# iSRITI_ Yesi_

*By* Bhakti Suprapto

# Improving Classification Attacks in IOT Intrusion Detection System using Bayesian Hyperparameter Optimization

Yesi Novaria Kunang[1,2]
[1]Doctoral Engineering Departement, Faculty of Engineering, Universitas Sriwijaya,
[2]Computer Science Departement, Universitas Bina Darma
[1,2]Palembang, Indonesia
yesinovariakunang@binadarma.ac.id

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

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

Bhakti Yudho Suprapto
Electrical Engineering Departement,
Universitas Srwijaya,
Palembang, Indonesia
bhakti@ft.unsri.ac.id

2020 3rd International Seminar on Research of Information Technology and Intelligent Systems (ISRITI) | 978-1-7281-8406-7/20/$31.00 ©2020 IEEE | DOI: 10.1109/ISRITI51436.2020.9315360

**Abstract— The growth of the Internet of Things (IoT) presents challenges in the field of security. The Intrusion Detection System is an alternative to protecting the internet of things. In this study, we propose an intrusion detection system model that combines unsupervised algorithm and a deep neural network. Autoencoder as unsupervised learning algorithm has a function as a feature extractor that feeds up the learning process on a deep neural network. The performance of a deep learning model depends heavily on the selection of hyperparameters of neural network architecture. In this case, we used Bayesian Hyperparameter Optimization to perform hyperparameter tuning of deep learning models with various activation and weight initialization techniques. The accumulation result is useful to help determine the correct activation function and weight initialization and the hyperparameters that most influence the deep learning model. The results of this study show that Bayesian hyperparameter optimization can improve classification results significantly. Evaluation using the BoT-IoT dataset, the classification accuracy results in deep learning model can reach 99.99%.**

***Keywords—Attack classification, hyperparameter, Bayesian optimization, Intrusion detection system, IoT***

## I. INTRODUCTION

The Internet of Things (IoT) is a significant component of industrial automation. A complex paradigm is a way of securing an IoT system that links billions of devices with different characteristics [1]. Different new threats begin to emerge and grow more sophisticated. This increases the need for intelligent security solutions to secure data in IoT networks, namely the Intrusion Detection System (IDS) [2].

Various machine learning (ML) and deep learning (DL) algorithms have been proven to be used in the research area of intrusion detection systems [3]. Especially for DL, some techniques have been widely implemented for anomaly-based IDS such as Deep Neural Network (DNN), Recurrent Neural Network (RNN), Deep Belief Network (DBN) [4]. However, the design of the deep learning model itself has several hyperparameters which need to be decided when constructing and training the model. In order to achieve optimal performance of DL architectural model, the right hyperparameters must be determined [5].

The optimisation method for hyperparameters in deep learning models is a costly computational problem. It takes several hours or even days to evaluation some hyperparameter configuration manually. The tuning process can be done automatically with various methods such as Grid search [12], Random search [14], and Bayesian optimization [6], [15] which is more flexible than manually tuning hyperparameters (trial and error) on deep learning models. Among these methods, Bayesian optimization can produce better and faster configurations than HPO with grid search and random search techniques [5], [15]

Contributions in this study include are: (1) we proposed deep learning models using unsupervised autoencoder and supervised deep neural networks using the Bayesian hyperparameter optimization approach; (2) The results of this study also evaluate the use of ReLU variant activation function in deep learning model so it can help determine the proper activation function in deep learning model; (3) In evaluation process we used various weight initialization techniques to evaluate the impact of initialization technique to the activation function.

We list the abbreviations and acronyms used in the paper as a quick and convenient guide in Table 1.

TABLE I. ACRONYMS AND ABBREVIATIONS LIST

| | |
|---|---|
| ANN | Artificial Neural Network |
| BO | Bayesian Optimization |
| DAE | Deep Autoencoder |
| DDoS | Distributted Denial of Services |
| DL | Deep Learning |
| DNN | Deep Neural Network |
| DoS | Denial of Service |
| ELU | Exponential Linear Unit |
| HPO | Hyperparameter Optimization |
| IDS | Intrusion Detection System |
| IoT | Internet of Things |
| ML | Machine Learning |
| NIDS | Nework Intrusion Detection System |
| PreLU | Parametric ReLU |
| ReLU | Rectified linear units |
| SELU | Scaled Exponential Linear Unit |
| SMOTE | Synthetic Minority Oversampling Technique |

The rest of this paper is structured accordingly. In Section 2, we briefly analyze some relevant works corresponding to a state-of-the-art method of deep learning and optimization of hyperparameters in the framework of intrusion detection. Section 3 presents our proposed DL architecture, and some

This research has received funding from Indonesia Ministry of Research, Technology, and Higher Education (grant agreement 170/SP2H/LT/DRPM/2020).

methods of our work are used. We give explanations of our experiments in Section 4 and present the results. Finally, in section 5, we end this paper with a few remarks.

## II. Related Work

Utilization of deep learning (DL) in intrusion detection systems has succeeded in improving accuracy and attack recognition [7]. The classification performance of the DL model depends on the architecture of the DL. Some researchers use various autoencoder variants in NIDS for feature learning processes combined with different classifiers [8]–[11]. Author Yang et al. [8] combined improved conditional variational autoencoder (ICVAE) with a deep neural network. In the process to get the best model, a manual grid search was performed to determine hyperparameters such as the number of hidden layers, the number of nodes, the learning rate, and L2.

Other researchers Rezvy et al. [9] used an AE autoencoder in combination with a dense neural network. AL-Hawawreh et al. [10] proposed Deep Autoencoder (DAE) in combination with the Deep Feed Forward Neural Network (DFFNN) as a classifier. Zhao et al. [11] proposed method to identify new forms of attacks through a Semi-Supervised Discriminant Auto-encoder (SSDA) combined with a heuristic denoising

Most of the selection of DL model structures such as the number of nodes, the number of hidden layers and several other hyperparameters in previous researches performed manually (trial and error) or with a limited combination of grids. In fact, model performance is very sensitive to hyperparameter layout as shown by Pawlicki 2019 [12] which uses manual grid search to find hyperparameter epoch, batch size, activation function, optimizer, number of hidden layers and number of neurons in Artificial Neural Network (ANN) based on NIDS. The hyperparameter tuning process is essential to improve deep learning performance [6], [13].

## III. Method and Design

### A. Proposed Model

In our previous study [16], we used the autoencoder based feature extraction with various activation functions and a varied number of nodes. The results of this study show the activation function, and the number of nodes in the hidden layer significantly determine the detection and classification performance. However, the process of feature extraction in the intrusion detection system [16] still uses manual search based trial and error and is very time-consuming. For this reason, in this research, we propose an intrusion detection system with an autoencoder pre-training process and a DNN classification with a hyperparameter tuning process based on Bayesian optimization (BO).

The phases of the proposed model can be seen in Figure 1, which starts by choosing a dataset and pre-processing dataset. For pre-training, the deep learning model uses an unsupervised autoencoder algorithm which extracts features. The encoder layer model will be transferred to a classifier using a deep neural network. To improve accuracy performance, we used automatic hyperparameter tuning based on a Bayesian optimization (BO). The evaluated hyperparameters are learning rate, number of nodes on a hidden layer, batch size, activation function and weight initialization. One of the focuses of this research is to investigate the effect of hyperparameter tuning on deep learning models using only 1 hidden layer. The Gaussian BO function automatically updates the hyperparameter value to the model evaluated by a certain number of iterations to obtain the best results.

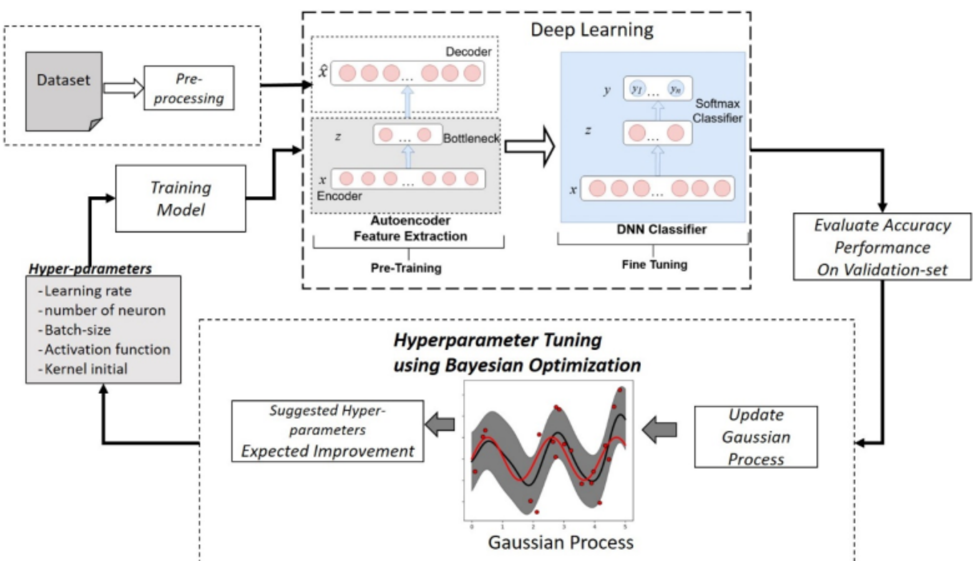


Fig. 1. Deep Learning Architecture-based Attacks classification

### B. Dataset

This study uses the Bot-IoT dataset [17], which represents an attack on the IoT environment. This dataset has been selected because it represents a quite realistic IoT attack environment compared to the previous dataset. This dataset consists of four types of attacks, namely DoS, DDoS, information gathering and information theft. The attack types are grouped into eleven sub-classes as in Table 2. The number of records from the dataset is quite large, more than 72 million records. In this experiment, we used the 5% version of the entire dataset that consists of 46 features.

TABLE II. COMPOSITION OF THE BOT-IOT DATASET USED

| Category | Sub Category | Flow count | Training-set | Testing-set | Training-set SMOTE |
|---|---|---|---|---|---|
| Normal | Normal | 477 | 381 | 96 | 50000 |
| DoS | UDP | 1032975 | 826412 | 206563 | 826412 |
| | TCP | 615800 | 492826 | 122974 | 492826 |
| | HTTP | 1485 | 1217 | 268 | 50000 |
| DDoS | UDP | 948255 | 758690 | 189565 | 758690 |
| | TCP | 977380 | 781607 | 195773 | 781607 |
| | HTTP | 989 | 787 | 202 | 50000 |
| Reconnaissance | Service_Scan | 73168 | 58470 | 14698 | 58470 |
| | OS_Fingerprint | 17914 | 14364 | 3550 | 50000 |
| Theft | Keylogging | 73 | 59 | 14 | 50000 |
| | Data_Exfiltration | 6 | 7 | 3 | 50000 |
| **Total** | | 3668522 | 2934820 | 733706 | 3218005 |

In feature selection From 46 features of the BoT-IoT dataset [17], there were several features were eliminated, i.e. pkSeqID, stime, flgs_number, proto_number, saddr, sport, daddr, dport, state_number, and ltime. These features were eliminated due to some duplication of features, and other features are more specific to the identity of packages that are not related to an attack's characteristics. Next step, we applied feature modification in label subcategory by merging label category-subcategory to facilitate the grouping of attack types. Then attack, category and subcategory features are used as labels. In the feature encoding process, we used a one-hot encoding method for nominal or categorical features in flags, protocol and state. The feature flag was mapped to 9 features, and the feature protocol became 5, the feature state became 11. Feature labels have also been mapped into five classes for categories and eleven classes for subcategory. After the encoding process, the overall features of the data set were transformed into 56 features, which became the input features of the deep learning model.

Furthermore, the dataset was divided into 80% into training sets and 20% into testing sets. Specifically, for the data_exfiltration class, we added data redundancy for the balanced process of the datasheet. The number of classes of the subcategory itself is very imbalanced, especially for the theft attack category (see table 2). We used the Synthetic Minority Oversampling Technique (SMOTE) method to generate balanced data then the IDS model could detect all class attacks correctly. This balanced SMOTE process was only carried out on the training set data by upsampling from small data to 50000 samples.

The last step, Feature scaling is a process to convert data into a specific range. In this phase, the feature scaling process on training-set and testing-set uses Min-max scaling with a range of [0.1]. Then the data is ready to be used on the deep learning model

*C. Deep Learning Model*

Figure 1 shows a proposed deep architectural model. The deep learning model uses a pre-training process using an unsupervised autoencoder and a fine-tuning process with a supervised DNN. The pre-training process with this autoencoder is useful for extracting knowledge from unsupervised training sets. Autoencoder consists of 2 stages of encoding and decoding. The encoding process will compress the input data into a low-dimensional representation. Then the decoder process will reconstruct the low-dimensional data from the encoder so that the output results from this autoencoder process produce an output that is close to the input data. The encoder vector function $h$ is denoted as given in (1).

$$z = f(W.x + b) \quad (1)$$

$W$ is the weight matrix, and $b$ is the bias vector, $x$ is the input feature vector, while $f\,(.)$ is the activation function used.

The decoder function returns the vector $z$ to the output vector $\hat{x}$, which has the same dimension as the input.

$$\hat{x} = f(W^T.h + b') \quad (2)$$

The activation function $f$ is in this decoded process used sigmoid. The loss function uses the mean square error ( MSE) that will minimize the difference between the input and output vectors.

Then by using the encoding vector z, which is a representation of the extracted data, will be transferred to the DNN model. So the DNN model will consist of 3 layers, namely the input layer and the hidden layer that comes from the encoding layer, is then added to the output layer using softmax activation. In supervised DNN, using bias values and matric weights transferred from the encoding layer is retrained to predict label class. Output labels will be evaluated in each iteration using categorical cross-entropy as loss function, and adam as optimizer function.

*D. Bayesian Optimisation*

Bayesian Optimization (BO) is an optimization model that generates predictive distributions of potential value with a probability approach to optimize the Blackbox function to be optimized [18]. This Bayesian optimization process will look for the next sampling point $x_t$ value by optimizing the acquisition function using the Gaussian process:

$$x_t = argmax_x u(x|D_{1:t-1}) \quad (3)$$

Where $t = 1,2,\cdots$ is an iteration for several samples of hyperparameter combinations to be evaluated, $x$ is the hyperparameter value to be observed. $u$ is the acquisition function, and $D_{1:t-1}$ is the posterior distribution to be optimized.

In this experiment, we use the Expected Improvement (EI) (in equation 4) as acquisition function to take a sample from the

$$EI(\boldsymbol{x}) = \mathbb{E}max(f(\boldsymbol{x}) - f(\boldsymbol{x}^{+}),0) \quad (4)$$

$f(x^{+})$ represents the best sample value to far and $x^{+}$ is the sample 's position.

*E. Performance Metrics and Environment Setup*

Evaluation of deep learning after hyperparameter optimization used accuracy as a metric performance. We also observed the time required to see the impact of the activation and kernel initialization functions on the model in the deep model training phase. After obtaining the most accurate performance of the model, it is evaluated using the testing-set.

The experiment was conducted using Python on the Google Collaboratory framework using the Tensorflow and Keras libraries. The Bayesian Optimization Hyperparameter Method used the Skopt Optimization Library [19] to produce the best model. Some of the hyperparameter values that the tuning process performs include the value of the learning rate, the number of hidden layer nodes, the activation function, and the initial kernel (see Figure 1).

The pre-training process for the autoencoder and fine-tuning of the deep learning model only uses 10 epochs each stage. When training the DNN model, the training set that has been upsampled with SMOTE becomes 3,218,005 records data, 80% split for the learning process, and 20% for the validation process. The best model obtained is re-evaluated using a testing-set of 733,706 data.

## IV. Results and Discussion

### A. *Results of Bayesian Optimization*

BO is only effective on continuous hyperparameters [15]. Therefore, BO was run in parallel to accelerate the tuning process for hyperparameters in categorical dimensions such as activation and weight initialization. Table 3 showed the effects of hyperparameter tuning for the entire activation function and the initial kernel function.

TABLE III. Best accuracy results for various activation and weight intialization.

| Activation function | Weight Initialization | base value [a] Accuracy | base value [a] training time | the best value after tuning Learning rate | nodes | batch size | Accuracy |
|---|---|---|---|---|---|---|---|
| ELU | lecun_normal | 0.8455 | 288.627 | 0.0100 | 45 | 256 | 0.99975 |
| | glorot_uniform | 0.8479 | 281.494 | 0.0019 | 45 | 32 | 0.99990 |
| | he_normal | 0.8207 | 282.519 | 0.0100 | 45 | 128 | 0.99977 |
| | lecun_uniform | 0.8407 | 282.028 | 0.0100 | 45 | 256 | 0.99974 |
| | glorot_normal | 0.8376 | 283.488 | 0.0100 | 45 | 256 | 0.99976 |
| | he_uniform | 0.8413 | 282.708 | 0.0100 | 45 | 256 | 0.99989 |
| | normal | 0.8233 | 269.454 | **0.0085** | **37** | **64** | **0.99991** |
| LeakyReLU | lecun_normal | 0.8250 | 266.204 | 0.0051 | 28 | 128 | 0.98645 |
| | glorot_uniform | 0.8385 | 266.55 | 0.0100 | 20 | 256 | 0.98652 |
| | he_normal | 0.8291 | 266.814 | 0.0014 | 45 | 256 | 0.98481 |
| | lecun_uniform | 0.8299 | 266.395 | 0.0029 | 45 | 32 | 0.98374 |
| | glorot_normal | 0.8377 | 267.113 | 0.0100 | 20 | 256 | 0.98526 |
| | he_uniform | 0.8377 | 264.356 | 0.0006 | 20 | 32 | 0.98369 |
| | normal | 0.8219 | 264.731 | 0.0007 | 44 | 32 | 0.98574 |
| PReLU | lecun_normal | 0.8353 | 232.745 | 0.0100 | 45 | 128 | 0.98474 |
| | glorot_uniform | 0.8398 | 235.674 | 0.0036 | 45 | 256 | 0.98598 |
| | he_normal | 0.8381 | 234.04 | 0.0040 | 21 | 32 | 0.98485 |
| | lecun_uniform | 0.8322 | **232.393** | 0.0020 | 45 | 32 | 0.98539 |
| | glorot_normal | 0.8278 | 233.031 | 0.0024 | 45 | 256 | 0.98501 |
| | he_uniform | 0.8317 | 233.07 | 0.0011 | 20 | 32 | 0.98512 |
| | normal | 0.8167 | 236.35 | 0.0100 | 43 | 256 | 0.98607 |
| ReLU | lecun_normal | 0.8527 | 250.305 | 0.0099 | 39 | 64 | 0.99989 |
| | glorot_uniform | 0.8432 | 248.926 | 0.0100 | 45 | 32 | 0.99985 |
| | he_normal | 0.8555 | 247.071 | 0.0100 | 45 | 256 | 0.99985 |
| | lecun_uniform | 0.8480 | 250.595 | 0.0100 | 45 | 256 | 0.99982 |
| | glorot_normal | 0.8569 | 252.598 | 0.0100 | 45 | 32 | 0.99979 |
| | he_uniform | 0.8594 | 249.746 | 0.0100 | 45 | 32 | 0.99986 |
| | normal | 0.8348 | 252.115 | 0.0061 | 45 | 256 | 0.99967 |
| SELU | lecun_normal | 0.8620 | 252.041 | 0.0017 | 45 | 32 | 0.99989 |
| | glorot_uniform | 0.8775 | 254.434 | 0.0100 | 45 | 32 | 0.99970 |
| | he_normal | 0.8764 | 254.358 | 0.0021 | 34 | 128 | 0.99958 |
| | lecun_uniform | 0.8657 | 254.35 | 0.0034 | 45 | 32 | 0.99982 |
| | glorot_normal | 0.8767 | 255.125 | 0.0100 | 45 | 64 | 0.99833 |
| | he_uniform | **0.8791** | 253.952 | 0.0053 | 25 | 32 | 0.99967 |
| | normal | 0.8590 | 251.503 | 0.0100 | 25 | 256 | 0.99958 |

[a] for base value learning rate= 0.00001, number of neuron =30, and batch size=256

There were 5 activation functions of ReLU variants evaluated in the experiment. Each of these activation functions was tested using various weight initialization functions. So there were 35 kinds of combinations between the activation function and the kernel initialization. The deep learning model is evaluated with 35 combinations with the hyperparameter tuning BO process using the Gaussian Process for the value of learning rate, number of nodes and batch_size. The base value of the variety of the activation and kernel initialization functions uses the learning rate = 0.00001, the number of neurons in the hidden layer = 30 and batch_size = 256.

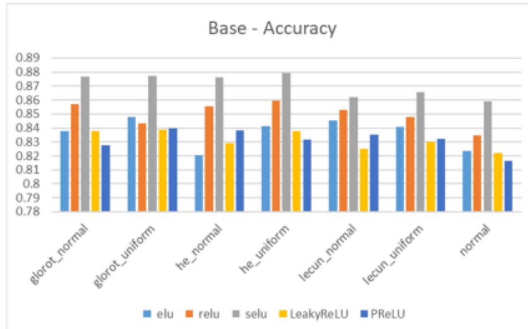


Fig. 2. Performance-based model based on combination activation and kernel initialization using base value before tuning hyperparameter

The SELU activation function dominates the other activation functions for the whole weight initialization function in table 3 and Figure 2. SELU is a self-normalizing development of ELU, makes learning particularly robust and faster [20]. And the highest accuracy of 87.91 % is achieved by he_uniform weight initialization. Figure 2 also shows that the ReLU activation function is relatively stable for base value.

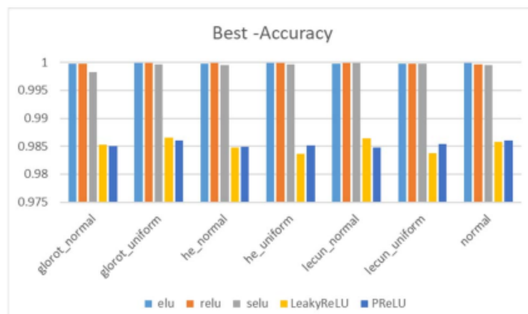


Fig. 3. Performance-based model based on combination activation and kernel initialization after tuning hyperparameter

We can observe the training process time for the base value in Table 3. With the same hyperparameters, it points out that PReLU activation function is the activation function with the fastest training process. It shows that PReLU tends to perform better for a limited number of epochs than ReLU. In the meantime, ELU activation function is the activation function with the longest learning time. It indicates that ELU is slightly slower to train from these findings. The ELU function has an exponential computation that slows down the process [21]. Meanwhile, the effect of weight initialization is not significant due to the unpredictable random factor.

After the hyperparameter tuning process with the total number of evaluations was carried out for each combination n calls = 15, the entire model has been evaluated with as much as 525 options. Table 3 and Figure 3 show that, after tuning with BO and Gaussian Process, the accuracy value increases to over 98.3%. Using the ELU activation function, normal weight initialization, learning rate value at 0.0085, number of nodes 37 and batch size 64, the results of the best accuracy value reaching 99.991%.

Figure 3 also indicates the accuracy of the LeakyReLU and PReLU activation functions underperformed due to other activation functions. For all the tests performed, the LeakyReLU alpha value was 0.01. The effects of LeakyReLU could be affected by choosing the α value which is not the optimal. Similarly, PReLU activation used only the default value of the keras library in the experiment. There are several alpha parameters PReLU itself, such as initialization, restriction and regularizer, which must be tuned also to obtain optimum value. Figure 3 also indicates the efficiency of the LeakyReLU and PReLU activation functions underperformed due to other activation functions.

The BO procedure searches hyperparameters based on the search-space dimension range on the learning rate, batch size and number of hidden nodes. Figure 4 described the search space dimension on the objective function for the ELU activation function and normal weight initialization. The value of partial dependencies is determined by calculating the average target value of many random samples for the dimension of the learning rate, batch size and the number of nodes. The red star indicates the best-observed value. In Figure 5, the best value for the study rate is 0.0085, batch_size is 64 ($2^6$) and the dense number 37. In the figure, the closer the average accuracy as objective value is to 0.01 (low = 1e-6, high = 1e-2). The higher the average accuracy. Similarly, the higher the number of nodes, the higher the average accuracy of the objective value. In table 3, most of the best values are obtained in the number of nodes in the hidden layer at the largest value of the dimensional interval (low = 20, high = 45).

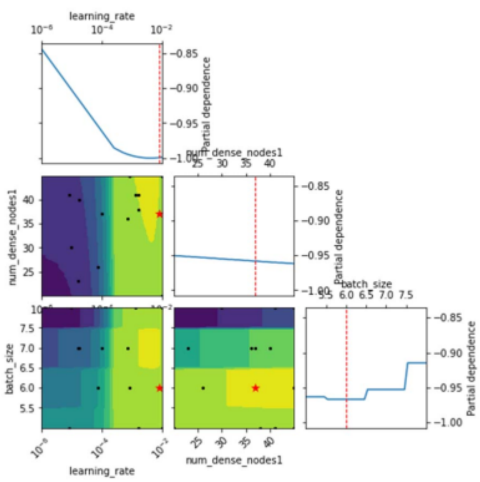


Fig. 4. Partial Dependence plots of the objective function in ELU as activation function and kernel_initialization normal

The effect of variable continuous hyperparameters that are tuned based on the impact of increasing accuracy performance can be evaluated from the general results of various combinations of hyper-parameters (Figure 5). The results of the different combinations of hyperparameters were tested with a random forest regressor to decide the hyperparameters features had the greatest effect on the increased accuracy. And results are shown to be the learning rate followed by the batch size value for that have a significant impact on improving accuracy. In contrast, the number of neurons doesn't have a considerable effect on classification accuracy.

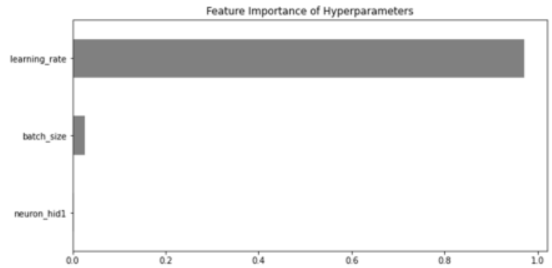


Fig. 5. Feature Importance of Continous hyperparameters

### *B. Performance Evaluation and comparison*

After finding the best model for all ReLU, ELU, SELU, LeakyReLU and PReLU activation functions, the best model is evaluated using a testing-set. Table 4 summarizes the findings of the best model assessment.

TABLE IV. EVALUATION OF BEST MODEL IN TESTING-SET

| Activation function | weight intialization | Data Train | | Data Test | |
|---|---|---|---|---|---|
| | | Loss | Accuracy | Loss | Accuracy |
| ELU | normal | 0.000657 | 0.999915 | 0.001214 | 0.999931 |
| SELU | lecun_normal | 0.000729 | 0.999905 | 0.000854 | 0.999920 |
| ReLU | lecun_normal | 0.000907 | 0.999892 | 0.001423 | 0.999890 |
| LeakyReLU | glorot_uniform | 0.066026 | 0.986534 | 0.069737 | 0.985508 |
| PReLU | normal | 0.067739 | 0.986068 | 0.070105 | 0.985264 |

The best score for the ELU activation function on the testing set is 99,993%. The best accuracy as objective function value is close to 100 % for any weight initialization with the hyperparameters of the search space (15 calls). Hyperparameter tuning with Gaussian method accelerates the deep-learning model's convergence, while in autoencoder as well as in deep neural network processes the number of the epoch was only 10.

After running all those experiments, the activation function indeed affect the final accuracies and the losses. ELU activation is robust than ReLU, SELU, Leaky ReLU and PReLU activation functions. The discussion of previous studies by Pedamonti [20] supported this finding. Furthermore, the stability of the activation function must be considered when selecting it. The results show that after tuning, LeakyReLU and PReLU are significantly less accurate than other activations. So it is necessary to determine the value of the alpha parameter for both activation functions.

Overall, the results obtained using just a small epoch with BO's hyperparameter tuning method are very strong with classification results will exceed 99.993% even with a small number of epochs (10 epochs in the pre-training method and 10 epochs in the classification process). Compared to some previous studies using the same data set for multiclass classification, the detection rate (weighted overall accuracy) results achieved are superior. As in previous studies by Alkadi et al. [22] using Mixture Localization-based Outliers (MLOs) and Gaussian mixture model to classify multiclass attacks trends of only 97.98%. Similarly, the best results for deep autoencoder models with a detection rate of 98.394% were obtained for research carried out by Ferrag et al. which evaluated several deep learning models. Khraisat et al. [23] proposed a hybrid intrusion detection system using an ensemble method for the same datasets with features selection. The detection rate of the proposed model was 99.97%.

Finally, our experiment has shown that the performance depends on how we change the optimizer hyperparameters and how we set up our deep learning model. In this study, ELU activation and normal weight initializer with Bayesian optimization using the Gaussian process were better than the others, but this may not be the case with other tasks and other data. In developing a deep learning model, multiple possibilities need to be tested, and hyper-parameter selections used to evaluate which options are best.

## V. CONCLUSION

We proposed in this study deep learning models in intrusion detection framework using unsupervised autoencoder and supervised deep neural networks using the Bayesian hyperparameter optimization method. With the hyperparameter tuning process using Bayesian optimization, the detection rate value of the BoT-IoT dataset has been increased to 99.993%, which is higher than the previous state of the art. From the results of the best model after hyperparameter tuning, it was found that compared to the evaluated ReLU variant, the ELU activation function provided the best performance. Based on the obtained results, the learning rate value is the continuous parameter, which influences most in deep learning performance, while batch sized and the number of hidden layer nodes has no significant effect.

In the future, we will evaluate the deep learning model by adding the number of hidden layers and other hyperparameters and comparing them with various deep learning methods and various datasets.

## ACKNOWLEDGMENT

The authors would like to thank data science research group, Universitas Bina Darma, and University for support and facilities.

## REFERENCES

[1] M. Serror, S. Hack, M. Henze, M. Schuba, and K. Wehrle, "Challenges and Opportunities in Securing the Industrial Internet of Things," IEEE Trans. Ind. Inf., pp. 1–1, 2020, doi: 10.1109/TII.2020.3023507.

[2] N. Chaabouni, M. Mosbah, A. Zemmari, C. Sauvignac, and P. Faruki, "Network intrusion detection for IoT security based on learning techniques," IEEE Communications Surveys & Tutorials, vol. 21, no. 3, pp. 2671–2701, 2019.

[3] R. Abdulhammed, M. Faezipour, A. Abuzneid, and A. AbuMallouh, "Deep and Machine Learning Approaches for Anomaly-Based Intrusion Detection of Imbalanced Network Traffic," IEEE Sens. Lett., vol. 3, no. 1, pp. 1–4, Jan. 2019, doi: 10.1109/LSENS.2018.2879990.

[4] M. A. Ferrag, L. Maglaras, S. Moschoyiannis, and H. Janicke, "Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study," Journal of Information Security and Applications, vol. 50, p. 102419, Feb. 2020, doi: 10.1016/j.jisa.2019.102419.

[5] M. Feurer and F. Hutter, "Hyperparameter Optimization," in Automated Machine Learning: Methods, Systems, Challenges, F. Hutter, L. Kotthoff, and J. Vanschoren, Eds. Cham: Springer International Publishing, 2019, pp. 3–33.

[6] M. Zhang, H. Li, J. Lyu, S. H. Ling, and S. Su, "Multi-level CNN for lung nodule classification with Gaussian Process assisted hyperparameter optimization," arXiv:1901.00276 [cs, eess], Jan. 2019, Accessed: Sep. 26, 2020. [Online]. Available: http://arxiv.org/abs/1901.00276.

[7] Y. Xin et al., "Machine Learning and Deep Learning Methods for Cybersecurity," IEEE Access, vol. 6, pp. 35365–35381, 2018, doi: 10.1109/ACCESS.2018.2836950.

[8] Y. Yang, K. Zheng, C. Wu, and Y. Yang, "Improving the Classification Effectiveness of Intrusion Detection by Using Improved Conditional Variational AutoEncoder and Deep Neural Network," Sensors, vol. 19, no. 11, p. 2528, Jun. 2019, doi: 10.3390/s19112528.

[9] S. Rezvy, M. Petridis, A. Lasebae, and T. Zebin, "Intrusion Detection and Classification with Autoencoded Deep Neural Network," in Innovative Security Solutions for Information Technology and Communications, vol. 11359, J.-L. Lanet and C. Toma, Eds. Cham: Springer International Publishing, 2019, pp. 142–156.

[10] M. AL-Hawawreh, N. Moustafa, and E. Sitnikova, "Identification of malicious activities in industrial internet of things based on deep learning models," Journal of Information Security and Applications, vol. 41, pp. 1–11, Aug. 2018, doi: 10.1016/j.jisa.2018.05.002.

[11] F. Zhao, H. Zhang, J. Peng, X. Zhuang, and S.-G. Na, "A semi-self-taught network intrusion detection system," Neural Comput & Applic, Apr. 2020, doi: 10.1007/s00521-020-04914-7.

[12] M. Pawlicki, R. Kozik, and M. Choraś, "Artificial Neural Network Hyperparameter Optimisation for Network Intrusion Detection," in Intelligent Computing Theories and Application, vol. 11643, D.-S. Huang, V. Bevilacqua, and P. Premaratne, Eds. Cham: Springer International Publishing, 2019, pp. 749–760.

[13] Y. Yoo, "Hyperparameter optimization of deep neural network using univariate dynamic encoding algorithm for searches," Knowledge-Based Systems, vol. 178, pp. 74–83, Aug. 2019, doi: 10.1016/j.knosys.2019.04.019.

[14] J. F. Torres, D. Gutiérrez-Avilés, A. Troncoso, and F. Martínez-Álvarez, "Random Hyper-parameter Search-Based Deep Neural Network for Power Consumption Forecasting," in Advances in Computational Intelligence, vol. 11506, I. Rojas, G. Joya, and A. Catala, Eds. Cham: Springer International Publishing, 2019, pp. 259–269.

[15] E. C. Garrido-Merchán and D. Hernández-Lobato, "Dealing with Categorical and Integer-valued Variables in Bayesian Optimization with Gaussian Processes," Neurocomputing, vol. 380, pp. 20–35, Mar. 2020, doi: 10.1016/j.neucom.2019.11.004.

[16] Y. N. Kunang, S. Nurmaini, D. Stiawan, A. Zarkasi, and F. Jasmir, "Automatic Features Extraction Using Autoencoder in Intrusion Detection System," in 2018 International Conference on Electrical Engineering and Computer Science (ICECOS), Pangkal Pinang, Oct. 2018, pp. 219–224, doi: 10.1109/ICECOS.2018.8605181.

[17] N. Koroniotis, N. Moustafa, E. Sitnikova, and B. Turnbull, "Towards the development of realistic botnet dataset in the internet of things for network forensic analytics: Bot-iot dataset," Future Generation Computer Systems, vol. 100, pp. 779–796, 2019.

[18] P. Murugan, "Hyperparameters Optimization in Deep Convolutional Neural Network / Bayesian Approach with Gaussian Process Prior," arXiv:1712.07233 [cs, stat], Dec. 2017, Accessed: Sep. 26, 2020. [Online]. Available: http://arxiv.org/abs/1712.07233.

[19] "skopt," Scikit-Optimize. https://pypi.org/project/scikit-optimize/ (accessed Oct. 07, 2020).

[20] D. Pedamonti, "Comparison of non-linear activation functions for deep neural networks on MNIST classification task," arXiv:1804.02763 [cs, stat], Apr. 2018, Accessed: Mar. 26, 2020. [Online]. Available: http://arxiv.org/abs/1804.02763.

[21] M. A. Mercioni and S. Holban, "The Most Used Activation Functions: Classic Versus Current," in 2020 International Conference on Development and Application Systems (DAS), Suceava, Romania, May 2020, pp. 141–145, doi: 10.1109/DAS49615.2020.9108942.

[22] O. AlKadi, N. Moustafa, B. Turnbull, and K.-K. R. Choo, "Mixture Localization-Based Outliers Models for securing Data Migration in Cloud Centers," IEEE Access, vol. 7, pp. 114607–114618, 2019, doi: 10.1109/ACCESS.2019.2935142.

[23] Khraisat, Gondal, Vamplew, Kamruzzaman, and Alazab, "A novel Ensemble of Hybrid Intrusion Detection System for Detecting Internet of Things Attacks," Electronics, vol. 8, no. 11, p. 1210, Oct. 2019, doi: 10.3390/electronics8111210.

# iSRITI_ Yesi_

ORIGINALITY REPORT

**17%**
SIMILARITY INDEX

PRIMARY SOURCES

| | | |
|---|---|---|
| 1 | sami muhaidat. "Intelligent Reflecting Surfaces Assisted UAV Communications for IoT Networks: Performance Analysis", Institute of Electrical and Electronics Engineers (IEEE), 2021<br>Crossref Posted Content | 60 words — 1% |
| 2 | journal.uad.ac.id<br>Internet | 53 words — 1% |
| 3 | Yesi Novaria Kunang, Siti Nurmaini, Deris Stiawan, Bhakti Yudho Suprapto. "Attack classification of an intrusion detection system using deep learning and hyperparameter optimization", Journal of Information Security and Applications, 2021<br>Crossref | 51 words — 1% |
| 4 | repository.unika.ac.id<br>Internet | 40 words — 1% |
| 5 | rie.binadarma.ac.id<br>Internet | 39 words — 1% |
| 6 | arxiv.org<br>Internet | 35 words — 1% |
| 7 | www.mdpi.com<br>Internet | 25 words — 1% |

| | | |
|---|---|---|
| 8 | downloads.hindawi.com<br>Internet | 24 words — 1% |
| 9 | Khraisat, Gondal, Vamplew, Kamruzzaman, Alazab. "A novel Ensemble of Hybrid Intrusion Detection System for Detecting Internet of Things Attacks", Electronics, 2019<br>Crossref | 23 words — 1% |
| 10 | YoungJun Yoo. "Hyperparameter optimization of deep neural network using univariate dynamic encoding algorithm for searches", Knowledge-Based Systems, 2019<br>Crossref | 20 words — < 1% |
| 11 | link.springer.com<br>Internet | 20 words — < 1% |
| 12 | dokumen.pub<br>Internet | 19 words — < 1% |
| 13 | "Intelligent Computing Theories and Application", Springer Science and Business Media LLC, 2019<br>Crossref | 18 words — < 1% |
| 14 | Xiuquan Du, Shiwei Sun, Changlin Hu, Yu Yao, Yuanting Yan, Yanping Zhang. "DeepPPI: Boosting Prediction of Protein–Protein Interactions with Deep Neural Networks", Journal of Chemical Information and Modeling, 2017<br>Crossref | 16 words — < 1% |
| 15 | "Advances in Computational Intelligence", Springer Science and Business Media LLC, 2019<br>Crossref | 15 words — < 1% |
| 16 | www.proceedings.com<br>Internet | 14 words — < 1% |

| | | |
|---|---|---|
| 17 | Hongpo Zhang, Lulu Huang, Chase Q. Wu, Zhanbo Li. "An Effective Convolutional Neural Network Based on SMOTE and Gaussian Mixture Model for Intrusion Detection in Imbalanced Dataset", Computer Networks, 2020<br>Crossref | 13 words — < 1% |
| 18 | Andrew Brandon, Melanie Seekins, Beulah Vedhavalli Joshua, Chelliah Samuel, John Haller. "Network Data Analysis to Support Risk Management in an IoT Environment", 2019 IEEE 10th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON), 2019<br>Crossref | 12 words — < 1% |
| 19 | Sungmin Yoon. "In-situ sensor calibration in an operational air-handling unit coupling autoencoder and Bayesian inference", Energy and Buildings, 2020<br>Crossref | 12 words — < 1% |
| 20 | www.spiedigitallibrary.org<br>Internet | 12 words — < 1% |
| 21 | "ICCCE 2020", Springer Science and Business Media LLC, 2021<br>Crossref | 11 words — < 1% |
| 22 | "New Knowledge in Information Systems and Technologies", Springer Science and Business Media LLC, 2019<br>Crossref | 11 words — < 1% |
| 23 | assets.researchsquare.com<br>Internet | 11 words — < 1% |

24 "[Front cover]", 2020 2nd IEEE International Conference on Artificial Intelligence Circuits and Systems (AICAS), 2020 — 10 words — < 1%
Crossref

25 Ahmed Akl, Ibrahim El-Henawy, Ahmad Salah, Kenli Li. "Optimizing deep neural networks hyperparameter positions and values", Journal of Intelligent & Fuzzy Systems, 2019 — 10 words — < 1%
Crossref

26 Ruizhe Zhao, Yingxue Mu, Long Zou, Xiumei Wen. "A hybrid intrusion detection system based on feature selection and weighted Stacking classifier", IEEE Access, 2022 — 10 words — < 1%
Crossref

27 Hudalizaman, Igi Ardiyanto, Sunu Wibirama. "Network Architecture Search Method on Hyperparameter Optimization of Convolutional Neural Network: Review", 2020 6th International Conference on Science and Technology (ICST), 2020 — 9 words — < 1%
Crossref

28 Muhammad Aqeel Aslam, Aslam, Daxiang Cui. "Breast Cancer Classification using Deep Convolutional Neural Network", Journal of Physics: Conference Series, 2020 — 9 words — < 1%
Crossref

29 Yong Fang, Yuetian Zeng, Beibei Li, Liang Liu, Lei Zhang. "DeepDetectNet vs RLAttackNet: An adversarial method to improve deep learning-based static malware detection model", PLOS ONE, 2020 — 9 words — < 1%
Crossref

30 hal.archives-ouvertes.fr

Internet

9 words — < 1%

31 mdpi-res.com
Internet

9 words — < 1%

32 portal.research.lu.se
Internet

9 words — < 1%

33 research.library.mun.ca
Internet

9 words — < 1%

34 www.jait.us
Internet

9 words — < 1%

35 "Advances in Security, Networks, and Internet of Things", Springer Science and Business Media LLC, 2021
Crossref

8 words — < 1%

36 "Soft Computing: Theories and Applications", Springer Science and Business Media LLC, 2022
Crossref

8 words — < 1%

37 B. G. Galuzzi, I. Giordani, A. Candelieri, R. Perego, F. Archetti. "Hyperparameter optimization for recommender systems through Bayesian optimization", Computational Management Science, 2020
Crossref

8 words — < 1%

38 Mohammad Al Razib, Danish Javeed, Muhammad Taimoor Khan, Reem Alkanhel, Mohammed Saleh Ali Muthanna. "Cyber Threats Detection in Smart Environments Using SDN-Enabled DNN-LSTM Hybrid Framework", IEEE Access, 2022
Crossref

8 words — < 1%

39 Wenliao Du, Pengjie Hu, Hongchao Wang, Xiaoyun Gong. "Bearing Fault Diagnosis Based on Wavelet Transform and Auto-Encoder Neural Network", 2019 International Conference on Sensing, Diagnostics, Prognostics, and Control (SDPC), 2019 — 8 words — < 1%
Crossref

40 Zainab Alothman, Mouhammd Alkasassbeh, Sherenaz Al-Haj Baddar. "An efficient approach to detect IoT botnet attacks using machine learning", Journal of High Speed Networks, 2020 — 8 words — < 1%
Crossref

41 aaltodoc.aalto.fi — 8 words — < 1%
Internet

42 ebin.pub — 8 words — < 1%
Internet

43 online-journals.tubitak.gov.tr — 8 words — < 1%
Internet

44 www.hindawi.com — 8 words — < 1%
Internet

45 "Intelligent Systems and Applications", Springer Science and Business Media LLC, 2021 — 7 words — < 1%
Crossref

46 "Machine Learning for Networking", Springer Science and Business Media LLC, 2020 — 7 words — < 1%
Crossref

47 Pinkey Chauhan, M. Atulkar. "Selection of Tree Based Ensemble Classifier for Detecting Network Attacks in IoT", 2021 International Conference on Emerging Smart Computing and Informatics (ESCI), 2021 — 7 words — < 1%

Crossref

48 Soheyl Massoudi, Cyril Picard, Jürg Schiffmann. "Robust design using multiobjective optimisation and artificial neural networks with application to a heat pump radial compressor", Design Science, 2022 — 7 words — < 1%
Crossref

49 Md Abdul Awal, Mehedi Masud, Md Shahadat Hossain, Abdullah Al-Mamun Bulbul, S. M. Hasan Mahmud, Anupam Kumar Bairagi. "A Novel Bayesian Optimization-based Machine Learning Framework for COVID-19 Detection from Inpatient Facility Data", IEEE Access, 2021 — 6 words — < 1%
Crossref

50 Sunanda Gamage, Jagath Samarabandu. "Deep learning methods in network intrusion detection: A survey and an objective comparison", Journal of Network and Computer Applications, 2020 — 6 words — < 1%
Crossref

51 Xiaoxuan Zhang, Jing Ran, Jize Mi. "An Intrusion Detection System Based on Convolutional Neural Network for Imbalanced Network Traffic", 2019 IEEE 7th International Conference on Computer Science and Network Technology (ICCSNT), 2019 — 6 words — < 1%
Crossref

52 Bengawan Alfaresi, Zainuddin Nawawi, Bhakti Yudho Suprapto. "Prediction of Path Loss Propagation in Case of Light Rail Transit (LRT) using Machine Learning Approaches", 2021 International Conference on Converging Technology in Electrical and Information Engineering (ICCTEIE), 2021 — 5 words — < 1%
Crossref

| | | | |
|---|---|---|---|
| EXCLUDE QUOTES | ON | EXCLUDE SOURCES | OFF |
| EXCLUDE BIBLIOGRAPHY | ON | EXCLUDE MATCHES | OFF |