Detecting Network Transmission Anomalies using Autoencoders-SVM Neural Network on Multi-class NSL-KDD Dataset

Shehram Sikander Khan
Department of Information Assurance
St. Cloud State University
St. Cloud, MN, USA

Akalanka Bandara Mailewa
Department of Computer Science & IT
St. Cloud State University
St. Cloud, MN, USA

***Abstract*— As modern manufacturing shifts towards industry 4.0, mass adoption of vulnerable Internet-of-Things (IoT), Operational Technology (OT), and IT-OT convergence have precipitated the rise of malware. Accordingly, there is a need for a security mechanism that is both low-resource and highly accurate to address sophisticated attacks like zero-day and Mirai botnets. The study proposed a novel scheme that combined Deep Autoencoder (DAE) and Support Vector Machine (SVM); the hybrid scheme was tested on NSL-KDD: an imbalanced, multi-class, and high-dimensional dataset. We conducted a grid search analysis on L1 and L2 regularization to avoid overfitting and examined varying neural network formations to improve F1-micro and balance accuracy. The paper thoroughly evaluated all four attack classes within the NSL-KDD dataset: U2R, Denial of Service, R2L, and Probe. We demonstrated that DAE-SVM had a significant classification advantage over PCA-SVM; our model outperformed the baseline models at detecting low-frequency attacks with a micro-average score of 0.72 compared to 0.63 for PCA-SVM. To minimize computational overhead, we examined optimal feature fusion usage on Principal Component Analysis (PCA) and deep autoencoders. Our models, namely SVM, PCA-SVM, and DAE-SVM, were evaluated based on train and test times for rapid predictions. The DAE-SVM with L1 penalty was the model of choice for binary classes with a train and test time of 145 seconds, while DAE-SVM without any penalty term outperformed other models in the multi-class scenario delivering 142.62 seconds compared to 300 seconds for PCA-SVM.**

Keywords—NSL-KDD, Network Security, Intrusion Detection System, Deep Autoencoders, Anomaly Detection, Support Vector Machine (SVM), t-SNE, Deep Learning

I. INTRODUCTION

Cybersecurity faces a complex interplay of attack vectors, from nation-state threats to the rapid adoption of vulnerable devices. IoT, for example, has not kept pace with the rest of the hardware and software industry, exposing itself to DDoS and botnet attacks. Owing to these vulnerabilities, new IoT malware variants have come to the fore, such as Mirai, Gafgyt, Miner, Tsunami, and Xhide [1]. These botnets conduct Distributed Denial of Service (DDoS) attacks by flooding the host system with connection requests [2], [3].

The Covid-19 crisis and the rise of remote work have triggered a shift in American households' relationship with technology. In a 2022 Security report, IBM indicates an increase in data breach recurrences due to remote work, costing organizations $1,000,000 more than where remote work was not a factor [4]. Furthermore, remote workers have caused an increase in Man-In-the-Middle (MITM) attacks, as users open malicious links via email or chat and disclose confidential enterprise credentials[5], [6]. Zero-day exploits are, perhaps, the most concerning; they are attacks with no prior signature and, thus, no resolution is readily available [7]–[9].

The cloud revolution has spurred companies to harness the cloud to gain a competitive edge. A report by McKinsey predicts that enterprises plan to migrate 60 percent of their IT infrastructure to cloud platforms by 2025 [10]. However, this shift has increased the severity of data breaches, especially for industries with a large user base. For instance, the 2017 Equifax data breach compromised the information of over 150 million Americans, prompting a settlement of $575 million with FTC, CFPB, and 50 U.S. States and Territories. Other enterprises, at some point in time, were in similar straits: Facebook (530 million users exposed), Capital One (100 million users), and Zynga (218 million users), among others. Cloud infrastructure accounts for forty-five percent of breaches based on the IBM Security Report 2022 [4].

Given the current trends, there is an increasing body of literature championing the use of neural networks to monitor devices for anomalous activity. Our paper posits that deep autoencoder (DAE), as opposed to Principal Component Analysis (PCA), is well-suited for multi-class feature fusion. In other words, non-linear dimensionality reduction would complement the imbalanced nature of anomalies in internet transmission packets.

We will evaluate the efficacy of our proposed scheme by comparing it against standalone SVM and PCA+SVM using classification performance metrics. This paper will perform sophisticated evaluative techniques that tailor to the multi-class and imbalanced nature of the NSL-KDD dataset. The previous statements can be synthesized into the three research questions as follows:

Research Question 1. Does Deep Autoencoder outperform Principal Component Analysis at anomaly detection on a large, non-linear feature space?

Research Question 2. Does Deep Autoencoder reduce train and test times compared to Principal Component Analysis?

Research Question 3. Does penalizing our model avoid the enhance our model's classification performance and avoid overfitting?

Researchers have performed in-depth analysis on NSL-KDD dataset by fitting variants of autoencoders and standalone deep learning classifiers like k-nearest neighbor (KNN) and artificial neural networks (ANN); the performance of these models were measured on resource consumption and precision-recall metrics [11]. Many researchers have demonstrated promising results in non-linear feature fusion techniques --autoencoders, for instance -- for anomaly detection than linear feature fusion methods [12]. In similar vein, we will attempt to compare linear and non-linear feature fusion techniques and measure which performs the best within the anomaly detection domain.

A. *Intrusion Detection Systems*

Conventionally, cybersecurity experts have relied on Intrusion Detection Systems (IDS) to combat cyber-attacks from malicious actors. IDS traditionally rely on signature-based detection methods, which rely on a prior attack repository. However, the traditional detection method performs poorly with zero-day exploits, attacks that have yet to occur. Zero-day attacks have prompted many companies to move towards AI-based intrusion detection systems[13].

Before big data and cloud architecture, cybersecurity experts relied primarily on signature-based detection systems. These systems drew on a repository of prior attack signatures and would disinfect host machines based on user-provided instructions. The signature-based system was accurate and required less memory consumption. However, one big drawback was zero-day exploits, attacks that have not occurred before. Signature-based systems are at a disadvantage because the system is ill-equipped to tackle zero-day attacks. On the other hand, anomaly-based detection systems are trained –using machine learning -- to detect outliers from standard internet transmission packets and, thus, are more effective at catching zero-day attacks[14]. One criticism of this anomaly-based systems is that it requires more resources from the host system and tends to generate high false-positive occurrences. Our paper aims to reduce this issue by closely studying the classification performance metrics.

B. *Deep Learning Algorithms*

In recent times, researchers have garnered considerable interest in machine learning techniques due to their classification efficacy compared to conventional classification models like logistic regression[11][15]. ML models are becoming commonplace for AI experts to solve business problems -- Random Forest [16], SVM, and KNN[17]. Deep learning, a subfield of machine learning, has taken center stage in research as it uses neural networks. Unlike classification models, neural networks borrow advanced concepts from linear algebra, statistics, and calculus to optimize for accurate classification.

Neural nets have shown promising results in the field of anomaly detection[18]. Neural networks contain three layers: outer, hidden, and inner layer. In addition, researchers can experiment with model parameters based on the use case or the dataset. Neural networks include parameters such as learning rate, epochs, number of layers or units, and type of activation or loss function.

C. *NSL-KDD Dataset*

We will perform our anomaly detection analysis on the NSL-KDD dataset, an enhanced version of the KDD99 dataset. MIT Lincoln Lab designed the dataset for cybersecurity researchers to validate the efficacy of their model [19]. The dataset results from a nine-week-long TCP dump containing over 37 attacks which can be categorized into four attack classes, namely U2R, DoS, Probe, and R2L (Refer to Table 1). To thoroughly analyze the NSL KDD dataset, our study will perform a multiclass analysis of all these attack types using multiclass evaluative techniques.

TABLE I. NSL-KDD DATASET

Attack Class	**NSL-KDD Dataset**		
	Attack Type	***Train***	***Test***
U2R	buffer_overflow, loadmodule, perl rootkit	52	200
Probe	ipsweep, nmap, portsweep, satan	11656	2421
R2L	ftp_write, guess_passwd, imap, multihop, phf, spy, wareclient, warezmaster	995	2756
DoS	back, land, neptune, pod, smurf, teardrop	45927	7456

II. BACKGROUND AND RELATED WORK

The following section will discuss the theoretical underpinnings of neural networks and ML classifiers. In addition, we will examine how various researchers have implemented neural schemes on the NSL-KDD dataset by culling through scholarly journals. Lastly, we will do an in-depth discussion on how our proposed model, namely DAE-SVM, compares to other research and the limitations of our study.

A. *Neural Networks*

The deep learning architecture can be interpreted as an information processing mechanism; it is composed of neurons, which consists of a number termed 'activation'. Neurons are the building block of a neural network, consisting of weighted sum of weights and biases. The architecture follows a layered structure that starts from the input layer to the hidden layer and ends at the outer layer (refer to figure 1). After calculating the optimal weights and biases in each neuron, the system implements an activation function – sigmoid being the most common -- that further transforms the activation based on our use case. The neural system is contingent on a feedback loop that aims to find the global minima of a loss function: the process is termed backpropagation [20].

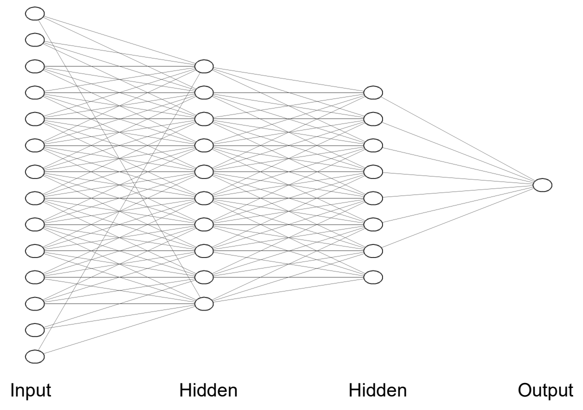

Fig. 1. Structure of Neural Nets

B. *Deep Autoencoders Neural Structure*

DAE follows an encoder-decoder architecture. They are analogous to a symmetrical ANNs, in that the encode and decode layers have same neurons in reverse. From a structure standpoint, the encoder layer consists of higher neurons than the successive bottleneck layer, and eventually decodes back to the original number of nodes [21]. The smaller bottleneck layer holds encoded representation after condensing the useful properties from the input layer. Once in the bottleneck layer, the data points are then reconstructed in the outer layer through the decode function. The encode layer can be represented by the function $h = f(x)$, whereas the decode function is represented through $r = g(h)$.

C. *SeLU Activation Function*

Although there are many activation functions used to linearize an activation; our model will use the SeLU function. It incorporates the full potential of ReLU function but avoids the issue of degrading AE performance. This function is equipped to handle more than three layers which is ideal for autoencoder layer structure. SeLU also performs normalization beforehand which can avoid gradient issues. In addition, our proposed activation function would allow our neural scheme to run faster, owing to its deep layered structure [22]. SELU is mathematically presented as:

$$SELU(x) = \lambda \begin{cases} x & if\ x > 0 \\ \alpha e^{x} - \alpha & if\ x \le 0 \end{cases}$$

D. *Autoencoders as a Feature Fusion Technique*

Deep AEs have emerged as an efficient feature fusion technique to their non-linear counterpart – namely manifold learning & Kernel PCA [23]. Based on various research, deep autoencoders are well suited to remove redundant information from large feature space, a central problem when building low-resource intrusion detection systems [24]. However, it should be noted that there are other researchers who have given preference to linear methods in other real-world scenarios [25].

Although autoencoders have a wide variety of applications, such as image denoising, hashing, automatic speech recognition, and data visualization, we will use AE as a feature fusion tool [26]. We will use an under complete formation – a structure when hidden layers have lower nodes than the outer layers. Autoencoders come in many variations: Deep AE, Denoising AE [27], Contractive AE [28], Convolutional AE [29] and sparse autoencoders. However, we decided to use Deep autoencoders, as they are more commonplace and suited for lightweight anomaly detection.

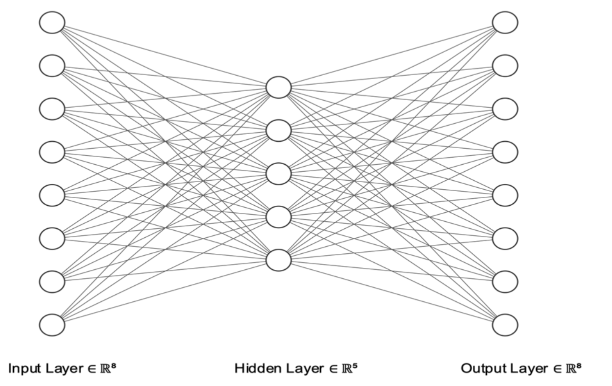

Fig. 2. Autoencoder (AE) Neural Architecture

E. *Support Vector Machine (SVM)*

This study will implement classification using the Support Vector Machine (SVM), as it cost-saving and high-dimension oriented. SVM is a robust algorithm that distinguishes different classes by finding the optimal threshold. The distance between the edge data point of one class and the defined hyperplane is termed 'margin'. As opposed to a maximal support classifier – which tends to maximize margin between class boundaries--, SVM uses soft margin which accommodates misclassification, leading to higher bias on training data but lower variance on the test data. To detect the optimal placement of hyperplane in each dataset, SVM can incorporate kernels to transform data points. The use of kernel enables SVM to work with high dimensional data, which is a central problem for this study [30].

Kernel comes in many forms: Radial, Polynomial, linear, etc.; they are vital to SVM, as they implement non-linear transformation to data points that are otherwise not linearly separable. Identifying the right type of kernel depends on the dataset and its concomitant class structure. For instance, Radial kernel finds the support vector classifiers, or soft margin classifier, in an infinite dimension. Polynomial, a popular choice, increases the dimension by *d,* defined as a degree of polynomials. For our purposes, we chose RBF kernel, as it is more suited for non-linear classification.

F. *Lasso and Ridge Regularization*

Since autoencoders are prone to overfitting, the use of regularization is crucial [31]. There are two types of regularization options: ridge regularization (L2 regularization) and lasso regularization (L1 regularization). The main distinction between L1 and L2 lies in the way it is mathematized. Ridge regularization introduces a bias to the model to account for overfitting; it does so by penalizing the model as follows:

$$Sum\ of\ Squared\ Residual + \lambda \times\ the\ slope^{2}$$

The sum of squared is added with the penalty term which represents lambda multiplied by the slope squared. The lambda can be adjusted based on the severity of the penalty we decide to impose. On the other hand, the L1 regularization takes the absolute of the slope, like so:

$$Sum\ of\ Squared\ Residual + \lambda \times\ |the\ slope|$$

The absolute, instead of squared, enables the user to remove useful features, which is quite useful when dealing with high dimensional features. We will perform grid search algorithm, an exhaustive search technique, to find what form of regularization would be well-suited for our dataset.

G. *Literature Review*

We will closely examine scholarly articles written in the context of anomaly detection and neural networks. A paper published in ICT Express incorporated Artificial Neural Networks (ANNs) to perform anomaly detection on malicious network traffic data sourced from exploitdb [32]. Their proposed ANN model resulted in a high accuracy of 98% and an average area under ROC of 0.98; furthermore, the false positive rate was lower than 2%. Although the paper thoroughly evaluated the classification metrics of the ANN model, the authors excluded discussion related dimensionality reduction, as shallow neural networks tend to underperform in classification with data exhibiting a large feature space.

This paper drew ideas initially presented by Majjed et al.; the authors extensively compared ML models' training and test times, from simple classifiers to more sophisticated ones [33]. Along with their in-depth attention to analyzing all types of models, they combined sparse autoencoders with simple vector machines. As a response, our study added a new dimension by discussing linear and non-linear dimensionality reduction; and examining which performed better based on various classification metrics. We also visualized our model's anomaly detection performance using T-SNE to spot all attack classes. Lastly, our study paid close attention to precision-recall scores for every attack class instead of binarizing the attack classes into 'normal' and 'abnormal,' which leaves the discussion of low-frequency attacks unaddressed.

Similarly, Mahmood et al. [34] follow a similar design by conducting a detailed comparison between PCA and ZCA against various autoencoders—the study data analysis on two datasets: NSL-KDD and Malware Classification Dataset. Although the authors outlined a multi-class analysis for the latter dataset, they demonstrated topline accuracy results for NSL-KDD, like the paper mentioned previously. Our research goes further by determining F-1 Score, Precision-Recall, and balanced accuracy for all attack types within NSL-KDD.

In a paper published by Shone et al. [35], the authors attempt to implement anomaly detection on NSL-KDD through nonsymmetrical deep autoencoders (NDAE) to reduce resource costs. To perform effective feature fusion, the authors fused NDAE-encoded features with a shallow learning classifier – namely, Random Forest. The authors correctly pointed out the challenges within the Network Intrusion Detection system, such as high volume, accuracy, low-frequency attacks, diversity, and adaptability. Furthermore, they diverge from other studies by utilizing a nonsymmetrical stacked autoencoder; however, our proposed model will remain symmetric. Unlike previously mentioned studies, the author included a multi-class analysis of 5-class classification and 14-class analysis.

Lastly, Clifford et al. [36] integrate three models in their detection scheme: deep neural net (DNN), Recurrent Neural Network (RNN), and self-taught learning (STL). The authors fit their model on KDD99 and NSL-KDD datasets. To gauge the validity of their proposed scheme, the authors correctly use Accuracy, Precision, Recall, and F-Measure. They concluded that autoencoders were better equipped to handle anomaly detection, as the accuracy resulted in 98.9%. The LSTM RNN model indicated a lower accuracy of 79.2%, whereas the STL variant resulted in 75.23%. There was a comprehensive discussion of classification on every attack type; however, there was a need to compare linear feature fusion techniques.

This section concludes the literature review after inspecting various studies implementing neural network-based anomaly detection. We examined studies deployed on multiple datasets, including NSL-KDD. Lastly, we compared their studies' models with our proposed neural scheme.

III. METHODOLOGY AND EXPERIMENTS

Since there is a considerable amount of complexity involved in the model-building process, in this section, we will outline why we chose our proposed model and compare it against the baseline models. Not only will we tabulate and analyze our performance metrics, but we will also give a detailed explanation of why we picked specific hyperparameters.

A. Data Preprocessing

From a data preprocessing standpoint, we imported standard data science libraries in the Python environment such as Numy, SKlearn, and Pandas. We leveraged one-hot encoder to transforms categorical values and normalized our data to effectively fit our proposed scheme on the dataset. Transforming data through normalization is vital to ensure outliers are addressed.

B. Design and Implementation of Study

To address the problem of large feature space, we decided to use a deep autoencoder (DAE) and complement it with a shallow classifier, namely, Support Vector Machine (SVM). Our grid search algorithm informed the hyperparameters and identified the ideal number of layers and neurons. The hyperparameters tuned by our grid search algorithms are SVM kernels, learning rate, epochs, and loss function. We have selected RMSE as our loss function, which is standard practice.

We used F-1 Micro and Balanced Accuracy to choose optimal hyperparameters, since these metrics sufficiently addresses the right balance within precision and recall tradeoff. We implemented our DAE-SVM model using the Keras library, a high-level deep learning API. We will gauge our model's overall effectiveness by comparing it against a standalone SVM classifier and a PCA-SVM classifier.

Having the PCA-encoded feature space against DAE-encoded features would address the central theme of the study: putting forward a lightweight neural network scheme. Since our study aims to analyze the NSL-KDD dataset thoroughly, we will perform both a binary class and a 5-class classification. To this end, we will present figures and tables that address the multi-class nature of the problem.

C. Classification Evaluation

To evaluate the efficacy of our proposed DAE-SVM model, we will rely on the classification confusion matrix [37]. The important terms to understand the matrix are as follows:

1) *True Negative:* Negative attack predictions that were objectively negative.

2) *True Positive:* Positive attack predictions that were objectively positive.

3) *False Negative:* Negative attack prediction that were objectively positive.

4) *False Positive:* Positive attack prediction that were objectively negative.

These metrics are the foundational to calculate crucial metrics such as precision, recall, accuracy, and F-measure.

D. Accuracy, Precision, Recall, F-Measure

Accuracy provides overall estimate of our correct predictions regardless of the attack class. It provides overall reliance on our model, and can be mathematically defined as follows:

$$Accuracy = \frac{True\ Positive + True\ Negative}{True\ Positive + True\ Negative + False\ Positive + False\ Negative}$$

Precision metric states the proportion of positive results that were correctly identified. Precision excludes true negatives from its formulation. On the other hand, recall indicates the proportion of positive predictions from all positive occurrences. The way precision and recall is formulated, we expect a trade-off relationship between recall and precision. To get a holistic view of our model, we will present precision-recall curves displaying precision and recall at every classification threshold.

$$Precision = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

$$Recall = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

We use the F-measure metric to consider both the precision and recall into one measure, it takes a harmonic mean of both precision and recall. All the mentioned metrics are mathematically formulated as follows:

$$F\ Measure = \frac{(1+\beta^1)\ Precision * Recall}{Precision + Recall}$$

E. *Train and Test Timing*

To gauge the resource consumption of our proposed model, DAE-SVM, we will compare the train and test times. We employed the time library within the Python environment to measure time taken for Models to train and test.

IV. Results

A. *Advanced visualization using T-SNE*

This section presents advanced visualizations sourced from running t-distributed Stochastic Neighbor Embedding (t-SNE) on our baseline and proposed models. Built on Kullback Leibler (KL) divergence, T-SNE considers the interpoint distances to avoid noise [38]. T-SNE reduces high-dimensional feature space into 2D; furthermore, it clusters together small neighborhoods of data points so that end users can visualize better. The axis in the visualization is calculated based on KL divergence. We ran different variations of t-SNE on our models based on tunable parameters – namely, perplexity and iterations. When dealing with normal-sized features, we can stay within the range of 5 to 50; however, in higher dimensional data, it is acceptable to pick a higher perplexity. We reduced the 122 NSL-KDD dimension space to a more manageable number of 10 dimensions before running T-SNE. These images were rendered using the SeaBorn Python library.

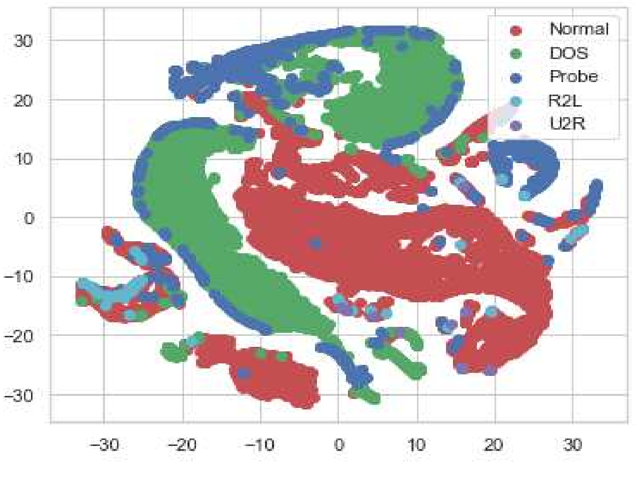

Fig. 3. AE-Encoded t-SNE Low-Dimension Representation (Perplexity 50, Iterations 500)

Figures 3 and 4 present t-SNE visualization of our proposed DAE-SVM model, the former exhibiting perplexity and iteration of 50 and 500, respectively. We can see 5-class attack types in a small neighborhood, distinctly coalescing into clusters. In the latter figure, we can see more interpoint spaces among the attack classes, which suggests better clustering.

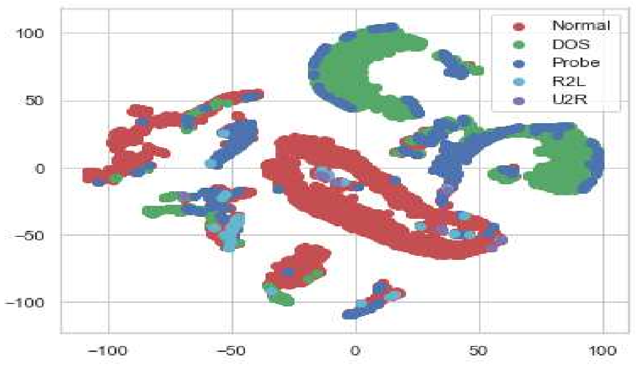

Fig. 4. AE-Encoded t-SNE Low-Dimension Representation (Perplexity 50, Iterations 1000)

If we notice t-SNE representations in Figures 5 and 6, the clusters in the first figure are not distinctly formed. We can see probe attack types being identified in the DOS clusters, suggesting bad handling of multi-class labels. When we increased the iterations for PCA-SVM, we noticed a bigger interpoint distance, but probe labels are still expressing in other class types. Another noticeable trend is that PCA-SVM and AE-SVM performed well at classifying 'normal' and 'attack' but had difficulty tracking low-frequency attack types. Overall, the low-frequency attack types seem difficult to classify based on t-SNE visualization, which suggests that we would need to refer to precision-recall tables.

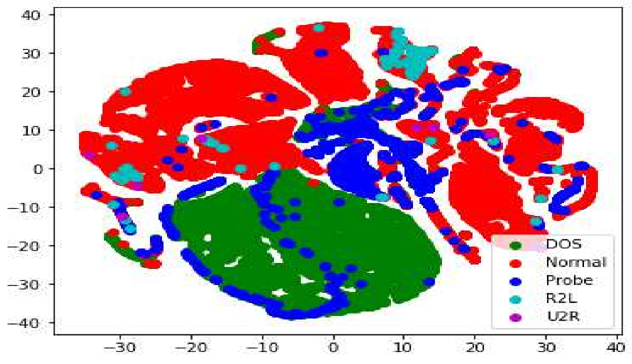

Fig. 5. PCA-Encoded t-SNE Representation (Perplexity 50, Iterations 500)

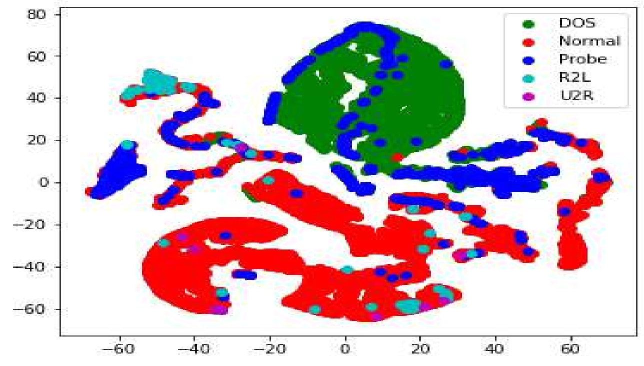

Fig. 6. PCA-Encoded t-SNE Representation (Perplexity 50, Iterations 1000)

B. GridSearchCV

To find optimal parameters for our SVM classifier, we tabulated penalty, kernel, and standard deviation using the Gridsearch algorithm. The parameters were listed with their concomitant balanced accuracy and F-1 Micro score. Tables 2 and 3 indicated polynomial with a penalty parameter of 100 as ideal and RBF with a penalty of 1000 as optimal SVM parameters, respectively.

TABLE II. BALANCED ACCURACY GRID SEARCH

Balanced Accuracy	Grid Search with 'Balanced Accuracy' Scoring		
	STD	*C*	*Kernel*
0.458	(+/-0.035)	1	linear
0.823	(+/-0.094)	1	rbf
0.755	(+/-0.081)	1	poly
0.66	(+/-0.079)	100	linear
0.849	(+/-0.126)	100	rbf
0.875	**(+/-0.106)**	**100**	**poly**
0.688	(+/-0.109)	1000	linear
0.828	(+/-0.091)	1000	rbf
0.845	(+/-0.135)	1000	poly

TABLE III. F1-MICRO GRID SEARCH

F1- Micro Score	Grid Search with 'F1-Micro' Scoring		
	Std	*C*	*Kernel*
0.704	(+/-0.258)	1	linear
0.815	(+/-0.011)	1	rbf
0.781	(+/-0.019)	1	poly
0.567	(+/-0.023)	100	linear
0.928	(+/-0.011)	100	rbf
0.903	(+/-0.014)	100	poly
0.665	(+/-0.028)	1000	linear
0.949	**(+/-0.013)**	**1000**	**rbf**
0.937	(+/-0.012)	1000	poly

C. Binary and Multiclass Classification:

With the detection of each class being paramount, we will closely evaluate the precision and recall of binary and multi-class attack types. Although accuracy is important, we can afford a lower score given the imbalanced nature of low-frequency attacks. Given the severity of Type 1 errors (flagging the 'U2R' attack as false and compromising the host machine), it is crucial that our model has high recall rates for each attack type. For a more nuanced discussion, we will also center our discussion on the f-1 score and precision toward a robust detection system.

The following tables examine the validity of the following baseline and proposed model: SVM, PCA-SVM, and DAE-SVM. Table 4 delineate the multiclass attack classes for standalone SVM; the figures indicating an accuracy of 71% and no precision and recall values for R2L and UR which are low-frequency attacks in the dataset.

TABLE IV. MULTI LABEL CLASSIFICATION REPORT

Label	Classification Report for Standalone SVM			
	precision	*recall*	*F1-score*	*support*
DOS	88%	71%	79%	7460
Normal	65%	97%	78%	9711
Probe	59%	51%	55%	2421
R2L	0%	0%	0%	2885
U2R	0%	0%	0%	67

As opposed to a shallow classifier, our PCA+SVM incorporated Principal Component Analysis to linearly reduce the high feature space of network transmission. We saw a drop of accuracy from 71% to 55%; however, upon a closer look, we see that low-frequency attacks had a higher f1 scores. Table 5 outlines multiclass classification report exhibits a lower precision and recall for each attack type except for a better recall score for the 'probe' attack.

TABLE V. MULTI-LABEL CLASSIFICATION REPORT

	Classification Report for PCA+SVM			
	Precision	*Recall*	*F1-Score*	*Support*
DOS	90%	68%	78%	7460
Normal	62%	55%	58%	9711
Probe	48%	67%	56%	2421
R2L	19%	10%	13%	2885
U2R	1%	52%	2%	67

Moving on towards our proposed model, table 6 surpasses the baseline models; it shows a much higher accuracy of 75% which is 400 basis points higher than standalone SVM. Additionally, it shows higher Precision, Recall, and F1-score against each attack type across all models; except for PCA+SVM model had a higher recall for U2R of 4%.

TABLE VI. MULTI-LABEL CLASSIFICATION REPORT

	Classification Report for AE-SVM			
	Precision	*Recall*	*F1-Score*	*Support*
DOS	95%	73%	82%	7460
Normal	73%	92%	81%	9711
Probe	65%	76%	70%	2421
R2L	48%	24%	32%	2885
U2R	9%	48%	16%	67

D. Precision-Recall Curves

Precision-recall curve provides information regarding a classifier's output quality. Given the precision-recall tradeoff, we can conveniently plot their relationship on each classification threshold. As the decision threshold decreases, the false positive occurrences tend to increase, thus, increasing the recall metric. Using sklearn, we have computed the weighted average of precision and recall along every classification threshold: it is termed mean average precision (mAP). MAP condenses the precision-recall curve in a single figure. With 1 being the highest number and 0 being the lowest, high mAP indicate the overall performance of our models.

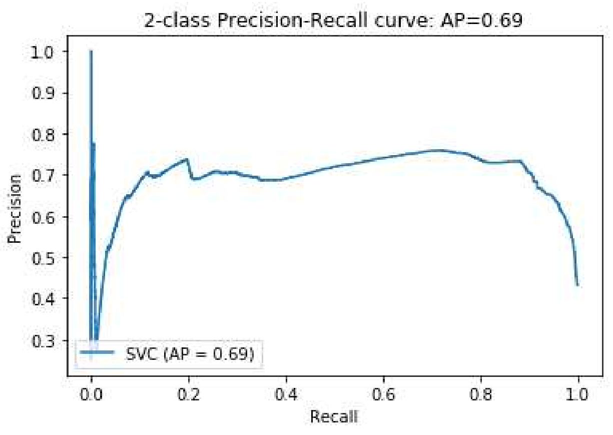

Fig. 7. Precision-Recall Curve for PCA-SVM in Binary Setting

Figures 7 and 8 present the precision-recall curves for our models in question. With average precision as our guiding principle, we find that PCA-SVM has an AP score of 0.69 and AE-SVM with a higher score of 0.90. Figure 8 has a bigger area under curve, indicating auto-encoded inputs as a clear choice compared to PCA-SVM.

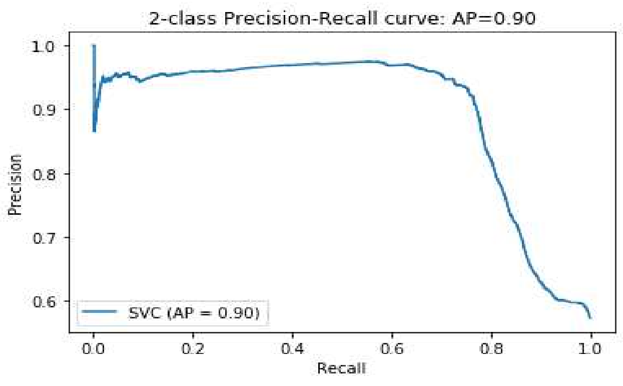

Fig. 8. Precision-Recall Curve for AE+SVM

Now we will closely interpret the Area under the curve (AUC) and f1-micro score for each attack type for PCA-SVM and AE-SVM. We selected micro f-1 score over macro as we wanted to center the discussion around each attack type given the class imbalance. Because precision-recall curves are drawn for binary classes, sklearn binarized each class and superimposed the output in one graph.

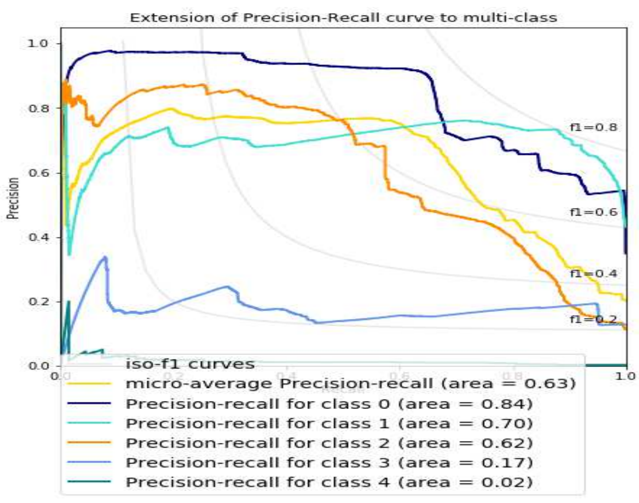

Fig. 9. PCA+SVM Multi-Class Precision-Recall Curves

Figure 9 summarizes the precision-recall curve for each attack type along with its corresponding Area under the curve metric. The PCA-SVM indicates a micro-average score of 0.63, compared to the AE-encoded inputs of 0.72 (refer to figure 10), suggesting that our deep autoencoder performs better in the multiclass scenario.

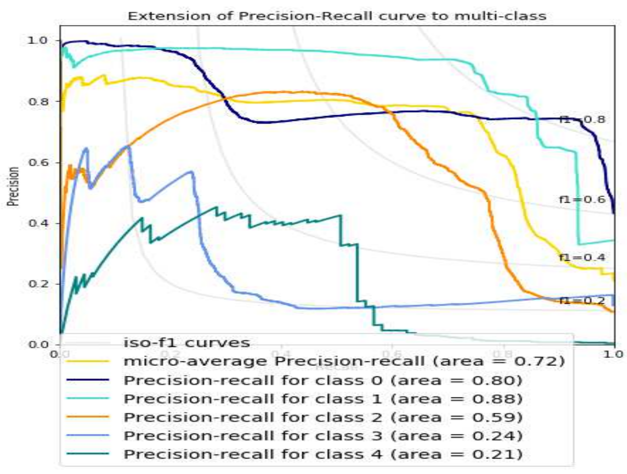

Fig. 10. AE+SVM Multi-Class Precision-Recall Curves

E. Performance Metrics: Train and Test Time

Train and test times are an important discussion, as it addresses the central problem of the paper: building a deployable lightweight anomaly detection system. Table 10 presented the training and test times for our baseline and proposed models. We also added the L1 and L2 regularization variants to see if it has any effect on resource computation. Even though the standalone SVM outdid the PCA-encoded SVM in the F-1 score, the time required to train the model was severely lagging. We find that the standalone classifier took a substantial amount of time to train in both multiclass and binarized scenarios. The DAE-SVM with L1 regularization was the model of choice for binary classes, while DAE-SVM without regularization component performed the best (142.62 seconds), with DAE-SVM L2 being a close second.

TABLE VII. TRAIN AND TEST TIME BASED ON SECONDS

Algorithm				
	Training Time (seconds)	***Testing Time (seconds)***	***Total Time (seconds)***	***Class***
SVM	392.70	39.68	432.38	Binary
DAE-SVM	59.13 + 86.60	2.28	148.01	Binary
DAE-SVM L1	52.55 + 90.49	2.80	145.84*	Binary
DAE-SVM L2	62.22 + 89.64	2.69	154.53	Binary
SVM	1545.64	133.23	1,678.87	Multi-Class
PCA-SVM	270.16	29.83	299.99	Multi-Class
DAE-SVM	48.09 + 89.75	7.78	145.62*	Multi-Class
DAE-SVM L1	49.20 + 121.04	10.72	180.96	Multi-Class
DAE-SVM L2	48.60 + 114.15	9.51	172.26*	Multi-Class

F. Conclusions

We will conclude our paper by addressing the three research questions posed earlier in the introduction section. First, we have, in fact, confirmed that DAE+SVM neural network scheme outperformed its PCA-SVM counterpart in a range of performance metrics. Given the importance of minimizing Type II errors, we put extra emphasis on precision and recall scores. False negatives have the potential to wreak havoc on host machines, therefore, are of central concern within anomaly detection. Excepting the higher 'Probe' recall score, our proposed model surpassed the baseline models in nearly all attack types because of DAE's superior feature fusion capabilities.

Second, we documented the training and test phases of the ML models in question; our proposed detection system had a significant advantage over the baseline models. Furthermore, we demonstrated through t-SNE embedding of high-dimensional feature space to a 2D plane that autoencoders exceeded at classifying attack types, thus, reducing analytical overhead.

Third, our findings indicate L1 regularization performs better than Lasso regularization on NSL-KDD. However, in a multiclass condition, our proposed model worked better without any penalty term. Our exhaustive parameter search algorithm, GridSearchCV, enabled us to make an informed decision in model selection. There is more to be done on this front; in future research, we would like to center RandomizedSearchCV as well as k-fold Cross-validation.

As we move towards industry 4.0, edge devices are becoming commonplace than cloud-led ones. Thus, there is a need in the industry for a security mechanism that is low-resource and highly accurate. Our paper contributes to the ongoing effort to incorporate neural networks in anomaly detection systems. Future research could build upon our study by exploring additional neural network architectures or incorporating other feature selection methods.

References

[1] "Microsoft Digital Defense Report 2022 | Microsoft Security." https://www.microsoft.com/en-us/security/business/microsoft-digital-defense-report-2022 (accessed Feb. 13, 2023).

[2] A. M. Dissanayaka, S. Mengel, R. R. Shetty, L. Gittner, S. Kothari, and R. Vadapalli, "A review of MongoDB and singularity container security in regards to HIPAA regulations," *UCC 2017 Companion - Companion Proceedings of the 10th International Conference on Utility and Cloud Computing*, pp. 91–97, Dec. 2017, doi: 10.1145/3147234.3148133.

[3] Mailewa Dissanayaka Akalanka *et al.*, "Secure NoSQL based medical data processing and retrieval: The Exposome project," *UCC 2017 Companion - Companion Proceedings of the 10th International Conference on Utility and Cloud Computing*, pp. 99–105, Dec. 2017, doi: 10.1145/3147234.3148132.

[4] "Cost of a data breach 2022 | IBM." https://www.ibm.com/reports/data-breach (accessed Feb. 18, 2023).

[5] S. Khan and A. B. Mailewa, "Discover botnets in IoT sensor networks: A lightweight deep learning framework with hybrid self-organizing maps," *Microprocess Microsyst*, vol. 97, p. 104753, Mar. 2023, doi: 10.1016/J.MICPRO.2022.104753.

[6] S. Thapa and A. M. Dissanayaka, "THE ROLE OF INTRUSION DETECTION/PREVENTION SYSTEMS IN MODERN COMPUTER NETWORKS: A REVIEW," 2020.

[7] A. Bandara Mailewa *et al.*, "Security threats/attacks via botnets and botnet detection & prevention techniques in computer networks: A Review Software Testing and Automation Project View project," 2019, Accessed: Feb. 13, 2023. [Online]. Available: https://www.researchgate.net/publication/334126617

[8] A. M. Dissanayaka, S. Mengel, L. Gittner, and H. Khan, "Vulnerability Prioritization, Root Cause Analysis, and Mitigation of Secure Data Analytic Framework Implemented with MongoDB on Singularity Linux Containers," *ACM International Conference Proceeding Series*, pp. 58–66, Mar. 2020, doi: 10.1145/3388142.3388168.

[9] A. M. Dissanayaka, S. Mengel, L. Gittner, and H. Khan, "Security assurance of MongoDB in singularity LXCs: an elastic and convenient testbed using Linux containers to explore vulnerabilities," *Cluster Comput*, vol. 23, no. 3, pp. 1955–1971, Sep. 2020, doi: 10.1007/S10586-020-03154-7.

[10] "$3 trillion is up for grabs in the cloud | McKinsey." https://www.mckinsey.com/capabilities/mckinsey-digital/our-insights/projecting-the-global-value-of-cloud-3-trillion-is-up-for-grabs-for-companies-that-go-beyond-adoption (accessed Feb. 18, 2023).

[11] S. Naseer *et al.*, "Enhanced network anomaly detection based on deep neural networks," *IEEE Access*, vol. 6, pp. 48231–48246, 2018, doi: 10.1109/ACCESS.2018.2863036.

[12] M. Sakurada and T. Yairi, "Anomaly detection using autoencoders with nonlinear dimensionality reduction," *ACM International Conference Proceeding Series*, vol. 02-December-2014, pp. 4–11, Dec. 2014, doi: 10.1145/2689746.2689747.

[13] D. J. Atul, R. Kamalraj, G. Ramesh, K. Sakthidasan Sankaran, S. Sharma, and S. Khasim, "A machine learning based IoT for providing an intrusion detection system for security," *Microprocess Microsyst*, vol. 82, p. 103741, Apr. 2021, doi: 10.1016/J.MICPRO.2020.103741.

[14] D. K. Kumar, "IoT-Edge Communication Protocol based on Low Latency for effective Data Flow and Distributed Neural Network in a Big Data Environment," *Microprocess Microsyst*, vol. 81, p. 103642, Mar. 2021, doi: 10.1016/J.MICPRO.2020.103642.

[15] M. K. Asif, T. A. Khan, T. A. Taj, U. Naeem, and S. Yakoob, "Network Intrusion Detection and its strategic importance," *BEIAC 2013 - 2013 IEEE Business Engineering and Industrial Applications Colloquium*, pp. 140–144, 2013, doi: 10.1109/BEIAC.2013.6560100.

[16] S. Mukkamala, G. Janoski, and A. Sung, "Intrusion detection using neural networks and support vector machines," *Proceedings of the International Joint*

Conference on Neural Networks, vol. 2, pp. 1702–1707, 2002, doi: 10.1109/IJCNN.2002.1007774.

[17] Y. Liao and V. R. Vemuri, "Use of K-Nearest Neighbor classifier for intrusion detection," *Comput Secur*, vol. 21, no. 5, pp. 439–448, Oct. 2002, doi: 10.1016/S0167-4048(02)00514-X.

[18] J. P. A. Yaacoub, O. Salman, H. N. Noura, N. Kaaniche, A. Chehab, and M. Malli, "Cyber-physical systems security: Limitations, issues and future trends," *Microprocess Microsyst*, vol. 77, p. 103201, Sep. 2020, doi: 10.1016/J.MICPRO.2020.103201.

[19] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," *IEEE Symposium on Computational Intelligence for Security and Defense Applications, CISDA 2009*, no. Cisda, pp. 1–6, 2009, doi: 10.1109/CISDA.2009.5356528.

[20] M. A. Nielsen, "Neural Networks and Deep Learning." Determination Press, 2015. Accessed: Feb. 18, 2023. [Online]. Available: http://neuralnetworksanddeeplearning.com

[21] Q. Xu, C. Zhang, L. Zhang, and Y. Song, "The learning effect of different hidden layers stacked autoencoder," *Proceedings - 2016 8th International Conference on Intelligent Human-Machine Systems and Cybernetics, IHMSC 2016*, vol. 2, pp. 148–151, 2016, doi: 10.1109/IHMSC.2016.280.

[22] B. Zoph and Q. v Le, "Searching for activation functions," *6th International Conference on Learning Representations, ICLR 2018 - Workshop Track Proceedings*, pp. 1–13, 2018.

[23] M. Premkumar and T. V. P. Sundararajan, "DLDM: Deep learning-based defense mechanism for denial of service attacks in wireless sensor networks," *Microprocess Microsyst*, vol. 79, p. 103278, Nov. 2020, doi: 10.1016/J.MICPRO.2020.103278.

[24] Y. Wang, H. Yao, and S. Zhao, "Auto-encoder based dimensionality reduction," *Neurocomputing*, vol. 184, no. November, pp. 232–242, 2016, doi: 10.1016/j.neucom.2015.08.104.

[25] L. J. P. van der Maaten, E. O. Postma, and H. J. van den Herik, "Dimensionality Reduction: A Comparative Review," *Journal of Machine Learning Research*, vol. 10, pp. 1–41, 2009, doi: 10.1080/13506280444000102.

[26] D. Charte, F. Charte, S. García, M. J. del Jesus, and F. Herrera, "A practical tutorial on autoencoders for nonlinear feature fusion: Taxonomy, models, software and guidelines", Accessed: Feb. 14, 2023. [Online]. Available: https://doi.org/10.

[27] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P. A. Manzagol, "Stacked denoising autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion," *Journal of Machine Learning Research*, vol. 11, pp. 3371–3408, 2010.

[28] S. Rifai, P. Vincent, X. Muller, X. Glorot, and Y. Bengio, "Contractive auto-encoders: Explicit invariance during feature extraction," *Proceedings of the 28th International Conference on Machine Learning, ICML 2011*, no. 1, pp. 833–840, 2011.

[29] J. Masci, U. Meier, D. Cireşan, and J. Schmidhuber, "Stacked convolutional auto-encoders for hierarchical feature extraction," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 6791 LNCS, no. PART 1, pp. 52–59, 2011, doi: 10.1007/978-3-642-21735-7_7.

[30] S. Khan, "Autoencoder-Based Representation Learning to Predict Anomalies in Computer Networks," *Culminating Projects in Information Assurance*, May 2020, Accessed: Feb. 18, 2023. [Online]. Available: https://repository.stcloudstate.edu/msia_etds/102

[31] O. Demir-Kavuk, M. Kamada, T. Akutsu, and E. W. Knapp, "Prediction using step-wise L1, L2 regularization and feature selection for small data sets with large number of features," *BMC Bioinformatics*, vol. 12, no. 1, pp. 1–10, Oct. 2011, doi: 10.1186/1471-2105-12-412/TABLES/2.

[32] A. Shenfield, D. Day, and A. Ayesh, "Intelligent intrusion detection systems using artificial neural networks," *ICT Express*, vol. 4, no. 2, pp. 95–99, Jun. 2018, doi: 10.1016/J.ICTE.2018.04.003.

[33] M. Al-Qatf, Y. Lasheng, M. Al-Habib, and K. Al-Sabahi, "Deep Learning Approach Combining Sparse Autoencoder with SVM for Network Intrusion Detection," *IEEE Access*, vol. 6, pp. 52843–52856, 2018, doi: 10.1109/ACCESS.2018.2869577.

[34] M. Yousefi-Azar, V. Varadharajan, L. Hamey, and U. Tupakula, "Autoencoder-based feature learning for cyber security applications," *Proceedings of the International Joint Conference on Neural Networks*, vol. 2017-May, pp. 3854–3861, 2017, doi: 10.1109/IJCNN.2017.7966342.

[35] N. Shone, T. N. Ngoc, V. D. Phai, and Q. Shi, "A Deep Learning Approach to Network Intrusion Detection," *IEEE Trans Emerg Top Comput Intell*, vol. 2, no. 1, pp. 41–50, 2018, doi: 10.1109/tetci.2017.2772792.

[36] B. Lee, S. Amaresh, C. Green, and D. Engels, "Comparative Study of Deep Learning Models for Network Intrusion Detection," *SMU Data Science Review*, vol. 1, no. 1, p. 8, 2018.

[37] N. Seliya, T. M. Khoshgoftaar, and J. van Hulse, "A study on the relationships of classifier performance metrics," *Proceedings - International Conference on Tools with Artificial Intelligence, ICTAI*, pp. 59–66, 2009, doi: 10.1109/ICTAI.2009.25.

[38] L. van der Maaten and G. Hinton, "Visualizing Data using t-SNE," *Journal of Machine Learning Research*, vol. 9, pp. 2579–2605, 2008.