symmetry

MDPI

Article

Learning Representations of Network Traffic Using Deep Neural Networks for Network Anomaly Detection: A Perspective towards Oil and Gas IT Infrastructures

Sheraz Naseer [1], Rao Faizan Ali [2], P.D.D Dominic [2,*] and Yasir Saleem [3]

1 Department of Computer Science, University of Management and Technology, Lahore 54728, Pakistan; sheraz.naseer@umt.edu.pk
2 Computer and Information Sciences Department, Universiti Teknologi PETRONAS, Perak 32610, Malaysia; rao_16001107@utp.edu.my
3 Department of Computer Engineering, University of Engineering and Technology, Lahore 54890, Pakistan; yasir@uet.edu.pk
* Correspondence: dhanapal_d@utp.edu.my

Received: 21 September 2020; Accepted: 14 November 2020; Published: 16 November 2020

Abstract: Oil and Gas organizations are dependent on their IT infrastructure, which is a small part of their industrial automation infrastructure, to function effectively. The oil and gas (O&G) organizations industrial automation infrastructure landscape is complex. To perform focused and effective studies, Industrial systems infrastructure is divided into functional levels by The Instrumentation, Systems and Automation Society (ISA) Standard ANSI/ISA-95:2005. This research focuses on the ISA-95:2005 level-4 IT infrastructure to address network anomaly detection problem for ensuring the security and reliability of Oil and Gas resource planning, process planning and operations management. Anomaly detectors try to recognize patterns of anomalous behaviors from network traffic and their performance is heavily dependent on extraction time and quality of network traffic features or representations used to train the detector. Creating efficient representations from large volumes of network traffic to develop anomaly detection models is a time and resource intensive task. In this study we propose, implement and evaluate use of Deep learning to learn effective Network data representations from raw network traffic to develop data driven anomaly detection systems. Proposed methodology provides an automated and cost effective replacement of feature extraction which is otherwise a time and resource intensive task for developing data driven anomaly detectors. The ISCX-2012 dataset is used to represent ISA-95 level-4 network traffic because the O&G network traffic at this level is not much different than normal internet traffic. We trained four representation learning models using popular deep neural network architectures to extract deep representations from ISCX 2012 traffic flows. A total of sixty anomaly detectors were trained by authors using twelve conventional Machine Learning algorithms to compare the performance of aforementioned deep representations with that of a human-engineered handcrafted network data representation. The comparisons were performed using well known model evaluation parameters. Results showed that deep representations are a promising feature in engineering replacement to develop anomaly detection models for IT infrastructure security. In our future research, we intend to investigate the effectiveness of deep representations, extracted using ISA-95:2005 Level 2-3 traffic comprising of SCADA systems, for anomaly detection in critical O&G systems.

Keywords: autoencoders; ANSI/ISO-95; convolutional neural networks; IT infrastructures; data driven security; deep learning; information security; intrusion detection; network anomaly detection; oil and gas IT infrastructure; representation learning

1. Introduction

According to the "BP Statistical Review of World Energy 2019" published by British Petroleum (BP), the Oil and Gas (O&G) sector accounts for 57% of total energy production [1]. Many research contributions have suggested the increased role of information technology (IT) in O&G industry to achieve more productivity [2]. The use of IT in the O&G industry has increased rapidly over the past decade, as shown by the adoption of modern marine digital platforms, intelligent drilling and smart reservoir prediction technologies. Furthermore, O&G organizations have geographically disparate sites, which needs secure communication to enable efficiency and effectiveness in decision making and production processes. Although the O&G industry is rapidly moving towards digitization and automation, the management and governance infrastructure of the O&G industry are still prone to many risks, including internal [3,4] , external, physical, reputational, and cybersecurity risks. A small disruption in IT and operation technology (OT) can cause very large financial and reputational losses to O&G organizations [5]. The world statistics shows that from 2015 to 2018, nearly three-quarters of O&G organizations faced at least one cyber-attack [6] most of which were carried out using Network infrastructure.

The oil and gas (O&G) organizations industrial automation infrastructure landscape is complex [7]. To perform focused and effective studies, Industrial systems infrastructure is divided into functional levels by The Instrumentation, Systems and Automation Society (ISA) Standard ANSI/ISA-95:2005 [8]. ISA-95 was developed to address the topic of how enterprise IT systems/business layer should be integrated with control systems/manufacturing layer. The standard is being used by many industries along with O&G industry where industrial automation is required and its contents are still valid and relevant. Industrial systems such as oil and gas are generally divided into various functional layers depending upon their needs and focus. ISA-95 comprises of five hierarchical levels ranging from the business layer to the actual physical processes layer as shown in Figure 1 adapted from [9]. Figure 1 shows well-defined information processing within six functionalities. The data is generally shared between contiguous layers. The planning layer determines the output goals and, in exchange, receives the total production amount. The scheduling layer produces the comprehensive schedule that determines the development sequence, assignments of equipment and critical timings. The scheduling input requires production progress (necessary for monitoring purposes), equipment availability, and data on capabilities. The execution layer regulates batch or production recipes identifying comprehensive production measures, controls, and other restrictions on production. Finally, the lower layers are directly connected with the physical process. Information systems automation pyramid shown in Figure 1 splits the functions into three main layers ERP, production, and control (MES/CPM) and Control Systems (DCS/PLC/SCADA); each of these components are part of the industrial information systems to handle vertical integration of the production.

This study is only focused on ISA-95:2005 level 4 of the organizational automation process which is responsible for Oil and Gas enterprise resource planning. The level 4 IT infrastructure enables the geographically separate teams to collaborate on tasks such as resource planning, process planning and operations management via distributed communication applications such as emails, chats, messaging, web platforms, etc. In essence, this IT infrastructure is not much different than the normal IT infrastructure and hence prone to the cyber security attacks such as intrusions, phishing, HTTP denial of Service, DDoS, Brute force password guessing, etc. All of these activities have one thing common i.e., they are all deviations from normal traffic and can be treated as anomalies. Anomaly detection describes the task of discovering atypical network traffic with respect to expected behavior patterns of normal network traffic. Anomaly detection is interesting because it enables recognition of patterns which can point to an underlying unknown phenomenon, a fault in system, a vulnerability or an unknown information security threat depending on the infrastructure under consideration.

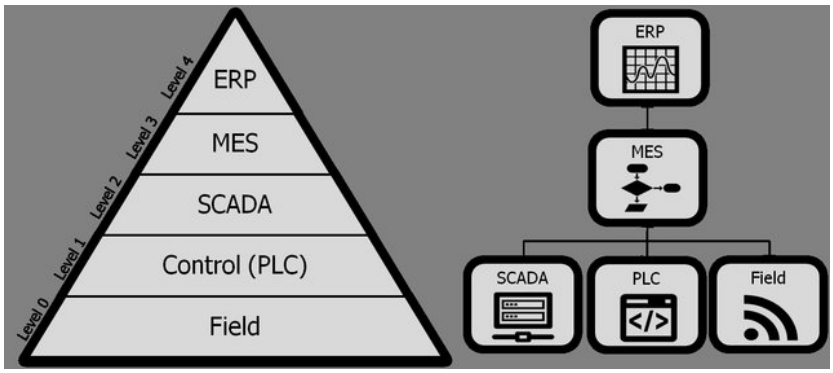

Figure 1. ISA-95 hierarchical view of components [9].

Anomaly detection helps us to understand the unknown underlying phenomenon and enables us to rectify a potentially dangerous situation arising out of anomalies. Network anomaly detection refers to the problem of differentiating anomalous network flows and activities which can adversely impact the security of information systems [10–12]. As IT infrastructures of Oil and Gas organizations get increasingly complex with each passing day, these complexities introduce more and more bugs and vulnerabilities that may be exploited by adversaries to mount attacks. Such attacks are also becoming exceedingly sophisticated which requires contemporary anomaly detection techniques for managing changing threat landscape. Current anomaly detection systems are usually based on a combination of supervised [13], semi-supervised [14] or unsupervised [15] learning methodologies which allow these systems to learn from network traffic, identify anomalous behavior and adapt to changing threat landscape. Given the large number of connections and traffic volume in contemporary networks, which requires monitoring and analysis, it is challenging to identify behavioral patterns over multiple days of data. Such complex, high-dimensional and unstructured network data representation causes the performance of conventional anomaly detection algorithms to become sub-optimal for learning patterns and behaviors from traffic due to enormous computational requirements. A solution is to create Human-engineered Handcrafted representations a.ka. features, which can be used by traditional Machine Learning (ML) algorithms. In fact, many of the successes in anomaly detection and classification for computer networks are dependent on human-engineered representations. Although human-engineered representations are useful for effective deployment of data driven systems, they too suffer from certain limitations. Fundamental limitations identified by Lecun et al. [16] are as follows:

- Creating Handcrafted representation from large datasets is resource intensive and laborious because it requires low-level sensing, preprocessing and feature extraction. This is especially true for unstructured data,
- Identifying and selecting optimal features from large feature-pool of preprocessed data is time consuming and requires expert domain knowledge,
- Handcrafted representations leads to difficulty in "scaling up" activity recognition to complex high level behaviors (e.g., second-long, minute-long or more).

LeCun et al. proposed deep learning to address aforementioned problems. Deep learning is an emerging subfield of Machine Learning which uses neuron-like mathematical structures for learning tasks [17]. Using Deep Learning, contemporary data scientists have made great strides in developing solutions for problems involving computer vision, speech processing, and natural language processing and online-advertisements. At its core, almost all deep learning models use multi-layered neural networks, which transform the inputs layer by layer until a label is generated by last layer of the neural network. Deep Neural Networks (DNNs) offer a versatile method for automatically discovering multiple levels of representations within data. This is due to the fact that each layer of deep neural network receives input from upper layer and transforms it into some representation which is used by subsequent layers. Layers of DNNs non-linearly transform their input, creating more abstract,

task-specific representations in hierarchical manner which are insensitive to unimportant variations, but sensitive to important features [16]. With sufficient training of neural network on input/output pairs of traffic data, the output of last fully connected layer provides an optimal low-dimensional representation of input record which is used by conventional classifier-like logistic regression or softmax to predict the label of input.

Application of Deep learning for solving information security problems is still a new area of research. Our previous works [18,19] discussed the use of DNNs as classifiers for network anomaly detection but important aforementioned research gaps identified by Lecun et al. [16] still remained open in context of network anomaly detection research. The first research gap is limited capability of existing anomaly detection models to use unstructured and high-dimensional network traffic data efficiently [16]. Due to this limitation, performance of contemporary anomaly detection models is dependent on quality of features, generated from raw traffic input. The better the quality of features used by underlying ML algorithm to train model, the better the performance of anomaly detection model. The second research gap stems from the dependence of current anomaly detectors on quality of features. Current representation learning and feature extraction techniques for network anomaly detection [20,21] works in isolation from learning subsystem of model. This means there is no automated feedback mechanism from learning subsystem of a model to feature extraction subsystem for improving the quality of learned data representation. Above-mentioned facts identify pressing need to find alternative automated methods for learning efficient and effective network data representations from complex and very high-dimensional network flow datasets which are comparable in performance to human-engineered (handcrafted) features but require minimum cost in terms of time and human expertise and can be readily used by traditional machine learning algorithms such as SVM, nearest neighbor and decision-tree. Such network data representations will be greatly helpful to implement and deploy better anomaly detection systems to secure ISA-95 level 4 IT infrastructure of oil and gas sector.

DNNs are inherently able to address aforementioned gaps. Sufficiently deep neural networks can use high volumes of raw, unstructured and high dimensional network data directly to learn and classify the network traffic records and are not limited by requirement of low-dimensional handcrafted network data representation. Additionally, since Deep learning models provides a generalized mechanism to automatically discover effective representations without domain knowledge and human intervention they are an ideal candidate for deriving network data representations for developing data driven intrusion/anomaly detection systems. The reader may ask why network traffic payload is included in representation learning process? Packet headers are used for developing handcrafted and statistical features [10,22]. This process requires human intervention and domain knowledge. The payload of a network packet is unstructured data which can be used to learn myriad of features to differentiate between normal and anomalous network traffic. Deep learning is especially useful for learning features from unstructured input and empirical results of this article indicate that deep features learned from payload data can effectively replace human engineered features. Deep feature learning from network payloads does not require human intervention and domain knowledge and thanks to inherent structure of DNNs it can be efficiently implemented using GPUs.

The primary contribution of this study is providing empirical evidence that network data representations learned from raw high dimensional network traffic flows using Deep learning are comparable or better than handcrafted ones in classification performance while removing the need for expert domain knowledge and automating the time-intensive feature extraction and selection process. This contribution will benefit in assuring the security of ISA-95 level-4 IT infrastructure of O&G sector as well as other industry sectors dependent on secure communication through computer networks. In absence of O&G level 4 network traffic, we used ISCX 2012 to represent level 4 network traffic and conduct our experiments. As mentioned earlier, Level 4 IT infrastructure enables the geographically separate teams of O&G organizations to collaborate on tasks such as resource planning, process planning and operations management via distributed communication applications such as

emails, chats, messaging, file transfers, web platforms, etc. In essence, this IT infrastructure is not much different than the normal IT infrastructure and its traffic can be represented reasonably by network traffic of a normal network traffic dataset.

ISCX 2012 is published by Shiravi et al. [23]. ISCX 2012 was generated using a systematic approach to minimize validity issues in existing datasets. Some of the issues encountered in previous datasets included non-availability of configuration information, lack of internal traffic, absence of payload data in captured network traffic, poor representation of realistic network traffic loads, biasness due to heavy packet anonymization, irregularities due to synthetic data generation approach, dearth of modern attack vectors, and availability of trace-only data. ISCX 2012 addresses the aforementioned dataset issues by using network traffic profiles. Profiles combine diverse sets of network traffic in a manner that unique features covering a portion of evaluation domain are contained within separate sets. Two general classes of profiles employed by ISCX 2012 included α and β profiles. An α-profile describes a network attack scenario in an unambiguous manner. Some of the α-profile scenarios used by included infiltration in network from inside, HTTP denial of service attack, DDoS using an IRC botnet, brute force SSH and different variants of DNS, SMTP and FTP attacks. β-profiles served as the templates for realistic background network traffic. To achieve realistic β-profiles, shiravi et al. [23] analyzed four weeks of network activity associated with the users and servers of Center of Cyber Security, University of New Burnswick for any abnormal or malicious activity. Realistic statistical properties of traffic were determined with respect to application and protocol composition after filtering suspicious activities from captured traffic. β-profiles were subsequently abstracted to minimize the likelihood of malicious traffic. Protocols chosen to create β-profiles included HTTP, SMTP, POP3, IMAP, SSH, and FTP. Other protocols including NetBios, DNS and transport layer protocols became available as an indirect consequence of using the aforementioned protocols. The capturing process of dataset in accordance with profile architecture was executed for seven days. ISCX 2012 contains 2,450,324 network flows in the form of seven packet capture files each corresponding to single day traffic. Due to limitations of computational resources, we used approximately 0.2 million records which form approximately 9% of dataset for our experiments. This change will not affect the research because this research is not aiming to develop an anomaly detector but investigating the creation of automated network traffic representations.

Remaining article is divided in six sections. Section 2 discusses on prominent works relevant to this study. Section 3 presents materials and methods of the research including models developed by authors. Section 4 sheds light on experimental setup. In Section 5, we present evaluation results of models using well-known model evaluation parameters along with discussion. This section is followed by Section 6 which discusses the conclusions drawn from this work. Finally, article is concluded by presenting references relevant to this study.

2. Related Works

Our approach is to employ DNNs as feature extractor for learning useful representations of network flow data. These representations can then be used directly by conventional machine learning algorithms without the need for expensive feature extraction and selection process. Recent works show tremendous success in employing DNNs including CNNs, Autoencoders and RNNs to learn useful representations of large-scale image, speech, video and language datasets [24–26]. As discussed in [16], the success of deep learning was due to profound ability of DNNs to learn representations from unstructured data which can then be effectively employed by machine models to solve real-world problems with unparalleled success.

Although a large research community exists which is focused on addressing information security use cases by machine learning applications [27,28], employing DNNs for network anomaly detection is relatively a new research area. Popular Neural Networks such as Recurrent neural networks (RNNs), Deep belief Networks (DBNs) and Autoencoders (AE) have been used for solving network anomaly detection problem. Our previous work [19] discussed the use of Convolutional neural networks (CNNs)

for developing an anomaly detection system and compared its performance with models trained using conventional ML and state of the art research in anomaly detection. This work was further extended in our other research contribution [18] in which anomaly detection models based on various deep learning structures including multiple autoencoders and LSTM-based RNN were developed and evaluated among each other and with conventional ML-based anomaly detectors. In both above mentioned contributions [18,19], DNNs were studied only as classifiers and not as the representation learning mechanisms. Other notable works using DNNs for information security include [29] in which Gao et al. proposed use of DBNs comprised of energy based reduced Boltzmann machines (RBMs) for an IDS architecture trained on KDDCup99 Dataset. Staudemeyer et al. [30] explored the use of RNNs with LSTMs to elucidate intrusion detection problem on the KDDCup99 dataset. A recent work by Tuor et al. [31] applied deep learning methods for anomaly-based intrusion detection. For network traffic identification, Wang [32] employed a deep network of stacked auto encoders (SAE). Ashfaq et al. [33] implemented an anomaly detection system trained on NSLKDD [13] using a semi-supervised learning approach with Neural Nets using Random weights (NNRw). A recent trend uses audit logging data generated by enterprise-grade security devices to learn representations which can help in detection of various advanced persistent security threats including malware and intrusions. Du et al. [34] proposed "Deeplog" a deep neural network model using LSTM cells [35], to model system logs as a natural language sequence. According to [34] Deeplog modeled sequence of log entries using an LSTM which allowed it to automatically learn a model of log pattern from normal execution and flag deviations from normal system execution as anomalies. On an HDFS log dataset, deeplog achieved F-measure of 96% and accuracy of 99%. Arnaldo et al. [36] proposed a generic framework to learn features from audit logging data using DNNs for command and control and control threat detection. Instead of treating audit-logs as natural language such as [34], auditlogs were preprocessed as aggregated, per entity, regular time series which resulted in regular multivariate time series for each entity-instance. They made use of Feed forward Networks (FFNN), LSTM and Convnets to generate deep representations from auditlog-based multivariate time series for each entity-instance in relational database and used Random-Forest as classification algorithm. The models were applied on ISCX 2014 botnet dataset and results remained inconclusive to verify whether deep features helped to achieve better results. We were unable to find any work which employed DNNs to learn representations from network traffic payloads. This research aims to achieve this by employing DNNs to learn Deep representations from network traffic payload data and compare the effectiveness of resultant network data representations among themselves and with Handcrafted representations for supervised learning.

3. Materials and Methods

In this section, we will provide brief overview of dataset preprocessing and methodology used for learning deep representation. We employed deep neural networks (DNNs) for representation learning. DNN models used in this research include:

- Autoencoders (Denoising, Convolutional)
- RNN with LSTM cells (LSTM)
- Convolutional Neural Networks (CNN)

Additionally, we will also discuss the algorithms and methodology of experiments conducted for this study.

The methodology is shown in Figure 2. The aforementioned methodology can be represented in following different modules:

1. Initial dataset preprocessing
2. Secondary dataset preprocessing
3. DNN design and optimization

4. Deep feature extraction using trained DNN models
5. Extraction of Handcrafted features
6. Conventional ML model training using deep and handcrafted features
7. Evaluation and Comparison

Discussion of methodology is provided in following subsections.

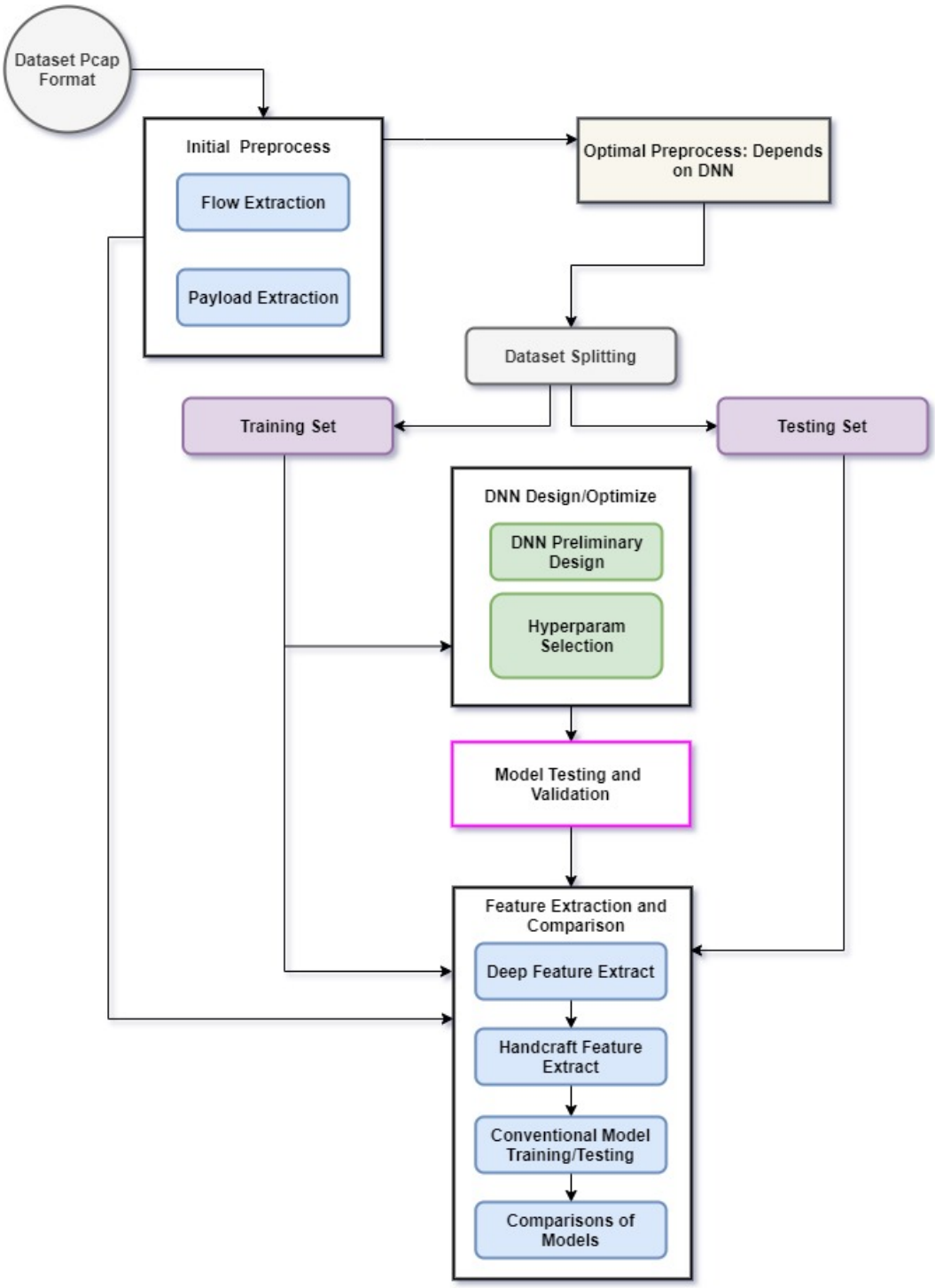

Figure 2. Methodology of Deep Representation learning for Anomaly Detection adopted by current study.

3.1. Dataset Sampling and Preprocessing

3.1.1. Initial Dataset Preprocessing

ISCX 2012 consist of 7 days' capture of network activity (both normal and anomalous) in form of pcap files. Initially, we preprocessed ISCX 2012 from pcap files to network flows so that it can be used by machine learning algorithms. A network flow, ϕ, is defined as an ordered set of all packets $\pi_1, \cdots, \pi_n$ where $\pi_i = \{t_i, S_i, Di, s_i, d_i, p_i, f_i\}$ represents a packet such that:

1. $\forall \pi_i, \pi_j \in \phi, p_i = p_j$
2. $\forall \pi_i, \pi_j \in \phi, (S_i = S_j, D_i = D_j, s_i = s_j, d_i = d_j) \wedge$
   $(S_i = D_j, S_j = D_i, s_i = d_j, d_i = s_j)$
3. $\forall \pi_{i \neq n} \in \phi (t_i \leq t_{i+1}) and (t_{i+1} - t_i \geq \alpha)$

where $t_i, S_i, Di, s_i, d_i, p_i, f_i$ represents time-stamp, source IP address, destination IP address, source port, destination port, protocol and TCP flags respectively. Various network parameters. We used ISCXFlowMeter proposed by Lashkari et al. [22] to generate flows from packet capture files. ISCXFlowMeter is a network traffic flow generator which can generate bidirectional flows from packet capture file where first packet determines both forward (source to destination) and backward (destination to source) directions. TCP flows are usually terminated upon connection teardown while UDP flows are terminated by a flow timeout which was set at 180 s for this study. ISCXFlowMeter can generate statistical time-related features separately in both forward and backward directions.

To label the flows, ISCX 2012 includes a list of malicious IPs and their corresponding attack types. Generally authors of ISCX2010 [23] reported both source and destination IPs in pairs but individual IPs were also reported in some cases. We labeled all flows as anomalous that included one of the individually listed IPs (either as source or destination) and all flows where both the source and destination IPs match a reported pair. All remaining flows were considered normal. Although attack types associated with the malicious IPs were reported by [23], we approached the problem as binary classification problem where flows were considered either normal or anomalous.

ISCX 2012 contained 2,450,324 network flows in the form of 7 packet capture files each corresponding to single day traffic. We used a random sample of approximately 0.2 million flows which constitutes 8.5% of Dataset due to limitation of computational and storage resources. As the purpose of this research is to compare the quality of deep representation with handcrafted representation for anomaly detection rather than the anomaly detection capability of any single algorithm, the statistics of sample chosen for this study are different from actual dataset. Out of 200,083 chosen flow records, 38,813 records were anomalous which constituted approximately 19.3% of chosen sample while 161,270 records were normal which constituted 80.6% of chosen sample. The chosen proportion assured sufficient imbalance in Dataset while containing adequate number of anomalous records for proper representation learning with both handcrafted and deep learning mechanisms.

3.1.2. Secondary Dataset Preprocessing

Each DNN used in this study requires input to be arranged in certain manner before it be used by DNN to learn model. For example, CNNs require input to be arranged in 2D spatial arrays while RNNs require input to be arranged in time-series fashion. Secondary preprocessing module takes care of such special input preprocessing needs for DNNs.

3.2. Deep Neural Networks for Learning Representations

Deep Neural Networks (DNNs) are usually employed as classifiers with last layer being a softmax or regression classifier. During training of DNNs, we can think of all layers of DNN, save last one, learning a representation which is then provided to last layer for predictions. Asserting supervised criterion while training naturally leads to a distinct representation at each subsequent hidden layer especially near the last hidden layer. The penultimate layer generate representation that makes the

classification task easier. For example, classes in data that were not linearly separable in the input space may become linearly separable in the representation fed by last hidden layer to output layer. In principle, the output layer could be another kind of model, such as a nearest neighbor classifier which consumes the representations learned by last hidden (penultimate) layer of deep neural network to perform classification [17].

Additionally, supervised training of DNNs does not impose any condition (except where imposed explicitly by regularization) on the learned intermediate features. Other representation learning algorithms such as PCA [37], t-SNE [38] or manifold learning are often explicitly designed to shape the representation in some particular way. This fact makes DNNs a perfect candidate for automated feature extractors. We used four DNNs for learning representations from aggregated application payload of each flow. These neural networks include Convolutional Neural Networks (CNN), Convolutional Auteoncoder (ConvAE), Denoising Autoencoder (DenoiseAE) and RNN with LSTM cells (LSTM).

3.2.1. Convolutional Neural Network

Convolutional Neural Network (CNN) [16] is a multi-layer architectures specifically designed to handle high dimensional or unstructured data with some spatial semantics. Examples of unstructured data with spatial semantics include images, video, sound signals in speech or character sequence in text [39].

Proposed Deep CNN-based feature extraction model uses an *input* layer, 4 pairs of *conv-subsample* layers, 2 *dropout* layers and two fully connected *FC* layers including *feature extraction* layer and *output* layer. Structure of proposed Deep CNN-based feature extraction model is shown in Figure 3.

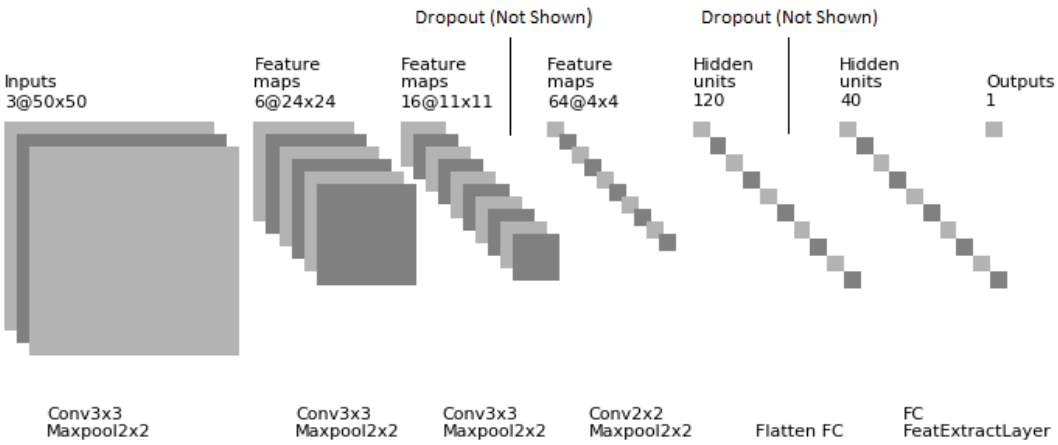

Figure 3. Architecture of Implemented CNN for Representation Learning.

To increase the generalizability of model, Dropout Layer suggested by Srivastav et al. [40] was used. The input layer receives each Network Flow training Dataset I_{train} record (originally having 7500 features) in the form of $50 \times 50 \times 3$ image which is achieved by arranging 7500 features in form of $50 \times 50 \times 3$ 3D array. Arranging input features of each I_{train} record as 3D array helps identifying localized features which are repeated in input record. The latent representations generated by Deep CNN are more tuned to transitive relationships existing in the input and help in learning more abstract relationships which are ignored by learning algorithms not taking into account the spatial structure of input. Architecture of implemented CNN is shown in Table 1.

Table 1. Architecture of implemented Convolutional Neural Network.

Layer Name	Output Shape	No. of Trainable Parameters
R0: Reshape layer (7500 to $50 \times 50 \times 3$ conversion)	(Batch-size,50,50,3)	0
C1: Conv layer with 3×3 kernels and 6 feature maps	(Batch-size,48,48,6)	$((3 \times 3 \times 3) + 1) \times 6 = 168$
S2: Subsample with 2×2 non-overlapping kernel	(Batch-size,24,24,6)	0
C3: Conv layer with 3×3 kernels and 16 feature maps	(Batch-size,22,22,16)	$(((3 \times 3) \times 6) + 1) \times 16 = 880$
S4: Subsample with 2×2 non-overlapping kernel	(Batch-size,11,11,16)	0
C5: Conv layer with 3×3 kernels and 64 feature maps	(Batch-size,9,9,64)	$(((3 \times 3) \times 16) + 1) \times 64 = 9280$
S6: Subsample with 2×2 non-overlapping kernel	(Batch-size,4,4,64)	0
D7: Dropout layer with 0.25 drop probability		Dropout layer
C8: Conv layer with 2×2 kernels and 120 feature maps	(Batch-size,3,3,120)	$((2 \times 2) \times 64) + 1) \times 120 = 30,840$
S9: Subsample with 2×2 non-overlapping kernel	(Batch-size,1,1,120)	0
Model Flattening	(Batch-size,120)	
D11: Dropout layer with 0.5 drop probability	Dropout layer	
FC12: Feature Extraction: Fully connected layer	(Batch-size,120)	$(120 + 1) \times 40 = 4840$
Output	(Batch-size,1)	$(40 + 1) \times 1 = 41$

The entry of vector y_n is zero if corresponding image representation of traffic record belongs to normal traffic and 1 otherwise. Selected non-linearities for CNN model training include *relu* activation for conv layers and *tanh* activation for feature extraction layer. Other hyper-parameters are comprised of *Adam* optimizer [41] with batch size of 256 and binary-crossentropy cost. Proposed CNN was trained for 30 epochs with above-mentioned parameters before it was used for feature extraction according to algorithm described in Section 3.2.6.

3.2.2. Convolutional Autoencoder

Convolutional Autoencoders (ConvAE) were proposed by Masci et al. [42]. ConvAEs are good at discovering initializations for CNNs which avoid distinct local minima of greatly non-convex cost functions used in various deep learning problems. Like CNNs, the weights of ConvAEs are shared among different locations of the input preserving spatial locality. Architecture of implemented ConvAE is shown in Table 2. The encoder part of ConvAE consist of 4 *Conv* layers, 3 subsampling layers and *bottleneck* layer which serves as feature extraction layer. Decoder part of *ConvAE* is responsible for recreating the input from the features learned at bottleneck layer. It is comprised of a *reshape* layer, a fully connected layer, 2 pairs of conv-subsampling layers and 2*Deconv* layers. *Deconv* layers use transpose-convolution to upsample the spatial resolution and perform a convolution. Although it is not a mathematical inverse of convolution, but for Encoder-Decoder architectures, this operation can combine the upscaling of an image with a convolution, instead of doing these two processes separately. Hyper-parameters for ConvAE include *relu* non-linearity, mean-squared-error (MSE) objective and *Adadelta* optimizer. *Adadelta* [43] dynamically adapts over time using only first order information and produces minimal computational overhead without requiring manual tuning of learning rate and shows robustness to noisy gradient information, various data modalities, different model architectures and selection of hyper-parameters. *Adadelta* is represented by following equation:

$$\Delta x_t = \frac{-RMS[\Delta x]_{t-1}}{RMS[g]_t}$$

where Δx_t is parameter update at time t, and $RMS[g]_t$ is exponentially decaying average of RMS at t. ConvAE was trained for 30 epochs with batch-size of 256 using aforementioned hyper-parameters before it was used to extract deep features for ISCX 2012 records using proposed feature extraction algorithm.

Table 2. Architecture of implemented Convolutional Autoencoder.

Layer Name	Output Shape	No. of Trainable Parameters
Input with Reshape	(Batch-size,50,50,3)	0
C1: Conv layer with 3×3 kernels and 8 feature maps	(Batch-size,48,48,8)	$((3 \times 3 \times 3) + 1) \times 8 = 224$
S2: Subsample with 2×2 non-overlapping kernel	(Batch-size,24,24,8)	0
C3: Conv layer with 3×3 kernels and 16 feature maps	(Batch-size,24,24,16)	$((3 \times 3 \times 8) + 1) \times 16 = 1168$
S4: Subsample with 2×2 non-overlapping kernel	(Batch-size,12,12,16)	0
C5: Conv layer with 3×3 kernels and 16 feature maps	(Batch-size,12,12,16)	$((3 \times 3 \times 16) + 1) \times 16 = 2320$
S6A: Subsample with 2×2 non-overlapping kernel	(Batch-size,6,6,16)	0
C7: Conv layer with 3×3 kernels and 16 feature maps	(Batch-size,6,6,16)	$((3 \times 3 \times 16) + 1) \times 16 = 2320$
S6B: Subsample with 2×2 non-overlapping kernel	(Batch-size,3,3,16)	0
Model Flattening	(Batch-size,144)	
FC8: **Bottleneck**: Fully connected layer	(Batch-size,40)	$(144 + 1) \times 40 = 5800$
FC9: Fully connected layer	(Batch-size,144)	$(40 + 1) \times 144 = 5904$
R10: Reshape layer (144 to 3x3x16 conversion)	(Batch-size,3,3,16)	0
C11: Conv layer with 3×3 kernels and 16 feature maps	(Batch-size,3,3,16)	$((3 \times 3 \times 16) + 1) \times 16 = 2320$
U12: Upsample with 2×2 kernels non-overlapping kernel	(Batch-size,6,6,16)	
C13: Convolution layer with 3×3 kernels and 16 feature maps	(Batch-size,6,6,16)	$((3 \times 3 \times 16) + 1) \times 16 = 2320$
U14: Upsample with 2×2 kernels non-overlapping kernel	(Batch-size,12,12,16)	
CT15: Transposed-Conv layer with 3×3 kernels, (2,2) Strides and 16 feature maps	(Batch-size,25,25,16)	$((3 \times 3 \times 16) + 1) \times 16 = 2320$
CT16: Transposed-Conv layer with 3×3 kernels, (2,2) and 3 feature maps	(Batch-size,50,50,3)	$((3 \times 3 \times 16) + 1) \times 3 = 435$

3.2.3. Denoising Autoencoder

Proposed by Vincent et al. [44], DenoiseAE receives as input a record corrupted with a noise source and is trained to predict at its output, the original, uncorrupted version of record. DenoiseAE minimizes loss function $L(x, g(f(\bar{x})))$. To generate the original record x at its input, DenoiseAE must remove this corruption while recreating their input at output. To achieve this, Denoise AE captures most significant features of training data by performing SGD on following expectation:

$$-\mathbb{E}_{x \sim \hat{p}_{data}} \mathbb{E}_{x \sim C(\bar{x}|x)} log p_{decoder}(x|h = f(\bar{x}))$$

Adding noise in input x also helps DenoiseAE to reduce over-fitting in learned models and increase the generalizability thereof.

Architecture of implemented DenoiseAE is shown in Table 3. First layer of Encoder part is a Guassian Noise layer with 25% impurity which serves as Corrupting function $C(\bar{x}|x)$ to impart corruption in input data points. Gaussian noise is followed by a conv-subsampling layer which was introduced to reduce the dimensionality of input space. Last layer of Encoder part is fully connected layer FC3, which serves as the *bottleneck* layer of our DenoiseAE. Decoder part of this DNN consist of a fully connected layer, a *reshape* layer and two *Deconv* layers to achieve simultaneous upsampling and representation learning by transposed-convolution. Hyper-parameters used to train *DenoiseAE* are same as that of *ConvAE* except the kernel-initializer function which drew sample from *RandomNormal* distribution for kernel initialization. This DNN was trained for 30 epochs and then deep features were extracted according to proposed algorithm described in Section 3.2.6.

Table 3. Architecture of implemented Denoising Autoencoder.

Layer Name	Output Shape	No. of Trainable Parameters
Input with Reshape	(Batch-size,50,50,3)	0
N0: Gaussian Noise layer with 25% Impurity		
C1: Conv layer with 5×5 kernels and 6 feature maps	(Batch-size,46,46,6)	$((5 \times 5 \times 3) + 1) \times 6 = 456$
S2: Subsample with 4×4 non-overlapping kernel	(Batch-size,11,11,6)	0
Model Flattening	(Batch-size,726)	
FC3: **Bottleneck**: Fully connected layer	(Batch-size,40)	$(726 + 1) \times 40 = 29,080$
FC4: Fully connected layer	(Batch-size,726)	$(40 + 1) \times 726 = 29766$
R5: Reshape layer (726 to $11 \times 11 \times 6$ conversion)	(Batch-size,11,11,6)	
CT6: Transposed-Conv layer with 5×5 kernels, (2,2) Strides and 6 feature maps	(Batch-size,25,25,6)	$((5 \times 5 \times 6) + 1) \times 6 = 906$
CT7: Transposed-Conv layer with 5×5 kernels, (2,2) and 3 feature maps	(Batch-size,50,50,3)	$((5 \times 5 \times 6) + 1) \times 3 = 453$

3.2.4. Recurrent NeuralNet with LSTM Cells

A Recurrent NeuralNet (RNN) is a connectivity pattern that performs computations on a sequence of vectors $x_1, \ldots, x_n$ using a recurrence formula of the form $h_t = f_\theta(h_{t-1}, x_t)$, allowing us to process sequences with arbitrary lengths where f is an activation function and θ is a parameter and both are used at every timestamp. Bengio et al. [45] presented theoretical and experimental evidence showing that gradient descent of an error criterion is inadequate to train RNNs for tasks involving long-term dependencies because such a model would either not be robust to input noise or would not be able to efficiently deliver long term context. Long Short term Memory (LSTM) is a special RNN architecture which addresses the problems in vanilla RNNs by allowing inputs x_t and hidden states h_{t-1}in a complex manner by combining multiplicative and additive interactions. The function of LSTM neuron can be represented by the following equations:

$$\begin{aligned}
\bar{H}^{<t>} &= tanh(W_c[\Gamma_r * a^{<t>}, X^t] + b_c \\
\Gamma_o &= \sigma(W_o[a^{t-1}, X^t] + b_o) \\
\Gamma_u &= \sigma(W_u[a^{t-1}, X^t] + b_u) \\
\Gamma_f &= \sigma(W_f[a^{t-1}, X^t] + b_f) \\
H^{<t>} &= \Gamma_u * \bar{H}^{<t>} + \Gamma_f * H^{t-1} \\
a^{<t>} &= \Gamma_u * tanh(H^{<t>})
\end{aligned}$$

In above equations, $W_u, W_o and W_f$ are respective weights, $b_u, b_o and b_f$ are respective bias terms, X^t represents input, H^t represents contents of memory cell, σ represents sigmoid function, and $a^{<t>}$ represents activations at time step t. Γ_x represent x gate where $x \in update, output, forget$.

Architecture of implemented LSTM-based RNN is presented in Table 4. This is by far our most expensive DNN with 634,577 trainable parameters as compared to that of other implemented DNNs, all of which required less than 50 k trainable parameters with the exception of DenoiseAE with 61 k trainable parameters. Implemented LSTM DNN consisted of a reshape layer, an LSTM layer with 128 LSTM cell units, a dropout layer to introduce resistance against over-fitting and 03 fully connected layers including feature extraction layer and output layer. All Fully connected layers except output layer used *tanh* activation function while output layer used *sigmoid* activation. Model was trained for 30 epochs with binary-crossentropy loss, *Adadelta* optimizer [43] and validation split of 25%.

Table 4. Architecture of implemented LSTM-based RNN.

Layer Name	Output Shape	No. of Trainable Parameters
Input with Reshape	(Batch-size,25,300)	0
L1: LSTM layer with 128 units	(Batch-size,25,128)	219,648
Model Flattening	(Batch-size,3200)	
D2: Dropout with 50% drop probability		Dropout
FC3: Fully connected layer	(Batch-size,128)	$(3200 + 1) \times 128$ = 409,728
FC4: Feature Extraction: Fully connected layer	(Batch-size,40)	$(128 + 1) \times 40 = 5160$
Output	(Batch-size,1)	$(40 + 1) \times 1 = 41$

3.2.5. Hyperparameter Optimization

In process of deep learning, hyper parameters include such "higher-level" properties of the model which cannot be learned from training set but have profound impact on learning capacity and accuracy of the model. For a Learning Algorithm, the process of determining "good" Hyperparameter values for a learning algorithm is called hyperparameter selection/optimization/tuning. Commonly used strategies for hyper-parameter optimization are grid-search and manual search, but their effectiveness is limited becuase of wasteful usage of computational budget [18,46]. Bergstra and Bengio [46] proposed Randomized Search Strategy for efficiently choosing trial-sets of hyperparameters from a

configuration space. Trial-sets are sampled independently and randomly from configuration space and represented by a random variable modeled by standard normal distribution. We chose acccuracy of model on validation set to evaluate performance of different trial-sets. According to [46] validation scores are typically relatively close and a few tens of hypothetical draws are sufficient to achieve optimized trial-set of hyperparameters.

For the purpose of current study, We limited the hyperparameter draws to 32 for ensuring more than 80% chance of achieving the optimial trial-set. The configuration space included different values of well-known hyperparameters including 'convlution kernels', Optimizers, Objective functions, Non-linearities for Convolution (Conv) layers and fully connected (FC) layers, kernel initializers, learning-rate and other DNN specific parameters. We assumed 5% interval around the true optimum in configuration space. In this case, each random draw would be independent and identically distributed and would have 5% probability to land in the optimum interval. If we draw 'n' samples independently then the probability that all 'n' of them miss the desired interval is $(1-0.05)^n$. The probability of atleast one of them landing in desired optimum would be $1-(1-0.05)^n$. As we limited the probability of optimum configuration to 80% probability due to limitation of computational resources, putting $n = 32$ in following inequality enable us to satisfy the desired condition:

$$1-(1-0.05)^n \geq 0.8$$

Major benefit of Random-search strategy is its efficient use of limited computational budget for finding better model parameters by performing effective search over otherwise large configuration space [46]. Selected important hyperparmeters of all discussed DNN models in this study are presented in Table 5. Architecture related hyper-parameters are presented separately in tables discussing architecture of specific DNNs.

Table 5. Important Hyperparameters selected using RandomizedSearch [46] for employed DNNs.

DNN	Optimizer	Objective	Neuron	Misc	Bias	Learn-Rate	Valid-Acc
CNN	Adam	Binary crossentropy	Conv:relu, FC:tanh	Init:He-uniform	Yes	0.01	0.9937
LSTM	Adadelta	Binary crossentropy	FC:tanh	lstm-cells:128	NA	0.05	0.9920
ConvAE	adadelta	MSE	Conv:relu	Init:glorot-uniform	Yes	0.01	0.6991
DenoiseAE	adadelta	MSE	Conv:relu, FC:relu	Noise: 0.25	Yes	0.001	0.5860

3.2.6. Deep Feature Extraction Using Trained DNN Models

Once a trained deep model is available, it is used to extract deep features from the network flow. We now present the general steps of feature extraction from each flow ϕ_i.

1. Extract first 7500 bytes I_i from each aggregated application payload P_i for 200,083 chosen flow records of ISCX 2012.
2. Reshape each input I_i according to requirement of DNN to be employed for model training.
3. Divide input I_i in training set of 160,000 records and validation set of 40,083 records.
4. Train a model $M_i$using each deep model on dataset I using backpropagation.
5. Propagate dataset I through each M_i and retrieve the outputs D_{M_i} generated at the penultimate layer of the Model M_i. The output D_{M_i} will be a matrix of shape $n * q$, where n is the number of samples and q is the number of learned features. For current study, q is fixed at 40 and n is 200,083 which means each D_{M_i} is a matrix of 200,083 $\times$ 40 stored in a numpy array.
6. For each D_{M_i} do the following:
   (a) Divide DM_iinto trainset and testset.
   (b) Train all 12 conventional ML algorithms on trainset and evaluate on test set.
   (c) Calculate model evaluation metrics on results of evaluation.

This methodology was used to extract deep representation from all four DNNs used in this study.

3.3. Handcrafted Representation

We used ISCXFlowmeter [22] to extract Handcrafted features from ISCX 2012. Extracted features by ISCXFlowmeter include 25 quantitative features and 9 symbolic features for every network flow as shown in Table 6.

Table 6. List of Handcrafted features extracted from ISCX 2012 [22].

Quantitative Features	Symbolic Features
Duration	application Name
Bytes per second	direction
Min/ Max/ Avg/ Std packet inter-arrival times	dstTCPflagDescription
Packets per second	srcTCPflagDescription
Min/ Max/ Avg/ Std idle time	dstIP
Min/ Max/ Avg/ Std active time	SrcIP
Min/ Max/ Avg/ Std inter-arrival times of received packets	StartDateTime
Min/ Max/ Avg/ Std inter arrival times of sent packets	transport protocol
srcPort	EndDateTime
dstPort	

As ML algorithms are unable to use symbolic features, these features need to be converted to numeric form. Different techniques [47–50] have been proposed by research community for encoding symbolic features to quantitative features. We studied effect of different encoding schemes on performance of Handcrafted features for ISCX 2012 dataset using a Random-Forest classifier. Random-Forest algorithm was chosen due to its time efficiency. Effect of different encoding schemes on dataset dimensionality, training time and accuracy of model on use of 09 encoded symbolic features to their quantitative counterparts is shown in Table 7.

Table 7. Effect of Different encoders on ISCX 2012 Handcrafted Symbolic Features.

Encoding Scheme	Dimensionality	Average Score	Score StDev	Training Time (Seconds)
BinaryEncoder	61	0.9900	0.001789	45.247959
BackwardDifference	11,136	0.9096	0.066437	90.317814
HelmertEncoder	11,136	0.8928	0.064478	78.638483
HashingEncoder	8	0.9304	0.003611	8.252981
OrdinalEncoder	9	0.9938	0.007909	44.707021
OneHotEncoder	11,145	0.9968	0.001600	206.595544
SumEncoder	11,136	0.9962	0.001939	79.257670
BaseNEncoder	61	0.9910	0.001414	65.117447
LeaveOneOutEncoder	9	0.9994	0.000800	44.738180

In Table 7, dimensionality represents number of additional features inserted by encoding algorithm in each dataset record during encoding of nine symbolic features. Average Training scores discuss training accuracy of selected Random-Forest classifier on employing any particular encoding scheme. Based on the performance shown by different encoders, we chose LeaveOneOutEncoding proposed by [50]. To normalize otherwise different numerical ranges of different feature vectors, we applied median and interquartile range (IQR) scaling. It is relevant to mention here that IQR preprocessing was only applied on Handcrafted features and it was not used for any of the Deep representations mentioned in the previous sections to verify the fully automatic feature generation hypotheses.

As handcrafted features of ISCX2012 dataset contained different numerical ranges, z-score scaling was applied for conversion of these features into standardize representation to ensure better learning by conventional ML algorithms.

3.4. Conventional ML Model Training Using Network Data Representations

Both Deep representations and handcrafted features were consumed by conventional machine learning classification algorithms to evaluate suitability of representation in anomaly detection and classification. Conventional ML Algorithms were selected to represent major supervised learning techniques including tree based, ensemble, nearest neighbor, support vector, neural networks and algebraic algorithms. Employed Conventional classification algorithms included Extreme learning Machine [51] with MLP and RBF hidden layer and its variation with Generalized RBF Neural Network [52], SVM, k-NN, Decision-Tree, Random-Forest, Extremely Randomized Trees (Extra Tree), Quadratic Discriminant Analysis (QDA), Multilevel perceptrons (MLP) and Boosting algorithms including Adaboost and Gradient boost. All ML algorithms mentioned in Table 8 are used in accordance with Algorithm 1 to train anomaly detection models. Important hyper-parameters for these conventional classification algorithms are presented in Table 8. Since all representations were subjected to same conventional ML algorithms with parameter presented in Table 8 for developing anomaly detector, no hyperparameter optimization was introduced for conventional ML algorithms to make the comparisons between deep and handcrafted representation fair.

In total, the experiments developed whooping 60 Anomaly Detection models where each representation R_i was used to develop 12 anomaly detection models using each ML algorithm mentioned in Table 8. For each ISCX2012 representation, the dataset was split into training set and test set. We tested each representation using 12 conventional ML algorithms. For each ML algorithms, an anomaly detection model was trained using training set and tested using test set. Relevant performance metrics were calculated for all 12 models for each data representation and comparisons were made using well-known classifier quality metrics.

Table 8. Important parameters for Binary classifiers used for comparison.

Sr. No.	Classification Algorithm	Hyper-Parameters
1	Decision-Tree	Max-Depth = 10, Split Quality Measure = "entropy", Max features considered for each best split = 40
2	Random-Forest	Max-Depth = 5, No. of Estimators = 10, Split Quality Measure = "gini", Max features considered for each best split = $\sqrt{40}$
3	QDA	Priors = None, Regularization Parameter = 0.01, Rank Estimation Threshold = 0.0001
4	Multilevel Perceptron	Hidden layer Units = 100, Activation = *relu*, Solver = SGD, *L*2 Penalty = 0.0001, Learning rate = adaptive, epochs = 200
5	Nearest Neighbor	Neighbors = 15, Algorithm = *Ball-Tree*, Leaf-size = 30, Distance-Metric = *Euclidean*
6	SVM	Kernel = RBF, Gamma = 1/40, epochs = 2500, Length scale = 1, Length scale bounds = $(1 \times 10^{-5}, 1 \times 10^{5})$
7	Extremely Randomized Trees	Max-Depth = Not Set, No. of Estimators = 30, Split Quality Measure = "gini", Max features considered for each best split = $\sqrt{40}$, Out-of-bag Sampling = enabled, Bootstrap-Sampling = enabled
8	Adaboost	Base-estimator = J48, No. of Estimators = 50
9	Gradient Boost	No. of Estimators = 100, max-leaf-nodes = 20, max-depth = 90, random-state = 24, min-samples-split = 5, learning-rate = 0.01
10	ELM MLP	Hidden Layer Units = 256, activation = *gaussian*, Hidden-Layer = MLP, epochs = 15
11	ELM RBF	Hidden Layer Units = 256, activation = *multiquadric*, Hidden Layer = *RBF*, epochs = 15
12	ELM Generalized	Hidden Layer Units = 256, activation = *grbf* [52], epochs = 15

Algorithm 1: Model Generation from Conventional ML Algorithms.

```
Input: ISCX Deep and Handcrafted Representations
for each ISCX Representation R_i (All 05 Representations) do
    Divide the input R_i into Training Set X^i_train and Testset X^i_test
    for each ML algorithms A_j presented in Table 8 do
        Train Anomaly Detection Model M_ij using Training Set X^i_train
        Test Anomaly Detection Model M_ij using Testset X^i_test
        Calculate relevant Quality metrics
    end
end
Output: Classification quality scores for each Anomaly Detection Model M_ij
```

Evaluation and Comparison of Conventional Anomaly Detectors

All learned models for each representation were evaluated using standard model evaluation metrics including Accuracy, ROC curve, AuC, precision-recall curve, mAP and F1-measure. We performed both representation-wise comparisons and algorithm-wise comparisons for all five ISCX2012 data representations (04 deep representations and one handcrafted representation) to ascertain the effectiveness of representation for learning models to make reliable predictions.

4. Experimental Setup

Following Hardware setup was used for experiments in this study:

- CPU: Quad Core Intel Xeon E-1650
- RAM: 16 GB
- GPU: nVidia GTX 1070

Software toolchain consisted of jupyter IDE using Keras 2.0.3 on tensorflow [53] backend and nVidia cuda 8.0. Training and testing data is manipulated using numpy. We used Sci-kit learn implementations of conventional ML for developing anomaly detection models.

5. Results and Discussion

In this section we present the performance of deep and handcrafted representations for anomaly classification using conventional machine learning algorithms. The evaluations are made on the assumption that if a model trained on a particular data representation shows better performance, then the representation used to train that model is better and vice versa. We used well-known evaluation parameters to evaluate and compare performance of anomaly detection models developed using all deep and handcrafted representations. These well-known evalaution parameters include Receiver Operating Characteristic (ROC) Curve, Area under Curve (AuC), Precision-Recall Curve, Accuracy, F1-measure and mean Average Precision (mAP). Aforementioned evaluation parameters are calculated on the basis of confusion matrix which presents four measures as follows:

- True Positive (TP): Model prediction is accepted as TP, if an anomaly is predicted as anomaly,
- False Positive (FP): Model prediction is accepted as FP, if a normal record is predicted as anomaly,
- True Negative (TN): Model prediction is accepted as TN, if a normal record is predicted as normal,
- False Negative (FN): Model prediction is accepted as FN, if an anomaly is predicted as normal.

In the following subsection, we give a brief introduction of relevant evaluation parameters and present the results. Results are presented with two different perspectives. Algorithm-wise performance evaluation combines and compare results of all Models M_{ij} where j represent a particular ML algorithm e.g., $M_{i(kNN)}$ compare results for all representations consumed by kNN algorithm and

$M_{i(MLP)}$ describe results for all representations consumed by MLP algorithm. The second perspective describes Representation-wise performance evaluation. From this perspective, results for all M_{ij} are compared where i represents a particular representation e.g., $M_{(ConvAE,j)}$ combine and compare model results of all ML algorithms consuming deep representations learned by ConvAE. We will use this convention to describe results for each performance evaluation parameter in following subsections.

5.1. Receiver Operating Characteristics (RoC) and Area under RoC

A receiver operating characteristics (ROC) graph is an evaluation parameter for visualizing, organizing and selecting classifiers based on their performance which depict the tradeoff between hit rates (true positive rate) and false alarm rates (false positive rate) of classifier [54]. ROC graphs are two-dimensional graphs in which true positive rate (tpr) is plotted on the $Yaxis$ and false positive rate (fpr) is plotted on the $Xaxis$. An ROC curve depicts relative trade-offs between benefits (true positives) and costs (false positives). Several points in ROC space are conceptually important. The lower left point (0,0) depicts the strategy of never predicting a positive classification i.e., such a model commits no false positive errors but also gains no true positives. In opposition to aforementioned strategy, the strategy of unconditionally issuing positive classifications is represented by the upper right point (1,1). The point (0,1), in ROC space, is of particular importance and represents perfect classification with maximal performance. ROC is able to provide a richer measure of classification performance than scalar measures such as accuracy, error rate or error cost [54].

To compare models using ROC, sometimes the requirement is to reduce ROC performance to a single scalar value representing expected performance. A common method is to calculate the area under the ROC curve which is known as AUC. In addition to reducing ROC scores to a scalear, AUC has an important statistical property. AUC of a classifier is equivalent to the probability that the model under consideration will rank a randomly chosen positive data sample higher than a randomly chosen negative data sample which is equivalent to wilcoxon test of ranks [54].

5.1.1. RoC and AuC Algorithm-Wise Performance

Algorithm-wise performance of Models is shown in Figures 4 and 5 while AuC scores for each Model M_ij are shown in legend section of aforementioned figures. In both figures, dotted line shows Auc score of 0.5 which marks the boundary between acceptable and non-acceptable AuC scores. A classifier of AuC score below 0.5 is considered poor and unsuitable for any practical applications. Comparisons are shown using mean AuC value which, as the name suggests, is measured by calculating mean of AuC scores of all models trained using a particular algorithm.

It is relevant here to mention that any representation which enables at least a single classifier to converge and achieve good generalization ability is a good representation. Mean AuC is a crude measure used only for ease of comparison and does not reflect that a representation is only usable if it achieves better mean AUC scores for all classifiers. Mean AuC just gives an idea of how much a representation is amenable to studied conventional ML algorithms and flexibility of representation to be usable by different algorithms. Additionally, the real benefit of Deep representations is their ability to be learned in fractional amount of time in comparison to handcrafted representations. Moreover, deep representations have the capability to capture important features, without the need for human intervention and domain expertise, using a feedback loop between learning subsystem and feature extraction subsystem to enhance their performance in supervised learning.

From Figures 4 and 5, it can be seen that both Averaging Classification Algorithms namely Random-Forest and ExtraTree came forth as the most robust algorithms for different type of representations and performed extremely well on both Deep and Handcrafted features. Mean AuC scores for both of these algorithms are 0.995 and 0.986 for all five representations. Performance of averaging algorithms was followed by kNN and Decision-Tree with respective mean AuC scores of 0.995 and 0.993 for all representations. Both Boosting algorithms including AdaBoost and GradientBoost also showed great promise for different types of representations. AuC scores for

each Model $M_i j$ are shown in legend section of Figure 4 and 5. Algorithm-wise mean Area under Roc scores are shown in Table 9.

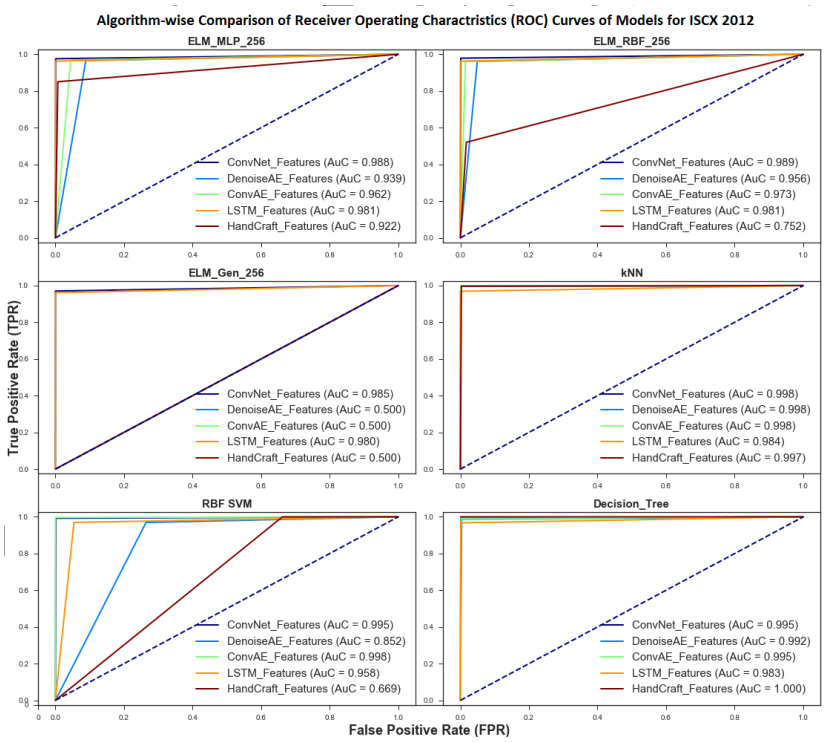

Figure 4. Algorithm-wise RoC curve and Area under Curve for ELM variants, kNN, SVM and Decision Tree Algorithm.

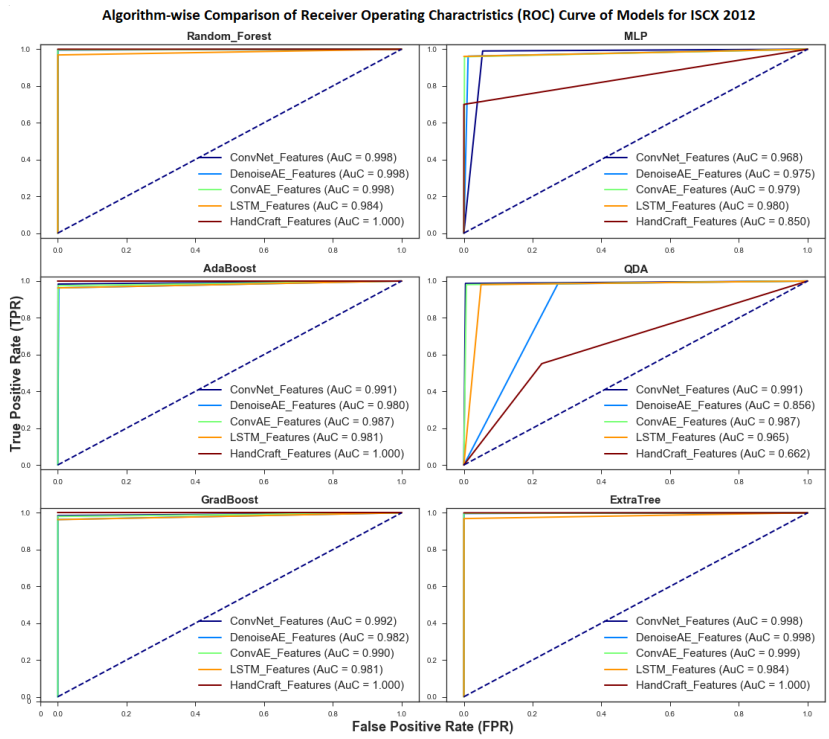

Figure 5. Algorithm-wise RoC curve and Area under Curve for MLP, QDA, Boosting and Averaging classification Algorithms.

Table 9. Algorithm-wise RoC curve and Area under Curve scores for all 12 ML algorithms.

Algorithm-Name	Mean-AuC	Algorithm-Name	Mean-AuC
ELM-MLP	0.9584	Random-Forest	0.9956
ELM-RBF	0.9302	MLP	0.9504
ELM-GRBF	0.693	Adaboost	0.9878
k-NN	0.995	QDA	0.8922
SVM	0.8944	Gradient Boost	0.989
Decision-Tree	0.993	ExtraTree	0.9958

5.1.2. RoC and AuC Representation-Wise Performance

RoC Curves and AuC scores of all network data representations under study for different ML algorithms are shown in Figure 6. Mean AuC scores for this section are measured by calculating mean of AuC scores of all models trained on a particular representation using conventional ML algorithms. Based on RoC Curves, best average performance is shown by CNN representation whose mean AuC score for all algorithm is 0.9925. Runner up with respect to RoC and AuC metric is LSTM representation with mean Auc score of 0.9775. Performance of HandCrafted representation remained on the bottom with AuC score of 0.8625. For Handcrafted features, MELM (Modified Extreme Learning Machine), QDA (Quadratic Discriminant Analysis) and RBF SVM performed poorly which had an adverse impact on mean AuC score of Handcrafted representation. As two of the aforementioned ML algorithms use RBF for classification, this may have a role to play in poor AuC scores. Averaging, boosting and Tree based classifiers performed exceptionally well for all representations including Handcrafted representation. In summary, the RoC and AuC results show that all Deep representations performed comparable to Handcrafted representation while automating the feature learning aspect of anomaly detection problem in fractional cost of time. The timing comparison is shown in Section 5.5.

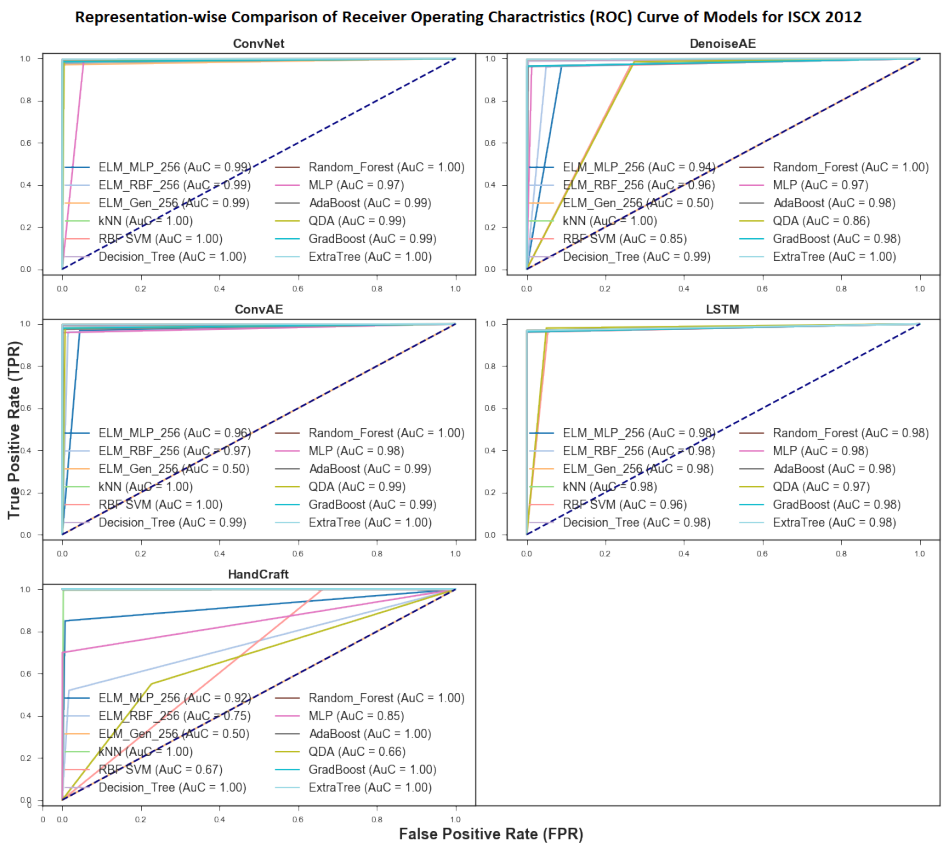

Figure 6. Representation-wise RoC curve and Area under Curve scores for all 12 ML algorithms.

5.2. Precision-Recall Curve and Mean Average Precision

Precision identifies the relevancy of model results, while recall is the ability of a model to find all the relevant cases i.e., finding all anomalies within a dataset. Typically, Precision and Recall show inverse relationship, i.e., as precision increases, recall falls and vice versa. Each model exhibits a trade-off between precision and recall. The trade-off between precision and recall is studied using precision-recall curve. The precision-recall curve (PR-Curve) is computed from the model's predictions by varying the model score threshold. This threshold determines which predictions fall in positive class and which predictions fall in negative class. Changing the threshold affects model-prediction scores for the classes in dataset. The closer a PR-Curve is to the point (1,1) in precision-recall space, the better the classifier is for practical use. PR-Curve is sometimes considered more useful for imbalanced datasets because it does not include negative class samples (TN) in calculation and measure the probability of correct detection of positive samples [54].

5.2.1. PR-Curve and mAP Algorithm-Wise Performance Evaluation

Algorithm-wise PR-Curves for all network anomaly detection models trained on five network data representations and twelve ML algorithms are shown in Figures 7 and 8. ML Algorithms were selected to represent major supervised classification techniques including tree based, ensemble, nearest neighbor, neural networks and algebraic algorithms. Comparisons are shown using mean mAP value which is measured by calculating mean of mAP scores of all models trained using a particular algorithm. Here again, mean mAP is used only for ease of comparison and does not reflect that a representation is only usable if it achieves better mean mAP scores for all classifiers. A higher mean mAP score shows that the representations is amenable to most of the ML algorithms for developing anomaly detectors. Achieved PR-curves and mAP scores corroborate the findings of Roc-Curve and AuC. The best mean mAP score is achieved by models trained using random-forest and Extreme randomized tree algorithms with score of 0.9924. These scores were followed by that of kNN and Decision-tree. The worst scores were shown by models developed using ELM variant proposed by [52]. Mean mAP scores of each ML algorithm for all five representations are shown in Table 10.

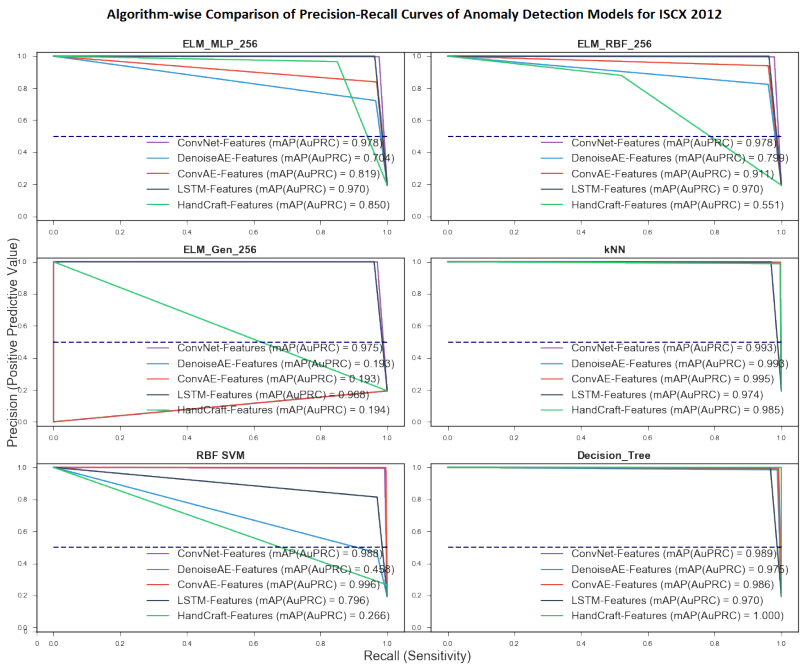

Figure 7. Algorithm-wise Precision-Recall curve and mAP for ELM variants, kNN, SVM and Decision Tree Algorithm.

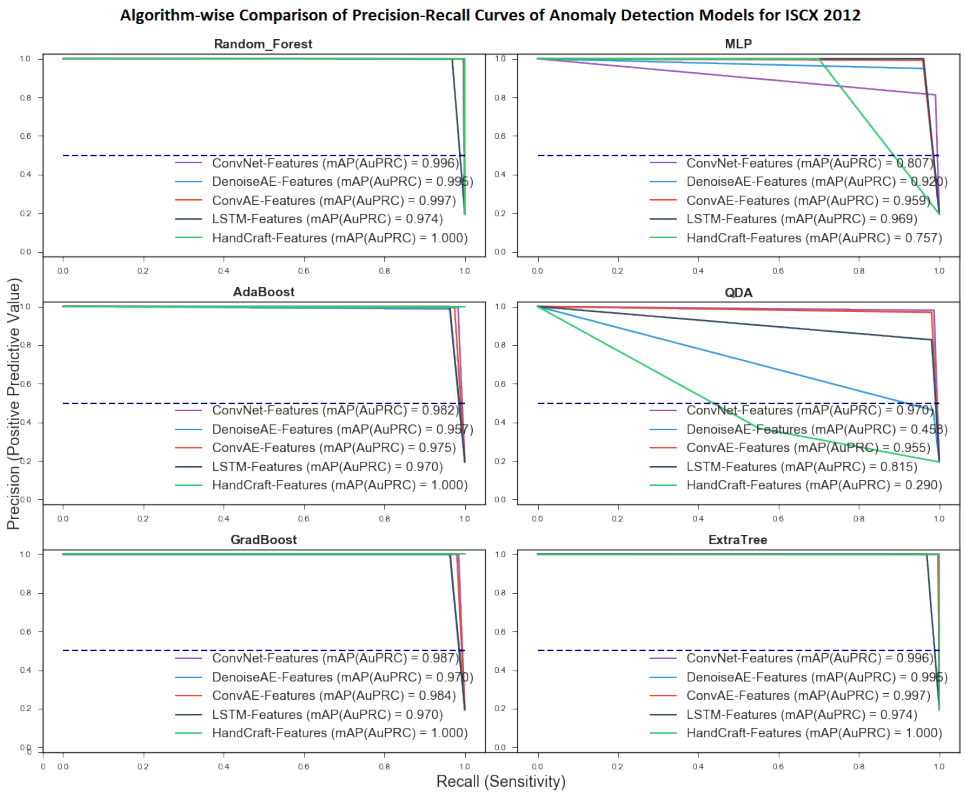

Figure 8. Algorithm-wise RoC curve and Area under Curve for MLP, QDA, Boosting and Averaging classification Algorithms.

Table 10. Algorithm-wise mean Average Precision (mAP) scores for all 12 ML algorithms.

Algorithm-Name	Mean mAP Score	Algorithm-Name	Mean mAP Score
ELM-MLP	0.864	Random-Forest	0.9924
ELM-RBF	0.8414	MLP	0.8824
ELM-GRBF	0.5046	Adaboost	0.9768
k-NN	0.988	QDA	0.6967
SVM	0.7008	Gradient Boost	0.9822
Decision-Tree	0.984	ExtraTree	0.9924

5.2.2. PR-Curve and mAP Representation-Wise Performance Evaluation

Figure 9 shows representation-wise scores of implemented models based on PR-Curve and mAP. Mean mAP scores for this section are measured by calculating mean of mAP scores of all models trained on a particular representation using conventional ML algorithms. The most affable representation turned out to be CNN representation which was amiable to all 12 algorithms as evident from mAP scores shown in Figure 9. ConvAE representation was affable for all algorithms except MELM [52] variant. Of all the representations the worst mean mAP score was depicted by Handcrafted Representation which was due to poor performance of models trained by SVM, ELM, and QDA.

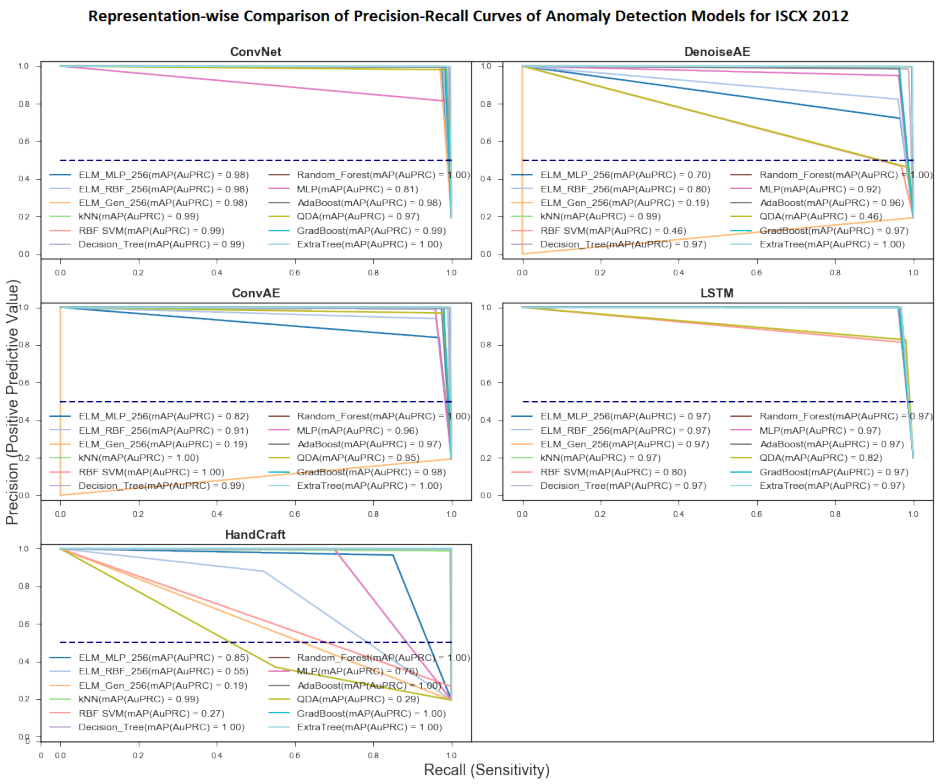

Figure 9. Representation-wise Precision-Recall curve and mean Average Precision scores for all 12 ML algorithms.

5.3. Accuracy

Although Accuracy is not the most effective evaluation parameter for measuring the performance of models trained on imbalanced datasets, we are providing the accuracy score of anomaly detection models to make this study comprehensive. Accuracy of a classifier is measured as the number of correctly classified observations over the total number of observations.

It is relevant to mention that all accuracy scores such as previous evaluation parameters were calculated over testset which is not exposed to ML algorithms during model training time. Figure 10 shows algorithms-wise accuracy scores for all models. As can be seen from Figure 10, kNN, Random-Forest, Adaboost, Gradboost, Decision-Tree and ExtraTree (Extremely Randomized Trees) classifier show comparable results in the range of 99% to 100% accuracy. SVM, QDA and ELM variants did not prove very useful for training anomaly detection models on different types of network data representaitons as depicted by capricious Accuracy scores. Representation-wise accuracy scores for all anomaly detection models are presented in Figure 11. CNN representation emerged as most amenable data representation for all ML algorithms used by current study. All algorithms trained on CNN representation showed remarkable accuracy scores in the range of 99% to 100% except MLP which depicted 95% accuracy score. LSTM representations achieved 2nd rank with slightly less accuracy scores ranging from 95% to 99%. Handcrafted representation showed fluctuating accuracy scores ranging from 73% to 100% except SVM which showed poor score of 47%. These Accuracy scores validated the findings revealed by previously discussed model evaluation parameters including RoC-Curve, AuC, PR-Curve, and mAP.

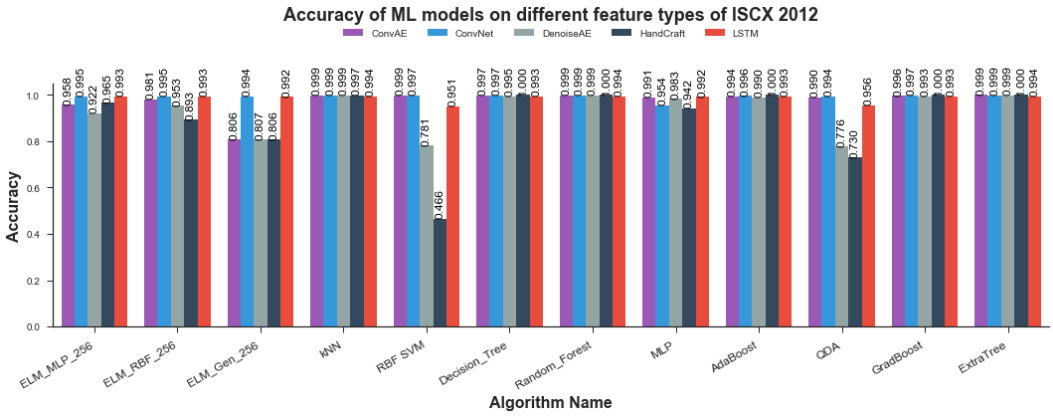

Figure 10. Algorithm-wise Test Accuracy scores for all 12 ML Algorithms.

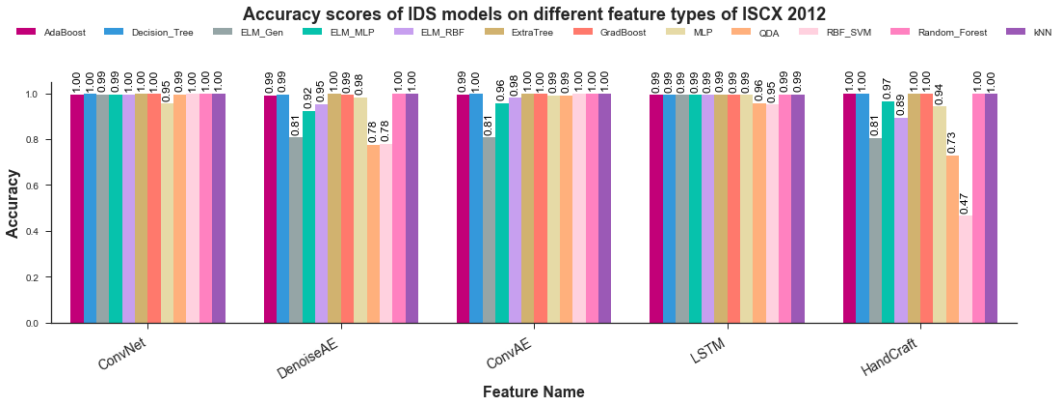

Figure 11. Representation-wise Test Accuracy scores for all 12 ML Algorithms.

5.4. F1-Measure

In cases where we want to find an optimal blend of precision and recall we can combine the two metrics into what is called the F1-score. The F1 score is the harmonic mean of precision and recall combining both metrics in the form of following equation:

$$F1 = 2 * \frac{precision * recall}{precision + recall}$$

Algorithm-wise F1-Measure of anomaly class for trained models are shown in Figure 12. The results closely replicates that of accuracy results. As can be seen from Figure 12, kNN, Random-Forest, Adaboost, Gradboost, Decision-Tree and ExtraTree (Extremely Randomized Trees) classifier show comparable results in the range of 0.97 to 0.99. SVM, QDA and ELM variants showed fluctuating F1 scores ranging from poor score of 0.47 to exceptional scores of 0.99. The value 1 shown in the figures is due to rounding-off 0.999 to two digits. Figure 13 provides representation-wise F1-measure scores for anomaly detection models. CNN representations once again emerged as most amenable Data Representation for all ML algorithms employed in current study. All algorithms trained on ConvNet representations showed significant F1-measure scores in the range of 0.98 to 0.99 except MLP which depicted score of 0.89. LSTM representations came forth as runner up with marginally less F1 scores ranging from 0.89 to 0.98. ConvAE representations scores depicted slightly more variations than that of LSTM but some of the models achieved maximum F1 score. Handcrafted representation showed inconsistent F1 scores ranging from poor score of 0.42 to exceptional score of 0.99 rounded-off to 1. Once again F1 scores corroborated the findings revealed by previously discussed model evaluation parameters including RoC-Curve, AuC, PR-Curve, mAP and Accuracy of classification.

The results of the study show with significant confidence that DNNs in general and CNNs in particular are very efficient in automating the process of learning network data representations from raw network packet Data for training state of the art anomaly detection models. We can enormously speedup the training of uptodate IDS systems by automating feature extraction, selection and dimensionality reduction processes using the innovations in deep learning.

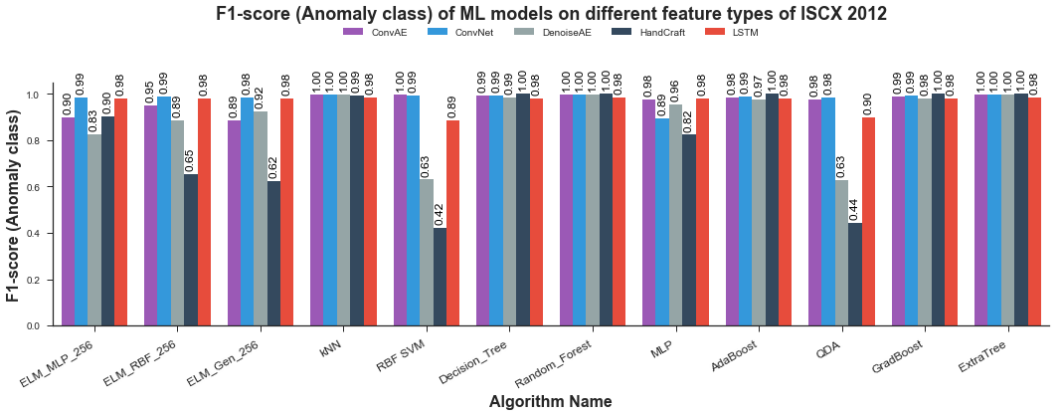

Figure 12. Algorithm-wise F1-measure scores for all 12 ML Algorithms.

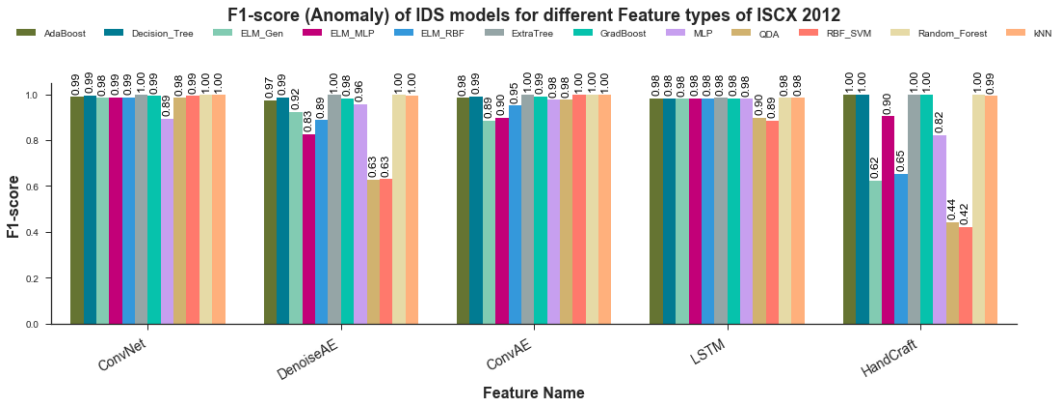

Figure 13. Representation-wise F1-measure scores for all 12 ML Algorithms.

5.5. Timing Information

This section presents the timing comparison between various network data representations created for this study and models trained on them. The comparison of timing information between handcrafted and Deep representations is not shown in any figure because the difference was staggering. It took our testbed more than 14 h to calculate Handcrafted representation from extracted flows of ISCX 2012. Deep Model training for calculating deep network data representations, on the other hand, took time of the order of minutes rather than hours due to usage of GPU. Model Training time for different DNN models is presented in Figure 14. In DNN models, LSTM-based Representation learning model came forth as most expensive in terms of time with training time of 1380 s for 30 epochs. CNN based model proved to be most cost effective DNN model with 22 s for each epoch resulting in total training time of 660 s for 30 epochs. We would like to mention that Feature Extraction time for both Handcrafted and Deep Representations did not include time to extract flows from pcap files which itself took more than 18 h.

Regarding Model training using conventional ML algorithms, change of representations for training anomaly detection models by conventional ML algorithms did show noticeable difference in some instances. kNN, SVM, MLP and Gradboost showed remarkable fluctuations on change of representations for model training. ConvAE representation proved to be most expensive in SVM and

GradBoost case and least expensive in the case of kNN and MLP. CNN representation remained runner up with respect to training times in 3 of the 4 above-mentioned cases. Timing fluctuations in remaining ML algorithms stayed relatively un-noticeable. Handcrafted representation proved most expensive for MLP but remained on the cost effective side for remaining algorithms. Training time information for Conventional ML Algorithms is shown in Figure 15.

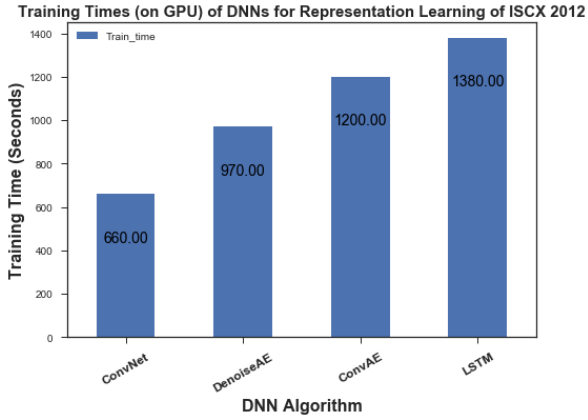

Figure 14. Model Training times of DNN models (on GPU) for Learning Representations of ISCX 2012.

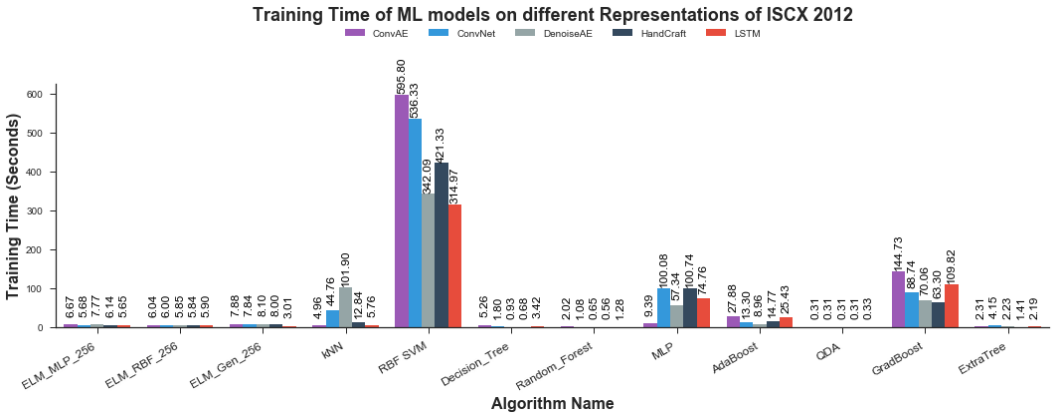

Figure 15. Model Training times of Conventional ML algorithms for Anomaly Detection using ISCX 2012.

6. Conclusions

Oil and Gas organizations are dependent on their IT infrastructure, which is a small part of their industrial automation infrastructure, to function effectively. This research focuses on the ISA-95:2005 level-4 IT infrastructure to address network anomaly detection problem for ensuring the security and reliability of O&G enterprise resource planning. To develop network anomaly detection systems, feature extraction and selection is a necessary but very resource intensive requirement. In this study, we proposed the use of deep learning algorithms to automate feature extraction and selection process for network anomaly detection. We trained four Deep Neural Network based models to extract deep representations from ISCX 2012 traffic flows. We trained multiple anomaly detection models using twelve Conventional ML algorithms to compare the performance of four Deep representations with that of handcrafted representation. The comparisons were performed using well known model evaluation parameters including RoC Curve, Area Under RoC Curve, F1-measure, Precision-Recall curve, mean Average Precision and Accuracy scores. Best average performance with respect to aforementioned performance metrics is shown by CNN representation while 2nd best performance was achieved by RNN-LSTM representation. Performance of Human-engineered representation was

better than autoencoder representation but remained inferior as compared to RNN-LSTM and CNN representation. The results showed that Deep network data representations are at least as effective as Human-Engineered representations for developing Network Anomaly Detection systems. In addition, learning deep representations require a fraction of the cost in terms of time using GPUs due to inherent support of parallel operations in DNNs without requiring human intervention and extensive domain knowledge. It was empirically shown that deep representations are an effective and extremely cost effective alternative for feature engineering for anomaly detection systems.

Author Contributions: Conceptualization, S.N. and R.F.A.; methodology, S.N. and P.D.D.D.; software, S.N. and Y.S.; validation, S.N. and R.F.A.; formal analysis, P.D.D.D. and Y.S.; data curation, S.N. and R.F.A.; writing—original draft preparation, S.N. and R.F.A.; writing—review and editing, P.D.D.D. and Y.S.; visualization, S.N. and Y.S.; supervision, P.D.D.D. and Y.S.; project administration, R.F.A. and P.D.D.D.; funding acquisition, P.D.D.D. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by YUTP-FRG Grant (015LCO-171).

Acknowledgments: We are thankful for the help and guidance provided by P.D.D. Dominic and Yasir Saleem; without their advice, it would have been impossible to achieve goals of this research.

Conflicts of Interest: The authors declare no conflict of interest. The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

1. Economics BEBP. *BP Energy Outlook*; Report; British Petrolium: London, UK, 2019.
2. Colwill, C. Human factors in information security: The insider threat—Who can you trust these days? *Inf. Secur. Tech. Rep.* **2009**, *14*, 186–196. [CrossRef]
3. Ali, R.F.; Dominic, P.; Karunakaran, P.K. Information Security Policy and Compliance In Oil And Gas Organizations—A Pilot Study. *Solid State Technol.* **2020**, *63*, 1275–1282.
4. Ali, R.F.; Dominic, P.; Ali, K. Organizational governance, social bonds and information security policy compliance: A perspective towards oil and gas employees. *Sustainability* **2020**, *12*, 8576. [CrossRef]
5. Lu, H.; Huang, K.; Azimi, M.; Guo, L.J.I.A. Blockchain technology in the oil and gas industry: A review of applications, opportunities, challenges, and risks. *IEEE Access* **2019**, *7*, 41426–41444. [CrossRef]
6. Wueest, C. *Targeted Attacks against the Energy Sector*; Symantec Security Response: Mountain View, CA, USA, 2014.
7. Lu, H.; Guo, L.; Azimi, M.; Huang, K. Oil and Gas 4.0 era: A systematic review and outlook. *Comput. Ind.* **2019**, *111*, 68–90. [CrossRef]
8. International Society of Automation. *Enterprise-Control System Integration*; Part 3: Activity Models of Manufacturing Operations Management; ISA: Durham, NC, USA, 2005.
9. Foehr, M.; Vollmar, J.; Calà, A.; Leitão, P.; Karnouskos, S.; Colombo, A.W. Engineering of Next Generation Cyber-Physical Automation System Architectures. In *Multi-Disciplinary Engineering for Cyber-Physical Production Systems*; Biffl, S., Lüder, A., Gerhard, D., Eds.; Springer: Cham, Switzerland, 2017; pp. 185–206.
10. Si, W.; Li, J.H.; Huang, X.J. *Features Extraction Based on Deep Analysis of Network Packets in Industrial Control Systems*; Springer: Singapore, 2020; Volume 595, pp. 524–529. [CrossRef]
11. Thudumu, S.; Branch, P.; Jin, J.; Singh, J.J. A comprehensive survey of anomaly detection techniques for high dimensional big data. *J. Big Data* **2020**, *7*, 42. [CrossRef]
12. Kurniabudi, K.; Purnama, B.; Sharipuddin, S.; Darmawijoyo, D.; Stiawan, D.; Samsuryadi, S.; Heryanto, A.; Budiarto, R. Network anomaly detection research: a survey. *Indones. J. Electr. Eng. Informatics* **2019**, *7*, 37–50. [CrossRef]
13. Tavallaee, M.; Bagheri, E.; Lu, W.; Ghorbani, A.A. A Detailed Analysis of the KDD CUP 99 Data Set. In Proceedings of the Second IEEE International Conference on Computational Intelligence for Security and Defense Applications, Piscataway, NJ, USA, 7–10 September 2009; pp. 53–58.
14. Zhu, X.; Goldberg, A.B. Introduction to Semi-Supervised Learning. *Synth. Lect. Artif. Intell. Mach. Learn.* **2009**, *3*. [CrossRef]

15. Luo, M.; Wang, L.; Zhang, H.; Chen, J. *A Research on Intrusion Detection Based on Unsupervised Clustering and Support Vector Machine*; Springer: Berlin/Heidelberg, Germany, 2003; pp. 325–336. [CrossRef]
16. LeCun, Y.; Bengio, Y.; Hinton, G. Deep learning. *Nature* **2015**, *521*, 436. [CrossRef]
17. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*; MIT Press: Cambridge, MA, USA, 2016.
18. Naseer, S.; Saleem, Y.; Khalid, S.; Bashir, M.K.; Han, J.; Iqbal, M.M.; Han, K. Enhanced Network Anomaly Detection Based on Deep Neural Networks. *IEEE Access* **2018**, *6*, 48231–48246. [CrossRef]
19. Naseer, S.; Saleem, Y. Enhanced Network Intrusion Detection using Deep Convolutional Neural Networks. *TIIS* **2018**, *12*, 5159–5178. [CrossRef]
20. Hamamoto, A.H.; Carvalho, L.F.; Sampaio, L.D.H.; Abrão, T.; Proença, M.L., Jr. Network anomaly detection system using genetic algorithm and fuzzy logic. *Expert Syst. Appl.* **2018**, *92*, 390–402. [CrossRef]
21. Song, J.; Zhao, W.; Liu, Q.; Wang, X. Hybrid feature selection for supporting lightweight intrusion detection systems. *J. Phys.* **2017**, *887*, 012031. [CrossRef]
22. Lashkari, A.H.; Gil, G.D.; Mamun, M.S.I.; Ghorbani, A.A. Characterization of Tor Traffic using Time based Features. In Proceedings of the 3rd International Conference on Information Systems Security and Privacy, Porto, Portugal, 19–21 February 2017; pp. 253–262. [CrossRef]
23. Shiravi, A.; Shiravi, H.; Tavallaee, M.; Ghorbani, A.A. Toward developing a systematic approach to generate benchmark datasets for intrusion detection. *Comput. Secur.* **2012**, *31*, 357–374. [CrossRef]
24. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems 25*; Pereira, F., Burges, C.J.C., Bottou, L., Weinberger, K.Q., Eds.; Curran Associates, Inc.: Red Hook, NY, USA, 2012; pp. 1097–1105.
25. Hinton, G.; Deng, L.; Yu, D.; Dahl, G.E.; Mohamed, A.R.; Jaitly, N.; Senior, A.; Vanhoucke, V.; Nguyen, P.; Sainath, T.N.; et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Process. Mag.* **2012**, *29*, 82–97. [CrossRef]
26. Sutskever, I.; Vinyals, O.; Le, Q.V. Sequence to sequence learning with neural networks. In Proceedings of the Advances in Neural Information Processing Systems, Montreal, QC, Canada, 8–13 December 2014; pp. 3104–3112.
27. Hodo, E.; Bellekens, X.; Hamilton, A.; Tachtatzis, C.; Atkinson, R. Shallow and Deep Networks Intrusion Detection System: A Taxonomy and Survey. *arXiv* **2017**, arXiv:1701.02145.
28. Ghorbani, A.A.; Lu, W.; Tavallaee, M. *Network Intrusion Detection and Prevention*, *Advances in Information Security*; Springer: Boston, MA, USA, 2010; Volume 47.
29. Gao, N.; Gao, L.; Gao, Q.; Wang, H. An Intrusion Detection Model Based on Deep Belief Networks. In Proceedings of the 2014 Second International Conference on Advanced Cloud and Big Data, Huangshan, China, 20–22 November 2014; pp. 247–252. [CrossRef]
30. Staudemeyer, R.C.; Omlin, C.W. Evaluating performance of long short-term memory recurrent neural networks on intrusion detection data. In Proceedings of the South African Institute for Computer Scientists and Information Technologists Conference, East London, South Africa, 7–9 October 2013; pp. 218–224.
31. Tuor, A.; Kaplan, S.; Hutchinson, B.; Nichols, N.; Robinson, S. Deep learning for unsupervised insider threat detection in structured cybersecurity data streams. *arXiv* **2017**, arXiv:1710.00811.
32. Wang, Z. *The Applications of Deep Learning on Traffic Identification*; BlackHat: Las Vegas, NV, USA, 2015.
33. Ashfaq, R.A.R.; Wang, X.Z.; Huang, J.Z.; Abbas, H.; He, Y.L. Fuzziness based semi-supervised learning approach for intrusion detection system. *Inf. Sci.* **2017**, *378*, 484–497. [CrossRef]
34. Du, M.; Li, F.; Zheng, G.; Srikumar, V. *DeepLog: Anomaly Detection and Diagnosis from System Logs through Deep Learning*; ACM Press: New York, NY, USA, 2017; pp. 1285–1298. [CrossRef]
35. Hochreiter, S.; Schmidhuber, J. Long Short-Term Memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef]
36. Arnaldo, I.; Cuesta-Infante, A.; Arun, A.; Lam, M.; Bassias, C.; Veeramachaneni, K. Learning Representations for Log Data in Cybersecurity. In *Cyber Security Cryptography and Machine Learning*; Dolev, S., Lodha, S., Eds.; Springer: Cham, Switzerland, 2017; Volume 10332, pp. 250–268. [CrossRef]
37. Ian, J. Principal Component Analysis. In *Encyclopedia of Statistics in Behavioral Science*; American Cancer Society: Atlanta, GA, USA, 2005. [CrossRef]
38. Maaten, L.V.D.; Hinton, G. Visualizing data using t-SNE. *J. Mach. Learn. Res.* **2008**, *9*, 2579–2605.
39. Karpathy, A. Connecting Images and Natural Language. Ph.D. Thesis, Stanford University, Stanford, CA, USA, 2016.

40. Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; Salakhutdinov, R. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *J. Mach. Learn. Res.* **2014**, *15*, 1929–1958.
41. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. *arXiv* **2014**, arXiv:1412.6980.
42. Masci, J.; Meier, U.; Cireşan, D.; Schmidhuber, J. Stacked convolutional auto-encoders for hierarchical feature extraction. In Proceedings of the Artificial Neural Networks and Machine Learning–ICANN 2011, Espoo, Finland, 14–17 June 2011; pp. 52–59.
43. Zeiler, M.D. ADADELTA: An Adaptive Learning Rate Method. *arXiv* **2012**, arXiv:1212.5701.
44. Vincent, P.; Larochelle, H.; Lajoie, I.; Bengio, Y.; Manzagol, P.A. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *J. Mach. Learn. Res.* **2010**, *11*, 3371–3408.
45. Bengio, Y.; Simard, P.; Frasconi, P. Learning Long-Term Dependencies with Gradient Descent is Difficult. *IEEE Trans. Neural Netw.* **1994**, *5*, 157–166. [CrossRef]
46. Bergstra, J.; Bengio, Y. Random search for hyper-parameter optimization. *JMLR* **2012**, 13, 281–305.
47. Mcginnis, W. *BaseN Encoding and Grid Search in Categorical Variables*. Available online: http://www.willmcginnis.com/2016/12/18/basen-encoding-grid-search-category_encoders/ (accessed on 10 May 2018).
48. Mcginnis, W. *Beyond OneHot: An Exploration of Categorical Variables*. Availableonline:http://www.willmcginnis.com/2015/11/29/beyond-one-hot-an-exploration-of-categorical-variables/ (accessed on 20 May 2018).
49. Group, S.C. *Contrast Coding Systems for Categorical Variables*. Availableonline:https://stats.idre.ucla.edu/spss/faq/coding-systems-for-categorical-variables-in-regression-analysis-2/ (accessed on 7 June 2018).
50. Zhang, O. *Strategies to Encode Categorical Variables with Many Categories*. Available online: https://towardsdatascience.com/smarter-ways-to-encode-categorical-data-for-machine-learning-part-1-of-3-6dca2f71b159 (accessed on 27 June 2018).
51. Huang, G.B.; Zhu, Q.Y.; Siew, C.K. Extreme learning machine: Theory and applications. *Neurocomputing* **2006**, *70*, 489–501. [CrossRef]
52. Fernandez-Navarro, F.; Hervas-Martinez, C.; Sanchez-Monedero, J.; Gutierrez, P.A. MELM-GRBF: A modified version of the extreme learning machine for generalized radial basis function neural networks. *Neurocomputing* **2011**, *74*, 2502–2510. [CrossRef]
53. Martin, A.; Ashish, A.; Paul, B.; Eugene, B.; Zhifeng, C.; Craig, C.; Greg, S.C.; Andy, D.; Jeffrey, D.; Matthieu, D.; et al. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. *arXiv* **2015**, arXiv:1603.04467.
54. Fawcett, T. An introduction to ROC analysis. *Pattern Recognit. Lett.* **2006**, *27*, 861–874. [CrossRef]