Journal of Network and Information Security
12 (2) 2024, 45-52
http://www.publishingindia.com/jnis

An Improved Deep Sparse Autoencoder Driven Network Intrusion Detection System (IDSAE-NIDS)

Jesse Ismaila Mazadu[1*] and Abayomi Jegede[2]
[1]Computer Science Department, Faculty of Computing and Information Systems, Federal University Wukari, Nigeria. Email: jesse@fuwukari.edu.ng
[2]Computer Science Department, Faculty of Natural Science, University of Jos, Nigeria. Email: jegedea@unijos.edu.ng
[*]Corresponding Author

Abstract: Computer network users experience persistent attacks due to vulnerabilities in the systems and network. An intrusion detection system (IDS) is a system that checks the flow of data in the network and alerts or detects abnormal traffic. Even though there is no perfect system anywhere on earth, there may be a cause for breakdown/downtime now and then. More so, the intrusion detection system may run into errors in a bid to detect malware in incoming traffic on a network. Hence, the need to develop a system that can detect intrusion in the network and do that with minimal error. This proposed paper is an improved deep sparse autoencoder-driven network intrusion detection system (IDSAE NIDS) that addresses the issue of interpretability of the L2 regularization technique employed in other works. The proposed IDSAE NIDS model was trained using a mini-batch gradient descent algorithm, L1 regularization technique, and ReLU activation function to achieve a better model performance. Experimental results based on the NSL-KDD dataset show that our approach provides significant performance and improvements over other deep sparse autoencoder NIDSs.

Keywords: Activation function, Autoencoder, Regularization, SoftMax and sparse autoencoder.

I. Introduction

An intrusion detection system (IDS) can be defined as a system that monitors network traffic for suspicious activities and issues alert when such activities are discovered. It is a tool that scans a network or system for harmful activities or breach of security policy and any malicious activities or violation is normally reported either to an administrator or collected centrally using a security information and event management system (SIEM). Although intrusion detection systems monitor network for potentially malicious activities, they are also susceptible to false alarms. Hence, organizations need to fine-tune their IDS product after installing them. This enables the intrusion detection systems to recognize normal traffic on the network and distinguish it from malicious activities [1].

Common types of IDSs are anomaly-based and signature-based. An anomaly-based IDS, also called behavioral-based intrusion detection system keeps track of activities within a specific scope, looking for instances of malicious behavior as they can define it. This is difficult because it can lead to false positives. For instance, outbound Uniform Resource Locator (URLs) of a web activity might be considered malicious and sites involving certain domains or URL length/contents might automatically be blocked

even though it is a human being trying to access them for legitimate purpose [2].

On the other hand, signature-based IDS also known as knowledge based, involves looking for specific signature byte combination that implies an attack. These solutions generate fewer false positives than anomaly solutions because the search criteria are so specific but only cover signatures that are already in the search database. This means novel attacks can bypass the IDS and penetrate the network. This technique, cannot detect novel attacks because they are based on regular expressions and strings. Other categories of IDSs are the host-based, application-based, and the network-based IDSs [1].

II. Related Work

A hybrid deep learning technique for intrusion detection [3], used an autoencoder in order to reduce the dimensionality of data and extract the main features of data. A Deep Belief Network (DBN) was used to train the intrusion detection systems based on KDDCUP'99 dataset. DBN is composed of multi-layer Restricted Boltzmann Machines (RBM) where each RBM consists of the visible units and hidden units. Similarly, RBM was used to detect anomalies by training a model with real workload traces from 42 hours workstation traffic and testing the accuracy with KDDCUP'99 dataset [4]. The model achieved about 85% accuracy on the total 10% KDDCUP'99 dataset. A related work, used an unsupervised greedy learning algorithm to learn a similarity representation over high-dimensional data [5]. An evaluation of the model on the KDDCUP'99 dataset shows that four-hidden-layer RBM produce higher accuracy in comparison with support vector machine and artificial network neural. On the other hand, long short-term memory (LSTM) architecture of a recurrent neural network (RNN) was used to train the IDS model using KDDCUP'99 dataset [6]. This model achieved higher accuracy and detection rate in comparison with other classifiers such as kernel network neural and support vector machine. Accelerated Deep neural network architecture was used to identify the abnormalities in the network traffic data [7]. The acceleration of the training process using the multicore CPUs was faster than the serial training mechanism. An artificial intelligence (AI) intrusion detection system using a deep neural network (DNN) was investigated and tested with the KDDCUP'99 dataset in response to ever-evolving network attacks [8]. This approach was based on the use of machine learning techniques to implement semi-supervised anomaly detection systems where the classifier is trained with "normal" traffic data only, so that knowledge about anomalous behavior can evolve and be constructed in a dynamic way. Authors in [6] explored the effectiveness of a detection approach based on machine learning, using the discriminative restricted Boltzmann machine [6] to combine the expressive power of generative models with good classification accuracy in order to infer part of its knowledge from a complete training data.

A novel deep learning technique for intrusion detection [9], used nonsymmetric deep autoencoder (NDAE) for unsupervised feature learning. Similarly, self-organization map (SOM) artificial neural network [10], was implemented as part of an intrusion detection system, to detect anomalies on KDDCUP'99 and NSL-KDD datasets of internet traffic activity simulation. The results obtained were compared and analyzed based on several performance metrics. The detection rate for KDDCUP'99 dataset is 92.37%, while detection rate for NSL-KDD dataset is 75.49%. In a bid to improve on the previous approach, a deep learning technique for intrusion detection was proposed in [11]. The model was built using deep autoencoder and trained in a greedy layer wise fashion in order to avoid overfitting. The performance of the model was evaluated on KDDCUP'99 (old version of NSL-KDD) using both binary and multiclass classification. Although the approach provides high accuracy for intrusion detection task, it is computationally expensive.

III. Methodology

This section contains the explanation of methods and techniques employed in developing the researchers' improved deep sparse autoencoder driven network intrusion detection system (IDSAE-NIDS). Fig. 1 is the research process flow for implementing the

proposed improved deep sparse autoencoder driven intrusion detection system (IDSAE-NIDS).

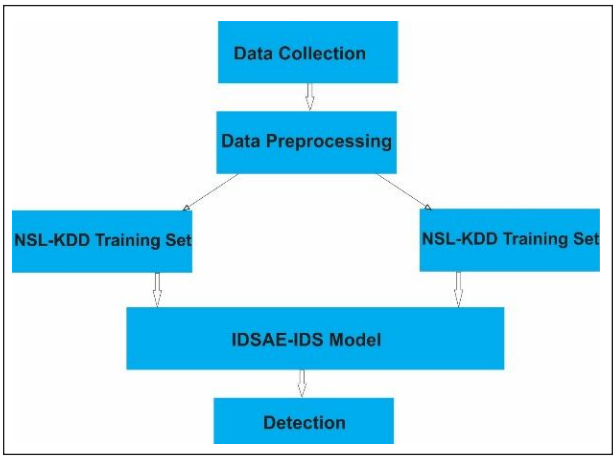

Fig. 1: Process-Flow Diagram

Data Collection

The proposed deep sparse autoencoder IDS was implemented using python programming language. The NSL-KDD dataset is the improve edition of the KDDCUP'99 [12]. NSL-KDD was proposed, which consists of selected records of the complete KDDCUP'99 dataset. The advantages of NSL-KDD dataset are:

- It has no redundant records in the train set, so the classifier will not produce any biased result
- No duplicate record in the test set which have better reduction rates.
- The number of selected records from each difficult level group is inversely proportional to the percentage of records in the original KDDCUP'99 dataset.

TABLE I: MAPPING OF ATTACK CLASS WITH ATTACK TYPE

1	DoS	Back, Land, Neptune, Pod, Smurf, Teardrop, Apache2, Udpstorm, Processtable, Worm (10)
2	Probe	Satan, I psweep, Nmap, Portsweep, Mscan, Saint (6)
3	R2L	Guess_Password, Ftp_write, Imap, Phf, Multihop, Warezmaster, Warezclient, Spy, Xlock, Xsnoop, Snmpguess, Snmpgetattack, Httptunnel, Sendmail, Named (16)
4	U2R	Buffer_overflow, Loadmodule, Rootkit, Perl, Sqlattack, Xterm, Ps (7)

A. Preprocessing

This phase transforms the data before loading it to the model [13]. Data preprocessing is used to convert the raw data into a clean dataset. This involves checking for redundant rows in the dataset and removing them. It is worthy of note that NSL-KDD dataset does not come with column names. Hence, the need to add column names to each feature of the dataset.

Training

The researchers used mini-batch gradient descent as the training algorithm. Gradient descent is an optimization algorithm used to find the values of a function's parameters (Coefficients or the weight) that minimize a cost function as less as possible. A gradient simply measures the change in all weights with regards to change in error. One can also think of it as the slope of a function. The higher the gradient, the steeper the slope and the faster a model can learn. But if the slope is zero, the model stops learning. In mathematics one can think of it as a partial derivative with respect to its inputs [14].

$$W_{ij}^{(l)} := W_{ij}^{(l)} - \alpha \frac{\partial}{\partial W_{ij}^{(l)}} J(W,b) \quad (1)$$

$$b_i^{(l)} := b_i^{(l)} - \alpha \frac{\partial}{\partial b_i^{(l)}} J(W,b) \quad (2)$$

Gradient descent Equation

Where α is the learning rate which is 'lr' in the work, with the value 0.001, it is a measure of the rate at which the network optimizes the minimization of the loss function in a neural network. The key step is computing the partial derivatives above. The researchers use backpropagation to compute the partial derivative above. *J(W, b)* is the cost function while b is the bias term. By training the neural network, what the researchers were doing is essentially minimizing the loss function. The loss

function gives the measure of how far from the perfect is the performance of the neural network model on the dataset and the product of loss function is the loss value or error value [15]. Moreover, the trained model is also learning good values for all the weights and bias from the sample data.

B. Modeling IDSEA-NIDS

Sparse Autoencoder, proffers an alternative method for introducing an information bottleneck without requiring reduction in the number of nodes at our hidden layers. It relies on activating a small number of neurons which is another way of imposing regularization [16].

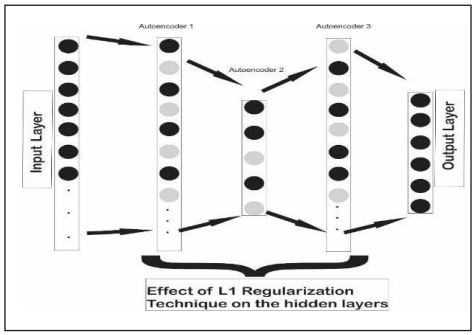

Fig. 2: IDSAE-NIDS Architecture

To apply sparsity to the model, the researchers used the L1 regularization technique which set weights to zero, and that will in turn lead to a lower loss value and more zeros weights produces a sparse model. They also used ReLu activation function that can actually allow the neurons to output a zero value that is, it penalizes the neurons as expected. The researchers have itemized the techniques employed in building this model as well as the strong reasons for using them and even how the used them below: Sparse Autoencoders are autoencoders whose training procedure involves sparsity constraints. In order to implement sparsity, the researchers used regularization techniques that imposes penalty to the neurons in our model. In the researchers' work – Improve deep sparse autoencoder driven network intrusion detection system, they used the L1 regularization technique, dropout and for desirable sparse representation, the Rectified Linear activation function (ReLu). L1 Regularization is one of the regularization techniques we chose for our model. Here, the researchers penalize the absolute value of the weights and as such the weight can be reduced to zero. For them to create a good model they must minimize the outcome of their loss function.

$$\text{Loss Function} = \text{Loss} + \lambda/2m * \sum||W|| \quad (3)$$

The loss function they used is the 'sparse categorical crossentropy' and lambda (λ) is the regularization parameter and its value is optimized for a better result. Whereas the m in the equation above represents the number of samples from the dataset and W is the weight of the input data [17]. Weight is the parameter that transfers the input data within the hidden layer of a neural network.

Let's say we have an equation:

$$y = Wx+b \quad (4)$$

Where x is the input, y the output, w is the weight, and b is the bias.

There are three (3) ways of applying regularizers as a neural network layer's parameter: Kernel regularizer, Bias regularizer and Activity regularizer [18]. Hence, in applying L1 regularization, the researchers will either apply it to the weight to reduce the layer's weights w with 'Kernel regularizer' or use 'Bias regularizer' which allows them to apply it on the bias to reduce the bias b. But because they aim at reducing the output y, they made use of 'Activity regularizer'. This reduced the weight and adjust the bias so that Wx+b is smallest. Since the researchers want the output of some nodes in a particular layer to be smaller or closer to 0 or even 0 that is for the node to be deactivated completely. The researchers employed L1 regularization which made the weight matrix of the layer's output y to return 0, in other words to impose sparsity in the neural network [19] there by helping to overcome model overfitting. Overfitting occurs when a model tries to predict a trend in data that is too noisy. This is caused due to an overly complex model with too many parameters. A model that is overfitted is inaccurate because the result does not reflect the reality present in the data.

Activation function is that function that is used to get the output of a neuron in a neural network and it is also called transfer function because it is responsible

for modifying and transferring the content (the input data) of the neuron from the current neuron to the next neuron [20].

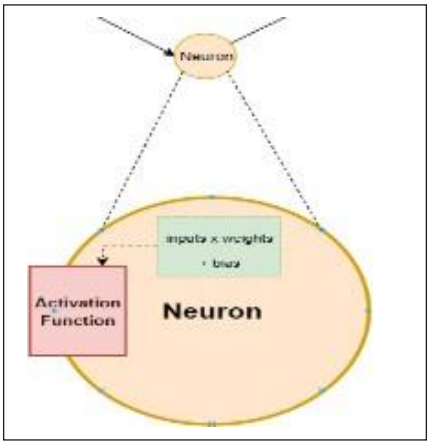

Fig. 3: A Typical Neuron and an Activation Function [21]

At first, the weighted sum of the inputs of the neuron were calculated.

$$Y = \sum(weight * input) + bias \quad (5)$$

For example, if the inputs are: $x_1, x_2,..x_m$

And the weights are: $w_1, w_2,\ldots w_m$

After which, the weighted sum (the weighted sum is the product of the inputs and the weights) is computed resulting to: $x_1 w_1 + x2\ w_2\ldots + x_m w_m$.

Subsequently, a bias is added to the weighted sum: $x_1 w_1 + x_2 w_2\ldots$ xmwm + bias

Lastly, the value computed is passed to the activation function, which then prepares an output.

Activation function $(x_1 w_1 + x_2 w_2\ldots x_m w_m + \text{bias})$ (6)

IV. Results and Discussion

The Fig. 4 below is the result of the last epoch (epoch 70) from our experiment. The study expects that as accuracy increases there should be a decrease in the loss (that is, accuracy is inversely proportional to loss). The relationship is captured in the equation:

$$\text{Accuracy } \alpha\ 1/\text{Loss} \quad (7)$$

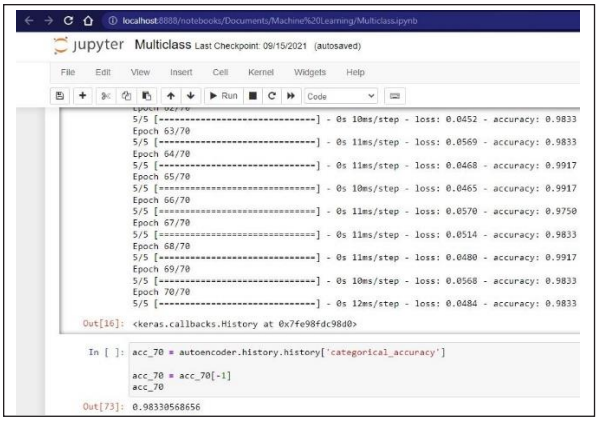

Fig. 4: Multiclass Classification Result

The model hyperparamters are:

Regularization Technique: L1 regularization

Output layer's Activation Function: ReLU

Epochs: 20, 30, 40, 50, 60 and 70.

Batch size: 64;

Drop out percentage: 10%

Numbers of hidden layers: 3

Table II: Number of Epochs vs Accuracy

Sr. No.	Number of Epochs	Accuracy	Loss
1.	20	58.23%	34.4202
2.	30	58.14%	0.2427
3.	40	60.00%	0.0890
4.	50	97.50%	0.0573
5.	60	97.50%	0.0672
6.	70	98.33%	0.0484

Since a single epoch can lead to underfitting and too many epochs may cause our model to overfit. From Table II above, we started with 20 epochs and this gave us 58.23% accuracy and we decided to increase the number of epochs to 30 and the accuracy turn out to be 58.14% which is a decrease in the accuracy. From the researchers result, the model performance is not bad because the loss (the model's error value) has reduced from 34.4202 to 0.2427 implying that their model performance is not bad. The loss value

is the penalty for a bad prediction, it is a number indicting how bad the model's prediction was at a given number of epochs. More so, at this point the model has not yet learnt sufficient features of data so as to have higher accuracy. When they increased the epochs to 40, the accuracy also increased to 60.00% and this shows that our model is having a better performance because higher accuracy corresponds to lower loss value of 0.0890. For epochs 50 the researchers had 97.50% as the accuracy and for epochs 60 they still had 97.50% as the accuracy, but their error value was 0.0573 and 0.0672 respectively.

Since the study is beginning to have an increase in the loss value, they had to affect early stopping. In training a neural network using an optimization algorithms like gradient descent, the model parameters that the weights are updated to reduce the training loss or error value. At the end of each forward propagation, the network weights are updated to reduce error in the next epochs. Too much training can result in network overfitting on the training data therefore, early stopping provides guidance as to how many epochs can run before the network begins to overfit. As the name suggests, the main idea in early stopping is to stop training when certain criteria are met. Usually, they stop training a model when generalization error (the loss value) starts to increase. So, we decided to add the last 70 epochs which has an accuracy of 98.33% with a better and lower loss value of 0.0484 and this bring us to a state where the network has learned to properly respond to the set of training patterns within some margin of error (0.0484) and we can also say that our model has converge at this point.

Graphical Representation of Result

Fig. 5: Accuracy vs Epochs

From Fig. 5 above the researchers can infer from the graph that y axis represents the accuracy from 0 to 1 (that is represented in percentage). The x axis is the epochs' axis that has numbers from 20 to 70 epochs in all. The curve above shows the expected increase in accuracy at the y axis as the epochs at the x axis increases. Very few epochs lead to underfitting but as the epochs increase additional number of times, the weights are adjusted in the neural network. Increasing the epochs is not always necessarily a bad idea since it can help make the researcher's model even more accurate but with an ever-increasing epoch, the researchers do run the risk of the neural network over-fitting the data [22]. The curve as seen in the Fig. 2 above moves down a bit then up after which it plateaus and at this point more and more epochs cannot cause any increase in the accuracy. Hence, at epochs 70 it converges, giving us an optimal model performance of 98.33%.

V. Conclusion and Future Work

In the experiment, the researchers had an accuracy of 98.33% which is higher than that of Systems [23] [24] because of the implementation of representational sparsity of ReLU, and the presence of L1 regularization technique in our experiment. The researchers have found out that the use of Rectified linear activation function (ReLU), L1 regularization technique and a suitable learning rate of 0.001, has increased the detection rate of our model more than previous works on Deep Sparse Autoencoders Network Intrusion Detection Systems [23] [24]. Since L2 regularization technique works well with data of high dimension, the researchers look forward to trying both L1 and L2 regularization techniques on a more recent network intrusion dataset of higher-dimensionality in our future work.

References

[1] G. Gross, "Intrusion detection techniques, methods and best practices: Detecting networking intrusion in 2019," *AT&A Cybersecurity-Security Essentials*, Feb. 2019. [Online]. Available: https://alienvault.com/

blogs/security-essentials/intrusion-detection-techquues-methods-best-practices

[2] V. Jyothsna, and V. V. Ranna Prasad "A "Review of anomaly based intrusion detection system," *International Journal of Computer Application*, vol. 28, no. 7, Sep. 2011.

[3] nibusinessinfo.co.uk, "Network security issues," Apr. 2021. [Online]. Available: https://www.nibusinessinfo.co.uk/content/network-security-issues

[4] K. A. Taher, B. M. Yasin Jisan, and Md. M. Rahman, "Network intrusion detection using supervised machine learning technique with feature selection," *International Conference on Robotics, Electrical and Signal Processing Techniques (ICREST)*, IEEE, 2019.

[5] W. Wang, "HAST-IDS: Learning hierarchical spatial-temporal features using deep neural networks to improve intrusion detection," *IEEE Access,* vol. 6, pp. 1792-1806, 2018.

[6] F. Fahimeh, and J. Heikkonen, "A deep auto-encoder based approach for intrusion detection system," *International Conference on Advanced Communications Technology (ICACT),* 2018.

[7] M. Al-Qatf, Y. Lasheng, Mohd. 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.

[8] B. Zhang, Y. Yu, and J. Li, "Network intrusion detection based on stacked sparse autoencoder and Binary Tree Ensemble method," *IEEE International Conference on Communications Workshops (ICC Workshops),* 2018, pp. 1-6, doi: https://doi.org/10.1109/ICCW.2018.8403759.

[9] C. Versloot, "What are L1, L2 and elastic net regularization in neural networks?," 2020. [Online]. Available: https://www.machinecurve.com/index.php/2020/01/21/what-are-l1-l2-and-elastic-net-regularization-in-neural-networks/

[10] K. Sadaf, and J. Sultana, "Intrusion detection based on autoencoder and IF in fog computing," *IEEE Access*, vol. 8, pp. 167059-167068, 2020, doi: https://doi.org/10.1109/ACCESS.2020.3022855.

[11] J. Jerry, "Variational autoencoders," 2020. [Online]. Available: https://www.jeremyjordan.me/variational-autoencoders/

[12] Edureka, Autoencoders Tutorial| Autoencoder in Deep Learning| Tensorflow Training| Edureka, 2019. [Online]. Available: https://youtube.be/nTt_ajul8NY

[13] S. Zhou, Dec. 2018. [Online]. Available: https://medium.com/@syoya/what-happens-in-sparse-autencoder-b9a5a69da5c6

[14] L. Yuancheng, M. Rong, and J. Ruhai, "A hybrid malicious code detection method based on deep learning," *Journal of Security and Its Applications*, vol. 9, no. 5, pp. 205-216, 2015.

[15] U. Fiore, F. Palmieri, A. Castiglione, and A. De Santis, "Network anomaly detection with the restricted Boltzmann machine," *Neurocomputing,* vol. 122, pp. 13-23, Dec. 2013, doi: http://dx.doi.org/10.1016/j.neucom.2012.11.050.

[16] N. Gao, L. Gao, Q. Gao, and H. Wang, "An intrusion detection model based on deep belief networks," *Second International Conference on Advanced Cloud and Big Data,* Nov. 2014, pp. 247-252.

[17] J. Kim, J. Kim, H. L. T. Thu, and H. Kim, "Long short-term memory recurrent neural network classifier for intrusion detection," *International Conference on Platform Technology and Service (PlatCon),* Feb. 2016, pp. 1-5.

[18] S. Potluri, and C. Diedrich, "Accelerated deep neural networks for enhanced intrusion detection system," *IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA),* Sep. 2016, pp. 1-8.

[19] K. Jin, N. Shin, S. Y. Jo, and S. H. Kim, "Method of intrusion detection using deep neural network," *2017 IEEE International Conference on Big Data and Smart Computing (BigComp),* 2017, pp. 313-316.

[20] N. Shone, T. Nguyen Ngoc, V. Dinh Phai, and Q. Shi, "A deep learning approach to network intrusion detection," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 2, no. 1, Feb. 2018.

[21] M. Laheeb Ibrahim, T. Dujan Basheer, and S. Mahmod Mahmod, "A comparison study for intrusion database (KDD99, NSL-KDD) based on self-organization map (SOM) artificial neural network," *Journal of Engineering Science and Technology,* vol. 8, no. 1, pp. 107-119, 2013.

[22] C. Liang, B. Shanmugam, S. Azam, M. Jonkman, F. D. Boer, and G. Narayansamy, "Intrusion detection system for internet of things based on a machine learning approach," *International Conference on Vision Towards Emerging Trends in Communication and Networking (ViTECoN),* 2019.

[23] N. Quamar, W. Sun, A. Y Javaid, and M. Alam, "A deep learning approach for network intrusion detection system," 2016, doi: https://doi.org/10.4108/eai.3-12-2015.2262516.

[24] C. Yin, Y. Zhu, J. Fei., and X. He, "A deep learning approach for intrusion detection using recurrent neural networks," *Complexity*, vol. 2019, Article ID 6516253, 11 pages, doi: https://doi.org/10.1155/2019/6516253.