

A Comparative Study of Off-Line Deep Learning Based Network Intrusion Detection

Jiaqi Yan*, Dong Jin*, Cheol Won Lee[†] and Ping Liu*

*Department of Computer Science, Illinois Institute of Technology, Chicago, Illinois 60616

Email: jyan31@hawk.iit.edu, dong.jin@iit.edu, pliu19@hawk.iit.edu

[†]National Security Research Institute, 1559 Yuseong-daero, Yuseong-gu, Daejeon, South Korea

Email: cheolee@nsr.re.kr

Abstract—Network intrusion detection systems (NIDS) are essential security building-blocks for today’s organizations to ensure safe and trusted communication of information. In this paper, we study the feasibility of off-line deep learning based NIDSes by constructing the detection engine with multiple advanced deep learning models and conducting a quantitative and comparative evaluation of those models. We first introduce the general deep learning methodology and its potential implication on the network intrusion detection problem. We then review multiple machine learning solutions to two network intrusion detection tasks (NSL-KDD and UNSW-NB15 datasets). We develop a TensorFlow-based deep learning library, called NetLearner, and implement a handful of cutting-edge deep learning models for NIDS. Finally, we conduct a quantitative and comparative performance evaluation of those models using NetLearner.

I. INTRODUCTION

As networking technology gets deeply integrated into our lives, protecting modern networked systems against cyber-attacks is no longer optional. Network intrusion detection systems (NIDS) are essential security solutions for today’s networked systems supporting military applications, social communications, cloud services, and other critical infrastructures. A NIDS automatically monitors traffic in a network to detect malicious activities and policy violations. The majority of NIDSes today adopt signature-based detection techniques, which can only identify known attacks via matching pre-installed signatures to observed network activities. The signature databases have to be frequently updated to include new types of attacks. Those limitations have motivated researchers to investigate anomaly detection based approaches [1]–[6].

Anomaly detection approaches use data mining or machine learning techniques to mathematically model the trustworthy network activities based on a set of training data, and detect deviations using the model with the observed data. A key advantage is the ability to detect unknown or novel malicious activities. An on-line model further enables the efficient identification of new patterns or even new types of the abnormal behaviors in a dynamic network environment. However, if the constructed model is not sufficiently generalized for distinguishing normal and abnormal traffic, anomaly-based approaches would suffer from high false positive, i.e., incorrectly treat unknown normal traffic as malicious.

Deep learning has gained a dramatic increase in popularity in the last couple of years, and has offered advanced solutions

in the areas of image and speech recognition [7], [8], natural language processing [9], Go playing [10], and many other domains [11]. This motivates us to study the feasibility to enhance the anomaly detection based NIDS with the state-of-art deep neural networks trained by innovative algorithms. We also study the unsupervised generative deep learning models for intrusion detection, because those models can extract useful and hierarchical features from the vast amount of unlabeled traffics.

In this paper, we introduce a bundle of deep learning models for the network intrusion detection task, including multilayer perceptron, restricted Boltzmann machine, sparse autoencoder, and wide & deep learning. We also develop and open-source our TensorFlow-based testing and evaluation platform, NetLearner [12], to the research community. NetLearner includes the implementations of the studied deep learning models as well as the training procedures, which facilitates the reproduction and further extension of this work. Finally, we conduct a quantitative and comparative study of those deep learning models using two network intrusion detection datasets (i.e., NSL-KDD [13] and UNSW-NB15 [14]) and measure the accuracy, precision, and recall. Our experimental results show that for the NSL-KDD task, sparse autoencoder achieves an equivalently good performance to the existing machine learning solutions; and for the UNSW-NB15 task, the deep neural network models with greater generalization capability achieve better accuracy than support vector machine (SVM) models.

II. DEEP LEARNING BACKGROUND

We introduce two main reasons behind the success of deep learning to the rise of artificial intelligence as well as the implications of deep learning for network intrusion detection.

In the supervised learning framework, given the feature representations and inference models, learning is an optimization process that minimizes a predefined loss function over the training examples. The most commonly-used optimization algorithm is back-propagation (BP) [15] with gradient descent, because computing gradient is Hessian-free and memorization saves a significant amount of computation when propagating backward level by level. However, it is challenging to train deep neural networks with optimal weights only using BP. The first problem is that the cost function is usually non-convex,

and optimizing algorithms only with the first-order gradient are likely to be stuck at a poor local minimum. Secondly, exploding and vanishing gradient makes back-propagation difficult to train models with many layers stacked together, such as deep recurrent neural networks. Even if we can tolerate the long training time and carefully deal with the gradient exploding and vanishing, the trained model is often over-fitted to the training dataset and thus fails to generalize to the testing or future datasets.

The emergence of many novel learning algorithms and training techniques enables us to train deep neural networks that achieve good suboptimal minimums. For example, stochastic gradient descent (SGD) with mini-batches can significantly increase the training speed comparing to normal gradient descent on the entire dataset. In each step of gradient descent, researchers have shown that momentum can prevent SGD from “oscillating across but pushing along the shallow ravine [16].” Along with decaying the learning rate, the momentum-based optimization algorithms (e.g., Adam [17]) often find better local minimums. To prevent the over-fitting problem, researchers have proposed dropout [18] to average over an exponential number of neural networks. These general learning algorithms and training techniques would directly help neural networks to achieve better performance in the network intrusion detection.

Another breakthrough in the deep learning area is that researchers have successfully trained a number of useful unsupervised generative models. Different from supervised models or discriminative models that aim to discover the relationship between input variables and target labels (or the conditional probability distribution of the targets given the inputs), these models aim to learn the joint probability distribution or the joint conditional distribution of all variables for one phenomenon from the given dataset. The resulting generative model is powerful in many ways. First, given the well-trained probability distribution, the model can synthesize meaningful data comparable to real examples in the training set. For example, Auxiliary-Classifer Generative Adversarial Nets (AC-GAN) [19] can generate high-quality images after training on ImageNet dataset [20]; both AC-GAN and deep brief nets [21] can synthesize handwritten digits after learning from the MNIST dataset. Second, the ability to generate high quality faked data indicates that the model has learned better feature representations from the unlabeled data. For example, the features extracted from the hidden units of sparse autoencoder can significantly improve the performance of support vector classifier [22]. Additionally, researchers have shown that it is an excellent strategy to initialize deep neural networks with the weights from a successfully trained generative model [16], [21].

In the area of network intrusion detection, network traffic data is massive and dynamic, which makes it hard for security analysts to find malicious patterns and label anomalies. This situation makes the unsupervised generative model a promising solution to traffic classification. First, it utilizes the vast amount of unlabeled data to learning useful and hierarchical features. Second, it effectively initializes the weights of the

hidden layers in a deep neural network, which allows further fine-tuning towards a high-performance classifier. Among the various deep learning models we investigate in this paper, there are two types of generative models, i.e., restricted Boltzmann machine and autoencoders.

III. OFF-LINE NETWORK INTRUSION DETECTION TASKS

Unlike image classification or natural language processing, the labeled datasets in network intrusion detection tasks lack a common feature space [13], [14], [23], [24]. For example, between the two datasets NSL-KDD [13] and UNSW-NB15 [14], we only identify five common features, i.e., `src_bytes`, `dst_bytes`, `service`, `flag`, and `duration`. As a result, we have to consider each dataset as a standalone network intrusion detection task. We choose NSL-KDD and UNSW-NB15 datasets as our targeted tasks.

A. NSL-KDD Dataset

The NSL-KDD dataset originates from the KDDCup 99 dataset [23], but addresses two issues of the KDDCup 99 dataset. First, it eliminates the redundant records in the KDDCup 99, i.e., 78% of the training set and 75% of the testing set. Second, it samples the dataset so that the number of records belonging to one difficulty level is inversely proportional to its difficulty. The changes make the NSL-KDD dataset suitable for evaluating intrusion detection systems. The training dataset consists of 125,973 TCP connection records, while the testing dataset consists of 22,544 records. A record is defined by 41 features, including 9 basic features of individual TCP connections, 13 content features within a connection, 9 temporal features computed within a two-second time window, and 10 other features. Connections in the training dataset are labeled as normal or one of the 24 attack types. There are additional 14 types of attacks in the testing dataset, intentionally designed to test the classifier’s ability to handle unknown attacks. A classifier identifies whether a connection is normal or belongs to one of the four categories of attacks, namely denial-of-service (DoS), remote-to-local (R2L), user-to-root (U2R), and probing (also known as the 5-class classification problem).

B. UNSW-NB15 Dataset

Similar to the KDDCup 99 dataset, the UNSW-NB15 dataset is generated by simulating normal and attack behaviors in a hardware testbed. The simulation is conducted in the Cyber Range Lab of the Australian Centre for Cyber Security (ACCS) and 49 features in the dataset are extracted by a chain of software tools developed by ACCS. The structure of the features is similar to that of KDDCup 99 including 5 flow features, 13 basic features, 8 content features, 9 time features, and 12 other features. However, there are only five common features between the UNSW-NB15 and NSL-KDD datasets. The dataset has 257,673 flow records, among which 175,341 are used for training set and the rest are for testing. There are nine types of attacks in the dataset. The only type of attack in common between UNSW-NB15 and NSL-KDD is DoS. The new attacks in UNSW-NB15 are analysis, backdoor, exploits,

fuzzers, generic, reconnaissance, shellcode, and worms. In this paper, we consider the 2-class classification problem for the UNSW-NB15 dataset. The task is to predict whether a given flow record is normal or malicious.

IV. RELATED WORK

Researchers have modeled the intrusion detection process as an unsupervised anomaly detection problem including Mahalanobis-distance based outlier detection [25], density-based outlier detection [2], and evidence accumulation for ranking outlier [3]. A key advantage of those unsupervised approaches is to address the unavailability of labeled network traffic data. Researchers also investigate methods to obtain useful attacking data and convert them into labeled data [13], [14], [23], which enable us to apply supervised machine learning algorithms to the intrusion detection problem. Examples include decision trees [26], linear and non-linear support vector machine [27], and NB-Tree [4]. To the best of our knowledge, two works achieved the best prediction accuracy on the two datasets mentioned above. For the UNSW-NB15 dataset, Ramp-KSVCR [5] can achieve the accuracy of 93.52% by extending K-support vector classification-regression with ramp loss. The creators of UNSW-NB15 dataset [14] proposed the Geometric Area Analysis techniques using trapezoidal area estimation [6], which achieved the best-known accuracy on the NSL-KDD dataset (99.7%) and a slightly worse accuracy on the UNSW-NB15 dataset (92.8%).

There are a limited number of pioneer work to explore deep learning based network intrusion detection systems. For example, [1] adopts the sparse autoencoder and the self-taught learning scheme [22] to handle the problem of insufficient labeled data for training supervised models. Similar semi-supervised approaches have also been applied to discriminative restricted Boltzmann machine [28].

V. DEEP LEARNING MODELS FOR NETWORK INTRUSION DETECTION

We explore a bunch of deep learning models that are promising to the network intrusion detection problem, including multilayer perceptron, restricted Boltzmann machine, sparse autoencoder, and wide and deep learning with embeddings. We develop a Python library, NetLearner [12], which allows us to build and train those deep learning models using TensorFlow [29].

A. Multilayer Perceptron

Multilayer perceptron (MLP) is a fully connected feed-forward neural network with multiple hidden layers. Each layer consists of non-linear neural units. By introducing non-linear neural units (perceptrons), it can distinguish data that are not linearly separable. However, the non-linearity make it challenging to train a deep MLP of more than three layers even with the back-propagation learning algorithm [15]. MLP-based models get revived recently because of various training techniques designed by the deep learning community, including Stochastic Gradient Descent (SGD), Adam optimizer [17],

batch normalization [30], and Dropout [18]. Besides the number of neurons in each layer and the number of layers, MLP can also be tuned with different activation functions or neural types. In this paper, we use the logistic function and rectifier linear unit.

We build two dual-hidden-layer MLPs with 800 neurons in the first layer and 480 neurons in the second layer followed by a softmax classifier. One MLP is for the NSL-KDD task, and the other is for the UNSW-NB15 task. Both MLPs are trained with Adam optimizer [17] for 160 epochs with a batch size of 80. During the training, the learning rate decays from 0.1 exponentially with the base of 0.96. We do not include regularization in the model, but we apply dropout of probability 0.2 to prevent over-fitting.

B. Restricted Boltzmann Machine

Restricted Boltzmann machine (RBM) [21] is a type of energy-based models, which associate scalar energy to each configuration vector of the variables in the network. In an energy-based model, learning is the process of configuring the network weights to minimize the average energy over the training data. Both Boltzmann machine and RBM consist of a layer of hidden units connected to a layer of visible units. The term “restricted” means that the connections are between the hidden and visible layers, but not within the hidden or visible layers. Thus, RBM has faster training speed than Boltzmann machine, and it is feasible to stack multiple separately-trained RBMs to form a deeper neural network. However, the classical way to train RBM is computational infeasible because it relies on a large number of iterations of alternating Gibbs sampling. [21] proposed contrastive divergence (CD- x) as a faster learning procedure. They discover that instead of doing the alternating Gibbs sampling for many iterations, applying x (a small number between 1 and 4, say) steps of the alternating Gibbs sampling can quickly obtain a set of good weights for both layers.

For each task, we build an RBM with 800-hidden units to perform unsupervised learning on the training dataset. We train the RBM using CD-1 with a batch size of 10 for 160 epochs. The learning rate is initialized to 0.01 and it decays by $10e-6$ over each gradient update. We create a separate MLP with the same configuration described in Section V-A, and initialize its weights in the first hidden layer (800 neurons equivalently) to those in the RBM model with the goal of improving the quality of MLP. We then fine-tune the MLP for 160 epochs with a small learning rate of 0.01.

C. Autoencoders

An autoencoder is an unsupervised neural network with one hidden layer that sets the output layer to be equal to the input. However, to prevent the network from learning the meaningless identity function, we have to place extra constraints on the network, which generates different flavors of autoencoders. The sparse autoencoder works by placing a sparsity constraint on the activities of the hidden neurons [22].

We build a sparse autoencoder and our implementation is different from the self-taught learning approaches, which also adopts the sparse autoencoder as the unsupervised feature learner [1], [22]. In those work, the hidden features learned by the sparse autoencoders are used directly by a classifier (e.g., a softmax regressor or an SVM). The functionality of autoencoder resembles a transformation of a raw dataset (e.g., using principal component analysis) with the goal of obtaining a new feature space beneficial to general supervised learning algorithms.

We initialize the first layer weights of an MLP in the same way of using an RBM. The size of the over-complete hidden layer in the sparse autoencoder is 800; the sparsity value ρ is 0.05. The autoencoder is trained with ADADELTA [31] for 160 epochs with a batch size of 80. We then create a separate MLP with the same configuration described in Section V-A, and initialize its first layer weights with the learned weights of the autoencoder. We use the SGD optimizer with a tiny initial learning rate 0.004 and decaying 1e-6 over each update to fine-tune the MLP model.

D. Wide and Deep Learning with Embeddings

The categorical and integer features are extremely sparse in the network intrusion datasets. For example, a categorical feature “proto” indicates one of the 133 protocol types that a traffic record belongs to. Therefore, one-hot encoding will generate a 133-dimension vector consisting only one field with the value one. Neural networks are often not good at utilizing sparse large dimension inputs. We tackle this problem in two ways concerning the combined model in [32]. The first solution is to embed the integer or categorical features. An embedding is a mapping from sparse discrete objects to a dense vector of real numbers. “word2vec” is widely used in the natural language processing and machine translation tasks, where embeddings are treated as points in the vector space so that the similarity between objects can be visually measured by the Euclidean distance or angle between the vectors. In this case, embedding provides a solution to converting large-vocabulary-size categorical features and sparse integer features to dense vectors of continuous values. Deep neural network fed with embedding inputs can generalize better even with less feature engineering. As stated in [32], these input features to the deep neural nets are denoted as deep components consisting of continuous and embedded features. Second, we leverage some simple linear models with nonlinear feature transformations to address the sparse inputs. The method is called wide components [32], which consists of the basis and crossed features. The basis features are the raw input features that are either integer or categorical. The crossed features are the cross-product transformations of basis features that memorize the interactions between raw features. The Wide and Deep model complements a deep neural network with embedded low-dimension input vectors for good generalization. Its linear sub-model is integrated with the deep neural network using a weighted sum of each model’s output for good memorization.

The wide and deep learning model requires engineering the raw attributes in datasets into the basis, crossed, continuous, and embedded components. In our implementation, the basis features are all the raw symbolic and integer attributes. Crossed features are built by a subset of combinations of the symbolic attributes in a dataset. The raw symbolic and integer attributes are fed to the deep neural network as embedded components after conducting embedding. The continuous components are the raw continuous attributes. To compare with other models, we set the structure of the deep neural network in the wide and deep model to be the same sizes as the baseline MLP, i.e., two hidden layers with the size of [800 480].

VI. EXPERIMENTAL EVALUATION

We evaluate all the models developed in NetLearner on the 5-class NSL-KDD task and the 2-class UNSW-NB15 task using the following metrics.

- **Accuracy** is the percentage of correctly classified connections over the total number of connections in the dataset:

$$A = \frac{\text{True Positives} + \text{True Negatives}}{\text{Number of Instances}} \quad (1)$$

Accuracy is not suitable for evaluating imbalance datasets where the number of records of one class is extremely larger than the number of records of another class. In the NSL-KDD dataset, the number of available U2R records (i.e., 67) is two orders of magnitude less than the number of records in other classes (i.e., 9711, 7458, 2887 and 2121). Therefore, we also consider the precision and recall.

- **Precision** is the percentage of the correctly classified positives over the total number of positives predicted by the classifier.

$$P = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \quad (2)$$

- **Recall** is the percentage of the correctly classified positives over the total number of relevant elements.

$$R = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \quad (3)$$

We train a radial basis function kernel support vector machine (SVM) and report its accuracy with the multilayer perceptron (MLP) model, the restricted Boltzmann machine fine-tuned neural network (RBM), the sparse autoencoder fine-tuned neural network (SAE), and the wide linear classifier and deep neural network combined model (WnD).

It is critical to search the optimal hyper-parameters that fit the problem domain and model before training machine learning models. We first manually set all the hyper-parameters, including the number of layers, number of neurons in each layer, learning rate, and batch size, to be identical across all the models. We then use 5-fold cross-validation on the training datasets to determine the optimal training time T for each model. Finally, we train the model for T epochs and report the metrics on the testing dataset, which is not touched during the training phase. Determining the training time (model

complexity) by cross-validation with fixed common hyper-parameters ensures that the deep learning models are neither overfitting nor underfitting. We plot the 5-fold cross validation loss of MLP in Figure 1. We can see that the validation loss converges to approximately 0.02 after 140 epochs, even though the training loss still decreases, and the optimal training time that validation loss is minimized is at $T \approx 150$. In this case, we train the MLP on NSL-KDD train dataset for exactly 150 epochs and report its metrics on NSL-KDD test dataset.

The accuracies of all considered classifier are shown in Figure 2 for the NSL-KDD task and Figure 3 for the UNSW-NB15 task. In the 2-class UNSW-NB15 dataset, the volumes of normal and attacking traffic are nearly balanced (i.e., 37,000 normal v.s. 45,332 attacking records). Therefore, we only report the precision and recall for the attacking traffics in Figure 3. For the 5-class NSL-KDD task, we adopt the approach in [1] to calculate the weighted precisions and recalls, and plot them together with accuracy in Figure 2. The per-class precisions and recalls are also listed in Table I.

For the NSL-KDD task, all classifiers achieve high training accuracy (no less than 99%). However, all classifiers show a gap between training accuracy and testing accuracy (as low as 78.4%). As the representative of the classic machine learning approach, SVM achieves a 78.5% accuracy comparable to the deep learning models. Note that our SAE model achieves the same accuracy performance to [1], which is the best among all the considered models (79.2%). RBM, SAE, and WnD all outperform MLP for two different reasons. RBM and SAE provide their underlying MLP with better initial weights in the first layer than randomly generated numbers. WnD has a slightly higher accuracy because of the extra linear model. Table I shows two remarkable facts. SVM performs better at Probe attacks (93% precision and 82% recall) than all the neural networks ($\approx 85\%$ precision and $\leq 70\%$ recall). However, it suffers hugely in case of the U2R attacks ($\approx 6\%$ precision and recall). On the other hand, the neural networks (MLP, RBM, SAE and WnD) miss many U2R attacks ($\leq 6\%$ recall), but they have much higher reliability in identifying these attacking traffic ($\approx 60\%$ precision).

For the UNSW-NB15 task, the average accuracies of RBM, SAE, and WnD are all higher than MLP for the same reason mentioned in the NSL-KDD task. We notice WnD has significantly improved MLP's performance by around 5%. Different from the NSL-KDD task, the training accuracies of all the approaches are mediocre (up to 94.4%) in contrast to the NSL-KDD case where the training accuracy of every model is more than 99%. The harder UNSW-NB15 training dataset is one primary reason that the testing accuracies of the UNSW-NB15 task are higher than that of the NSL-KDD task, since classifiers only have access to the training dataset. Therefore, even though SVM has an equivalent training accuracy to the neural networks (93% v.s. 94%), its testing accuracy falls far behind by 5% (comparing to MLP) to 9% (comparing to WnD), showing the superior generalization capability of the deep neural network models. The detection alarms are mostly correct for every classifier, due to the high recall values \geq

97%, and the deep learning models have better precision ($\geq 81\%$) than SVM (75%).

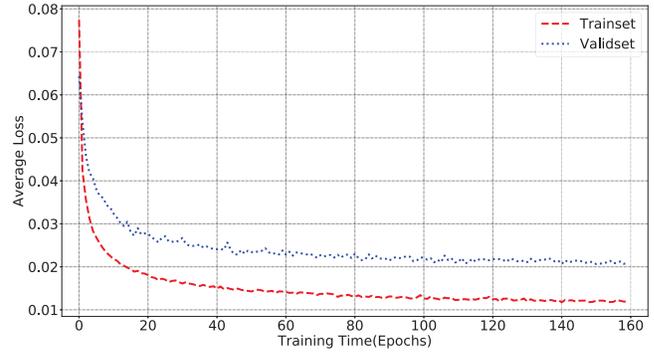


Fig. 1: History of MLP's Average Loss During Cross Validation

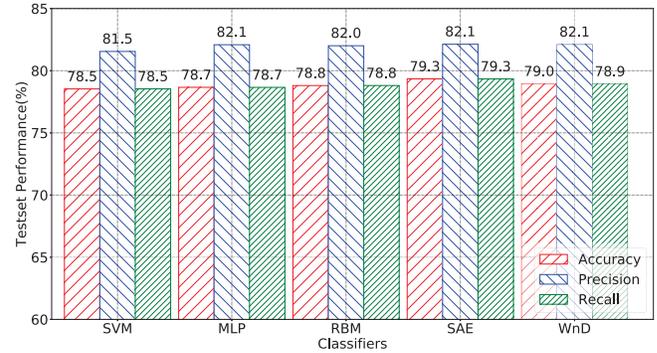


Fig. 2: Metrics Comparison, NSL-KDD Task

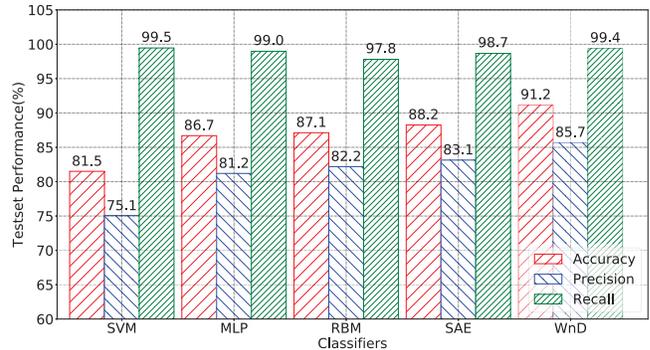


Fig. 3: Metrics Comparison, UNSW-NB15 Task

VII. CONCLUSION

We study cutting-edge deep learning models to the design of network intrusion detection systems. With the help of our open-source Tensorflow-based deep learning library NetLearner, we perform a comparative evaluation of those models on two network intrusion detection tasks using the NSL-KDD and UNSW-NB15 datasets. Preliminary experimental results show that for the NSL-KDD task, sparse

TABLE I: Per-class Precision/Recall on the NSL-KDD Task

		Traffic Class				
		Normal	DoS	Probe	U2R	R2L
SVM	Precision	72.82	75.27	93.85	6.32	96.65
	Recall	96.18	72.16	82.16	6.06	13.34
MLP	Precision	69.18	95.49	85.58	59.52	92.01
	Recall	96.66	82.31	69.28	6.31	15.03
RBM	Precision	69.41	95.41	85.23	41.51	93.86
	Recall	96.81	83.89	64.74	5.56	15.48
SAE	Precision	70.20	95.63	84.70	65.00	87.76
	Recall	96.92	83.34	70.56	3.28	16.29
WnD	Precision	70.08	95.59	84.02	60.34	91.57
	Recall	96.88	83.64	67.84	4.53	15.34

autoencoder achieves accuracy similar to the existing machine learning solutions; for the UNSW-NB15 dataset, deep neural network models with greater generalization capability deliver better accuracy than SVM based solutions.

ACKNOWLEDGMENT

The authors are grateful to the support by the Air Force Office of Scientific Research under Grant YIP FA9550-17-1-0240, the Maryland Procurement Office under Contract No. H98230-18-D-0007, and a cooperative agreement between IIT and National Security Research Institute (NSRI) of Korea. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of AFOSR, NSRI, and the Maryland Procurement Office.

REFERENCES

- [1] A. Javaid, Q. Niyaz, W. Sun, and M. Alam, "A Deep Learning Approach for Network Intrusion Detection System," in *Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies*, New York, NY, USA, vol. 35, 2015, pp. 2126–2132.
- [2] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "LOF: Identifying Density-based Local Outliers," in *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '00. New York, NY, USA: ACM, 2000, pp. 93–104.
- [3] P. Casas, J. Mazel, and P. Owezarski, "Unsupervised Network Intrusion Detection Systems: Detecting the Unknown without Knowledge," *Computer Communications*, vol. 35, no. 7, pp. 772 – 783, 2012.
- [4] R. Kohavi, "Scaling Up the Accuracy of Naive-Bayes Classifiers: A Decision-tree Hybrid," in *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, ser. KDD'96. AAAI Press, 1996, pp. 202–207.
- [5] S. M. Hosseini Bamakan, H. Wang, and Y. Shi, "Ramp Loss K-Support Vector Classification-Regression; a Robust and Sparse Multi-class Approach to the Intrusion Detection Problem," *Know.-Based Syst.*, vol. 126, no. C, pp. 113–126, Jun. 2017.
- [6] N. Moustafa, J. Slay, and G. Creech, "Novel Geometric Area Analysis Technique for Anomaly Detection using Trapezoidal Area Estimation on Large-Scale Networks," *IEEE Transactions on Big Data*, vol. PP, no. 99, pp. 1–1, June 2017.
- [7] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," *Commun. ACM*, vol. 60, no. 6, pp. 84–90, May 2017.
- [8] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath *et al.*, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, 2012.
- [9] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in Neural Information Processing Systems*, 2013, pp. 3111–3119.
- [10] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot *et al.*, "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.
- [11] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.
- [12] "NetLearner," <https://github.com/littlepretty/NetLearner>, accessed: 2018-3-15.
- [13] M. Tavallaei, E. Bagheri, W. Lu, and A. A. Ghorbani, "A Detailed Analysis of the KDD CUP 99 Data Set," in *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, July 2009, pp. 1–6.
- [14] N. Moustafa and J. Slay, "The Evaluation of Network Anomaly Detection Systems: Statistical Analysis of the UNSW-NB15 Data Set and the Comparison with the KDD99 Data Set," *Inf. Sec. J.: A Global Perspective*, vol. 25, no. 1-3, pp. 18–31, Apr. 2016.
- [15] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning Representations by Back-propagating Errors," in *Neurocomputing: Foundations of Research*. Cambridge, MA, USA: MIT Press, 1988, pp. 696–699.
- [16] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the Importance of Initialization and Momentum in Deep Learning," in *Proceedings of the 30th International Conference on International Conference on Machine Learning*, ser. ICML'13. JMLR.org, 2013, pp. III–1139–III–1147.
- [17] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *ArXiv e-prints*, Dec. 2014.
- [18] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, Jan. 2014.
- [19] A. Odena, C. Olah, and J. Shlens, "Conditional Image Synthesis With Auxiliary Classifier GANs," *ArXiv e-prints*, Oct. 2016.
- [20] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.
- [21] G. E. Hinton and R. R. Salakhutdinov, "Reducing the Dimensionality of Data with Neural Networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.
- [22] R. Raina, A. Battle, H. Lee, B. Packer, and A. Y. Ng, "Self-taught Learning: Transfer Learning from Unlabeled Data," in *Proceedings of the 24th International Conference on Machine Learning*, ser. ICML '07. New York, NY, USA: ACM, 2007, pp. 759–766.
- [23] "KDD Cup 1999 Data," <http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html>, accessed: 2017-3-10.
- [24] R. Lippmann, J. W. Haines, D. J. Fried, J. Korba, and K. Das, "The 1999 DARPA Off-line Intrusion Detection Evaluation," *Computer Networks*, vol. 34, no. 4, pp. 579–595, Oct. 2000.
- [25] A. Lazarevic, L. Ertoz, V. Kumar, A. Ozgur, and J. Srivastava, *A Comparative Study of Anomaly Detection Schemes in Network Intrusion Detection*, pp. 25–36.
- [26] J. R. Quinlan, *Learning Efficient Classification Procedures and Their Application to Chess End Games*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1983, pp. 463–482.
- [27] C. Cortes and V. Vapnik, "Support-vector Networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, Sep 1995.
- [28] U. Fiore, F. Palmieri, A. Castiglione, and A. D. Santis, "Network Anomaly Detection with the Restricted Boltzmann Machine," *Neurocomputing*, vol. 122, pp. 13–23, 2013.
- [29] "TensorFlow: An open-source software library for Machine Intelligence," <https://www.tensorflow.org/>, accessed: 2018-3-15.
- [30] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," *CoRR*, vol. abs/1502.03167, 2015. [Online]. Available: <http://arxiv.org/abs/1502.03167>
- [31] M. D. Zeiler, "ADADELTA: an adaptive learning rate method," *CoRR*, vol. abs/1212.5701, 2012. [Online]. Available: <http://arxiv.org/abs/1212.5701>
- [32] H.-T. Cheng, L. Koc, J. Harmsen, T. Shaked, T. Chandra, H. Aradhye, G. Anderson, G. Corrado, W. Chai, M. Ispir, R. Anil, Z. Haque, L. Hong, V. Jain, X. Liu, and H. Shah, "Wide and Deep Learning for Recommender Systems," *ArXiv e-prints*, Jun. 2016.