Network Intrusion using Deep Learning

(Cascaded model of Stacked Denoising Autoencoder with SVM + SGD)

ABSTRACT

Deep learning models are having a significant impact in the field of computer network security. Network Intrusion Detection Systems, popularly known as Anomaly Detection systems are being built on machine learning models for effective detection of attack network packets and to prevent them from causing any theft or loss of confidential or important data across the devices connected to that network. We propose a cascaded model of **Stacked Denoising A.E** for unsupervised deep feature extraction and a **S.V.M model boosted with Stochastic Gradient Descent** for supervised classification.

1. INTRODUCTION

Introduction to NIDS(Network Intrusion Detection System)

Network intrusion detection systems (NIDS) are strategically placed at various points within the network to analyze traffic to and from all devices on the network. It performs an analysis of traffic flow of the whole subnet, and matches the traffic that is passed on the subnets to a record of known attacks. Once an attack is identified, or abnormal behavior is sensed, the alerts are sent to the network admin.

Types of NIDS'

There are two types of NIDS available based on their functionality

1.SNIDS(Signature based Network Intrusion Detection System)

2.ANIDS(Anomaly based Network Intrusion Detection System)

SNIDS:

These kind of systems look for byte sequences and patterns in network traffic which are malicious instruction sequences.This method is similar to the working of an antivirus software.SNIDS compares the patterns with it's stored memory of harmful patterns and alerts the admin.While it is effective to block old attacks, the system cannot recognise a new kind of attack unless it already knows the sequence.

ANIDS:

Since, SNIDS cannot always detect and prevent attacks,ANIDS systems have come into play.These systems are primarily designed to detect unknown attacks,and use machine learning models to detect network intrusion.The basic approach is to train a machine learning model and then compare new behaviours/patterns against these models.Proper feature selection algorithms are very essential for these systems to be effective in detecting malicious traffic otherwise they might give false positives

2. LITERATURE SURVEY

Research Paper - I

Machine Learning and Deep Learning Methods for Intrusion Detection Systems: A Survey

Abstract:

In this paper ,the authors have provided an intuitive and comprehensive explanation of various existing deep learning models.This paper has highlighted various events and attempts by many other researchers who proposed models using machine learning and deep learning techniques.The authors also described various datasets related to network intrusion detection; the oldest being the KDD-99 dataset,and other successor datasets like NSL-KDD and UNSW-NB15 which emerged as improved versions of the predecessors.

- Deep learning methods can directly work on raw data, allowing them to learn features and perform classification simultaneously.
- Unsupervised deep learning models can also be used to extract features; then, shallow models can be used to perform classification.
- However, the deep learning methods do not yield expected results all the time,especially when the datasets are small or unbalanced.So care should be taken while choosing appropriate deep learning methods based on the dataset.

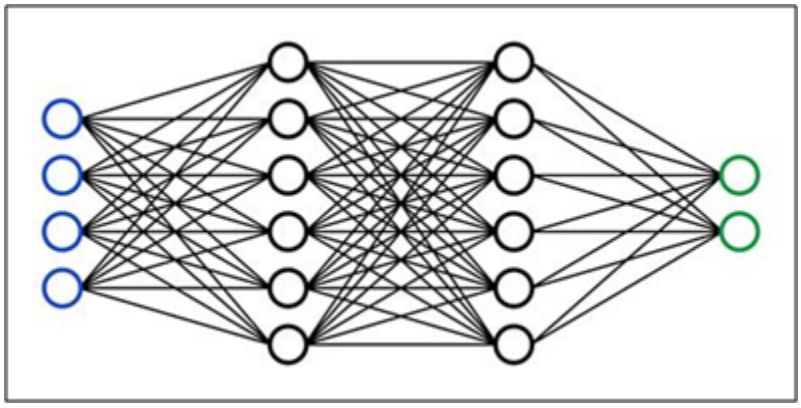
Some deep learning methods highlighted in this paper.

- **Autoencoders**:Auto encoders are unsupervised deep learning models which are mostly popular for learning feature representations from unlabelled data.
- **Restricted Boltzmann Machine (RBM):**It is a supervised deep learning model,basically a neural network that can learn probability distribution over its set of inputs.It is called restricted because the hidden layers should form a bipartite graph They can be trained in both supervised and unsupervised way depending on the task.

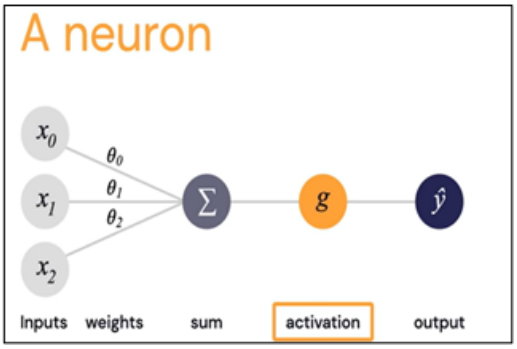
- **Deep Belief Network (DBN):**These networks are simply considered as layers of RBM'S stacked .It is combined with a logistic regression model.
- **Convolutional Neural Network (CNN):**A CNN is stacked with alternate convolutional and pooling layers. The convolution layers are used to extract features, and the pooling layers are there to improve feature generalizability.
- **Recurrent Neural Network (RNN):** RNNs are networks designed for sequential data and are widely used in natural language processing (NLP).They are unique because their "memory" as they take gather info from previous inputs to influence the current input and output.
- **Generative Adversarial Network (GAN):**This is a special type of deep neural networks, inshort they have two different neural networks ,a generator and discriminator.Basically, the generative network learns to map from a latent space to a data distribution of focus, on the other hand the discriminator network distinguishes records produced by the generator from the actual data distribution. They are usually used in image processing and analysis apps

Basics of a Neural Network:

A neural network basically contains an input layer, single or multiple hidden layer and output layer.Each layer in this network has nodes or neurons.When data passes from one layer to other ;starting from input layer to output layer, each layer learns features from the data and tries to extract latent features(i.e derived features).



How does a neural network interpret data?



A neuron in a layer is basically contains a mathematical function called the activation function, which takes certain inputs then calculates and passes the output to the next layer.This means when data passes through a layer in the network,that layer outputs a representation of the input data it is feeded.Before the activation function, the input data from the neurons of the previous layer is converted into a linear weighted sum (∑Xi.Θi).Generally these weights are tuned via back propagation.

Research Paper - II

Deep Learning Approach for Network Intrusion Detection System

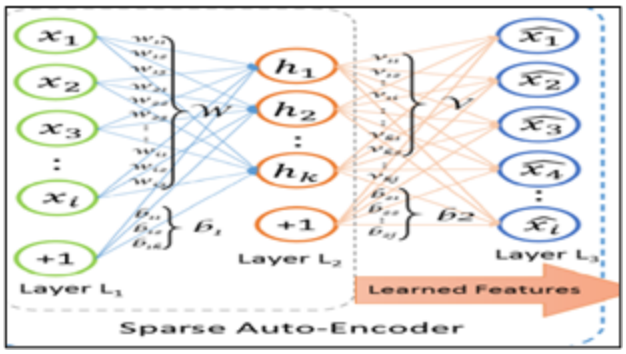
Abstract:

In this paper, the authors proposed a “Self Taught Learning” model.In theory, it is a two stage classification model ;where an unsupervised deep learning algorithm has been implemented for feature representation ,and a supervised model has been used for classification.The deep learning model is a sparse autoencoder and the supervised model is a softmax regression model(multinomial regression model).The NSL-KDD dataset has been chosen for network intrusion problems.

Comprehensive overview of the methodology

- Autoencoders : An autoencoder is an deep learning model that learns to reconstruct an output from its input.It contains two symmetrical components, an encoder and a decoder.The encoder converts input data to a hidden representation , and the decoder reconstructs the data from that representation.

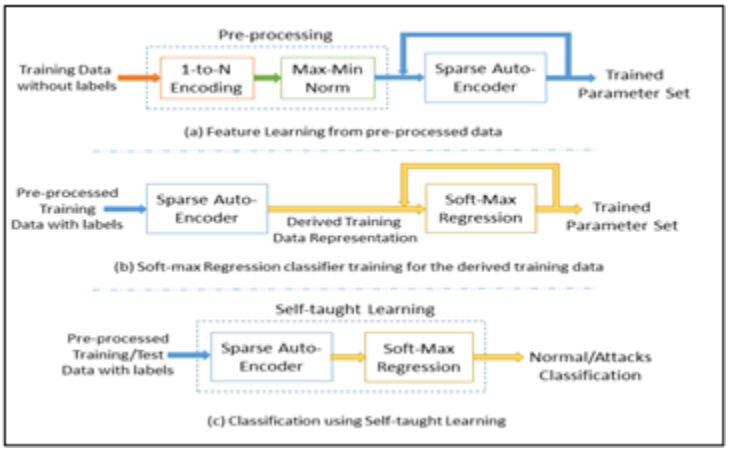
- Sparse Autoencoder : A sparse autoencoder is a neural network consisting of an input layer, hidden, and output layers. The activation function inside the neurons is a sigmoid activation function.It compresses to the inputs given to a value between 0 and 1. A sparse autoencoder tries to ensure that a neuron doesn't fire up most of the time. . This means that the average value of act.func is close to zero.



    - The AE has a parameter called sparsity parameter (rho).
    - Now the autoencoder tries to enforce a constraint that the mean value of the activation function is = to rho.which is already very near to zero eg 0.005.
    - The important point is this restriction ensures that a neuron only fires for meaningful patterns in the data as a result tries to reduce **overfitting**.

Preprocessing and training phase:

For preprocessing,the authors fed the training data and they have used 1-N encoding and max-min normalization and fed it to the autoencoder, to derive a trained parameter set. The authors trained a softmax classifier,with the output from the reconstructed data from the sparse AE.Now the S.A.E and the softmax regression have been joined and testing has been done.



1-N encoding is used to encode categorical variables into binary variables.

Testing and Results

For the 2-class classification, the model achieved 88.39% accuracy rate, whereas SM achieved 78.06%.

For the extended 5-class classification, the model achieved an accuracy of 79.10%

Research Paper - III

A Novel Two-Stage Deep Learning Model for Efficient Network Intrusion Detection

Abstract:

In this paper the authors proposed a two-stage deep learning model . A stacked autoencoder cascaded with a softmax regression model for classification.. The proposed model has successfully been able to acquire useful feature representations from huge amounts of unlabeled data and classifies them automatically and efficiently. To evaluate its effectiveness, several experiments are conducted on two public datasets, specifically the benchmark **KDD99** and **UNSW-NB15** datasets.

Comprehensive overview of the methodology:

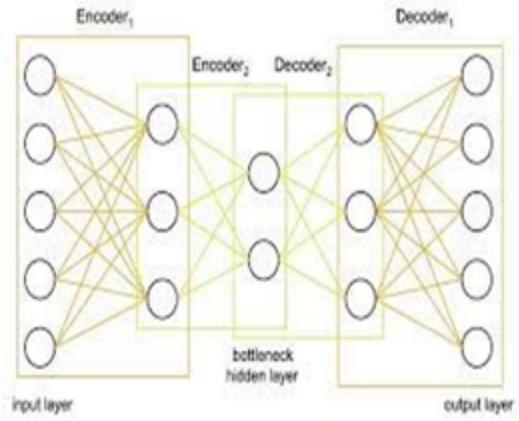
Stacked Autoencoder:

A stacked autoencoder is a neural network consisting of several layers of autoencoders where output of each hidden layer is connected to the input of the succeeding hidden layer.

Stacked autoencoder mainly consists of three steps:

1. Train the autoencoder using input data and acquire the learned data.

2. The learned data from the previous layer is used as an input for the next layer and this continues until the training is completed.



3. Once all the hidden layers are trained, use the backpropagation algorithm to minimize the cost function and weights are updated with the training set to achieve fine tuning.

This is done to improve accuracy.

Preprocessing and Training phase:

The data-preprocessing task includes necessary tasks:

the nominal-to-numeric data conversion and resampling of the data. Starting with, nominal-to-numeric data conversion is applied to both the datasets.Then for the two datasets namely the first KDD-99 ,it contains many duplicate records which often create bias in training,so undersampling is performed on this dataset.UNSW-NB15 is subjected to over sampling for maintain balance because there is a significant difference between the number of safe and unsafe traffic.

In training ,each layer of the SAE is pre-trained individually using an unsupervised learning technique. At each layer, the error from reconstruction of the input features is reduced.The initial unlabeled features are inputs to the first layer and the compressed features are inputs to the second layer. When the first and second layers are pre-trained separately, we stack them together and add a soft-max layer on top as a classifier for the next task, which is termed a fine-tuning task.

Testing and Results

For KDD-99:

KDD-99 for a 2-class dataset:2 class refers to one being normal(safe) traffic other being abnormal(harmful traffic).The model achieved 99.931% accuracy.

KDD-99 for multi-class dataset:5-class dataset; one being normal and other being 4 different types of attacks.The model achieved 99.996% accuracy.

FOR UNSW-NB15:

For UNSW-NB15 2class: The model achieved 89.711% accuracy

For UNSW-NB15 10 class:one being normal , the other are 9 different types of attacks.The model achieved 89.134% .

Research Paper - IV

A Deep Learning Approach for Effective Intrusion Detection in Wireless Networks Using CNN

Abstract:

B. Riyaz et al proposed a deep learning model which uses a feature selection algorithm called conditional random field and linear correlation coefficient-based feature selection**(CRF-LCFS)** algorithm for choosing most contributed features and classifying them using convolutional neural network**(CNN)**.

KDD 99 Cup dataset used for evaluating the proposed intrusion detection system.

Comprehensive overview of the methodology

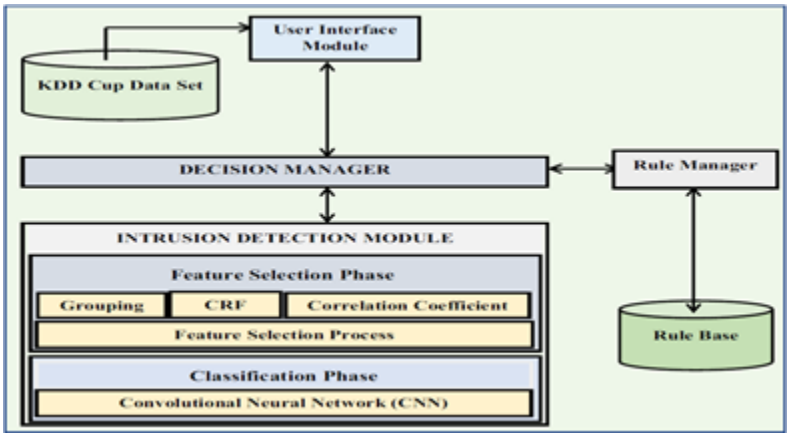
Conditional Random Field and Linear correlation Coefficient-based Feature Selection (CRFLCFS).

- In CRF LCFS, group the features by checking the distance between the attributes. Then, apply CRF for choosing two features that are useful for taking decisions over the records. After selecting the features, the correlation of coefficient value is calculated by using standard formulas and it also selects most contributed features for enhancing the classification accuracy.

Convolutional Neural Network(CNN)

CNN performs well in the process of classification and produces good accuracy for the large data. The CNN contributes the identical convolutional kernels that decrease the number of features and the size of training data.CNN has three layers, namely convolutional layer, pooling layer and fully connected layer. These convolutional layers use more kernels (filters) for the given input features and generate various feature maps in the process. In addition, the pooling layer of CNN used to shorten the size of the feature map for reducing the processing time. Finally, fully connected layer performs classification process on the extracted features which we get from convolutional and pooling layers

Preprocessing and training phase:



- Feature selection phase : CRF LCFS is used to select the most contributed features for performing the classification process from the dataset.
- CNN is responsible for classifying the records as normal users and intruders in the Classification phase.
- Decision manager responsible for forwarding input to the two phases and takes decision over data record with help of rule manager.

- Rule manager is used to manage the rules that are available in the rule base and adds new rules to the rule base if necessary.
- Rule base contains all the rules.

Results

The data set is split into 63% for training purposes and 33% for testing purposes.

The proposed model achieved detection accuracy of (98.8%)

Research Paper - V

Machine Learning Based Intrusion Detection Systems for IoT Applications

Abstract:

Abhishek Verma et al focused on utilizing ML classification algorithms for building IDS for securing IoT against DoS attacks.The performance of single classifiers including CART and MLP, and classifier ensembles namely Random forest (RF), AdaBoost (AB), Extreme gradient boosting(XGB), Gradient boosted machine (GBM),Extremely randomized trees (ETC) is measured in terms of prominent metrics, i.e., accuracy, specificity, sensitivity, false positive rate, area under the receiver operating characteristic curve (AUC).

CIDDS-001, **UNSWNB15** and **NSL-KDD** datasets are used.

Comprehensive overview of the methodology

- **AdaBoost (AB)**: AB is an adaptive meta-estimator that learns the initial training weights on the dataset. These weights act as input to additional copies of the classifier based on incorrectly classified instances.The

subsequent classifiers adjust the weights of classified instances. AB improves the performance of learning algorithms by boosting weak learners so that the final model converges to a strong learner.

- **Random Forest (RF)**:RF, classification algorithm consisting of many decision trees. Random forest is trained on bagging method. Each decision tree in a random forest gives class predictions and the class with the most votes will become a prediction model.
- **Gradient Boosted Machine (GBM)**:GBM aims to enhance the performance of decision trees (DT). Like other boosting algorithms, it sequentially combines weak classifiers, i.e., DT, and allows weak classifiers to optimize an arbitrary differential loss function in order to obtain strong prediction model.Each present learner (tree) relies on the predictions of previous learners for improving the prediction errors.
- **Extreme Gradient Boosting (XGB)**: XBM is an improved version of GBM. It uses more regularized model formalization in order to control over-fitting and increase generalization ability while GBM focuses only on the variance.
- **Extremely Randomized Trees (ETC)**: ETC is a tree induction algorithm for performing supervised classification and regression.builds an ensemble of unpruned DTs.
- **Classification and Regression Trees (CART)**: CART, predictive algorithm utilized in machine learning. It is used for predicting the targeted variable based on other variables . It's a DT where each fork is a split in a predictor variable and every node at the end has a prediction for the target variable.
- **Multi-layer Perceptron (MLP)**: MLP is a logical unit of connected nodes (artificial neurons) that attempts to mimic the biological brain behavior commonly referred to as a feed-forward artificial neural network.

Evaluation Metrics and Validation Methods

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

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

$$Senstivity = \frac{TP}{TP + FN}$$

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

$$AUC = \int_0^1 \frac{TP}{TP + FN} d\frac{FP}{FP + TN}$$

Testing and Results

RF outperforms other Machine learning classifiers in terms of accuracy (94.94%) and specificity (91.6%). GBM performs best in terms of sensitivity (99.53%). In terms of AUC , XGB performed the best by achieving 98.76%.

Research Paper - VI

Comparative Analysis of ML Classifiers for Network Intrusion Detection

Abstract:

Ahmed M. Mahfouz et al compared six ML classifiers regarding classification accuracy, TPR, FPR, precision, recall, f-measure,and ROC area. The chosen classifiers are Naïve Bayes, Logistic,Multilayer Perceptron (MLP), SMO (SVM), IBK (KNN) and J48 (DT).The six Machine learning classifiers are applied to the NSL-KDD dataset.

Preprocessing and Training phase:

The training process went in three phases:

- In phase 1 the ML classifiers are compared with none of the preprocessing done on the dataset. The training dataset is provided by NSL-KDD using 10-fold cross-validation for training the model.
- In phase 2 the NSL-KDD dataset is preprocessed by using InfoGainAttributeEval algorithm for selection 14 of 41 features.
- In phase 3 the dataset is balanced by under-sampling the dominant classes and over-sampling the minority classes. For undersampling WEKA is used and for oversampling SMOTE is used.

Testing and Results

For phase 1 J48(Decision tree) outperforms with an accuracy of 81.6%

Phase 2 IBK(KNN) outperforms with an accuracy of 86.8%

Phase 3 MLP outperforms with an accuracy of 97.5%

Research Paper - VII

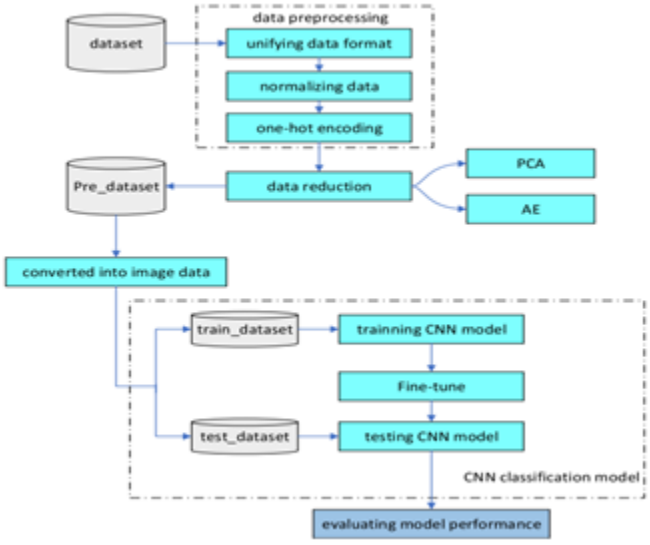
An Intrusion Detection Model Based on Feature Reduction and Convolutional Neural

Methodology:

The traffic generated in network communication is a one-dimensional byte stream.The CNN algorithm requires two dimensional data type.Considering this lack of compatibility between the structure of the CNN and the network data of IDS, we apply data preprocessing methods to get rid of the redundant and irrelevant features within network traffic data. The traffic is then transformed into a two-dimensional matrix form, which can be used in the proposed CNN network. This method not only solves the problem of preventing traditional machine learning models from determining the relationship between data features but also it can more clearly elucidate the features compared with the general neural network.

Proposed framework for intrusion detection

The overall framework of the model proposed consists of three steps:



Step1: Data preprocessing and data type conversion: The symbolic attributes within the KDD dataset are converted into a digital form and then are normalized to obtain standardized datasets. After that standardized datasets undergo dimensionality reduction, each network dataset is converted into a two-dimensional dataset, which conforms to the input file form of the CNN.

Step2:. After training the transformed dataset with the CNN, the optimal features are obtained.The transformed dataset is trained with CNN algorithm and optimal features are extracted from the transfromed dataset. Five attack states within the dataset are identified using the Softmax classifier.

Step3: The back propagation (BP) algorithm fine-tunes the features of the network model. After the optimal parameters of the network model are determined, the performance of the model is evaluated by the classification results of the test dataset.

Preprocessing and training phase:

The preprocessing was done in 3 phases: Symbolic feature numerization, Normalization and Label numerization.

Symbolic feature numerization: Attribute mapping transforms symbolic features into numerical form.

Normalization: Min max standardization was used without altering the relationship between original data.The min-max standardization formula is as follows:

$$y = (y - MIN)/(MAX - MIN)$$

Label numerization: Class identification of the label record is numerically processed

Testing and Results

Evaluation metrics: AC , DR, FAR

Testing :We use different dimensionality reduction datasets to measure the performance of the CNN–IDS model. The model uses the training dataset to coach the algorithm and verifies the training results on the test set. AC, DR, and FAR are used to measure the performance of the model

Results:The experimental results show that the CNN–IDS model proposed in the study efficiently detects network intrusion data. AC of 94.0%, DR of 93.0%, and FAR of 0.5% can be reached.

The experimental results indicate that the proposed CNN–IDS model not only considerably improves the classification detection performance of the intrusion network traffic but also significantly reduces the classification time, which satisfies the requirements of the intrusion detection system.

Research Paper - VIII

Towards Detecting and Classifying Network IntrusionTraffic Using Deep Learning Frameworks

Abstract:

The authors compared the utility and capability of different deep learning frameworks within area of network intrusion detection**.**Various deep learning frameworks (e.g., Keras, TensorFlow, Theano, fast.ai, and PyTorch) are applied and compared in detecting network intrusion traffic and also in classifying common network attack types using the recent CSE-CIC-IDS2018 dataset.

The Deep Learning Frameworks used :

Keras :The Keras interface is a portable, high-level neural networks API, written in Python, usable for TensorFlow, CNTK, and Theano as a back-end, and was originally developed as part of the Open-ended Neuro-Electronic Intelligent Robot Operating System (ONEIROS). This interface has multiple advantages when it comes to research and development; primarily its portability.

TensorFlow: TensorFlow is a machine learning open-source platform developed by Google with extensive industry use. It has a comprehensive, flexible ecosystem with a plethora of tools, libraries and community resources that helps researchers

and developers to easily build and deploy ML powered applications.This framework is available in Python, Java, JavaScript, and C++

Theano: Theano is a Python-based deep learning framework primarily popular due to its speed. This library utilizes various techniques to reach high speeds that, when used in along with a contemporary GPU, supersedes the speed of custom C programs.

Fast.ai: fast.ai was designed with the aim of making deep learning solutions more accessible to researchers and developers of diverse backgrounds. fast.ai is an open-source Python-based library that uses PyTorch as a backend.

PyTorch: PyTorch is a deep learning framework popular within academia and industry thanks to its wide support and research to release features. THis development framework is supported by most of the big cloud services and also allows intense customization and optimization.

Dataset Used : The datasets contain normal network traffic and malicious traffic generated from several different network attacks like Brute force attack, Denial of Service (DoS) attack , Bot attack, Web attack, Infiltration attack.

After downloading the dataset it is analysed to find its characteristics and is cleaned.Each of the cleaned dataset approximately contains 79 features; out of which 2 (Destination Port and Protocol) are treated as categorical using 1-to-n encoding and the rest are all numeric.Then they experimented with several deep learning frameworks discussed above for every dataset.

Conclusion

Three state of the art deep learning frameworks namely Theano, TensorFlow, and fast.ai in detecting and classifying various intrusion types. It is observed that fast.ai outperformed the other competing frameworks consistently among all the experiments yielding accuracy results up to 99% with a very low false positives and negative rates of less than 1%.

Research Paper - IX

Deep Learning Methods on Network Intrusion Detection Using NSL-KDD Dataset

The BAT-MC model contains five different layers, namely the input layer, multiple convolutional Layers, BLSTM layer, attention layer and output layer, from bottom to top.

1.In the input layer, BAT-MC model converts each of the traffic bytes into a one format. Each byte is encoded and then converted into a numerical form. After the traffic byte is converted into a numerical form, we perform normalization.

2. In the multiple convolutional layer, we convert the numerical data into traffic images. Convolutional operations are used to extract features and the data is converted to an image representation of a data packet.

3. In the BLSTM layer, BLSTM model links the forward LSTM and the backward LSTM. These are employed to extract features from traffic bytes of every network packet. BLSTM models can learn the sequential characteristics within the traffic bytes because it is very suitable to the structure of network traffic.

4.In the attention layer, attention mechanism is used to analyze the important degree of packet vectors to obtain fine-grained features which are more common in malicious traffic.

5.At the output layer, the features generated in the previous layer are then imported into a fully connected layer for feature fusion, which obtains the key features that can accurately characterize network traffic behavior.

Proposed System:

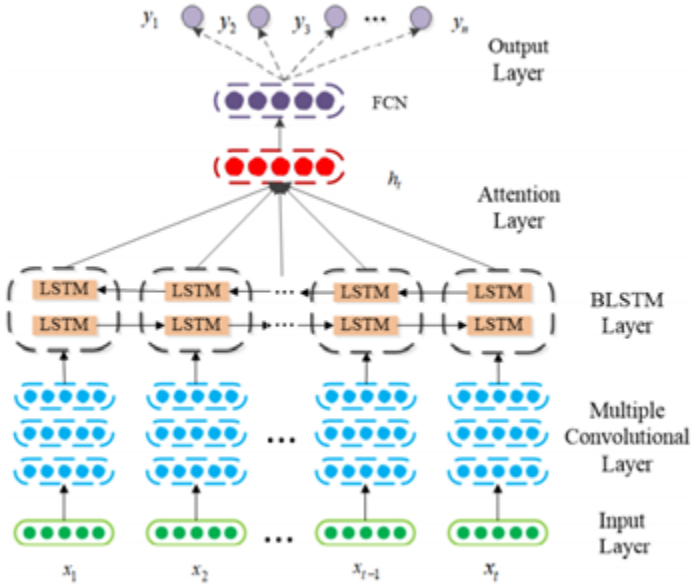


FIGURE 1. The Architecture of RAT-MC model. The whole architecture is divided into five parts.

Preprocessing and training phase:

Preprocessing

The NSL-KDD dataset consists of three symbolic data types namely, protocol type, flag and service.A One-hot encoder was used to map these features into binary vectors.

$$r_i c \frac{r - r_{min}}{r_{max} - r_{min}},$$
$$r_{max} = max\{r\},$$

Normalization : The standard scaler was used to normalize the continuous data into the range between [0, 1].

Testing and Results:

Evaluation metrics: AC , DR, FAR

Testing :The dataset was divided into two parts, training dataset and testing dataset.The operating environment of all experiments is Keras with tensorflow as the backend

Results: The accuracy of the proposed model on the KDDTest-21 data set is 69.42% and the accuracy on the KDDTrain+ data set is 99.21%.

Conclusion

Experimental results on the NSL-KDD dataset indicate that the BAT-MC model achieves pretty high accuracy.

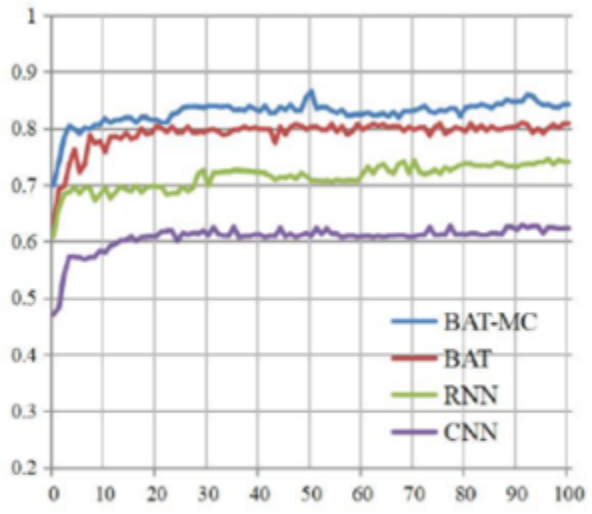


FIGURE 8. Comparison of Accuracy with different models.

3. PROPOSED METHODOLOGY:

For our deep learning model, we used the UNSW NB-15 dataset which has been provided by the official website.The training and testing data has been provided separately. The training set contains 82332 records and the testing set contains 175341 records.We have preprocessed the dataset using 1-N encoding and z-score normalization. Then we created a numpy array which contains randomly generated noise or Gaussian noise.

We corrupted the input (training data) with this noise.We then initialized the layers of the encoder and decoder of the first part of the SDAE.We created 4 dense layers of encoder and 4 of decoder.We then compiled the model and trained it on the noisy data.

Now we used dimensionality reduction and brought down the number of features from 43 to 20.We then initialized the second denoising autoencoder's dimension to 20. We then further use dimensionality reduction and bring the 20 features down to just 6 features. After training the second denoising autoencoder separately we now stack both the denoising autoencoders and then we obtain the data with deep feature representation. This data is used as input for the Linear svm .The linear svm is boosted with Stochastic gradient descent.

System Architecture:

The overall architecture of the proposed system consists of 5 components such asUNSW-NB15 dataset, preprocessing module, Stacked denoising auto encoder module, Support Vector Machine module and Stochastic gradient decent module.

UNSW-NB15 dataset:

UNSW-NB 15 dataset was created by Cyber Range Lab of the Australian Centre for Cyber Security (ACCS).This dataset has nine types of attacks,Analysis, Fuzzers, DoS,Backdoors, Exploits, Generic, Reconnaissance, Shellcode and Worms and has a total of 49 features.

Preprocessing module

Out of the 44 features , 4 features are nominal variables. Since the autoencoder can work only with numeric variables , we use 1-N encoding to convert those variables to numeric. Then data is normalized using z-score normalization.

Stacked denoising auto encoder module

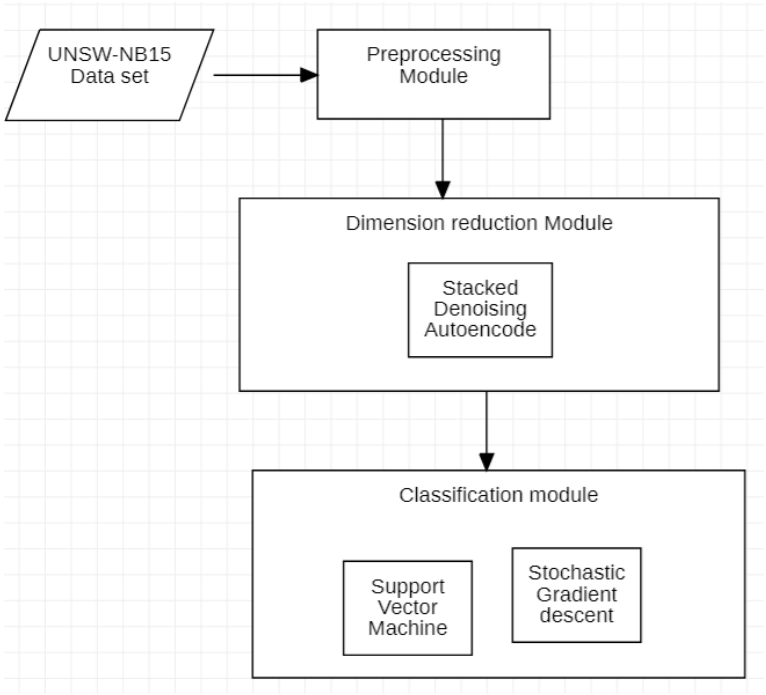
In the proposed model we are stacking two denoising auto encoders.**Denoising auto encoder (DAE)**takes the data to which noise has been added deliberately,which prevents Auto Encoder from learning an identity mapping between the input and reconstructed output and also captures more effective representations from noisy data. DAE helps in minimizing the reconstruction error more between z and x.

Support vector machine module

Support vector machines are machine learning models which are generally used for binary classification or regression purposes.An SVM aims to find the best hyperplane(a plane of n-1th dimension drawn in an n dimensional space) which would separate the points of two classes.

Stochastic Gradient descent module:

Stochastic Gradient descent is used to boost the support vector module. Stochastic(random) gradient descent selects a few random data points at every iteration and adjusts the weights to minimize the error. This makes the gradient adjustment system fast and efficient.



3.1 Dataset Description:

UNSW-NB15 :

UNSW-NB 15 dataset was created by the IXIA PerfectStorm and Tcpdump tools in the Cyber Range Lab of the Australian Centre for Cyber Security (ACCS).This dataset has nine types of attacks,Analysis, Fuzzers, DoS,Backdoors, Exploits, Generic, Reconnaissance, Shellcode and Worms. Argus, Bro-IDS tools have been used and 12 algorithms are developed to generate totally 49 features with the class label. This dataset consists of 2.5 million records.

3.1.1 Preprocessing:

Out of the 44 features , 4 features are nominal variables. Since the autoencoder can work only with numeric variables , we use 1-N encoding to convert those variables to numeric.For example, if a certain attribute takes values “**red**”,”**blue**”,”**green**”

only, then we encode red as 1 and blue as 2 and green as 3.For normalization of data, z-score normalization has been used.

- Sample data before preprocessing:

	dur	proto	service	state	spkts	dpkts	sbytes	dbytes	rate	sttl	dttl	sload	dload	sloss	dloss	sinpkt	dinpkt	
0	0.000011	udp	-	INT	2	0	496	0	90909.090200	254	0	1.803636e+08	0.000000	0	0	0.011000	0.0	0.00
1	0.000008	udp	-	INT	2	0	1762	0	125000.000300	254	0	8.810000e+08	0.000000	0	0	0.008000	0.0	0.00
2	0.000005	udp	-	INT	2	0	1068	0	200000.005100	254	0	8.544000e+08	0.000000	0	0	0.005000	0.0	0.00
3	0.000006	udp	-	INT	2	0	900	0	166666.660800	254	0	6.000000e+08	0.000000	0	0	0.006000	0.0	0.00
4	0.000010	udp	-	INT	2	0	2126	0	100000.002500	254	0	8.504000e+08	0.000000	0	0	0.010000	0.0	0.00
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
82327	0.000005	udp	-	INT	2	0	104	0	200000.005100	254	0	8.320000e+07	0.000000	0	0	0.005000	0.0	0.00
82328	1.106101	tcp	-	FIN	20	8	18062	354	24.410067	254	252	1.241044e+05	2242.109863	7	1	55.880051	143.7	4798.13

- Sample data after preprocessing:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	
0	0.151353	1.24742	-1.542786	-0.661729	-0.044028	-0.056242	-0.047047	-0.063537	-0.654855	-1.100636	1.574395	-0.636446	-0.198251	-0.040186	-0.057365	0.27738
1	0.357155	1.24742	-1.542786	-0.661729	-0.044028	-0.041793	-0.047000	-0.063025	-0.654886	-1.100636	1.574395	-0.636474	-0.201151	-0.040186	-0.057365	0.51135
2	0.179528	1.24742	-1.542786	-0.661729	-0.044028	-0.056242	-0.046981	-0.063744	-0.654863	-1.100636	1.574395	-0.636451	-0.199270	-0.040186	-0.057365	0.30946
3	0.153844	1.24742	-1.542786	-0.661729	-0.044028	-0.041793	-0.047103	-0.062219	-0.654843	-1.100636	1.574395	-0.636448	-0.195911	-0.040186	-0.057365	0.28021
4	0.270660	1.24742	-1.542786	-0.661729	-0.031872	-0.056242	-0.046665	-0.063820	-0.654872	-1.100636	1.574395	-0.636458	-0.201062	-0.040186	-0.057365	0.31250
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
35174	0.089544	1.24742	-1.542786	-0.661729	0.259855	-0.012895	0.266900	-0.066586	-0.654406	0.790668	1.574395	-0.627894	-0.201607	0.270161	-0.072148	-0.08940
35175	0.092117	1.24742	-1.542786	-0.661729	0.259855	-0.012895	0.266900	-0.066586	-0.654411	0.790668	1.574395	-0.627966	-0.201663	0.270161	-0.072148	-0.08737

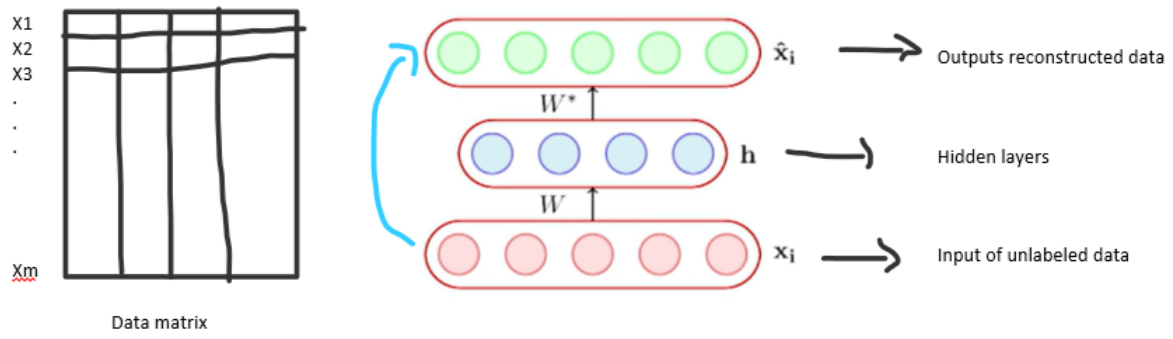
3.2 Working Model:

Stacked Denoising Autoencoder + SVM+SGD

Stacked denoising autoencoder:

In a simple Autoencoder(AE) there are three layers:

- Encoder layer: The encoder transforms the input vector x (non-linear vector) into a hidden layer representation (linear vector) through a non-linear mapping with help weights(w1).
- Decoder layer: The decoder maps the hidden layer representation back to non-linear vector z in a similar way using weight(w2).
- The AE aims to optimise the parameter w1, w2 to minimize the reconstruction error between z and x by training. The reconstruction error is measured by using mean square error.



The autoencoder encodes the input xi to a hidden representation h.
Let g be the activation function within a neuron , But as we know that before the input reaches activation function, it incurs linear transformation; i.e (weighted sum) Wxi.

Here W is the weight matrix ,and let b be the bias for each linear transformation

The linear transformation will be

Now this value of linear transformation will be incur a non linear transformation g.

g can be any function (most commonly a sigmoid function $\frac{1}{1+\exp(-z)}$)

This means that $h = g(z)$.

Now the reconstructed vector

$\hat{x}_i$

$=f(W * (h) + c)$

Here the function f will be usually decided by the nature of the data to be reconstructed:

It can be a logistic function ,
Because the logistic function in nature will generate the values between 0 and 1 (probability distribution)

Or else it can be a linear function; if the output is real values; else it can be tanh function.

When the model performs reconstruction of the data, some error occurs; the model is trained in such
a way that the error between the reconstructed output and the original input is minimum. In general the objective function is also referred to as the **loss function**.

It can be functions like Mean squared error $L(W)= \frac{sigma\left((xi-x')^2\right)}{n}$

As like any neural network , the weights(parameters) will be adjusted using BACK PROPAGATION METHOD TO MINIMIZE ERROR WITH EVERY PREDICTION.

Denoising Auto encoder (DAE)takes the data to which noise has been added deliberately,which prevents Auto Encoder from learning an identity mapping between the input and reconstructed output and also captures more effective representations from noisy data. DAE helps in minimizing the reconstruction error more between z and x.

When several layers of DAE are stacked up it represents SDAE.The stacking of encoders can build a high level representation output and this can be used as input for supervised learning algorithms like SVM and logistic regression.

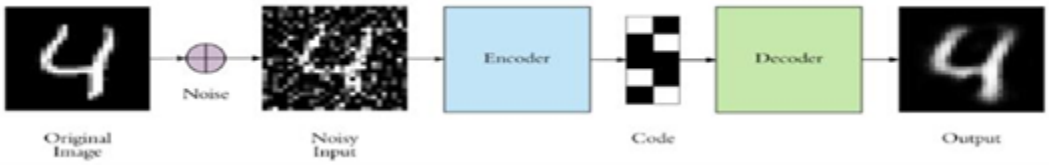


FIG: SAMPLE WORKING OF A DENOISING AUTOENCODER

SVM(Support Vector Machines):

Support Vector machines are shallow machine learning models which are generally used for binary classification or regression purposes.All an SVM aims to do is to

find the best hyperplane(a plane of n-1th dimension drawn in an n dimensional space) which would separate the points of two classes.Its name comes from support vectors;

Support vectors are the points which are closer to the hyperplane after it is drawn.

Support vector machines can also be used for multi-class classification by treating the classification problem in a different way, it can be made to do it by making some changes to the kernel function.

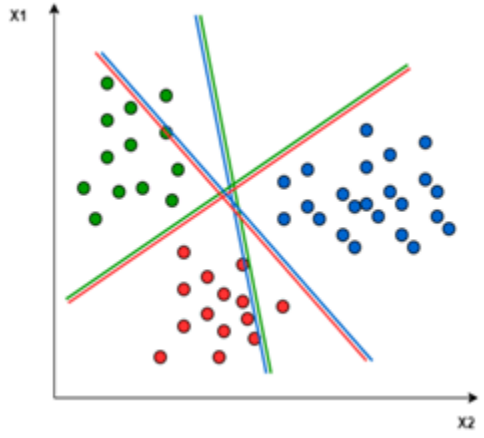
There are various kernel functions: Linear, Polynomial, Gaussian, Radial Basis Function (RBF), and Sigmoid.

There are two methods; one-to-one and one-to-rest approach.

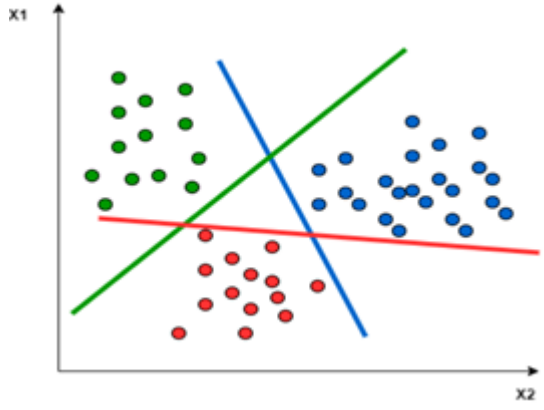
In a one-one approach, svms are used to recursively partition and classify,where an svm cares about separating only two classes at once.

In a one-rest approach,svms are also used to recursively partition and classify, but now the svm aims to draw a hyperplane between one class and the rest of the other classes**.**

One-one approach



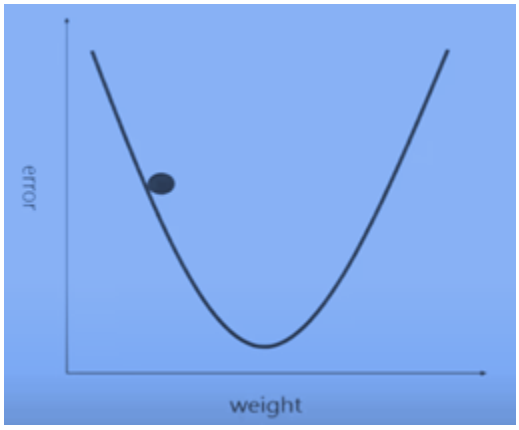
one-rest approach



Boosting an SVM with Stochastic Gradient descent

Gradient descent:

A graph between weights and error has been taken, This is usually a U -shaped graph.When we want to adjust weights in the neural network using back propagation, we want to minimize the errors that means we are finding the optimum weights by getting the minima of the graph.



Gradient descent does this by adjusting weights by comparing all the outputs for the tuples in the dataset.

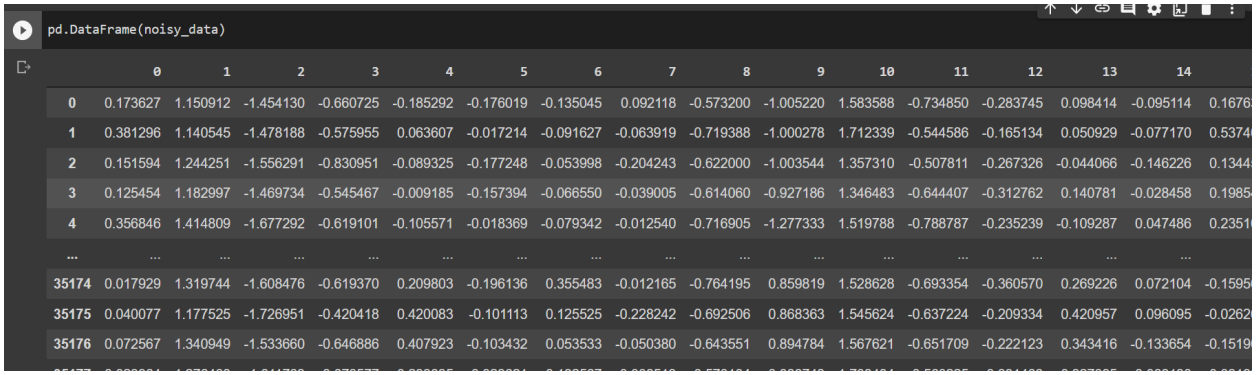
But this is not advisable for huge datasets like in our case , so we opt for stochastic gradient descent for optimization.

But stochastic(random) gradient descent selects a few random data points at every iteration and adjusts the weights to minimize the error.So this gives us the advantage of a faster and efficient method for gradient adjustment for huge datasets.

*****So our idea is that we can combine an SVM with this SGD ensemble for effective classification*****

4.Experimentation Results:

Noisy data:



pd.DataFrame(noisy_data)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	
0	0.173627	1.150912	-1.454130	-0.660725	-0.185292	-0.176019	-0.135045	0.092118	-0.573200	-1.005220	1.583588	-0.734850	-0.283745	0.098414	-0.095114	0.1676
1	0.381296	1.140545	-1.478188	-0.575955	0.063607	-0.017214	-0.091627	-0.063919	-0.719388	-1.000278	1.712339	-0.544586	-0.165134	0.050929	-0.077170	0.5374
2	0.151594	1.244251	-1.556291	-0.830951	-0.089325	-0.177248	-0.053998	-0.204243	-0.622000	-1.003544	1.357310	-0.507811	-0.267326	-0.044066	-0.146226	0.1344
3	0.125454	1.182997	-1.469734	-0.545467	-0.009185	-0.157394	-0.066550	-0.039005	-0.614060	-0.927186	1.346483	-0.644407	-0.312762	0.140781	-0.028458	0.1985
4	0.356846	1.414809	-1.677292	-0.619101	-0.105571	-0.018369	-0.079342	-0.012540	-0.716905	-1.277333	1.519788	-0.788787	-0.235239	-0.109287	0.047486	0.2351
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
35174	0.017929	1.319744	-1.608476	-0.619370	0.209803	-0.196136	0.355483	-0.012165	-0.764195	0.859819	1.528628	-0.693354	-0.360570	0.269226	0.072104	-0.1595
35175	0.040077	1.177525	-1.726951	-0.420418	0.420083	-0.101113	0.125525	-0.228242	-0.692506	0.868363	1.545624	-0.637224	-0.209334	0.420957	0.096095	-0.0262
35176	0.072567	1.340949	-1.533660	-0.646886	0.407923	-0.103432	0.053533	-0.050380	-0.643551	0.894784	1.567621	-0.651709	-0.222123	0.343416	-0.133654	-0.1519

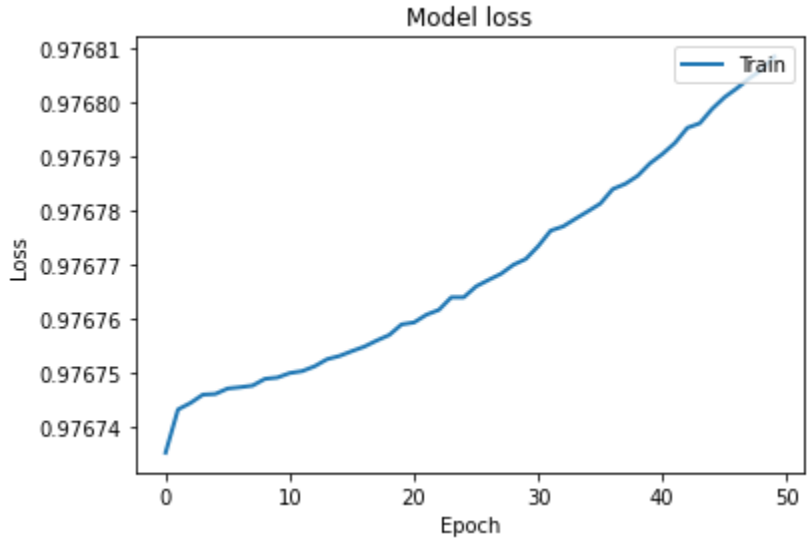
We initially declare a noise array with the same dimensions as the training data , then we add the noise array generated to the training data.

Summary of 1st DAE:

```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 43)]              0
_________________________________________________________________
dense (Dense)                (None, 35)                1540
_________________________________________________________________
dense_1 (Dense)              (None, 30)                1080
_________________________________________________________________
dense_2 (Dense)              (None, 25)                775
_________________________________________________________________
dense_3 (Dense)              (None, 20)                520
_________________________________________________________________
dense_4 (Dense)              (None, 25)                525
_________________________________________________________________
dense_5 (Dense)              (None, 30)                780
_________________________________________________________________
dense_6 (Dense)              (None, 35)                1085
_________________________________________________________________
dense_7 (Dense)              (None, 43)                1548
=================================================================
Total params: 7,853
Trainable params: 7,853
Non-trainable params: 0
_________________________________________________________________
```

In the above figure we can observe that we have compiled the autoencoder model, the tensor flow libraries will give us various details about the autoencoder, for example the number of parameters the output shape, the number of trainable parameters and the number of non trainable parameters in a tabular model

Epoch vs loss graph of 1st DAE in SDAE



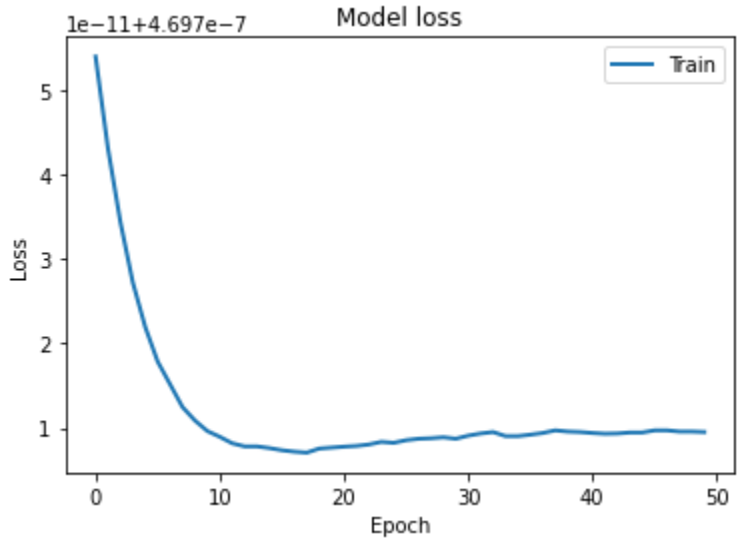
After compiling and generating the model we then start to train the autoencoder using the noisy data. Initially the loss of training the first autoencoder will raise , but in the next phases (for 2nd denoising autoencoder the training loss will decrease significantly.

Summary of 2nd DAE:

```
Model: "model_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_2 (InputLayer)         [(None, 20)]              0
_________________________________________________________________
dense_12 (Dense)             (None, 18)                378
_________________________________________________________________
dense_13 (Dense)             (None, 14)                266
_________________________________________________________________
dense_14 (Dense)             (None, 10)                150
_________________________________________________________________
dense_15 (Dense)             (None, 6)                 66
_________________________________________________________________
dense_16 (Dense)             (None, 10)                70
_________________________________________________________________
dense_17 (Dense)             (None, 14)                154
_________________________________________________________________
dense_18 (Dense)             (None, 18)                270
_________________________________________________________________
dense_19 (Dense)             (None, 20)                380
=================================================================
Total params: 1,734
Trainable params: 1,734
Non-trainable params: 0
_________________________________________________________________
```

Now we have initialized the second denoising autoencoder after reducing and extracting features from phase 1(first denoising autoencoder). The intializing dimensions for the second denoising autoencoder will be the dimension obtained after feature extraction. That is from 43 to 20. Now we set the in put dimensions of the second denoising AE to be 20 and then we train it.

Epoch vs loss graph of 2nd DAE in SDAE



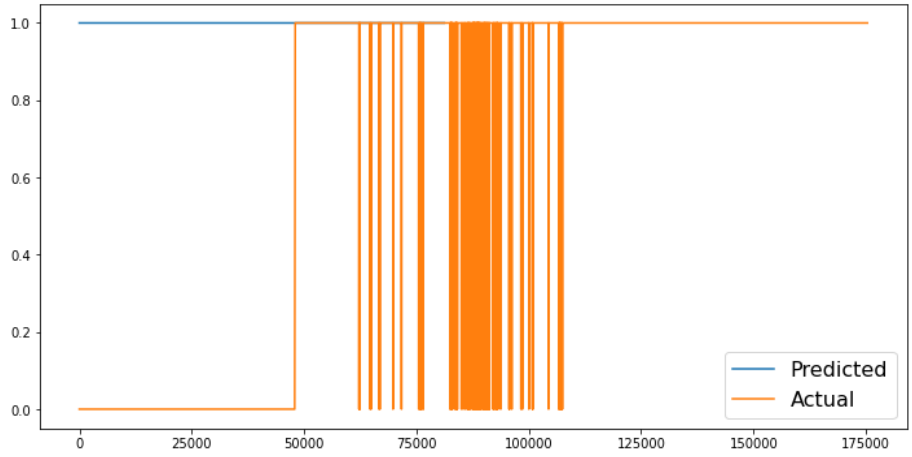
We can observe that after second denoising autoencoder the training loss has dropped significantly compared to the first phase.

Accuracy:

```
[39] from sklearn.linear_model import SGDClassifier
     from sklearn.metrics import accuracy_score

     clf = SGDClassifier(loss="hinge", penalty="l2")
     clf.fit(featuremode2.predict(featuremodel.predict(train_set)), target)

     y_pred = clf.predict(featuremode2.predict(featuremodel.predict(test_set)))

     print('Accuracy: {:.2f}'.format(accuracy_score(test_target, y_pred)))

     Accuracy: 0.76
```

Eventually we obtained 76% accuracy after passing the high feature representation data that is obtained from the autoencoder to the svm+ sgd boosted classifier.Here in the code, if we set the loss parameter to hinge it means that we are using linear svm as our classifier.

Graph for comparing Actual vs Predicted values



We have obtained a mean squared error of 0.24

Research Paper Number	Proposed Model	Our Model
Research Paper - II Deep Learning Approach for Network Intrusion Detection System	A sparse autoencoder cascaded with a softmax regression model Data set: NSL-KDD Accuracy:79.10%	A stacked denoising autoencoder cascaded with support vector machine and boosted using Stochastic gradient descent Dataset: UNSW-NB15 Accuracy: 76 %
Research Paper - III A Novel Two-Stage Deep Learning Model for Efficient Network Intrusion Detection	A stacked autoencoder cascaded with a softmax regression model Data set: UNSW-NB15 Accuracy: 89.134%	
Research Paper - IX Deep Learning Methods on Network Intrusion Detection Using NSL-KDD Dataset	BAT-MC Data set:NSL-KDD Accuracy: 84.25%	

5.Conclusion:

We have managed to implement the proposed model using google colb software and Python 3 and tensor flow and pandas libraries.Using those libraries has made the implementation much more simpler and efficient.Autoencoders have been widely used for the intrusion detection problem.Our cascaded model is completely new and has not been implemented before.We were able to achieve decent accuracy score of 76% with the UNSW NB15 data set, close to what other papers from esteemed journals written by well known researchers with vast experience.This model certainly has scope of improvement and necessary developments and modifications will be implemented in the near future.

4. REFERENCES

[1] Liu, H., & Lang, B. (2019). Machine learning and deep learning methods for intrusion detection systems: A survey. *applied sciences*, *9*(20), 4396.

link:(PDF) Machine Learning and Deep Learning Methods for Intrusion Detection Systems: A Survey (researchgate.net)

[2]Javaid, A., Niyaz, Q., Sun, W., & Alam, M. (2016, May). A deep learning approach for network intrusion detection system. In *Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies (formerly BIONETICS)* (pp. 21-26).

link:(PDF) A Deep Learning Approach for Network Intrusion Detection System (researchgate.net)

[3]Khan, F. A., Gumaei, A., Derhab, A., & Hussain, A. (2019). A novel two-stage deep learning model for efficient network intrusion detection. *IEEE Access*, *7*, 30373-30385

link:(PDF) A Novel Two-Stage Deep Learning Model for Efficient Network Intrusion Detection (researchgate.net)

[4]Riyaz, B., & Ganapathy, S. (2020). A deep learning approach for effective intrusion detection in wireless networks using CNN. *Soft Computing*, *24*, 17265-17278.

Link: https://link.springer.com/content/pdf/10.1007/s00500-020-05017-0.pdf

[5]Verma, A., & Ranga, V. (2020). Machine learning based intrusion detection systems for IoT applications. *Wireless Personal Communications*, *111*(4), 2287-2310.

Link: https://link.springer.com/article/10.1007/s11277-019-06986-8#Sec19

[6]Mahfouz, A. M., Venugopal, D., & Shiva, S. G. (2020). Comparative analysis of ML classifiers for network intrusion detection. In Fourth international congress on information and communication technology (pp. 193-207). Springer, Singapore.

Link: https://www.researchgate.net/profile/Ahmed-Mahfouz-8/publication/335430444_Comparative_Analysis_of_ML_Classifiers_for_Network_Intrusion_Detection/links/5e5d1a70a6fdccbeba12d5cb/Comparative-Analysis-of-ML-Classifiers-for-Network-Intrusion-Detection.pdf

[7]Xiao, Y., Xing, C., Zhang, T., & Zhao, Z. (2019). An intrusion detection model based on feature reduction and convolutional neural networks. IEEE Access, 7, 42210-42219.

Link: https://ieeexplore.ieee.org/document/8666014

[8]Basnet, R. B., Shash, R., Johnson, C., Walgren, L., & Doleck, T. (2019). Towards Detecting and Classifying Network Intrusion Traffic Using Deep Learning Frameworks. J. Internet Serv. Inf. Secur., 9(4), 1-17.

Link: https://www.researchgate.net/publication/337936440_Towards_Detecting_and_Classifying_Network_Intrusion_Traffic_Using_Deep_Learning_Frameworks

[9]Su, T., Sun, H., Zhu, J., Wang, S., & Li, Y. (2020). BAT: deep learning methods on network intrusion detection using NSL-KDD dataset. IEEE Access, 8, 29575-29585.

Link: https://ieeexplore.ieee.org/abstract/document/8988230