Journal on Information Technologies & Communications

https://www.ictmag.vn

Volume 2024, Number 2

September

TABLE OF CONTENTS

Network Traffic Feature Behaviour Augmentation Fusion based Attacks Classification for Intrusion Detection System in SDN Framework ..... ***Duong Van Dan, Tran Nam Khanh, Ta Minh Thanh*** 70

A Human Retrieval System based on Human Attribute Ontology and Deep Multi-task Neural Network ..... ***Trang T.T. Phung, Ngoc Q. Ly, Fukuzawa Masayuki*** 80

Mellifluous Viewport Bitrate Adaptation for 360° Videos Streaming over HTTP/2 ..... ***Nguyen Duc Thien, Tran Trieu Hai, Nguyen Viet Hung*** 94

Empirical Study of Ovarian Tumors Detection and Classification in Ultrasound Images ..... ***Thi-Loan Pham, Van-Hung Le, Thi-Lan Le, Hai Vu, Duy-Hai Vu, Thanh-Hai Tran*** 101

Method for Mining High-utility Patterns in Transaction Stream Data based on Linked List Structure ..... ***Minh-Thai Tran, Anh-Duy Tran, Duc-Thanh Pham, Minh-Nguyen Le*** 116

Mining Association Rules on Time-sensitive Data ..... ***Dang-Ngo Huu, Nghia Le, Khac-Chien Nguyen*** 130

Network Traffic Feature Behaviour Augmentation Fusion based Attacks Classification for Intrusion Detection System in SDN Framework

Duong Van Dan[1], Tran Nam Khanh[1], Ta Minh Thanh[1]
[1] Le Quy Don Technical University
Correspondence: Ta Minh Thanh, thanhtm@mta.edu.vn
Communication: received 02 Dec 2023, revised 22 Jan 2024, accepted 13 Mar 2024
Digital Object Identifier: 10.32913/mic-ict-research.v2024.n2.1247

Abstract: The implementation of Software-Defined Network (SDN) across multiple data centers aims to simplify the control and management of networks. However, the increasing popularity of SDN has also attracted the attention of attackers. To tackle this problem, it is essential to have an intrusion detection system (IDS) in place, which plays a crucial role in cybersecurity by addressing external threats. The advantage of SDN's centralized nature is that it facilitates the training of an IDS based on machine learning. However, there is a scarcity of research specifically focused on intrusion detection in SDN. Existing literature often treats SDN intrusion detection as similar to traditional computer systems and relies on intrusion datasets generated for those systems. We explore the issue of intrusion detection in SDN using the most recent public dataset (InSDN). However, InSDN is an imbalanced data set. In this paper, we have recommended a method to balance the data as well as a method to find the best features to improve the quality of IDS using Machine Learning. In addition, we also suggest a method of classifying SDN network traffic and normal network traffic. At the same time, we also evaluate the efficiency of the SDN system with the load balancing system and without the load balancing system.

Keywords: *Software-Defined Network, Intrusion Detection System, Machine Learning, Data Augmentation, Load Balancing, Network Flow Feature, SDN Detection, Feature Fusion.*

I. Introduction

1. Overview

The demand for network usage is continually growing in complexity, driven by advancements in both hardware and software technologies. Operating and managing the network manually shows certain disadvantages when errors occur, and flexibility is not high. Therefore, modern networks are moving towards automation as well as softwareization using advanced technologies such as SDN. SDN is a network architecture that allows centralized control over the network by separating the control plane from the data plane. There, the control plane will do things like traffic routing, congestion control, and so on.

Instead, the data plane just forwards the network traffic based on the configurations provided by the control plane. The control plane has the ability to control and view the network centrally. Therefore, it can make smart decisions to manage and operate the network securely. The advantage of SDN is indisputable, and it is increasingly widely used in data centers and enterprises. It is estimated that by 2027, the SDN market could reach 52 billion USD [1].

With its numerous advantages, SDN is gaining popularity and is increasing widely in network management. However, along with its increasing adoption, SDN has also drawn the attention of attackers. In SDN, the control plane holds centralized control and management over the network. If the control plane is compromised by an attacker, they can potentially gain control over the entire network, inadvertently turning SDN into a target for malicious attacks. This highlights the importance of implementing robust security measures to protect the control plane and mitigate the risk of such attacks in SDN environments.

Infiltrating the SDN server grants attackers unauthorized access, allowing them to acquire crucial system information. Their primary objective is to target the SDN controller, recognized as the brain of the network. Attackers may execute various malicious activities, including Denial-of-Service (DoS) attacks, Distributed Denial-of-Service (DDoS) attacks, Brute Force attacks, and other methods to compromise the controller's functionality and disrupt network operations. Protecting the SDN controller from such attacks is crucial to maintaining the security and

integrity of the entire SDN infrastructure.

It is extremely important to detect such attacks quickly so that prevention methods can be put in place. Normally, we would be able to think of IDS [2] as a common way to detect such attacks. More importantly, current IDS mostly use rules to detect attacks that have certain weaknesses, while IDS rely on network traffic analysis and packet characteristics to detect intrusions. Supposedly, such network traffic does not fall under the rules that the network administrator sets for a traditional IDS, but with an IDS using packet analysis, they can still be detected. This improves the quality of detecting malicious packets and attacks on network systems, helping network administrators quickly prevent attacks.

The utilization of Machine Learning (ML) for analyzing network packets and constructing IDS has become a well-established approach. ML has demonstrated notable advantages in terms of accuracy compared to traditional IDS methods. However, prior research has often overlooked the detection of specific attack details. Detecting these details can enable administrators to take targeted measures to prevent attacks effectively.

Furthermore, a prevalent issue in the ML-based IDS domain is the usage of imbalanced datasets for training ML algorithms. These datasets often exhibit a significant disparity in the frequency of different labels, which can result in learning algorithms becoming biased toward the more prevalent label during training. Consequently, when these algorithms are used to predict network flows, they may produce errors due to their skewed learning experience. Addressing this data imbalance challenge is crucial for developing reliable and effective ML-based IDS systems. Techniques such as data balancing methods, including over-sampling and under-sampling, can help mitigate this issue and improve the performance and accuracy of the ML algorithms for intrusion detection.

In addition, it is also important to detect what is SDN traffic and what is traditional network traffic. This helps attackers determine what their targets are and how they attack. However, there has not been much research on how to detect traditional network traffic and SDN network traffic. The use of modern technologies such as ML, DL is almost nonexistent. Therefore, we also focus on proposing a method that can classify whether the traffic belongs to the SDN network traffic or not.

2. Our contributions

In this paper, we focus on developing an IDS for SDN to detect and predict attacks and what types of attacks they are. In addition, we also develop a method for detecting SDN networks.

Our contributions can be explained as follows:

- Create a balanced dataset: The collected dataset obtained from real-world scenarios often exhibits a significant data imbalance, with the majority of data records representing instances when no attack is present. This imbalance can greatly skew the learning models used for IDS in SDN. The success of an IDS system in SDN heavily relies on the availability of a well-balanced dataset for training the models. Consequently, our first contribution in this paper is focused on the creation of a balanced dataset specifically tailored for SDN networks.

- Identifying a set of optimal features: Our second contribution is to evaluate the features from the balanced dataset in order to efficiently feed to our proposed algorithm. In general, using too many features leads to long training times and large file sizes that make it difficult to implement in practice. Therefore, we propose a method to find the feature set with the best classification ability while the model still ensures accuracy.

- Discover the best method: The third contribution is to conduct experiments and to find the most suitable method as well as an algorithm for detecting and also classifying data of attacks. Our suggested algorithms can improve IDS performance.

- Find a suitable load balancing algorithm: Lastly, we evaluate the performance of web servers in SDN without load balancing and with load balancing.

3. Roadmap

The rest of this paper is arranged as follows. Section II discusses the related work and theory analysis. In section III, the proposed method algorithm is described in detail. The experimental results and comparisons are shown in section IV. Finally, a short conclusion is drawn in section V.

II. Related Work

Intrusion detection is a difficult problem [3], which has been attempted by a lot of previous researchers to solve particularly in SDN. Most recent research works focused on applying ML [4–6] to build IDS for SDN. Deep Learning (DL) algorithms are also used in some research but not much. In this section, we discuss some of the previous studies concerned IDS for SDN.

In the article [7], the authors have built an IDS system using ML algorithms on the available data set. The results show that using ML to build IDS gives relatively good results, reducing dependence on leading experts and replacing the need for a large amount of data to train models. However, the method proposed by the authors has many

limitations, not overcoming the inherent weaknesses of the data set as well as improving the efficiency of IDS attack detection.

In more detail, IDS using ML in SDN has made great contributions to generating a relatively large and rich dataset of attacks in SDN [8]. This is the foundation for subsequent researchers to develop highly transparent ML-using IDSs for SDN systems. The authors also evaluated the generated dataset with basic ML algorithms and achieved very good results with specific detection of each type of attack. However, the authors have not solved the imbalance problem and are using a model to detect attacks in binary form, which has not been done at the multi-class level. In order to detect and defense the specific network attack such as flooding DDoS attack for SDN, Song Wang *et al*. [9] had employed lightweight supervised learning techniques. However, their experimental dataset is not belong to real SDN. Tuan *et al*. [10] proposed to compute entropy and logarithmic values to detect TCP-SYN and ICMP flood attacks in SDN, respectively. KNNs (K-Nearest Neighbor) are recruited to find the K nearest Euclidean distance points from the current entropy or logarithmic point to determine if the network is subject to a DDoS attack. This model is evaluated with CAIDA 2007 dataset and self-generated traffic using Bonesi, accuracy is over 99% when $K = 9$.

In another paper[11], the authors propose a method to detect early attacks in SDN networks. Their method took a set of 9 features and then transform them through CNN to obtain a 128-dimensional feature vector. Then let that feature vector go through algorithms like XGBoost and Random Forest (RF) for training. The solution method is relatively good and practical, and the training time is relatively low. However, they can only do it at the level of detecting attacks and not being attacked, not solving the problem of what type of attack. There are also some studies [12] on applying ML and Deep Learning (DL) to IDS problems of SDN. They summarized in detail the methods that have been applied and the technologies applied are relatively new. However, the authors use data sets that are not large enough, and the data imbalance is not resolved.

In the article "Network intrusion detection in software defined networking with self-organized constraint-based intelligent learning framework" [13], the authors have also outlined a number of ways to use Machine Learning to detect network intrusions, but the problem they focus on solving is to reduce the error when the model predicts as well as to avoid problems such as over-fitting. Recently, the application of AI technology to systems is gradually gaining popularity. There are many research works related to network intrusion detection using AI. A group of authors conducted a synthesis of methods to apply AI network intrusion detection in SDN [14]. However, the article has not yet provided solutions for the weaknesses of the existing works.

In the paper "Survey on SDN based network intrusion detection system using machine learning approaches" [15], the authors have synthesized methods of network intrusion detection using ML. The article mentioned tools that can develop NIDS (Network Based Intrusion Detection System) in a real environment. Conversely, the methods they came up with have not yet solved the problem of data imbalance. In another way, applying deep learning for network intrusion detection is also a very positive and potential direction. There are many groups of authors who have applied different deep learning methods to realize that. In the paper "A Novel Approach To Network Intrusion Detection System Using Deep Learning For Sdn: Futuristic Approach" [16], the authors used 12 features out of 41 features in the NSL-KDD dataset[1] by using the feature selection method. It is also considered to choose the effective feature of real network traffic. The results of the paper are relatively good and capable of applying it to practical implementation. However, the paper uses a general data set for the IDS system and it has not been specialized for SDN application.

In the paper "Early Detection of Intrusion in SDN" [17], Towhid *et. al.* focused on detecting network intrusions as early as possible. Using real-time stream-based traffic network features, they built a method for early detection of network intrusions. They used the InSDN dataset [8], then employed the 9 features outlined in the paper "A novel hybrid model for intrusion detection systems in SDNs based on CNN and a new regularization technique" [18]. However, their method also reduced the effectiveness of intrusion detection, not solving the problem of data imbalance. They also have not specifically detected each type of network attack.

According to previous studies, we found that there are several problems that are not still solved for SDN environment. Those are data imbalance and type of network attack detection for SDN. Our method proposes a method to balance the data and detect the details of each type of attack for IDS of SDN applications.

III. Our Proposed Method

In traditional networks, IDSs are commonly positioned behind firewalls. These IDSs typically receive data from the SPAN port of the switch, enabling them to analyze network data and make predictions. In the case of software-defined networking (SDN), all network traffic flows through the

[1] https://www.unb.ca/cic/datasets/nsl.html

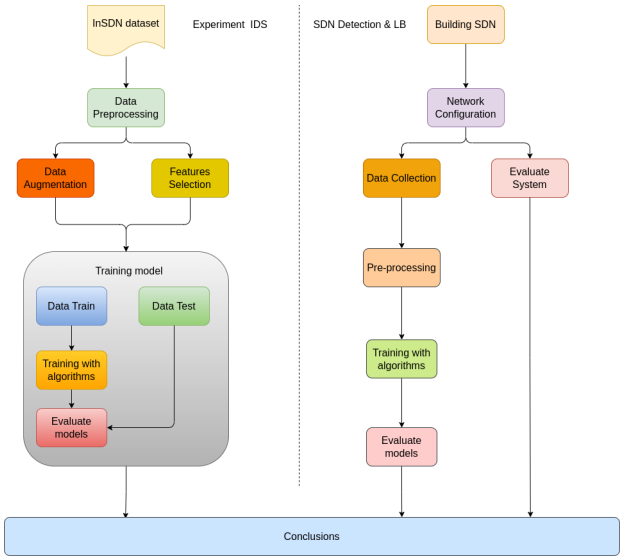


Figure 1. Pipeline for experiments of this paper

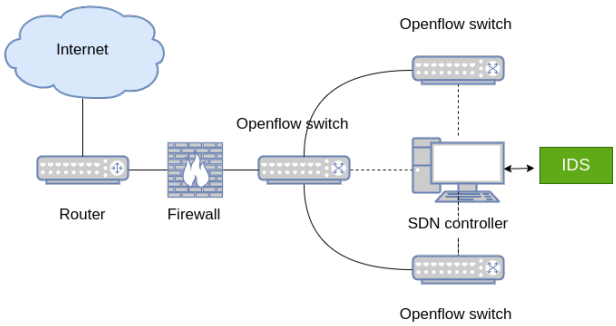


Figure 2. IDS location in SDN

SDN controller. Therefore, it is recommended to place the IDS behind the firewall and retrieve network traffic from the SDN controller. Our proposed IDS setup is illustrated in Fig. 2. Following the training of the model, the Machine Learning (ML) algorithm identified the model with the most effective classification capabilities. This model is then utilized for network intrusion detection in SDN. The network traffic originates from the client and passes through the SDN controller before reaching the server. During this process, the network traffic is directed through the IDS by the SDN controller. If the IDS detects any abnormal network traffic behavior and classifies it as intrusive, an alarm is triggered.

In this section, we present the issues that have been resolved in the article. Firstly, data imbalance is a big problem. Therefore, we have studied and proposed different methods to solve the data imbalance problem. Secondly, we propose a method to find the set of data imbalances features with the best classifier. This can improve the network intrusion detection classification ability for IDS. In addition, we also focus on reducing the number of features used in order to reduce the training time and weight of the model after training. This is the basis for being able to deploy IDS on actual devices. We perform this test on the dataset InSDN [7]. Thirdly, we propose an algorithm to classify traditional network traffic and that of SDN. To the best of our knowledge, this is the first method to classify the network traffic of traditional network (physical-based network) and that of SDN (software-based network). After distinguishing the type of network traffic from physical-based network or from software-based network (SDN), we can build different attacks and defense solutions for the SDN network. Finally, we evaluate the SDN system with and without load balancing for real applications.

In the second experiment, we build an SDN network with Mininet[2] and Ryu controller[3]. Conduct network traffic data collection of traditional networks and SDN networks. After collecting the data from the traditional network and the SDN network, we build a classifier of these two networks. Finally, based on the built-in network model, we evaluate the SDN system with load balancing and without load balancing in order to evaluate the its performance in real applications.

1. Our Approach

Data imbalance poses a significant challenge in Machine Learning (ML) [19] as it can adversely affect the learning process of algorithms. When the data is imbalanced, the algorithm tends to assign more weight to the labels that appear more frequently. To address this issue, various techniques such as down-sampling and over-sampling can be employed. However, down-sampling the data significantly reduces its size, then leads potentially resulting in the loss of important training data for the trained model.

In our paper, we present a novel approach for addressing data imbalance through the use of over-sampling, specifically employing the Synthetic Minority Oversampling Technique (SMOTE) [20]. The SMOTE technique, introduced by Nitesh Chawla *et al*. [21], operates by identifying instances that are close to each other in the feature space. It establishes a connection between these instances and generates synthetic samples along that connection, effectively increasing the representation of the minority class in the dataset [22].

Once we successfully addressed the issue of data imbalance, our focus shifted toward developing a method to identify features that serve as effective classifiers for our IDS. By identifying such features, we were able to

[2]http://mininet.org/
[3]https://ryu-sdn.org/

reduce the number of features required to construct the IDS. This reduction not only decreased the training time but also reduced the overall weight of the model. The resulting streamlined IDS was more efficient and effective in detecting network intrusions.

In this paper, we recognized the significance of distinguishing between SDN network traffic and traditional network traffic. To address this, we collected relevant data and constructed models to classify network traffic into these two categories. By developing these models, we aimed to accurately differentiate between SDN-specific traffic and traffic associated with traditional networks. This classification process provided valuable insights and contributed to a better understanding of network traffic patterns in the context of SDN.

Our project involved the design and implementation of a basic Software-Defined Networking (SDN) infrastructure using Mininet and Ryu controllers. The purpose of this setup was to assess the performance of the SDN system under two conditions: with load balancing and without load balancing. By comparing the outcomes of these two scenarios, we aimed to evaluate the impact of load balancing on the overall performance of the SDN system.

2. Data Pre-processing

In our data preprocessing phase, we performed cleaning operations on the existing dataset. Specifically, we removed any records that contained missing values represented as "NaN" fields. Subsequently, we addressed the data imbalance problem by applying techniques such as over-sampling and down-sampling. However, during our experimentation, we observed that down-sampling led to the loss of crucial data points, and therefore, we decided not to proceed with this method.

After addressing the data imbalance, we proceeded to split the dataset into three sets: training, testing, and validation. The split was performed at a ratio of 80% for training, 10% for testing, and 10% for validation. This division allowed us to have separate sets for training the model, evaluating its performance, and validating its generalization capabilities.

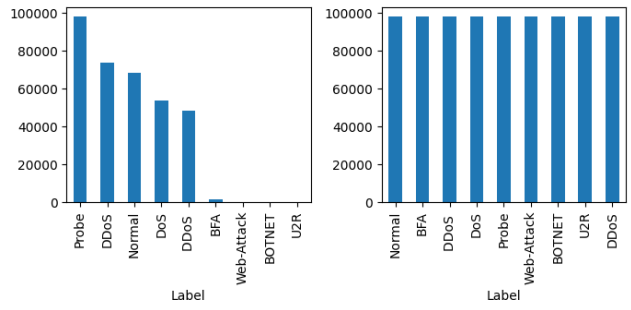


Figure 3. Before and After Using Data Augmentation

3. Data Augmentation

In our study, we utilize the Synthetic Minority Oversampling Technique (SMOTE) [22] to enhance the existing dataset. By employing SMOTE algorithm, our objective is twofold: to augment the data and to ensure a balanced representation across different IoT devices. To achieve this purpose, we leverage the largest number of device samples present in the dataset and apply SMOTE to generate an equivalent number of samples for the remaining devices [23]. This approach allows us to address the data imbalance issue and create a more representative and balanced dataset for further analysis and modeling.

Our approach follows a two-step process. First, we utilize the Imblearn [24] library, specifically leveraging the oversampling technique, to augment the dataset. This step involves generating synthetic samples to balance the class distribution in the dataset. Next, we apply this data augmentation method to our specific dataset, resulting in outcomes comparable to those shown in Fig.3. This approach allows us to effectively address the data imbalance and create a more balanced dataset for further analysis and modeling.

4. Features Selection

In fact, the collected IoT data includes many different features in which there are many features of less importance and features with low classification ability. Using all the features can cause interference with the model during the learning process, increase model weight after training, and training time.

In order to select the efficient features, we propose an algorithm, which allows us to find features with high classification ability for many ML algorithms. The input of this algorithm is all the features of the data set, the output is the features with the best classification ability.

In this problem, we use the function available in the libraries as feature "_importances_". This function is created by the library based on the mathematical formula of the authors Trevor Hastie, Robert Tibshirani, and Jerome Friedman [25]. They define the calculation of feature importance using the following equation:

$$I_l^2(T) = \sum_{t=1}^{J-1} i_t^2 \prod [v(t) = l], \tag{1}$$

where T is the whole decision tree, l feature in question, J number of internal nodes in the decision tree, i squared the reduction in the metric used for splitting, $\prod$ is indicator

Algorithm 1 Find the best features

Input: List of all features
Output: List of best features

```
Initialize Algorithms: models = {RandomForest, Xgboost, LightGBM}
Initialize important feature of models: iF = [ ]
Initialize list of best features: bestFeature = [ ]
for each in models do
    Model training with all features
    Append to iF
end for
for i = 0 to 50 do
    for j = 0 to 50 do
        for k = 0 to 50 do
            if iF['RF'][i]=iF['Xgb'][j]=iF['LGBM'][k] then
                Append feature to bestFeature
            end if
        end for
    end for
end for
Return bestFeature
```

function, $v(t)$ is a feature used in splitting of the node t used in splitting of the node. This math formula will sum up all the decreases in the metric for all the features across the tree (Random Forest, Xgboost, and LightGBM algorithms are all based on Decision Tree algorithm).

We train algorithms like RandomForest, XGBoost, and LightGBM with all features. We then superimpose the significant features of each algorithm. Within the 50 most important features of each algorithm, if it is also the important feature of the rest of the algorithms, we add the most important feature.

Once we have found those important features, we'll retrain them with basic Machine Learning algorithms to evaluate accuracy of IoT devices classification.

Based on Formula (1), we utilize the "_importances_" function in the libraries to conduct experiments as described earlier. As a result, we got 12 features, as shown in Table I.

5. Data Collection and Pre-processing

To be able to distinguish traditional network traffic and SDN network traffic. We collect network traffic data of traditional networks and SDN networks. We use Wireshark to capture packets exchanged in the experimental network.

To do this, we built an experimental system. We use docker to create a traditional network. This network consists of 10 clients and 3 servers. Similarly, in SDN we use Mininet and Ryu controller to build an equivalent network of 10 clients and 3 servers.

Then, we collect the data in turn of the networks. We let the devices in the traditional network communicate with each other and used Wireshark[4] to capture the packets it exchanged and label them as "0". Next, we let the devices in the SDN communicate with each other and also use Wireshark to capture those packets and label them as "1".

In order to be able to use this data to classify traditional networks and SDN networks. We need to clean the data first. The collected data will include a lot of "NaN" data fields. We need to get rid of them. For example, the problem we are solving is based on IPv4 data fields, so the information in the network related to IPv6 will be corrupted. We need to remove the records with IPv6 data fields away.

In general, most of our experiments are built on a relatively small network model. The allocation of IP addresses to devices in network models is relatively limited. When the number of IP addresses allocated in the network is relatively small, the IP address data collection that exists in the training dataset is also relatively small. This causes the model during training to be biased. It will default when it sees IP address A as the traditional network. Otherwise, it will default when it sees IP address B as SDN. To solve this problem, we can do it in two ways. We can build a large enough network with a large enough number of source and destination IP addresses to avoid this problem. The above is only suitable for large systems, we do not recommend it for small experimental systems. Another simpler way, we can remove the source IP and destination IP address features to train the model. Based on our preprocess, the bias of small dataset can be avoided.

IV. Results and Comparison

1. Experimental environment

Our experimental environment is on a PC with the following configuration: Intel i7-10700f CPU, 32 GB RAM, and 16GB VRAM RTX A4000 VGA card. We choose Python language and jupyter notebook platform to perform the experiments. When we train the model, we use Python version 3.6.13, sci-kit learn 0.24.2, TensorFlow 2.6.2, Numpy 1.19.2, pandas 1.1.5, and some other tools to analyze and train the model in the experiment.

In this paper, we use some popular Machine learning algorithms for testing such as Random Forest (RF), XGBoost (XGB), LightGBM (LGBM), and some others. They require the use of relatively different libraries. RF belongs

[4]https://www.wireshark.org/

to sci-kit learn library, XGB belongs to XGBoost library[5] and LGBM belongs to Lightgbm library[6].

2. Dataset preparation

In this section, we provide a description of the dataset utilized for training our Machine Learning algorithms. We employed the InSDN dataset [8], which was created in 2020. This dataset is notably extensive and encompasses a wide range of SDN network data. It incorporates data related to diverse types of attacks, including DoS, DDoS, Web-attacks, and numerous other attack types. The dataset contains a total of 343889 distinct records of network traffic in SDN. It comprises 84 columns, with 83 columns representing various network features, and 1 column indicating the label. Among the different types of attacks, Probe attacks make up the majority with 98129 records, while U2R attackers have the smallest representation, comprising only 17 records.

It is evident that the dataset is significantly imbalanced, with a substantial difference in the number of records between different labels. To address this issue, we employ the SMOTE method with over-sampling. By utilizing SMOTE, we successfully generate synthetic samples and achieve a balanced dataset, as depicted in Fig. 3. That is the first training dataset in our experiment.

In the second training dataset, we use the selected features. To select these features, we used all the features of the dataset to train with machine learning algorithms such as Random Forest, XGBoost, and LightGBM. After training, we obtain the order of important features for each algorithm. Within the 50 most important features of each algorithm, we superimpose each other to obtain the most important features. This method has been described in the section III.4.

We then obtain a list of features that fall within the 50 most important features of each algorithm. This list includes 12 features which is shown in Table I.

3. Result of ML application in intrusion detection

In this section, we do experiment with two training datasets. These two data sets are the result of our two different experimental directions. Both datasets we created for the experiment are based on the original data set, InSDN. For the first experiment, we used an incremental dataset. In the second experiment, we use the dataset after the important features have been extracted based on our proposed method.

[5]https://xgboost.readthedocs.io/
[6]https://lightgbm.readthedocs.io/

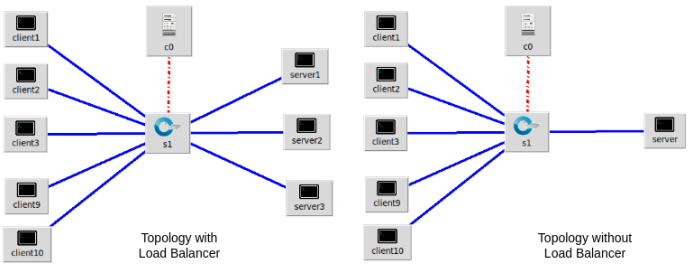


Figure 4. Topology of SDN

TABLE I
LIST OF IMPORTANT FEATURES

Ft/Algo	XGBoost	LightGBM	RandomForest
Bwd Pkts/s	**0.179**	425	0.039
Dst Port	0.073	987	0.052
Bwd Header Len	0.067	110	0.015
Timestamp	0.046	**1753**	**0.093**
Src IP	0.034	1470	0.048
Init Bwd Win Byts	0.027	351	0.061
Flow ID	0.014	1715	0.038
Dst IP	0.006	564	0.031
Src Port	0.005	892	0.035
Flow IAT Min	0.001	394	0.025
Fwd Pkts/s	0.0006	302	0.027
Flow Duration	0.0002	822	0.029

We tested in turn with algorithms such as Decision Tree, Random Forest, XGBoost, Logistic Regression, and LightGBM. First, we do experiment with the original dataset. Then, we implement experiment with the balanced and enriched dataset. Finally, we evaluate with the dataset after having extracted the important features.

Table II presents a comparison of the results before and after applying the data balancing method. The initial dataset was unbalanced, leading to low values for evaluation metrics such as F1-score. Furthermore, there were noticeable differences between the performance indexes. However, after applying the data balancing method, significant improvements are observed in our experimental results. The F1-score shows substantial enhancement. This indicates that balancing the data has positively impacted the performance of the model, resulting in improved accuracy and effectiveness in the classification task.

The algorithm for the highest accuracy is Random Forest with accuracy up to 99.95%. However, the XGBoost algorithm has the largest Recall and F1-scores at 99.90% and 99.86%, respectively. This index reflects the learning level of the model with the unbalanced dataset. With the current stats showing, the XGBoost algorithm with augmented dataset (XGBoost + Aug) is superior compared with another ML methods. The details of experimental results are shown in Table II.

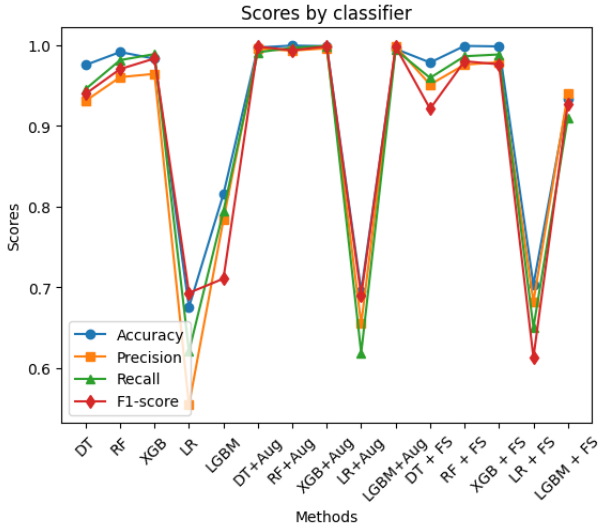


Figure 5. Scores of The Methods

TABLE II
RESULTS USING THE METHODS

Method	Accuracy	Precision	Recall	F1-score
Decision Tree	0.9755	0.9311	0.9457	0.9398
Random Forest	0.9915	0.9604	0.9815	0.9704
XGBoost	0.9833	0.9645	0.9890	0.9835
Logistic Regression	0.6755	0.5545	0.6208	0.6932
LightGBM	0.8156	0.7839	0.7943	0.7108
Decision Tree + Aug	0.9972	0.9955	0.9912	0.9983
Random Forest + Aug	**0.9995**	0.9932	0.9967	0.9936
XGBoost + Aug	0.9988	0.9962	**0.9990**	**0.9986**
Logistic Regression + Aug	0.6955	0.6549	0.6188	0.6898
LightGBM + Aug	0.9954	**0.9987**	0.9943	0.9981
Decision Tree + FS	0.9785	0.9512	0.9595	0.9212
Random Forest + FS	0.9991	0.9759	0.9863	0.9801
XGBoost + FS	0.9985	0.9789	0.9885	0.9763
Logistic Regression + FS	0.7032	0.6821	0.6501	0.6132
LightGBM + FS	0.9338	0.9402	0.9101	0.9271
InSDN[8] + Random Forest	-	0.9942	0.9969	0.9955
NIDS-DL[16] + CNN	0.9863	0.9845	0.9898	0.9872

As shown in Table II, the accuracy of algorithms such as Random Forest, and CNN on datasets like InSDN[8] and NIDS-DL[16] is relatively high. However, algorithms such as Random Forest, XGBoost, and LightGBM improved performance when combined with augmented datasets.

In addition, our Features Selection (FS) method gives less accurate results than the data enrichment method. However, after applying the FS method, the model training time is significantly reduced and the model weight is also reduced relatively. It implies that our FS method is effective for feature selection from many less efficient features.

TABLE III
TRAINING TIME AND FILE SIZE WITH FS

Method	Accuracy	Training time	File size
Random Forest	99.15	1m16s	4.9mb
Xgboost	98.33	5m59s	821.3kb
LightGBM	81.56	35.5s	2.1mb
Random Forest + FS	**99.91**	48.8s	2.3mb
Xgboost + FS	99.85	1m14s	**815.1kb**
LightGBM + FS	93.38	**15.7s**	2.0mb

TABLE IV
RESULTS USING ML FOR SDN DETECTION

Method	Accuracy	Precision	Recall	F1-score
Decision Tree	0.9935	0.9201	0.9655	0.9638
Random Forest	0.9951	0.9356	**0.9832**	0.9614
XGBoost	**0.9973**	**0.9745**	0.9830	**0.9795**
Logistic Regression	0.9765	0.9641	0.9608	0.9432
LightGBM	0.9906	0.9539	0.9747	0.9632

Table III shows that after applying FS method, the training time and file size of the model after training are greatly reduced. There is a slight increase in the accuracy of the model compared to using the original data set. The file size of the Random forest algorithm is halved.

4. SDN Detection

In this section, we explain the results of traditional network and SDN detection based on network traffic. With the dataset collected in both types of networks, we proceed to process, label and train the model.

The dataset we collected includes 91489 rows and 35 columns. After preprocessing the data, we removed 2 columns, so the training data remained 33 columns. The two columns of data that we leave out are ip.src and ip.dst. As we explained above, because our dataset is small, it is necessary to remove the source and destination IP addresses to avoid the bias of experimental results.

With the collected data set as described above, we train the model to detect SDN. The algorithms we tested cover the basics of Machine Learning. Our experiment works on algorithms like Decision Tree, Random Forest, and XGBoost. With our dataset, when testing with the above basic algorithms, the results are significantly improved.

The dataset we collected has not yet resolved the imbalance problem, so the results still have a difference between F1-score and Accuracy. Table IV shows that when using ML, we can be completely possible to detect SDN. The above results are also very positive and have potential for real development.

TABLE V
SYSTEM RESPONSE TIME

Method	max	avg	min
Non-LB	23.8ms	13.2ms	0.5ms
LB-RR	**13.8ms**	**7.2ms**	**0.4 ms**
LB-WRR	15.3ms	7.5ms	0.51ms

5. Evaluation of SDN system with load balancing

In this section, we have successfully constructed a basic SDN network using the Mininet network emulator and the Ryu controller. Building a simple SDN network is straightforward with these tools. For our setup, we opted for the simplest network topology, which is a star topology. The network architecture includes 10 clients, 3 web servers, 1 switch, and 1 controller. This configuration allows for the evaluation and testing of SDN functionalities and applications within a controlled environment.

Setup three web servers to open port 80 providing HTTP services. With the web server installed as Apache. Program the Ryu controller to test all 3 cases: Non-Load Balancing, Load Balancing with Round Robin, and Load Balancing with Weight Round Robin. With Weight Round Robin we set the weights for servers 1,2,3 to be 1,3,5, respectively.

In order to evaluate the response time of the system, we conducted experiments using three different network configurations. The network topology was set up as depicted in Fig. 4, utilizing Mininet and the Ryu controller. For the load balancing experiment, we programmed the Ryu controller to manage the traffic distribution among the servers. In the Round Robin algorithm case, equal weights of 1 were assigned to all servers. In the Weight Round Robin scenario, the weights were set as 1, 3, and 5 for servers 1, 2, and 3 respectively. Through these experiments, we measured and analyzed the response time of the system under different load balancing strategies.

We can see the response time is extremely small. The response time of the system with load balancing is comparatively much better than the system without load balancing. In particular, the Round Robin algorithm proved to be superior when there were significantly better coefficients. The fastest response time was 0.4 ms and the longest was 23.8 ms.

V. Conclusions

SDN is a relatively new and highly applicable field. However, it also has many potential security problems when centralizing control in the controller. The study of network intrusion detection methods is extremely necessary for the system.

In this paper, we have proposed a method to build IDS based on ML using network traffic. Our results are relatively good and can be deployed into real systems. We also propose methods to solve the problem of data imbalance. In addition, we also found the features with the best classification. The detection of SDN traffic has also been demonstrated by us. Finally, we also evaluate the performance of the SDN network with the load balancing system and without the load balancing system.

In the future, we will continue to develop our research. We will study and apply Deep Learning models to the problem. Test the actual implementation of the model on a given company.

References

[1] J. A. Sava, "Software-defined networking (sdn) market size worldwide from 2021 to 2027," Available at https://www.statista.com/statistics/468636/global-sdn-market-size/ (2023/25/5).

[2] A. Chawla, B. Lee, S. Fallon, and P. Jacob, "Host based intrusion detection system with combined cnn/rnn model," in *ECML PKDD 2018 Workshops*, C. Alzate, A. Monreale, H. Assem, A. Bifet, T. S. Buda, B. Caglayan, B. Drury, E. García-Martín, R. Gavaldà, I. Koprinska, S. Kramer, N. Lavesson, M. Madden, I. Molloy, M.-I. Nicolae, and M. Sinn, Eds. Cham: Springer International Publishing, 2019, pp. 149–158.

[3] Y. Al-Nashif, A. A. Kumar, S. Hariri, Y. Luo, F. Szidarovsky, and G. Qu, "Multi-level intrusion detection system (ml-ids)," in *2008 International Conference on Autonomic Computing*, 2008, pp. 131–140.

[4] S. S. Gopalan, D. Ravikumar, D. Linekar, A. Raza, and M. Hasib, "Balancing approaches towards ml for ids: A survey for the cse-cic ids dataset," in *2020 International Conference on Communications, Signal Processing, and their Applications (ICCSPA)*, 2021, pp. 1–6.

[5] A. S. Dina and D. Manivannan, "Intrusion detection based on machine learning techniques in computer networks," *Internet of Things*, vol. 16, p. 100462, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2542660521001037

[6] P. Parkar and A. Bilimoria, "A survey on cyber security ids using ml methods," in *2021 5th International Conference on Intelligent Computing and Control Systems (ICICCS)*, 2021, pp. 352–360.

[7] M. Almgren and E. Jonsson, "Using active learning in intrusion detection," in *Proceedings. 17th IEEE Computer Security Foundations Workshop, 2004.*, 2004, pp. 88–98.

[8] M. S. Elsayed, N.-A. Le-Khac, and A. D. Jurcut, "Insdn: A novel sdn intrusion dataset," *IEEE Access*, vol. 8, pp. 165 263–165 284, 2020.

[9] S. Wang, J. F. Balarezo, K. G. Chavez, A. Al-Hourani, S. Kandeepan, M. R. Asghar, and G. Russello, "Detecting flooding ddos attacks in software defined networks using supervised learning techniques," *Engineering Science and Technology, an International Journal*, vol. 35, p. 101176, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2215098622000842

[10] N. N. Tuan, P. H. Hung, N. D. Nghia, N. V. Tho, T. V. Phan, and N. H. Thanh, "A ddos attack mitigation scheme in isp networks using machine learning based on sdn," *Electronics*, vol. 9, no. 3, 2020. [Online]. Available: https://www.mdpi.com/2079-9292/9/3/413

[11] N. S. Shamim Towhid, "Early detection of intrusion in sdn," Available at https://uregina.ca/~nss373/papers/Early-detection-SDN.pdf (2023/25/5).
[12] M. R. Ahmed, salekul Islam, S. Shatabda, A. K. M. M. Islam, and M. T. I. Robin, "Intrusion Detection System in Software-Defined Networks Using Machine Learning and Deep Learning Techniques –A Comprehensive Survey," 12 2021. [Online]. Available: https://www.techrxiv.org/articles/preprint/Intrusion_Detection_System_in_Software-Defined_Networks_Using_Machine_Learning_and_Deep_Learning_Techniques_A_Comprehensive_Survey/17153213
[13] A. Bhardwaj, R. Tyagi, N. Sharma, A. Khare, M. S. Punia, and V. K. Garg, "Network intrusion detection in software defined networking with self-organized constraint-based intelligent learning framework," *Measurement: Sensors*, vol. 24, p. 100580, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2665917422002148
[14] S. Dahiya, V. Siwach, and H. Sehrawat, "Review of ai techniques in development of network intrusion detection system in sdn framework," in *2021 International Conference on Computational Performance Evaluation (ComPE)*, 2021, pp. 168–174.
[15] N. Sultana, N. Chilamkurti, W. Peng, and R. Alhadad, "Survey on sdn based network intrusion detection system using machine learning approaches," *Peer-to-Peer Networking and Applications*, vol. 12, no. 2, pp. 493–501, Mar 2019. [Online]. Available: https://doi.org/10.1007/s12083-017-0630-0
[16] M. R. Hadi and A. S. Mohammed, "A novel approach to network intrusion detection system using deep learning for SDN: Futuristic approach," in *Machine Learning &amp Applications*. Academy and Industry Research Collaboration Center (AIRCC), jun 2022. [Online]. Available: https://doi.org/10.5121%2Fcsit.2022.121106
[17] M. S. Towhid and N. Shahriar, "Early detection of intrusion in sdn," in *NOMS 2023-2023 IEEE/IFIP Network Operations and Management Symposium*, 2023, pp. 1–6.
[18] M. S. ElSayed, N.-A. Le-Khac, M. A. Albahar, and A. Jurcut, "A novel hybrid model for intrusion detection systems in sdns based on cnn and a new regularization technique," *Journal of Network and Computer Applications*, vol. 191, p. 103160, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1084804521001739
[19] S. Ray, "A quick review of machine learning algorithms," in *2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon)*, 2019, pp. 35–39.
[20] P. Jeatrakul, K. W. Wong, and C. C. Fung, "Classification of imbalanced data by combining the complementary neural network and smote algorithm," in *Neural Information Processing. Models and Applications*, K. W. Wong, B. S. U. Mendis, and A. Bouzerdoum, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 152–159.
[21] N. Chawla, K. Bowyer, L. Hall, and W. Kegelmeyer, "Smote: Synthetic minority over-sampling technique," *J. Artif. Intell. Res. (JAIR)*, vol. 16, pp. 321–357, 06 2002.
[22] G. Douzas and F. Bacao, "Geometric smote a geometrically enhanced drop-in replacement for smote," *Information Sciences*, vol. 501, pp. 118–135, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0020025519305353
[23] Z. Wang, J. Hu, G. Min, Z. Zhao, and J. Wang, "Data-augmentation-based cellular traffic prediction in edge-computing-enabled smart city," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 6, pp. 4179–4187, 2021.
[24] G. Lemaître, F. Nogueira, and C. K. Aridas, "Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning," *Journal of Machine Learning Research*, vol. 18, no. 17, pp. 1–5, 2017. [Online]. Available: http://jmlr.org/papers/v18/16-365
[25] T. Hastie, R. Tibshirani, and J. Friedman, *The elements of statistical learning: data mining, inference and prediction*, 2nd ed. Springer, 2009. [Online]. Available: http://www-stat.stanford.edu/~tibs/ElemStatLearn/



Van Dan Duong graduated with a degree in Information Technology from Le Quy Don Technical University in Vietnam in 2023. His research interests lie in the areas of deep learning, steganography, and computer vision.
Email: duongvandan2806@gmail.com



Nam Khanh Tran obtained an impressive IT engineering degree from Le Quy Don Technical University in Vietnam in 2020. He has since been contributing to the academic community as a lecturer at the same university, specializing in the research fields of cybersecurity, technology network, and image processing.
Email: khanhtn107195@lqdtu.edu.vn



Minh Thanh Ta is currently an Associate Professor and Vice Dean of Institute of Information and Communication Technology in Le Quy Don Technical University, Vietnam. He is also a Postdoctoral Fellow of the Department of Mathematical and Computing Sciences at Tokyo Institute of Technology. He received his B.S. and M.S in Computer Science from National Defense Academy, Japan, in 2005 and 2008 and his Ph.D. from Tokyo Institute of Technology, Japan, in 2015, respectively. He is a member of IPSJ Japan and IEEE. His research interests lie in the area of watermarking, network security, and computer vision.
Email: thanhtm@lqdtu.edu.vn

A Human Retrieval System based on Human Attribute Ontology and Deep Multi-task Neural Network

Trang T.T. Phung[1,2], Ngoc Q. Ly[1], Fukuzawa Masayuki[3]

[1] Faculty of Information Technology, HCM University of Science –VNUHCM, Vietnam.

[2] Saigon University, HCM City, Vietnam

[3] Kyoto of Institue of Technology (KIT), Japan

Correspondence: Trang T.T. Phung, trangphung@sgu.edu.vn

Communication: received 31 Jan 2024, revised 12 Mar 2024, accepted 05 Apr 2024

Digital Object Identifier: 10.32913/mic-ict-research.v2024.n2.1255

Abstract: The goal of this research is to enhance the capability of image retrieval systems to understand images more effectively. We present a model designed for searching human objects (such as pedestrians or persons) within expansive image datasets. Our unique approach involves developing an image retrieval system that incorporates attribute learning and the Human Attribute Ontology (HAO). This research offers several key contributions: (1) The development of the Human Attribute Ontology (HAO) which serves as a repository for storing prior knowledge about images. Thanks to its hierarchical structure, this ontology facilitates the reuse of prior knowledge, optimizing the subsequent stages of attribute learning and image retrieval; (2) The implementation of a Convolutional Neural Network (CNN) to spearhead attribute learning, leveraging the HAO to enhance accuracy; (3) The creation of a Human Image Retrieval system that utilizes both attribute learning and the HAO. Our system delves deeper by understanding images at the attribute level, highlighting the advantages of harnessing the ontology to reuse existing knowledge. The efficacy of our methodology is validated through experiments on benchmark datasets like PETA and Pa100k achieving state-of-the-art results.

Keywords: *Image Retrieval, Object Retrieval, Human Attribute Ontology, Attribute Learning, Deep Learning, Deep feature learning.*

I. INTRODUCTION

Image retrieval (IR) is a useful application for people in many aspects from daily life to special business. Studies about IR have been developed in methods for retrieval systems with more and more variety and popularity. Human image retrieval (HR) has an important role in surveillance and security [1, 2]. Recently, the approaches for building person retrieval systems are popular such as part-based [3, 4], graph-based [5–7], attribute-based [8–13], and text-based [14, 15], and used deep learning models [16] for feature extraction or fine-tuning models from famous CNNs such as ResNet [17], VGG [14], and EfficientNet [18]. In the semantic image approach, models have used ontology [19] for building retrieval systems, such as facial attribute retrieval with FAO (face attribute ontology) [20], and Bio-FAO (Bio-inspired Facial Aesthetic Ontology)[21].

Human image retrieval systems have met some challenges in images such as motion blur, occlusion, pose, camera angles, lighting conditions, and other kinds of image degradation.

Recently, with The Industry Revolution 4.0, intelligent products have been developed for society's demand. It is also a big challenge. How to solve the challenges is a difficult question.

Using attribute recognition for image retrieval will help the retrieval be more intelligent because it could find out attributes of objects in images instead of looking at objects only. For example, finding a person with the attribute "red shirt", the recognition will help the retrieval can detect the attribute "red shirt" instead of the shirt only.

Ontology is a tool to represent knowledge for sharing and reusing [17]. Using ontology for organizing and storing prior knowledge of images will help image retrieval systems at the semantic level. With the information on images in ontology, we can gain relevant information on images such as attributes of objects, relations between objects and objects as intra-relations, and attributes and objects as inter-relations. On the other hand, ontology supports the reuse of prior knowledge without learning (training and testing) or re-learning by relations in ontology.

In this paper, we try to build an HR by using three main tasks: (1) Designing the HAO for organizing and storing

prior knowledge of images, (2) proposing a new Attribute Learning method based on HAO, and (3) building a human image retrieval based on attribute learning and HAO. The model has solved some challenges in image understanding and closed the gap in image semantics as follows: First, improving attribute learning to increase accuracy and more intelligence for the retrieval system. Second, building the HAO to support attribute learning and reusing prior knowledge, more understanding of images at the semantic level, and effective performance for the retrieval system.

In short, our contribution to this topic is as follows:

- Building a new ontology named: Human Attribute Ontology (HAO) for organizing and storing information of human object attributes.
- Proposing an Attribute Learning method based on deep neural network and HAO.
- Building a new human attribute retrieval system based on HAO and Attribute Learning.
- The results of the proposal gained good results on the benchmark datasets PETA, and Pa100k.

II. RELATED WORKS

1. Deep Attribute Learning

Visual attributes are generally defined as mid-level semantic visual concepts [22] between low-level features and high-level semantics of images. Attribute learning has an important role in supporting image understanding in a fine-grained approach. In the traditional approach, studies used hand-crafted feature SIFT, HOG for feature extraction and SVM for attribute learning. For example, instead of searching for zebras, it is easier to find a black and white striped pattern [23] or to find a spotted dog is replaced by looking for a black dot pattern [24]. Popular works about attribute learning are in [5, 6, 25, 26], attribute learning for face retrieval of [27], Relative Attributes learning [28]. Recently, most studies have used deep learning to improve the accuracy of attribute learning. Many works about object attributes are studied and developed for building effective applications such as person attributes, Face attributes [29–31], vehicle attributes [32], and fashion attributes [27].

In human attributes learning, most studies use transfer learning of famous neural networks, such as VGG, ResNet, etc for feature extraction. After the feature extraction step, the feature vector will be input for the attribute learning method, such as SVM, Bayesian, or other Neural Networks. At the end of the attribute learning step, an attribute vector is returned with a one-hot type vector or list of scores on each attribute. Some popular studies in person attribute learning [33–43].

In [44], Yuxuan Shi et al. (2020) proposed an appearance attribute model named “Attribute Mining and Reasoning” (AMR) for person retrieval by using CNN and Graph Convolution to build two components: ALE (Attribute Attribute Localization Ensemble) and AR (Attribute Reasoning).

In [15], Chiat-Pin Tay et al. proposed an Attribute Attention Network (AANet) by using ResNet-50 architecture. The AANet consists of a global person ID task, a part detection task, and a crucial attribute detection task. Focused attributes are the six attributes such as hair, upper clothing color, lower clothing color, etc.

In [45], Jingya Wang et al. (2018) used encoder-decoder to build a Transferable Joint Attribute-Identity Deep Learning (TJ-AIDL) for simultaneously learning an attribute-semantic for reID and gained the state-of-the-art methods on four challenging benchmarks including VIPeR, PRID, Market-1501, and DukeMTMC.

In [46], Yifang Men et al. (2020) used GAN for person attribute learning and reID with attribute pose, head, upper clothes, and pants.

In [10], the authors proposed a Person re-identification (re-ID) and attribute recognition to share a common target for learning pedestrian descriptions. They used two steps in the model: (1) feature extraction step and (2) classifier step by building a multi-task attribute-person recognition (ARN)network using Resnet50 for feature extraction and classifier block with linear transform and sigmoid. The result of attribute learning is to return a one-hot attribute vector for each image.

In [40], the authors used EfficientNet for attribute learning based on the human part and feature pyramid structure in video surveillance to gain good results in datasets PETA, RAP, and PA-100k.

In [47], authors Yiqiang Chen et al. proposed a pedestrian attribute recognition such as gender, clothing, or hair by combining deep features (feature extraction from CNN) and handcrafted features Local Maximal Occurrence (LOMO) and part-based learning to model attribute patterns related to different body parts. Their experiments achieved state-of-the-art results on PETA, ApiS, and VIPeR datasets.

In [48], authors Xiangpeng Song et al. used GCN to build a pedestrian attribute recognition model and gained the state-of-the-art on the RAP dataset.

In [42], Zichang Tan et al. used GCN to build a new end-to-end network, named Joint Learning of Attribute and Contextual Relations (JLAC) for pedestrian attribute recognition with two modules (1) an attribute graph Attribute Relation Module (ARM) and (2) Contextual Relation Module (CRM) to explore the contextual relations among those

regions. the experiments on PA-100K, RAP, and PETA attribute datasets gained good results.

In [5], to exploit the inter-local relation and the intra-local relation among parts from different images for person re-identification (Re-ID), they proposed a novel deep graph model named Heterogeneous Local Graph Attention Networks (HLGAT). The model demonstrated effectiveness and gained state-of-the-art on Market-1501, CUHK03, DukeMTMC-reID, and MSMT17.

In [49], Qi Dong et al. proposed an Attribute-Image Hierarchical Matching (AIHM) model for attribute learning and retrieval with text query input. They used zero-shot learning and deep Multitask learning (MTL) for the system and gained good results on benchmark datasets PA100K, Market1501, and DukeMTMC.

Sarafianos et al. [50] focused on visual attributes of people such as backpack, hat, gender, height, and clothing to propose a method by combining multi-task learning and curriculum learning for attribute classification to try to transfer knowledge from a strong group to a weakly correlated group. Their experiments gained state-of-the-art on PETA and WIDER datasets.

In [51], Shengyu Pei et al. suggested a multitask model for person re-identification based on attributes. They used ResNet for the feature extraction performance. After using generalized mean pooling for feature aggregation, the identity-based and attribute-based double-stream network modules pay attention to the relationship between identities and pedestrian features, and the relationship between attributes and pedestrian features, to fully activate the relationship between pedestrian attributes and identities. Their experiments are on datasets Market-1501 and DukeMTMC-reID and gained good results to prove the effectiveness of the method.

In [52], authors Ngoc Q. Ly et al. suggested a deep attribute learning for Object retrieval based on object ontology, and deep neural network. With improving imbalanced data and multitask learning, the model gained state-of-the-art on the DeepFashion dataset.

In this paper, we followed [52] to build human attribute learning based on HAO and deep learning. We tried to use transfer learning for feature extraction by using a CNN EfficientNet [18] for the feature extraction step and with the support of HAO and a linear classifier, our attribute learning model gained some state-of-the-art on PETA and PA100k datasets.

2. Image retrieval based on attribute learning

The image retrieval system tries to find candidates that are similar to the image input (query). In the traditional model, most studies used the similarity of feature vectors of candidate images and query images to collect results. Recently, to improve the accuracy of the image retrieval system, studies built image retrieval systems based on attribute learning of objects in images. It is the fine-grained approach to make the retrieval more intelligent than the old system. Because it could recognize attributes of objects and retrieve the attributes, instead of comparing feature vectors in low-level with blind semantics of images. Using deep learning for building attribute learning in retrieval systems is a robust step.

Recently, Person Re-Identification based on attribute learning has developed progress. Some popular works that used deep learning for attribute learning in Person Re-ID are [17, 44, 45]. In [12], Yutian Lin et al. (2019) built a person re-identification and attribute recognition share a common target for learning pedestrian descriptions. They proposed an attribute-person recognition (APR) network and achieved the state-of-the-art on Market-1501.

The authors in [53] proposed SAL (Symbiotic Adversarial Learn) by using two GANs to build the model of person search and gained high results in the PETA and Market1501 datasets. In [35], the authors proposed the UPAR to unique the pedestrian attribute recognition and person search. They used ConvNeXt [54] to build Attribute recognition and build person search based on attributes, and gained state-of-the-art on four datasets Market1501, Peta, Pa100k, and Rapv2.

- Text-based query for Image Retrieval

Most image retrieval uses image query. Some previous works admit using text query for image retrieval without the caption of images by annotation.

In [55], Zhe Wang et al. built a person search named ViTAA(Visual-Textual Attributes Alignment) by natural language. They tried to match specific attribute phrases to the corresponding visual regions with attribute learning support. The model achieved state-of-the-art performances.

In [49] Qi Dong et al. proposed a person search method using a text attribute query. They built an Attribute-Image Hierarchical Matching (AIHM) model to match text attribute descriptions with noisy surveillance person images by jointly learning global category-level and local attribute-level textual-visual embedding as well as matching. The AIHM model gained state-of-the-art on Market-1501, DukeMTMC, and PA100K datasets.

In [56], Tinghuai Ma et al. proposed a novel Dual-path CNN with MaxGatedblock(DCMG) for text-based person re-identification. The model is based on two deep residual CNNs and gained state-of-the-art on the CUHK-PEDES dataset.

In [57], Sijin Wang et al. built an Image-text retrieval of natural scenes by using graph matching. Focusing on two kinds of visual concepts, objects, and their relationships, they proposed to represent image and text with two kinds of scene graphs: visual scene graph (VSG) and textual scene graph (TSG). The model achieved state-of-the-art results on Flickr30k and MS COCO.

In [58], the authors proposed ASMR (Adaptive Semantic Margin Regularizer) for Attribute-based person search by building a network with a special loss function in embedding space of images and semantics of images as attributes or person categories. The study gained high results on datasets PETA, PA100k, and Market-1501.

3. Image retrieval based on attribute learning and ontology

Ontology is a tool to represent knowledge for sharing and reusing [19]. Especially. It is useful to support the semantics of images in building applications for image retrieval image interpretation [52, 59–62]. In attribute learning, ontology supports building the hierarchical semantic tree for facial attributes [20, 63, 64]. In [63], Garcıa-Rojas et al. built an ontology for facial representation for emotional face expression profiles. Combining deep imbalanced data learning and ontology, Hung M Nguyen et al. [20] proposed a facial attribute ontology (FAO) for attribute learning and achieved high accuracy with 40 facial attributes on CelebA and LFWA datasets. Focusing on aesthetics, Mingliang Xu et al. [21] suggested Bio-FAO (Bio-inspired Facial Aesthetic Ontology) and combined it with CNN to predict facial aesthetics. Using fuzzy reasoning, Renan Contreras et al. [63] proposed an emotion recognition method based on ontology. In [52], authors Ngoc Q. Ly et al. proposed an attribute learning model based on ontology and attribute learning for person re-identification on the Market-1501 dataset and object retrieval on the DeepFashion dataset.

In our paper, we used deep learning and ontology to build an image retrieval based on attributes. By combining deep learning and ontology, our image retrieval has increased accuracy and effectiveness. In the next part, we will show the design of Human Attribute Ontology in section III, the attribute learning in section IV, the human image retrieval in section V, the experiments in section VI and results in section VII, and the conclusion in section VIII.

III. HUMAN ATTRIBUTE ONTOLOGY

In this paper, we propose a novel ontology named Human Attribute Ontology (HAO) that facilitates the organization and storage of knowledge related to human objects in the real world.

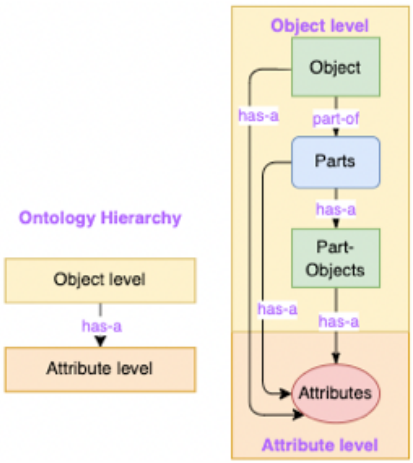


Figure 1: HAO with Hierarchical concepts in general

We utilize attribute learning and ontology to construct a human retrieval system operating at the attribute level. The ontology plays a crucial role in facilitating both the attribute learning stage and the retrieval stage. The reason for using ontology in the retrieval system is to (1) Reuse prior knowledge by rules and hierarchy in the ontology will support the attribute learning step; (2) Decrease the gap between the semantics of images from low-level features and high-level semantics of images, the ontology is to connect concrete images in the real world to the semantics of images in the ontology.

1. Ontology definition

Our HAO has five components with (C, P, Re, Ru, I):

- C (Concepts): They include names of objects, parts of objects, and attributes of objects. An example of the hierarchy of concepts is represented in the Figure 1 and Figure 2
- P (Properties): Each concept has a set of properties for describing itself.
- Re (Relations): A set of relationships of the concepts.
- Ru (Rules): A set of rules for reasoning concepts in the system.
- I (Instances): A set of Images with objects represented knowledge in HAO.

There are some definitions for constraining between concepts in the HAO:

- Definition 1: Each object has one part or some parts that are made in the object. The whole part is the biggest part of the object. One object has one whole part. For example, a human object has head part, top part, bottom part, foot part and whole part.
- Definition 2: Each part has one or many objects inside the part and is called PartObject. Each part may be

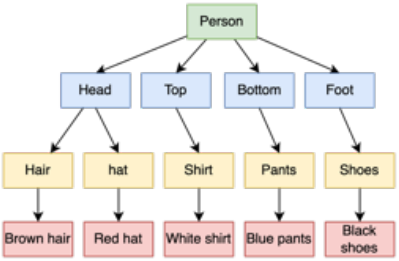


Figure 2: An example of hierarchical concepts in HAO

included null PartObject. For example, the top part of the person has a shirt object; the head part has a hat object.

- Definition 3: Each part has some attributes from itself or attributes of PartObjects in the part. And then each object has some attributes from its Parts and PartObjects.
- Definition 4: Each attribute belongs to one attribute group. For example, Asian is in the race group, and yellow skin is in the physical appearance group.
- Definition 5: There are two types of attributes, they are (1) Common type (also called visual concept type or general type) such as color, shape, texture, and size; (2) Specific type for character of each object. For example, a person object has a specific type: emotion, race, age, and gender. Fashion objects include material, office clothing, and party clothing.
- Definition 6: Visual concepts such as color, shape, and texture have a role in describing objects and attributes, they are in properties P of HAO.

2. An example of HAO

In this paper, we focused on human objects and human attributes to build HAO. We collect concepts about humans and relevant concepts about humans to build the ontology.

a) Concepts

Example of concepts of human objects and its hierarchy in Figure 1, Figure 2, Table I, Table II

b) Relations

There are relations between concepts of human object in the Table III.

c) Rules

A Rule has the following form:

$$Input \xrightarrow{\text{Condition}} Output$$

Each rule is defined by a tuple:

$$(Inputs, Outputs, Conditions).$$

TABLE I: CONCEPTS: OBJECTS AND PARTS.

Concepts	Type/Description	Context
Person	Object	Person
Fashion	Object	Fashion
Face	Object	Person
Shirt	Object	Person, Fashion
Pant	Object	Person, Fashion
Hat	Object	Person, Fashion
Head	Part of Person	Person
Top	Part of Person	Person
bottom	Part of Person	Person
Foot	Part of Person	Person
body	Part of Person	Person
wholepart	Part of Person	Person

TABLE II: CONCEPTS: ATTRIBUTES.

Attributes	Objects	Parts	Visual concepts	Groups
Blue jean	Person	bottom	Color	Fashion
Tall	Person	Whole	Size	Height
Short	Person	Whole	Size	Height
Thin	Person	Whole	Size	Body type
Fat	Person	Whole	Size	Body type
Bright skin	Person	Whole	Color	Complex
Straight hair	Person	Head	Texture	Hair style

Example rule: ”smile”→ ”happy” with null condition with a tuple (”smile”, ”happy”, null), ”long hair” → ”female” with rate=90% with a tuple ”long hair”, ”female”, rate=0.9). default of rate=1.0. Part= ”top” and color= ”red” → ”upred”.

There are some rules in the HAO as follows in the Table IV. Our HAO gains some characteristics in the standard of ontology: systematic, flexible, and extensible. With the knowledge in HAO, it can support some functions as follows:

- Support semantics of images by concepts hierarchy in HAO
- Support reasoning by rules in HAO to gain new knowledge without learning.
- Reuse prior knowledge: How many attributes are for training?

In this paper, we use HAO for support attribute learning in the following functions: (1) Support attribute training and matching for decreasing time and data space search Attribute for fine-grained learning, representation, and searching. (3) in the retrieval stage, HAO helps with preprocessing queries and reranking results of the retrieval system.

In the next section, we present the Attribute Learning

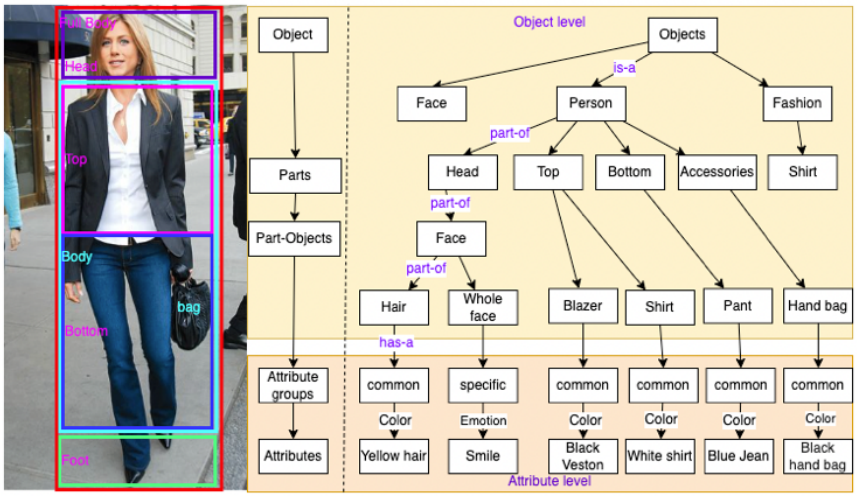


Figure 3: An example of HAO

TABLE III: Relations.

Relations	Examples
Is-a	Person ”is-a” object (sub-object and parent-object)
Has-a	Person “has-a” Face
Part-of	Head “part-of” Person (component)
Group-of	Black hair “group-of” black color, physical appearance, Asian race, Black hair “belong-to” physical appearance
Same-as	black skin “same-as” Dark skin, boy same-as male
Belong-to	Red “belong-to” Color, Color “has-a” red, Attribute “belong-to” part, Part ”belong-to” object, Attribute “belong-to” attribute-group

TABLE IV: RULES.

Rule ID	Input	Output	Condition
R1	Smile	Happy	Null
R2	Long hair	female	Rate=0.9
R3	Mustache	Male	
R4	Bushy Beard	Male	
R5	Red dress	Young girl	Rate=0.9
R6	Veston	Formal	
R7	backpack	young style	

method that used deep learning and HAO support to improve accuracy of the human attribute recognition.

IV. ATTRIBUTE LEARNING

Attributes are a mid-level between the low-level feature to the high-level semantics of images [9]. In our system, Attribute Learning (AL) is a very important module for the representation of images. It is a middle step for the retrieval system to transform from a raw image to an attribute vector

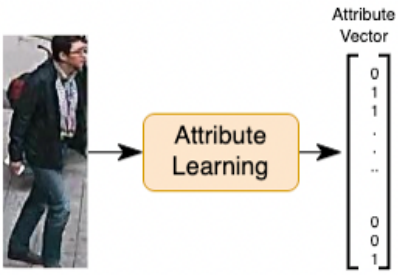


Figure 4: Attribute learning model in general

(shown in Fig. 4), that is conveying important information about the image at the semantic level to improve the image understanding capacity of the retrieval system.

The results of attribute learning help our retrieval to recognize images based on attributes. It shows some advantages: (1) the system is a fine-grained approach, can understand the image in detail and then increase accuracy, and (2) the system is more intelligent because it can retrieve on attributes of the object instead of based on low-level features in traditional ways or the whole object only.

In formal statements of attribute learning as follows:
Let $X = \{x_1, x_2, \ldots, x_N\}$, a set of images of the dataset, N is the number of the images.
Let $A = \{a^1, a^2, \ldots, a^M\}$, a set of M attribute labels of the system. Let $Y = \{y_1, y_2, \ldots, y_N\}$, a set of attribute labels of images, y_i is a binary vector that indicates the presence with 1 and absence with 0 of a particular attribute in the image (img), $y_i \in \{0,1\}^m$, $y_i = \{y_{i,1}, y_{i,2}, \ldots, y_{i,M}\}$, with:

$$y_{i,j} = \begin{cases} 1 \text{ if attribute } j^{th}\ (a^j) \text{ presents in img } i^{th} \\ 0 \text{ otherwise.} \end{cases} \quad (1)$$

Our method consisted of three main steps. They are (1)

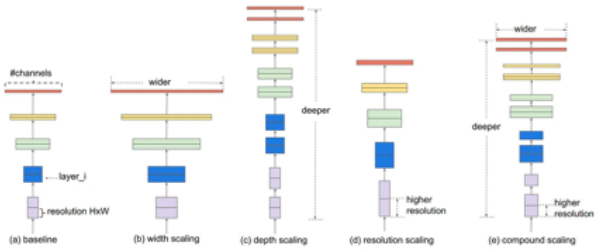


Figure 5: EfficientNet [65]

feature extraction, (2) Partition, and (3) classification step:

- Feature Extraction (FE)

In the FE step, we used EfficientNetB5 [65] for extracting a feature vector length of 2048. With HAO knowledge, we separate the feature vector into parts of the object that are declared in the ontology and collect the attribute list of each part, meaning which attributes are in which part, and then input them to the next step which is the attribute classification step.

EfficientNet [65] is the CNN, that used a technique of scaling up width, and depth to build the model. The authors of EfficientNet developed seven models, named B1 to B7. In the topic, we used EfficientNet-B5 for attribute learning in the feature extraction step with a 2048-length of the feature vectors.

- Part-based attribute learning: Connecting HAO and AL

We build AL based on the part objects with the support of ontology HAO. With the support of HAO, our attribute learning approach is based on parts of the object. From the part information of object in HAO, we get the part information and divide the image object into a number of parts corresponding to the part of the object in HAO. For example, with human object, there are five parts of the human object, they are head part, top part, bottom part, foot part, and whole part.

With each part, we get the list of attributes of the part and input them for the training. For example, with the top part, there are attributes in the top part such as shirt and attributes of shirt: red shirt, black shirt, ... With part and attribute in part gained from the HAO, they are input for the training in the next step.

Part-based attribute learning will be effective: (1) to decrease the cost for the training instead of training on the whole part; (2) Each attribute will be determined by a part of an object to avoid training with attributes and parts are not relevant; (3) attributes are grouped on part and hierarchy, then it is not the same level difference from attribute 1 to attribute m make a list for input training.

After the partition step, the next step is the training step for attribute classification.

- Attribute classification

In the classification step, we adopted the attribute learning model [12] as a multi-label classification task also multiple binary classifiers with Binary Cross-Entropy loss, SGD optimization, and sigmoid function for the probability of the result attribute vector are used. The loss function of the step is as follows:

$$L_{CLS} = \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M}\omega_j(y_{i,j}\log(\sigma(p_{i,j})) + (1-y_{i,j})\log(1-\sigma(p_{i,j}))) \quad (2)$$

With

$$\sigma(z) = \frac{1}{1+e^{-z}}$$

and $p_{i,j}$ is the output of the classifier layer, and ω_j adopted from [12] for imbalanced data between attributes with the formula (3):

$$\omega_j = \begin{cases} e^{1-r_j}, y_{i,j} = 1 \\ e^{r_j}, y_{i,j} = 0 \end{cases} \quad (3)$$

With r_j is the positive sample ratio of j^{th} attribute in the training set.

- Multitask learning: Connecting HAO and AL

Multitask Learning (MTL) is an advanced machine learning technique to improve performance [66] by sharing common data and combining related tasks to train together. There are two stages in the MTL, the shared stage and the separate stage.

In multi-task learning, We build MTL for attribute learning with two stages: a shared stage and a separate stage. In the shared stage, we used a CNN (EfficientNet) for feature extraction of images. After that, the feature map of the previous step will be partitioned into parts (Head, Top, Bottom, Foot) corresponding to parts of human information in HAO (shown in Figure 6). The feature vector of each part will be input into the next step. In the separate stage, we build attribute classifiers of attributes following attribute groups (gender, race, ...) corresponding to attribute groups and attribute lists on each part of human information in HAO. A summary of MTL of attribute learning on the Head part is shown in Figure 7.

- Loss function

Loss function in the attribute learning stage includes loss of FE and loss of classification, as shown in formula (4).

$$L_{AL} = L_{FE} + L_{CLS} \quad (4)$$

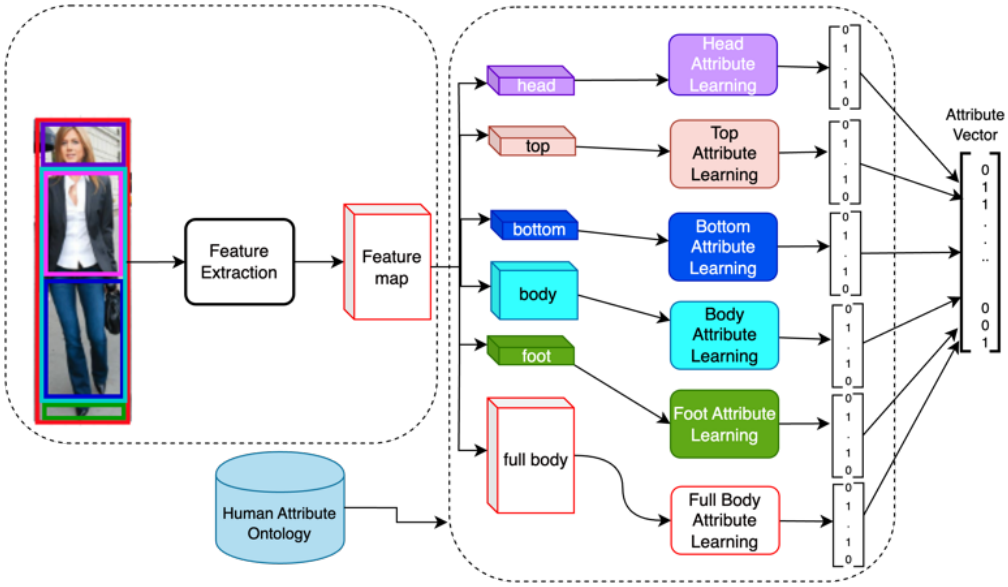


Figure 6: Attribute learning model in Detail

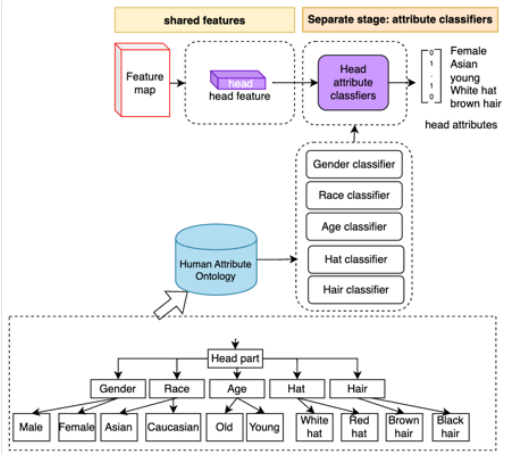


Figure 7: A summary MTL of AL on Head part

With L_{FE} is the loss of feature extraction method, L_{CLS} is the loss of classification. In the next section, we will present the retrieval system that the main task of the topic.

V. HUMAN IMAGE RETRIEVAL BASED ON ONTOLOGY AND ATTRIBUTE LEARNING

In this section, we present our human image retrieval system based on HAO and AL. The system comprises two stages: the offline stage and the online stage. The offline stage includes steps: the attribute learning and collecting attribute vectors of the dataset and saving them to the attribute vector database. The online stage has two steps: (1) Query set and gallery set, and (2) searching with matching, filtering and Ranking for the results of the retrieval. The model of the retrieval is shown in Figure 8.

- Query set and gallery set

In the offline stage, after the training step, we collected the parameters of the model and computed attribute vectors of images in the test set. The attribute vectors will be stored in the attribute vectors database. We divided the test set into query set Q and gallery set G that satisfies: $Q \cap G = \emptyset$ With an image query input $q \in Q$, we used the attribute learning model to infer the attribute vector of the query: aq. It is also in the attribute vector database if the query is in the test set. After gaining the attribute vector of query, we used it as input for the next step which is the matching and filtering step.

- Searching: Matching and filtering

In the step, from the attribute vector of the query in the previous step, we input it into the matching and filtering module by comparing the attribute vectors in the attribute vector database in the offline stage. We used the $L2$ measurement for comparing the distance of the query and the attribute vectors of the gallery in the database, to get the top 10 smallest distances. It returns the result of the retrieval. The search algorithm presents the method for matching and filtering. The algorithm is presented in Algorithm. 1.

VI. EXPERIMENTS

1. System configuration and datasets

We used Google Colab Pro for experiments with the programming language Python 3.9 with libraries such as Torch,

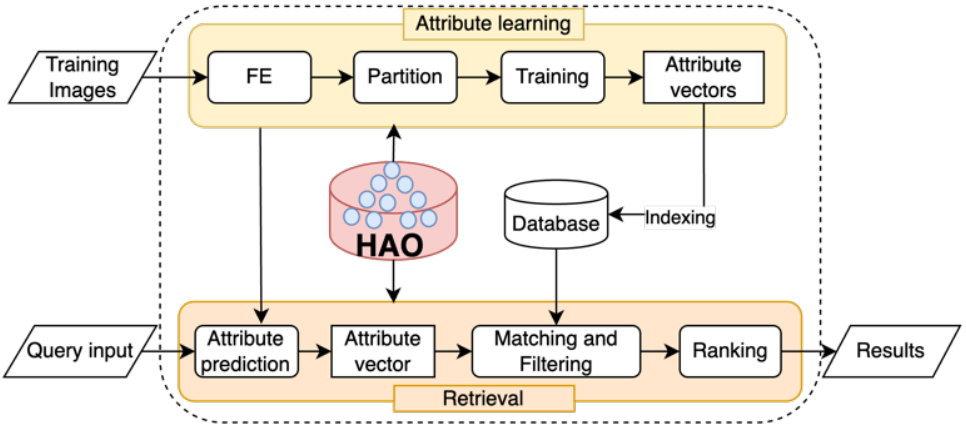


Figure 8: Human retrieval system

Algorithm 1: Search Algorithm

```
Inputs: image query q, image galleries G.
Outputs: top-10 image results.
begin
    aq=attributePredicting (q); //get attribute vector of q
    aG=attributePredicting (G);//get attribute vectors of G
    dis=distance(aG,aq); //computing distance attribute
      vectors of aG and aq
    candidates=argSort(dis); //sort the best candidates go
      to the head list
    result=getTopK(candidates); //get top-10 candidates
    Return result;
end
```

TABLE V: DATASETS.

Datasets	No. Img	No. Atts	Train set	Val set	Test set
PETA	19,000	35	9,500	1,900	7,600
Pa100k	100,000	26	80,000	10,000	10,000

keras, tensorflow, numpy, and OpenCV. We demonstrated our proposed methods on a large Re-ID dataset named PETA [67], and PA100k [36]. The PETA (PEdesTrian Attribute) dataset contains 19,000 indoor and outdoor images with 35 attributes. The PA100k (pedestrian attributes) dataset consists of 100,000 outdoor pedestrian images with 26 attributes. The information on datasets is shown in the Table V.

2. Ontology building

In the experiment of building HAO, we focused on the human knowledge domain with three relevant objects: Person (Pedestrian), Fashion, and Face object for HAO. We described Person object as follows the Figure 3. with parts: head, top, bottom, foot, wholebody. Each part has a set of attributes.

3. Attribute learning

In attribute learning, we use transfer learning EfficientNetB5 [65] for feature extraction by resizing the input image to 224x224. The result of the step is to return a feature vector of 2048 lengths (the default length of the feature vectors of EfficientNetB5). After that, we used the information about parts of objects in HAO to divide the feature vector into the number of parts of objects. In each part, we used a classifier for them. In this case, we used a Linear classifier combining the sigmoid function of the Torch library for classifiers. In the training step, we used SGD optimization, Binary Entropy Loss by following the experiment of author Yutian Lin [12]. We trained 40 epochs and a learning rate of 10^{-4}. We evaluate our attribute learning with attributes to follow each dataset. The result of AL will be shown in the next section.

Partitition is depended on the object in experiments. In the case, without loss of generality, we divided an image (as an object) into 4 segments lengthwise with the rate: the head part(1/6), the top part(3/8), the bottom part(2/6), the foot part(2/6), and the wholepart (6/6). Each part will input to the Attribute training step.

4. Image retrieval

With the query set and gallery set, we implemented the search algorithm (in Algorithm 1) and gained the results. We separated into three cases for the retrieval stage belonging to the organizing query set and gallery set as follows Table X. In case 1, we filtered images from the test set with each person having greater or equal 10 images, in case 2 with each person having greater or equal 5 images, and in case 3 with 500 random images from the test set for query set, and the remaining images of the test set for

gallery set. The results of the retrieval are shown in the next section.

VII. RESULTS

1. Evaluation Metrics

In the evaluation metric, we followed the work [68] with measurements: Mean Accuracy (mA), Accuracy (Acc), Precision (Prec), Recall (Rec), F1-score. In which, mA is as follows the equation (5):

$$mA = \frac{1}{2N}\sum_{i=1}^{M}\left(\frac{TP_i}{P_i} + \frac{TN_i}{N_i}\right), \quad (5)$$

where N is the number of examples and M is the number of attributes; P_i and TP_i are the number of positive examples and correctly predicted positive examples of the i^{th} attribute respectively; N_i and TN_i are defined for negative cases. The measurements: Accuracy (Acc), Precision (Prec), Recall (Rec), F1-score have display in [68] in detail.

In the evaluation of the retrieval system, we used the metric Top-k with k={1, 5, 10}, and Mean Average Precision (mAP)[68]. Top-k is the true positive in fisrt k results, it is also the rank-k accuracy.

2. Attribute learning

We compare our AL method with and without using HAO to popular studies about attribute learning on two datasets PETA and Pa100k. The results are shown in Table VI and Table VII.

The results of 35 attributes on PETA and Pa100k datasets in Table VI and Table VII were not on top 1, but they kept sTable With the support of ontology HAO and attribute multitask learning, the results are effective in the retrieval stage. Besides, we evaluated 6 attributes on the PETA dataset and gained high results in the Table VIII and Table IX.

3. Image retrieval

we experimented on the PETA dataset with cases in the Table X, our results in the Table XI, and compared to some popular studies and gained high results on the PETA dataset as follows Table XII. An example is in Figure 9 with query is in blue box.

The same experiment on the Pa100k dataset is shown in the Table XIII, our method results in Table XIV, and compared to popular studies in the Table XV.

Our results showed that the person search based on attributes effectively, not only finding out the person in the query image but also finding out people with the same attributes of the query. For example in the Figure 11 and 12. (the query is in the blue box).

TABLE VI: AL RESULTS OF 35 ATTRIBUTES ON THE PETA DATASET.

Topics	ma	Acc	Prec	Rec	F1-score
Zichang Tan [33]	84.88	79.46	87.42	86.33	86.87
HydraPlus-Net [36]	81.77	76.13	84.92	83.24	84.07
ALM [37]	86.3	79.52	85.65	88.09	86.85
Strong Baseline [38]	85.11	79.14	86.99	86.33	86.09
Jian Jia 2021 [39]	86.52	78.95	86.02	87.12	86.99
feature pyramid [40]	87.69	81.2	87.59	89.2	88.32
DAFL[41]	87.07				86.4
JLAC [42]	86.96	80.38	87.81	87.09	87.45
Rein-PAR[43]	85.51	78.45	84.08	88.77	85.91
Dai[34]	88.24	79.14	88.79	84.7	86.7
UPAR [35]	88.40				89.9
Ours:AL+w/o HAO	85.50	79.50	87.32	86.45	86.60
Ours:AL+HAO	85.41	80.17	88.09	86.59	87.07

TABLE VII: AL RESULTS OF 26 ATTRIBUTES ON THE PA100K DATASET.

Topics	ma	Acc	Prec	Rec	F1-score
Zichang Tan [33]	81.61	78.89	86.83	87.73	87.27
HydraPlus-Net [36]	74.21	72.19	82.97	82.09	82.53
ALM [37]	80.68	77.08	84.21	88.84	86.46
Strong Baseline [38]	79.38	78.56	89.41	84.78	86.25
Jian Jia 2021 [39]	81.87	78.86	85.98	89.1	86.87
feature pyramid [40]	81.45	79.13	86.24	89.46	87.94
DAFL[41]	83.54				88.09
JLAC [42]	82.31	79.47	87.45	87.77	87.61
Rein-PAR[43]	80.55	77.2	84.76	87.67	85.7
Dai[34]	77.89	79.71	90.26	85.37	87.75
UPAR [35]	84.80				90.2
Ours:AL+w/o HAO	82.49	79.89	87.39	88.32	87.49
Ours:AL+HAO	75.46	75.34	88.86	81.34	84.42

TABLE VIII: AL RESULTS: ACCURACY OF TOP 6 ATTRIBUTES ON THE PETA DATASET.

Attributes	Acc	Prec	Rec	F1-score
lowerBodyCasual	94.22	96.13	97.94	97.03
upperBodyCasual	93.72	95.87	97.66	96.76
personalLarger60	90.19	98.14	91.76	94.84
accessoryMuffler	89.97	97.33	92.25	94.72
personalMale	89.16	93.73	94.82	94.27
accessoryNothing	88.80	93.41	94.74	94.07
Average	91.01	95.77	94.86	95.28



Figure 9: A result top-10 in the best case on the PETA dataset



Figure 10: A result top-10 in the best case on the Pa100k dataset



Figure 11: Example search person based on attributes: ✓accessoryNothing, ✓upperBodyCasual, ✓upperBodyShortSleeve, ✓upperBodyTshirt, ✓lowerBodyCasual, ✓lowerBodyJeans, ✓footwearShoes, ✓carryingNothing, ✓personalLess30



Figure 12: Example search person based on attributes: ✓accessoryNothing, ✓upperBodyPlaid, ✓upperBodyShortSleeve, ✓lowerBodyCasual, ✓lowerBodyJeans, ✓footwearShoes, ✓carryingNothing, ✓personalLess30, ✓personalMale.

- Discussion

In the dataset PETA, there are over 4000 persons who have one image/person, which is why the random case for query (case 3 in Table XI) is the worst.

Our results in case 1 and case 2 also show that: with attribute learning, we could re-identification a person without training person-ID. It improves the performance of the system because it eliminates a training stage for the Person Re-Identification task.

Our result on the Pa100k dataset is not better than on the PETA dataset because the number of images in PA100k is bigger. It is a disadvantage for improving in the future works.

VIII. CONCLUSION

This topic is to improve the image understanding capacity of image retrieval systems. First, we developed HAO for storing the knowledge of images. Second, we built an MTL

TABLE IX: AL RESULTS: COMPARING THE ACCURACY OF THE TOP 6 ATTRIBUTES ON THE PETA DATASET.

Attributes	SAR [69]	MAR [69]	Ours	+/-
lowerBodyCasual	81.60	84.90	94.22	+
upperBodyCasual	81.10	84.40	93.72	+
personalLarger60	92.00	94.80	90.19	-
accessoryMuffler	94.40	96.10	89.97	-
personalMale	85.10	89.90	89.16	-
accessoryNothing	81.50	85.80	88.80	+
Average	85.95	89.32	91.01	+

TABLE X: CASES OF QUERY AND GALLERY SET IN THE PETA DATASET.

Cases	description	query set	gallery set
1	10 images/person	144	2313
2	5 images/person	250	2953
3	500 random images	500	7100

TABLE XI: 3 CASE RESULTS ON THE PETA DATASET

Cases	Top-1	Top-5	Top-10	mAP
1	92.36	97.22	97.22	94.98
2	86.00	92.80	93.20	87.66
3	40.60	48.30	49.80	40.39
Average	72.99	79.44	80.07	74.34

TABLE XII: RESULTS IR ON THE PETA DATASET

Cases	Top-1	Top-5	Top-10	mAP
UPAR [35]	29.70			30.20
SAL [53]	47	66.50	74	41.50
ASMR [58]	56.50	80.00	83.50	50.20
Ours (avg)	72.99	79.44	80.07	74.34

TABLE XIII: CASES FOR RETRIEVAL IN THE PA100K DATASET.

Cases	description	query set	gallery set
1	10 images/person	191	8917
2	5 images/person	248	9244
3	500 random images	500	9500

attribute learning with the support of HAO and deep neural networks. Finally, we constructed the original HR system based on HAO and AL. The retrieval could understand human images in detail at the attribute level, and be more intelligent. We demonstrated that the performance of our retrieval system was better than some state-of-the-art on

TABLE XIV: 3 CASE RESULTS ON THE PA100K DATASET

Cases	Top-1	Top-5	Top-10	mAP
1	65.44	67.53	69.63	25.35
2	62.90	64.51	66.12	15.15
3	19.40	33.60	43.60	7.22
Average	49.25	55.21	59.78	15.91

TABLE XV: RESULTS IR ON THE PA100K DATASET

Cases	Top-1	Top-5	Top-10	mAP
UPAR [35]	39.50			30.50
ASMR [58]	31.9	49.1	58.2	20.6
AIHM [49]	31.3	45.1	50.0	17.0
Ours (avg)	49.25	55.21	59.78	15.91

datasets PETA and Pa100k.

In the future, we could use the rule component of HAO for attribute reasoning based on rules. This helps AL decrease the costs of training and improve the performance of the system because no need for training the reasonable attributes from the rules. With rule-based reasoning, we can infer new knowledge from prior knowledge without training. The reasoning module also supports detecting the relation between an attribute and relevant attributes in an object, an object, and relevant objects. HAO and reasoning modules based on rules will help to reuse prior knowledge and close the semantic gap between low-level features and high-level semantics of images.

ACKNOWLEDGEMENTS

This research is supported by research funding from the Faculty of Information Technology, University of Science, Vietnam National University-Ho Chi Minh City, Saigon University, and by the JSPS Core-to-Core Program (grant number: JPJSCCB20230005).

REFERENCES

[1] E. Yaghoubi, F. Khezeli, D. Borza, S. Kumar, J. Neves, and H. Proença, "Human attribute recognition-a comprehensive survey," *Appl. Sci*, 2020.

[2] S. Dubey, "A decade survey of content based image retrieval using deep learning," 2021.

[3] M. Kalayeh, E. Basaran, M. Gokmen, M. Kamasak, and M. Shah, "Human semantic parsing for person re-identification," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit*, p. 1062–1071, 2018.

[4] T. He, X. Shen, J. Huang, Z. Chen, and X.-S. Hua, "Partial person re-identification with part-part correspondence learning," *Cvpr*, p. 9105–9115, 2021.

[5] J. Liu, Z.-J. Zha, W. Wu, K. Zheng, and Q. Sun, "Spatial-temporal correlation and topology learning for person re-identification in videos," *CVPR*, p. 4370–4379, 2021.

[6] Z. Zhang, H. Zhang, and S. Liu, "Person re-identification using heterogeneous local graph attention networks," *Cvpr*, p. 12136–12145, 2021.

[7] B. Nguyen, B. Nguyen, T. Do, E. Tjiputra, Q. Tran, and A. Nguyen, "Graph-based person signature for person re-identifications," 2021.

[8] D. Parikh and K. Grauman, "Relative attributes learning," p. 503–510, 2011.

[9] K. Grauman and B. Leibe, *Visual object recognition*, 2011.

[10] X. Wang, S. Zheng, R. Yang, B. Luo, and J. Tang, "Pedestrian attribute recognition: A survey," p. 1–32, 2019.

[11] J. Joo, S. Wang, and S. Zhu, "Human attribute recognition by rich appearance dictionary," *Proc. IEEE Int. Conf. Comput. Vis*, p. 721–728, 2013.

[12] Y. Lin, "Improving person re-identification by attribute and identity learning," *Pattern Recognit*, vol. 95, p. 151–161, 2017.

[13] G. Zhang and J. Xu, "Person re-identification by mid-level attribute and part-based identity learning," *Proc. Mach. Learn. Res*, vol. 95, p. 220–231, 2018.

[14] S. Li, T. Xiao, H. Li, B. Zhou, D. Yue, and X. Wang, "Person search with natural language description," p. 5187–5196, 2017.

[15] H. Galiyawala and M. Raval, "Person retrieval in surveillance using textual query: a review," *Multimed. Tools Appl*, 2021.

[16] M. Ye, J. Shen, G. Lin, T. Xiang, L. Shao, and S. Hoi, "Deep learning for person re-identification: A survey and outlook," p. 1–20, 2020.

[17] C. Tay, S. Roy, and K. Yap, "Aanet: Attribute attention network for person re-identifications," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit*, vol. 2019-June, p. 7127–7136, 2019.

[18] M. Wieczorek, A. Michalowski, A. Wroblewska, and J. Dabrowski, "A strong baseline for fashion retrieval with person re-identification models," 2020.

[19] T. Gruber, "Toward principles for the design of ontologies used for knowledge sharing," *Int. J. Hum. - Comput. Stud*, vol. 43, no. 5–6, p. 907–928, 1993.

[20] H. Nguyen, N. Ly, and T. Phung, "Large-scale face image retrieval system at attribute level based on facial attribute ontology and deep neuron network," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics*, vol. 10752, p. 539–549, 2018.

[21] M. Xu, "Bio-inspired deep attribute learning towards facial aesthetic prediction," *IEEE Trans. Affect. Comput*, vol. 3045, no. c, p. 1–1, 2018.

[22] R. Feris, C. Lampert, and D. Parikh, *Visual Attributes*. Springer, 2017.

[23] V. Ferrari and A. Zisserman, "Learning visual attributes," *Adv. Neural Inf. Process. Syst*, vol. 20, 2008.

[24] A. Farhadi, I. Endres, D. Hoiem, and D. Forsyth, "Describing objects by their attributes," 2009.

[25] C. Lampert, H. Nickisch, and S. Harmeling, "Attribute-based classification for zero-shot visual object categorization," 2013.

[26] O. Russakovsky and L. Fei-Fei, "Attribute learning in large-scale datasets," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics*, no. PART 1, p. 1–14, 2012.

[27] N. Kumar, "Describable visual attributes for face images," 2011.

[28] Y. Fu, "Attribute learning for image / video understanding," *Queen*, 2015-03.

[29] E. Rudd, M. Günther, and T. Boult, "Moon: A mixed objective optimization network for the recognition of facial attributes," *LNCS Bioinforma*, p. 19–35, 2016.

[30] K. He, Z. Wang, Y. Fu, R. Feng, Y. Jiang, and X. Xue, "Adaptively weighted multi-task deep network for person attribute classification," *MM*, p. 1636–1644, 2017.

[31] D. Jayaraman, F. Sha, and K. Grauman, "Decorrelating semantic visual attributes by resisting the urge to share," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit*, p. 1629–1636, 2014.

[32] Z. Huo, Y. Xia, and B. Zhang, "Vehicle type classification and attribute prediction using multi-task rcnn," p. 564–569, 2016.

[33] Z. Tan, Y. Yang, J. Wan, H. Hang, G. Guo, and S. Li, "Attention-based pedestrian attribute analysis," *IEEE Trans. Image Process*, vol. 28, no. 12, p. 6126–6140, 2019.

[34] Y. Hu, X. Bai, P. Zhou, F. Shang, and S. Shen, "Data augmentation imbalance for imbalanced attribute classification," 2020.

[35] A. Specker, M. Cormier, and J. Beyerer, "Upar: Unified pedestrian attribute recognition and person retrieval," *Appl. Comput. Vision, WACV*, p. 981–990, 2023.

[36] X. Liu, "Hydraplus-net: Attentive deep features for pedestrian analysis," *Proc. IEEE Int. Conf. Comput. Vis*, no. c, p. 350–359, 2017.

[37] C. Tang, L. Sheng, Z. Zhang, and X. Hu, "Improving pedestrian attribute recognition with weakly-supervised multi-scale attribute-specific localization," *Proc. IEEE Int. Conf. Comput. Vis*, no. c, p. 4996–5005, 2019.

[38] J. Jia, H. Huang, W. Yang, X. Chen, and K. Huang, "Rethinking of pedestrian attribute recognition: Realistic datasets and a strong baseline," p. 1–12, 2020.

[39] J. Jia, X. Chen, and K. Huang, "Spatial and semantic consistency regularizations for pedestrian attribute recognition," *Proc. IEEE Int. Conf. Comput. Vis*, p. 942–951, 2021.

[40] M. Moghaddam, M. Charmi, and H. Hassanpoor, "Jointly human semantic parsing and attribute recognition with feature pyramid structure in efficientnets," *IET Image Process*, vol. 15, no. 10, p. 2281–2291, 2021.

[41] J. Jia, N. Gao, F. He, X. Chen, and K. Huang, "Learning disentangled attribute representations for robust pedestrian attribute recognition," *Proc. AAAI Conf. Artif. Intell*, vol. 36, no. 1, p. 1069–1077, 2022.

[42] Z. Tan, Y. Yang, J. Wan, G. Guo, and S. Li, "Relation-aware pedestrian attribute recognition with graph convolutional networks," 2020.

[43] Z. Ji, Z. Hu, Y. Wang, Z. Shao, and Y. Pang, "Reinforced pedestrian attribute recognition with group optimization," *SSRN Electron. J*, 2022.

[44] Y. Shi, Z. Wei, H. Ling, Z. Wang, J. Shen, and P. Li, "Person retrieval in surveillance videos via deep attribute mining and reasoning," *IEEE Trans. Multimed*, vol. 14, no. 8, 2020.

[45] J. Wang, X. Zhu, S. Gong, and W. Li, "Transferable joint attribute-identity deep learning for unsupervised person re-identification," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit*, no. 1, p. 2275–2284, 2018.

[46] Y. Men, Y. Mao, Y. Jiang, W. Ma, and Z. Lian, "Controllable person image synthesis with attribute-decomposed gan," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit*, p. 5083–5092, 2020.

[47] Y. Chen, S. Duffner, A. Stoian, J. Dufour, and A. Baskurt, "Pedestrian attribute recognition with part-based cnn and combined feature representations," *Jt. Conf. Comput. Vision, Imaging Comput. Graph. Theory Appl*, vol. 5, p. 114–122, 2018.

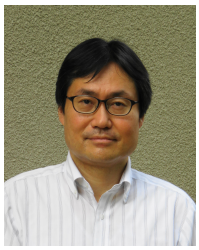
[48] X. Song, H. Yang, and C. Zhou, "Pedestrian attribute recognition with graph convolutional network in surveillance scenarios," *Futur. Internet*, vol. 11, no. 11, 2019.

[49] Q. Dong, X. Zhu, and S. Gong, "Person search by text attribute query as zero-shot learning," *Proc. IEEE Int. Conf.*

Comput. Vis, vol. 2019-Octob, p. 3651–3660, 2019.
[50] N. Sarafianos, C. Nikou, T. Giannakopoulos, and I. Kakadiaris, "Curriculum learning for multi-task classification of visual attributes," p. 2608–2615, 2017.
[51] S. Pei, "Multitask model for person re-identication by," 2021.
[52] N. Ly, T. Do, and B. Nguyen, "Large-scale coarse-to-fine object retrieval ontology and deep local multitask learning," *Comput. Intell. Neurosci*, vol. 2019, 2019.
[53] Y. Cao, J. Wang, and D. Tao, "Symbiotic adversarial learning for attribute-based person search," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics*, vol. 12359, p. 230–247, 2020.
[54] Z. Liu, H. Mao, C. Wu, C. Feichtenhofer, T. Darrell, and S. Xie, "A convnet for the 2020s," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit*, vol. 2022-June, p. 11966–11976, 2022.
[55] Z. Wang, Z. Fang, J. Wang, and Y. Yang, "Vitaa: Visual-textual attributes alignment in person search by natural language," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics*, vol. 12357, p. 402–420, 2020.
[56] T. Ma, M. Yang, H. Rong, Y. Qian, Y. Tian, and N. Al-Nabhan, "Dual-path cnn with max gated block for text-based person re-identification," *Image Vis. Comput*, vol. 111, p. 1–14, 2021.
[57] S. Wang, R. Wang, Z. Yao, S. Shan, and X. Chen, "Cross-modal scene graph matching for relationship-aware image-text retrieval," *Appl. Comput. Vision, WACV*, p. 1497–1506, 2020.
[58] B. Jeong, J. Park, and S. Kwak, "Asmr: Learning attribute-based person search with adaptive semantic margin regularizer," *Proc. IEEE Int. Conf. Comput. Vis*, vol. 1, no. 1, p. 11996–12005, 2021.
[59] N. Maillot, M. Thonnat, and C. Hudelot, "Ontology based object learning and recognition: Application to image retrieval," *Proc. - Int. Conf. Tools with Artif. Intell. ICTAI*, no. 1, p. 620–625, 2004.
[60] N. Maillot, "Ontology based object learning and recognition-thesis," 2008.
[61] V. Mezaris, I. Kompatsiaris, and M. Strintzis, "An ontology approach to object-based image retrieval," p. –511–14, 2004.
[62] C. Hudelot, "Towards a cognitive vision platform for semantic image interpretation; application to the recognition of biological organisms," 2008.
[63] R. Contreras, O. Starostenko, V. Alarcon-Aquino, and L. Flores-Pulido, "Facial feature model for emotion recognition using fuzzy reasoning," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics*, vol. 6256, p. 11–21, 2010.
[64] R. Bashar, S. Kang, P. Dawadi, and P. Rhee, "A context-aware statistical ontology approach for adaptive face recognition," p. 704–709, 2007.
[65] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," 2019.
[66] Y. Zhang and Q. Yang, "A survey on multi-task learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 34, no. 12, pp. 5586–5609, 2022.
[67] Y. Deng, P. Luo, C. Loy, and X. Tang, "Pedestrian attribute recognition at far distance," *Proc. ACM Int. Conf. Multimed*, vol. MM'14, p. 789–792, 2014.
[68] D. Li, Z. Zhang, X. Chen, and K. Huang, "A richly annotated pedestrian dataset for person retrieval in real surveillance scenarios," *IEEE Transactions on Image Processing*, vol. 28, no. 4, pp. 1575–1590, 2019.
[69] D. Li, X. Chen, and K. Huang, "Multi-attribute learning for pedestrian attribute recognition in surveillance scenarios," p. 111–115, 2015.



Ly Quoc Ngoc is currently Associate Professor and Head of the Computer Vision and Cognitive Cybernetics Department at the Faculty of Information Technology, University of Science, Ho Chi Minh City. He received his Ph.D in Computer Science from VNUHCM-HCMUS, Vietnam, in 2009. His research interests lie in the fields of Computer Vision, Digital Image and Video Processing, Computer Graphics, Artificial Intelligence, Multimodal Deep Learning, Medical Image Processing.
Email: lqngoc@fit.hcmus.edu.vn



Masayuki Fukuzawa is currently an associate professor of Faculty of Information and Human Sciences in Kyoto Institute of Technology, Japan. His specialty is image instrumentation and processing as multidimensional signal. His current research interests include image and video processing for clinical diagnosis, optical instrumentation of semiconductor crystals, and intelligent image sensors.
Email: fukuzawa@kit.ac.jp



Trang T.T. Phung is currently a lecturer in the Faculty of Information Technology at Saigon University (SGU), Vietnam. Her research interests lie in the fields of Computer Vision, especially in Image Retrieval and Image Classification.
Email: trangphung@sgu.edu.vn

Mellifluous Viewport Bitrate Adaptation for 360° Videos Streaming over HTTP/2

Nguyen Duc Thien[1], Tran Trieu Hai[2], Nguyen Viet Hung[1]
[1]Faculty of Information Technology, East Asia University of Technology, Bacninh, Vietnam
[2]Faculty of General Education - Hanoi Open University, Hanoi, Vietnam
Correspondence: Nguyen Duc Thien (e-mail: thiennd@eaut.edu.vn)
Communication: received 05 Jan 2024, revised 11 Mar 2024, accepted 11 Apr 2024
Digital Object Identifier: 10.32913/mic-ict-research.v2024.n2.1251

Abstract: Virtual reality provides an immersive experience for users. There has recently been much research on adaptive methods of playing 360° video over HTTP/2 to excite users. This paper analyzes users' behaviors and perceptions when watching 360° online videos. We also provide some measurements and evaluation of the main features of 360° video streaming. Experiments show that our method improves bitrate by up to 45.8% and buffer size counter-up to 25.0% compared to referenced methods.

Keywords: *VR, HTTP/2, 360° video, QoE, Viewport Adaptive Streaming.*

I. Introduction

Virtual classrooms are deployed more and more popularly, studied, and applied to many fields, especially in information technology. Virtual reality [1] or 360° video has been widely used in this field to meet human requirements in providing a realistic user experience, including learner experience, student satisfaction assessment, or even about the cost of streaming 360° video, which requires much higher bandwidth and a much higher 60fps frame rate [2–4].

Viewport Adaptive Streaming (VAS) applications were used to reduce bandwidth requirements. A VAS system that also delivers a portion of the currently available video to the user can result in a view with a higher bitrate in the provided area than in other segments [2, 5–7]. As a result, 360° video streaming will be scaled up based on the corresponding argument, as shown in Fig. 1.

On the other hand, since users tend to change their viewing direction when they are watching, this view will change over time during streaming sessions, or status will be updated to infer topics they are interested in. This may affect the viewer's perception described in [8]. Furthermore, bitrates in a VAS system should be adjusted to adapt to different views over time. Besides, studies [9, 10] have demonstrated when throughput being reduced due to network problems, users' viewing experience will be significantly reduced.

Besides, a large number of videos must be cached into versions to ensure smoothness when watching 360° videos in different network conditions, as shown in Fig. 1. Finally, the client will choose the version to download and play the video [9]. However, there is a limitation that the performance of VAS can be reduced due to the large buffer size [11]. Therefore, it is necessary to consider the viewport's adaptability and network condition.

In this paper, our objective is to assess the impact of users' head movement on existing methods, explicitly focusing on the following key aspects:

- Evaluation of current VAS methods: We aim to examine how existing VAS methods perform when there are variations in bandwidth and image quality uniformity within the users' field of view. By doing so, we intend to enhance the smoothness of video playback on the client side.
- Development of an adaptive algorithm: We propose an algorithm that dynamically adapts to changing network conditions, specifically in video streaming over HTTP/2 streams. This algorithm will ensure that the video streaming experience remains optimal by adjusting to fluctuations in available bandwidth.

By conducting this evaluation and developing the adaptive algorithm, we aim to contribute to improving video streaming systems, particularly in scenarios where head movement and varying network conditions can significantly impact the user experience. The remaining of this paper is the paper's content in the following sections.

The first presents related works in part II, part III details the proposed algorithm, and the fourth quarter evaluates



Figure 1: General chunk-based Viewport Adaptive Streaming system

the algorithm in part IV. Finally, the article concludes in part V.

II. Related work

In this section, we will briefly describe the context of some of the current HTTP video streaming methods of the scalable 360° video streaming method considered in the study.

1. Adaptive streaming over HTTP

Thanks to the strong development of information technology and 5G and 6G network systems today [12], 360° videos are becoming more popular. Moreover, 360° video requires a high resolution and viewport rate to cover an omnidirectional view. Therefore, in research [13], the authors propose an optimization method using video encoding layers, which increases the possibility of viewport adaptation over HTTP. Furthermore, research[2] shows that encoding and attention to regions for streaming are significantly improved to bring immersion to the viewer.

Besides, Streaming over HTTP has received much attention thanks to the Dynamic Adaptive Streaming over HTTP (DASH) feature. However, the development history is primarily private and not made public [14]. It is, therefore, difficult to compare. The paper [14] showed that using the open-source DASH method provides a basic assessment of the different segment lengths and their impact on the server. HTTP. According to a study [15], Streaming over HTTP (DASH) is a trusted standard for streaming video on the Internet because it provides a rich experience for users that the server side will support video streaming DASH-based content delivery.

2. Adaptive streaming over HTTP/2

With the growing popularity of HTTP/2 video streaming and the dominance of large amounts of video on the Internet, providers have moved toward high-quality video streaming. Besides, thanks to the development of HTTP/2, the protocol demonstrated that the multiplexing feature of HTTP/2 allows audio and video segments to be mixed in streaming [16, 17], bringing more solutions for video streaming over HTTP/2 protocol. The problem of mixed transmissions has been solved using monitoring techniques.

On the other hand, with data growth and high requirements from users, 360° video content is a big challenge [18], as they need to save bandwidth and optimize methods to improve stream quality and increase streaming speed. It is known that random transmission will slow down the process because it takes time to wait for processing. Therefore, researchers have taken advantage of providing more transmission lines to avoid user discomfort. HTTP/2 also shows flexibility for video streaming and improved QoE [19].

Furthermore, a method [20] has also shown that HTTP adaptive streaming plays a vital role in reducing video downtime and improving video quality by adjusting the quality of transmitted video segments according to the conditions of the network. The method also shows that buffering and bit rate factors are also important. Therefore, in this paper, we propose an algorithm to analyze and evaluate the quality of video streaming over HTTP/2 streams.

III. Proposed method

In this section, we propose an algorithm that uses a sorting algorithm to select view regions, improving bitrate and cache quality with the HTTP/2 protocol.

1. Version Selection

Figure 2 shows that choosing the version to stream video content depending on user requirements and network conditions is significant. Although different videos are

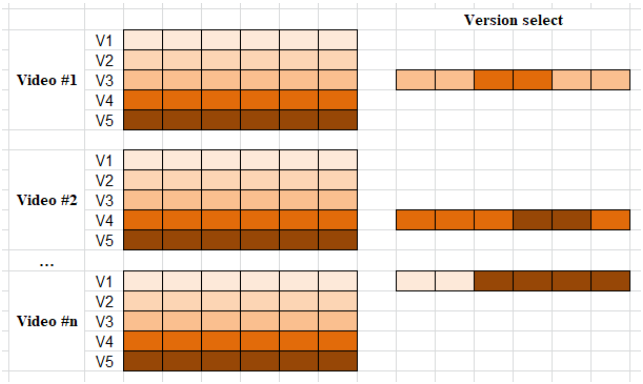


Figure 2: Selecting the streaming version

Algorithm 1: *Version Selection*

Input: V_n, W_c, R
Output: $\{v_n\}_{1 \le n \le N}$

```
while R > 0 do
    for v = 1 to V − 1 do
        if W_c < R then
            W_c ∈ V_n
            Calculate to Max W_c ≈ R.
            l_n ← l_n + 1;
        end
    end
end
return {v_n}_{1≤n≤N};
```

divided into different versions, from primary to high level, version transmission depends on network factors. Therefore, we hope that even when network conditions are limited, it still brings the best quality to viewers.

On the other hand, to perform such work, our method not only improves the user's viewport adaptation quality but also divides the video into segments. Each of them will have selected versions, which means that with good network quality conditions, the method that will achieve the best quality level, Version 5 in algorithm 1. However, the network's reality is always changing, and 360° video content is relatively large; studies [2] also show that the influence levels of each region also play an important role. We can improve the quality of important areas and limit unimportant regions. This means that areas within the mind will get much attention, and others will get less attention.

2. Chunk Selection

Research by the authors [21] has shown that high level caching is necessary. After dividing the versions, we analyze and divide them into Chunks. Each Chunk will have a different weight, corresponding to this weight is

Algorithm 2: *Chunk Weight Selection*

```
for c = 1 to C do
    pivot = Buffer[high]
    for low to high do
        Calculate Buffer[high];
        if Buffer[low]> pivot then
            Buffer[high] = Buffer[low]
            low = low + 1;
        end
    end
end
return Buffer[high];
```

TABLE I: Quantization Parameter (QP) of Chunk

Scalable	QP
Level of Enhancement of Chunk 01	38
Level of Enhancement of Chunk 02	32
Level of Enhancement of Chunk 03	28
Level of Enhancement of Chunk 04	24
Level of Enhancement of Chunk 05	20

the improved quality. In algorithm 2, we improved the calculation to fill the buffer head, which means that some chunks of version 4 can be used for the version.

For algorithm 2, we assume the desire to choose the best value named Buffer. To do this, we divide the versions into chunks as shown above; here, we conduct evaluation and analysis to select the corresponding threshold to achieve the best value, and the quick sort algorithm has been used. We calculate in this step. This method will not only improve time and help the system calculate better based on the buffer threshold, but also calculate and improve quality for regions with the largest number of viewers.

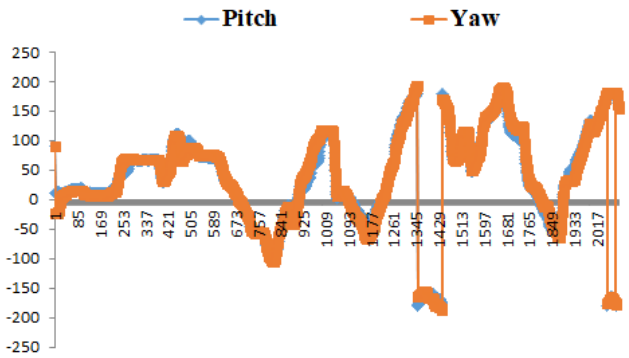
IV. Evaluation

1. Experimental Settings

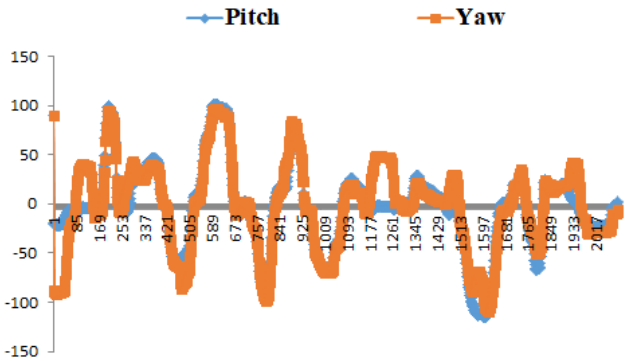
To evaluate the proposed solution, we will build a test system consisting of a server and a client; the server side

TABLE II: Chunk's average bitrate (kbps) for video experiments.

	Average chunk bitrate (kbps)		
Scalable Chunk	Roller Coaster	Diving	Venice
L. Enhance. Chunk 01	55	159	50
L. Enhance. Chunk 02	107	314	114
L. Enhance. Chunk 03	179	505	216
L. Enhance. Chunk 04	389	1037	484
L. Enhance. Chunk 05	633	1418	825

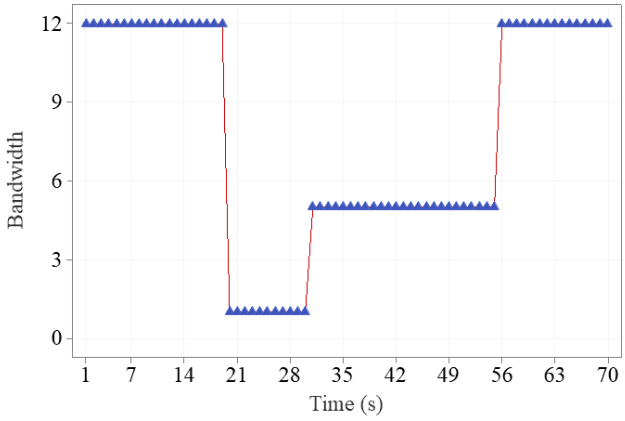


((a)) Head-Movement Trace 01

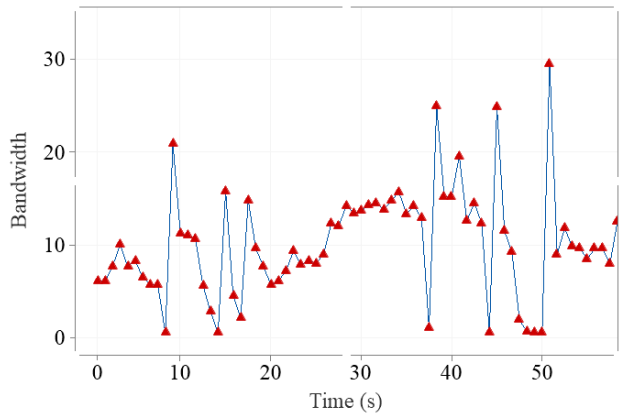


((b)) Head-Movement Trace 02

Figure 3: Head-movement traces under two simulations



((a)) Bandwidth 01



((b)) Bandwidth 02

Figure 4: Bandwidth traces under two simulations

will be analyzed and divided into five different versions. Experiments are run with varying simulations of evaluation.

- First, we divided the experiment into two different head movements in Fig. 3.
- Second, we experiment with actual bandwidth and our designed bandwidth to test the proposed algorithm further in Fig. 4.
- Third, we simulate a system algorithm that typically limits intentional selection and instead runs automatically.

On the other hand, we use three videos, Rollercoaster, Diving, and Venice from [22]. Each 360° video will be converted to CubeMap format using 360Lib software [23], with the viewport set to 70s. The videos will be divided into 24 chunks; each chunk has the same resolution of 480x480, and each video will be divided into five versions, with each version being V_n $(0 < n < 6)$, V_1 the lowest version, the highest version is V_5, Let R is the network bandwidth, and W_c the weight of each chunk $(1 \leq c \leq 24)$. We use the HEVC [24] standard for encoding. The parameters we calculated for testing are in Tables I and II.

2. Performance Analysis

In this section, we create a standard algorithm (called No) and an improved algorithm (called proposed) to analyze the evaluations and three referenced methods including WZUQI [2], GLVP [4], and BL [25] with the following descriptions:

- No: we perform the normal setting as the Proposed algorithm. However, No will choose the version corresponding to the network conditions without chunk selection, while our algorithm will calculate the chunk. The chunk value then arranges the chunks in order and selects the appropriate chunk based on network conditions.
- WZUQI [2]: WZUQI is the algorithm focusing on the video's central center. Therefore, for our problem, the limitation is that it is not fixed but moves in the direction of the head, so for this algorithm, with continuously changing motion, the algorithm results are relatively limited.
- GLVP [4]: GLVP is the algorithm that focuses on

TABLE III: Average bitrate (Kbps) of reference vs. the proposed method

Videos	Head movement trace #1									
	WZUQI		No		GLVP		BL		The Proposed	
	BW #1	BW #2	BW #1	BW #2	BW #1	BW #2	BW #1	BW #2	BW #1	BW #2
RollerCoaster	468,60	469,28	202.63	202.46	503.75	494.26	454.94	459.13	555.46	547.36
Diving	231.77	198.56	505.54	509.74	343.87	343.68	433.10	415.69	519.70	513.19
Venice	222.10	185.26	232.95	233.13	462.14	454.36	436.86	428.24	514.00	514.77
Average	**307.49**	**284.37**	**313.71**	**315.11**	**436.59**	**430.77**	**441.63**	**434.36**	**529.72**	**525,11**

TABLE IV: Average bitrate (Kbps) of reference vs. the proposed method

Videos	Head movement trace #2									
	WZUQI		No		GLVP		BL		The Proposed	
	BW #1	BW #2	BW #1	BW #2	BW #1	BW #2	BW #1	BW #2	BW #1	BW #2
RollerCoaster	444.09	**444.15**	189.78	189.60	495.06	486.46	408.12	396.11	543.55	508.92
Diving	304.20	478.48	451.29	478.48	285.94	293.70	367.58	362.88	545.82	478.93
Venice	282.81	233.56	218.30	218.22	442.29	434.93	366.22	362.99	460.16	479.06
Average	**343.70**	**385.39**	**286.46**	**295.43**	**407.76**	**405.03**	**380.64**	**373.99**	**516.51**	**488.97**

TABLE V: Average total buffer size (s) of all test videos of reference vs. the proposed method under Head-movement trace 01

Videos	Bandwidths	WZUQI	No	GLVP	BL	Proposed
RollerCoaster	01	4.683	4.991	6.409	6.349	6.409
	02	4.676	4.988	6.417	6.350	6,417
Diving	01	4.872	4.549	6.367	6.358	6.383
	02	4.778	4.529	6.329	6.328	6.341
Venice	01	4.699	4.890	6.169	6.143	6.169
	02	4.692	4.890	6.172	6.140	6.172
Average Buffer		**4.734**	**4.806**	**6.310**	**6.278**	**6.315**

TABLE VI: Average total buffer size (s) of all test videos of reference vs. the proposed method under Head-movement trace 02

Videos	Bandwidths	WZUQI	No	GLVP	BL	Proposed
RollerCoaster	01	444.094	189.777	495.061	408.120	543.547
	02	444.150	189.595	486.461	396.113	508.918
Diving	01	304.200	451.289	285.942	367.580	445.818
	02	264.239	478.475	293.704	362.878	478.934
Venice	01	282.812	218.301	442.287	366.219	460.162
	02	233.556	218.220	434.929	362.990	479.055
Average Buffer		**328.842**	**290.943**	**406.397**	**377.317**	**486.072**

improving and estimating the typical view of the VAS system. In general, the algorithm is the same as currently mentioned. Still, for the proposed method, in addition to estimating, the algorithm needs to focus on calculating load calculation and buffer filling.

- BL [25]: BL is a proposed tiling-based 360° video streaming method that maintains smooth viewport quality despite network bandwidth fluctuations and user head movements. BL dynamically adjusts the number of extension tiles based on viewport prediction errors to ensure a smooth viewport bitrate across video segments. In addition, the total bitrate allocated to tiles is chosen to accommodate sudden changes in network bandwidth. Finally, a tile selection algorithm helps to reduce intra-segment bitrate variations.

On the other hand, according to recent research, the most recent scalable 360° video streaming methods adapt the video to variations in network throughput and user behavior. None of the methods listed above perform well in either measured parameters: buffer size or average view bitrate. Our solution is to create a stable connection as network bandwidth changes while improving user experience. Furthermore, improving these two parameters will fine-tune the video streaming quality and expertise. Proper consideration and adjustment of these parameters help ensure that videos play smoothly and are of good quality on viewing devices.

a) Viewport bitrate

In this paper, we measured the viewport bitrate parameter to evaluate the human-perceived quality of the same viewport when streaming 360° videos over the same HTTP/2 stream. This parameter also shows that the higher the value of the bit rate, the better the perceived quality of the service. However, we also see that increasing the quality too high also affects the user's perception when it goes up and down suddenly.

Experiments show that the average bit rate of the proposed method improves better than the referenced methods in Table III and IV. For head movements #1, the proposed method improves up to 40.8% compared to method No. Besides, the most negligible improvement is 17.3% for the BL method. Meanwhile, for head movements #2, our method can improve up to 44.5%, while the lowest improvement is 17.2% in the GLVP method.

b) Buffer size

The purpose of our experiment is to understand cache activation and cache impact parameters. A bigger storage buffer means faster and better data streaming for users. This parameter also indicates that buffering has a net impact on video streaming because the quicker the buffer transmits, the more data will be downloaded.

Furthermore, we also found that this effect holds under strong bandwidth fluctuations and normal throughput. In practical testing, we found that the view bit rate fluctuates greatly under practically zero network fluctuations and does not change much with normal throughput. However, our algorithm is stable in these two cases of fluctuations.

Experiments show that five methods have buffer sizes between 4.7s and 6.4s. Although, in this test, the maximum allowed cache threshold is 7s. However, neither technique has reached the maximum threshold; this is considered a limitation for future expansion. However, the experiment shows that although the proposed method has not reached the maximum threshold, filling the buffer high and close to the threshold value is relatively good. Compared to the proposed method, the reference methods have deficient buffer levels, reaching 5,529s for head movement #1 and 5,535s for head movement #2. While measuring, the proposed method reaches the most minor threshold of 6,159s. Overall, with the current buffer level, the proposed method can improve up to 25%; we can see that the buffer level is shown in Tables V and VI.

In summary, the method has also improved and expanded on the commonly used methods. Further research on multi-directional motion is needed to gain a broader perspective.

V. CONCLUSIONS

In this paper, we studied 360° video streaming with extensive content via HTTP/2 protocol. The proposed method performs significantly more than the conventional method based on bit rate and buffer parameters. Overall, the proposed method improves significantly compared to the referenced methods; for head movements, the bit rate ratio can improve from 17.2% to 45.8%. Besides, the proposed method to enhance the buffer can achieve a maximum improvement of up to 25.0%.

REFERENCES

[1] T. Stockhammer. MPEG Immersive Media. Accessed: 2023-12-27. [Online]. Available: https://www.itu.int/en/ITU-T/studygroups/2017-2020/16/Documents/ws/201701ILE/S2P2-1701-01-19-MPEG-Immersive-Media-Thomas-Stockhammer.pdf

[2] V. H. Nguyen, N. N. Pham, C. T. Truong, D. T. Bui, H. T. Nguyen, and T. H. Truong, "Retina-based quality assessment of tile-coded 360-degree videos," *EAI Endorsed Transactions on Industrial Networks and Intelligent Systems*, vol. 9, no. 32, 2022.

[3] Huewei. (2017) Virtual reality/augmented reality white paper. [Online]. Available: http://www-file.huawei.com/-/media/CORPORATE/PDF/ilab/vr-ar-en.pdf

[4] H. Nguyen, T. N. Dao, N. S. Pham, T. L. Dang, T. D. Nguyen, and T. H. Truong, "An accurate viewport estimation method for 360 video streaming using deep learning," *EAI Endorsed Transactions on Industrial Networks and Intelligent Systems*, vol. 9, no. 4, pp. e2–e2, 2022.

[5] C. Ozcinar, A. De Abreu, and A. Smolic, "Viewport-aware adaptive 360° video streaming using tiles for virtual reality," in *2017 IEEE International Conference on Image Processing (ICIP)*, Beijing, China, Sep. 2017, pp. 2174–2178.

[6] D. Nguyen, N. V. Hung, N. T. Phong, T. T. Huong, and T. C. Thang, "Scalable multicast for live 360-degree video streaming over mobile networks," *IEEE Access*, vol. 10, pp. 38 802–38 812, 2022.

[7] N. V. Hung, P. H. Thinh, N. H. Thanh, T. T. Lam, T. T. Hien, V. T. Ninh, and T. T. Huong, "Lvsum-optimized live 360 degree video streaming in unicast and multicast over mobile networks," in *2023 IEEE 15th International Conference on Computational Intelligence and Communication Networks (CICN)*, 2023, pp. 29–34.

[8] W. L. S. Ramos, M. M. Silva, E. R. Araujo, A. C. Neves, and E. R. Nascimento, "Personalizing fast-forward videos based on visual and textual features from social network," in *2020 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2020, pp. 3260–3269.

[9] T. C. Thang, H. T. Le, A. T. Pham, and Y. M. Ro, "An evaluation of bitrate adaptation methods for HTTP live streaming," *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 4, pp. 693–705, 2014.

[10] H. T. T. Tran, N. P. Ngoc, A. T. Pham, and T. C. Thang, "A Multi-Factor QoE Model for Adaptive Streaming over Mobile Networks," in *2016 IEEE Globecom Workshops (GC Wkshps)*, Dec 2016, pp. 1–6.

[11] D. V. Nguyen and H. T. T. Tran and T. C. Thang, "Impact of delays on 360-degree video communications," in *2017 TRON Symposium (TRONSHOW)*, 12 2017, pp. 1–6.

[12] F. Muheidat, K. Dajani, and L. A. Tawalbeh, "Security concerns for 5g/6g mobile network technology and quantum communication," *Procedia Computer Science*, vol. 203, pp. 32–40, 2022, 17th International Conference on Future Networks and Communications / 19th International Conference on Mobile Systems and Pervasive Computing / 12th International Conference on Sustainable Energy Information Technology (FNC/MobiSPC/SEIT 2022), August 9-11, 2022, Niagara Falls, Ontario, Canada. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050922006123

[13] X. Chen, D. Wu, and I. Ahmad, "Optimized viewport-adaptive 360-degree video streaming," *CAAI Transactions on Intelligence Technology*, vol. 6, no. 3, pp. 347–359, 2021.

[14] S. Lederer, C. Müller, and C. Timmerer, "Dynamic adaptive streaming over http dataset," in *Proceedings of the 3rd multimedia systems conference*, 2012, pp. 89–94.

[15] J. Kua, G. Armitage, and P. Branch, "A survey of rate

adaptation techniques for dynamic adaptive streaming over http," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, pp. 1842–1866, 2017.

[16] H. Wu, X. Li, G. Wang, G. Cheng, and X. Hu, "Resolution identification of encrypted video streaming based on http/2 features," *ACM Trans. Multimedia Comput. Commun. Appl.*, vol. 19, no. 2, feb 2023. [Online]. Available: https://doi.org/10.1145/3551891

[17] V. H. Nguyen, D. T. Bui, T. L. Tran, C. T. Truong, and T. H. Truong, "Scalable and resilient 360-degree-video adaptive streaming over http/2 against sudden network drops," *Computer Communications*, vol. 216, pp. 1–15, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S01404036642400001X

[18] M. Mahmoud, S. Rizou, A. S. Panayides, N. V. Kantartzis, G. K. Karagiannidis, P. I. Lazaridis, and Z. D. Zaharis, "A survey on optimizing mobile delivery of 360° videos: Edge caching and multicasting," *IEEE Access*, 2023.

[19] A. Kumar, S. Banerjea, and M. Pandey, "Churn-tolerant cdn at the edge for adaptive video streaming: towards multi-connection approach using http/2," *International Journal of Information Technology*, vol. 15, no. 7, pp. 3491–3497, 2023.

[20] M. Nguyen, H. Amirpour, F. Tashtarian, C. Timmerer, and H. Hellwagner, "Performance analysis of h2br: Http/2-based segment upgrading to improve the qoe in has," *Multimedia Tools and Applications*, pp. 1–35, 2023.

[21] N. V. Hung, T. D. Chien, N. P. Ngoc, and T. H. Truong, "Flexible http-based video adaptive streaming for good qoe during sudden bandwidth drops," *EAI Endorsed Transactions on Industrial Networks and Intelligent Systems*, vol. 10, no. 2, pp. e3–e3, 2023.

[22] X. Corbillon, F. De Simone, and G. Simon, "360-degree video head movement dataset," in *Proceedings of the 8th ACM on Multimedia Systems Conference*, ser. MMSys'17. Taipei, Taiwan: Association for Computing Machinery, 2017, p. 199–204.

[23] JVET, "360lib," https://jvet.hhi.fraunhofer.de/, October 2016. [Online]. Available: https://jvet.hhi.fraunhofer.de/

[24] J. M. Boyce, Y. Ye, J. Chen, and A. K. Ramasubramonian, "Overview of shvc: Scalable extensions of the high efficiency video coding standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 26, no. 1, pp. 20–34, 2015.

[25] H. L. D. Huong, D. V. Nguyen, T. T. Huong, P. N. Nam, and T. C. Thang, "Smooth viewport bitrate adaptation for 360 video streaming," in *2019 6th NAFOSTED Conference on Information and Computer Science (NICS)*. IEEE, 2019, pp. 512–517.



NGUYEN DUC THIEN received the B.Sc.degree in Informatics of Vietnam National University, Hanoi, in 2005, the M.Sc. degree Master of Data transmission and computer network of Pots and Telecommunications Institute of Technology, Vietnam, in 2011.

He is a lecturer in the Faculty of Information Technology at the East Asia University of Technology, Trinh Van Bo Street, Nam Tu Liem District, Hanoi, Vietnam.

His research interests include multimedia communications, network security, artificial intelligence, and applications.

Email: thiennd@eaut.edu.vn



TRAN TRIEU HAI received a B.Sc.degree in Bachelor of Information Technology, Faculty of Information Technology, Hanoi National University, Vietnam, in 2004 and an M.Sc. degree from Master of Information Management in Information Management, Shu-te University, Taiwan, in 2013.

He is also an IT Lecturer at the Faculty of General Education - at Hanoi Open University, Nguyen Hien Street, Hai Ba Trung District, Hanoi, Vietnam.

His research interests include Network Security, Artificial Intelligence, Natural Language Processing, and applications.

Email: haitt@hou.edu.vn



NGUYEN VIET HUNG received a B.Sc.degree in Bachelor of Informatics pedagogy from the Faculty of Engineering Technology of Ha Tinh University, Vietnam, in 2012, the M.Sc. degree Master of Information Technology from the Faculty of Information Technology of Ho Chi Minh City University of Technology, Vietnam, in 2016, and has been learning the Ph.D. degree in Telecommunications Engineering from the Hanoi University of Science and Technology, Vietnam.

He is also a lecturer in the Faculty of Information Technology at the East Asia University of Technology, Trinh Van Bo Street, Nam Tu Liem District, Hanoi, Vietnam.

His research interests include multimedia communications, network security, artificial intelligence, QoE/QoS guarantee for network services, Virtual Reality, Natural Language Processing, Data Mining, and applications.

Email: hungnv@eaut.edu.vn

Empirical Study of Ovarian Tumors Detection and Classification in Ultrasound Images

Thi-Loan Pham [1,3], Van-Hung Le [2], Thi-Lan Le [3], Hai Vu [3], Duy-Hai Vu [3], Thanh-Hai Tran [3]

[1] Hai Duong University, Vietnam

[2] Tan Trao University, Vietnam

[3] School of Electrical and Electronic Engineering (SEEE), Hanoi University of Science and Technology, Vietnam

Correspondence: Van-Hung Le, email: van-hung.le@mica.edu.vn

Communication: received 11 Oct 2023, revised 09 Feb 2024, accepted 16 May 2024

Digital Object Identifier: 10.32913/mic-ict-research.v2024.n2.1237

Abstract: Ovarian cancer is one of the leading causes of death in women. Ultrasound images often provide initial findings and diagnosis before further tests are performed. Despite some progress in medical image analysis, the detection and recognition of ovarian cancers remains still quite limited. Most existing works focused on detecting or classifying two tumor categories, namely benign and malignant lesions. This paper presents an initial comprehensive study of ovarian tumor detection, and at the same time classifies eight classes of ovarian tumors in ultrasound images using deep-learning models. Due to the lack of training data, we implemented a data augmentation process and examined the improvement of performance with and without data augmentation. Experiments carried out on the OTU-2D set of MMOTU dataset with YOLOv5, YOLOv7, and YOLOv7 variants show promising detection and classification results.

Keywords: *Ovarian tumor, Classification, Data augmentation, Ultrasound images, YOLOv5 and YOLOv7.*

I. INTRODUCTION

According to statistics, the new incidence of ovarian cancer is 10.3 per 100,000 women per year, and the mortality rate is 6.3 per 100,000 women per year [1]. Currently, overcrowding in hospitals, particularly in developing countries such as Vietnam [2], leads to missed or misdiagnosed diseases, causing unintended consequences for patients and an economic burden on families and society. In addition, the diagnosis of the disease often depends on the doctor's experience. Therefore, to tackle these issues, it is necessary to develop tools to encourage the automatic detection and determination of ovarian tumors from ovarian ultrasound images.

Today, with the increased development of artificial intelligence, many applications of deep learning in medical image analysis has achieved expressive results. For instance, automatic detection of lung tumors from x-ray images [3], automatic detection and segmentation of lesions in gastrointestinal endoscopy images [4–6], breast and cervical cancer images analysis [7], [8], MRI images analysis [9], etc. However, the studies dedicated to ovary ultrasound analysis are limited and few datasets are publicly available for the research community. Most existing works focus on classifying images as normal or abnormal, or detecting abnormalities [10]. Further analysis such as classifying tumors into different tumor classes is still not thoroughly investigated.

Regarding ovarian ultrasound image analysis, recently Pham et al. [11] performed a study on the semantic segmentation of ovarian tumors on the OTU_2D subset of the MMOTU dataset. However, as the analysis is carried out at the pixel level, the computation requires GPU and is quite consuming time. The current best accuracy results for segmentation of 8 classes of ovarian tumors when using the accuracy measure is 71.56%, because the ovarian tumors on ultrasound images are relatively similar in appearance and structure, especially two classes (Mucinous cystadenoma, High grade serous).

Therefore, for faster and more accurate prediction, our approach is to employ a CNN to detect and classify ovarian tumors on ultrasound images. From there, it is possible to build an application to detect and classify ovarian tumors that can be processed on a regular computer and has low computational space (CPU-only desktops). Among many existing CNN-based detectors, YOLO (You Only Look Once) is a reasonable solution. YOLO is a deep learning network that allows detecting accurately objects. Especially YOLO is capable of operating on low-resource computers and running on CPUs [12]. However, when applying deep learning for computer vision tasks, the availability and data

distribution have a huge impact on the performance. In the medical domain, we may face a few training data and imbalance issues. Therefore, before implementation of the fine-tuning and evaluation process, detect, segment, and classify ovarian tumors in ultrasound images. We perform data augmentation with some simple methods (e.g. scaling, rotation, and flipping of images) on the training set of the original OTU_2D set (*OTU_2D-OS*) of the MMOTU dataset. The augmented dataset is called "*OTU_2D-DAS*".

In this paper, we conduct a comprehensive study to (1) evaluate the effectiveness of detecting with or without ovarian tumors in images; (2) detect and classify 8 classes of ovarian tumors by fine-tuning the ovarian tumor detection model and classifying ovarian tumors by YOLOv5 [13], and YOLOv7 [14]; Two evaluation studies (1) and (2) were performed on the *OTU_2D-OS* and *OTU_2D-DAS*. Below are the findings regarding the quantitative assessment, analysis, and hurdles encountered in the classification of ovarian tumors.

II. Related works

In recent years, there have been several works on automatic diagnosis based on different kinds of medical images. In medicine, several reports have shown AI to have high accuracy in diagnostics, such as in head CT scans, skin cancer, and retinopathy in diabetic patients. In gynecology, the application of AI has been previously studied, such as the automatic diagnosis of Pap-smear and digital colposcopy. In the last decade, the study of CAD has progressed given the remarkable development of computer science. Therefore, the study aimed to use AI to predict pathological diagnosis, using blood biomarkers, in patients with ovarian tumors.

Srivastava et al. [15] have used a fine-tuned VGG-16 network to detect whether an ovarian cyst is present or not in the image. In fine-tuning, an already existing pre-trained model is modified with a similar type of task that it earlier performed. This helps in taking benefit of the feature extraction that happens in the front layers of the network. The authors have used the VGG-16 model and modified its last four layers by training it on the ovarian cyst dataset using a CNN. This is done to achieve a high accuracy of 92.11%.

In the research of Akazawa et al. [16], the real-time dataset generated at the Institute of Tokyo Women's Medical University Medical Center East was used under the approval of the Institutional Review Board (IRB). Sample Size: 53 ovarian cancers, 23 borderline malignant tumors 126 benign ovarian tumors. Accuracy of five machine learning algorithms. The highest accuracy was 0.80 in the XGBoost algorithm, followed by 0.78 in a random forest, 0.67 in logistic regression, and 0.62 in support vector machine.

Qi et al. [17] proposed an MMOTU dataset with a set of *OTU_2D-OS* and *OTU_CEUS* consist of 1469 2D ultrasound images and 170 contrast-enhanced ultrasonography. In particular, the alignment feature was also proposed to the DS2Net network for semantic segmentation of ovarian tumor data on ultrasound images.

Data augmentation aims to generate additional data which is used to train the model and has been shown to improve performance when validated on a separate unseen dataset [18]. This approach has become commonplace so to help understand the types of data augmentation techniques used in state-of-the-art deep learning models, so conducted a systematic review of the literature where data augmentation was utilized on medical images to train a deep learning model. In this article, we employ data augmentation methods and various versions of YOLO to detect ovarian tumors. Experimental results have shown that the combination of data augmentation and YOLO is a promising approach for accurately detecting and delineating ovarian tumors in medical images. This technology has the potential to greatly aid doctors in accurately diagnosing and pinpointing the location of tumors, providing valuable advantages for the treatment and care of ovarian diseases.

III. Detecting and classifying ovarian tumors in ultrasound images

Detecting and classifying ovarian tumors is a very important task in the process of building a system to help doctors diagnose ovarian cancer. In this paper, we propose to conduct a comparative study on detecting and classifying ovarian tumors on ultrasound images, as illustrated in Fig. 1. In Fig. 1, the input is *OTU_2D-OS*, and through data augmentation, we get the *OTU_2D-DAS*. Then use YOLO to fine-tune the ovarian tumors detection and classification model on the *OTU_2D-OS* and the *OTU_2D-DAS*. The first output is the result of detecting the with or without ovarian tumors on ultrasound image, the second output is the result of classifying 8 classes of ovarian tumors. The results of the quantitative evaluation, analysis, and challenges of ovarian tumor classification are also presented below.

1. Background

Currently, many deep learning networks have been studied and applied to the problem of object detection and classification. In which YOLO version (v) 1 [19] is a typical network with average accuracy, has very fast computation speed, and can perform computations on CPU. From 2020

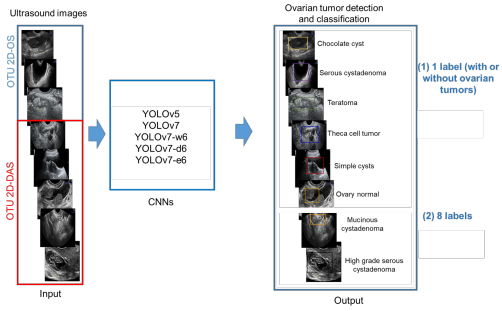


Figure 1. Our proposed framework for comparative study on ovarian tumors detection with and without data augmentation.

and early 2023, YOLOv5 [13] and YOLOv7 [14] networks were developed and announced.

YOLOv5 [20]: YOLOv5 focuses on speed and ease of use. YOLOv5 model architecture [20] has three important parts: Backbone, Neck, and Head, as illustrated in Fig. 2.

- The backbone aims to extract rich features from the input image. A Cross Stage Partial Network (CSPNet) is used as a model backbone. CSPNets are faster than deeper networks. YOLOv5 improves YOLOv4's CSPResBlock [21] into a new module, little more than a Convolution layer called a C3 module. In YOLOv5 still use CSPDarkNet53 for predicting the original object candidates according to the boxes. For the activation function, YOLOv4 [21] employs Mish or LeakyReLU in its lightweight version, whereas YOLOv5 utilizes SiLU.

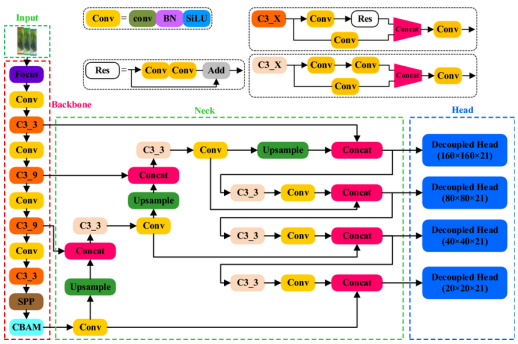


Figure 2. YOLOv5 architecture [20].

- The Neck constructs feature pyramids. This component helps the model to detect and identify the same objects at different sizes and scales. This leads to better performance with unlearned data. Different types of feature pyramids are used by many models, like FPN, BiFPN, and PANet which is used in YOLOv5. YOLOv5 adopts a module similar to SPP, but twice as fast and calls it SPP-Fast (SPPF). Instead of using parallel MaxPooling as in SPP, YOLOv5 SPPF uses sequential MaxPooling. Furthermore, the kernel size in the MaxPooling of SPPF is all 5 instead of (5,9,13) like the SPP of YOLOv4.
- The Head performs the final detection. It applies anchor boxes on generated features and outputs final vectors containing class probabilities, objectiveness scores, and bounding boxes. YOLOv5 has the same model head as YOLOv3 and YOLOv4.

Other changes in YOLOv5 compared to YOLOv4: about data augmentation: mosaic augmentation, copy-paste augmentation, mixup augmentation; about loss function: add scale factor for Objectness Loss; about anchor boxes: auto anchor using genetic algorithm; about removing grid sensitivity but the formula is different; about exponential moving average weight.

YOLOv7 [22]: YOLOv7 surpasses all known object detectors in both speed and accuracy in the range from 5 FPS to 160 FPS and has the highest accuracy at 0.568 AP among all known real-time object detectors with 30 FPS or higher on GPU V100.

The architecture of YOLOv7 is illustrated in Fig 3.

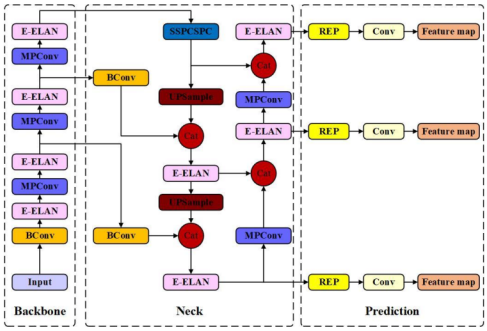


Figure 3. YOLOv7 architecture [14].

This architecture also includes three main parts.

- The backbone used the ELAN (YOLOv7-p5, YOLOv7-p6) and E-ELAN (YOLOv7-E6E).
- The neck used the SPPCSPC + (CSP-OSA)PANet (YOLOv7-p5, YOLOv7-p6) + RepConv.
- The head used the YOLOR + Auxiliary Head (YOLOv7-p6).

YOLOv7 has some improvements over than previous versions. One of the main improvements of YOLOv7 is the use of anchor boxes. Anchor boxes, which consist of predefined boxes with various aspect ratios, are utilized to detect objects of diverse shapes. YOLOv7 uses nine anchor boxes, allowing YOLO to detect a wider range of object shapes and sizes than previous versions, thus helping to decrease the number of wrong identifications. An important improvement in YOLO v7 is the use of a

new loss function called "focal loss". Previous versions of YOLO used the standard cross-entropy loss function, which is known to be less effective at detecting small objects. Focal loss solves this problem by reducing loss weights for well-classified examples and focusing on hard examples such as hard-to-detect objects. YOLOv7 also has different versions like YOLOv7-p5, YOLOv7-p6. Which YOLOv7-p5 group including YOLOv7-tiny, YOLOv7, and YOLOv7x is to use the images with size (640×640) for training. Which YOLOv7-p6 group including YOLOv7-w6, YOLOv7-e6, and YOLOv7-d6 is to use the images with size (1280×1280) for training.

2. Data augmentation

Augmenting medical image data is increasingly prevalent in deep learning applications [18]. Basic techniques like scaling, rotation, and flipping of images are commonly employed in model design. As more advanced methods such as deformable augmentation with randomized displacement fields and deep learning-based approaches like GANs are developed, they are increasingly integrated into model design practices. Although realism can be a desired outcome for augmented data, it's not always essential. Unrealistic augmentations might still yield a more generalizable model. Data augmentation proves particularly advantageous in scenarios where there's inadequate training data, helping to mitigate model over-fitting.

Data augmentation is a crucial technique in the realm of deep learning, especially when faced with a limited amount of training data. The primary concept behind this technique is to produce new variations of the existing data by implementing minor alterations. The objective is to generate a diverse dataset that enables the model to learn significant features without requiring additional data. By leveraging the diversity of the original dataset, data augmentation can enhance the model's ability to learn generalized features. Common transformations, such as horizontal and vertical flipping, rotation, and mirroring, can be particularly effective for tasks involving images.

When implementing data augmentation in situations where there is a dependency between samples in the data, it is crucial to carefully consider the potential loss of intrinsic characteristics or the creation of misleading samples that could negatively affect the performance of the model. Preprocessing techniques can be utilized to mitigate this issue. Prior to applying data augmentation, the data should be divided into independent sets to ensure the integrity of the data.

In this paper, we use five methods (x5) of data augmentation: (1) flip the vertical image, (2) flip the image horizontally, (3) rotate the image 90 degrees, (4) HUE, and (5) blur the image. They are illustrated in Fig. 4. Raising the hue of an image refers to the process of shifting the colors in the image along the hue spectrum. Hue is one of the components of the HSL (Hue, Saturation, Lightness) color model or the HSV (Hue, Saturation, Value) color model, which are commonly used in image processing. Hue represents the dominant wavelength of light that is perceived as color. By adjusting the hue value of an image, you can change the overall color tone while preserving the brightness and saturation of the image.

Average blurring, also known as box blurring or mean blurring, is a type of blurring operation commonly used in image augmentation. It involves replacing each pixel in an image with the average value of its neighboring pixels within a defined neighborhood. Here's how the average blurring operation works:

- Step 1: Define a kernel or filter size: The kernel size determines the size of the neighborhood used for blurring. It is typically represented as a square or rectangular matrix. The size of the kernel determines the extent of blurring. A larger kernel size will result in a more pronounced blurring effect.
- Step 2: Traverse through the image: For each pixel in the image, calculate the average value of the pixel intensities within the defined neighborhood. This is done by summing up the pixel intensities in the neighborhood and dividing by the total number of pixels in the neighborhood.
- Step 3: Replace the pixel value: Replace the original pixel value with the calculated average value. This step is repeated for every pixel in the image. The average blurring operation effectively smooths out the image by reducing high-frequency details and noise. It is a simple blurring technique that produces a uniform blurring effect across the image. Average blurring can be beneficial during image augmentation for various reasons. It helps to reduce noise and unwanted artifacts, which can improve the robustness and generalization of machine learning models. Additionally, it can be used to create a more visually pleasing appearance or simulate certain real-world effects.

This system classifies the test image into one of eight labels by leveraging features extracted from the test image. These features are processed through classifiers previously trained using the training parameters evaluated by the offline learning system. The offline classification system assesses the training parameters of the classifiers by employing a combination of features extracted from the training dataset and the corresponding ground truth training class labels.

Vikas et al [23], explained that the number of parameters

required corresponds to the data complexity of the operation in the machine learning model that must be carried out by using this kind of augmentation process. Sometimes it might still go wrong even if we have the correct size of the training set. To regularize convolution neural network models, it can be upgraded dataset by including more images dataset, using the data augmentation methods as shown in Fig 4.

By using data augmentation methods in the dataset, the image ratio has been improved at a 1: 9 ratio.

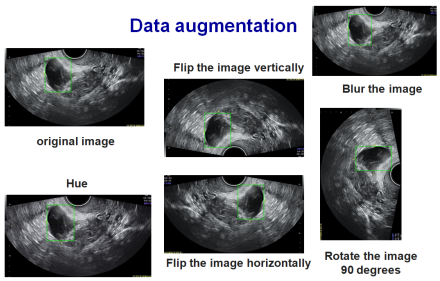


Figure 4. Five augmentation techniques are employed to enrich data in the training set.

IV. Experimental Results

1. Dataset

In this work, we show a follow-up study based on the *OTU_2D-OS*) of the MMOTU dataset [17]. The *OTU_2D-OS* includes 1469 2D ultrasound images and eight labels, as illustrated in Fig. 5.

For example, for an ovarian tumor, in OTU_2D set of MMOTU [17] dataset, some classes have a small amount of data as shown in Fig. 6. (0: Chocolate cyst, 1: Serous cystadenoma, 2: Teratoma, 3: Theca cell tumor, 4: Simple cysts, 5: Ovary normal, 6: Mucinous cystadenoma, 7: High grade serous cystadenoma). The number of samples in the classes is imbalanced, with 226 images in class 0, while class 7 has only 38 images. Imbalanced data often leads to inaccurate predictions for the minority group. This is because the majority class dominates the prediction results, while the minority class is poorly represented. However, accurately predicting a sample belonging to the minority group is much more important than predicting samples from the majority group. To improve prediction results, appropriate adjustments are needed to ensure that the model achieves high accuracy on the minority group.

Before performing data enhancement, we perform data normalization of the *OTU_2D-OS*, with the image size being 320 × 240 pixels. This process includes ground truth standardization. At the same time, we perform data preparation according to two datasets, which are single-label data with or without ovarian tumors (1-label) and 8-label data on ovarian tumor classes as shown in of MMOTU dataset [17]. In particular, the bounding box of the tumors was normalized against the bounding box of the YOLO's bounding box format with the COCO 2017 dataset [24], as presented in Eq. 1.

$$a = \frac{\frac{x_max+x_min}{2}}{w_b}; \quad b = \frac{\frac{y_max+y_min}{2}}{h_b} \\ c = \frac{w_b}{w_im}; \quad d = \frac{h_b}{h_im} \tag{1}$$

where (x_max, y_max) is the coordinates of the top right corner of the ovarian tumor bounding box on the image, (x_max, y_max) is the coordinates of the bottom left corner of the ovarian tumor bounding box on the image, w_b is the width size of the bounding box of the ovarian tumor on the image, h_b is the height size of the ovarian tumor bounding box on the image, w_im is the width size of the image, h_im is the height size of the image.

The data structure of the bounding box in the format of COCO 2017 used for training and evaluation has the form (l, a, b, c, d), where l is the label of the object.

After data augmentation, the *OTU_2D-DAS* obtained the following number of images in Fig. 7. The statistical data before and after augmentation are presented in Tab. I.

We performed data augmentation exclusively on some minor classes (i.e. theca cell tumor, simple cysts, mucinous cystadenoma, and high-grade serous cystadenoma) in the training set of the entire dataset, increasing the sample number of each of such classes by a factor of five. After the augmentation phase, the number of data samples in the training set increased from 1000 to 1852 images. We have precisely summarized the number of original data samples along with the augmented samples in Tab. I.

Data augmentation is a common technique used to enrich training data when the training set is limited. Numerous studies have demonstrated that augmentation contributes to enhancing the performance of image-based tasks. Conventional techniques for augmenting data involve applying geometric and photo-metric transformations to the original images. In our work, we employ five transformations across four classes, as mentioned previously, including: (1) vertical flipping, (2) horizontal flipping, (3) rotating by 90 degrees, (4) adjusting hue, and (5) blurring. To ensure the variation and diversity of the augmented data, we did not apply any additional pre-processing steps to remove dependent data. In our work, we apply five data augmentation techniques to a subset of our training set. The augmented data is utilized to train the network. However, we do not currently evaluate

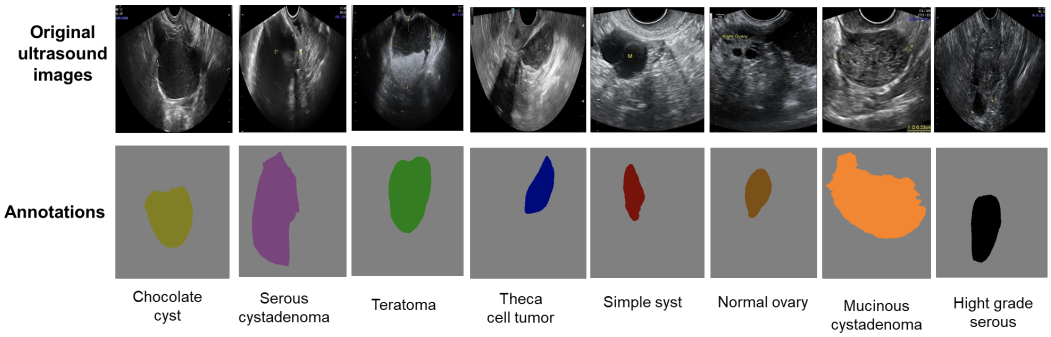


Figure 5. Illustration of eight ovarian tumor categories in the *OTU_2D-OS* dataset.

TABLE I
THE STATISTICS OF ORIGINAL DATA (OTU_2D-OS) AND DATA AFTER ENHANCEMENT (OTU_2D-DAS).

Classes	Index classes	Number of original training samples	Augmentation factor	Number of augmented training samples	Number of testing samples
Chocolate cyst	0	226		226	110
Serous cystadenoma	1	153		153	66
Teratoma	2	228		228	108
Theca cell tumor	3	57	x5	285	31
Simple cysts	4	47	x5	235	19
Ovary normal	5	180		180	87
Mucinous cystadenoma	6	71	x5	355	33
High grade serous cystadenoma	7	38	x5	190	15
Total		1000		1852	469

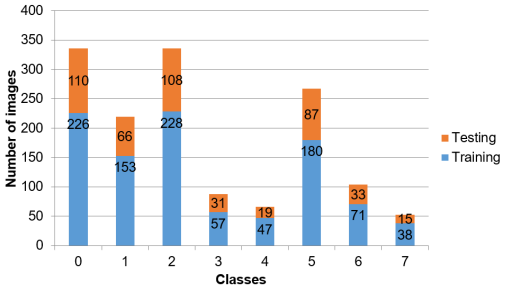


Figure 6. The number of training and testing images in the dataset OTU_2D.

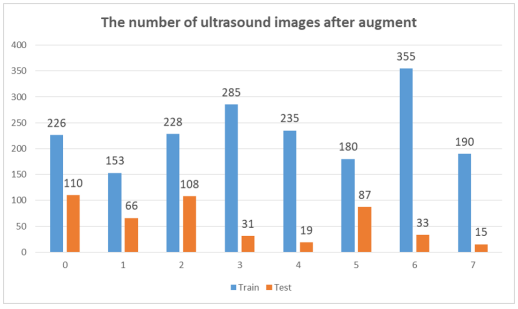


Figure 7. The number of ovarian tumors ultrasound images after augmentation.

the specific role of each transformation. Investigating the role of each augmentation technique will be a focus of our future work. This aspect is also highlighted in the conclusion section of the revised manuscript.

2. Evaluation metrics and measurement

We use *IOU* (Intersection Over Union) - metric of evaluation to evaluate the performance of detection of ovarian tumors as Fig. 8, Precision (P) as Eq. 2, Recall (R) as Eq. 2, and mAP (mean Average Precision) as Eq. 3.

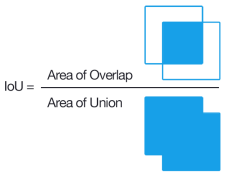


Figure 8. The formula to calculate the Intersection Over Union (IOU).

$$P = \frac{TP}{TP + FP}; \qquad R = \frac{TP}{TP + FN}; \tag{2}$$

$$mAP = \frac{1}{|\ \text{Classes}\ |} \sum_{c \in \text{Classes}} \frac{TP(c)}{TP(c) + FP(c)} \tag{3}$$

where TP is the correct prediction, FP is the false prediction, TN is a false prediction (the label is positive but the prediction is negative), and FN is the no predicted

ground truth data area. The threshold of IOU we use is 0.5.

In this work, we implemented the examined models on a server featuring an NVIDIA GeForce RTX 2080 Ti, 12GB GPU, for the purposes of fine-tuning, training, and testing. The programs were written in the Python language (≥3.7 version) with the support of the CUDA 11.2/cuDNN 8.1.0 libraries. In addition, there are some libraries such as Matplotlib, OpenCV, Mmcls≥0.20.1, Packaging, Numpy, Prettytable, PyTorch 1.5+, etc. Especially the source code is inherited and developed on the YOLOv5 ([1]) and YOLOv7 ([2]). The models of YOLOv5 and YOLOv7 are fine-tuned with 100 epochs. The source code to fine-tune the ovarian tumor detection and classification model of YOLOv5 and YOLOv7 is shown in links ([3]),([4]), respectively.

3. Results and Discussions

Ovarian cancer stands as the foremost cause of death among women, underscoring the critical significance of early diagnosis in the medical realm. In this paper, we conducted a comparative study utilizing YOLO for the segmentation of ovarian tumors. In this study, two main problems were solved: the detection and classification of 1-label (1 class) and 8-label (8 classes) of ovarian tumors in the OTU_2D subset of the MMOTU dataset.

The results of ovarian tumor detection and one class classification (1-label) with YOLOv5, YOLOv7, YOLOv7-w6, YOLOv7-d6, and YOLOv7-e6 on the *OTU_2D-OS* and *OTU_2D-DAS* are shown in Tab. II. The best detection and classification results are from YOLOv5 with ($P = 0.91, R = 0.89$) of the *OTU_2D-OS* and ($P = 0.90, R = 0.90$) of the *OTU_2D-DAS*. It can be seen that the results in Tab. II, are not improved on the *OTU_2D-OS* and *OTU_2D-DAS* sets. This result is due to the augmented data process being just some simple processing methods of image processing on the original data image, so the change of the distinguishing features with or without ovarian tumor is not great. And this evaluation is only concerned with or without ovarian tumors on the image so the detection results are not improved. Specifically, in Tab. II with the P metric, the average results of the models with the *OTU-OS* dataset are 0.87, while with the *OTU-DAS* dataset, it's 0.86. With the R metric, the average results of the models with the *OTU-OS* dataset are also 0.87, and with the *OTU-DAS* dataset, it's 0.86. As for the mAP50 metric, the average results of the models with the *OTU-OS* dataset are 0.90, while with the *OTU-DAS* dataset, it's 0.91, which is 1% higher compared to the non-augmented data.

Figure 9(a) presents the distribution of ovarian tumor classification results on the training data of the *OTU_2D-OS* and *OTU_2D-DAS* when using YOLOv7-e6 to classify with or without ovarian tumors in ultrasound images. We show the results of YOLOv7-e6 because based on the results of Tab. II, YOLOv7-e6 has the best results among the variants of YOLOv7. In Fig. 9(b), the training results of YOLOv7-e6 are not much improved on the original data ($R = 0.97, P = 0.901$) and the augmented data ($R = 0.98, P = 0.942$).

Figure 10(a) and Figure 10(b) presents the distribution of ovarian tumor classification results on the training data of the *OTU_2D-OS* and *OTU_2D-DAS* when using YOLOv5 to classify with or without ovarian tumors in ultrasound images.

The results in Tab. II also show that R is much lower than P, which shows that the with or without ovarian tumors classifier of YOLO is highly wrong classified. Because R is calculated as the number of correct classifications divided by the number of correct classifications plus the number of false negatives. If R is higher then the lower the number of missed ovarian tumors.

The results of ovarian tumor detection and eight classes classification (8-labels) with YOLOv5, YOLOv7, YOLOv7-w6, YOLOv7-d6, and YOLOv7-e6 on the *OTU_2D-OS* and *OTU_2D-DAS* are shown in Tab. III.

In our work, we applied five data augmentation techniques to a subset of our training set. The augmented data is utilized to train the network. In the original dataset *OTU_2D-OS*, there are 1239 gray-scales images and 216 color images (doppler images). Among five data augmentation techniques, only "adjusting hue" affects the color of 216 color images. To evaluate the impact of "adjusting hue" technique on the final result, we conducted an additional experiment where only 4 augmentation techniques without "hue adjusting" (vertial flipping, horizontal flipping, rotating by 90 degrees, and blurring) to create a dataset *OTU_2D-DAS** (without hue adjusting). We select Yolov5 for testing and experimental result is shown in Table IV. In average, without "hue adjusting" augmentation, the recall is reduced significantly from 0.72 to 0.675. However, mAP50 remains similar. In conclusion, hue adjusting helps to improve the recall of detection. We have included this experiment in the revised manuscript.

Table V shows the results of ovarian tumor detection and eight classes classification (8-labels) on the *OTU_2D-OS* by YOLOv5 (epoch = 100) with threshold $IOU = 0.25$ and $IOU = 0.75$. The average results of detecting and

[1] https://github.com/ultralytics/yolov5

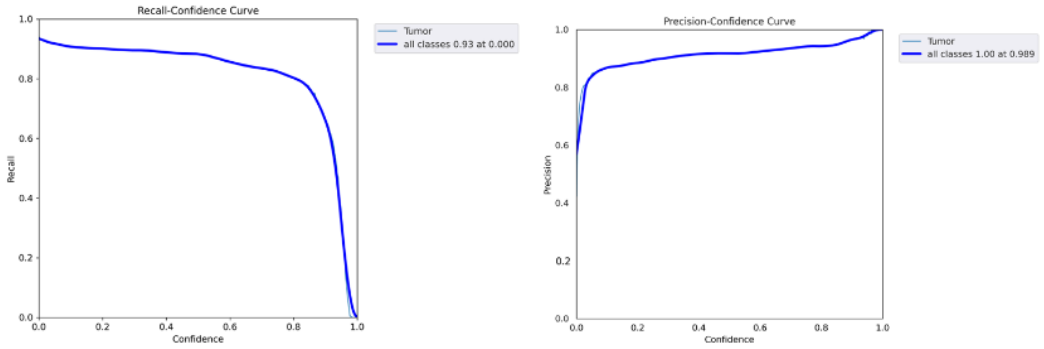
[2] https://github.com/WongKinYiu/yolov7

[3] https://drive.google.com/file/d/1aJS8hEisvQu-ugBDYxq5e9_Zw5EceisB/view?usp=sharing

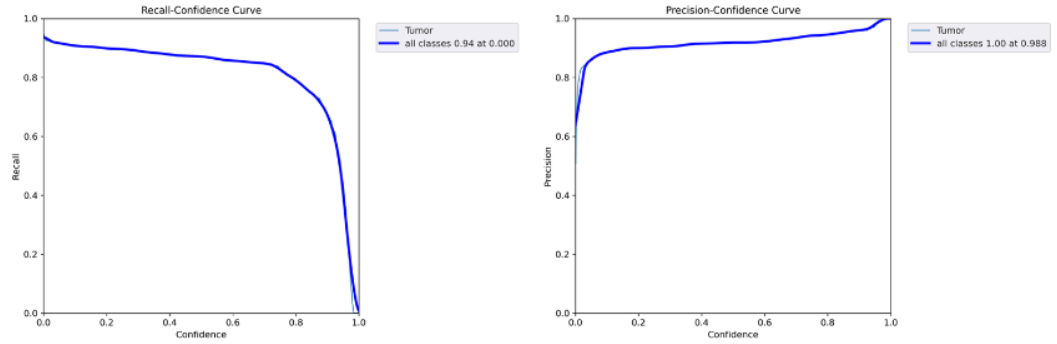
[4] https://drive.google.com/file/d/1Qg1umKPCToi2WyOsKJ_Kq4Gc1Mz-lE7O/view?usp=sharing

TABLE II
THE RESULTS OF OVARIAN TUMOR DETECTION AND BINARY CLASSIFICATION (1-LABEL) ON THE *OTU_2D-OS* AND *OTU_2D-DAS* BY YOLOV5, YOLOV7, AND OTHER YOLOV7 VARIATIONS (ALL ARE TRAINED WITH THE NUMBER OF EPOCHS = 100).

Class/ Measurements	Dataset	Precision (*P*)					Recall (*R*)					*mAP50*				
		YOLO v5	YOLO v7	YOLO v7-w6	YOLO v7-d6	YOLO v7-e6	YOLO v5	YOLO v7	YOLO v7-w6	YOLO v7-d6	YOLO v7-e6	YOLO v5	YOLO v7	YOLO v7-w6	YOLO v7-d6	YOLO v7-e6
Tumor	OTU 2D-OS	**0.91**	0.86	0.83	0.84	0.90	**0.89**	0.87	0.82	0.88	0.87	**0.92**	0.91	0.86	0.92	0.92
	OTU 2D-DAS	**0.90**	0.84	0.84	0.90	0.84	**0.90**	0.83	0.87	0.86	0.89	**0.93**	0.89	0.91	0.92	0.91



(a) Distribution of classification results on the training data of the original data when using YOLOv7-e6 to classify with or without ovarian tumors in ultrasound images



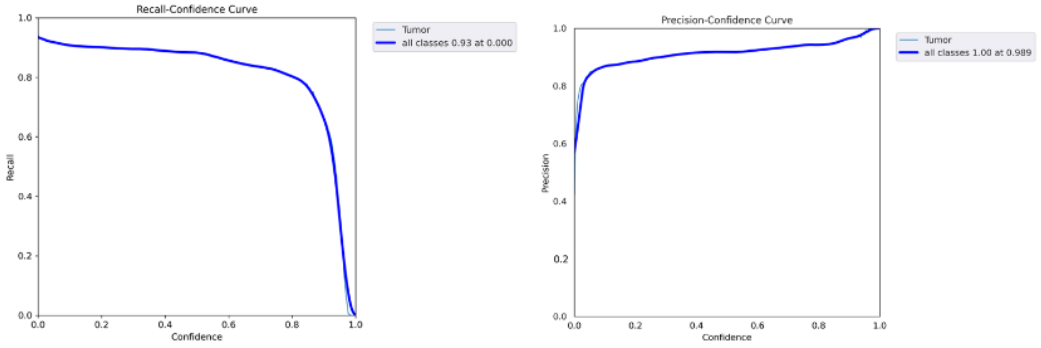
(b) Distribution of classification results on the training data of the data augmentation when using YOLOv7-e6 to classify with or without ovarian tumors in ultrasound images

Figure 9. Illustration of the distribution of ovarian tumor classification results on the training data of the *OTU_2D-OS* and *OTU_2D-DAS* when using YOLOv7-e6 to classify with or without ovarian tumors in ultrasound images.

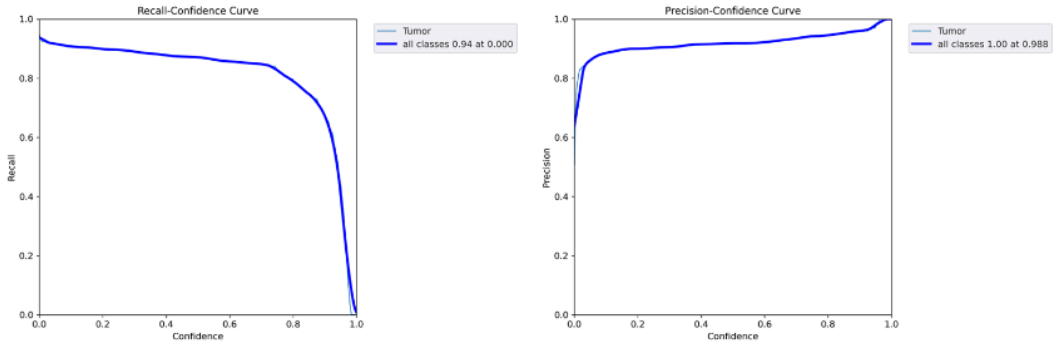
classifying 8 ovarian tumor classes of YOLOv5 are ($P = 0.853, P = 0.83, P = 0.748$), ($R = 0.683, R = 0.66, R = 0.6$), ($mAP25 = 0.781, mAP50 = 0.75, mAP75 = 0.628$) with threshold $IOU = 0.25, IOU = 0.5, IOU = 0.75$, respectively. The detection and classification results are gradually decreasing with increasing IOU.

Table VI shows the results of ovarian tumor detection and eight classes classification (8-labels) on the *OTU_2D-DAS* by YOLOv5 (epoch = 100) with threshold $IOU = 0.25$ and $IOU = 0.75$, respectively. The results of detecting and classifying ovarian tumors also decrease as IOU increases as shown in Tab. V.

Figure 11 presents the distribution of ovarian tumor classification results on the training data of the *OTU_2D-OS* and *OTU_2D-DAS* when using YOLOv7-e6 to classify 8 classes of ovarian tumors in ultrasound images. In Fig. 11, the training results of YOLOv7-e6 are not much improved on the original data ($R = 0.95, P = 0.96$) and the augmented data ($R = 0.914, P = 0.972$). Figure 11 also shows the distribution of prediction results R and P of labels (3: Theca cell tumor, 4: Simple cysts, 6: Mucinous cystadenoma, 7: High grade serous cystadenoma) which have ten times more confidence than labels (0: Chocolate cyst, 1: Serous cystadenoma, 2: Teratoma, 5: Ovary normal). Therefore, the results of classifying 8 ovarian tumor classes of labels (0: Chocolate cyst, 1: Serous cystadenoma, 2: Teratoma, 5: Ovary normal) are better than labels (3: Theca cell tumor, 4: Simple cysts, 6: Mucinous cystadenoma, 7: High grade serous cystadenoma). The results in Tab. III are also presented on YOLOv5, and YOLOv7 and the results show



(a) Distribution of ovarian tumor classification results on the training data of the original data when using YOLOv5 to classify with or without ovarian tumors in ultrasound images



(b) Distribution of ovarian tumor classification results on the training data of the data augment when using YOLOv5 to classify with or without ovarian tumors in ultrasound images

Figure 10. Illustration of the distribution of ovarian tumor classification results on the training data of the *OTU_2D-OS* and *OTU_2D-DAS* when using YOLOv5 to classify with or without ovarian tumors in ultrasound images.

that the detection and classification results of YOLOv5 have the highest results at 83%.

Figure 12 shows the confusion matrix in classifying 8 ovarian tumor classes on ultrasound images of the *OTU_2D-OS* and *OTU_2D-DAS* using YOLOv5. On the *OTU_2D-OS* (left), the prediction result of label 7 (High grade serous cystadenoma) is mistaken as background with a rate of 0.2, and label 0 (Chocolate cyst) is also mistakenly classified as background with a rate of 0.33. On the *OTU_2D-DAS* (right), the prediction result of label 7 (High grade serous cystadenoma) is mistaken as background with a rate of 0.33, and label 1 (Serous cystadenoma) is also mistakenly classified as background with a rate of 0.24.

Figure 13 shows the results of R in comparison to the distribution of YOLOv5 to classifier 8 ovarian tumor classes when training on the *OTU_2D-OS* and *OTU_2D-DAS* with 100 epochs and 469 epochs. In this paper, we performed YOLOv5 training to build a model to detect and classify 8 classes on the *OTU_2D-OS* and *OTU_2D-DAS* with a number of epochs of 500. However, when training to epoch 469^{th}, the training process stopped due to the model can not be improved further, because, after each parameter adjustment at each epoch, the model will be evaluated on the validation set. This shows that the ovarian tumor detection and classification model that we trained at 100 epochs has reached the highest possible performance, it can not get better with more epochs of training. Therefore, it is reasonable to stop training the model at 100 epochs to compare the results.

From the performance results, overfitting gradually decreased, leading to higher accuracy, which is better than CNN models using *OTU_2D-OS*. For instance, when using the YOLOv7 model with *OTU_2D-DAS*, the Precision metric increased by 0.15. Similarly, when using the YOLOv5 model, the average R metric increased by 0.06 compared to using *OTU_2D-OS*. It can be said that the issue of detecting and classifying ovarian tumors is very important in the process of examining and diagnosing ovarian tumors for patients. The results in Tab. II, Tab. III show that R is low. This proves that many ovarian tumors have not been detected and classified properly. This is very dangerous in detecting and diagnosing ovarian cancer. For example, a woman has an ovarian tumor but it is not detected through medical examination, and this person will not receive early treatment for the tumor, which will lead to the tumor becoming larger and metastasizing to organs, and others in

TABLE III
THE RESULTS OF OVARIAN TUMOR DETECTION AND EIGHT CLASSES CLASSIFICATION (8-LABELS) ON THE ***OTU_2D-OS*** AND ***OTU_2D-DAS*** BY YOLOV5, YOLOV7, AND OTHER YOLOV7 VARIATIONS (NUMBER OF EPOCHS FOR TRAINING THE MODELS = 100).

Classes/ Measurements	**Dataset**	**Precision (*P*)**					**Recall (*R*)**					***mAP*50**				
		YOLO v5	**YOLO v7**	**YOLO v7-w6**	**YOLO v7-d6**	**YOLO v7-e6**	**YOLO v5**	**YOLO v7**	**YOLO v7-w6**	**YOLO v7-d6**	**YOLO v7-e6**	**YOLO v5**	**YOLO v7**	**YOLO v7-w6**	**YOLO v7-d6**	**YOLO v7-e6**
Chocolate cyst	OTU 2D-OS	0.82	0.57	0.58	0.55	0.53	0.65	0.76	0.75	0.75	0.80	0.74	0.69	0.70	0.70	0.71
	OTU 2D-DAS	0.71	0.72	0.66	0.71	0.73	0.77	0.72	0.74	0.70	0.77	0.76	0.74	0.75	0.74	0.76
Serous cystadenoma	OTU 2D-OS	0.91	0.44	0.42	0.37	0.39	0.64	0.86	0.85	0.94	0.86	0.81	0.62	0.56	0.57	0.62
	OTU 2D-DAS	0.62	0.61	0.53	0.61	0.54	0.82	0.70	0.70	0.61	0.74	0.78	0.71	0.61	0.69	0.62
Teratoma	OTU 2D-OS	0.93	0.61	0.62	0.54	0.41	0.76	0.82	0.77	0.81	0.90	0.88	0.78	0.82	0.81	0.81
	OTU 2D-DAS	0.80	0.80	0.82	0.81	0.65	0.86	0.69	0.76	0.74	0.82	0.89	0.79	0.83	0.81	0.80
Theca cell tumor	OTU 2D-OS	0.82	0.49	0.38	0.38	0.27	0.65	0.34	0.19	0.52	0.36	0.68	0.41	0.28	0.31	0.22
	OTU 2D-DAS	0.79	0.80	0.76	0.64	0.54	0.60	0.68	0.61	0.57	0.58	0.68	0.74	0.61	0.64	0.65
Simple cysts	OTU 2D-OS	0.83	0.99	0.00	0.15	0.12	0.74	0.11	0.00	0.37	0.16	0.72	0.27	0.13	0.15	0.12
	OTU 2D-DAS	0.87	0.80	0.63	0.27	0.26	0.74	0.63	0.58	0.42	0.58	0.72	0.69	0.60	0.45	0.38
Ovary normal	OTU 2D-OS	0.85	0.39	0.52	0.54	0.47	0.61	0.63	0.75	0.81	0.82	0.76	0.52	0.69	0.68	0.65
	OTU 2D-DAS	0.74	0.70	0.56	0.59	0.66	0.70	0.48	0.81	0.75	0.71	0.76	0.61	0.72	0.66	0.68
Mucinous cystadenoma	OTU 2D-OS	0.81	0.28	0.24	0.21	0.18	0.67	0.49	0.55	0.82	0.67	0.74	0.40	0.30	0.29	0.27
	OTU 2D-DAS	0.84	0.67	0.53	0.61	0.53	0.65	0.48	0.39	0.57	0.48	0.77	0.60	0.53	0.50	0.55
High grade serous cystadenoma	OTU 2D-OS	0.65	0.39	0.33	0.27	0.23	0.61	0.20	0.27	0.13	0.30	0.64	0.22	0.17	0.16	0.20
	OTU 2D-DAS	0.77	0.65	0.69	0.65	0.38	0.66	0.40	0.44	0.38	0.33	0.67	0.53	0.44	0.47	0.34
AVERAGE	OTU 2D-OS	**0.83**	0.52	0.39	0.38	0.32	**0.66**	0.53	0.51	0.64	0.61	**0.75**	0.49	0.46	0.46	0.45
	OTU 2D-DAS	**0.77**	0.72	0.65	0.61	0.54	**0.72**	0.60	0.63	0.59	0.63	**0.75**	0.67	0.63	0.62	0.60

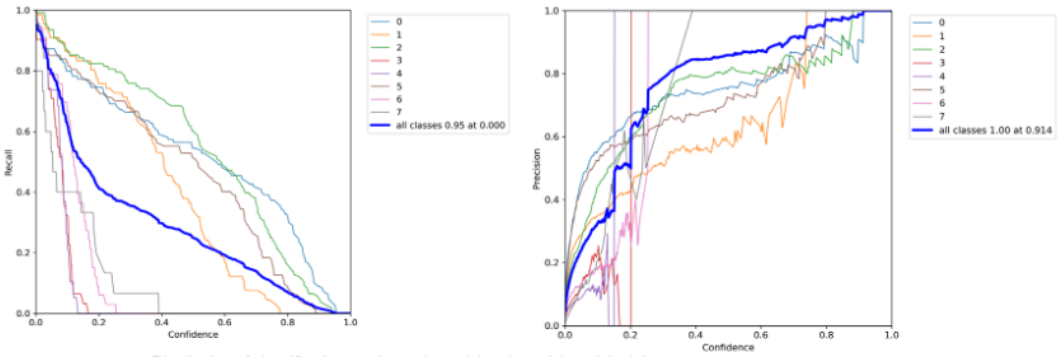
TABLE IV
COMPARISON OF PERFORMANCE OF YOLO-V5 WITH AND WITHOUT "HUE ADJUSTING" AUGMENTATION TECHNIQUE

Class ID	**Precision**		**Recall**		**mAP50**	
	OTU_2D -DAS	OTU_2D -DAS* (without hue adjusting)	OTU_2D -DAS	OTU_2D -DAS* (without hue adjusting)	OTU_2D -DAS	OTU_2D -DAS* (without hue adjusting)
0	0.71	0.811	0.77	0.703	0.76	0.789
1	0.62	0.727	0.82	0.769	0.78	0.806
2	0.8	0.885	0.86	0.785	0.89	0.887
3	0.79	0.794	0.6	0.613	0.68	0.669
4	0.87	0.769	0.74	0.684	0.72	0.741
5	0.74	0.789	0.7	0.643	0.76	0.752
6	0.84	0.815	0.65	0.666	0.77	0.719
7	0.83	0.782	0.66	0.533	0.67	0.661
Average	0.77	0.797	**0.72**	**0.675**	0.75	0.757

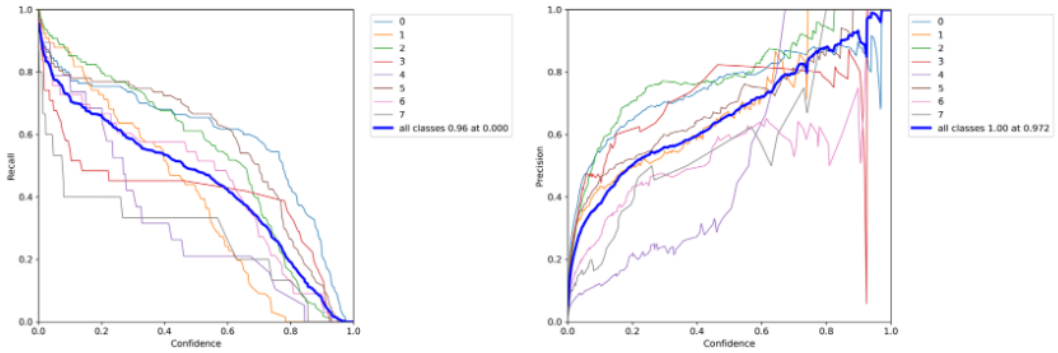
the body When detected, the disease is in a late stage and very difficult to treat. This reduces the patient's chance of survival. In medical examination and treatment, especially when detecting ovarian tumors, it is often better to accept "a mistake than to miss" it. Thereby increasing the ability to detect early and promptly treat ovarian tumors. The study also highlights significant obstacles in constructing a support system for diagnosing ovarian tumors. Moving forward, we will persist in enhancing CNNs for detecting ovarian tumors and assessing their performance on additional datasets related to ovarian tumors.

The processing time for detecting and classifying 8 ovarian tumor classes on the *OTU_2D-OS* when performing calculations on GPU, is shown in Tab. VII.

Based on the results in Tab. II, Tab. III, Fig. 11, Fig. 12, Fig. 13, it can be seen that the results of detecting ovarian tumors on ultrasound images with YOLOv5 are better than those of YOLOv7. This problem occurs because the



(a) Distribution of ovarian tumor classification results on the training data of the original data when using YOLOv7-e6 to classify to 8 class ovarian tumors in ultrasound image



(b) Distribution of ovarian tumor classification results on the training data of the original data when using YOLOv7-e6 to classify to 8 class ovarian tumors in ultrasound image

Figure 11. Illustration of the distribution of ovarian tumor classification results on the *OTU_2D-OS* and *OTU_2D-DAS* training data when using YOLOv7-e6 to classify to 8 class ovarian tumors in ultrasound images.

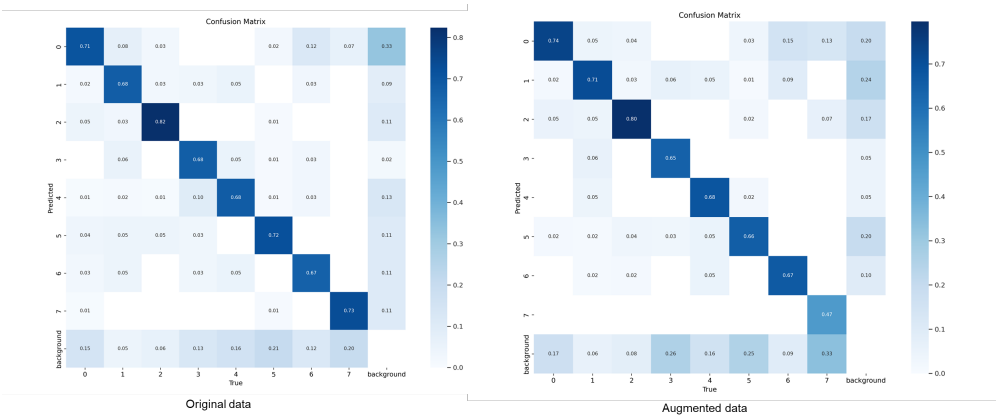
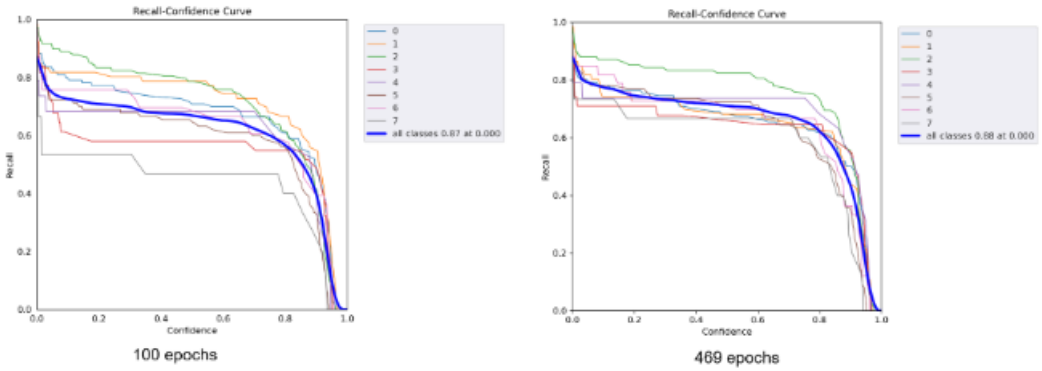


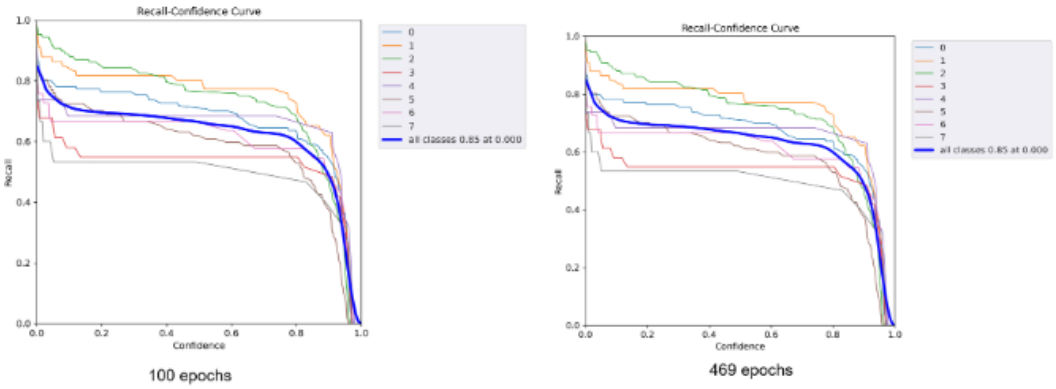
Figure 12. The confusion matrix in classifying 8 ovarian tumor classes on ultrasound images of the *OTU_2D-OS* and *OTU_2D-DAS* using YOLOv5.

difference between YOLOv5 and YOLOv7 is mainly about computational speed in object detection, higher resolution allows YOLO v7 to detect smaller objects. In this paper, the size of the normalized input image is 320×240 pixels. When used in YOLOv5, the size of the image is changed to 640×640 pixels, and when using YOLOv7, the input image size is mainly changed to 1280×1280 pixels. Therefore, the object prediction on a much larger image makes the learning set less efficient. Hence, the object detection result of YOLOv7 is lower than that of YOLOv5. The results in Tab. III have increased 0.116 because the data has been augmented for the training set, which only YOLOv7 has increased results and YOLOv5 has not improved results.

Figure 15 shows the detection and classification of the 8 classes of ovarian tumors. The label of the class and the confidence coefficient of the prediction are shown on the



(a) Distribution of ovarian tumor classification results on the training data of the original data when when training on the OTU 2D-OS and OTU 2D-DAS with 100 epochs and 469 epochs



(b) Distribution of ovarian tumor classification results on the training data of the original data when training on the OTU 2D-OS and OTU 2D-DAS with 100 epochs and 469 epochs

Figure 13. The recall distribution of YOLOv5 when classifying 8 ovarian tumor classes when training on the *OTU_2D-OS* and *OTU_2D-DAS* with 100 epochs and 469 epochs.

bounding box. Each label will be represented by a distinct color of the bounding box.

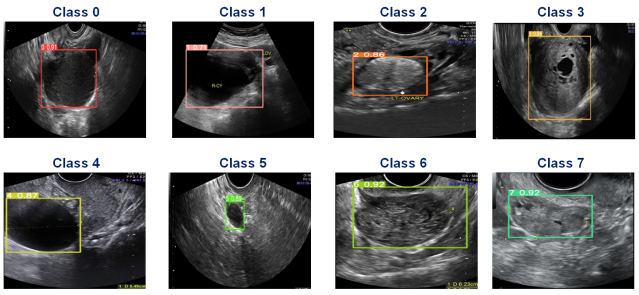


Figure 14. The result of detection and classification by YOLOv5 with 8 classes of ovarian tumors.

Figure 16 shows a wrong classification result using YOLOv5, on an area of ovarian tumor with 2 delineated bounding boxes. Where the bounding box on the outside is the wrong prediction.

During the experimentation process, there were some failure cases in which the proposed method could not detect tumors as shown in Fig. 17, or incorrect delineation of the

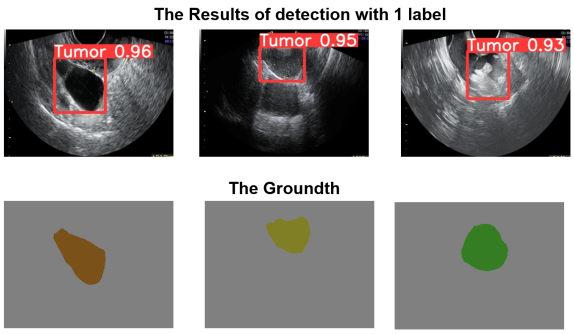


Figure 15. The result of detection and classification by YOLOv5 with 1-label.

regions. It can be seen that the case in Fig. 17 is very challenging, the tumor area data is weakly differentiated from the normal data area of the ultrasound image.

V. Conclusion

In this work, we augmented only the training dataset without making any changes to the testing set. This is a common practice when applying augmentation techniques

TABLE V
THE RESULTS OF OVARIAN TUMOR DETECTION AND EIGHT CLASSES CLASSIFICATION (8-LABELS) ON THE *OTU_2D-OS* BY YOLOV5 (EPOCH = 100) WITH THRESHOLD IOU = 0.25 AND IOU = 0.75.

IOU	Measurement/ Index classes	P	R	mAP25
IOU=0.25	0	0.849	0.665	0.81
	1	0.913	0.635	0.827
	2	0.943	0.766	0.896
	3	0.818	0.645	0.681
	4	0.838	0.737	0.721
	5	0.87	0.616	0.799
	6	0.807	0.667	0.736
	7	0.786	0.736	0.779
	Average	**0.853**	**0.683**	**0.781**
	Measurement/ Index classes	**P**	**R**	**mAP75**
IOU=0.75	0	0.778	0.609	0.644
	1	0.869	0.605	0.742
	2	0.749	0.609	0.65
	3	0.777	0.613	0.628
	4	0.84	0.737	0.715
	5	0.555	0.391	0.347
	6	0.771	0.636	0.696
	7	0.643	0.6	0.602
	Average	**0.748**	**0.6**	**0.628**

TABLE VI
THE RESULTS OF OVARIAN TUMOR DETECTION AND EIGHT CLASSES CLASSIFICATION (8-LABELS) ON THE *OTU_2D-DAS* BY YOLOV5 (EPOCH = 100) WITH THRESHOLD IOU = 0.25 AND IOU = 0.75.

IOU	Measurement/ Index classes	P	R	mAP25
IOU=0.25	0	0.765	0.809	0.839
	1	0.701	0.788	0.799
	2	0.827	0.806	0.864
	3	0.822	0.548	0.644
	4	0.913	0.632	0.755
	5	0.722	0.782	0.731
	6	0.961	0.755	0.839
	7	0.713	0.499	0.615
	Average	**0.803**	**0.702**	**0.761**
	Measurement/ Index classes	**P**	**R**	**mAP75**
IOU=0.75	0	0.599	0.625	0.603
	1	0.707	0.788	0.786
	2	0.658	0.639	0.617
	3	0.824	0.548	0.64
	4	0.922	0.626	0.713
	5	0.483	0.517	0.349
	6	0.961	0.748	0.813
	7	0.616	0.43	0.453
	Average	**0.721**	**0.615**	**0.622**

to enrich the training set for deep learning models. The testing set should be kept in its original form. In the future, we will conduct more experiments with larger datasets and compare the results with the current findings.

We perform an initial comparative study in tumor detection (classifying one class with or without a tumor) and

TABLE VII
THE PROCESSING TIME (FPS) FOR DETECTING AND CLASSIFYING 8 OVARIAN TUMOR CLASSES ON THE ***OTU_2D-OS*** WHEN PERFORMING CALCULATIONS ON GPU AND CPU.

Methods	Type	YOLO v5	YOLO v7
Processing time	GPU	166	**204**
Processing time	CPU	0.7	**2.04**

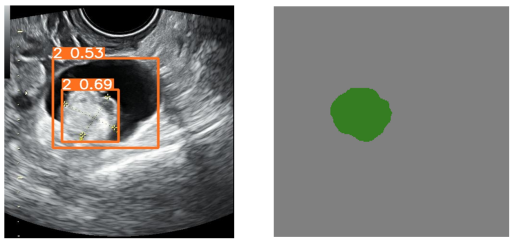


Figure 16. The result shows a misclassification result using YOLOv5, on an area of ovarian tumor with 2 delineated bounding boxes.

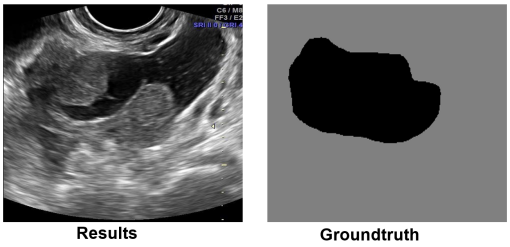


Figure 17. Illustrating the case of undetectable ovarian tumor on ultrasound image.

classifying eight classes of ovarian tumors on ultrasound using YOLOv5 and YOLOv7-p6. With mAP of ovarian tumors detection and one class classification is 0.86 to 0.92 of YOLOv5 and YOLOv7-p6. From 0.45 to 0.75 of mAP when ovarian tumors were detected and eight classes of classification.

At the same time, we also fine-tuned the tumor detection and classification model on an augmented training dataset. The results show that data augmentation increases the results by 0.116. In particular, we also show YOLO's superior computation time compared to other methods for ovarian tumor detection and classification (YOLOv7's processing speed is 204fps on GPU and 2.04fps on CPU). In the future, other deep learning networks could be investigated and improved for tumor detection and and classification.

ACKNOWLEDGMENT

This research is funded by Ministry of Science and Technology (MOST) under grant number KC-4.0-45/19-25 "Research and development of a computer-aided support system for ovarian cancer diagnosis using ultrasound images".

REFERENCES

[1] N. C. Institute, "Cancer Stat Facts: Ovarian Cancer," https://seer.cancer.gov/statfacts/html/ovary.html, 2023, [Online; accessed 14-July-2023].

[2] Vietnam, "Overcrowding strains HCM City hospitals, doctors," https://en.vietnamplus.vn/vietnam-to-publicly-rank-quality-of-hospitals/87505.vnp, 2023, [Online; accessed 14-July-2023].

[3] J. R. Astley, J. M. Wild, and B. A. Tahir, "FUNCTIONAL IMAGING OF THE LUNG SPECIAL FEATURE : Artificial intelligence in functional imaging of the lung," *Br J Radiol. 2022 Apr 1; 95(1132): 20201107*, no. April 2021, 2022.

[4] Z. Jin, T. Gan, P. Wang, Z. Fu, C. Zhang, Q. Yan, X. Zheng, X. Liang, and X. Ye, "Deep learning for gastroscopic images: computer-aided techniques for clinicians," *BioMedical Engineering Online*, vol. 21, no. 1, pp. 1–41, 2022. [Online]. Available: https://doi.org/10.1186/s12938-022-00979-8

[5] P.-T. Nguyen, M.-Q. Le, Q.-T. Dao, V. A. Tran, V.-H. Dao, and T.-H. Tran, "Automatic classification of upper gastrointestinal tract diseases from endoscopic images," in *2022 11th International Conference on Control, Automation and Information Sciences (ICCAIS)*. IEEE, 2022, pp. 442–447.

[6] T.-H. Tran, D. H. Vu, D. H. Tran, K. L. Do, P. T. Nguyen, V. T. Nguyen, L. T. Nguyen, N. K. Ho, H. Vu, and V. H. Dao, "Dcs-unet: Dual-path framework for segmentation of reflux esophagitis lesions from endoscopic images with u-net-based segmentation and color/texture analysis," *Vietnam Journal of Computer Science*, vol. 10, no. 02, pp. 217–242, 2023.

[7] P. Xue, J. Wang, D. Qin, H. Yan, Y. Qu, S. Seery, Y. Jiang, and Y. Qiao, "Deep learning in image-based breast and cervical cancer detection: a systematic review and meta-analysis," *npj Digital Medicine*, vol. 5, no. 1, 2022.

[8] E. Mahoro and M. A. Akhloufi, "Applying Deep Learning for Breast Cancer Detection in Radiology," *Current Oncology*, vol. 29, no. 11, pp. 8767–8793, 2022.

[9] A. S. Lundervold and A. Lundervold, "An overview of deep learning in medical imaging focusing on MRI," *Zeitschrift fur Medizinische Physik*, vol. 29, no. 2, pp. 102–127, 2019.

[10] B. Usha and S. Sandya, "Abnormality Detection in Ovarian Ultrasound Images using Active Contours," *Journal of Biomedical Engineering and Medical Imaging*, vol. 1, no. 4, pp. 14–23, 2014.

[11] T.-l. Pham, V.-h. Le, T.-h. Tran, and D. H. Vu, "Comprehensive study on Semantic Segmentation of Ovarian Tumors from Ultrasound Images," in *Conference on Information Technology and its Applications*, 2023.

[12] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," *https://arxiv.org/pdf/1804.02767.pdf*, 2018. [Online]. Available: http://arxiv.org/abs/1804.02767

[13] X. Zhu, S. Lyu, X. Wang, and Q. Zhao, "TPH-YOLOv5: Improved YOLOv5 Based on Transformer Prediction Head for Object Detection on Drone-captured Scenarios," *Proceedings of the IEEE International Conference on Computer Vision*, vol. 2021-October, pp. 2778–2788, 2021.

[14] G. Weng, D. Liu, L. Liu, Q. Wan, and S. Wu, "Research on an improved yolov7 algorithm for road surface crack detection," 2023.

[15] S. Srivastava, P. Kumar, V. Chaudhry, and A. Singh, "Detection of ovarian cyst in ultrasound images using fine-tuned vgg-16 deep learning network," *SN Computer Science*, vol. 1, pp. 1–8, 2020.

[16] M. Akazawa and K. Hashimoto, "Artificial intelligence in ovarian cancer diagnosis," *Anticancer research*, vol. 40, no. 8, pp. 4795–4800, 2020.

[17] Q. Zhao, S. Lyu, W. Bai, L. Cai, B. Liu, M. Wu, X. Sang, M. Yang, and L. Chen, "A multi-modality ovarian tumor ultrasound image dataset for unsupervised cross-domain semantic segmentation," *CoRR*, vol. abs/2207.06799, 2022.

[18] P. Chlap, H. Min, N. Vandenberg, J. Dowling, L. Holloway, and A. Haworth, "A review of medical image data augmentation techniques for deep learning applications," *Journal of Medical Imaging and Radiation Oncology*, vol. 65, no. 5, pp. 545–563, 2021.

[19] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[20] Y. Zhu, S. Li, W. Du, and Y. Du, "Identification of table grapes in the natural environment based on an improved yolov5 and localization of picking points," *Precision Agriculture*, vol. 24, no. (4), pp. 1–22, 2023.

[21] A. Bochkovskiy, C. Y. Wang, and H. Y. M. Liao, "YOLOv4: Optimal Speed and Accuracy of Object Detection," *arXiv*, 2020.

[22] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, "YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors," *https://arxiv.org/pdf/2207.02696.pdf*, pp. 7464–7475, 2022. [Online]. Available: http://arxiv.org/abs/2207.02696

[23] N. Sathiadhas Puvaneswari, "Detection of polycystic ovarian syndrome using convolutional neural network in conjunction with transfer learning models," Ph.D. dissertation, Dublin, National College of Ireland, 2022.

[24] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European conference on computer vision*. Springer, 2014, pp. 740–755.



Thi-Loan Pham received Bachelor degree at Faculty Information Technology Hanoi Pedagogical University 2 (2007). She received M.Sc. degree at University of Engineering and Technology (2012). Currently, she is a lecture of HaiDuong University and is a PhD student of School of Electrical and Electronic Engineering (SEEE), Hanoi University of Science and Technology, Vietnam. Her research interests include Ovarian cancer on Ultrasound images.

Email: phamthiloan2011@gmail.com



Van-Hung Le received M.Sc. degree at Faculty Information Technology- Hanoi National University of Education (2013). He received PhD degree at International Research Institute MICA HUSTC-NRS/UMI - 2954 - INP Grenoble (2018). Currently, he is a lecture of Tan Trao University. His research interests include Computer vision, RANSAC and RANSAC variation.

Email: van-hung.le@mica.edu.vn



Thi-Lan Le graduated in Information Technology from Hanoi University of Science and Technology (HUST), Vietnam. She obtained an MS. degree in Signal Processing and Communication from HUST, Vietnam. In 2009, she received her Ph.D. degree at INRIA Sophia Antipolis, France in video retrieval. She is currently associate professor at School of Electrical and Electronic Engineering (SEEE), HUST, Vietnam. Her research interests include images processing, computer vision, content-based indexing and retrieval, video understanding and human-robot interaction.
Email: Thi-Lan.Le@mica.edu.vn



Thanh-Hai Tran graduated an engineer's degree in Information Technology from Hanoi University of Science and Technology (HUST) in 2001. She holds a M.S. degree and a Ph.D degree in Imagery Vision Robotic from Grenoble INP, France, in 2002 and 2006 respectively. Currently, she is a lecturer/researcher at School of Electronics and Telecommunications and Computer Vision Department, International Research Institute in Multimedia, Information, Communication and Application, HUST. Her main research interests are visual object recognition, video understanding, human-robot interaction and text detection for applications in Computer Vision.
Email: hai.tranthithanh1@hust.edu.vn



Duy-Hai Vu is an associate professor of biomedical engineering, director of Biomedical Electronics Center at Hanoi University of Science and Technology. He received the PhD degree from the Hanoi University of Science and Technology in 2013 and became the associate professor of biomedical engineering in 2016. His research ranges are bio-signal processing and applications, medical devices and technology, integrated electronic medical record, medical IoT and AI for health. Manufacturing equipment used in the covid-19 pandemic: BKVM-HF1 (HFNC), BK-Vent (Ventilator), BK-NICO (ICG Cardiac Output Measurement Equipment.
Email: hai.vuduy@hust.edu.vn



Hai Vu received BE in Electronics and Telecommunications in 1999 and M.E. in Information Processing and Communication in 2002, both from HUST, Hanoi, Vietnam. He received Ph.D. in Computer Science from Osaka University, Japan, in 2009. He has been a lecturer/researcher in HUST, Hanoi, Vietnam since 2012. His current research interests are in Computer Vision, Pattern Recognition, and Human-Computer Interactions.
Email: hai.vu@hust.edu.vn

Method for Mining High-utility Patterns in Transaction Stream Data based on Linked List Structure

Minh-Thai Tran[1], Anh-Duy Tran[1], Duc-Thanh Pham[1], Minh-Nguyen Le[1]

[1] Faculty of Information Technology, Ho Chi Minh City University of Foreign Languages - Information Technology, Vietnam

Correspondence: Minh-Thai Tran, thaitm@huflit.edu.vn,
Communication: received 15 Dec 2023, revised 18 Apr 2024, accepted 23 May 2024
Digital Object Identifier: 10.32913/mic-ict-research.v2024.n2.1249

Abstract: Mining valuable patterns in data streams presents a significant challenge in the field of data mining. This task is crucial as it allows for the identification of highly profitable item sets within transaction databases. However, as new transactions are continually added, new valuable patterns emerge, thus changing the usefulness of previously analyzed data. It is essential to promptly update information regarding these changes to enable effective business decision-making. Consequently, existing mining methods applied to transaction flow datasets require considerable time to identify new patterns and update information related to new transactions. This article focuses on the research and proposal of a new transaction stream data mining method called High-Utility Stream Linked-List Mining. The method utilizes a linked list structure known as the High-Utility Stream Linked List (HUSLL) to store information about patterns in the database. Mining and updating transaction information are directly performed on the HUSLL structure. Experimental results demonstrate that this novel mining method exhibits more efficient execution times compared to previous solutions.

Keywords: *transaction stream data, data mining, high-utility item set.*

I. INTRODUCTION

Mining association rules [1] is one of the important topics in the field of data mining. Typically, association rules in mining are bifurcated into two stages: frequent pattern mining, also known as frequent item sets, followed by the generation of rules from these frequent patterns. However, some patterns, although not frequent, carry a higher value, such as instances where users purchase a few items with each item holding greater value than the remaining items. Consequently, frequent pattern mining encounters limitations when not considering both the number of items (internal utility value) and the value of the items (external utility value) purchased. To address these limitations, high-utility pattern mining is introduced, which considers both internal and external utility values to determine high-utility patterns within the transaction database [2]. Various high-utility pattern mining algorithms are proposed, including HMiner [3], EFIM [4], MAHI [5] and UBP-Miner [6].

However, existing algorithms are designed for static datasets, where data remains unchanged over time. Yet, the utility of a pattern or set of items may vary over time, such as beach tourism-related items being highly useful in the summer but rarely purchased in the winter. Consequently, methods for mining high-utility patterns over time are proposed to extract useful patterns in continuously updated databases. These methods include mining high-utility patterns on growing databases, such as HUI-list-INS algorithms [7], EIHI [8], LIHUP [9], and others. Additionally, transactional data stream mining methods, such as MHUI-BIT [10], MHUI-TID [10], and HUPMS [11] are introduced.

The current limitation of transaction stream data mining methods lies in the generation of considerable intermediate data when using prefix tree or condition tree structures for mining. Furthermore, during data update operations, specifically sliding window movement operations, updating the state of intermediate data structures incurs significant time and memory costs. This article proposes a list structure called High-Utility Stream Linked List (HUSLL), designed for efficient storage of data related to highly useful patterns in the database, enabling quick updates whenever a sliding window shift occurs. Specifically, the article introduces the High-Utility Stream Linked-List Mining (HSLM) algorithm, exploiting the HUSLL structure to detect high-utility patterns in the transaction stream database.

By using a list structure, pattern mining can be ap-

proached differently than with the HUPMS algorithm. The list structure can provide more flexible mining methods, optimizing the process of extracting pattern information. This can ensure that data is efficiently and accurately mined. Furthermore, with the list structure, updating the pattern information storage structure when there is a window shift can be optimized. When the sliding window moves, the list can be updated flexibly and efficiently. This helps save time and resources in updating the pattern information. Additionally, using a list structure can also change the way information is stored for parallel mining operations. The list structure can enhance the efficiency of parallel mining processes by partitioning and managing pattern information better.

The remainder of the article is structured as follows. Section II presents related research, followed by concepts and definitions in Section III. Section IV details the proposed method, while Section V showcases experimental results. Finally, conclusions and future research directions are discussed in Section VI.

II. RELATED WORKS

Frequent pattern mining has received significant attention and research since the introduction of the Apriori algorithm by Agrawal and Srikant in 1994 [1]. This algorithm utilizes the Apriori property, which states that "All nonempty subsets of a frequent itemset are frequent." In the pattern generation process, if an infrequent pattern emerges, there is no need to generate a parent pattern for that candidate, allowing for the early elimination of candidates to reduce space and exclude non-frequent patterns. However, a notable drawback is the algorithm's tendency to generate a large number of candidates, requiring multiple database scans to verify pattern popularity and thus increasing the overall mining process cost. In 2000, Han et al. proposed the FP-Growth algorithm [12] to address this limitation by exploiting frequent patterns without generating candidate patterns. This algorithm utilizes a tree structure called FP-Tree to store pattern information and employs a divide-and-conquer strategy for frequent pattern generation. Initially, the algorithm scans the database to identify sets with a length of 1 (1-itemset). Subsequently, the FP-Tree structure is constructed to identify frequent patterns within the 1-item set. While finding frequent patterns in trees incurs lower costs, the time required to generate FP-Trees may increase with larger databases. Following this, several other algorithms have been proposed to enhance the efficiency of frequent pattern mining, including LP-Growth [13], PMRARIM-IEG [14], and based on a linear table [15].

Frequent pattern mining algorithms are known for their focus on examining the occurrence frequency of patterns. However, many real-world problems require considering additional characteristics of the data, such as the internal utility value and external utility value of each item within a transaction. Frequent pattern mining algorithms do not meet this requirement. As a result, high-utility pattern mining algorithms, such as those proposed by [3], [4], [16], and [17], have been introduced. In 2005, Liu et al. [18] presented the Two-Phase algorithm for mining high-utility patterns, which builds upon the Apriori algorithm. To leverage the characteristics of the Apriori algorithm, Liu introduced the concept of transaction utility weights (TWUs) to estimate high-utility patterns based on utility weights. Currently, other mining methods such as FHM [16] and D2HUP [17] have been proposed for mining high-utility patterns using data structures like lists. The effectiveness of applying these list structures has been demonstrated on various experimental datasets.

Business analytics and decision support applications face the challenge of handling large and continuously growing volumes of data over time. A prime example is sales data, which is consistently updated with each customer purchase. While there exist high-utility pattern mining algorithms such as HMiner [3], EFIM [4], D2HUP [17], MAHI [5] and UBP-Miner [6]. these algorithms are designed to operate on static data. In other words, they do not accommodate data updates during the mining process, making them unsuitable for addressing the real-world mining needs of continuously evolving datasets.

To address the limitations of static data mining algorithms, several approaches have been proposed for incremental databases. In 2008, Yeh et al. [19] introduced two algorithms based on the Apriori approach: Incremental Utility Mining and Fast Incremental Utility Mining. These algorithms focus on generating candidates and testing to identify high-utility patterns. In 2009, Ahmed et al. [20] presented the IHUP algorithm, which utilizes a tree structure for mining high-utility patterns in incremental data. Several tree-based mining algorithms have since been proposed, such as iCHUM [21] and HUPID-Growth [22]. In 2015, Lin et al. [7] introduced a list-based mining algorithm named HUI-list-INS. Additionally, Fournier-Viger et al. [8] proposed EIHI, another list-based algorithm in the same year. In 2017, Yun et al. [9] presented the LIHUP algorithm, which stands out for its ability to mine high-utility patterns in a dynamic environment without the need to generate candidates.

Furthermore, research has extended to mining algorithms for incremental data and transaction stream data. Transaction stream data, characterized by continuous updates and item ordering over time, presents unique challenges. Items or sets of items may exhibit varying levels of utility

over different periods. Sliding window-based algorithms, such as MHUI-BIT [10] and MHUI-TID [10], have been proposed to handle the dynamic nature of continuously changing datasets. However, these algorithms are noted for generating a significant number of candidates, which poses a limitation.

Three main models are currently being used to mine high-utility itemsets in data streams. The first model is the landmark model: It mines all high-utility itemsets from the past to the present at a given time point from the data stream. The MAHUSP [23] algorithm that was proposed in 2017 is included among them. To store the necessary patterns, this algorithm utilizes a tree structure called MAS-Tree. MAHUSP remains limited in its memory capacity when adding new potential patterns to the tree. The second model is time decay models: This model is designed to find the latest relevant information from the data by distinguishing the importance of old and new transactions through a weight. In this model, GENHUI [24] and HAUPM [25] are two typical algorithms that are included. Lastly, the sliding window model: Separate the stream data into multiple batches, each batch having multiple transactions. The window will only keep a specific number of the latest batches in the stream data. Once there is new stream data, the data in the old batches will be excluded and the new transactions will be added to the window as the last batch. In this paper, we focus on the approach of mining high-utility itemsets from stream data using the sliding window model.

In 2010, Ahmed et al. [11] introduced a high-utility pattern mining algorithm designed for stream data known as HUP Mining over Stream Data (HUPMS). This algorithm utilizes a tree structure named HUS-Tree to store data stream information and conducts high-utility pattern mining based on this structure. Each time the sliding window changes, the algorithm updates the information of the nodes in the tree. The HUPMS algorithm involves generating prefix trees and condition trees as intermediate structures during the mining process. High-utility patterns are then derived from the conditional tree. However, a limitation of this algorithm is the necessity to generate numerous prefix trees and condition trees each time a pattern is mined, resulting in significant time consumption. Additionally, employing a storage tree structure requires updating the state of each node whenever the sliding window moves, especially if non-leaf nodes must be deleted, leading to increased processing time. Also, approaching from the perspective of a sliding window, the SHU-Growth [26] algorithm used a tree structure similar to HUS-Tree to store highly useful pattern information in the current sliding window. However, the strength of SHU-Growth lies in the addition of two techniques, Reducing Global Estimated Utilities and Reducing Local Estimated Utilities, aimed at reducing the search space and pruning out unpromising candidate patterns. In some datasets, experimental results have indicated that it is more effective than HUPMS.

Most recently, in 2022, Reddy et al. proposed an improved algorithm based on a tree structure called EGUI-Tree [27] to mine high utility itemsets from streaming data under the sliding window model. In this algorithm, each node in the tree stores information about the candidate patterns, including the transaction utility value and the remaining utility value. The authors proposed a formula to recalculate the utility value of patterns after each information update to reduce search space. The limitation of this algorithm is that the update operation involves updating the entire tree structure, including modifying the pattern values in each node, using batch information update operations, and shifting batch positions, which requires a significant amount of time and memory to execute. In addition, the deletion operation still has the limitation of the HUPMS algorithm when it needs to check all the information of the child nodes of the node to be deleted.

To overcome these limitations, this article proposes a high-utility pattern mining method based on a linked list structure called HUSLL. Unlike the HUPMS algorithm, this method eliminates the need to construct prefix trees or condition trees while still effectively identifying high-utility patterns in the transaction stream database. Utilizing a list structure simplifies node updates, thereby reducing processing time when handling data updates.

III. PROBLEM DEFINITION

This article adopts and builds upon definitions from previous studies [28]. Let $I = \{i_1, i_2, \ldots, i_n\}$ represent the set of items (or events), and $D = \{T_1, T_2, \ldots, T_m\}$ represent the transaction database, where each $T_j \in D$ is a children's set of items from I.

Definition 1: The internal utility of each item, denoted as $iu(i, T_j)$, is the frequency of item i appearing in transaction T_j. For instance, in Table I, item a appears twice in transaction T_1, so the internal utility of item a in transaction T_1 is denoted as $iu(< a >, T_1) = 2$.

Definition 2: External utility of each item, denoted as $eu(i)$, represents the value of item i. For example, in Table II, the external utility of item a is $eu(< a >) = 5$.

Definition 3: The utility of item i in transaction T_j, denoted as $u(i, T_j)$, is calculated by the formula (1):

$$u(i, T_j) = eu(i) \times iu(i, T_j) \tag{1}$$

For example $u(< a >, T_1) = 2 \times 5 = 10$, using data from Table I and Table II.

Definition 4: The utility of itemset X in transaction T_j, denoted as $u(X, T_j)$, is calculated by the formula (2):

$$u(X, T_j) = \sum_{i \in X} u(i, T_j) \tag{2}$$

Where $X = \{i_1, i_2, \ldots, i_k\}$ is the set of k items, $X \subseteq T_j$ and $1 \leq k \leq n$. For example, the item set $< ac >$ in transaction T_1 has a utility of $u(< ac >, T_1) = 2 \times 5 + 3 \times 10 = 40$, using data from Table I and Table II.

Definition 5: The utility of itemset X in the database, denoted as $u(X)$, is calculated by the formula (3):

$$u(X) = \sum_{X \subseteq T_j, T_j \in D} u(X, T_j) \tag{3}$$

For example, the utility of $< ac >$ in Table I is $u(< ac >) = u(< ac >, T_1) + u(< ac >, T_5) + u(< ac >, T_6) + u(< ac >, T_7) = 235$.

Definition 6: The utility of transaction T_j, denoted as $tu(T_j)$, is the total utility value of the items included in transaction T_j, calculated by formula (4):

$$tu(T_j) = \sum_{i \in T_j} u(i, T_j) \tag{4}$$

For example, the utility of transaction T_1 in Table I is $tu(T_1) = u(< a >, T_1) + u(< c >, T_1) + u(< d >, T_1) = 2 \times 5 + 3 \times 10 + 4 \times 7 = 68$.

Definition 7: The minimum utility threshold δ is a predetermined percentage value based on the total utility value of all transactions in the database. In Table I, the total utility of all transactions is 734. The minimum utility value can be defined using formula (5):

$$minutil = \delta \times \sum_{T_j \in D} tu(T_j) \tag{5}$$

For example, when $\delta = 20\%$, the minimum utility value is calculated as $minutil = 20\% \times 734 = 146.8$. A set of items X is deemed a high-utility pattern if $u(X) \geq minutil$. The problem of mining high-utility patterns is framed as identifying all sets X that satisfy the condition $u(X) \geq minutil$.

In the frequent pattern mining problem, the Apriori property proves valuable for the early elimination of infrequent patterns by showing that all extensions of an infrequent event are also infrequent. However, this property does not apply to the problem of mining high-utility patterns. For instance, if $minutil$ is set at 146.8, then the pattern $< a >$ in Table I is not considered a high-utility pattern because $u(< a >) = 115$. Nevertheless, the extended set of $< a >$ is $< ac >$, which qualifies as a high-utility pattern since $u(< ac >) = 235$. Consequently, a measure called utility weight has been introduced to address the high-utility pattern mining problem.

Definition 8: The transaction weight utility of a set of items X, denoted as $twu(X)$, is calculated as the sum of the utility values of transactions containing X, defined by formula (6):

$$twu(X) = \sum_{X \subseteq T_j, T_j \in D} tu(T_j) \tag{6}$$

For example, in Table I the transaction weight utility of the item $< d >$ is calculated as $twu(< d >) = 68 + 43 = 111$. The Apriori property can be applied to this measure. For example, with $minutil = 146.8$, then $twu(< d >) < minutil$. Subsequently, the expanded pattern of $< d >$ is $< dc >$, which also exhibits a transaction weight utility smaller than minutil or $twu(< dc >) < minutil$.

A pattern X is considered a high transaction-weighted utility pattern if $twu(X) \geq minutil$.

Transaction stream data is a prevalent type of data in practical applications, encompassing sales data, website operations data, weather data, etc. In this type of data, the volume is unlimited and can grow over time. A batch of transactions comprises numerous individual transactions, while a window encompasses several batches. Figure 1 illustrates an example of transaction stream data, with the database divided into four equal-sized batches and two windows. Each batch contains two transactions, and each window comprises three batches. Window W_1 includes batches B_1, B_2, and B_3, while window W_2 includes batches B_2, B_3, and B_4.

Definition 9: A transaction T is said to be reorganized, if the items present in T are arranged in the increasing order of their twu values [27]. For example, reorganized transaction for $T_1 = \{< a >, < c >, < d >\}$ is $\{< d >, < a >, < c >\}$ because $twu(< d >) = 111 < twu(< a >) = 439 < twu(< c >) = 647$.

Definition 10: Given an itemset X and a reorganized transaction T with $X \subseteq T$, the set of items after X in T is represented as $A(T/X)$ and are available in T followed by X. For example, set of items after itemset $(< d >, < a >)$ in T_1 given by $A(T_1/\{< d >, < a >\}) = \{< c >\}$.

Definition 11: The remaining utility [27] of itemset X in a reorganized transaction T_j, denoted as $ru(X, T_j)$, is calculated by the formula (7):

$$ru(X, T_j) = \sum_{i_k \subseteq A(T_j/X)} u(i_k, T_j) \tag{7}$$

For example, $ru(< d >, T_1) = u(< a >, T_1) + u(< c >, T_1) = 2 \times 5 + 3 \times 10 = 40$.

Definition 12: The utility of itemset X in a batch B_k is defined by formula (8):

$$u_{B_k}(X) = \sum_{T_j \in B_k} u(X, T_j) \tag{8}$$

TABLE I
TRANSACTION DATABASE

TID	$< a >$	$< b >$	$< c >$	$< d >$	$< e >$	tu
T_1	2	0	3	4	0	68
T_2	4	3	0	0	0	44
T_3	0	5	7	0	0	110
T_4	0	2	0	1	5	43
T_5	6	7	2	0	0	106
T_6	3	0	5	0	1	69
T_7	8	8	4	0	2	152
T_8	0	9	5	0	5	142

TABLE II
EXTERNAL UTILITY OF ITEMS

Item	eu(i)
$< a >$	5
$< b >$	8
$< c >$	10
$< d >$	7
$< e >$	4

For example, in Figure 1, and considering data in Table II, $u_{B_3}(< ac >) = u(< ac >, T_5) + u(< ac >, T_6) = 45 + 70 = 115$.

Definition 13: The utility of itemset X in a window W_l is defined by formula (9):

$$u_{W_l}(X) = \sum_{B_k \in W_l} u_{B_k}(X) \tag{9}$$

For example, in Figure 1, and considering data in Table II, $u_{W_2}(ac) = u_{B_2}(ac)+u_{B_3}(ac)+u_{B_4}(ac) = 0+115+80 = 195$.

Definition 14: The minimum utility threshold in a window W_l, denoted by δ_{W_l}, is a given percentage value based on the total utility value of all transactions in the W_l window. In Figure 1, the total utility of all transactions in window W_2 is 622. If $\delta_{W_l} = 20\%$, then the minimum utility in window W_l can be defined by formula (10):

$$minutil_{W_l} = \delta_{W_l} \times \sum_{T_j \in W_l} tu(T_j) \tag{10}$$

For example, with $\delta_{W_2} = 20\%$, the minimum utility value is $minutil_{W_2} = 20\% \times 622 = 124.4$.

An item set X is called a high-utility pattern in window W_l if $u_{W_l}(X) \geq minutil_{W_l}$. The problem of finding a high-utility pattern is defined as identifying all item sets X that satisfy the condition $u_{W_l}(X) \geq minutil_{W_l}$.

For example, for $\delta_{W_2} = 20\%$, the pattern $< ac >$ is a high-utility pattern in window W_2 because $u_{W_2}(< ac >) = 195$.

Definition 15: The transaction weight utility of an item set X in window W_l, denoted by $twu_{W_l}(X)$, is calculated as the total utility value of the transactions in window W_l containing X. For example, $twu_{W_2}(< ac >) = twu_{B_2}(< ac >)+twu_{B_3}(< ac >)+twu_{B_4}(< ac >) = 106+69+152 = 327$.

X is said to be a high transaction weight utility pattern in window W_l if $twu_{W_l}(X) \geq minutil_{W_l}$. For example, with $minutil_{W_2} = 124.4$, the pattern $< ac >$ is a high-weighted utility pattern because $twu_{W_2}(< ac >) = 327$.

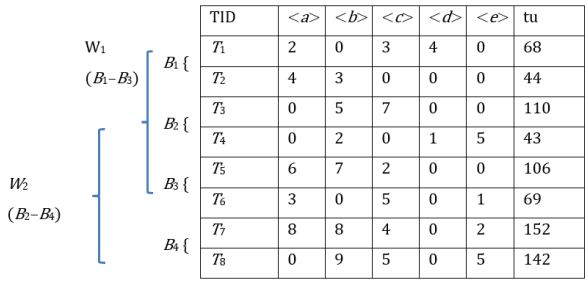
Figure 1 illustrates a transaction stream database. In this representation, the symbols T_i, with $1 \leq i \leq 8$, represent each transaction occurring the ith time. The letters in the set $\{< a >, < b >, < c >, < d >, < e >\}$ denote distinct items in the database. The digits in each row T_i in the column of letters in the set $\{< a >, < b >, < c >, < d >, < e >\}$ represent the internal utility of the item in transaction T_i. For example, in transaction T_1, item $< a >$ has an internal utility of 2. The symbol tu represents the utility of each transaction. The symbol B_j, with $1 \leq j \leq 4$, represents trading batches, each containing a certain number of transactions. Different batches have the same number of transactions, and transactions in each batch do not overlap. The notation W_k, with $1 \leq k \leq 2$, represents sliding windows, each containing a certain number of batches. Batches in different sliding windows may overlap or two different sliding windows may contain the same batches. As shown in Figure 1, sliding window W_2 can contain batches B_2 and B_3, which are also present in sliding window W_1. Window W_1 is the first window created, and it can only contain a maximum of three trading batches. After inserting batches B_1, B_2, and B_3 into window W_1, the window is full. Upon examining transaction T_7 in transaction batch B_4, sliding window W_1 moves to sliding window W_2. In the sliding window model, the algorithm only performs pattern mining on a sliding window containing a certain number of transaction batches.

The problem of mining high-utility patterns on transaction stream data is stated as follows: Given a transaction stream database D and a user-defined minimum utility threshold ($minutil$), mining high-utility patterns in the sliding window W is to find the set of all patterns X whose utility weight $twu(X)$ is greater than or equal to minutil in the sliding window W.

IV. THE PROPOSED ALGORITHM

a) High-Untility Stream Linked List Data

In this section, the article introduces a linked-list data structure called HUSLL, designed for storing information about transactions in the database.



TID	<a>	<b>	<c>	<d>	<e>	tu
T_1	2	0	3	4	0	68
T_2	4	3	0	0	0	44
T_3	0	5	7	0	0	110
T_4	0	2	0	1	5	43
T_5	6	7	2	0	0	106
T_6	3	0	5	0	1	69
T_7	8	8	4	0	2	152
T_8	0	9	5	0	5	142

Figure 1. Example of transaction stream data

The most apparent difference between the HUSLL structure and the Prefix Tree structure is the transformation of organizing nodes in the tree structure into nodes organized in a linked list. A sorted list can be used to execute the mining algorithm without the necessity of storing an intermediate structure mapping to nodes with the same item id. Additionally, deleting nodes in the tree requires more processing steps compared to handling a linked list. This is because the node deletion in a tree involves checking and traversing each child node in the branch to be deleted. In contrast, with a linked list, the deletion operation occurs by simply removing the node from the linked list via the next link, without needing to consider the linked child nodes of the node being examined.

For example, suppose that in window W_1 of batch B_1, there are two items $< m >$ and $< o >$ satisfying the threshold in T_1 and T_2. However, $< m >$ and $< o >$ do not appear in the transactions of window W_2. Therefore, when transitioning from W_1 to W_2, the data in batch B_1 will be deleted. Consequently, items $< m >$ and $< o >$ will also be deleted. In the case of organizing data in a tree structure, to delete the node containing $< m >$, we need to check if $< m >$ has any child nodes because the deletion operation will be performed on the child nodes first. Additionally, to perform mining on the tree using the HUPMS algorithm, an intermediate structure is needed to store the mapping to node $< m >$ in the tree. Therefore, deleting a node $< m >$ requires updating the intermediate structure. This operation consumes resources, especially in the case of continuous updates to the transaction stream database, where batch addition and deletion occur more frequently than in a static database. In contrast, when the data is organized as a linked list, node deletion is much simpler because there is no need to consider child nodes or update intermediate mapping structures.

Each node in the HUSLL structure stores identification information about an item (or event), the value of the node, and the values of transaction batches, which consist of multiple consecutive transactions containing that item. Transaction batches are numbered according to the order

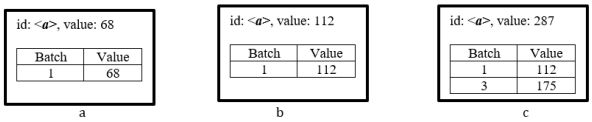


Figure 2. Structure of a node that stores information on item $< a >$

of transactions in the database. To achieve this, a hash table structure is employed to store information about the values of transaction batches. Utilizing a hash table for batch information facilitates quick updates of batch values when the mining algorithm examines transactions within a specific batch. Additionally, using a hash table helps conserve storage space by avoiding the need to store values of batches that do not contain the item under consideration. Deleting a batch within a node is also more straightforward with a hash table when there is sliding window movement, compared to using a list. Figure 2 illustrates the structure of a node in the list. Figure 2a demonstrates the structure of the node storing information about item $< a >$ with batch B_1 containing the value of transaction T_1 in Figure 1. Figure 2b illustrates the data update activity on batch B_1 when exploring transaction T_2. Figure 2c showcases the structure of the node storing information about item $< a >$ after exploring transactions in three batches: B_1, B_2, and B_3. Since item $< a >$ does not appear in batch B_2, information about batch B_2 is not stored in the list node. For ease of presentation, the node structure of item $< a >$ in Figure 2c is described as follows: $\{< a >: [1 : 112], [3 : 175]\}$. The value of a node represents the total value of the transaction batches that the node contains. In Figure 2c, the node value is 287, which is the sum of the values of batches B_1 and B_3.

To represent the relationships between nodes, links are established. An "after" link is employed to signify the sequential order of items in a transaction. For instance, in transaction T_1 in Figure 1, item $< c >$ appears after item $< a >$ in alphabetical order. Subsequently, an "after" link is created to indicate that item $< c >$ and item $< a >$ belong to the same transaction, and item $< c >$ follows item $< a >$. The "after" link plays a crucial role in constructing the HUSLL list when exploring transactions in the database. A single node can have multiple after-nodes associated with it. Once the list construction operation is completed, the after links can be removed to reduce memory size before the extraction process takes place. Figure 3 illustrates the after link connecting two nodes, $< a >$ and $< c >$.

The "before" link is employed to represent the preceding order of items within a transaction. For instance, in transaction T_1 in Figure 1, item $< a >$ precedes item $< c >$ in alphabetical order. A "before" link is then established

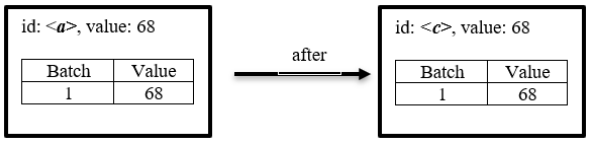


Figure 3. After link between two nodes $< a >$ and $< c >$

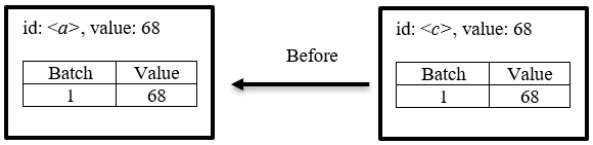


Figure 4. Before link between two nodes $< a >$ and $< c >$

to signify that item $< a >$ and item $< c >$ belong to the same transaction, and item $< a >$ precedes item $< c >$. The "before" link plays a crucial role in the high-utility pattern mining process. For example, if node $< c >$ has a "before" link with node $< a >$, a new pattern $< ac >$ can be created during the candidate generation process. Figure 4 illustrates the "before" link between node $< c >$.

A HUSLL root node is created by default with a null value. As each item in a transaction is examined, the root node establishes an after link with the node containing the first item in the transaction. These nodes also create a before link with the root node. Subsequently, when inspecting other items in the transaction, a new node is generated containing the item, along with before and after links between the nodes.

The next link is employed to connect all the nodes, forming a HUSLL structure. During the list-building process, the nodes created are linked together and sorted based on item identifier information. The next link is utilized in both the mining process and the update process of the HUSLL structure when the sliding window moves.

During mining operations, the algorithm considers items with the same identifier to generate patterns containing a specific identifier. The use of the next link facilitates the easy identification of patterns with the same identifier, as nodes are sorted by item identification. This approach avoids the necessity of using an intermediate mapping structure to locate nodes with the same identifier.

When a sliding window shift operation occurs, a loop traverses all the nodes in the HUSLL based on the next link to identify nodes with updated information about the transaction batch. Once found, a delete operation is performed to eliminate the selected batch of transactions and update the node's value. If the value of a node is 0, the node is removed from the HUSLL by deleting the before, after, and next links to that node. The deletion

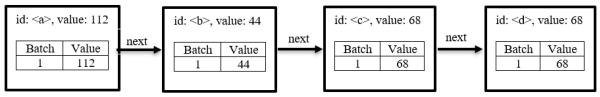


Figure 5. Next link between nodes in transactions T_1 and T_2

of such a link is simpler than updating a tree structure. Simultaneously, not using an intermediate structure to store the links mapping to the node allows update operations on HUSLL to be faster than on the HUS-Tree structure. Figure 5 illustrates the next link between the nodes in transactions T_1 and T_2 in Figure 1. In this figure, transaction T_1, the node containing item $< c >$, has a before link with the node containing item $< a >$ and has no connection with the node containing item $< b >$. However, because item $< b >$ is placed before item $< c >$ in the alphabet, when the next link is created, item $< b >$ will be positioned before item $< c >$.

The process of constructing the HUSLL structure involves creating nodes to store item information and establishing before, after, and next links connecting these nodes. Initially, an empty root node is created. Utilizing the information from Table I, the algorithm sequentially goes through each item in each transaction in alphabetical order to add each item to the HUSLL structure. For example, during the traversal operation of transaction T_1 (with a total utility value, tu, of 68 and items $< a, c, d >$), the item $< a >$ is added to the list by creating a new node with item-id $< a >$ and twu value 68. The resulting node is $\{< a >: [1 : 68]\}$. A next link is established from the root node to $\{< a >: [1 : 68]\}$, and the root node will have an after link to $\{< a >: [1 : 68]\}$. Simultaneously, $\{< a >: [1 : 68]\}$ will have a before link to the root node. Subsequently, item $< c >$ is added to the HUSLL by creating the node $\{< c >: [1 : 68]\}$. Since, in transaction T_1, item $< c >$ comes after item $< a >$, an after link is created between node $\{< c >: [1 : 68]\}$ and node $\{< a >: [1 : 68]\}$. Additionally, a before link is established between node $\{< a >: [1 : 68]\}$ and node $\{< c >: [1 : 68]\}$. Node $\{< c >: [1 : 68]\}$ is inserted at the end of the HUSLL list, creating a next link between node $\{< a >: [1 : 68]\}$ and node $\{< c >: [1 : 68]\}$. Similarly, item $< d >$ is added to the HUSLL by creating node $\{< d >: [1 : 68]\}$. As item $< d >$ follows item $< c >$ in transaction T_1, a before link is formed from node $\{< d >: [1 : 68]\}$ to node $\{< c >: [1 : 68]\}$. Two links, after and next, are generated at node $\{< c >: [1 : 68]\}$ to point to node $\{< d >: [1 : 68]\}$. Figure 6 illustrates the HUSLL structure that stores the items contained in transaction T_1.

For simplicity, the HUSLL structure is represented as

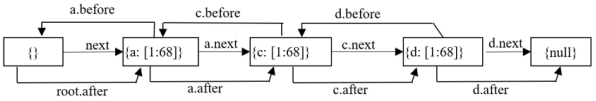


Figure 6. The HUSLL stores information on transaction T_1

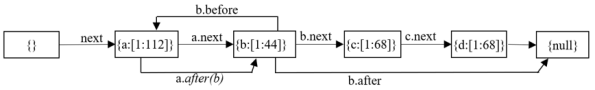


Figure 7. The HUSLL after adding transaction T_2

$\{\} \rightarrow \{< a >: [1 : 68]\} \rightarrow \{< c >: [1 : 68]\} \rightarrow \{< d >: [1 : 68]\}$, where the direction of the arrow signifies the next link.

Next, transaction T_2 is processed to add items to the HUSLL. With a *tu* value of 44, T_2 includes 2 items $< a, b >$. Item $< a >$ from T_2 is incorporated into the HUSLL, but since there is already node $< a >$ with an after link to the root node, and transactions T_1 and T_2 belong to the same batch B_1 (as shown in Figure 1), the *twu* value of item $< a >$ in T_2 is accumulated to node $\{< a >: [1 : 68]\}$, resulting in $\{< a >: [1 : 112]\}$. Subsequently, item $< b >$ is added as the next node after $\{< a >: [1 : 112]\}$ in alphabetical order. The HUSLL structure is designed to maintain alphabetical order, placing item $< b >$ before item $< c >$. Figure 7 illustrates the HUSLL structure after incorporating transaction T_2, with some details omitted for clarity.

Moving on, transaction T_3 is processed to add items to the HUSLL. With a tu value of 110, T_3 includes two items, $< b, c >$. Item $< b >$ from T_3 is introduced into the HUSLL with a *twu* value of 110. Given that there is no after link from the root node to the identifier $< b >$, a new node is created. Additionally, since T_3 belongs to transaction batch B_2, the newly added node will take the form $\{< b >: [2 : 110]\}$. As the list already contains node $\{< b >: [1 : 44]\}$, the new node will be inserted after node $\{< b >: [1 : 44]\}$. Similarly, a new node $\{< c >: [2 : 110]\}$ is generated and added after node $\{< c >: [1 : 68]\}$. Since a new transaction batch is involved, nodes $\{< b >: [2 : 110]\}$ and $\{< c >: [2 : 110]\}$ only store the value of the current batch, which is batch 2. Figure 8 illustrates the HUSLL structure after incorporating transaction T_3.

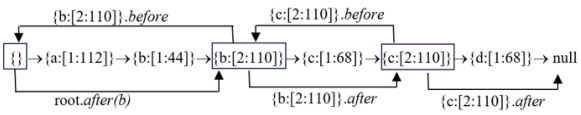


Figure 8. The HUSLL after adding transaction T_3

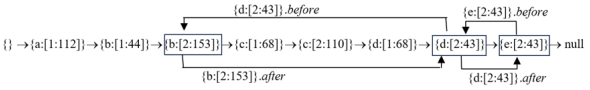


Figure 9. The HUSLL after adding transaction T_4

$\{\} \rightarrow \{< a >: [1 : 112], [3 : 106]\} \rightarrow \{< b >: [1 : 44], [3 : 106]\} \rightarrow \{< b >: [2 : 153]\} \rightarrow \{< c >: [1 : 68]\} \rightarrow \{< c >: [2 : 110]\} \rightarrow \{< c >: [3 : 106]\} \rightarrow \{< d >: [1 : 68]\} \rightarrow \{< d >: [2 : 43]\} \rightarrow \{< e >: [2 : 43]\} \rightarrow null$

Figure 10. The HUSLL after adding transaction T_5

Continuing with the process, transaction T_4 is processed to add items to the HUSLL. With a *tu* value of 43, T_4 encompasses three items: $< b, d, e >$. As the root node has an after link to node $\{< b >: [2 : 110]\}$ and transaction T_4 belongs to batch B_2, the *tu* value of T_4 should be added to node $\{< b >: [2 : 110]\}$ to result in $\{< b >: [2 : 153]\}$. Subsequently, two new nodes $\{< d >: [2 : 43]\}$ and $\{< e >: [2 : 43]\}$ are generated and appended to the list in the following order: $\{< d >: [1 : 68]\}$. Figure 9 illustrates the list structure after the addition of transaction T_4.

Moving forward, transaction T_5 is processed to add items to the HUSLL. With a *tu* value of 106, T_5 includes three items: $< a, b, c >$. Given that the root node already has an after link to node $\{< a >: [1 : 112]\}$ and transaction T_5 belongs to batch B_3, we update node $\{< a >: [1 : 112]\}$ to $\{< a >: [1 : 112], [3 : 106]\}$. The order of *twu* values for batches B_1 and B_3 in node $\{< a >: [1 : 112], [3 : 106]\}$ is represented in the hash table structure. Since there is an after link from node $\{< a >: [1 : 112], [3 : 106]\}$ to node $\{< b >: [1 : 44]\}$, it is updated to $\{< b >: [1 : 44], [3 : 106]\}$. As there is no after link after $\{< b >: [1 : 44], [3 : 106]\}$ to the node with the identifier $< c >$, a new node $\{< c >: [3 : 106]\}$ is created and connected after node $\{< c >: [2 : 110]\}$. The HUSLL structure after adding items in transaction T_5 is illustrated in Figure 10.

Figure 10 illustrates the HUSLL structure after incorporating the items from transaction T_5. In this representation, link information is omitted for simplicity.

Continuing with the process, transaction T_6 is processed to add items to the HUSLL, and the outcomes are illustrated in Figure 11.

With the HUSLL structure, when the window shifts from W_1 to W_2, the data in transaction batch B_1 is removed, and the data from transaction batch B_4 is added to the list. To delete the data of transaction batch B_1, the HUSLL structure is traversed from the first node to the last node based on the next link. At each node traversal, the data

$\{\} \rightarrow \{< a >: [1:112], [3:175]\} \rightarrow \{< b >: [1:44], [3:106]\} \rightarrow \{< b >: [2:153]\} \rightarrow \{< c >: [1:68], [3:69]\} \rightarrow \{< c >: [2:110]\} \rightarrow \{< c >: [3:106]\} \rightarrow \{< d >: [1:68]\} \rightarrow \{< d >: [2:43]\} \rightarrow \{< e >: [2:43]\} \rightarrow \{< e >: [3:69]\} \rightarrow null$

Figure 11. The HUSLL after adding transaction T_6

(twu) of batch B_1 is deleted from the hash table. Utilizing a hash table structure makes finding and deleting the value of batch B_1 in each node easier compared to using a list of batch values. Subsequently, nodes with a twu value of 0 are removed from the list. For example, from the HUSLL list in Figure 11, when the algorithm reaches transaction T_7, the sliding window will shift from window W_1 to W_2. At that point, all the values of batch 1 in each node will be deleted. A loop will iterate through each node in the HUSLL list to find and delete batch 1 data in each node. In Figure 11, we can see that the nodes $\{< a >: [1:112], [3:175]\}$, $\{< b >: [1:44], [3:106]\}$, $\{< c >: [1:68], [3:69]\}$, and $\{< d >: [1:68]\}$ contain data of batch 1. After performing the deletion operation, the values of these nodes will be $\{< a >: [3:175]\}$, $\{< b >: [3:106]\}$, $\{< c >: [3:69]\}$, and $\{< d >:\}$. At this point, the node $\{< d >:\}$ does not contain any data and will be removed from the list. The deletion operation is performed by deleting all links created to the node $\{< d >:\}$ and the next link of the node just before the node $\{< d >:\}$, which is the node $\{< c >: [3:106]\}$ in Figure 11, will be directly linked to the node just after $\{< d >:\}$, which is the node $\{< d >: [2:43]\}$. Finally, the items of transaction batch B_4 are incorporated into the HUSLL. The advantage of maintaining a HUSLL list is that traversing and deleting nodes is considerably more straightforward than with the HUS-Tree structure. The use of a hash table enables the storage of only batches with the item identifier present, without having to store zero values, as is the case in the HUS-Tree structure. Consequently, the update only occurs on nodes containing the batch under consideration, and the deletion of batch B_1 is performed solely on nodes where this batch appears, such as nodes $\{< a >: [1:112], [3:175]\}$, $\{< b >: [1:44], [3:106]\}$, $\{< c >: [1:68], [3:69]\}$, and $\{< d >: [1:68]\}$, instead of having to update all nodes as required by the HUS-Tree structure. Figure 12 illustrates the outcome of the shift operation to window W_2.

The process of constructing the HUSLL list is detailed in algorithm 1.

b) Mining High-Ultility Patterns on the HUSLL List

From the list structure in Figure 12, we can recognize the following characteristics:

Property 1. Nodes containing information about items appearing in the same transaction will possess a "before" link to a node containing information about the item whose identifier immediately precedes the item identifier of the node under consideration. For instance, in Figure 12, the node $\{< e >: [3:69]\}$ has a "before" link to node $\{< c >: [3:69]\}$, and node $\{< c >: [3:69]\}$ has a "before" link to node $\{< a >: [3:175], [4:152]\}$. Simultaneously, the items $< a, c, e >$ appear in the same transactions T_6 and T_7.

Property 2. A high-utility pattern can be generated by com-

$\{\} \rightarrow \{< a >: [3:175], [4:152]\} \rightarrow \{< b >: [3:106], [4:152]\} \rightarrow \{< b >: [2:153], [4:142]\} \rightarrow \{< c >: [3:69]\} \rightarrow \{< c >: [2:110], [4:142]\} \rightarrow \{< c >: [3:106], [4:152]\} \rightarrow \{< d >: [2:43]\} \rightarrow \{< e >: [2:43]\} \rightarrow \{< e >: [3:69]\} \rightarrow \{< e >: [4:152]\} \rightarrow \{< e >: [4:142]\} \rightarrow null$

Figure 12. The HUSLL after sliding to window W_2

Algorithm 1: BuildHUSLL (D, minutil)

```
1  Inputs: Transaction database D, minimum utility
     threshold minutil.
2  Outputs: HUSLL data structure.
3  begin
4      Create root-node R;
5      Set current node C is null;
6      foreach T ∈ D do
7          i=T[0];
8          if R.hasAfter(i) = False then
9              CreateNode for i AS iN;
10             R.setAfter(iN);
11             iN.setBefore = R;
12             call AddToHUSLL (iN,R);
13             C=iN;
14         end
15         else
16             C= R.getNode(i);
17             Update twu of C;
18         end
19         k=1;
20         while k < Length(T) do
21             i=T[k];
22             if C.hasAfter(i) = TRUE then
23                 C=C.getNode(i);
24                 Update twu of C;
25             end
26             else
27                 CreateNode for i AS iN;
28                 C.setAfter(iN);
29                 iN.setBefore = C;
30                 AddToHUSLL (iN,R);
31                 C=iN;
32             end
33             k=k+1;
34         end
35     end
36 end
```

Algorithm 2: AddToHUSLL (iN,R)

```
Inputs: considered node iN, root node R
Outputs: HUSLL
begin
    C = R;
    while c.next is not null do
        if c.next > iN.id then
            iN.next = c.next;
            c.next = iN;
            return;
        end
    end
    c.next = iN
end
```

bining items in the same branch through the "before" link, and the utility value of the pattern is the list of values of the transaction batch of the last node in the branch. For example, in Figure 12, the "before" link of the nodes includes nodes $\{< a >: [3 : 175], [4 : 152]\}$ and node $\{< c >: [3 : 69]\}$. The node $\{< e >: [3 : 69]\}$ can generate the pattern $< ec >$ by concatenating the identifiers of node $\{< c >: [3 : 69]\}$ and node $\{< e >: [3 : 69]\}$. Subsequently, the utility of the $< ec >$ pattern will be the transaction batch value of the first node in the "before" link, which is node $\{< e >: [3 : 69]\}$. The utility value of the pattern $< ec >$ is $\{< ec >: [3 : 69]\}$. Then, the pattern $< eca >$ can be generated by concatenating the identifier of node $\{< ec >: [3 : 69]\}$ and node $\{< a >: [3 : 175], [4 : 152]\}$. Since, in the "before" link, node $\{< e >: [3 : 69]\}$ is the first node, the utility of the $< eca >$ pattern is the transaction batch value of node $\{< e >: [3 : 69]\}$. The utility value of the $< eca >$ pattern is $\{< eca >: [3 : 69]\}$.

Based on the two characteristics mentioned above, the pattern mining process on HUSLL can be outlined as follows: From the existing HUSLL list, iterate through each node using the next link. For each node, determine the linked branch based on the before link. For each branch, concatenate the item identifiers of each node within the branch to create a candidate pattern. Next, assess the utility of the candidate pattern by summing the utility values in the transaction batch of that pattern and comparing it with the minutil value. Candidate patterns exceeding the minutil threshold will be retained, and the concatenation operation to create new candidate patterns will be iterated. After this browsing and merging process, all high-utility patterns will be obtained. Algorithm 3 outlines the process of mining high-utility patterns from the HUSLL list.

For instance, with *minutil* set to 260, item $< c >$ has a utility weight equal to the total utility value in the trading batches, summing up to 69 + 106 + 152 + 110 + 142 = 579. Next, retrieve the id_item_list of nodes containing item $< c >$ in HUSLL, which includes $\{< c >: [3 : 69]\}, \{< c >: [2 : 110], [4 : 142]\}$, and $\{< c >: [3 : 106], [4 : 152]\}$. Starting with node $\{< c >: [3 : 69]\}$, traverse the list based on the before link to generate a pattern $\{< ca >: [3 : 69]\}$. For node $\{< c >: [3 : 106], [4 : 152]\}$, perform a list traversal based on the before link to generate patterns $\{< cb >: [3 : 106], [4 : 152]\}$ and $\{< ca >: [3 : 106], [4 : 152]\}$. From node $\{< c >: [2 : 110], [4 : 142]\}$, traverse the list based on the before link to generate the pattern $\{< cb >: [2 : 110], [4 : 142]\}$. Then, calculate the utility weights of the patterns by summing up the utility values. With the pattern $< ca >$, comprising $\{< ca >: [3 : 69]\}$ and $\{< ca >: [3 : 106], [4 : 152]\}$, the utility is 69 + 106 + 152 = 327. For pattern $< cb >$, consisting of $\{< cb >: [3 : 106], [4 : 152]\}$ and $\{< cb >: [2 : 110], [4 : 142]\}$, the utility weight is 106 + 152 +110 + 142 = 510. Both patterns $< ca >$ and $< cb >$ qualify as high-utility patterns since their utility weights exceed *minutil*. The pattern generation process continues with $< ca >$ and $< cb >$. For the pattern $< ca >$, the node containing $< a >$ is the one with the root node as the before link, so no further patterns can emerge. However, for the pattern $< cb >$, comprising two nodes $\{< cb >: [3 : 106], [4 : 152]\}$ and $\{< cb >: [2 : 110], [4 : 142]\}$, the node $\{< cb >: [2 : 110], [4 : 142]\}$ has node $< b >$ with the before link as the root node, indicating that no additional patterns can be generated. Regarding the node $\{< cb >: [3 : 106], [4 : 152]\}$, the preceding node is $\{< a >: 0, 175, 152\}$, enabling the pattern $\{< cba >: [3 : 106], [4 : 152]\}$ to arise. However, the pattern $< cba >$ with a utility weight of 106 + 152 = 258 does not meet the criteria for a high-utility pattern. The mining process persists until all nodes in HUSLL are traversed.

The HUSLL structure shares similarities with the HUS-Tree structure, but it differs in the use of after links during the extraction process. In contrast to the HUPMS algorithm, which relies on intermediate structures like condition trees, the HSLM algorithm utilizes the previous link, rendering after links unnecessary for mining operations. As a result, after links can be removed during mining to optimize memory usage. Each node in the HUSLL structure contains information such as the item identifier, total batch value, and before and next links. To enhance performance, mining information can be stored in text files for separate mining operations on another processor. This allows for the optimization of both mining structure construction and mining activities by dividing these tasks. One processor can focus on building the HUSLL structure and continuously updating the data, while mining operations can be carried out on one or more separate processors.

Additionally, the use of a hash table to store information

Algorithm 3: HSLM (T, minutil)

```
Inputs: HUSLL T, minimum utility threshold (minutil)
Outputs: High-utility patterns
begin
    foreach id ∈ T do
        Identify the nodes belonging to the branch
          associated with the id;
        Add the nodes to the id_item_list ;
        foreach node n ∈ id_item_list do
            Browse T and combine node n with before
              nodes of n to create pattern nx;
            Save nx in K;
        end
        foreach pattern m ∈ K do
            if utility of m ≥ minutil then
                MiningSubPattern(T,m,minsup);
                Output m;
            end
        end
    end
end
```

Algorithm 4: MiningSubPattern (T,minutil,m)

```
Inputs: HUSLL T, minimum utility threshold (minutil),
  pattern m
Outputs: High-utility patten (mx)
begin
    Create L to save list;
    Browse T and combine node m with before nodes of
      m to create pattern mx;
    Save mx to L;
    foreach pattern mx∈L do
        if utility of mx ≥ minutil then
            MiningSubPattern(T,mx,minsup);
            Output mx;
        end
    end
end
```

about item batches in each node facilitates easy updating of node information when the sliding window moves. This hash table structure speeds up transaction searches and optimizes storage operations by excluding information about batches that do not contain the item identifier of the node under consideration. The next link enables straightforward node deletion operations. When a node's value reaches 0, the node is removed from the HUSLL list by deleting all before, after, and next links and updating the value to null. In contrast, the HUS-Tree structure requires more complex deletion operations, especially for branch nodes, where updating is not easily accomplished. Traversing the next link from the root node to the last node in the list simplifies state updates for the entire mining structure.

V. EXPERIMENTAL RESULTS

In this section, we present the experimental results comparing the effectiveness of the HUPMS [11] algorithm , EGUI-TREE [27] algorithm and the proposed HSLM algorithm. The study utilizes four real-life datasets for comparison: chainstore, foodmart, ecommerce, and chicago_crimes. Detailed information about each dataset is provided in Table III. The program is implemented in Java and executed on a system running Windows 10, equipped with an Intel® Core™ i7-9700 processor and 16384 MB RAM.

TABLE III
EXPERIMENTAL DATA

Dataset name	Trans No.	Item No.	AvgLen
foodmart_utility	4,141	1,559	4.42
chainstore_utility	1,112,949	46,086	7.23
ECommerce_retail_utility	14,975	3,468	11.71
Chicago_Crimes	2,662,309	35	1.795

The input data for all three algorithms is a transactional database with three parameters: (i) the number of transactions per batch, (ii) the number of batches in each window, and (iii) the minimum useful threshold value, *minutil*. In contrast to the HUPMS algorithm, the EGUI-Tree algorithm incorporates the *ru* value for every pattern to facilitate the early pruning of unpromising patterns. Specifically, in the first phase, the EGUI-Tree algorithm constructs the global utility 1-item sets to store information about the *tu* and *ru* values of each item per transaction and batch for the current window. Next, based on the *ru* value, EGUI-Tree decides whether to expand the items in the 1-item set to create candidate patterns and add these candidate patterns to the EGUI-Tree. Each node in this structure stores information about the pattern. The EGUI-Tree is subdivided into various levels. The candidate patterns at each level have the same number of items in each pattern. Finally, the algorithm traverses the EGUI-Tree to find high utility patterns by comparing the utility value of each pattern stored on the tree with a minutil. The algorithm updates the information for both the global utility 1-item sets and the EGUI-Tree whenever the window changes.

a) Comparision of execution time when performing

In this section, we present experimental results from the high-utility pattern mining process on various datasets, with a focus on adjusting the number of batches per window, ranging from 2 to 6 for different datasets. Each dataset is associated with specific parameters. For the chainstore dataset, the minimum utility threshold is set to 0.25%, and the number of transactions per batch is 100K. The foodmart dataset has a minimum utility threshold of 0.25%, with 200 transactions per batch. In the case of the ecommerce dataset, the minimum utility threshold is 0.75%, and there are 2000 transactions per batch. Lastly, for the Chicago dataset, a minimum utility threshold of 0.15% is chosen, and there are 300K transactions per batch. The article

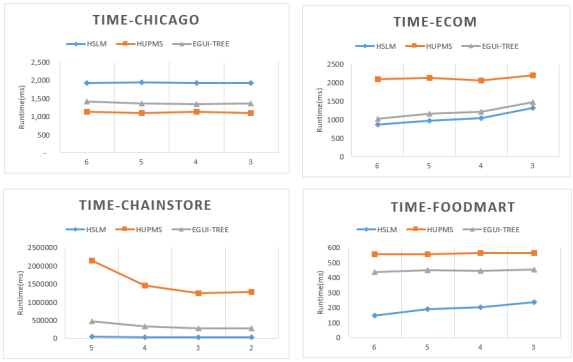


Figure 13. Comparison results according to execution time

compares the execution time to assess the effectiveness of both the proposed algorithm (HSLM) and the HUPMS algorithm. The results depicted in Figure 13 illustrate the execution times.

Figure 13 reveals that for highly dense datasets like chainstore, ecommerce, and foodmart, the HUPMS algorithm requires operations involving the construction of prefix trees, condition trees, and utility value calculations for patterns. As a result, HUPMS utilizes more execution time and memory. Conversely, HSLM can perform extraction without the need to generate prefix trees and condition trees, leading to a lower execution time for the proposed HSLM algorithm. Additionally, the EGUI-Tree algorithm performs at a faster rate than the HUPMS algorithm on these datasets. However, the update operation of the EGUI-Tree requires traversing the batch list to update patterns, which consumes more time than the HSLM algorithm. Besides, with sparse datasets like Chicago, where the density is very low, the time spent creating and deleting the condition tree and prefix tree of the HUPMS algorithm becomes negligible. In such cases, the execution time of HUPMS surpasses that of the proposed algorithm.

The above experimental results indicate that the proposed HSLM algorithm outperforms the HUPMS algorithm on dense datasets. Additionally, the results reveal that as the number of batches per window increases, the mining time also tends to increase.

b) Comparision of excution time with different batch

In this section, the article examines the experimental results by varying the number of transactions in each batch. Figure 14 illustrates the experimental outcomes on the chainstore dataset. In Figure 14a, as the number of transactions per batch increases from 50K to 100K, the HUPMS, EGUI-Tree, and HSLM algorithms exhibit a decrease in mining time. Conversely, when reducing the number of transactions per batch, the mining time

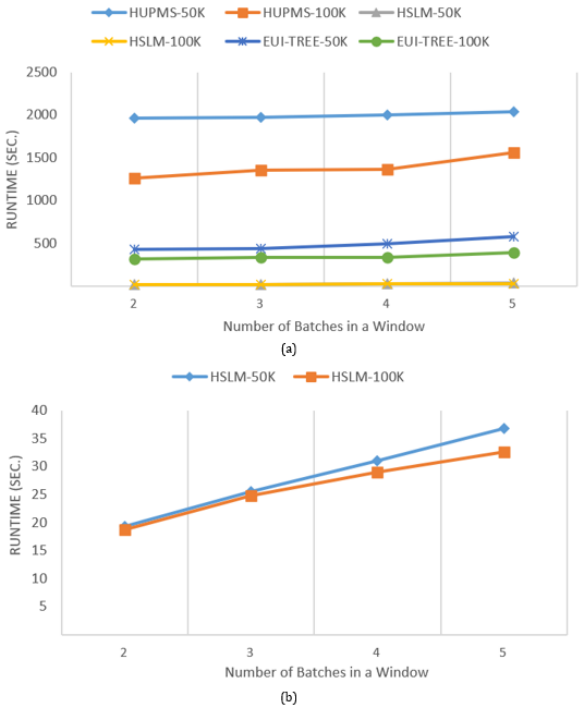


Figure 14. Comparison results based on the number of transactions per batch

increases correspondingly. Figure 14b specifically presents the mining time of the HSLM algorithm on the chainstore dataset while varying the number of transactions per batch from 50K to 100K. The results indicate that a decrease in the number of transactions per batch leads to an increase in the number of mining algorithm executions, subsequently elevating the overall mining time.

VI. CONCLUSIONS

This article introduced a high-utility pattern mining technique for transaction stream data based on the HUSLL list data structure. The experimental results demonstrated that the proposed method outperforms the HUPMS algorithm, particularly on dense datasets such as chainstore, foodmart, or ecommerce.

The HUSLL structure, resembling the HUS-Tree structure, offers the advantage of not using after links in mining operations. This feature facilitates storing HUSLL list information in text files, enabling external memory storage. Consequently, mining operations can be conducted on a different processor using the file containing the HUSLL list structure. This separation of update and exploitation steps allows efficient handling of the latest sliding window information stored in the text file without the need to rerun the algorithm from the beginning. Storing in text files also enables users to retain information about data status at

various exploitation stages, facilitating the comparison of patterns and strategic planning for subsequent stages.

Moreover, the linked list organization proves to be more effective in updating the state compared to tree data organization, particularly in a stream data mining environment where transaction information undergoes regular updates. This adaptability contributes to the practical effectiveness of the HSLM algorithm.

The proposed algorithm's performance is not particularly outstanding in sparse datasets. This is a limitation when mining high-utility sets for sparse datasets. Moreover, the process of building the linked list sequentially does not reflect parallel processing during list creation. Therefore, the mining has not been optimized for the case of a large window size. Our research will continue to focus on parallelizing the list-building process and using early candidate pruning techniques to optimize mining in the future.

The utilization of textual data opens avenues for research in parallel and simultaneous processing of multiple datasets from various sources. It also provides users with the flexibility to transform shared data while retaining valuable information gleaned from exploitation activities.

Additionally, the mining of closed high-utility sets of stream data via a sliding window [29] helps eliminate highly useful but non-repeating patterns to reduce the time and space required for mining, which also needs to be further researched.

ACKNOWLEDGEMENT

This research is funded by Ho Chi Minh City University of Foreign Languages – Information Technology under grant number H2022-05.

REFERENCES

[1] R. Agrawal, R. Srikant *et al.*, "Fast algorithms for mining association rules," in *Proc. 20th int. conf. very large data bases, VLDB*, vol. 1215. Santiago, Chile, 1994, pp. 487–499.

[2] P. Fournier-Viger, J. Chun-Wei Lin, T. Truong-Chi, and R. Nkambou, "A survey of high-utility itemset mining," *High-utility pattern mining: Theory, algorithms and applications*, pp. 1–45, 2019.

[3] S. Krishnamoorthy, "Hminer: Efficiently mining high-utility itemsets," *Expert Systems with Applications*, vol. 90, pp. 168–183, 2017.

[4] S. Zida, P. Fournier-Viger, J. C.-W. Lin, C.-W. Wu, and V. S. Tseng, "Efim: a fast and memory efficient algorithm for high-utility itemset mining," *Knowledge and Information Systems*, vol. 51, no. 2, pp. 595–625, 2017.

[5] M. K. Sohrabi, "An efficient projection-based method for high utility itemset mining using a novel pruning approach on the utility matrix," *Knowledge and Information Systems*, vol. 62, pp. 4141–4167, 2020.

[6] P. Wu, X. Niu, P. Fournier-Viger, C. Huang, and B. Wang, "Ubp-miner: An efficient bit based high utility itemset mining algorithm," *Knowledge-Based Systems*, vol. 248, p. 108865, 2022.

[7] J. C.-W. Lin, W. Gan, T.-P. Hong, B. Zhang *et al.*, "An incremental high-utility mining algorithm with transaction insertion," *The Scientific World Journal*, vol. 2015, 2015.

[8] P. Fournier-Viger, J. C.-W. Lin, T. Gueniche, and P. Barhate, "Efficient incremental high-utility itemset mining," in *Proceedings of the ASE BigData & SocialInformatics 2015*, 2015, pp. 1–6.

[9] U. Yun, H. Ryang, G. Lee, and H. Fujita, "An efficient algorithm for mining high-utility patterns from incremental databases with one database scan," *Knowledge-Based Systems*, vol. 124, pp. 188–206, 2017.

[10] H.-F. Li, H.-Y. Huang, Y.-C. Chen, Y.-J. Liu, and S.-Y. Lee, "Fast and memory efficient mining of high-utility itemsets in data streams," in *2008 eighth IEEE international conference on data mining*. IEEE, 2008, pp. 881–886.

[11] C. F. Ahmed, S. K. Tanbeer, and B.-S. Jeong, "Efficient mining of high-utility patterns over data streams with a sliding window method," *Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing 2010*, pp. 99–113, 2010.

[12] J. Han, J. Pei, and Y. Yin, "Mining frequent patterns without candidate generation," *ACM sigmod record*, vol. 29, no. 2, pp. 1–12, 2000.

[13] G. Pyun, U. Yun, and K. H. Ryu, "Efficient frequent pattern mining based on linear prefix tree," *Knowledge-Based Systems*, vol. 55, pp. 125–139, 2014.

[14] Y. Mao, B. Wu, Q. Deng, S. Mahmoodi, Z. Chen, and Y.-C. Chen, "A novel parallel frequent itemset mining algorithm for automatic enterprise," *Enterprise Information Systems*, p. 2204317, 2023.

[15] J. Lu, W. Xu, K. Zhou, and Z. Guo, "Frequent itemset mining algorithm based on linear table," *Journal of Database Management (JDM)*, vol. 34, no. 1, pp. 1–21, 2023.

[16] P. Fournier-Viger, C.-W. Wu, S. Zida, and V. S. Tseng, "Fhm: Faster high-utility itemset mining using estimated utility co-occurrence pruning," in *Foundations of Intelligent Systems: 21st International Symposium, ISMIS 2014, Roskilde, Denmark, June 25-27, 2014. Proceedings 21*. Springer, 2014, pp. 83–92.

[17] J. Liu, K. Wang, and B. CM Fung, "Direct discovery of high-utility itemsets without candidate generation," in *2012 IEEE 12th international conference on data mining*, 2012, pp. 984–989.

[18] Y. Liu, W.-k. Liao, and A. Choudhary, "A two-phase algorithm for fast discovery of high-utility itemsets," in *Advances in Knowledge Discovery and Data Mining: 9th Pacific-Asia Conference, PAKDD 2005, Hanoi, Vietnam, May 18-20, 2005. Proceedings 9*. Springer, 2005, pp. 689–695.

[19] J.-S. Yeh, C.-Y. Chang, and Y.-T. Wang, "Efficient algorithms for incremental utility mining," in *Proceedings of the 2nd international conference on Ubiquitous information management and communication*, 2008, pp. 212–217.

[20] C. F. Ahmed, S. K. Tanbeer, B.-S. Jeong, and Y.-K. Lee, "Efficient tree structures for high-utility pattern mining in incremental databases," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 12, pp. 1708–1721, 2009.

[21] H.-T. Zheng and Z. Li, "ichum: an efficient algorithm for high-utility mining in incremental databases," in *Knowledge Science, Engineering and Management: 8th International Conference, KSEM 2015, Chongqing, China, October 28-30, 2015, Proceedings 8*. Springer, 2015, pp. 212–223.

[22] U. Yun and H. Ryang, "Incremental high-utility pattern mining with static and dynamic databases," *Applied intelligence*, vol. 42, pp. 323–352, 2015.

[23] M. Zihayat, Y. Chen, and A. An, "Memory-adaptive high

utility sequential pattern mining over data streams," *Machine Learning*, vol. 106, pp. 799–836, 2017.
[24] D. Kim and U. Yun, "Mining high utility itemsets based on the time decaying model," *Intelligent Data Analysis*, vol. 20, no. 5, pp. 1157–1180, 2016.
[25] U. Yun, D. Kim, E. Yoon, and H. Fujita, "Damped window based high average utility pattern mining over data streams," *Knowledge-Based Systems*, vol. 144, pp. 188–205, 2018.
[26] H. Ryang and U. Yun, "High utility pattern mining over data streams with sliding window technique," *Expert Systems with Applications*, vol. 57, pp. 214–231, 2016.
[27] P. A. Reddy and M. K. Prasad, "Sliding window-based high utility item-sets mining over data stream using extended global utility item-sets tree," *International Journal of Software Innovation (IJSI)*, vol. 10, no. 1, pp. 1–16, 2022.
[28] H. Yao and H. J. Hamilton, "Mining itemset utilities from transaction databases," *Data & Knowledge Engineering*, vol. 59, no. 3, pp. 603–626, 2006.
[29] M. Li, M. Han, Z. Chen, H. Wu, and X. Zhang, "Fchm-stream: fast closed high utility itemsets mining over data streams," *Knowledge and Information Systems*, vol. 65, no. 6, pp. 2509–2539, 2023.



Dr. Tran Minh Thai received a Bachelor's degree in 2001 and a Master's degree in Computer Science in 2006, both from the University of Natural Sciences, Vietnam National University, HCMC. In 2017, he was awarded a Ph.D. in Computer Science from Vietnam National University, HCMC. From 2002 to 2015, he served as a lecturer and manager of the Information Technology Department at the Information Technology College HCMC. Since 2015, he has been a lecturer and Head of the Information Systems Department of the Faculty of Information Technology at the HCMC University of Foreign Languages - Information Technology. Dr. Thai's primary research interests lie in data mining, data privacy, big data processing, and pattern recognition.
Email: thaitm@huflit.edu.vn



Anh-Duy Tran received a Master's degree in Computer Science from the University of Natural Sciences, Vietnam National University, HCMC in 2017. He serves as a Lecturer at the Department of Information Technology at the University of Foreign Languages and Information Technology in Ho Chi Minh City. His contributions to the field of data mining aim to empower businesses and organizations to make data-driven decisions, fostering innovation and advancements in various domains.
Email: duyta@huflit.edu.vn



Duc-Thanh Pham received his Master's degree in 2006 from Ho Chi Minh City National University; Currently working as a Lecturer at the Faculty of Information Technology, Ho Chi Minh City University of Foreign Languages and Information Technology; The research field of interest is: data mining
Email: thanhpd@huflit.edu.vn



Minh-Nguyen Le received his Master's degree in 2007 from Ho Chi Minh City National University; Currently working as a Lecturer at the Faculty of Information Technology, Ho Chi Minh City University of Foreign Languages and Information Technology; The research field of interest is: data mining.
Email: nguyenltm@huflit.edu.vn

Mining Association Rules on Time-sensitive Data

Dang-Ngo Huu[1,3], Nghia Le[2,3], Khac-Chien Nguyen[4]
[1]University of Science, Ho Chi Minh City, Viet Nam, Vietnam
[2]University of Information Technology, Ho Chi Minh City, Viet Nam, Vietnam
[3]Vietnam National University, Ho Chi Minh City, Vietnam
[4]People's Police University, Ho Chi Minh City, Vietnam

Correspondence: Khac-Chien Nguyen, nkchienster@gmail.com
Communication: received 13 Feb 2024, revised 15 Apr 2024, accepted 26 May 2024
Digital Object Identifier: 10.32913/mic-ict-research.v2024.n2.1256

Abstract: Real-world data is continuously produced and collected quickly; hence, mining association rules are used to find relationships between item sets. One of these combined rule mining techniques is Rare Association Rule Mining (RARM), a method that extracts rare, combined rules with low support but high confidence from the database. Additionally, a large amount of time-sensitive data is created in many fields, and combining valid and outdated data leads to low efficiency in extracting combined rules. This paper explores a mining method that increases rare, combined rules on time-sensitive data to extract common and rare patterns in the database. The primary approach of the research proposes a new tree structure called ILC-tree, inspired by the LC-tree structure published in 2022. Specifically, our approach includes a new tree structure and a fast reconstruction method. The proposed approach reduces execution time and memory consumption by approximately 2 times compared to the old tree structure.

Keywords: *Association rules, rare association rules, rare patterns, common patterns, damped window model*

I. INTRODUCTION

Nowadays, data, most of which is non-static, is generated and collected daily. Among these, time-sensitive data is continuously generated and frequently changes over time, characterized by being associated with specific time intervals. Applying methods to optimize such time-sensitive data in terms of factors like memory, speed, etc., can bring many benefits.

Many association rule mining (ARM) methods [1] have been applied to discover relationships between itemsets in databases, and rare association rule mining (RARM) [2], has been proposed. However, when published throughout this process, each method tends to address only part of the problem or still has existing limitations. In 2022, a group of researchers in China proposed a tree structure called LC-tree to mine rare and common patterns in the association rule mining problem [3]. The LC-tree structure combines features resembling the lifecycle of a real tree in data management and has addressed most of the limitations listed. However, due to some inappropriate data structures used in LC-tree, the tree-building time and memory consumption have not been optimized. This research will introduce new approaches, namely a new tree structure improved from the old LC-tree structure, called ILC-tree, and a new restructuring strategy. Overall, with this approach, we set three main goals:

- The proposed tree structure must ensure the advantages of the old LC-tree structure, solving the limitations of previous methods that the LC-tree has already addressed.

- The tree-building time must be faster than LC-tree and previous methods.

- Memory consumption must be less than LC-tree and previous methods.

To achieve these goals, we proposed a new approach to replace the hash table with a single linked list in the child node list, where the parent node will only point to the beginning of the linked list instead of having to point to the entire child node as in the old structure. Next, we propose a method to adjust new nodes in tree restructuring, allowing the permutation of child nodes with a group of nodes from its parent to its predecessors. Our approach helps save memory and reduce the complexity of the node adjustment process from $O\left(n^2\right)$ to $O(n)$.

II. RELATED ISSUES

1. Basic concepts

This section will introduce critical definitions of RARM and the data structure employed in RARM methods.

Let $I = \{x_1, x_2 \ldots, x_n\}$ be a set containing n items, and each subset of I is called an itemset. Let D be a database containing a list of m transactions $T = \{t_1, t_2, \ldots, t_m\}$, where each transaction $t_i \in T$ is an itemset $\{i_1, i_2, \ldots, i_k\}$. A combination rule between two sets X and Y is denoted as $X \rightarrow Y$, where $X \subset I, Y \subset I$ and $X \cap Y \neq \varnothing$.

Definition 1: Support of X, denoted as $Supp(X)$ is the fraction of transactions containing itemsets X in the database D.

$$Supp(X) = \frac{|X|}{|D|} \tag{1}$$

(where $|D|$ is the total number of transactions and $|X|$ is the total number of transactions containing itemsets X)

Definition 2: Support of the rule, denoted as $Supp(X \rightarrow Y)$ is the fraction of transactions containing both itemsets X and Y in the database D.

$$Supp(X \rightarrow Y) = \frac{|X \cup Y|}{|D|} \tag{2}$$

(where $|X \cup Y|$ is the total number of transactions containing both itemsets X and Y)

Definition 3: Confidence of the rule, denoted as $Conf(X \rightarrow Y)$ is the fraction of transactions that contain both itemsets X and Y in transactions that contain itemsets X.

$$Conf(X \rightarrow Y) = \frac{|X \cup Y|}{|X|} \tag{3}$$

Definition 4: (Rare Pattern) A rare pattern is a pattern with a count greater than or equal to the minimum rare threshold *minRare* and less than the minimum frequent threshold *minFre*, expressed as:

$$minFre > Count(X) \geq minRare \tag{4}$$

Definition 5: (Frequent Pattern) A frequent pattern is a pattern with a count greater than or equal to the minimum frequent threshold *minFre*, expressed as:

$$Count(X) \geq minFre \tag{5}$$

Current RARM methods often apply two data structures: 1 tree structure and 1 header table. For example, in the case of SSP-Tree, its tree structure stores item names and frequencies of occurrence. They include tree nodes and links, with tree nodes divided into root nodes, leaf nodes, and branch nodes, defined as follows:

Definition 6: (Root Node) A tree structure has only 1 unique root node, from which the tree path to other nodes begins. The tree node does not contain item information.

Definition 7: (Leaf Node) A leaf node is a node following the tree path, storing the name and frequency of the item.

Definition 8: (Branch Node) A branch node is a node within the tree path connecting the root node and the leaf node. The branch node also stores the name and frequency of the item.

2. Related works

Srikant and Agrawal proposed the first RARM method in 1997, using the Apriori algorithm to generate rare association rules with a minimum support level (min-support) [4]. Based on this foundation, Haglin and Manning applied min-support and max-support to generate rare association rules [5]. Subsequently, improved algorithms with similar mechanisms were developed, such as RSAA [6], Apriori-Inverse [7], and Apriori-Rare [8]. However, Apriori-based RARM algorithms face two significant issues: they must traverse the database multiple times and generate many candidate patterns. Therefore, FP-growth-based RARM algorithms were proposed based on the original FP-growth algorithm [9], applying a tree structure that only requires two passes through the database to extract patterns without generating candidates. However, these methods are designed for static databases, and in practice, continuous data is generated and changes over time. Therefore, these methods may not be well-suited as they must start from scratch for newly updated databases.

To improve the performance of RARM on dynamic databases, researchers have made enhancements to FP-growth-based RARM algorithms. AFPIM [10], FUFP [11], and IFP-tree [12] are algorithms designed for dynamic databases. They all use the FP-tree structure to maintain the compactness of information and avoid scanning as much data as possible to enhance efficiency. Additionally, some methods can update the tree structure without rescanning the initial database. For example, in 2019, Borah and Nath proposed the SSP-Tree algorithm [13] to mine rare association rules in medical diagnosis. The ISSP-tree algorithm [14] is an improved version of the SSP-tree algorithm. Mahdi and colleagues proposed the FR-tree algorithm [15], which can create, filter, and classify association rules without needing an additional threshold. Furthermore, some rare association rule mining methods have been further improved by considering the characteristics of items and databases. However, due to limitations in inefficient data structures, the current rare association rule mining process still consumes considerable time and memory for storing and searching data.

In 2022, a group of authors in China proposed a new tree structure based on the idea of the life cycle of a real tree in nature called the LC-tree. The tree structure is illustrated in Figure 1. Adding the ”null” item to the header table to store a list of pure leaf nodes helps optimize the tree to become faster and easier. This tree structure addresses the shortcomings of previous methods, such as its applicability

to dynamic databases, not requiring multiple scans of the database, and making changes to optimize pattern mining time and consume less resources than existing methods.

However, let's consider the aspects of execution time and memory consumption when building the tree. This structure is still not fully optimized because some applied data structures are unreasonable. Specifically, LC-tree uses a hash table structure, increasing memory costs due to storing crucial information in the table. Additionally, this tree structure can only perform a 1-level permutation between 2 adjacent nodes and continue recursive swapping until adjustment is complete, increasing the cost of both memory and time.

III. PROPOSED METHOD

To improve memory consumption and execution time, we propose a new tree data structure called ILC-tree, illustrated in Figure 2. We change the link between parent nodes and the list of child nodes from a hash table structure to a single linked list. By adopting this new tree structure, the child nodes are linked together into a single linked list, and the parent node points to the first child node in the list. This change helps reduce memory costs during the tree construction process. The following enhancement involves proposing a method to adjust new nodes by permuting the node that needs adjustment with its group of predecessor nodes. In the worst case, when adjusting a branch with a depth of n itemsets, it only requires $n - 1$ comparisons and a single permutation, compared to the previous method that required $n - 1$ permutations. This proposal helps reduce complexity from $O\left(n^2\right)$ to $O(n)$. In addition to the transition to a linked list structure, our approach incorporates the utilization of the damped window model [16]. This model is instrumental in optimizing the tree structure by gradually removing invalid transactions over time during pattern mining tasks.The changes significantly enhance ILC-Tree efficiency compared to the old LC tree, and will be elaborated on in detail in section 3.1 of this paper.

1. Construction of ILC-Tree

To better understand visually, an example database containing 7 transactions is introduced, as shown in Table I. The TID stores the ID of each transaction, Itemset contains the items in the corresponding transaction, Sum_vitality is the total vitality value of the transaction, Vitality contains information about the vitality value of each item, and Timestamp is the time when the transaction occurs. The larger the timestamp value of a transaction, the newer the transaction is. The larger the vitality value of each item in the tree branch, the higher the vitality of that branch. Regarding the process of building the tree, we divide it into two subtasks: tree restructuring and structure updating. These two tasks will be described respectively in the fast restructuring strategy and the tree structure updating.

TABLE I
EXAMPLE DATASET

TID	Itemset	Sum_vitality	Vitality	Timestamp
1	A, B, C, E, F, G	6	1, 1, 1, 1, 1, 1	50
2	A, C, D, G	4	1, 1, 1, 1	53
3	A, D, H	3	1, 1, 1	55
4	E, F, H, I	4	1, 1, 1, 1	60
5	A, C, E, G	4	1, 1, 1, 1	70
6	I, C, E, F	4	1, 1, 1, 1	75
7	D, A, H, B	4	1, 1, 1, 1	80

a) Fast restructuring strategy

Before each structure update operation, the header table is updated to obtain the order of the most recent item frequencies. The ILC-tree will automatically restructure when the items on the path are not in the order of the current item frequency. Instead of creating a change list of limited items, we will directly compare the items in the current header table with the previous header table to determine the items that need adjustment. Our method will swap the relevant nodes through parent and child node links for each item until all nodes are linked to the traversed list. In summary, the fast ILC-tree restructuring steps are automatically performed in the alternating sequence as follows:

Step 1: Compare the current header table with the previous header table. Starting from the high-frequency item in the list, the Q tree nodes related to the item are accessed through node links.

Step 2: If a parent node R exists with an order after the Q node, move the Q node above the parent node R.

Step 3: Use nodeLink to move to the node P, which has the same name as Q.

Step 4: If P exists, perform step 2. If not, proceed to the next step.

Step 5: Perform steps 2 to 4 until the header table is fully traversed.

During the exchange process, the following properties must be observed:

Property 1: Temporary transaction information is stored in the new leaf node of the transaction. [3]

Property 2: The number of child nodes must not exceed the number of its parent node.[3]

Property 3: A node cannot have a sibling node with the same name.[3]

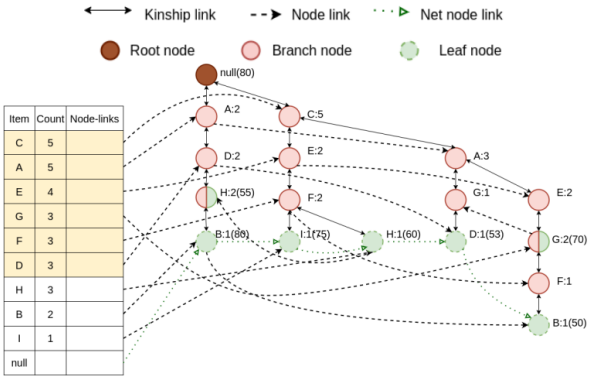


Figure 1. ILC-Tree Structure.

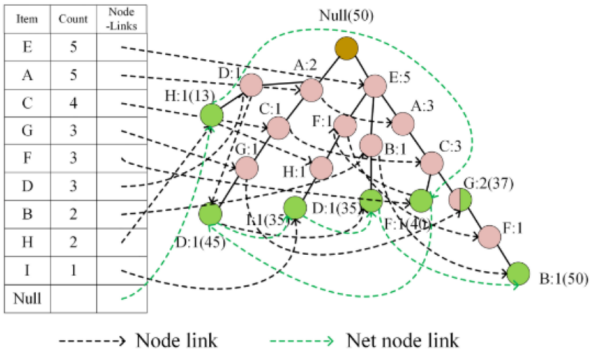


Figure 2. LC-Tree Structure[3]

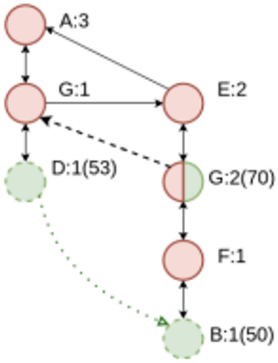


Figure 3. Branch EGFB before splitting

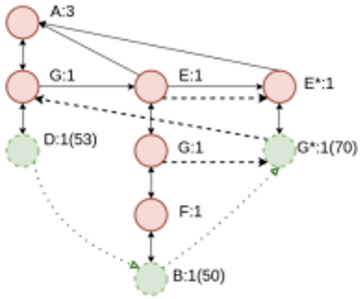


Figure 4. Branch EGFB after splitting

To understand the node permutation process, we divide it into 3 parts: Separation, permutation, and merging of duplicate nodes. The permutation operation between nodes is only performed when the number of Q nodes and the parent R nodes of Q are equal, meaning $R.count = Q.count$. Therefore, if the number of Q nodes is less than the parent R node, it is necessary to separate from the parent node of Q to the node R. Suppose there is a branch consisting of a group of nodes $P = \{P_0, P_1, P_2, \ldots, P_n\}$, with $n > 0$, where P_0 is the node that needs adjustment, in this case, P_0 is the Q node. P_1 is the parent node of P_0, P_2 is the parent node of P_1, ..., P_n is the parent node of P_{n-1}, and R is P_n. In that case, it is necessary to split the group of nodes P into two branches $P = \{P_0, P_1, P_2, \ldots, P_n\}$ and $P^* = \{P_0^*, P_1^*, P_2^*, \ldots, P_n^*\}$. The quantity of the P^* node group will be updated as follows:

$$P_1^* = P_i.count - P_0.count, \ for\ i : 1 \rightarrow n$$

The child nodes of the nodes in $P \setminus \{P_0\}$ that do not belong to this node branch will be sequentially assigned as child nodes of the nodes in P^*. For example, with the tree structure in Figure 2, assuming the node that needs adjustment is node B:1(50), its correct position is above the predecessor node E:2, as illustrated in Figure 3.

Before splitting, we have the set $P = \{E, G, F, B\}$. After separating, we have the set $P = \{E, G, F, B\}$ and the set $P^* = \{E^*, G^*\}$. The number of nodes in the P node group is equal, so the conditions for permutation are met. It is important to note that if any node P_i is a leaf node of another path, its temporary information will be transferred and stored in the node P_i^* when splitting. In this case, the node G^* is a leaf node, so its temporary information will be transferred from node G to node G^*, and since this leaf node has no child nodes, it will become a pure leaf node and will be linked to other pure leaf nodes.

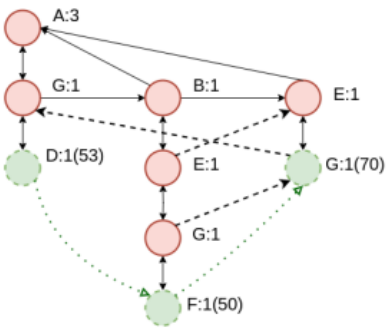


Figure 5. Tree after restructuring

Permutation: Based on the example in Figure 3(b), we will perform the permutation of node B:1(50) with the node group E:1, G:1, F:1. In this case, the order of the branch will change from $A \rightarrow E \rightarrow G \rightarrow F \rightarrow B$ *to* $A \rightarrow B \rightarrow E \rightarrow G \rightarrow F$. After the exchange, the result will be as shown in Figure 4. Since B is a pure leaf node, its old parent node F will become a new pure leaf node, and the temporary information at node B will be transferred to node F.

Merge Duplicate Nodes: If there exists a sibling node Q^* representing the same item as node Q and having the same parent node during the node exchange process, then the quantity and child links of node Q^* are merged into node Q. If Q^* is a leaf node, its temporary information will be updated to Q. If Q^* is a pure leaf node and Q is not, then Q^* will be removed from the pure leaf node linked list. Similarly, if Q is a pure leaf node and Q^* is not, then Q will be removed from the pure leaf node linked list. Afterward, Q^* is removed from the tree. This process is recursively applied to the children of Q until there are no more duplicate nodes.

b) Update Tree Structure

Updating the structure involves adding, deleting, and modifying the existing tree structure. Except for the first transaction data, adding the remaining data can be seen as an update to the existing structure. Modification operations can be considered a combination of addition and removal. This section details the process of adding transactions and deleting transactions.

- Adding Transaction: The process of adding a transaction is illustrated in Algorithm 1. The steps of adding a transaction are automatically performed in an interleaved manner as follows:

- Reordering the header table and transaction: Information about the frequency of items in the scanned transactions is store in a header table. When a new transaction is examined, each item is checked to see if it is already in the header table. If it is, its quantity in the header table is increased by 1. Otherwise, the item is inserted into the header table. After scanning, the header table and transactions are reordered based on the decreasing order of item frequencies.

- Restructuring the ILC-tree: Based on the order of the newly sorted header table, the ILC-tree is updated using the fast-restructuring strategy. It allows traversing through as few nodes as possible during the restructuring process.

- Adding New Transaction to ILC-tree: After restructuring the ILC-tree, the transactions items are rearranged and inserted into the ILC-tree in order. If a node containing an existing item is encountered, the quantity of that node is incremented by 1. Otherwise, a new node containing the new item is created in the ILC-tree and linked to the item with the same name through a nodeLink. The transaction time is stored in the leaf node of the path. If the timestamp value of the root node is less than the timestamp value of the new transaction, then the timestamp value of the root node is updated to the timestamp value of the new transaction.

- Linking Pure Leaf Nodes: When a data operation involves a leaf node, that node is identified as a pure leaf node based on its number of children. If a leaf node has no children, it is identified as a pure leaf node, and the new pure leaf node is linked to the ”null”

Entry in the header table and other pure leaf nodes. If a pure leaf node gains additional children during restructuring or adding a new transaction, its link is removed. Finally, the header table points to the most recently inserted pure leaf node.

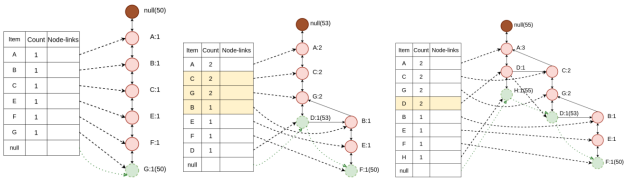


Figure 6. Process of inserting transactions 1 to 3

For example, based on the dataset in Table I, the process of adding a new transaction is illustrated in Figure 6. The set of items covered by the yellow-shaded area in the header table represents the differences between the old and new header tables. These nodes need to be adjusted before inserting new transactions. After adjustment, the last node in the path becomes the new leaf node, and the temporary information of the transaction is stored in the leaf node. The header table points to the newly inserted nodes and the pure leaf node when transactions are inserted. For example, when inserting the second transaction <A, C, D, G>, the header table updates the count of items 'A', 'C', 'G', and creates a new entry for item 'D', as shown in Figure 6. By comparing the updated and original frequencies in the header table, we identify the nodes that need adjustment, which are the nodes containing items 'C', 'G', and 'B'. After adjusting the path, node 'F' becomes the new leaf node containing the timestamp of the first transaction, and the items of the second transaction are inserted into the current ILC-tree.

The 'D' and 'Null' entries in the header table point to the new pure leaf node D and other nodes with the same item name in the tree structure are linked together. Similar processes for inserting transactions follow this example.

Algorithm 1: Add Transaction to ILC-Tree

Input: Transaction dataset D

Output: ILC-tree

Step 1: Create a header table with a key *null* and a root node labeled *null*;

Step 2: Procedure *SortHeaderTableAndTransactions*;

begin

  Read a transaction T from D;

  for *each item I in T* **do**

    if *I exists in the header table* **then**

      Increase the count of I by 1;

    end

    else

      Add I to the header table;

    end

  end

  Sort items in the header table in descending order of frequency;

  Sort each transaction T based on the order of the header table;

end

Step 3: Procedure *Restructure ILC-Tree*;

begin

  Compare the order of items between the old and new header tables;

  Rearrange the paths of the tree based on the links between different items in the two header tables;

  for *each path $P = \{P_0, P_1, \ldots, P_n\}$ belonging to a node branch in the ILC-tree* **do**

    if *the quantity of node P_i is greater than node P_0* **then**

      Split the P node group into two branches $P = \{P_0, P_1, \ldots, P_n\}$ and $P^* = \{P_0^*, P_1^*, \ldots, P_n^*\}$;

      Update the quantity of the P^* node group as $P_j^*.count = P_j.count - P_0.count$ for $j = 1 \rightarrow n$;

    end

  end

  Permute node P_0 and the node group $\{P_1, \ldots, P_i\}$;

  if *the child node W of P_{i+1} has the same name as P_0* **then**

    Merge node W with node P_0, including quantity and temporary information;

  end

end

Step 4: Procedure *Add New Transaction to ILC-Tree*;

begin

  After restructuring, insert the items of T into the ILC-tree;

  if *a node Y containing an item in T already exists* **then**

    Increase the quantity of Y by 1;

  end

  else

    Initialize node Y with a quantity of 1;

    Link it to the header table and linked nodes with the same item;

  end

  Add netNodeLink if it is a pure leaf node;

  Continue until all transactions are inserted into the tree;

end

Deletion of transaction: The ILC-tree algorithm performs the operation of deleting a transaction at a specific time and applies a top-down traversal mechanism to access nodes. The process of deleting a transaction is illustrated in Algorithm 2. For each deleted transaction, the first step is to sort it in descending order of item frequency in the current header table. Then, starting from the first item of the transaction, the count of each item in the header table is reduced by m, where m is the quantity to be deleted, as well as the count of the corresponding node. Nodes with a count of 0 after reduction will be deleted. After all transactions are deleted, the items in the header table are sorted in descending order of frequency, and the paths of the ILC-tree are rearranged based on the updated frequency decreasing order of items.

An example is provided to illustrate the transaction deletion process for the initial tree structure of Table I. Table II is the set of transactions to be deleted, and the process is illustrated in Figure 7. Nodes with a quantity reduced to 0 after reduction are marked with gray dots. After processing all nodes of transactions, the algorithm performs a fast restructuring to maintain the properties of the ILC-tree and improve the compactness of the tree structure.

Algorithm 2: Delete transaction and update the tree

Require: Transaction dataset D, ILC-tree root node, deleted transaction dataset db^- from D

Ensure : Updated ILC-tree $(D - db^-)$

From db^-, read a transaction T at a certain moment, decrease the quantity by 1 for each item i of T;
Sort the items in the header table in descending order, sort each transaction t in db^- based on the order of the updated header table. Set the ordered transaction list as $[m|M]$, where m is the first element of the ordered list, and M is the rest of the list;
Set X as the root node;
for *each child node* Y *of node* X *such that* $Y.ItemID = m.ItemID$ **do**
    Decrease the quantity of Y by 1;
    Update X to be Y, the first item of M will be m, and the rest of the items will be M;
    if Y *is a leaf node* **then**
        Remove the temporary information of transaction T;
    end
end
Continue until all transactions in db^- are deleted;
Execute steps 11-20 of Algorithm 1;

TABLE II
THE TRANSACTION SET HAS BEEN DELETED

TID	Itemset	Sum_vitality	Vitality	Timestamp
3	A, D, H	3	1, 1, 1	55
7	D, A, H, B	4	1, 1, 1, 1	80

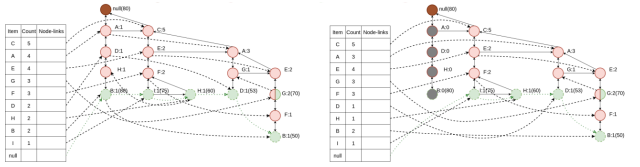


Figure 7. Process of deleting transactions 3 and 7

c) Damped window model for tree optimization

The damped window [16] model is used to optimize the tree structure by removing invalid transactions over time in pattern mining. The model is combined with our ILC-tree structure. First, the model accesses each pure leaf node through the node links linked to the ”null” entry in the header table. Each pure leaf node is analyzed and evaluated based on its time and frequency. When a pure leaf node is identified for deletion, its corresponding tree path will also be deleted. If a new pure leaf node appears during the deletion process, its decayed vitality value needs to be calculated and evaluated. If the decayed vitality value is greater than the $vit_{\min}$ value, then proceed to the next pure leaf node; otherwise, continue deleting the branch with this leaf node. After evaluating all pure leaf nodes in the tree and adjusting the tree structure, the tree structure is optimized. In this process, the model also addresses two main issues:

Determining when a leaf node needs to be deleted: A leaf node may be deleted if it infrequently occurs, and its last occurrence was a long time ago. To quantify this predictive process, the following definitions of node vitality are introduced. Firstly, we determine a value to quantify the timeliness of transactions. The lifespan of an item in a transaction $v(i, t_d)$ is calculated as: $v\,(i, t_d) = u_i$ (6)[16] In which i is an item of the transaction t_d, $i \in t_d$, and u_i is the relative importance, utility, of i. Based on this, the vitality value of the transaction (the vitality of the tree branch) $v(t_d)$ is calculated as: $v\,(t_d) = \sum\ v(i, t_d)/|t_d|$ (7)[16]

Where $|t_d|$ is the number of items in t_d. For example, for the transaction <A, C, D, G> in Table I, where all items have an importance of 1, the vitality value of the transaction is 1.

For each transaction, the value of decayed vitality $v\,(l_d)$ is calculated as follows:

$$v\,(l_d) = \sum\ v(t_d) * f^{T-d} \quad (8)[16]$$

where d is the time of occurrence of each transaction t_d, T is the current time contained in the root node, and f is the decay factor. For example, for the leaf node at transaction 2, the transaction occurrence time is $T = 80$, decay factor $f = 0.99$, then the decayed vitality value of this leaf node is calculated as: $v\left(l_{d(2)}\right) = 1 * 0.99^{80-53} \approx 0.76$. If the decayed vitality value at the leaf node falls below the $vit_{\min}$ threshold, the leaf node is identified as weak, and the related paths and nodes in the tree will be deleted.

Performing the deletion operation and adjusting the ILC-tree: The deletion operation starts from the pure leaf node to be deleted and follows the transmission mechanism from bottom to top. The quantity of the leaf node is marked as N_i. When the pure leaf node is deleted, the corresponding item quantity in the header table will decrease by N_i.

For the remaining nodes of the transaction connected to the deleted node, their quantity will reduce similarly by N_i. The amount of the corresponding item in the header table will decrease by N_i. Nodes with a quantity of 0 after the reduction will be deleted. After the deletion operations are completed, the ILC-tree performs rapid restructuring to adjust the tree structure.

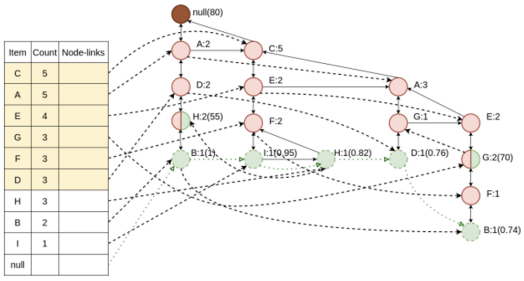
Algorithm 3: Delete transaction and update the tree

Require: Table of headers, root node
Ensure : Updated ILC-tree

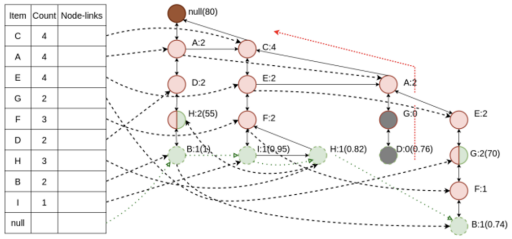
foreach *leaf node X of the ILC-tree* **do**
  Calculate $v\left(l_{d(x)}\right)$ for node X;
  if $v\left(l_{d(x)}\right) < vit_{\min}$ **then**
    Delete node X and set N_i to the number of nodes X;
    foreach *parent node Y of node X* **do**
      Decrease N_i by the number of nodes Y;
      if *the number of nodes* $Y = 0$ **then**
        Delete node Y;
      end
    end
    if *Y is a pure leaf node after deletion* **then**
      Repeat steps 2-14 to evaluate the new pure leaf node;
    end
  end
end

2. Pattern Mining from ILC-tree Structure

After the construction of the ILC, the process of creating subtrees is based on two support thresholds: $minFreq$ and $minRare$. In mining, the ILC-tree applies a backward propagation mechanism, starting from the lowest node



(a) Tree after calculating the decayed vitality value of each pure leaf node



(b) Branch t1 is deleted

Figure 8. Tree optimization process

and moving gradually up to the root node. Subsequently, frequent, and rare patterns for each item are collected based on the configured thresholds. Rare patterns need to satisfy condition 4 in Section 2, and frequent patterns need to satisfy condition 5 in Section 2. The entire pattern mining process is illustrated in Algorithm 4.

Algorithm 4: Pattern Mining from ILC-tree

Require: Header table H, minFreq, minRare
Ensure : Frequent patterns, rare patterns

$result \leftarrow \varnothing$;
for *each item I in the header table H with* $I.support > minRare$ **do**
  Create a set of prefix paths in the ILC-tree that occur with I;
  Build $Tree_{Pattern}$ containing frequent and rare items satisfying minFreq, minRare;
  Create FrequentPattern and RarePattern by appending I to frequent and rare items in $Tree_{Pattern}$;
  $result \leftarrow [FrequentPattern \cup RarePattern]$;
end
return $result$;

With the final tree structure obtained in Figure 7(f), the results for frequent and rare patterns are presented in Table

TABLE III
FREQUENT AND RARE PATTERNS EXTRACTED FROM THE ILC-TREE

Item	Base sample	Subtree	Common sample	Rare sample
A	{C:1}	{C:1}	-	-
E	{C:2}, {C, A:1}	{C:3}, {A:1}	{C, E:3}	-
H	{A:2}, {C, E:1}	{A:2}, {C, E:1}	-	{A, H:2}
F	{C, E:1}, {C, E, H:1}	{C, E:2}, {H:1}	-	{C, E, F:2}
D	{A, H:2}	{A, H:2}	-	{A, H, D:2}

III, where $minFreq = 3$ and $minRare = 2$. In this table, Item E has an empty base pattern, so it is impossible to build a subtree for E. Items G, B, and I do not satisfy the support thresholds, so no subtrees can be constructed for them, resulting in no patterns generated for these items. For Item E, the base patterns include {C: 2} and {C, A: 1}. According to the support thresholds, a frequent pattern is {C, E: 3}, and no rare patterns are generated for Item E. Frequent and rare patterns for the remaining items in the header table are generated similarly.

3. Time Complexity Analysis

The proposed ILC-tree method comprises two stages: updating and optimizing the tree structure. Therefore, the time complexity analysis is performed separately for each phase.

a) Tree Structure Update

Let D be the database with T transactions, I be the total number of items in the database, and L be the length of the most extended transaction. Updating the tree structure is divided into adding transactions and deleting transactions. Adding a transaction involves four steps: rearranging the header table and the transaction, fast restructuring, adding the new transaction to the ILC-tree, and linking the pure leaf nodes.

Adding transactions: The worst-case scenario for the first step is when the length of each transaction equals the number of items I, meaning $L = I$. However, this is the best case for rearranging the header table because there will be no changes in the order. Therefore, the time complexity of the first step is $O(TI^2)$. In the second step, the worst case is when all tree paths need adjustment when the data is inserted. In other words, the total number of swaps equals the total number of items I. Hence, the overall time complexity of the second step is $O(TI^2)$. In the third step, the worst case is when the tree has a maximum depth of $L = I$, and the ILC-tree requires traversing a maximum of I nodes to insert a node. Therefore, the total time complexity of inserting transactions is $O(TI^2)$. In the final step, the worst case is when all T transactions require linking node information, and the total time complexity is $O(T)$.

Deleting transactions: The worst-case scenario for deleting transactions is when all transactions are deleted, and each deletion requires traversing through nodes, resulting in a total time complexity of $O(TI^2)$. Hence, the overall time complexity of updating the tree structure can be asymptotically expressed as $O(TI^2)$.

b) Tree Optimization

Tree optimization includes calculating the decayed vitality value of each pure leaf node, deleting paths, and quick restructuring. The worst case for calculating the degraded vitality value is when the number of paths is T, and the time complexity is $O(T)$. For deleting paths, the time complexity depends on the number of paths in the tree and the depth of the tree. The maximum depth of the tree is $L = I$, and the maximum number of tree paths is T. Therefore, the time complexity is $O(TI^2)$. The total time complexity of quick restructuring is $O(TI^2)$, as the maximum number of swaps required is I. Thus, the overall time complexity of tree optimization can be asymptotically expressed as $O(TI^2)$. In summary, ILC-tree requires $O(TI^2)$ for updating the tree structure and $O(TI^2)$ for optimizing the tree. The total time complexity of building the ILC-tree can be asymptotically expressed as $O(TI^2)$.

IV. EXPERIMENTAL

1. Experimental Environment and Dataset

TABLE IV
CHARACTERISTICS OF THE DATASETS

Dataset	Category	Transaction count	Item count (I)	Average Length (A)
Retail utility timestamp	Synthetic	88,162	16,470	10.30
Ecommerce utility timestamp	Real	14,975	3,468	11.71
Mushroom utility timestamp	Synthetic	8,416	119	23

In this section, we will conduct experiments to verify the effectiveness of the proposed algorithm and compare it with SSP-tree, ISSP-tree, and LC-tree. SSP-tree and ISSP-tree are two methods for mining rare itemset patterns using their proposed structures to store data, while LC-tree utilizes a life-cycle tree structure to optimize the tree-building process compared to the two aforementioned structures. We will perform a comparison evaluating the real-time execution and memory consumption of these

methods on various datasets. We use authentic and synthetic datasets, including Retail utility timestamps, ECommerce utility timestamps, and Mushroom utility timestamps. These datasets are widely used to evaluate performance in pattern mining and can be downloaded from the SPMF - Open-Source Data Mining Library. Table IV shows the characteristics of the datasets, such as the time span, the number of transactions, the number of distinct items, the average length, and the density of an item in a transaction. Datasets with more distinct items incur higher memory and execution time costs. Additionally, datasets with lower item density result in lower chances of item repetition in different transactions, leading to longer structure update times. Therefore, we selected three datasets with high, medium, and low density (19.33%, 0.34%, 0.06%) to evaluate our proposed tree structure objectively compared to other tree structures.

To run the experiments, we divided each dataset into five smaller or equal-sized databases, with each small dataset accounting for approximately 20% of the original dataset. After each transaction is inserted into the tree structure, the algorithm will run the reconstruction function promptly. After inserting each dataset, the algorithm will use the damped window model to optimize the tree once.

TABLE V
BRANCHING DENSITY IN THE DATASET

Dataset	f	$[vit_{\min}]$	$minRare$	$[minFreq]$
Retail utility timestamp	0.9	0.28	800	1000
ECommerce utility timestamp	0.9	27.3	75	110
Mushroom utility timestamp	0.9	2.35	800	1000

All methods build an initial tree structure with temporary information and perform pattern mining. Then, the data structure is updated and optimized every time a supplementary database is inserted, and the pattern mining task is re-executed. Decay factor f, minimum $vit_{\min}$ threshold, and support thresholds are set as in Table V. All methods generate similar rare and common patterns and spend equal amounts of time on the pattern mining process due to the same input data and parameter settings. We evaluate the performance of the methods and the obtained results by taking the average values from multiple runs. All algorithms are implemented in Python and executed on a computer with the following configuration: Ubuntu 18.04.6 LTS, Intel 2.2 GHz CPU, 256 GB RAM.

2. Results

a) Execution Time

From Figure 9 (a), it is evident that our tree structure has a faster execution time compared to the old version of LC-tree and SSP-tree, ISSP-tree, with times of 72.9 hours, 81.4 hours, 87.9 hours, and 83.6 hours, respectively, in the Retail utility timestamp dataset. Figure 9 (b) shows 48.6 minutes, 59.4 minutes, 56.7 minutes, and 56.8 minutes, respectively, in the Ecommerce utility timestamps dataset. The relatively long execution time in the Retail utility timestamp dataset is due to its large and sparse nature, leading to the need to build more nodes. Our ILC-tree has a mechanism for quickly adjusting nodes, thus helping to reduce the reconstruction time.

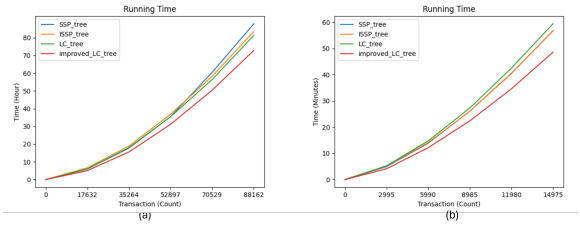


Figure 9. Real-time execution time chart of the ILC-tree structure compared with 3 structures: LC-tree, SSP-tree, and ISSP-tree, on 2 datasets

b) Memory Consumption

In terms of memory consumption, our algorithm stands out in comparison to LC-tree, SSP-tree, and ISSP-tree. In the Retail utility timestamp dataset, depicted in Figure 10 (a), our algorithm exhibits memory footprints of 158.9 MB, 321.7 MB, 540.2 MB, and 402.8 MB. Meanwhile, in the Ecommerce utility timestamps dataset illustrated in Figure 10 (b), the memory footprints are observed to be 30.0 MB, 61.9 MB, 87.3 MB, and 60.8 MB respectively. These figures underscore the superior efficiency of our approach.

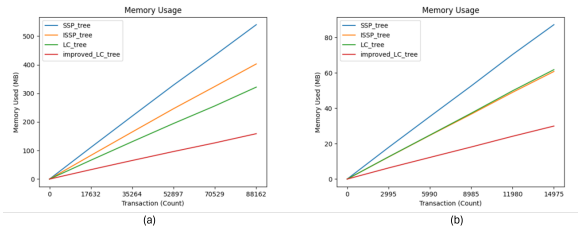


Figure 10. Memory consumption of the ILC-tree structure compared to 3 other structures: LC-tree, SSP-tree, and ISSP-tree, on 2 datasets

Moreover, when directly compared to the legacy LC-tree structure, our ILC-tree demonstrates a remarkable performance, executing approximately twice as fast while conserving more memory than its predecessors SSP-tree and ISSP-tree. This significant enhancement is primarily attributed to the implementation of our novel tree structure, which effectively reduces the memory overhead associated with the list of child nodes at each parent node.

V. CONCLUSION

In this study, we have proposed improvements to optimize execution time and memory consumption compared to the old tree structure. This includes changing the tree structure and using a method to adjust new nodes. For the tree structure, we use a singly linked list to store child nodes instead of a hash table, reducing the storage cost of keys in the hash table. For the tree reconstruction task, our method allows the node that needs adjustment to swap positions with a group of nodes rather than just the parent node, improving execution time, especially for large and densely branched datasets.

In the future, we will explore new algorithms and strategies to optimize the pattern mining process. We will consider using more efficient libraries like Dask or DataTable to handle large-sized files, improving execution time. We will also investigate methods for extracting rules from patterns and continue researching and building new tree structure to find more optimal solutions. Despite the remaining challenges, our proposed enhancements have significant potential for the future, thanks to their processing speed and memory optimization.

REFERENCES

[1] Hipp, Jochen, Ulrich Güntzer, and Gholamreza Nakhaeizadeh, "Algorithms for association rule mining—a general survey and comparison," *ACM SIGKDD Explorations Newsletter*, vol. 2, no. 1, 2000, pp. 58–64.

[2] Abbasi, Ameer Ahmed, and Mohamed Younis. "A survey on clustering algorithms for wireless sensor networks," *Computer Communications*, vol. 30, no. 14-15, 2007, pp. 2826–2841.

[3] Hu, Kerui, Lemiao Qiu, Shuyou Zhang, Zili Wang, and Naiyu Fang, "An incremental rare association rule mining approach with a life cycle tree structure considering time-sensitive data," *Applied Intelligence*, vol. 53, no. 9, 2023, pp. 10800–10824.

[4] Srikant, Ramakrishnan, and Rakesh Agrawal, "Mining generalized association rules," *Future Generation Computer Systems*, vol. 13, no. 2-3, 1997, pp. 161–180.

[5] Haglin, David J., and Anna M. Manning, "On minimal infrequent itemset mining," In *Proceedings of DMIN*, 2007, pp. 141–147.

[6] Yun, Hyunyoon, Danshim Ha, Buhyun Hwang, and Keun Ho Ryu, "Mining association rules on significant rare data using relative support," *Journal of Systems and Software*, vol. 67, no. 3, 2003, pp. 181–191.

[7] Koh, Yun Sing, and Nathan Rountree, "Finding sporadic rules using apriori-inverse," In *Proceedings of the 9th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)*, Springer, 2005, pp. 97–106.

[8] Szathmary, Laszlo, Amedeo Napoli, and Petko Valtchev, "Towards rare itemset mining," In *Proceedings of the 19th IEEE International Conference on Tools with Artificial Intelligence (ICTAI)*, vol. 1, 2007, pp. 305–312.

[9] Han, Jiawei, Jian Pei, and Yiwen Yin, "Mining frequent patterns without candidate generation," *ACM SIGMOD Record*, vol. 29, no. 2, 2000, pp. 1–12.

[10] Koh, Jia-Ling, and Shui-Feng Shieh, "An efficient approach for maintaining association rules based on adjusting FP-tree structures," In *Proceedings of the International Conference on Database Systems for Advanced Applications*, Springer, 2004, pp. 417–424.

[11] Hong, Tzung-Pei, Chun-Wei Lin, and Yu-Lung Wu, "Incrementally fast updated frequent pattern trees," *Expert Systems with Applications*, vol. 34, no. 4, 2008, pp. 2424–2435.

[12] Adnan, Muhaimenul, Reda Alhajj, and Ken Barker, "Alternative method for incrementally constructing the FP-tree," In *Proceedings of the 3rd International IEEE Conference on Intelligent Systems*, 2006, pp. 494–499.

[13] Borah, Anindita, and Bhabesh Nath, "Incremental rare pattern based approach for identifying outliers in medical data," *Applied Soft Computing*, vol. 85, 2019, pp. 105824.

[14] Ahmed, Shafiul Alom, and Bhabesh Nath, "ISSP-tree: An improved fast algorithm for constructing a complete prefix tree using single database scan," *Expert Systems with Applications*, vol. 185, 2021, pp. 115603.

[15] Mahdi, Mahmoud A., Khalid M. Hosny, and Ibrahim Elhenawy, "FR-Tree: A novel rare association rule for big data problem," *Expert Systems with Applications*, vol. 187, 2022, pp. 115898.

[16] Yun, Unil, Donggyu Kim, Eunchul Yoon, and Hamido Fujita, "Damped window based high average utility pattern mining over data streams," *Knowledge-Based Systems*, vol. 144, 2018, pp. 188–205.



Dang-Ngo Huu received the B.E. degree in Data Science from Faculty of Information Technology, University of Science in 2023. Research direction: Data Science, Privacy-preserving data mining.
Email: hudn@gmail.com

Khac-Chien Nguyen received his Bachelor's degree in Information Technology from the People's Security Academy in 2003 and received the Master degree in Computer Science from the University of Natural Sciences - Vietnam National University, Ho Chi Minh City in 2009, and his PhD in Communication Engineering from the Post and Telecommunication Institute of Technology Hanoi in 2019. His research interests include: Cloud computing and Data mining.
Email: nkchienster@gmail.com

Nghia Le received the B.E. degree from Faculty of Electronics and Telecommunications, Ho Chi Minh City University of Technology in 1995. In 1998, he received Master degree in Computer Science, Ho Chi Minh City University of Science. His research interests include: Artificial Intelligence in Embedded Systems and Data Security.
Email: nghiale@gmail.com