LPRNet: A Novel Approach for Novelty Detection in Networking Packets

Anshumaan Chauhan, Ayushi Agarwal, Angel Arul Jothi, Sangili Vadivel
Department of Computer Science, Birla Institute of Technology and Science
Pilani, Dubai Campus, Dubai, United Arab Emirates

***Abstract*—Novelty Detection is a task of recognition of abnormal data points within a given system. Recently, this task has been performed using Deep Learning Autoencoders, but they face several drawbacks which include the problem of identity mapping, adversarial perturbations and optimization algorithms. In this paper, we have proposed a novel approach LPRNet, a Denoising Autoencoder which uses algorithms such as Least Trimmed Square, Projected Gradient Descent and Robust Principal Component Analysis, to solve the above-mentioned problems. LRPNet is then trained and tested on NSL-KDD dataset, and experiments have been performed using Accuracy as performance metric for comparing the existing models with the proposed model. The results show that LRPNet has the maximum accuracy of 95.9% and performed better than all the previous state-of-the-art algorithms.**

Keywords*—Novelty detection; deep learning; autoencoders; unsupervised learning

I. Introduction

Novelty detection is classification of points whose characteristics are different from that of normal data [1]. These points which are not like the normal data are called anomalies. The process of anomaly detection is also known as outlier detection or out of distribution detection. Automatic anomaly detection is a task that is of high demand in the areas of fraud detection, network intrusion detection and several other fields.

Earlier the task of anomaly detection was a binary classification problem, where they used to train machine learning algorithms on the normal data as one class, and all the other data was treated as other class. Data which was categorized in this other class was taken as an anomaly point. Classic machine learning algorithms such as Support Vector Machines, Isolation Forest (also known as iForest), and many more algorithms have been used for the task of binary classification. Novelty detection approaches can be classified into 3 types, Density based classification algorithms, distance-based classification algorithms (also referred as clustering algorithms), and deep learning algorithms. The drawback of classical anomaly detection models was that they were inconsiderate about the temporal nature of data, that is, they classified points based on its value and not on the value of previous data fed to it. Due to this their results were not up-to the mark.

In recent years, there has been many advances in deep learning which has led to models which have performed significantly well in anomaly detection as compared to classical anomaly detection models. Algorithms and models which have shown significant results [2][3] are:

- Deep learning models: Autoencoders, Recurrent Neural Networks (RNNs) along with Long Short-Term Memory (LSTM)
- Dimensionality reduction techniques such as Self-Organized Maps (SOM), Randomized Principal Component Analysis (RPCA)
- State space models

This paper discusses the drawbacks of existing deep learning models used for anomaly detection. Also, this work proposes a methodology which overcomes some of those drawbacks. Deep neural networks follow two learning approaches: supervised or unsupervised. The problem with supervised learning, is that they require enough anomaly data along with the normal data, which is very hard to find, as anomalies do not happen every now and then. So due to this, the model is not trained well and does not give promising results. Therefore, most of the models follow an unsupervised approach.

Few drawbacks faced by existing methods [5] are:

- Problem of identity mapping
- Adversarial perturbations
- Optimization algorithms
- Appropriate data.

In this paper, we propose a denoising autoencoder model following an active learning approach, which incorporates projected gradient descent to overcome the drawback of adversarial perturbations and optimization. Robust estimation using Least Trimmed Squares (LTS) is used to prevent the model from adverse effects of outliers on the reconstruction error. After going through a lot of existing literature we found out that 2D-CNN architecture of autoencoder is well suited for this problem, as it has enough layers and uses nonlinear activation functions, which solves our problem of identity mapping.

The paper is organized as follows. Section II comprises a brief working information about the different Autoencoders. In Section III we have provided the issues involved with the currently used Autoencoder methodologies. Section IV comprises the brief description of relevant papers on Novelty Detection. In Section V a description of the dataset being used

is given. Section VI mentions the model development for LRPNet and Section VII comprises proposed methodology. Experiments and results have been given in Section VIII. Finally, in Section IX we conclude the paper along with future scope.

II. Background Theory

In this section we will see about autoencoders and have a brief description of different types of autoencoders.

A. Concept of Autoencoders

In Convolutional Neural Network (CNN), after a convolution layer, the size of input decreases. This decreased output is the features which are useful for solving the problem at hand. But it should be noted that the model learns those features in an unsupervised manner. After the loss of information, we want that information left at hand should be the one crucial for problem solving. From this idea emerged the concept of Autoencoders. Autoencoders is a feed-forward type of artificial neural network, which uses the concept of compression, that is, first the original data is reduced to data of low dimensional space. This job is done by one half of the network called as encoder. The other half does the exact opposite job, it tries to reconstruct the input from this latent space representation, generated by encoder. This part is known as a decoder.

Some of the important properties of Autoencoders are:

- Data-specific: Autoencoders are good at finding coding of only those kinds of data for which it is trained on, i.e., we cannot use autoencoders to compress a geospatial image which is trained on NSL-KDD dataset.
- Lossy: The output of autoencoders may not be exactly the same as the input provided to it. This property is also good for some reasons that it is not just performing identity mapping. Therefore, a reconstruction error is calculated to check how much different is the output from the input.
- Unsupervised: Autoencoders follow an unsupervised learning method, as we do not provide them with the labels, we just provide it with raw data.

Autoencoders are used for many purposes except for just dimensionality reduction, there are many applications where autoencoders were used for classification and generative purposes too. In Fig. 1, the working of an autoencoder is shown.

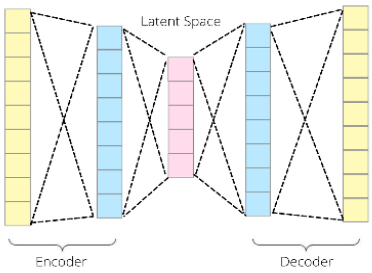

Fig. 1. Working of Autoencoders.

Encoder and decoder part of autoencoders are generally fully connected artificial neural networks and exact mirror images of each other.

B. Briefs of different Types of Autoencoders

Different varieties of autoencoders are used for novelty detection based on the type of input and application. They are as follows:

- Denoising Autoencoder: In these autoencoders, noise (using anomaly data or making some of our inputs as 0) is purposely introduced to prevent the model from simply copying the input to output without learning important features about the data.
- Sparse Autoencoder: Sparsity constraint is known to be a method which leads to good feature extraction. It is done by introducing an extra term to the cost function, which forces the optimization algorithm to reduce the weights of some neurons to almost zero. Due to these reduction in weights, neurons represent their output as a summation of very small values, which ensures the latent space representation to showcase important features (usually the number of neurons in the hidden layer are greater than the number of neurons in the input layer).
- Variational Autoencoder: These are also known as probabilistic autoencoders since their output is sometimes determined by chance even after the training phase is over. They are also known as generative autoencoders, this is because they have capability to generate new outputs that seem to have been taken from training dataset. They are preferred over the Restricted Boltzmann Machine (RBM) as they are easier to train and do faster computation.

Apart from these, there are many other autoencoders such as WTA autoencoders and adversarial autoencoders (but these are not much used in real life application). Amongst all, Denoising autoencoders are the most preferred for novelty detection.

III. Issues Involved in Novelty Detection

There are many challenges related to the input data when given to artificial neural networks (ANN) for training and testing. A combination of numeric and continuous data in the dataset needs to be separated (as a neural network only works with numeric data). Even after separating the different types of data, it is still not ready to give it as an input to the ANN. Some pre-processing steps such as data normalization must be performed so that later when the optimization algorithm changes the weights of the links, it does not change it in a way so that less importance is given to an important feature.

Another important task is to select an appropriate autoencoder. As described in Section II, there are various autoencoders that are used for the job of novelty detection. For novelty detection of cyber-attacks, many autoencoder algorithms such as denoising autoencoder, adversarial autoencoder, convolutional autoencoder and conventional autoencoders have been used. There have been many research papers that review the existing autoencoder architecture based

on the cyber-attack novelty detection, but for different datasets different kinds of autoencoder works fine and others do not.

Issue that arises after deciding on an appropriate autoencoder and doing all the preprocessing steps for the input data is, what will be the architecture of this artificial feed-forward neural network. In architecture of neural network there are many factors involved such as:

- Number of layers for compression (or reconstruction) and respective number of nodes: The number of nodes for the hidden layers should be chosen carefully; it should be in a proper decreasing order (increasing order for decoder), example 128:64:16. We cannot have too few layers as it would make our model underfitting and neither should we have too many as it will make it overfitting. In accordance with the input size, we have used 3 layers for compression (and 3 layers for decompression from the latent space). We have used a brute force method for deciding the number of layers and the number of neurons to be in each layer (shown in Fig. 6). Also, while designing the model we aimed to keep the model as simple as possible, so that it has less computational cost and provides a good accuracy at the same time.
- Different activation functions (or same) for the hidden layers: We can have different activation functions for different layers, but we should keep in mind that the result of these activation functions does not lead us to identity mapping, which is one the most discussed problem of autoencoders. To overcome this problem, in this work, we have used the non-linear activation function Rectified Linear Unit (ReLU), which makes sure there is no instance of identity mapping.
- Optimization algorithm: There are many optimization algorithms that work well, such as Projected Gradient Descent, Adam optimization algorithm, RMSProp, Gradient Descent with momentum and many more. In this work, we have used the Projected Gradient Descent algorithm for the optimization purpose and to prevent adversarial perturbations. Adversarial perturbations are the situation where the autoencoder usually ignores the important data or features that it should learn during compression, and learns the common characteristics that are usually also shared between the normal data and the anomaly data. This mainly leads to the low reconstruction error for normal as well as anomaly data.
- We have used Least Trimmed Squares for the task of Robust Estimation. As Denoising Autoencoders only take the normal data as input, we have to separate out the Outliers from the dataset. In our dataset, when the data points are plotted on a two-dimensional graph, there are some normal data points that are far away from others, and those are removed from the training dataset using LTS.
- Regularization method (to prevent from overfitting): Artificial neural networks often tend to overfit on the training set, to prevent from that as well as for faster convergence we use the concept of regularization. Algorithms such as dropout, L_1 and L_2 regularization, batch norm regularization and many more can be used for regularization. Sometimes one single method is not enough to implement regularization. Therefore, a combination of two methods can also be used. If dropout is not performing well for regularization, we can add batch normalization layers in between the network, and it often tends to solve the problem. In this work, we have used this above-mentioned method for regularization.
- Value of the hyperparameter learning rate is crucial so that the model doesn't have jumps that make it go away from the minimum error point. In this work, we have used 0.01 as the value of learning rate.
- To prevent the model from overfitting, we have used early stopping, which will stop the training phase as soon as the accuracy reaches 95%.

IV. Literature Survey

The problem of novelty detection can be solved using three approaches, as described earlier, that is, density-based approach, distance or clustering based approach and the deep learning-based approach. Density-based approach includes algorithms like GMM, etc. for classifying the anomaly data from the normal data. They use the concept of density of normal data, and make a Gaussian distribution out of it, data which lied in the maximum variance region was classified as anomaly data. OC-SVM, OC-KNN are algorithms which use distance-based approaches for novelty detection. The calculate the distance between A review of all the existing approaches have been given in. In all the experiments, the deep learning approach has performed significantly better than the other two approaches.

Novelty detection is an important task for learning systems in which a subset of the dataset does not fit well on the trained model [1]. Paper concluded that if this data is not in accordance with the data which was used to train the model, then its performance will be affected. In the review, it was mentioned that it is better if we train the model without giving any anomaly data as input and use a statistical approach for classifying the anomaly data. Review of some most used novelty detection techniques was conducted by Dubravko Miljković et al. [2] also concluded the same. After detailing about some important algorithms from each of the 4 approaches (classification-based approach, nearest neighbor-based approach, clustering based approach, statistical based approach) taken for novelty detection, it was concluded that factors such as labelled or n-labeled data, continuous or symbolic features (type of data), and many other factors related to data helps us to decide which will be the most appropriate algorithm for our novelty detection.

A survey of existing outlier techniques, where all different approaches including statistical models (further classified), neural network algorithms, machine learning algorithms and hybrid systems were taken for comparison and conclusion was that models should always be selected based on the dataset. The distribution of dataset, attribute types, and other factors

decide the speed and the accuracy of the model.[3]. It also mentioned that based on whether the data is labelled or not, we decide whether to go for distance-based approach, density-based approach, or novelty approach.

A comparison of different unsupervised anomaly-based approaches used for novelty detection in spatio-temporal data was conducted [4]. They proposed an algorithm that showed better results when the data used for training is scarce as well as having more than 5% of anomaly data. They proposed a hybrid autoencoder based approach which uses convolutional encoder (CAE) along with convolutional Long Short-Term Memory. After testing this model proved to be far better than all the considered methods including iForest, LSTM autoencoder and Convolutional autoencoder.

There are many different possible architectures of autoencoders which can be used for novelty detection. [5] compared some of the best architectures based on computational complexity and accuracy. After experimenting with architectures and other algorithms such as 1D-CNN,2D-CNN, MSCRED, OC-SVM, they concluded that for solving real-time problems, 2D-CNN is the best architecture (showed 100% accuracy in both tests and took minimum time for computation).

Emanuele Principi and Damiano Rossetti et. al [6] evaluated different autoencoder algorithms such as Multi-Layered Perceptron (MLP) autoencoder, Convolutional Neural Network Autoencoder and LSTM for detection of failure of the motor of an electric car. AUC (area under the curve) was the performance metrics, and these methods were trained on 1178 signals (1170 non defective signals and 8 defective signals that were considered as anomaly) and tested on 22 signals (8 normal signals and 14 anomaly signals). Experiments showed that MLP Autoencoder was the best and showed 99.11% accuracy.

Zhiwei Zhnag and Lei Sun [7] proposed an algorithm which uses the concept of along Progressive Knowledge Distillation with Generative Adversarial Networks (GANs), where two different GAN models were combined using the distillation loss. They compared this novel approach with OC-SVM (One Class- Support Vector Machine), Kernel Density Estimation (KDE) and Variational Autoencoder (VAE). Accuracy achieved by the proposed algorithm was 97.8% whereas VAE, KDE and OC-SVM were 96.96%, 81.43% and 95.13% respectively.

Tangqing Li et al. [8] came up with an approach using the concept of re-evaluation of examples after every epoch of training phase has been completed in an autoencoder. They tested this approach on datasets such as MNIST, KDDCUP, and many more, and compared their approach with many baseline models such as OC-SVM and Deep autoencoding Gaussian mixture model (DAGMM). Except for MNIST dataset, where OC-SVM performed better than the proposed algorithm (OC-SVM had an accuracy of 90.2%, Proposed algorithm had an accuracy of 84.2%), for rest all datasets, the proposed algorithm had a better accuracy when compared to all the baseline models.

Stainslav Pidhorskyi and Ranya Almohsen et al. [9] used an adversarial autoencoder with a probabilistic approach for solving the novelty detection problem. They first pre-processed the data by “linearizing the parameterized manifold”, which helps to understand deeply about the normal data (inliers) and then feed it to an adversarial autoencoder. Performance metrics used during experimentations were Area under ROC curve, F1-measure, Area under precision-recall curve and the FPR at 95% TPR (it is the chance that a normal data will be misclassified as anomaly data). The experiment was conducted on MNIST, CIFAR-10, Coil-100 datasets and showed results which were comparable to that of state-of-art algorithms.

The author in [10] explains why autoencoders are a better option for novelty detection than GANs (Generative Adversarial Networks). They stated that GANs during training can face a problem known as mode-collapse, that is, it may map more than 1 input image to a single output image. A full mode collapse situation is rarely encountered, but partial mode collapse can be frequent. Not only mode collapse, GANs are very sensitive to the choice of hyperparameters, non-convergence problems, and many more are the reasons that they are not preferred for novelty detection. Learning of non-semantic features was stated as a problem of autoencoder, that is, it may learn features that share common characteristics between normal and anomaly data, which leads to classification of anomaly data as normal data.

Jorge Meira et al. [11] did a comparative study on the unsupervised anomaly detection techniques used for cyber-attacks. They tested and checked the performance of algorithms such as Autoencoders, Isolation Forest (iForest), One Class-K-means and One Class- Nearest Neighbor on datasets ISCX and NSL-KDD. F1-score, Recall and Accuracy were taken into consideration for comparing the performances of the algorithms. It was noticed that Autoencoder when applied with pre-processing steps such as Z-score and Equal Frequency (EF) showed the best results for both the datasets.

Vishal M. Patel and Pramuditha Perera et al. [12] uses the concept of membership loss function in addition to the mostly used cross entropy error during the training phase of their neural network. For training they also used the knowledge gathered from data apart from what we have in our training dataset to make it learn generic feature filters. When tested and compared performance with VGG16 model on Caltech256 dataset, their proposed model showed superior performance. Accuracy of the proposed model was 93.9% whereas that of VGG16 model was 90.8%.

Autoencoder have shown to perform better when combined with other clustering techniques [13][14]. In [13], autoencoders were combined with. It was experimented on UCSD dataset along with algorithms such as ConvLSTM-AE, Conv2D-AE, Conv-3D AE, and many more. The results clearly showed the supremacy of the proposed algorithms over the others. AUC of the proposed algorithm was 96.5%, whereas the maximum other autoencoders reached was 91.2%. The author [14] used autoencoders with density-based clustering. The latent space encoding and the reconstruction error is sent to a density-based cluster. Points which exceed a

certain error threshold limit are categorized as anomalies. Taking AUC as performance metrics, the model was tested on a range of 20 datasets along with OC-SVM, PCA Based methods and combinations of these methods with density-based clustering. Out of 20 data sets, the proposed method performed better than all in 9 datasets and had the highest average AUC score of 78.29%.

Erik Marchi et al. [15] used Denoising autoencoders with bidirectional LSTM (BLSTM) for acoustic novelty detection. Experiments were conducted on PASCAL CHime speech separation and recognition challenge dataset along with algorithms such as LSTM-CAE (LSTM Convolutional Autoencoder), BLSTM-CAE, Gaussian Mixture Models (GMM) and Hidden Markov Models (HMM). The performances were compared on the basis of precision, recall and F1-measure. BLSTM-DAE (proposed algorithm) performed the best amongst all, having a precision of 94.7%, recall of 92.0% and F1-measure of 93.4%.

Yoshinao Ishii and Masaki Takanashi et al. [16] introduced the concept of robust estimation, which not only reduced the computational cost, but also guarantees robustness as its thresholds/restricts the capability of reconstruction of the autoencoder. Group of 15 datasets was used for comparing the performance of the proposed method with normal autoencoder, OC-SVM, iForest and Local outlier factor (LOF). Out of 15, in 7 datasets the AUC score of proposed algorithms was the highest. Proposed algorithm has the highest average AUC score of 85.15.

Chong Zhou and Randy C. Paffenroth et al. [17] stated that Denoising autoencoders are better to use than Maximum correntropy autoencoders, as denoising autoencoders purely trains on the noise free data, due to which its hidden layers are not corrupted (unlike Maximum correntropy autoencoder). They use the concept of RPCA which divides the dataset into two parts (noise free and noise data). Now this noise free data is used for the training of denoising autoencoders. During training they also used L_1 and L_2 regularization techniques as anomaly regularization penalties. The result showed that "the optimal F1-score achieved by iForest was approximately 73% worse than the score achieved by RDA. (Robust Deep Autoencoder)"

Zhaomin Chen and Chai Kiat Yeo et al. [18] evaluated Convolutional Autoencoder, Conventional Autoencoders and Dimensionality based reduction methods on NSL-KDD dataset. It stated that autoencoders are better as, along with performing dimensionality reduction, they also learn the non-linear relationship between the features of the training dataset. The performance metrics was AUC score. For network traffic type UDP, conventional autoencoder performed best, and for rest 3, convolutional autoencoder performed best (not much difference with AUC score of conventional autoencoder).

V. Data Collection

Kaggle is an open-source website, which is used by many machine learning learners and experts for different datasets and participating in various competitions. We are using the NSL-KDD dataset [19] which was provided in the Kaggle Dataset repository for the purpose of novelty detection. NSL-KDD dataset has 2 CSV files, one is used for training purposes and the other for testing purposes. NSL-KDD is a dataset that has 42 features which are shown in Fig. 2.

F#	Feature name	F#	Feature name	F#	Feature name
F1	Duration	F15	Su attempted	F29	Same srv rate
F2	Protocol type	F16	Num root	F30	Diff srv rate
F3	Service	F17	Num file creations	F31	Srv diff host rate
F4	Flag	F18	Num shells	F32	Dst host count
F5	Source bytes	F19	Num access files	F33	Dst host srv count
F6	Destination bytes	F20	Num outbound cmds	F34	Dst host same srv rate
F7	Land	F21	Is host login	F35	Dst host diff srv rate
F8	Wrong fragment	F22	Is guest login	F36	Dst host same src port rate
F9	Urgent	F23	Count	F37	Dst host srv diff host rate
F10	Hot	F24	Srv count	F38	Dst host serror rate
F11	Number failed logins	F25	Serror rate	F39	Dst host srv serror rate
F12	Logged in	F26	Srv serror rate	F40	Dst host rerror rate
F13	Num compromised	F27	Rerror rate	F41	Dst host srv rerror rate
F14	Root shell	F28	Srv rerror rate	F42	Class label

Fig. 2. Features in the Dataset.

Training file is composed of 125973 tuples and the testing file is composed of 22544 tuples. Out of these 148517, 78588 tuples have their label value as 'normal', and the rest all are anomaly packets as shown in Table I. There are a total of 36 other label values that are being treated as anomaly packets.

Our focus is to reject or drop any packet which has even a slight chance of being a malicious packet. Therefore, we are categorizing all 37 labels (types of packets) into broadly two categories; normal and malicious. The traffic proportions of these packets in our dataset are given in Fig. 3.

TABLE I. Frequency of each Label within the Dataset

normal	78588
neptune	47868
satan	4331
ipsweep	4078
portsweep	3302
smurf	3186
nmap	1699
back	1183
warezclient	997
teardrop	996
guess_passwd	464
mscan	310
warezmaster	299
pod	236
apache2	228
processtable	211
snmpguess	99
mailbomb	94
saint	93
buffer_overflow	47
snmpgetattack	43
httptunnel	41
land	20
multihop	16
rootkit	14
loadmodule	13
imap	13
ftp_write	10
ps	9
sendmail	8
phf	5
perl	4
xlock	4
xterm	3
named	2
spy	2
xsnoop	1

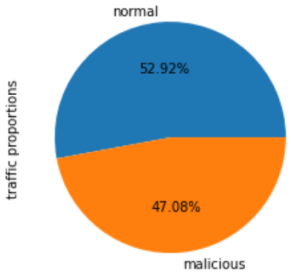

Fig. 3. Traffic Proportions of Dataset.

VI. Theoretical Background on Model Development

In this work, we have used Denoising Autoencoder, Z-score normalization and RPCA. In this section we have provided a brief explanation about these algorithms.

1) Denoising autoencoders: An autoencoder is a special type of deep neural network which is used for the purpose of dimensionality reduction. dimensionality reduction is a process of features selection as well as extraction from a n-featured dataset. Autoencoders are symmetrical in nature, and are basically composed of three components: Encoder, latent space encoding and decoder. The number of layers and the nodes within each layer in encoder and decoder are the same.

Denoising autoencoder is an autoencoder which purposely takes input which has some noise in it, or by making the input corrupt and then do the reconstruction or denoising part. One of the parameters that denoising autoencoder takes is the amount of noise that we want to introduce in the input. The most optimal value used for this parameter is 0.2 provided we have sufficient data. If data is limited then we can also go for higher values. Various applications of denoising autoencoder are feature imputation, anomaly detection, feature extraction and category embedding.

2) Z-score normalization: Z-score is also known as Standard score and it is a technique used to know how far the data point is from the mean of the attribute. More specifically it measures the standard deviation of the data point from the mean of the attribute. It is given by Eq. 1

$$z_i = \frac{x_i - \bar{x}}{s} \quad (1)$$

where x_i is the value of a data point, x bar is the calculated mean of the attribute and s is the standard deviation of the attribute. So, one of the prerequisites to use the z-score is to calculate the mean and standard deviation of the attribute first.

Z-score is used to standardize our dataset to a common range of values so that the autoencoder does not give more importance to a feature which has a high range of values as compared to others. It is one of the most used pre-processing steps for numerical data.

3) Robust principal component analysis: Robust Principal Component analysis is an extension to the most used dimensionality reduction technique Principal Component Analysis (PCA). RPCA is used when we are handling corrupted data or noise data. RPCA returns a low-ranking matrix L_o from a corrupted matrix composed of L_o+S_o. S_o here is a sparse matrix. This decomposition into low-rank matrix and sparse matrix can be achieved by using any of the following techniques: Quantized Principal Component Pursuit method (PCP), Local PCP or Stable PCP. Working of RPCA is shown in Fig. 4.

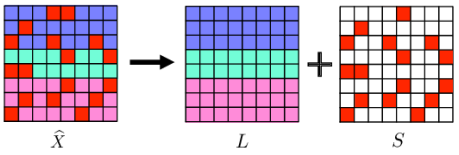

Fig. 4. Working of RPCA [20].

RPCA is used in anomaly detection, face detection, video surveillance and many more applications.

4) Projected gradient descent: It can be considered as a stricter version of Gradient Descent algorithm. In the Gradient Descent algorithm, we use Equation 2 for changing the value of weights and bias of a Neural Network

$$min\, f(\theta)\ \Theta(\text{new}) = \theta(old) - \alpha \Delta J(\theta) \quad (2)$$

Where, α is the learning rate, ΔJ(Θ) is the error (difference between the predicted outcome and the actual outcome) and Θ is the weight of neurons.

We can see that the error is being minimized by moving in a negative gradient direction. In Projected Gradient descent there is a constraint in this equation. It minimizes the error by moving in a negative gradient direction and then project that value onto a valid meaningful set, say *C*. By doing this we make the algorithm more general [21][22].

VII. Proposed Methodology

The steps in the proposed method are explained in this section.

Step 1: Load the dataset D and convert it into a binary dataset, one class being 'normal' and combining all other classes into 'malicious' class.

Step 2: Split the dataset into numerical and categorical column lists based on the datatype (do not include labels in the categorical columns).

Step 3: Remove column 'num_outbound_cmds' from the numerical column list.

Step 4: Encode the labels using Label Encoder. (1- normal and 0-malicious).

Step 5: Normalize the values in all the numerical columns using z-normalization.

Step 6: Form 2 datasets Dnormal and Dattack based on the label values. Use LTS for transferring some data points from Dnormal to Dattac that are far away from other normal data points in a 2-D graph representation.

Step 7: Convert the categorical data into numeric type using get dummies function of Pandas library (does one hot encoding) and combine them with the numerical data.

Step 8: After appending these numerical data in Dnormal and Dattack. Split Dnormal into two: Training (67%) and Testing (33%). Finally, we have the following three datasets: Dnormal-test, Dnormal-train, and Dattackt.

Step 9: Make a copy of all the datasets created in the previous step and process all using Robust Principal Component Analysis. This results in the following three datasets: Dnormal-test-RPCA, Dnormal-train-RPCA and Dattack-RPCA.

Step 10: Create 2 instances of the Proposed Model named as Model 1 and Model 2. Train Model 2 (LRPNet) on the Dnormal-train-RPCA and validate the model on Dnormal-test-RPCA.

Train Model 1, on Dnormal-train and validate the model on Dnormal-test.

Step 11: Test Model 2 on dataset Dattack-RPCA and Model 1 on dataset Dattack using different threshold values and compare them on their reconstruction error score.

The various steps of the proposed methodology are shown in Fig. 5.

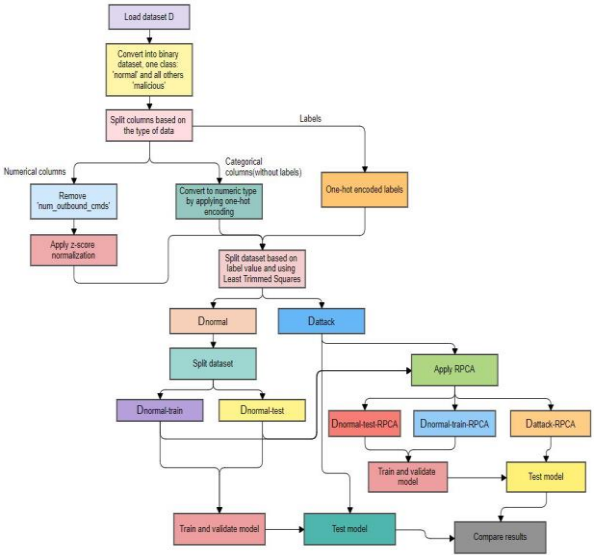

Fig. 5. Proposed Methodology.

VIII. Result and Discussion

We have used Python and Google Collaboratory for experimentation of our model on the NS- KDD dataset. The architecture of the autoencoder used is shown in Fig. 6. Early stopping is used to prevent the model from overfitting.

The results obtained by training and testing the model (Model 1) on the dataset without applying RPCA is shown in Fig. 7 and 8.

```
Model: "sequential_19"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_103 (Dense)            (None, 96)                11712
_________________________________________________________________
dense_104 (Dense)            (None, 64)                6208
_________________________________________________________________
dense_105 (Dense)            (None, 32)                2080
_________________________________________________________________
dense_106 (Dense)            (None, 16)                528
_________________________________________________________________
dense_107 (Dense)            (None, 32)                544
_________________________________________________________________
dense_108 (Dense)            (None, 64)                2112
_________________________________________________________________
dense_109 (Dense)            (None, 96)                6240
_________________________________________________________________
dense_110 (Dense)            (None, 121)               11737
=================================================================
Total params: 41,161
Trainable params: 41,161
Non-trainable params: 0
_________________________________________________________________
```

Fig. 6. Architecture of the Autoencoder.

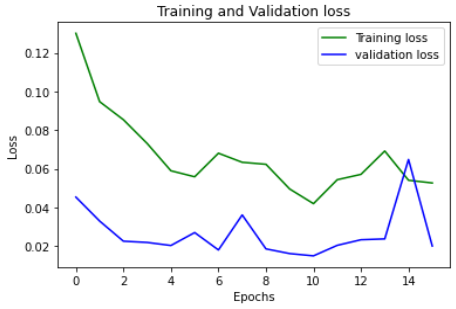

Fig. 7. Loss V/S Epochs Graph of Model 1.

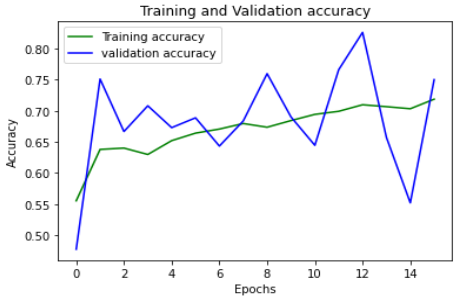

Fig. 8. Accuracy V/S Epochs Graph of Model 1.

The results of training the model (Model 2) on the dataset after applying RPCA is shown in Fig. 9 and 10.

Fig. 9. Loss V/S Epochs Graph of Model 2.

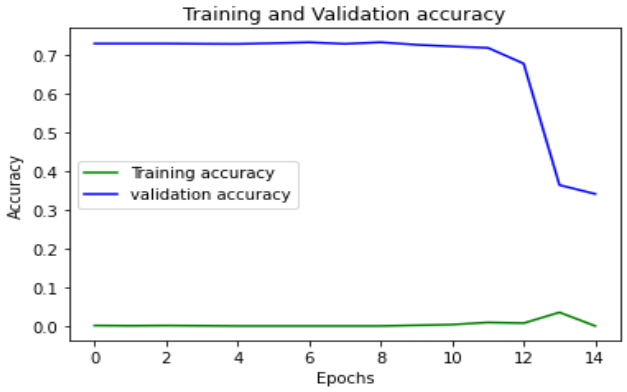

Fig. 10. Accuracy V/S Epochs Graph of Model 2.

We compared the accuracy of the models when RPCA is applied and when it is not applied. For Model 2 when RPCA was applied the reconstruction error for the normal data was high, around 33% and the reconstruction error of malicious packets was around 56%. In this case deciding a threshold value is very difficult because it has to be selected using a brute force method and selecting the one which corresponds to the maximum accuracy, but the prediction accuracies are very high. In Model 1, where RPCA was not applied before training the model, the reconstruction error for normal data was around 14% and for malicious packets was 55%, in this case the difference is quite large, and we can choose an appropriate value accordingly.

For different values of threshold, we got different levels of accuracies for both models which are formulated in Tables II and III.

TABLE II. ACCURACY RESULTS FOR DIFFERENT THRESHOLD VALUES FOR MODEL 2 (LPRNET)

Threshold value	Accuracy (%)
0.33	95.9%
0.35	91.4%
0.37	86.19%
0.39	79.2%
0.41	69.9%
0.43	46%

TABLE III. ACCURACY RESULTS FOR DIFFERENT THRESHOLD VALUES FOR MODEL 1

Threshold value	Accuracy (%)
0.12	95.8%
0.14	94%
0.16	90.91%
0.18	87.27%
0.20	85.76%

TABLE IV. COMPARISON OF MAXIMUM ACCURACY ACHIEVED BY EXISTING METHODS: COMPRESSION AUTOENCODER WITH BLSTM (BLSTM-CAE), COMPRESSION AUTOENCODER WITH LSTM (LSTM-CAE), DENOISING AUTOENCODER WITH BLSTM (BLSTM-DAE) AND DENOISING AUTOENCODER WITH LSTM (LSTEM-DAE)

Model	Accuracy (%)
LSTM-CAE	91.5%
BLSTM-CAE	92.7%
LSTM-DAE	93.4%
BLSTM-DAE	93.6%
LPRNet	95.9%

It could be noted from Tables II and III that the accuracy scores are better for both the models. Though RPCA has shown great results in other fields such as face recognition and video surveillance, in this work RPCA is not useful to a great extent as it is an overhead and the difference in accuracy is also not very high. Another disadvantage that we encountered was, with slight change in the value of threshold, the accuracy of the model depreciated at a very fast rate. Unlike Model 2, Model 1 (RPCA not applied) has a good accuracy over a range of threshold values. In Table IV comparative results with the existing architectures have been formulated and the proposed methodology shows the best results amongst all. The denoising autoencoder architecture proposed is efficient as it takes less than a minute in training and has shown great results provided the applicant choses proper threshold value for classifying between normal and malicious packets.

IX. CONCLUSION

This paper surveys different types of autoencoders that are used for novelty detection and states the issues that are involved while using autoencoders. The survey depicts that denoising autoencoders is the best approach for novelty detection and can have high performance if combined with techniques such as RPCA, clustering based methods and other tools. It also reveals that a proper architecture such as 2D-CNN should be used. Moreover, it could also be concluded that conventional autoencoders are not efficient for novelty detection until used with LSTM. Convolutional and LSTM Autoencoders have better performance but also have a drawback of high computational cost, and in real-time novelty detection, time is an important factor of consideration, therefore they are not preferred over denoising autoencoders. In this work, we experimented with the proposed denoising autoencoder model on the NSL-KDD dataset after applying proper pre-processing for novelty detection purposes. The results showed that the proposed denoising autoencoder achieved a maximum accuracy of 95.9%. The training is also not time consuming, and the accuracy achieved also shows high accuracy scores.

REFERENCES

[1] Marsland, Stephen. "Novelty detection in learning systems." *Neural computing surveys* 3.2 (2003): 157-195

[2] Miljković, Dubravko. "Review of novelty detection methods." *The 33rd International Convention MIPRO*. IEEE, 2010.

[3] Hodge, Victoria, and Jim Austin. "A survey of outlier detection methodologies." *Artificial intelligence review* 22.2 (2004): 85-126.

[4] Karadayi, Yildiz, Mehmet N. Aydin, and Arif Selçuk Öğrencí. "Unsupervised Anomaly Detection in Multivariate Spatio-Temporal Data Using Deep Learning: Early Detection of COVID-19 Outbreak in Italy." *IEEE Access* 8 (2020): 164155-164177.

[5] Meire, Maarten, and Peter Karsmakers. "Comparison of deep autoencoder architectures for real-time acoustic based anomaly detection in assets." *2019 10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS)*. Vol. 2. IEEE, 2019.

[6] Principi, Emanuele, et al. "Unsupervised electric motor fault detection by using deep autoencoders." *IEEE/CAA Journal of Automatica Sinica* 6.2 (2019): 441-451.

[7] Zhang, Zhiwei, Shifeng Chen, and Lei Sun. "P-kdgan: Progressive knowledge distillation with gans for one-class novelty detection." *arXiv preprint arXiv:2007.06963* (2020).

[8] Li, Tangqing, et al. "Deep Unsupervised Anomaly Detection." *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 2021.

[9] Pidhorskyi, Stanislav, et al. "Generative probabilistic novelty detection with adversarial autoencoders." *arXiv preprint arXiv:1807.02588* (2018).

[10] Salehi, Mohammadreza, et al. "Arae: Adversarially robust training of autoencoders improves novelty detection." *arXiv preprint arXiv:2003.05669* (2020).

[11] Meira, Jorge, et al. "Comparative Results with Unsupervised Techniques in Cyber Attack Novelty Detection." *International Symposium on Ambient Intelligence*. Springer, Cham, 2018.

[12] Perera, Pramuditha, and Vishal M. Patel. "Deep transfer learning for multiple class novelty detection." *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019.

[13] Chang, Yunpeng, et al. "Clustering Driven Deep Autoencoder for Video Anomaly Detection." *European Conference on Computer Vision*. Springer, Cham, 2020.

[14] Amarbayasgalan, Tsatsral, Bilguun Jargalsaikhan, and Keun Ho Ryu. "Unsupervised novelty detection using deep autoencoders with density based clustering." *Applied Sciences* 8.9 (2018): 1468.

[15] Marchi, Erik, et al. "A novel approach for automatic acoustic novelty detection using a denoising autoencoder with bidirectional LSTM neural networks." *Proceedings 40th IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP 2015*. 2015.

[16] Chang, Yunpeng, et al. "Clustering Driven Deep Autoencoder for Video Anomaly Detection." *European Conference on Computer Vision*. Springer, Cham, 2020.

[17] Zhou, Chong, and Randy C. Paffenroth. "Anomaly detection with robust deep autoencoders." *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*. 2017.

[18] Chen, Zhaomin, et al. "Autoencoder-based network anomaly detection." *2018 Wireless Telecommunications Symposium (WTS)*. IEEE, 2018.

[19] M. Tavallaee, E. Bagheri, W. Lu, and A. Ghorbani, "A Detailed Analysis of the KDD CUP 99 Data Set," *Submitted to Second IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA)*, 2009.

[20] Arrigoni, Federica & Fusiello, Andrea & Rossi, Beatrice & Fragneto, Pasqualina. (2015). Robust Rotation Synchronization via Low-rank and Sparse Matrix Decomposition. Computer Vision and Image Understanding. 174. 10.1016/j.cviu.2018.08.001.

[21] Gupta, Harshit, et al. "CNN-based projected gradient descent for consistent CT image reconstruction." *IEEE transactions on medical imaging* 37.6 (2018): 1440-1453.

[22] Bolduc, E., Knee, G.C., Gauger, E.M. *et al.* Projected gradient descent algorithms for quantum state tomography. *npj Quantum Inf* 3, 44 (2017).