Automatic Features Extraction Using Autoencoder in Intrusion Detection System

Yesi Novaria Kunang
Intelligent System Research Group
Faculty of Computer Science
Department,
Universitas Sriwijaya
Palembang, Indonesia
yesinovariakunang@binadarma.ac.id

Siti Nurmaini *
Intelligent System Research Group
Faculty of Computer Science
Department,
Universitas Sriwijaya
Palembang, Indonesia
sitinurmaini@gmail.com

Deris Stiawan
Computer Networking & Information
System, Faculty of Computer Science
Departement,
Universitas Sriwijaya
Palembang, Indonesia
deris@unsri.ac.id

Ahmad Zarkasi
Intelligent System Research Group
Faculty of Computer Science
Department,
Universitas Sriwijaya
Palembang, Indonesia
zarkasi98@gmail.com

Firdaus
Intelligent System Research Group
Faculty of Computer Science
Department,
Universitas Sriwijaya
Palembang, Indonesia
virdauz@gmail.com

Jasmir
Intelligent System Research Group
Faculty of Computer Science
Department,
Universitas Sriwijaya
Palembang, Indonesia
ijay_jasmir@yahoo.com

***Abstract*— Intrusion Detection System (IDS) can detect attacks by analysing the patterns of data traffic in the network. With a large amount of data that is processed in the IDS, then need to do a feature extraction to reduce the computational cost of processing raw data in IDS. Feature extraction will transform features to the lower dimension to accelerate the learning process and improve the accuracy. This research on automatic feature extraction using simple autoencoder and SVM to classify attacks on IDS. We use various functions activation and loss to see how far this feature extraction feature can improve accuracy. We use Datasets KDD Cup ' 99 NSL-KDD and to evaluate the effectiveness of the mechanisms of detection after extraction features process. In the proposed model, the activation functions autoencoder hyperparameter ReLU activation and loss function cross-entropy gives best accuracy value than other functions.**

Keywords—Intrusion Detection System; Machine Learning, Features Extraction, Autoencoder

I. INTRODUCTION

Feature extraction is a process of transforming the existing features into a lower dimensional space. The feature extraction algorithm has improved the accuracy of detection and segmentation in various fields, including digital images or video streams [1], audio and language recognition [2], text document classification [3], biomedical signal processing [4] and speech recognition [5]. The learning process may increase the accuracy by reducing the data with the Principal Component Analysis (PCA) [6] and Linear Discriminant Analysis (LDA) [7].

Some researches on features extraction use a mathematical approach to features extraction such as Multidimensional Scaling (MDS), ISOMAP, Locally Linear Embedding (LLE) and Laplacian Eigenmaps (LE) [8]. Other issues have proposed mapping data into a lower dimension to extracting features with a single process. Unfortunately, the approach will be less accurate if the data extracted is extensive. One of the machine learning techniques to overcome the problem of feature extraction is using Autoencoder (AE).

AE uses the concept of an artificial neural network to reduce dimensionality with the stack on the hidden layer by minimising the reconstruction layer. Autoencoder with its variants such as Sparse Autoencoder, Variational Autoencoder [9], Denoising Autoencoder [10], and Relational Autoencoder [8], demonstrated the ability of the autoencoder to extract meaningful features.

Feature extraction is one of the stages in classification in the field of Informatics Engineering, especially in the area of network and information security. One of the challenges is the increase in volume traffic data in the growing network. As the impact of the rise in connectivity by of the Internet of Things and cloud-based services. With increasingly more extensive and complex data as well as a large number of new attack types have an impact on system IDS to detect normal and abnormal behaviour. There have been many methods used in IDS to identify attacks by using the Decision tree [11], Back Propagation Neural Network [12], K-Means [13] and SVM [14]. Unfortunately, the lack of such models is the feature extraction process is not automatic. Learning process using feature extraction can effectively detect attacks [15]. Features extraction become an essential issue in the transformation of raw data into features suitable to improve the accuracy of the IDS algorithm.

In these early studies, we propose to implement the automatic extraction of features on the intrusion detection system (IDS). We use Autoencoder which will perform automatic features extraction to reduce the dimensions of the data being processed. The aim is to study the various functions of the activation and the loss autoencoder hyperparameter can increase the level of accuracy of attack detection. SVM is used to evaluate the level of accuracy scores.

Yesi Novaria Kunang is with the Department Computer Science at Bina Darma University Palembang.

II. PREVIOUS WORK

Many models have been proposed in earlier studies to overcome the limitations of the anomaly intrusion detection system. This section will discuss some approaches related to feature extraction on IDS. In the traditional approach, many intrusion detection models use features selection as a preprocessing step to reduce data dimensions of network traffic. Chae and Choi [11] develops feature selection models using the average total value of each class and implements the decision tree classifier algorithm to evaluate the features reduction method. For the highest accuracy obtained using 22 features.

Associated with feature extraction, some research focuses on reducing feature dimensions without losing information. Liu et al. [16] developed an anomaly detection model using feature reduction with Principal Component Analysis (PCA) and classified using Neural Networks. This approach extracts only 22 features from 41 features. The main advantage of this approach is the reduced computation time as the number of features decreases. However, the selection of the principal component is not optimal because it only processes certain features. Datti and Lakhina [17] compares the reduction feature of Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA) techniques. Having reduced the classification is done using the back-propagation algorithm. The result shows PCA is more accurate than LDA for smaller datasets, whereas for more massive datasets LDA is more accurate.

Kuang et al. [18] combine Kernel Principal Component Analysis (KPCA) and SVM for intrusion detection. Multi-layer classifier model is performed to determine attacks. In the same year, Thaseen and Kumar [19] developed an intrusion detection model based on the principal component analysis (PCA) and support vector machines (SVM) using RBF kernel. With SVM optimisation parameters for the RBF kernel and cross-validation successfully reduced training and testing time. It has better accuracy for minor U2R and R2L attacks.

On the other hand, Aminanto [15] introduced the Deep-Feature Extraction and Selection (D-FES) method that combines stacked feature extraction and weighted feature selection techniques to detect impersonation attacks on Wi-Fi networks. However, the earlier study no one has comprehensively examined in detail hyperparameter autoencoder as extraction features associated with the effect of the activation function and loss function of the level of accuracy produced.

III. METHOD AND DESIGN

A. Autoencoder

Autoencoder is one of unsupervised machine learning algorithm on the artificial neural network. Autoencoder is trained to reconstruct output near original input. Autoencoder consists of an input layer, output layer (with the same number of dimensions as the input layer) and the hidden layer which usually has a dimension smaller than the input. An example of a simple autoencoder can be shown in figure 1.

One of the characteristics possessed by an autoencoder is more powerful for finding data structures by reducing data with non-linear transformations than Principle Component Analysis (PCA) [20], [21]. The process is done based on the encoder-decoder paradigm, by applying backpropagation and setting the target value equal to the input. The input is first transformed into a lower-dimensional layer (encode), and then expand to reproduce the initial data (decoder). After the layer was trained, the output of the layer is forwarded to the next layer to obtain a highly non-linear dependencies model on the input. This process aims to reduce the dimensions of input data. Encoded layer in the middle of the autoencoder is used as a feature extracted for classification [20].

The neural network with one hidden layer has the encoder equation in equation (1) and equation (2) on the decoder.

$$Y = f_\theta(X) = s(WX + b_X) \qquad (1)$$

$$X' = g_{\theta'}(Y) = s(W'Y + b_Y) \qquad (2)$$

Where f(X) is an encoded function g(Y) is a decoded function, the parameter W (weighting), and b (bias) for data x. S is a function activation. The decoder function g maps the hidden representation of Y back to reconstruct X '.

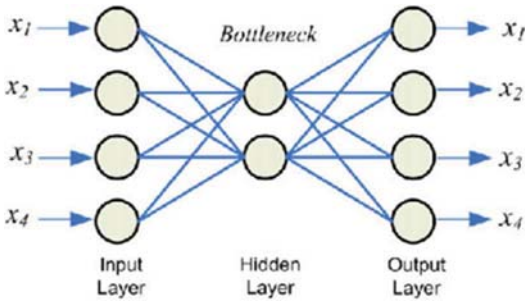

Fig. 1. Simple Autoencoder

The autoencoder training process is used to determine the parameters $\theta = (W, b_X, b_Y)$ to minimize reconstruction loss on the X dataset with the objective function given as follows:

$$\theta = \min L(X, X') = \min L(X, g((f(X)))) \qquad (3)$$

For Linear reconstruction, Loss Reconstruction (L_1) is obtained from squares error:

$$L_1(\theta) = \sum_{i=1}^{n} ||x_i - x'_i||^2 \qquad (4)$$

As for nonlinear functions, reconstruction loss (L_2) is calculated based on cross-entropy:

$$L_2(\theta) = -\sum_{i=1}^{n} [x_i \log(y_i) + (1 - x_i)\log(1 - y_i)] \qquad (5)$$

B. Model of Automatic Feature Extraction

The model proposed in the Preprocessing stage for the preparation of data in the Deep Learning process for attack detection shown in figure 2.

The initial stage of this feature extraction is Preprocessing feature selection of 42 features available. In this study, we use One Hot Encoding. Another researcher using ordinal coding for variable encoding Technique [22]. We select this approach

because it gives a better result than ordinal coding [23]. We handle all the features including non-numerical. Non-numerical features data in the dataset are converted to numerical data (especially on features 2, 3 and 4 of protocol type, service, and flag). For each certain level, the variable on the map forms the dummy variable in binary form [23]. In this case, the features used in the next stage to 120 features and 1 feature label.

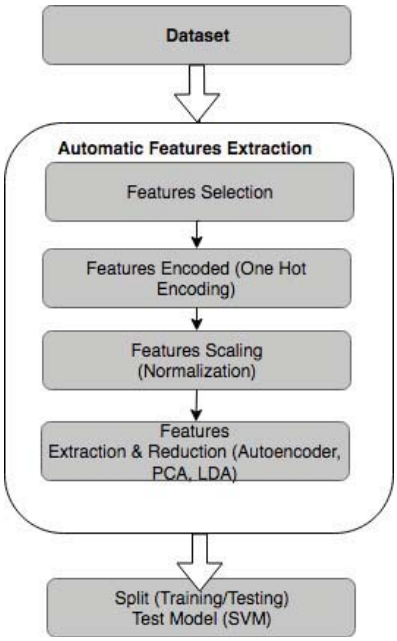

Fig. 2. Proposed Model for Features Extraction

Features labels were grouped into numeric categories consisting of normal data, DOS, Probe, R2L, and attacks that are mapped into integer values as in table 2 and table 3.

For some features not to dominate the other features, it is necessary to scale all the features. This process is called Feature scaling. There are several alternatives used to perform feature scaling that is widely used is Standardization, Scaling, and Normalization. In this research, we used Z Score Normalization. This method is commonly used in machine learning algorithms (such as SVM, logistic regression and neural network) [24]. Normalized values are calculated by determining the mean and standard deviation of each feature.

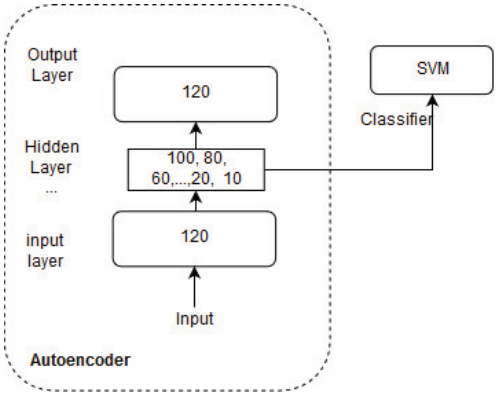

Fig. 3. Features Extraction Results for Autoencoder

The next process is dimension reduction by utilising the Autoencoder model. The autoencoder model we used in this preliminary study uses a simple autoencoder consisting of 3 layers with 1 hidden layer in it. We use One Hot coding model for preprocessing. The autoencoder process includes of 120 input neurons. In the hidden layer, we transformed X (repeatedly into 100, 80, 60, 40, 22, 20, 15, 10 and 8 neurons to get the best accuracy value (see figure 3). The output in the hidden layer becomes the features extracted according to the number of neurons. The data have extracted become smaller dimensions of the features. The output can be processed in training and testing.

C. Datasets.

KDD Cup '99 dataset is a standard dataset used for surveys and research in the field of attack detection developed by MIT Lincoln Labs [25]. This dataset itself is old and has flaws as suggested by Mc Huge [26] and may be irrelevant for network real today. However, current research NIDS still often use it as a benchmark in research to compile different methods.

KDD Cup '99 Dataset consists of 24 types of attacks and 42 attributes. The attacks themselves are divided into four main categories namely DOS, R2L, U2R, and Probing. The data were tested using 10% KDD data totalling 494,021 as training records and data corrected 311,029 as testing records.

We also evaluated our model on NSL-KDD dataset. The NSL-KDD dataset is a development of KDD Cup '99 [27]. For training data using data train as much as 125,973 records and data testing as much as 18,794 records. NSL-KDD data is almost the same as KDD Cup '99 consisting of 42 attributes.

D. Classifier

We used the SVM algorithm for 5 class classification. Based on several studies showed the effectiveness and suitability of SVM in IDS classification [14].

At the initial stage to see the activation function and loss in the autoencoder process we use Training-Testing split with ratio 70-30 from KDD Cup Data '99 training data. This ratio is commonly used for train/test split [28]. From this stage, the best epoch and batch values are obtained, as well as activation and optimisation functions that provide the best accuracy values. Then cross-validation testing (cv = 10) was carried out with epoch, batch and activation functions which gave the best accuracy. This cross-validation process is to see more accurate values of the model. The value of k = 10 is a value commonly used in machine learning [29].

In the next stage, KDD Cup '99 data is tested with testing data (corrected data) for 5 class classification. The model is also tested for NSL KDD data training and testing data in 5 class classification. Table 2 and 3 show data for training and testing.

We try our SVM Classifier model using kernel RBF, linear and Polynomial kernel. Other parameters like gamma, degree, and others use the default values of scikit learn. In the next step, we will tune other parameters.

IV. EXPERIMENTAL RESULTS

A. Model Hyperparameter

The evaluation process is run using GPU-enabled TensorFlow. It was running on a Windows 10 PC Operating System with Intel Core i7-7700HQ processor, 32 Gb RAM and Nvidia GTX 1060 6GB.

After it extracted using autoencoder, the dimension transformed become 100, 80, 60, 40, 22, 20, 15, 10 and 8 features. Features extraction used a simple autoencoder model with one hidden layer. We investigate hyper-parameter epochs 50, 100, 150, 200 and 250. Hyper-parameter mini-batch size respectively 64, 128, 256 (powers of 2). For KDD Cup '99, the best value hyper-epochs and batch size parameters are 200 and 256. Whereas for NSL-KDD epochs and batch size are 200-128.

We applied backpropagation and Adam optimiser in our model. The empirical results show that Adam optimiser works better than other stochastic optimisation methods [30].

In this paper, Automatic Features Extraction used function loss MSE and cross-entropy as a comparison. Activation function linear, sigmoid, ReLU, SoftMax, SoftPlus, SoftSign and tanh activation on the hidden layer are evaluated for the most suitable function activation for the IDS model. As benchmarks, we compare with PCA and LDA.

The output of the feature extraction results was evaluated using SVM to see the level of classification accuracy and times processing for feature extraction. In the experiment, we also carried out the processing of machine learning classification without features extraction to evaluate how automatic features extraction can extract important information from the initial features. Table 1 shows The results of feature extraction (20 neurons) for KDD Cup ’99 datasets. It is obtained by classifying SVM using the RBF kernel. In our tests for SVM linear and polynomial kernel using default parameters, it gives lower accuracy results. From several experiments conducted for the number of features, the number of features 20 for KDD Cup '99 gives the highest accuracy value (see figure 4).

Automatic features extraction successfully retains important information on the features before being reduced. Almost all of the various features extraction function activation and function loss give higher accuracy than the SVM models without any extraction features.

From table 1, the best feature extraction model is using ReLU activation and cross-entropy loss. Accuracy rate is higher than the machine learning model without automatic features extraction that is 99.947 (increased 0.024%). Other activation functions that also show better results are on features extraction models using activation linear and Softplus and loss cross-entropy (increased 0.016%). Softmax activation models show the longest optimisation process during the training process.

For the loss function in the automatic extraction, features show function loss cross-entropy give a more accurate result. It means the loss cross-entropy function gives local optimum better than MSE (in line with Golik et al. [31]).

To evaluate the model, we used 10-fold cross-validation in the dataset. For NSL-KDD dataset, the final structured model is 99.464 +/- 0.019. The average accuracy of KDD cup '99 value is 99.447 +/- 0.048.

B. 5 Class Classifications

The final model of Automatic features extraction was tested in KDD Cup '99 Training and Testing data, to get best results from the number of features extracted. Figure 4, it can be seen that KDD Cup '99 data produces the best accuracy performance of 99,358 for features extraction with the number of neurons 20, linear SVM kernel and ReLU Activation function.

Figure 5 shows the performance of the model in testing NSL-KDD data using training and testing data. The best accuracy value is 86.964% on the 22 features extracted. These results are obtained using the Relu Activation function and the RBF SVM kernel.

Table 2 and 3 show the overall performance. For the model proposed, it produces better accuracy, precision, and FScore than SVM without autoencoder or SVM using feature reduction with PCA and LDA.The result proves the automatic feature extraction model has succeeded in extracting and transforming intrusion data. All models of Automatic Features Extraction with autoencoder work more powerful (in line with other studies [32]). While for LDA features extraction indicates the level of accuracy of approaching the process of SVM without features extraction.

TABLE I. THE PERFORMANCES AE-SVM IDS FOR KDD CUP 99 DATASET (20 NEURONS IN HIDDEN LAYER)

Methods	Number of features	Accuracy (%)	Time for Feat. Extract (ms)	Training Time (ms)	Testing Time (ms)
SVM (without features extraction)	120	99.923	0	889.11	45.399
SVM-AE (actv=linear, loss=MSE)	20	99.868	1822.686	57.397	12.096
SVM-AE (actv=linear, loss= cross-entropy)	20	99.939	2427.775	59.931	9.784
SVM-AE (actv=sigmoid, loss= MSE)	20	99.937	1849.008	46.071	7.833
SVM-AE (actv=sigmoid, loss=cross-entropy)	20	99.926	2542.503	42.939	8.333
SVM-AE (actv=ReLU, loss=MSE)	20	99.939	2778.225	59.964	11.482
SVM-AE (actv=ReLU, loss=cross-entropy)	20	99.947	2331.131	56.61	7.453
SVM-AE (actv=SoftMax, loss=MSE)	20	99.894	2123.37	1955.198	13.836
SVM-AE (actv=SoftMax, loss=cross-entropy)	20	99.883	2303.416	2416.059	17.715
SVM-AE (actv=Softplus, loss=MSE)	20	99.939	2337.375	56.724	9.235
SVM-AE (actv=Softplus, loss=cross-entropy)	20	99.939	2696.421	58.52	8.993
SVM-AE (actv=Softsign, loss=MSE)	20	99.922	2600.325	109.234	25.627
SVM-AE (actv=Softsign, loss=cross-entropy)	20	99.936	2693.109	49.867	9.069
SVM-AE (actv=tanh, loss=MSE)	20	99.939	2309.318	51.203	7.377
SVM-AE (actv=tanh, loss=cross-entropy)	20	99.928	2861.439	72.119	11.883

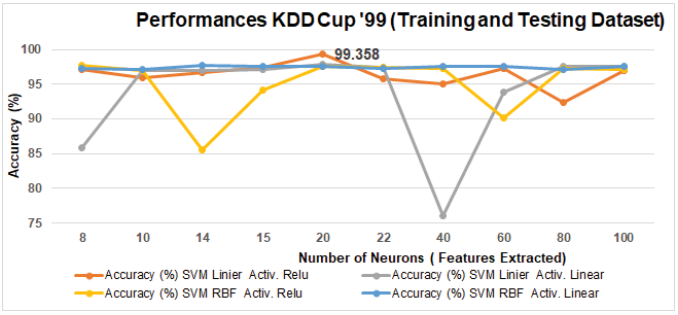

Fig. 4. Overall Accuracy in KDD Cup' 99 data Training and Testing.

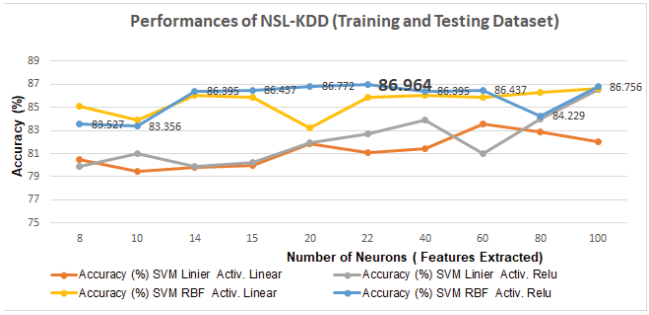

Fig. 5. Overall Accuracy in NSL-KDD data Training and Testing

TABLE II. KDD CUP '99 5 CLASS PERFORMANCE

Attack Class	No. Training	No. Testing	Accuracy (%)				Precision (%)				F-Score (%)			
			SVM	SVM-AE	SVM-PCA	SVM-LDA	SVM	SVM-AE	SVM-PCA	SVM-LDA	SVM	SVM-AE	SVM-PCA	SVM-LDA
DoS	391458	223298	99.932	99.965	75.066	99.507	99.704	99.758	98.847	99.513	99.818	99.861	85.33	99.51
Probe	4107	2377	82.162	89.062	27.85	91.965	91.134	91.289	1.131	92.706	86.416	90.162	2.174	92.334
R2L	1126	5993	14.484	88.27	0	16.186	98.19	95.436	0	95.85	25.244	91.713	0	27.695
U2R	52	39	0	23.077	0	0	0	6.338	0	0	0	9.945	0	0
normal	97278	60593	98.686	98.67	95.475	98.086	91.343	98.763	90.199	90.538	94.873	98.716	92.762	94.161
Total	494021	292300	97.764	**99.358**	77.364	97.43	97.857	**99.382**	94.219	97.509	97.142	**99.366**	84.434	96.857

TABLE III. NSL-KDD 5 CLASS PERFORMANCE

Attack Class	No. Training	No. Testing	Accuracy (%)				Precision (%)				F-Score (%)			
			SVM	SVM-AE	SVM-PCA	SVM-LDA	SVM	SVM-AE	SVM-PCA	SVM-LDA	SVM	SVM-AE	SVM-PCA	SVM-LDA
DoS	45927	5741	99.129	97.91	56.053	93.207	97.033	99.452	64.104	98.31	98.07	98.675	59.809	95.69
Probe	11656	1106	91.591	88.065	37.071	87.432	60.119	78.107	13.585	61.164	72.59	82.788	19.884	71.976
R2L	995	2199	11.642	12.779	0.045	0	92.754	97.569	100	0	20.687	22.598	0.091	0
U2R	52	37	24.324	8.108	0	0	69.231	50	0	0	36	13.953	0	0
normal	67343	9711	91.947	97.467	84.626	92.926	81.506	81.588	76.411	76.669	86.412	88.823	80.309	84.018
Total	125973	18794	84.591	**86.964**	63.036	81.632	86.282	**88.648**	71.564	73.246	81.371	**83.581**	60.947	76.879

Processing time, automatic feature extraction takes a longer time to extract features compared with the features of reduction using PCA, LDA. It caused, the local optima process run in the hidden layer is sophisticated enough to achieve the convergence of weight values during the autoencoder training process. The optimisation process depends on the number of epoch and batch size. But for the training process in SVM, particularly for large data testing, automatic feature extraction much faster than the method of training at PCA and LDA. To further evaluate it needs to be tested with other dataset and with larger data. Using more extensive data and lower accuracy rates, we can analyze noise factor using more complex autoencoder model.

C. Comparison With Related Work

In contrast, we compared our results with previous research. Author [33] divides 70% training data and 30% for KDD data and classifies attacks for five classes for KDD data. Accuracy reaches only 99.85%, while our model reach 99,947 (table 1).

For testing the NSL-KDD training and testing data, the accuracy we got was also better compared to [33], those using the J48 algorithm only reached 84.84% while we reached 86.964% (table 3). This result is also better than [34] whose accuracy is only 83.7%. With feature selection that uses a decision tree using 14 features. Shone [35] for the same test the accuracy value just reached 85.42%, but the precision value produced is almost 100%. They use more layers (3 hidden layers and 2 stacked layers). Whereas in this initial research we only used 1 hidden layer.

For KDD Cup 99 data with training and testing data, our accuracy is better than the deep autoencoder model [36] or [35] The accuracy values we got were 99,358 (table 2), while Farahnakian [36] claimed their accuracy reached 94.71% and Shone [35] 97.85%. However, for precision with the deep learning model S-NDAE [35] gives a better precision value of 99.99%.

The selection of the encoding technique affects the accuracy of the data even not significant. The one hot coding maps the categorical data into a binary categorical form in features shows better accuracy than using ordinal coding. The encoding technique is important because several studies that research IDS use ordinal coding [22], [35], which maps features of only 41 features. Mapping the categorical data with the number format will result in the false structure (since in fact, the categorical data has no ordering).

In general, the results of the feature extraction models tested, provide promising results to adapt to more complex IDS systems with deep learning models for a larger dataset.

V. Conclusions and Future Works

This initial research focuses on the process of automatic features extraction on IDS data. Test results Automatic features extraction by using autoencoder shows the dimensional reduction process performed ably to maintain the relation of information from the features that are compressed. It shows from the value of accuracy using ReLU activation, and linear activation with cross-entropy loss function gives a better result than the process without extraction features and linear transformations using PCA or LDA.

From the testing done the best hyperparameter model for autoencoder is a ReLU activation function with cross-entropy as loss function. The accuracy value of this model reaches 99.947% with fast relative processing times compared to other functions.

The limitations of this study are only done using less experiment of the dataset used. The dataset is less up to date for the networks technology and the latest types of attacks. For the future researchers, we will evaluate for newer, more complex datasets, with more extensive data. We plan to develop a further hybrid clustering model on the IDS system with a more complex autoencoder process.

References

[1] Yuhandri, S. Madenda, E. P. Wibowo, and - Karmilasari, "Object Feature Extraction of Songket Image Using Chain Code Algorithm," *International Journal on Advanced Science, Engineering and Information Technology*, vol. 7, no. 1, p. 235, Feb. 2017.

[2] F. Richardson, D. Reynolds, and N. Dehak, "Deep Neural Network Approaches to Speaker and Language Recognition," *IEEE Signal Processing Letters*, vol. 22, no. 10, pp. 1671–1675, Oct. 2015.

[3] C. H. P. Ferreira, D. M. R. de Medeiros, and F. O. de França, "DCDistance: A Supervised Text Document Feature extraction based on class labels," *arXiv:1801.04554 [cs]*, Jan. 2018.

[4] R. Jenke, A. Peer, and M. Buss, "Feature Extraction and Selection for Emotion Recognition from EEG," *IEEE Transactions on Affective Computing*, vol. 5, no. 3, pp. 327–339, Jul. 2014.

[5] J. Barker, R. Marxer, E. Vincent, and S. Watanabe, "The third 'CHiME' speech separation and recognition challenge: Dataset, task and baselines," 2015, pp. 504–511.

[6] U. Demšar, P. Harris, C. Brunsdon, A. S. Fotheringham, and S. McLoone, "Principal Component Analysis on Spatial Data: An Overview," *Annals of the Association of American Geographers*, vol. 103, no. 1, pp. 106–128, Jan. 2013.

[7] A. Sharma and K. K. Paliwal, "Linear discriminant analysis for the small sample size problem: an overview," *International Journal of Machine Learning and Cybernetics*, vol. 6, no. 3, pp. 443–454, Jun. 2015.

[8] Q. Meng, D. Catchpoole, D. Skillicorn, and P. J. Kennedy, "Relational Autoencoder for Feature Extraction," *arXiv:1802.03145 [cs, stat]*, pp. 364–371, May 2017.

[9] J. An and S. Cho, "Variational Autoencoder Based Anomaly Detection Using Reconstruction Probability," Technical Report, 2015.

[10] Y. Wang, W. Cai, and P. Wei, "A Deep Learning Approach for Detecting Malicious Javascript Code," *Security Comm. Networks*, vol. 9, no. 11, pp. 1520–1534, Jul. 2016.

[11] H. Chae and S. H. Choi, "Feature Selection for efficient Intrusion Detection using Attribute Ratio," vol. 8, p. 6, 2014.

[12] B. Shah and B. H. Trivedi, "Reducing Features of KDD CUP 1999 Dataset for Anomaly Detection Using Back Propagation Neural Network," 2015, pp. 247–251.

[13] D. Y. Mahmood and M. A. Hussein, "Feature-Based Unsupervised Intrusion Detection," vol. 8, no. 9, p. 5, 2014.

[14] P. Kushwaha, H. Buckchash, and B. Raman, "Anomaly-based intrusion detection using filter-based feature selection on KDD-CUP 99," in *Region 10 Conference, TENCON 2017-2017 IEEE*, 2017, pp. 839–844.

[15] M. E. Aminanto, R. Choi, H. C. Tanuwidjaja, P. D. Yoo, and K. Kim, "Deep Abstraction and Weighted Feature Selection for Wi-Fi Impersonation Detection," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 3, pp. 621–636, Mar. 2018.

[16] G. Liu, Z. Yi, and S. Yang, "A hierarchical intrusion detection model based on the PCA neural networks," *Neurocomputing*, vol. 70, no. 7–9, pp. 1561–1568, Mar. 2007.

[17] R. Datti and S. Lakhina, "Performance Comparison of Features Reduction Techniques for Intrusion Detection System," vol. 3, no. 1, p. 4, 2012.

[18] [18] F. Kuang, W. Xu, and S. Zhang, "A novel hybrid KPCA and SVM with GA model for intrusion detection," *Applied Soft Computing*, vol. 18, pp. 178–184, May 2014.

[19] I. S. Thaseen and C. A. Kumar, "Intrusion detection model using fusion of PCA and optimized SVM," 2014, pp. 879–884.

[20] Y. Wang, H. Yao, and S. Zhao, "Auto-encoder based dimensionality reduction," *Neurocomputing*, vol. 184, pp. 232–242, Apr. 2016.

[21] T. Manning-Dahan, "PCA and Autoencoders."

[22] S. Potluri and C. Diedrich, "Accelerated deep neural networks for enhanced Intrusion Detection System," 2016, pp. 1–8.

[23] K. Potdar, T. S., and C. D., "A Comparative Study of Categorical Variable Encoding Techniques for Neural Network Classifiers," *International Journal of Computer Applications*, vol. 175, no. 4, pp. 7–9, Oct. 2017.

[24] A. Harasimowicz, "Comparison of Data Preprocessing Methods and the Impact on Auto-encoders Performance in Activity Recognition Domain," *ICT Young 2014*, p. 6, Sep. 2014.

[25] "KDD Cup 1999 Data.", [online] Available: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html.

[26] J. McHugh, "Testing intrusion detection systems: a critique of the 1998 and 1999 DARPA intrusion detection system evaluations as performed by Lincoln laboratory," *ACM Transactions on Information and System Security (TISSEC)*, vol. 3, no. 4, pp. 262–294, 2000.

[27] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," 2009, pp. 1–6.

[28] H. Liu and M. Cocea, "Semi-random partitioning of data into training and test sets in granular computing context," *Granular Computing*, vol. 2, no. 4, pp. 357–386, Dec. 2017.

[29] M. Kuhn and K. Johnson, *Applied predictive modelling*, vol. 26. Springer, 2013.

[30] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *arXiv:1412.6980 [cs]*, Dec. 2014.

[31] P. Golik, P. Doetsch, and H. Ney, "Cross-entropy vs. squared error training: a theoretical and experimental comparison.," in *Interspeech*, 2013, vol. 13, pp. 1756–1760.

[32] J. Almotiri, K. Elleithy, and A. Elleithy, "Comparison of autoencoder and Principal Component Analysis followed by neural network for e-learning using handwritten recognition," 2017, pp. 1–5.

[33] N. Paulauskas and J. Auskalnis, "Analysis of data pre-processing influence on intrusion detection using NSL-KDD dataset," in *Electrical, Electronic and Information Sciences (eStream), 2017 Open Conference of*, 2017, pp. 1–5.

[34] B. Ingre, A. Yadav, and A. K. Soni, "Decision Tree Based Intrusion Detection System for NSL-KDD Dataset," in *Information and Communication Technology for Intelligent Systems (ICTIS 2017) - Volume 2*, vol. 84, S. C. Satapathy and A. Joshi, Eds. Cham: Springer International Publishing, 2018, pp. 207–218.

[35] N. Shone, T. N. Ngoc, V. D. Phai, and Q. Shi, "A deep learning approach to network intrusion detection," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 2, no. 1, pp. 41–50, 2018.

[36] F. Farahnakian and J. Heikkonen, "A deep auto-encoder based approach for intrusion detection system," in *Advanced Communication Technology (ICACT), 2018 20th International Conference on*, 2018, pp. 178–183.