

# Improving Multilayer-Perceptron(MLP)-based Network Anomaly Detection with Birch Clustering on CICIDS-2017 Dataset

Yuhua Yin\*, Julian Jang-Jaccard\*, Fariza Sabrina<sup>†</sup> and Jin Kwak<sup>‡</sup>

\*Cybersecurity Lab, School of Mathematical and Computational Sciences, Massey University, New Zealand

<sup>†</sup>School of Engineering and Technology, Central Queensland University, AUSTRALIA

<sup>‡</sup>Department of Cyber Security, Ajou University, REPUBLIC OF KOREA

yuhua.yin.1@uni.massey.ac.nz, j.jang-jaccard@massey.ac.nz, f.sabrina@cqu.edu.au, security@ajou.ac.kr

**Abstract**—The network intrusion threats are increasingly severe with the application of computer supported cooperative work. Machine learning algorithms have been widely used in intrusion detection systems, including Multi-layer Perceptron (MLP). In this study, we proposed a two-stage model that combines the Birch clustering algorithm and MLP classifier to improve the performance of network anomaly multi-classification. In our proposed method, we first apply Birch or Kmeans as an unsupervised clustering algorithm to the CICIDS-2017 dataset to pre-group the data. The generated pseudo-label is then added as an additional feature to the training of the MLP-based classifier. The experimental results show that using Birch and K-Means clustering for data pre-grouping can improve intrusion detection system performance. Our method can achieve 99.73% accuracy in multi-classification using Birch clustering, which is better than similar researches using a stand-alone MLP model.

**Keywords**—Intrusion Detection, Anomaly Detection, CICIDS-2017 dataset, Multilayer Perceptron, Multi-classification, Clustering Algorithm

## I. INTRODUCTION

Organizations and individuals face increasing threats of cyber-attacks while enjoying the convenience of computer supported cooperative work. According to a report by Check Point Research, cyber-attacks in 2021 have increased by 50%, and each organization would experience approximately 900 attacks per week [1]. In addition, the types of network intrusions have become more diverse, and more attacks come from zero-day attacks [2]. Network intrusion detection systems are designed to identify network attacks by analyzing normal and malicious behavior of network traffic [3]. Traditional signature-based intrusion detection systems have been unable to deal with complex and diverse modern network attacks, so increasing research has focused on anomaly detection-based methods. Machine learning models have been widely used in anomaly-based intrusion detection systems and have good detection performance [3]. Supervised learning and unsupervised learning are two common machine learning methodologies. Models based on supervised learning, such as MLP, decision

tree, random forest, SVM, etc., can learn and classify labeled data [4]. Models based on unsupervised learning, including Kmeans, Birch, hidden Markov models, etc., can cluster similar unlabeled data to find anomalies [5]. Furthermore, datasets used for intrusion detection are essential and can be divided into packet-based and flow-based data [6]. The packet-based dataset mainly collects some meta-information such as protocol and size in the packet. In contrast, the flow-based dataset treats the packets that share attributes within a specific time window as a flow. Many modern intrusion detection datasets are flow-based, such as CICIDS-2017, CICIDS-2018, etc. [6]. The timeliness of intrusion detection datasets also affects their effectiveness. Early datasets such as NSL-KDD only contain some simple network attack classes [7]. With the continuous emergence of network attack types and variants, newer datasets such as CICIDS-2017 can better reflect modern and complex network anomalies [8].

This paper proposed a network anomaly detection method that combines both supervised and unsupervised learning models. In our method, the IDS data is pre-labelled using the Birch clustering algorithm, and then the generated pseudo-label is used as an extra feature in the training of the MLP classifier. The CICIDS-2017 as a modern IDS dataset was used to validate our method.

The contributions of our research are as follows:

- We proposed an approach that can improve MLP-based intrusion detection systems by using an unsupervised clustering algorithm for pre-labeling, which is believed to help the classifier distinguish between similar samples.
- During data pre-processing of CICIDS-2017, we removed duplicate data, regrouped the labels based on recent research, and resampled benign traffic to make benign samples and attack samples equal. Also, we applied information gain as a feature selection method to remove 18 unimportant features.
- Our experimental results demonstrated that our model can achieve a multi-class accuracy of 99.73%, which

is better than using MLP only. After comparison, our method outperformed similar research.

We organized the rest of the paper as follows. In Section II, we reviewed related works on machine learning-based intrusion detection systems. In Section III we described the background of our proposed method. In Section IV, we introduced our proposed method. In Section V, we described our experiment process and demonstrated the results. Finally, in Section VI, we concluded our work and discussed our future works.

## II. RELATED WORK

There have been many studies using different machine learning methods and IDS datasets for network anomaly detection. In this section, we reviewed some related work.

Wen et al. proposed an autoencoder-based network anomaly detection method in the NSL-KDD dataset [7]. The researchers used the percentile method to remove outliers before applying the autoencoder, and then implemented a five-layer autoencoder model for classification. The proposed autoencoder model can achieve a state-of-art accuracy of 90.61% on the multi-classification task for NSL-KDD.

Kasongo and Sun proposed a feature selection method using XGBoost as an intrusion detection dataset and applied it to the UNSW-NB15 dataset [9]. In the experiments, multiple machine learning methods including SVM, KNN, LR, ANN and DT are used to verify the performance of the intrusion detection system. The results show that the feature selection method of XGBoost enables the decision tree to obtain the highest binary classification performance up to 90% in UNSW-NB15.

Jing and Chen proposed a network intrusion detection method based on SVM [10]. A new data scaling method based on the log function is used, which is different from the traditional min-max scaling. Experimental results show that the proposed method can achieve 85.99% binary classification accuracy on UNSW-NB15.

Khan et al. proposed a novel two-stage network anomaly detection method [11]. Their method is divided into two steps, first using an autoencoder to perform binary classification on the data and then adding the obtained binary classification result as an extra feature to the multi-classification in the second step. The method achieves 99.99% and 89.13% multi-class accuracy on KDD99 and UNSW-NB15, respectively.

Rosay et al's study used MLP as a classifier to validate their intrusion detection method on CICIDS-2017 and CICIDS-2018 [12]. The proposed MLP model consists of two hidden layers, each containing 256 neurons. The authors removed constant-valued features from the datasets. Compared with the studies that also used MLP as a classifier, the experiments achieved better multi-classification accuracy of 99.46% and 95.47% on CICIDS-2017 and CICIDS-2018.

Ho et al. explored resampling methods for intrusion detection systems using CICIDS-2017 [13]. Random undersam-

TABLE I  
RECORDS OF ORIGINAL CICIDS-2017 DATASET

Label	Instances
BENIGN	2273097
DoS Hulk	231073
PortScan	158930
DDoS	128027
DoS GoldenEye	10293
FTP-Patator	7938
SSH-Patator	5897
DoS slowloris	5796
DoS Slowhttptest	5499
Bot	1966
Web Attack - Brute Force	1507
Web Attack - XSS	652
Infiltration	36
Web Attack - Sql Injection	21
Heartbleed	11
<b>Total</b>	<b>2830743</b>

pling, SMOTE, and their combination were used with the C4.5 classifier separately in the experiment. The results show that the best performance is obtained using random undersampling.

In Ustebay et al.'s study, recursive feature elimination(RFE) was used as a feature selection method for intrusion detection [14]. On the CICIDS-2017 dataset, the researchers used RFE to obtain the ten most important features, which were then used in the Deep Multilayer Perceptron(DMLP) classifier. The experiment achieved 91% accuracy.

Kong et al. proposed an intrusion detection system using a hybrid deep learning approach combining 1D CNN and LSTM [15]. In the proposed method, 1D CNN is first used to extract spatial features. Then the connected LSTM can extract temporal features, so that the model can extract both temporal and spatial features simultaneously. The results of the experiments can achieve higher than 97% accuracy on CICIDS-2017.

## III. PRELIMINARIES

### A. CICIDS2017 Dataset

CICIDS-2017 is a flow-based intrusion detection dataset created by the Canadian Institute for Cyber Security (CIC) to solve the problems of previous datasets [16]. The dataset recorded network traffic for five days and contained 2,830,743 samples consisting of 15 classes and 78 features. Among 15 classes, there were benign instances and 14 types of attack instances. The number of each class is shown in Table I. The 78 features are all integer or float numeric features.

### B. Birch Clustering Algorithm

Birch (balanced iterative reducing and clustering using hierarchies) is an efficient unsupervised hierarchical clustering algorithm proposed by Zhang et al. [17]. Birch is suitable for large datasets and only needs to read the data once to cluster the data. Birch constructs a CF tree data structure and uses the Clustering Feature at each leaf node to represent a sub-cluster by compressing instances and features. Each

Clustering Feature consists of a vector with three elements, as shown in equation 1. Birch clustering algorithm usually has three important parameters: *threshold*, *branching\_factor*, and *n\_clusters*. *Threshold* determines the radius threshold of the largest sample in a single leaf node. *branching\_factor* determines the largest number of samples in a node, and *n\_clusters* represents the final number of clusters to decide the aggregation of nodes.

$$CF = (N, LS, SS) \quad (1)$$

where N represents the number of data points, LS represents the linear sum of N data points, and SS represents the square sum of N data points.

### C. K-Means Clustering Algorithm

K-means is a centroid-based unsupervised clustering algorithm that is efficient and easy to implement. In the K-means algorithm, k centroids are randomly initialized, and then the samples are grouped according to the minimum euclidean distance between the data points and the centroids. After that, new centroids are calculated based on the mean of the data points in each cluster. This process runs iteratively until each cluster is converged and no longer changes.

---

#### Algorithm 1: K-means algorithm

---

**Input:**

D: data points  $\{d_1, d_2, \dots, d_m\}$

K: centroids  $\{k_1, k_2, \dots, k_n\}$

**Output:**

C: clustering result  $\{c_1, c_2, \dots, c_m\}$

**begin**

    initialize centroids for  $\{k_1, k_2, \dots, k_n\}$

**repeat**

        assign clusters for  $\{c_1, c_2, \dots, c_n\}$  by  
         calculating the minimum euclidean distance  
         between data points and each centroid.  
         calculate new centroids by cluster.

**until** centroids do not change

**end**

---

### D. Multi-layer Perceptron Classifier

As seen in Figure 1, Multi-layer Perceptron (MLP) is a feed-forward artificial neural networks composed of multiple layers with activation functions. An MLP typically consists of at least three layers including an input layer, a hidden layer, and an output layer. All layers are fully connected and use a supervised backpropagation for training. When an MLP is used for a classification task, it creates the same number of neurons in the input layer according to the number of features while the number of neurons in the last output layer is decided by the number of classes to be classified. The output is calculated

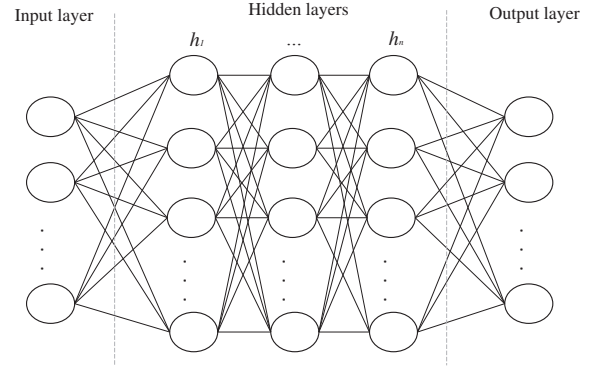


Fig. 1. Basic MLP Model

by computing weighted inputs and a bias associated with the layer (shown in 2).

$$Z^{[l]} = W^{[l]}A^{[l-1]} + b^{[l]} \quad (2)$$

where  $W^{[l]}$  indicates the weight inputs,  $b^{[l]}$  depicts a bias, and  $Z^{[l]}$  is the output.

An activation function is used at each layer to normalize the output in a certain range (i.e., typically between -1 and 1 and its variations) to improve computation efficiency (shown in equation 3).

$$A^{[l]} = g(Z^{[l]}) \quad (3)$$

where  $Z^{[l]}$  is the output,  $g$  represents an activation function, and  $A^{[l]}$  is the activated result.

The error between the value predicted by the model and the original value is calculated by a loss function as shown in equation 4. The result of the loss function at each interaction is used by the supervised backpropagation mechanism to update the weights associated with each input in the layer and a bias.

$$L(y, \hat{y}) = \frac{1}{m} \sum_{i=1}^m (y_i - \hat{y}_i)^2 \quad (4)$$

where m indicates the total number of input samples,  $\hat{y}$  represents the predicted value by the model, and y is the original value.

### E. Information Gain

We used Information Gain(IG) as a filter-based feature selection technique to select the most relevant subset of available features in a dataset to reduce the noise from the data and improve the classification accuracy.

The IG utilizes information entropy to measure the disorder of the system. A system with high entropy is considered unpredictable and more disordered while low entropy is associated with highly predictable and less disorderly. In machine

learning training, information entropy is used to measure the level of predictability of data distribution. For example, low entropy means many similar values in data distribution while high entropy indicates a lot of different values.

Mathematically, entropy is defined as follows:

$$H(Y) = - \sum_{i=1}^n p(y_i) \log_2 p(y_i) \quad (5)$$

where,  $n$  is the number of classes in the dataset  $Y$  and  $p(y_i)$  indicates the probability of picking an element  $y_i$  in each class of the dataset  $Y$ .

The average specific conditional entropy (i.e., another entropy value given that we already know the entropy of  $X$ ) can be defined as follows:

$$H(Y|X) = - \sum_{i=1}^m p(x_i) H(Y|X = x_i) \quad (6)$$

where,  $m$  is the number of classes in the dataset  $X$  and  $p(x_i)$  indicates the probability of picking an element  $x_i$  in each class of the dataset  $X$ .

In a simple term, IG can be seen as the amount of entropy removed. Mathematically, this can be defined as follows:

$$IG(Y, X) = H(Y) - H(Y|X) \quad (7)$$

In summary, the higher the IG, the more entropy is removed, and the more information the dataset  $Y$  carries about  $X$ . After calculating the IGs for all different features in the input, we rank them and choose the top  $n$  features to feed to the MLP model.

#### IV. PROPOSED METHOD

##### A. Overview

This section described the workflow of our proposed two-stage intrusion detection method. The CICIDS-2017 dataset provides 2.8 million instances of data with 78 numerical features, and they need to be pre-processed at first. In the data pre-processing step, we performed cleaning, label grouping, resampling, information gain feature selection, and normalization on the data (see Figure 2). We divided the pre-processed data into a training set, a validation set, and a test set according to the 80:10:10 proportion, which is unseen to each other. Our method mainly involved two machine learning modules, an unsupervised clustering model and an MLP deep learning classifier. Only training data is used to train the Birch or K-Means unsupervised clustering algorithm because the training data is assumed to be known. We used the elbow method when choosing the cluster number  $k$ , which was described in detail in the following subsection. After obtaining a clustering model, we generated cluster labels for the training, validation, and test set. This pseudo-label is added as an additional feature to the MLP model along with the original features. Finally, we will use the test set and the trained MLP model to verify the effectiveness of our method.

TABLE II  
HYPER-PARAMETER SETTING FOR OUR MLP MODEL

Parameter	Setting
neurons for hidden layer 1	256
neurons for hidden layer 2	256
output activation	softmax
optimizer	adam
learning rate	0.0001
batch size	64
epochs	200
early stopping patient	20

##### B. Elbow Method to Select Cluster Number $K$

The elbow method is a heuristic cluster number  $k$  selection method applied to unsupervised clustering algorithms with a cluster number parameter [18]. As the number of clusters increases, the mutual information within each cluster will be higher, but too many clusters can cause overfitting problems. The Elbow method selects the inflection points from the cluster purity line graph, which can help us choose a potentially suitable cluster number before overfitting. In our method, we used information gain to measure the clustering performance by calculating the entropy between the generated cluster label and the actual class label of the training set. After plotting the information gain metric under different cluster numbers, we can decide potential cluster numbers at the elbow points to test our model.

##### C. Specified MLP Classifier

We implemented an MLP model with two hidden layers as the classifier in our proposed method. As shown in Figure 3, each hidden layer contains 256 neurons and uses relu activation. Batch Normalization is added after each hidden layer for regularization, and it helps the model to avoid overfitting caused by gradient vanishing and gradient exploding. In the output layer, softmax activation is used to output a normalized probability vector for each class. The final classification result uses the argmax function to obtain the class with the highest probability. In addition, the Adam optimization algorithm is used in our MLP model to adjust the learning rate automatically. The early-stopping technique is used to prevent overfitting. If the validation loss do not continue to decrease in the specified patient's epochs, the training can be terminated early. The relevant hyper-parameter settings were listed in Table II.

#### V. EXPERIMENTS AND RESULTS

##### A. Hardware and Environment Setting

Our research was conducted on a desktop computer running Ubuntu 20.04.4 LTS. The Desktop computer is equipped with 16GB of RAM, a Ryzen 2700 processor, as well as an RX580 graphics card. TensorFlow 2.4.1 was utilized to develop the MLP model in our Python 3.8-based experimental environment. For our work, Scikit-Learn, Numpy, pandas,



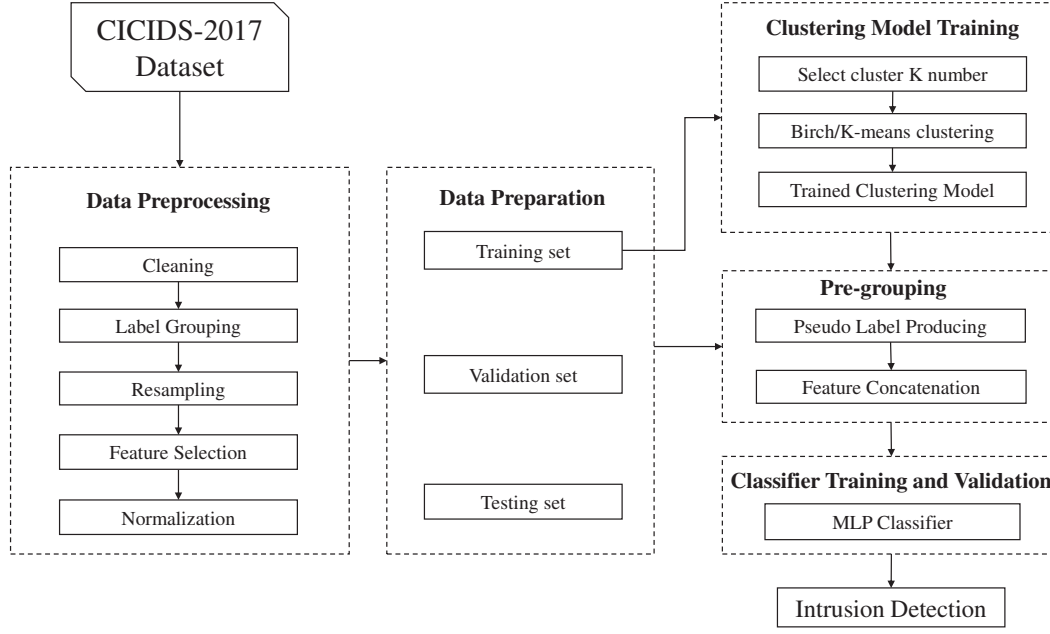


Fig. 2. Our Proposed Method

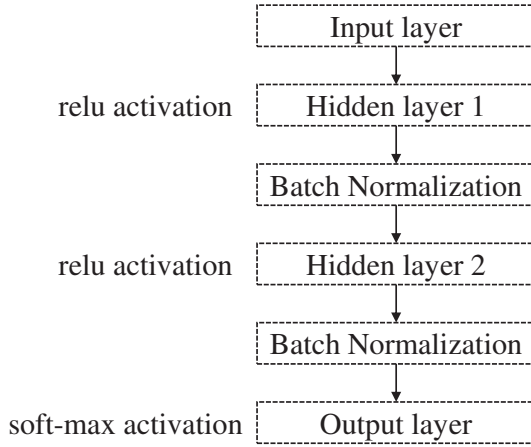


Fig. 3. Our Specified MLP Classifier

and matplotlib supplied pre-processing, feature selection, and visualization functions. Table III details hardware and environmental parameters.

#### B. Data Pre-processing

The procedure and methods we applied for data pre-processing were described in this subsection.

TABLE III  
HARDWARE AND ENVIRONMENT SPECIFICATION

Unit	Description
Processor	AMD Ryzen 7 2700
RAM	16 GB
GPU	AMD RX580
Operating System	Ubuntu 20.04.4 LTS
Packages	Tensorflow 2.4.1, Sklearn 1.0.2, Numpy, Pandas and Matplotlib

1) *Cleaning*: We eliminated 2867 rows with null values from the CICIDS-2017 dataset, as well as the Label column, which specified the class type.

2) *Label Grouping*: There are 15 labels for instances in the original CICIDS-2017 dataset, some of which can be grouped. To help reduce the output vector of classification problems, Panigrahi et al [19] and Kurniabudi et al [20] proposed a grouping method that can group 15 labels into 7 classes. We described the labels before and after grouping in Table IV.

3) *Resampling*: In the original CICIDS-2017 dataset, benign samples account for almost 80% of the total 2.8 million samples, which is imbalanced. We used the random under-sampling method which randomly remove majority instances on benign instances and made the benign and attack samples equal (See Figure 4).

4) *Feature Selection*: Figure 5 depicted the ranking results after we calculated the importance of each feature using Information Gain. The figure shows that the importance scores

TABLE IV  
GROUPING LABELS FOR CICIDS-2017

New Labels	Original Labels	Number
Benign	Bengin	2271320
Bot	Bot	1956
Brute Force	FTP-Patator, SSH-Patator	13832
DoS/DDoS	DDoS, DoS, GoldenEye, DoS Hulk, DoS Slow, httpstest, DoS slowloris, Heartbleed	379748
Infiltration	Infiltration	36
PortScan	PortScan	158804
Web Attack	Web Attack-Brute Force, Web Attack-Sql Injection, Web Attack-XSS	2180
<b>Total</b>		<b>2827876</b>

Benign and attack proportion in CICIDS-2017 dataset before resampling Benign and Attack proportion in CICIDS-2017 dataset after resampling

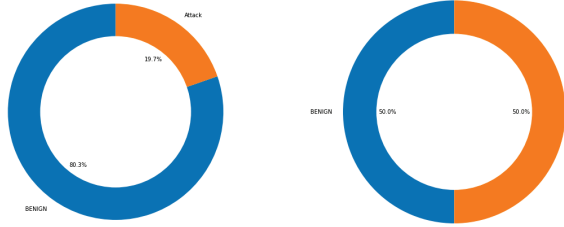


Fig. 4. Resampling benign samples in CICIDS-2017

of some features are very low, implying that these features are either irrelevant or redundant. We removed 18 features with information gain importance less than 0.1, leaving 60 features for our model.

5) *Normalization*: Normalization can unify the value range of each feature and remove bias produced by disparate value scales during MLP model training. To transform the range of feature values between 0 and 1, we utilized MinMax Normalization [21]. The new value is computed by dividing the difference between the min and max values by the scale size, as depicted in equation 8.

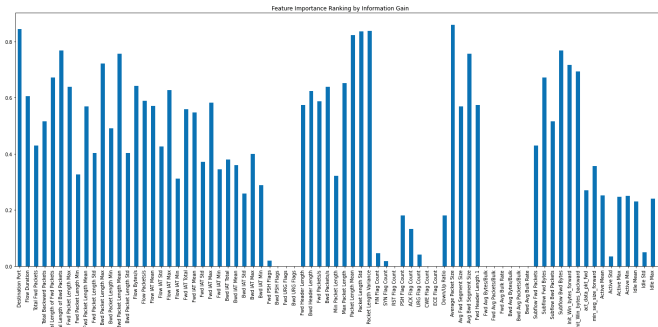


Fig. 5. Information Gain Feature Importance Ranking

TABLE V  
PREPARED TRAINING, VALIDATION AND TEST DATASET

Label	Training set	Validation set	Test set
Benign	445244	55656	55656
DoS/DDoS	303798	37975	37975
PortScan	127043	15881	15880
Brute Force	303798	1383	1383
Web Attack	1744	218	218
Bot	1565	196	195
Infiltration	29	3	4
<b>Total</b>	<b>890489</b>	<b>111312</b>	<b>111311</b>

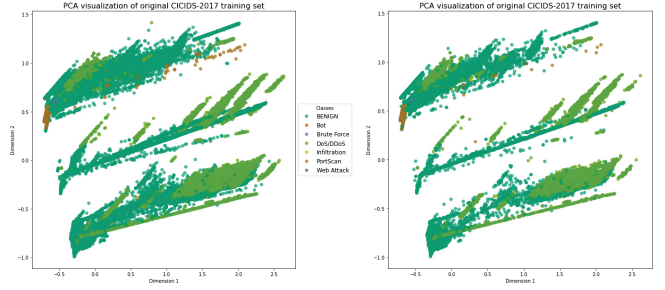


Fig. 6. The PCA visualization of training and test set for CICIDS-2017

$$x_i' = \frac{x_i - \min(x_i)}{\max(x_i) - \min(x_i)} \quad (8)$$

where  $x_i$  depicts all features,  $\min(x_i)$  is the minimum value among all the features, and  $\max(x_i)$  is the the maximum value among all the features.

6) *Training, validation and test data preparation*: In the ratio of 80:10:10, we set aside a training set, validation set, and test set from the resampled dataset for the CICIDS-2017 dataset. The number of occurrences is shown in Table V. Figure 6 also displays the hold-out training and test set shown using PCA.

In our experiments, the elbow method was used to select the cluster number  $k$  of the Birch and K-means clustering algorithm, and information gain was used as the metric of the cluster purity. A higher information gain metric means a better cluster purity. Default hyper-parameter  $Threshold=0.5$  and  $branching\_factor=50$  in sklearn were used for Birch algorithm. Figure 7 shows the line graph when 2-14 clusters are selected using the Elbow method, which can help find the elbow point. In K-means clustering, the potential elbow points are [3, 4, 6, 8]. In Birch clustering, [3, 9, 12] can be observed as elbow points. These cluster numbers will be used in our further experiments.

### C. Evaluation Metrics

We employed accuracy, recall (also known as true positive rate), precision, FPR( also known as false positive rate), f1 score, and Receiver operating characteristic(ROC) as performance indicators. We use True Positive(TP) to identify correctly classified positive samples, False Negative(FN) for

TABLE VI  
RESULTS OF DIFFERENT CLUSTER NUMBERS UNDER CLUSTERING ALGORITHMS

Method	K(Cluster) number	Precision(%)	Recall(%)	F1 score(%)	Accuracy(%)
MLP(78 features)	-	99.01	99.01	98.94	99.01
MLP(60 features)	-	99.45	99.45	99.44	99.45
K-means+MLP	3	99.68	99.69	99.68	99.69
K-means+MLP	4	99.70	99.70	99.69	99.70
K-means+MLP	6	99.41	99.41	99.41	99.41
K-means+MLP	8	99.61	99.60	99.60	99.60
Birch+MLP	3	99.59	99.59	99.57	99.59
Birch+MLP	9	99.69	99.69	99.69	99.69
<b>Birch+MLP</b>	<b>12</b>	<b>99.73</b>	<b>99.73</b>	<b>99.73</b>	<b>99.73</b>

TABLE VII  
EVALUATION METRICS OF MULTI-CLASSIFICATION FOR CICIDS-2017

	Precision	Recall	F1 score	FPR	Accuracy
Benign	0.9985	0.9965	0.9975	0.0015	<b>99.73%</b>
Bot	0.9034	0.6718	0.7706	0.0001	
Brute Force	0.9788	1.0000	0.9893	0.0003	
DoS/DDoS	0.9962	0.9989	0.9980	0.0020	
Infiltration	1.0000	0.7500	0.8571	0.0000	
PortScan	0.9986	0.9994	0.9990	0.0002	
Web Attack	0.9904	0.9450	0.9671	0.0000	
Avg.	0.9973	0.9973	0.9973	0.0015	

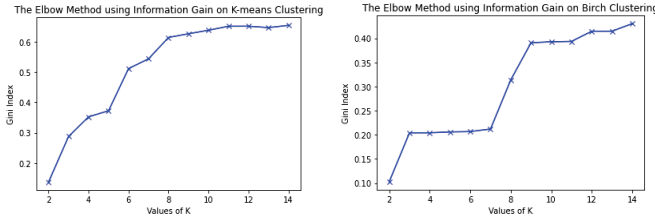


Fig. 7. The Elbow Method using Birch and K-means

incorrectly classified positive samples, False Positive(FP) for incorrectly classified negative samples, and True Negative(TN) for correctly classified negative samples. Other measures using TP, FN, FP, and TN are as follows.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (9)$$

$$Recall(TruthPositiveRate) = \frac{TP}{TP + FN} \quad (10)$$

$$Precision = \frac{TP}{TP + FP} \quad (11)$$

$$FPR(FalsePositiveRate) = \frac{FP}{TN + FP} \quad (12)$$

$$F1 - score = 2 \times \left( \frac{Precision \times Recall}{Precision + Recall} \right) \quad (13)$$

$$AUC_{ROC} = \int_0^1 \frac{TP}{TP + FN} d \frac{FP}{TN + FP} \quad (14)$$

BENIGN	55462	14	17	139	0	22	2
Bot	64	131	0	0	0	0	0
Brute Force	0	0	1383	0	0	0	0
DoS/DDoS	14	0	3	37958	0	0	0
Infiltration	2	0	0	0	2	0	0
PortScan	3	0	1	6	0	15870	0
Web Attack	3	0	9	0	0	0	206
	BENIGN	Bot	Brute Force	DoS/DDoS	Infiltration	PortScan	Web Attack

Fig. 8. Confusion Matrix

#### D. Results

According to the experimental results in Table VI, both k-means and Birch can achieve better performance than MLP alone. When using K-means with MLP, the best cluster number is 4, and it can get 99.70% accuracy. When using Birch with MLP, the optimal cluster number is 12, which can achieve 99.73% accuracy and is higher than using k-means.

Figure 8 and figure 9 demonstrated the confusion matrix and ROC curve obtained by the model. As shown in Figure 8, only a few samples are misclassified. In Figure 9, the ROC of each class is plotted using the one versus all principle. The roc area of almost all classes reaches a score over 99%, which means that the model has good generalization.

Table VII presented the performance of each class using Birch with a cluster number of 12. Benign, Brute Force, DoS/DDoS and Web Attack achieved over 95% accuracy. The performance of Bot and infiltration is slightly lower than other

TABLE VIII  
COMPARISON OF SIMILAR WORK

Work	Model	Precision(%)	Recall(%)	FPR(%)	F1 score(%)	Accuracy(%)
Rosay et al. [12]	MLP	99.51	99.41	0.49	99.46	99.46
Ustebay et al. [14]	DMLP	-	-	-	-	91
Jiang et al. [22]	MLP	99.87	99.60	-	99.41	99.23
Jabbar and Mohammed [23]	MLP	-	-	-	-	98.98
Azzaoui et al. [24]	DNN	80.33	-	0.07	-	99.43
Alrowaily et al. [25]	MLP	95.36	96.25	-	95.57	96.26
<b>Proposed Method</b>	<b>Birch+MLP</b>	<b>99.73</b>	<b>99.73</b>	<b>0.15</b>	<b>99.73</b>	<b>99.73</b>

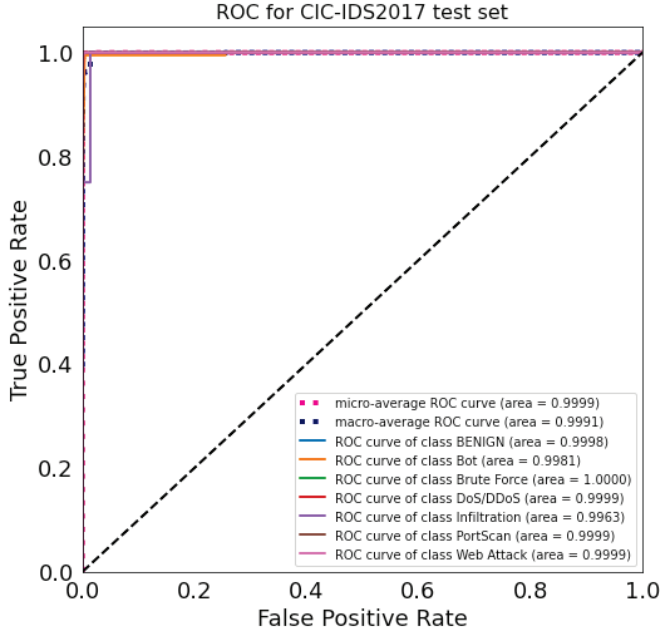


Fig. 9. ROC Curve

classes, which may be caused by their insufficient training samples. Furthermore, our model achieves a very low FPR of 0.15%.

#### E. Comparison

In table VIII, we compared the performance of our model with other studies using MLP as a classifier. It can be found that our model achieves the best performance in both f1 score and accuracy. Although the FPR is not the lowest, it is also a relatively good performance.

#### VI. CONCLUSION AND FUTURE WORK

In this paper, we proposed a two-stage intrusion detection system that combines both an unsupervised clustering algorithm and a supervised machine learning model. In our method, we used the Birch or k-means algorithm as an unsupervised clustering algorithm to group unlabeled data. The pseudo-label generated by the clustering algorithm is then added as an extra feature along with other features to our MLP model for multi-classification of network anomalies. In the experiment, we

used the elbow method and information gain to determine the potential cluster numbers. The experimental results show that using Birch and K-Means in our approach can help improve the multi-classification performance of the MLP classifier. The best accuracy performance of 99.73% can be obtained when using Birch with cluster number 12. After comparison, our model outperformed similar studies.

This paper initially explored the impact of two unsupervised clustering algorithms on MLP models. There are many variants of Birch and K-Means unsupervised clustering algorithms that can cluster unlabeled data more accurately. In the future, we plan to explore the effects of Birch and K-Means variants and other hyper-parameters of these algorithms on our hybrid model.

#### VII. ACKNOWLEDGEMENT

This work is supported by the Cyber Security Research Programme-Artificial Intelligence for Automating Response to Threats from the Ministry of Business, Innovation, and Employment (MBIE) of New Zealand as a part of the Catalyst Strategy Funds under the Grant Number MAUX1912.

#### REFERENCES

- [1] "Check Point Research: Cyber Attacks Increased 50% Year over Year," accessed 2022-06-03. [Online]. Available: <https://blog.checkpoint.com/2022/01/10/check-point-research-cyber-attacks-increased-50-year-over-year/>
- [2] A. Khraisat, I. Gondal, P. Vamplew, and J. Kamruzzaman, "Survey of intrusion detection systems: techniques, datasets and challenges," *Cybersecurity*, vol. 2, no. 1, pp. 1–22, 2019.
- [3] Z. Ahmad, A. Shahid Khan, C. Wai Shiang, J. Abdullah, and F. Ahmad, "Network intrusion detection system: A systematic study of machine learning and deep learning approaches," *Transactions on Emerging Telecommunications Technologies*, vol. 32, no. 1, p. e4150, 2021.
- [4] R. Zebari, A. Abdulazeez, D. Zeebaree, D. Zebari, and J. Saeed, "A comprehensive review of dimensionality reduction techniques for feature selection and feature extraction," *Journal of Applied Science and Technology Trends*, vol. 1, no. 2, pp. 56–70, 2020.
- [5] M. Dua et al., "Machine learning approach to ids: A comprehensive review," in *2019 3rd International conference on Electronics, Communication and Aerospace Technology (ICECA)*. IEEE, 2019, pp. 117–121.
- [6] M. Ring, S. Wunderlich, D. Scheuring, D. Landes, and A. Hotho, "A survey of network-based intrusion detection data sets," *Computers & Security*, vol. 86, pp. 147–167, 2019.
- [7] W. Xu, J. Jang-Jaccard, A. Singh, Y. Wei, and F. Sabrina, "Improving performance of autoencoder-based network anomaly detection on nsll-kdd dataset," *IEEE Access*, vol. 9, pp. 140 136–140 146, 2021.
- [8] R. Panigrahi and S. Borah, "A detailed analysis of cids2017 dataset for designing intrusion detection systems," *International Journal of Engineering & Technology*, vol. 7, no. 3.24, pp. 479–482, 2018.



- [9] S. M. Kasongo and Y. Sun, "Performance analysis of intrusion detection systems using a feature selection method on the unsw-nb15 dataset," *Journal of Big Data*, vol. 7, no. 1, pp. 1–20, 2020.
- [10] D. Jing and H.-B. Chen, "Svm based network intrusion detection for the unsw-nb15 dataset," in *2019 IEEE 13th international conference on ASIC (ASICON)*. IEEE, 2019, pp. 1–4.
- [11] F. A. Khan, A. Gumaei, A. Derhab, and A. Hussain, "A novel two-stage deep learning model for efficient network intrusion detection," *IEEE Access*, vol. 7, pp. 30 373–30 385, 2019.
- [12] A. Rosay, K. Riou, F. Carlier, and P. Leroux, "Multi-layer perceptron for network intrusion detection," *Annals of Telecommunications*, pp. 1–24, 2021.
- [13] Y.-B. Ho, W.-S. Yap, and K.-C. Khor, "The effect of sampling methods on the cids2017 network intrusion data set," in *IT Convergence and Security*. Springer, 2021, pp. 33–41.
- [14] S. Ustebay, Z. Turgut, and M. A. Aydin, "Intrusion detection system with recursive feature elimination by using random forest and deep learning classifier," in *2018 international congress on big data, deep learning and fighting cyber terrorism (IBIGDELFT)*. IEEE, 2018, pp. 71–76.
- [15] X. Kong, C. Wang, Y. Li, J. Hou, T. Jiang, and Z. Liu, "Traffic classification based on cnn-lstm hybrid network," in *International Forum on Digital TV and Wireless Multimedia Communications*. Springer, 2022, pp. 401–411.
- [16] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," *ICISSp*, vol. 1, pp. 108–116, 2018.
- [17] T. Zhang, R. Ramakrishnan, and M. Livny, "Birch: an efficient data clustering method for very large databases," *ACM sigmod record*, vol. 25, no. 2, pp. 103–114, 1996.
- [18] D. Marutho, S. H. Handaka, E. Wijaya *et al.*, "The determination of cluster number at k-mean using elbow method and purity evaluation on headline news," in *2018 international seminar on application for technology of information and communication*. IEEE, 2018, pp. 533–538.
- [19] R. Panigrahi and S. Borah, "A detailed analysis of cids2017 dataset for designing intrusion detection systems," *International Journal of Engineering & Technology*, vol. 7, no. 3.24, pp. 479–482, 2018.
- [20] D. Stiawan, M. Y. B. Idris, A. M. Bamhdi, R. Budiarto *et al.*, "Cids-2017 dataset feature analysis with information gain for anomaly detection," *IEEE Access*, vol. 8, pp. 132 911–132 921, 2020.
- [21] S. Patro and K. K. Sahu, "Normalization: A preprocessing stage," *arXiv preprint arXiv:1503.06462*, 2015.
- [22] J. Jiang, Q. Yu, M. Yu, G. Li, J. Chen, K. Liu, C. Liu, and W. Huang, "Alldd: a hybrid traffic-user behavior detection method for application layer ddos," in *2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/12th IEEE International Conference On Big Data Science And Engineering (Trust-Com/BigDataSE)*. IEEE, 2018, pp. 1565–1569.
- [23] A. F. Jabbar and I. J. Mohammed, "Development of an optimized botnet detection framework based on filters of features and machine learning classifiers using cids2017 dataset," in *IOP Conference Series: Materials Science and Engineering*, vol. 928, no. 3. IOP Publishing, 2020, p. 032027.
- [24] H. Azzaoui, A. Z. E. Boukhamla, D. Arroyo, and A. Bensayah, "Developing new deep-learning model to enhance network intrusion classification," *Evolving Systems*, pp. 1–9, 2021.
- [25] M. Alrowaily, F. Alenezi, and Z. Lu, "Effectiveness of machine learning based intrusion detection systems," in *International Conference on Security, Privacy and Anonymity in Computation, Communication and Storage*. Springer, 2019, pp. 277–288.