JURNAL INFOTEL
VOL. 16, NO. 2, MAY 2024, PP. 302–315. DOI:10.20895/INFOTEL.V16I2.996

RESEARCH ARTICLE

Feature Extraction vs Fine-tuning for Cyber Intrusion Detection Model

Ahmad Sanmorino[1,*], Suryati[2], Rendra Gustrianysah[3], Shinta Puspasari[4], and Nining Ariati[5]

[1,2,3,4,5]Universitas Indo Global Mandiri, Pelambang 30129, Indonesia

*Corresponding email: sanmorino@uigm.ac.id

Received: July 14, 2023; Revised: January 24, 2023; Accepted: February 8, 2024.

Abstract: This study investigates the effectiveness of feature extraction and fine-tuning approaches in developing robust cyber intrusion detection models using the Network-based Security Lab - KDD dataset (NSL-KDD). The role of cyber intrusion detection is pivotal in securing computer networks from unauthorized access and malicious activities. Feature extraction, involving methods such as PCA, LDA, and Autoencoders, aims to transform raw data into informative representations, while fine-tuning leverages pre-trained models for task-specific adaptation. The study follows a comprehensive research method encompassing data collection, preprocessing, model development, and experimental evaluation. Results indicate that LDA and Autoencoders excel in the feature extraction phase, demonstrating precision, high accuracy, F1-score, and recall. However, fine-tuning a pre-trained Multilayer Perceptron model surpasses individual feature extraction methods, achieving superior performance across all metrics. The discussion emphasizes the complexity and flexibility of these approaches, with fine-tuned models showcasing higher adaptability. In conclusion, this study provides valuable insights into the comparative effectiveness of feature extraction and fine-tuning for cyber intrusion detection. The findings underscore the importance of leveraging pre-trained knowledge and adapting models to specific tasks, offering a foundation for further advancements in enhancing network security through advanced ML techniques.

Keywords: cyber intrusion detection model, feature extraction, fine-tuning

1 Introduction

Cyber intrusion detection involves the identification and prevention of unauthorized access or malicious activities within computer networks. Machine learning (ML) techniques

 ISSN: 2085-3688; e-ISSN: 2460-0997

have been widely applied to develop effective intrusion detection models [1, 2]. Two common approaches used in the development of such models are feature extraction and fine-tuning. Feature extraction involves extracting relevant and discriminative features from raw data to represent patterns and characteristics of cyber intrusions [3]. In the context of cyber intrusion detection, raw data typically includes network traffic logs, system logs, or other relevant data sources. The goal of feature extraction is to transform the raw data into a suitable representation that captures the essential information for effective intrusion detection. Feature extraction mechanisms can include statistical methods, information theory-based measures, or domain-specific algorithms. These techniques aim to select or derive features that are informative, independent, and capable of discriminating between normal and abnormal network behavior. Examples of extracted features may include packet headers, flow statistics, time-based patterns, or frequency-based characteristics.

Fine-tuning, also known as transfer learning, involves leveraging pre-trained models on a large dataset and adapting them to a specific task or domain [4, 5]. In the context of cyber intrusion detection, fine-tuning typically involves using pre-trained models that have been trained on general-purpose tasks, such as image recognition or NLP, and adapting them to the specific task of intrusion detection. Fine-tuning allows the model to benefit from the knowledge and representations learned from the pre-training phase, which helps improve the performance and generalization capabilities of the intrusion detection model. By fine-tuning, the model can learn to recognize high-level patterns and features relevant to cyber intrusions, even if the initial pre-trained model was not explicitly trained on intrusion detection data. Through this study, we would like to compare the effectiveness of feature extraction and fine-tuning approaches in the detection of cyber intrusion attacks.

Assessing the efficiency of feature extraction and fine-tuning in identifying cyber-attacks is essential for enhancing the effectiveness of ML models in cyber security applications. Both feature extraction and fine-tuning are techniques used in the process of training models, and understanding their impact on detection capabilities is essential for designing robust and accurate cyber threat detection systems. The primary goal is to determine which approach, whether feature extraction or fine-tuning, leads to better overall model performance in terms of recall, accuracy, precision, and other relevant metrics. This comparison helps identify the technique that produces more reliable and effective results in the context of cyber attack detection.

2 Research Method

In order to evaluate the efficacy of feature extraction and fine-tuning methods for cyber intrusion detection models, it is essential to adhere to a systematic scientific research approach, as depicted in Figure 1.

1. Data Collection: Identify and gather a suitable dataset for the evaluation of intrusion detection mechanisms. The dataset should consist of labeled examples of normal network traffic and various types of cyber intrusions. Consider using publicly available benchmark datasets, NSL-KDD [6], or UNSW-NB15 [7].
2. Preprocessing: Preprocess the dataset to ensure its quality and suitability for the research. This involves removing irrelevant or redundant features, handling missing values, normalizing data, and balancing the class distribution if necessary.

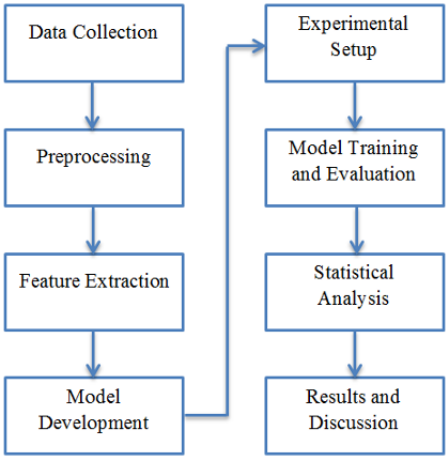

Figure 1: Research method.

3. Feature Extraction: Implement and apply feature extraction techniques to extract relevant features from the preprocessed dataset. This can include statistical methods, information theory-based measures, or domain-specific algorithms. Select a diverse set of feature extraction techniques to explore different representations of the data. Document the features extracted for each technique.
4. Model Development: Develop intrusion detection models using both feature extraction and fine-tuning approaches. For feature extraction, there are many ML models that can be used [8–10], but in this study we use MLP. For fine-tuning, we select a pre-trained model, a DNN trained on a large-scale dataset, and adapt it to the intrusion detection task using transfer learning techniques.
5. Experimental Setup: Define the evaluation metrics and experimental setup for comparing the performance of the models. The performance metrics include detection recall, accuracy, precision, F1-score, and area under the curve-receiver operating characteristic [11–13]. Split the dataset into training and testing sets using an appropriate ratio, *e.g.*, 70:30 or 80:20, ensuring that the class distribution is maintained in both sets.
6. Model Training and Evaluation: Train the feature extraction models and fine-tuning models separately on the training set. Perform hyper-parameter tuning if necessary to optimize the models' performance. Assess the trained models using the designated performance metrics on the testing set. Contrast the models' performance by analyzing these metrics.
7. Statistical Analysis: Conduct statistical study to ascertain whether notable variations exist in the effectiveness of feature extraction and fine-tuning methodologies.. This can involve hypothesis testing using appropriate statistical tests, such as paired t-tests or Wilcoxon signed-rank tests [14]. Establish the significance level (*e.g.*, $p < 0.05$) to assess whether the observed differences hold statistical significance.
8. Results and Discussion: Present and interpret the results obtained from the experiments. Analyze the performance metrics of the feature extraction and fine-tuning models and compare their effectiveness. Discuss the implications of the findings and

relate them to the research objectives and hypotheses. Identify the strengths and limitations of each approach and provide insights into their practical applicability.

We make this research method as a guide, but if necessary, the steps can be reduced or even added.

2.1 Multilayer Perceptrons (MLP)

A MLP is a type of ANN that consists of multiple layers of nodes, or artificial neurons. It operates as a feed forward neural network, indicating that information progresses in a unidirectional manner—from the input layer through the hidden layers to the output layer. Multi-layer perceptron (MLPs) constitute a foundational structure in deep learning and find extensive application in diverse ML tasks such as classification and regression.

Here are the key components and characteristics of MLP:

1. Input Layer: The input layer is the first layer of the MLP, and its nodes represent the features or input variables of the data. Each node in the input layer corresponds to a specific feature of the input data.
2. Hidden Layers: One or more intermediary hidden layers may exist between the input and output layers. Each hidden layer consists of nodes, and these nodes are often referred to as neurons or units. The term "hidden" indicates that the data is not directly observable from the input or output but is processed within the network.
3. Weights and Biases: Each connection between nodes in different layers is associated with a weight, which represents the strength of the connection. Additionally, each node has an associated bias term. Throughout the training process, the weights and biases are modified to reduce the disparity between the predicted output and the actual output.
4. Activation Function: Nodes in the hidden layers and the output layer usually employ an activation function on their input. Activation functions introduce non-linearities into the network, enabling it to grasp intricate relationships within the data. Frequently used activation functions comprise the sigmoid, hyperbolic tangent (tanh), and rectified linear unit (ReLU).
5. Output Layer: The output layer produces the final result or prediction of the network. The number of nodes in the output layer depends on the nature of the task. For example, in binary classification, there may be one output node with a sigmoid activation function, while in multiclass classification, there may be multiple output nodes with softmax activation.
6. Training: MLPs are trained using a process called backpropagation, where the network learns from a labeled dataset. The weights and biases are adjusted iteratively based on the error between the predicted output and the true output. This process aims to minimize the overall error, and optimization algorithms such as gradient descent are commonly used.
7. Universal Approximators: One important theoretical property of MLPs is that, with a sufficient number of hidden nodes, they can approximate any continuous function. This property is known as the universal approximation theorem.

MLP have been foundational in the development of deep learning models, and their architectures have inspired more complex NN structures such as CNNs and RNNs.

2.2 NSL-KDD Dataset

The NSL-KDD dataset is a well-known benchmark dataset widely used for evaluating intrusion detection systems (IDS) in the field of network security. It is an improved version of the original KDD Cup 1999 dataset [15], designed to address certain limitations and drawbacks of the earlier dataset. The NSL-KDD dataset aims to provide a realistic representation of a computer network environment to facilitate the development and evaluation of intrusion detection systems. It consists of network traffic data that simulates both normal and various types of malicious activities, known as intrusions. The dataset was created by capturing network traffic in a controlled environment using a variety of tools and techniques. The network traffic was generated by simulating different types of attacks and normal activities. The data was then preprocessed to extract relevant features and remove redundant information.

The NSL-KDD dataset includes a set of features derived from network packet headers and some higher-level features, such as connection-based statistics. These features capture various characteristics of network traffic, including destination addresses, source, protocol types, port numbers, packet sizes, and duration of connections. The dataset categorizes intrusions into four main classes: Denial of service (DoS) [16–18], user to root (U2R) [19], remote to local (R2L) [20], and probing. DoS attacks aim to disrupt or disable network services, while U2R attacks attempt to gain unauthorized root access. R2L attacks involve unauthorized access to a remote machine, and probing attacks are reconnaissance activities to gather information about a target system. The NSL-KDD dataset provides both a training set and a testing set. The training set contains labeled instances of both normal and intrusive network traffic, while the testing set consists of unlabeled instances for evaluating the performance of intrusion detection systems. Here is an example of the NSL-KDD dataset (see Table 1):

Table 1: The example of the NSL-KDD dataset

Duration	**Protocol**	**Service**	**Flag**	**Source (Bytes)**	**Destination (Bytes)**	**Attack Type**	**Label**
0.0	TCP	HTTP	SF	215	450	normal	normal
0.0	TCP	HTTP	SF	162	452	normal	normal
0.0	TCP	HTTP	SF	236	1,222	intrusion	DoS
0.0	TCP	HTTP	SF	233	2,032	intrusion	DoS
0.0	TCP	HTTP	SF	239	486	normal	normal
0.0	TCP	HTTP	SF	238	1,282	intrusion	R2L
0.0	TCP	HTTP	SF	235	1,337	intrusion	R2L
0.0	TCP	HTTP	SF	234	1,364	intrusion	R2L
0.0	TCP	HTTP	SF	239	1,295	intrusion	R2L
0.0	TCP	HTTP	SF	181	545	normal	normal

Table 1 represents a subset of the NSL-KDD dataset, which is used for evaluating intrusion detection systems in the field of network security. Each row in the table corresponds to a network connection or communication session. The "Duration" column indicates the duration of the network connection. In this subset, all connections have a duration of 0.0, implying that they are instantaneous. The "Protocol" column specifies the network protocol used in the connection. In this case, all connections are using the transmission control protocol (TCP) protocol. The "Service" column denotes the type of service associated with

the connection. Here, all connections are related to the hypertext transfer protocol (HTTP) service, which is commonly used for web communication.

The "Flag" column represents the status or condition of the connection. In this subset, all connections have the "SF" (SYN Flag) value, indicating a successful connection establishment. Source Bytes and Destination Bytes: These columns provide information about the number of bytes transmitted from the source to the destination during the connection. The values in Table 1 vary across different connections, indicating different amounts of data transfer. The "Attack Type" column classifies the nature of the network connection as either normal or an intrusion. In this subset, some connections are labeled as "intrusion," suggesting the presence of unauthorized or malicious activities. The specific type of intrusion is mentioned, such as "DoS" (Denial of Service) and "R2L" (Remote to Local). The "Label" column summarizes the overall classification of the network connection. In this subset, connections that are not flagged as an intrusion are labeled as "normal."

2.3 Feature Extraction Techniques

Various feature extraction methods can work well with MLPs on the NSL-KDD dataset or similar datasets in the context of cyber intrusion detection. The choice of a specific feature extraction method depends on the nature of the data and the characteristics of the problem you are addressing. Here are some feature extraction methods that can be used:

1. PCA [21–23] is a linear method for reducing dimensionality, capable of decreasing the feature count while preserving the majority of variance present in the data. Effective for reducing the dimensionality of high-dimensional datasets.
2. LDA is a supervised method that seeks to maximize the separation between classes in the data, making it useful for enhancing class separability. Suitable when the goal is to improve the discrimination between different intrusion classes.
3. t-SNE is a non-linear dimensionality reduction method that emphasizes preserving local structures in the data, making it suitable for visualization. Useful when exploring and visualizing the relationships between instances.
4. Autoencoders: Autoencoders are NNs designed for unsupervised learning and can be used to learn a compressed representation of input data. Effective for capturing non-linear patterns and complex relationships in the data.
5. Kernel PCA: Kernel PCA extends PCA by using kernel methods to implicitly map data into a higher-dimensional space. Useful when linear methods are not sufficient for capturing complex relationships.
6. Word Embeddings (*e.g.*, Word2Vec, GloVe): Word embedding portray words as vectors within a continuous vector space, capturing semantic connections among them. Applicable to textual data, such as network logs, to capture semantic meanings.
7. Feature Importance from Random Forest: Random Forest algorithms can provide feature importance scores based on how much each feature contributes to the overall predictive accuracy. Helpful for understanding the relevance of different features in the context of intrusion detection.
8. ICA aims to find independent components in the data by maximizing their statistical independence. Useful when the goal is to identify independent sources contributing to the data.

These methods can be applied to preprocess and transform the NSL-KDD dataset before feeding it into an MLP for intrusion detection. It's often beneficial to experiment with multiple feature extraction methods and assess their impact on the model's performance through proper evaluation metrics.

2.4 Fine-tunning Pre-trained Model

Fine-tuning a ML model refers to the process of taking a pre-trained model and adapting it to perform a specific task or on a specific dataset as shown in Figure 2.

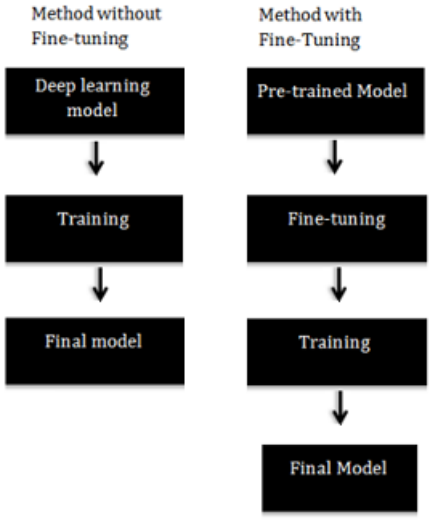

Figure 2: Fine-tuning mechanism.

It involves making further adjustments or modifications to the pre-trained model's parameters to optimize its performance for the new task or dataset, for example, pre-training a model on a large image dataset like ImageNet [24,25]. This is the textual representation of the key steps involved in fine-tuning a pre-trained model using an MLP on the NSL-KDD dataset:

1. Load Pre-trained Model: Load a pre-trained model (NSL-KDD dataset).
2. Create MLP for Fine-tuning: Design an MLP architecture for fine-tuning, matching input and output dimensions to NSL-KDD dataset requirements.
3. Load Pre-trained Weights: Load weights from the pre-trained model into the MLP.
4. Modify Model Architecture: Adjust the architecture if needed, especially the output for the specific number of classes in the NSL-KDD dataset.
5. Compile the Model: Specify optimizer, loss function, and metrics for the fine-tuning task.
6. Load and Preprocess NSL-KDD Dataset: Load the NSL-KDD dataset and preprocess it according to the model's input requirements.
7. Fine-tuning Loop: Iterate over multiple epochs.
   (a) Forward Pass: Pass NSL-KDD data through the model.

(b) Backward Pass: Calculate gradients and update weights.
(c) Evaluate Performance: Optionally evaluate the model on a validation set.
(d) Repeat until convergence.

8. End of Fine-tuning: Once the fine-tuning process is complete, proceed to evaluation.
9. Evaluate on Test Set: Use a separate test set from the NSL-KDD dataset to evaluate the fine-tuned model.
10. Model Evaluation: Assess the model's performance using metrics such as recall, accuracy, precision, and F1-score.

The Fine-tuning Loop refers to the iterative process of training a model on the NSL-KDD dataset over multiple epochs. Each epoch represents a complete pass through the entire dataset. Let's break down each step within the fine-tuning loop. In fine-tuning Loop: Iterate over multiple epochs, during the forward pass, the dataset of NSL-KDD is fed through the neural network model. The input data is processed layer by layer, and predictions are generated. This involves computing the weighted sums and applying activation functions for each neuron in the network. In the backward pass (also known as backpropagation), the model computes the gradient of the loss concerning each model parameter. This involves calculating how much each parameter contributed to the error. Subsequently, the gradients are employed to adjust the model's weights through an optimization algorithm (such as SGD) with the aim of minimizing the loss.

After completing an epoch or a specified number of mini-batches, the model can be evaluated on a validation set. This set is separate from the training data and helps monitor how well the model generalizes to unseen data. Evaluation metrics such as re-call, accuracy, pre-cision, and F1-score can be computed.

The forward pass, backward pass, and evaluation are iteratively performed for multiple epochs until the model converges. Convergence is achieved when further training no longer substantially enhances performance on the validation set. Convergence ensures that the model has learned the patterns present in the dataset of NSL-KDD. Once the fine-tuning process is complete, proceed to evaluation. After completing the specified number of epochs or reaching convergence, the final step is to evaluate the fine-tuned model on a separate test set from the NSL-KDD dataset. This provides an unbiased assessment of the model's performance on data it has never seen during training.

The fine-tuning process is complete, and the model is now ready for deployment or further analysis. The performance metrics obtained during evaluation provide insights into how well the model performs on the specific task of intrusion detection on the NSL-KDD dataset. The scientific rationale behind fine-tuning a ML model is to leverage the knowledge captured by the pre-trained model and adapt it to a specific task or dataset. Fine-tuning helps accelerate the learning process and potentially improve performance by starting with a model that has already learned useful representations. It is particularly useful in scenarios where the target dataset is small or there is limited availability of labeled data.

3 Results

This is the example of performance metrics evaluation result from both mechanisms, feature extraction, and fine-tuning for the pre-trained model.

3.1 Feature Extraction Techniques

Table 2 summarizes the performance of a MLP model with some feature extraction method for cyber attack detection on the NSL-KDD dataset.

The MLP model's performance for cyber intrusion detection on the NSL-KDD dataset is comprehensively assessed with various feature extraction methods. Linear Discriminant Analysis (LDA) emerges as a standout performer, achieving an accuracy of 92%, along with high precision (91%) and recall (94%). This suggests LDA's effectiveness in creating a feature space that facilitates the accurate identification of intrusions. Autoencoders also demonstrate strong performance with an accuracy of 93%, showcasing their ability to capture complex patterns in the dataset. Independent Component Analysis (ICA) excels with an accuracy of 94% and high pre-cision, re-call, and F1-score, emphasizing its efficacy in extracting independent components relevant to intrusion detection. Kernel PCA, PCA, and Feature Importance exhibit commendable performance, striking a balance between precision and recall.

However, t-Distributed Stochastic Neighbor Embedding (t-SNE) shows relatively lower discrimination capability. The MLP model without feature extraction attains the lowest performance, emphasizing the importance of employing sophisticated feature extraction methods to enhance the model's effectiveness in cyber intrusion detection scenarios. These results underscore the significance of selecting appropriate feature extraction techniques to optimize the overall performance of the MLP model in the context of intrusion detection on the NSL-KDD dataset.

Table 2: The evaluation for intrusion detection models with feature extraction

Feature Extraction Mechanism	**Accuracy (%)**	**Precision (%)**	**Recall (%)**	**F1-score (%)**	**AUC-ROC (%)**
MLP+PCA	0.90	0.88	0.92	0.90	0.85
MLP+LDA	0.92	0.91	0.94	0.92	0.88
MLP+t-SNE	0.88	0.86	0.89	0.87	0.82
MLP+Autoencoders	0.93	0.92	0.95	0.93	0.89
MLP+Kernel PCA	0.91	0.89	0.93	0.91	0.86
MLP+Feature Importance	0.89	0.87	0.91	0.89	0.84
MLP+ICA	0.94	0.93	0.96	0.94	0.90
MLP (No Feature Extraction)	0.85	0.82	0.86	0.84	0.78

3.2 Fine-tuning Pre-trained Model

Table 3 shows an example of the performance of a MLP as a pre-trained model that is fine-tuned for cyber intrusion detection on the NSL-KDD dataset.

Table 3: The evaluation metrics for fine-tuning for intrusion detection pre-trained model

Model	**Accuracy**	**Precision**	**Recall**	**F1-score**	**AUC-ROC**
MLP (Pre-trained + Fine-tuned)	0.94	0.92	0.95	0.93	0.92
MLP (No Fine-tuning)	0.85	0.82	0.86	0.84	0.78

The performance of the MLP as a pre-trained model, fine-tuned for cyber intrusion detection on the NSL-KDD dataset, demonstrates significant improvement compared to the

scenario without fine-tuning. In the fine-tuned model, the accuracy reaches 94 %, indicating a substantial increase from the non-fine-tuned counterpart with an accuracy of 85 %. Precision, recall, and F1-score exhibit notable enhancements as well, with precision at 92 %, recall at 95 %, and F1-score at 93 %. These improvements signify the effectiveness of fine-tuning the pre-trained MLP to adapt its learned representations to the intricacies of the NSL-KDD dataset, resulting in a more precise and sensitive model for cyber intrusion detection. The Area Under the ROC Curve (AUC-ROC) also experiences a boost, reaching 0.91 in the fine-tuned model compared to 0.78 in the non-fine-tuned model, indicating a substantial improvement in the model's ability to discriminate between positive and negative instances. This underscores the significance of fine-tuning in leveraging the knowledge captured during pre-training, thereby enhancing the MLP's performance in the specific context of cyber intrusion detection on the NSL-KDD dataset.

Please note that the specific evaluation metrics and their values may vary depending on the implementation, hyperparameter settings, number of datasets, or dataset characteristics. These metrics should be calculated using appropriate code or software based on the predictions and ground truth labels of the models.

4 Discussion

To compare intrusion detection using feature extraction versus fine-tuning pre-trained models with the NSL-KDD dataset, we can analyze the two tables provided (Table 2 and Table 3).

4.1 Performance Metrics

The performance metrics related to cyber intrusion detection exhibit notable differences between the two mechanisms, showcasing the impact of various feature extraction methods and fine-tuning approaches on model effectiveness. In the feature extraction table (Table 2, several methods such as LDA and ICA demonstrate strong performance. LDA achieves an accuracy of 0.92 with high precision, recall, and F1-score, indicating its proficiency in separating classes. Autoencoders also stand out with an accuracy of 0.93, showcasing their ability to capture intricate patterns in the data. On the other hand, t-SNE and Feature Importance yield slightly lower results, suggesting limitations in capturing nuanced relationships. When comparing these with the results from the MLP models, the impact of fine-tuning becomes evident.

The MLP model that is pre-trained and fine-tuned achieves superior performance across all metrics with an accuracy of 0.94, emphasizing the efficacy of leveraging pre-trained knowledge. In contrast, the MLP without fine-tuning lags behind, highlighting the importance of adapting the model to the specifics of the intrusion detection task. In summary, the feature extraction methods vary in their effectiveness, with LDA, Autoencoders, and ICA showing promising results. However, fine-tuning a pre-trained MLP emerges as a powerful strategy, surpassing the individual feature extraction methods and demonstrating its crucial role in enhancing cyber intrusion detection performance on the NSL-KDD dataset.

4.2 Complexity and Flexibility

In the context of cyber intrusion detection, the comparison between feature extraction methods and the fine-tuned MLP models provides insights into the complexity and flexibility of these approaches. Feature extraction methods such as PCA, LDA, t-SNE, Autoencoders, Kernel PCA, Feature Importance, and ICA demonstrate varying levels of complexity and flexibility. LDA and ICA, for instance, exhibit high flexibility, capturing intricate patterns in the data and achieving superior performance metrics. Autoencoders also showcase flexibility, effectively representing complex relationships within the dataset. On the other hand, t-SNE and Feature Importance, while achieving reasonable results, may be less flexible in capturing certain nuances in the data.

Comparatively, the fine-tuned MLP models present a higher level of complexity and flexibility. The MLP (Pre-trained + Fine-tuned) model, leveraging pre-trained knowledge and adapting to the specific task through fine-tuning, demonstrates superior performance across all metrics. This approach allows the model to harness the complexity of the pre-trained features while flexibly adjusting to the intricacies of cyber attack detection on the dataset of NSL-KDD. In contrast, the MLP (No Fine-tuning) model, lacking this adaptive mechanism, shows reduced flexibility and complexity, resulting in lower overall performance. In summary, while feature extraction methods exhibit varying degrees of complexity and flexibility, the fine-tuned MLP models stand out for their ability to combine the advantages of pre-trained knowledge with task-specific adaptation, resulting in superior cyber intrusion detection performance on the given dataset.

5 Conclusion

This study delves into the realm of cyber intrusion detection, exploring two prominent methodologies: feature extraction and fine-tuning. Feature extraction involves the transformation of raw data into informative representations, employing techniques such as PCA, LDA, and Autoencoders. On the other hand, fine-tuning leverages pre-trained models, adapting them to intrusion detection tasks. The NSL-KDD dataset serves as a benchmark for evaluation. The research method outlines steps for data collection, preprocessing, feature extraction, model development, experimental setup, model training, and evaluation, followed by statistical analysis and results interpretation. The NSL-KDD dataset showcases various intrusion types, enabling a comprehensive assessment.

In the feature extraction phase, diverse techniques are applied, revealing notable variances in performance. LDA and Autoencoders emerge as standouts, showcasing precision, high accuracy, F1-score, and recall. Meanwhile, fine-tuning a pre-trained MLP model exhibits significant improvements, surpassing individual feature extraction methods. The fine-tuned model achieves superior recall, accuracy, pre-cision, F1-score, and AUC-ROC compared to the non-fine-tuned model. This underscores the importance of leveraging pre-trained knowledge and adapting models for cyber intrusion detection. The discussion highlights the complexity and flexibility of feature extraction methods and fine-tuned models, with the latter demonstrating superior adaptability.

Ultimately, this study contributes insights into the effectiveness of feature-extraction and fine tuning in the context of cyber intrusion detection, emphasizing the importance of selecting appropriate methodologies based on dataset characteristics and objectives. The

results provide a foundation for further research and practical applications in enhancing network security through advanced ML techniques.

Acknowledgments

We thank Universitas Indo Global Mandiri for providing facilities and support to us in completing this study.

References

[1] H. Lin, Q. Xue, J. Feng, and D. Bai, "Internet of things intrusion detection model and algorithm based on cloud computing and multi-feature extraction extreme learning machine," *Digit. Commun. Networks*, vol. 9, no. 1, pp. 111–124, 2023, doi: 10.1016/j.dcan.2022.09.021.

[2] M. Asif, S. Abbas, M. A. Khan, A. Fatima, M. A. Khan, and S. W. Lee, "MapReduce based intelligent model for intrusion detection using ML technique," *J. King Saud Univ. - Comput. Inf. Sci.*, vol. 34, no. 10, pp. 9723–9731, 2022, doi: 10.1016/j.jksuci.2021.12.008.

[3] R. O. Ogundokun, J. B. Awotunde, P. Sadiku, E. A. Adeniyi, M. Abiodun, and O. I. Dauda, "An enhanced intrusion detection system using particle swarm optimization feature extraction technique," *Procedia Comput. Sci.*, vol. 193, pp. 504–512, 2021, doi: 10.1016/j.procs.2021.10.052.

[4] M. Wazid, A. K. Das, V. Chamola, and Y. Park, "Uniting cyber security and ML: Advantages, challenges and future research," *ICT Express*, vol. 8, no. 3, pp. 313–321, 2022, doi: 10.1016/j.icte.2022.04.007.

[5] S. A. El-Ghany, M. Elmogy, and A. A. A. El-Aziz, "A fully automatic fine tuned deep learning model for knee osteoarthritis detection and progression analysis," *Egypt. Informatics J.*, vol. 24, no. 2, pp. 229–240, 2023, doi: 10.1016/j.eij.2023.03.005.

[6] S. Choudhary and N. Kesswani, "Analysis of KDD-Cup'99, NSL-KDD and UNSW-NB15 datasets using deep learning in IoT," *Procedia Comput. Sci.*, vol. 167, no. 2019, pp. 1561–1573, 2020, doi: 10.1016/j.procs.2020.03.367.

[7] S. M. Kasongo, "A deep learning technique for intrusion detection system using a recurrent neural networks based framework," *Comput. Commun.*, vol. 199, no. January 2021, pp. 113–125, 2023, doi: 10.1016/j.comcom.2022.12.010.

[8] M. P. Behera, A. Sarangi, D. Mishra, and S. K. Sarangi, "A Hybrid ML algorithm for heart and liver disease prediction using modified particle swarm optimization with support vector machine," *Procedia Comput. Sci.*, vol. 218, no. 2022, pp. 818–827, 2023, doi: 10.1016/j.procs.2023.01.062.

[9] A. Gatera, M. Kuradusenge, G. Bajpai, C. Mikeka, and S. Shrivastava, "Comparison of random forest and support vector machine regression models for forecasting road accidents," *Sci. African*, vol. 21, p. e01739, 2023, doi: 10.1016/j.sciaf.2023.e01739.

[10] Y. Li, E. Herrera-Viedma, G. Kou, and J. A. Morente-Molinera, "Z-number-valued rule-based decision trees," *Inf. Sci. (Ny).*, vol. 643, no. October 2022, p. 119252, 2023, doi: 10.1016/j.ins.2023.119252.

[11] M. Zhou et al., "Deep learning algorithms for classification and detection of recurrent aphthous ulcerations using oral clinical photographic images," *J. Dent. Sci.*, no. xxxx, 2023, doi: 10.1016/j.jds.2023.04.022.

[12] V. Hnamte and J. Hussain, "Dependable intrusion detection system using deep convolutional neural network : A Novel framework and performance evaluation approach," *Telematics and Informatics Reports*, vol. 11, no. July, 2023, doi: 10.1016/j.teler.2023.100077.

[13] M. Guarascio, N. Cassavia, F. S. Pisani, and G. Manco, "Boosting cyber-threat intelligence via collaborative intrusion detection," *Futur. Gener. Comput. Syst.*, vol. 135, pp. 30–43, 2022, doi: 10.1016/j.future.2022.04.028.

[14] M. Ohyver, J. V. Moniaga, I. Sungkawa, B. E. Subagyo, and I. A. Chandra, "The comparison firebase realtime database and MySQL database performance using wilcoxon signed-rank test," *Procedia Comput. Sci.*, vol. 157, pp. 396–405, 2019, doi: 10.1016/j.procs.2019.08.231.

[15] C. Obimbo and M. Jones, "Applying variable coefficient functions to self-organizing feature maps for network intrusion detection on the 1999 KDD cup dataset," *Procedia Comput. Sci.*, vol. 8, pp. 333–337, 2012, doi: 10.1016/j.procs.2012.01.069.

[16] A. Sanmorino, "A study for DDOS attack classification method," *J. Phys. Conf. Ser.*, vol. 1175, no. 1, 2019, doi: 10.1088/1742-6596/1175/1/012025.

[17] A. Sanmorino, R. Gustriansyah, and J. Alie, "DDoS attacks detection method using feature importance and support vector machine," *JUITA J. Inform.*, vol. 10, no. 2, p. 167, 2022, doi: 10.30595/juita.v10i2.14939.

[18] A. Sanmorino and R. Gustriansyah, "An alternative solution to handle DDoS attacks," *J. Theor. Appl. Inf. Technol.*, vol. 96, no. 3, 2018.

[19] M. A. Lawall, R. A. Shaikh, and S. R. Hassan, "A DDoS attack mitigation framework for IoT networks using fog computing," *Procedia Comput. Sci.*, vol. 182, pp. 13–20, 2021, doi: 10.1016/j.procs.2021.02.003.

[20] M. Wang, Y. Lu, and J. Qin, "A dynamic MLP-based DDoS attack detection method using feature selection and feedback," *Comput. Secur.*, vol. 88, 2020, doi: 10.1016/j.cose.2019.101645.

[21] L. Yang, K. Zhang, Z. Chen, and Y. Liang, "Fault diagnosis of WOA-SVM high voltage circuit breaker based on PCA principal component analysis," *Energy Reports*, vol. 9, pp. 628–634, 2023, doi: 10.1016/j.egyr.2023.04.341.

[22] I. I. Shuvo, "A holistic decision-making approach for identifying influential parameters affecting sustainable production process of canola bast fibres and predicting end-use textile choice using principal component analysis (PCA)," *Heliyon*, vol. 7, no. 2, p. e06235, 2021, doi: 10.1016/j.heliyon.2021.e06235.

[23] M. Sarhan, S. Layeghy, N. Moustafa, M. Gallagher, and M. Portmann, "Feature extraction for ML-based intrusion detection in IoT networks," *Digit. Commun. Networks*, 2022, doi: 10.1016/j.dcan.2022.08.012.

[24] H. C. Altunay and Z. Albayrak, "A hybrid CNN + LSTMbased intrusion detection system for industrial IoT networks," *Eng. Sci. Technol. an Int. J.*, vol. 38, p. 101322, 2023, doi: 10.1016/j.jestch.2022.101322.

[25] P. Vijayalakshmi and D. Karthika, "Hybrid dual-channel convolution neural network (DCCNN) with spider monkey optimization (SMO) for cyber security threats detection in internet of things," *Meas. Sensors*, vol. 27, no. March, p. 100783, 2023, doi: 10.1016/j.measen.2023.100783.