosch, M., Granville, L. Z., & Pras, A. (2015). Booters—an analysis of DDoS-as-a-service attacks. In *Ifip/ieee international symposium on integrated network management* (pp. 243–251).

Shekyan, S. (2020). *Configurable slow HTTP attack.* `https://github.com/shekyan/slowhttptest/wiki/InstallationAndUsage`. ([Online; accessed 19-June-2020])

Shiaeles, S. N., & Papadaki, M. (2015). Fhsd: an improved ip spoof detection method for web ddos attacks. *The Computer Journal*, *58*(4), 892–903.

Shrikumar, A., Greenside, P., & Kundaje, A. (2017). Learning important features through propagating activation differences. In *Proceedings of the 34th international conference on machine learning-volume 70* (pp. 3145–3153).

Shyong, K., Frankowski, D., & Riedl, J. (2006). Do you trust your recommendations? an exploration of security and privacy issues in recommender systems. *Emerging Trends in ICS*.

Simonyan, K., Vedaldi, A., & Zisserman, A. (2013). Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*.

Singh, K., Singh, P., & Kumar, K. (2018). User behavior analytics-based classification of application layer http-get flood attacks. *Journal of Network and Computer Applications*, *112*, 97–114.

slowloris ddos tool. (n.d.). Retrieved from `https://github.com/gkbrk/slowloris`

Smilkov, D., Thorat, N., Kim, B., Viégas, F., & Wattenberg, M. (2017). Smoothgrad: removing noise by adding noise. *arXiv preprint arXiv:1706.03825*.

Soares, E., Costa Jr, P., Costa, B., & Leite, D. (2018). Ensemble of evolving data clouds and fuzzy models for weather time series prediction. *Applied Soft Computing*, *64*, 445–453.

Soeller, G., Karahalios, K., Sandvig, C., & Wilson, C. (2016). Mapwatch: Detecting and monitoring international border personalization on online maps. In *Proc. of the 25th international conference on world wide web* (pp. 867–878).

Soheil, J., Reza, R., & Jun, L. (2016-18). Characterizing the incentivized online reviews. *Technical Report , University of Oregon*. Retrieved from `https://www.cs.uoregon.edu/Reports/TR-2018-001.pdf`

Sommer, R., & Paxson, V. (2010). Outside the closed world: On using machine learning for network intrusion detection. In *Security and privacy (sp), ieee symposium on* (pp. 305–316).

Soule, A., Lakhina, A., Taft, N., Papagiannaki, K., Salamatian, K., Nucci, A., . . . Diot, C. (2005). Traffic matrices: balancing measurements, inference and modeling. In *Acm sigmetrics performance evaluation review* (Vol. 33, pp. 362–373).

Sreeram, I., & Vuppala, V. P. K. (2019). Http flood attack detection in application layer using machine learning metrics and bio inspired bat algorithm. *Applied computing and informatics*, *15*(1), 59–66.

Sushil, M., Šuster, S., & Daelemans, W. (2018). Rule induction for global explanation of trained models. *arXiv preprint arXiv:1808.09744*.

Sutiene, K., Vilutis, G., & Sandonavicius, D. (2011). Forecasting of grid job waiting time from imputed time series. *Elektronika ir Elektrotechnika*, *114*(8), 101–106.

Tofino. (2020). `https://www.barefootnetworks.com/products/brief-tofino-2/`. ([Online; accessed 10-May-2020])

Van der Laan, M. J., Polley, E. C., & Hubbard, A. E. (2007). Super learner. *Statistical applications in genetics and molecular biology*, *6*(1).

Vartouni, A. M., Teshnehlab, M., & Kashi, S. S. (2019). Leveraging deep neural networks for anomaly-based web application firewall. *IET Information Security*, *13*(4), 352–361.

Vinayakumar, R., Soman, K., & Poornachandran, P. (2017). Applying deep learning approaches for network traffic prediction. In *Advances in computing, communications and informatics (icacci)* (pp. 2353–2358).

Wang, J., Ghose, A., & Ipeirotis, P. (2012). Bonus, disclosure, and choice: What motivates the creation of high-quality paid reviews? In *Proc. of the international conference on information systems.*

Wang, X., Zhou, Z., Yang, Z., Liu, Y., & Peng, C. (2017). Spatio-temporal analysis and prediction of cellular traffic in metropolis. In *International conference on network protocols (icnp)* (pp. 1–10).

Wang, Z., & Oates, T. (2015a). Encoding time series as images for visual inspection and classification using tiled convolutional neural networks. In *Workshops at the twenty-ninth aaai conference on artificial intelligence* (Vol. 1).

Wang, Z., & Oates, T. (2015b). Imaging time-series to improve classification and imputation. *arXiv preprint arXiv:1506.00327*.

Web page replay. (n.d.). Retrieved from `https://github.com/catapult-project/catapult/`

Xie, Y., & Yu, S.-Z. (2008). Monitoring the application-layer ddos attacks for popular websites. *IEEE/ACM Transactions on networking*, *17*(1), 15–25.

Xie, Z., & Zhu, S. (2015). Appwatcher: Unveiling the underground market of trading mobile app reviews. In *Proc. of the acm conference on security & privacy in wireless and mobile networks.* Retrieved from `http://doi.acm.org/10.1145/2766498.2766510` doi: 10.1145/2766498.2766510

Xu, H., Chen, W., Zhao, N., Li, Z., Bu, J., Li, Z., . . . Feng, Y. (2018). Unsupervised anomaly detection via variational auto-encoder for seasonal kpis in web applications. In *Proceedings of the 2018 world wide web conference* (pp. 187–196).

Yadav, S., & Subramanian, S. (2016). Detection of application layer ddos attack by feature learning using stacked autoencoder. In *2016 international conference on computational techniques in information and communication technologies (icctict)* (pp. 361–366).

Yamansavascilar, B., Guvensan, M. A., Yavuz, A. G., & Karsligil, M. E. (2017). Application identification via network traffic classification. In *Computing, networking and communications (icnc)* (pp. 843–848).

Yang, B., Sun, S., Li, J., Lin, X., & Tian, Y. (2019). Traffic flow prediction using LSTM with feature enhancement. *Neurocomputing*, *332*, 320–327.

Yeganeh, B., Rejaie, R., & Willinger, W. (2017). A view from the edge: A stub-as perspective of traffic localization and its implications. In *2017 network traffic measurement and analysis conference (tma)* (pp. 1–9).

Yu, R., Li, Y., Shahabi, C., Demiryurek, U., & Liu, Y. (2017). Deep learning: A generic approach for extreme condition traffic forecasting. In *Siam international conference on data mining* (pp. 777–785).

Zang, Y., Ni, F., Feng, Z., Cui, S., & Ding, Z. (2015). Wavelet transform processing for cellular traffic prediction in machine learning networks. In *Signal and information processing (chinasip), ieee china summit and international conference on* (pp. 458–462).

Zarate, L., Vimieiro, R., & Vieira, N. (2006). Reasoning based on rules extracted from trained neural networks via formal concept analysis. In *2006 ieee international conference on engineering of intelligent systems* (pp. 1–6).

Zeiler, M. D., & Fergus, R. (2014). Visualizing and understanding convolutional networks. In *European conference on computer vision* (pp. 818–833).

Zhang, J., Chen, X., Xiang, Y., Zhou, W., & Wu, J. (2015). Robust network traffic classification. *IEEE/ACM Transactions on Networking (ToN)*, *23*(4), 1257–1270.

Zhang, X., Wang, N., Ji, S., Shen, H., & Wang, T. (2018). Interpretable deep learning under fire. *arXiv preprint arXiv:1812.00891*.

Zhang, Y., Roughan, M., Duffield, N., & Greenberg, A. (2003). Fast accurate computation of large-scale ip traffic matrices from link loads. In *Acm sigmetrics performance evaluation review* (Vol. 31, pp. 206–217).

Zhou, B., He, D., & Sun, Z. (2006). Traffic modeling and prediction using arima/garch model. In *Modeling and simulation tools for emerging telecommunication networks* (pp. 101–121). Springer.

Zhou, C., & Paffenroth, R. C. (2017). Anomaly detection with robust deep autoencoders. In *Proceedings of the 23rd acm sigkdd international conference on knowledge discovery and data mining* (pp. 665–674).

Zhu, L., & Laptev, N. (2017). Deep and confident prediction for time series at uber. In *Data mining workshops (icdmw), ieee international conference on* (pp. 103–110).