Received 26 March 2020; accepted 14 April 2020. Date of publication 27 April 2020; date of current version 29 May 2020.

Digital Object Identifier 10.1109/OJCOMS.2020.2989925

Incidents Information Sharing Platform for Distributed Attack Detection

KONSTANTINA FOTIADOU [1], TERPSICHORI-HELEN VELIVASSAKI [2], ARTEMIS VOULKIDIS [1], KONSTANTINOS RAILIS [1], PANAGIOTIS TRAKADAS [3], AND THEODORE ZAHARIADIS [3]

[1] Power Operations Ltd., Swindon SN3 5HQ, U.K.

[2] SingularLogic, Attica 145 64, Greece

[3] National and Kapodistrian University of Athens, Athens 157 72, Greece

CORRESPONDING AUTHOR: K. FOTIADOU (e-mail: fotiadou@power-ops.com)

This work was supported by the H2020 Framework Program of the European Commission (DEFENDER Project) under Contract 740898 and (PHOENIX Project) under Contract 832989.

ABSTRACT Intrusion detection plays a critical role in cyber-security domain since malicious attacks cause irreparable damages to cyber-systems. In this work, we propose the *I2SP* prototype, which is a novel Information Sharing Platform, able to gather, pre-process, model, and distribute network-traffic information. Within the I2SP prototype we build several challenging deep feature learning models for network-traffic intrusion detection. The learnt representations will be utilized for classifying each new network measurement into its corresponding threat level. We evaluate our prototype's performance by conducting case studies using cyber-security data extracted from the Malware Information Sharing Platform (MISP)-API. To the best of our knowledge, we are the first that combine the MISP-API in order to construct an information sharing mechanism that supports multiple novel deep feature learning architectures for intrusion detection. Experimental results justify that the proposed deep feature learning techniques are able to predict accurately MISP threat-levels.

INDEX TERMS Malware information sharing platform, network intrusion detection, anomaly detection, deep feature learning, convolutional neural networks, long-short memory neural networks, stacked-sparse autoencoders.

I. INTRODUCTION

NOWADAYS, the demand for designing efficient, end-to-end Network Intrusion Detection Systems (NIDS) for Cyber Physical Systems (CPS) has grown tremendously. Typically, CPS are composed of a combination of physical components and modern sensors. Modern sensors are responsible for monitoring the physical environment and provide high-quality multivariate time-series measurements, while physical components are related with the structure of each application. CPS have been widely utilized in electric power Critical Energy Infrastructures (CEIs), automobile systems, and oil and natural gas distributions, among others. In practice they present high similarity with Internet of Things (IoT) systems, since they put extra attention on the connection of the individual components with the networks. Towards this direction, CPS are considered as the main target for cyber-attacks. Consequently, it is highly important not only to closely monitor the behaviours of these systems for intrusion events, but also to design efficient, large-scale and flexible mechanisms, able to detect abnormal behaviours.

An anomaly in CPS can be considered any pattern in certain time period that depicts significantly different or suspicious behaviour from the previous normal phase. The main task of network-traffic anomaly detection is to identify whether a new event can be considered as normal or suspicious, and provide further details regarding the structure of the detected threat type. Traditionally, statistical-based approaches [1] and probabilistic frameworks [2] tackle the problem of network anomaly detection. Nevertheless, these techniques are proved to work well only with binary classification tasks. Another limitation of probabilistic-based approaches lie into the prior knowledge of input signals distributions, since these techniques suggest that the normal input measurements adhere to a standard distribution.

Consequently, for the testing phase, the probability distributions of the new measurements are compared with a provided empirical threshold in order to determine whether the new signal examples are considered as normal or anomalous. However, in real-world scenarios, these techniques are not able to model the rapidly changing structure of multivariate datasets that are generated by modern CPSs. Consequently, novel and large-scale machine learning formulations should be developed in order to exploit the enormous amounts of training data that were synthesized by these systems.

Moving towards to this direction, we provide the prototype version of the proposed Information Sharing Platform (I2SP), utilizing network incidents extracted from the Malware Information Sharing Platform (MISP)-API [3]. MISP is a trusted system that distributes important indicators of compromise (IoC) of targeted attacks, and threat information, such as financial indicators or vulnerabilities that are used in fraud cases. The main goal of MISP is to define a standard framework of preventive counter-measures that are used against targeted attacks, existing malware and other threats. The I2SP prototype exploits the MISP architecture, and thus extracts significant information regarding coordinated activities for the minimization of cascading effects. Additionally, within the I2SP, we develop an innovative network-traffic intrusion detection system able to detect complex attack patterns or new threats, and then share the information regarding the cyber security incidents to the responsible authorities.

The key contributions of this paper can be summarized as:

- the development of a novel Incidents Sharing Centre (I2SP), that combines attributes from variant network measurements and provides the complete overview of the security status of the system, exploiting the MISP architecture;
- the design of several challenging Deep Feature Learning architectures for semi-supervised anomaly detection.

A key benefit of the proposed method is its flexibility, since the anomaly detection platform may support variant Machine Learning (ML) techniques. Additionally, we are the first to propose multiple innovative deep-feature learning architectures towards the problem of MISP network-traffic measurements modelling, with upper goal the threat-level identification, and distribution of this information through the proposed I2SP platform. Additionally, the proposed scheme can be easily extended in dealing with multiple sources of cyber-physical data. Figure 1 illustrates the proposed system's block diagram, while the following Sections provide the complete analysis.

The rest of the paper is organized as follows: Section II provides an overview of the related state-of-the-art regarding the network information sharing mechanisms and the network traffic anomaly detection. Section III presents the proposed Incidents Information Sharing Platform (I2SP) prototype, while Section IV demonstrates the variant deep feature learning techniques that were implemented towards the network intrusion detection problem. Section V provides the data sources, along with the experimental results of the comparable Deep Feature learning architectures. Extensions of this work are discussed in Section VI.

II. RELATED WORK

A. NETWORK INFORMATION SHARING MECHANISMS

Computer Emergency Response Teams (CERTs) are a vital part of the cyber security community, since they collect information of new threat types, issue early warnings and provide assistance in requests made by the users. These organizations are able to collect huge amounts of data from multiple distributors using threat libraries. Consequently, threat intelligence tools should be utilized in order to handle the information, and perform decision making. In the following paragraphs we highlight several representative information sharing tools [4].

In [5] the *Collaborative Research into Threats* (CRITs) platform is demonstrated. CRIT is a data repository, providing analysts with the necessary tools in order to operate collaborative research into malware and threats. Another threat intelligence tool is the *Soltra Edge* [6] that supports a highly extensible community defence model. Moreover, the *Collective Intelligence Framework* (CIF) [7], helps to aggregate, store, post-process, share and produce threat intelligence datasets. Likewise, *Threatelligence* [8] is a cyber-threat intelligence feed collector, using Elastic-Search, Kibana and Python. It fetches data from various custom or public sources into Elastic-Search while simultaneously adds geospatial information. Additionally, *AlliaCERT* tool [9] collects security-related information from multiple sources and provides mechanisms to query, correlate, and share it.

Finally, the *Malware Information Sharing Platform* (MISP) [3], is an open source software solution mainly developed by the Belgian Defense CERT and the NATO Computer Incident Response Capability (NCIRC), aiming at gathering information about malware and attacks, storing data in a standardized format, and distributing cyber security indicators and malware analysis with trusted parties.

B. NETWORK TRAFFIC ANOMALY DETECTION

Anomalies can be defined as patterns that appear infrequently and do not conform to the already defined, normal behaviour. Abnormal can be considered for instance the network traffic that does not comply with the normal class. However, anomaly detection systems should be able to provide flexibility in discriminating whether the new network connections contain suspicious content or not. Over the last years, significant research has been conducted in the framework of network-traffic anomaly detection. From a high-level perspective, anomaly detection techniques may be separated into three main categories:

- *supervised*: where label information regarding the threat type is available [10]
- *unsupervised*: where no labelling data appear on the dataset [11], [12]

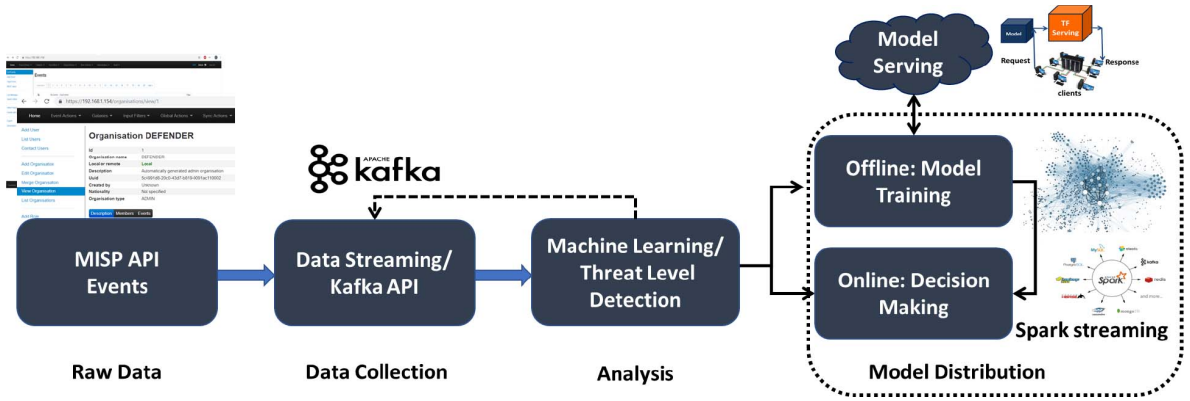


FIGURE 1. The I2SP Architecture.

- *semi-supervised*: where partial knowledge regarding the anomaly type is available [13], [14], [15]

Towards the supervised direction, the key objective is to construct proper training datasets that include all the anomalous examples along with their corresponding labels. Since this procedure can be considered as a standard classification approach, the main advantage is its flexibility in identifying if a new pattern is suspicious or not, based on already existing attack patterns. Therefore, classification techniques require a training period where the normal profile activity is constructed, and a testing period where every new network measurement can be determined as anomalous or normal. In the scenario where we are dealing with more threat types, proper labelling should be also implemented for each available class, and thus we are dealing with a multi-class supervised classification problem. Early classification approaches rely on the widely utilized framework of Support Vector Machines (SVMs) [16], [17]. Typically, SVM is considered as a supervised learning technique that requires labelled data to create the classification rule. However, a major limitation of the supervised frameworks lie in the insufficiency of labelled training data, since these techniques rely on extensive prior knowledge regarding the characteristics of the network attacks, i.e., the attack pattern.

Regarding the unsupervised direction, clustering-based techniques are proved to be a convenient solution [18], [19]. However, the major drawback of clustering approaches lie in the fact that they are not able to guarantee that the identified classes correspond to the desired ones, even in the case where a large amount of data is available for different threat type scenarios. Recently, generative models have also proved to be an intelligent choice towards the unsupervised network anomaly detection problem [20], [21], [22] Concerning the semi-supervised anomaly detection scenario, state-of-the-art approaches [14] aim to provide accurate estimates of the probability distribution of the normal and abnormal classes, extracted from a sufficiently large amount of collected network measurements. The main difference with the supervised methods is that there exists partial or no information regarding the properties of the distortion (i.e., anomalous region).

Finally, over the last years, multiple deep learning formulations for network intrusion detection have been proposed in literature. For instance, the authors in [23], [24] propose Long-Short Term Memory (LSTM) network architectures that confront the problem of network traffic intrusion detection. Moreover, in [25], [26], [27] several Convolutional Neural Network (CNNs) anomaly detection techniques were proposed. Additionally, several deep learning formulation for network anomaly detection employ the intelligent scheme of Stacked Autoencoders (SAE) [28], [29], [30]. Despite the fact that deep learning architectures are demanding, since a great amount of data and internal iterations are required, these methods usually outperform traditional supervised and unsupervised intrusion detection approaches.

III. THE I2SP PROTOTYPE

This section demonstrates the proposed Information Sharing Platform (I2SP) prototype. The upper goal of this architecture is to provide continuous monitoring, improve current cyber-threat detection capabilities, and notify and report to higher level the corresponding Computer Emergency Response Teams (CERTs). A key component in providing the prototype version of the I2SP is the proper selection of the data-measurements to analyse and distribute. Out of the multiple Network Information Sharing Mechanisms that were highlighted in the Related Work Section, we selected the *Malware Information Sharing Platform* (MISP) framework, due to its comprehensive and extensive structure.

A. MALWARE INFORMATION SHARING PLATFORM (MISP)

MISP is an open-source threat information sharing platform, where users from various communities are able to share all kind of cyber-threats, indicators of compromise, and financial indicators among others. Due to its peer-to-peer structure, multiple instances exchange information with each other, while its synchronization protocol is relied on three main criteria: efficiency, accuracy and scalability. The users are able to determine the granularity of the information they want to distribute in MISP, for instance with respect to the organization level, the community-level, or within their sharing groups.

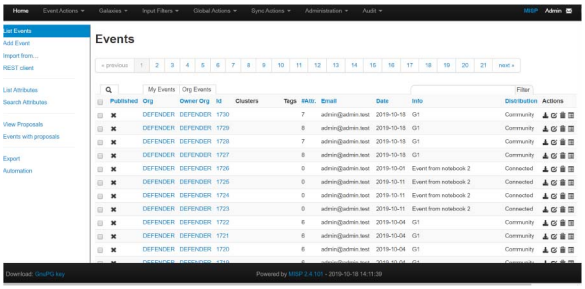


FIGURE 2. The Malware Information Sharing Platform (MISP) Interface.

TABLE 1. MISP API's threat levels.

Threat Id	Definition
['EventThreatLevelId']['1']	Sophisticated APT malware or 0-day attack
['EventThreatLevelId']['2']	APT malware
['EventThreatLevelId']['3']	Mass-malware
['EventThreatLevelId']['4']	Undefined Attack/Risk

A shared piece of information in MISP is called an event. An event is composed of a list of attributes, including destination IP addresses and file hashes. An attribute is identified with the tuple: (i) category, (ii) type, (iii) value. Additionally, an event is linked with textual information, where it is available, such as date, threat level, description, organization and galaxies about threat actors, among others. In this study, we exploit the intelligent MISP architecture in order to create and upload events, store them properly, extract them, perform the proposed analysis, and finally distribute the desired outcomes to the responsible authorities.

Figure 2 provides a characteristic snapshot of MISP interface. Within the MISP interface we are able to add any supported information we wish, such as the main attributes (i.e., event ID, date, corresponding organization), economical and financial risks, and threat level. Additionally, Table 1 provides the threat level information that is available within the MISP interface.

Since our ML analysis towards the intrusion detection will rely on the threat level identification, threat levels will be included only at the models' training phase.

B. PROPOSED MECHANISM

The system presented in this paper is designed to upload, monitor and analyse network-traffic measurements utilizing the MISP-API. Specifically, for each new instance that is uploaded on MISP interface, we use Apache Spark Streaming [31] as the tool for real-time processing of the network traffic. Packets from the input data are stored into Apache Kafka [32] topics and are ingested into Spark Streaming using the Kafka consumer API for Spark (Spark-Kafka), synthesizing discretized streams. Spark Streaming provides its own high-level abstraction for continuous data-streams, the so-called DStream, that processes data into mini-batches relied on a configured streaming interval [31], [33].

At the next stage, after we parse the input raw network measurements, we perform the proposed data-preprocessing and analysis. Data-preprocessing is a significant step towards the upcoming modelling stage. In this step, we perform internal data checks, such as the problems of missing values and inappropriate characters. Additionally, since we are dealing with multivariate time-series, we separate the numeric from the categorical variables in order to encode them properly. Specifically, for the categorical variables (e.g., *threat level*, etc.), we use the *state-of-the-art* one-hot-encoding [34] technique, while the numeric variables (i.e., columns) are normalized with respect to the sum of the unit-norm (l_1-norm) [35].

From the data pre-processing step, we proceed to the analysis framework including the development of variant deep-feature learning models for intrusion detection. Specifically, in the data-acquisition stage from the MISP-API, we have also considered the threat-level information. However, multiple MISP instances appear either without threat-level, or with a corrupted version. In these scenarios, the proposed algorithmic formulations aim to extract the missing or corrupted threat level information, in order to distribute it, in a later stage, to the responsible authorities. The analysis stage may be separated into two phases: *(i)* the offline, referring to the model training, and (*ii*) the online, referring to the decision making. Regarding the offline - training stage, we store the learnt models on a Redis server [36], in order to be able to use them and distribute them for the final stage of decision making.

In the following section, we explain the variant deep learning formulations that were exploited in this study for the problem of threat-level detection of MISP instances.

IV. DEEP FEATURE LEARNING FOR NETWORK ANOMALY DETECTION

In this study, we propose challenging deep feature learning architectures towards the problem of anomaly detection of the MISP database. The first model that we developed is a traditional multi-perceptron model [37] composed of several stacked hidden layers, the second one is a Stacked Sparse Autoencoders [38] architecture, the third one adheres to a Convolutional Neural Network (CNN) scheme [39], while the final model follows a CNN-LSTM (Long-Short Term Memory) procedure [25], [40]

A. MULTI-LAYER PERCEPTRON MODEL

Multi-layer perceptron neural networks (MLP-NNs) [41] are the most widely recognized feed-forward neural networks due to their ease of implementation, and smaller training set requirements. The neurons of each subsequent layer (except the input) are generated by the previous layers via an activation function, while the whole network is trained utilizing an unsupervised back-propagation approach [42]. Figure 3 illustrates the main MLP structure that was employed in this work. The input layer contains the encoded information that was extracted from the MISP-API. The hidden layers contain the learnt representation of the input features,

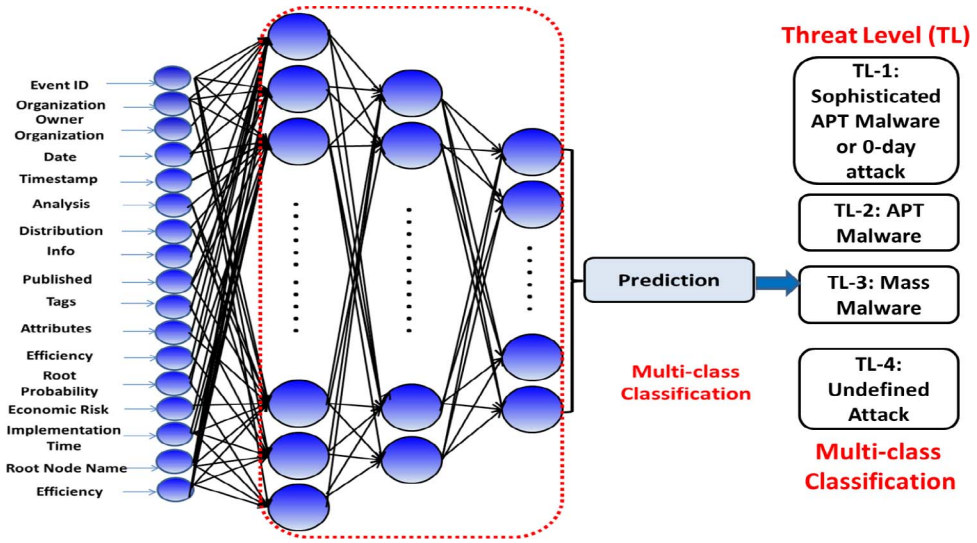


FIGURE 3. Typical structure of a n-layer Multi-Layer Perceptron network: The input layer contains the dimensionality of the input feature space, the hidden layers are composed of a pre-defined number of hidden units (neurons) that are connected with the input and the hidden layers via the weight matrices, and are activated using standard mathematical formulations, like the sigmoid, the tanh, or the ReLU functions. Finally, the output layer (i.e., classification layer) corresponds to the learnt feature vectors, and is the outcome of the minimization among the hidden and input network's layers. The final layer assigns the corresponding probabilities to each threat-level (TL) class.

while they process and transmit this information to the output layer, which is responsible for the classification into the corresponding threat-level. The output layer, was activated using the *Softmax* [43] function, and thus it assigns the corresponding probabilities to each class.

B. STACKED SPARSE AUTOENCODERS FOR NETWORK ANOMALY DETECTION

A classical autoencoder is a deterministic feed-forward artificial neural network comprised of an input and an output layer of the same size with a hidden layer in between, which is trained with back propagation in a fully unsupervised manner, aiming to learn a faithful approximation of the input [38]. Specifically, the formulation encodes the information between input and output data through a nonlinear function $\sigma : \mathbb{R}^N \rightarrow \mathbb{R}^M$, such that each input vector $s \in \mathbb{R}^N$, is mapped to a new feature space via M hidden units.

Formally, a single layer network consists of the input layer units $\mathbf{s} \in \mathbb{R}^N$, the hidden layer unit $\mathbf{h} \in \mathbb{R}^M$, and the output units $\hat{\mathbf{s}} \in \mathbb{R}^N$, which for the case of SAE are set to be equal to the input units. The objective is to learn a set of weights $\mathbf{W} \in \mathbb{R}^{M \times N}$, along with associated encoding bias $\mathbf{b} \in \mathbb{R}^M$, in order to generate compact and descriptive features as:

$$\mathbf{h} = \sigma(\mathbf{W}\mathbf{s} + \mathbf{b}_1), \tag{1}$$

able to accurately reconstruct the input patch $\mathbf{s}$. The function σ is usually selected to be the logistic sigmoid function, defined as: $\sigma(z) = \frac{1}{1+e^{-z}}$. Decoding of $\mathbf{h}$ is performed using the separate weight matrix $\mathbf{V} \in \mathbb{R}^{N \times M}$, that connects the hidden layer with the output units as:

$$\hat{\mathbf{s}} = \sigma(\mathbf{V}\mathbf{h} + \mathbf{b}_2), \tag{2}$$

where $\mathbf{b}_2$ stands for the decoding bias. Following standard approaches, we consider tied weights such that: $\mathbf{W} = \mathbf{V}$.

While an autoencoder is closely related to the Principal Component Analysis (PCA), by performing an internal dimensionality reduction, an over-complete nonlinear mapping of the input vector can be made by applying a sparsity constraint to the target activation function. Consequently, in order to learn representative features, the error of the loss function:

$$\mathcal{L}(\mathbf{S}, \hat{\mathbf{S}}) = \frac{1}{2} \sum_{j=1}^{N} ||\hat{\mathbf{s}}_j - \mathbf{s}_j||_2^2 \tag{3}$$

should be minimized, adhering to a sparsity constraint [44]. In the aforementioned formulation, $\mathbf{S}$ and $\hat{\mathbf{S}}$ correspond to the input and the output data, respectively [45].

Stacked sparse autoencoders (SSAE), is considered the deep learning architecture of multiple shallow sparse autoencoders that are built by stacking additional unsupervised feature layers, and can be trained using greedy methods for each additional layer. Applying a SSAE architecture into the network intrusion problem can be considered by itself as a novel problem for the network security community.

Figure 4 stands as an example, illustrating the basic block diagram for the Stacked Sparse Autoencoders network intrusion detection scheme. The input layer contains the MISP features, while the output layer provides approximately the representation of the input layer. The intermediate layers that are learnt via the networks connections, are directly fed into a softmax layer, which is responsible for the classification to the proper threat-level.

C. CONVOLUTIONAL NEURAL NETWORKS FOR INTRUSION DETECTION

While in fully-connected deep neural networks, the activation of each hidden unit is computed by multiplying the entire input by the correspondent weights for each neuron

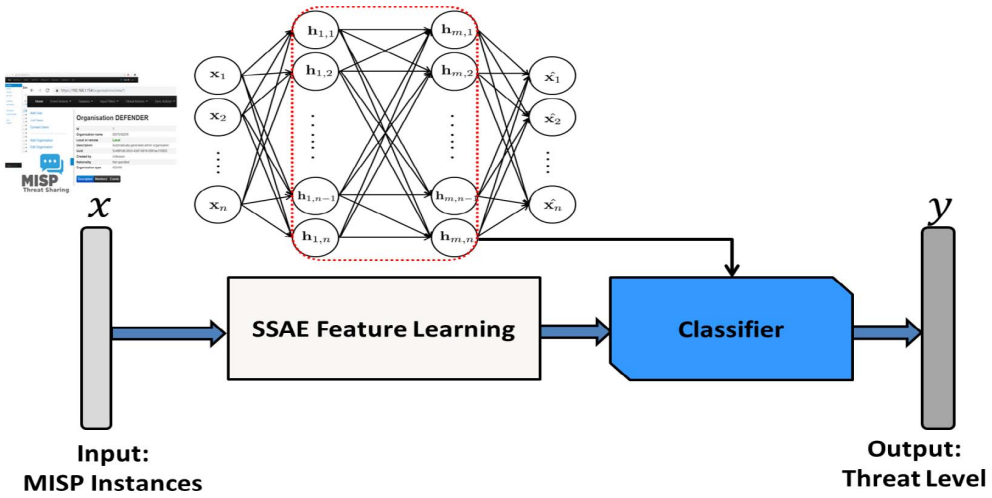


FIGURE 4. Proposed classification scheme utilizing the Stacked Sparse Autoencoders architecture (S-SAE). The input MISP measurements are passed through the S-SAE network, where the learnt features (i.e., hidden layers) are directly fed to the classifier. The output vector y contains the classes for each testing example.

in that layer, in CNNs, the activation of each hidden unit is computed for a small input area [46]. CNNs are composed of convolutional layers which alternate with subsampling (pooling) layers, resulting in a hierarchy of increasingly abstract features, optionally followed by fully connected layers to carry out the final labeling into categories. At the convolution layer, the previous layer's feature maps are first convolved with learnable kernels and then are passed through the activation function to form the output feature map. Specifically, let $n \times n$ be a square region extracted from a training input two-dimensional array $\mathbf{X} \in \mathbb{R}^{N \times M}$, and $\mathbf{w}$ be a filter of kernel size $m \times m$. The output of the convolutional layer $\mathbf{h} \in \mathbb{R}^{(n-m+1)\times(n-m+1)}$ is formulated as: $\mathbf{h}_{ij}^{l} = f(\sum_{k=0}^{m-1} \sum_{l=0}^{m-1} \mathbf{w}_{ab} \mathbf{x}_{(i+k)}^{l-1} + \mathbf{b}_{ij}^{l})$, where $\mathbf{b}$ is the additive bias term, and $f(\cdot)$ stands for the neuron's activation unit. The activation function $f(\cdot)$ is a formal way to model a neuron's output as a function of its input. Typical choices for the activation function are the logistic sigmoid function, the hyperbolic tangent function: $f(x) = \tanh(x)$, and the Rectified Linear Unit (ReLU), given by: f(x)=max(0,x). The majority of state-of-the-art approaches employ the ReLU as the activation function for the CNNs [47].

The output of the convolutional layer is directly utilized as input to a sub-sampling (i.e., pooling) layer that produces down-sampled versions of the input maps. There are several types of pooling; two common types are the max- and the average-pooling. Pooling operators partition the input table into a set of non-overlapping or overlapping samples and output the maximum or average value for each such sub-region. By pooling, the model can reduce its computational complexity for upper layers, and can provide a form of translation invariance. The last layer of a CNN, namely the fully connected layer, is a logistic regression layer [48], where each unit of the output represents a class membership probability, as:

$$P(Y = i|\mathbf{x}) = softmax(\mathbf{W}x + \mathbf{b}) = \frac{e^{\mathbf{W}_i \mathbf{x} + \mathbf{b}_i}}{\sum_j e^{\mathbf{W}_j \mathbf{x} + \mathbf{b}_j}} \quad (4)$$

The network parameters, along with the multiplicative and additive biases, are learnt via a back-propagation procedure [42]. The final prediction of the classification model is implemented by finding the maximum value of:

$$y_{pred} = \underset{i}{argmax} P(\mathbf{Y} = i|\mathbf{x}, \mathbf{W}, \mathbf{b}) \quad (5)$$

Figure 5 illustrates the threat level detection block diagram using the CNN architecture. The initial time-series data are fed into the 1-D convolutional architecture, where appropriate features are learnt, while the output softmax layer is responsible for the classification procedure.

D. CNN-LONG SHORT-TERM MEMORY (LSTM) NETWORKS FOR ANOMALY DETECTION

1) LONG SHORT-TERM MEMORY (LSTM)

Long Short-Term Memory (LSTM) [23] is a variation of Recurrent Neural Networks (RNNs) [49], that utilizes memory blocks to replace the traditional neurons in the hidden layer. Regarding the architecture, a typical LSTM block is composed of a memory cell ($\mathbf{C}_t$), an input gate (i_t), an output gate (o_t), and a forget gate (f_t). At time instance t, $\mathbf{x}_t$ denotes the input and h_t the hidden state, while $\hat{\mathbf{C}}_t$ is the candidate state of the memory cell, which determines how much the input is received in the cell state. The calculations for each gate, input candidate, hidden state, and cell state are presented as follows:

$$i_t = \sigma(\mathbf{W}_i * \mathbf{x}_t + \mathbf{U}_i * \mathbf{h}_{t-1} + \mathbf{b}_i) \quad (6)$$

$$\hat{\mathbf{C}}_t = \tan(\mathbf{W}_c * \mathbf{x}_t + \mathbf{U}_c * \mathbf{h}_{t-1} + \mathbf{b}_c) \quad (7)$$

$$f_t = \sigma(\mathbf{W}_f * \mathbf{x}_t + \mathbf{U}_f * \mathbf{h}_{t-1} + \mathbf{b}_f) \quad (8)$$

$$\mathbf{C}_t = i_t * \hat{\mathbf{C}}_t + f_t * \mathbf{C}_{t-1} \quad (9)$$

$$\mathbf{o}_t = \sigma(\mathbf{W}_o * \mathbf{x}_t + \mathbf{U}_o * \mathbf{h}_{t-1} + \mathbf{b}_o) \quad (10)$$

$$\mathbf{h}_t = \mathbf{o}_t * \tan(\mathbf{C}_t), \quad (11)$$

where σ and *tan* stand for the sigmoid and tanh activation functions, respectively, while $\mathbf{W}_i, \mathbf{W}_f, \mathbf{W}_c, \mathbf{W}_o, \mathbf{U}_i, \mathbf{U}_f, \mathbf{U}_c$ and $\mathbf{U}_o$ represent the weight matrices.

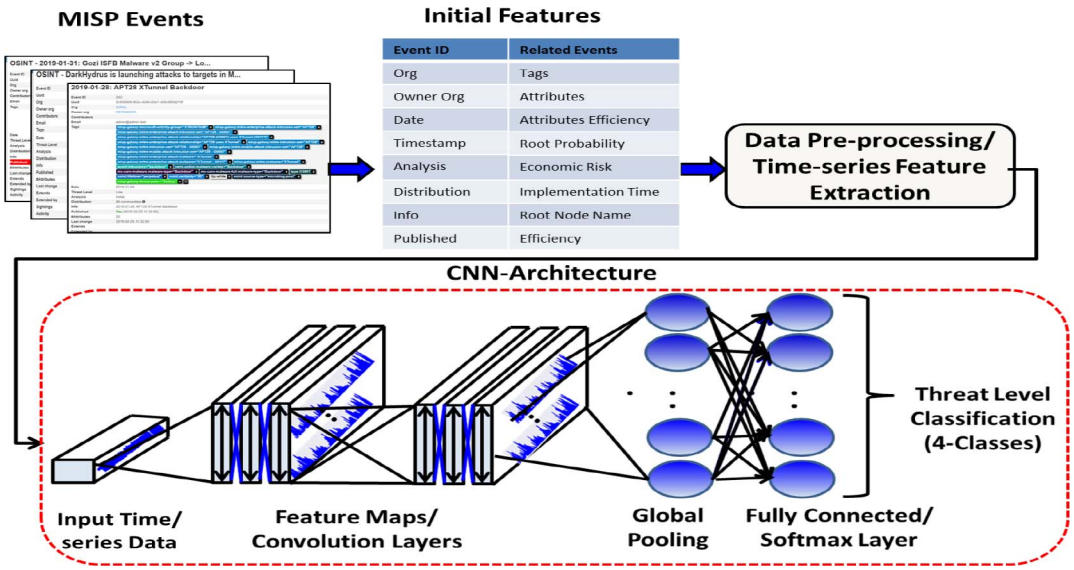


FIGURE 5. CNN Architecture for Threat Level Classification: In this scenario we employ a CNN deep feature approach towards the problem of threat level prediction of MISP instances. The extracted features from MISP API are considered as time-series data from which we learn significant features using the proposed deep CNN structure constructed from a sequence of convolutional and pool layers. The final layer is responsible for the classification into the threat levels, and it is activated using the *softmax* function.

2) CNN-LSTM NETWORKS

The CNN-LSTM approach for predicting the threat level for every new MISP instance consists of a series connection of CNN and LSTM networks. The proposed CNN-LSTM architecture is able to extract complex features from the MISP connections, and thus can store complex and irregular trends. The upper layer of the proposed CNN-LSTM architecture consists of a sequence of CNNs. The CNN architecture is composed of: *(i)* an input CNN layer that receives multiple variables that represent the MISP framework, including event IDs, distribution, information, attributes, economic risk, implementation time, and probability that a possible attack reaches the root of the attack tree, among others, *(ii)* an output layer that extracts the features to the LSTM network, and *(iii)* several hidden layers. The hidden layers are typically consisted of convolution layers, ReLU activation layers, and pooling layers. The convolution layer applies the convolution operation to the incoming multivariate time series sequence and passes the results to the next layer. The main advantage of the specific scheme relies on the flexibility of the convolutional operators in reducing the number of parameters and make the CNN-LSTM network architecture deeper. CNN-LSTM networks using LSTM cells provide superior performance through time information modelling of signals and provide high-performance detection of MISP events' threat levels. The last layer of CNN-LSTM architecture, i.e., the classification layer, is a fully connected layer, that provides the final decision regarding the threat level within a certain period of time for every new MISP instance.

Figure 6 demonstrates the threat level detection block diagram using the CNN-LSTM scheme. The initial time-series data are fed into the 1-D convolutional architecture, which is followed by a recurrent LSTM network. The learnt features are directly provided to the final, softmax layer in order to assign the probabilities to the corresponding classes.

V. EXPERIMENTAL SETUP

A. DATASET DESCRIPTION

The classification task involves predicting the threat level using MISP API's instances. In Table 2 we provide the complete set of features that we utilize in our analysis. To evaluate the performance of the comparable machine learning formulations, experiments were implemented on approximately 40, 000 examples, extracted from the MISP-API. The dataset was divided in a manner such that 80% was used for training (i.e., 32, 000 examples) and the rest 20% was used for testing (i.e., 8, 000 examples).

B. IMPLEMENTATION AND EVALUATION METRICS

The performance of the aforementioned deep feature learning classification approaches is first evaluated in terms of the accuracy, and the reconstruction error of the model's loss function. Accuracy, is determined in the light of true positive T_p (i.e., genuine measurements correctly classified), true negative T_n (i.e., fake measurements correctly classified), false negative F_n (i.e., fake measurements incorrectly classified) and false positive F_p (i.e., genuine measurements incorrectly classified) [50]. It is defined as the ratio between the true outcomes towards the total number of outcomes, formulated as:

$$\text{Accuracy} = \frac{T_p + T_n}{T_p + T_n + F_p + F_n} \tag{12}$$

Regarding the loss function, since we confront a multi-class classification problem, the categorical cross-entropy function

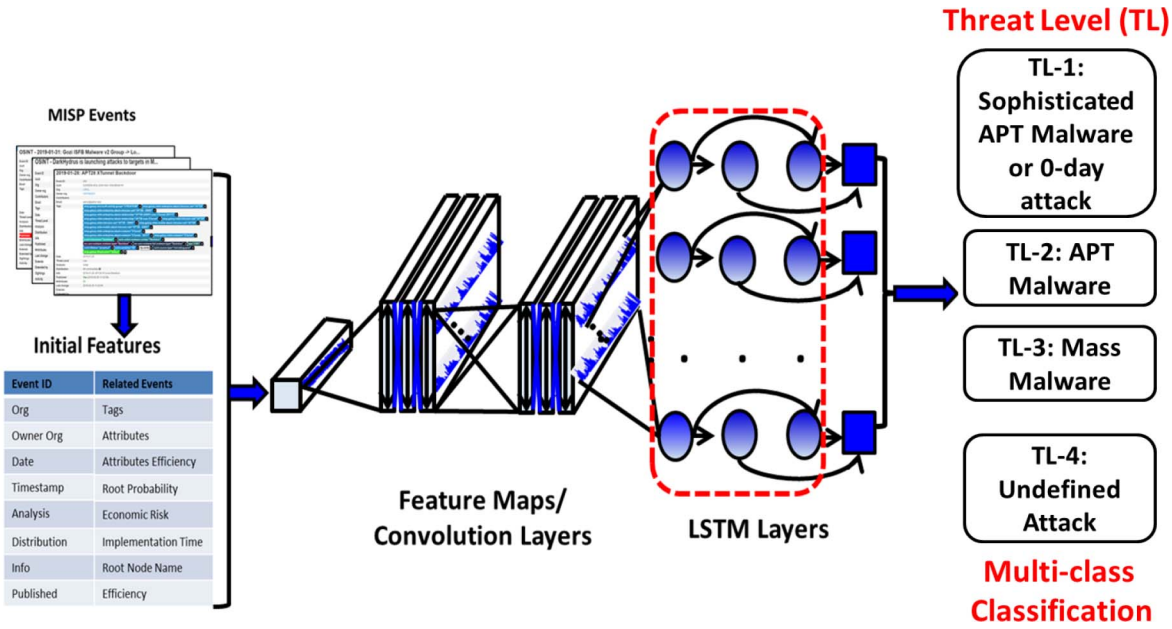


FIGURE 6. CNN-LSTM Architecture for Threat Level Classification: In this scenario we employ a CNN followed by an LSTM architecture, in order to provide even more higher and robust accuracy towards the problem of threat level prediction of MISP events. The learnt features from the convolutional layers are directly fed into a recurrent architecture (i.e., LSTM model). Respectively, in this scenario the output/softmax full-connected layer is responsible for the assignment into the proper threat-level.

TABLE 2. MISP features.

Feature	Type
Event Id	Integer
Information	Character
Published	Character
Analysis	Character
Timestamp	Numeric
Distribution	Character
Organization ID	Numeric
Owner Organization ID	Numeric
Year	Numeric
Month	Numeric
Day	Numeric
Day of the Week	Numeric
Hour	Numeric
Minute	Numeric
Attribute Count	Numeric
Attributes Cost	Numeric
Attributes Implementation Time	Numeric
Attributes Economic Risk	Numeric
Attributes Root Node Name	Character
Attributes Probability to Root	Numeric
Creator e-mail	Character

is exploited and defined as:

$$CE = -\log\left(\frac{e^{\mathbf{s}_p}}{\sum_j^{\mathcal{C}} e^{\mathbf{s}_j}}\right), \tag{13}$$

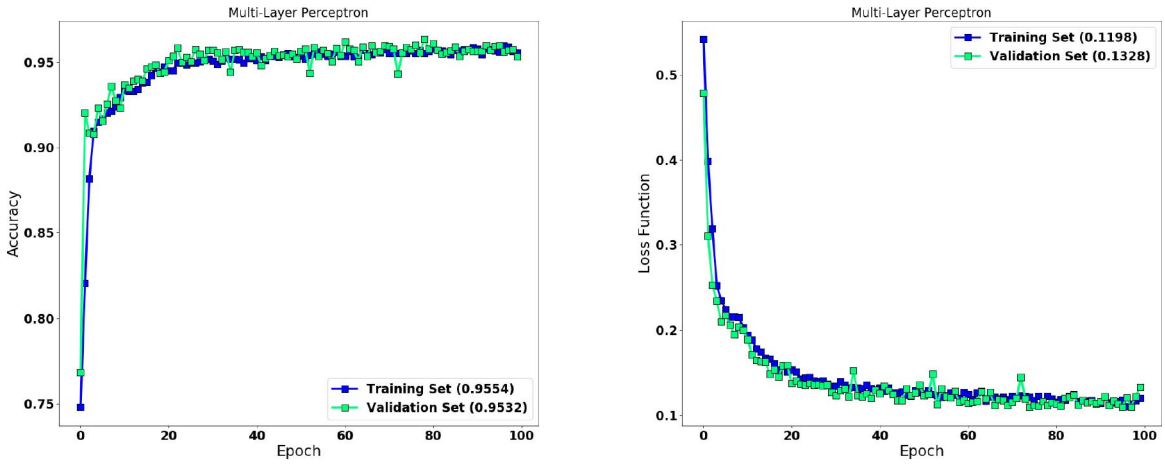
where $\mathbf{t}_i$ and $\mathbf{s}_i$ are the ground truth and model's output scores for each class $i \in \mathcal{C}$.

Additionally, we have selected the Area Under the Receiver Operating Characteristic (ROC) Curve (ROC-AUC) score as an evaluation metric to determine the degree of separability among the different categories, since it measures the classification performance of each model/per each class. ROC curve illustrates the ratio between the True Positive Rate (TPR), i.e., TPR $= \frac{T_n}{T_n+F_p}$, and the False Positive Rate (FPT), i.e., FPR $= \frac{F_p}{F_p+T_n}$. Using the specific metric, we evaluate how the model works while distinguishing between the variant threat levels (TL). ROC-AUC scores that are close to 1.0 indicate highly robust models, that can perfectly determine the variant different classes.

Finally, in order to further validate the quality of our developed models we exploit the Precision $= \frac{T_p}{T_p+F_p}$, Recall $= \frac{T_p}{T_p+F_n}$, and F1-score $= \frac{2T_p}{2T_p+F_p+F_n}$ metrics. High score on Precision metric indicates a lower False Positive Rate. On the other hand, high score on the Recall metric demonstrates low ratio of False Negatives, and thus prevents from false detection. Finally, F1-score provides the harmonic mean of Precision and Recall, by capturing these two measures on a single metric.

C. EXPERIMENTAL RESULTS

In this paragraph we demonstrate the evaluation results obtained using the proposed Machine Learning (ML) formulations. Regarding the Multi-Layer Perceptron (MLP) model, we considered as input layer the features extracted from the MISP-API, and thus we have constructed a deep feature learning architecture composed of 5 hidden layers. The number of hidden nodes was set to: (**128**, **64**, **64**, **32**, **16**) respectively for each hidden layer. The output layer (i.e., classification layer) was activated using the *Softmax* activation function, assigning the corresponding probabilities to the output classes (i.e., the classification levels) that we investigate. The number of batch-size was set to 64, after a cross



(a) *MLP-Model Accuracy* (b) *MLP-Model Loss*

FIGURE 7. **(From Left to Right): In this figure we illustrate the training/validation loss and classification accuracy of the proposed Multi-Layer Perceptron (MLP) neural network applied on the four-class threat level dataset. We observe that after a fixed number of iterations (approximately 60) the proposed model reaches and preserves its highest classification accuracy. Correspondingly, the loss function is constantly reduced into the whole iterations frame.**

TABLE 3. **Confusion matrix of multi-layer perceptron model.**

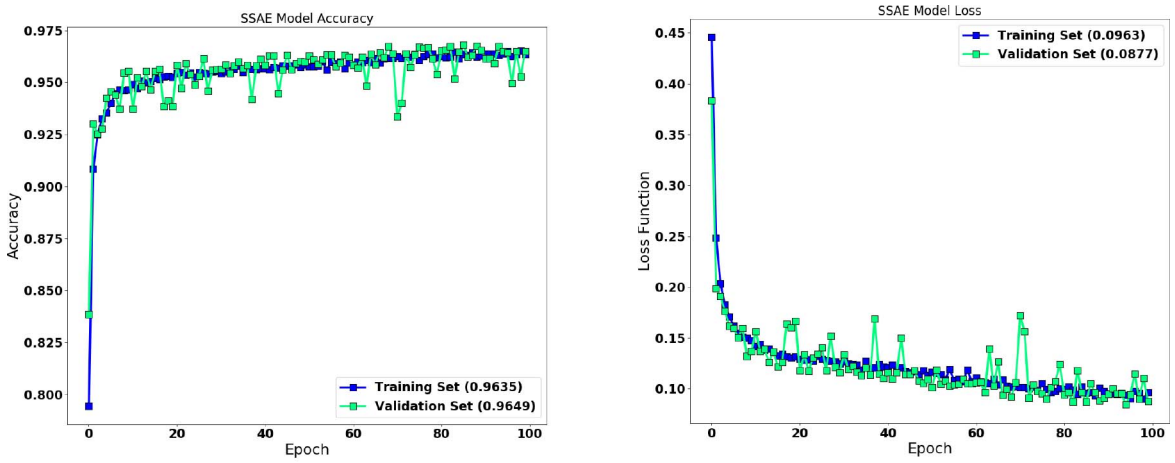
Actual vs.Predicted	TL-1	TL-2	TL-3	TL-4
TL-1	**1714**	0	78	0
TL-2	96	**1510**	163	0
TL-3	196	81	**1518**	0
TL-4	29	0	6	**1859**

validation process, while the number of internal epochs in order to achieve convergence in terms of the reduction of the loss function error, and in terms of achieving a high ratio in accuracy, was set to 100. In Figure 7 we illustrate the MLP model's accuracy and loss function with respect the number of epochs. The total accuracy for the training and validation phases was **95**.**54**% and **95**.**32**% respectively, while the total error of the loss function was **0**.**1198** and **0**.**1328** for the training and testing phases, respectively. Additionally, Table 3 illustrates the confusion matrix of the MLP approach towards the four-threat level (TL) classification scenario. In terms of the Precision metric, the proposed MLP scheme achieves: **84**% for the TL-1: Sophisticated APT-Malware or 0-day attack, **95**% for the TL-2: APT-Malware, **86**% for the TL-3: Mass-malware, and **100**% for the TL-4: Undefined attack type. All results are summarized, illustrated and compared with the other developed schemes in Table 7. As we may notice, the proposed MLP scheme provides high performance detection rate between the actual and the predicted states.

In contrast with the multi-perceptron model, where the output nodes are the network's connection classification outcome, in the Stacked Sparse Autoencoders scenario, the output is approximately the same as the input (i.e., containing the same network measurements, affected by a dramatically small loss function). Consequently, in the SSAE case, we consider as input the 21 feature space containing MISP's network-traffic measurements, and thus we learn the internal representations (i.e., features). The learnt internal representations are directly fed into the final classification layer, which in our case is a *softmax* layer. Specifically, we have constructed a SSAE architecture equiped with 5 hidden layers, containing (**64**, **32**, **16**, **32**, **32**) hidden nodes respectively in each layer, and with a 64 batch size. Additionally, the number of epochs in order to achieve a fair convergence rate was fixed into 100.

Figure 8 depicts the loss function and classification accuracy of the proposed SSAE scheme, when applied on the four-category threat-level classification scenario. After 100 epochs, the training loss is fixed into **0**.**0963**, while the validation loss into **0**.**0877**. Likewise, the training accuracy after 100 epochs is **96**.**35**%, and the validation accuracy **96**.**49**%. As we may observe, the SSAE model built with the aforementioned network architecture achieves high classification accuracy. Referring to the loss function we may notice the high convergence rate of the proposed SSAE scheme. Moreover, Table 4 illustrates the Confusion Matrix of the SSAE scheme with respect to each category (i.e., Threat Level), where we may depict the detection ratio among the actual and predicted classes. Finally, in terms of Precision metric, the proposed SSAE scheme achieves: **91**% for the TL-1: Sophisticated APT-Malware or 0-day attack, **97**% for the TL-2: APT-Malware, **85**% for the TL-3: Mass-malware, and **100**% for the TL-4: Undefined attack type. All results are compared with the other developed architectures in Table 7. From the aforementioned metrics we are able to validate the high prediction quality of the proposed SSAE scheme when applied on MISP network measurements dataset.

Additionally, we have further investigated the performance of an 1D-Convolutional Neural Networks (CNNs)

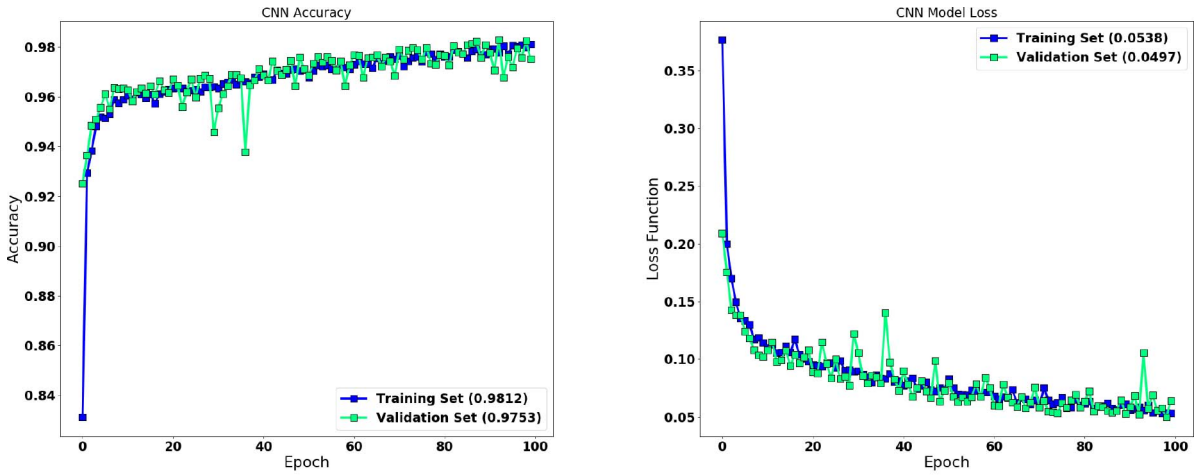


(a) *SSAE-Model Accuracy* (b) *SSAE-Model Loss*

FIGURE 8. **(From Left to Right): In this figure we illustrate the training/validation loss and classification accuracy of the proposed SSAE scheme applied on the four-class MISP dataset. We observe that the loss function is continuously reduced within the 100-epochs interval.**

TABLE 4. **Confusion matrix of stacked AE architecture.**

Actual vs.Predicted	TL-1	TL-2	TL-3	TL-4
TL-1	**1628**	0	164	0
TL-2	55	**1587**	127	0
TL-3	69	47	**1679**	0
TL-4	28	1	6	**1859**



(a) *CNN Model Accuracy* (b) *CNN Model Loss*

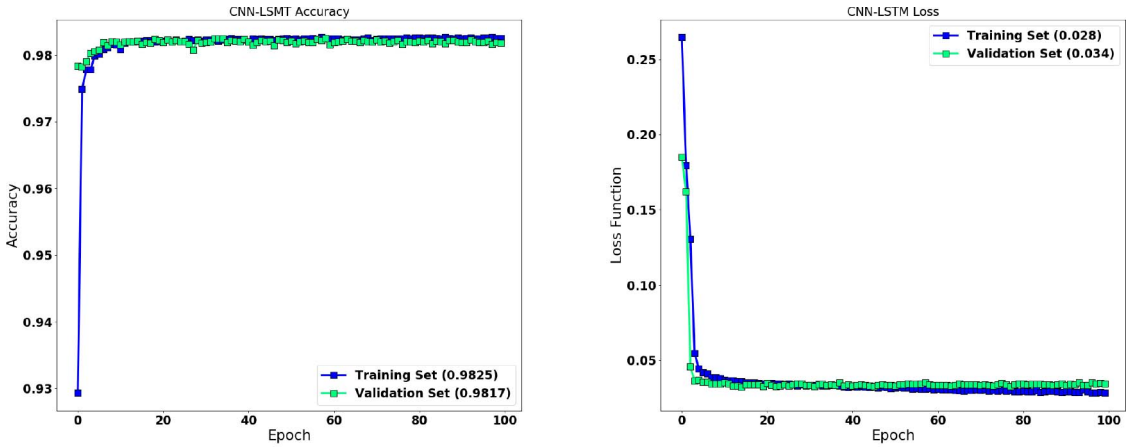
FIGURE 9. **(From Left to Right): In this experiment we illustrate the training/validation loss and classification accuracy of the proposed CNN Model. As we may notice, in terms of the classification accuracy, the proposed technique achieves high performance in both validation and training phases. Additionally, the loss function is constantly reduced in both training and validation stages.**

architecture for the threat-level detection of MISP network-traffic measurements. Specifically, we trained our CNNs using approximately $32,000$ examples, and thus we further validated the CNN model using $8,000$ examples. The exploited CNN architecture is composed of 5 CNN hidden layers, each one followed by a Max-pooling and a Dropout layer, for eliminating gradient's fluctuations. The Dropout layer's parameter was set into 0.2 in all layers, while the pool size was fixed in 1. Moreover, the number of hidden nodes was set to $(\mathbf{64}, \mathbf{64}, \mathbf{32}, \mathbf{32}, \mathbf{64})$, and the filter size into 3. For the CNN layers, we choose the RELU activation function, while the classification layer utilizes the *Softmax* function.

Figure 9 presents the loss function and classification accuracy of the proposed CNN scheme, when applied on the four-threat level classification scenario. After 100 epochs, the training loss is fixed into $\mathbf{0.0538}$, while the validation loss into $\mathbf{0.0497}$. Despite the loss function presents moderate fluctuations, we observe that it is continuously reducing within the 100 iterations interval, validating the convergence of

TABLE 5. Confusion matrix of CNN architecture.

Actual vs.Predicted	TL-1	TL-2	TL-3	TL-4
TL-1	**1765**	0	27	0
TL-2	32	**1607**	130	0
TL-3	20	48	**1727**	0
TL-4	23	4	8	**1859**



(a) *CNN-LSTM Model Accuracy* (b) *CNN-LSTM Model Loss*

FIGURE 10. (From Left to Right): In this figure we highlight the high-quality classification performance achieved by the proposed CNN-LSTM architecture, in terms of the training/validation accuracy and loss. As we may observe, the classification accuracy stabilizes into its highest value after only few epochs, while simultaneously the loss function is constantly reduced in both training and validation stages.

TABLE 6. Confusion matrix of CNN-LSTM architecture.

Actual vs.Predicted	TL-1	TL-2	TL-3	TL-4
TL-1	**1427**	0	365	0
TL-2	19	**1642**	105	3
TL-3	2	59	**1730**	4
TL-4	3	6	19	**1866**

the proposed scheme. On the other hand, the classification accuracy in the training set converges significantly fast, after almost 40 epochs, while the validations accuracy presents small variations but with simultaneously preserving its high value. The training accuracy after 100 epochs is **98**.**12**%, and the validation accuracy **97**.**53**%. As we may observe, the CNN model built with the aforementioned network architecture achieves high classification accuracy, validating the high quality of the proposed scheme. Additionally, Table 5 provides the Confusion Matrix of the proposed CNN architecture with respect to each investigating category (i.e., threat level), where we illustrate the distribution between the actual and predicted threat levels. In terms of Precision metric, the proposed CNN architecture achieves: **96**% for the TL-1: Sophisticated APT-Malware or 0-day attack, **97**% for the TL-2: APT-Malware, **91**% for the TL-3: Mass-malware, and **100**% for the TL-4: Undefined attack type. All results are summarized, and compared with our other constructed architectures in Table 7. Consequently, the proposed CNN-based scheme also achieves high quality threat-level predictions.

Finally, we exploited a CNN-LSTM architecture towards the problem of threat-level identification for network-traffic intrusion detection. For this purpose, we considered four 1-D CNN layers, followed by Max-pooling and Dropout layers, with (**64**, **64**, **32**, **32**) hidden nodes. The sub-sequent LSTM architecture is composed of 5 hidden layers and (**64**, **64**, **32**, **32**, **16**)-hidden nodes. The filter size for the CNN structure was set into 3, while the dropout value was fixed into 0.2. The activation function in all layers, except the output (i.e., *Softmax*-classification) is the RELU.

Figure 10 illustrates the classification accuracy and loss-function with respect to the number of iterations. In this simulation, we have also fixed the number of epochs into 100. The classification accuracy of both training and validation sets converges into a stationary point after approximately 20 iterations, validating the high-performance of the proposed CNN-LSTM technique. Specifically, the classification accuracy for the training set was fixed into **98**.**25**%, while for the validation set into **98**.**17**%. Additionally, the loss-functions of the training and validation datasets, after approximately 40 iterations, stabilize into its lowest value, of **0**.**028** for the training set, and **0**.**034** for the validation set. Moreover, Table 6 illustrates the Confusion Matrix of the proposed CNN-LSTM scheme, where we illustrate the ground truth versus the predicted results per each category (i.e., threat level). In all cases, the proposed scheme

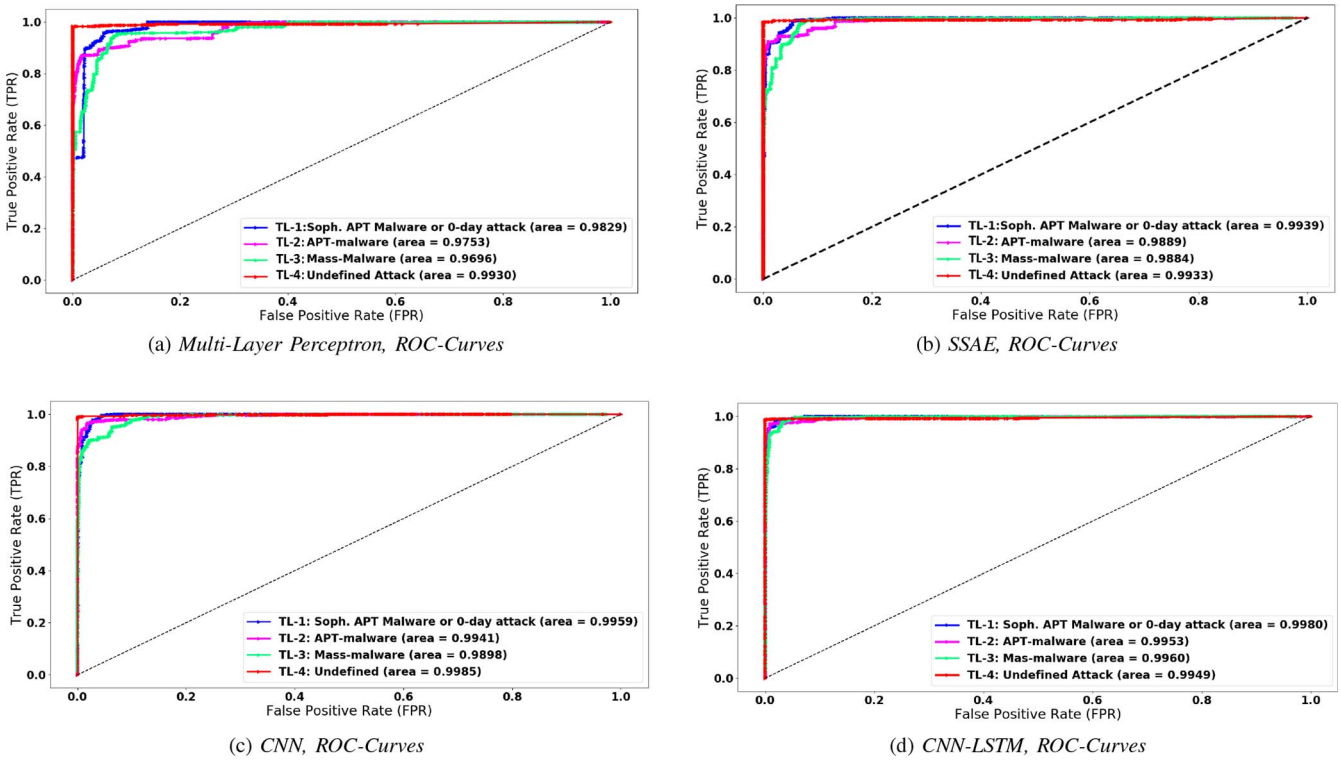


FIGURE 11. AUC-ROC Curves for the four comparable deep learning architectures. As we may observe all architectures depict high quality ROC curves, validating the high-accuracy of the proposed classification/prediction approaches.

TABLE 7. Quantitative performance evaluation of the developed methods in terms of precision, recall and F-1 score metrics.

Methods	MLP			Stacked SAE			CNN			CNN-LSTM		
Metrics	Precision	Recall	Metrics	Precision	Recall	F-1 Score	Precision	Recall	F-1 Score	Precision	Recall	F-1 Score
TL-1 (Sophisticated APT-malware or 0-day attack)	0.84	0.96	0.90	0.91	0.91	0.91	0.96	0.98	0.97	0.98	0.80	0.88
TL-2 (APT-malware)	0.95	0.85	0.90	0.97	0.90	0.93	0.97	0.91	0.94	0.96	0.93	0.94
TL-3 (Mass-malware)	0.86	0.85	0.85	0.85	0.94	0.89	0.91	0.96	0.94	0.88	0.96	0.86
TL-4 (Undefined Attack)	1.00	0.98	0.98	1.00	0.98	0.99	1.00	0.98	0.99	1.00	0.98	0.99

achieves high quality predictions. In further detail, in terms of the Precision metric, the proposed CNN-LSTM scheme achieves: **98**% for the TL-1: Sophisticated API-malware or 0-day attack, **96**% for the TL-2: APT malware, **88**% for the TL-3: Mass-malware, and **100**% for the TL-4: Undefined attack type.

Comparing the four deep feature learning approaches (i.e., Multi-Perceptron Model, SSAE, CNN, and CNN-LSTM), in terms of the classification accuracy on the validation set, we may notice that the highest accuracy for the four-class threat level scenario of MISP's network-traffic dataset, is achieved first by the CNN-LSTM scheme (**98**.**17**%), followed by the CNN architecture (**97**.**53**%), then by the SSAE model (**96**.**49**%), and finally by the Multi-Layer Perceptron model (**95**.**32**%). Finally, Figure 11 illustrates the ROC-AUC curves/per category for the four comparable techniques, while Table 7 highlights and summarizes the Precision, Recall and F-1 scores of the proposed architectures. Regarding the AUC-ROC curves, we observe that all comparable deep feature learning approaches depict high quality results. To be more precise, the best score (i.e., **99**.**80**% for TL-1, **99**.**53**% for TL-2, **99**.**60**% for TL-3, and **99**.**49**% for TL-4) was achieved using the CNN-LSTM architecture, validating our claim that the specific scheme outperforms the comparable deep learning approaches for the problem of MISP network measurements threat level detection.

VI. CONCLUSION

In this paper we presented an end-to-end novel Information Sharing Prototype (I2SP), able to gather, process, and distribute information regarding network-traffic events.

Specifically, we exploited the architecture of the MISP interface for gathering network measurements, and thus we proposed four challenging machine learning formulations that learn internal representations from the multi-variate time-series dataset. The ultimate of this study is to perform intrusion detection, by learning the threat level of each upcoming measurement. All comparable techniques present high-quality classification results in terms of the accuracy and the error of the loss function.

REFERENCES

[1] C. Manikopoulos and S. Papavassiliou, "Network intrusion and fault detection: A statistical anomaly approach," *IEEE Commun. Mag.*, vol. 40, no. 10, pp. 76–82, Oct. 2002.

[2] M. Burgess, "Probabilistic anomaly detection in distributed computer networks," *Sci. Comput. Program.*, vol. 60, no. 1, pp. 1–26, 2006.

[3] C. Wagner, A. Dulaunoy, G. Wagener, and A. Iklody, "MISP: The design and implementation of a collaborative threat intelligence sharing platform," in *Proc. ACM Workshop Inf. Sharing Collaborative Security*, 2016, pp. 49–56.

[4] W. Tounsi and H. Rais, "A survey on technical threat intelligence in the age of sophisticated cyber attacks," *Comput. Security*, vol. 72, pp. 212–233, Jan. 2018.

[5] CSIRT Gadgets. (2016). *CIF V3*. [Online]. Available: https://csirtgadgets.com

[6] M. Bernier and A. Magar. (2015). *Soltra Edge Open Cyber Intelligence Platform*. [Online]. Available: http://cradpdf.drdc-rddc.gc.ca/PDFS/unc196/p802346_A1b.pdf

[7] S. Barnum, *Standardizing Cyber Threat Intelligence Information With the Structured Threat Information Expression (STIX)*, Miter Corporat., McLean, VA, USA, 2012.

[8] *Threatintelligence*. Accessed: Mar. 24, 2020. [Online]. Available: https://www.threatintelligence.com/

[9] *Alliacert*. Accessed: Mar. 24, 2020. [Online]. Available: https://www.alliacert.com

[10] N. Görnitz, M. Kloft, K. Rieck, and U. Brefeld, "Toward supervised anomaly detection," *J. Artif. Intell. Res.*, vol. 46, pp. 235–262, Jan. 2014.

[11] S. Ahmad, A. Lavin, S. Purdy, and Z. Agha, "Unsupervised real-time anomaly detection for streaming data," *Neurocomputing*, vol. 262, pp. 134–147, Nov. 2017.

[12] V. Filimonov *et al.*, "Unsupervised anomaly detection for arbitrary time series," U.S. Patent 9 652 354, May 16, 2017.

[13] S. Akcay, A. Atapour-Abarghouei, and T. P. Breckon, "GANOMALY: Semi-supervised anomaly detection via adversarial training," in *Proc. Asian Conf. Comput. Vis.*, 2018, pp. 622–637.

[14] L. Ruff *et al.*, "Deep semi-supervised anomaly detection," 2019. [Online]. Available: arXiv:1906.02694.

[15] H. Song, Z. Jiang, A. Men, and B. Yang, "A hybrid semi-supervised anomaly detection model for high-dimensional data," *Comput. Intell. Neurosci.*, vol. 2017, p. 9, Nov. 2017. [Online]. Available: https://doi.org/10.1155/2017/8501683

[16] S. M. Erfani, S. Rajasegarar, S. Karunasekera, and C. Leckie, "High-dimensional and large-scale anomaly detection using a linear one-class SVM with deep learning," *Pattern Recognit.*, vol. 58, pp. 121–134, Oct. 2016.

[17] W. Han, J. Xue, and H. Yan, "Detecting anomalous traffic in the controlled network based on cross entropy and support vector machine," *IET Inf. Security*, vol. 13, no. 2, pp. 109–116, 2019.

[18] A. P. Muniyandi, R. Rajeswari, and R. Rajaram, "Network anomaly detection by cascading *k*-means clustering and C4.5 decision tree algorithm," *Procedia Eng.*, vol. 30, pp. 174–182, Mar. 2012.

[19] C. Aytekin, X. Ni, F. Cricri, and E. Aksu, "Clustering and unsupervised anomaly detection with L2 normalized deep auto-encoder representations," in *Proc. IEEE Int. Joint Conf. Neural Netw. (IJCNN)*, 2018, pp. 1–6.

[20] H. Zenati *et al.*, "Efficient GAN-based anomaly detection," 2018. [Online]. Available: arXiv:1802.06222.

[21] Y. Intrator, G. Katz, and A. Shabtai. (2018). *MDGAN: Boosting Anomaly Detection Using—Multi-Discriminator Generative Adversarial Networks*. [Online]. Available: https://arxiv.org/abs/1810.05221.

[22] D. Li, D. Chen, B. Jin, L. Shi, J. Goh, and S.-K. Ng, "MAD-GAN: Multivariate anomaly detection for time series data with generative adversarial networks," in *Proc. Int. Conf. Artif. Neural Netw.*, 2019, pp. 703–716.

[23] P. Malhotra, L. Vig, G. Shroff, and P. Agarwal, "Long short term memory networks for anomaly detection in time series," Presses universitaires de Louvain, Ottignies-Louvain-la-Neuve, Belgium, 2015, p. 89.

[24] L. Bontemps, J. McDermott, and N.-A. Le-Khac, "Collective anomaly detection based on long short-term memory recurrent neural networks," in *Proc. Int. Conf. Future Data Security Eng.*, 2016, pp. 141–152.

[25] D. Kwon, K. Natarajan, S. C. Suh, H. Kim, and J. Kim, "An empirical study on network anomaly detection using convolutional neural networks," in *Proc. IEEE 38th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, 2018, pp. 1595–1598.

[26] R. Chalapathy and S. Chawla, "Deep learning for anomaly detection: A survey," 2019. [Online]. Available: arXiv:1901.03407.

[27] A. Chawla, B. Lee, S. Fallon, and P. Jacob, "Host based intrusion detection system with combined CNN/RNN model," in *Proc. Joint Eur. Conf. Mach. Learn. Knowl. Disc. Databases*. 2018, pp. 149–158.

[28] Y. Meidan *et al.*, "N-BaIot—Network-based detection of IoT botnet attacks using deep autoencoders," *IEEE Pervasive Comput.*, vol. 17, no. 3, pp. 12–22, Jul.–Sep. 2018.

[29] A. Borghesi, A. Bartolini, M. Lombardi, M. Milano, and L. Benini, "Anomaly detection using autoencoders in high performance computing systems," in *Proc. AAAI Conf. Artif. Intell.*, vol. 33, 2019, pp. 9428–9433.

[30] Q. P. Nguyen, K. W. Lim, D. M. Divakaran, K. H. Low, and M. C. Chan, "GEE: A gradient-based explainable variational autoencoder for network anomaly detection," 2019. [Online]. Available: arXiv:1903.06661.

[31] M. Zaharia *et al.*, "Apache spark: A unified engine for big data processing," *Commun. ACM*, vol. 59, no. 11, pp. 56–65, 2016.

[32] G. Wang *et al.*, "Building a replicated logging system with Apache Kafka," *Proc. VLDB Endow.*, vol. 8, no. 12, pp. 1654–1655, 2015.

[33] X. Meng *et al.*, "MLLIB: Machine learning in Apache spark," *J. Mach. Learn. Res.*, vol. 17, no. 1, pp. 1235–1241, 2016.

[34] K. Potdar, T. S. Pardawala, and C. D. Pai, "A comparative study of categorical variable encoding techniques for neural network classifiers," *Int. J. Comput. Appl.*, vol. 175, no. 4, pp. 7–9, 2017.

[35] M. Schmidt. (2005). *Least Squares Optimization with L1-Norm Regularization*. Accessed: Mar. 24, 2020. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/versions?doi=10.1.1.186.3602

[36] *Redis Server*. Accessed: Mar. 24, 2020. [Online]. Available: https://oss.redislabs.com/redisai/

[37] A. K. Jain, J. Mao, and K. M. Mohiuddin, "Artificial neural networks: A tutorial," *Computer*, vol. 29, no. 3, pp. 31–44, 1996.

[38] A. Ng. *Sparse Autoencoder*. Accessed: Mar. 24, 2020. [Online]. Available: https://web.stanford.edu/class/cs294a/sparseAutoencoder_2011new.pdf

[39] R. Girshick, "Fast R-CNN," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2015, pp. 1440–1448.

[40] F. A Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: Continual prediction with LSTM," *Neural Comput.*, vol. 12, no. 10, pp. 2451–2471, 2000.

[41] S. K. Pal and S. Mitra, "Multilayer perceptron, fuzzy sets, and classification," *IEEE Trans. Neural Netw.*, vol. 3, no. 5, pp. 683–697, Sep. 1992.

[42] S.-I. Horikawa, T. Furuhashi, and Y. Uchikawa, "On fuzzy modeling using fuzzy neural networks with the back-propagation algorithm," *IEEE Trans. Neural Netw.*, vol. 3, no. 5, pp. 801–806, Sep. 1992.

[43] Z. Tüske, M. A. Tahir, R. Schlüter, and H. Ney, "Integrating Gaussian mixtures into deep neural networks: Softmax layer with hidden variables," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, 2015, pp. 4285–4289.

[44] R. Gribonval and M. Nielsen, "Sparse representations in unions of bases," *IEEE Trans. Inf. Theory*, vol. 49, no. 12, pp. 3320–3325, Dec. 2003.

[45] A. G. de G Matthews, J. Hensman, R. Turner, and Z. Ghahramani, "On sparse variational methods and the Kullback–Leibler divergence between stochastic processes," in *Proc. Artif. Intell. Stat.*, 2016, pp. 231–239.

[46] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.

[47] H. Ide and T. Kurita, "Improvement of learning for CNN with ReLU activation by sparse regularization," in *Proc. IEEE Int. Joint Conf. Neural Netw. (IJCNN)*, 2017, pp. 2684–2691.

[48] Z. Zhang, "Model building strategy for logistic regression: Purposeful selection," *Ann. Transl. Med.*, vol. 4, no. 6, p. 111, 2016.

[49] W. Zaremba, I. Sutskever, and O. Vinyals, "Recurrent neural network regularization," 2014. [Online]. Available: arXiv:1409.2329.

[50] M. Sokolova, N. Japkowicz, and S. Szpakowicz, "Beyond accuracy, F-score and ROC: A family of discriminant measures for performance evaluation," in *Proc. Aust. Joint Conf. Artif. Intell.*, 2006, pp. 1015–1021.