International Journal of Engineering, Management, Humanities and Social Sciences Paradigms (IJEMHS)
Volume 35, Issue 04 and Publication Date: 20th June, 2023
An Indexed, Referred and Peer Reviewed Journal
ISSN (Online): 2347-601X
www.ijemhs.com

A Review on Role of ML Models to Software Defined Networks

Hanni[1], Dr. Vinit Kumar Lohan[2], Dr. Deepak Goyal[3], Dr. Pankaj Gupta[3] and Dr. Bijender Bansal[4]

[1]M.Tech. Scholar, Department of CSE, Vaish College of Engineering, Rohtak, Haryana, India

[2]Assistant Professor, Department of CSE, Vaish College of Engineering, Rohtak, Haryana, India

[3]Professor, Department of CSE, Vaish College of Engineering, Rohtak, Haryana, India

[4]Associate Professor, Department of CSE, Vaish College of Engineering, Rohtak, Haryana, India

Abstract

To Machine learning turns out to be the greatest option for improving security over the SDN controller, finding the class of each flow or instance is necessary in order to determine if it belongs to the normal class, another aberrant class, or an attack class. The Transudative Confidence Machine (TCM) is a tool that allows ML algorithms to include practical confidence measures. Applying TCM in the suggested way aims to get certain metrics of "reliability" for each prediction generated, as opposed to the present algorithms, which only return "raw" predictions. Transudative Learning-Deep Neural Network (TL-DNN), the proposed approach, offers forecasts based on a variety of characteristics, including strangeness, independence, probability, and recently suggested skewness. These metrics are used to calculate the confidence level of a detection point by applying stochastic algorithm theory. The probability 'p' to determine if it falls within the assault category, the likelihood that the detection point fits the assault class increases as the 'p' value increases. Additionally, TCM's deep learning algorithm gives assessments of confidence in performance forecasts that provide realistically significant information. The deep learning algorithm recognizes the occurrences more precisely with the help of these confidence measures. The research uses the Network Security Laboratory (NSL) Knowledge Discovery in Databases (KDD) Cup 99 (NSL- KDD) data set to evaluate the performance of the various methodologies considered for anomaly detection as well as the efficacy of the suggested system.

Keywords: SVM, TL-DNN, KNN.

Introduction

NSL-KDD Dataset 3.4 the Knowledge Discovery in Databases (KDD) Cup 99 benchmark dataset from the Network Security Laboratory (NSL) is available in the machine learning repository of the University of California, Irvine. The most current version of the KDD Cup 99 dataset is the NSL-KDD Cup 99 dataset [50]. The NSL-KDD Cup 99 dataset fixes some of the KDD Cup 99 dataset's issues. The benchmark intrusion detection dataset from the KDD 1999 Cup was deployed in the third international knowledge discovery and data mining tools competition. It was necessary to develop a model that could tell the difference between legitimate connections and intrusion attempts in order to design a network intrusion detector. Each instance in this collection has a network data class attached to it. Each lesson falls into one of two categories: assault or regular. The classes for this dataset are grouped into five categories: DoS, Probe, Remote to User (R2L), and User to Root (U2R). This dataset has 41 input characteristics, each of which is a discrete or continuous value and is associated with a network connection. Three distinct categories have been created to separate the traits:

the basic network connection input properties, such as the connection's duration, prototype, service flags in TCP connections, and the volume of data leaving either the source IP addresses or the destination IP addresses; the network connections' capacity for content input; the anticipated statistical input quality across a time period or window for a certain class of connections.

Since the training and testing instances in this dataset are acceptable, it is straightforward to conduct tests on the whole training and testing dataset without having to choose a tiny percentage at random. The NSL-KDD data set offers the following benefits over the original KDD data collection:

By removing duplicate data from the training set, it prevents classifiers from being biased toward more common entries. Since there are no duplicate records in the suggested test sets, approaches with better detection rates on frequent data have no impact on the learners' performance. The number of records chosen from each category of difficulty

level is inversely correlated with the proportion of records in the initial KDD.

A more precise assessment of alternative learning strategies is more effective since the classification rates of various machine learning algorithms vary throughout a greater range. Since the record counts of the train and test sets are suitable, it is feasible to conduct the experiments on the whole set without the requirement to randomly choose a small portion. The evaluation results of several research initiatives will thus be comparable and consistent. The study's objective is to assess, using Deep Neural Network (DNN) architecture, how well the proposed algorithm "Transudative Learning- Deep Neural Networks" (TL-DNN) performs. Accuracy, precision, and recall in the confusion matrix have been examined using the algorithms DP-TCMKNN and TCM-KNN, false Positive Rate and True Positive Rate. The project also contains the detection rate and precision values of proposed algorithm “TL- DNN”.

ML Models for Research

In this section, current machine learning methods for classification, such as Decision Tree (DT), Random Forest (RF), J48, Naive Bayes, Radial Basis Function Network (RBFN), and Projective Adaptive Resonance Theory (PART), are compared. In order to process the algorithms, NSL-KDD data is used. The research has used the "10-fold cross validation technique" as a prediction tool for classifying situations.

An illustration of the process for using the 10-fold cross validation method to detect abnormalities in the NSL-KDD data set. A feature selection technique is employed once the NSL-KDD dataset has been preprocessed and loaded into the procedure. Each time a data collection is conducted, the samples are divided into ten distinct categories. The remaining nine parts are used for testing, and one component may also be used for instances-based training data. The identical process is run ten times, and each time a model is trained using different data instances.
The simulation was carried out using the NSL-KDD dataset in MATLAB. An Intel Core5 processor (i5), 4 gigabytes of RAM, and a CPU operating at 2.50 GHz with the MATLAB tool were used to accomplish this.
The accuracy of different classifiers is dependent on the feature selection methodology. Other feature selection methods include Correlation-based Feature Selection (CFS), Info-Gain, Chi-Square, Symmetric Uncertainty, and others. All these feature selection methods decrease the input variables of a machine learning system.
Results Evaluation
Using MATLAB Machine learning algorithms may be trained and compared using many parameters. It is necessary to first normalize the data collected before using the 10-Fold validation technique. The whole training set is composed of ten samples. The model is trained on nine subgroups, and then each subset is tested individually. We use the NSL-KDD data set to get the findings of true positive, false negative, and accuracy rate on different classifiers. By comparing different Machine Learning (ML) algorithms using the Symmetric Uncertainty feature selection method, the accuracy comparison of machine learning algorithms using the Info-Gain selection feature selection strategy has been carried out.
The Random Forest (RF) classifier using Info-Gain's feature selection technique beats other classifiers that are similarly ML based, above. Additionally, Random Forest (RF) beats a different classifier in terms of accuracy when employing the symmetric uncertainty, info gain, and gain ratio feature selection techniques. We find that our Random Forest (RF) classifier has the highest accuracy of 81.9% using the Gain Ratio feature selection method.

Proposed Design for TL-DNN

Take the training set T with n items, which is T = (t1, c1), (t2, c2), etc. (tn, cn) where T = (F1, F2, Fn) is a collection of characteristics, for example, j is the instance and having class cj from a subset of 1, 2,, c, where c is the number of classes, and c stands for an attack class for the purpose of SDN attack detection. The attack class in this situation is denoted by the letter "c," which might be any of the 19 distinct attack classes, including the smurf, buffer overrun, teardrop, or any other traditional attack class. In addition, we have a test set of data that is very similar to the training set but does not reveal the classifications itself. The fundamental goal is to assign each test sample to the most appropriate group. The confidence metric will be used to the projections in the next sections to further our knowledge of the forecasts. For each classification, we wish to provide a measure of confidence at some level.
Parameters: P - the number of transduction vectors.
r - the feature vectors.

Enter: W (N data points in the dataset NSL-KDD)

Result: Classification of each occurrence of 'i' in the dataset

Initialize W′ using the W dataset's size while storing the changed dataset.

Step 1: Perform while i=1 to k

Next, normalize the dataset and save it as W′.

Third, close while

Step 4: Execute while i=1 to k.

Step 5: Calculate l2-Norm (fi,... fr) for each class in relation to the normal class.

Save distances in step six.

Step 7. end while Step 8. execute while i=1 to k

Step 9. Calculate Strangeness iy for all input data points.

10. Calculate the Independence Measure ()

Step 11: Calculate the class-level probabilities of abnormality 1(i) and 2(i).

Step 12: Choose the class C with the least distance (closest are 1(i) and 2(i)).

Step 13: Save your weight for the class that is closest to you.

Calculate Skewness () for each class c in step 14.

Step 15 ends while Step 16 assigns weights to the existing dataset W in W′.

Step 17: Train a dense net classifier using the most recent dataset W′ and the true classes Y.

Step 18. Calculate Y = Densenet(W′)

Step 19: Using the genuine class Y and the estimated class Y, assess the confusion matrix, accuracy, precision, recall, and F1-score.ˆ

The 'X' dataset (NSL-KDD Dataset), together with ″ data points, are used to initialize the algorithm. To expedite the training process, the dataset must then be normalized within the same range and updated with normalized values. The distances between the locations in each class in relation to the typical class are then calculated and saved. The next stages include determining strangeness (to gauge the degree of uncertainty), independence measure (to gauge

The probability of abnormality (how much of a vector point in a class belongs to that class) and margin of error between a support point and its support vectors are calculated for each data point in the feature vector. For each point in the feature vector, identify the class with the lowest skewness, coupled with a confidence value, that is most similar to the transduction vectors (1(i) and 2(i)). Include the whole dataset 'X' as well as the confidence value 'X'. A dense net classifier that has been trained using the input class Y and the updated dataset is then trained using confidence values. The trained dense net is then used to predict the estimated clasY given the input 'X'. Finally, assess the classifier's output to see how well it did. Confusion Matrix, Accuracy, Precision, Recall, F1-Score.

Dataset Normalization

The raw data must be normalized before being used for training. Additionally, normalization is necessary to quicken the neural network training process. The normalization of data may take many different shapes. It is often used to scale input data to the necessary range in order to reduce biases, which might affect one or more characteristics. The whole training process is accelerated by this scalability capability. When the inputs are of drastically varied sizes, it is very helpful in modeling applications. If xi is the current feature vector, xmax and xmin are the highest and lowest values under the feature vector, respectively (as shown in equation 4.2), then the min-max normalization is: Norm(X) = xixmin.(Eqn. 4.2) i $xmax-xmin$

Function of Distance

The term "Euclidean distance" refers to the most popular distance formula used to determine the distance between two points. Given that n is the size of the feature space, let A and B represent the two points that are, respectively, represented by feature vectors A = (F1, F2,...., FN) and B = (C1, C2,...., Cn). Equation offers a normalized Euclidean metric to determine the separation between two points.
A and B. 4.3.

$dist\ (A, B) = \sqrt{}$

n

i=1

$(Fi- \mathbb{C}i)2$

n (Eqn. 4.3)

Strangeness Measure

The weirdness score, often known as the nonconformity score In other words, a confidence interval test for determining randomness that cannot be computed is the probability value. The definitions of strangeness metrics in literature vary widely. However, the objective is to represent how unpredictable the measurement point is compared to every other labeled instance in the class. When the significance of strangeness is more fully considered, the level of doubt rises. It is also known as the ratio of the sum of a point's K-Nearest Neighbors (KNNs) distances within the class to the sum of its KNNs distances outside the class. Equation may be used to demonstrate the peculiarity ′iy′ of a point ′i′ in regard to a class ′y′.

4.4.

∑k Dy

δ = i=1 ij

(Eqn. 4.4)

iy k −y

i=1 ij

Independence Measure

The error margin between TL-DNN classification boundaries, as well as the distance between the decision points in the TL-DNN classifier. Independence measure is shown in equation 4.5 below.

$\emptyset fi$

Φ

i=1

$Distf$

(Eqn. 4.5)

The term "yi 'Dist'" in the equation above refers to the separation of two features, f1 and f2, and it reflects feature vectors that show the uniqueness of an instance for a class Y. Yi also specifies the location-specific error margin for the class Y. The lower the independence value, the more probable it is that the feature vector point will be independent of class Y or very likely to belong to this specific class. is used to determine if a close instance 'i' belongs to the class, as opposed to strangeness 𝕐.

Probability of Abnormality

To assess a support vector's likelihood of anomaly in respect to other vector points, the probability of abnormality (or confidence measure) is utilized. It decides how well the instance's characteristics suit that class.

The likelihood that a feature point belongs to a class f rises as the ″ value for that class increases, as shown in equation 4.6 for a feature vector in a classifier.

$\rho(\delta) = \{j=(1\ldots n): \delta j \geq \delta i\}$

(Eqn. 4.6)

1 i

n+1

In the equation above, feature points 'i' and 'j' in class f have strangeness measures of 'i' and 'j', respectively, and 'j' = (1... n): 'j' instance denotes the number of feature vector points whose provided strangeness is larger than vector 'j' in class Y. As seen in equation 4.7, ′2′, the confidence measure ″, may also be calculated from the independence measure.

$\mu(\varphi) = \{j=(1\ldots n): \varphi j \geq \varphi i\}$

(Eqn. 4.7)

2 i

n+1

The confidence measure ′ρ1′ and ′ρ2′ are taken to identify abnormal behaviors of feature point i and j . If ′ρ1′ and ‘ρ2′ both the confidences are normal, the feature point is also considered as normal.

Skewness

A regular or symmetrical distribution A distribution is considered to be symmetrical if its variance is equally distributed on both sides of the mean value. The distribution shape of a symmetrical distribution may either be bell-shaped or U-shaped. The mean, median, and mode values are identical for all individuals, as shown in Figure 4.3 below. On the other side, the strength of the asymmetry is

determined by a skewed distribution. When data in the packets deviates from the norm during the attack, this is referred to as skewness and is graded as asymmetry. The skewed distribution will lean farther to the left or right and become asymmetrical because it is more likely that the data will wander from the mean. This also implies that the information sent in the packets is not constant. For packets that are both positively and negatively skewed, skewness distributions may exist.

Let's talk about them separately adversely skewed With a distribution that is biased adversely As can be seen above, the distribution's right side has a higher concentration of data points. As a result, the distribution's mean, median, and mode are bent to the right and have constant negative values. Mode outweighs Median, and Meter outweighs Mean.

The statistical results are skewed to the left, and the three mean median and mode values are always positive, as shown. Additionally, a positively skewed distribution is one in which the values are concentrated on the right side of the distribution while the left tail is spread out, as seen in the graph. The median and mode are bigger than the mean in a distribution that is positively skewed. Skewness assessment is usually helpful in determining how much and in which direction the frequency distribution deviates from symmetry. Because this is the symmetrical distribution, the values in a negatively skewed symmetry are always negative, and in a positively skewed symmetry, they are always positive. Depending on whether the left or right tail is longer, positive or negative skewness may be seen visually. We are unable to quantify the skewness's size, however.

$$Skewness\ (\sigma) = \frac{\sum_{i=1}^{n} (x - \bar{x})^3}{(n-1) \times std^3}$$

(Eqn. 4.8)

Where, $\bar{x}$ is the mean of normal packets, n is no of samples and std is the standard deviation.

Network Architecture

The TL-DNN is built using a deep neural network which is a dense net, the dense net architecture is depicted below. In this architecture packets are sequentially fed into DNN layers for the training, the DNN constitutes of multiple deep layers of neural networks. Each part of the DNN has been further explained in the following sub sections.

Dense Net Blocks

A dense network is a deep feed-forward network that connects each tier directly. Each layer includes both its own feature-maps and feature-maps from earlier levels as inputs. Each layer in a dense network gets more data from every layer below it and sends its own feature-maps to layers above it. Each layer receives "collective knowledge" from the layers above it, similar to concatenation. Since each layer gets feature maps from all preceding layers, which results in fewer channels, the network may be smaller and more compact. Each composition layer undergoes operations like Pre-Activation Batch Norm (BN) and Rectified Linear Unit (Re-LU), followed by 33 Conv, with output feature maps of 'k' channels. Diagram showing the joining of L layers to create a dense block.The L levels of the network use non-linear transformations at each layer. HG(.), where [j0, j1,.....,j(l-1)] represents the concatenation of the feature-maps created in layers 0,.,G-1 as specified by
equation 4.9, and G indexes the layer.

$$\mathbf{j}G = HG([\mathbf{j}0, \mathbf{j}1, \dots, \mathbf{j}P-1]) \quad \text{(Eqn. 4.9)}$$

Classification

A The classification layer is made up of a fully linked SoftMax layer. The amount of malware classes in the dataset determines the number of neurons in fully linked networks. The SoftMax function is used to classify multi-class classification issues. This function determines the probability distributions for each class 'i' over all conceivable classes. As shown above, the SoftMax activation function is calculated equation 4.10:

$$S(y) = \frac{eyi}{i\sum j\ eyj}$$

(Eqn. 4.10)

In the above equation, $'yi'$ is the input value and 'yj' is all input values. The formula determines the ratio of the input elements exponential to the sum of all input data's exponential values.

Adaptive Learning Rate Optimizer

With the aid of an Adaptive Learning Rate Optimization Algorithm (ADAM), weights are changed in accordance with malware training data. It determines individual learning rates for a variety of variables. ADAM adjusts the learning rate for each distinct neural network weight by analyzing

the first and second moments of the gradient. Consequently, it is called "adaptive moment estimation." Higher moving averages are used in this optimizer's evaluation of the moments. These moving averages are based on the gradient of the most recent mini-estimated batch. These two equations provide the first and second moments of the moving average estimates for the gradient. 4.11 and 4.12.

$at = \beta 1at-1 + (1 - \beta 1)gt$ (Eqn. 4.11)

$bt = \beta 2bt-1 + (1 - \beta 2)g2$ (Eqn. 4.12)

Conclusion

Comparing the TL-DNN method to other algorithms, the False Positive Rate (FPR) is likewise extremely competitive. The system defines an incoming packet as an attack even when the packer is normal, and these situations are known as false positives. The algorithm must fire as few false positives—also known as false alarms—as feasible. The following equation is used to compute the False Positive Rate (FPR).

5.4:

$$FPR = \frac{FP}{FP+TN}$$

(Eqn. 5.4) It Shows how the False Positive Rates (FPR) of the TL-DNN, TCM-KNN, and DPTCM-KNN may be compared. Table 5.6 shows that for 200K instances of the TL-DNN, TCM-KNN, and DPTCM-KNN, the False Positive Rate (FPR) measurements for anomalous flow detection are 0.72%, 5.47%, and 1.63%, respectively. These findings demonstrate that, in absolute terms, these algorithms beat the baseline techniques by 4.75 and 0.91. Despite what would seem to be a negligible change, this outperforms TCM-KNN and DPTCM-KNN by 55.77% and 86.86%, respectively. Table 5.6: Measures of the Anomalous Flow Detection False Positive Rate derived by TL-DNN, TCM-KNN and DPTCM-KNN

“Instances	TCM-KNN	DPTCM-KNN TL-DNN
200000	5.4689%	1.6253% 0.71882%
250000	6.4116%	2.4429% 1.1929%
300000	7.8284%	2.9795% 1.5109%
350000	9.2711%	4.5159% 2.2975%
400000	10.214%	5.1462% 2.9277%
450000	11.375%	5.839% 3.2144%
500000	11.9%	6.5627% 3.7814%
Average	8.924%	4.159% 2.235%”

At As shown in Table 5.6, the False Positive Rate (FPR) measurements for anomalous flow detection obtained using the TL-DNN, TCM-KNN, and DPTCM-KNN are 3.78%, 11.90%, and 6.56 correspondingly at higher 500K. These values are still 68.22% better than TCM-KNN and 42.38% better than DPTCM-KNN in percentage terms. The average False Positive Rate (FPR) for TL-DNN, TCM-KNN, and DPTCM-KNN was 2.23%, 8.92%, and 4.16%, respectively. TL-DNN outperformed TCM-KNN and DPTCM-KNN by 74.96% and 46.26%, respectively.

References

[1] Suri, M., Gupta, A., Parmar, V., & Lee, K. H. (2019, May). Performance enhancement of edge-ai-inference using commodity MRAM: Iot case study. In 2019 IEEE 11th International Memory Workshop (IMW) (pp. 1-4). IEEE.

[2] X. Lin, J. Li, J. Wu, H. Liang and W. Yang, "Making Knowledge Tradable in Edge-AI Enabled IoT: A Consortium Blockchain-Based Efficient and Incentive Approach," in IEEE Transactions on Industrial Informatics, vol. 15, no. 12, pp. 6367-6378, Dec. 2019, doi: 10.1109/TII.2019.2917307.

[3] T. A. Al-Janabi and H. S. Al-Raweshidy, "A Centralized Routing Protocol With a Scheduled Mobile Sink-Based AI for Large Scale I-IoT," in IEEE Sensors Journal, vol. 18, no. 24, pp. 10248-10261, 15 Dec.15, 2018, doi: 10.1109/JSEN.2018.2873681.

[4] I. García-Magariño, R. Muttukrishnan and J. Lloret, "Human-Centric AI for Trustworthy IoT Systems With Explainable Multilayer Perceptrons," in IEEE Access, vol. 7, pp. 125562-125574, 2019, doi: 10.1109/ACCESS.2019.2937521.

[5] T. Sutjarittham, H. Habibi Gharakheili, S. S. Kanhere and V. Sivaraman, "Experiences With IoT and AI in a Smart Campus for Optimizing Classroom Usage," in IEEE Internet of Things Journal, vol. 6, no. 5, pp. 7595-7607, Oct. 2019, doi: 10.1109/JIOT.2019.2902410.

[6] Y. Lin, Y. Lin, C. Liu, J. Lin and Y. Shih, "Implementing AI as Cyber IoT Devices: The House Valuation Example," in IEEE Transactions on Industrial Informatics, vol. 16,

no. 4, pp. 2612-2620, April 2020, doi: 10.1109/TII.2019.2951847.

[7] F Zafari, A Gkelias and K.K Leung, "A Survey of Indoor Localization Systems and Technologies", IEEE Communications Surveys & Tutorials, vol. 21, pp. 2568-2599, 2019.

[8] A. Longo, M. Rizzi, D. Amendolare, S. Stanisci, R. Russo, G. Cice, et al., "Localization and monitoring system based on ble fingerprint method", CEUR Workshop Proceedings, vol. 1982, pp. 25-32, 2017.

[9] S. Xia, Y. Liu, G. Yuan, M. Zhu and Z. Wang, "Indoor fingerprint positioning based on Wi-Fi: an overview", ISPRS Int. J. Geo-Inf., vol. 6, pp. 135, 2017.

[10] H. Suining and ChanS. H. Gary, "Wi-Fi Fingerprint-Based Indoor Positioning: Recent Advances and Comparisons", IEEE Commun. Surv Tut, vol. 18, pp. 466-490, 2016.

[11] J.S. Lee, M. F. Dong and Y.H. Sun, "A preliminary study of low power wireless technologies: ZigBee and Bluetooth Low Energy", 10th IEEE Conf. on. Industrial Electronics and Applications (ICIEA), 2015.

[12] M D'Aloia, F Cortone, G Cice, R Russo, M Rizzi and A Longo, "Improving energy efficiency in building system using a novel people localization system", 2016 IEEE Workshop on Environmental Energy and Structural Monitoring Systems EESMS 2016, 2016.

[13] Paterna Cantón, Vicente et al., "A bluetooth low energy indoor positioning system with channel diversity weighted trilateration and kalman filtering", Sensors, vol. 17.12, 2017.

[14] M. D'Aloia, A. Longo, R. Ruggero, S. Stanisci, D. Amendolare, M. Rizzi, et al., "An innovative LPWA network scheme to increase system reliability in remote monitoring", 2017 IEEE Workshop on Environmental Energy and Structural Monitoring Systems EESMS 2017, 2017.

[15] Zhang Zhongheng, "Introduction to machine learning: k-nearest neighbors", Annals of translational medicine, vol. 4.11, 2016.

[16] C.H. Chen and R.S. Cheng, "Improving Indoor Localization Based on Artificial Neural Network Technology", EAI Endorsed Transactions on Internet of Things, vol. 4, 2018.

[17] M. D'Aloia, M. Rizzi, R. Russo, M Notarnicola and L. Pellicani, "A marker-based image processing method for detecting available parking slots from UAVs", Lecture Notes in Computer Science, vol. 9281, pp. 275-281, 2015.

[18] A.B. Adege, Y. Yayeh, G Berie, H. Lin, L. Yen and Y.R. Li, "Indoor localization using K-nearest neighbor and artificial neural network back propagation algorithms", the 27th Wireless and Optical Communications Conference (WOCC2018), 2018.

[19] A. Gholoobi and S. Stavrou, "RSS based localization using a new WKNN approach", Computational Intelligence Communication Systems and Networks (CICSyN) 2015 7th International Conference on, pp. 27-30, 2015.