Journal on
Information Technologies & Communications
https://www.ictmag.ictvietnam.vn/ict
Volume 2025, No.1
March

TABLE OF CONTENTS

Low-Barrier Object Detection for Mobile Applications

Khoi Nguyen The[1], Tuong Ho Vinh[2], Anh Hoang[3]

[1] Department of Information Technology, FPT University HCMC, Vietnam
[2] Department of Information Technology, TDT University, Vietnam
[3] School of Business Information Technology, College of Technology and Design, University of Economics Ho Chi Minh City, Vietnam

Correspondence: Anh Hoang. Email: anhh@ueh.edu.vn
Communication: received 12 Aug 2024, revised 1 Oct 2024, accepted 25 Oct 2024
DOI: 10.32913/mic-ict-research.v2025.n1.1343

***Abstract*: This paper investigates innovative applications of image processing and object recognition methods aimed at simplifying the creation and deployment of object detection models. By doing so, we seek to expand access to advanced computer vision technologies for small and medium-sized businesses. Our research leverages a combination of modern technologies including Flask for web development, Firebase for database management, and Kotlin and Jetpack Compose for mobile application development. We integrate these with automatic training methods provided by the Autodistill library, utilizing models such as Detic and YOLOv8.**

The results demonstrate that this technological combination significantly enhances the performance of our object detection models, contributing to AI solutions in the digital intelligence era. A notable advancement is Autodistill's capability to bypass the traditional dataset creation step by automatically generating labeled datasets from unlabeled input data. This feature markedly improves the efficiency and effectiveness of both data preparation and model training processes. Overall, our findings underscore the potential of these integrated technologies to democratize access to sophisticated computer vision capabilities for smaller enterprises, fostering greater innovation and competitiveness in the marketplace.

Keywords: *Distillation, object detection, android application*

I. Introduction

This paper presents a novel, user-friendly approach to mobile object detection, offering a low-barrier solution for identifying and locating objects in images and videos. Object detection is increasingly vital across various industries, playing a key role in tasks like autonomous vehicle navigation in complex environments and enhancing medical imaging systems for more accurate diagnoses. The proposed application aims to deliver efficient and precise object detection, meeting the growing demand for accessible solutions in both everyday and specialized use cases.

Traditional object detection methods often involve complex procedures that require specialized technical knowledge. Data preparation, feature engineering, and model training are time-consuming and can be challenging for non-experts. These hurdles have limited the accessibility of object detection to a select group of individuals with advanced technical skills.

In the realm of traffic safety, Computer Vision (CV) is crucial for the recognition of traffic signs and lane markings, significantly enhancing vehicular safety and enabling the development of Advanced Driver-Assistance Systems (ADAS) and autonomous vehicles [1, 2]. Additionally, CV technology is vital in defense and healthcare sectors. In defense, it facilitates the use of drones and robots to access and operate in hazardous areas that are unsafe or unreachable for humans [3]. In healthcare, CV applications include automated diagnosis, medical imaging analysis, and surgical assistance, thereby improving the accuracy and efficiency of medical procedures [4].

Despite the transforming potential of CV, several challenges impede its widespread adoption. Technical barriers, such as the complexity of developing and deploying CV models, remain significant hurdles. Additionally, the high costs associated with implementing and maintaining CV systems can be prohibitive for many businesses, particularly smaller ones. These factors contribute to the slow uptake of CV technology across different industries, despite its proven benefits.

To address these limitations, we propose a user-friendly approach to object detection that eliminates the complexities of data processing and training. Our app allows users to create custom object detection models with minimal input. Users simply need to provide the names of the objects they want to detect, along with a few sample images. The app then automatically handles the data preprocessing, model

training, and evaluation, providing users with a ready-to-use object detection model.

By democratizing object detection, our method enables a wider range of individuals to benefit from this powerful technology. This has the potential to drive innovation in various fields and address real-world challenges.

The following sections delve deeper into our research. Section 2 reviews existing contributions in the field, highlighting their strengths and weaknesses. Section 3 details our methodology, which is the core of our innovation. Section 4 presents the evaluation results, assessing our product on various aspects. If applicable, Section 5 explores experiments and ablation studies we conducted. Finally, Section 6 concludes the report by summarizing our findings.

II. Problem Statement and Related works

1. Problem Statement

Object detection, while a powerful tool, presents significant challenges for non-technical users. Traditional methods often involve complex procedures that require specialized knowledge and skills, limiting accessibility to a select group of individuals. Data preparation, including data collection, annotation, and preprocessing, is time-consuming and error-prone. Moreover, training deep learning models for object detection can be computationally intensive and requires expertise in machine learning.

Existing deep learning frameworks for object detection, while powerful, can be difficult to use for non-technical users. The steep learning curve, the requirement for extensive data preprocessing, and the complexity of model training can be daunting for individuals without specialized knowledge. Additionally, the performance of deep learning models heavily relies on the quality and quantity of training data, which can be a significant limitation for many applications.

To address these challenges, there is a clear need for a user-friendly approach to object detection that eliminates the complexities of data processing and training. Such a solution would enable a wider range of individuals to benefit from this powerful technology, democratizing AI and driving innovation in various fields.

2. Open vocabulary object detection

In the field of object detection, the shift towards Open Vocabulary object Detection (OVD) [5, 6] has presented a significant challenge and opportunity. Traditionally, object detection models have relied on predefined sets of objects, known as closed vocabulary systems, which limit the range of detectable objects to those included in the training data. These closed vocabulary models, while efficient, are constrained by their inability to recognize objects outside their predefined sets, which can be a significant limitation in real-world applications where the variety of objects can be vast and unpredictable.

In response to these limitations, the OVD models have emerged as a critical advancement. Among the pioneers in this field is Detic [7], which stands for Detecting Twenty-thousand Classes using Image-level Supervision. Detic is notable for its ability to identify a large number of objects, up to 21,000 different classes, far surpassing the capabilities of traditional models. This expansive ”vocabulary” allows Detic to recognize a much broader range of objects, including many that were previously difficult or impossible to detect with closed vocabulary systems.

The key innovation of Detic lies in its use of image-level supervision to expand its detection capabilities. By leveraging vast amounts of image data, Detic can learn to identify a wide array of objects without being limited to a predefined set. This method enables the model to generalize better and detect objects that it has not seen explicitly during training.

However, this strength in open vocabulary recognition comes with the trade-off of increased computational time and complexity. The processing power required to handle such a large number of potential object classes is significantly higher compared to closed vocabulary models. This increased computational demand can affect the speed and efficiency of the model, especially in real-time applications where quick detection is crucial.

Despite this trade-off, the capabilities of Detic represent a significant advancement in the field of object detection. Its ability to accurately identify a vast array of different object classes, including those that were previously difficult to detect, opens up new possibilities for applications in various domains. For instance, in dynamic environments such as autonomous driving, surveillance, and healthcare, the ability to recognize a wide variety of objects with high accuracy is invaluable.

Overall, the shift towards open vocabulary object detection, exemplified by Detic, marks a transformative step forward. By overcoming the limitations of closed vocabulary systems, OVD models like Detic are pushing the boundaries of what is possible in object detection, paving the way for more versatile and robust AI systems that can operate effectively in the diverse and unpredictable real world.

The Detic model consists of two stages: The first stage utilizes a specific CNN model, such as RPN (Region Proposal Network) [8], to generate a set of proposals. The second stage processes object features, providing classifi-

cation scores and refined locations for each object (using CLIP) [9].

3. Model distillation

The model distillation approach is a technique designed to leverage a large, bulky pre-trained model to label data, which is then used to train a smaller, more efficient model. This smaller model is optimized for real-time performance while being limited to recognizing a specific number of classes. The primary objective is to transfer the knowledge from the larger model to the smaller model, enabling it to achieve high accuracy without the extensive computational resources required by the larger model.

One of the pioneers in this area is Autodistill [10], developed by Roboflow. Autodistill exemplifies the model distillation approach by utilizing the capabilities of large, pre-trained models to generate high-quality labeled datasets. These datasets are then used to train smaller models that can perform object detection tasks with impressive speed and efficiency. This process significantly reduces the time and effort required to manually label data, while also ensuring that the smaller models retain a high level of accuracy and robustness.

Autodistill's approach addresses several critical challenges in the field of computer vision. Firstly, it mitigates the issue of computational resource constraints by enabling the deployment of lightweight models capable of real-time performance on devices with limited processing power. Secondly, it enhances the accessibility of advanced object detection capabilities, making them available to a wider range of applications, from mobile devices to embedded systems.

By streamlining the model training process and improving the efficiency of object detection tasks, Autodistill and similar model distillation techniques are driving significant advancements in the field of computer vision. These innovations are paving the way for more practical and scalable AI solutions, enabling broader adoption and integration of sophisticated computer vision technologies across various industries.

4. Real-time object detection

Various versions of the You Only Look Once (YOLO) [11–13] object detection model prioritize real-time processing speed with high accuracy, making them ideal for applications that demand fast processing. These single-stage models process the entire image at once, achieving high speed with minimal latency. However, this emphasis on speed comes with a trade-off: traditional YOLO models can only detect objects they have been trained on.

YOLOv8 [14], the current stable version of their YOLO, focuses on real-time object detection capabilities. Similar to other single-stage models, YOLOv8 excels in processing speed, achieving high frame rates for quick processing efficiency. While the recent release of YOLOv9 [15] marks further advancements in their YOLO lineage, YOLOv8 remains a robust and well-tested choice for applications that require real-time processing. Thanks to its balance of speed and accuracy, YOLOv8 emerges as an ideal choice for scenarios that demand immediate results, while YOLOv9 promises further improvements in the future. YOLOv8 is one of the most-endorsed real-time object detection models for the sake of application purposes.

5. Object Detection for everyone

Fit-Q [16] is one of the pioneers in bringing AI to Android app, this app implements the pose detection to locate key-joints of a person, then counts the number of reps that is performed, this app can detect various types of exercise including push-up, sit-up, burpee, etc.

Vmake [17] is a fashion app, this application takes in a normal image of a clothes item, then generates a high-quality advertizable image of that item tried on by a model, who can be customized with various types of body shape and facial expression.

Moreover, RepDetect[18] is also a fitness app that is not yet published onto Google Play, offers a wide number of exercise rep counting, similar to Fit-Q.

However, the approach of making the training stages accessible to everyday users with no technical expertise has not been adequately addressed by any publicly available work. While there have been significant advancements in user-friendly interfaces and simplified tools, the process of training models, especially in the context of machine learning and computer vision remains largely confined to those with specialized knowledge. Existing platforms and applications often assume a level of technical proficiency that excludes a significant portion of potential users. This gap highlights a critical area for development, as empowering non-experts to engage in the training process could democratize access to AI and significantly expand its practical applications. Despite the growing demand for more accessible AI solutions, there is a noticeable lack of public resources or tools that fully address this need.

III. Methodology

Here we will describe the core pipeline of the model creation process, from user's input to the result model. Then, in Section V, we will explain our application design as well as user flow.

1. Model pipeline

The overall concept of our model creation pipeline is inspired by Autodistill's distillation approach. This method uses a large pre-trained model to train a lightweight model. Our model pipeline, illustrated in Figure 1, follows this approach. We implemented knowledge distillation by using the annotated dataset from the Detic model to train the smaller YOLOv8 model. This process involved transferring the learned knowledge from the larger Detic model to the smaller one, following the distillation concept of extracting a portion of a large model's knowledge into a smaller model. Specifically, an open-vocabulary object detection model (the base model) is employed to label raw images of the desired objects, creating a dataset for training the target model. The setup of our pipeline is as follows:

a) Dataset preparation: images preparation

Since user-uploaded images may not provide sufficient coverage of all object angles and the diverse environments in which the objects may appear, we enhance the dataset by incorporating additional images. These supplementary images are sourced from public databases through APIs, broadening the dataset's scope. This approach ensures better representation of various object perspectives and environmental contexts, improving the model's performance and overall accuracy in object detection tasks.

b) Dataset preparation: image labeling

The image set is then labeled automatically by a base model named by Autodistill [19], which is pre-trained on gigantic image datasets that has the ability to detect any object in the vocabulary.

Furthermore, dataset augmentation is added to improve the dataset's coverage of environments. Augmentation methods that are used include: rotation, shear, brightness, saturation, exposure, etc. These augment operations are added at a fixed ratio of the dataset for everytime a new dataset is created.

c) Target model training

With the dataset prepared, we train a relatively small model on the dataset, with the expectation that it will be able to find user-defined objects at real-time speed.

We train a YOLOv8 model on the dataset with hyperparameters mostly kept as default, with some modifications to match our resources such as batch size, amp (turn off to prevent nan loss, as advised by YOLOv8 owner).

d) Model deployment

We will run the model on a cloud server, the customer's camera frames will be sent to the server for processing, and the annotated frames with bounding box indicating objects will be sent to customer's devices to show onto screen.

As a result, our pipeline is able to produce real-time object detection models with easy input of only object classes and sample images for each class.

IV. Results and Discussions

We evaluate the two most crucial aspects of our apps: the target model produced by the pipeline, and the intermediate dataset created from user input and the base model used to train the target model.

1. Evaluation of the target model

We run a full pipeline and evaluate the resulting model's detection on mAP50-95 score. We evaluate on two sets of object, *common objects* and *rare objects*.

a) Common Objects

The common objects were randomly selected from the classes in COCO dataset, all 80 classes of COCO dataset can be found in Table I.

Here we only randomly select 10 classes to evaluate, 10 classes are shown in Table I, including *keyboard*, *scissors*, etc.

Table I
Object classes used to evaluate our pipeline

Common Objects	Uncommon Objects
Keyboard	Snail
Scissors	Mango
Cow	Durian
Motorcycle	Dragon Fruit
Giraffe	Mangosteen
Laptop	Frangipani
Mouse	Lotus
Potted Plant	Mini Crispy Pancakes
Banana	Vietnamese Banana Cake
Carrot	Chung Cake

b) Rare Objects

In addition to common objects, which frequently appear in object recognition datasets, we selected a number of special object categories, such as certain dishes or flowers. The 10 selected objects are shown in Table I, including *snail*, *mango*, *durian*, etc..

The quantitative recognition results are presented in Table II. The Detic model, designed for labeling and creating datasets, exhibits acceptable processing time given its complexity. However, the overall performance remains relatively low compared to the practical expectations for an object recognition model.

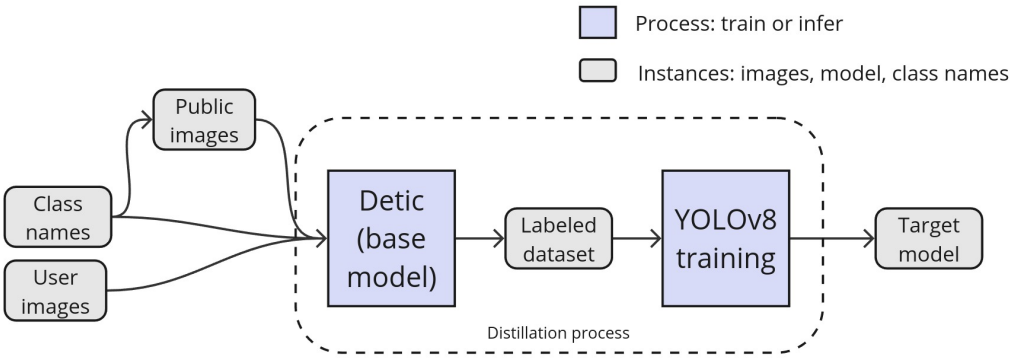


Figure 1. The model training pipeline

Table II
RESULT OF TARGET MODEL (YOLOV8) AFTER TRAINING

Results	Common objects	Rare objects
mAP50 – 95	0.305	0.055
Time (hh:mm)	1:28	1:32

2. Quality of the created dataset

Since the target model's mAP score is calculated on the base-model-labeled dataset, the quality of this dataset should also be evaluated.

To evaluate the base model's accuracy of label results, we make Detic [19] detect on labeled dataset to see how much Detic matches expected results. We also evaluate on common objects and rare objects.

a) Common Objects

We run Detic in inference mode on the valid set of COCO 2017 dataset [20], which contains about five thousand images, the results of Detic compared to labels from COCO is as below:

- mAP50-95: 0.55
- Inference time: 0.70s/image (total time 58:34)

Speaking of accuracy, Detic performs pretty well, comparable to state-of-the-art models in object detection. However, inference time of Detic is relatively high due to the size of the model, but as we use Detic as a base model, processing times does not pose a concern in our work.

b) Rare Objects

To measure rare object performance of the Detic model, we select a public dataset from Roboflow [21], which contains 100 labeled images of snails. The inference result of Detic is:

- mAP50-95: 0.144
- Inference time: 3.94s/image (total time 06:34)

Compared to common objects, rare objects cause Detic more difficulties, with the mAP score achieved at only 0.144. The inference time of 3.94s per image is signifi-canyly high, possibly due to the small number of images to infer, and Detic run from cold-start.

V. DEPLOYMENT

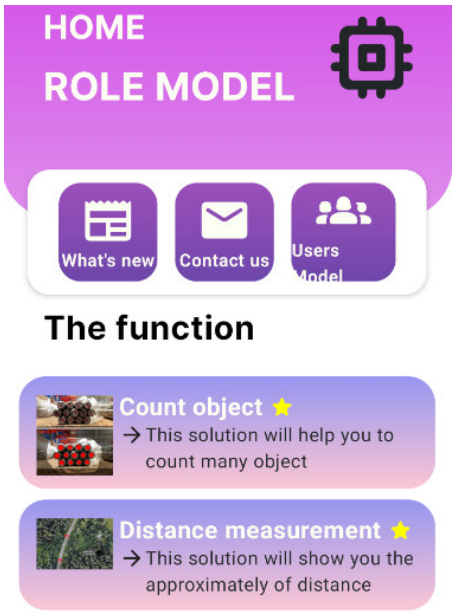
1. Front-end: user interface

Our app design targets making advanced technology accessible to everyday users, regardless of their technical background. The primary goal is to create an intuitive and user-friendly interface that allows individuals with no prior technical knowledge to easily navigate and utilize the app's features. By prioritizing simplicity and ease of use, we aim to democratize access to powerful tools and functionalities that were previously available only to tech-savvy users or professionals.

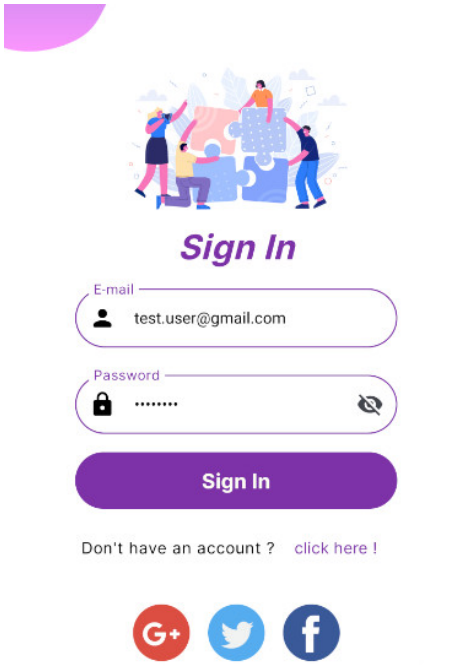
To achieve this, our design incorporates clear instructions, visual aids, and interactive tutorials that guide users through each step of the process. The interface is streamlined to minimize complexity and avoid technical jargon, ensuring that users can quickly grasp how to operate the app, as demonstrated in Figure 2, other snapshots of the application are reported in Figure 2.

The simple stages that a user will experience while creating a model include:

- Keyword input: User needs to provide the system with names of objects expected to be found in images/videos.
- Submit and wait for the model to be trained.
- Once model training is done, the user can load the model for detection on live camera.
- Nevertheless, our app allows users to create multiple models, and call one model to run at a time.



(a) Homepage



(b) Sign in

Figure 2. UI samples

2. Backend

Behind the scenes, the app is composed of several key components that work together seamlessly to deliver real-time object detection and user-friendly functionality. At the core of the system is a robust server infrastructure, equipped with powerful computational units capable of running complex detection algorithms in real time. This server processes the visual data, executes the detection tasks, and ensures that the results are delivered swiftly to the user.

User and model-related data, including user profiles, preferences, and trained models, are securely stored and managed within Firebase, a cloud-based platform that provides scalable and reliable data management services. Firebase serves as the backbone for data synchronization, enabling real-time updates and interactions between the app's various components.

The user interface (UI) application acts as the crucial intermediary that connects the server, Firebase, and the end user. This UI is designed to be intuitive and responsive, allowing users to interact with the app effortlessly. It facilitates communication between the user's device and the server, ensuring that detection requests are processed efficiently and that results are displayed promptly. Additionally, the UI provides access to various features and settings, enabling users to customize their experience and manage their data with ease.

The entire system is designed to be both scalable and efficient, capable of handling multiple simultaneous users while maintaining high performance. This architecture ensures that the app can deliver advanced object detection capabilities in a way that is both accessible and practical for everyday use.

The system's overall design, including the interaction between the server, Firebase, and the UI application, is illustrated in Figure 3. This diagram provides a visual representation of how the components are integrated, highlighting the flow of data and the processes that drive the app's functionality.

VI. Conclusion

This paper has presented a novel approach to democratizing object detection model creation. By introducing a streamlined pipeline for dataset creation and model training, wrapped in a user-friendly application, we have lowered the barrier of entry for individuals and SMEs to leverage the power of computer vision. Our proposed solution emphasizes simplicity and accessibility, enabling users to build effective object detection models with minimal technical expertise.

By making computer vision more accessible, we aim to contribute to the broader application of AI in everyday life and business operations. Our work has the potential to reduce the cost and complexity associated with implementing AI solutions, thereby fostering innovation and driving economic growth.

Looking ahead, we envision further enhancing our platform by expanding dataset capabilities and refining the base

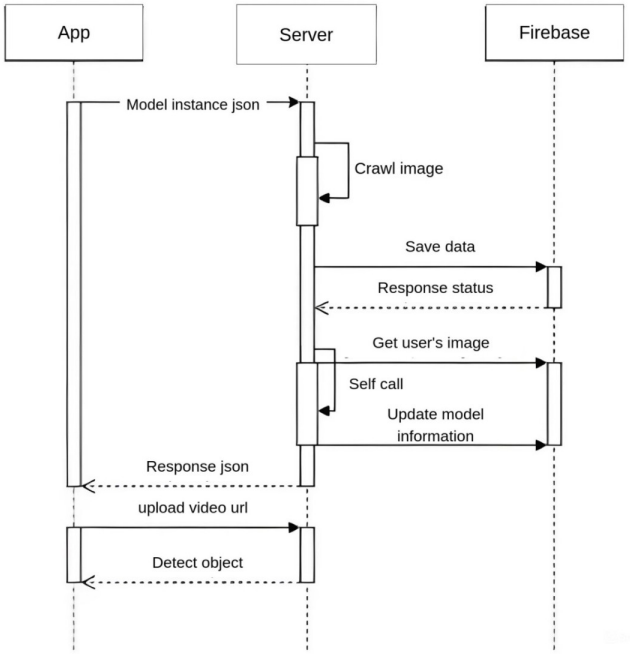


Figure 3. Backend components of the application

model to achieve even higher detection accuracy. These advancements will solidify our solution's position as a leading tool for rapid and cost-effective object detection model development.

REFERENCES

[1] V. K. Kukkala, J. Tunnell, S. Pasricha, and T. Bradley, "Advanced driver-assistance systems: A path toward autonomous vehicles," *IEEE Consumer Electronics Magazine*, vol. 7, no. 5, pp. 18–25, 2018.

[2] M. M. Antony and R. Whenish, "Advanced driver assistance systems (adas)," in *Automotive Embedded Systems: Key Technologies, Innovations, and Applications*. Springer, 2021, pp. 165–181.

[3] M. Payal, P. Dixit, T. Sairam, and N. Goyal, "Robotics, ai, and the iot in defense systems," *AI and IoT-Based Intelligent Automation in Robotics*, pp. 109–128, 2021.

[4] A. Khang, V. Abdullayev, E. Litvinova, S. Chumachenko, A. V. Alyar, and P. Anh, "Application of computer vision (cv) in the healthcare ecosystem," in *Computer Vision and AI-Integrated IoT Technologies in the Medical Ecosystem*. CRC Press, 2024, pp. 1–16.

[5] A. Zareian, K. D. Rosa, D. H. Hu, and S.-F. Chang, "Open-vocabulary object detection using captions," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 14 393–14 402.

[6] H. Bangalath, M. Maaz, M. U. Khattak, S. H. Khan, and F. Shahbaz Khan, "Bridging the gap between object and image-level representations for open-vocabulary detection," *Advances in Neural Information Processing Systems*, vol. 35, pp. 33 781–33 794, 2022.

[7] X. Zhou, R. Girdhar, A. Joulin, P. Krähenbühl, and I. Misra, "Detecting twenty-thousand classes using image-level supervision," in *European Conference on Computer Vision*. Springer, 2022, pp. 350–368.

[8] P. Tang, X. Wang, A. Wang, Y. Yan, W. Liu, J. Huang, and A. Yuille, "Weakly supervised region proposal network and object detection," in *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

[9] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever, "Learning transferable visual models from natural language supervision," *CoRR*, vol. abs/2103.00020, 2021. [Online]. Available: https://arxiv.org/abs/2103.00020

[10] Roboflow, "autodistill." [Online]. Available: \url{https://github.com/autodistill/autodistill}

[11] 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.

[12] P. Jiang, D. Ergu, F. Liu, Y. Cai, and B. Ma, "A review of yolo algorithm developments," *Procedia computer science*, vol. 199, pp. 1066–1073, 2022.

[13] J. Du, "Understanding of object detection based on cnn family and yolo," in *Journal of Physics: Conference Series*, vol. 1004. IOP Publishing, 2018, p. 012029.

[14] G. Jocher, A. Chaurasia, and J. Qiu, "Ultralytics YOLO," jan 2023. [Online]. Available: https://github.com/ultralytics/ultralytics

[15] C.-Y. Wang, I.-H. Yeh, and H.-Y. M. Liao, "Yolov9: Learning what you want to learn using programmable gradient information," *arXiv preprint arXiv:2402.13616*, 2024.

[16] real BI GmbH, "Fit-q: Ai fitness + gaming." [Online]. Available: https://play.google.com/store/apps/details?id=com.statletics.bodyweightconnect

[17] Pixocial technology PTE. LTD., "Vmake ai fashion model studio." [Online]. Available: https://play.google.com/store/apps/details?id=com.airbrush.vmake

[18] G. Ngo, "Repdetect," https://github.com/giaongo/RepDetect, 2024-07-19.

[19] X. Zhou, R. Girdhar, A. Joulin, P. Krähenbühl, and I. Misra, "Detecting twenty-thousand classes using image-level supervision," *CoRR*, vol. abs/2201.02605, 2022. [Online]. Available: https://arxiv.org/abs/2201.02605

[20] T. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: common objects in context," *CoRR*, vol. abs/1405.0312, 2014. [Online]. Available: http://arxiv.org/abs/1405.0312

[21] NBDuy, "Snail dataset," https://universe.roboflow.com/nbduy/snail-edpnm, dec 2021, visited on 2024-07-21. [Online]. Available: https://universe.roboflow.com/nbduy/snail-edpnm

Khoi Nguyen The received the bachelor's degree in artificial intelligence from FPT University. His current research interests include computer vision, object detection, deep learning, and self-supervised learning approaches.
Email: nguyenthekhoig7@gmail.com

Tuong Ho Vinh is a final-year Software Engineering student at Ton Duc Thang University, specializing in computer vision with a focus on self-supervised learning and multimodal approaches for open-vocabulary detection tasks. His research direction is to explore the integration of textual and visual modalities to improve the adaptability and generalization of computer vision models.
Email: viktorho210@gmail.com

Anh HOANG received the B.S. degree in Telecommunication engineer from the Department of Electrical, Electronic, and Information Engineering, Hanoi University of Transport and Communication, in 2007, and the M.S. degree in Computer Science (major in Wireless Networks Security) from the National Taiwan University of Science and Technology (NTUST), Taiwan, in 2010. He completed Ph.D. program at the Graduate School of Advanced Science and Technology (major in Knowledge Science), Japan Advanced Institute of Science and Technology (JAIST), Japan, in September 2021. His research interests are related to AI/Machine Learning, Data Science/Data Mining/Data Analytics, CyberSecurity, and Business Intelligence/ Business Analytics. He is currently teaching at School of Business Information Technology, College of Technology and Design, University of Economics Ho Chi Minh City (UEH), Vietnam.
Email: anhh@ueh.edu.vn

Applying Deep Learning Models for Weapon Detection in Public Environments Through Serveil-lance Cameras

Dung Nguyen[1], Van-Dung Hoang[2], Van-Tuong-Lan Le[3]

[1] University of Sciences, Hue University, Hue city, Vietnam,
[2] HCMC University of Technology and Education, Ho Chi Minh city, Vietnam,
[3] Hue University, Hue city, Vietnam.

Correspondence: Van-Dung Hoang, dunghv@hcmute.edu.vn
Communication: received 25 Sep 2024, revised 28 Oct 2024, accepted 28 Nov 2024
DOI: 10.32913/mic-ict-research.v2025.n1.1318

***Abstract*: In the context of public security, the ability to detect weapons in real time through surveillance cameras is of utmost importance. This study focuses on applying fine-tuning techniques and data augmentation strategies to a Transformer-based model to enhance the capability of identifying weapons in public environments. The fine-tuning process optimized the model parameters, along with these augmentations, to improve weapon detection performance. Experimental results show that after fine-tuning, the model achieved an mAP0.5 score of up to 96.5%, representing a significant improvement in accuracy compared to previous object detection models. These results demonstrate the great potential for applying the model to real-time security surveillance systems, effectively detecting and addressing threats.**

Keywords: *Transformer, Real time, Weapon detection, Surveillance system*

I. Introduction

Weapon detection in public environments through surveillance cameras is a crucial and challenging task in the field of security. To address this issue, the application of modern deep learning techniques, including Convolutional Neural Networks (CNN) [1–4], and Transformer networks [5], has become a prominent research trend. These methods not only improve object recognition accuracy but also enhance processing capabilities under complex real-world conditions. In the paper [6], the authors developed a dataset for evaluating handgun detection from surveillance camera images, aiming to establish a new standard for weapon detection methods. This dataset supports deep learning models for training and testing in complex conditions, enhancing the ability to detect handguns even in crowded environments with high noise levels. However, a major limitation of this dataset is the lack of diversity in the surveillance scenarios and conditions. The handgun detection scenarios are mainly simulated in environments with a certain level of noise and complexity, but it does not cover all real-world situations, such as nighttime conditions or low-light footage.

The paper [7] introduces a multi-level feature pyramid technique for deep learning models, which helps improve the detection of atypical guns in surveillance video. This method enables the system to effectively detect guns even when they have unusual shapes and sizes, overcoming the limitations of traditional object detection methods. However, this technique requires complex computations and may face challenges when deployed in real-time systems, especially in crowded environments or when there are many objects to detect simultaneously. Furthermore, the ability to detect objects in low-light or blurry images remains limited.

In [8], the authors highlight the challenges of real-time gun detection from CCTV, particularly in scenarios with various sources of noise such as lighting changes, complex viewing angles, and crowds. This study recommends developing more robust methods to address these challenges in security surveillance. However, the paper does not provide a complete solution for issues such as handling lighting changes or difficult viewpoints, which means that current models still lack high accuracy in real-world situations.

Finally, the paper [9] presents an anomaly detection method for surveillance video, allowing the system to automatically identify abnormal or dangerous situations. This method contributes to enhancing security by assisting in the detection of abnormal behaviors related to violence or threats. However, one limitation of this approach is the classification of abnormal behaviors, especially in non-dangerous situations. The system may generate false alerts,

particularly when similar behaviors occur in contexts with no clear threat, reducing the effectiveness and reliability of the system.

Current research and methods have made significant contributions to the development of weapon and anomaly detection models in public surveillance. However, a major weakness of these models is their generalization ability in real-world scenarios, especially under complex conditions such as changing lighting, difficult viewpoints, or crowded environments. Additionally, current methods have not fully addressed the issue of accuracy in detecting objects in high-noise situations or low-quality footage. Minimizing false alerts in situations without clear threats remains a challenge that must be overcome to improve the efficiency and reliability of surveillance systems.

In this study, we focus on applying Fine-tuning techniques [10–16] to several object detection models, including Transformer-based models like RT-DETR [17] and some versions of YOLO [18–26]. RT-DETR, a variant of DETR [27], has been fine-tuned on the Weapon dataset to optimize weapon detection capabilities in public environments. Additionally, we conducted fine-tuning on YOLO, a method well-known for its speed and effectiveness in object detection. RT-DETR combines the strengths of Transformer and CNN, effectively handling spatial features and enhancing performance in real-world conditions. Meanwhile, YOLO offers fast and accurate detection capabilities. By fine-tuning both models, we expect to significantly improve the accuracy and detection capabilities of surveillance systems, thereby contributing to the enhancement of public security measures.

II. RELATED WORKS

Object Detection. In the field of computer vision, object detection is a crucial and complex task, requiring models not only to recognize objects but also to determine their locations within an image. Object detection methods are generally classified into two main categories: two-stage models and one-stage models.

Two-stage models, such as R-CNN and its variants [28–30], operate through the following process: **(1) Region Proposal Stage:** The model identifies regions in the image that are likely to contain objects, typically using a Region Proposal Network (RPN) [30]; **(2) Classification and Localization Stage:** After obtaining the proposed regions, the model classifies each region and precisely locates the objects. Two-stage models often achieve high accuracy because the detection and classification steps are clearly separated. However, the downside is slower processing speed, making it challenging to meet real-time requirements.

One-stage models, such as YOLO [18–26], SSD [31], and RetinaNet [32], are designed to detect and classify objects in a single step. These models usually have a simpler network structure, allowing for quick predictions of object positions and types directly from feature points. The biggest advantage of one-stage models is their fast speed, making them suitable for real-time applications. However, their accuracy may be lower compared to two-stage models, especially in complex situations.

Transformer-based Models. Recently, Transformer-based models such as DETR [27] and its variants [17, 33, 34] have emerged as modern solutions, combining high performance with real-time processing capabilities. DETR is one of the pioneering models that apply Transformers to the object detection task. DETR uses a Transformer encoder and decoder to directly predict bounding boxes and object labels from the input image, eliminating the need for region proposal extraction as seen in two-stage models. The architecture of DETR consists of the following key components: **(1) Backbone:** The backbone is typically a deep convolutional network, such as ResNet [35], responsible for extracting features from the input image. These features are then passed through several convolutional and pooling layers to reduce their size while enriching the spatial information; **(2) Encoder:** The encoder takes in the features from the backbone and applies a multi-head attention mechanism to model the relationships between different feature points. The encoder helps extract relevant information from the entire image, unrestricted by the size of feature regions; **(3) Decoder:** The decoder uses object queries, which are learnable embeddings, to predict the bounding boxes and labels of objects. The decoder also applies attention mechanisms to focus on regions with a high probability of containing objects, leading to more accurate predictions; **(4) Hungarian Matching:** DETR employs the Hungarian algorithm to optimize the matching of predicted bounding boxes with actual objects in the image. This approach avoids issues such as duplication or missed objects, which are common in traditional methods; **(5) Set Prediction Loss:** DETR employs a composite loss function that combines bounding box losses (L1 loss and GIoU loss) with classification label loss (cross-entropy loss). The set prediction loss allows the model to assign labels to objects more accurately and efficiently. However, DETR faces several drawbacks, including: long training times, poor performance with small objects, high computational resource requirements, and slow optimization on small datasets. Figure 1 illustrates the structure of the DETR model.

RT-DETR Model. RT-DETR [17] is an enhanced version of DETR, specifically optimized for real-time appli-

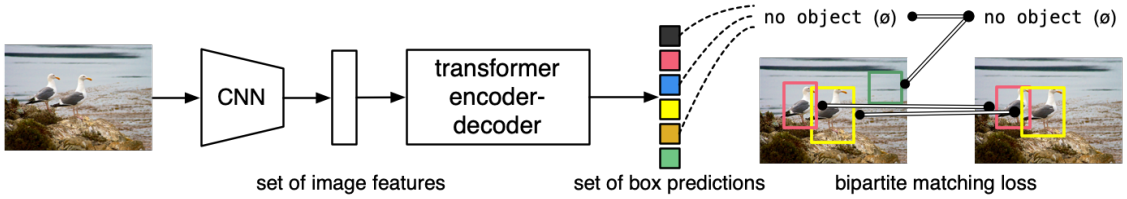


Figure 1. DETR model architecture [27]

cations. It maintains the advantages of the Transformer architecture while improving processing speed without compromising accuracy, thanks to a lighter and more efficient design. RT-DETR inherits many characteristics from DETR but is optimized for performance: **(1) Attention mechanism optimization:** The attention mechanism in RT-DETR is optimized to reduce computational complexity. Instead of applying attention over the entire image space, RT-DETR uses several strategies to offload computation, such as hierarchical attention or region-based attention; **(2) Reduction in the number of queries:** RT-DETR reduces the number of queries in the decoder, which helps speed up inference without significantly affecting the model's accuracy. The number of queries is adjusted to cover all objects in the image without wasting computational resources; **(3) Lighter backbone network:** Instead of using complex and heavy backbones like ResNet-101 [35], RT-DETR often uses lighter backbones like ResNet-50 [35] or even MobileNet [36, 37] to increase inference speed while still ensuring detection performance. With these improvements, the RT-DETR model achieves notable efficiency: **(1) High processing speed:** RT-DETR is designed to meet real-time requirements, capable of processing tens to hundreds of frames per second, depending on hardware configuration and input size; **(2) High accuracy:** Despite being optimized for speed, RT-DETR still maintains high accuracy in detecting and classifying objects; **(3) Generalization ability:** With the Transformer-based attention mechanism, RT-DETR can handle complex scenes and difficult-to-recognize objects, a challenge for models purely based on CNNs.

Fine-Tuning Technique. Fine-tuning is a powerful technique in deep learning, particularly useful when working with pre-trained models on large datasets. Fine-tuning is typically applied in scenarios where the target dataset is small or specific compared to the original dataset, or when the new task is related but distinct from the one the original model was trained on. The fine-tuning process generally begins with a model pre-trained on a large, common dataset like ImageNet [38] or COCO [39]. The main steps in the fine-tuning process include: **(1) Using the pre-trained model:** The model is loaded along with the weights learned from training on a large dataset. These weights help the model capture general features of images; **(2) Freezing the initial layers:** The early layers of the model, where general features are learned, are usually kept unchanged and not re-trained. This helps preserve the general features learned from the original dataset. Specifically, the model freezes at layer S3, meaning that the layers before S3 will remain unchanged, while the subsequent layers (from S4 onwards) will be fine-tuned to adapt to the new task; **(3) Training the later layers:** The final layers of the model, where more complex features are learned, are re-trained on the new dataset. This allows the model to adjust and learn the specific features of the new task; **(4) Tuning hyperparameters.** During fine-tuning, hyperparameters like learning rate, batch size, and optimization function can be adjusted to optimize training on the new data. Benefits of fine-tuning: **(1) Saves time and resources:** Since the model already has foundational knowledge from previous training, fine-tuning doesn't require learning from scratch, reducing both training time and computational resources; **(2) Improves performance:** Fine-tuning helps the model better adapt to the target data, improving accuracy and generalization in practical applications; **(3) Flexible application:** Fine-tuning can be applied to a variety of tasks, from image classification and object detection to natural language processing, broadening the applicability of deep learning models.

III. WEAPON DETECTION METHOD

1. IMPLEMENTATION DETAIL

The workflow of the model begins with processing the input images using data augmentation techniques such as flipping, rotating, and color adjustments to increase the diversity of the training set. After augmentation, the images are fed into the Backbone, which has been pre-trained on the ImageNet dataset [38], to extract essential features. Additionally, we use a model pre-trained on the COCO dataset [39] to initialize the parameters. The model is then fine-tuned on the dataset, as described in Section III.2.

The features are then passed through two key components: AIFI (Attention-based Intra-scale Feature Interac-

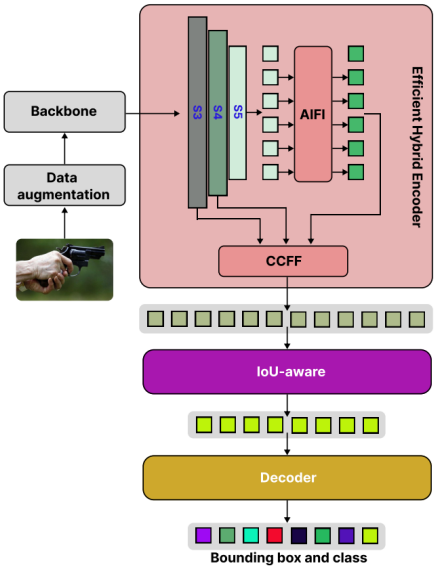


Figure 2. Structure of the model

tion) and CCFF (CNN-based Cross-scale Feature Fusion) for fusion. AIFI utilizes attention mechanisms to interact with features within the same scale, focusing on important information at each level of object detail, enhancing the key features, especially for weapon detection. CCFF combines features from multiple scales, ensuring the model accurately detects objects with varying sizes and positions in the image.

After processing through AIFI and CCFF, the features are merged into a unified representation, then passed through the IoU-Aware module to improve accuracy in predicting object locations, and finally through the Decoder to predict the labels and locations of weapons in public surveillance environments. Figure 2 illustrates the structure of the model.

2. DATASET

The Weapon Dataset [40] is a dataset containing images of various types of weapons in public environments. This dataset provides images along with labels describing the type of weapon and its location. The dataset is divided into two sets: a training set and a testing set. Figure 3 illustrates the distribution of the number of samples for objects in the training dataset.

From the chart, it can be observed that: Knife is the most prevalent object in the dataset, with over 3,000 samples, significantly surpassing other objects; Gun and Pistol have relatively similar numbers of samples, with around 1,500 samples for Gun and approximately 1,200 for Pistol; Handgun has the fewest samples in the dataset,

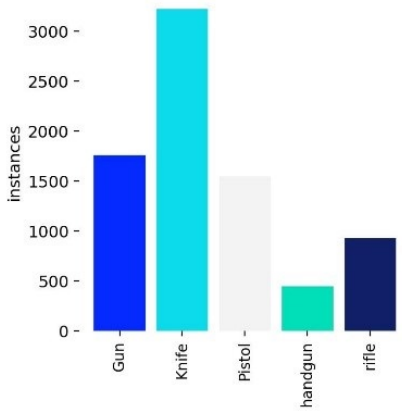


Figure 3. Object distribution by label in the training set

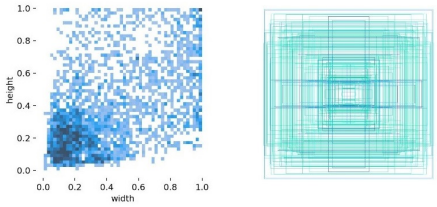


Figure 4. The ratio of bounding box area to frame area in the training set

with only around 500; Rifle has a moderate number of samples, with about 1,000. The large disparity in the number of samples between objects may impact the model's performance, particularly for objects with fewer samples, such as Handgun, which may be harder to detect accurately.

To address this imbalance, Focal Loss [32] has been employed during the training process. This loss function is particularly effective in scenarios where the dataset is imbalanced by assigning higher weights to less frequent objects. By focusing more on these underrepresented classes, Focal Loss enhances the model's ability to detect objects such as Handgun and Rifle, which have fewer samples, thereby improving overall classification accuracy and robustness in real-world applications. This technique ensures that the model does not overly favor the detection of frequent objects like Knife, contributing to a more balanced and effective weapon detection system.

Figure 4 illustrates the ratio of object bounding box areas to the areas of the images containing them in the training set. Based on the distribution of bounding box sizes relative to the frame, we observe that most detected objects are small, with widths and heights predominantly below 0.2 of the overall frame size. The high density of values in this region indicates that the model faces challenges in detecting small objects. This emphasizes the importance of fine-tuning the model to enhance sensitivity and accuracy

in detecting small objects, reducing missed or misidentified detections.

3. DATA AUGMENTATION

Data augmentation techniques are an important method in the field of deep learning, used to expand and enrich the training dataset. By creating variations of the original data, such as resizing, flipping, shifting, and adjusting colors, this technique helps the model learn to recognize objects under different conditions, thereby improving the model's generalization capability. The application of data augmentation techniques not only helps reduce overfitting but also enhances the model's performance in real-world scenarios. In this study, data augmentation techniques such as scaling, horizontal flipping, shifting, and adjusting colors in the HSV space, as described in Table I, have been used to improve the training efficiency of the RT-DETR model on the Weapon dataset.

4. MODEL TRAINING

To train the model on the Weapon dataset, the RT-DETR model was initialized from a pre-trained version on the COCO dataset. The final layers of the model were then adjusted to be compatible with the number of classes in the Weapon Dataset. The fine-tuning technique was applied by adjusting the model's parameters, as described in Table II, ensuring that the model learns the specific features of the new dataset. The AdamW optimizer was used with an appropriate learning rate, optimizing the training process without causing overfitting.

IV. EXPERIMENTS AND RESULTS

1. TRAINING MACHINE SPECIFICATIONS

In this experiment, the model was implemented on a device with the configuration described in Table III.

2. EVALUATION OF TRAINING RESULTS

Common criteria used to evaluate the performance of object detection algorithms include IoU, Precision, Recall, and mAP. Detailed definitions are listed below.

A. IoU: Intersection over Union (IoU) is calculated by taking the area of intersection between the predicted region (A) and the ground truth region (B) and dividing it by the total area of the union of these two regions. The formula can be expressed as follows:

$$IoU = \frac{A \cap B}{A \cup B} \tag{1}$$

The IoU value ranges from 0 to 1. A higher value indicates better model accuracy. Specifically, a lower numerator indicates inaccurate predictions of the ground truth region, while a larger denominator suggests that the predicted region is wider, leading to a lower IoU value.

B. Precision: Precision represents the ratio of correctly predicted samples among the set of predicted samples. It can be expressed as follows:

$$Precision = \frac{TP}{TP + FP} \tag{2}$$

where:

- TP (True Positive): The number of correctly predicted samples
- FP (False Positive): The number of incorrectly predicted samples

C. Recall: Recall represents the ratio of correctly predicted samples among the total set of actual samples. It can be expressed as follows:

$$Recall = \frac{TP}{TP + FN} \tag{3}$$

where:

- TP (True Positive): The number of correctly predicted samples
- FN (False Negative): The number of samples that are incorrectly predicted and not predicted at all

D. mAP: Average Precision (AP) is a measure of Precision values at different thresholds on the Precision-Recall (PR) curve, and is calculated as a weighted average. Mean Average Precision (mAP) is the average value of AP across all classes. Specifically, mAP@0.5 indicates the mAP when IoU is 0.5, and mAP@[0.5:0.95] represents the mean mAP when IoU ranges from 0.5 to 0.95.

$$mAP = \frac{1}{N} \sum_{i=1}^{N} \mathrm{AP}_i \tag{4}$$

where:

- AP: The area under the Precision-Recall (PR) curve calculated at different thresholds
- N: The number of classes

3. TRAINING RESULTS

Figure 5 illustrates the training process of the model through the error function values over epochs. Specifically, the GIoU loss during training starts at 0.55605 and fluctuates slightly across epochs, indicating that the model is gradually improving its localization ability. The Cls loss starts high (around 2.6170) and decreases over time, demonstrating that the model is learning to classify better. The L1 loss fluctuates within a narrow range, from 0.44467

Table I
A FEW DATA AUGMENTATION TECHNIQUES

Techniques	Value	Description
scale	0.5	This technique resizes the image by scaling it up or down by a certain ratio. With a factor of 0.5, the image will be reduced to 50% of its original size. This helps the model learn to recognize objects at different sizes, thus improving the model's generalization ability
fliplr	0.5	This technique flips the image horizontally with a 50% probability. This creates variations of the original image by inverting it along the horizontal axis. Horizontal flipping helps the model learn to recognize objects from different angles, potentially improving recognition in situations where the object may appear from different directions
translate	0.1	This technique shifts the image horizontally and vertically by a certain percentage. With a factor of 0.1, the image can be shifted by up to 10% of the original size in both directions. This helps the model learn to recognize objects even when they are not centered in the image, thereby improving recognition in various scenarios
hsv_h	0.015	This technique changes the hue of the image in the HSV (Hue, Saturation, Value) color space. With a factor of 0.015, the hue of the image can change by ±1.5% of its original hue value. Changing the hue helps the model learn to recognize objects under different lighting and color conditions, thereby enhancing the model's generalization capability
hsv_s	0.7	This technique adjusts the saturation of the image in the HSV color space. With a factor of 0.7, the saturation can vary by ±70% of the original saturation value. This helps the model learn to recognize objects under different color conditions, ranging from highly saturated colors to more muted tones
hsv_v	0.4	This technique changes the brightness (value) of the image in the HSV color space. With a factor of 0.4, the brightness of the image can vary by ±40% of its original brightness value. Changing the brightness helps the model recognize objects under different lighting conditions, including both low-light and high-light environments

Table II
TRAINING HYPERPARAMETERS

Parameter	Value	Description
Epoch	50	The number of epochs refers to how many times the entire training dataset is used to train the model
Learning rate	0.00001	The learning rate determines the extent to which the model's weights are adjusted after each update step based on the loss function
Batch size	4	Batch size is the number of data samples used in each step of weight updates for the model
Image size	640 x 640	Input image size for the model. All images will be resized to 640 pixels x 640 pixels before being fed into the model
Backbone pre-trained	ImageNet	The backbone is the main part of the deep learning model, and in this case, the model uses the ResNet network. The backbone was pre-trained on the ImageNet dataset, a large dataset with various types of images and object classes
Model Pre-trained	COCO	The object detection model was pre-trained on the COCO dataset, a common dataset for object detection tasks
Optimizer	AdamW	AdamW is a variant of the Adam optimization algorithm, with the addition of Weight Decay to control overfitting and improve training performance
Max-Det	300	This is the maximum number of objects the model can detect in an image
Mometum	0.937	Momentum is a parameter in the optimization algorithm that helps improve the model's convergence speed by maintaining part of the gradient direction from previous update steps
Weight decay	0.0005	Weight decay is a technique to reduce overfitting by adding a penalty to the model's weight values

Table III
A FEW SPECIFICATIONS OF THE TRAINING MACHINE

Specifications	Value
Python version	3.9
PyTorch	1.10
CUDA	11.1
GPU	Quadro RTX 4000, 7783MiB

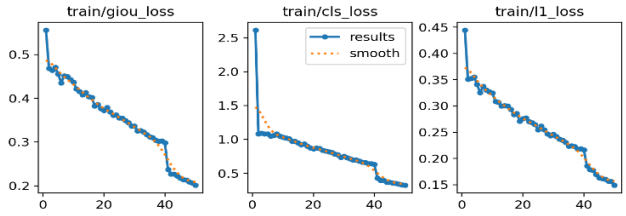


Figure 5. Training results based on Loss Functions

to 0.34084, reflecting stability in matching the predicted positions.

The three charts in Figure 6, 7, 8 provide a comprehensive view of the model's performance through precision, recall, and F1-score metrics across various confidence thresholds. The Precision-Confidence chart shows that the model's precision increases as confidence rises, with the highest value reaching 1.00 at a confidence threshold of around 0.917, indicating excellent model performance at high confidence levels. Meanwhile, the Recall-Confidence chart reveals that recall decreases as confidence increases,

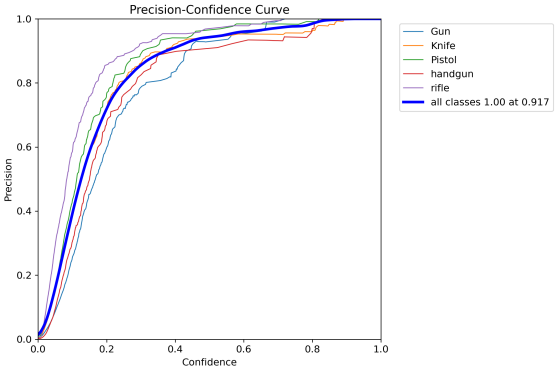


Figure 6. Precision-Confidence Curve

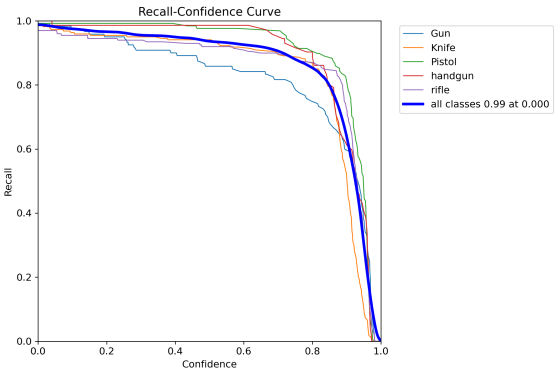


Figure 7. Recall-Confidence Curve

meaning the model overlooks more true positives at higher confidence thresholds. Finally, the F1-Confidence chart peaks at a threshold of 0.58 with an F1-score of 0.94, reflecting a balance between precision and recall at this level. Overall, the model exhibits high precision with higher confidence but has to trade off recall and F1-score as confidence continues to increase.

From Figure 9, the Precision-Recall chart shows the relationship between the model's precision and coverage across different weapon classes. The model performs exceptionally well, with an mAP@0.5 value of 0.965 (96.5%) across all classes. Classes such as Knife, Pistol, Handgun, and Rifle exhibit both high precision and recall, indicating that the model effectively detects these objects. The Gun class has slightly lower precision, at 0.947, but still maintains good performance. This demonstrates that the model operates effectively across various object types, with high accuracy and coverage.

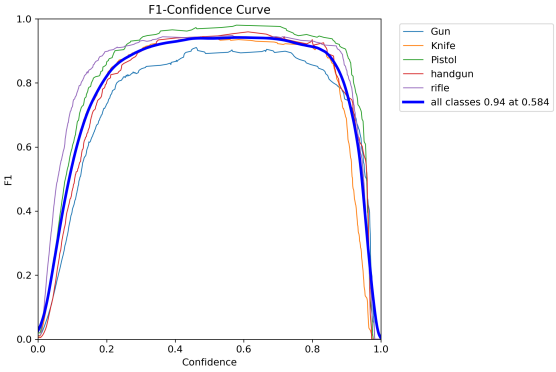


Figure 8. F1-Confidence Curve

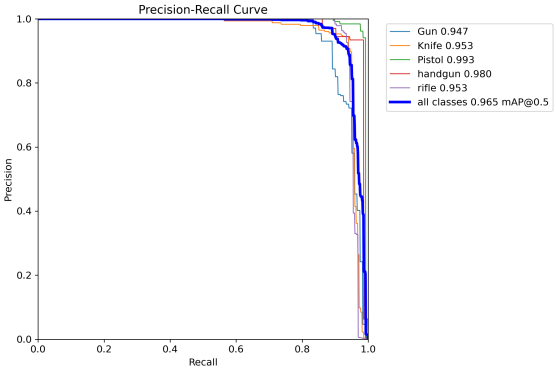


Figure 9. Precision-Recall Curve

Figure 10 illustrates the confusion matrix of the model after training, showing the classification performance across different object categories. The confusion matrix indicates that the model classifies weapons with fairly good performance for the main categories such as Gun, Knife, and Pistol, with high accuracy rates of 96%, 96%, and 99%, respectively. However, there is a slight confusion between different types of weapons, such as confusion between gun and background (29%) and between knife and background (30%). These errors suggest that the model struggles to distinguish objects that may visually resemble weapons. On the other hand, the background category has the highest confusion rate with all weapon types, indicating that the model still needs improvement in detecting non-weapon environments.

Figure 11, 12 illustrates the model's prediction results on several images from the test set.

Table IV compares the performance and efficiency of the YOLOv8, YOLOv9, YOLOv10, RT-DETR, Faster RCNN + FPN, and UGR models on a specific dataset. For small models, YOLOv8-n, YOLOv9-t, and YOLOv10-n achieve a good balance between accuracy and inference speed. Among them, YOLOv9-t stands out by reaching a mAP@0.5 of 85.5% and mAP@0.5-0.95 of 64.3%, while maintaining an inference time of 18.9ms. YOLOv8-n offers a faster inference time of 9.8ms but has a lower mAP@0.5-0.95 of 61.1%. For large models, YOLOv8-l, YOLOv9-c,

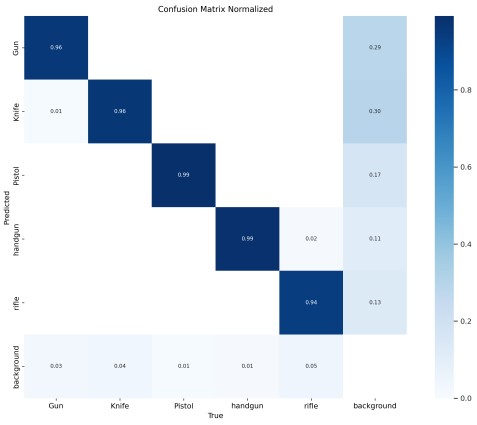


Figure 10. Confusion Matrix on the Test Dataset

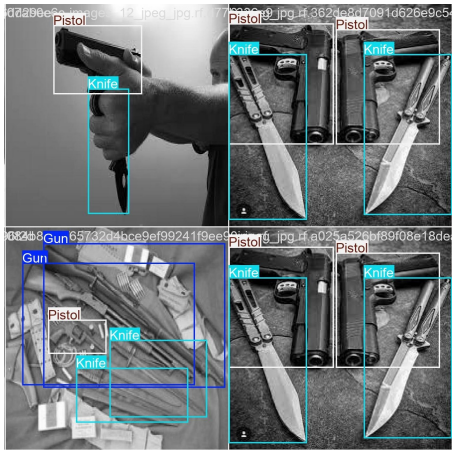


Figure 11. Ground-Truth

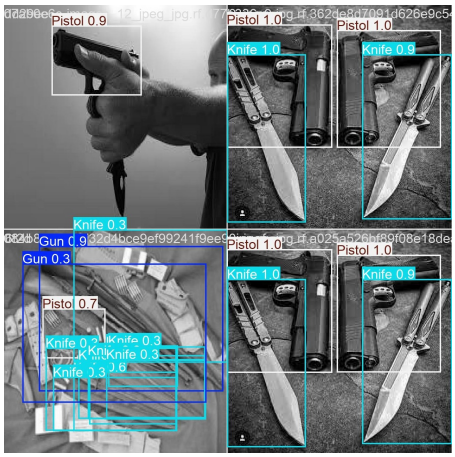


Figure 12. Predicted

and YOLOv10-l require higher computational resources. Notably, YOLOv8-l, with the highest GFlops, achieves a mAP@0.5-0.95 of only 64.0%, which is lower compared to the smaller, more efficient models. In contrast, RT-DETR-l excels in terms of accuracy, achieving a mAP@0.5 of 96.5% and mAP@0.5-0.95 of 79.7%, although it has a slower inference time of 45.1ms. Compared to Faster RCNN + FPN and UGR, RT-DETR-l outperforms both in accuracy, with Faster RCNN + FPN achieving 91.7% mAP@0.5 and UGR reaching 92.6% mAP@0.5. However, both models have lower mAP@0.5-0.95 values (68.3% for UGR), indicating that RT-DETR-l provides better generalization across detection thresholds. This comparison highlights that while RT-DETR-l demands more computational resources, it is the most accurate model, making it highly suitable for tasks that prioritize precision.

V. Conclusions

In this paper, we conducted fine-tuning of the RT-DETR model on the Weapon dataset to improve the detection of weapons in public environments through surveillance cameras. The experimental results show that the fine-tuned RT-DETR model achieved a mAP@0.5 score of up to 96.5%, demonstrating high effectiveness in detecting and classifying weapon objects. This indicates that RT-DETR can accurately and consistently recognize objects in various situations, enhancing the reliability of security surveillance systems. The use of fine-tuning significantly improved the model's performance compared to the original version, especially in real-world conditions where there is significant variation in the shapes and sizes of objects. These results not only affirm the feasibility of RT-DETR in challenging object detection tasks but also open up new avenues for applying Transformer models in surveillance and security systems.

In the future, several improvements and follow-up research can be pursued to further enhance the effectiveness of weapon detection models: (1) Dataset expansion: Expanding the dataset with more diverse weapon types and scenarios can improve the model's accuracy, especially under different environmental conditions. This expansion could also include gathering data from various sources to increase the model's generalization capability; (2) Hyperparameter tuning: Further tuning the hyperparameters of the RT-DETR model, such as the size of the Transformer network and training parameters, could help achieve better performance and reduce training time; (3) Combining with other techniques: Researching the combination of RT-DETR with other deep learning techniques, such as machine learning augmentation methods or lighter deep learning models, could improve inference speed and reduce

Table IV
TRAINING HYPERPARAMETERS

Model	Layers	Parameters	GFlops	Epoch	mAP@0.5	mAP@0.5-0.95	Inference time
YOLOv8-n	255	03.2M	8.90	50	85.5%	61.1%	09.8ms
YOLOv9-t	917	02.0M	7.90	50	85.5%	64.3%	18.9ms
YOLOv10-n	385	02.8M	8.70	50	80.3%	59.3%	28.6ms
Faster RCNN + FPN [9]	N/A	N/A	N/A	N/A	91.7%	N/A	N/A
UGR [8]	N/A	N/A	N/A	N/A	92.6%	68.3%	N/A
YOLOv8-l	365	43.7M	165.7	50	84.9%	64.0%	52.2ms
YOLOv9-c	618	25.5M	104.0	50	85.0%	63.7%	45.0ms
YOLOv10-l	628	25.9M	127.9	50	81.5%	63.5%	42.9ms
RT-DETR-l	**673**	**33.0M**	**108.0**	50	96.5%	**79.7%**	**45.1ms**

computational resource requirements; (4) Application in real-world scenarios: Deploying the model in real-world security surveillance systems and evaluating its performance in live environments could help identify existing issues and improve the model based on the specific requirements of the application. Continued research and development in these areas will not only improve weapon detection technology but also contribute to the overall advancement of more intelligent and efficient security surveillance systems.

ACKNOWLEDGEMENT

This research is funded by Vietnam National Foundation for Science and Technology Development (NAFOSTED) under grant number 102.05-2021.04.

REFERENCES

[1] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, 2012.

[3] K. Simonyan, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[4] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[5] A. Vaswani, "Attention is all you need," *Advances in Neural Information Processing Systems*, 2017.

[6] S. Yellapragada, Z. Li, K. B. Doshi, P. M. Mhasakar, H. Fan, J. Wei, E. Blasch, B. Zhang, and H. Ling, "Cctv-gun: Benchmarking handgun detection in cctv images," *arXiv preprint arXiv:2303.10703*, 2023.

[7] J. Lim, M. I. Al Jobayer, V. M. Baskaran, J. M. Lim, J. See, and K. Wong, "Deep multi-level feature pyramids: Application for non-canonical firearm detection in video surveillance," *Engineering applications of artificial intelligence*, vol. 97, p. 104094, 2021.

[8] J. L. S. González, C. Zaccaro, J. A. Álvarez-García, L. M. S. Morillo, and F. S. Caparrini, "Real-time gun detection in cctv: An open problem," *Neural networks*, vol. 132, pp. 297–308, 2020.

[9] W. Sultani, C. Chen, and M. Shah, "Real-world anomaly detection in surveillance videos," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 6479–6488.

[10] C. B. Do and A. Y. Ng, "Transfer learning for text classification," *Advances in neural information processing systems*, vol. 18, 2005.

[11] L. Torrey and J. Shavlik, "Transfer learning," in *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques*. IGI global, 2010, pp. 242–264.

[12] L. Zhao, S. Pan, E. Xiang, E. Zhong, Z. Lu, and Q. Yang, "Active transfer learning for cross-system recommendation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 27, no. 1, 2013, pp. 1205–1211.

[13] K. Weiss, T. M. Khoshgoftaar, and D. Wang, "A survey of transfer learning," *Journal of Big data*, vol. 3, pp. 1–40, 2016.

[14] N. Agarwal, A. Sondhi, K. Chopra, and G. Singh, "Transfer learning: Survey and classification," *Smart Innovations in Communication and Computational Sciences: Proceedings of ICSICCS 2020*, pp. 145–155, 2021.

[15] A. Hosna, E. Merry, J. Gyalmo, Z. Alom, Z. Aung, and M. A. Azim, "Transfer learning: a friendly introduction," *Journal of Big Data*, vol. 9, no. 1, p. 102, 2022.

[16] Z. Lin, D. Liu, W. Pan, and Z. Ming, "Transfer learning in collaborative recommendation for bias reduction," in *Proceedings of the 15th ACM Conference on Recommender Systems*, 2021, pp. 736–740.

[17] Y. Zhao, W. Lv, S. Xu, J. Wei, G. Wang, Q. Dang, Y. Liu, and J. Chen, "Detrs beat yolos on real-time object detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 16 965–16 974.

[18] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 779–788.

[19] J. Redmon and A. Farhadi, "Yolo9000: Better, faster, stronger," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 6517–6525.

[20] J. Redmon, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[21] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "Yolov4: Optimal speed and accuracy of object detection," *arXiv preprint arXiv:2004.10934*, 2020.

[22] G. Jocher, A. Stoken, J. Borovec, L. Changyu, A. Hogan, L. Diaconu, F. Ingham, J. Poznanski, J. Fang, L. Yu *et al.*, "ultralytics/yolov5: v3. 1-bug fixes and performance improvements," *Zenodo*, 2020.

[23] C. Li, L. Li, Y. Geng, H. Jiang, M. Cheng, B. Zhang, Z. Ke, X. Xu, and X. Chu, "Yolov6 v3. 0: A full-scale reloading," *arXiv preprint arXiv:2301.05586*, 2023.

[24] 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," in *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023, pp. 7464–7475.
[25] C.-Y. Wang, I.-H. Yeh, and H.-Y. Mark Liao, "Yolov9: Learning what you want to learn using programmable gradient information," in *European Conference on Computer Vision*. Springer, 2025, pp. 1–21.
[26] A. Wang, H. Chen, L. Liu, K. Chen, Z. Lin, J. Han, and G. Ding, "Yolov10: Real-time end-to-end object detection," *arXiv preprint arXiv:2405.14458*, 2024.
[27] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," in *European conference on computer vision*. Springer, 2020, pp. 213–229.
[28] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 580–587.
[29] R. Girshick, "Fast r-cnn," in *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 1440–1448.
[30] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, vol. 39, no. 06, pp. 1137–1149, June 2017.
[31] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*. Springer, 2016, pp. 21–37.
[32] T.-Y. Ross and G. Dollár, "Focal loss for dense object detection," in *proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2980–2988.
[33] X. Zhu, W. Su, L. Lu, B. Li, X. Wang, and J. Dai, "Deformable detr: Deformable transformers for end-to-end object detection," *arXiv preprint arXiv:2010.04159*, 2020.
[34] D. Meng, X. Chen, Z. Fan, G. Zeng, H. Li, Y. Yuan, L. Sun, and J. Wang, "Conditional detr for fast training convergence," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 3651–3660.
[35] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.
[36] A. G. Howard, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.
[37] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4510–4520.
[38] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.
[39] 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 *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*. Springer, 2014, pp. 740–755.
[40] J. Assalim, "Fireguns and knifes detector dataset," https://universe.roboflow.com/joo-assalim/fireguns-and-knifes-detector, nov 2023, visited on 2024-11-29. [Online]. Available: https://universe.roboflow.com/joo-assalim/fireguns-and-knifes-detector

Dung Nguyen was born on June 13, 1988 in Thua Thien Hue. He graduated with a bachelor's degree in information technology from University of Sciences, Hue University in 2010. In 2013, he graduated with a master's degree in computer science from University of Sciences, Hue University. Currently he works at University of Sciences, Hue University. Research fields: Software technology, artificial intelligence, machine learning, deep learning, databases.
Email: nguyendung@hueuni.edu.vn

Van-Dung Hoang received a Ph.D. degree in Electrical and Computer Engineering from University of Ulsan, Ulsan city, Korea, in 2015. After a year of experience with Telecom SudParis as a Postdoctoral Research Fellow, he joined the Faculty of Information Technology, HCMC University of Technology and Education, Ho Chi Minh City, Vietnam, where he is currently serving as a professor of Artificial Intelligence Department. His research interests cover a wide area, which focuses on computer vision, robotics, medical image processing, autonomous vehicles, and ambient intelligence.
Email: dunghv@hcmute.edu.vn

Van-Tuong-Lan Le was born on November 10, 1974, in Thua Thien Hue. In 1996, he graduated with a degree in Mathematics and Informatics from the College of Sciences, Hue University. He obtained a Master's degree in Information Technology from Hanoi University of Science and Technology in 2002 and earned a Ph.D. in Computer Science from the College of Sciences, Hue University, in 2018. He is currently working at the Department of Training and Student Affairs, Hue University.
Email: lvtlan@hueuni.edu.vn

A Rich High-Order Mutation Testing Dataset for Software Fortification

Van-Nho Do[1], Giang T.C. Tran[2], Duc-Thuan Nguyen[3], Ngoc-Anh Nguyen-Thi[4], Quang-Vu Nguyen[5], Thanh-Binh Nguyen[6]

[1] Le Quy Don High School for the Gifted, Danang, Vietnam
[2] University of Quebec in Trois-Rivières, Canada. The University of Danang, University of Economics, Vietnam
[3] University of Engineering and Technology, Vietnam National University of Hanoi, Vietnam
[4] University of Engineering and Technology, Vietnam National University of Hanoi, Vietnam
5 The University of Danang, Vietnam-Korea University of Information and Communication Technology, Vietnam
[6] The University of Danang, Vietnam-Korea University of Information and Communication Technology, Vietnam.

Correspondence: Quang-Vu Nguyen, nqvu@vku.udn.vn
Communication: received 26 Jul 2024, revised 8 Nov 2024, accepted 1 Dec 2024
DOI: 10.32913/mic-ict-research.v2025.n1.1277

***Abstract*: High-order mutation (HOM) testing is a rigorous technique for evaluating the effectiveness of test suites by introducing mutations with multiple concurrent faults into the source code. In this study, we present the development and analysis of a comprehensive dataset tailored for HOM testing purposes. The dataset comprises 2,839,792 instances categorized into Survived and Killed classes, representing instances correctly identified as surviving and not surviving the mutation testing process, respectively. We employ four prominent machine learning algorithms—Logistic Regression, Random Forest Classifier, LightGBM, and XGBoost—to classify instances within these categories. Experimental results demonstrate varying levels of accuracy, precision, recall, and F1-score across the algorithms, with LightGBM and XGBoost exhibiting superior performance. These findings underscore the importance of high-quality datasets in facilitating effective HOM testing and provide valuable insights into the capabilities of machine learning algorithms in this context.**

Keywords: *High order mutation testing, dataset, data generation, machine learning.*

I. Introduction

Software testing plays a pivotal role in ensuring the reliability and robustness of modern software systems. Among the abundance of testing methodologies, mutation testing stands out as a rigorous technique for assessing the quality of test suites by injecting artificial faults, or mutants, into the source code. This method, dating back to the pioneering work of DeMillo et al. in the 1970s [1], has since evolved into a powerful tool for evaluating the effectiveness of test cases in detecting faults [2].

While traditional mutation testing typically focuses on introducing single faults into the codebase, recent advancements have led to the development of HOM testing techniques, capable of generating mutants with multiple simultaneous faults [3]. These techniques offer a more comprehensive assessment of test suite quality by exploring the interactions between faults and the ability of tests to detect them [3].

The availability of comprehensive datasets comprising diverse mutants and their associated test cases is a significant factor influencing the effectiveness of HOM testing. However, the success of HOM testing is not exclusively contingent upon data-driven methodologies; a plurality of algorithmic approaches contributes to the overall efficacy of HOM techniques [4]. Such datasets are essential for evaluating the effectiveness of HOM testing tools and algorithms, as well as for benchmarking their performance against existing techniques [4]. However, the advancement of HOM testing is hampered by a lack of suitable datasets.

In this paper, we address this need by presenting the results of our efforts to generate a novel dataset specifically tailored for HOM testing using the MutPy tool. Our dataset aims to provide researchers and practitioners with a valuable resource for advancing the state-of-the-art in mutation testing. Furthermore, this dataset comprises mutants generated using over 20 mutation operators applied to a representative set of software programs selected from diverse domains based on code complexity, programming language diversity, and domain representation. This ensures a broad range of code characteristics.

Our approach to dataset generation follows a systematic and rigorous methodology, leveraging the MutPy mutation testing framework. We carefully selected a representative set of software programs from various domains and applied a comprehensive set of mutation operators to create a

diverse population of mutants. Additionally, we curated a collection of test suites designed to exercise different aspects of the software under test, ensuring thorough coverage of its functionality and behavior.

The selection of software programs and their associated test suites prioritized three key criteria to ensure a representative dataset. First, popularity within the software community was considered to focus on widely used and relevant applications. Second, the reliability of each program was assessed based on its GitHub star rating, indicating community validation and maturity. Finally, the availability of comprehensive test suites was essential to adequately exercise diverse aspects of the software under test, allowing for thorough mutation analysis. This multi-faceted approach aimed to create a robust and representative dataset for evaluating mutation testing techniques.

Throughout this paper, we provide detailed insights into our methodology for dataset generation, as well as a comprehensive description of the dataset itself. We evaluate the quality and usefulness of the dataset through various experiments and comparisons with existing datasets, demonstrating its efficacy in facilitating HOM testing research and development.

In summary, our work contributes to the ongoing evolution of mutation testing techniques by providing a valuable resource that empowers researchers and practitioners to explore new frontiers in HOM testing. We believe that our dataset will serve as a catalyst for innovation and collaboration in the field of software testing, ultimately leading to the development of more robust and reliable software systems. The rest of the paper is structured as follows. The Related Work section reviews existing literature, recent advancements in HOM testing and current datasets. The Data Generation Process section details the methodology employed to create the dataset, including the use of MutPy and various mutation operators. Following this, the Dataset Evaluation section presents an in-depth analysis of the generated dataset. Finally, the Conclusion summarizes the findings, emphasizes the importance of the dataset for advancing mutation testing research, and suggests potential future work in this domain.

II. Related Work

Mutation testing has been a subject of extensive research, with numerous advancements made in recent years. In this section, we review the latest publications related to HOM testing and data generation using tools like MutPy.

Recent studies have focused on advancing mutation testing techniques to address the limitations of traditional mutation testing. For example, Dang et al. introduced a novel approach for HOM testing, leveraging advanced mutation operators and mutation strategies to generate mutants with multiple faults simultaneously [5]. Their study demonstrated the effectiveness of HOM testing in improving test suite quality and fault detection capability.

In the realm of data generation for mutation testing, the use of automated tools such as MutPy has gained significant attention. MutPy, a mutation testing tool for Python programs, has been widely adopted for generating mutation datasets due to its flexibility and extensibility [4]. Recent study by Dakhel et al. explored the capabilities of MutPy for generating diverse sets of mutants and corresponding test cases, showcasing its effectiveness in facilitating mutation testing research and development [6].

Furthermore, efforts have been made to enhance the mutation testing process by integrating advanced techniques for mutation generation and test case generation. For instance, the study by [7] et al. proposed a novel framework that combines HOM testing with evolutionary algorithms to optimize test case generation for improved fault detection.

Tushar Swamy et al. [8] proposed Homunculus, which is a compiler for efficient machine learning pipelines in network data planes. Addressing limitations of traditional network management, it simplifies machine learning model deployment by automatically generating optimized code from high-level specifications (using the Alchemy interface). Microbenchmarks show superior performance and resource efficiency compared to hand-tuned baselines. Future work focuses on co-compilation with traditional network applications and improving dataset quality and retraining speed. Le Van Phoi and Nguyen Thanh Binh [9] address the performance bottleneck of mutant generation in Lustre mutation testing. It proposes and evaluates a multi-threading approach for parallel generation of both first-order and higher-order mutants, demonstrating significant (nearly 50%) time improvements. Future work suggests exploring machine learning techniques for further optimization.

Additionally, research has been conducted on the evaluation and benchmarking of mutation testing techniques. Studies by [10] et al. and [11] et al. investigated the effectiveness of different mutation operators and mutation strategies in detecting faults and improving test suite quality [10, 11]. These studies provide valuable insights into the strengths and limitations of mutation testing approaches, informing the development of more robust mutation testing methodologies.

Our findings show that previous studies on mutation score prediction mainly rely on the source code of projects [12], or are combined with test suite metrics [13]. It can be said that up to now, there has not been a dataset that

includes mutants of projects. This is the direction of our research.

While these studies have made significant contributions to the field of mutation testing, there remains a need for further research on the generation of comprehensive datasets for HOM testing. Our work builds upon these efforts by presenting a novel dataset specifically tailored for HOM testing, generated using the MutPy tool. By incorporating advanced mutation operators and comprehensive test suites, our dataset aims to provide researchers and practitioners with a valuable resource for advancing the state-of-the-art in mutation testing.

III. Data Generation Process

1. HOM generation strategies background

Mutation testing is a well-established technique in software testing for evaluating the effectiveness of test suites. Traditional mutation testing involves the introduction of single faults, or mutations, into the source code to assess the ability of the test suite to detect these faults. However, as software systems become more complex, there is a growing need for mutation testing techniques capable of generating mutants with multiple simultaneous faults. This is where the HOM strategy of MutPy comes into play.

MutPy was chosen for dataset generation due to its suitability for our research goals. Its established position as a widely-used Python mutation testing tool, coupled with robust support for HOM testing—including the ability to generate mutants with multiple simultaneous faults—aligned perfectly with our need for a comprehensive and complex dataset. We leveraged MutPy's extensibility to implement a custom "ALL_PAIR" HOM strategy, further enhancing the dataset's scope. The tool's integration with scikit-learn provided essential machine learning capabilities for data analysis and model evaluation. MutPy's focus on Python, the language of our target codebase, ensured seamless integration and facilitated the reproducibility of our results.

Mathematically, let M denote the set of mutation operators and n represent the number of faults to be introduced into the codebase. The HOM strategy involves selecting n mutation operators from the set M and applying them simultaneously to create a mutant. This process can be represented by the combination formula:

$$C(|M|, n) = \frac{|M|!}{n! \times (|M| - n)!}. \quad (1)$$

This formula calculates the total number of distinct n-order mutants that can be generated. Each n-order mutant is created by applying n different mutation operators simultaneously from the set of M operators. By systematically combining mutation operators, MutPy's HOM strategy enables the generation of mutants with varying degrees of complexity, providing a more comprehensive assessment of test suite quality. This approach helps uncover subtle faults and vulnerabilities that may go undetected by traditional mutation testing techniques.

The HOM strategy of MutPy also incorporates advanced mutation operators tailored to target specific aspects of the codebase. These mutation operators cover a wide range of programming constructs and language features, ensuring thorough mutation testing coverage. Additionally, MutPy's HOM strategy allows for the integration of HOM testing with other mutation testing techniques, providing testers with the flexibility to customize their mutation testing approach based on project requirements and constraints. Overall, MutPy's HOM strategy represents a significant advancement in mutation testing methodologies, empowering testers to identify and address potential weaknesses in their test suites more effectively.

2. Dataset Generation Process

The Algorithm 1 defines the AllPairHOMStrategy class, which is responsible for generating second-order mutations by combining compatible first-order mutations. It iterates through the list of mutations, selects a pair of mutations, and yields a combination of these mutations to create a second-order mutant. This code should be integrated into the controller.py file of MutPy as part of the mutation testing process. Additionally, other code snippets are integrated into the MutPy source code to extract important mutation attributes and output them to a CSV file.

The dataset generation process for HOM testing using MutPy involves several steps, primarily utilizing MutPy for mutation extraction and coverage analysis.

Step 1: Utilizing MutPy for Mutation Extraction. In the first step, MutPy is employed to extract attributes of mutations. However, the built-in HOM strategies of MutPy typically generate a limited number of second-order mutants, approximately half the number of first-order mutations. To address this limitation, a custom HOM Strategy named ALL_PAIR is implemented to generate second-order mutants by combining compatible first-order mutants. This custom strategy significantly increases the number of second-order mutants that can be generated, providing a more comprehensive dataset for evaluation.

Step 2: Mutation Attribute Extraction. Following mutant generation, essential attributes of mutations are extracted to create a dataset. These attributes include:

- typeStatement: The type of statement mutated, extracted based on the mutated node or its parent nodes.

Algorithm 1 AllPairHOMStrategy class

```
import random
class AllPairHOMStrategy(HOMStrategy):
    name = 'ALL_PAIR'
    def __init__(self, *args, shuffler=random.shuffle, **kwargs):
        super().__init__(*args, **kwargs)
        self.shuffler = shuffler
    def generate(self, mutations):
        size_of_mutations = len(mutations)
for i in range(size_of_mutations) do
            first_mutations = []
            first_mutations.append(mutations[i])
            available_mutations = []
    for j in range(i + 1, size_of_mutations) do
                available_mutations.append(mutations[j])
    end for
            self.remove_bad_mutations(first_mutations, available_mutations)
    for mutation in available_mutations do
                mutations_to_apply = []
                mutations_to_apply.append(mutations[i])
                mutations_to_apply.append(mutation)
                yield mutations_to_apply
    end for
end for
```

- typeOperator: The mutation operator used, extracted from the mutation.operator field in the code.
- typeReturn: The return type of the mutated operator, obtained from the mutation.node.
- distanceBetweenTwoMutants: The distance between two mutated nodes in the abstract syntax tree (AST), calculated using the lowest common ancestor (LCA) method.
- line and end_line: The starting and ending lines of the mutation.
- depthOnAST: The depth of the mutant on the AST.
- result: The outcome of the mutation when tested, categorized as killed, survived, incompetent, or timeout.

Step 3: Code Coverage Analysis. Next, code coverage analysis is performed using the coverage tool. This involves running the test suite with coverage to determine which lines of code are covered. Additionally, each test case in the test suite is executed individually with coverage to assess its code coverage.

Step 4: Additional Attribute Generation. Based on the results of MutPy and coverage analysis, five additional attributes are generated:

- ancestor: Indicates whether two first-order mutants have a parent-child relationship.
- start_line_coverage: Determines if the starting line of the mutation is covered by the test suite.
- start_line_test_coverage: Calculates the percentage of tests covering the starting line of the mutation.
- body_coverage: Measures the overall code coverage of the mutated code body.
- body_test_coverage: Calculates the percentage of tests covering each line of the mutated code body.

By following these steps, a comprehensive dataset is generated for high-order mutation testing, facilitating the evaluation and improvement of test suites and mutation testing tools.

3. Dataset Description

Table I encapsulates the results of HOM testing conducted using the MutPy framework across various software modules. Each module is assessed based on several key metrics, including the number of survived mutants, killed mutants, timeouts, instances categorized as incompetent, and a specific metric denoted as Nummodulo.

A ”killed” mutant refers to a mutated version of the code that is successfully detected by the test cases. This means that the test cases, when executed against the mutated code, identify the introduced fault and fail, indicating that the test cases are effective in detecting this specific type of mutation. Conversely, a ”survived” mutant represents a

mutated version of the code that is not detected by the test cases. This implies that the test cases, when executed against the mutated code, do not identify the introduced fault and pass, indicating that the test cases are ineffective in detecting this specific type of mutation.

Survived mutants are those that remain undetected after the test cases have been executed. High numbers of survived mutants might indicate that the test cases are not thorough or effective enough to catch certain types of faults. This could be due to insufficient test coverage, focusing on specific code paths, or failing to address certain types of errors.

Opposite to Survived mutants, Killed mutants are the mutants that are detected by the test cases. A high number of killed mutants indicates that the test cases are effective in finding faults, thus demonstrating good coverage and robustness. This suggests that the test cases are capable of identifying a wide range of potential errors, leading to a higher confidence in the software's quality.

The classification of mutants into "killed" and "survived" categories is crucial for evaluating the effectiveness of test cases in detecting software faults [14]. The mutation score, often calculated as the percentage of killed mutants, provides a quantitative measure of test case effectiveness, guiding researchers and developers in improving the quality and coverage of their test cases.

Timeouts are instances where test cases take too long to execute and exceed a predefined time limit. This indicates performance issues or infinite loops within the software. Incompetent mutants are the mutants that are invalid or cause the program to crash immediately, suggesting that the mutation operators might need refinement to produce more realistic and useful mutations. Nummodulo is a metric representing the number or type of mutation operations applied during the testing process. This reflects the diversity and range of mutations tested, which is crucial for comprehensive coverage.

The *AirBnB_clone* module exhibits a relatively low count of 6 survived mutants, suggesting a limited capacity to withstand alterations, juxtaposed with a higher count of 30 killed mutants, indicating instances where the software failed to meet expected outcomes. Notably, this module demonstrates no occurrences of timeouts or incompetence.

Conversely, the *asynctest* module demonstrates robustness with a substantial count of 79,978 survived mutants, although it encounters occasional timeouts, as evidenced by 253 occurrences. Additionally, a notable number of instances, 6,939, are categorized as incompetent, indicating areas where the software may require optimization or refinement.

The *avocado* module showcases impressive resilience with over 1.2 million survived mutations, yet it is plagued by a significant count of timeouts and instances of incompetence, pointing towards potential areas for improvement.

Similarly, the *connect_four* module exhibits a blend of successes and challenges, with notable counts in both survived and killed mutants, alongside occurrences of timeouts and incompetence.

The *cpython3.7* module stands out with an extensive dataset, indicating substantial testing efforts, with millions of survived and killed mutants, alongside occurrences of timeouts and incompetence.

Overall, the dataset[1] provides a comprehensive snapshot of the performance and robustness of each software module under the rigorous scrutiny of HOM testing, offering insights into areas of strength and opportunities for enhancement in software quality and reliability.

IV. Dataset Evaluation

1. Evaluation metrics

In this study, we employed several evaluation metrics to assess the performance of the machine learning algorithms in classifying instances within the HOM testing dataset. These metrics provide valuable insights into the algorithms' effectiveness in correctly classifying instances across different categories.

Accuracy: Accuracy measures the proportion of correctly classified instances among all instances in the dataset. It provides an overall assessment of the algorithm's classification performance and is calculated as the ratio of the number of correctly classified instances to the total number of instances.

Precision: Precision measures the proportion of correctly identified instances among those predicted to belong to a particular category. It focuses on the algorithm's ability to avoid false positives and is calculated as the ratio of the number of true positive instances to the total number of instances predicted to belong to the positive category.

Recall (Sensitivity): Recall, also known as sensitivity, quantifies the proportion of correctly identified instances among all instances that actually belong to a particular category. It assesses the algorithm's ability to identify all positive instances and is calculated as the ratio of the number of true positive instances to the total number of instances belonging to the positive category.

F1-score: The F1-score is the harmonic mean of precision and recall, offering a balanced measure of a classifier's performance. It takes into account both false positives and

[1] https://rb.gy/1928xn

Table I
MUTANTS GENERATED BY MUTPY

Module	Survived	Killed	Timeout	Incompetent	Nummodulo
AirBnB_clone	6	30	0	0	4
asynctest	79978	0	253	6939	2
avocado	1287486	26426	15448	219190	15
callee	187537	0	0	36335	9
connect_four	12452	89592	8662	5080	4
cpython3.7	13373132	12227565	3118778	2792673	73
cricri	682	9067	4796	1386	3
green	309015	178806	19082	12920	8
nose2	36722	0	0	4784	4
RinnBlackJackPro	951269	20177	5536	89989	1
virtue	198693	0	8	19551	3

false negatives and is calculated as the harmonic mean of precision and recall. A higher F1-score indicates better overall performance.

2. Predictive machine learning models

To rigorously assess the suitability of the newly generated dataset for machine learning-based HOM testing, we employed four machine learning classification algorithms and evaluated their performance using evaluation metrics in section 4.1. These metrics provide valuable insights into the algorithms effectiveness in correctly classifying instances, while the successful classification across diverse algorithms provides strong empirical evidence of the dataset's internal consistency, absence of significant bias, and overall structural integrity—essential characteristics for a reliable HOM testing resource. This multifaceted evaluation, therefore, directly demonstrates the dataset's utility and robustness for future research in data-driven HOM testing methodologies.

Logistic Regression: Logistic Regression is a classic linear model widely used for binary classification tasks [15]. Despite its simplicity, Logistic Regression provides interpretable results and is computationally efficient, making it a popular choice for baseline classification tasks.

Random Forest Classifier: Random Forest Classifier is an ensemble learning method that constructs multiple decision trees during training and combines their predictions through voting or averaging [16]. It excels in handling high-dimensional data and is robust against overfitting, thanks to its built-in mechanism of feature randomness and bagging.

LightGBM: LightGBM is a gradient boosting framework developed by Microsoft that uses a novel technique called Gradient-Based One-Side Sampling (GOSS) to handle large-scale datasets efficiently [17]. It partitions the data in a depth-wise manner and can handle categorical features directly, making it highly efficient and accurate for classification tasks. Additionally, LightGBM supports GPU acceleration, enabling faster training times and scalability to larger datasets.

XGBoost: XGBoost, or Extreme Gradient Boosting, is another popular gradient boosting framework known for its scalability, speed, and performance [18]. It employs a regularized objective function and parallel tree boosting to achieve high accuracy and generalization on various datasets. XGBoost's advanced regularization techniques and parallel computing capabilities make it suitable for large-scale classification tasks with complex data structures.

3. Experimental Results

Table II shows the experimental results, provides a granular analysis of the performance metrics of four distinct machine learning algorithms—Logistic Regression, Random Forest Classifier, LightGBM, and XGBoost—across two pivotal categories: "Survived" and "Killed". These categories represent the outcomes of a binary classification task, where "Survived" denotes instances correctly identified as surviving and "Killed" represents instances correctly identified as not surviving.

In the "Survived" category, there are 1,633,032 instances for Logistic Regression, 1,633,844 instances for Random Forest Classifier, 1,634,353 instances for LightGBM, and 1,633,032 instances for XGBoost. Similarly, in the "Killed" category, there are 1,205,089 instances for Logistic Regression, 1,205,901 instances for Random Forest Classifier, 1,204,580 instances for LightGBM, and 1,205,901 instances for XGBoost.

Across the algorithms, we observe varying levels of accuracy, precision, recall, and F1-score, which offer nuanced insights into their classification capabilities. Logistic Regression achieves an accuracy of 0.81. In comparison, Random Forest Classifier achieves slightly higher accuracy scores of 0.85, and moving to more advanced algorithms, LightGBM and XGBoost exhibit even higher accuracy levels, with LightGBM achieving 0.90 and XGBoost achieving 0.91.

Analyzing precision, recall, and F1-score metrics provides further insights into algorithm performance. Light-

Table II
EXPERIMENTAL RESULTS

Algorithm	Accuracy	Kind of SOM	Amount	Precision	Recall	F1-score
Logistic Regression	0.81	Survived	1,633,844	0.27	0.79	0.54
		Killed	1,205,089	0.27	0.70	0.50
Random Forest Classifier	0.85	Survived	1,633,032	0.23	0.85	0.71
		Killed	1,205,901	0.23	0.78	0.67
LightGBM	0.90	Survived	1,634,353	0.90	0.94	0.92
		Killed	1,204,580	0.91	0.85	0.88
XGBoost	0.91	Survived	1,633,032	0.91	0.94	0.93
		Killed	1,205,901	0.92	0.88	0.90

GBM demonstrates remarkable precision values of 0.90 and 0.91 for ”Survived” and ”Killed” categories, respectively, indicating a high proportion of correctly classified instances among those predicted to belong to a particular category. Similarly, XGBoost exhibits impressive recall values of 0.94 and 0.88 for ”Survived” and ”Killed” categories, respectively, showcasing its ability to correctly identify a substantial proportion of instances belonging to each category.

By employing these established classification algorithms, the suitability of the generated dataset for machine learning applications is evaluated. The results provide insights into key dataset characteristics, including size and completeness (a large dataset with minimal missing values is desired), data quality (the absence of errors, inconsistencies, and bias), and transparency (a detailed description of the dataset creation process, encompassing selection criteria, data collection methods, and preprocessing steps). These findings inform the development of robust criteria for evaluating the quality of datasets for mutation testing research.

V. CONCLUSION

The dataset presented herein contributes significantly to the advancement of research in the field of HOM testing. By meticulously generating a comprehensive dataset using MutPy and coverage analysis, this research provides a valuable resource for researchers and practitioners interested in evaluating and enhancing the effectiveness of mutation testing techniques.

One of the primary contributions of this dataset lies in its focus on HOM testing, a cutting-edge approach that introduces mutants with multiple simultaneous faults. Traditional mutation testing techniques often overlook complex faults that can only be detected through HOM testing. By generating mutants with varying degrees of complexity, this dataset enables researchers to explore the effectiveness of HOM testing in uncovering subtle faults and vulnerabilities in software systems.

Moreover, the dataset encompasses a diverse set of mutation operators and strategies, including custom strategies like the AllPairHOMStrategy, designed to overcome limitations in existing mutation testing tools. This diversity allows researchers to evaluate the performance of different mutation operators and strategies and identify the most effective approaches for mutation testing.

This dataset provides a valuable resource for researchers in the Software Engineering (SE) community to advance their understanding and development of HOM testing techniques. Researchers can utilize the dataset to evaluate the effectiveness of existing HOM testing tools and algorithms, benchmark their performance against existing techniques, and explore new approaches for generating and analyzing mutants. The dataset's rich collection of mutants, categorized as ”Survived” and ”Killed,” allows researchers to investigate the impact of various mutation operators and strategies on test suite quality. Furthermore, the dataset can be used to train and evaluate machine learning models for predicting mutation outcomes, thereby improving the efficiency and effectiveness of HOM testing.

Overall, this dataset serves as a valuable resource for advancing the state-of-the-art in mutation testing and software quality assurance. By providing researchers with access to a diverse and meticulously curated dataset, this research aims to foster collaboration, innovation, and knowledge sharing in the field of HOM testing and beyond.

While the dataset presented here offers valuable insights into HOM testing and data generation, it is essential to acknowledge its limitations and identify areas for future research.

One limitation of the dataset is its reliance on MutPy for mutation generation and coverage analysis. While MutPy is a versatile and widely used tool, it may have inherent limitations in its mutation operators and strategies, which could impact the diversity and effectiveness of the generated mutants. Future research could explore alternative mutation testing tools or develop custom mutation operators to enhance the diversity and quality of mutants generated.

In terms of future work, we will explore novel mutation operators and strategies tailored to specific application domains or programming paradigms and focus on developing automated techniques for mutation testing, such as machine

learning-based approaches for mutation generation and test case prioritization. Additionally, enhancing the dataset's scope and documentation to improve its generalizability for broader mutation testing research should be focused. This includes expanding the dataset to encompass a wider variety of software projects and programming paradigms, providing a more detailed analysis of the dataset's statistical properties, and thoroughly documenting all data collection, preprocessing, and selection criteria. Furthermore, we plan to investigate the application of the dataset to different mutation testing techniques and explore the development of more robust evaluation metrics beyond the mutation score.

REFERENCES

[1] R. A. DeMillo, R. J. Lipton, and F. G. Sayward, "Hints on test data selection: Help for the practicing programmer," *Computer*, vol. 11, no. 4, pp. 34–41, 1978.

[2] Y. Jia and M. Harman, "An analysis and survey of the development of mutation testing," *IEEE transactions on software engineering*, vol. 37, no. 5, pp. 649–678, 2010.

[3] G. Fraser and A. Zeller, "Mutation-driven generation of unit tests and oracles," in *Proceedings of the 19th international symposium on Software testing and analysis*, 2010, pp. 147–158.

[4] M. Papadakis, M. Kintis, J. Zhang, Y. Jia, Y. Le Traon, and M. Harman, "Mutation testing advances: an analysis and survey," in *Advances in computers*. Elsevier, 2019, vol. 112, pp. 275–378.

[5] X. Dang, D. Gong, X. Yao, T. Tian, and H. Liu, "Enhancement of mutation testing via fuzzy clustering and multi-population genetic algorithm," *IEEE Transactions on Software Engineering*, vol. 48, no. 6, pp. 2141–2156, 2021.

[6] A. M. Dakhel, A. Nikanjam, V. Majdinasab, F. Khomh, and M. C. Desmarais, "Effective test generation using pre-trained large language models and mutation testing," *Information and Software Technology*, p. 107468, 2024.

[7] Y. Li, W. Shen, T. Wu, L. Chen, D. Wu, Y. Zhou, and B. Xu, "How higher order mutant testing performs for deep learning models: A fine-grained evaluation of test effectiveness and efficiency improved from second-order mutant-classification tuples," *Information and Software Technology*, vol. 150, p. 106954, 2022.

[8] T. Swamy, A. Zulfiqar, L. Nardi, M. Shahbaz, and K. Olukotun, "Homunculus: Auto-generating efficient data-plane ml pipelines for datacenter networks," in *Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3*, 2023, pp. 329–342.

[9] N. T. Binh *et al.*, "Optimizing mutant generation for lustre programs with multi-threading," in *2020 5th international conference on innovative technologies in intelligent systems and industrial applications (CITISIA)*. IEEE, 2020, pp. 1–5.

[10] J. H. Andrews, L. C. Briand, Y. Labiche, and A. S. Namin, "Using mutation analysis for assessing and comparing testing coverage criteria," *IEEE Transactions on Software Engineering*, vol. 32, no. 8, pp. 608–624, 2006.

[11] Q. Zhu, A. Panichella, and A. Zaidman, "A systematic literature review of how mutation testing supports quality assurance processes," *Software Testing, Verification and Reliability*, vol. 28, no. 6, p. e1675, 2018.

[12] Y. Gil and D. Ma'ayan, "Better prediction of mutation score," *Authorea Preprints*, 2023.

[13] K. Jalbert and J. S. Bradbury, "Predicting mutation score using source code and test suite metrics," in *2012 First International Workshop on Realizing AI Synergies in Software Engineering (RAISE)*. IEEE, 2012, pp. 42–46.

[14] Y. Jia and M. Harman, "Higher order mutation testing," *Information and Software Technology*, vol. 51, no. 10, pp. 1379–1393, 2009.

[15] J. H. Friedman, "Greedy function approximation: a gradient boosting machine," *Annals of statistics*, pp. 1189–1232, 2001.

[16] L. Breiman, "Random forests," *Machine learning*, vol. 45, pp. 5–32, 2001.

[17] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, "Lightgbm: A highly efficient gradient boosting decision tree," *Advances in neural information processing systems*, vol. 30, 2017.

[18] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, 2016, pp. 785–794.

Van-Nho Do heads the Informatics group at Le Quy Don Gifted High School, Da Nang, Vietnam. He is a doctoral candidate in Computer Science at Da Nang University of Technology, with a research specialization in advanced mutation testing. His research interests include software engineering, which he applies to his leadership role in education.

Email: dovannho@gmail.com

Giang T.C. Tran is a postdoctoral researcher at the University of Quebec in Trois-Rivières, specializing in the application of artificial intelligence to improve and enhance information systems. She completed her Ph.D. in Computer Science and Engineering at Chung-Ang University (2023) and holds a B.S. (with honors) in Management Information Systems from Da Nang University of Economics (2019). Her research utilizes data mining, machine learning, and logical reasoning techniques.

Email: thi.chau.giang.tran@uqtr.ca

Duc-Thuan Nguyen is a fourth-year student majoring in Information Technology at the University of Engineering and Technology, Vietnam National University, Hanoi.
Email: thuanbn03@gmail.com

Ngoc-Anh Nguyen Thi is a fourth-year student majoring in Information Technology at the University of Engineering and Technology, Vietnam National University, Hanoi.
Email: hannibalvera1412@gmail.com

Quang-Vu Nguyen is PhD in field of Computer Science from Wroclaw University of Science and Technology, Poland and currently is Head of the Department of Science – Technology and International Cooperation at Vietnam-Korea University of Information and Communication Technology, the University of Danang. His research focus is on artificial intelligence, software engineering, data science, software quality assurance, and testing.
Email: nqvu@vku.udn.vn

Thanh-Binh Nguyen graduated in Information Technology from the University of Danang - University of Science and Technology in 1997. He received PhD. degree in Information Technology at Grenoble Institute of Technology, France in 2004. He has been qualified as Associate Professor since 2013. He is currently working at the University of Danang - Vietnam-Korea University of Information and Communication Technology. His research interests include software engineering and software quality.
Email: ntbinh@vku.udn.vn

Predicting Stock Market Trends in Vietnam Using SVM, XGBoost, and LSTM Architectures

Truong Dinh Tu[1], Bhagawan Nath[2]

[1]Faculty of Information Technology, Ton Duc Thang University, Ho Chi Minh City, Vietnam,
[2]Department of Information Technology, Greenwich University, Hanoi, Vietnam.

Correspondence: Truong Dinh Tu, truongdinhtu@tdtu.edu.vn
Communication: received 9 Oct 2024, revised 12 Dec 2024, accepted 16 Jan 2025
DOI: 10.32913/mic-ict-research.v2025.n1.1356

Abstract*: Predicting the behavior of a stock market with higher accuracy has been a critical task, however, application of various cutting-edge Machine Learning (ML) and Deep Learning (DL) architectures have shown better results compared to that of classical statistical models. This study explores the Vietnamese stock market with the aim of predicting closing price movements. Various machine learning models, including Support Vector Machines (SVM), Extreme Gradient Boosting (XGBoost), and Long Short-Term Memory (LSTM) networks, are employed for the analysis. The data for this research has been gathered from various companies listed on the Ho Chi Minh City Stock Exchange (HOSE) and Hanoi Stock Exchange (HNINDEX) in Vietnam. These datasets were systematically preprocessed and thoroughly analyzed to calculate performance metrics, including the Root Mean Squared Error (RMSE) for assessing prediction quality and Mean Absolute Percentage Error (MAPE) to gauge forecast accuracy. In this study, technical features such as Relative Strength Index (RSI), Moving Average Convergence Divergence (MACD), Stochastic Oscillator (SO), among others, are employed. Additionally, Simple Moving Averages (SMA) and Weighted Moving Averages (WMA) are considered for various time frames. The significance of these features is systematically evaluated and determined. The experimental results indicate that the LSTM model outperforms both SVM and XGBoost in terms of prediction accuracy, as evidenced by its lower RMSE and MAPE values, as well as higher accuracy and F1-scores. However, the computational time of LSTM is significantly greater compared to that of SVM and XGBoost.

Keywords: *Support vector machines, extreme gradient boosting, long short-term memory, relative strength index, stochastic oscillator, moving average convergence divergence.*

I. Introduction

Vietnam is classified as a lower middle-income country, characterized by a multi-sectoral market economy where the state sector plays a central role in guiding economic development, ultimately with the overarching aim of advancing socialism. As of 2022, Vietnam ranks as the 38th-largest economy globally in terms of nominal Gross Domestic Product (GDP) and the 26th-largest when measured by Purchasing Power Parity (PPP) [1]. The VNINDEX serves as a capitalization-weighted index encompassing all companies listed on both the HOSE and HNINDEX. Established on July 28, 2000, with a base index value of 100, this index provides a comprehensive overview of the stock market's performance. Presently, the Ho Chi Minh City Stock Exchange boasts 404 listed companies [2], while the Hanoi Stock Exchange has 341 listed companies [3].

The stock market functions as a platform where traders and investors engage in buying and selling shares of publicly traded companies. It holds a pivotal role in shaping the economic landscape of a country. The prediction of stock market behavior involves forecasting the future values of stocks of various companies and other financial instruments traded on an exchange. The primary goal of such predictions is to mitigate risks and maximize profits [4]. A stock market is mostly characterized by a combination of nonlinear behaviors such as demand-supply, fundamental components, government policies, political situations, inflation, economy etc., which are highly volatile and complex. From economic, political and investment decisions to unknown causes, in some respects, there are many difficulties in predicting the stock market [5]. Thus, although stock markets attract a lot of investors due to its high profitability, it also brings along a comparable size of risk factors.

Stock markets are known for their high volatility and unpredictability, making traditional statistical methods like Simple Moving Average, Weighted Moving Average, and Exponential Smoothing only moderately effective. The inherent linearity of these statistical approaches can result

in suboptimal prediction performances, particularly when faced with abrupt fluctuations in stock prices, whether they experience a sudden rise or fall. Thus, sophisticated state-of-the-art approaches are required to deal with it more robustly [6, 7]. Recent advancement and application of technologies such as ML, DL, big data, cloud technologies etc. has helped significantly in developing higher performing algorithms in various sectors and it has largely drawn the attention of industries and academia as well [8]. Various ML and DL based algorithms such as Decision Trees, Logistic Regression, Support Vector Machines, Extended Gradient Boosting etc. have been applied in predicting stock market behaviors in research [9]. However, the Long Short-Term Memory (LSTM) network algorithms have been predominantly used to classify sequential data because of their ability to learn long-term dependencies between time steps of data and higher accuracy rates. The LSTM network, classified as a type of realizable recurrent neural network model, is equipped with selective memory and intra-temporal influence. This particular architecture proves highly suitable for analyzing stochastic non-stationary time series data, such as stock prices [10].

This paper uses LSTM, SVM and XGBoost network architectures to predict stock market behavior, especially the closing price prediction and this study is specific to Vietnam stock market. Datasets are collected from different sources, five company datasets (VRE, SSI, VCB, STB, and VNM) are extracted from the VN30 dataset. Three more datasets are collected from the Ho Chi Minh Stock Exchange (HOSE), HNINDEX and a combination of both, known as Vietnam Stock Exchange (VNINDEX) respectively. The time span of the datasets is mostly from either from 2006 or 2009, and until 2021. The datasets are preprocessed and certain technical features such as MACD, RSI, SO etc. are computed for a better understanding of the problem and to check their impacts in the analysis. Two moving averages (SMA and EMA) are calculated for different time intervals such as weekly, monthly, quarterly etc.

Finally, analysis is done using the algorithms to determine forecasting of the stock market. Several performance metrics, including MAPE, RMSE, accuracy score, F1-score, and computational time, are employed to assess the efficiency of the models. MAPE, widely utilized for assessing forecast accuracy, falls under percentage errors, making it scale-independent and suitable for comparing series on different scales.

The second metric considered is RMSE, a commonly used measure for evaluating prediction quality. It quantifies how far predictions deviate from measured true values, utilizing Euclidean distance. The accuracy score measures the models' performance by calculating the ratio of the sum of True Positives (TP) and True Negatives (TN) to all predictions made. F1-score, another crucial metric, gauges a model's accuracy by combining precision and recall scores, reflecting the frequency of correct predictions across an entire dataset. Accuracy is preferred when True Positives (TP) and True Negatives (TN) are of greater importance, whereas F1-score is prioritized when False Negatives (FN) and False Positives (FP) carry more significance.

The last metric, computational time, is used to determine the swiftness of the algorithms, which is equally important looking at the current scenario. Another crucial outcome of this paper is the determination of the importance of the technical features, that is done using the XGBoost architecture and presented in the paper.

The contributions of this study are:
- The study applies advanced machine learning (SVM, XGBoost) and deep learning (LSTM) architectures to predict stock market behavior in Vietnam, focusing on closing price prediction across multiple datasets, including company-specific and stock exchange datasets (HOSE, HNINDEX, VNINDEX).
- The study introduces a comprehensive analysis of technical indicators (e.g., MACD, RSI, SMA, EMA) and evaluates their impact on stock market forecasting for improved prediction accuracy.
- Experimental results demonstrate that LSTM outperforms SVM and XGBoost in prediction accuracy (e.g., RMSE, MAPE, F1-score), although it requires significantly higher computational time.
- The study focuses on the Vietnamese stock market, offering insights into an emerging market that has been underexplored in financial forecasting research. It encourages future research to develop more effective prediction models, offering valuable insights to investors, analysts, and other stakeholders in stock market forecasting.

The rest of the paper is organized as follows: Section II briefly describes the technical background, Section III discusses the related work done in the same domain, Section IV explains the methodology, Section V explores the results, and Section VI includes the conclusion and future work.

II. Technical Background

The ML and DL based algorithms and other technological components used in this investigation are briefly explained below.

1. Support Vector Machines (SVM)

SVMs are supervised ML algorithms mostly used for classification, regression, and outlier detection problems.

The basic idea of SVM is to create a hyperplane to separate the given data points into classes. The dimension of the hyperplane depends on the number of input features, for example, a set of 2-feature data points have a single line hyperplane, a 3-feature set will have a 2-D hyperplane and so on. A hyperplane is a decision boundary that helps to classify the data points. The data points that are closer to the hyperplane and influence the position and orientation of the hyperplane are known as the support vectors. These support vectors help in building the SVM with the basic idea of maximizing the margin between the data points and the hyperplane. SVMs can handle both classification and regression on linear and non-linear data, and are used in various applications such as malware detection, handwriting recognition, intrusion detection, face detection, email classification, gene classification etc.

2. Extended Gradient Boosting Algorithm (XGBoost)

XGBoost stands as a scalable and distributed Gradient-Boosted Decision Tree (GBDT) supervised machine learning technique, widely applied in tasks such as regression and classification. Its efficacy lies in its ability to handle large datasets and deliver robust performance across various domains. It is known as “extreme” because of its ability to do parallel computation on a single machine, which makes it more efficient than normal gradient boosting frameworks. XGBoost is an efficient implementation of gradient boosting algorithms, known for its scalability and high accuracy. It is specifically designed to boost the computing power of tree-based algorithms, primarily focused on enhancing both the performance of machine learning models and their computational speed [11]. XGBoost employs a parallel tree-building approach, distinguishing itself from GBDT, where tree construction is sequential. Its methodology adheres to a level-wise strategy, involving a thorough scan of gradient values. Utilizing these gradient values, XGBoost calculates partial sums to assess the quality of splits at each potential split point within the training set.

3. Long Short-Term Memory (LSTM)

LSTM, a deep learning-based recurrent neural network (RNN), excels in efficiently retaining long-term dependencies and is resilient against the vanishing gradient problem. To facilitate learning long-term dependencies, LSTMs leverage Gated Recurrent Units (GRUs). GRUs possess the capability to selectively forget information from the previous timestep and control the amount of information from the current timestep to be passed on to the next. Moreover, LSTMs offer versatility in training methods, including backpropagation through time and reinforcement learning. This adaptability makes LSTMs particularly well-suited for tasks requiring the retention and interpretation of information from extended sequences [12]. LSTMs find prominent application in learning, processing, classifying, and predicting based on time series data, especially in scenarios where there are lags of unknown duration between critical events [13]. They demonstrate effectiveness in complex problem domains such as machine translation, language modeling, speech recognition, and various other applications.

4. Moving Average Convergence Divergence (MACD)

The Moving Average Convergence Divergence is a momentum oscillator mainly used to deal with trends. It is an indicator that shows the relationship between two exponential moving averages (EMA). We obtain MACD by subtracting the longterm EMA (slow EMA) from the shortterm EMA (fast EMA). It is one of the most popular tools in technical analysis because it gives traders the ability to identify the short-term trend directions quickly and easily [14]. MACD is calculated using the formula:

$$MACD = fastEMA - slowEMA \tag{1}$$

5. Relative Strength Index (RSI)

Relative Strength Index (RSI) indicates magnitude of recent price changes. It can show that a stock is either overbought or oversold. Typically, RSI value ranges between 0 and 100, where 70 and above signal that a stock is becoming overbought/overvalued, whereas a value of 30 and less can mean that it is oversold. RSI can be calculated as:

$$RSI = 100 - [100/(1 + (AUPC/ADPC))] \tag{2}$$

where: AUPC = Average of Upward Price Change, and ADPC = Average of Down-ward Price Change.

6. Stochastic Oscillator (SO)

The Stochastic Oscillator serves as a momentum indicator, revealing the position of the closing price concerning the high-low range within a specified number of periods. In an uptrend, the closing price typically concludes near the high, while in a downtrend, it tends to approach the low. When the closing price deviates from the high or low, it signals a slowdown in momentum. Stochastics demonstrate optimal effectiveness in broader trading ranges or during periods of gradual trend movements.

$$\%K = [(C - L14)/(H14 - L14)] \times 100 \tag{3}$$

where: %K = The current value of the stochastic indicator, C = The most recent closing price, L14 = The lowest price traded of the 14 previous trading sessions, H14 = The highest price traded during the same 14-day period.

7. Moving Average (MA)

Moving average is used to smoothen stock prices on a chart by filtering out shortterm price fluctuations. Moving averages are calculated over a defined period, such as the last 7, 15, 30 or 200 days. There are two (most common) averages used in technical analysis.

a) Simple Moving Average (SMA)

A simple moving average is a straightforward average computed over a specified period. This technical indicator is useful for assessing whether the price of an asset will sustain its current trend or undergo a reversal in a bull or bear market.

$$SMA = (A1 + A2 + ... + An)/n \tag{4}$$

where: An is the price of an asset at period n, n is the number of total periods.

b) Exponential Moving Average (EMA)

The Exponential Moving Average (EMA) is a type of average that assigns higher weights to recent prices. Widely recognized as one of the most effective indicators for scalping, the EMA responds more swiftly to recent price fluctuations compared to older price changes. This responsiveness makes it particularly valuable for capturing short-term market movements and trends in scalping strategies.

$$EMA = Price(t) \times k + EMA(y) \times (1 - k) \tag{5}$$

where: t = current day / today, y = previous day / yesterday, k = 2 / (N + 1), and N = number of days in EMA.

III. Related Work

Yun et. al (2023), introduced an algorithm that utilizes two distinct input feature sets: internal technical indicators and external market prices. This algorithm undergoes a two-stage feature selection process, which includes feature set expansion, a hybridized approach integrating genetic algorithm-machine learning regressions for the selection of crucial features, and importance score filtering to identify the optimal features. Their effort aimed to select the most effective feature subset with a parsimoniously reduced number of features, prioritizing interpretability. The results demonstrated a 10.42% improvement in average forecasting RMSE for the optimal feature set and a 13.47% improvement for the best feature subset of the internal technical indicators. In their study, the researchers employed Savitzky–Golay smoothing as an integral component of piecewise optimal curve fitting to analyze various potential groupings of external features. The introduced local interpretability technique, incorporating piecewise optimal curve fitting and piecewise best feature subset, offered a more adaptable interpretation of stock price behavior. This method utilized a select few best features for different segments of the piecewise data, providing timely and flexible insights. In contrast to other feature-importance interpretability techniques that rely on either a singular data point or an entire data period, the proposed interpretability technique successfully overcame these limitations. It accurately reflected the primary characteristics inherent in time series data [15]. Houssein et. al (2022) implemented a hybrid model that integrated the Support Vector Regression (SVR) method with Equilibrium Optimizer (EO) to predict the closing prices of the Egyptian stock exchange (EGX). The study focused on three indices: EGX 30, EGX 30 capped, and EGX 50 EWI. Evaluation of the model was conducted using metrics such as mean absolute percentage error, average, standard deviation, best fit, worst fit, and CPU time. Furthermore, the researchers compared their model with recently developed metaheuristic optimization algorithms from the literature, including the whale optimization algorithm, salp swarm algorithm, Harris Hawks optimization, grey wolf optimizer, Henry gas solubility optimization, Barnacle's mating optimizer, Manta ray foraging optimization, and slime mould algorithm. The results demonstrated that the proposed EO-SVR model outperformed its counterparts. Notably, the study concluded that there was no discernible need to incorporate technical indicators and statistical measures, as their impact was not significant [16].

Hanh, Le Hong et al. (2022), used about 69,950 articles from four Vietnam based reputable online newspapers as their input data, which were divided into 8 stock market trend related parts, and they were collected between 2001 and 2021. The data were analyzed using Machine Learning architectures such as decision trees, random forests, K-Nearest Neighbors (KNNs) and Support Vector Machines (SVMs). They found SVM as the best performing model and the best dataset being Vietstock, digging deeper and refining the findings helped them to raise the accuracy up to 60.1% [17].

Hung et al. (2021) conducted a study examining the repercussions of COVID-19 on the performance of the Vietnamese Stock Market. Their analysis was based on panel data encompassing stock returns of 733 listed companies on both the Ho Chi Minh Stock Exchange (HOSE) and the Hanoi Stock Exchange (HNX) during the period from

January 2, 2020, to December 13, 2020. Employing the Random-Effect Model (REM), the researchers observed that the impacts were more pronounced during the pre-lockdown and second-wave periods, in comparison to the impact during the lockdown period. Furthermore, the study revealed variations in the impacts across sectors, with the financial sector experiencing the most significant effects [18].

Nti et. al (2020) conducted a comprehensive comparative analysis of ensemble techniques, including boosting, bagging, blending, and super learners (stacking). The study utilized Decision Trees (DT), Support Vector Machine (SVM), and Neural Network (NN) to construct twenty-five different ensembled regressors and classifiers. The evaluation encompassed execution times, accuracy, and error metrics using stock data from the Ghana Stock Exchange (GSE), Johannesburg Stock Exchange (JSE), Bombay Stock Exchange (BSE-SENSEX), and New York Stock Exchange (NYSE) spanning from January 2012 to December 2018. The findings indicated that stacking and blending ensemble techniques exhibited higher prediction accuracies, ranging from 90% to 100% and 85.7% to 100%, respectively, in comparison to bagging (53% to 97.78%) and boosting (52.7% to 96.32%). Additionally, the study recorded the root mean squared error (RMSE) for stacking within the range of 0.0001 to 0.001, and for blending, it ranged from 0.002 to 0.01 [19].

Hiransha et. al (2018) In 2018, Hiransha and collaborators employed deep learning architectures, including Multilayer Perceptron (MLP), Recurrent Neural Networks (RNN), Long Short-Term Memory (LSTM), and Convolutional Neural Network (CNN), to analyze two distinct stock markets: the National Stock Exchange (NSE) of India and the New York Stock Exchange (NYSE). The models were trained using the stock prices of a single company from NSE and then applied to predict stock prices for five different companies from both NSE and NYSE. Comparing the results with the Autoregressive Integrated Moving Average (ARIMA) model, the study revealed that the neural networks, including MLP, RNN, LSTM, and CNN, consistently outperformed the existing linear model (ARIMA) in terms of predictive accuracy [20].

Phuoc, Tran, et. al (2024) [21] proposed a model to predict stock price trends in an emerging economy, utilizing the LSTM algorithm combined with technical indicators predict such as Simple Moving Average (SMA), Moving Average Convergence Divergence (MACD), and Relative Strength Index (RSI). Their study, based on VN-Index and VN-30 stocks, demonstrated a high prediction accuracy, highlighting the effectiveness of LSTM in analyzing and forecasting stock price movements. Unlike their work, our study employs a broader comparison by using multiple models (SVM, XGBoost, and LSTM) and evaluates multiple performance metrics, including RMSE, MAPE, F1-score, and computational time of each algorithm, for a comprehensive assessment of each algorithm. Additionally, our work incorporates an assessment of the importance of technical features on datasets, which was not emphasized in [21].

IV. Methodology

The investigation is done using three ML and DL based architectures - SVM, LGBoost and LSTM on the datasets - VRE, SSI, VCB, STB, VNM, HOSE, HNINDEX and VNINDEX to predict the stock market behavior of Vietnam. The datasets are pre- processed and various technical features such as MACD, SO, RSI, SMA and EMA are calculated after checking the suitability of different time frames. The following algorithm (Algorithm-1) is used to calculate the MACD.

Algorithm 1 Computation MACD

(1) Compute_MACD(dataset, slowEMA, fastEMA)
(2) slow ← compute EMA for slowEMA
(3) fast ← computione EMA for fastEMA
(4) df[‘MACD’] ← fastEMA - slowEMA
(5) df[‘MACD signal’] ← Moving Average of the MACD (line)
(6) end

The algorithm below (Algorithm-2) is used to calculate the SO.

Algorithm 2 SO Calculation

(1) Compute_SO(dataset, k value, d value)
(2) low min ← Minimum of df[‘low’] based on window size = k value
(3) high min ← Maximum of df[‘high’] based on window size = k value
(4) df[‘stock k’] ← 100 * (df[‘Close’] - low min) / (high max - low min)
(5) df[‘stock d’] ← Mean of df[‘stock k’] with window size = d
(6) end

RSI is used to determine the magnitude of recent price changes, which is calculated using the Algorithm-3, given below.

The EMA and SMA are calculated for different time spans, such as 7, 9, 10, 15, 30, and 90 days respectively. These technical features are added to the datasets and the datasets are normalized for the ease of analysis. The

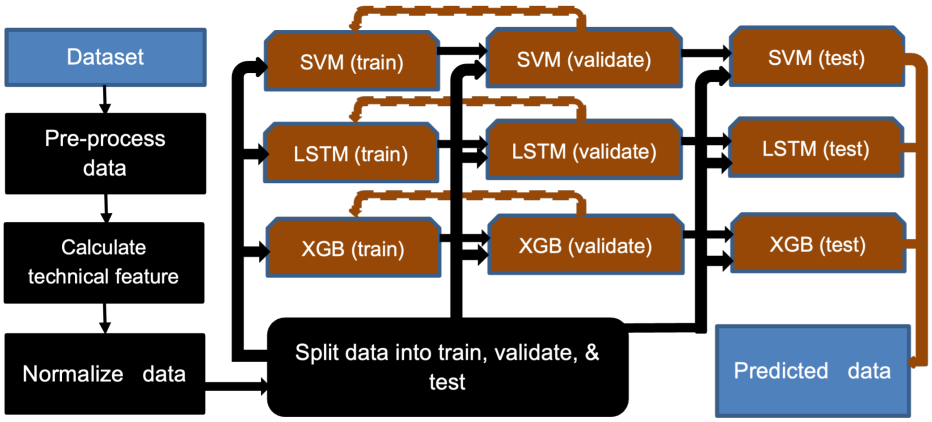


Figure 1. Flow diagram of the methodology.

Algorithm 3 RSI Calculation

(1) Compute_RSI(dataset, k value, d value)
(2) rollUp ← Average of Upward Price Change
(3) rollDown ← Average of Downward Price Change
(4) RS← rollUp / rollDown
(5) RSI←100-(100/(1+RS)
(6) end

datasets are split into three parts -train at 70%, test as 15% and validation with the rest 15%, and the architectures are then trained and validated. The algorithms are fine-tuned by resetting certain hyperparameters to achieve a better performance and finally they are tested. Various performance metrics such as RMSE, MAPE, accuracy score and F1-score are computed to see the effectiveness of the models on each dataset. The study has also considered the computational time as a final performance metric to justify the swiftness of the algorithms. The figure 1 represents the general methodology used in the study.

The study also investigated and determined the technical features that are crucial in the prediction of the stock market behavior. The RGBoost network architecture is specifically used to do that, and the results have been presented in the later section.

Table I
DATASET INFORMATION

Datasets	Start Date	End Date	Size
SSI	15/12/2009	25/06/2021	3606
VCB	30/06/2009	25/06/2021	2991
STB	02/07/2006	25/06/2021	3727
VNM	19/01/2006	25/06/2021	3842
VRE	28/07/2000	25/06/2021	5056
HOSE	15/07/2010	25/06/2021	65535
HNINDEX	24/05/2006	14/10/2019	3180
VNINDEX	07/08/2000	14/10/2019	4550

V. EXPERIMENTAL

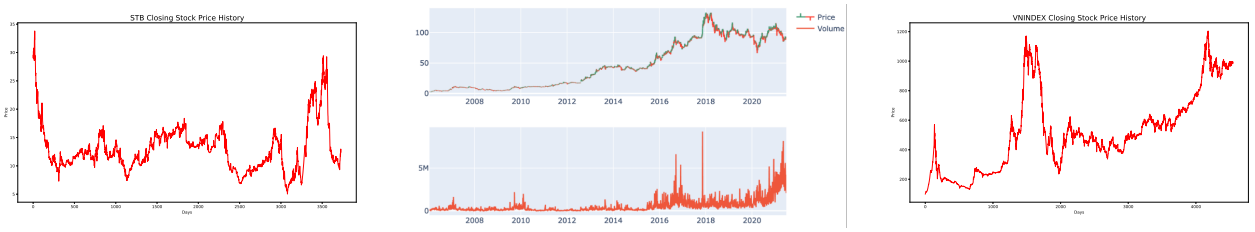
The results obtained by using SVM, RGBoost and LSTM networks to predict the stock market behavior of Vietnam have been divided into different sections and presented here. The first two sections represent raw data and different technical features that are used in the experiment. The third section represents the confusion matrices of the algorithms on different datasets, and the fourth section depicts the importance of various technical features used in the study. The fifth section represents the closing price prediction of the datasets, and the final section shows various performance metrics scores obtained by the algorithms for individual datasets and a summary of it.

1. Dataset

The objective of this study is specific to Vietnam, the datasets being used are collected from the two stock exchanges of HOSE and HNINDEX. A total of eight datasets are used in this study, five of them are from the HOSE based companies extracted from the VN30 dataset, the other two datasets are from the individual stock exchanges and the last one is the VNINDEX. HOSE, HNINDEX and VNINDEX are collected from different sources. The time span and number of records vary from dataset to dataset, however, most of them are from either 2006 or 2009, and until 2021. A Table 1 representation of the time and size of the datasets is given below.

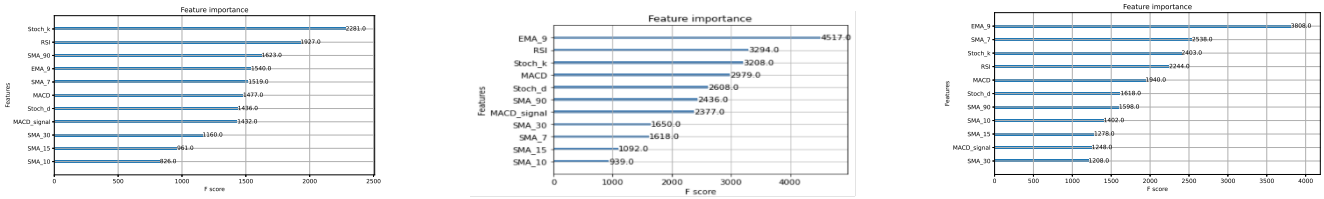
2. Raw data Visualization

The figure 2 (a) is the representation of the raw closing price plot of the STB dataset. The figure 2 (b) shows the raw OLHC (Open, High, Low, Close) graph of the VCB dataset. The figure 2(c) depicts the raw closing price data of VNINDEX.



(a) STB Closing Stock Price History (b) VCB dataset (c) The closing price data of VNINDEX

Figure 2. Raw data Visualization



(a) Features ranking in SSI dataset (b) Features ranking in VCB dataset (c) Technical features ranking in HtNINDEX dataset

Figure 3. The importance of technical features on SSI, VCB and HNINDEX datasets.

(a) VRE prediction graph using LSTM (b) VCB prediction graph using LSTM

Figure 4. The prediction of the stock market behavior using the LSTM model.

Figure 5. MACD of VNINDEX dataset.

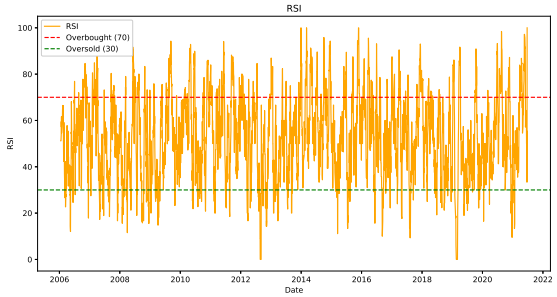


Figure 6. RSI of VNM dataset.

3. Technical features visualization

The visuals of different technical features used in this experiment are presented here. The trade trends of the VNINDEX as MACD graph is shown in figure 5. The Figure 6 represents the speed and change of price move-

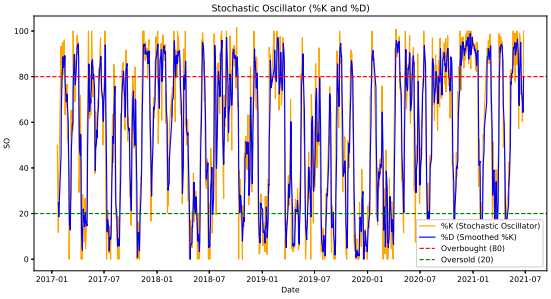


Figure 7. SO of SSI dataset.

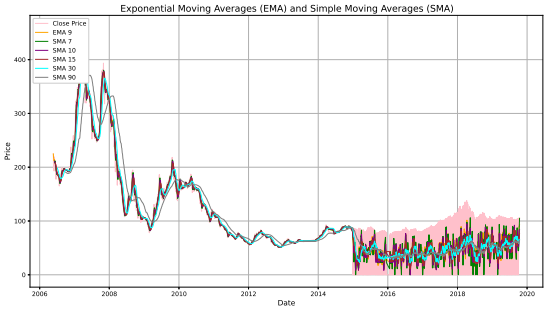


Figure 8. MA of HNINDEX dataset.

ments of the VNM dataset as an RSI graph. The Figure 7 shows the location of the closing price relative to the high-low range over a set number of periods as the stochastic oscillation of the SSI dataset. The moving averages (EMA and SMA) of HNINDEX dataset are presented in the Figure 8.

4. Confusion Matrices

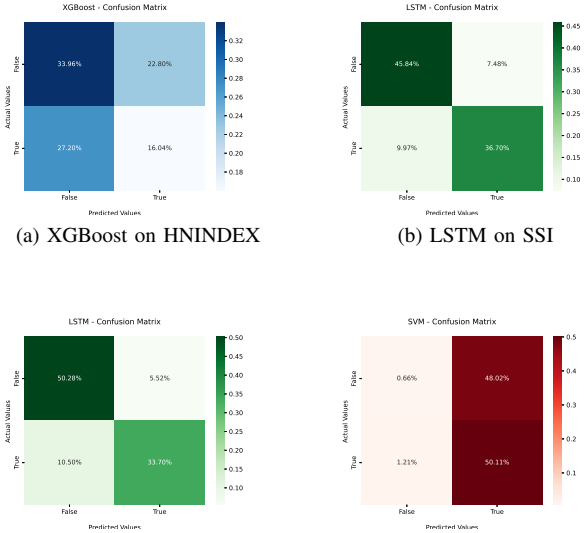
Confusion matrices of the algorithms are computed on each dataset and four of them are presented in their normalized form Figure 9.

5. Importance of technical features

The technical features such as MACD, SO, RSI, EMA, and SMA are used in the investigation and their importances are determined using the XGBoost algorithm. Figures 3(a), 3(b) and 3(c) represent the importance of those technical features on SSI, VCB and HNINDEX datasets.

6. Stock Market Prediction Graphs

Below are the diagrams showing the prediction of the stock market behavior using the LSTM model. Figure 4(a) shows the VRE closing stock price prediction and Figure 4(b) represents the closing stock price prediction of VCB.



(a) XGBoost on HNINDEX (b) LSTM on SSI

(c) LSTM on VRE (d) SVM on VNINDEX

Figure 9. Confusion matrices on each dataset

7. Performance metrics

The Table 2 shows the RMSE, MAPE and computational time of each algorithm on different datasets. Accuracy score and F1-score of the architectures are computed and represented in the Table 3. The Table 4 represents a summary of all four performance metrics scores by the algorithms.

VI. Conclusion

The experimental findings indicate that, across all performance metrics, LSTM outperforms the other two architectures. However, it's noteworthy that LSTM comes with a trade-off, as it consumes a significantly higher amount of computational time compared to the alternative architectures. XGBoost, on the other hand, has a lower F1-score than the SVM, but it performs better than the SVM in other performance metrics. It is also observed that the technical features such as exponential moving average (EMA9), relative strength index (RSI), simple moving averages (SMA7, SMA90), stochastic oscillator (stochK) etc. are crucial in determining the stock market behavior and may be used as strong indicators in the same domain. The findings of the investigation are completely specific to the datasets used in the study and it is, therefore, tricky to generalize them. However, this will encourage researchers to use novel approaches to develop more effective prediction methods, and thereby assisting investors, analysts and other stakeholders involved in the stock market by providing more definite and accurate information of the future stock market.

The proposed models can assist investors and financial analysts in making more informed trading decisions by

Table II
PERFORMANCE COMPARISON BETWEEN LSTM, SVM, AND XGB MODELS ON VARIOUS STOCKS.

Models	Metrics	SSI	VCB	STB	VNM	VRE	HOSE	HNINDEX	VNINDEX	Average
LSTM	RMSE	0.036	0.744	0.122	0.700	0.023	0.186	0.628	0.383	0.353
	MAPE	16.604	13.790	14.719	15.461	17.136	15.659	17.544	19.672	16.323
	TIME	69.799	18.832	36.368	76.301	47.920	567.473	21.220	21.681	107.449
SVM	RMSE	0.422	0.134	0.831	0.247	0.271	0.370	0.432	0.468	0.397
	MAPE	19.541	17.390	13.761	12.090	19.325	23.211	19.932	16.721	17.746
	TIME	6.249	5.659	8.879	7.280	3.580	368.566	9.564	4.307	51.761
XGB	RMSE	0.273	0.830	0.670	0.026	0.057	0.317	0.676	0.081	0.366
	MAPE	17.755	15.811	15.904	13.641	20.954	20.871	18.431	15.596	17.370
	TIME	9.230	8.769	11.172	9.114	1.141	17.771	31.911	2.240	11.419

Table III
ACCURACY SCORE AND F1 SCORE

Datasets	Metrics	SVM	LSTM	XGBoost
SSI	Accuracy Score	0.822	0.832	0.805
	F1-score	0.802	0.808	0.799
VCB	Accuracy Score	0.806	0.808	0.793
	F1-score	0.714	0.704	0.713
STB	Accuracy Score	0.823	0.840	0.879
	F1-score	0.782	0.788	0.784
VNM	Accuracy Score	0.799	0.868	0.860
	F1-score	0.809	0.858	0.797
VRE	Accuracy Score	0.707	0.735	0.851
	F1-score	0.496	0.525	0.730
HNINDEX	Accuracy Score	0.460	0.761	0.475
	F1-score	0.427	0.658	0.459
VNINDEX	Accuracy Score	0.679	0.695	0.475
	F1-score	0.671	0.681	0.454
HOSE	Accuracy Score	0.348	0.695	0.393
	F1-score	0.322	0.691	0.388

Table IV
SUMMARY OF PERFORMANCE METRICS

Metrics	SVM	LSTM	XGBoost
Accuracy Score	0.605	0.679	0.629
F1 Score	0.662	0.732	0.597
RMSE	0.397	0.353	0.366
MAPE	17.746	16.323	17.370
Compu. Time	51.761	107.449	11.419

providing insights into stock price trends. The integration of these predictive models into automated trading systems or financial advisory tools can help optimize portfolio management and risk mitigation strategies.

Additionally, our analysis of technical indicators can aid financial professionals in refining their feature selection for stock forecasting in similar emerging markets.

REFERENCES

[1] Economy of Vietnam. In *Wikipedia*, retrieved from https://en.wikipedia.org/wiki/Economy_of_Vietnam

[2] Hanoi Stock Exchange. "In Sustainable Stock Exchanges (SSE) initiative", available at: https://sseinitiative.org/stock-exchange/hnx/

[3] Ho Chi Minh Stock Exchange, "In Sustainable Stock Exchanges (SSE) initiative," available at: https://sseinitiative.org/stock-exchange/hsx/

[4] Biswas, M., Shome, A., Islam, M. A., Nova, A. J., & Ahmed, S. ", Predicting stock market price: A logical strategy using deep learning, " *In 2021 IEEE 11th IEEE Symposium on Computer Applications & Industrial Electronics (ISCAIE)* (pp. 218-223). IEEE. 2021. doi: 10.1109/ISCAIE51753.2021.9431817.

[5] Sharma, A., Bhuriya, D., & Singh, U., "Survey of stock market prediction using machine learning approach, " *In 2017 International Conference of Electronics, Communication and Aerospace Technology (ICECA)*, Vol. 2, pp. 506-509. IEEE, 2017. doi: 10.1109/ICECA.2017.8212715.

[6] Bhattacharjee, I., & Bhattacharja, P.,"Stock Price Prediction: A Comparative Study between Traditional Statistical Approach and Machine Learning Approach," *In 2019 4th International Conference on Electrical Information and Communication Technology (EICT)*, (pp. 1-6), 2019, IEEE. doi: 10.1109/EICT48899.2019.9068850.

[7] Koukaras, P., Nousi, C., & Tjortjis, C., "Stock Market Prediction Using Microblogging Sentiment Analysis and Machine Learning," *Telecom*, 3(2):358-378., 2022. https://doi.org/10.3390/telecom3020019

[8] Zhang, R., "LSTM-based Stock Prediction Modeling and Analysis," *Proceedings of the 2022 7th International Conference on Financial Innovation and Economic Development (ICFIED 2022)*, pp. 2537-2542, 2022. doi: 10.2991/aebmr.k.220307.414

[9] Obthong, M., Tantisantiwong, N., Jeamwatthanachai, W., & Wills, G., "A Survey on Machine Learning for Stock Price Prediction: Algorithms and Techniques," 2020. doi: 10.5220/0009340700630071.

[10] Khang, P. Q., Kaczmarczyk, K., Tutak, P., Golec, P., Kuziak, K., Depczyński, R., ... & Rot, "Machine learning for liquidity prediction on Vietnamese stock market, " *Procedia Computer Science*, 192, 3590-3597, 2021. doi: 10.1016/j.procs.2021.09.132

[11] Gumelar, A. B., Setyorini, H., Adi, D. P., Nilowardono, S., Widodo, A., Wibowo, A. T., ... & Christine, E., "Boosting the accuracy of stock market prediction using xgboost and long short-term memory, " *In 2020 International Seminar on Application for Technology of Information and Communication (iSemantic)*, (pp. 609-613), 2020. doi: 10.1109/iSemantic50169.2020.9234256.

[12] Nabipour, M., Nayyeri, P., Jabani, H., Mosavi, A., Salwana, E., & Shahab, S., "Deep Learning for Stock Market Prediction, " *Entropy* 22(8), 840, 2020. doi: 10.3390/e22080840

[13] Moghar, A., & Hamiche, M, "Stock Market Prediction Using LSTM Recurrent Neural Network, " *Procedia Computer Science*, 170, 1168-1173, 2020. doi: 10.1016/j.procs.2020.03.049

[14] Mokhtari, S., Yen, K. K., & Liu, J., "Effectiveness of

artificial intelligence in stock market prediction based on machine learning, ” arXiv preprint arXiv:2107.01031, 2021. https://doi.org/10.48550/arXiv.2107.01031

[15] Yun, K. K., Yoon, S. W., & Won, D., “Interpretable stock price forecasting model using genetic algorithm-machine learning regressions and best feature subset selection, ” *Expert Systems with Applications*, 213, 2023. doi: 10.1016/j.eswa.2022.118803

[16] Houssein, E.H., Dirar, M., Abualigah, L., ”An efficient equilibrium optimizer with support vector regression for stock market prediction, ” *Neural Comput & Applic*, 34, 3165–3200, 2022. doi: 10.1007/s00521-021-06580-9.

[17] Hanh, L. H., ”Stock Market Prediction: The Application of Text-Mining in Vietnam, ” *VNU Journal of Economics and Business*, 2(2), 2021. doi: 10.57110/jeb.v2i2.4715.

[18] Hung, D., Hue, N., & Duong, V., ”The Impact of COVID-19 on Stock Market Returns in Vietnam,” *Journal of Risk and Financial Management*, 2021.

[19] Nti, I.K., Adekoya, A.F. & Weyori, B.A, ”A comprehensive evaluation of ensemble learning for stock-market prediction, ” *Journal of Big Data*, 7, 20, 2020. doi: 10.1186/s40537-020-00299-5.

[20] Hiransha, M., Gopalakrishnan, E.A., Vijay Krishna Menon, Soman, K.P, ”NSE Stock Market Prediction Using Deep-Learning Models, ” *Procedia Computer Science*, 132, 1351-1362, 2018. doi: 10.1016/j.procs.2018.05.050.

[21] Phuoc, T., Anh, P. T. K., Tam, P. H., & Nguyen, C. V, ”Applying machine learning algorithms to predict the stock price trend in the stock market–The case of Vietnam, ”*Humanities and Social Sciences Communications*, 11(1), 1-18, 2024.

Truong Dinh Tu received his Ph.D. degree from Southeast University, Nanjing, China, in 2016. He is currently a lecturer and researcher at the Faculty of Information Technology, Ton Duc Thang University, Ho Chi Minh City, Vietnam. His research interests include Cybersecurity, Data Analysis, Machine Learning, and Artificial Intelligence.
Email: truongdinhtu@tdtu.edu.vn

Bhagawan Nath Bhagawan Nath is currently pursuing a Ph.D. program at the University of Jyväskylä, Finland. Since May 2024, he has been a lecturer at the Faculty of Information Technology, Greenwich University, Hanoi, Vietnam. His research interests include cybersecurity and data analysis using Machine Learning, Deep Learning, and other advanced technologies.
Email: jitu.nath.vn@gmail.com

Resolving Multi-Class Imbalance using Counterfactual Data Augmentation

Thai-Nguyen Xuan[1,2], Hung-Nguyen Quang[3], Bac-Le[1,2]

[1] Faculty of Information Technology, University of Science, Ho Chi Minh, Vietnam
[2] Vietnam National University, Ho Chi Minh, Vietnam
[3] Posts and Telecommunications Institute of Technology, Hanoi, Vietnam

Correspondence: Bac Le. Email: lhbac@fit.hcmus.edu.vn
Communication: received 04 Oct 2024, revised 05 Nov 2024, accepted 01 Dec 2024
DOI: 10.32913/mic-ict-research.v2025.n1.1328

***Abstract*: Data imbalance in multi-class classification, where some classes have significantly fewer samples than others, is a prevalent challenge in machine learning. This paper evaluates the effectiveness of Counterfactual Augmentation for Multi-Class (CFA-MC), a data augmentation technique that utilizes "native counterfactuals" to generate synthetic samples for minority classes, compared to two popular oversampling techniques, SMOTE and ADASYN. We conduct experiments on multiple benchmark multi-class datasets and employ three different classifiers (Random Forest, k-Nearest Neighbors, and Multilayer Perceptron) to assess the performance of the three methods. Experimental results demonstrate that CFA-MC consistently outperforms SMOTE and ADASYN in terms of macro-averaged F1-score, suggesting its ability to generate more plausible synthetic samples, improve decision boundary representation, and mitigate overfitting.**

Keywords: *Multi-class imbalance, data augmentation, counterfactuals, SMOTE, ADASYN, classification.*

I. Introduction

Data imbalance in multi-class classification, where some classes (minority classes) have considerably fewer samples than others (majority classes), is a common problem in machine learning [**?** **?**]. This issue can lead to classification models being biased towards the majority classes, resulting in poor performance on the minority classes. Oversampling techniques such as SMOTE (Synthetic Minority Oversampling Technique) [**?**] and ADASYN (Adaptive Synthetic Sampling Approach) [**?**] are often used to address data imbalance. However, these methods can generate noisy or non-representative synthetic samples that do not accurately reflect the true data distribution.

This paper introduces CFA-MC (Counterfactual Augmentation for Multi-Class), a novel data augmentation technique based on the concept of "native counterfactuals" as introduced by Keane and Smyth [**?**]. CFA-MC identifies pairs of samples that are close in feature space but belong to different classes, then generates synthetic samples for the minority classes by adapting these real samples. This approach aims to produce more plausible synthetic samples, situated near the decision boundaries, and enhance the model's discriminative ability.

II. Related Methods

1. SMOTE (Synthetic Minority Over-sampling Technique)

SMOTE [**?**] is a popular oversampling technique that operates by generating new synthetic samples for the minority class through interpolation between neighboring samples. Specifically, SMOTE randomly selects a sample from the minority class, finds its k nearest neighbors (typically k=5), then creates a new synthetic sample along the line segment connecting the chosen sample to one of its neighbors.

2. ADASYN (Adaptive Synthetic Sampling Approach)

ADASYN [**?**] is an adaptive oversampling technique that extends SMOTE by generating more synthetic samples for minority samples that are difficult to classify. ADASYN calculates a degree of difficulty for each minority sample based on the ratio of majority class samples among its k nearest neighbors. More synthetic samples are generated for harder samples.

3. Counterfactual Augmentation (CFA)

The Counterfactual Augmentation (CFA) method, as introduced by Keane and Smyth [**?** **?**], presents a novel approach to data augmentation by leveraging the concept of "native counterfactuals" within a dataset. These native counterfactuals represent pairs of existing instances that

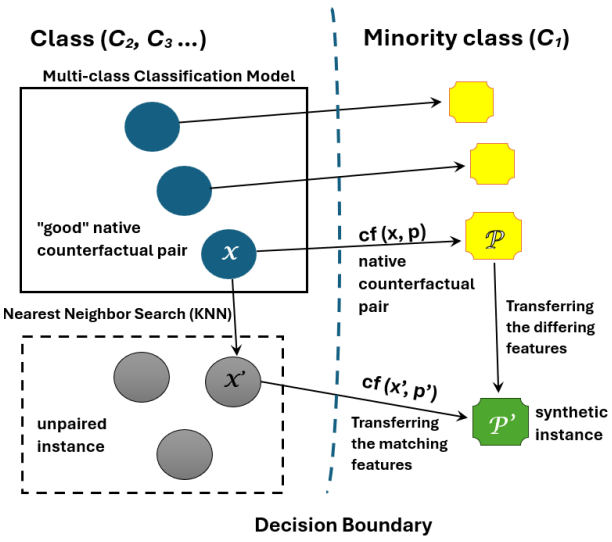


Figure 1. CFA-MC

are close in feature space but belong to different classes, effectively highlighting potential decision boundaries. CFA utilizes these pairs as templates for generating synthetic instances in the minority class. For an unpaired majority class instance, CFA identifies its nearest neighbor that is part of a native counterfactual pair. It then generates a synthetic minority class instance by combining the matching features of the unpaired instance with the differing features of the minority class instance in the counterfactual pair. This process allows CFA to generate plausible synthetic instances that lie near decision boundaries, strengthening the minority class representation and facilitating a more balanced learning process. Notably, CFA distinguishes itself from other oversampling techniques by utilizing actual feature values from the dataset rather than relying on interpolation, potentially reducing the risk of introducing artificial noise.

III. Proposed Method: CFA-MC (Counterfactual Augmentation for Multi-Class)

Inspired by CFA [**? ?**], which is a data augmentation technique that utilizes "native counterfactuals" to generate synthetic samples for the minority class, we extend CFA to handle multi-class datasets. The core idea is to exploit existing "native counterfactuals" within the dataset to guide the generation of synthetic instances for minority classes. A native counterfactual is a pair of instances from different classes that are close in feature space but exhibit a change in class label, suggesting a potential decision boundary.

Figure 1 effectively illustrates the CFA-MC process detailed as follows:

Multi-Class Classification Model: The blue dashed line represents the decision boundary separating multiple classes (C2, C3, etc.). The upper box depicts the majority classes, while the right side represents the minority class (C1).

"Good" Native Counterfactual Pair: the blue circle (instance 'x' from class C2) connected to the yellow octagon (instance 'p' from class C1) represents a "good" native counterfactual pair. They are "close" in feature space but belong to different classes. This closeness is determined by the tol (tolerance) parameter, while "goodness" is based on the fd (feature difference) parameter.

Unpaired Instances: the gray circles in the lower box represent "unpaired instances" of a minority class (C1). These are instances not yet identified as part of a "good" native counterfactual pair.

K-Nearest Neighbor (KNN) Search: the arrow from the gray circle (unpaired instance 'x") to the blue circle (paired instance 'x') represents KNN search for the nearest neighbor among instances that are part of "good" counterfactual pairs.

Feature Transfer: the arrow from the gray circle (x') to the green octagon (synthetic instance 'p") represents the transfer of "matching features" from the unpaired instance. The arrow from the yellow octagon (p) to the green octagon (p") indicates the transfer of "differing features" from the minority class instance in the native counterfactual pair.

Synthetic Instance: the green octagon (p") represents the newly created synthetic instance, ideally belonging to the minority class (C1) and situated near the decision boundary.

Experiment:

The CFA-MC algorithm operates as follows:

Step 1. *Initialization*: The original dataset D is initialized as the augmented dataset D'.

Step 2. *Identify Minority Classes*: Classes with fewer instances than the average number of instances across all classes are identified as minority classes.

Step 3. *Find Native Counterfactuals*: For each minority class C_i, the algorithm identifies "good" native counterfactuals: Nearest Neighbor Search: Using KNN, for each instance in C_i, the algorithm finds its nearest neighbors from other classes. Filter Counterfactuals: Pairs of instances from different classes (C_i and C_j) that are "close" in feature space (based on a predefined tolerance threshold) are selected as potential counterfactuals. Select "Good" Pairs: From the potential counterfactuals, "good" pairs are those where the number of differing features between the two instances is less than or equal to a predefined maximum feature difference count (k).

Step 4. *Generate Synthetic Instances*: For each unpaired instance x' in the minority class C_i: Nearest Neighbor in

Algorithm 1 The CFA-MC algorithm

Input: D: Original dataset (instances and labels), k: Number of nearest neighbors for counterfactual search, tol: Tolerance threshold for feature distance, fd: Maximum feature difference for "good" counterfactuals, M: Classification model for validation, Balancing_ratio (optional): Target ratio for class balance.

Output: D': Augmented dataset.

1. Initialize D' = D
2. Identify minority classes in D
3. While not achieved desired class balance (or maximum iterations reached) Do
4. For each minority class C_i Do
5. Find "good" native counterfactuals for C_i using k, tol, and fd
6. For each unpaired instance x' in C_i Do
7. Find nearest paired instance x in a "good" counterfactual pair (x, p)
8. Generate synthetic instance p' by transferring differing features from p to x'
9. If M predicts p' belongs to C_i Then
10. Add p' to D'
11. End If
12. End For
13. End For
14. End While
15. Return D'.

Counterfactual Pair: The algorithm identifies the nearest paired instance x belonging to another class C_j that is part of a "good" native counterfactual pair (C_i, C_j). Transfer Feature Values: A synthetic instance p' is created by copying the feature values of x', but transferring the "differing features" from the counterfactual instance p (belonging to class C_j) in the identified counterfactual pair (x, p). Validation: The synthetic instance p' is validated using a classification model M. If p' is predicted to belong to the intended minority class C_i, it is added to D'.

Step 5. *Repeat*: Steps 2-4 are repeated until the desired class balance is achieved.

Step 6. *Return*: The augmented dataset D' is returned.

1. Complexity Identifying minority classes:

$O(N)$ complexity, where N is the number of classes. *KNN search*: Complexity depends on the KNN algorithm used. With brute-force KNN, the complexity is $O(M \times K)$ for each search, where M is the number of data points and K is the number of nearest neighbors. *Identifying "good" native counterfactuals*: $O(M \times F)$ complexity, where F is the number of features, to calculate the tolerance threshold and identify "differing" and "matching" features. *Generating synthetic data*: $O(M \times (N-1) \times K \times F)$ complexity per iteration, as it's necessary to search for nearest neighbors and generate synthetic data for each minority data point, with each remaining majority class. *Label prediction*: Complexity depends on the multi-class classification model M used.

Since the algorithm iterates until the desired balance is achieved, the overall complexity depends on the number of iterations needed. In the worst case, the time complexity of CFA-MC can reach O(L×M×N ×K×F), where L is the number of iterations.

2. Memory Space

The CFA-MC algorithm needs to store: Original dataset D: $O(M \times F)$. Augmented dataset D': $O(M' \times F)$, where M' is the number of data points after augmentation. Multi-class classification model M: Depends on the size of the model. Intermediate data structures: $O(M \times K)$ for the list of nearest neighbors and $O(M \times F)$ for "good" native counterfactual pairs. In the worst case, the memory space of CFA-MC can reach $O(M' \times F + M \times K + M \times F + size(M))$.

3. Model-Dependent Parameters (tol and fd):

The CFA-MC algorithm relies on two main parameters, tol (tolerance) and fd (feature difference), to control the generation of synthetic instances. These parameters guide the selection of "good" native counterfactuals, which is crucial for generating plausible and effective synthetic data points.

Tolerance (tol): This parameter defines the threshold for considering two instances "close" in feature space. A lower tol value enforces a stricter matching criterion, meaning only instances with very similar feature values will be considered for forming counterfactual pairs. Conversely, a higher tol allows for more flexibility, including pairs with larger differences in feature values.

Feature Difference (fd): This parameter limits the maximum number of features allowed to differ between instances in a "good" counterfactual pair. A smaller fd value restricts the algorithm to using counterfactual pairs that exhibit minimal differences, promoting simpler and potentially more interpretable adaptation rules. On the other hand, a larger fd permits the use of pairs with more feature differences, potentially capturing more complex relationships in the data but also increasing the risk of generating less plausible synthetic instances.

In our experiments, we initially set tol to 10% of the standard deviation for each feature and fd to 2. This

configuration balances ensuring the plausibility of synthetic instances and allowing for sufficient diversity in the augmented data. We also experimented with different tol and fd values; however, preliminary results indicated that the chosen configuration yielded the most promising performance.

The selection of appropriate values for tol and fd depends on the specific characteristics of the dataset and the desired properties of the synthetic instances. Therefore, further investigation into the impact of tol and fd values across different datasets and classifiers is an avenue for future research, potentially leading to adaptive strategies for parameter selection.

IV. Experiments

Datasets: We utilize three benchmark multi-class datasets from the UCI repository [7] to evaluate the effectiveness of the three methods:

D1 Yeast Dataset: Prediction of protein cellular localization.

D2 Abalone Dataset: Prediction of abalone age from physical measurements.

D3 Wine Quality White Dataset: Classification of white wine quality based on physicochemical characteristics

The imbalance ratio in the target classes ranges from 1.86 to 82, as illustrated in Figures 2, 3, and 4 below.

Classifiers: We employ three commonly used classifiers, implemented using scikit-learn [14], for performance evaluation:

Random Forest (RF) [8]

K-Nearest Neighbors (K-NN) [9]

Multilayer Perceptron (MLP) [10]

Evaluation: We utilize the macro-averaged F1-score [11] to assess the performance of the three methods.

Experimental Procedure

Data Preprocessing: Convert categorical features to numerical representation using one-hot encoding. Split the dataset into training (80%) and testing (20%) sets.

Data Augmentation: The CFA-MC method is experimentally compared against Baseline, SMOTE-MC, ADASYN-MC (Technically, SMOTE and ADASYN are not directly designed for multi-class problems. Initially, SMOTE and ADASYN were proposed to handle data imbalance in binary classification. However, they can be adapted for multi-class problems by employing the One-vs-All (OVA) strategy [13], which divides the multi-class problem into multiple binary problems, each distinguishing one class from all the others. Then SMOTE and ADASYN are applied to each of these binary problems. Hence, we refer to them as SMOTE-MC and ADASYN-MC).

Classifier Training and Evaluation: Train each classifier on the augmented training set using 5-fold cross-validation for hyperparameter tuning. Evaluate the best model for each method on the original test set Experimental Results:

Figure 5, 6, 7: CFA-MC Data Balance Results. All experimental results are published at https://github.com/mrnxthai/CFA-MC.git

V. Discussion and Analysis

The experimental results show that CFA-MC consistently outperforms SMOTE and ADASYN in terms of F1-score across all datasets and classifiers. This can be attributed to: **CFA-MC generates more plausible synthetic samples**: By utilizing "native counterfactuals," CFA-MC leverages information from real samples near the decision boundaries, leading to the generation of synthetic samples that are more representative of the data distribution. *CFA-MC improves decision boundary representation*: CFA-MC focuses on generating synthetic samples near the decision boundaries, which helps the model learn more accurately and better discriminate between classes. *CFA-MC mitigates overfitting*: CFA-MC utilizes actual feature values and avoids interpolation, reducing the risk of generating noisy synthetic samples, thereby mitigating overfitting.

Despite its advantages, CFA-MC also has some limitations: *Dependence on native counterfactuals*: The effectiveness of CFA-MC depends on the availability of "good" native counterfactuals in the dataset. If the dataset lacks suitable counterfactual pairs, or if the identified pairs do not represent true decision boundaries, the performance of CFA-MC can be affected. *Computational complexity*: Searching for and validating counterfactuals can be computationally expensive, especially for large datasets with many features.

VI. Conclusion

The main goals of CFA-MC are: *Plausible synthetic instances*: By leveraging real feature values from native counterfactuals, CFA-MC generates synthetic instances that are more realistic and representative of the underlying data distribution.

Targeted data augmentation: CFA-MC focuses on generating synthetic instances for minority classes near the decision boundaries, strategically strengthening their representation and improving the model's ability to discriminate between classes.

Table I
EXPERIMENTAL DATASETS

ID	Dataset	Features	Instances	Prediction Task
D1	Yeast	8	1484	Predicting the cellular localization sites of proteins (non-numerical).
D2	Abalone	8	4177	Predicting the age of abalone from physical measurements.
D3	Wine Quality White	11	4898	Predicting the quality of white wines (score from 1 to 10).

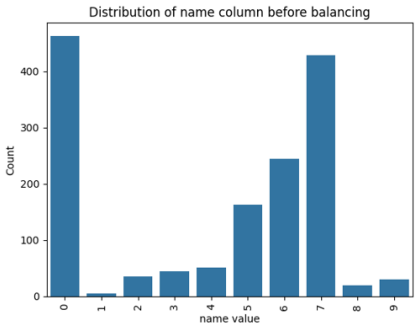


Figure 2: Yeast data distribution

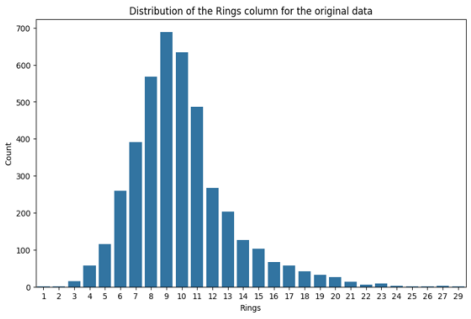


Figure 3: Abalone data distribution

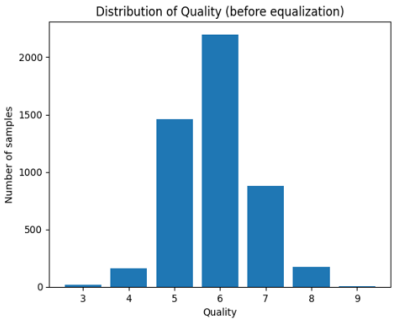


Figure 4: Wine Quality White data distribution

Figure 2. The imbalance ratio

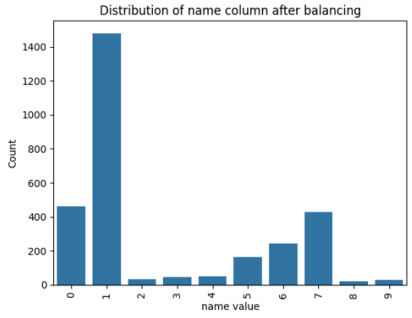


Figure 5: Yeast data distribution

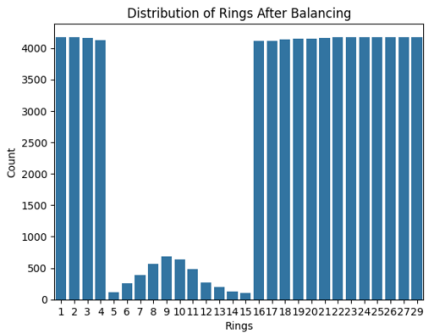


Figure 6: Abalone data distribution

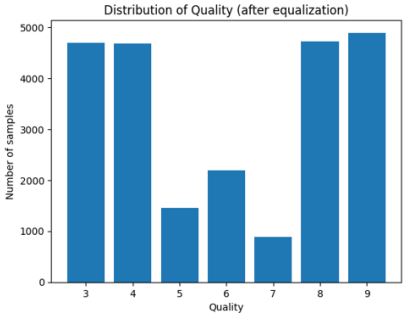


Figure 7: Wine Quality White data distribution

Figure 3. CFA-MC Data Balance Results

Table II
F1-SCORES FOR RF CLASSIFIER

Dataset	Baseline	SMOTE-MC	ADASYN-MC	CFA-MC
D1	0.63	0.58	0.55	**0.78**
D2	0.25	0.2	0.21	**0.5**
D3	0.69	0.64	0.69	**0.8**

Table III
F1-SCORES FOR K-NN CLASSIFIER

Dataset	Baseline	SMOTE-MC	ADASYN-MC	CFA-MC
D1	0.58	0.46	0.42	**0.73**
D2	0.23	0.17	0.17	**0.42**
D3	0.54	0.39	0.48	**0.7**

Table IV
F1-SCORES FOR MLP CLASSIFIER

Dataset	Baseline	SMOTE-MC	ADASYN-MC	CFA-MC
D1	0.61	0.47	0.43	**0.74**
D2	0.31	0.2	0.19	**0.49**
D3	0.54	0.30	0.46	**0.59**

This study demonstrates that CFA-MC is an effective technique for addressing multi-class imbalance, outperforming SMOTE and ADASYN in terms of F1-score across multiple datasets and classifiers. CFA-MC generates more plausible synthetic samples, improves decision boundary representation, and mitigates overfitting.

Future research directions include *Adaptive parameter selection*: Developing strategies to automatically determine

optimal tol and fd values based on the characteristics of the dataset. *Handling noisy data*: Investigating techniques to identify and mitigate the influence of noisy or unreliable native counterfactuals. *Application to deep learning*: Exploring the feasibility and effectiveness of incorporating CFA-MC into deep learning frameworks to address multi-class imbalance in image and text data. CFA-MC has the potential to be further enhanced as a robust and effective technique for handling multi-class imbalance in various machine learning applications.

ACKNOWLEDGEMENT

This research is funded by Vietnam National Foundation for Science and Technology Development (NAFOSTED) under grant number 102.05-2023.44

REFERENCES

[1] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 9, pp. 1263–1284, Sep. 2009.

[2] N. Japkowicz and S. Stephen, "The class imbalance problem: A systematic study," *Intelligent Data Analysis*, vol. 6, no. 5, pp. 429–449, 2002.

[3] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic Minority Over-sampling Technique," *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, Jun. 2002.

[4] H. He, Y. Bai, E. A. Garcia, and S. Li, "ADASYN: Adaptive synthetic sampling approach for imbalanced learning," *Proceedings of the IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, pp. 1322–1328, 2008.

[5] M. Temraz and M. T. Keane, "Solving the class imbalance problem using a counterfactual method for data augmentation," *Machine Learning Applications*, vol. 9, p. 100375, Jun. 2022.

[6] F. Mantiuk and L. Kurth, "Comment on Solving the Class Imbalance Problem Using a Counterfactual Method for Data Augmentation," *Machine Learning Applications*, vol. 11, p. 100412, Dec. 2022.

[7] D. Dua and C. Graff, "UCI Machine Learning Repository," *University of California, Irvine, School of Information and Computer Sciences*, 2017. [Online]. Available: http://archive.ics.uci.edu/ml

[8] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, Oct. 2001.

[9] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE Transactions on Information Theory*, vol. 13, no. 1, pp. 21–27, Jan. 1967.

[10] S. Haykin, *Neural Networks and Learning Machines*, 3rd ed. Upper Saddle River, NJ, USA: Pearson Education, 2009.

[11] D. M. Powers, "Evaluation: From precision, recall and F-measure to ROC, informedness, markedness & correlation," *Journal of Machine Learning Technologies*, vol. 2, no. 1, pp. 37–63, 2011.

[12] M. T. Keane and B. Smyth, "Good counterfactuals and where to find them: A case-based technique for generating counterfactuals for explainable AI (XAI)," *Proceedings of the 28th International Conference on Case-Based Reasoning*, pp. 163–178, 2020.

[13] A. Fernández, S. García, F. Herrera, and N. V. Chawla, "SMOTE for learning from imbalanced data: progress and challenges, marking the 15-year anniversary," *Journal of Artificial Intelligence Research*, vol. 61, pp. 863-905, 2018.

[14] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, ... and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12(Oct), pp. 2825-2830, 2011.

Thai-Nguyen Xuan is a Computer Science teacher at Trinh Hoai Duc High School in Thuan An, Binh Duong province, Vietnam. He received his Bachelor of Science in Computer Science Education from Ho Chi Minh City University of Education in 2008 and is currently pursuing a Master's degree in Computer Science at the University of Science, Vietnam National University, Ho Chi Minh City, Vietnam. His research interests include Deep Learning, Computer Vision, and Bioinformatics.
Email: thainx@trinhhoaiduc.sgdbinhduong.edu.vn

Hung-Nguyen Quang received Bachelor's and Master's degrees from Hanoi University in 1994 and 2000, respectively, and a Ph.D. degree from the Posts and Telecommunications Institute of Technology in 2007. He is currently working at the Posts and Telecommunications Institute of Technology, Hanoi, Vietnam. His research interests include AI applications in e-commerce, multimedia, and policy.
Email: hungnq1@ptit.edu.vn

Bac Le is currently a Professor and Head of the Department of Computer Science, Faculty of Information Technology, University of Science, Vietnam National University, Ho Chi Minh City, Vietnam. His research interests include AI, XAI, Machine Learning, Data Science, and Soft Computing.
Email: lhbac@fit.hcmus.edu.vn

Low-Complexity FNN-Based Transmit Antenna Selection for Enhanced Performance in VASM Systems over Rician Fading Channels

Tran Viet Vinh, Pham Thanh Hiep, Nguyen Thu Phuong
Advanced Wireless Communications Group, Le Quy Don Technical University, Hanoi, Vietnam.

Correspondence: Nguyen Thu Phuong: E-mail: phuong.nt@lqdtu.edu.vn
Communication: received xxx, revised xxx, accepted xxx
DOI: 10.32913/mic-ict-research.v2025.n1.1346

***Abstract*: Variable Active Antenna Spatial Modulation (VASM) is a spatial modulation variant designed to improve spectral efficiency and offer greater flexibility in system configuration. In this paper, we propose a feedforward neural network (FNN) framework to address the transmit antenna selection (TAS) problem, aiming to enhance performance in VASM systems operating over Rician fading channels. Computational results demonstrate that our novel FNN-TAS algorithm provides a significant reduction in computational costs compared to the standard Euclidean distance-based method. Additionally, simulation results indicate that the bit error rate (BER) of the VASM system increases with the Rician factor. However, the proposed FNN-TAS method effectively improves BER performance for VASM systems, particularly at high Rician factors, and outperforms conventional TAS methods based on channel gain.**

Index Terms: *Variable active antenna spatial modulation (VASM), antenna selection (TAS), feedforward neural network (FNN), Euclidean distance-based TAS (ED-TAS), Rician fading channel.*

I. Introduction

Driven by the rapid advancement of sixth-generation (6G) wireless technology [1], future communication systems must meet unprecedented requirements, including ultra-high data rates, ultra-low latency, and seamless connectivity. Achieving these goals while ensuring spectral and energy efficiency is crucial for the sustainable and scalable deployment of next-generation networks.

Among the key technologies shaping 6G, Multiple-Input Multiple-Output (MIMO) has been widely recognized for its ability to enhance spectral efficiency, improve signal reliability, and support massive connectivity [2]. Within the broader family of multi-antenna transmission techniques, Spatial Modulation (SM) [3] has gained increasing attention due to its ability to reduce hardware complexity while maintaining high spectral efficiency, making it a promising candidate for energy-efficient next-generation systems [4].

In SM systems, the bitstream is divided into equally sized groups, with each group further split into two parts. The first part selects an Amplitude and Phase Modulation (APM) symbol, while the second part activates an antenna index to transmit the selected APM symbol. Due to this straightforward operational principle, SM systems enhance spectral efficiency, reduce inter-antenna interference, and minimize radio frequency chain requirements.

Unlike conventional spatial modulation systems, such as SM, Generalized SM (GSM) [5], and Quadrature SM (QSM) [6], which activate a fixed number of transmit antennas (TAs) in a time slot, Variable Active Antenna Spatial Modulation (VASM) [7] is an SM variant that offers the flexibility to vary the quantity of active TAs. This approach further enhances spectral efficiency while reducing the number of required TAs. Additionally, VASM enables more straightforward TA configuration selection, as the TA count is not restricted to powers of two. VASM thus presents itself as a promising candidate for high-speed wireless systems with flexible antenna usage [8–11].

In MIMO systems applying spatial modulation principles, a portion of the information is conveyed through the indices of the active TAs. Studies have shown that increasing the number of TAs can enhance spectral efficiency, but it may come at the cost of increased BER [4]. In MIMO systems with a large number of TAs, potentially reaching hundreds (massive MIMO [12]), using all TA indices to transmit additional information through SM principles is impractical, as this would result in an excessively high BER, compromising system quality. Moreover, each TA corresponds to a unique transmission channel, potentially offering varying transmission effectiveness. Therefore, selecting the appropriate TAs for modulation and signal trans-

mission is crucial in systems operating on SM principles like VASM.

Various antenna selection techniques have been proposed to address this critical aspect in family SM systems [13–16]. Channel-Gain-based TAS (CG-TAS) as well as Euclidean distance-based TAS (ED-TAS) [13] are two fundamental techniques underpinning most TAS schemes for SM and its variants. CG-TAS selects TAs based on their channel gains, while ED-TAS uses an exhaustive search to optimize the Euclidean distance between transmit signals, resulting in the best BER performance.

To reduce the computational complexity of ED-TAS while maintaining near-optimal BER performance, various antenna selection strategies have been proposed for VASM systems. One such strategy, the hierarchical approach combining CG-TAS together with ED-TAS, termed HC-TAS, is presented in [17–19]. This algorithm offers a flexible trade-off between computational complexity and BER performance, adapting to diverse system requirements.

The Rician fading channel is a widely observed propagation model in wireless communications, characterized by the presence of a dominant line-of-sight (LoS) component alongside multiple weaker multipath components [20]. This channel model is particularly relevant in scenarios where a strong direct transmission path coexists with scattered reflections, such as in satellite communications, urban mobile networks, and millimeter-wave systems.

However, previous studies [21–23] have demonstrated that the BER performance of spatial modulation-based systems significantly degrades in Rician fading channels as the Rician factor increases. This degradation occurs because, in scenarios where the LoS component is dominant, the receiver faces increased difficulty in distinguishing which antenna transmitted the signal, leading to a higher probability of antenna index detection errors. In such conditions, conventional TAS schemes that rely on channel gain metrics, such as CG-TAS and HC-TAS, may become ineffective in Rician fading environments. This challenge underscores the necessity of developing advanced TAS techniques tailored to the unique characteristics of Rician channels to maintain robust system performance.

Recently, several studies have explored the use of neural networks to replace traditional TAS methods for SM family systems [24–28]. These approaches have shown promising improvements in performance metrics. For example, studies such as [24, 25] proposed using deep neural networks (DNNs) for TAS and signal detection in SM-MIMO systems, achieving near-optimal BER performance with low computational complexity. Three typical DNNs—feedforward neural networks (FNN), convolutional neural networks (CNN), and recurrent neural networks (RNN)—have been employed for TAS in SM-MIMO systems [26] and for fully generalized spatial modulation (FGSM) systems in [27]. These studies demonstrated that leveraging DNNs for TAS can significantly reduce computational burden while improving BER performance.

However, these prior works have primarily focused on minimizing computational complexity and achieving near-optimal BER performance under Rayleigh fading channel conditions. As far as we are aware, no existing study has explored TAS under Rician fading conditions, which are commonly encountered in practical scenarios, or within VASM systems.

Additionally, CNNs are particularly effective for spatially correlated data due to their ability to extract local features, while RNNs excel at modeling sequential data owing to their capacity to retain temporal dependencies. For data such as random wireless channels, where spatial and temporal correlations are minimal or nonexistent, FNNs are a practical and efficient choice [29].

For these reasons, in this paper, we propose employing an FNN-based TAS approach for VASM systems operating over Rician fading channels.

The main contributions of this paper can be summarized as follows:

- This paper develops an FNN structure for efficient antenna selection in VASM systems, achieving notable computational complexity reductions over the optimal ED-TAS method and delivering improved BER performance relative to CG-TAS and HC-TAS under high Rician factor conditions.
- We propose five distinct techniques for constructing datasets to train neural networks for antenna selection and compare the effectiveness between them.
- We calculate and compare computational complexity to demonstrate the superiority of FNN-based TAS (FNN-TAS) over ED-TAS.
- Reliable simulation results are presented to evaluate the influence of the Rician factor and the efficacy of various TAS techniques on the BER performance of VASM systems employing the proposed methods.

The remainder of the paper is organized as follows: Section II presents the VASM system model with TAS and the Rician channel model. Section III details the antenna selection method using FNN. Simulation results are provided in Section IV. Section V concludes the paper and suggests future research directions.

Notations: The notations used throughout this paper are defined as follows. Italic lowercase letters represent variables, while bold uppercase letters denote matrices, and bold lowercase letters signify vectors. The notation $(\bullet)^T$

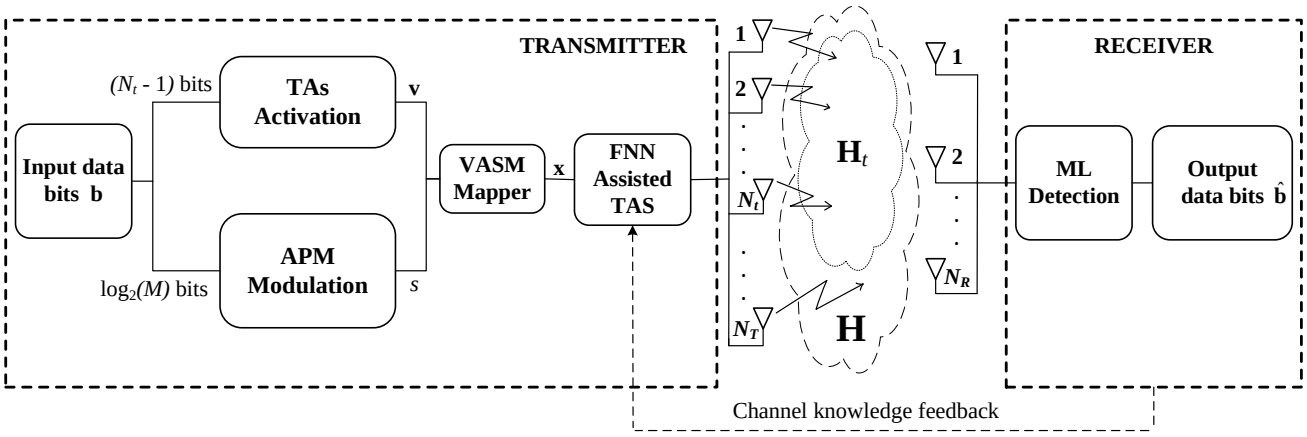


Figure 1. VASM system architecture incorporating TAS.

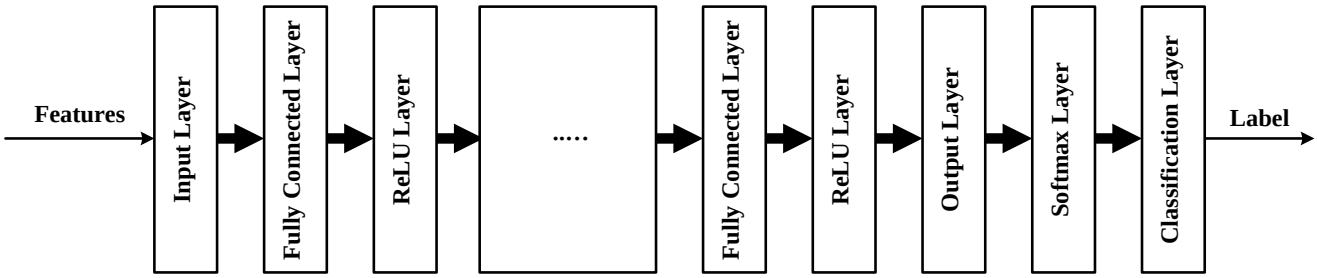


Figure 2. Proposed FNN architecture.

is used for the transpose operation, and $(\bullet)^H$ indicates the Hermitian transpose operation. The symbol $\|\bullet\|$ denotes the calculation of a norm. A matrix of size $A \times B$ is represented by $\mathbb{C}^{A\times B}$, and C_b^a signifies the combination of a elements chosen from b elements. Finally, the expectation operator is denoted by $\mathbb{E}[\bullet]$.

II. System Model

The VASM transceiver model [11] incorporating FNN-TAS is illustrated in Fig. 1. This model differs from previous VASM models by employing FNN-TAS for transmit antenna selection instead of traditional methods [17].

In this setup, the transmitter is equipped with N_T TAs, but only N_t of these TAs are actively used for data transmission, where $N_t < N_T$. On the receiver side, there are N_R receive antennas (RAs). The channel between the transmitter and receiver is represented by the channel matrix $\mathbf{H} \in \mathbb{C}^{N_R\times N_T}$. When N_t TAs are selected, the channel matrix is reduced to an effective channel matrix $\mathbf{H}_t \in \mathbb{C}^{N_R\times N_t}$, obtained by selecting the corresponding columns of $\mathbf{H}$.

This study considers the Rician fading channel model, which is widely used to describe environments where a strong LoS component is present alongside several scattered non-LoS (NLoS) paths. This channel can be mathematically expressed as the sum of a deterministic LoS component $\mathbf{H}_{\text{LoS}} \in \mathbb{C}^{N_R\times N_T}$ and a stochastic NLoS component $\mathbf{H}_{\text{NLoS}} \in \mathbb{C}^{N_R\times N_T}$ with $\mathcal{CN}(0,1)$. The channel model is given by [20]:

$$\mathbf{H} = \sqrt{\frac{K}{K+1}}\mathbf{H}_{\text{LoS}} + \sqrt{\frac{1}{K+1}}\mathbf{H}_{\text{NLoS}}, \tag{1}$$

here K is Rician factor that represents the power ratio of the LoS to NLoS components.

In the system, the receiver first estimates the channel state information (CSI) and feeds this information return to the transmitter. Utilizing the received channel knowledge, the transmitter selects N_t TAs out of the total available N_T TAs to transmit the data, aiming to maximize the quality of the signal and thus improve the overall system performance.

The input data bit sequence b is segmented into groups of fixed length $(N_t-1)+\log_2(M)$ bits, where M denotes the modulation order of the APM scheme. Each group is then divided into two parts: the first part, comprising N_t-1 bits, represents the spatial bits, while the second part, containing $\log_2(M)$ bits, represents the symbol bits.

The spatial bits specify the activation states of the TAs, with a bit value of 1 indicating an active TA and a bit value of 0 indicating an inactive one. In cases where all spatial bits are 0, the last TA within the chosen set of N_t TAs is activated. The remaining $\log_2(M)$ bits are modulated into an APM symbol, $s_m \in \mathbb{S}$, $m \in [1; M]$, which is then mapped to the positions of the active TAs, forming the transmitted VASM signal vector $\mathbf{x} \in \mathbb{X}$.

Using the above simple mapping principle, VASM activates between 1 and $N_t - 1$ TAs to emit signals within a single time slot, achieving a spectral efficiency of $\log_2(M)+(N_t-1)$ bits per channel use. Consequently, it can operate with an unrestricted number of TAs, achieving considerably greater spectral efficiency in comparison to other SM family variants [18].

The received signal vector $\mathbf{y} \in \mathbb{C}^{N_R \times 1}$ is represented as follows:

$$\mathbf{y} = \sqrt{\gamma}\mathbf{H}_t\mathbf{x} + \mathbf{n}, \tag{2}$$

here $\mathbf{n} \in \mathbb{C}^{N_R \times 1}$, $\mathcal{CN}(0, 1)$, represents the additive white Gaussian noise (AWGN) vector. γ represents the average signal-to-noise ratio (SNR) at the receiver.

With the assumption of perfect CSI knowledge at the receiver, the received signal is detected using optimal maximum likelihood detection [30], leading to the following expression:

$$\widehat{\mathbf{x}} = \underset{\mathbf{x}\in\mathbb{X}}{\operatorname{argmin}} \left\|\mathbf{y} - \sqrt{\gamma}\mathbf{H}_t\mathbf{x}\right\|^2, \tag{3}$$

here $\hat{\mathbf{x}}$ denotes the estimated transmitted signal. $\hat{\mathbf{x}}$ is, finally, mapped back to the original data bits.

III. Transmit Antenna Selection

In a VASM-MIMO system, there are a total of N_T TAs at the transmitter, but the number of available radio frequency chains is $N_f < N_T$. The required number of TAs to be selected for transmission is $N_t = N_f$. The TAS problem can thus be defined as the task of selecting N_t columns out of the N_T columns of the channel matrix $\mathbf{H}$ with the best quality for signal transmission.

1. Conventional transmit antenna selection methods

a) Transmit antenna selection based on channel gain

The CG-TAS method is a selection algorithm that chooses the best N_t TAs out of N_T TAs based on the criterion of maximizing individual channel gains. In particular, the usage channel matrix $\mathbf{H}_t^{\mathrm{CG}}$ is constructed by choosing N_t columns from the channel matrix $\mathbf{H}$, as follows:

$$\mathbf{H}_t^{\mathrm{CG}} = \begin{bmatrix}\mathbf{h}_1 & \mathbf{h}_2 & \cdots & \mathbf{h}_{N_t}\end{bmatrix}. \tag{4}$$

These columns are selected by sorting the column norms in descending order:

$$||\mathbf{h}_1|| > ||\mathbf{h}_2|| > \cdots > ||\mathbf{h}_{N_t}|| > \cdots > ||\mathbf{h}_{N_T}||. \tag{5}$$

b) Transmit antenna selection using Euclidean distance

The ED-TAS approach is an optimization algorithm designed to identify the optimal subset of TAs that maximizes the minimum Euclidean separation between transmitted signal vectors, thereby enhancing system performance.

$$\mathbf{H}_t^{\mathrm{ED}} = \arg \max_{c\in[1;C_{N_T}^{N_t}]} \{ \min_{\mathbf{x}_i\neq\mathbf{x}_j\in\mathbb{X}} ||\mathbf{H}_t^c(\mathbf{x}_i - \mathbf{x}_j)||^2\}. \tag{6}$$

c) Combination of CG-TAS and ED-TAS in a hierarchical structure

The HC-TAS approach employs a sequential two-stage process to select the optimal subset of TAs. In the initial stage, a subset of N_s TAs with the highest channel gains is chosen from the total N_T TAs, forming a reduced channel matrix $\mathbf{H}_{N_s}^{\mathrm{CG}}$:

$$\mathbf{H}_{N_s}^{\mathrm{CG}} = \begin{bmatrix}\mathbf{h}_1 & \mathbf{h}_2 & \cdots & \mathbf{h}_{N_s}\end{bmatrix}, \tag{7}$$

where the columns are sorted in descending order of their norm. Subsequently, the ED-TAS algorithm is applied to the reduced set of N_s TAs to select the final N_t TAs, as expressed by the following optimization problem:

$$\mathbf{H}_t^{\mathrm{HC}} = \arg \max_{d\in[1;C_{N_s}^{N_t}]} \min_{\mathbf{x}_i\neq\mathbf{x}_j\in\mathbb{X}} ||\mathbf{H}_t^d(\mathbf{x}_i - \mathbf{x}_j)||^2. \tag{8}$$

Thus, the HC-TAS method effectively leverages CG-TAS for coarse-grained selection and ED-TAS for fine-grained optimization.

2. Feedforward neural network-based transmit antenna selection

a) FNN architecture design

The proposed FNN architecture is illustrated in Fig. 2. It consists of several sequential layers, including an input layer, an output layer, and fully connected hidden layers. The network begins with an input layer that receives the feature vector, followed by some fully connected (dense) layers, each activated by a rectified linear unit (ReLU) layer to introduce non-linearity. The output from the final dense layer is passed to an output layer with a Softmax function to compute a probability distribution across classes, assigning the class with the highest probability as the final prediction. This architecture enables the FNN to learn complex patterns within the input data, facilitating accurate classification.

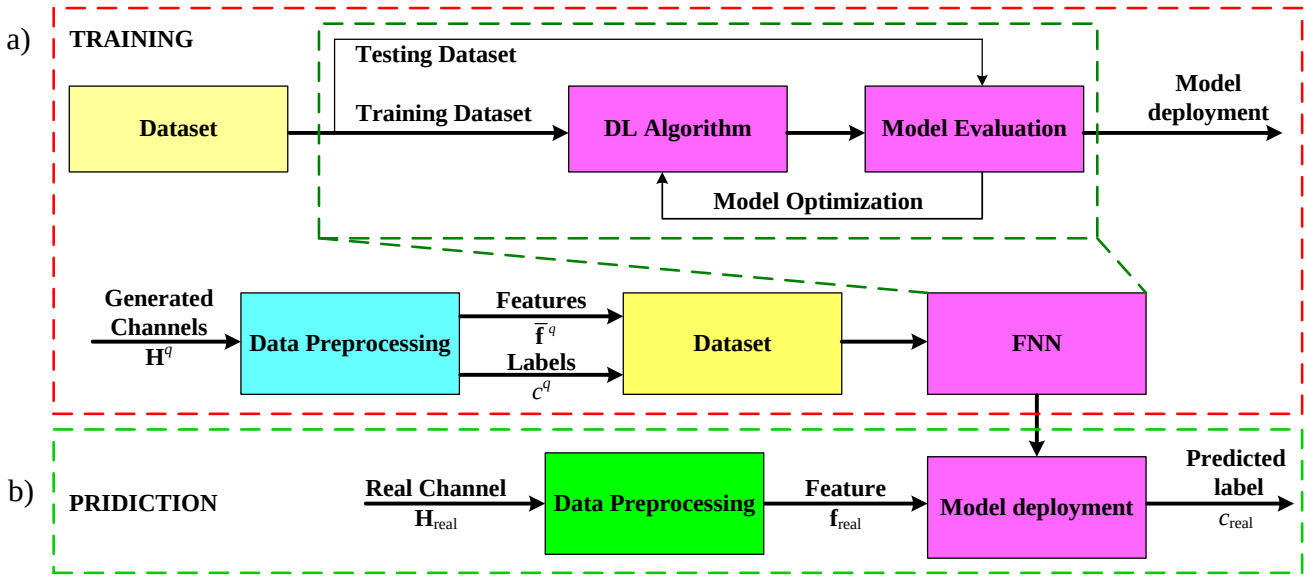


Figure 3. Training and prediction procedure of proposed FNN.

b) Preparation of training data

To create the training dataset, the channel matrices with complex values must be transformed into real-valued feature vectors compatible with FNN input requirements. The steps for preparing the training data are as follows:

- ***Step 1***: Generate a total of Q random channel matrices $\mathbf{H}^q$, where $1 \le q \le Q$, as given in (1).
- ***Step 2***: Feature extraction from each channel matrix $\mathbf{H}^q$. In this study, we derive five distinct feature types from $\mathbf{H}^q$, each representing different channel attributes. The extraction process for these features is outlined below:

  - Real and imaginary components (R&I):

$$\mathbf{f}^q_{\text{R\&I}} = \left[\Re(h^q_{1,1}), \Im(h^q_{1,1}), \ldots, \Re(h^q_{N_R,N_T}), \Im(h^q_{N_R,N_T})\right], \tag{9}$$

  where $h^q_{a,b}$, with $1 \le a, b \le N_T$, is the (a,b)-th entry of $\mathbf{H}^q$.

  - Magnitude (Mag):

$$\mathbf{f}^q_{\text{Mag}} = [|h^q_{1,1}|, |h^q_{1,2}|, \ldots, |h^q_{N_R,N_T}|], \tag{10}$$

  with $|h^q_{a,b}| = \sqrt{\Re(h^q_{a,b})^2 + \Im(h^q_{a,b})^2}$.

  - Norm (Norm):

$$\mathbf{f}^q_{\text{Norm}} = \left[||\mathbf{h}^q_1||, ||\mathbf{h}^q_2||, \ldots, ||\mathbf{h}^q_{N_T}||\right]. \tag{11}$$

  - Phase (Phase):

$$\mathbf{f}^q_{\text{Phase}} = \left[\angle(h^q_{1,1}), \angle(h^q_{1,2}), \ldots, \angle(h^q_{N_R,N_T})\right], \tag{12}$$

  where $\angle(h^q_{a,b}) = \tan^{-1}\left(\frac{\Im(h^q_{a,b})}{\Re(h^q_{a,b})}\right)$.

  - Correlation (Corr):

$$\mathbf{f}^q_{\text{Corr}} = \left[|\mathbf{h}^{q\,H}_1 \mathbf{h}^q_1|, |\mathbf{h}^{q\,H}_1 \mathbf{h}^q_2|, \ldots, |\mathbf{h}^{q\,H}_{N_T} \mathbf{h}^q_{N_T}|\right]. \tag{13}$$

  After obtaining the features, normalization is performed to prevent bias effects. Max-min normalization is applied as follows:

$$\bar{\mathbf{f}}^q = \frac{\mathbf{f}^q - \mathbb{E}[\mathbf{f}^q]}{\mathbf{f}_{\max} - \mathbf{f}_{\min}}, \tag{14}$$

  where $\mathbf{f}_{\min} = \min(\mathbf{f}^1, \mathbf{f}^2, \ldots, \mathbf{f}^Q)$ and $\mathbf{f}_{\max} = \max(\mathbf{f}^1, \mathbf{f}^2, \ldots, \mathbf{f}^Q)$ are the minimum and maximum values of the features, respectively.
- ***Step 3***: Labeling. Each label c^q corresponds to the index of the optimal TA combination, derived based on the following criterion:

$$c^q = \underset{c \in [1; C^{N_t}_{N_T}]}{\arg\max} \left\{ \min_{\mathbf{x}_i \ne \mathbf{x}_j \in \mathbb{X}} ||\mathbf{H}^q_{t,c}(\mathbf{x}_i - \mathbf{x}_j)||^2 \right\}, \tag{15}$$

  where $\mathbf{H}^q_{t,c} \in \mathbb{C}^{N_R \times N_t}$ are the sub-channel matrices generated by selecting N_t columns from the N_T columns in $\mathbf{H}^q \in \mathbb{C}^{N_R \times N_T}$.

As a result, the training dataset is composed of two parts: the feature vector set $\mathbb{F} = \{\bar{\mathbf{f}}^1, \bar{\mathbf{f}}^2, \ldots, \bar{\mathbf{f}}^Q\}$ and the associated label vector $\mathbf{c} = \{c^1, c^2, \ldots, c^Q\}$. The neural network is trained to map these features to labels by minimizing a loss function, allowing it to predict optimal antenna configurations in real-world channel scenarios.

c) Training and testing

Figure 3a provides a detailed illustration of the training process of the proposed FNN. Initially, a dataset comprising

generated channel realizations $\mathbf{H}^q$ undergoes preprocessing to extract relevant feature vectors and their corresponding labels. For each channel matrix $\mathbf{H}^q$, a feature vector $\mathbf{f}^q$ is extracted based on one of the expressions from (9) to (13), depending on the selected feature extraction technique, and is subsequently normalized according to (14) to obtain $\bar{\mathbf{f}}^q$. The corresponding label c^q is determined using (15). During training, the neural network learns the mapping relationship between the feature vector $\bar{\mathbf{f}}^q$ and the label c^q. The dataset is then divided into a training set and a testing set, where the former is used for parameter optimization, while the latter is employed to assess model generalization. The training dataset is fed into a deep learning algorithm, which iteratively updates the network parameters through an optimization process. The trained model is subsequently validated using the testing dataset, and performance evaluation is conducted to ensure satisfactory accuracy. Once optimal performance is achieved, the trained model is deployed for real-time inference.

d) Prediction

Figure 3b demonstrates the prediction (inference) phase of the deployed FNN. In this phase, the real channel matrix $\mathbf{H}_{\text{real}}$, representing the actual communication environment, undergoes the same preprocessing steps as in the training phase to extract the feature vector $\mathbf{f}_{\text{real}}$. This feature vector is then fed into the trained FNN model, which predicts the label c_{real}. This label corresponds to the optimal set of transmitting antennas for the current channel conditions, thereby facilitating enhanced transmission performance in practical scenarios.

3. Computational complexity

In this study, we define the computational complexity of TAS methods as the number of floating-point operations (flops) required to determine the TA combination for the VASM system.

According to the analysis results in [17, 18], the computational complexity of conventional TAS methods can be expressed as:

$$\Omega_{\text{CG-TAS}} = N_T(4N_R - 1) + N_T N_t - \frac{N_t(N_t + 1)}{2} \tag{16}$$

$$\Omega_{\text{ED-TAS}} = C_{N_T}^{N_t} \times \left(\frac{2^{2\varepsilon} - 2^{\varepsilon}}{2} \times (8N_R N_t + 2N_R - 1) + 1 \right) - 1, \tag{17}$$

$$\Omega_{\text{HC-TAS}} = N_T(4N_R - 1) + N_T N_s - \frac{N_s(N_s + 1)}{2} + C_{N_s}^{N_t} \times \left(\frac{2^{2\varepsilon} - 2^{\varepsilon}}{2} \times (8N_R N_t + 2N_R - 1) + 1 \right) - 1. \tag{18}$$

Here, $\varepsilon = (N_t - 1) + log_2(M)$ is the spectral efficiency of VASM.

For FNN-TAS, since the training process is conducted offline, the computational complexity of FNN-TAS is the sum of the flops from the data preprocessing step on real channel and the flops from the FNN prediction phase. However, the feature extraction process requires a relatively small number of flops compared to the prediction phase, and thus its computational cost can be neglected. Assuming the FNN has L hidden layers, with the number of neurons in each layer denoted as $[n_1, n_2, ..., N_L]$ the computational complexity of the prediction phase in the FNN is the total number of dot-product operations and activations across each layer. The total number of flops for the entire FNN is given by [25]:

$$\text{flop}_{\text{FNN}} = n_{\text{in}} n_1 + n_1 + \sum_{l=1}^{L-1} (n_l \; n_{l+1} + n_{l+1}) + n_L n_{\text{out}} + n_{\text{out}}, \tag{19}$$

where n_{in} and n_{out} are the number of neurons in the input and output layers of the FNN, respectively.

Since different feature extraction techniques lead to varying computational complexities for the data preprocessing step and input size (n_{in}), we calculate the computational complexity of FNN-TAS for the five proposed feature extraction methods as follows:

$$\Omega_{\text{FNN-TAS (R \& I)}} = 2N_R N_T n_1 + n_1 + \sum_{l=1}^{L-1} (n_l n_{l+1} + n_{l+1}) + n_L n_{\text{out}} + n_{\text{out}}, \tag{20}$$

$$\Omega_{\text{FNN-TAS (Mag)}} = N_R N_T n_1 + n_1 + \sum_{l=1}^{L-1} (n_l n_{l+1} + n_{l+1}) + n_L n_{\text{out}} + n_{\text{out}}, \tag{21}$$

$$\Omega_{\text{FNN-TAS (Phase)}} = N_R N_T n_1 + n_1 + \sum_{l=1}^{L-1} (n_l n_{l+1} + n_{l+1}) + n_L n_{\text{out}} + n_{\text{out}}, \tag{22}$$

$$\Omega_{\text{FNN-TAS (Norm)}} = N_T n_1 + n_1 + \sum_{l=1}^{L-1} (n_l n_{l+1} + n_{l+1}) + n_L n_{\text{out}} + n_{\text{out}}, \tag{23}$$

$$\Omega_{\text{FNN-TAS (Corr)}} = {N_T}^2 n_1 + n_1 + \sum_{l=1}^{L-1} (n_l n_{l+1} + n_{l+1}) + n_L n_{\text{out}} + n_{\text{out}}, \tag{24}$$

Table I
SIMULATION PARAMETERS FOR THE SYSTEM.

Parameter	Value
Total number of TAs	$N_T = 10$
Number of TAs selected	$N_t = 3$
Number of temporary TAs in HC-TAS	$N_s = 4, 5, 6$
Number of RAs	$N_R = 4$
APM modulation level	$M = 4$ (QPSK)

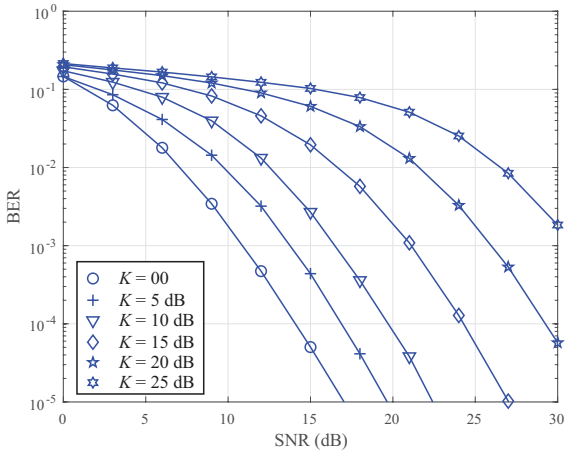


Figure 4. BER performance of the VASM system with Ran-TAS versus SNR for $K = 0 : 25$ dB.

Equations (20) through (24) clearly show that the computational complexity of the proposed FNN-TAS scheme depends only on the total number of TAs, the number of RAs, and the architecture of the neural network. Importantly, it is unaffected by both the count of selected TAs and the modulation order of APM.

IV. SIMULATION RESULTS AND DISCUSSIONS

This section presents simulations of the VASM system, employing the previously discussed TAS techniques under various Rician channel coefficients, to assess the effectiveness of the proposed FNN-TAS approach. For clarity, the simulation parameters used are listed in Table I and II. Ran-TAS refers to the method of randomly selecting N_t TAs from a total of N_T TAs.

1. Investigation of the impact of Rician factor on BER performance of VASM systems

It can be observed from Fig.4 that the system's BER performance declines with increasing K-factor. The reason is that, with a higher K-factor, the LoS component becomes dominant, causing the differences between the channel paths to become smaller. As a result, the receiver has more difficulty distinguishing which TAs transmitted the signal.

Table II
TRAINING HYPERPARAMETERS OF THE FNN

Hyperparameter	Value
Sample size (Q)	10^5
Holdout ratio	80-20%
Optimizer	Adam
Activation function	ReLU
Max epochs	500
Mini batch size	64
Initial learn rate	0.00015

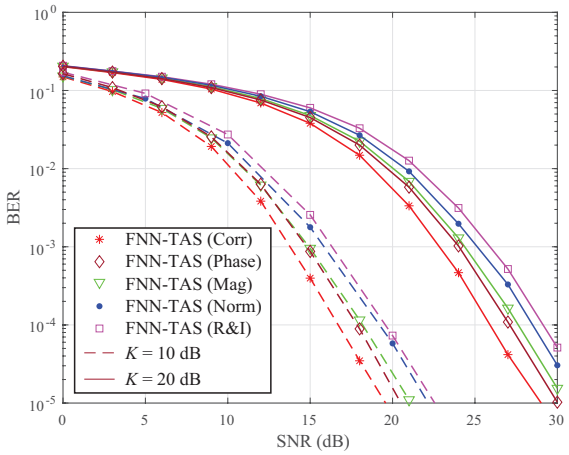


Figure 5. Comparison of the efficiency of feature extraction techniques through the BER performance of the VASM system employing FNN-TAS.

This reduces the accuracy of TAs index detection, which, in turn, leads to an increased BER. These results are fully consistent with those reported in [21–23].

2. Assessment of the effectiveness of feature extraction techniques

The paper extracted five important channel features to create the training dataset. However, it is unclear which feature will yield the best performance. Therefore, in this subsection, we evaluate the BER performance achieved by FNN-TAS using five datasets generated by extracting these five channel features. For this analysis, we employ an FNN (64/32/16) with $K = 10$ dB and $K = 20$ dB. The results in Fig. 5 indicate that FNN-TAS (Corr) achieves the best BER performance, followed by FNN-TAS (Phase), FNN-TAS (Mag), FNN-TAS (Norm), and finally, FNN-TAS (R&I). This demonstrates that the correlation feature among the columns of the channel matrix contains valuable information that enhances the learning capability of the FNN more effectively than the other features. Based on these results, we will focus on using dataset extracted by this feature in subsequent simulations.

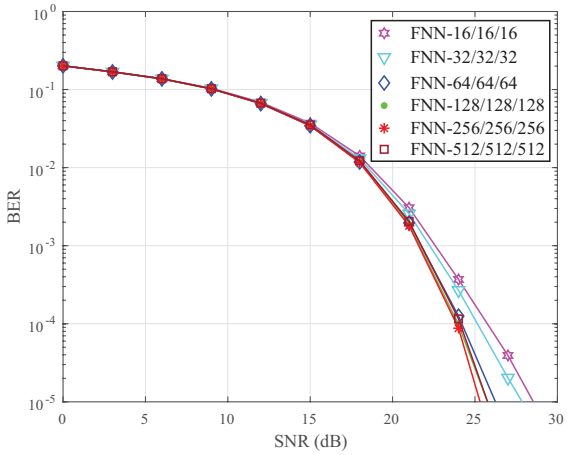


Figure 6. FNN-TAS BER performance in VASM system for different hidden layer setups at $K = 20$ dB.

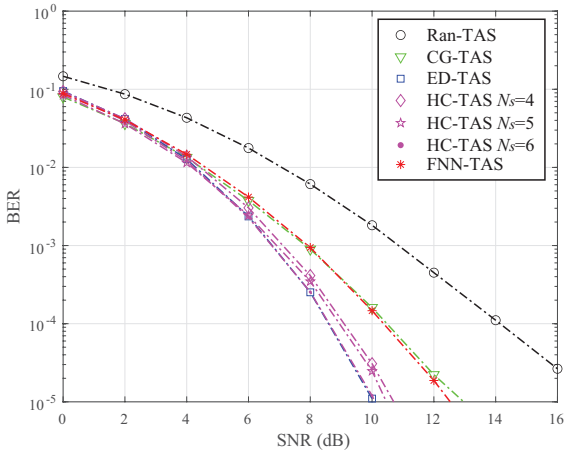


Figure 7. Comparison of BER performance enhancement achieved by TAS methods in VASM system at $K = 0$.

3. Evaluation of BER performance of FNN-TAS with different network configurations

Figure 6 illustrates the BER performance results of the FNN-TAS-VASM system, where neural networks with different numbers of hidden neurons share a common dataset extracted from correlation features between the columns of the channel matrix, with $K = 20$ dB. It is evident that as the number of hidden neurons increases, the BER performance improves significantly. However, when the network configuration changes from 256/256/256 to 512/512/512, the BER performance not only fails to improve but even degrades, whereas the computational complexity increases significantly. Therefore, the FNN (256/256/256) configuration can be considered an optimal choice for subsequent simulations.

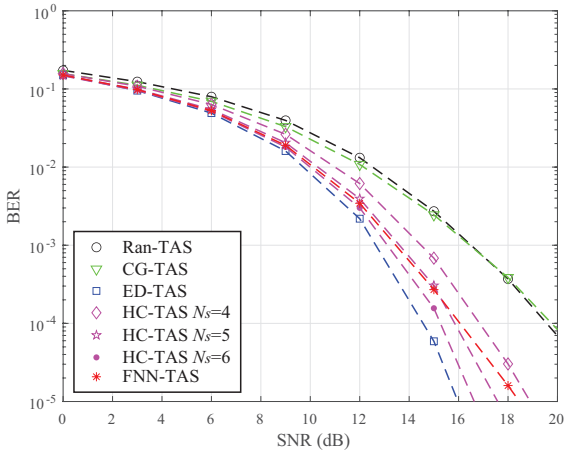


Figure 8. Comparison of BER performance for VASM systems employing various TAS techniques at K=10 dB.

4. BER performance evaluation of TAS-VASM systems under varying Rician factors

In Figs. 7, 8, and 9, we compare the BER performance of the VASM system achieved by the proposed FNN-TAS (256/256/256) and traditional TAS selection methods at K-factors of 0 (Rayleigh channel), 10 dB, and 20 dB. The results show that as the K-factor increases, the system's BER performance deteriorates for all TAS methods. However, the introduced FNN-TAS consistently surpasses CG-TAS, Ran-TAS, and even HC-TAS ($N_s = 4$ and $K = 10$ dB), with a lower rate of degradation. Notably, in Fig. 9, at the higher K-factor of 20 dB, the BER performance of FNN-TAS surpasses HC-TAS ($N_s = 4, 5, 6$) and nearly matches the optimal performance of ED-TAS.

This result clearly demonstrates that FNN-TAS performs effectively in Rician channel conditions with high Rician factors. The reason is that under high Rician factor conditions, the transmission channel is relatively stable, facilitating feature extraction and learning from the characteristics by the FNN. Consequently, higher performance is achieved.

5. Computational complexity comparison

Based on expressions from (16) to (24), along with the system parameters outlined above and the FNN (256/256/256), we have determined the computational complexity of the TAS algorithms, as illustrated in Fig. 10. The various feature extraction techniques result in disparate levels of computational complexity, albeit the differences are not significant. Notably, the FNN-TAS approach requires substantially less computational burden in comparison with ED-TAS and is comparable to HC-TAS.

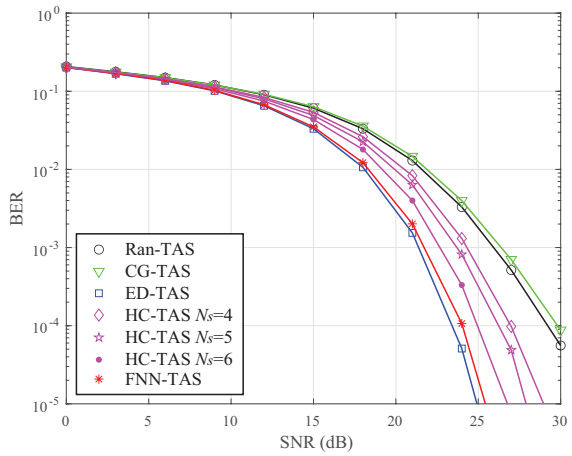


Figure 9. BER performance comparison of VASM systems using different TAS approaches at K=20 dB.

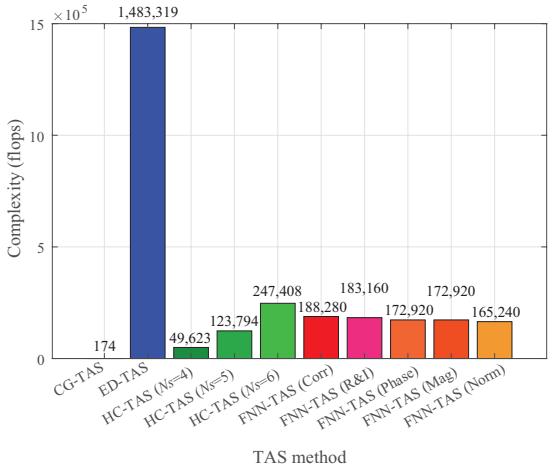


Figure 10. Computational complexity of TAS methods.

These results suggest that FNN-TAS is a favorable option for TAS in VASM systems under Rician fading channels, as it balances computational complexity with significant improvements in BER performance.

V. Conclusion and Future Works

This paper proposed the FNN-TAS scheme for VASM systems under Rician fading channels, achieving low complexity while maintaining superior BER performance. The proposed scheme demonstrates notable advantages in Rician channels with a dominant LoS component, providing valuable insights into TAS design tailored to channel conditions. Performance evaluations indicate that correlation-based features between channel matrix columns are the most informative for FNN training. Additionally, an optimal configuration with 256 neurons per hidden layer achieves a favorable trade-off between computational complexity and classification accuracy. The proposed FNN-TAS outperforms conventional HC-TAS and CG-TAS while closely approximating ED-TAS performance at a significantly reduced computational cost.

In future research, we will focus on extending FNN-TAS to low-latency applications, such as intelligent transportation and UAV communications. Furthermore, the potential of advanced neural network architectures for TAS in multi-antenna systems will be explored to enhance both efficiency and adaptability.

References

[1] M. N. A. Siddiky, M. E. Rahman, M. S. Uzzal, and H. M. D. Kabir, "A comprehensive exploration of 6G wireless communication technologies," *Computers*, vol. 14, no. 1, Jan. 2025.

[2] Z. Wang, J. Zhang, H. Du, D. Niyato, S. Cui, B. Ai, M. Debbah, K. B. Letaief, and H. V. Poor, "A tutorial on extremely large-scale MIMO for 6G: Fundamentals, signal processing, and applications," *IEEE Communications Surveys and Tutorials*, vol. 26, no. 3, pp. 1560–1605, Jan. 2024.

[3] R. Y. Mesleh, H. Haas, S. Sinanovic, C. W. Ahn, and S. Yun, "Spatial modulation," *IEEE Transactions on Vehicular Technology*, vol. 57, no. 4, pp. 2228–2241, Jul. 2008.

[4] M. Wen, B. Zheng, K. J. Kim, M. Di Renzo, T. A. Tsiftsis, K.-C. Chen, and N. Al-Dhahir, "A survey on spatial modulation in emerging wireless systems: Research progresses and applications," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 9, pp. 1949–1972, Sep. 2019.

[5] A. Younis, N. Serafimovski, R. Mesleh, and H. Haas, "Generalised spatial modulation," in *2010 Conference Record of the Forty Fourth Asilomar Conference on Signals, Systems and Computers*, Apr. 2010, pp. 1498–1502.

[6] R. Mesleh, S. S. Ikki, and H. M. Aggoune, "Quadrature spatial modulation," *IEEE Transactions on Vehicular Technology*, vol. 64, no. 6, pp. 2738–2742, Jun. 2015.

[7] O. Osman, "Variable active antenna spatial modulation," *IET Microwaves, Antennas and Propagation*, vol. 9, Oct. 2015.

[8] Y. Li, X. Lei, Y. Xiao, X. Zhou, and H. Niu, "Variable active pre-coding aided spatial modulation," in *2019 IEEE 19th International Conference on Communication Technology (ICCT)*. IEEE, Otc. 2019, pp. 670–673.

[9] S. Gadhai and R. Budhiraja, "Joint-mapping-based variable active antenna spatial modulation," *IEEE Wireless Communications Letters*, vol. 9, no. 10, pp. 1668–1672, Oct. 2020.

[10] T. V. Vinh, P. T. Hiep, and N. T. Phuong, "Combined variable active antenna spatial modulation and NOMA to enhance spectral efficiency for multiple users MIMO systems," in *2022 International Conference on Advanced Technologies for Communications (ATC)*, Nov. 2022, pp. 29–34.

[11] F. Zhu, H. Hai, Y. Peng, J. Hou, and X.-Q. Jiang, "Extended variable active antenna generalized spatial modulation," *IEEE Wireless Communications Letters*, vol. 13, no. 2, pp. 265–269, Feb. 2024.

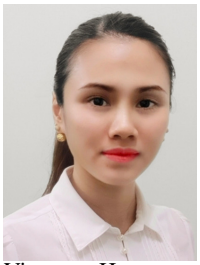
[12] J. Zheng, J. Zhang, H. Du, D. Niyato, B. Ai, M. Debbah, and K. B. Letaief, "Mobile cell-free massive MIMO: Challenges, solutions, and future directions," *IEEE Wireless Communications*, Feb. 2024.

[13] R. Rajashekar, K. Hari, and L. Hanzo, "Antenna selection in spatial modulation systems," *IEEE Communications Letters*, vol. 17, no. 3, pp. 521–524, Jan. 2013.

[14] P. Yang, Y. Xiao, Y. L. Guan, S. Li, and L. Hanzo, "Transmit antenna selection for multiple-input multiple-output spatial modulation systems," *IEEE Transactions on Communications*, vol. 64, no. 5, pp. 2035–2048, Mar. 2016.
[15] R. Zhu, H. Chen, Y. Xiao, X. Ji, W. Tang, and Y. Zhao, "Reduced-complexity transmit antenna selection for spatial modulation," in *2020 IEEE 20th International Conference on Communication Technology (ICCT)*, Oct. 2020, pp. 888–892.
[16] G. Altın, "A novel and low-complexity algorithm of EDAS for spatial modulation systems," *IEEE Communications Letters*, vol. 24, no. 12, pp. 2922–2925, Aug. 2020.
[17] T. V. Vinh, A. H. Phan, N. T. Thanh, and N. T. Phuong, "Proposal of a hierarchical combination of CG-TAS and ED-TAS to improve BER performance at low complexity for VASM systems," in *2023 12th International Conference on Control, Automation and Information Sciences (ICCAIS)*. IEEE, Jan. 2024, pp. 448–453.
[18] T. V. Vinh, N. H. Minh, P. T. Hiep, and N. T. Phuong, "Transmit antenna selection for multi-user VASM systems: Simplicity and fairness," *AEU-International Journal of Electronics and Communications*, vol. 170, p. 154842, Oct. 2023.
[19] K. X. Thuc, T. V. Vinh, P. L. Nguyen, T. Van Luyen, H. M. Kha, and N. T. Phuong, "Investigation of transmit antenna selection for MU-VASM systems over correlated channels," in *International Conference on Ad Hoc Networks*. Springer, Mar. 2024, pp. 112–124.
[20] W. C. Jakes and D. C. Cox, *Microwave mobile communications*. Wiley-IEEE press, 1994.
[21] M. Xie, X. Yu, J. Chen, and T. Teng, "BER performance of unmanned aerial vehicle assisted spatial modulation system in Rician channels," *Physical Communication*, vol. 49, p. 101471, Dec. 2021.
[22] A. Kabil, A. Bendary, A. F. Darwesh, and H. Dahshan, "Spatial index modulation in fading channels: A fair comparison approach," in *2024 14th International Conference on Electrical Engineering (ICEENG)*, Jun. 2024, pp. 300–304.
[23] S. Shrestha, S. Naser, L. Bariah, S. Muhaidat, P. C. Sofotasios, H. Elgala, and E. Damiani, "Autoencoder-based spatial modulation for the next generation of wireless networks," *IEEE Internet of Things Journal*, vol. 11, no. 22, pp. 36 322–36 334, Nov. 2024.
[24] Y. Zhang, J. Wang, X. Wang, Y. Xue, and J. Song, "Efficient selection on spatial modulation antennas: Learning or boosting," *IEEE Wireless Communications Letters*, vol. 9, no. 8, pp. 1249–1252, Apr. 2020.
[25] A. Mohamed, Z. Bai, F.-P. Oloruntomilayo, K. Pang, Y. Yang, D. Zhou, and K. S. Kwak, "Low complexity deep neural network based transmit antenna selection and signal detection in SM-MIMO system," *Digital Signal Processing*, vol. 130, p. 103708, Sep. 2022.
[26] G. Altın and İ. A. Arslan, "Joint transmit and receive antenna selection for spatial modulation systems using deep learning," *IEEE Communications Letters*, vol. 26, no. 9, pp. 2077–2080, Jun. 2022.
[27] H. K. Jadhav and V. B. Kumaravelu, "Deep learning-assisted transmit antenna classifiers for fully generalized spatial modulation: Online efficiency replaces offline complexity," *Applied Sciences*, vol. 13, no. 8, p. 5134, Apr. 2023.
[28] F. Zhu, H. Hai, and X.-Q. Jiang, "Efficient antenna selection for adaptive enhanced spatial modulation: A deep neural network approach," *IEEE Communications Letters*, vol. 27, no. 5, pp. 1352–1356, May. 2023.
[29] H. Zhang and N. Yang, *Deep Learning in Wireless Communications*. Springer, 2025.
[30] A. Bhowal and R. S. Kshetrimayum, *Advanced spatial modulation systems*. Springer, 2021.

Tran Viet Vinh received a B.S. degree in Telecommunication Technological Command at Telecomunications University, Vietnam, in 2011, and an MS degree in Electronics Engineering from Le Quy Don Technical University, Vietnam, in 2017. He is currently studying for a Ph.D degree at Le Quy Don Technical University, Vietnam. His research interests include MIMO systems, spatial modulation, and antenna selection, deep learning for wireless communications.
E-mail: tranvietvinhsqtt@lqdtu.edu.vn

Pham Thanh Hiep received a B.E. degree in Communications Engineering from the National Defense Academy, Japan, in 2005; and received the M.E. and Ph.D. degrees in Physics, Electrical and Computer Engineering from Yokohama National University, Japan, in 2009 and 2012, respectively. He worked as an associate researcher at Yokohama National University, Yokohama, Japan, from 2012 to 2015. Now, he is a lecturer at Le Quy Don Technical University, Hanoi, Vietnam. His research interests lie in the area of wireless communication technologies and signal processing.
E-mail: thanhhiep@lqdtu.edu.vn



Nguyen Thu Phuong received the B.S, M.S and PhD degrees from Le Quy Don Technical University, Vietnam in 2008, 2012 and 2016, respectively. She is now a lecturer at Faculty of Radio-Electronics Engineering, and a significant member of the Advanced Wireless Communication Group, Le Quy Don Technical University, Hanoi, Vietnam. Her research interests are in the area of emerging technologies for future wireless communications: Energy harvesting, Non-orthogonal multiple access (NOMA), space-time processing, space-time coding, spatial modulation and index modulation, and massive MIMO systems.
E-mail: phuong.nt@lqdtu.edu.vn

viTBI-BERT: A Vietnamese Language Model for Prediction of Traumatic Brain Injury

Duc-Khiem Doan[1], Thanh Hai Tran[1], Trung-Kien Tran[2], Thi-Lan Le[1], Hai Vu[1], Huu-Khanh Nguyen[3], Van Mao Can[4], Thanh Bac Nguyen[3]

[1] School of Electrical and Electronic Engineering, Hanoi University of Science and Technology, Hanoi, Vietnam
[2] Military Insitute of Science and Technology, Hanoi, Vietnam
[3] Military Hospital 103, Hanoi, Vietnam
[4] Vietnam Military Medical University, Hanoi, Vietnam

Correspondence: Thanh-Hai Tran, email: hai.tranthithanh1@hust.edu.vn
Communication: received 12 Sep 2024, revised 28 Oct 2024, accepted 04 Mar 2025
DOI: 10.32913/mic-ict-research.v2025.n1.1303

Abstract: Artificial Intelligence (AI) is increasingly utilized for disease prediction by analyzing large and complex datasets. Traditionally, AI-based traumatic brain injury (TBI) prediction relies on multimodal data, including structured information like test results and unstructured data such as CT and/or MRI scans. However, physician conclusions in text form at admission and discharge represent an underutilized source of valuable insights into a patient's condition. This paper proposes a novel approach to classifying TBI severity using these physicians' written conclusions. We introduce viTBI-BERT, a model built on the ViHealthBERT backbone, incorporating pre-processing techniques such as acronym normalization, special character removal, word segmentation, and textual data augmentation. Following pre-processing, the data is passed through the pre-trained ViHealthBERT model, augmented with a fully connected layer and a softmax layer for classification. Evaluated on three sets of medical notes from a self-collected dataset of 503 Vietnamese patients, viTBI-BERT achieved a sensitivity of 71% in classifying four levels of injury severity. These findings demonstrate the potential of language-based analysis, which, when integrated with structured clinical and subclinical data, could further improve TBI classification outcomes.

Keywords: *Traumatic Brain Injury, Large Language Model, Transformer, Classification.*

I. INTRODUCTION

Traumatic brain injury (TBI) is a critical medical condition resulting from external forces impacting the head, leading to varying degrees of brain dysfunction. TBI can be caused by various incidents such as traffic accidents, falls, and sports injuries, with consequences ranging from mild concussions to severe brain damage. TBI significantly influences the patient's quality of life and, in some cases, leads to long-term disability or death.

Current diagnostic and treatment procedures for TBI involve a combination of clinical examinations, imaging techniques such as CT scans or MRIs, and continuous monitoring of the patient's neurological status. However, the overload situation and the limited experience of healthcare professionals, particularly at remote medical facilities, may cause errors in prognosis and treatment, exacerbate patient outcomes, and increase healthcare costs.

In recent years, there has been a growing interest in applying AI models to enhance TBI diagnosis. Most existing AI approaches have focused on analyzing structured clinical information [1]. Some recent works focused on the medical analysis of CT (Computerized Tomography) or MRI (Magnetic Resonance Imaging) brain images [2]. While these models have shown promise in detecting abnormalities in brain images, they often do not provide comprehensive diagnostic conclusions, particularly regarding the severity of the injury.

Accurate conclusions about the severity of TBI are crucial, as they directly influence treatment decisions and prognoses. For example, a conclusion like "Máu tụ dưới màng cứng mạn tính bán cầu phải" ("Chronic subdural hematoma, right hemisphere" in English) can indicate a severe level of TBI. Meanwhile, a conclusion like "Máu tụ dưới màng cứng trán đỉnh phải nghiện rượu" ("Subdural hematoma, right frontal-parietal, alcoholic" in English) may correspond to a moderate TBI case. These conclusions typically appear at the end of the medical report. To our knowledge, there is a gap in the research concerning the analysis of diagnostic conclusions for determining the severity of TBI. Existing studies have not explored the integration of textual information from medical reports into AI models for generating comprehensive diagnostic conclusions. On one hand, no dataset containing the conclusions of TBI patients

is available for research purposes. On the other hand, if such a dataset exists, the linguistic characteristics of the reports must be taken into consideration.

This paper addresses the existing gap by investigating text analysis models and their application in the field of traumatic brain injury diagnosis. We propose a novel approach that categorizes TBI into four severity levels: mild, moderate, severe, and critical, based on the analysis of Vietnamese textual conclusions in medical reports. This study represents the first effort to employ text analysis for TBI classification, aiming to enhance diagnostic accuracy and improve patient outcomes. The proposed framework, called **viTBI-BERT**, consists of two main blocks: a preprocessing block that includes acronym normalization, special character removal, word segmentation, and data augmentation; and a classification block composed of a pretrained ViHealthBERT model [3], followed by an additional fully connected layer and a softmax layer. The method is validated on three sets of medical notes extracted from our self-collected dataset, **TBI_MH**, showing promising results.

In summary, the contributions of this paper are threefold.

- First, we introduce an original idea for TBI outcome prediction using Vietnamese medical notes, which has not been investigated in the literature.
- Second, we propose a framework, **viTBI-BERT**, which leverages ViHealthBERT to analyze TBI Vietnamese medical notes, incorporating specific preprocessing steps.
- Finally, we validate our proposed framework on three sets of TBI medical notes extracted from our self-collected dataset.

The remainder of this paper is organized as follows. In section II, we present some existing works for the automatic prognosis of traumatic brain injury using machine learning. In section III, we present the general framework, and a detailed description of each component is presented. Section IV presents the dataset and experimental results. We conclude and give some ideas for future works in Section V.

II. RELATED WORKS

1. Machine learning for TBI outcome prediction

Recent studies have explored using ML to predict TBI outcomes. One study analyzed data from 1653 TBI patients using algorithms like random forest (RF) and decision tree (DT)[1], identifying key predictors such as the motor component of the Glasgow Coma Scale (GCS), pupillary condition, and age. RF was found to be the best for short-term mortality prediction, while a generalized linear model (GLM) performed best for long-term survival prediction. The study also highlights the potential of deep learning (DL) for handling high-dimensional data but recommends using simple models to avoid overfitting.

In 2020, Matsuo et al., similarly, [4] evaluated the effectiveness of nine machine learning algorithms in predicting outcomes for traumatic brain injury (TBI) patients. By analyzing clinical data from 232 patients, the research identifies the random forest and ridge regression models as the most effective for predicting poor outcomes and mortality. Key factors influencing predictions include age, the Glasgow Coma Scale, and specific blood markers. The findings suggest that modern machine learning techniques hold promise for improving TBI prognosis, though further validation is needed. Bruschetta et al. compared traditional regression models with various machine learning (ML) algorithms for predicting clinical outcomes in traumatic brain injury (TBI) patients [5]. The study evaluates the performance of a Classical Linear Regression Model (LM) against a Support Vector Machine (SVM), k-nearest Neighbors (k-NN), Naïve Bayes (NB), Decision Tree (DT), and an ensemble ML approach. The findings indicate that ML algorithms, including ensemble methods, do not significantly outperform traditional regression models when using a limited set of predictor variables. Naïve Bayes showed the poorest performance among the models tested. The study underscores the importance of rigorous comparisons between traditional and ML approaches in TBI prognosis and suggests further validation on more diverse datasets incorporating dynamic predictors like neuroimaging data.

2. Deep learning for TBI outcome prediction

The study by Adil et al. (2020) focuses on developing a deep learning model to predict outcomes after TBI, particularly in low- and middle-income countries (LMICs) [6]. Using clinical data from Uganda, the authors compared the performance of deep neural networks, shallow neural networks, and elastic-net logistic regression. Their results showed that deep learning outperformed traditional methods in specific metrics like AUROC and partial AUPRC, highlighting the potential of advanced algorithms in resource-limited settings, though optimal algorithm choice depends on the clinical context. Matthew et al. proposed a prognostic model combining deep learning from head CT scans with clinical data to predict long-term outcomes in patients with severe traumatic brain injury (sTBI) [7]. In a cohort of 537 patients, the fusion model demonstrated superior performance over the established IMPACT model in predicting mortality and unfavorable outcomes on internal

data. However, in external validation with 220 patients, the fusion model did not significantly outperform the IMPACT model. Despite the mixed results, the study underscores the promise of integrating imaging and clinical information for sTBI prognostication. Another study employs a deep learning framework to evaluate the sensitivity of a CNN model in detecting anatomical changes in brain MRI scans related to traumatic brain injury (TBI) outcomes [2]. The objective is to validate the model's clinical relevance using a dataset from the TRACK-TBI study, which includes 203 patients, and to analyze the model's performance across varying severity subgroups. Results indicate that the model can identify latent predictive information from MR images that is not captured in ground truth labels, suggesting promise in classifying TBI-related MR images.

3. Language models for disease prediction

Large language models (LLMs) have demonstrated impressive capabilities not only in the domain of natural language processing but also in various other tasks. In medical analysis, some early works have applied LLMs to enhance the performance of existing deep-learning models within a single modality. We present recent significant works that apply LLMs to address medical-related problems.

In [8], raw clinical texts have been refined and augmented by an LLM then combined with embedding computed from tabular data from the Electrical Medical Report of Patients. The final model, namely called CKLE, has been trained for two health event prediction tasks in the field of cardiology, heart failure prediction and hypertension prediction. Health-LLM incorporated health records and medical knowledge into a large model to ask relevant questions for a given disease [9]. Med-BERT is an adaptation of the BERT framework originally developed for the text domain to the structured Electrical Health Records domain [10]. Hegselmann et al. proposed TabLLM for the classification task of tabular data [11]. Each feature in the tabular data was converted into a natural language string using a manual template, table-to-text, or LLM. Li et al. introduced LViT that combined text description with images as input for segmentation task [12]. In LViT, medical text annotation is incorporated to compensate for the quality deficiency in image data. Le et al. conducted an evaluation of TBI 6-month outcome prediction using ChatGPT4 model [13] and showed that it was less performant than the conventional IMPACT model and that physicians.

Although there have been some attempts to use large language models (LLMs) for disease prediction, to the best of our knowledge, no studies have yet explored the use of LLMs with medical notes from CT images of TBI patients. In our work, we will utilize both pre- and post-CT medical notes from doctors, as well as a combination of these notes, to predict the outcomes of TBI. We believe this is the first initiative to explore the potential of LLMs for TBI prediction using medical text.

III. PROPOSED METHOD FOR TBI PROGNOSIS

1. General framework

As aforementioned, our work focuses on analyzing the medical notes of doctors from CT scans of TBI patients and outputs one among four severity levels of TBI: mild, moderate, severe, and critical. In this study, we propose a framework for the prognosis of TBI using the medical text of the doctors from CT scans as illustrated in Fig. 1. It is composed of the following steps:

- *Data splitting*: First, raw data (i.e. medical notes of physicians from CT scans of TBI patients) are split into two subsets, one for training and one for testing. In our work, we employ K folds (with K = 10) so nine folds will be used for training and one fold will be used for testing.
- *Pre-processing*: The raw medical notes must be pre-processed before being input into the classification model. We deploy two techniques: acronym normalization and special character removal to convert raw text to normalized text. This step is applied for both training and testing datasets.
- *Data augmentation*: To enrich the training set, we implement various techniques to create new medical texts with the same meaning. This step is applied only to the training dataset.
- *Classifier training*: We utilize the pre-train ViHealth-BERT model to fine-tune the viTBI-BERT using an augmented training dataset.
- *Inference*: The test samples are first pre-processed and then inputted into the trained viTBI-BERT to infer the severity level of the TBI.

Subsequently, we provide a detailed explanation of each of these steps, providing a comprehensive view of our methodology and the underlying rationale.

2. Pre-processing

a) Acronym normalization

In medical texts, acronyms are often used to quickly denote specific conditions, symptoms, or medical procedures. In our TBI dataset, a total of 77 acronyms have been found in medical notes. For example, "CTSN" is the acronym for "Chấn thương sọ não" (Traumatic Brain Injury), "MTTN" is the acronym for "Máu tụ trong não" ("intracerebral hemorrhage"), TNGT means "Tai nạn giao

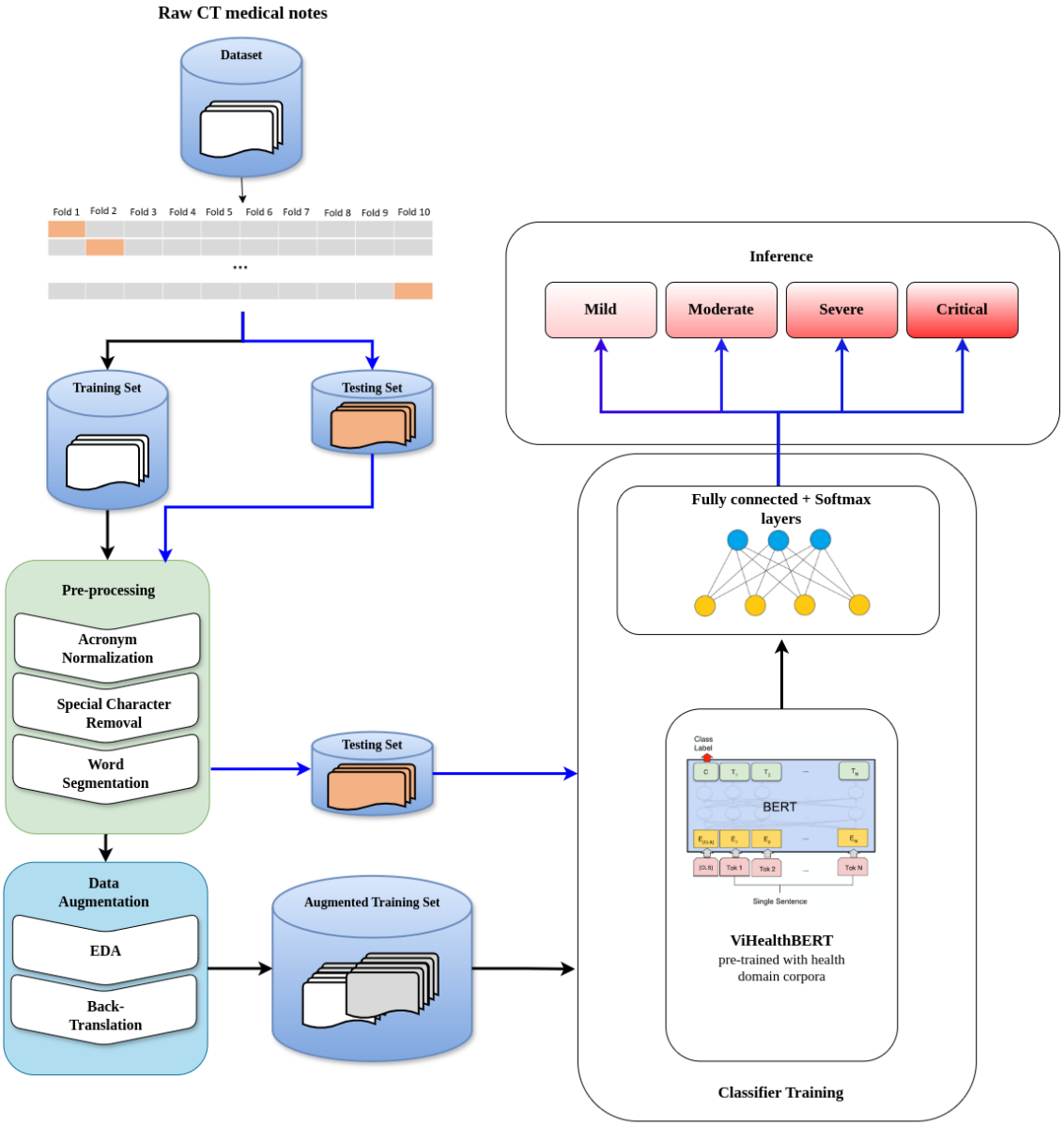


Figure 1. Proposed viTBI-BERT for outcome prediction after Traumatic Brain Injury from Vietnamese medical notes.

thông" ("traffic accident"). To ensure accurate interpretation and improve consistency in the data, all acronyms must be normalized.

In the medical text domain, Vo et al. proposed a method to detect abbreviation in Vietnamese clinical texts [14]. However, this method does not encompass all acronyms present in our dataset related to traumatic brain injury. Following the approach used in [3], we generate a dictionary of 77 acronyms. This involves extracting all noun phrases (NP) from the dataset using the *underthesea* POS tagger. Consequently, we manually decrypt the data by clarifying their meanings provided by expert doctors at 103 Military Hospital. For example, acronyms such as "CTSN" are expanded to "Chấn thương sọ não", and "MTMNC" is expanded to "Máu tụ ngoài màng cứng" (epidural hematoma).

b) Special characters removal

Machine learning models often struggle to interpret punctuation or special characters. In our dataset, the medical notes frequently contain various punctuation marks such as exclamation points, apostrophes, and commas [15]. To address this, we preprocess the medical notes by removing all special characters and punctuation, including ":", ",", ".", "/", "(", ")", "-", and ";".

c) Word segmentation

Word segmentation is a crucial pre-processing step in natural language processing (NLP), especially for languages like Vietnamese, which lack explicit word boundaries. In our workflow, we perform word segmentation using the RDRSegmenter, a component of the VNCoreNLP toolkit. RDRSegmenter, developed by [16], is a Vietnamese word segmentation tool that uses a rule-based approach to accurately separate text into meaningful units. This tool is part of VNCoreNLP, a comprehensive suite of NLP tools for Vietnamese developed by Vu et al. [17]. Since Minh et al. [3] used this method to pre-process the pre-training data and also recommended it for fine-tuning tasks, we use the same word segmenter for our downstream applications. By incorporating these methods, we enhance the quality and precision of our text processing pipeline, making it well-suited for classification tasks. In our experiment, we directly utilize the pre-trained RDRSegmenter for word segmentation.

Figure 2 illustrates an example of pre-precessing raw data "CĐ:CTSN:MTNMC+TKNS do tai nạn sinh hoạt G7". In this sentence, "CĐ" is an acronym for "Chấn động" ("concussion"), "CTSN" is an acronym for "Chấn thương sọ não" ("traumatic brain injury"), "MTNMC" is an acronym for "Máu tụ ngoài màng cứng" ("epidural hematoma"). "TKNS" means "Tụ khí nội sọ" ("intracranial pneumatoma"). "G7" means "giờ thứ 7" ("7th hour"). After normalizing all acronyms, we obtain "chấn động: chấn thương sọ não:máu tụ ngoài màng cứng+tụ khí nội sọ do tai nạn sinh hoạt giờ thứ 7" ("concussion: traumatic brain injury: epidural hematoma+intracranial pneumatoma due to domestic accident 7th hour"). This new data contains special characters and punctuation. We remove all of these and apply RDRSegmenter for word segmentation to finally achieve: "chấn_động chấn_thương sọ não máu tụ ngoài màng_cứng tụ khí nội sọ do tai_nạn sinh_hoạt giờ thứ 7".

3. Data augmentation

a) Augmentation methods

In image classification tasks, data augmentation enhances data diversity through techniques such as rotation, translation, scaling, and noise addition. Similarly, text classification employs augmentation to enrich text data, with numerous studies developing automated methods to generate and improve data quality. This approach improves model accuracy, particularly for small datasets such as ours.

We will first present the EDA (Easy Data Augmentation) techniques introduced by Wei and Zou [18] and apply them to Vietnamese medical texts. The techniques include:

- *Synonym Replacement (SR)*: This technique involves replacing non-stop words with randomly chosen synonyms to create a new sentence. For our experiments, we used the Vietnamese WordNet from [19] to find synonyms and a Vietnamese stopword dictionary to exclude stop words from the sentences.
- *Random Swap (RS)*: Swapping two non-stop words randomly n times.
- *Random Insert (RI*: Finding a random synonym of a non-stop word and inserting it randomly n times in the sentence.
- *Random Delete (RD)*: Removing each word in the sentence with a probability p.

The number of augmentations should be calibrated according to the dataset size, as an excessive amount of augmented data can result in overfitting. A notable limitation of these methods is their inability to maintain the contextual meaning of sentences. To overcome this, we investigate more advanced techniques that preserve the original sentence's meaning.

- *Back Translation (BT)*: This technique generates additional training samples by utilizing translators. It has been used by various research teams to improve translation models [20][21]. The process involves translating the original text into another language and then translating it back to the original language using a different translator. In this work, we employ the *mtranslate* library to implement the back translation method, utilizing English, French, and Russian as intermediate languages.

Table I presents the results of applying the Easy Data Augmentation (EDA) method to the sentence: "chấn_động não do tai_nạn giao_thông" (brain_injury due to traffic_accident). Similarly, Table II shows the results of back-translation applied to the sentence "chấn động: chấn thương sọ não máu tụ ngoài màng cứng tụ khí nội sọ do tai nạn sinh hoạt giờ thứ 7" using three different intermediate languages.

Table I
EXAMPLES OF EASY DATA AUGMENTATION METHODS

Augmented Sentence	Type
chấn_động do tai_nạn giao_thông (deleted "não")	RD
chấn_động **đầu_óc** do tai_nạn giao_thông	SR
não chấn_động do tai_nạn giao_thông	RS
chấn_động não do **đầu** tai_nạn giao_thông	RI

4. Classifier training

a) Model architecture

In our experiments, we employed monolingual domain-specific pre-trained BERT models, designed specifically

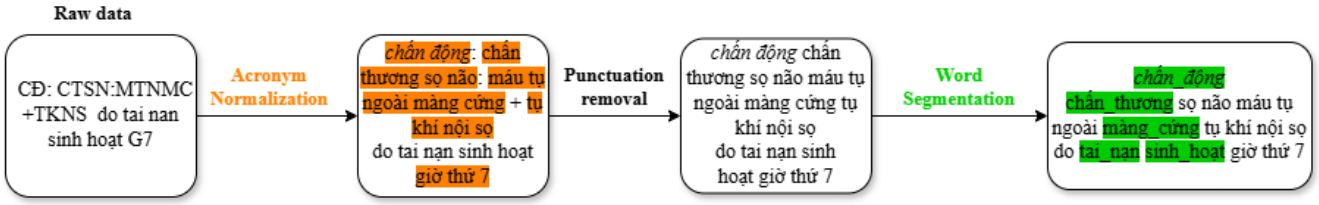


Figure 2. Example of pre-processing steps.

Table II
EXAMPLE OF DATA AUGMENTATION USING BACK-TRANSLATION WITH THREE LANGUAGES: ENGLISH, FRENCH AND RUSSIAN. WE ALSO ADDED THE TRANSLATION OF THE AUGMENTED DATA FROM VIETNAMESE TO ENGLISH USING GOOGLE TRANSLATOR.

Language	Back-Translated Sentence
English	Chấn động não: chấn thương sọ não, tụ máu ngoài màng cứng, tràn khí nội sọ do tai nạn gia đình, giờ thứ 7 (Concussion: traumatic brain injury, epidural hematoma, intracranial pneumothorax due to domestic accident, 7th hour)
French	Chấn động: chấn thương đầu, tụ máu ngoài màng cứng, tụ khí nội sọ do tai nạn hàng ngày vào giờ thứ 7 (Concussion: head trauma, epidural hematoma, intracranial emphysema due to daily accidents at 7 o'clock)
Russian	Chấn động: chấn thương sọ não, tụ máu ngoài màng cứng, tích tụ khí nội sọ do hậu quả của một sự kiện trong đời vào giờ thứ 7 (Concussion: traumatic brain injury, epidural hematoma, intracranial gas accumulation as a result of a life event in the 7th hour).

for Vietnamese healthcare text for the TBI outcome classification task based on medical notes. ViHealthBERT, a highly effective baseline model for this domain, was rigorously evaluated using various training strategies. It achieved state-of-the-art (SOTA) results across three downstream tasks: Named Entity Recognition (NER) for COVID-19 and ViMQ datasets, Acronym Disambiguation, and Summarization [3].

In this study, we used the word-level version of ViHealthBERT, which contains 135 million parameters and utilizes a word-level tokenizer. This model follows the same architecture as $\text{BERT}_{\text{base}}$ [22]. One of the key advantages of using BERT-based Transformers for sequence classification is their ability to capture the meaning and context of the input sequence more effectively than traditional neural networks. This capability stems from BERT's pretraining on a vast corpus of text, where it learns to predict missing words or sentences based on their surrounding context. This pretraining enables BERT to understand the relationships between different words and phrases within the input, resulting in a more informative representation of the sequence.

The pre-trained model consists of 12 encoder-only Transformer layers [23], with 768 hidden units and 12 attention heads. Each encoder layer has two main components: a Multi-Head Self-Attention mechanism and a Feedforward Neural Network. The attention mechanism allows the model to weigh the importance of different words in the sequence, capturing long-range dependencies. Mathematically, the output of the Multi-Head Attention for a single head can be expressed as:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad (1)$$

where Q, K, and V are the query, key, and value matrices, respectively, and d_k is the dimension of the key vectors. In practice, the model employs multiple heads, each with different learned parameters, allowing it to focus on various parts of the input.

After the attention mechanism, the output passes through a feedforward network, which typically consists of two linear transformations with a ReLU activation in between:

$$\text{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2 \quad (2)$$

where W_1, W_2, b_1, and b_2 are learned parameters.

As shown in Figure 1, we added an additional classifier head. This classifier head operates on the [CLS] token (Classifier token) in the last hidden layer output by the pretrained BERT model. The [CLS] token is a special token prepended to every input sequence, representing the entire sequence for classification tasks. The classifier head consists of a fully connected layer followed by a softmax activation function, which produces probabilities for each class label.

5. Inference

To gain deeper insights into the model's decisions, we employed Integrated Gradients (IG) [24], an explainable AI technique. This method computes attribution scores for each word in a clinical note, revealing which words most significantly influenced the model's predictions.

Formally, let $F : \mathbb{R}^n \to \mathbb{R}$ represent a deep neural network, where $F(x)$ is the output for input x. The attribution of each input feature x_i is given by:

$$\text{IG}_i(x) = (x_i - x_i') \times \int_0^1 \frac{\partial F(x' + \alpha(x - x'))}{\partial x_i} d\alpha \quad (3)$$

Here, x is the input, x' is the baseline (neutral or zero input), and α interpolates between x' and x. The gradient $\frac{\partial F(x)}{\partial x_i}$ is computed at points along this path. The integral can be approximated as:

$$\text{IG}_i(x) \approx (x_i - x_i') \times \frac{1}{m} \sum_{k=1}^{m} \frac{\partial F(x' + \frac{k}{m}(x - x'))}{\partial x_i} \quad (4)$$

IG satisfies two properties: **Sensitivity**, ensuring non-zero attributions when output changes are due to a single feature, and **Implementation Invariance**, guaranteeing identical attributions for functionally equivalent models.

IV. EXPERIMENTS

1. Dataset

In the literature, there is no dataset of medical notes of traumatic brain injury. As a result, to validate our proposed framework, we collected a dataset consisting of 503 patients treated at the Military Hospital 103 in Hanoi, Vietnam, in 2023. This dataset, referred to as **TBI_MH** (**T**raumatic **B**rain **I**njury collected at **M**ilitary **H**ospital), was carefully curated to exclude records with incomplete information, ensuring data accuracy. The patients had a mean age of 43.89 years with a standard deviation of 21.3 years. A significant portion of the cohort, approximately 76.09%, were male, as described in Table III and Figure 3.

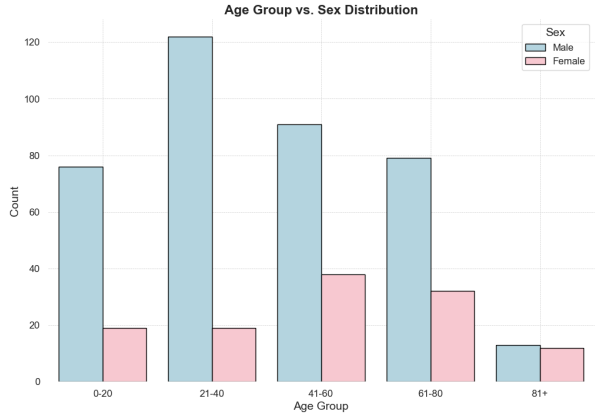


Figure 3. Age groups and sex distribution of patients in the TBI_MH dataset

Table III
AGE AND SEX DISTRIBUTION IN TBI_MH DATASET

Category	Subcategory	Percentage (%)
Age Group	0-20	18.92
	21-40	28.29
	41-60	25.70
	61-80	22.11
	81+	4.98
Sex	Male	76.10
	Female	23.90

The dataset includes a comprehensive range of patient data, covering demographic details, clinical indicators, laboratory results, and CT (computed tomography) imaging information. The demographic data encompasses age, sex, and key patient identifiers. Clinical data provides insights into health status, blood pressure, and temperature, along with details on traumatic brain injuries as documented in medical histories. The Glasgow Coma Scale (GCS) [25] was assessed during physical examinations, and the dataset also records related conditions such as stroke, diabetes, hypertension, and cardiovascular diseases. CT scan data plays a critical role in diagnosing traumatic brain injuries, particularly through the use of the Rotterdam score [26]. Additionally, the dataset includes laboratory test results, such as blood testing, serum biochemistry, electrolyte levels, and electrocardiogram readings.

In this research, our focus is on analyzing the pre-discharge and post-discharge medical notes provided by doctors to assess patient severity using AI-based natural language processing (NLP) techniques. Patient outcomes were categorized into four severity levels (1 = Mild, 2 = Moderate, 3 = Severe, 4 = Critical) by experienced clinicians and physicians, as illustrated in Table IV and Figure 4. Table V illustrates some examples of medical notes at pre-discharge and post-discharge for four severity levels.

Table IV
OUTCOMES OF THE TBI_MH DATASET

Severity Level	Description	Number of Patients
1	Mild	39
2	Moderate	265
3	Severe	155
4	Critical	44

Before using this dataset for research, we obtained the necessary approvals from the relevant organizational authorities to ensure compliance with established data usage protocols. The study's methodology strictly adhered to the guidelines and regulations governing research practices, and ethical clearance was obtained from the hospital for access to the dataset.

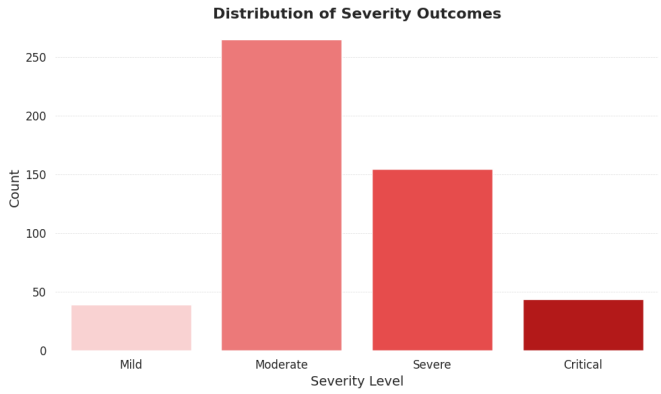


Figure 4. Outcomes distribution of TBI_MH Dataset

We consider three sets of medical notes extracted from the raw **TBI_MH** dataset as follows:

1) **PreMN_TBI_MH**: The first dataset includes the medical notes of doctors before the patient was admitted to the hospital.
2) **PostMN_TBI_MH**: The second dataset includes the medical notes of doctors after the patient was discharged from the hospital.
3) **MN_TBI_MH**: The third dataset is a combination of both the pre-admission and post-discharge medical notes.

Table VI provides a comprehensive summary of the statistics related to the length of notes across three datasets PostMN_TBI_MH, PreMN_TBI_MH, and MN_TBI_MH by severity levels (Mild, Moderate, Severe, and Critical). The metrics include the number of medical notes, along with descriptive statistics about their lengths: mean, standard deviation (Std), minimum (Min), 25th percentile (25%), median (50%), 75th percentile (75%), and maximum (Max). It is noted that the lengths of medical notes at pre-discharge and post-discharge are similar on average. However, the more severe the TBI, the longer the medical notes tend to be.

We separate the dataset into the training set and the testing set using K-Fold. Then the training set is augmented to re-balance the samples among classes. Figure 5 shows the distribution of original and augmented training sets.

2. Implementation details

a) Data Augmentation

According to Wei and Zou (2019) [18], for the SR, RI, and RS methods, the number of changed words n is calculated as $n = \alpha \times l$, where α is the percentage of words to be replaced and l is the sentence length. For the RD method, the deletion probability p equals α, with α defined by the user.

Table V
EXAMPLE OF PRE- AND POST-MN_TBI_MH SET WITH SEVERITY LEVELS

PreMN_TBI_MH	PostMN_TBI_MH	Severity
Xuất huyết dưới nhện liềm đại não gãy hở 13 dưới xương mác trái tụ máu dưới da quanh mi mắt phải do tai nạn giao thông (Subarachnoid hemorrhage, open fracture of the left fibula, subcutaneous hematoma around the right eyelid due to traffic accident)	Xuất huyết dưới nhện liềm đại não gãy kín 13d xương mác và vết thương phần mềm cẳng chân bên trái do tai nạn giao thông giờ thứ 3 đã khâu vết thương ngày thứ nhất (Subarachnoid hemorrhage, closed fracture of the 13d fibula and soft tissue injury of the left leg due to a traffic accident. The wound was sutured on the first day.)	Mild
Máu tụ dưới màng cứng trán đỉnh phải nghiện rượu (Subdural hematoma right frontal parietal alcoholic)	Máu tụ dưới màng cứng trán đỉnh phải đám mờ không thuần nhất ngoại vi 13 dưới phổi phải teo não tuổi già nghiện rượu (Right frontal subdural hematoma peripheral heterogeneous opacity 13 right lower lung cerebral atrophy senile alcoholism)	Moderate
Máu tụ dưới màng cứng mạn tính bán cầu phải (Chronic subdural hematoma right hemisphere)	Máu tụ dưới màng cứng mạn tính bán cầu phải đã phẫu thuật bơm rửa di lệch ổ máu tụ ngày thứ nhất (Chronic subdural hematoma of the right hemisphere was surgically discharged and displaced on the first day)	Severe
Chấn thương sọ não nặng máu tụ dưới màng cứng bán cầu trái xuất huyết dưới nhện rải rác 2 bán cầu vỡ xương đá phải do tai nạn giao thông g3 (Severe traumatic brain injury, left hemisphere subdural hematoma, scattered subarachnoid hemorrhage, bilateral hemisphere fracture of right petrous bone due to traffic accident g3)	Chấn thương sọ não nặng máu tụ dưới màng cứng bán cầu trái dập não xuất huyết trán và thái dương và đỉnh trái xuất huyết dưới nhện rải rác 2 bán cầu máu tụ ngoài màng cứng và vỡ xương thái dương đỉnh phải vỡ xương thái dương bên trái xương đá phải thân xương bướm do tai nạn giao thông viêm phổi do a baumanii toàn kháng (Severe traumatic brain injury, left hemisphere subdural hematoma, brain contusion, left frontal and temporal and parietal hemorrhage, scattered subarachnoid hemorrhage in both hemispheres, epidural hematoma and right temporal-parietal bone fracture, left temporal bone fracture, right petrous bone, sphenoid bone body due to traffic accident, pan-resistant A baumannii pneumonia)	Critical

Table VI
STATISTICS OF DIAGNOSIS LENGTH ACROSS DATASETS AND SEVERITY LEVELS

Dataset	Severity Level	Count	Mean	Std	Min	25%	50%	75%	Max
PostMN_TBI_MH	Mild	39	16.59	9.45	5	10	14	22.50	48
	Moderate	265	28.17	14.70	3	18	25	37	85
	Severe	155	35.37	18.34	7	22	30	48	141
	Critical	44	49.45	24.91	17	29.75	47	57.25	140
PreMN_TBI_MH	Mild	39	18.33	8.41	7	11	17	26	37
	Moderate	265	29.09	14.30	5	19	26	36	86
	Severe	155	34.47	14.96	6	22.50	31	45	79
	Critical	44	45.07	24.18	13	29.75	40.50	53.25	121
MN_TBI_MH	Mild	39	34.92	16.86	12	22	32	46	75
	Moderate	265	57.26	28.30	11	36	52	74	171
	Severe	155	69.85	32.29	22	44.50	61	94.50	220
	Critical	44	94.52	47.45	35	62.75	82.50	111.50	261

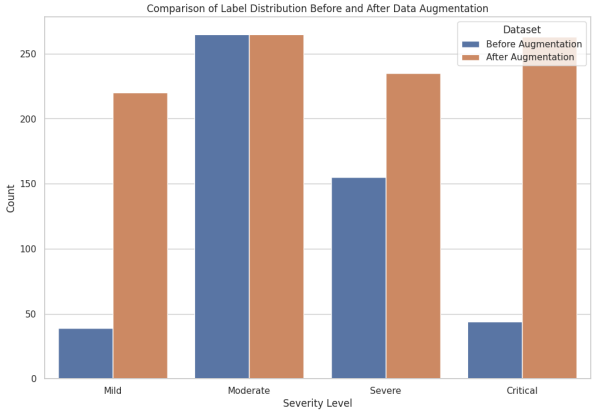


Figure 5. Distribution of original and augmented training dataset

In this experiment, we used the number of augmentations equal to 2 for each sentence. The augmentation techniques applied include synonym replacement, random insertion, random swap, and random deletion, with the following parameters: $\alpha_{\text{sr}} = 0.1$, $\alpha_{\text{ri}} = 0.1$, $\alpha_{\text{rs}} = 0.1$, and $\alpha_{\text{rd}} = 0.1$. Additionally, back-translation is applied using intermediate languages from a set of {English, French, Russian}. Figure 5 illustrates the distribution of a training fold before and after applying the aforementioned augmentation methods. It is noted that the original data across classes is quite imbalanced, with the first and fourth classes having fewer samples. After applying data augmentation, the number of samples in each class becomes more balanced.

b) Classifier Training

viTBI-BERT is implemented in Python. To track and fine-tune the model for optimal hyperparameters. After all, we use the hyperparameters set as described in Table VII below. The optimizer is AdamW, and the method to find hyperparameters is Bayesian. The model training process spans 40 epochs.

Table VII
HYPERPARAMETERS SETTING IN THE EXPERIMENTS.

Hyperparameter	Value
Batch Size	16
Dropout Rate	0.2
Learning Rate	0.00005
Epoch	40
Number layers	1
Pretrained Model	word-level

c) Inference

We applied the Integrated Gradients (IG) method to evaluate the contribution of individual features to the model's predictions. As the baseline input, we used a zero embedding vector, consistent with the approach described in [24], which is widely adopted in text classification tasks. For calculating and visualizing the results, we leveraged the Captum library [27].

3. Evaluation Metrics

To evaluate the performance of the viTBI-BERT, we use various metrics derived from the confusion matrix. This matrix is essential for assessing the performance of classifiers by including counts of True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN). The following formulas define the evaluation metrics, which help in determining the most suitable classification method.

*a) **Accuracy***

Accuracy measures the proportion of correct predictions among the total number of cases examined. It is given by (5):

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{5}$$

*b) F_1 **score***

The F_1 score is a harmonic mean of precision and recall, providing a single metric for a model's performance. It is defined as (6):

$$F_1 = \frac{2 \times TP}{2 \times TP + FP + FN} \quad (6)$$

c) *Precision*

Precision measures the proportion of true positive predictions among all positive predictions made by the model. It is calculated using (7):

$$\text{Precision} = \frac{TP}{TP + FP} \quad (7)$$

d) *Recall*

Recall, also known as sensitivity or the true positive rate, measures the model's ability to correctly identify true positives. It is given by (8):

$$\text{Recall} = \frac{TP}{TP + FN} \quad (8)$$

4. Experimental results

a) *Results on three datasets*

Table VIII illustrates the average performance of our proposed method on three sets: PreMN_TBI_MH, PostMN_TBI_MH, and MN_TBI_MH. In terms of Sensitivity (or Recall), viTBI-BERT achieved scores of 0.67, 0.68, and 0.71 on the three sets on average, respectively. It is reasonable that the sensitivity on the PostMN_TBI_MH set is higher than on the PreMN_TBI_MH set because the information about the disease is more detailed and precise when recorded after discharge, especially in cases involving severe and critical patients. When combining both pre-discharge and post-discharge medical notes, the sensitivity increases to 0.71, which is 4% higher than using only pre-discharge notes and 3% higher than using only post-discharge notes.

Table VIII
COMPARISON OF PERFORMANCE METRICS ACROSS THREE DATASETS.

Set of data	Precision	Recall	F_1 score	Accuracy
PreMN_TBI_MH	0.72	0.67	0.67	0.68
PostMN_TBI_MH	0.71	0.68	0.67	0.68
MN_TBI_MH	0.71	0.71	0.74	0.71

To delve deeper, we extracted the model with the highest accuracy, trained on the third fold using the *PostMN_TBI_MH* dataset, as shown in Table IX. Figure 6 displays the confusion matrix, offering a comprehensive view of the model's classification performance. Figure 7 illustrates the training and testing accuracy over epochs, clearly demonstrating the model's learning progression. Lastly, we utilized the model to create visualizations using the Integrated Gradients method, emphasizing its interpretability in practical applications.

Table IX
PERFORMANCE METRICS FOR THE THIRD FOLD OF THE *PostMN_TBI_MH* DATASET

Metric	Value
Precision	0.7861
Recall	0.8007
F1	0.7861
Accuracy	0.7736

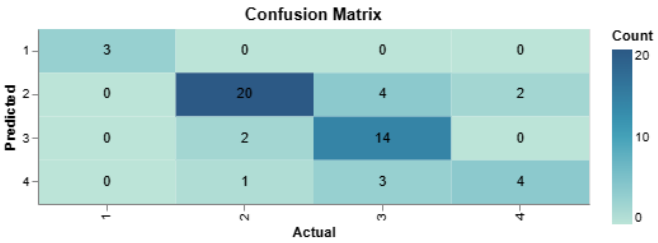


Figure 6. Confusion matrix for third fold of the applied method on PostMN_TBI_MH.

Figure 8 illustrates the application of the Integrated Gradients (IG) method to the medical note "chấn_động não vết_thương phần_mềm đỉnh trái do ngã" (concussion soft tissue injury, left parietal, due to fall). In this case, the true label is 1 (moderate), and the model correctly predicted the outcome. The words contributing most significantly to the model's decision are "chấn_động" (concussion), "vết_thương" (injury), and "do_ngã" (due to fall). These insights can potentially aid doctors in identifying critical terms relevant to patient diagnosis.

Comparision with conventional machine learning methods We applied several traditional machine learning methods as described in [28], along with additional machine learning models:

- **Multinomial Naive Bayes:** A probabilistic classifier based on Bayes' theorem, commonly used for text classification tasks.
- **Decision Tree Classifier:** A simple, interpretable model that splits data into branches based on feature

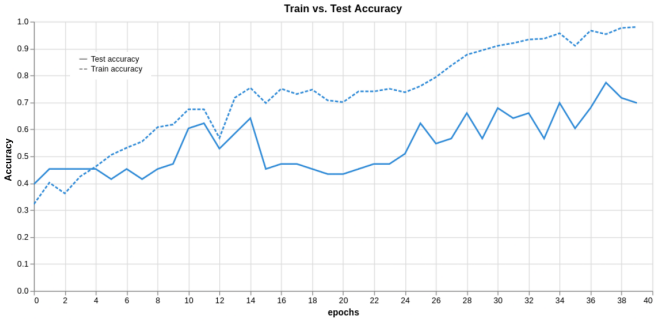


Figure 7. Training and testing progress of Fold 3

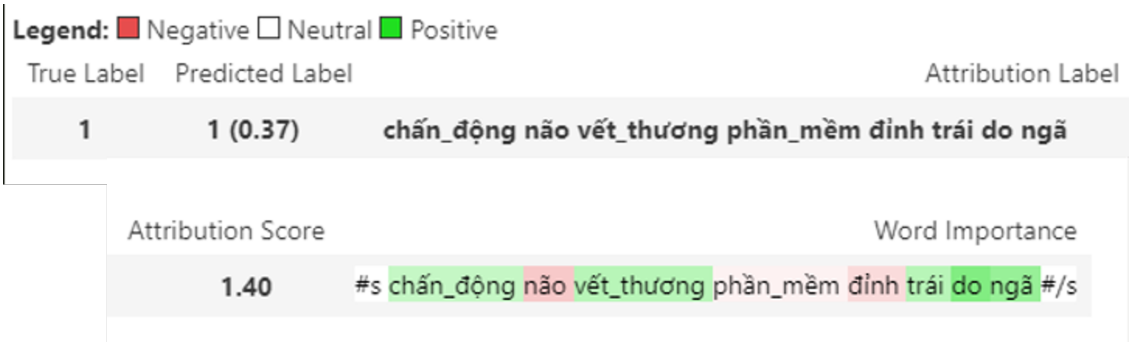


Figure 8. Intepreation of the result using IG method.

values to make predictions.

- **Logistic Regression:** A linear model for binary or multi-class classification that predicts probabilities using the logistic function.
- **Stochastic Gradient Descent Classifier:** A linear classifier optimized using stochastic gradient descent, suitable for large-scale and sparse data.
- **Linear Support Vector Classifier:** A linear SVM implementation configured with Cramer-Singer multi-class support ($C = 1$) to handle multi-class classification tasks.
- **XGBoost Classifier:** A high-performance gradient boosting method.

These traditional models provide a baseline for comparison with our proposed approach, with the results summarized in Table X. The results demonstrate that our method outperformed the others, while the Stochastic Gradient Descent Classifier also achieved relatively high performance.

5. Ablation study

a) Impact of pre-trained models

We compared the performance of viTBI-BERT using three baseline pretrained BERT models: PhoBERTbase [29], trained on general Vietnamese text, to evaluate the impact of general linguistic knowledge, ViPubmedDeBERTaxsmall [30] and ViHealthBERT [3], a domain-specific biomedical model to assess the benefits of specialized pretraining. Using the same hyperparameters in Table VII, we found that ViHealthBERT achieved the highest accuracy. The results are presented in Figure 9. Across all experimented datasets, ViHealthBERT consistently enabled viTBI-BERT to achieve higher accuracy.

b) Impact of pre-processing steps

We have conducted additional experiments along with four different strategies to assess the impact of various pre-processing techniques on model performance. These

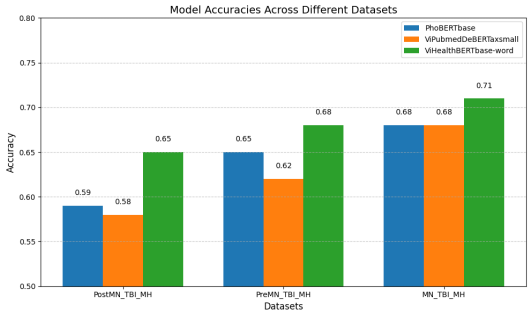


Figure 9. Performance comparison of various pre-trained on our three datasets.

strategies are outlined below and their responding results are indicated by Figures 10, 11, and 12.

- **Strategy 1: Raw data without augmentation:** In this strategy, we use the original, unprocessed text data without applying any additional preprocessing or data augmentation. This serves as a baseline to assess the effectiveness of subsequent preprocessing techniques.
- **Strategy 2: Punctuation removal without augmentation:** In this strategy, we remove punctuation marks from the text, keeping the data otherwise unchanged, to assess how simplifying the text in this way impacts the model's performance. No data augmentation is applied at this stage.
- **Strategy 3: Acronym normalization without augmentation:** In this strategy, we replace acronyms in the text with their full forms to enhance the model's ability to understand medical terminology in its complete form. Again, no data augmentation is introduced at this stage.
- **Strategy 4: Full pre-processing without data augmentation:** This approach includes the complete set of preprocessing techniques—punctuation removal, acronym normalization, and other necessary steps—without applying any data augmentation. This allows us to evaluate the effects of the entire prepro-

Table X
COMPARISION WITH CONVENTIONAL MACHINE LEARNING METHODS

Classifier	PostMN_TBI_MH				PreMN_TBI_MH				MN_TBI_MH			
	Precision	**Recall**	**F1 Score**	**Accuracy**	**Precision**	**Recall**	**F1 Score**	**Accuracy**	**Precision**	**Recall**	**F1 Score**	**Accuracy**
MultinomialNB	0.46	0.34	0.34	0.56	0.46	0.34	0.34	0.56	0.46	0.34	0.34	0.56
DecisionTreeClassifier	0.44	0.47	0.42	0.51	0.40	0.43	0.41	0.46	0.36	0.41	0.36	0.46
LogisticRegression	0.31	0.34	0.32	0.52	0.31	0.35	0.33	0.52	0.31	0.35	0.33	0.51
SGDClassifier	0.61	0.59	0.56	0.57	0.62	0.48	0.51	0.56	0.58	0.45	0.41	0.50
LinearSVC	0.64	0.56	0.52	0.57	0.35	0.42	0.38	0.51	0.46	0.50	0.51	0.56
XGBClassifier	0.33	0.32	0.31	0.47	0.44	0.44	0.42	0.50	0.38	0.36	0.33	0.40
viTBI-BERT (our)	**0.71**	**0.68**	**0.67**	**0.68**	**0.72**	**0.67**	**0.67**	**0.68**	**0.71**	**0.71**	**0.74**	**0.71**

cessing pipeline in isolation.

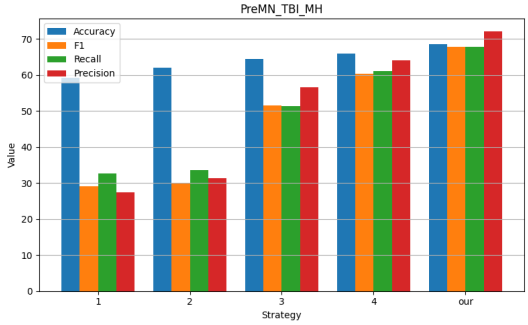


Figure 10. Results of applying several strategies on PreMN_TBI_MH dataset.

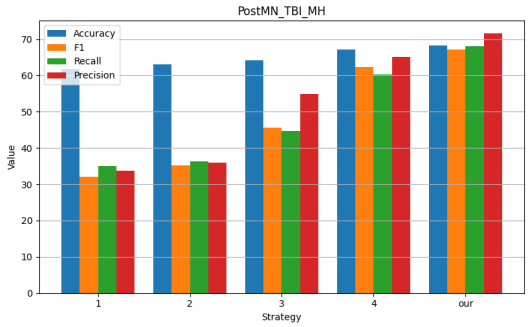


Figure 11. Results of applying several strategies on PostMN_TBI_MH dataset.

We observed an improvement across four evaluation metrics on all three datasets after applying each strategy. This demonstrates the positive impact of the updated preprocessing steps on the model's performance. Compared to the our final framework, the four strategies achieved lower performance.

V. CONCLUSIONS

In this paper, we presented a new framework, viTBI-BERT, for outcome prediction after traumatic brain injury from medical notes of CT scan using a large language model. Our framework consists of two main stages. The preprocessing stage involves expanding all acronyms, removing

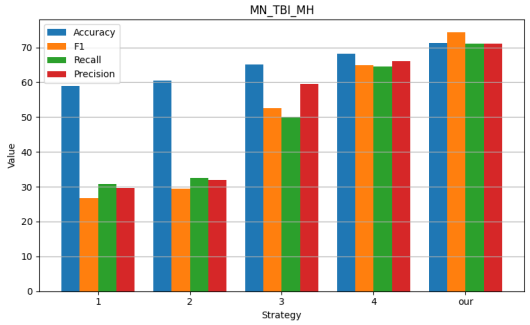


Figure 12. Results of applying several strategies on MN_TBI_MH dataset.

special characters, augmenting data, and performing word segmentation. This stage was executed using dictionary mapping, EDA, back-translation, and RDRSegmenter techniques. The processed data is then fed into the pre-trained ViHealthBERT model, enhanced with an additional fully connected layer and a softmax layer for classification. Our framework was validated on three self-collected medical note sets, comprising 503 patients across four severity levels. Experimental results showed that the sensitivity ranged from 0.67 to 0.68 and 0.71 on medical notes collected before admission, after discharge, and from both periods, respectively. Despite the limited amount of medical notes, the framework was able to effectively distinguish the severity of TBI. In the future, we plan to combine these results with clinical tests and CT scan data for further improvement of the framework.

ACKNOWLEDGEMENTS

This research is funded by the Ministry of Education and Training (MOET) under grant number B2023.BKA.09, titled "*Research and development of supporting tools for prognosis of traumatic brain injury using multi-modal information and artificial intelligence.*"

REFERENCES

[1] H. Khalili, M. Rismani, M. A. Nematollahi, M. S. Masoudi, A. Asadollahi, R. Taheri, H. Pourmontaseri, A. Valibeygi, M. Roshanzamir, R. Alizadehsani *et al.*, "Prognosis prediction in traumatic brain injury patients using machine learning algorithms," *Scientific reports*, vol. 13, no. 1, p. 960, 2023.

[2] D. Yeboah, H. Nguyen, D. B. Hier, G. R. Olbricht, and T. Obafemi-Ajayi, "A deep learning model to predict traumatic brain injury severity and outcome from mr images," in *2021 IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB)*, Oct 2021, pp. 1–6.

[3] M. P. Nguyen, V. H. Tran, V. Hoang, T. D. Huy, T. H. Bui, and S. Q. H. Truong, "ViHealthBERT: Pre-trained Language Models for Vietnamese in Health Text Mining," *13th Edition of its Language Resources and Evaluation Conference*, 2022.

[4] K. Matsuo, H. Aihara, T. Nakai, A. Morishita, Y. Tohma, and E. Kohmura, "Machine learning to predict in-hospital morbidity and mortality after traumatic brain injury," *Journal of Neurotrauma*, vol. 37, no. 1, pp. 202–210, 2020. [Online]. Available: https://doi.org/10.1089/neu.2018.6276

[5] R. Bruschetta, G. Tartarisco, L. F. Lucca, E. Leto, M. Ursino, P. Tonin, G. Pioggia, and A. Cerasa, "Predicting outcome of traumatic brain injury: Is machine learning the best way?" *Biomedicines*, vol. 10, no. 3, 2022. [Online]. Available: https://www.mdpi.com/2227-9059/10/3/686

[6] S. M. Adil, C. Elahi, D. N. Patel, A. Seas, P. I. Warman, A. T. Fuller, M. M. Haglund, and T. W. Dunn, "Deep learning to predict traumatic brain injury outcomes in the low-resource setting," *World Neurosurgery*, vol. 164, pp. e8–e16, 2022.

[7] M. Pease, D. Arefan, J. Barber, E. Yuh, A. Puccio, K. Hochberger, E. Nwachuku, S. Roy, S. Casillo, N. Temkin, D. O. Okonkwo, S. Wu, , N. Badjatia, Y. Bodien, A.-C. Duhaime, V. R. Feeser, A. R. Ferguson, B. Foreman, R. Gardner, S. Gopinath, C. D. Keene, C. Madden, M. McCrea, P. Mukherjee, L. B. Ngwenya, D. Schnyer, S. Taylor, and J. K. Yue, "Outcome prediction in patients with severe traumatic brain injury using deep learning from head ct scans," *Radiology*, vol. 304, no. 2, pp. 385–394, 2022. [Online]. Available: https://doi.org/10.1148/radiol.212181

[8] S. Ding, J. Ye, X. Hu, and N. Zou, "Distilling the knowledge from large-language model for health event prediction," *medRxiv*, pp. 2024–06, 2024.

[9] M. Jin, Q. Yu, D. Shu, C. Zhang, L. Fan, W. Hua, S. Zhu, Y. Meng, Z. Wang, M. Du *et al.*, "Health-llm: Personalized retrieval-augmented disease prediction system," *arXiv preprint arXiv:2402.00746*, 2024.

[10] L. Rasmy, Y. Xiang, Z. Xie, C. Tao, and D. Zhi, "Med-bert: pretrained contextualized embeddings on large-scale structured electronic health records for disease prediction," *NPJ digital medicine*, vol. 4, no. 1, p. 86, 2021.

[11] S. Hegselmann, A. Buendia, H. Lang, M. Agrawal, X. Jiang, and D. Sontag, "Tabllm: Few-shot classification of tabular data with large language models," in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2023, pp. 5549–5581.

[12] Z. Li, Y. Li, Q. Li, P. Wang, D. Guo, L. Lu, D. Jin, Y. Zhang, and Q. Hong, "Lvit: language meets vision transformer in medical image segmentation," *IEEE transactions on medical imaging*, 2023.

[13] C. Le Barbey, "Evaluation of artificial language model chatgpt4 to predict 6-month outcome after traumatic brain injury," Ph.D. dissertation, 2023.

[14] C. Vo, T. Cao, and B. Ho, "Abbreviation detection in vietnamese clinical texts," *VNU Journal of Science: Computer Science and Communication Engineering*, vol. 34, no. 2, 2018. [Online]. Available: //jcsce.vnu.edu.vn/index.php/jcsce/article/view/211

[15] A. Tabassum and D. R. R. Patil, "A survey on text pre-processing & feature extraction techniques in natural language processing," 2020. [Online]. Available: https://api.semanticscholar.org/CorpusID:235211496

[16] D. Q. Nguyen, D. Q. Nguyen, T. Vu, M. Dras, and M. Johnson, "A fast and accurate vietnamese word segmenter," *CoRR*, vol. abs/1709.06307, 2017. [Online]. Available: http://arxiv.org/abs/1709.06307

[17] T. Vu, D. Q. Nguyen, D. Q. Nguyen, M. Dras, and M. Johnson, "VnCoreNLP: A Vietnamese natural language processing toolkit," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*. Association for Computational Linguistics, 2018, pp. 56–60. [Online]. Available: https://aclanthology.org/N18-5012

[18] J. Wei and K. Zou, "EDA: Easy data augmentation techniques for boosting performance on text classification tasks," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Association for Computational Linguistics, nov 2019, pp. 6382–6388. [Online]. Available: https://aclanthology.org/D19-1670

[19] T. P. Nguyen, V.-L. Pham, H.-A. Nguyen, H.-H. Vu, N.-A. Tran, and T.-T.-H. Truong, "A two-phase approach for building Vietnamese WordNet," in *Proceedings of the 8th Global WordNet Conference (GWC)*. Global Wordnet Association, 27–30 2016, pp. 261–266. [Online]. Available: https://aclanthology.org/2016.gwc-1.38

[20] M. Xia, X. Kong, A. Anastasopoulos, and G. Neubig, "Generalized data augmentation for low-resource translation," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, jul 2019, pp. 5786–5796. [Online]. Available: https://aclanthology.org/P19-1579

[21] A. Sugiyama and N. Yoshinaga, "Data augmentation using back-translation for context-aware neural machine translation," in *Proceedings of the Fourth Workshop on Discourse in Machine Translation (DiscoMT 2019)*. Association for Computational Linguistics, nov 2019, pp. 35–44. [Online]. Available: https://aclanthology.org/D19-6504

[22] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *North American Chapter of the Association for Computational Linguistics*, 2019. [Online]. Available: https://api.semanticscholar.org/CorpusID:52967399

[23] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2023. [Online]. Available: https://arxiv.org/abs/1706.03762

[24] M. Sundararajan, A. Taly, and Q. Yan, "Axiomatic attribution for deep networks," in *International conference on machine learning*. PMLR, 2017, pp. 3319–3328.

[25] Z. Zhao, R. Anand, and M. Wang, "Maximum relevance and minimum redundancy feature selection methods for a marketing machine learning platform," in *2019 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, 2019, pp. 442–452.

[26] F. Gaillard, "Rotterdam CT score of traumatic brain injury | Radiology Reference Article | Radiopaedia.org — radiopaedia.org," https://radiopaedia.org/articles/rotterdam-ct-score-of-traumatic-brain-injury.

[27] N. Kokhlikyan, V. Miglani, M. Martin, E. Wang, B. Alsallakh, J. Reynolds, A. Melnikov, N. Kliushkina, C. Araya, S. Yan, and O. Reblitz-Richardson, "Captum: A unified and generic model interpretability library for pytorch," 2020.

[28] Q. T. Nguyen, T. L. Nguyen, N. H. Luong, and Q. H. Ngo, "Fine-tuning BERT for sentiment analysis of vietnamese reviews," *CoRR*, vol. abs/2011.10426, 2020. [Online]. Available: https://arxiv.org/abs/2011.10426

[29] D. Q. Nguyen and A. T. Nguyen, "Phobert: Pre-trained language models for vietnamese," *CoRR*, vol. abs/2003.00744, 2020. [Online]. Available: https://arxiv.org/abs/2003.00744

[30] L. Phan, T. Dang, H. Tran, T. H. Trinh, V. Phan, L. D. Chau, and M.-T. Luong, "Enriching biomedical knowledge for low-resource language through large-scale translation," 2022. [Online]. Available: https://arxiv.org/abs/2210.05598

Duc-Khiem Doan is currently a final-year undergraduate student in Biomedical Engineering at Hanoi University of Science and Technology. His main research interests include medical data processing and the application of machine learning and deep learning models to biomedical data.
Email: khiem.DD210483@sis.hust.edu.vn

Thanh-Hai Tran graduated with an engineer's degree in Information Technology from Hanoi University of Science and Technology (HUST) in 2001. She holds an M.S. degree and a Ph.D. degree in Imagery Vision Robotics from Grenoble INP, France, in 2002 and 2006 respectively. Currently, she is a lecturer/researcher at the School of Electrical and Electronics Engineering and 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.vnnd

Trung-Kien Tran graduated from Hanoi University of Science and Technology in 2004 and worked at the Military Information Technology Institute, AMST, as a researcher in the field of cybersecurity until 2014. After that, he began pursuing a Masterś degree and Ph.D. in Machine Learning at the Artificial Intelligence Laboratory, National Defense Academy of Japan, and graduated in March 2020. Currently, he is an AI specialist at the Military Information Technology Institute, AMST, with primary research interests in Edge Artificial intelligence, Human-Robot Interaction, and Unmanned Vehicle.
Email: t2kien@gmail.com

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 an associate professor at the School of Electrical and Electronic Engineering (SEEE), HUST, Vietnam. Her research interests include image processing, computer vision, content-based indexing and, retrieval, video understanding, and human-robot interaction.
Email: lan.lethi1@hust.edu.vn

Hai Vu received a PhD in Computer Science from Osaka University, Japan, in 2009. Currently, the author is a lecturer at the School of Electrical and Electronic Engineering (SEEE), HUST, Vietnam. His research interests include computer vision-based techniques in smart-surveillance camera networks, medical image analysis for diagnostic assistance and Human-Machine Interaction.
Email: hai.vu@hust.edu.vn

Huu-Khanh Nguyen is a neurosurgery resident at 103 Hospital in Hanoi, Vietnam. He graduated from Vietnam Military Medical University in 2022, where he completed seven years of comprehensive medical training. Since then, he has been specializing in neurosurgery, developing his skills and knowledge at one of Vietnam's leading medical institutions.
Email: HuukhanhHVQY@gmail.com

Van Mao Can graduated from the Military Academy of Medicine, Hanoi, Vietnam, in 2000. He earned his Master's Degree in Medicine in 2003 and a Doctoral Degree in System Emotional Science from Toyama University, Japan in 2009. He was appointed Associate Professor in Medicine in March 2018. Since 2018, he has been the Head of the Pathophysiology Department at the Military Medical University, contributing significantly to education, research, and medical advancements.
Email: canvanmao@vmmu.edu.vn

Thanh Bac Nguyen graduated from the Military Academy of Medicine, Hanoi, in 2000 and completed his residency in neurosurgery at Military Hospital 103 in 2005. He earned his Ph.D. in 2020. Since then, he has been the Head of the Department of Neurosurgery at Military Hospital 103. In December 2024, he was appointed Associate Professor. He is dedicated to advancing neurosurgery and mentoring the next generation of medical professionals in Vietnam.
Email: bacnt103@gmail.com