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 [1, 2]. 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) [3] and ADASYN (Adaptive Synthetic Sampling Approach) [4] 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 [12]. 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 [3] 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 [4] 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 [5, 12], 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

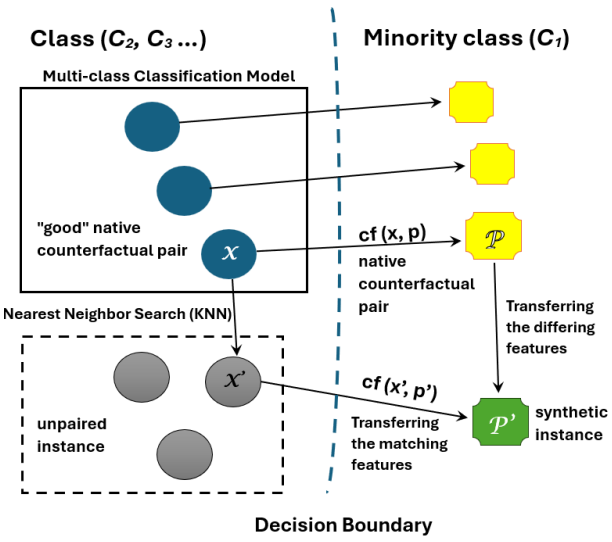

Figure 1. CFA-MC

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 [5, 6], 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).

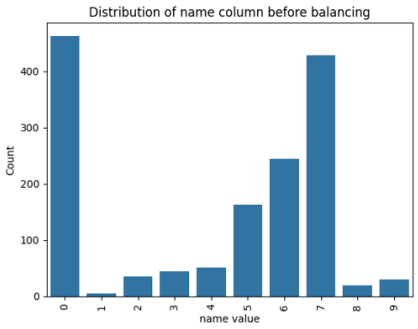

Figure 2: Yeast data distribution

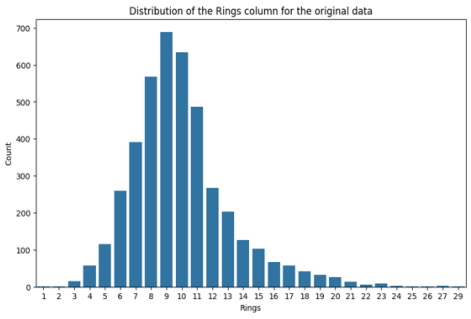

Figure 3: Abalone data distribution

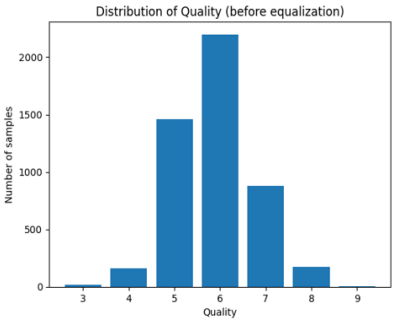

Figure 4: Wine Quality White data distribution

Figure 2. The imbalance ratio

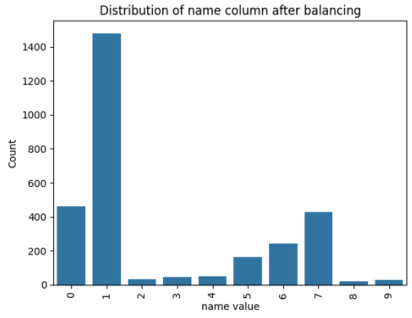

Figure 5: Yeast data distribution

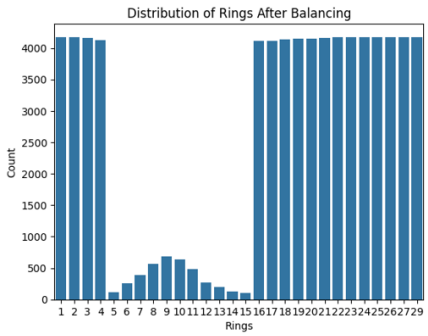

Figure 6: Abalone data distribution

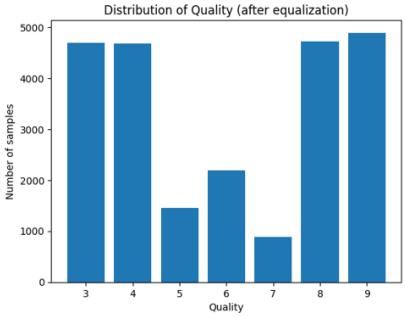

Figure 7: Wine Quality White data distribution

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