ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

Nguyễn Khánh Tùng

NGHIÊN CỨU PHÁT TRIỂN KỸ THUẬT HỌC MÁY PHÁT HIỆN VÀ PHÂN LỚP TRÔI KHÁI NIỆM

LUẬN ÁN TIẾN SĨ HỆ THỐNG THÔNG TIN

HÀ NỘI - 2025

ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

Nguyễn Khánh Tùng

NGHIÊN CỨU PHÁT TRIỂN KỸ THUẬT HỌC MÁY PHÁT HIỆN VÀ PHÂN LỚP TRÔI KHÁI NIỆM

Ngành đào tạo : Hệ thống thông tin
Mã số : 9480104

LUẬN ÁN TIẾN SĨ HỆ THỐNG THÔNG TIN

NGƯỜI HƯỚNG DẪN KHOA HỌC
1. PGS. TS. HÀ QUANG THỤY
2. PGS. TS. PHAN XUÂN HIẾU

HÀ NỘI - 2025

Lời cam đoan

Tôi xin cam đoan luận án này là công trình nghiên cứu của riêng tôi. Các kết quả được viết chung với các tác giả khác đều được sự đồng ý của các đồng tác giả trước khi đưa vào luận án. Các kết quả nêu trong luận án là trung thực và chưa từng được công bố trong các công trình nào khác.

Nghiên cứu sinh

Nguyễn Khánh Tùng

Lời cảm ơn

Trước hết, tôi xin bày tỏ lòng biết ơn sâu sắc tới PGS.TS. Hà Quang Thụy và PGS.TS. Phan Xuân Hiếu, những người thầy đã tận tình hướng dẫn, hỗ trợ và đồng hành cùng tôi trong suốt quá trình thực hiện luận án này. Những góp ý quý báu, sự nghiêm túc trong khoa học và tinh thần tận tụy của các thầy là nguồn động lực to lớn giúp tôi hoàn thành công trình nghiên cứu này.

Tôi cũng xin chân thành cảm ơn các thầy cô trong Khoa Công nghệ thông tin, Trường Đại học Công nghệ, Đại học Quốc gia Hà Nội đã giảng dạy, truyền đạt kiến thức quý giá và tạo điều kiện thuận lợi cho tôi trong suốt thời gian học tập và nghiên cứu tại trường.

Xin cảm ơn Hội đồng đánh giá luận án và các nhà khoa học đã dành thời gian đọc, nhận xét, góp ý giúp tôi hoàn thiện hơn nội dung và chất lượng của luận án.

Tôi cũng xin gửi lời tri ân tới các đồng nghiệp, bạn bè, và nhóm nghiên cứu DS&KTLab, các bạn sinh viên đã luôn đồng hành, chia sẻ kinh nghiệm, khích lệ tôi trong suốt hành trình nghiên cứu và học tập đầy thử thách này.

Cuối cùng, tôi xin gửi lời cảm ơn sâu sắc nhất tới gia đình – đặc biệt là cha mẹ, vợ và con – những người luôn ở bên, yêu thương và động viên tôi không ngừng, là chỗ dựa vững chắc để tôi vượt qua mọi khó khăn trong suốt quá trình thực hiện luận án tiến sĩ.

Nghiên cứu sinh

Nguyễn Khánh Tùng

Mục lục

Danh mục thuật ngữ và viết tắt

Thuật ngữ tiếng Anh	**Thuật ngữ tiếng Việt**	**Viết tắt**
Application Programming Interface	Giao diện lập trình ứng dụng	API
Artificial Intelligence	Trí tuệ nhân tạo	AI/TTNT
Bayes Optimization	Tối ưu hóa Bayes	BO
Convolutional Neural Networks	Mạng nơ-ron tích chập	CNN
Cosine Similarity and Angular Margin based Softmax Classifier	Bộ phân lớp Softmax dựa trên độ tương đồng Cosin và biên góc	CS&AM_SC
Drift Detection Method	Phương pháp phát hiện trôi khái niệm	DDM
Effective and Robust Identification of Concept Shift	Phát hiện trôi khái niệm hiệu quả và chắc chắn (mô hình)	ERICS
Ensemble ERICS	Tổng hợp ERICS	E-ERICS
Extreme Gradient Boosting	Tăng cường độ dốc cực đại (Thuật toán)	XGBoost
False Negative	Âm tính sai	FN
False Positive	Dương tính sai	FP
Feedforward Neural Network	Mạng nơ-ron truyền thẳng	FNN
Fully Attention Network	Mạng nơ-ron với cơ chế chú ý đầy đủ	FAN
Fully Convolutional Network	Mạng nơ-ron tích chập đầy đủ	FCN

Internet of Things	Internet vạn vật	IoT
Jensen-Shannon Divergence	Khoảng cách Jensen-Shannon	J-SD
Kernel Density Estimation	Ước lượng mật độ nhân	KDE
k-Nearest Neighbors	k-láng giềng gần nhất	kNN
Kolmogorov-Smirnov Test	Kiểm định thống kê Kolmogorov–Smirnov	K-ST
Kullback-Leibler Divergence	Khoảng cách Kullback Leibler	K-LD
Massive Online Analysis	Phân tích trực tuyến loạt (Thư viện)	MOA
Meta learning Lightweight Drift Detection Finetune	Bộ phát hiện trôi nhẹ được tinh chỉnh	MetaLDD-Finetune
Multilayer Perceptron	Mạng Perceptron nhiều lớp	MLP
Recurrent Neural Network	Mạng nơ-ron hồi quy	RNN
Support Vector Machine	Máy vector hỗ trợ	SVM
True Negative	Âm tính đúng	TN
True Positive	Dương tính đúng	TP
Variation Windowing Strategy	Chiến lược cửa sổ linh hoạt	VAR-WIND
Wilcoxon Test	Kiểm định Wilcoxon	Wilcoxon
Thuật ngữ không có viết tắt		
Angular margin	Biên góc	
Batch	Lô (dữ liệu)	
Center loss	Mất mát trung tâm	
Change Point Detection	Phát hiện điểm trôi	

Concept drift	Trôi khái niệm	
Concept drift adaptation	Thích nghi với trôi khái niệm	
Data stream	Luồng dữ liệu	
Ensemble learning	Học kết hợp	
Entropy regularization	Điều chuẩn độ bất định	
Episode	Vòng huấn luyện	
Episodic training	Huấn luyện theo tập nhiệm vụ	
Expanding Window	Cửa sổ mở rộng	
Few-shot learning	Học từ ít mẫu	
Gradual drift	Trôi dần	
Incremental drift	Trôi gia tăng	
Inter-class separation	Độ phân tách giữa các lớp	
Interleaved test-then-train	Xen kẽ việc kiểm tra với huấn luyện	
Intra-class variation	Biến thiên nội lớp	
Meta-learning	Học cách học	
Moving Average	Trung bình động	
Prototype	Tâm lớp	
Prototypical Network	Mạng nguyên mẫu	
Query Set	Tập truy vấn	
Reinforcement Learning	Học tăng cường	
Reoccuring drift	Trôi tái diễn	
Sliding window	Cửa sổ trượt	
Standard supervised training	Huấn luyện theo phương thức học giám sát tiêu chuẩn	

Sudden drift	Trôi đột ngột	
Support Set	Tập hỗ trợ	
Timestamp	Tem (dấu) thời gian	
Tumbling window	Cửa sổ rời rạc	

Danh mục các bảng

Danh mục các hình vẽ

Mở đầu

Trong một thế giới thực luôn động, các khái niệm thường không ổn định mà thay đổi theo thời gian nhưng hầu hết các mô hình học máy được xây dựng dựa trên các tập dữ liệu quá khứ đều tĩnh. Vì vậy, việc triển khai trực tuyến ngày càng nhiều các mô hình học máy tĩnh đã làm gia tăng tính cấp thiết cho việc phát triển các cơ chế hiệu quả cao và hiệu suất cao để giải quyết vấn đề học máy đối với phân phối dữ liệu không tĩnh, đặc biệt trong bối cảnh trực tuyến. Đồng thời, các phương pháp học máy trực tuyến phải đối phó được với các thách thức như dung lượng bộ nhớ hạn chế, các yêu cầu thời gian thực và trôi khái niệm (concept drift). Hơn nữa, các phương pháp học máy trực tuyến phải cung cấp các dự đoán có chất lượng cao, ổn định khi có nhiễu đầu vào và khả năng diễn giải tốt để có thể sử dụng đáng tin cậy trong thực tế. Trôi khái niệm ([43], [59], [19], [49], [28], [36]) mô tả những thay đổi không lường trước được phân phối dữ liệu được ghi nhận theo thời gian và các mô hình học máy cần có khả năng tự động thích nghi được với những thay đổi theo thời gian như vậy. Trôi khái niệm là một vấn đề nổi bật trong khai phá dữ liệu, đề cập tới "*hiện tượng mà các tính chất thống kê của biến mục tiêu thay đổi theo thời gian một cách tuỳ ý không biết trước*" [28] hoặc "*mối quan hệ giữa dữ liệu đầu vào và biến mục tiêu có sự thay đổi theo thời gian*" [36].

Nói một cách hình thức, trôi khái niệm tại thời điểm t được định nghĩa là sự thay đổi phân phối kết hợp $P_t(X, y)$ của dữ liệu đầu vào X và biến mục tiêu y tại thời điểm t theo công thức $P_t(X, y) = P_t(X) \times P_t(y|X)$; trong học không giám sát, trôi khái niệm còn được định nghĩa là sự thay đổi phân phối biên của dữ liệu đầu vào (marginal distribution of input features) $P_t(X)$ [49]. Luận án này được định hướng theo tiếp cận trôi khái niệm dựa trên học giám sát.

Trôi khái niệm được phân lớp thành bốn kiểu là đột ngột (*sudden*), dần (*gradual*), gia tăng (*incremental*), tái diễn (*reoccuring*) ([55]). Các nghiên cứu về thích nghi với trôi khái niệm (*concept drift adaptation*) đối với ba kiểu trôi khái niệm đầu tiên tập trung vào việc làm thế nào để cực tiểu hoá việc suy giảm độ chính xác và đạt được sự phục hồi mô hình học nhanh nhất có thể trong suốt quãng thời gian thay đổi xảy ra, trong khi đó đối với kiểu trôi khái niệm cuối thì tập trung

vào việc sử dụng phương pháp học dựa trên dữ liệu lịch sử, bằng cách tìm ra một sự thay đổi trùng khít nhất từ dữ liệu đã xảy ra trong lịch sử theo khoảng thời gian ngắn nhất có thể [28]. Nghiên cứu về trôi khái niệm liên quan đến việc phát triển các phương pháp luận và kỹ thuật giải quyết ba bài toán trôi khái niệm chính là Phát hiện trôi khái niệm, Hiểu trôi khái niệm và Thích nghi với trôi khái niệm.

Tồn tại ba nhóm phương pháp và thuật toán ***phát hiện trôi khái niệm*** là: dựa trên tỷ lệ lỗi, dựa trên phân phối dữ liệu và dựa trên kiểm thử đa giả thuyết. Khung tổng quát Phát hiện trôi khái niệm bao gồm bốn giai đoạn. *Tìm kiếm dữ liệu* (Data Retrieval) nhằm thu được các đoạn dữ liệu từ các luồng dữ liệu (data streams). *Mô hình hóa* dữ liệu (Data Modeling) nhằm bao gói các đoạn dữ liệu thu được và trích chọn thành các đặc trưng chứa đựng các thông tin có tính nhận dạng tốt, hay nói cách khác là bước này sẽ chọn ra các đặc trưng có ý nghĩa trong việc đánh giá hệ thống khi nó thay đổi. *Tính toán kiểm thử thống kê* (Test Statistics Calculation) là bước tính toán khoảng cách để xác định có sự thay đổi hay không. Bước này là bước khó khăn nhất trong tất cả các bước. *Kiểm thử giả thuyết* (Hypothesis Test) tiến hành một kiểu kiểm thử giả thuyết để kiểm thử một giả thuyết cụ thể để đánh giá tính thống kê các thay đổi được xác định ở bước Tính toán kiểm thử thống kê. Lưu ý rằng, phát hiện trôi khái niệm là một quá trình phản ứng mà không phải là quá trình phòng ngừa [36].

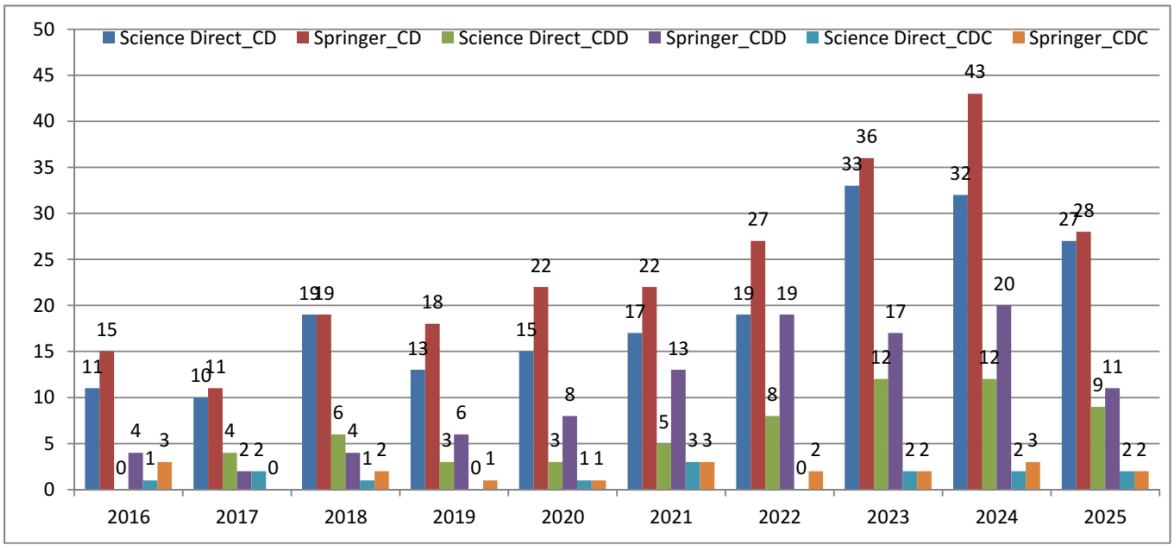
Ba bài toán ***hiểu trôi khái niệm*** là Phát hiện điểm trôi (Change Point Detection) xác định thời điểm xảy ra trôi khái niệm, Mức nghiêm trọng của trôi khái niệm và Vùng trôi khái niệm, trong đó, phát hiện điểm trôi là bài toán được quan tâm nhiều nhất. Hiểu biết về trôi khái niệm cần dự báo được thời điểm bắt đầu, giai đoạn thay đổi và thời điểm kết thúc của trôi khái niệm. Mức nghiêm trọng của trôi khái niệm đề cập đến việc sử dụng giá trị định lượng để đo mức độ tương đồng giữa khái niệm mới và khái niệm trước đó. Vùng trôi của khái niệm trôi là vùng xung đột giữa khái niệm mới và khái niệm trước đó, được định vị bằng cách tìm vùng trong không gian đặc trưng dữ liệu mà tại đó các phân phối dữ liệu có khác biệt đáng kể. Các kỹ thuật xác định vùng trôi phụ thuộc rất nhiều vào mô hình dữ liệu được sử dụng trong các phương pháp/thuật toán phát hiện trôi khái niệm.

Thích nghi với trôi đề cập tới việc cập nhật các mô hình học máy hiện có để phù hợp với trôi khái niệm. Ba tiếp cận chính cho thích nghi trôi là Huấn luyện

mô hình mới cho trôi toàn cục, Mô hình tổng hợp cho trôi tái diễn và Điều chỉnh các mô hình hiện có theo trôi theo vùng.

Các phương pháp tiếp cận khác nhau để xử lý trôi khái niệm đã được đề xuất và nhiều phương pháp trong số đó đã minh chứng được tiềm năng của chúng trong nhiều lĩnh vực ứng dụng, chẳng hạn như, phát hiện gian lận, kiểm soát hệ thống thích nghi, mô hình hóa người dùng, truy xuất thông tin, khai phá văn bản, y sinh học, chăm sóc sức khỏe, giáo dục, tài chính hoặc quảng cáo ([43], [59], [19], [49]).

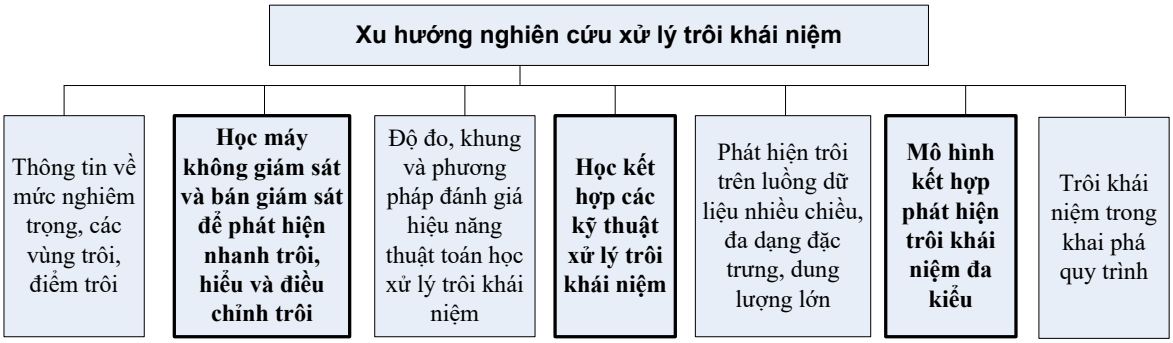
Nghiên cứu về Xử lý trôi khái niệm đã tăng trưởng không ngừng trong khoảng thời gian mười năm trở lại đây. Hình 0.1 cho một minh họa về số lượng các công trình khoa học có tiêu đề liên quan tới các cụm từ *"Concept Drift" (CD), "Concept Drift Detection" (CDD)* và *"Concept Drift Classification"* được công bố trên Science Direct và Springer có xu hướng tăng dần hàng năm. Như vậy, nghiên cứu về Xử lý trôi khái niệm là có tính thời sự và rất cần thiết.



Hình 0.1 Một thống kê vào tháng 09/2025 về số công bố khoa học có tiêu đề liên quan tới Trôi khái niệm (CD), Phát hiện trôi khái niệm (CDD) và Phân lớp trôi khái niệm (CDC) trên Science Direct và Springer

Như được mô tả trên Hình 0.2, các xu hướng nghiên cứu điển hình để giải quyết các thách thức đối với xử lý trôi khái niệm là ([28], [2], [Cetrulo23], [29]): (1) Cung cấp thông tin về mức nghiêm trọng, các vùng trôi cùng với điểm trôi; (2) Học máy không giám sát và bán giám sát để phát hiện nhanh trôi, hiểu và điều chỉnh trôi khái niệm, đặc biệt trong xử lý trực tuyến; (3) Độ đo, khung và phương pháp đánh giá hiệu năng thuật toán học xử lý trôi khái niệm; (4) Học kết hợp các

kỹ thuật xử lý trôi khái niệm; (5) Phát hiện trôi trên luồng dữ liệu nhiều chiều, đa dạng đặc trưng (số, phân lớp, đa phân lớp, thời gian, nhị phân và độ lệch) và dung lượng lớn; (6) Mô hình kết hợp phát hiện trôi khái niệm đa kiểu (đột ngột, dần dần, gia tăng); (7) Trôi khái niệm trong khai phá quy trình.



Hình 0.2 Các xu hướng nghiên cứu xử lý trôi khái niệm

Các xu hướng nghiên cứu điển hình đối với trôi khái niệm trên đây đặt ra nhiều vấn đề nghiên cứu hấp dẫn cần giải quyết đầy thách thức, nảy sinh ra các câu hỏi và chủ đề nghiên cứu cho các luận án Tiến sĩ trên thế giới (một số luận án như vậy được giới thiệu sơ bộ tại Chương 1 của luận án này).

Định hướng tới các chủ đề nghiên cứu quan trọng theo xu hướng xử lý trôi khái niệm như đã được đề cập, **nghiên cứu của luận án** hướng tới **ba câu hỏi nghiên cứu với mục tiêu nghiên cứu** sau đây.

- *Câu hỏi nghiên cứu đầu tiên* là về khung và mô hình học kết hợp xử lý trôi khái niệm cũng như cơ chế quyết định trong học kết hợp, tương ứng với xu hướng nghiên cứu về Học kết hợp các kỹ thuật xử lý trôi khái niệm (Xu hướng 4). Mục tiêu nghiên cứu tương ứng là đề xuất các mô hình phát hiện trôi khái niệm dựa trên Học kết hợp và tối ưu hóa siêu tham số mô hình.
- *Câu hỏi nghiên cứu thứ hai* là về cách thức sử dụng mô hình cửa sổ linh hoạt để phân lớp trôi khái niệm đa kiểu, tương ứng với xu hướng nghiên cứu về mô hình kết hợp phát hiện trôi khái niệm đa kiểu (đột ngột, gia tăng, dần) (Xu hướng 6). Mục tiêu nghiên cứu tương ứng là đề xuất các mô hình phân lớp trôi khái niệm dựa trên mạng nơron nguyên mẫu (Prototypical Network) xử lý cửa sổ phát hiện trôi khái niệm đa kiểu.
- *Câu hỏi nghiên cứu thứ ba* là về nâng cao khả năng phân biệt của mô hình, phù hợp với xu hướng Học máy không giám sát và bán giám sát để phát hiện nhanh trôi, đặc biệt trong môi trường trực tuyến với số lượng mẫu có

nhãn rất ít (Xu hướng 2). Mục tiêu nghiên cứu là đề xuất mô hình phân lớp trôi khái niệm dựa trên phân tích và tối ưu không gian biểu diễn của mạng nguyên mẫu.

Đối tượng nghiên cứu của luận án là các mô hình xử lý trôi khái niệm và các kỹ thuật được áp dụng trong các mô hình đó.

Phạm vi nghiên cứu của luận án tập trung tới các kỹ thuật và giải pháp cải tiến được áp dụng trong các mô hình xử lý trôi khái niệm dựa trên học máy tổng hợp và học xử lý trôi khái niệm đa kiểu.

Luận án sử dụng **phương pháp nghiên cứu kết hợp** phương pháp nghiên cứu định tính và phương pháp nghiên cứu định lượng. Tiến hành một quá trình đọc-nghĩ-giải thích nhằm phân tích định tính các khái niệm và mô hình từ hệ thống tài liệu liên quan, luận án tập trung vào các tài liệu khoa học tổng quan (sách, bài báo, luận án Tiến sĩ, chẳng hạn như [59], [19], [49], [36], [12], [41], [31], [37], [32]) và chuyên sâu (chẳng hạn như [39], [23], [24], [1], [25], [55], [56]) phù hợp với các mục tiêu nghiên cứu của luận án để đề xuất các cải tiến để hình thành các kỹ thuật và mô hình xử lý trôi khái niệm mới. Đồng thời, luận án tiến hành triển khai các hệ thống thực nghiệm tương ứng theo các kịch bản thực nghiệm phù hợp để đánh giá hiệu năng của các kỹ thuật và mô hình đề xuất nhằm kiểm chứng, đánh giá kết quả về hiệu năng của các kỹ thuật và mô hình do luận án đề xuất. Để nhận được đánh giá khách quan về chất lượng nghiên cứu, các kết quả nghiên cứu của luận án được gửi công bố tới các ấn phẩm khoa học quốc tế uy tín.

Tham gia vào dòng nghiên cứu trên thế giới theo các xu hướng xử lý trôi khái niệm như đã được đề cập, luận án có ba nhóm đóng góp chính sau đây:

- Luận án đề xuất một khung phát hiện trôi khái niệm gồm hai pha nối tiếp nhau, là pha tối ưu hóa siêu tham số mô hình và pha tổng hợp. Trong pha tổng hợp xây dựng hai mô hình: E-ERICS và ERICS+3. Mô hình E-ERICS (Ensemble-ERICS) được thiết kế bằng cách tổng hợp bốn mô hình cơ sở tốt nhất sau pha tối ưu hóa siêu tham số. Mô hình sử dụng cơ chế bỏ phiếu để đưa ra quyết định về việc một mẫu dữ liệu trong luồng có phải là điểm trôi khái niệm đột ngột hoặc trôi gia tăng hay không. Trong khi đó, mô hình ERICS+3 được xây dựng bằng cách kết hợp ba mô hình cơ sở tốt nhất nhằm phát hiện các điểm trôi dần dần trên từng trường hợp dữ liệu trong luồng. Các kết quả nghiên cứu tương

ứng với hai mô hình này đã được công bố trong các công trình [TungNK1] đối với mô hình E-ERICS và [TungNK2] đối với mô hình ERICS+3.

- Luận án đề xuất hai khung cải tiến cho nhiệm vụ phân lớp trôi khái niệm, bao gồm VAR-WIND và MetaLDD-Finetune. Cả hai khung đều được phát triển dựa trên khung phân lớp trôi Meta-ADD [55], đồng thời kế thừa một số thành phần từ khung Type-LDD [56] – phiên bản mở rộng của Meta-ADD. Khung VAR-WIND cải tiến giai đoạn trích xuất đặc trưng của Meta-ADD bằng cách áp dụng các chiến lược cửa sổ khác nhau phù hợp với từng kiểu trôi: sử dụng cửa sổ lật cho trôi đột ngột, cửa sổ trượt cho trôi dần và cửa sổ mở rộng cho trôi tăng dần. Mỗi chiến lược được lựa chọn dựa trên đặc điểm của từng kiểu trôi, từ đó giúp mô hình học được các biểu diễn đặc trưng phù hợp hơn cho từng loại trôi khái niệm. Khung MetaLDD-Finetune được xây dựng bằng cách bổ sung một bước tinh chỉnh (fine-tuning) sau giai đoạn tiền huấn luyện, nhằm cải thiện khả năng thích ứng của bộ phân lớp với phân phối dữ liệu mới. Bước tinh chỉnh này giúp tối ưu hóa hiệu quả phân lớp cho các kiểu trôi khái niệm trong luồng dữ liệu. Các kết quả nghiên cứu liên quan đến hai khung đề xuất đã được công bố trong các công trình: [TungNK5] cho VAR-WIND và [TungNK3] cho MetaLDD-Finetune.
- Từ nền tảng của khung MetaLDD-Finetune, luận án phát triển một khung phân lớp trôi khái niệm mới có tên là CS&AM_SC (Cosine Similarity and Angular Margin based Softmax Classifier), nhằm nâng cao độ chính xác trong phân loại kiểu trôi thông qua việc phân tích và tối ưu hóa không gian biểu diễn đặc trưng trong mạng nguyên mẫu. Cụ thể, khung CS&AM_SC sử dụng ba kỹ thuật chính: (i) thay thế phép nhân vô hướng bằng độ tương đồng Cosine trong bộ phân lớp Softmax, (ii) tích hợp biên góc (Angular Margin) để tăng cường khả năng phân tách giữa các lớp, và (iii) sử dụng hàm mất mát trung tâm (Center Loss) nhằm giảm biến thiên nội lớp. Sự kết hợp các kỹ thuật này giúp tối ưu hóa không gian biểu diễn theo hướng vừa đảm bảo tính gắn kết trong lớp, vừa tăng cường khả năng phân tách giữa các lớp, từ đó cải thiện hiệu quả phân lớp kiểu trôi. Kết quả nghiên cứu thuộc nhóm đóng góp này của luận án đã được công bố trong [TungNK4].

Bố cục của luận án gồm Mở đầu và bốn chương nội dung, Kết luận và Danh mục các tài liệu tham khảo. Nội dung chính của bốn chương của luận án được mô tả sơ bộ như sau:

- ***Chương 1*** trình bày một cách hệ thống về xử lý trôi khái niệm, bao gồm các khái niệm về trôi khái niệm và liên quan, các bài toán, phương pháp, xu hướng, một số bộ dữ liệu và độ đo đánh giá phổ biến trong xử lý trôi khái niệm. Mối liên hệ của các nghiên cứu trong luận án với một sô mô hình và phương pháp xử lý trôi khái niệm cũng được giới thiệu.
- ***Chương 2*** trình bày đề xuất khung xử lý trôi khái niệm gồm hai pha, trong đó đề xuất hai mô hình xử lý trôi khái niệm E-ERICS và ERICS+3 theo tiếp cận học kết hợp của luận án. Chương này bắt đầu bằng việc giới thiệu khung mô hình tối ưu hóa tham số xử lý trôi khái niệm ERICS của J. Haug và G. Kasneci [24]. Luận án chỉ ra những điểm còn tồn tại của ERICS để từ đó đề xuất cải tiến bằng một khung hai pha, trong đó pha tổng hợp đề xuất hai mô hình E-ERICS và ERICS+3. Các thực nghiệm đánh giá hiệu năng của hai mô hình được trình bày cụ thể và được đối sánh với khung ERICS gốc để minh chứng sự hợp lý của đề xuất cải tiến.
- ***Chương 3*** trình bày đề xuất của luận án về hai khung phân lớp trôi khái niệm cải tiến từ khung Meta-ADD, là VAR-WIND và MetaLDD-Finetune. Các khung nêu trên đều sử dụng mạng nguyên mẫu từ đề xuất của J. Snell và cộng sự [39]. Phần đầu của chương giới thiệu mạng nguyên mẫu học ít mẫu. Phần tiếp theo giải thích về khung phân lớp trôi Meta-ADD và chỉ ra những vấn đề còn tồn tại của khung này. Từ đó luận án đề xuất khung VAR-WIND trong đó cải tiến thành phần trích xuất đặc trưng của Meta-ADD bằng cách sử dụng các chiến lược cửa sổ khác nhau cho mỗi kiểu trôi khái niệm. Khung MetaLDD-Finetune đề xuất thêm giai đoạn tinh chỉnh sau giai đoạn tiền huấn luyện để gia tăng độ chính xác phân lớp và tính thích nghi của mô hình.
- ***Chương 4*** trình bày đề xuất cải tiến khung MetaLDD-Finetune thành khung CS&AM_SC, trong đó phân tích và đưa ra phương pháp tối ưu không gian biểu diễn đặc trưng trong mạng nguyên mẫu, sử dụng: (i) độ tương đồng Cosine thay cho sử dụng tích vô hướng (ii) biên góc và (iii) hàm mất mát trung tâm, nhằm giảm biến thiên trong lớp và phân tách giữa các lớp rõ ràng.

Chương 1. Giới thiệu chung về trôi khái niệm và xử lý trôi khái niệm

Chương này trình bày một cách hệ thống về xử lý trôi khái niệm, bao gồm các khái niệm về trôi khái niệm và liên quan, các bài toán, phương pháp, xu hướng, một số bộ dữ liệu và độ đo đánh giá phổ biến trong xử lý trôi khái niệm. Mục đầu tiên trình bày về các khái niệm trôi khái niệm, các kiểu trôi khái niệm và xử lý trôi khái niệm. Ba mục tiếp theo giới thiệu về ba bài toán chính trong xử lý trôi khái niệm là Phát hiện trôi khái niệm, Hiểu trôi khái niệm và Thích nghi trôi khái niệm. Mục thứ năm giới thiệu về các tập dữ liệu xử lý trôi khái niệm được dùng trong luận án. Đánh giá hiệu năng đối với phát hiện trôi khái niệm được giới thiệu trong mục thứ sáu; ở đây, nhấn mạnh về cách thức đo lường các giá trị trong ma trận nhầm lẫn. Mục tiếp đó giới thiệu về xu hướng xử lý trôi khái niệm và một số luận án Tiến sĩ trên thế giới. Mối liên hệ của các nghiên cứu trong luận án với một số mô hình và phương pháp theo xu hướng xử lý trôi khái niệm được giới thiệu khái quát trong mục thứ tám.

1.1 Trôi khái niệm

1.1.1 Định nghĩa về trôi khái niệm

Định nghĩa 1.1 (Luồng dữ liệu) [49]

Luồng dữ liệu đề cập đến một tập dữ liệu mà mỗi phần tử dữ liệu được gắn với một dấu thời gian tương ứng.

Theo một cách hình thức, luồng dữ liệu là tập $D_{tream} = \{d_t\}, t \in T$ với T là một đoạn số nguyên và d_t là phần tử dữ liệu tại thời điểm t; luồng dữ liệu thường được minh họa như một dòng các phần tử theo thời gian (xem Hình 1.3).

Thông thường, các hệ thống tương tác với luồng dữ liệu chỉ có thể truy cập các phần tử dữ liệu có dấu thời gian trước thời điểm hiện tại. Tuy nhiên, các mô hình được xây dựng từ dữ liệu này nhằm mục đích xử lý và đưa ra dự đoán cho các phần tử dữ liệu có dấu thời gian trong tương lai. Theo [49], quá trình sinh ra luồng dữ liệu có thể được mô hình hóa như một biến ngẫu nhiên Z mà từ đó các đối tượng cụ thể $o \in dom(Z)$ được lấy ra ngẫu nhiên, trong đó, $dom(\cdot)$ biểu thị miền giá trị của một biến ngẫu nhiên. Trong học phân lớp, chúng ta cần phân biệt

một biến lớp hoặc nhãn, $y \in dom(Y)$, trong đó Y biểu thị một biến ngẫu nhiên trên các nhãn lớp và các biến $x \in dom(X)$, trong đó X biểu thị một biến ngẫu nhiên trên các vector của các giá trị thuộc tính. Trong trường hợp này, Z biểu thị phân phối kết hợp XY và o biểu thị bởi một cặp $< x, y >$.

Phân phối dữ liệu là một biểu diễn xác suất mô tả cách mà dữ liệu xuất hiện trong không gian đầu vào và đầu ra. Về mặt toán học, nó thường được biểu diễn dưới dạng phân phối kết hợp giữa đặc trưng đầu vào X và nhãn mục tiêu Y.

"Khái niệm" đề cập đến mối quan hệ cơ bản giữa các thuộc tính của dữ liệu và nhãn của chúng. Về mặt xác suất, khái niệm được định nghĩa là phân phối kết hợp giữa đặc trưng đầu vào X và nhãn mục tiêu Y, ký hiệu là $P(X, Y)$.

Định nghĩa 1.2 (Khái niệm) [49]

$$\text{Khái niệm} = P(X, Y) \quad (1.1)$$

Khái niệm tại một thời điểm t được biểu diễn như sau:

$$\text{Khái niệm} = P_t(X, Y) \quad (1.2)$$

Ngoài việc học phân lớp (ví dụ trong trường hợp học không giám sát, khi không có thuộc tính lớp đặc biệt), thì có thể biểu diễn khái niệm như sau:

$$\text{Khái niệm} = P(Z) \quad (1.3)$$

Một khái niệm không phải lúc nào cũng tĩnh trong luồng dữ liệu vì có thể khái niệm đó thay đổi theo thời gian do những biến đổi trong quá trình sinh dữ liệu cơ bản. Đây là một yếu tố cần quan tâm khi xử lý các luồng dữ liệu có hiện tượng trôi khái niệm.

Cho một khoảng thời gian $[0, t]$, một tập các mẫu, $S_{0,t} = d_0, \dots, d_t$ trong đó $d_i = (X_i, y_i)$ là một quan sát (hoặc một thể hiện) với X_i là vector đặc trưng, y_i là nhãn, và $S_{0,t}$ tuân theo một phân phối định rõ $F_{0,t}(X, y)$.

Định nghĩa 1.3 (Trôi khái niệm) [28]

Trôi khái niệm xuất hiện tại thời điểm $t + 1$ nếu:

$$F_{0,t}(X, y) \neq F_{0,t+1}(X, y), \text{ được ký hiệu là } \exists t: P_t(X, y) \neq P_{t+1}(X, y). \quad (1.4)$$

Như vậy, trôi khái niệm tại thời điểm t được định nghĩa là sự thay đổi xác suất đồng thời của X và y tại thời điểm t theo công thức $P_t(X, y) = P_t(X) \times P_t(y|X)$.

Ví dụ trong bài toán "Dự báo thời tiết":

- Đầu vào: các đặc trưng khí tượng như nhiệt độ, độ ẩm, áp suất $X = (x_1, x_2, \dots, x_n)$
- Nhãn: $y = 1$ nếu ngày mai mưa, $y = 0$ nếu không.

Ban đầu (giả sử ở tháng ba của năm), phân phối có thể là:

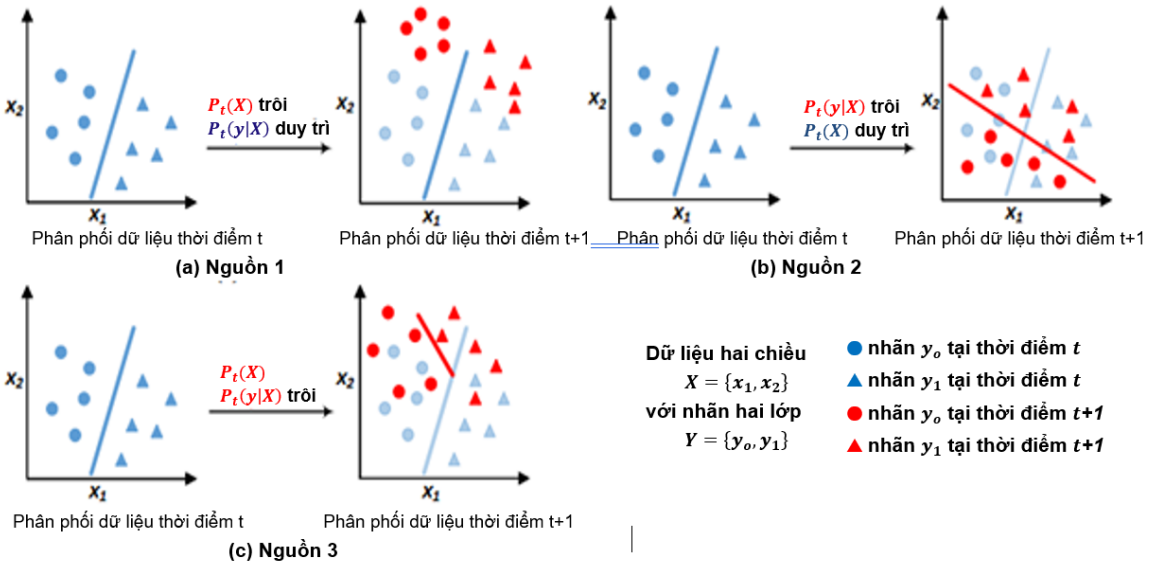
$$P_{t1}(y = 1 \mid X) = f_1(X)$$

Sau đó (tháng sáu), do thay đổi khí hậu hoặc mùa mưa đến, mối quan hệ này thay đổi:

$$P_{t2}(y = 1 \mid X) = f_2(X) \neq f_1(X)$$

Như vậy mô hình học từ dữ liệu tháng ba sẽ sai lệch khi áp dụng cho tháng sáu, vì $P(Y|X)$ đã thay đổi.

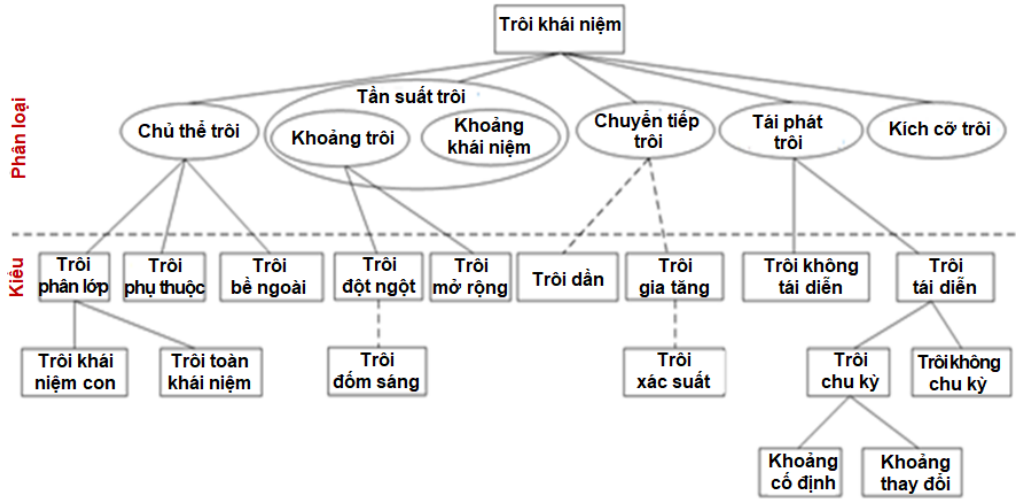
Như vậy, trôi khái niệm sẽ được phát hiện dựa vào sự thay đổi của $P_t(X)$, của $P_t(y|X)$, của cả $P_t(X)$ và $P_t(y|X)$. Trường hợp khi $P_t(X)$ thay đổi mà không làm thay đổi $P_t(y|X)$ thì được gọi là trôi khái niệm ảo (virtual concept drift) hay thay đổi hiệp phương sai (covariate shift) để phân biệt với trôi khái niệm thực (real concept drift) có sự thay đổi của $P_t(y|X)$ [19], [29] Trôi khái niệm ảo không thuộc chủ đề nghiên cứu của luận án này. Hình 1.1 cho một minh họa về ba nguồn trôi khái niệm.



Hình 1.1. Ba nguồn trôi khái niệm [28]

Hình 1.2 minh họa một khung phân loại xử lý trôi khái niệm được G. I. Webb và cộng sự đề nghị [49]. Lưu ý rằng, các đường liền chỉ ra rằng các khái niệm con

tạo thành một tập hợp hoàn chỉnh các lựa chọn thay thế loại trừ lẫn nhau, các đường đứt nét có nghĩa là các khái niệm con không loại trừ nhau.



Hình 1.2 Khung phân loại xử lý trôi khái niệm [49]

1.1.2 Các kiểu trôi khái niệm

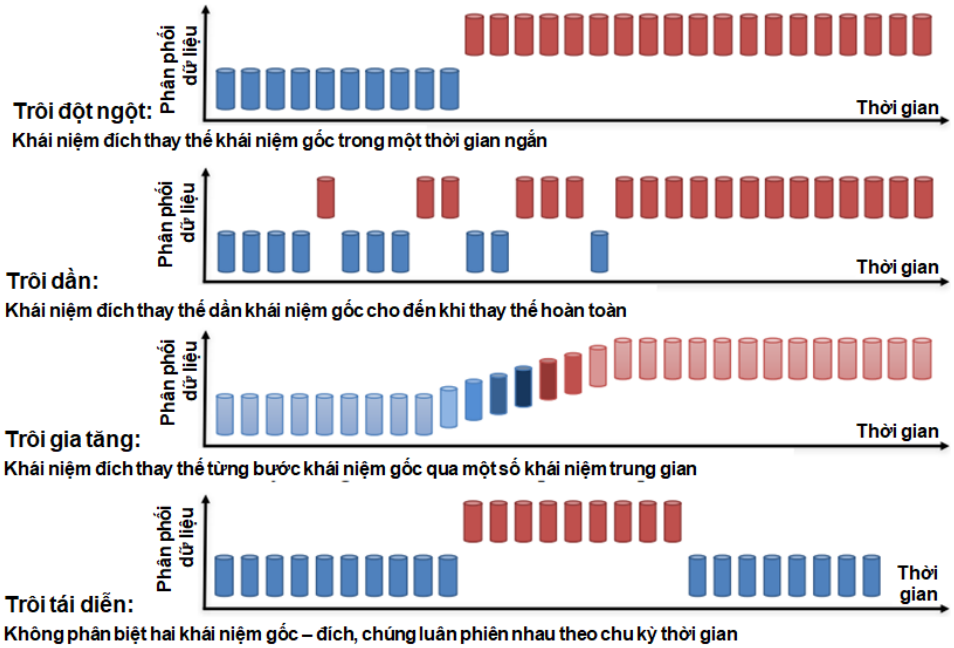
Trôi khái niệm được phân thành bốn kiểu là đột ngột (*sudden*), dần dần (*gradual*), gia tăng (*incremental*), tái diễn (*reoccuring*) như được minh họa ở Hình 1.3.

Trôi đột ngột xảy ra khi một khái niệm (*khái niệm gốc*) được thay thế tức thời bằng một khái niệm khác (*khái niệm đích*).

Trong trôi dần dần, khái niệm đích thay thế dần khái niệm gốc trong một khoảng thời gian cho tới thời điểm khái niệm đích thay thế hoàn toàn khái niệm gốc.

Trong trôi gia tăng, khái niệm gốc được chuyển hóa từng bước qua *các khái niệm trung gian* và cuối cùng tới khái niệm đích.

Trôi tái diễn xảy ra khi một khái niệm từng xuất hiện trong quá khứ quay trở lại sau một khoảng thời gian. Kiểu trôi này không phân biệt khái niệm gốc và khái niệm đích, hai khái niệm này luân phiên nhau một cách định kỳ.

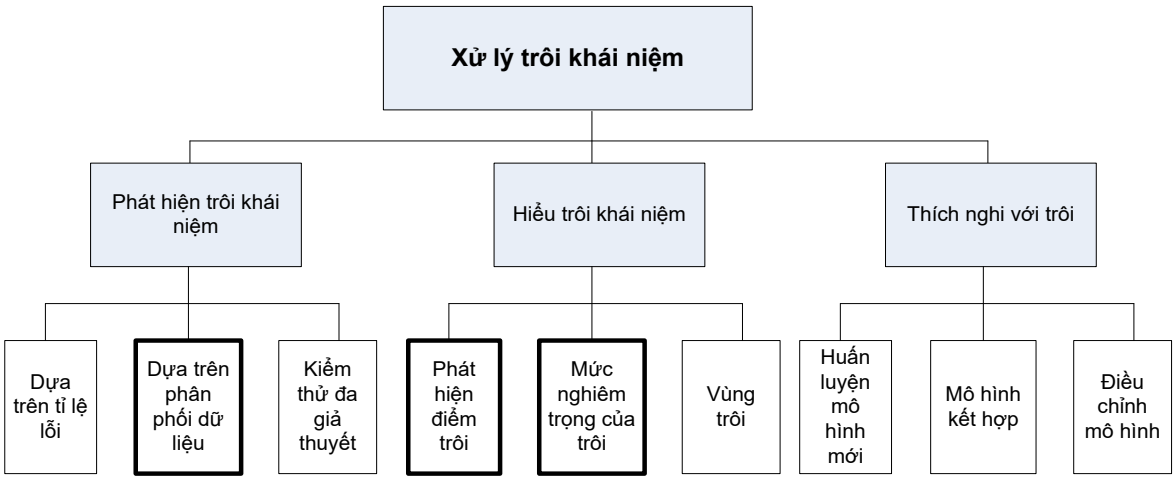


Hình 1.3. Các kiểu trôi khái niệm [55]

Một trong những thách thức của xử lí trôi khái niệm là không được lẫn trôi khái niệm với ngoại lai hoặc nhiễu mà chỉ là sự chuyển ngẫu nhiên trong một khung thời gian ngắn.

1.1.3 Xử lý trôi khái niệm

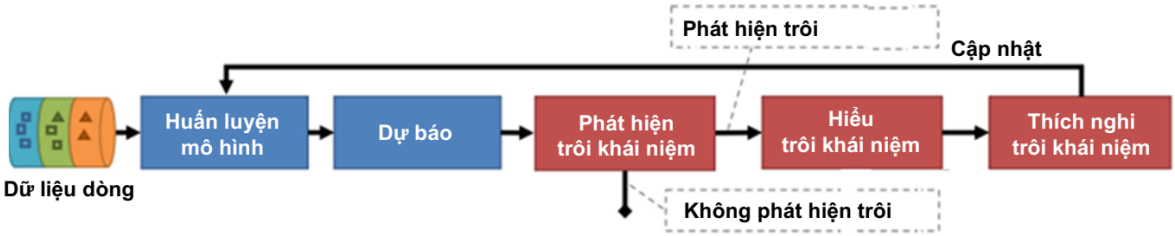
Hình 1.4 cung cấp một khung phân loại các bài toán xử lý trôi khái niệm.



Hình 1.4 Các bài toán liên quan đến xử lý trôi khái niệm

Một khung quy trình chung xử lý trôi khái niệm được minh họa tại Hình 1.5. Đầu tiên, các dữ liệu liên tục từ luồng dữ liệu được đưa vào mô hình để tiến hành huấn luyện ban đầu. Sau đó, mô hình sử dụng kiến thức đã học để thực hiện dự

báo. Trong quá trình dự báo, một cơ chế phát hiện trôi khái niệm được kích hoạt nhằm kiểm tra sự thay đổi trong dữ liệu.



Hình 1.5. Khung xử lý trôi khái niệm [28]

Nếu hiện tượng trôi được phát hiện, hệ thống sẽ thực hiện hai bước tiếp theo: (i) hiểu rõ loại trôi khái niệm đang xảy ra nhằm xác định bản chất của sự thay đổi, và (ii) thích nghi với trôi khái niệm thông qua việc cập nhật mô hình học. Ngược lại, nếu không phát hiện thấy trôi, hệ thống tiếp tục quá trình huấn luyện và dự báo mà không cần điều chỉnh. Vòng lặp này đảm bảo rằng mô hình luôn duy trì hiệu năng dự đoán cao, ngay cả khi môi trường dữ liệu thay đổi theo thời gian.

Theo Agrahari và cộng sự [2], xử lý luồng dữ liệu (xử lý trôi khái niệm, nói riêng) cần đáp ứng bốn yêu cầu: (i) Xử lý một ví dụ nhiều nhất một lần; (ii) Có giới hạn bộ nhớ để học một ví dụ; (iii) Yêu cầu xử lý trong một khoảng thời gian giới hạn; (iv) Sẵn sàng dự đoán nhãn lớp bất cứ khi nào cần. Chính vì lý do đó, các phương pháp học ít mẫu "few-shot learning" (thậm chí học một mẫu "one-shot learning") là phổ biến trong mô hình phát hiện và phân lớp trôi khái niệm.

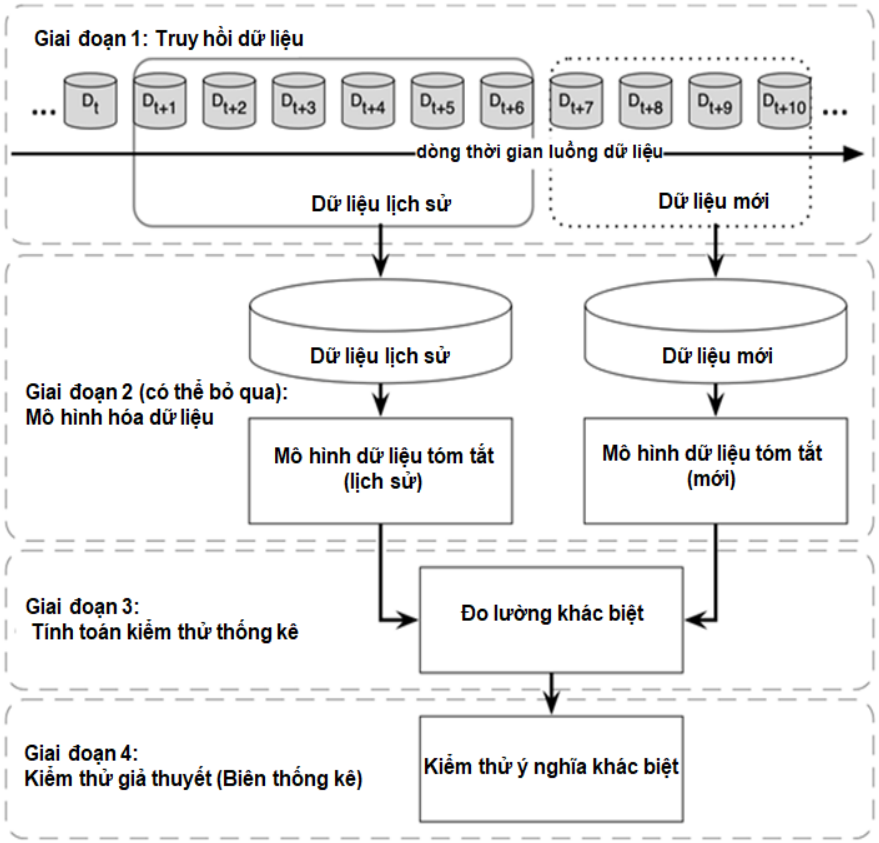
1.2 Phát hiện trôi khái niệm

Cho trước một luồng dữ liệu D_{tream} và một khái niệm $P(X, Y)$ với X là một biến đặc trưng đầu vào và Y là lớp mục tiêu, phát hiện trôi khái niệm là kiểm tra xem khái niệm $P(X, Y)$ trong D_{tream} có bị trôi hay không. Khung tổng quát phát hiện trôi khái niệm bao gồm bốn giai đoạn như được minh họa trong Hình 1.6.

Giai đoạn 1: Truy hồi dữ liệu nhằm thu các đoạn dữ liệu từ các luồng dữ liệu. Vì một mẫu dữ liệu đơn lẻ không thể có đủ thông tin để suy luận trên toàn bộ phân phối, hiểu được cách tổ chức các đoạn dữ liệu để xây dựng các mẫu hay tri thức có ý nghĩa là một tác vụ phân tích quan trọng.

Giai đoạn 2: Mô hình hóa dữ liệu nhằm bao gói các tập dữ liệu thu được và trích chọn thành các đặc trưng mà chứa đựng các thông tin có tính nhận dạng tốt, hay nói cách khác là bước này sẽ chọn ra các đặc trưng có ý nghĩa trong việc đánh

giá hệ thống khi nó thay đổi. Giai đoạn này có thể bỏ qua bởi vì đa số các trường hợp phải giảm chiều dữ liệu (nhằm tối ưu thời gian tính toán và tiết kiệm lưu trữ) và không giữ được các trường dữ liệu gốc.



Hình 1.6 Khung tổng quát phát hiện trôi khái niệm [55]

Giai đoạn 3 Tính toán kiểm thử thống kê là bước tính toán khoảng cách để xác định có sự thay đổi hay không. Giai đoạn này là khó khăn nhất trong tất cả các giai đoạn, với một thách thức chính là xác định độ đo.

Giai đoạn 4 Kiểm thử giả thuyết sử dụng một kiểm thử giả thuyết để kiểm thử một giả thuyết cụ thể để đánh giá tính thống kê các thay đổi xác định được ở giai đoạn 3. Việc đo lường sự khác biệt hoặc khoảng cách giữa các phân phối dữ liệu là công việc ở bước 3, tuy nhiên, những thống kê thô này không thể tự mình xác định liệu một thay đổi có ý nghĩa hay chỉ đơn giản là do nhiễu hoặc ngoại lai. Giai đoạn 4 sử dụng kiểm định giả thuyết để xác nhận liệu những thay đổi quan sát được có ý nghĩa thống kê hay không. Điều này giúp phân biệt trôi khái niệm thực sự với những dao động ngẫu nhiên có trong dữ liệu.

Trên cơ sở khung tổng quát phát hiện trôi khái niệm trên đây, tồn tại các nhóm phương pháp và thuật toán phát hiện trôi khái niệm là: dựa trên tỷ lệ lỗi, dựa trên phân phối dữ liệu, kiểm thử đa giả thuyết và dựa trên học kết hợp [55].

1.2.1 Dựa trên tỷ lệ lỗi

Trong phương pháp này, các đại lượng thống kê liên quan đến tỷ lệ lỗi của bộ phân lớp được khai thác để xác định sự xuất hiện của trôi khái niệm. Cụ thể, khi trôi khái niệm xảy ra, tỷ lệ lỗi của bộ phân lớp có xu hướng tăng đột ngột hoặc biến đổi dần theo thời gian.

Các thuật toán phát hiện trôi khái niệm dựa trên phương pháp này chủ yếu tập trung vào việc theo dõi sự biến động của tỷ lệ lỗi và áp dụng các ngưỡng hoặc tiêu chí đánh giá cụ thể để xác định thời điểm trôi khái niệm diễn ra. Các ngưỡng này có thể được thiết lập dựa trên phân tích thống kê hoặc học từ dữ liệu thực nghiệm, nhằm đảm bảo tính chính xác và hiệu quả trong phát hiện trôi khái niệm.

Một trong những thuật toán phát hiện trôi khái niệm được nghiên cứu và ứng dụng rộng rãi là thuật toán DDM (Drift Detection Method) [Gama04]. Đây là thuật toán đầu tiên xác định hai ngưỡng "cảnh báo" và "trôi" nhằm hỗ trợ quá trình phát hiện trôi khái niệm một cách hiệu quả.

EDDM [Baena-García06] là một phương pháp để phát hiện sớm hiện tượng trôi khái niệm trong dữ liệu luồng, đặc biệt hiệu quả với các thay đổi dần dần. Phương pháp này cải thiện so với phương pháp DDM bằng cách không chỉ dựa vào tỷ lệ lỗi mà còn xem xét khoảng cách giữa các lỗi phân loại, giúp phát hiện sự thay đổi sớm hơn mà không làm tăng tỷ lệ cảnh báo sai.

Một thuật toán phát hiện trôi phổ biến khác dựa trên hai cửa sổ thời gian là ADaptive WINdowing với hai phiên bản là ADWIN và ADWIN2 [Bifet07]. Thay vì sử dụng cửa sổ trượt có kích thước cố định như các phương pháp truyền thống, ADWIN tự động điều chỉnh kích thước cửa sổ dựa trên tốc độ thay đổi của dữ liệu. Khi dữ liệu ổn định, cửa sổ sẽ mở rộng để tận dụng tối đa thông tin lịch sử; ngược lại, khi phát hiện sự thay đổi trong phân phối dữ liệu, thuật toán sẽ thu nhỏ cửa sổ để phản ứng nhanh với các mẫu mới. ADWIN2 cải thiện ADWIN bằng cách tối ưu hóa hiệu suất bộ nhớ và thời gian xử lý, nhờ áp dụng các kỹ thuật từ lĩnh vực thuật toán luồng dữ liệu.

1.2.2 Dựa trên phân phối dữ liệu

Phát hiện trôi khái niệm dựa trên sự thay đổi trong phân phối dữ liệu là một phương pháp quan trọng trong giám sát luồng dữ liệu, giúp nhận diện sự biến động trong phân phối xác suất của các đặc trưng hoặc đầu ra theo thời gian. Nguyên lý cốt lõi của phương pháp này là khi trôi khái niệm xảy ra, phân phối của các điểm dữ liệu thay đổi đáng kể. Do đó, việc so sánh phân phối của các mẫu hiện tại với phân phối trong quá khứ có thể giúp phát hiện những khác biệt có ý nghĩa thống kê, từ đó xác định sự xuất hiện của trôi khái niệm.

Luồng dữ liệu thường được chia thành các cửa sổ thời gian. Trong mỗi cửa sổ, phân phối dữ liệu được ước lượng bằng các phương pháp khác nhau, ví dụ Phân phối thực nghiệm, Ước lượng mật độ hạt nhân (Kernel Density Estimation - KDE), Biểu đồ tần suất… Để đo lường sự khác biệt giữa các phân phối, người ta thường sử dụng các độ đo như Kullback-Leibler Divergence (K-LD), Jensen-Shannon Divergence (J-SD), Kolmogorov-Smirnov Test (K-ST), hàm hạt nhân, Wilcoxon Test (Wilcoxon). Khi độ đo khoảng cách vượt qua một ngưỡng nhất định, trôi được phát hiện.

Một trong những nghiên cứu điển hình trong phương pháp này được đề xuất bởi tác giả Kifer và cộng sự [27]. Tác giả đề xuất một phương pháp phát hiện sự thay đổi dựa trên mô hình hai cửa sổ trượt, trong đó một cửa sổ tham chiếu giữ dữ liệu cũ và một cửa sổ hiện tại cập nhật dữ liệu mới. Bằng cách so sánh sự khác biệt thống kê giữa hai cửa sổ này, thuật toán có thể xác định thời điểm xảy ra sự thay đổi trong phân phối dữ liệu.

Nhóm tác giả Haug và cộng sự đề xuất một khung phát hiện trôi có tên là ERICS (Effective and Robust Identification of Concept Shift) [24], coi tham số của mô hình dự đoán như những biến ngẫu nhiên, dựa trên một nhận định rằng có sự tương ứng giữa trôi khái niệm và sự thay đổi phân bố của các tham số tối ưu.

1.2.3 Kiểm thử đa giả thuyết

Phát hiện trôi khái niệm dựa trên kiểm định đa giả thuyết cũng sử dụng các kỹ thuật phát hiện trôi giống như hai phương pháp đã nêu trên. Sự khác biệt đó là phương pháp này thực hiện đồng thời nhiều phép kiểm định thống kê trên các cửa sổ dữ liệu nhằm phát hiện trôi khái niệm. Ưu điểm nổi bật của phương pháp này là đảm bảo tính chính xác và độ tin cậy cao nhờ điều chỉnh sai số khi thực hiện

nhiều kiểm định, đồng thời phát hiện được cả trôi đột ngột lẫn trôi dần dần. Tuy nhiên, chi phí tính toán là một thách thức lớn, đặc biệt khi xử lý các luồng dữ liệu tốc độ cao và đa chiều. So với phương pháp dựa trên tỉ lệ lỗi, phương pháp kiểm định nhiều giả thuyết không phụ thuộc vào nhãn dữ liệu, do đó phù hợp với các tình huống nhãn bị trễ hoặc thiếu hụt. So với phương pháp dựa trên phân phối dữ liệu, phương pháp này tăng cường độ tin cậy nhờ kiểm soát chặt chẽ sai số thống kê.

1.2.4 Học kết hợp phát hiện trôi khái niệm

Bên cạnh ba hướng nghiên cứu truyền thống đã nêu, một trong những hướng nghiên cứu triển vọng về kỹ thuật phát hiện trôi khái niệm là học kết hợp (ensemble learning) là một phương pháp mạnh mẽ trong học máy, kết hợp nhiều mô hình để cải thiện hiệu năng dự đoán, tính ổn định và khả năng tổng quát hóa. Bằng cách tận dụng ưu điểm của các mô hình đa dạng, phương pháp kết hợp giúp khắc phục nhược điểm của từng mô hình riêng lẻ, dẫn đến các dự đoán chính xác và đáng tin cậy hơn.

Dietterich và cộng sự [14] đưa ra định nghĩa về học kết hợp như sau: "một phương pháp học xây dựng một tập hợp các bộ phân lớp (mô hình cơ sở) và sau đó phân lớp các điểm dữ liệu mới bằng cách lấy (bỏ phiếu có trọng số) từ các dự đoán của chúng". Trong định nghĩa trên thì "*mô hình cơ sở*" là một mô hình học máy đơn giản và "*phiếu*" được biểu diễn bằng một số thực gọi là trọng số.

Các mô hình học máy thường được thiết kế để giải quyết các vấn đề phức tạp bằng cách học các mẫu từ dữ liệu. Tuy nhiên, các mô hình riêng lẻ có thể gặp phải những hạn chế như quá khớp, độ lệch cao hoặc phương sai cao. Học kết hợp giải quyết những vấn đề này bằng cách kết hợp các dự đoán từ nhiều mô hình, thường mang lại hiệu năng vượt trội so với một mô hình đơn lẻ. Ý tưởng cốt lõi là một nhóm các "mô hình yếu" có thể cùng nhau tạo thành một "mô hình mạnh".

Một mô hình kết hợp tính tổng các phiếu bầu từ các mô hình cơ sở bằng cách cộng dồn các đóng góp của chúng. Trong một số bài toán phân lớp nhị phân, một phiếu bầu a_i thường có giá trị trong tập 0,1, trong đó:

- $a_i = 1$ nếu mô hình cơ sở thứ i dự đoán điểm dữ liệu thuộc lớp dương.
- $a_i = 0$ nếu mô hình cơ sở thứ i dự đoán điểm dữ liệu thuộc lớp âm.

Mỗi phiếu bầu đi kèm với một trọng số w_i. Để quyết định dự đoán cuối cùng, mô hình tổng hợp so sánh giá trị trung bình có trọng số của các phiếu bầu với một ngưỡng T (thông thường, $T = 0.5$). Mô hình tổng hợp sẽ nhận diện một điểm dữ liệu thuộc lớp dương nếu thỏa mãn điều kiện sau:

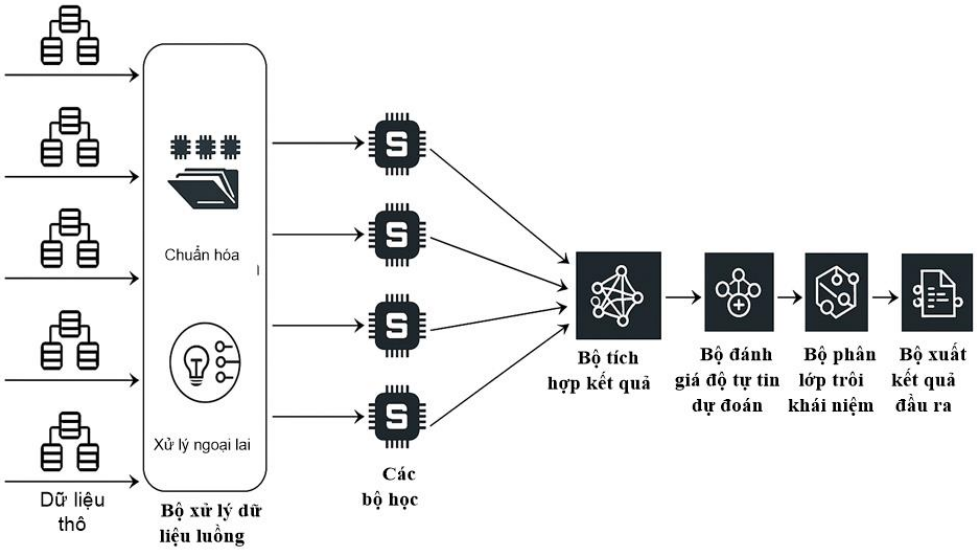
$$P(\text{ output} = 1 \mid a) \geq T \tag{2.1}$$

$$\Leftrightarrow \frac{\sum_{i=1}^{N} a_i w_i}{\sum_{i=1}^{N} w_i} \geq T \tag{2.2}$$

Ngược lại, nếu không thỏa mãn điều kiện trên, điểm dữ liệu sẽ được xếp vào lớp âm.

Abbasi và cộng sự [1] đề xuất mô hình học kết hợp ElStream để phát hiện trôi khái niệm trong luồng dữ liệu lớn, đặc biệt là dữ liệu từ mạng xã hội có tính động cao, sử dụng một tổ hợp linh hoạt các bộ học, mỗi bộ được huấn luyện để hoạt động trong các điều kiện phân phối dữ liệu khác nhau, và toàn bộ hệ thống có thể thích nghi khi phát hiện có trôi khái niệm. Các thành phần cơ bản của ElStream được minh họa trên Hình 1.7, bao gồm:

- Bộ xử lý dữ liệu luồng: phân tích luồng dữ liệu đầu vào và trích xuất đặc trưng.
- Các bộ học: là trung tâm của mô hình, nơi tập hợp các bộ phân lớp. Sau khi tiền xử lý, dữ liệu được đưa qua nhiều bộ phân lớp, mỗi bộ là một mô hình học máy được huấn luyện độc lập. Thành phần các bộ học trong Elstream bao gồm nhiều mô hình học máy chạy song song, như rừng ngẫu nhiên (Random Forest), Thuật toán tăng cường độ dốc cực đại (XGBoost), Mạng Perceptron nhiều lớp (Multilayer Perceptron - MLP), k láng giềng gần nhất (kNN)… Mỗi mô hình học một phần khác nhau của không gian đặc trưng hoặc trạng thái dữ liệu.
- Bộ tích hợp kết quả (Meta Classifier): tích hợp kết quả từ các bộ phân lớp bằng cơ chế bỏ phiếu. Tuy nhiên, chỉ những mô hình có độ tự tin cao hơn ngưỡng (ví dụ 80%) mới được tham gia biểu quyết. Nhờ đó, hệ thống lọc ra những dự đoán đáng tin cậy, tránh ảnh hưởng của các mô hình yếu trong trường hợp xảy ra trôi khái niệm mạnh.



Hình 1.7 Mô hình Elstream [1]

- Bộ đánh giá độ tự tin dự đoán (Prediction confidence): để đánh giá độ tự tin của dự đoán hợp nhất, nếu độ tự tin vượt ngưỡng, hệ thống tiếp tục; nếu không, cần thêm thông tin.
- Bộ phân lớp trôi khái niệm (Drift class): thực hiện phân lớp dữ liệu hiện tại thuộc kiểu trôi khái niệm nào hoặc không có trôi.
- Bộ xuất kết quả đầu ra (Output classification): đưa ra kết quả.

Khi phát hiện có sự giảm sút hiệu năng (theo dõi thông qua các chỉ số lỗi hoặc độ chính xác), bộ phát hiện trôi sẽ kích hoạt quá trình tái huấn luyện hoặc cập nhật trọng số của các mô hình trong tổ hợp. ElStream sử dụng kỹ thuật học trực tuyến cho phép cập nhật mô hình theo thời gian thực, không cần lưu trữ toàn bộ dữ liệu quá khứ. Khi phát hiện thấy có trôi khái niệm, những mô hình kém hiệu quả trong tổ hợp sẽ được thay thế. Các mô hình mới được huấn luyện từ dữ liệu mới nhất được thêm vào tổ hợp.

Một nghiên cứu điển hình khác dựa trên tập hợp có chọn lọc các bộ phát hiện là e-Detector [15] nhằm nâng cao hiệu quả phát hiện trôi khái niệm trong luồng dữ liệu, đặc biệt khi dữ liệu có nhiều loại trôi khác nhau để khắc phục các hạn chế của các phương pháp phát hiện trôi đơn lẻ, thường chỉ hiệu quả với một loại trôi nhất định, dẫn đến khả năng phát hiện sai hoặc bỏ sót điểm trôi trong môi trường phức tạp.

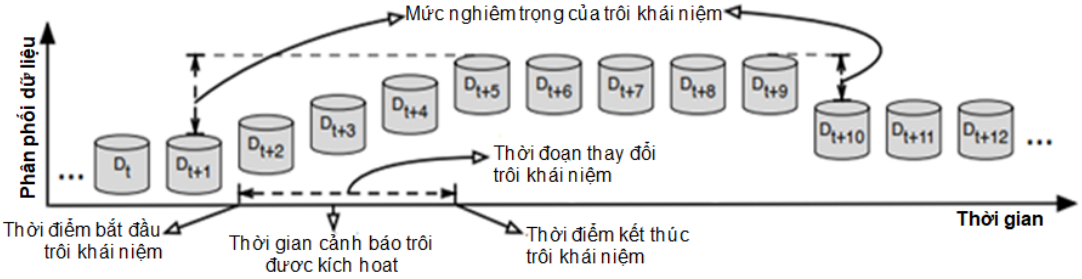
1.3 Hiểu trôi khái niệm

Vấn đề hiểu về trôi khái niệm bao gồm xác định các thông tin về thời gian trôi, phân lớp trôi (được giới thiệu chi tiết trong mục con 1.3.4), mức độ nghiêm trọng trôi và vùng trôi.

1.3.1 Thời gian trôi khái niệm

Hiểu về thời gian trôi gồm các mốc thời gian như thời điểm bắt đầu trôi (điểm trôi), quãng thời gian thay đổi khái niệm, thời điểm kết thúc trôi như minh họa tại Hình 1.8. Những thông tin thời gian này có thể đóng vai trò là đầu vào quan trọng cho quá trình thích nghi của hệ thống học máy.

Theo định nghĩa của trôi khái niệm như được mô tả tại Công thức (1.4) biến t biểu thị thời gian mà hiện tượng trôi khái niệm xảy ra. Trong các phương pháp hoặc thuật toán phát hiện trôi, một tín hiệu cảnh báo được sử dụng để chỉ ra liệu hiện tượng trôi khái niệm đã xảy ra hay chưa tại thời điểm hiện tại. Tín hiệu này cũng là thông báo cho hệ thống học máy thích nghi với một khái niệm mới. Việc xác định chính xác điểm trôi là yếu tố rất quan trọng đối với quá trình thích nghi của hệ thống học máy; *sự chậm trễ hoặc cảnh báo sai* có thể dẫn đến việc hệ thống không theo dõi được các khái niệm mới. Do đó, cần làm thế nào để phát hiện điểm trôi càng sớm càng tốt.



Hình 1.8. Ví dụ về thời điểm và mức nghiêm trọng của trôi khái niệm [28]

1.3.2 Mức độ nghiêm trọng của trôi khái niệm

Hình 1.8 cũng cho một ví dụ minh họa về mức độ nghiêm trọng của trôi khái niệm: Một trôi gia tăng bắt đầu thay đổi tại $t + 1$ và kết thúc tại $t + 5$, một trôi đột ngột xảy ra giữa $t + 9$ và $t + 10$. Mức độ nghiêm trọng của hai trôi khái niệm này (D_{t+1} - D_{t+5} và D_{t+9} - D_{t+10}) được chỉ dẫn trên hình.

Hiểu về mức độ nghiêm trọng của trôi khái niệm, được định nghĩa là sử dụng một giá trị định lượng để đo lường mức độ tương đồng giữa khái niệm mới và khái niệm trước đó. Về mặt hình thức, mức độ nghiêm trọng của trôi có thể được biểu diễn bởi $\Delta = \delta\big(P_t(X,y), P_{t+1}(X,y)\big)$, trong đó δ là một hàm đo lường sự khác biệt giữa hai phân phối dữ liệu, và t là thời điểm xảy ra hiện tượng trôi. Giá trị Δ thường là một số không âm, phản ánh giá trị mức độ nghiêm trọng của trôi khái niệm. Giá trị Δ càng lớn thì mức độ nghiêm trọng của trôi khái niệm càng lớn.

1.3.3 Vùng trôi khái niệm

Hiểu về vùng trôi của hiện tượng trôi khái niệm là xác định vùng xung đột giữa khái niệm mới và khái niệm trước đó. Các vùng trôi này được xác định bằng cách tìm các vùng trong không gian đặc trưng dữ liệu X nơi $P_t(X,y)$ và $P_{t+1}(X,y)$ có sự khác biệt đáng kể.

1.3.4 Phân lớp trôi khái niệm

Phân lớp trôi khái niệm là bước tiếp theo của phát hiện trôi khái niệm nhằm xác định cụ thể là kiểu trôi nào đã xảy ra, có nghĩa là, nếu khái niệm $P(X,Y)$ bị trôi trong D_{tream} thì cần chỉ ra là khái niệm đó đã bị trôi sang một khái niệm mới theo kiểu nào.

Việc xác định kiểu trôi khái niệm giúp tối ưu hóa tài nguyên tính toán. Nếu trôi xảy ra đột ngột và có tác động lớn, hệ thống có thể cần tái huấn luyện mô hình ngay lập tức với mức tài nguyên cao. Ngược lại, nếu trôi diễn ra dần dần, việc cập nhật trực tuyến với tốc độ phù hợp sẽ giúp giảm tải tài nguyên mà vẫn đảm bảo hiệu quả mô hình.

Hiểu rõ bản chất của trôi khái niệm giúp thiết lập các ngưỡng phát hiện phù hợp, đảm bảo độ chính xác trong việc nhận diện sự thay đổi. Nếu hệ thống quá nhạy, có thể dẫn đến nhiều cảnh báo giả; ngược lại, nếu kém nhạy, có thể bỏ sót các sự kiện trôi quan trọng, ảnh hưởng đến hiệu suất của mô hình dự đoán.

Có hai nhóm phương pháp chính phân lớp trôi khái niệm [46]:

- Các nghiên cứu của phương pháp phân lớp kiểu trôi dựa trên phân phối dữ liệu có đặc điểm chung là định lượng quãng thời gian thay đổi khái niệm để xác định kiểu trôi. Guo và cộng sự [22], H. Wang [48] cung cấp hai nghiên cứu điển hình trong nhóm này.

- Dựa trên học cách học (Meta learning). Theo Santoro và cộng sự [Santoro16], học cách học là quá trình mô hình học từ nhiều tác vụ khác nhau ở cấp độ ngắn hạn và cấp độ dài hạn. Ở mỗi tác vụ (cấp ngắn hạn), mô hình học nhanh để đưa ra dự đoán trong một phạm vi hẹp, nhưng qua nhiều tác vụ khác nhau (cấp dài hạn), nó tích lũy kiến thức chung về cách mà các mẫu và cấu trúc nhiệm vụ thay đổi giữa các miền dữ liệu khác nhau. Quá trình hai tầng này thường được mô tả là mô hình đang học cách học. Một mô hình học cách học có thể học cách đánh giá độ tương đồng giữa các mẫu dữ liệu để áp dụng cho nhiều tác vụ phân lớp khác nhau mà không cần phải huấn luyện lại hoàn toàn. Như vậy có thể áp dụng phương pháp học cách học để phân lớp các kiểu trôi dựa trên việc đánh giá độ tương đồng giữa các mẫu dữ liệu trôi mới với các kiểu đã học trong quá khứ. Một số nghiên cứu điển hình về sử dụng học cách học để phân lớp trôi là [5], [60], Meta-ADD [55], Type-LDD [56].

Meta-ADD [55], Type-LDD [56] là các nghiên cứu có chung phương pháp sử dụng mạng nguyên mẫu của Snell và cộng sự [39] để thực hiện phân lớp kiểu trôi. Đặc điểm chung của các nghiên cứu này là thực hiện tiền huấn luyện một mạng nguyên mẫu, sử dụng bộ dữ liệu tổng hợp có nhãn về các dạng trôi. Việc tiền huấn luyện là dạy cho mạng học cách đánh giá độ tương đồng giữa các mẫu dữ liệu truy vấn với tâm lớp của từng kiểu trôi. Mô hình thu được sẽ được sử dụng để phát hiện kiểu trôi khái niệm trong luồng dữ liệu mới.

Khung Meta-ADD hoạt động theo hai giai đoạn chính: đầu tiên, trong giai đoạn huấn luyện, hệ thống thu thập các đặc trưng của các kiểu trôi. Sau đó, một bộ phát hiện và phân lớp gọi là Meta-detector dựa trên mạng nguyên mẫu được huấn luyện để học cách phân biệt giữa các mẫu trôi thuộc các kiểu đột ngột, gia tăng, hoặc bình thường. Trong giai đoạn dự đoán, bộ Meta-detector được tinh chỉnh qua một thuật toán học chủ động để liên tục để thích nghi với luồng dữ liệu mới.

Nghiên cứu Type-LDD [56] được mở rộng trực tiếp từ Meta-ADD, vốn chỉ tập trung vào phân lớp kiểu trôi mà không xác định chính xác điểm trôi. Khung này được mở rộng thêm thành phần là bộ định vị điểm trôi – một mô hình hồi quy giúp xác định chính xác thời điểm xảy ra trôi trong luồng dữ liệu. Như vậy Type-LDD gồm hai thành phần: bộ Meta-detector (từ Meta-ADD) và bộ định vị điểm

trôi được huấn luyện đồng thời thông qua hàm mất mát chia sẻ (sharing-loss). Đây là điểm khác biệt nổi bật, giúp Type-LDD không chỉ nhận diện loại trôi mà còn chỉ ra được vị trí trôi với độ trễ thấp và độ chính xác cao hơn rõ rệt.

1.4 Thích nghi trôi khái niệm

Các nghiên cứu về thích nghi trôi khái niệm (*concept drift adaptation*) về ba kiểu trôi khái niệm đầu tiên tập trung vào việc làm thế nào để cực tiểu hoá việc giảm độ chính xác và đạt được sự phục hồi mô hình học nhanh nhất có thể trong suốt quãng thời gian thay đổi xảy ra, ngược lại, kiểu thứ ba thì tập trung vào việc sử dụng phương pháp học dựa trên dữ liệu lịch sử, bằng cách tìm ra một sự thay đổi trùng khít nhất từ dữ liệu đã xảy ra trong lịch sử trong khoảng thời gian ngắn nhất có thể.

Mỗi kiểu trôi khái niệm có thể yêu cầu một phương pháp thích ứng khác nhau. Chẳng hạn, trôi đột ngột có thể yêu cầu khởi tạo lại mô hình một cách nhanh chóng để đảm bảo độ chính xác, trong khi trôi gia tăng hoặc trôi dần dần thường đòi hỏi cập nhật mô hình liên tục nhằm thích nghi với sự thay đổi theo thời gian.

Trong thực tế có nhiều ứng dụng trong việc phát hiện và *thích ứng* với trôi khái niệm. Ví dụ trong an ninh mạng, các mô hình phát hiện xâm nhập sử dụng dữ liệu lịch sử để phân lớp các sự kiện mạng thành hợp lệ hoặc độc hại. Tuy nhiên, các kỹ thuật tấn công liên tục thay đổi, làm cho mô hình cũ trở nên kém hiệu quả. Bằng cách giám sát dữ liệu mạng để phát hiện sự thay đổi trong các mẫu hành vi của lưu lượng truy cập, từ đó cập nhật các mô hình phát hiện tấn công theo thời gian. Vấn đề thích ứng với trôi khái niệm được giải quyết theo ba phương pháp phổ biến đó là: (i) Huấn luyện lại mô hình; (ii) Sử dụng tập hợp các mô hình và (iii) Điều chỉnh mô hình.

1.4.1 Huấn luyện lại mô hình

Phương pháp huấn luyện lại mô hình với dữ liệu mới nhất để thay thế mô hình cũ (đã lỗi thời) cần có một bộ phát hiện trôi để biết được khi nào cần thực hiện huấn luyện lại. Phương pháp này thường áp dụng chiến lược cửa sổ nhằm bảo lưu dữ liệu mới nhất cho việc huấn luyện lại và/hoặc dữ liệu cũ để kiểm tra sự thay đổi của phân phối.

1.4.2 Tập hợp các mô hình

Phương pháp sử dụng tập hợp các mô hình thường áp dụng với dạng trôi khái niệm tái diễn. Khi có trôi tái diễn, việc lưu trữ và tái sử dụng các mô hình cũ giúp giảm đáng kể công sức huấn luyện lại. Phương pháp này dùng một tập hợp các bộ phân lớp cơ sở có thể có loại khác nhau hoặc thông số khác nhau. Đầu ra của mỗi bộ phân lớp cơ sở được kết hợp lại với nhau bằng quy tắc bỏ phiếu để dự đoán trên dữ liệu mới.

1.4.3 Điều chỉnh mô hình

Trong phương pháp điều chỉnh mô hình thì khi thiết kế, mô hình đã có khả năng học thích nghi từ dữ liệu thay đổi. Các mô hình này có thể cập nhật một phần khi phát hiện sự thay đổi cục bộ trong phân phối dữ liệu, thay vì phải thay đổi toàn bộ mô hình.

Phương pháp này hiệu quả hơn so với việc huấn luyện lại toàn bộ khi sự trôi khái niệm chỉ xảy ra ở một số vùng cục bộ trong không gian dữ liệu. Thuật toán cây quyết định được sử dụng phổ biến vì có khả năng kiểm tra và thích nghi với từng vùng riêng biệt.

1.5 Một số bộ dữ liệu phổ biến

Tồn tại một số bộ dữ liệu thực tế và tổng hợp (nhân tạo) được xây dựng cho xử lý trôi khái niệm [19, 28, 2], trong đó, bốn bộ dữ liệu thực tế và năm bộ dữ liệu tổng hợp dưới đây được sử dụng trong luận án.

1.5.1 Bốn bộ dữ liệu thực tế

Các bộ dữ liệu thực tế được lấy từ kho lưu trữ các bộ dữ liệu trực tuyến dành cho học máy - UCI Machine Learning Repository[1], bao gồm *Spambase*, *Adult*, *KDD 1999*, *Dota2*.

- Bộ dữ liệu **Spambase**[2] được sử dụng để phân lớp email thành thư rác hoặc không phải thư rác. Đây là một tập dữ liệu đa biến thuộc lĩnh vực khoa học máy tính và thường được áp dụng cho bài toán phân lớp. Bộ dữ liệu bao gồm 4.601 mẫu với 57 đặc trưng, trong đó có cả biến số nguyên và biến số thực.

[1] https://archive.ics.uci.edu/

[2] https://archive.ics.uci.edu/dataset/94/spambase

Các đặc trưng chủ yếu phản ánh tần suất xuất hiện của các từ hoặc ký tự đặc biệt trong nội dung email, giúp mô hình học máy nhận diện thư rác dựa trên mô hình thống kê và xử lý ngôn ngữ tự nhiên. Với tính ứng dụng cao, bộ dữ liệu này thường được sử dụng để phát triển hệ thống lọc thư rác, nghiên cứu về phát hiện gian lận trực tuyến và cải tiến thuật toán phân lớp văn bản.

- Bộ dữ liệu **Adult** (còn được gọi là Census Income Dataset)[3] sử dụng để dự đoán xem thu nhập hàng năm của một cá nhân có vượt quá 50.000 USD hay không dựa trên dữ liệu điều tra dân số. Đây là một tập dữ liệu đa biến (multivariate) thuộc lĩnh vực khoa học xã hội và thường được sử dụng cho bài toán phân lớp. Bộ dữ liệu bao gồm 48.842 mẫu với 14 đặc trưng, trong đó có cả biến phân lớp và biến số nguyên. Nhờ tính đa dạng của các đặc trưng, bộ dữ liệu này thường được sử dụng để nghiên cứu về mô hình phân lớp, phân tích bất bình đẳng thu nhập, hoặc phát triển thuật toán học máy trong lĩnh vực kinh tế - xã hội.
- Bộ dữ liệu **KDD Cup** được sử dụng trong cuộc thi Khai phá tri thức và Khai thác dữ liệu quốc tế lần thứ ba năm 1999 (KDD-99)[4], một sự kiện quan trọng trong lĩnh vực phát hiện xâm nhập mạng. Đây là một tập dữ liệu đa biến, thuộc lĩnh vực khoa học máy tính, và được áp dụng cho bài toán phân lớp. Bộ dữ liệu bao gồm 4.000.000 mẫu, với các đặc trưng thuộc loại biến phân lớp và biến số nguyên. Các đặc trưng trong bộ dữ liệu mô tả các kết nối mạng, trong đó mỗi mẫu đại diện cho một phiên giao tiếp mạng và được gán nhãn là bình thường hoặc tấn công. Nhờ vào quy mô lớn và tính đa dạng của các cuộc tấn công, bộ dữ liệu này được sử dụng rộng rãi trong nghiên cứu an ninh mạng, phát triển hệ thống phát hiện xâm nhập (IDS), và thử nghiệm các thuật toán học máy trong bảo mật mạng.
- Bộ dữ liệu **Dota2**[5] được xây dựng từ trò chơi Dota 2, một trò chơi điện tử nổi tiếng với hai đội, mỗi đội gồm 5 người chơi. Trong mỗi trận đấu, mỗi người chơi sẽ chọn một tướng (hero) duy nhất với những điểm mạnh và điểm yếu riêng. Đây là một tập dữ liệu đa biến, thuộc lĩnh vực trò chơi (Games) và

[3] https://archive.ics.uci.edu/dataset/2/adult

[4] https://archive.ics.uci.edu/dataset/130/kdd+cup+1999+data

[5] https://archive.ics.uci.edu/dataset/367/dota2+games+results

thường được sử dụng cho bài toán phân lớp. Bộ dữ liệu bao gồm 102.944 mẫu, với 115 đặc trưng, trong đó mỗi đặc trưng mô tả các yếu tố liên quan đến lựa chọn tướng, thông tin đội hình, và các chỉ số khác có thể ảnh hưởng đến kết quả trận đấu. Mục tiêu chính khi sử dụng bộ dữ liệu này là xây dựng mô hình dự đoán đội nào sẽ giành chiến thắng, dựa trên các thông tin về tướng được chọn trước khi trận đấu diễn ra. Bộ dữ liệu này thường được ứng dụng trong phân tích chiến thuật, tối ưu hóa lựa chọn đội hình, và nghiên cứu về trí tuệ nhân tạo trong trò chơi.

1.5.2 Năm bộ dữ liệu tổng hợp

Nền tảng Massive Online Analysis (MOA) được tác giả Bifet và cộng sự giới thiệu tại nghiên cứu [7]. Đây là một nền tảng mã nguồn mở được viết bằng Java, thiết kế chuyên biệt cho việc khai thác dữ liệu luồng và đánh giá các thuật toán học máy trong môi trường dữ liệu thay đổi liên tục theo thời gian. MOA hỗ trợ nhiều bộ dữ liệu tổng hợp tổng hợp, mô phỏng hiện tượng trôi khái niệm như *SEA*, *Agrawal*, và *Hyperplane*, *Stagger*, *Sine*,… đồng thời cung cấp các thuật toán học thích nghi và các phương pháp đánh giá hiệu năng trực tuyến. Với khả năng mở rộng cao, dễ tích hợp và bộ công cụ phong phú, MOA đã trở thành một công cụ tiêu chuẩn trong nghiên cứu và thử nghiệm các hệ thống học trực tuyến. Mỗi bộ dữ liệu tổng hợp nói trên có những đặc điểm riêng, phù hợp để mô phỏng các dạng trôi khái niệm khác nhau, góp phần tạo ra các kịch bản thử nghiệm đa dạng và sát với thực tế hơn trong nghiên cứu về học máy trực tuyến.

Dựa trên ý tưởng từ MOA, Scikit-multiflow[6] là một thư viện Python mã nguồn mở sau đó được phát triển chuyên biệt cho học máy trên luồng dữ liệu, cung cấp các công cụ mạnh mẽ để xử lý phân lớp, hồi quy, phân cụm và phát hiện trôi khái niệm trong môi trường dữ liệu thay đổi liên tục. Thư viện hỗ trợ nhiều phương pháp phát hiện trôi cũng như các thuật toán học trực tuyến như cây Hoeffding (Hoeffding Tree), Naïve Bayes, kNN, rừng ngẫu nhiên thích nghi (Adaptive Random Forest), giúp mô hình có thể cập nhật mà không cần huấn luyện lại toàn bộ dữ liệu. Ngoài ra, scikit-multiflow còn cung cấp các thuật toán cụm trực tuyến và bộ tạo dữ liệu luồng để thử nghiệm mô hình. Với API tương tự scikit-learn, thư

[6] https://scikit-multiflow.github.io/

viện này giúp đơn giản hóa việc triển khai mô hình trên dữ liệu động, có ứng dụng rộng rãi trong phát hiện gian lận, an ninh mạng, hệ thống đề xuất, IoT…

Trong các công trình nghiên cứu về xử lý trôi khái niệm, thư viện *Scikit-multiflow*[7] được sử dụng rất phổ biến, trong đó có năm bộ dữ liệu tổng hợp điển hình sau đây [19, 28, 2]:

- Bộ dữ liệu **SEA** là một bộ dữ liệu tổng hợp đơn giản, thường được dùng để đánh giá khả năng thích nghi của mô hình với trôi khái niệm. Mỗi quan sát trong bộ dữ liệu gồm ba thuộc tính thực, trong đó chỉ có hai thuộc tính đầu tiên là có ý nghĩa trong việc phân lớp, thuộc tính thứ ba đóng vai trò nhiễu. Giá trị nhãn trong SEA là nhị phân, được xác định dựa trên tổng trọng số của hai thuộc tính đầu tiên, so sánh với một ngưỡng xác định trước. Các kịch bản trôi khái niệm được thiết lập bằng cách thay đổi ngưỡng phân lớp hoặc điều chỉnh phân phối của các thuộc tính đầu vào. Bộ dữ liệu SEA có thể được tạo từ lớp SEAGenerator trong gói scikit-multiflow với các tham số điều khiển như lựa chọn hàm phân lớp và tỷ lệ nhiễu.
- Bộ dữ liệu **Agrawal** thường được sử dụng để mô phỏng các bài toán liên quan đến tài chính và ngân hàng. Bộ dữ liệu này chứa chín thuộc tính, bao gồm sáu thuộc tính số và ba thuộc tính phân lớp. Các thuộc tính này biểu diễn các thông tin như tuổi, thu nhập, nợ, giới tính, tình trạng hôn nhân, và trình độ học vấn. Nhãn phân lớp là nhị phân, phản ánh quyết định cho vay hoặc từ chối dựa trên tổng hợp các thuộc tính đầu vào. Bộ dữ liệu Agrawal được thiết kế với mười hàm phân lớp, mỗi hàm biểu diễn một quy tắc ra quyết định khác nhau, có thể thay đổi để tạo ra trôi khái niệm đột ngột hoặc dần. Trong thư viện scikit-multiflow, lớp AGRAWALGenerator cung cấp khả năng sinh bộ dữ liệu này với các tham số tùy chỉnh như lựa chọn hàm phân lớp và mức độ nhiễu.
- Bộ dữ liệu **Hyperplane** bao gồm các điểm dữ liệu thuộc không gian nhiều chiều, mỗi điểm được gán nhãn nhị phân dựa trên khoảng cách của nó tới một siêu phẳng trong không gian thuộc tính. Mỗi quan sát chứa một số lượng thuộc tính thực (thông thường là 10), và nhãn được xác định dựa trên tổng hợp tuyến tính của các thuộc tính đó. Đặc điểm nổi bật của bộ dữ liệu này là khả năng thay đổi vị trí và hướng của siêu phẳng theo thời gian, tạo ra quá trình trôi khái

[7] https://scikit-multiflow.readthedocs.io/en/stable/api/api.html

niệm liên tục. Lớp HyperplaneGenerator trong gói scikit-multiflow cung cấp các tùy chỉnh như số lượng thuộc tính, số lượng thuộc tính tham gia vào quá trình trôi khái niệm, và mức độ nhiễu.

- Bộ dữ liệu **Stagger** là một bộ dữ liệu tổng hợp luồng gồm ba đặc trưng chính: kích thước (nhỏ, trung bình, lớn), hình dạng (tròn, vuông, tam giác) và màu sắc (đỏ, xanh dương, xanh lá). Nhãn của dữ liệu được xác định theo các hàm logic khác nhau, chẳng hạn như một đối tượng được gán nhãn dương nếu có kích thước nhỏ và màu đỏ, hoặc nếu màu là xanh lá hoặc hình dạng là tròn. Trôi khái niệm có thể được tạo ra bằng cách thay đổi hàm phân lớp theo thời gian, giúp kiểm tra khả năng thích nghi của các thuật toán học máy trực tuyến. Ngoài ra, bộ dữ liệu này còn hỗ trợ cân bằng lớp, đảm bảo phân phối nhãn đồng đều, giúp đánh giá chính xác hơn hiệu năng của mô hình.
- Bộ dữ liệu **Sine** là một bộ dữ liệu tổng hợp luồng bao gồm tối đa bốn đặc trưng số, trong đó hai đặc trưng có ý nghĩa đối với bài toán phân lớp, còn hai đặc trưng tùy chọn có thể được thêm vào như nhiễu. Bộ dữ liệu tổng hợp Sine hỗ trợ bốn hàm phân lớp: SINE1, Reversed SINE1, SINE2, và Reversed SINE2, trong đó các hàm SINE xác định nhãn dựa trên giá trị của hai đặc trưng quan trọng và một đường cong phân chia dữ liệu. Trôi khái niệm có thể được tạo ra bằng cách thay đổi hàm phân lớp theo thời gian, có thể thực hiện thủ công hoặc thông qua *ConceptDriftStream*. Ngoài ra, bộ dữ liệu này còn hỗ trợ cân bằng lớp, giúp phân phối nhãn gần như đồng đều, cũng như khả năng thêm nhiễu, làm tăng độ phức tạp của bài toán phân lớp.

Việc mô phỏng hiện tượng trôi khái niệm khi tạo sinh các bộ dữ liệu trên đây được thực hiện qua cơ chế thay đổi xác suất sinh mẫu giữa hai khái niệm trong một khoảng thời gian xác định, gọi là cửa sổ trôi. Cụ thể, tại mỗi thời điểm trong cửa sổ, xác suất sinh một mẫu từ khái niệm A hiện tại hoặc khái niệm mới B được xác định qua một hàm sigmoid. Khi bắt đầu cửa sổ, phần lớn các mẫu được sinh từ A; khi đến cuối cửa sổ, phần lớn các mẫu được sinh từ B; sau khi cửa sổ kết thúc, toàn bộ dữ liệu sẽ được sinh từ khái niệm B, nghĩa là khái niệm mới đã trở nên ổn định.

Việc tạo sinh các bộ dữ liệu thực nghiệm của luận án được mô tả chi tiết trong Chương 3.

1.6 Đánh giá hiệu năng phát hiện trôi khái niệm

Như đã được đề cập, hầu hết nghiên cứu trên thế giới về xử lý trôi khái niệm giám sát tập trung vào các phương pháp giám sát, vì vậy, đánh giá hiệu năng phát hiện trôi khái niệm thường theo khung đánh giá hiệu năng học máy giám sát, tuy nhiên, dữ liệu luồng dã tạo ra nhiều thách thức không nhỏ [19, 49, 36]. Hai vấn đề chính trong đánh giá hiệu năng học máy giám sát (nói chung) và phát hiện trôi khái niệm (nói riêng) là về các độ đo được dùng và phương thức tính toán các giá trị liên quan tới các độ đo này.

1.6.1 Các độ đo hiệu năng phổ biến

Các độ đo hiệu năng trong phát hiện trôi khái niệm được giới thiệu như sau đây [4].

Bảng 1.1 Ma trận nhầm lẫn

Kết quả phân lớp / Thực tế	**Được phân vào lớp dương (P)**	**Được phân vào lớp âm (N)**
Thực tế là ví dụ dương (P)	TP (True Positive)	FN (False Negative)
Thực tế là ví dụ âm (N)	FP (False Positive)	TN (True Negative)

Trong phân lớp nhị phân, thường một lớp được quan tâm hơn (ví dụ, lớp "*trôi*") so với lớp còn lại và được gọi là lớp "dương", lớp còn lại (ví dụ, lớp "*không trôi*") được gọi là lớp "âm". Tiến hành áp dụng bộ phân lớp trên tập dữ liệu kiểm tra, nhận được các đại lượng TP (True Positive), TN (True Negative), FP (False Positive), FN (False Negative) như trình bày trong Bảng 1.1 Ma trận nhầm lẫn (*confusion matrix*).

Các đại lượng TP (dương tính đúng), FN (âm tính sai), FP (dương tính sai), TN (âm tính đúng) trong bảng được diễn giải như sau: TP là số lượng dữ liệu đánh giá có nhãn dương được dự đoán đúng vào lớp dương, FN là số lượng dữ liệu đánh giá có nhãn dương bị dự đoán sai vào lớp âm, FP là số lượng dữ liệu đánh giá có nhãn âm bị dự đoán sai vào lớp dương, TN là số lượng ví dụ đánh giá có nhãn âm được dự đoán đúng vào lớp âm.

Từ ma trận nhầm lẫn, các độ đo chính xác P (*Precision*), hồi tưởng R (*Recall* còn được gọi là tỷ lệ dương thật TPR: *True Positive Rate*), tỷ lệ dương giả FPR (*False Positive Rate*), tỷ lệ âm giả FNR (*False Negative Rate*) và tỷ lệ phân lớp CR (*Classification rate*) hay độ đo Accuracy được tính toán như sau:

$$P = \frac{TP}{TP+FP}, R \equiv TPR = \frac{TP}{TP+FN}, FPR = \frac{FP}{FP+TN}, FNR = \frac{FN}{TP+FN}, \quad (1.4)$$

$$CR \equiv Accurary = \frac{TP+TN}{TP+TN+FP+FN}. \quad (1.5)$$

Độ chính xác và độ hồi tưởng không có quan hệ trực tiếp với nhau, điều này vừa có điểm tích cực là cung cấp một khung nhìn đo lường hai chiều đánh giá mô hình phân lớp lại vừa có điểm hạn chế là tạo khó khăn khi xem xét, so sánh độ hiệu quả của các mô hình phân lớp khác nhau. Để việc so sánh đánh giá các bộ phân lớp khác nhau được thuận tiện, luận án sử dụng độ đo F_β **(F_β-score)** như sau:

$$F_\beta = \frac{(\beta^2+1)*P*R}{\beta^2*P+R} \quad (1.6)$$

Trường hợp đặc biệt khi chọn giá trị β=1, độ đo điểm F1 được gọi là trung bình điều hòa (*harmonic mean*) của độ chính xác và độ hồi tưởng. Để tường minh thêm ý nghĩa "trung bình điều hòa", độ đo F1 được trình bày dưới dạng sau đây:

$$F1 = \frac{2}{\frac{1}{P}+\frac{1}{R}} = \frac{2\times P\times R}{P+R} = \frac{2\times TP}{2\times TP+FP+FN} \quad (1.7)$$

Công thức (1.7) cho thấy trung bình điều hòa F1 của hai số sẽ tiến gần đến số nhỏ hơn, do đó, giá trị độ đo F1 cao chỉ trong trường hợp cả P và R đều cao. Như vậy, F1 là rất "nhạy" đối với sự thay đổi của P hoặc R vì chỉ cần P hay R có một thay đổi nhỏ cũng dẫn tới sự thay đổi tương ứng của F1.

1.6.2 Cách thức tính toán giá trị các độ đo hiệu năng

Như đã đề cập ở trên, các độ đo đánh giá hiệu năng như độ chính xác P và độ hồi tưởng R được tính toán giá trị dựa trên việc đo đếm các đại lượng TP, FN, FP, TN. Trong học giám sát thông thường, bốn đại lượng này được đo đếm trên từng thể hiện dữ liệu: nhãn lớp thực tế và nhãn lớp qua mô hình dự đoán. Trong học giám sát đối với phát hiện trôi khái niệm từ dữ liệu luồng, việc đo đếm các đại lượng này gặp thách thức lớn [19, Weeb16, 28, 36]. Vì vậy, theo thời gian, các quan sát thuộc các tập dữ liệu luồng được gom thành các lô (batch), kích thước lô có thể khác nhau theo từng tập dữ liệu. Ví dụ, J. Haug và G. Kasneci [24] chọn lô kích thước 10 cho Spambase và HAR, lô kích thước 50 cho Adult, lô kích thước

100 cho mọi tập dữ liệu khác. Việc đo đếm các đại lượng TP, FN, FP, TN để tính toán độ chính xác P và độ hồi tưởng R có thể có khác biệt. Dưới đây là cách thức được S. Yu và Z. Abraham [Yu17], J. Haug và G. Kasneci [24] sử dụng để tính toán độ đo chính xác P và độ hồi tưởng R:

- Chỉ xem xét các điểm trôi được mô hình phát hiện sau 80 lô dữ liệu đầu tiên để loại bỏ tác động không ổn định khi học 80 lô dữ liệu đầu.
- Với độ chính xác ***P***: Ghi nhận một dương tính đúng $TP1$ khi một điểm trôi khái niệm được phát hiện trong 50 lô dữ liệu đầu tiên kế từ điểm trôi thực sự xảy ra và ghi một dương tính sai *FP1* nếu điểm trôi khái niệm được phát hiện sau **50** lô dữ liệu kể trên. Khi đó, $P = \frac{|TP1|}{|TP1|+|FP1|}$.
- Với độ hồi tưởng ***R***: Ghi nhận một âm tính sai $FN2$ nếu không có một điểm trôi khái niệm nào được phát hiện trong 50 lô dữ liệu đầu tiên kể từ khi điểm trôi thực sự xảy ra và ghi một dương tính đúng $TP2$ nếu một điểm trôi được phát hiện trong khoảng thời gian 50 lô, Khi đó, $R = \frac{|TP2|}{|TP2|+|FN2|}$. Lưu ý là $TP2$ có thể khác với $TP1$ vì có thể ghi nhận thêm các phát hiện trôi khái niệm nằm sau điểm trôi khái niệm thực sự.

Luận án sử dụng cách thức tính toán giá trị các độ đo chính xác P và hồi tưởng R theo cách thức được sử dụng trong các công trình liên quan [24], [55].

1.7 Xu hướng nghiên cứu xử lý trôi khái niệm và một số luận án Tiến sĩ trên thế giới

1.7.1 Xu hướng nghiên cứu xử lý trôi khái niệm

Như được đề cập, xu hướng nghiên cứu điển hình để giải quyết các thách thức đối với xử lý trôi khái niệm bao gồm [28], [2], [36], [Cetrulo23], [52], [29]:

- *Cung cấp thông tin về mức nghiêm trọng, các vùng trôi cùng với điểm trôi*. Không có phân tích toàn diện nào về luồng dữ liệu thực tế từ khía cạnh trôi khái niệm, chẳng hạn như thời gian xảy ra trôi, mức nghiêm trọng của trôi và các vùng trôi, vì vậy, nghiên cứu phát hiện trôi không chỉ tập trung vào việc xác định chính xác thời gian xảy ra trôi mà còn cần cung cấp thông tin về mức nghiêm trọng và các vùng trôi; những thông tin này có thể được sử dụng để điều chỉnh trôi khái niệm tốt hơn [28].

- *Học máy không giám sát và bán giám sát để phát hiện nhanh trôi, hiểu và điều chỉnh trôi khái niệm, đặc biệt trong xử lý trực tuyến*. Rất ít nghiên cứu được tiến hành để giải quyết vấn đề phát hiện và thích ứng trôi không giám sát hoặc bán giám sát; trong kịch bản thực tế, chi phí để có được nhãn thực sự có thể rất tốn kém, nghĩa là phát hiện và điều chỉnh trôi không giám sát hoặc bán giám sát vẫn có thể hứa hẹn trong tương lai [28], đồng thời, cần có các luồng dữ liệu tổng hợp phức tạp hơn để kiểm tra đáng tin cậy các bộ phát hiện trôi khái niệm không giám sát [29].
- *Độ đo, khung và phương pháp đánh giá hiệu năng thuật toán học xử lý trôi khái niệm*. Hệ thống đánh giá là một phần quan trọng đối với các thuật toán học [28]; đánh giá các bộ phát hiện trôi khái niệm là một xu hướng nghiên cứu điển hình [Cetrulo23]; không có phương pháp đánh giá đơn giản nào có sẵn để đánh giá chất lượng phát hiện trôi khái niệm khi có thông tin thực tế [29].
- *Tích hợp các kỹ thuật xử lý trôi khái niệm*. Một trong những phương pháp tiếp cận phổ biến và hiệu quả nhất để xử lý trôi khái niệm là Học kết hợp [43]; nghiên cứu về việc tích hợp hiệu quả các kỹ thuật xử lý trôi khái niệm với các phương pháp học máy cho các ứng dụng dựa trên dữ liệu là rất được mong muốn [28]; vấn đề tích hợp đa mô hình với các phương pháp tích hợp trực tuyến phổ biến hơn trong các phương pháp thích ứng trôi khái niệm [52].
- *Dữ liệu nhiều chiều, đa dạng đặc trưng (số, phân lớp, đa phân lớp, thời gian, nhị phân và độ lệch) và dung lượng lớn*. Bộ phát hiện trôi phải xử lý các luồng dữ liệu có các đặc trưng như số, phân lớp, đa phân lớp, thời gian, nhị phân và độ lệch, đồng thời, phân tích các đặc trưng có tác động lớn hơn đến sự thay đổi khái niệm là một nhiệm vụ đầy thách thức [2].
- *Phát hiện trôi khái niệm không đột ngột và kết hợp phát hiện trôi khái niệm đa kiểu (đột ngột, gia tăng, dần)*. Phát hiện trôi khái niệm không đột ngột cần được nghiên cứu kỹ lưỡng hơn; nhiều bộ phát hiện trôi sẽ xóa các cửa sổ dữ liệu của chúng sau khi phát hiện ra trôi khái niệm, cho nên, vẫn chưa rõ liệu hành vi này có phù hợp với trôi khái niệm gia tăng và dần dần hay không, đồng thời, một thách thức nghiên cứu mới là làm thế nào để kết hợp phát hiện trôi khái niệm đột ngột, gia tăng và dần dần trong một hệ thống duy nhất bao gồm khả năng phân biệt các loại trôi khác nhau [29].

- *Trôi khái niệm trong khai phá quy trình*. Trong khai phá quy trình, tồn tại mối quan hệ giữa dữ liệu sự kiện và mô hình quy trình, tuy nhiên, việc xử lý các mô hình quy trình chứa các cấu trúc phức tạp (như đồng thời, lựa chọn và vòng lặp) dẫn đến việc giải quyết trôi khái niệm trong khai phá quy trình là một thách thức [36], [37]; hơn nữa, đưa ra lời giải thích dạng nhân-quả cho trôi khái niệm là hết sức cần thiết [61].

Ngoài ra, xử lý trôi khái niệm theo nghĩa "chuyển dịch phân bố" (*distribution shift*) cũng nhận được sự quan tâm trong thời gian gần đây [17].

1.7.2 Một số luận án Tiến sĩ liên quan.

Luận án của Gu Feng [16] đề xuất một loạt phương pháp nhằm cải thiện khả năng phát hiện trôi khái niệm trong môi trường dữ liệu luồng. Trước hết, tác giả giới thiệu Neighbor Search Discrepancy (NSD), một chỉ số mới đo lường sự khác biệt của ranh giới phân lớp giữa hai mẫu dữ liệu, giúp phát hiện chính xác trôi khái niệm thực trong khi bỏ qua trôi ảo. Dựa trên NSD, hai thuật toán chọn mẫu mới là Decision Region Support Set (DRS) và Decision Region Border Set (DRB) được đề xuất, cho phép giảm số lượng mẫu phục vụ cả mục tiêu phát hiện trôi và phân lớp mà không làm tăng chi phí tính toán. Đối với bài toán phát hiện trôi khi không có nhãn, tác giả phát triển phương pháp Equal Density Estimation (EDE) – một giải pháp phi tham số, không cần giả định trước về phân phối dữ liệu, nhằm khắc phục hạn chế của các kỹ thuật phân vùng không gian cố định. Cuối cùng, luận án xây dựng khái niệm mới về *trôi duy trì được* và *trôi không duy trì được*, từ đó đề xuất phương pháp để chỉ phát hiện các loại trôi mà việc cập nhật mô hình thực sự cần thiết, ngay cả khi không có dữ liệu nhãn.

W. Chiu [12] tập trung nghiên cứu bài toán phân lớp trên luồng dữ liệu trong điều kiện xuất hiện đồng thời trôi khái niệm và mất cân bằng lớp, hai thách thức phổ biến nhưng ít được xử lý đồng thời trong các nghiên cứu trước đây. Các đóng góp chính của luận án gồm ba hướng lớn. Thứ nhất, tác giả tiến hành nghiên cứu sâu về vai trò của chiến lược quản lý bộ nhớ đối với khả năng thích ứng với trôi khái niệm. Trên cơ sở đó, luận án đề xuất một chiến lược quản lý bộ nhớ dựa trên độ đa dạng nhằm tối đa hóa xác suất bảo tồn các mô hình có khả năng tái sử dụng khi xuất hiện trôi khái niệm, đặc biệt hữu ích đối với các trường hợp trôi lặp lại. Thứ hai, tác giả phát triển khung phương pháp mới có tên CDCMS (Concept Drift Handling based on Clustering in the Model Space), tận dụng cụm các mô hình

trong không gian đầu ra để phát hiện và thích nghi với nhiều loại trôi khái niệm khác nhau trong luồng dữ liệu cân bằng lớp. Tiếp tục, tác giả mở rộng CDCMS để xử lý hiệu quả các luồng dữ liệu mất cân bằng lớp, bằng cách kết hợp chiến lược quản lý bộ nhớ theo độ đa dạng và kỹ thuật cân bằng lớp. Thứ ba, nhằm giải quyết vấn đề khan hiếm mẫu thiểu số trong môi trường luồng dữ liệu, luận án đề xuất một phương pháp dựa trên việc sinh mẫu thiểu số tổng hợp từ các cụm vi mô được duy trì trực tuyến, cho phép vừa cân bằng lớp, vừa thích ứng linh hoạt với các dạng trôi khái niệm mà không cần lưu trữ toàn bộ dữ liệu trước đó.

Luận án của H. Tan [41] trình bày một khung thống nhất để phát hiện đồng thời dị thường và trôi khái niệm trong luồng dữ liệu động, đồng thời cung cấp khả năng giải thích quyết định của mô hình. Các thách thức được giải quyết bao gồm: (i) xử lý khối lượng dữ liệu không giới hạn với tốc độ cao, (ii) thích ứng với sự thay đổi động của dữ liệu, (iii) tối ưu hóa việc sử dụng nhãn dữ liệu vốn khan hiếm, và (iv) nâng cao tính minh bạch trong quá trình ra quyết định của mô hình. Đầu tiên, tác giả đề xuất phương pháp phát hiện dị thường hoàn toàn không giám sát, dựa trên bộ ước lượng Mahalanobis với khả năng học gia tăng. Bộ ước lượng này không chỉ phát hiện dị thường hiệu quả mà còn khai thác các cập nhật gia tăng để đồng thời phát hiện trôi khái niệm trong luồng dữ liệu. Sau đó, luận án phát triển một khung phân tích tài nguyên thông tin, cho phép ước lượng số lượng nhãn và mẫu cần thiết để đạt được độ chính xác nhất định trong việc phát hiện trôi khái niệm, từ đó tối ưu hóa chi phí gán nhãn dữ liệu. Để nâng cao tính minh bạch của mô hình, luận án mở rộng phương pháp trên với khả năng giải thích: (i) phân tách điểm dị thường theo đóng góp của từng đặc trưng, và (ii) mô tả hướng và độ lớn của trôi khái niệm thông qua thay đổi tương quan và phân phối dữ liệu. Các phương pháp được đánh giá kỹ lưỡng trên cả tập dữ liệu tổng hợp và thực tế công nghiệp (khai thác than, điện lực), cho thấy hiệu quả cao trong việc phát hiện và giải thích các sự kiện bất thường trong môi trường không giám sát.

Trong luận án [32], H. Mehmood tập trung nghiên cứu bài toán phát hiện và thích nghi với trôi khái niệm trong các bài toán liên quan đến thành phố thông minh, nơi dữ liệu từ các thiết bị IoT và cảm biến rất lớn, liên tục thay đổi và không đồng nhất. Tác giả chỉ ra rằng sự thay đổi ngữ cảnh và các tính chất thống kê của dữ liệu theo thời gian có thể gây suy giảm hiệu quả của các hệ thống dự đoán truyền thống. Để giải quyết vấn đề này, luận án đề xuất hai thuật toán mới tích

hợp phát hiện trôi khái niệm vào quá trình học máy dự đoán, đồng thời phát triển hai kiến trúc xử lý dữ liệu phân tán: một dựa trên nền tảng điện toán đám mây và một dựa trên tính toán biên cho các trung tâm dữ liệu vi mô.

Luận án của Pingfan Wang [46] tập trung vào vấn đề phát hiện trôi khái niệm trong học máy trên dữ liệu luồng, đặt vấn đề rằng các phương pháp phát hiện trôi khái niệm chủ yếu chỉ xác định thời điểm trôi xảy ra mà không cung cấp thông tin chi tiết về đặc điểm của nó, chẳng hạn như thời điểm bắt đầu, mức độ nghiêm trọng hay thời điểm kết thúc. Ngoài ra, mặc dù các mô hình học sâu có tiềm năng lớn trong việc xử lý dữ liệu phức tạp, chúng chưa được khai thác nhiều trong lĩnh vực này do chi phí tính toán cao. Để giải quyết những hạn chế này, luận án đề xuất một loạt phương pháp mới trong đó có phương pháp QuadCDD, một phương pháp tiếp cận bốn thành phần giúp cung cấp cái nhìn toàn diện hơn về trôi khái niệm. Thay vì chỉ xác định thời điểm bắt đầu trôi, QuadCDD thu thập thông tin về toàn bộ vòng đời của trôi khái niệm, bao gồm điểm bắt đầu, điểm kết thúc, mức độ nghiêm trọng và kiểu trôi, từ đó hỗ trợ mô hình thích nghi hiệu quả hơn với sự thay đổi của dữ liệu.

1.8 Liên hệ với nghiên cứu trong luận án

Trước hết, việc luận án đề xuất các mô hình phát hiện trôi khái niệm như E-ERICS và ERICS+3 thể hiện rõ định hướng tiếp cận hiện đại thông qua học tổng hợp, kết hợp với tối ưu hóa siêu tham số để nâng cao hiệu quả phát hiện. Cách tiếp cận hai pha, gồm pha tối ưu và pha nhận diện, đồng thời khai thác cả tri thức từ các mô hình hiện có lẫn khả năng tổng quát hóa qua quá trình chọn lọc tham số tối ưu, phù hợp với xu hướng áp dụng tri thức liên miền và tăng cường khả năng thích nghi của mô hình theo thời gian. Việc sử dụng chiến lược bỏ phiếu đa mô hình trong E-ERICS và ERICS+3 cũng cho thấy định hướng biểu diễn tri thức đa hình thức, giảm thiểu độ nhiễu trong việc phát hiện trôi.

Thứ hai, luận án đóng góp vào hướng phân lớp kiểu trôi khái niệm bằng cách xây dựng các khung mô hình phân lớp trôi như VAR-WIND, MetaLDD-Finetune, trong đó áp dụng phương pháp học từ ít mẫu (few-shot learning) trên nền tảng mạng nguyên mẫu (prototypical network). Dựa trên những ưu điểm của phương pháp học cách học và cảm hứng từ các nghiên cứu dựa trên mạng nguyên mẫu để phân lớp kiểu trôi, luận án đi sâu tìm hiểu về khung Meta-ADD và đưa ra các đề xuất trong Chương 3 và Chương 4.

Trong quá trình nghiên cứu bài toán phân lớp trôi khái niệm trong dữ liệu dòng, các phương pháp dựa trên phân phối dữ liệu như CDT_MSW hay QuadCDD đã cho thấy hiệu quả nhất định trong việc phát hiện và phân biệt các loại trôi dựa trên sự thay đổi thống kê giữa các cửa sổ dữ liệu. Tuy nhiên, qua thực nghiệm và phân tích lý thuyết, các phương pháp này vẫn tồn tại những hạn chế đáng kể, đặc biệt là khả năng dễ bị ảnh hưởng bởi nhiễu, phụ thuộc mạnh vào tham số cửa sổ, khó xử lý dữ liệu nhiều chiều, và thiếu khả năng học hỏi từ lịch sử để cải thiện hiệu năng theo thời gian. Trước những thách thức này, luận án lựa chọn tiếp cận theo hướng *học cách học* như một giải pháp tiềm năng nhằm khắc phục các nhược điểm trên. Phương pháp học cách học không chỉ cho phép mô hình học được từ nhiều tín hiệu khác nhau của quá trình trôi trong quá khứ mà còn giúp tích hợp linh hoạt các đầu ra từ nhiều thuật toán phát hiện trôi để ra quyết định phân loại chính xác hơn. Việc học các mẫu thay đổi đặc trưng của từng loại trôi qua thời gian giúp mô hình có khả năng phân biệt tốt hơn giữa trôi thực và nhiễu, đồng thời nâng cao hiệu quả trong các môi trường dữ liệu động, phức tạp và thay đổi liên tục.

VAR-WIND triển khai các chiến lược cửa sổ theo từng kiểu trôi, giúp mô hình nắm bắt đặc điểm riêng biệt và biểu diễn khái niệm theo cách thích nghi hơn. Điều này có liên hệ với xu hướng biểu diễn tri thức phân biệt theo loại hình thay đổi, cũng như khả năng suy luận nhanh từ ít quan sát. MetaLDD-Finetune là phiên bản cải tiến từ Meta-ADD bổ sung một bước tinh chỉnh sau tiền huấn luyện Meta-ADD để tăng cường khả năng thích nghi với luồng dữ liệu mới từ đó nâng cao độ chính xác của dự đoán.

Thứ ba, luận án đề xuất mô hình phân lớp trôi khái niệm CS&AM_SC dựa trên phân tích và tối ưu không gian biểu diễn đặc trưng trong mạng nguyên mẫu, cho thấy luận án không chỉ khai thác các phương pháp hiện có, mà còn mở rộng chúng theo hướng nâng cao khả năng phân biệt của mô hình – phù hợp với định hướng hiện nay về học máy ít giám sát, mạnh về biểu diễn.

1.9 Kết luận Chương 1

Chương 1 đã giới thiệu một hệ thống nội dung cơ bản nhằm cung cấp một khung nhìn tổng quát về trôi khái niệm và phát hiện trôi khái niệm. Đồng thời, các xu hướng nghiên cứu về xử lý trôi khái niệm đã được giới thiệu. Hơn nữa, mối

liên hệ của các nghiên cứu trong luận án với các xu hướng nghiên cứu cũng đã được chỉ ra.

Chương 2 trình bày đề xuất của luận án về mô hình học kết hợp phân phối tham số để phát hiện trôi khái niệm.

Chương 2. Mô hình học kết hợp phân phối tham số phát hiện trôi khái niệm

Chương này trình bày hai mô hình phát hiện trôi khái niệm là E-ERICS phát hiện trôi đột ngột và trôi gia tăng và ERICS+3 phát hiện trôi dần dần. Cả hai mô hình đều được xây dựng theo hướng tiếp cận *học kết hợp* như đã đề cập ở Mục 1.2.4 và kỹ thuật tối ưu hóa siêu tham số nhằm nâng cao độ chính xác và tính thích ứng của hệ thống phát hiện trôi.

Đầu tiên, luận án trình bày mô hình ERICS gốc của J. Haug và G. Kasneci [23] với phương pháp phát hiện trôi dựa trên việc theo dõi sự thay đổi trong phân phối tham số của mô hình, đại diện cho một hướng tiếp cận học sự thay đổi phân phối tham số. Từ phân tích những hạn chế của ERICS và những những ưu điểm của phương pháp học kết hợp, luận án đề xuất hai mô hình phát hiện trôi khái niệm E-ERICS và ERICS+3 nhằm khắc phục những điểm còn hạn chế của ERICS đồng thời nâng cao hiệu quả phát hiện các loại trôi khái niệm khác nhau.

2.1 Mô hình phát hiện trôi khái niệm ERICS

Bài toán phát hiện trôi khái niệm được phát biểu một cách hình thức như sau:

- **Đầu vào**: Một luồng dữ liệu D_{tream} có kích thước n, một tập dữ liệu huấn luyện $D^*_{tream} = \{(d_t, l_t\}, t \in [1, n],\ d_t \in D_{tream},\ l_t \in [0,1]$ cho biết thời điểm t thuộc một vùng trôi khái niệm ($l_t = 1$) hay không ($l_t = 0$).
- **Đầu ra**: Một mô hình phát hiện trôi khái niệm M cho luồng dữ liệu D_{tream} sau thời điểm n: áp dụng mô hình M vào thời điểm $t = n + k, k > 0$ cho dự báo trôi khái niệm có xảy ra hay không.

Mô hình phát hiện trôi ERICS [23] thuộc nhóm các phương pháp phát hiện trôi khái niệm dựa trên phân phối dữ liệu (Mục 1.2.2). Điểm mới của ERICS là không theo dõi trực tiếp dữ liệu, mà giám sát sự thay đổi của phân phối các tham số của mô hình học dựa trên việc sử dụng lý thuyết thông tin để suy ra trôi khái niệm. Cụ thể, mô hình này coi các tham số của mô hình dự đoán như những biến ngẫu nhiên và chỉ ra rằng trôi khái niệm có thể được nhận biết thông qua sự thay đổi trong phân phối của các tham số tối ưu. Gọi tham số ϕ như là một biến ngẫu nhiên, có nghĩa là $\phi \sim P(\phi; \psi)$ và tối ưu các tham số phân phối ψ ở mỗi thời điểm

theo hàm khả năng (log-likelihood), vấn đề tối ưu có thể được giải quyết dưới dạng khả năng cận biên $P(Y|X,\psi)$. Kết hợp trôi khái niệm giữa hai thời điểm t và u với sự khác nhau của khả năng cận biên về các tham số phân phối ψ_t và ψ_u:

$$P(Y|X;\psi_t) \neq P(Y|X;\psi_u) \quad (2.1)$$

$$\Leftrightarrow |P(Y|X;\psi_t) - P(Y|X;\psi_u)| > 0 \quad (2.2)$$

$$\Leftrightarrow |\int P(Y|X,\phi)[P(\phi;\psi_t) - P(\phi;\psi_u)]d\phi| > 0 \quad (2.3)$$

Biến đổi lại (2.3) dưới dạng entropy vi phân (sự không chắc chắn) và phân kỳ Kullback–Leibler (KL) của phân phối tham số tại thời điểm t, thu được:

$$\underbrace{|h[P(\phi;\psi_u)] - h[P(\phi;\psi_t)]}_{\Delta\text{Sự không chắc chắn}} + \underbrace{D_{KL}[P(\phi;\psi_u) \mid P(\phi;\psi_t)]|}_{\Delta\text{Phân phối}} > 0 \quad (2.4)$$

Trong (2.4), trôi khái niệm thực tế tương ứng với sự thay đổi của "sự không chắc chắn của các tham số tối ưu" và "sự phân kỳ của phân phối tham số".

Trong thực tế, chúng ta chủ yếu quan tâm đến trôi khái niệm giữa các thời điểm liên tiếp $t - 1$ và t. Tuy nhiên, nếu áp dụng (2.4) chỉ cho hai thời điểm, mô hình phát hiện trôi khái niệm có thể trở nên quá "nhạy cảm" (có nghĩa là sự biến động nhỏ của tham số xác suất cũng dẫn tới sự biến động lớn của giá trị biến ngẫu nhiên) với các biến ngẫu nhiên của mô hình dự đoán. Do đó, để mô hình đỡ nhạy cảm với nhiễu, ERICS sử dụng trung bình chuyển động (Moving Average – MA) của (2.4), bằng cách tính tính trung bình chuyển động tại thời điểm t trên cửa sổ có kích thước M:

$$MA_t = \frac{1}{M}\sum_{i=t-M+1}^{t}(|h[P(\phi;\psi_i)] - h[P(\phi;\psi_{i-1})] + D_{KL}[P(\phi;\psi_i) \mid P(\phi;\psi_{i-1})]|) \quad (2.5)$$

Có thể điều chỉnh độ nhạy của ERICS bằng cách lựa chọn M phù hợp. Nhìn chung (M càng lớn, ERICS càng trở nên mạnh mẽ. Tuy nhiên, M lớn cũng có thể làm ẩn đi trôi khái niệm có cường độ nhỏ hoặc khoảng thời gian ngắn. Tiếp tục tính tổng cộng công thức (2.5) trong một cửa sổ phát hiện trôi có kích thước W:

$$\sum_{j=t-W+1}^{t}\left(MA_j - MA_{j-1}\right) > \alpha_t \Leftrightarrow \text{Trôi ở thời điểm } t \quad (2.6)$$

với $\alpha \geq 0$ là một ngưỡng thích ứng.

Có thể điều khiển độ nhạy của phát hiện trôi khái niệm bằng cách thay đổi kích thước cửa sổ W. Khi phát hiện trôi khái niệm, ngưỡng α_t được tính bởi:

$$\alpha_t = \sum_{j=t-W+1}^{t}\left(MA_j - MA_{j-1}\right) \quad (2.7)$$

Với cách này, ERICS tạm thời chấp nhận mọi thay đổi đối với mô hình dự đoán lên tới một độ lớn nào đó. Sau đó cập nhật α_t sau mỗi lần lặp. Đặt β là một siêu tham số thể hiện tốc độ cập nhật ngưỡng α do người dùng định nghĩa trong khoảng [0, 1]. Mỗi cập nhật phụ thuộc vào giá trị α hiện tại, siêu tham số β và thời gian đã trôi qua kể từ cảnh báo trôi khái niệm cuối cùng, ký hiệu là Δ_{Drift}:

$$\alpha_t = \alpha_{t-1} - \left(\alpha_{t-1} * \beta * \Delta_{Drift}\right) \quad (2.8)$$

Chú ý rằng α_t sẽ tiệm cận đến 0 theo thời gian, nếu không có trôi khái niệm. Theo cách này, mức chấp nhận của ERICS sau khi một trôi khái niệm được phát hiện sẽ giảm dần.

ERICS nhấn mạnh rằng không phụ thuộc vào mô hình cơ sở cụ thể nhưng trong các thực nghiệm, hồi quy Probit được sử dụng làm mô hình cơ sở cho việc theo dõi sự thay đổi của các tham số. Đây là một dạng hồi quy nhị phân, trong đó xác suất một quan sát thuộc về lớp nhãn dương được mô hình hóa thông qua hàm phân phối tích lũy (CDF) của phân phối chuẩn. Trong hồi quy Probit thông thường, hàm liên kết xác suất có dạng:

$$P(y = 1 \mid \boldsymbol{x}) = \Phi(\boldsymbol{x}^T \boldsymbol{w}) \quad (2.9)$$

Trong đó:

$\Phi(\cdot)$ là hàm phân phối tích lũy của phân phối chuẩn.

$\boldsymbol{w}$ là vector trọng số của các đặc trưng.

Ở đây, trọng số mô hình $\boldsymbol{w}$ thay đổi theo thời gian, được theo dõi để xác định trôi khái niệm.

2.2 Nhận xét và ý tưởng cải tiến mô hình ERICS

So với các phương pháp phát hiện trôi dựa trên phân phối dữ liệu, ERICS có ưu điểm là thay vì trực tiếp xem xét phân phối của luồng dữ liệu, ERICS theo dõi phân phối tham số của mô hình, phản ánh trực tiếp sự thay đổi trong quá trình học. Bằng việc quan sát thay đổi trong không gian tham số, mô hình ERICS có thể phát hiện những thay đổi nhỏ nhưng quan trọng, đặc biệt là với những trôi phức tạp, khó phát hiện bằng các phương pháp truyền thống. Do không phụ thuộc hoàn toàn vào hiệu năng dự đoán, ERICS có khả năng phát hiện trôi sớm hơn trong một số trường hợp. Có thể áp dụng ERICS với nhiều mô hình học máy và thuật toán tối ưu khác nhau.

Mặc dù mô hình ERICS cho thấy khả năng phát hiện trôi khái niệm hiệu quả trên nhiều loại dữ liệu, tuy nhiên một hạn chế đáng chú ý là mô hình này thường phát sinh nhiều báo động sai, dẫn đến độ chính xác không cao. Nguyên nhân chủ yếu xuất phát từ việc hiệu năng của ERICS phụ thuộc nhiều vào lựa chọn bộ siêu tham số (M, W, β). Nếu các tham số này không được chọn tối ưu, mô hình dễ rơi vào tình trạng nhầm lẫn giữa điểm trôi thực sự và nhiễu.

Từ thực tế đó, luận án đề xuất hai cải tiến chính nhằm khắc phục điểm yếu nêu trên:

- *Thứ nhất*, áp dụng phương pháp tối ưu hóa siêu tham số dựa trên kỹ thuật Tối ưu Bayes, nhằm tìm ra bộ tham số tối ưu giúp cải thiện hiệu năng phát hiện trôi của ERICS.
- *Thứ hai,* áp dụng ý tưởng từ phương pháp học kết hợp (đã đề cập tại Mục 1.2.4) để kết hợp các mô hình ERICS (đã tối ưu siêu tham số). Kết quả phát hiện điểm trôi từ từng mô hình đơn lẻ này sẽ được kết hợp bằng một cơ chế bỏ phiếu, từ đó đưa ra kết quả cuối cùng, xem điểm trôi đó có phải là một điểm trôi thực sự hay không. Phương pháp bỏ phiếu này nhằm tăng tính tin cậy và giảm thiểu ảnh hưởng của các quyết định sai lệch so với từ một mô hình đơn lẻ.

Hai cải tiến trên kết hợp tạo thành một *mô hình hai pha* phát hiện trôi mới, gồm pha tối ưu hóa siêu tham số và pha học kết hợp, nhằm duy trì ưu điểm của ERICS trong khả năng phát hiện trôi, đồng thời nâng cao đáng kể độ chính xác và tính ổn định trong các tình huống thực tế.

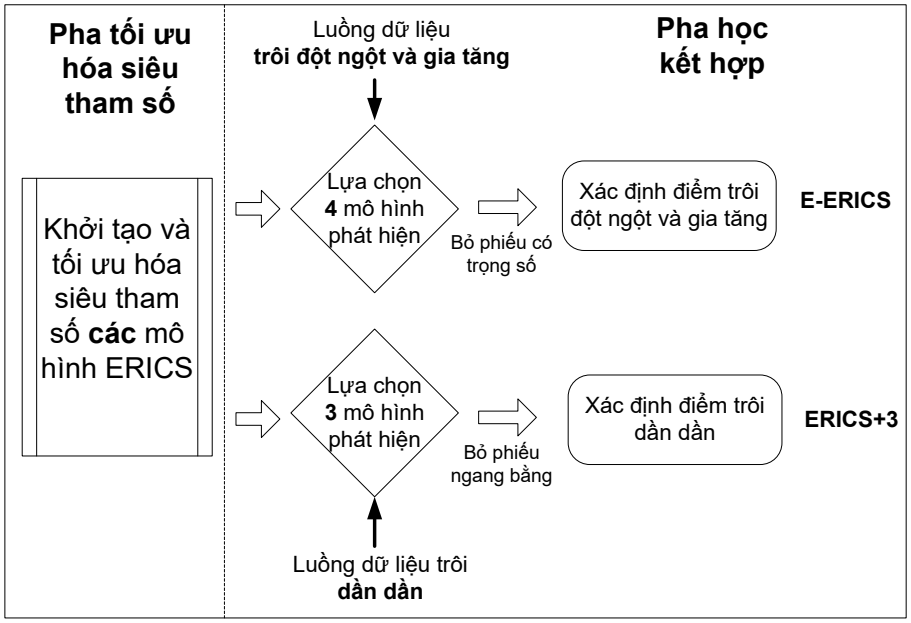
2.3 Hai mô hình đề xuất E-ERICS và ERICS+3

2.3.1 Mô hình cải tiến đề xuất

Mô hình E-ERICS và ERICS+3 đề xuất đều có hai pha nối tiếp nhau, được minh họa tại Hình 2.1 gồm:

- Pha tối ưu siêu tham số mô hình (gọi tắt là pha BO): áp dụng phương pháp tối ưu hóa Bayes để tìm kiếm các siêu tham số tối ưu cho mô hình ERICS. Quá trình này được thực hiện đồng nhất cho các kiểu trôi
- Pha học kết hợp: Đưa ra các phương án lựa chọn mô hình có hiệu năng tốt nhất với cơ chế bỏ phiếu khác nhau, được thiết kế riêng để phù hợp với đặc

điểm của từng kiểu trôi khái niệm, giúp nâng cao độ chính xác trong việc phát hiện trôi.



Hình 2.1 Hai mô hình E-ERICS và ERICS+3 được đề xuất

2.3.2 Pha tối ưu hóa siêu tham số

Tối ưu hóa siêu tham số đóng vai trò then chốt trong việc nâng cao hiệu năng của các mô hình học máy. Tuy nhiên, việc tìm kiếm bộ siêu tham số tối ưu thường gặp thách thức do không gian tìm kiếm lớn, các siêu tham số có mối quan hệ phi tuyến, và mỗi lần đánh giá hàm mục tiêu (ví dụ: huấn luyện và kiểm thử mô hình) đều rất tốn kém chi phí tính toán.

ERICS sử dụng nhiều siêu tham số, nhưng ba siêu tham số điển hình liên quan đến độ chính xác phát hiện trôi là kích thước cửa sổ tính trung bình trượt (M), kích thước cửa sổ phát hiện trôi (W) và tốc độ cập nhật ngưỡng (β). Do đó, cần tập trung vào việc tìm kiếm các giá trị tối ưu cho bộ ba tham số này.

Một số phương pháp tối ưu hóa phổ biến gồm: Tối ưu hóa ngẫu nhiên, tối ưu hóa lưới, tối ưu bầy đàn, và tối ưu hóa Bayes. Tuy nhiên, theo [58], mỗi phương pháp đều có những hạn chế:

- Tối ưu hóa ngẫu nhiên đơn giản, dễ thực hiện, nhưng thiếu định hướng và không tận dụng được thông tin từ các lần đánh giá trước, dẫn đến hiệu quả tìm kiếm thấp trong không gian tham số lớn.

- Tối ưu hóa lưới có hệ thống nhưng lại thiếu hiệu quả tính toán khi số lượng tham số tăng, vì phải đánh giá toàn bộ tổ hợp trong lưới.
- Tối ưu bầy đàn có khả năng tìm kiếm toàn cục nhưng nhược điểm là dễ mắc kẹt tại cực trị cục bộ và phụ thuộc nhiều vào điều chỉnh các siêu tham số của chính thuật toán.

Trong khi đó, xét trong bối cảnh của ERICS, tối ưu hóa Bayes (Bayesian Optimization - BO) khắc phục được hầu hết các nhược điểm trên, và phù hợp cho các bài toán tối ưu hộp đen mà mỗi phép đánh giá đều tốn kém [38] – như trường hợp tìm tham số tối ưu cho mô hình phát hiện trôi trong ERICS.

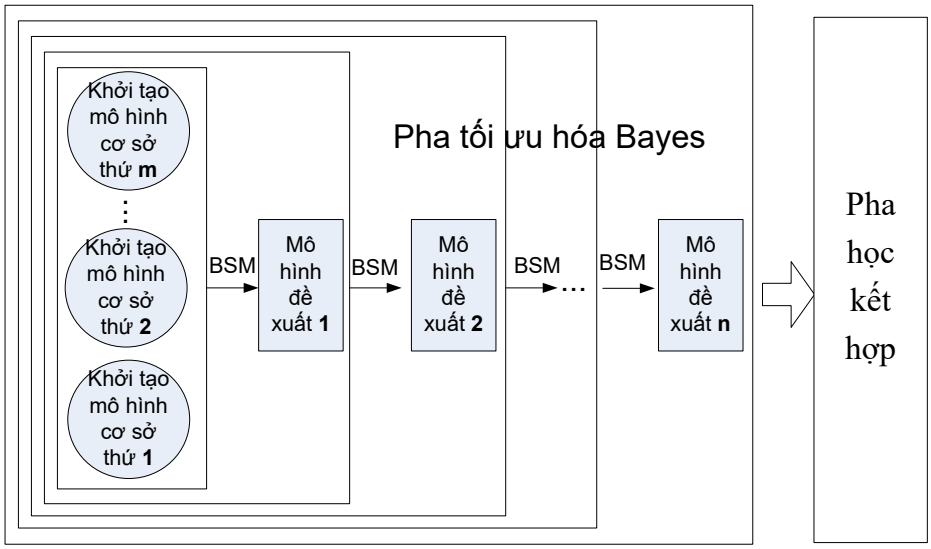
Cụ thể, BO sử dụng mô hình xác suất (quá trình Gaussian) để xấp xỉ hàm mục tiêu — là độ chính xác hoặc điểm F1 trên tập kiểm thử tương ứng với các siêu tham số *(M, W, β)*. Nhờ vậy, BO có thể:

- Tận dụng thông tin từ các lần đánh giá trước để chọn điểm đánh giá tiếp theo một cách thông minh, thông qua một hàm thu nhận, thay vì chọn ngẫu nhiên.
- Cân bằng giữa khám phá (đánh giá các vùng chưa biết) và khai thác (tập trung vào các vùng tiềm năng cao).
- Giảm số lần đánh giá cần thiết, giúp tiết kiệm thời gian và tài nguyên tính toán.

Đặc biệt, trong mô hình ERICS, không gian tìm kiếm cho ba siêu tham số *(M, W, β)* có thể rộng và phức tạp, nhất là khi hiệu năng mô hình có thể biến động theo các tổ hợp phi tuyến giữa chúng. Việc sử dụng BO giúp tập trung vào các tổ hợp có triển vọng cao, giảm thiểu đáng kể chi phí tối ưu hóa và nâng cao xác suất tìm được nghiệm tối ưu toàn cục.

Do đó, luận án chọn tối ưu hóa Bayes để phù hợp về mặt lý thuyết và bản chất của bài toán tối ưu siêu tham số trong ERICS. Cụ thể, quá trình tối ưu sử dụng điểm F1 làm hàm mục tiêu và quá trình Gaussian [34] để mô phỏng hàm hộp đen cần tối ưu, từ đó dẫn hướng tìm kiếm thông qua các hàm thu nhận.

Quy trình tối ưu hóa được thể hiện ở Hình 2.2. Đầu tiên khởi tạo $\boldsymbol{m}$ mô hình cơ sở phát hiện trôi khái niệm độc lập dựa trên mô hình ERICS. Mỗi mô hình được cấu hình bằng một bộ siêu tham số ngẫu nhiên *(M, W, β)* được chọn từ miền giá trị *D* đã xác định trước.



*Ghi chú: **BSM** là viết tắt của mô hình thay thế (Bayes Surrogate Model)

Hình 2.2. Pha tối ưu hóa Bayes

Tiếp đó thực hiện quy trình tối ưu hóa Bayes, các mô hình cơ sở được đánh giá bằng điểm F1 làm hàm thu thập. Dựa trên kết quả của các mô hình đã khởi tạo, tối ưu hóa Bayes đưa ra ***n* mô hình đề xuất** với các giá trị siêu tham số được tối ưu hóa hơn. Mũi tên có nhãn BSM chỉ ra rằng mỗi mô hình tiếp theo được tạo ra theo từng vòng lặp, cải thiện việc lựa chọn siêu tham số bằng cách sử dụng quá trình Gaussian làm mô hình thay thế. Công việc tối ưu hóa trên được lặp lại nhằm tinh chỉnh các siêu tham số để xác định các mô hình cơ sở có hiệu năng cao nhất.

Thủ tục tối ưu hóa:

*Bước 1. Khởi tạo **m** mô hình cơ sở*

Mỗi mô hình là một phiên bản của ERICS với một bộ siêu tham số *(M, W, β)* được chọn ngẫu nhiên từ không gian xác định trước *D*. Dữ liệu huấn luyện giống nhau, chỉ khác nhau về siêu tham số.

Bước 2. Tính toán điểm F1 cho từng mô hình cơ sở

Đánh giá hiệu năng từng mô hình bằng điểm F1. Đây là hàm mục tiêu để tối ưu.

Bước 3: Tối ưu hóa Bayes

- Từ các điểm ban đầu (*m* mô hình), BO sử dụng một mô hình thay thế (Bayes Surrogate Model - BSM) — cụ thể là Gaussian Process — để ước lượng điểm F1 trong toàn bộ không gian D.

- BSM giúp chọn ra **n** điểm mới (***n* mô hình đề xuất**) với bộ siêu tham số có khả năng cao sẽ cho kết quả tốt hơn.
- Các mô hình đề xuất này được huấn luyện và đánh giá tiếp tục để cải thiện dần.

Mỗi vòng lặp của quá trình này là: dự đoán điểm F1 của các điểm chưa thử → chọn điểm tốt nhất để thử → cập nhật mô hình BSM.

Kết thúc pha BO ta có một tập các mô hình có siêu tham số đã được tối ưu hóa, và biết mô hình nào có hiệu năng (điểm F1) tốt nhất. Sau đó chuyển sang pha học kết hợp.

Việc áp dụng Tối ưu Bayes để ước lượng các siêu tham số *(M, W, β)* trong ERICS mang lại hiệu quả cao hơn so với các phương pháp tìm kiếm truyền thống. Tuy nhiên, phương pháp này vẫn tồn tại hạn chế là chi phí tính toán cao do phải đánh giá nhiều cấu hình tham số trên dữ liệu luồng, làm giảm hiệu năng khi triển khai thực tế.

2.3.3 Pha học kết hợp

Sau khi kết thúc pha tối ưu hóa Bayes, ta có nhiều mô hình đã được huấn luyện với các siêu tham số khác nhau. Pha học kết hợp được thực hiện để xác nhận lại điểm trôi khái niệm một cách chắc chắn hơn bằng cách kết hợp các mô hình tốt nhất.

2.3.3.1 Mô hình E-ERICS

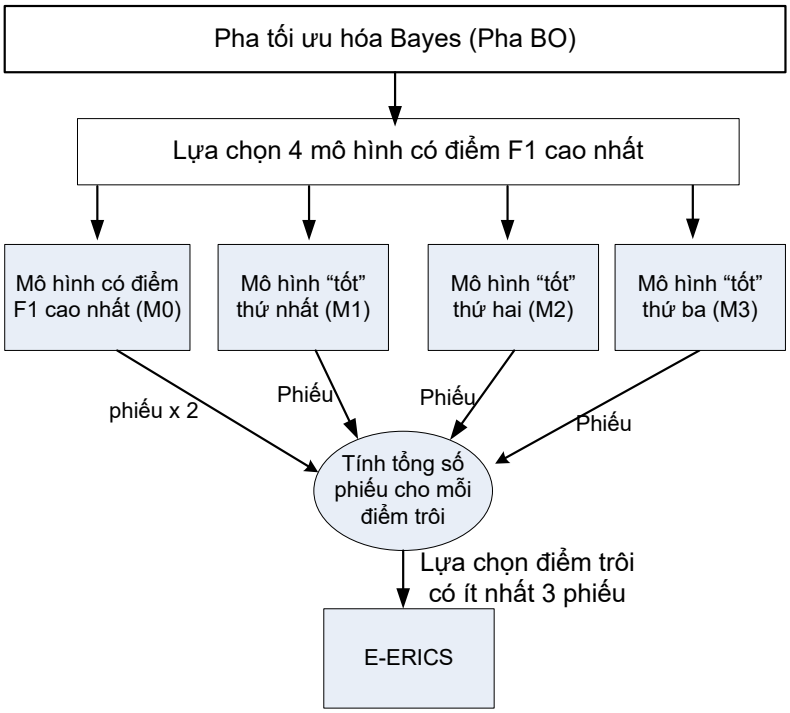
Để xác định các điểm trôi đột ngột hoặc gia tăng một cách chính xác và ổn định hơn, luận án xây dựng cơ chế học kết hợp trên các mô hình đã được tối ưu siêu tham số.

Sau pha BO, thu được n mô hình phát hiện trôi và biết được những mô hình nào có điểm F1 cao nhất (sắp xếp theo thứ tự từ cao đến thấp $M_0, M_1, M_2, \ldots M_n$) trong đó M_0 là mô hình có điểm F1 cao nhất.

Từng mô hình trong n mô hình sẽ độc lập dự đoán xem mỗi điểm trong luồng dữ liệu có phải là điểm trôi hay không. Tuy nhiên nếu lấy cả n mô hình thì khi n lớn sẽ rất khó khăn khi tổng hợp phiếu. Do vậy câu hỏi là: *cần lấy bao nhiêu mô hình để thực hiện bỏ phiếu?*

Để chọn số lượng mô hình bỏ phiếu phù hợp, luận án tiến hành thực nghiệm trong khoảng từ **2 đến 6**. Phương án thực nghiệm cụ thể như sau:

Điểm F1 của tổ hợp không phải là trung bình cộng điểm F1 của từng mô hình riêng lẻ, mà được tính trên kết quả phát hiện cuối cùng của tổ hợp các mô hình thông qua cơ chế bỏ phiếu có trọng số: Mô hình tốt nhất (M_0) có trọng số 2 phiếu, các mô hình còn lại mỗi cái 1 phiếu. Một điểm được xác định là điểm trôi khái niệm nếu có tổng số phiếu quá bán. Đây là kết quả dự đoán hợp nhất của tổ hợp mô hình. Sau đó, điểm F1 được tính bằng công thức thông thường nhưng trên tập kết quả dự đoán hợp nhất đó.



Hình 2.3. Mô hình E-ERICS

Bằng cách thực nghiệm này, luận án nhận thấy khi **số lượng mô hình là 4** sẽ cho kết quả điểm F1 cao nhất. Vì vậy luận án lựa chọn 4 mô hình để bỏ phiếu. Cụ thể, bốn mô hình là M_0 M_1, M_2, M_3. Mỗi mô hình sẽ bỏ phiếu quyết định xem một điểm dữ liệu có phải là điểm trôi hay không. Phiếu của M_0 được gán trọng số gấp đôi so với ba mô hình còn lại. Một điểm được xác nhận là điểm trôi nếu nhận được ít nhất 3 phiếu, như minh họa Hình 2.3.

M_0 là mô hình có F1 cao nhất, phản ánh cấu hình siêu tham số tối ưu nhất do vậy giữ vai trò chủ đạo trong quyết định. Do đó M_0 có số phiếu là 2. Tuy nhiên không để M_0 quyết định hoàn toàn tránh tình huống M_0 sai sót gây báo trôi giả.

Việc gán trọng số gấp đôi cho M_0 thể hiện sự tin tưởng có kiểm soát. Ba mô hình còn lại giúp tạo ra đa dạng hóa trong phát hiện, giảm ảnh hưởng của mô hình đặc biệt nào đó bị tối ưu lệch. Đây là một dạng kết hợp đơn giản nhưng hiệu quả, đủ để tạo tính chắc chắn (qua biểu quyết) nhưng vẫn đảm bảo tính nhanh nhạy (mô hình tốt vẫn phản ứng được). Quy tắc quyết định điểm trôi được tóm tắt như sau:

- Một điểm trôi khái niệm phải được nhiều mô hình xác nhận mới được coi là hợp lệ.
- Các điểm trôi chỉ được phát hiện bởi M_0 (nhưng không phải bởi M_1, M_2, hoặc M_3) sẽ bị loại bỏ vì có thể là điểm báo động giả.
- Các điểm trôi không được M_0 phát hiện nhưng được phát hiện bởi cả M_1, M_2 và M_3 sẽ được thêm vào tập điểm trôi, giúp cải thiện khả năng phát hiện đúng.

Thuật toán: Xác định tập điểm trôi đột ngột và gia tăng

Đầu vào:
M_0, M_1, M_2, M_3: các mô hình phát hiện trôi

Đầu ra:
s: tập các điểm trôi đột ngột hoặc tăng dần

```
begin
  1:  s1 := TapDiemTroi(M1);
  2:  s2 := TapDiemTroi(M2);
  3:  s3 := TapDiemTroi(M3);
  4:  s0 := TapDiemTroi(M0);

  5:  s := (s0 ∩ s1) ∪ (s0 ∩ s2) ∪ (s0 ∩ s3);

  6:  s := s ∪ (s1 ∩ s2 ∩ s3);

  7:  return s;
end;
```

Ghi chú:

- `TapDiemTroi(Mi)` là một thủ tục/hàm giả định trả về tập các điểm trôi đột ngột/gia tăng do mô hình M_i phát hiện. Mục tiêu của hàm: trích xuất

các điểm thời gian trong dòng dữ liệu mà mô hình M_i (với $i \in \{0, 1, 2, 3\}$) đánh dấu là có sự kiện trôi khái niệm.

Giả sử có luồng dữ liệu `data_stream`, sử dụng cửa sổ `W`, hàm `TapDiemTroi(Mi)` được minh họa như sau:

```
def TapDiemTroi(Mi):
begin
    s = set()
    for t in range(W, len(data_stream)):
      indicator  =  Mi.detect_at(data_stream[t-W:t])#áp
      dụng phát hiện trong cửa sổ W
      if indicator == True:
        s.add(t)
    return s
end
```

- Các phép toán như ∩ (giao), ∪ (hợp) được hiểu là trên các tập hợp.

Phương pháp học kết hợp này đảm bảo rằng chỉ những điểm trôi đáng tin cậy nhất được giữ lại. Quá trình này chuyển bài toán thành một bài toán phân lớp nhị phân, trong đó mỗi điểm dữ liệu được phân lớp là điểm trôi khái niệm hoặc không phải điểm trôi dựa trên số phiếu bầu. Mô hình cuối cùng có độ chính xác cao hơn và đáng tin cậy hơn so với bất kỳ mô hình cơ sở nào từ pha tối ưu Bayes.

Đánh giá độ phức tạp tính toán

Thuật toán xác định điểm trôi đột ngột/gia tăng hoạt động dựa trên việc kết hợp thông tin từ bốn mô hình phát hiện trôi độc lập. Cụ thể, thuật toán xây dựng bốn tập điểm trôi từ các mô hình và xác định các điểm trôi tiềm năng theo hai nguyên tắc: (i) các điểm được đồng thuận giữa mô hình M_0 và ít nhất một mô hình còn lại; và (ii) các điểm được đồng thuận giữa cả ba mô hình không phải là mô hình M_0. Quá trình này bao gồm ba phép giao từng cặp giữa M_0 và các mô hình khác, một phép giao ba tập, và tổng cộng ba phép hợp.

Hàm *Tapdiemtroi(M_i)* sử dụng kết quả từ mô hình gốc ERICS [23] mà độ phức tạp xây dựng mô hình gốc này có độ phức tạp $O(lK)$ trong đó l là kích thước của luồng dữ liệu và K số lượng tham số của mô hình (Trong [23], độ phức tạp tính

toán cho một thời điểm t là $O(K)$ cho Công thức 8, một hằng số cho Công thức 9 và cần thực hiện với l thời điểm trong luồng dữ liệu).

Giả sử mỗi tập có kích thước trung bình là l, và các phép toán giao và hợp giữa hai tập có thể được thực hiện trong thời gian tuyến tính theo kích thước tập (dựa trên giả định rằng có thể sử dụng cấu trúc dữ liệu cho phép tra cứu phần tử hiệu quả), độ phức tạp tính toán của thuật toán được đánh giá là $O(l)$.

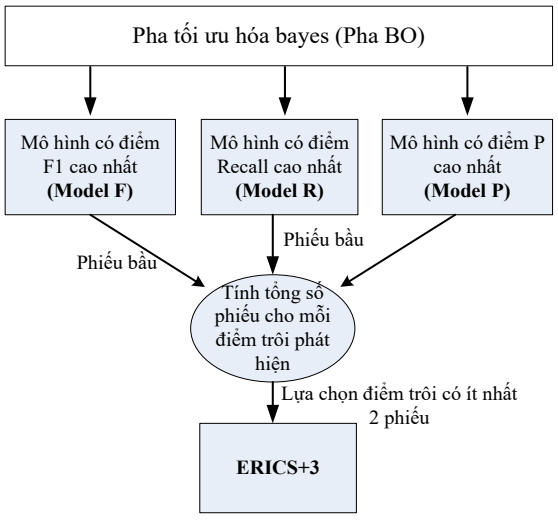
Mặc dù số lượng phép toán tập hợp tương đối lớn, thuật toán vẫn duy trì được độ phức tạp tuyến tính tổng thể, đảm bảo tính hiệu quả khi áp dụng cho bài toán xử lý luồng dữ liệu.

Tóm lại pha học kết hợp tinh chỉnh quá trình phát hiện trôi đột ngột và trôi gia tăng bằng cách sử dụng một hệ thống bỏ phiếu có trọng số.

- Mô hình tốt nhất *(*M_0*)* được kết hợp với ba mô hình bổ trợ (M_1, M_2, M_3) để tạo ra một hệ thống phát hiện trôi mạnh mẽ và chính xác hơn.
- Kỹ thuật học kết hợp này giúp lọc bỏ các điểm trôi giả và khôi phục các điểm trôi bị bỏ sót, dẫn đến một mô hình phát hiện trôi tối ưu hơn.

2.3.3.2 Mô hình ERICS+3

Đối với việc phát hiện kiểu trôi dần dần, sau pha BO, **chọn ba** mô hình bỏ phiếu gồm mô hình có điểm F1 cao nhất (M_F), mô hình có độ hồi tưởng cao nhất (M_R) và mô hình có điểm chính xác cao nhất (M_P). Sau đó, sử dụng cơ chế bỏ phiếu ngang bằng để xác định điểm trôi. Phương án này gọi là ERICS+3 (Hình 2.4).



Hình 2.4. Mô hình ERICS+3

Giải thích lý do lựa chọn 3 mô hình bỏ phiếu như trên:

Trôi dần dần là dạng trôi có tính chất đặc biệt: không có ranh giới rõ ràng giữa khái niệm cũ và khái niệm mới mà chỉ có sự giao thoa [49]. Do đó xác suất phân phối của dữ liệu thay đổi một cách mờ dần theo thời gian, khiến các đặc trưng của lớp cũ và mới có thể cùng tồn tại trong một khoảng thời gian tương đối dài. Điều này dẫn đến hai thách thức chính:

- Tín hiệu trôi thường yếu và không rõ rệt, dễ bị che khuất bởi nhiễu ngẫu nhiên.
- Các mô hình đơn lẻ thu được từ pha BO có thể không ổn định – có mô hình nhạy hơn với tín hiệu mới (R cao), có mô hình lại quá thận trọng để tránh báo động giả (P cao), và có mô hình đạt cân bằng giữa hai yếu tố này (F1 cao).

Chính vì thế, việc chỉ dựa vào một mô hình duy nhất sẽ dễ dẫn đến thiên lệch – hoặc là bỏ sót trôi (nếu quá thận trọng), hoặc là báo động sai (nếu quá nhạy). Để khắc phục điều này, luận án lựa chọn một bộ ba mô hình đại diện cho ba góc nhìn bổ sung lẫn nhau:

- M_F cung cấp quyết định dựa trên sự cân bằng tổng thể (Độ chính xác P và Độ hồi tưởng R).
- M_P đại diện cho tính cẩn trọng trong phát hiện – giúp giảm báo động sai.
- M_R mang lại độ nhạy cao – đảm bảo khả năng không bỏ sót trôi.

Ba mô hình đại diện cho ba loại phản ứng khác nhau, nên cơ chế bỏ phiếu giúp trung hòa quan điểm, phù hợp với các giai đoạn mơ hồ và chuyển tiếp trong trôi dần dần.

Từ kết quả của pha BO, ba mô hình có hiệu năng tốt nhất theo các tiêu chí khác nhau được lựa chọn để tạo thành hệ thống học kết hợp, cụ thể:

- M_F: Là mô hình có điểm F1 cao nhất. Nếu có nhiều mô hình có cùng giá trị F1, thì mô hình có độ chính xác cao nhất sẽ được chọn.
- M_P: Là mô hình có độ chính xác cao nhất trong số các mô hình còn lại. Nếu có nhiều mô hình có cùng độ chính xác cao nhất, thì mô hình có điểm F1 cao nhất sẽ được ưu tiên.

- M_R: Là mô hình có độ hồi tưởng cao nhất trong số các mô hình còn lại. Nếu có nhiều mô hình có cùng độ hồi tưởng cao nhất, thì mô hình có điểm F1 cao nhất sẽ được chọn.

Sau khi chọn được ba mô hình tối ưu, quá trình học kết hợp sử dụng cơ chế *bỏ phiếu ngang bằng* để đưa ra quyết định cuối cùng về việc một điểm dữ liệu có phải là điểm trôi khái niệm hay không.

- Mỗi mô hình M_F, M_P, M_R sẽ đưa ra một dự đoán về một điểm dữ liệu nhất định.
- Nếu **ít nhất hai trong ba mô hình** xác nhận điểm dữ liệu là điểm trôi khái niệm, thì điểm đó sẽ được ghi nhận là điểm trôi trong mô hình cuối cùng.

Nếu chỉ có một mô hình phát hiện trôi trong khi hai mô hình còn lại không phát hiện, thì điểm đó sẽ bị loại bỏ để giảm thiểu tỷ lệ báo động sai.

Sau khi quá trình bỏ phiếu hoàn tất, mô hình kết hợp cuối cùng sẽ được hình thành. Mô hình này có hai đặc điểm chính:

- Giảm thiểu báo động giả: Các điểm trôi chỉ được một mô hình phát hiện nhưng không được hai mô hình còn lại xác nhận sẽ bị loại bỏ. Điều này giúp đảm bảo rằng mô hình chỉ nhận diện những điểm trôi thực sự đáng tin cậy.
- Tăng cường khả năng phát hiện điểm trôi bị bỏ sót: Nếu một điểm trôi không được mô hình có điểm F1 cao nhất (M_F) phát hiện nhưng lại được cả hai mô hình còn lại (M_P và mô hình M_R) phát hiện, thì điểm đó sẽ được thêm vào danh sách điểm trôi. Điều này giúp mô hình có khả năng phát hiện trôi toàn diện hơn, giảm thiểu nguy cơ bỏ sót những thay đổi quan trọng trong dữ liệu.

Thuật toán: Xác định tập điểm trôi dần dần

Đầu vào: M_F, M_R, M_P: các mô hình phát hiện trôi dần dần
Đầu ra: s: tập các điểm trôi dần

```
Begin
  1:  s1 := TapDiemTroi(M_F);
  2:  s2 := TapDiemTroi(M_R);
  3:  s3 := TapDiemTroi(M_P);

  4:  s := (s1 ∩ s2) ∪ (s2 ∩ s3) ∪ (s1 ∩ s3);

  5:  return s;
End;
```

- `TapDiemTroi(M_x)` là một thủ tục/hàm giả định trả về tập các điểm trôi *dần dần* do mô hình M_x phát hiện.
- Các phép toán như ∩ (giao), ∪ (hợp) được hiểu là trên các tập hợp.

Đánh giá độ phức tạp tính toán

Thuật toán xác định điểm trôi dần dần sử dụng ba mô hình phát hiện độc lập để đưa ra các tập điểm trôi ứng viên. Các điểm được xem là trôi dần dần nếu chúng xuất hiện đồng thời trong ít nhất hai trong ba tập này. Quá trình này được hiện thực thông qua việc tính giao từng cặp giữa các tập và hợp nhất các kết quả giao lại thành tập điểm trôi đầu ra.

Với giả định mỗi tập có kích thước trung bình là l, và mỗi phép giao hoặc hợp có thể thực hiện trong thời gian tuyến tính, tổng số phép toán trong thuật toán bao gồm ba phép giao và hai phép hợp, dẫn đến độ phức tạp tính toán tổng thể là $O(l)$. Do không cần xử lý đồng thời nhiều hơn hai tập ở mỗi phép toán, thuật toán này có chi phí tính toán thấp hơn trong thực tiễn và vẫn đáp ứng tốt yêu cầu phát hiện trôi dần dần trong luồng dữ liệu có quy mô lớn.

2.4 Thực nghiệm và đánh giá

2.4.1 Mục tiêu và kịch bản thực nghiệm

Trong luận án này, các mô hình E-ERICS và ERICS+3 được đề xuất như là những cải tiến trực tiếp của ERICS. Do đó, quá trình đánh giá tập trung vào việc so sánh trực tiếp với ERICS, nhằm chứng minh hiệu quả vượt trội của các phương pháp được đề xuất. Trước đó, ERICS đã được đánh giá rộng rãi và cho thấy hiệu năng cao hơn các thuật toán phát hiện trôi phổ biến như DDM hay ADWIN [23].

Thực nghiệm của luận án nhằm đạt được hai mục tiêu chính sau đây

i. Mục tiêu thứ nhất là đánh giá hiệu năng của E-ERICS trong việc phát hiện điểm trôi đột ngột và gia tăng, so sánh với mô hình ERICS gốc.

ii. Mục tiêu thứ hai là đánh giá hiệu năng của ERICS+3 trong việc phát hiện điểm trôi dần dần, so sánh với mô hình ERICS gốc.

Kịch bản thực nghiệm:

i. Kịch bản đầu tiên, so sánh hiệu năng của ERICS với hiệu năng của BO-ERICS (mô hình thu được với bộ siêu tham số đã được tối ưu) và của E-ERICS (mô hình thu được sau pha học kết hợp). Trong kịch bản này, luận án sử dụng bộ dữ liệu có trôi đột ngột và gia tăng (bao gồm cả dữ liệu tổng hợp và dữ liệu thực tế). Sau đó lần lượt chạy thực nghiệm ERICS, BO-ERICS và E-ERICS trên cùng bộ dữ liệu để có kết quả và so sánh hiệu năng, với kỳ vọng rằng BO-ERICS và E-ERICS sẽ phát hiện đúng được nhiều điểm trôi đột ngột và gia tăng hơn so với ERICS.

ii. Kịch bản so sánh ERICS+3 và ERICS: luận án sử dụng bộ dữ liệu có trôi dần dần, sau đó lần lượt chạy ERICS và ERICS+3 trên cùng bộ dữ liệu để có kết quả và so sánh hiệu năng, với kỳ vọng rằng ERICS+3 sẽ phát hiện được nhiều điểm trôi dần dần hơn so với ERICS.

2.4.2 Thiết lập thực nghiệm

Luận án kế thừa phương thức thực nghiệm mà ERICS đã thực hiện, được mô tả chi tiết dưới đây.

2.4.2.1 Bộ dữ liệu

Bộ dữ liệu trôi đột ngột/gia tăng sử dụng cho mô hình E-ERICS

- *Bộ dữ liệu tổng hợp*: luận án sử dụng thư viện Scikit-multiflow để tạo ra hai bộ dữ liệu tổng hợp mô phỏng các dạng trôi khái niệm khác nhau. Mỗi bộ dữ liệu bao gồm **100.000** mẫu dữ liệu và được lưu dưới định dạng .csv để phục vụ huấn luyện và đánh giá các mô hình phát hiện trôi. Sử dụng bộ sinh *SEA* để sinh các luồng trôi đột ngột và bộ sinh *Hyperplane* để sinh các luồng trôi gia tăng.
    - Bộ dữ liệu *SEA*

      Bộ dữ liệu này được sinh ra từ *SEAGenerator* của thư viện Scikit-multiflow, trong đó mỗi mẫu gồm 3 thuộc tính thực. Quy tắc phân loại

được xác định dựa trên tổng của hai thuộc tính là *attr1* và *attr2*: Nếu (*attr1*+*attr2* $\leq$ *ngưỡng*) $\Rightarrow$ lớp 0, ngược lại là lớp 1. Giá trị ngưỡng thuộc tập {8,9,7} sẽ được thay đổi theo từng giai đoạn để tạo ra trôi khái niệm đột ngột. Dữ liệu được bổ sung 10% nhiễu ngẫu nhiên nhằm tăng tính thực tế cho bài toán. Hiện tượng trôi khái niệm đột ngột được mô phỏng bằng cách luân phiên thay đổi giữa bốn hàm phân loại khác nhau (tương ứng với các giá trị khác nhau của tham số *classification_function* trong *SEAGenerator*).

- Bộ dữ liệu *Hyperplane*

  Bộ dữ liệu này được sinh ra từ *HyperplaneGenerator* trong Scikit-multiflow, mô phỏng bài toán phân lớp trong không gian 20 chiều. Mỗi mẫu dữ liệu có d đặc trưng đại diện cho vị trí tương đối của một điểm so với một siêu phẳng phân tách. Mỗi mẫu được gán nhãn dựa trên dấu của biểu thức:

  $$\sum_{i=1}^{d} w_i x_i > \theta \Rightarrow$$ lớp 0, ngược lại là lớp 1.

  $w_i \in [0,1]$ là hệ số trọng số của mặt phẳng siêu phẳng.

  Trôi khái niệm được sinh ra bằng cách thay đổi tuyến tính hoặc ngẫu nhiên các hệ số w_i theo thời gian, dẫn đến sự thay đổi ranh giới phân lớp trong không gian nhiều chiều. Đây là điển hình của trôi gia tăng. Dữ liệu được bổ sung 10% nhiễu, và hiện tượng trôi khái niệm tăng dần được mô phỏng bằng cách thay đổi phương trình của siêu phẳng sau mỗi 100 mẫu**.**

- *Bộ dữ liệu thực tế:* sử dụng 4 bộ dữ liệu KDD, Spambase, Dota2, Adult. Trong đó bộ KDD có tổng là 4 triệu mẫu, nhưng luận án chỉ lấy ra ngẫu nhiên 100 nghìn mẫu trong đó để tiến hành thử nghiệm (bằng số lượng mẫu mà mô hình ERICS thử nghiệm). Các bộ khác được giữ nguyên số lượng mẫu. Để đánh giá định lượng mô hình cần phải biết điểm trôi thật. Nhưng các bộ dữ liệu thực tế không cung cấp thông tin này, do đó J. Haug và G. Kasneci trong ERICS chèn điểm trôi khái niệm nhân tạo vào các bộ dữ liệu thực tế. Phương pháp thực hiện chèn điểm trôi vào các bộ dữ liệu thực tế này như sau:

- Bước 1. Xáo trộn ngẫu nhiên dữ liệu gốc: Dữ liệu ban đầu được xáo trộn để loại bỏ các hiện tượng trôi tự nhiên tiềm ẩn mà người dùng không thể kiểm soát được. Điều này đảm bảo rằng mọi sự thay đổi trong dữ liệu sau đó là do can thiệp nhân tạo.
- Bước 2. Xếp hạng thuộc tính theo thông tin thu được: Các đặc trưng được đánh giá và xếp hạng dựa trên mức độ đóng góp của chúng vào phân biệt lớp nhãn thông qua độ đo thông tin thu được.
- Bước 3. Lựa chọn 50% thuộc tính có giá trị thông tin cao nhất: Sau khi xếp hạng, chọn ra một nửa thuộc tính quan trọng nhất để thực hiện chèn can thiệp.
- Bước 4. Xáo trộn giá trị các thuộc tính được chọn: Giá trị của các thuộc tính được chọn sẽ được hoán vị một cách ngẫu nhiên để làm mất đi mối liên hệ ban đầu với lớp nhãn, từ đó mô phỏng sự thay đổi đột ngột trong phân phối dữ liệu.

  Tần suất chèn điểm trôi: Cứ mỗi 20% số mẫu, một điểm trôi được đưa vào. Quá trình trên được lặp lại để chèn một điểm trôi nhân tạo. Như vậy có tổng **cộng bốn điểm trôi đột ngột** được tạo ra.

Bộ dữ liệu trôi dần dần sử dụng cho ERICS+3

Luận án sử dụng các bộ dữ liệu tổng hợp gồm: SEA, Stagger, Sine, và Agrawal. Mỗi bộ dữ liệu chứa 100.000 hoặc 200.000 mẫu và được lưu trữ dưới định dạng .csv. Đặc trưng cụ thể của từng bộ dữ liệu tổng hợp được trình bày trong Bảng 2.1.

Bảng 2.1. Các khoảng trôi dần dần được tạo ra

Bộ dữ liệu	Số lượng mẫu	Khoảng trôi (từ lô – đến lô)
SEA	200,000	20000 - 40000, 90000 - 100000, 160000 - 195000
Stagger	100,000	15000-20000, 35000-40000, 55000 - 60000, 75000 – 80000
Sine	100,000	15000 - 20000, 35000 - 40000, 55000 - 60000, 75000 – 80000
Agrawal	100,000	10000 - 20000, 45000 - 50000, 80000 - 95000

2.4.2.2 Cấu hình mô hình

- Phần cứng: các mô hình được đánh giá trên máy ảo Google Colab với cấu hình 2 bộ vi xử lý Intel(R) Xeon(R) CPU @ 2.20GHz, RAM: 12.69 GB.
- Phần mềm: Trong pha BO, các thực nghiệm sử dụng thư viện Python *BayesOptimization.*[8] Đây là một gói phần mềm tối ưu hóa toàn cục có ràng buộc, được xây dựng dựa trên suy luận Bayes và các quá trình Gaussian, với mục tiêu tìm giá trị cực đại của một hàm chưa biết chỉ trong số vòng lặp ít nhất có thể.
- Trong thí nghiệm so sánh giữa E-ERICS và ERICS: đối với mỗi bộ dữ liệu, khởi tạo (***m = 10***) mô hình cơ sở với tập giá trị siêu tham số ban đầu và sau đó tự động tạo thêm (***n = 350***) mô hình đề xuất với tập giá trị siêu tham số mới nhằm tối ưu hóa điểm F1 của mô hình cơ sở. Không gian tìm kiếm các siêu tham số được liệt kê ở Bảng 2.2. Trong ERICS, tác giả thực hiện huấn luyện các mô hình cơ sở (hồi quy Probit) theo từng lô, với kích thước lô là **10** cho bộ Spambase, **50** cho bộ Adult, và **100** cho bộ KDD, Dota2, và các bộ dữ liệu tổng hợp. Luận án cũng tiến hành các thực nghiệm của mình trong cùng điều kiện như trên.
- Trong thí nghiệm so sánh giữa ERICS+3 và ERICS: trong pha BO khởi tạo **10** mô hình cơ sở với tập giá trị siêu tham số ngẫu nhiên từ không gian tìm kiếm, sau đó tự động tạo thêm **250** mô hình với tập tham số mới. Không gian tìm kiếm của các siêu tham số được xác định như sau: M từ 50 đến 200, W từ 10 đến 75, và β từ 10^{-5}đến 10^{-2}. Đối với tất cả các mô hình được tạo ra trong quá trình BO, luận án sử dụng kích thước lô là **100**.

2.4.2.3 Chiến lược huấn luyện và đánh giá

Sử dụng cách đánh giá xen kẽ việc kiểm tra với huấn luyện (interleaved test-then-train), mỗi mẫu dữ liệu có hai vai trò: đầu tiên, mô hình dự đoán nhãn cho mẫu và cập nhật các chỉ số đánh giá hiệu năng. Sau đó, mẫu dữ liệu được sử dụng để huấn luyện (tối ưu dần) mô hình. Phương pháp này giúp sử dụng toàn bộ dữ

[8] https://github.com/fmfn/BayesianOptimization

liệu cho quá trình huấn luyện, đồng thời tiết kiệm bộ nhớ vì không cần lưu trữ một tập kiểm tra riêng biệt, phù hợp với các mô hình học máy trực tuyến.

2.4.3 Các độ đo đánh giá

Sử dụng độ đo chính xác **P**, độ hồi tưởng **R** và điểm **F1** là những chỉ số hiệu năng đạt được của mô hình ERICS, đã được mô tả chi tiết trong Mục 1.6.2.

Với độ chính xác **P**: Ghi nhận một dương tính đúng TP1 khi một điểm trôi khái niệm được phát hiện **trong 50 lô** dữ liệu *kể từ điểm trôi thực sự xảy ra* và ghi một dương tính sai FP1 nếu điểm trôi khái niệm được phát hiện **sau 50 lô** dữ liệu kể trên.

Với độ hồi tưởng **R**: Ghi một âm tính sai FN2 nếu *không* có một điểm trôi khái niệm nào được phát hiện **trong 50 lô** dữ liệu *kể từ khi điểm trôi thực sự xảy ra* và ghi một dương tính đúng TP2 nếu một điểm trôi được phát hiện **trong 50 lô**.

2.4.4 Trình bày và phân tích kết quả

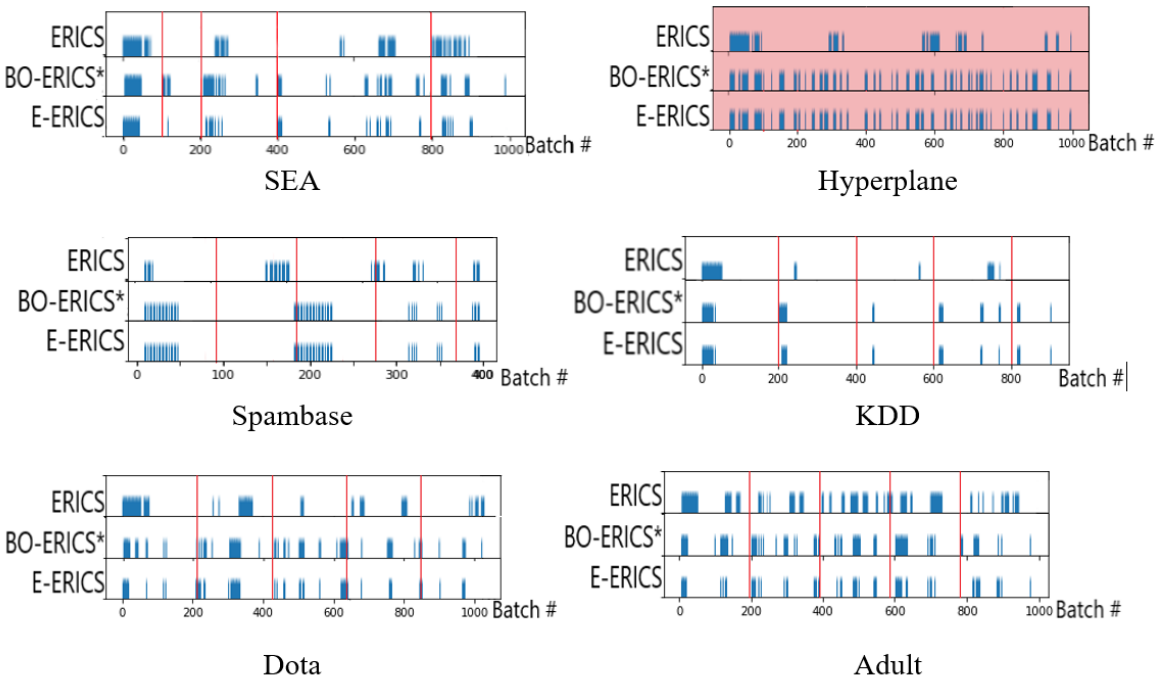
2.4.4.1 Kết quả phát hiện trôi đột ngột và gia tăng của E-ERICS

Sau khi thực hiện pha BO trên từng bộ dữ liệu, giá trị bộ siêu tham số tối ưu *(M, W, β)* được xác định như trong Bảng 2.2 dưới đây.

Bảng 2.2. Bảng giá trị các siêu tham số tối ưu

Bộ dữ liệu	**M**			**W**			**-logβ**		
	Không gian tìm kiếm	ERICS	Pha BO	Không gian tìm kiếm	ERICS	Pha BO	Không gian tìm kiếm	ERICS	Pha BO
SEA	(20, 90)	75	35	(10,70)	50	50	(2, 5)	4	3.535
Hyperplane	(20,110)	100	53	(10,70)	50	10	(2, 5)	4	2.019
KDD	(20, 90)	50	26	(8, 80)	50	24	(2, 5)	4	3.677
Spambase	(10, 70)	35	53	(10,60)	25	46	(2, 5)	3	2.032
Dota2	(20, 90)	75	38	(10,75)	50	10	(2, 5)	4	3.179
Adult	(10, 60)	50	24	(10,60)	50	31	(2, 5)	3	3.096

Độ trễ và độ chính xác phát hiện trôi được mô tả ở Hình 2.5.



Hình 2.5. So sánh độ trễ và độ chính xác phát hiện trôi đột ngột/gia tăng

Trong Hình 2.5, trục ngang thể hiện vị trí các lô (batch). Đường dọc đỏ là các điểm trôi khái niệm đột ngột thực tế xảy ra; đường xanh ngắn là các điểm trôi được mô hình phát hiện; vùng màu đỏ của bộ Hyperplane thể hiện rằng trong bộ này *tất cả các điểm dữ liệu đều chứa điểm trôi gia tăng liên tiếp nhau*. Như vậy ngoài vùng đỏ Hyperplane, các bộ dữ liệu khác chỉ có bốn điểm trôi dữ liệu thực tế (bốn đường đỏ). Ký hiệu BO-ERICS là mô hình thu được sau pha tối ưu hóa siêu tham số. E-ERICS là mô hình thu được sau pha học kết hợp.

Quan sát các điểm trôi màu xanh ngắn mà mỗi mô hình phát hiện, có thể thấy sự cải thiện đáng kể về độ chính xác và độ trễ của E-ERICS so với ERICS.

Ví dụ trong bộ SEA, quan sát điểm trôi đỏ đầu tiên (tính từ trái qua phải), các điểm xanh bên trái là các điểm phát hiện sai vì như đã đề cập ở mục 2.4.3 về các độ đo đánh giá, điểm đúng chỉ tính trong 50 lô *kể từ khi* điểm trôi đỏ xảy ra. Bên phải điểm trôi đỏ này có các điểm màu xanh. Nếu điểm xanh nào nằm trong khoảng 50 lô tính từ điểm đỏ, thì đó là điểm trôi được phát hiện đúng, ngoài phạm vi 50 lô thì coi là điểm trôi phát hiện sai. Độ trễ phát hiện là khoảng cách từ điểm đỏ đến các điểm xanh nằm trong khoảng 50 lô về bên phải. Quan sát bên *trái* điểm đỏ đầu tiên bộ SEA, thấy rằng ERICS phát hiện sai nhiều điểm trôi hơn so với BO-ERICS và E-ERICS. Quan sát bên *phải* điểm đỏ đầu tiên, ERICS không phát

hiện ra điểm trôi nào (bỏ sót) trong khi BO-ERICS và E-ERICS phát hiện đúng một số điểm trôi.

Quan sát điểm đỏ thứ hai trong bộ SEA thấy rằng về bên phải điểm này, cả BO-ERICS và E-ERICS phát hiện ra nhiều điểm trôi đúng hơn so với ERICS.

Quan sát điểm đỏ thứ ba trong bộ SEA, ERICS tiếp tục bỏ sót điểm trôi trong khi BO-ERICS và E-ERICS phát hiện thấy.

Quan sát điểm đỏ thứ tư trong bộ SEA thì thấy rằng ERICS phát hiện ra nhiều điểm trôi đúng hơn so với BO-ERICS và E-ERICS nhưng đồng thời cũng phát hiện nhiều điểm trôi sai vì nhiều điểm trôi nằm quá giới hạn 50 lô.

Quan sát kết quả trong bộ Hyperplane là bộ mà *mọi lô dữ liệu đều chứa điểm trôi tăng dần*, cho thấy BO-ERICS và E-ERICS phát hiện đúng nhiều điểm trôi hơn so với ERICS.

Quan sát kết quả trên bộ Spambase cho thấy tại điểm trôi đỏ đầu tiên, cả ERICS, BO-ERICS và E-ERICS đều không phát hiện ra. Đến điểm trôi đỏ thứ thứ hai, E-ERICS phát hiện ra còn ERICS thì không. Điểm đỏ thứ ba thì ERICS phát hiện thấy còn E-ERICS không phát hiện ra. Điểm trôi đỏ thứ tư thì ba mô hình đều phát hiện các điểm trôi đúng, BO-ERICS vượt trội hơn trong khi ERICS và E-ERICS gần tương đương.

Quan sát kết quả trên các bộ KDD, Dota2, Adult đều cho thấy rõ sự cải thiện đáng kể trong độ chính xác và độ trễ của E-ERICS.

Kết quả trong các Bảng từ 2.3 đến 2.6 là kết quả thực nghiệm của luận án.

Bảng 2.3. So sánh hiệu năng trên bộ SEA và Hyperplane

Các độ đo	SEA			Hyperplane		
	ERICS	BO-ERICS	E-ERICS	ERICS	BO-ERICS	E-ERICS
R	0.5000	1.0000	**1.0000**	0.1701	**1.0000**	**1.0000**
P	0.3103	0.5306	**0.5778**	1.0000	**1.0000**	**1.0000**
Điểm F1	0.3830	0.6933	**0.7324**	0.1748	**1.0000**	**1.0000**

Trong **Bảng 2.3** cho thấy rõ hiệu quả của từng pha của mô hình hai pha đề xuất. Trên bộ dữ liệu **SEA**, mô hình ERICS ban đầu chỉ đạt R = 0.5, cho thấy chỉ phát hiện được một nửa số điểm trôi thực sự. P cũng khá thấp (0.3103), khiến điểm

F1 chỉ đạt 0.3830. Khi áp dụng tối ưu hóa Bayes, khả năng phát hiện trôi tăng mạnh (R = 1.0), P cũng cải thiện lên 0.5306, giúp điểm F1 đạt 0.6933. Đáng chú ý, pha học kết hợp đã nâng P lên 0.5778, trong khi vẫn duy trì R = 1.0. Nhờ đó, điểm F1 tiếp tục tăng lên 0.7324, cao nhất trong ba giai đoạn. Điều này cho thấy cơ chế bỏ phiếu trong pha học kết hợp giúp loại bỏ các báo động giả, làm tăng độ chính xác mà không ảnh hưởng đến khả năng bao phủ.

Trên tập dữ liệu **Hyperplane**, **đây là tập dữ liệu mà mọi lô dữ liệu đều chứa điểm trôi gia tăng.** Chính vì vậy khi sử dụng kết hợp bốn mô hình phát hiện trong E-ERICS mô hình đạt hiệu năng lý tưởng với P = R = F1 = 1.0 (không phải do quá khớp). Điều này khẳng định rằng E-ERICS, sau pha tối ưu và học kết hợp, giúp nâng cao cả độ chính xác và độ đầy đủ trong phát hiện trôi.

Bảng 2.4. So sánh hiệu năng trên bộ KDD và Spambase

Các độ đo	KDD			Spambase		
	ERICS	BO-ERICS	E-ERICS	ERICS	BO-ERICS	E-ERICS
R	0.2500	**1.0000**	**1.0000**	0.7500	0.7500	0.7500
P	0.2667	0.7333	**0.7917**	0.6061	**0.8611**	0.8571
Điểm F1	0.2581	0.8462	**0.8837**	0.6704	**0.8017**	0.8000

Bảng 2.5. So sánh hiệu năng trên bộ Dota2 và Adult

Các độ đo	Dota2			Adult		
	ERICS	BO-ERICS	E-ERICS	ERICS	BO-ERICS	E-ERICS
R	0.2500	1.0000	**1.0000**	1.0000	1.0000	**1.0000**
P	0.0417	0.3913	**0.4000**	0.2212	0.3929	**0.3934**
Điểm F1	0.0714	0.5625	**0.5714**	0.3623	0.5641	**0.5647**

Các **Bảng từ 2.4 đến Bảng 2.5** cho thấy rằng, nhờ cải thiện các siêu tham số quan trọng (M, W, β), một số mô hình trong BO-ERICS đã đạt được hiệu năng vượt trội so với ERICS trong tất cả các bộ dữ liệu.

Bảng 2.6. So sánh hiệu năng trung bình trên các bộ dữ liệu

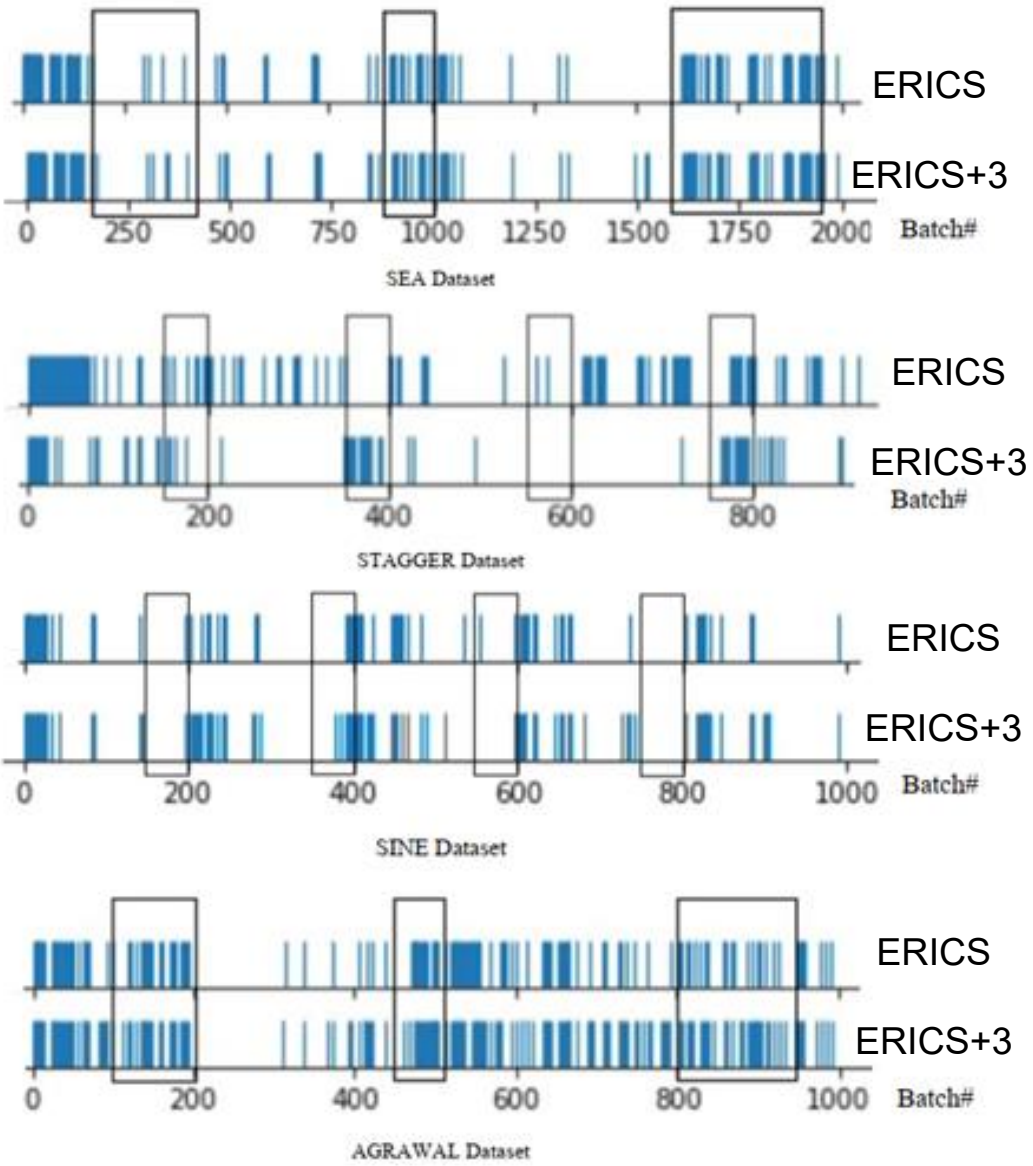
Các độ đo	Giá trị trung bình		
	ERICS	**Pha BO**	**Pha học kết hợp**
R	0.4867	**0.9583**	**0.9583**
P	0.4077	0.6515	**0.6700**
Điểm F1	0.3200	0.7446	**0.7509**

Đối với **Bảng 2.6** so sánh hiệu năng trung bình, BO-ERICS giúp nâng R lên mức cao (≈ 0.96), đảm bảo không bỏ sót các điểm trôi khái niệm. Sau đó, E-ERICS với cơ chế bỏ phiếu từ nhiều mô hình tối ưu tiếp tục cải thiện P bằng cách lọc bỏ các báo động giả. Nhờ đó, điểm F1 – thước đo cân bằng giữa độ chính xác và độ bao phủ – được cải thiện một cách rõ rệt và ổn định trên cả 4 bộ dữ liệu. Trung bình điểm F1 tăng từ 0.32 (ERICS) lên 0.7509 (E-ERICS).

Những kết quả này chứng minh rằng việc kết hợp giữa tối ưu hóa Bayes và học kết hợp giúp hệ thống phát hiện trôi khái niệm đột ngột và gia tăng, không chỉ nhạy hơn, mà còn chính xác và đáng tin cậy hơn.

2.4.4.2 Kết quả phát hiện trôi dần dần của ERICS+3

Hình 2.6 so sánh độ chính xác và độ trễ phát hiện trôi dần dần dưới dạng biểu đồ trực quan. *Khung chữ nhật* là khoảng thời gian xảy ra trôi dần dần. Khác với trôi đột ngột là điểm trôi chỉ có một điểm còn trôi dần dần diễn ra trong một khoảng thời gian. Các điểm *màu xanh ngắn* là các điểm trôi mà mô hình phát hiện. Trong mỗi bộ dữ liệu, hàng trên đầu tiên là kết quả phát hiện của ERICS, hàng bên dưới là kết quả của ERICS+3. Những điểm trôi đúng sẽ nằm trong phạm vi hình chữ nhật. Quan sát các điểm màu xanh trong các khung chữ nhật, ở các bộ dữ liệu, cho thấy ERICS+3 phát hiện nhiều điểm trôi dần dần hơn so với ERICS.



Hình 2.6 So sánh độ trễ và độ chính xác phát hiện trôi dần dần

Cụ thể trong các bộ dữ liệu có thể thấy trên bộ SEA, số lượng các điểm trôi mà hai mô hình phát hiện ra gần tương đương, ERICS+3 phát hiện được nhiều hơn. Trong bộ Stagger, ERICS+3 tỏ ra vượt trội hơn so với ERICS. Trong bộ Sine và Agrawal, hai mô hình gần tương đương.

Những nhận xét này phù hợp với kết quả định lượng được chỉ ra trong Bảng 2.7.

Bảng 2.7 là các chỉ số hiệu năng của hai mô hình đạt được trong thí nghiệm.

Bảng 2.7. So sánh hiệu năng giữa các mô hình

Mô hình	Bộ dữ liệu	P	R	Điểm F1
ERICS	**SEA**	0.5586	0.9168	0.6943
ERICS+3		**0.5714**	**0.9169**	**0.7041**
ERICS	**Stagger**	**0.5588**	0.7500	0.6404
ERICS+3		0.5385	**1.0000**	**0.7000**
ERICS	**Sine**	0.5155	0.9850	0.6768
ERICS+3		**0.5190**	**0.9858**	**0.6798**
ERICS	**Agrawal**	0.4860	0.9831	0.6505
ERICS+3		**0.5401**	**0.9836**	**0.6973**

Bảng 2.8. So sánh trung bình hiệu năng giữa các mô hình trên các bộ dữ liệu

Độ đo	ERICS	ERICS+3
P trung bình	0.5297	**0.5423**
R trung bình	0.9088	**0.9713**
Điểm F1 trung bình	0.6655	**0.6953**

Kết quả từ Bảng 2.7 thu được từ bốn bộ dữ liệu tổng hợp (SEA, Stagger, Sine, Agrawal) cho thấy mô hình ERICS+3 đạt hiệu năng vượt trội so với mô hình ERICS gốc ở cả ba chỉ số đánh giá: P, R và điểm F1. Cụ thể, trên tập dữ liệu SEA, mô hình ERICS+3 đạt P cao hơn ERICS (0.5714 so với 0.5586), trong khi R cao hơn một chút, giúp điểm F1 tăng từ 0.6943 lên 0.7041. Trên tập **Stagger**, mặc dù ERICS+3 có điểm P thấp hơn một chút so với ERICS, nhưng độ đo R thể hiện khả năng phát hiện tất cả các điểm trôi dần dần thực sự (R = 1.0), cao hơn đáng kể so với ERICS (R = 0.75). Điều này dẫn đến sự cải thiện rõ rệt của điểm F1 (0.7000 so với 0.6404). Đối với bộ **Sine**, mặc dù sự khác biệt là không lớn, ERICS+3 vẫn ghi nhận mức P và điểm F1 cao hơn, cho thấy tính ổn định của mô hình ngay cả khi xử lý các loại trôi dao động đều. Trên tập dữ liệu **Agrawal**, một bộ có đặc trưng phức tạp và nhiều biến thể ngầm, ERICS+3 cải thiện đáng kể P (0.5401 so với 0.4860) nhờ giảm số lượng báo động sai, trong khi vẫn duy trì độ bao phủ rất cao (R = 0.9833).

Sự cải thiện này được tổng quát hóa trong Bảng 2.8, thể hiện giá trị trung bình trên tất cả các bộ dữ liệu. Trung bình P của ERICS+3 đạt 0.5423 (so với 0.5297 của ERICS), R đạt 0.9713 (so với 0.9088), và điểm F1 đạt 0.6953 (so với 0.6655).

Các kết quả này cho thấy rằng ERICS+3 không chỉ giúp phát hiện đầy đủ hơn các điểm trôi (R tăng), mà còn làm giảm đáng kể các báo động giả (P), từ đó cải thiện hiệu năng tổng thể của hệ thống phát hiện trôi khái niệm.

Tổng thể, có thể khẳng định rằng mô hình ERICS+3 mang lại sự cân bằng tốt hơn giữa độ nhạy và độ chính xác, đồng thời cho thấy tính hiệu quả và ổn định cao hơn so với ERICS gốc trên các loại dữ liệu tổng hợp khác nhau.

2.5 Kết luận Chương 2

Chương này giới thiệu mô hình phát hiện trôi khái niệm gồm hai pha là pha tối ưu siêu tham số và pha tổng hợp, trong đó đề xuất hai mô hình E-ERICS phát hiện trôi đột ngột/gia tăng và mô hình ERICS+3 phát hiện trôi dần dần, cho kết quả chính xác hơn với độ trễ thấp hơn so với mô hình ERICS. Một phần các kết quả nghiên cứu của chương này được công bố trong [TungNK1] (đã nhận được một tham chiếu Scopus từ các tác giả nước ngoài[9]), [TungNK2].

Trong chương sau, luận án sẽ trình bày các đề xuất của luận án về các kỹ thuật học máy phân lớp trôi khái niệm trong dữ liệu luồng.

[9] [Trat23] Martin Trat, Jivka Ovtcharova. *Designing Concept Drift Detection Ensembles: A Survey*. DSAA 2023: 1-10.

Chương 3. Mô hình phân lớp trôi khái niệm dựa trên mạng nguyên mẫu

Để giải quyết bài toán phân lớp trôi, luận án tiếp cận theo hướng học ít mẫu thông qua mạng nguyên mẫu, như đã đề cập trong Mục 1.3.4. Phương pháp này cho phép mô hình phân biệt được các kiểu trôi dựa trên đặc trưng biểu hiện của từng loại, ngay cả khi số lượng mẫu cho mỗi loại trôi là rất hạn chế – điều thường gặp trong môi trường dữ liệu luồng thực tế.

Vì vậy mục đầu tiên của chương này trình bày tiếp cận phân lớp trôi khái niệm dựa trên mạng nguyên mẫu. Hai mục tiếp theo trình bày về đề xuất của luận án về hai mô hình phân lớp trôi khái niệm VAR-WIND và MetaLDD-Finetune, kế thừa phương pháp từ Meta-ADD.

3.1 Phân lớp trôi khái niệm dựa trên mạng nguyên mẫu

Trong học máy trên luồng dữ liệu, khi một điểm trôi được phát hiện, nhiệm vụ phân lớp là xác định kiểu trôi tại điểm đó, ví dụ: trôi đột ngột, dần dần, gia tăng, hoặc bình thường (không trôi). Phân lớp trôi khái niệm là một bài toán phân lớp đa lớp và được phát biểu một cách hình thức như sau:

- **Đầu vào**: Một luồng dữ liệu D_{tream} có kích thước n, một tập dữ liệu huấn luyện $D^*_{tream} = \{(d_t, l_t\}, t \in [1, n], d_t \in D_{tream}, l_t \in [0, 4]$ (l_t được gọi là nhãn kiểu trôi, cho biết thời điểm t thuộc một vùng trôi kiểu nào ($l_t = 1, 2, 3, 4$) hay không ($l_t = 0$)).
- **Đầu ra**: Một mô hình phân lớp trôi khái niệm M cho luồng dữ liệu D_{tream} sau thời điểm n: áp dụng mô hình M vào thời điểm tiềm năng trôi khái niệm $t = n + k, k > 0$ thì xác định kiểu trôi khái niệm xảy ra.

Trong bối cảnh dữ liệu luồng, việc xây dựng tập dữ liệu huấn luyện cho bài toán phân lớp trôi khái niệm đặt ra một số thách thức đặc thù. Khác với dữ liệu tĩnh truyền thống, dữ liệu trong luồng là liên tục theo thời gian và không có sẵn nhãn phân loại trôi cho từng đoạn. Do đó, cần có một cách tiếp cận phù hợp để trích xuất các biểu hiện đặc trưng cho hiện tượng trôi và gán nhãn tương ứng.

Để trích xuất các biểu hiện đặc trưng, hướng tiếp cận phổ biến là sử dụng chiến lược chia luồng dữ liệu thành các đoạn nhỏ gọi là cửa sổ. Sau đó, từ các cặp cửa

sổ liên tiếp, các đặc trưng phản ánh sự thay đổi – như sự khác biệt về phân phối, sai số mô hình hoặc biểu diễn đặc trưng – được trích xuất. Những biểu hiện này đóng vai trò là đầu vào cho mô hình phân lớp [55], [56].

Để gán nhãn cho các mẫu dữ liệu huấn luyện, cách tiếp cận là sử dụng các bộ dữ liệu tổng hợp, trong đó các điểm trôi và kiểu trôi đã được chèn trước, từ đó có thể gán nhãn tương ứng cho từng mẫu: trôi đột ngột, trôi dần dần, trôi gia tăng hoặc không trôi. Tập dữ liệu thu được dưới dạng các đoạn dữ liệu có biểu hiện thay đổi và nhãn loại trôi được sử dụng để huấn luyện các thuật toán phân lớp. Việc này giúp xây dựng được một tập huấn luyện có nhãn rõ ràng, phục vụ cho các mô hình học có giám sát để phân lớp các kiểu trôi.

Trong các phương pháp phân lớp trôi, luận án lựa chọn mạng nguyên mẫu để huấn luyện cho mạng cách thức phân biệt các kiểu trôi khác nhau dựa trên phương pháp huấn luyện theo từng tập nhiệm vụ. Lý do lựa chọn mạng nguyên mẫu đã được nêu trong Mục 1.3.4.2. Dữ liệu sử dụng để huấn luyện trong phương pháp này được chia thành tập huấn luyện và tập kiểm tra, được mô tả chi tiết trong phần dưới đây.

3.1.1 Mạng nguyên mẫu học ít mẫu

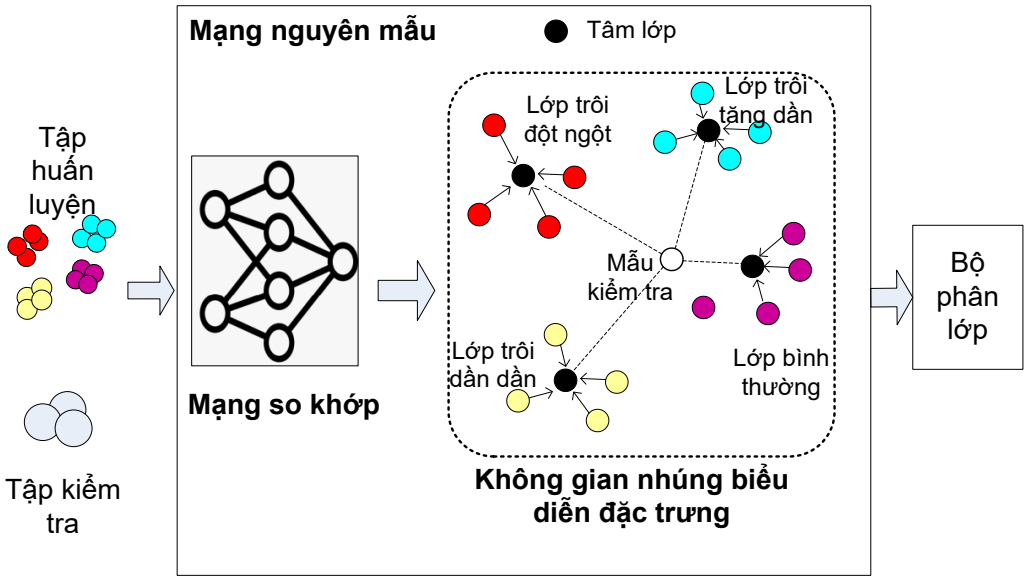
Trong bối cảnh phân lớp trôi khái niệm với số lượng mẫu có nhãn hạn chế, mạng nguyên mẫu là một phương pháp đơn giản nhưng hiệu quả, đặc biệt phù hợp với bài toán học từ ít mẫu [55]. Thay vì huấn luyện một bộ phân lớp truyền thống với nhiều tham số, mạng nguyên mẫu học một hàm ánh xạ từ không gian đầu vào sang một không gian nhúng sao cho các điểm thuộc cùng một lớp sẽ tập trung gần nhau trong không gian này, như minh họa tại hình 3.1.

Trong không gian nhúng, mỗi lớp được đại diện bởi một tâm lớp (prototype) – được tính bằng trung bình của các điểm có nhãn trong lớp đó. Khi cần phân loại một mẫu mới, mô hình chỉ cần ánh xạ mẫu này vào không gian nhúng, sau đó so sánh khoảng cách đến các tâm lớp để xác định nhãn tương ứng.

Quá trình huấn luyện mạng nguyên mẫu thường tuân theo chiến lược huấn luyện theo tập nhiệm vụ – tức mô phỏng quá trình học qua nhiều nhiệm vụ nhỏ (episodes). Mỗi tập dữ liệu để huấn luyện cho một nhiệm vụ nhỏ bao gồm hai phần:

- Tập huấn luyện: chứa một số ít mẫu có nhãn cho mỗi lớp, dùng để xây dựng tâm lớp.
- Tập kiểm tra: chứa các mẫu cần dự đoán nhãn, dùng để đánh giá khả năng phân loại của mô hình dựa trên các tâm lớp đã tính.

Trong từng nhiệm vụ nhỏ, mô hình học cách ánh xạ sao cho các mẫu cùng lớp nằm gần tâm lớp tương ứng, trong khi các mẫu khác lớp bị đẩy ra xa. Nhờ đó, mô hình dần học được một không gian biểu diễn ổn định và có tính khái quát, giúp phân loại chính xác các biểu hiện trôi khái niệm mới dù chỉ có rất ít dữ liệu huấn luyện.



Hình 3.1 Minh họa mạng nguyên mẫu [39]

Trong luận án này, mạng nguyên mẫu được sử dụng như một mô hình cơ sở để xử lý bài toán phân lớp kiểu trôi. Mỗi biểu hiện trôi được biểu diễn dưới dạng một vector đặc trưng, sau đó được ánh xạ vào không gian nhúng để so sánh với các tâm lớp. Phần tiếp sau đây mô tả cách thức xây dựng bộ phân lớp trôi khái niệm.

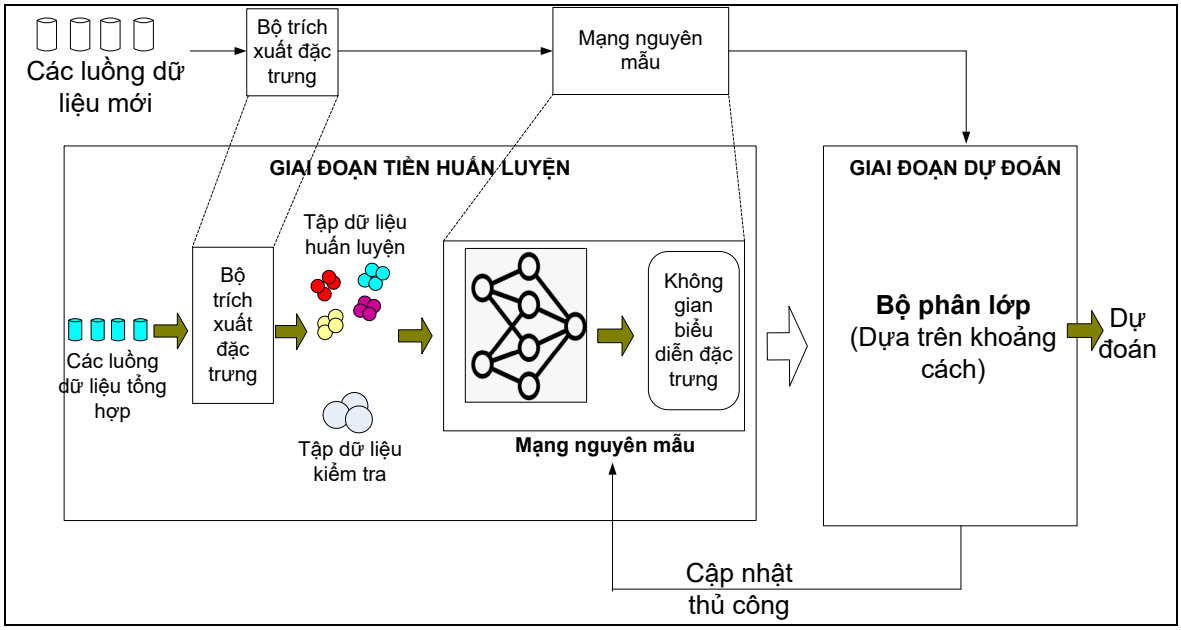
3.1.2 Mô hình phân lớp trôi khái niệm Meta-ADD

Mô hình Meta-ADD [55] đề xuất một quy trình phát hiện và phân lớp trôi khái niệm trong luồng dữ liệu, được minh họa trong Hình 3.2.

Trong giai đoạn *tiền huấn luyện*, đầu vào là các luồng dữ liệu tổng hợp có chứa các kiểu trôi khái niệm khác nhau. Dữ liệu này được đưa qua bộ trích xuất đặc trưng để tạo ra các vector đặc trưng tương ứng với từng kiểu trôi. Sau đó, các

vector đặc trưng được phân tách thành hai tập: tập huấn luyện (còn được gọi là tập hỗ trợ) và tập kiểm tra (còn được gọi là tập truy vấn). Các tập dữ liệu này được sử dụng để huấn luyện mạng nguyên mẫu nhằm tối ưu các tham số của mạng. Mạng nguyên mẫu bao gồm hai thành phần chính: một mạng so khớp ký hiệu là f_ϕ, và một không gian biểu diễn đặc trưng. Việc huấn luyện mạng so khớp được thực hiện theo chiến lược huấn luyện theo từng tập nhiệm vụ (episodic training).

Trong *giai đoạn dự đoán*, một luồng dữ liệu cần phân loại sẽ được đưa vào bộ trích xuất đặc trưng và mạng nguyên mẫu (đã được tiền huấn luyện) để thu được vector nhúng trong không gian biểu diễn – đại diện cho mẫu kiểm tra. Vector này sau đó được đưa vào bộ phân lớp để dự đoán kiểu trôi khái niệm. Bộ phân lớp hoạt động bằng cách tính khoảng cách giữa mẫu kiểm tra và các tâm lớp đã được xác định trong không gian biểu diễn. Nhãn của kiểu trôi sẽ được gán cho lớp có khoảng cách ngắn nhất đến mẫu kiểm tra.



Hình 3.2. Quy trình phân lớp trôi chi tiết của Meta-ADD [55]

Chi tiết hơn, trong giai đoạn tiền huấn luyện, bộ trích xuất đặc trưng thực hiện trích xuất các đặc trưng của luồng dữ liệu liên quan đến các kiểu trôi. Vì khi trôi xảy ra, tỉ lệ lỗi dự đoán của mô hình dự đoán (ví dụ: Naïve Bayes, Gaussian, Decision Tree, SVR…) sẽ tăng lên, do đó sự thay đổi của tỷ lệ lỗi rất quan trọng để xác định kiểu trôi nào đã xảy ra:

- Trôi đột ngột: tỷ lệ lỗi sẽ giảm mạnh trong một thời gian ngắn.

- Trôi dần dần: tỷ lệ lỗi sẽ dao động trong một khoảng thời gian dài.
- Trôi tăng dần: tỷ lệ lỗi sẽ tăng dần trong một khoảng thời gian dài.
- Bình thường: tỷ lệ lỗi sẽ duy trì ổn định.

Dựa vào những đặc điểm trên, Meta-ADD sử dụng khoảng cách giữa tỷ lệ lỗi trung bình giữa hai cửa sổ rời rạc liên tiếp là đặc trưng cần trích xuất. Cụ thể, đối với mỗi kiểu trôi, sẽ tạo ra các luồng dữ liệu tổng hợp, sau đó sử dụng mô hình dự đoán Naïve Bayes để thu được tỉ lệ lỗi e_t của mô hình tại mỗi thời điểm t. Trên một luồng, chia thành nhiều cửa sổ, mỗi cửa sổ W_i với độ dài n (tức mỗi W_i chứa n mẫu e_t), vì vậy tỷ lệ lỗi trung bình $\hat{e}_t$ của W_i là:

$$\hat{e}_t = \frac{\sum_{t=1}^{n} e_t}{n} \tag{3.1}$$

Khoảng cách giữa tỷ lệ lỗi trung bình của hai cửa sổ liên tiếp là:

$$gap_i = \hat{e}_{t+1} - \hat{e}_t \tag{3.2}$$

Giả sử mỗi luồng dữ liệu có l cửa sổ thì số gap là $l-1$. Do vậy một mẫu huấn luyện i là:

$$G_i: \{\widehat{\boldsymbol{X}} \times \hat{y}\} \in R^{l-1} \tag{3.3}$$

Trong đó: $\widehat{\boldsymbol{X}} = \{gap_1, \dots, gap_{l-1}\}$ và $\hat{y} \in 1,..,k..,4$ là nhãn tương ứng 4 loại trôi.

Nếu có N luồng, mỗi luồng có l cửa sổ, mỗi cửa sổ có n mẫu, vậy tổng mỗi luồng có $m = l * n$ mẫu dữ liệu thì ta có tập dữ liệu $g: \{\boldsymbol{X} \times y\}^{N \times m}$ sẽ được ánh xạ vào tập đặc trưng là $G: \widehat{\boldsymbol{X}} \times \hat{y} \in R^{N \times l}$. Có thể thấy với đặc trưng G này sẽ độc lập với dữ liệu thực tế của luồng sinh ra.

Chia tập G thành tập huấn luyện và tập kiểm tra. Các tập này sau đó được sử dụng để để huấn luyện mạng nguyên mẫu theo phương pháp *huấn luyện theo tập nhiệm vụ*, như sau:

- Bước 1 - Lấy dữ liệu: Trong mỗi vòng huấn luyện, một số ít mẫu dữ liệu được chọn làm tập huấn luyện, và một số lượng lớn hơn được chọn làm tập kiểm tra cho từng kiểu trôi khái niệm.
- Bước 2 – Ánh xạ sang không gian nhúng: sử dụng một mạng so khớp ánh xạ cả tập huấn luyện và tập kiểm tra vào không gian nhúng thông qua hàm f_ϕ:

$$f_\phi: R^N \rightarrow R^M \quad (3.4)$$

với ϕ là tham số được học, ánh xạ $\widehat{\boldsymbol{X}}_i$ vào không gian nhúng $f_\phi(\widehat{\boldsymbol{X}}_i)$,

- Bước 3 – Tính toán các tâm lớp $\boldsymbol{c}_k$ của mỗi lớp k bằng cách lấy trung bình các vector nhúng của *tập huấn luyện* của lớp *k*:

$$\boldsymbol{c}_k = \frac{1}{|S_k|}\sum_{(\widehat{\boldsymbol{x}}_i, y_i)\in R^{N\times l}} f_\phi(\widehat{\boldsymbol{X}}_i) \quad (3.5)$$

S_k là tập dữ liệu luồng với nhãn k

- Bước 4 – Phân lớp: các mẫu $\widehat{\boldsymbol{x}}$ trong tập kiểm tra được phân lớp bằng cách tính toán khoảng cách từ mẫu $\widehat{\boldsymbol{x}}$ tới tâm của từng lớp trong không gian nhúng. Mẫu $\widehat{\boldsymbol{x}}$ sẽ thuộc về lớp có khoảng cách gần nhất.

$$p_\theta(\hat{y} = k|\widehat{\boldsymbol{x}}) = \frac{\exp\left(-d(f_\phi(\widehat{\boldsymbol{x}}), \boldsymbol{c}_k)\right)}{sum_{k'}\exp\left(-d(f_\phi(\widehat{\boldsymbol{x}}), \boldsymbol{c}_k)\right)} \quad (3.6)$$

- Bước 5 – Tính toán hàm mất mát: tính bằng xác suất log âm (negative log-likelihood) của nhãn thực sự của các mẫu trong tập kiểm tra, dựa trên xác suất của lớp được dự đoán theo khoảng cách đến các tâm lớp.

$$J(\phi) = -\log p_\phi(y = k|\widehat{\boldsymbol{x}}) \quad (3.7)$$

- Bước 6 - Tối ưu hóa: Các tham số của mạng nhúng f_ϕ được cập nhật bằng thuật toán hạ gradient ngẫu nhiên (SGD) để giảm thiểu lỗi phân lớp.

Sau giai đoạn tiền huấn luyện là giai đoạn dự đoán kiểu trôi trên các luồng dữ liệu khác nhau. Vì đặc điểm về trôi khái niệm đối với mỗi loại luồng cũng khác nhau, vì vậy cần cập nhật mạng nhúng để thích nghi với các đặc điểm trôi của dạng luồng cụ thể. Để cập nhật mạng nhúng, khi thu thập một mẫu $W_i = \{e_{t-n}, \dots, e_i\}$ trong đó e_t là tỷ lệ lỗi thu được tại thời điểm t, cần biết kiểu trôi nào *thực sự* đã xảy ra trên mẫu W_i này (nhãn thực tế). Tuy nhiên, vì các luồng dữ liệu rất đa dạng nên rất khó biết được thông nhãn thực tế này, Meta-ADD sử dụng cách gán nhãn thủ công thông qua phân tích kiểu trôi của mẫu W_i trên. Tuy nhiên làm thủ công là rất tốn kém nên trong Meta-ADD cách thức làm là chỉ gán nhãn thủ công cho các mẫu có giá trị độ bất định cao (high entropy). Độ bất định ở đây thể hiện mức độ không chắc chắn của việc phân lớp. Độ bất định càng lớn thì việc phân lớp càng không chắc chắn, và do đó phải gán nhãn lớp thủ công cho mẫu này. Một bộ phân lớp biểu thị xác suất $P(c)$ để mẫu W_i thuộc lớp c với $\sum_{c\in C} P(c) = 1$, độ bất định của việc phân lớp này được tính như sau:

$$H(C) = -\sum_{c \in C} P(c) \log p\,(C) \quad (3.8)$$

3.1.3 Nhận xét

- **Nhận xét 1**: Trong Meta-ADD, ở bộ trích xuất đặc trưng, H. Yu và cộng sự áp dụng duy nhất chiến lược cửa sổ rời rạc. Tuy nhiên mỗi kiểu trôi có đặc trưng khác nhau nên nếu chỉ sử dụng một chiến lược cửa sổ rời rạc sẽ không trích xuất được các đặc trưng điển hình của từng kiểu. Do vậy đối với mỗi kiểu trôi cần áp dụng một chiến lược cửa sổ phù hợp. Từ nhận xét này, luận án đề xuất mô hình **VAR-WIND** trong đó bộ trích xuất đặc trưng sẽ áp dụng các chiến lược của sổ linh hoạt đối với mỗi kiểu trôi, được trình bày chi tiết trong Mục 3.2.

- Nhận xét 2: Trong giai đoạn dự đoán của Meta-ADD cần xác định kiểu trôi trên các mẫu dữ liệu mới, chưa từng thấy trước đây. Để đảm bảo tính tổng quát, mô hình này đưa ra phương án cập nhật mạng nguyên mẫu theo cách thủ công. Điều này rất tốn kém. Do vậy luận án đề xuất phương án khác, đó là *thêm giai đoạn tinh chỉnh sau giai đoạn tiền huấn luyện*, để có mô hình cải tiến **MetaLDD-Finetune** với kỹ thuật huấn luyện đảm bảo tính tổng quát hóa trên các luồng dữ liệu mới**,** được trình bày chi tiết trong Mục 3.3.

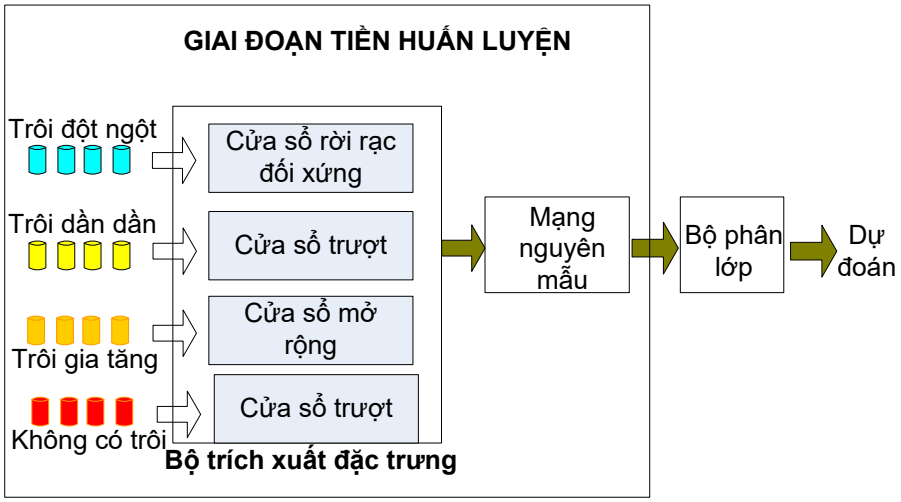
3.2. Mô hình phân lớp trôi khái niệm VAR-WIND đề xuất

3.2.1 Mô hình VAR-WIND đề xuất

Trong giai đoạn tiền huấn luyện, ở bước trích xuất đặc trưng, mô hình Meta-ADD chỉ sử dụng duy nhất một chiến lược cửa sổ rời rạc áp dụng trên luồng các tỉ lệ lỗi e_t để tính toán tỉ lệ lỗi trung bình $\hat{e}_t$ trong từng cửa sổ, sau đó tính được khoảng cách *gap* giữa các cửa sổ kề nhau, chính là đặc trưng cần trích xuất. Tuy nhiên mỗi kiểu trôi có đặc điểm khác nhau, trong khi chỉ sử dụng một chiến lược cửa sổ để trích xuất đặc trưng, dẫn đến độ chính xác phân lớp kiểu trôi còn hạn chế. Vấn đề này đã được H. Yu và cộng sự đề cập trong phần hạn chế của Meta-ADD.

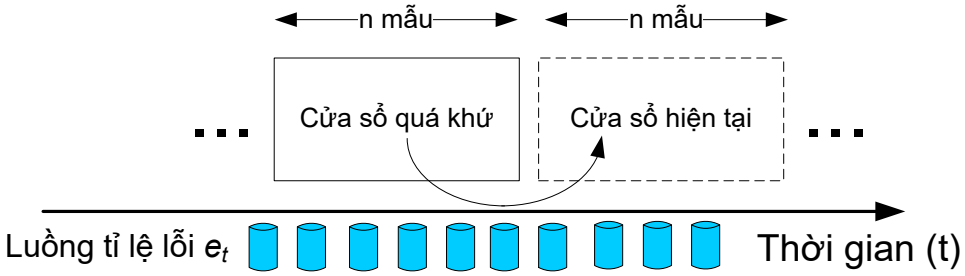
Vì vậy, trong phần này, luận án đề xuất phương pháp cải tiến như trong Hình 3.3. Lấy cảm hứng từ các nghiên cứu sử dụng cửa sổ linh hoạt trong việc lấy mẫu và trích xuất đặc trưng luồng dữ liệu của trong [44] và Mansalis [30], luận án đề xuất phương pháp cải tiến bộ trích xuất đặc trưng của Meta-ADD bằng cách sử dụng các chiến lược cửa sổ khác nhau cho mỗi kiểu trôi: *cửa sổ rời rạc đối xứng cho luồng có trôi đột ngột, cửa sổ trượt cho luồng có trôi dần dần, cửa sổ mở rộng*

cho luồng có trôi gia tăng, cửa sổ trượt cho luồng không có trôi. Mỗi chiến lược này có sự phù hợp với đặc điểm của từng kiểu trôi, được phân tích trong phần dưới đây.



Hình 3.3 Mô hình VAR-WIND cải tiến bộ trích xuất đặc trưng

Trong Meta-ADD, giai đoạn trích xuất đặc trưng của chiến lược cửa sổ rời rạc được sử dụng, như minh họa trong Hình 3.4. Luồng dữ liệu được chia thành các cửa sổ liên tiếp có kích thước cố định. Hai cửa sổ được chọn từ hai thời điểm khác nhau: một cửa sổ hiện tại và một cửa sổ quá khứ. Các cửa sổ này không chồng lấp nhau và mỗi cửa sổ được phân tích độc lập để tính toán tỷ lệ lỗi trung bình $\hat{e}_t$, đại diện cho phân phối dữ liệu trong từng cửa sổ.



Hình 3.4 Cửa sổ rời rạc

Phương pháp cửa sổ rời rạc tỏ ra hiệu quả trong việc phát hiện những thay đổi rõ rệt khi so sánh giữa hai khoảng thời gian khác nhau, chẳng hạn như trôi đột ngột hoặc sự thay đổi mạnh trong phân phối dữ liệu khi quan sát ở hai giai đoạn tách biệt [44]. Tuy nhiên, trôi dần dần và trôi gia tăng lại diễn ra dưới dạng sự chuyển tiếp liên tục hoặc sự thay đổi "nhỏ" trong phân phối dữ liệu theo thời gian. Do chiến lược cửa sổ rời rạc không có phần chồng lấp giữa các cửa sổ liên tiếp,

các mô hình chuyển tiếp của trôi giữa các cửa sổ có thể bị mất hoàn toàn. Điều này dẫn đến việc phân đoạn dữ liệu một cách không liên tục, gây khó khăn trong việc nhận diện các dạng trôi này và làm tăng nguy cơ phân lớp sai.

Các chiến lược cửa sổ khác nhau—cửa sổ rời rạc, cửa sổ trượt và cửa sổ mở rộng—có khả năng nắm bắt các khía cạnh khác nhau của sự thay đổi trong phân phối dữ liệu [30]. Việc lựa chọn chiến lược cửa sổ phù hợp với từng kiểu trôi giúp tạo dữ liệu huấn luyện phản ánh chính xác hơn đặc trưng của từng kiểu trôi, từ đó nâng cao hiệu năng của mô hình phân lớp trôi khái niệm.

3.2.1.1 Cửa sổ rời rạc đối xứng cho luồng trôi đột ngột

Trôi đột ngột được đặc trưng bởi sự thay đổi đột ngột và rõ rệt trong phân phối dữ liệu.

Trong DDM, J. Gama và cộng sự [18] giới thiệu phương pháp phát hiện trôi sử dụng hai cửa sổ rời rạc để phát hiện các thay đổi đột ngột bằng cách so sánh tỷ lệ lỗi giữa các cửa sổ liên tiếp.

Luận án cũng áp dụng chiến lược cửa sổ rời rạc để so sánh sự thay đổi của tỉ lệ lỗi trung bình giữa hai cửa sổ liên tiếp. Mỗi cửa sổ chứa một tập mẫu có độ dài cố định n và không có sự chồng lấp, giúp đảm bảo rằng các đặc trưng được trích xuất từ hai pha dữ liệu rõ ràng trước và sau trôi.

Tuy nhiên, trong thực tế, điểm trôi không phải lúc nào cũng nằm đúng ranh giới giữa hai cửa sổ rời rạc để từ đó so sánh nắm bắt được đã có trôi xảy ra một cách kịp thời. Trong một số trường hợp, điểm trôi có thể rơi vào chính giữa một cửa sổ, dẫn đến hiện tượng trộn lẫn giữa hai khái niệm trong cùng một cửa sổ, khi một cửa sổ chứa cả dữ liệu thuộc khái niệm cũ lẫn khái niệm mới. Điều này làm giảm độ phân biệt của đặc trưng, do giá trị trung bình của cửa sổ trộn lẫn có thể không khác biệt nhiều so với cửa sổ trước đó.

Để khắc phục hạn chế này, luận án sử dụng thêm biến thể *cửa sổ rời rạc đối xứng*, trong đó lựa chọn hai cửa sổ nằm hoàn toàn trước và sau điểm trôi, tránh cửa sổ có khả năng chứa trôi. Có được điều này là trong quá trình sinh dữ liệu tổng hợp từ các bộ sinh đã đề cập ở Mục 1.5.1.2, khi thay đổi từ hàm sinh này (khái niệm cũ) sang hàm sinh khác (khái niệm mới), sẽ đánh dấu được điểm trôi. Do vậy giả sử trên một luồng dữ liệu e_t, một cửa sổ W_i được gắn nhãn có điểm trôi nằm trong đó, sẽ chỉ trích xuất đặc trưng từ cặp cửa sổ nằm trước và nằm sau W_i,

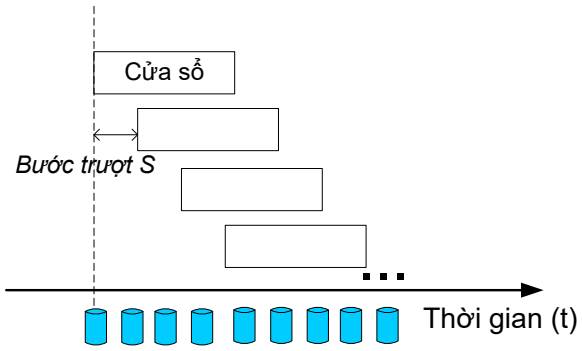
để đảm bảo mỗi cửa thuộc một phân phối dữ liệu thuần nhất. Cách tiếp cận này giúp tăng độ khác biệt giữa hai phân phối, từ đó mô hình có thể dễ dàng nhận diện đặc trưng thay đổi đột ngột — đặc trưng vốn là biểu hiện chính của loại trôi này.

3.2.1.2 Cửa sổ trượt cho luồng trôi dần dần

Trôi dần dần được đặc trưng bởi sự thay đổi chậm và liên tục trong phân phối dữ liệu theo khoảng thời gian đủ dài.

G. Widmer và M. Kubat [51] đã giới thiệu phương pháp cửa sổ trượt để xử lý trôi dần dần, theo đó chiến lược này duy trì một cửa sổ có độ dài cố định, liên tục dịch chuyển về phía trước khi có dữ liệu mới. Chiến lược này có ưu điểm là luôn cập nhật liên tục phân phối dữ liệu, giúp phát hiện nhanh các xu hướng mới và thích hợp cho trôi dần dần.

Như minh họa tại Hình 3.5, thay vì chỉ sử dụng các cặp cửa sổ rời rạc như trong chiến lược ban đầu, các cửa sổ có độ dài cố định n và trượt đi với bước trượt S (xác định mức độ chồng lấp giữa các cửa sổ liên tiếp) từ luồng các giá trị lỗi e_t.



Hình 3.5 Cửa sổ trượt

Mỗi cửa sổ trượt W_i tính tỉ lệ lỗi trung bình:

$$\hat{e}_i = \frac{1}{n}\sum_{j=1}^{n} e_{(i-1)S+j} \tag{3.9}$$

Tiếp theo, tính khoảng cách giữa hai cửa sổ liên tiếp:

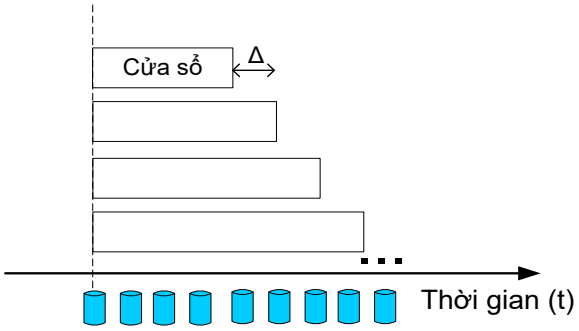
$$gap_i = \hat{e}_{i+1} - \hat{e}_i \tag{3.10}$$

Từ đó, thu được một vector đặc trưng là chuỗi các giá trị *gap* thu được.

Chiến lược cửa sổ trượt giúp tăng mật độ thông tin bằng cách tạo ra các cửa sổ có vùng chồng lấn, từ đó mô hình học có thể nhận diện rõ hơn các biểu hiện thay đổi dần dần trong dữ liệu, vốn thường không rõ rệt tại một điểm cụ thể.

3.2.1.3 Cửa sổ mở rộng cho luồng trôi gia tăng

Trôi gia tăng thường diễn ra dưới dạng sự thay đổi tiến triển theo thời gian. Chiến lược cửa sổ mở rộng giúp nắm bắt xu hướng này bằng cách xây dựng một lịch sử tích lũy, hỗ trợ trong việc phát hiện các thay đổi mang tính tiến triển. Baena-García và cộng sự [6] thảo luận về phương pháp phát hiện trôi sớm (EDDM), sử dụng cách tiếp cận cửa sổ mở rộng để tăng độ nhạy đối với trôi gia tăng.



Hình 3.6 Cửa sổ mở rộng

Bằng cách phân tích các xu hướng gia tăng trong dữ liệu theo thời gian, cửa sổ mở rộng giúp phát hiện hiệu quả các kiểu trôi này. Hình 3.6 minh họa cửa sổ mở rộng với bước tăng độ dài Δ.

Cụ thể, cửa sổ được mở rộng dần từ một độ dài ban đầu nhỏ nhất L_{min} sau đó tăng dần độ dài thêm một lượng Δ cho đến độ dài tối đa n. Với mỗi bước mở rộng tính tỉ lệ lỗi trung bình:

$$\hat{e}_L = \frac{1}{L}\sum_{i=1}^{L} e_i \tag{3.11}$$

Sau đó tính khoảng cách giữa các mức mở rộng liên tiếp:

$$gap_i = \hat{e}_{L_i+\Delta} - \hat{e}_{L_i} \tag{3.12}$$

với $L_i \in \{L_{min},\ L_{min} + \Delta,\ \ldots, n - \Delta\}$

Cửa sổ mở rộng giúp mô hình nắm bắt được cách tỷ lệ lỗi tích lũy thay đổi khi có thêm dữ liệu, phản ánh tốt quá trình chuyển tiếp chậm rãi trong trôi gia tăng.

3.2.1.4 Cửa sổ trượt cho luồng không có trôi

Đối với luồng dữ liệu không có trôi, đặc điểm nổi bật là sự ổn định trong phân phối dữ liệu và hiệu năng của mô hình theo thời gian. Để phản ánh đặc trưng này trong quá trình trích xuất đặc trưng, luận án sử dụng chiến lược cửa sổ trượt với kích thước cố định và bước trượt đều, tương tự như với luồng có trôi dần dần.

Cụ thể, thay vì trích xuất một chuỗi đặc trưng duy nhất từ một cửa sổ rời rạc, việc áp dụng cửa sổ trượt cho phép sinh ra nhiều đoạn đặc trưng từ chuỗi *gap*, trong đó mỗi đoạn phản ánh mức biến động lỗi trong một khoảng thời gian ngắn. Trong trường hợp không có trôi, luồng các giá trị lỗi trung bình giữa các cửa sổ trượt kế tiếp thường có độ biến thiên rất thấp, dẫn đến dãy đặc trưng gần như ổn định hoặc dao động ngẫu nhiên nhỏ.

Chiến lược này vừa giúp tăng số lượng mẫu huấn luyện, vừa đảm bảo rằng mô hình học được đặc trưng của trạng thái ổn định, làm cơ sở để phân biệt rõ ràng với các luồng có trôi (vốn thể hiện sự thay đổi trong đặc trưng). Ngoài ra, việc duy trì cùng một chiến lược cửa sổ với trôi dần cũng giúp đảm bảo tính công bằng về mặt kỹ thuật, tránh việc mô hình phân biệt các loại trôi dựa vào khác biệt do cấu hình cửa sổ thay vì do bản chất dữ liệu.

3.2.2. Thực nghiệm và đánh giá

Luận án kế thừa phương thức thực nghiệm mà Meta-ADD đã thực hiện, được mô tả chi tiết dưới đây.

3.2.2.1 Mục tiêu thực nghiệm

Mục tiêu thực nghiệm nhằm so sánh hiệu năng trong *giai đoạn tiền huấn luyện* giữa mô hình Meta-ADD và mô hình VAR-WIND trên từng bộ dữ liệu thực nghiệm.

Để đảm bảo tính tái lập và tính công bằng trong đánh giá, luận án sử dụng lại bộ dữ liệu mô phỏng các kiểu trôi khái niệm cũng như mã nguồn thực nghiệm do mô hình Type-LDD [56] công bố[10].

3.2.2.2. Xây dựng bộ dữ liệu thực nghiệm

Từ các bộ dữ liệu tổng hợp điển hình (Mục 1.5), luận án tạo ra ba bộ dữ liệu thực nghiệm **Dataset_50-50-4800**, **Dataset_200-25-3800**, **Dataset_50-15-4800** với các kiểu trôi khác nhau (đột ngột, dần dần, gia tăng, bình thường); ở đây, kiểu "bình thường" là luồng dữ liệu không có trôi. Các thực nghiệm ở Chương 3 và Chương 4 sử dụng một phần hay toàn bộ ba bộ dữ liệu được tạo ra này.

Bộ thứ nhất: Dataset_50-50-4800:

[10] https://github.com/liaub/Type-LDD/tree/main

Mỗi kiểu trôi có 4800 luồng, mỗi luồng dữ liệu chứa 2550 mẫu, được chia đều thành 51 cửa sổ, mỗi cửa sổ chứa 50 mẫu.

Trong mỗi luồng có trôi (đột ngột, dần dần, gia tăng), một cửa sổ duy nhất được gắn nhãn 1 (tức là có trôi tại cửa sổ này). Việc gắn nhãn trôi này là do trong quá trình sinh dữ liệu, tại thời điểm thay đổi hàm sinh khái niệm ban đầu sang hàm sinh khái niệm mới, sẽ gán nhãn trôi tại thời điểm đó.

Đối với luồng bình thường tức không có trôi thì các luồng sẽ không đánh dấu điểm trôi.

Phương pháp sinh trôi khái niệm

- Trôi đột ngột: Sử dụng bộ sinh Agrawal và SEA trong thư viện Scikit-multiflow, sinh luồng trôi bằng hàm *ConceptDriftStream* với quãng thay đổi đột ngột (tham số *width=1)*. Hai hàm sinh khác nhau được sử dụng trong các bộ này.
- Trôi dần dần: Sinh luồng trôi với quãng thay đổi dài (tham số *width=1000)*, mô phỏng chuyển tiếp mượt giữa hai khái niệm trong bộ sinh SEA với các giá trị ban đầu (seed) ngẫu nhiên và tỷ lệ nhiễu 28%.
- Trôi gia tăng: Sử dụng bộ sinh Hyperplane với các mức thay đổi biên độ và độ bất định khác nhau, nhằm tạo sự dịch chuyển ranh giới gia tăng dần.
- Không trôi: Luồng dữ liệu được sinh ra hoàn toàn từ một khái niệm ổn định, không có trôi, sử dụng một trong các bộ sinh: Agrawal, SEA, Hyperplane hoặc LED.

Cấu trúc dữ liệu sinh ra

Như đã đề cập, sau khi sinh các luồng và phân chia thành các cửa sổ, với mỗi cửa sổ trên luồng dữ liệu, mô hình học trực tuyến Naive Bayes được huấn luyện và đánh giá trên từng mẫu nằm trong cửa sổ để thu được tỉ lệ lỗi dự đoán e_t tại mỗi thời điểm t, sau đó tính tỷ lệ lỗi trung bình $\hat{e}_t$ trên cửa sổ đó.

- Tất cả dữ liệu được ghi ra các tệp *.csv*, mỗi tệp lưu kết quả về tỉ lệ lỗi của một luồng dữ liệu. Tệp .csv gồm 51 hàng (ứng với 51 cửa sổ), mỗi hàng chứa thông tin về số thứ tự cửa sổ, tỉ lệ lỗi trung bình trên cửa sổ đó và nhãn trôi tương ứng.

Bảng 3.1 Cấu trúc một luồng lưu thành một tệp .csv bộ 50-50-4800

Số thứ tự cửa sổ	Tỉ lệ lỗi trung bình ($\hat{e}_t$)	Nhãn trôi
0	0.666	0
1	0.6	0
2	0.8	**1**
3	0.533	0
4	0.533	0
…	…	…
50	0.666	0

- Đối với luồng không có trôi, toàn bộ cửa sổ chỉ ghi lại tỷ lệ lỗi của mô hình, không có cột nhãn.

Bảng 3.2 Cấu trúc một tệp .csv tương ứng với 1 luồng dữ liệu không có trôi

Số thứ tự cửa sổ	Tỉ lệ lỗi trung bình ($\hat{e}_t$)
0	0.7
1	0.76
2	0.88
3	0.86
4	0.84
5	0.8
6	0.84
7	0.9
8	0.9
…	--
50	0.88

Bộ dữ liệu thứ hai: Dataset_200-25-3800

Mỗi kiểu trôi gồm 3800 luồng, mỗi luồng dữ liệu bao gồm 5025 mẫu, được chia đều thành 201 cửa sổ thời gian, mỗi cửa sổ chứa 25 mẫu. Với các luồng có trôi khái niệm, mỗi luồng được gán duy nhất một cửa sổ có nhãn trôi, các cửa sổ còn lại được gán nhãn không trôi (0). Đối với luồng không có trôi, toàn bộ cửa sổ chỉ ghi lại tỷ lệ lỗi của mô hình, không có cột nhãn.

Phương pháp sinh trôi khái niệm: Giống như cách thực hiện trong bộ Dataset_50-50-4800.

Cấu trúc dữ liệu sinh ra

Lưu trong các tệp *.csv* với cấu trúc như trong Bảng 1.1 và Bảng 1.2, chỉ khác là mỗi tệp có 201 hàng, tương ứng với 201 cửa sổ, mỗi cửa sổ chứa 25 mẫu, mỗi hàng cũng lưu thông tin về số thứ tự cửa sổ, tỉ lệ lỗi trung bình (của 25 mẫu) và nhãn trôi của từng cửa sổ.

Bộ dữ liệu thứ ba: Dataset_50-15-4800

Mỗi kiểu trôi bao gồm 4.800 luồng dữ liệu riêng biệt. Mỗi luồng dữ liệu được sinh ra có tổng cộng 765 mẫu, chia thành 51 cửa sổ thời gian đều nhau, mỗi cửa sổ gồm đúng 15 mẫu. Đối với các luồng có trôi khái niệm, chỉ có một cửa sổ được gắn nhãn là có trôi (với giá trị 1), tất cả các cửa sổ còn lại đều được gắn nhãn 0.

Cấu trúc dữ liệu sinh ra

Lưu trong các tệp *.csv* với cấu trúc như trong Bảng 1.1 và Bảng 1.2, chỉ khác là mỗi tệp có 51 hàng, tương ứng với 51 cửa sổ, mỗi hàng cũng lưu thông tin về số thứ tự cửa sổ, tỉ lệ lỗi trung bình (của 15 mẫu) và nhãn trôi của từng cửa sổ.

3.2.2.3 Phần cứng, phần mềm và bộ dữ liệu

Phần cứng sử dụng: máy tính Dell PowerEdge T40, CPU Intel Xeon E-2224G 3.5 Ghz, 32GB RAM, 2TB HDD.

Phần mềm sử dụng: hệ điều hành Windows 10, Pycharm IDE Community.

Bộ dữ liệu: **Dataset_50-50-4800** và **Dataset_200-25-3800.** Mỗi bộ dữ liệu nói trên được chia thành hai phần: **80%** dành cho tập huấn luyện và **20%** dành cho tập kiểm tra. Trong tập kiểm tra, các luồng trôi thuộc 4 kiểu được *trộn lẫn*. Việc trộn lẫn được thực hiện nhằm chứng minh tính ổn định của mô hình đề xuất khi gặp nhiều kiểu trôi xen kẽ hoặc dữ liệu thay đổi liên tục.

3.2.2.4 Triển khai quá trình thực nghiệm

- **Cấu hình và chiến lược cửa số cho bộ trích xuất đặc trưng**

Bảng 3.3 chỉ dẫn các tham số cho Bộ trích xuất đặc trưng sử dụng đối với hai bộ dữ liệu **Dataset_50-50-4800** và **Dataset_200-25-3800**. Lựa chọn kích thước cửa sổ W=25 cho bộ dữ liệu **Dataset_200-25-3800** vì nó sinh ra 3800 luồng, mỗi luồng 201 cửa sổ, mỗi cửa sổ độ dài 25.

Bảng 3.3. Bảng tổng hợp trích xuất đặc trưng cho hai bộ dữ liệu

Loại luồng	Chiến lược cửa sổ	Tham số chính	
		Dataset_50-50-4800	Dataset_200-25-3800
Trôi đột ngột	Rời rạc đối xứng	W = 50	W = 25
Trôi dần dần	Trượt	W = 50, S = 5	W = 25, S = 5
Trôi gia tăng	Mở rộng	Lmin = 25, Lmax=50, Δ = 5	Lmin = 5, Lmax =25, Δ = 5
Bình thường	Trượt	W = 50, S = 5	W = 25, S = 5

Luồng trôi đột ngột – Cửa sổ rời rạc đối xứng

Trên mỗi luồng chứa trôi đột ngột (tương ứng với một file .csv), vị trí điểm trôi đã biết trước. Áp dụng cửa sổ rời rạc đối xứng có độ dài $W = 50$, một cửa sổ nằm trước và một cửa sổ sau điểm trôi. Lý do chọn kích thước 50 là bởi vì đối với bộ dữ liệu Dataset_50-50-4800, trong quá trình sinh sẽ sinh ra 4800 luồng dữ liệu, mỗi luồng có 51 cửa sổ và mỗi cửa sổ có độ dài 50.

Luồng trôi dần – Cửa sổ trượt

Với mỗi luồng chứa trôi dần dần, sử dụng cửa sổ trượt có độ dài $W = 50$ và bước trượt $S = 5$ để tạo ra chuỗi cửa sổ liên tiếp. Chọn bước trượt nhỏ S=5 để đảm bảo không bỏ sót các mẫu phản ánh đặc trưng. Mỗi lần trượt như vậy sẽ tạo ra một cửa sổ tương ứng với một vector đặc trưng.

Luồng trôi gia tăng – Cửa sổ mở rộng

Với mỗi luồng trôi tăng dần, sử dụng cặp cửa sổ mở rộng liên tiếp với độ dài tăng dần từ $L_{min} = 25$ đến $L_{max} = 50$ với bước nhảy Δ = 5. Chọn bước nhảy Δ để

đảm bảo không bỏ sót các mẫu phản ánh đặc trưng Mỗi lần mở rộng sẽ sinh ra một vector đặc trưng phản ánh sự tích lũy.

Luồng không trôi – Cửa sổ trượt

Áp dụng chiến lược trượt giống trôi dần, sử dụng cửa sổ $W = 50$, bước trượt $S = 5$, nhằm sinh ra nhiều mẫu trong điều kiện phân bố ổn định.

- **Chiến lược huấn luyện và kiểm tra**

Sau khi thiết lập xong cấu hình trích xuất đặc trưng, áp dụng chiến lược huấn luyện theo từng tập nhiệm vụ, theo cách thức mà mô hình Meta-ADD đã thực hiện. Cụ thể:

- Giai đoạn huấn luyện: Với mỗi kiểu trôi khái niệm, 5 mẫu dữ liệu hỗ trợ được chọn ngẫu nhiên để tính tâm lớp, 15 mẫu dữ liệu truy vấn được sử dụng để đánh giá mô hình.
- Giai đoạn kiểm tra: 20 mẫu dữ liệu hỗ trợ được chọn để tạo tâm lớp.

- **Cấu trúc mạng so khớp**

  - Mạng so khớp FAN (Fully Attention Network) được sử dụng với hàm kích hoạt ReLU tại các tầng ẩn. Lý do sử dụng mạng FAN được giải thích trong Meta-ADD và TypeLDD là vì mạng FAN được thiết kế dựa trên các lớp tích chập kết hợp với cơ chế chú ý, giúp tập trung vào các đặc trưng quan trọng trong dữ liệu đầu vào, thay vì xử lý toàn bộ đặc trưng một cách đồng đều
  - Độ tương đồng cosine được sử dụng làm hàm đo khoảng cách trong không gian biểu diễn đặc trưng.
  - Quá trình huấn luyện sử dụng thuật toán hạ gradient ngẫu nhiên (SGD) với bộ tối ưu Adam và tốc độ học (learning rate) được thiết lập là 0.01.

3.2.2.5 Trình bày và phân tích kết quả

Luận án sử dụng hai đọ đo hiệu năng là độ chính xác và điểm F1.

Trong thực nghiệm này, luận án so sánh độ chính xác và điểm F1 đạt được giữa Meta-ADD và VAR-WIND, như được thể hiện trên Bảng 3.4 và Bảng 3.5.

Bảng 3.4 So sánh hiệu năng trên bộ 50-50-4800

Loại trôi	Độ chính xác		Điểm F1	
	Meta-ADD	VAR-WIND	Meta-ADD	VAR-WIND
Đột ngột	0.9612	**0.9855**	0.9519	**0.9865**
Dần dần	0.9425	**0.9516**	0.9522	**0.9624**
Gia tăng	0.8911	**0.9860**	0.8904	**0.9862**
Bình thường	0.9478	**0.9707**	0.9433	**0.9649**

Bảng 3.5 So sánh hiệu năng trên bộ 200-25-3800

Loại trôi	Độ chính xác P		Điểm F1	
	Meta-ADD	VAR-WIND	Meta-ADD	VAR-WIND
Đột ngột	0.5698	**0.9103**	0.7043	**0.9324**
Dần dần	0.9496	**0.9632**	0.7867	**0.9514**
Gia tăng	0.9167	**0.9741**	0.9246	**0.9743**
Bình thường	0.9744	**0.9804**	0.9532	**0.9645**

Kết quả trình bày trong Bảng 3.4 và 3.5 cho thấy VAR-WIND đạt hiệu năng vượt trội so với phương pháp tham chiếu Meta-ADD trên cả hai bộ dữ liệu 50-50-4800 và 200-25-3800, với độ chính xác và điểm F1 đều cao hơn ở hầu hết các loại trôi.

Đáng chú ý nhất là ở trường hợp trôi đột ngột trên bộ dữ liệu Dataset-200-25-3800, độ chính xác của Meta-ADD chỉ đạt 0.5698, trong khi VAR-WIND đạt tới 0.9103. Điều này cho thấy chiến lược cửa sổ rời rạc cố định của Meta-ADD dễ bị ảnh hưởng bởi vị trí trôi không xác định trong các luồng dài, đặc biệt khi điểm trôi không rơi vào vị trí trung tâm. Trong khi đó, VAR-WIND sử dụng chiến lược cửa

sổ rời rạc đối xứng, tập trung vào vùng giữa luồng – nơi khả năng xảy ra trôi cao nhất – từ đó cải thiện đáng kể khả năng phát hiện trôi đột ngột.

Ở các loại trôi phức tạp hơn như trôi dần và trôi gia tăng, VAR-WIND tiếp tục thể hiện ưu thế. Với trôi dần dần, chiến lược cửa sổ trượt giúp mô hình ghi nhận tốt hơn các thay đổi nhỏ và kéo dài – vốn dễ bị bỏ sót nếu chỉ sử dụng cặp cửa sổ đơn lẻ như trong Meta-ADD. Với trôi gia tăng, VAR-WIND khai thác cửa sổ mở rộng, mô phỏng quá trình tích lũy dần sai số – đúng với đặc trưng của dạng trôi này – và giúp mô hình học được xu hướng thay đổi hiệu quả hơn.

Ngay cả trong luồng bình thường (không trôi), VAR-WIND vẫn giữ được hiệu năng ổn định và cao hơn Meta-ADD. Điều này cho thấy các chiến lược linh hoạt trong VAR-WIND không chỉ hữu ích trong việc phát hiện trôi mà còn giúp mô hình phân biệt rõ ràng giữa luồng ổn định và luồng có biến đổi.

Một trong những đánh đổi rõ ràng nhất khi áp dụng VAR-WIND là sự gia tăng chi phí tính toán và độ phức tạp trong *quá trình tiền xử lý*. Khác với Meta-ADD – vốn sử dụng chiến lược rời rạc cố định cho mọi kiểu trôi – VAR-WIND đề xuất ba chiến lược trích xuất đặc trưng linh hoạt. Cách tiếp cận này cho phép mô hình khai thác đặc trưng riêng biệt của từng kiểu trôi, từ đó nâng cao độ chính xác phân lớp. Tuy nhiên, việc này cũng làm gia tăng đáng kể số lượng mẫu vector đặc trưng sinh ra từ mỗi luồng dữ liệu ban đầu, đồng nghĩa với việc tăng kích thước tập huấn luyện. Đồng thời, việc phải tính toán nhiều loại cửa sổ với cấu hình khác nhau cũng khiến quá trình tiền xử lý trở nên phức tạp hơn và tiêu tốn nhiều tài nguyên xử lý hơn. Do đó, mặc dù VAR-WIND cải thiện hiệu năng phân lớp, đặc biệt với các kiểu trôi phức tạp, nhưng đi kèm là chi phí tính toán cao hơn so với phương pháp Meta-ADD.

Bảng 3.6 So sánh thời gian giai đoạn dự đoán (ms: mili-giây)

Dataset_50-50-4800		**Dataset_200-25-3800**	
Meta-ADD	**VAR-WIND**	**Meta-ADD**	**VAR-WIND**
453	**426**	562	**541**

Mặc dù VAR-WIND áp dụng các chiến lược trích xuất đặc trưng phức tạp hơn so với Meta-ADD, kết quả Bảng 3.6 cho thấy thời gian dự đoán của VAR-WIND

thậm chí còn thấp hơn hoặc tương đương so với Meta-ADD trên cả hai bộ dữ liệu. Cụ thể, trên tập Dataset_50–50–4800, VAR-WIND đạt thời gian dự đoán 426 ms so với 453 ms của Meta-ADD. Tương tự, trên Dataset_200–25–3800, VAR-WIND cũng có thời gian dự đoán thấp hơn một chút (541 ms so với 562 ms). Điều này cho thấy rằng, mặc dù quá trình trích xuất đặc trưng của VAR-WIND có thể tốn nhiều tài nguyên hơn trong giai đoạn tiền xử lý, nhưng khi mô hình đã được huấn luyện xong, *quá trình dự đoán* không bị ảnh hưởng đáng kể và vẫn duy trì hiệu suất tốt. Đây là một điểm mạnh đáng chú ý, giúp VAR-WIND không chỉ cải thiện độ chính xác phân loại mà còn giữ được tính khả thi trong các hệ thống yêu cầu dự đoán nhanh thời gian thực.

Bảng 3.7 So sánh độ chính xác phân lớp và phát hiện trôi trung bình

Thông số	Dataset_50-50-4800		Dataset_200-25-3800	
	Meta-ADD	**VAR-WIND**	**Meta-ADD**	**VAR-WIND**
Độ chính xác phân lớp trung bình	0.9321	**0.9785**	0.8433	**0.9549**
Độ chính xác phát hiện trôi trung bình	0.9743	**0.9856**	0.8867	**0.9741**

Bảng 3.7 so sánh độ chính xác phân lớp trung bình và độ chính xác phát hiện trôi trung bình giữa hai mô hình. Dựa trên kết quả này cho thấy VAR-WIND cho hiệu suất tốt hơn so với Meta-ADD trên cả hai bộ dữ liệu.

Các kết quả trên chứng minh rằng chiến lược cửa sổ linh hoạt luôn cho hiệu năng cao hơn chiến lược cửa sổ rời rạc trên cả hai tập dữ liệu. Điều này cho thấy khả năng thích ứng của phương pháp cửa sổ linh hoạt, giúp cải thiện hiệu quả trong việc nắm bắt và phản hồi sự thay đổi của luồng dữ liệu, từ đó nâng cao độ chính xác trong các nhiệm vụ phân lớp và phát hiện trôi khái niệm.

Một giả định của VAR-WIND là biết trước điểm trôi trong quá trình huấn luyện để áp dụng các chiến lược cửa sổ chuyên biệt cho từng kiểu trôi, đây không phải là một hạn chế mà là thiết kế có chủ đích. Mục tiêu là tối ưu hóa quá trình trích xuất đặc trưng, từ đó giúp mô hình học phân biệt rõ hơn các biểu hiện đặc trưng của từng loại trôi khái niệm. Trong các ứng dụng thực tế, điểm trôi là không biết trước. Tuy nhiên,

VAR-WIND không đảm nhiệm nhiệm vụ *phát hiện trôi*, mà được thiết kế để huấn luyện bộ phân lớp kiểu trôi trên tập dữ liệu đã có gán nhãn. Điều này tương tự như nhiều phương pháp trong học máy sử dụng thông tin trong giai đoạn huấn luyện để đạt được khả năng phân biệt mạnh hơn, rồi tổng hợp vào hệ thống phát hiện – phân lớp hoàn chỉnh. Do đó, VAR-WIND đóng vai trò như một mô-đun chuyên biệt để nâng cao khả năng nhận diện loại trôi, có thể được tích hợp vào các hệ thống thực tế, miễn là có mô-đun phát hiện điểm trôi đứng trước. Mô hình không mâu thuẫn với thực tế nếu được triển khai trong kiến trúc phát hiện - phân loại trôi tách biệt, phổ biến trong nhiều hệ thống xử lý luồng.

Hạn chế của VAR-WIND

Trong thiết kế ban đầu, VAR-WIND được xây dựng nhằm tối ưu khả năng học biểu hiện đặc trưng của từng loại trôi một cách rõ ràng, do đó mỗi luồng huấn luyện đều giả định chỉ chứa một kiểu trôi duy nhất. Trong trường hợp thực tế, luồng chứa nhiều điểm trôi, hoặc trôi kéo dài không rõ ranh giới — phương pháp VAR-WIND có thể được áp dụng như một thành phần trong hệ thống xử lý lớn hơn, nơi luồng dữ liệu dài được chia thành các đoạn nhỏ (bằng thuật toán phát hiện điểm trôi). Khi đó, VAR-WIND sẽ thực hiện phân loại kiểu trôi trên từng đoạn riêng biệt, đảm bảo tính linh hoạt và khả năng mở rộng. Trong các nghiên cứu tiếp theo, luận án có thể mở rộng VAR-WIND bằng cách huấn luyện trên các luồng có nhiều loại trôi xen kẽ, hoặc tích hợp cơ chế tự chú ý để tự động học các vùng biểu hiện trôi nổi bật, từ đó cải thiện khả năng áp dụng vào luồng trôi không rõ ràng.

3.3 Mô hình phân lớp trôi khái niệm MetaLDD-Finetune đề xuất

Trong giai đoạn dự đoán, Meta-ADD thực hiện đoán kiểu trôi trên các mẫu dữ liệu mới, chưa từng thấy trước đây. Điều này đòi hỏi mô hình đề xuất cần có tính thích nghi để đảm bảo dự đoán chính xác các mẫu dữ liệu mới.

Phần này đề xuất mô hình MetaLDD-Finetune - là cải tiến của mô hình Meta-ADD đã trình bày ở Phần 3.1.2 với mục tiêu làm tăng độ chính xác phân lớp các mẫu dữ liệu trôi mới mà không cần huấn luyện lại mạng nguyên mẫu.

3.3.1 Mô hình MetaLDD-Finetune đề xuất

Trong bối cảnh dự đoán trực tuyến, mô hình cần có khả năng thích nghi để phân lớp chính xác các kiểu trôi khái niệm chưa từng xuất hiện trong tập huấn luyện ban đầu. Điều này đặc biệt quan trọng khi dữ liệu thay đổi liên tục và số mẫu gán nhãn rất hạn chế.

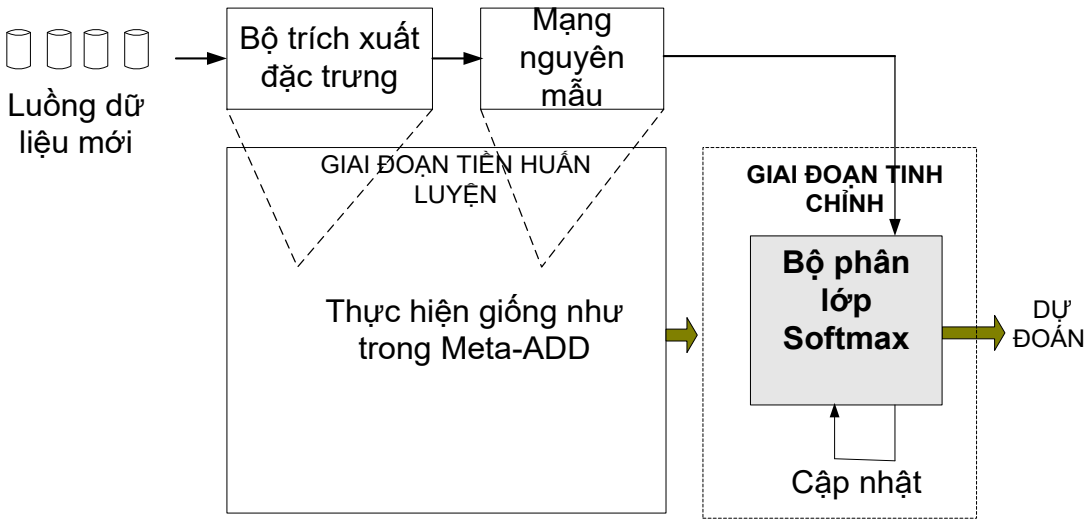
Trên cơ sở mô hình Meta-ADD đã trình bày ở Mục 3.1.2, luận án đề xuất mô hình **MetaLDD-Finetune**, với mục tiêu cải thiện độ chính xác phân lớp mà không cần huấn luyện lại toàn bộ mạng nguyên mẫu. Ý tưởng chính là thay thế bộ phân lớp dựa trên khoảng cách với tâm các lớp trong Meta-ADD bằng bộ phân lớp Softmax, sau đó tiến hành tinh chỉnh bộ phân lớp này trên tập dữ liệu mới, sử dụng đồng thời cả tập huấn luyện và tập kiểm tra.

Trong **giai đoạn tiền huấn luyện**, mạng so khớp được huấn luyện theo cách thức huấn luyện theo tập nhiệm vụ, giống Meta-ADD, nhằm học được không gian đặc trưng tổng quát.

Giai đoạn tinh chỉnh: mô hình giữ nguyên mạng nguyên mẫu từ giai đoạn tiền huấn luyện, chỉ thay thế bộ phân lớp trong Meta-ADD bằng bộ phân lớp Softmax. Mục tiêu chính của giai đoạn này là cập nhật bộ phân lớp Softmax sao cho nó có thể phân lớp chính xác các lớp mới với chỉ một vài mẫu huấn luyện của luồng dữ liệu mới trong tập dữ liệu tinh chỉnh. Luận án đề xuất ý tưởng tinh chỉnh như sau:

(i) Khởi tạo trọng số của bộ phân lớp Softmax.

(ii) Cập nhật lại bộ phân lớp Softmax bằng cách thức huấn luyện có giám sát trên cả các mẫu hỗ trợ và mẫu truy vấn của tập dữ liệu tinh chỉnh. Hàm mất mát được bổ sung thêm điều chuẩn độ bất định (Entropy Regularization) nhằm tăng tính phân biệt và khả năng tổng quát hoá.



Hình 3.7 Mô hình MetaLDD-Finetune đề xuất

Gọi $(\boldsymbol{x}_i, y_i)$ là một mẫu trong tập huấn luyện S, $\boldsymbol{f}_\phi(\boldsymbol{x}_i)$ là vector đặc trưng đầu ra của mạng so khớp. Sử dụng bộ phân lớp Softmax ta thu được xác suất dự đoán p_i của x_i là:

$$p_i = softmax\ (\mathbf{W}\cdot\boldsymbol{f}_\phi(\boldsymbol{x}_i) + \boldsymbol{b}) \tag{3.13}$$

trong đó $\boldsymbol{W}$ là vector trọng số và $\boldsymbol{b}$ là giá trị thiên lệch.

Khởi tạo trọng số bộ phân lớp:

Ta khởi tạo $\boldsymbol{b} = 0$ và $\boldsymbol{W}$ được khởi tạo như sau: đối với mỗi lớp trôi k, vector trọng số $\boldsymbol{w}_k$ của bộ phân lớp Softmax được khởi tạo bằng trọng số của vector tâm lớp $\boldsymbol{c}_k$ của các mẫu hỗ trợ thuộc lớp đó. Tâm lớp $\boldsymbol{c}_k$ được tính bằng cách lấy trung bình của tất cả các vector đặc trưng của các mẫu trong tập hỗ trợ S_k của lớp k:

$$\boldsymbol{w}_k = \boldsymbol{c}_k = \frac{1}{|S_k|}\sum_{\boldsymbol{x}_i \in S_k} \boldsymbol{f}_\phi(\boldsymbol{x}_i) \tag{3.14}$$

Các vector này trở thành trọng số khởi tạo của bộ phân lớp Softmax, giúp mô hình có điểm xuất phát tốt hơn trước khi cập nhật.

Cập nhật bộ phân lớp Softmax

Bộ phân lớp Softmax sau đó tiếp tục được cập nhật bằng cách huấn luyện theo cách thức *học có giám sát*, sử dụng cả các mẫu hỗ trợ và mẫu kiểm tra. Ta cần tối thiểu hóa hàm mất mát tổng thể $L_{fine-tune}$, là kết hợp của cả hàm mất mát độ bất định chéo và điều chuẩn độ bất định:

$$L_{fine-tune} = L_{cross-entropy} + \lambda H(p) \tag{3.15}$$

Trong đó:

- $L_{cross-entropy}$ là mất mát entropy chéo trung bình của các mẫu trong tập S, của vòng huấn luyện hiện tại.

$$L_{cross-entropy} = \frac{1}{N_S}\sum_{(\boldsymbol{x},y)\in S} -\log\left(p_\phi(y \mid \boldsymbol{x})\right) \tag{3.16}$$

- λ là tham số điều chuẩn độ bất định. Giá trị tham số λ ảnh hưởng đến hiệu năng mô hình như sau: λ nhỏ sẽ ưu tiên giảm mất mát độ bất định, nhưng có thể gây ra dự đoán không chắc chắn. λ trung bình tạo ra sự cân bằng giữa độ chính xác và độ chắc chắn của dự đoán. λ lớn sẽ dễ dẫn đến quá khớp do mô hình quá tự tin.
- $H(p)$: độ bất định trung bình của phân phối dự đoán trên mẫu trong tập Q, của vòng huấn luyện hiện tại:

$$H(p) = \frac{1}{N_Q}\sum_{(x,y)\in Q} H\left(p_\phi(y \mid \boldsymbol{x})\right) \quad (3.17)$$

Sử dụng cực tiểu hóa độ bất định để giúp mô hình dự đoán chắc chắn hơn trên các mẫu trong tập truy vấn. Mục tiêu của cực tiểu hóa độ bất định là nếu một mẫu có xác suất dự đoán phân tán giữa nhiều lớp, hàm mất mát sẽ khuyến khích mô hình đưa ra dự đoán chắc chắn hơn. Điều này giúp tận dụng thông tin thống kê trong tập truy vấn để cải thiện khả năng phân lớp. Bản chất của $H(p)$ là thúc đẩy mô hình đưa ra dự đoán mang tính chắc chắn cao, giúp cải thiện hiệu năng phân lớp.

3.3.2 Thực nghiệm và đánh giá

Luận án kế thừa phương thức thực nghiệm mà Meta-ADD đã thực hiện, được mô tả chi tiết dưới đây.

3.3.2.1 Mục tiêu

Như được đề cập, mô hình MetaLDD-Finetune được đề xuất như một cải tiến trực tiếp từ mô hình Meta-ADD – một phương pháp dựa trên học cách học, có khả năng phân lớp các kiểu trôi khái niệm. Vì vậy, việc đánh giá hiệu quả của mô hình MetaLDD-Finetune chủ yếu tập trung vào so sánh với Meta-ADD nhằm chứng minh hiệu năng phân lớp kiểu trôi được cải thiện.

Mục tiêu thực nghiệm: so sánh hiệu năng giữa mô hình Meta-ADD và mô hình MetaLDD-Finetune.

Kịch bản thực nghiệm: Thực nghiệm MetaADD và và MetaLDD-Finetune trên từng bộ dữ liệu.

3.3.2.2 Thiết lập thực nghiệm

- **Phần cứng**: máy tính Dell PowerEdge T40, CPU Intel Xeon E-2224G 3.5 Ghz, 32GB RAM, 2TB HDD.
- Phần mềm: hệ điều hành Windows 10, Pycharm IDE Community, Python 3.11.
    - Bộ dữ liệu: Tương tự như VAR-WIND, luận án sử dụng hai bộ dữ liệu thực nghiệm là Dataset_50-15-4800 và Dataset_200-25-3800; mỗi bộ dữ liệu này được chia thành ba phần: **tiền huấn luyện (60%), tinh chỉnh (20%) và kiểm tra (20%).**

3.3.2.3 Triển khai thực nghiệm

- **Chiến lược tiền huấn luyện, tinh chỉnh và kiểm tra**

  Giai đoạn tiền huấn luyện

  Huấn luyện cả MetaADD và MetaLDD-Finetune bằng phương pháp huấn luyện theo từng tập nhiệm vụ, gồm **600** vòng. Trong mỗi vòng huấn luyện, thực hiện:

  - Chọn ngẫu nhiên các luồng dữ liệu từ 4 kiểu trôi và chia thành hai tập:
    - Tập huấn luyện: gồm 5 luồng cho mỗi kiểu trôi.
    - Tập kiểm tra: gồm 15 luồng cho mỗi kiểu trôi.
  - Các luồng trong tập huấn luyện được đưa qua mạng nhúng FAN (Fully Attention Network) để tạo ra các vector đặc trưng. Trung bình các vector đặc trưng thuộc cùng một kiểu trôi sẽ tạo ra vector tâm lớp đại diện cho kiểu trôi đó. Lý do sử dụng mạng FAN được giải thích trong Meta-ADD là vì mạng FAN được thiết kế dựa trên các lớp tích chập kết hợp với cơ chế chú ý, giúp tập trung vào các đặc trưng quan trọng trong dữ liệu đầu vào, thay vì xử lý toàn bộ đặc trưng một cách đồng đều. Mạng FAN có kích thước nhỏ gọn, đảm bảo chi phí tính toán thấp và khả năng triển khai nhanh, phù hợp với môi trường xử lý luồng dữ liệu.
  - Các luồng trong tập kiểm tra cũng được đưa qua mạng FAN để tạo vector đặc trưng. Mỗi vector đặc trưng trong tập kiểm tra được so sánh với các tâm lớp bằng hàm khoảng cách Cosine để dự đoán kiểu trôi tương ứng.
  - Hàm mất mát trong quá trình huấn luyện là logarit âm của xác suất dự đoán chính xác.
  - Tối ưu hóa được thực hiện bằng thuật toán Adam với tốc độ ban đầu là 0.01, sau đó giảm dần theo lịch học sử dụng StepLR với hệ số giảm gamma = 0.9 và bước giảm là 30.

  Sau khi tiền huấn luyện, mô hình và ma trận tâm lớp được lưu lại để phục vụ cho giai đoạn tinh chỉnh và giai đoạn kiểm tra.

  Giai đoạn tinh chỉnh MetaLDD-Finetune:

Trong giai đoạn này, bộ phân lớp Softmax được cập nhật bằng cách huấn luyện lại trên tập tinh chỉnh bằng phương pháp học có giám sát, sử dụng cả các mẫu

trong tập huấn luyện và tập kiểm tra. Quá trình tối ưu được thực hiện bằng giảm dần gradient ngẫu nhiên (SGD) với thuật toán Adam (tốc độ học 0.01), và sử dụng hàm mất mát tổng hợp gồm:

- Mất mát entropy chéo (cross-entropy),
- Điều chuẩn độ bất định với hệ số điều chỉnh λ=0.1.

Giai đoạn kiểm tra

- Với Meta-ADD, việc đánh giá được thực hiện như sau:
    - Lấy 20 luồng từ tập huấn luyện (5 luồng cho mỗi kiểu trôi) để tính tâm lớp cho từng kiểu trôi. Việc sử dụng luồng huấn luyện thay vì kiểm tra nhằm đảm bảo khả năng tổng quát hóa cho các biểu hiện mới của từng kiểu trôi – đây là đặc điểm thiết kế có chủ đích trong Meta-ADD.
    - Mỗi tập kiểm tra cũng được chia thành tập huấn luyện và tập kiểm tra giống như trong giai đoạn tiền huấn luyện. Dự đoán kiểu trôi được thực hiện bằng so sánh khoảng cách giữa vector đặc trưng và các tâm lớp. Độ chính xác phân lớp được tính trên các tập kiểm tra.
- Với MetaLDD-Finetune, kiểm tra được thực hiện bằng cách khởi tạo và tinh chỉnh bộ phân lớp Softmax trực tiếp trên tập kiểm tra, sử dụng cả tập huấn luyện và tập kiểm tra. Không sử dụng lại các tâm lớp cố định từ tập huấn luyện, mô hình thích nghi nhanh với các biểu hiện mới trong mỗi tập kiểm tra.

Kết quả trung bình trên các tập kiểm tra được sử dụng để đánh giá hiệu năng tổng thể của mô hình.

3.3.2.4 Trình bày và phân tích kết quả

Luận án sử dụng hai độ đo hiệu năng là độ chính xác và điểm F1. Hơn nữa, luận án còn so sánh thời gian dự đoán của hai mô hình.

Thực nghiệm 1: Đánh giá độ chính xác và điểm F1 cho phân lớp trôi.

Bảng 3.8 Độ chính xác và điểm F1 trên Dataset_50-15-4800

Kiểu trôi	Độ chính xác		Điểm F1	
	Meta-ADD	MetaLDD-Finetune	Meta-ADD	MetaLDD-Finetune
Đột ngột	0.8913	**0.9104**	0.8910	**0.8915**
Dần dần	**0.7411**	0.7109	**0.7544**	0.7508
Gia tăng	0.7415	**0.7805**	0.7447	**0.7607**
Bình thường	0.8908	**0.9001**	0.8901	**0.9002**

Bảng 3.9 Trung bình độ chính xác phân lớp – Dataset_50-15-4800

Mô hình	Độ chính xác
Meta-ADD	0.87
MetaLDD-Finetune	**0.88**

Bảng 3.8 và Bảng 3.9 trình bày kết quả thực nghiệm so sánh giữa hai mô hình Meta-ADD và MetaLDD-Finetune trên bộ dữ liệu Dataset_50–15–4800. Bảng 3.8 cho thấy hiệu năng theo từng kiểu trôi, còn Bảng 3.9 tổng hợp độ chính xác trung bình.

Chi tiết hơn ở Bảng 3.8, MetaLDD-Finetune cho độ chính xác tốt hơn MetaADD trong ba kiểu trôi đó là trôi đột ngột (0.9104 so với 0.8913), trôi gia tăng (0.7805 so với 0.7415) và kiểu bình thường (0.9001 so với 0.8908). Điều này khẳng định rằng việc cập nhật bộ phân lớp Softmax dựa trên dữ liệu mới, cùng với khởi tạo trọng số bằng tâm lớp và sử dụng điều chuẩn độ bất định, đã giúp mô hình phát hiện hiệu quả hơn với các thay đổi bất ngờ hoặc gia tăng trong phân phối dữ liệu.

Tuy nhiên, trong trường hợp trôi dần dần, Meta-ADD lại chiếm ưu thế hơn (0.7411 so với 0.7109). Điều này có thể lý giải bởi ***đặc trưng mờ nhạt của trôi dần dần*** khiến việc cập nhật bộ phân lớp có thể chưa mang lại lợi ích rõ rệt và thậm chí dễ bị sai lệch nếu mô hình chưa nhận diện rõ ràng được biểu hiện trôi.

Xét tổng thể, MetaLDD-Finetune đạt độ chính xác trung bình trên tập kiểm tra cao hơn một chút so với Meta-ADD (0.88 so với 0.87, theo Bảng 3.8), cho thấy

mô hình đề xuất có tiềm năng cải thiện hiệu suất phân loại trong bối cảnh học ít mẫu. Dù mức tăng không lớn, nhưng đây là kết quả đáng chú ý bởi *việc tinh chỉnh bộ phân lớp Softmax chỉ diễn ra trên một lượng dữ liệu rất hạn chế từ luồng mới*.

Kiểm định Wilcoxon trên tập Dataset_50-15-4800

Trên tập dữ liệu Dataset_50-15-4800, luận án tiến hành so sánh hiệu quả phân lớp kiểu trôi khái niệm giữa hai mô hình Meta-ADD và MetaLDD-Finetune. Mỗi mô hình được thực thi **30** lần lặp với các giá trị khởi tạo ngẫu nhiên khác nhau nhằm đảm bảo tính khách quan.

Để kiểm tra xem sự chênh lệch này có phải là ngẫu nhiên hay không, luận án sử dụng kiểm định Wilcoxon – một phương pháp phi tham số phù hợp để so sánh hai tập dữ liệu có liên hệ. Cụ thể:

- Mỗi cặp độ chính xác tương ứng từ hai mô hình, trên cùng một giá trị khởi tạo ngẫu nhiên, được sử dụng để tính hiệu sai khác.
- Sau đó, thực hiện xếp hạng các sai khác và tính tổng thứ hạng dương và âm.

Kết quả thu được sau 30 lần chạy, hai mô hình Meta-ADD và MetaLDD-Finetune lần lượt đạt độ chính xác trung bình là **0.8712** và **0.8778.**

- Giá trị thống kê Wilcoxon thu được là **79**
- Giá trị p tương ứng là **p = 0.0010**

Với $p < 0.05$, kết luận rằng sự khác biệt giữa MetaLDD-Finetune và Meta-ADD là có ý nghĩa thống kê. Nói cách khác, mô hình MetaLDD-Finetune đã thể hiện hiệu quả hơn một cách đáng tin cậy trên tập dữ liệu này trong việc phân lớp các kiểu trôi khái niệm.

Luận án tiếp tục thử nghiệm trên bộ dữ liệu thứ hai là bộ Dataset_200-25-3800 với kỳ vọng rằng MetaLDD-Finetune sẽ cho kết quả hiệu năng tốt hơn so với Meta-ADD. Bảng 3.10 và Bảng 3.11 trình bày kết quả so sánh giữa hai mô hình phân lớp Meta-ADD và MetaLDD-Finetune trên tập dữ liệu Dataset_200–25–3800.

Bảng 3.10 Độ chính xác và điểm F1 trên Dataset_200-25-3800

Kiểu trôi	Độ chính xác		Điểm F1	
	Meta-ADD	MetaLDD-Finetune	Meta-ADD	MetaLDD-Finetune
Đột ngột	**0.7104**	0.6908	**0.8100**	0.8002
Dần dần	0.9211	**0.9310**	0.8320	**0.8433**
Gia tăng	0.9202	**0.9401**	0.9201	**0.9405**
Bình thường	0.9480	**0.9520**	0.9302	**0.9411**

Bảng 3.11 Trung bình độ chính xác phân lớp – Dataset_200-25-3800

Mô hình	Độ chính xác
Meta-ADD	0.82
MetaLDD-Finetune	**0.83**

Nhìn tổng thể ở cả hai bảng, MetaLDD-Finetune tiếp tục cho thấy hiệu năng cạnh tranh hoặc vượt hơn trong phần lớn các tình huống, đặc biệt là đối với những kiểu trôi có tính chất thay đổi phức tạp như trôi dần và trôi gia tăng.

Xét chi tiết trên từng kiểu trôi (Bảng 3.10), MetaLDD-Finetune vượt Meta-ADD ở cả độ chính xác và điểm F1 đối với hai kiểu trôi có biểu hiện mờ dần theo thời gian là trôi dần dần (0.9310 so với 0.9211) và trôi gia tăng (0.9401 so với 0.9202). Đây là minh chứng rõ ràng cho hiệu quả của việc tận dụng thông tin từ cả tập huấn luyện và tập kiểm tra để cập nhật bộ phân lớp, đặc biệt khi dữ liệu biểu hiện trôi không rõ ràng và có tính tích lũy. Trong trường hợp không có trôi, MetaLDD-Finetune cũng nhỉnh hơn (0.9520 so với 0.9480), chứng minh rằng việc tinh chỉnh bộ phân lớp không gây ra hiệu ứng tiêu cực hay làm giảm khả năng tổng quát hóa trong điều kiện ổn định. Tuy nhiên, đối với trôi đột ngột, MetaLDD-Finetune có hiệu năng thấp hơn Meta-ADD (0.6908 so với 0.7104 về độ chính xác). Điều này có thể lý giải do trên một luồng dài với số cửa sổ lớn và số mẫu trong một cửa sổ ít (Dataset 200_25_3800 mỗi luồng có tới 200 cửa sổ, mỗi cửa sổ có 25 mẫu), khi có ảnh hưởng của nhiễu khả năng dự đoán sai sẽ cao. Việc cập

nhật bộ phân lớp trên lượng nhỏ dữ liệu mới có thể gây nhiễu, nhất là khi biểu hiện thay đổi nhanh và đột ngột.

Đối với điểm F1, MetaLDD-Finetune cũng cho kết quả tốt hơn so với Meta-ADD ở ba kiểu là dần dần, gia tăng và bình thường. Riêng trôi đột ngột thì MetaADD cho kết quả tốt hơn. Điều này cũng phù hợp với nhận xét ở trên.

Về độ chính xác trung bình trên tập kiểm tra (Bảng 3.11), MetaLDD-Finetune đạt giá trị 0.83 so với 0.82 của Meta-ADD. Dù mức chênh lệch không lớn, kết quả này khẳng định rằng việc cập nhật trực tiếp bộ phân lớp Softmax dựa trên dữ liệu mới có thể hỗ trợ mô hình thích nghi nhanh hơn trong bối cảnh học ít mẫu, mà không cần thay đổi cấu trúc mạng so khớp.

Kiểm định Wilcoxon trên tập Dataset_200-25-3800

Trên tập dữ liệu Dataset_200-25-3800, mô hình MetaLDD-Finetune đạt độ chính xác trung bình là **0.83**, cao hơn so với **0.82** của mô hình Meta-ADD. Để đánh giá mức độ khác biệt này, luận án cũng thực hiện kiểm định Wilcoxon trên 30 cặp kết quả tương ứng. Kết quả kiểm định thu được giá trị thống kê **Wilcoxon = 5.0 với p = 0.0391**, cho thấy sự khác biệt là có ý nghĩa thống kê ở mức tin cậy 95%. Điều này khẳng định rằng mô hình MetaLDD-Finetune có hiệu quả phân loại trôi khái niệm tốt hơn một cách đáng tin cậy.

Thực nghiệm 2: Đánh giá độ chính xác phát hiện trôi và thời gian dự đoán

Trong thực nghiệm này, luận án tính toán độ chính xác phát hiện trôi trung bình (*không phân biệt kiểu trôi)* và thời gian dự đoán trung bình của của hai mô hình.

Bảng 3.12 Độ chính xác phát hiện trôi và thời gian dự đoán bộ 50-15-4800

Mô hình	Độ chính xác trung bình	Thời gian trung bình (ms)
Meta-ADD	0.94	546
MetaLDD-Finetune	**0.95**	**396**

Bảng 3.13 Độ chính xác phát hiện trôi và thời gian dự đoán bộ 200-25-3800

Mô hình	Độ chính xác trung bình	Thời gian trung bình (ms)
Meta-ADD	0.87	521
MetaLDD-Finetune	**0.88**	**369**

Bảng 3.12 và Bảng 3.13 trình bày kết quả so sánh giữa hai mô hình Meta-ADD và MetaLDD-Finetune về độ chính xác *phát hiện* trôi trung bình và thời gian dự đoán trung bình trên hai bộ dữ liệu Dataset_50–15–4800 và Dataset_200–25–3800. Kết quả cho thấy rằng MetaLDD-Finetune không chỉ cải thiện nhẹ về độ chính xác mà còn có thời gian dự đoán ngắn hơn đáng kể so với Meta-ADD.

Cụ thể, trên tập Dataset_50–15–4800, độ chính xác *phát hiện* trôi của MetaLDD-Finetune đạt 0.95 so với 0.94 của Meta-ADD. Đặc biệt, thời gian xử lý trung bình mỗi lượt dự đoán của MetaLDD-Finetune chỉ là 396ms, giảm gần 27.5% so với 546ms của Meta-ADD. Tương tự, trên tập Dataset_200–25–3800, MetaLDD-Finetune cũng vượt nhẹ về độ chính xác (0.88 so với 0.87) trong khi thời gian dự đoán giảm từ 521ms xuống còn 369ms.

Kết quả này khẳng định rằng việc thay thế bộ phân lớp của mạng nguyên mẫu bằng bộ phân lớp Softmax và cập nhật linh hoạt trên dữ liệu mới không chỉ nâng cao hiệu quả phát hiện trôi mà còn rút ngắn thời gian tính toán. Điều này có ý nghĩa thực tiễn quan trọng, đặc biệt trong các hệ thống phát hiện trôi hoạt động theo thời gian thực hoặc có tài nguyên tính toán hạn chế.

Hạn chế của MetaLDD-Finetune

Mặc dù MetaLDD-Finetune cho thấy hiệu năng tốt hơn so với Meta-ADD cả về độ chính xác phân lớp lẫn thời gian dự đoán, việc áp dụng mô hình này cũng đòi hỏi một số đánh đổi nhất định:

- *Đầu tiên* là chi phí huấn luyện tăng lên do cần thực hiện giai đoạn tinh chỉnh bộ phân lớp Softmax trên từng tập kiểm tra mới, trong khi Meta-ADD chỉ cần sử dụng khoảng cách đến tâm lớp cố định. Điều này làm tăng thêm độ phức tạp thuật toán và yêu cầu phải lưu giữ mô hình nhúng đã tiền huấn luyện, đồng thời thực hiện tối ưu hóa bổ sung cho từng luồng dữ liệu mới.

- *Thứ hai*, chất lượng phân lớp phụ thuộc vào khả năng biểu diễn của mạng so khớp được huấn luyện trước. Trong những trường hợp mà các biểu hiện trôi trong dữ liệu mới có khác biệt lớn với kiểu trôi đã được thấy trong tập huấn luyện, hoặc khi dữ liệu đầu vào chứa nhiễu cao, mạng so khớp có thể không tạo ra các vector đặc trưng đủ phân tách giữa các lớp. Khi đó, dù bộ phân lớp Softmax có được tinh chỉnh, hiệu năng tổng thể vẫn có thể bị suy giảm.
- *Thứ ba*, việc khởi tạo bộ phân lớp Softmax dựa trên tâm lớp của tập huấn luyện cũng có thể gặp khó khăn khi các mẫu trong cùng một lớp phân tán rộng hoặc bị chồng lấn với các lớp khác trong không gian đặc trưng. Trong điều kiện lý tưởng, các vector đặc trưng của cùng một lớp cần hội tụ gần tâm lớp, nhưng thực tế, điều này có thể không xảy ra nếu tập dữ liệu mới có phân bố sai lệch hoặc bị ảnh hưởng bởi hiện tượng trôi không đồng nhất.

Những yếu tố trên cho thấy rằng phương pháp MetaLDD-Finetune vẫn dựa nhiều vào sự tương đồng giữa dữ liệu mới và dữ liệu huấn luyện ban đầu. Điều này mở ra hướng nghiên cứu tiếp theo về việc tăng cường tính linh hoạt của mạng so khớp. Đây chính là các yếu tố để luận án đi sâu hơn vào Chương 4 nhằm giải quyết các điểm nêu trên.

3.4 Kết luận chương 3

Chương này đã trình bày nhóm đóng góp thứ hai của luận án thông qua việc đề xuất hai mô hình phân lớp trôi khái niệm nhằm nâng cao hiệu quả phân lớp các dạng trôi trong bối cảnh học ít mẫu. Các mô hình được phát triển dựa trên mạng nguyên mẫu. Kết quả nghiên cứu theo nhóm đóng góp này của luận án được công bố trong [TungNK5] cho VAR-WIND (đã nhận được một tham chiếu Scopus từ các tác giả nước ngoài[11]), [TungNK3] cho MetaLDD-Finetune.

Trong chương tiếp theo, luận án đề cập đến vấn đề tối ưu không gian biểu diễn để từ đó đề xuất các kỹ thuật phân lớp kiểu trôi hiệu quả.

[11] Patil, M.A., Kumar, S. & Kumar, S. *GraphDrift-net: a dynamic graph-based framework for concept drift detection in short unstructured text streams*. Eur. Phys. J. Plus 140, 884 (2025).

Chương 4. Mô hình phân lớp trôi khái niệm dựa trên tối ưu không gian biểu diễn đặc trưng

Chương này trình bày mô hình đề xuất CS&AM_SC (Cosine Similarity and Angular Margin based Softmax Classifier), như một cải tiến của mô hình MetaLDD-Finetune nhằm nâng cao chất lượng biểu diễn đặc trưng.

Phần đầu của chương phân tích đặc điểm của không gian biểu diễn đặc trưng và chỉ ra hai yếu tố then chốt ảnh hưởng đến hiệu quả phân lớp, đó là: (i) sự biến thiên trong lớp và (ii) hiện tượng chồng lấn giữa các lớp.

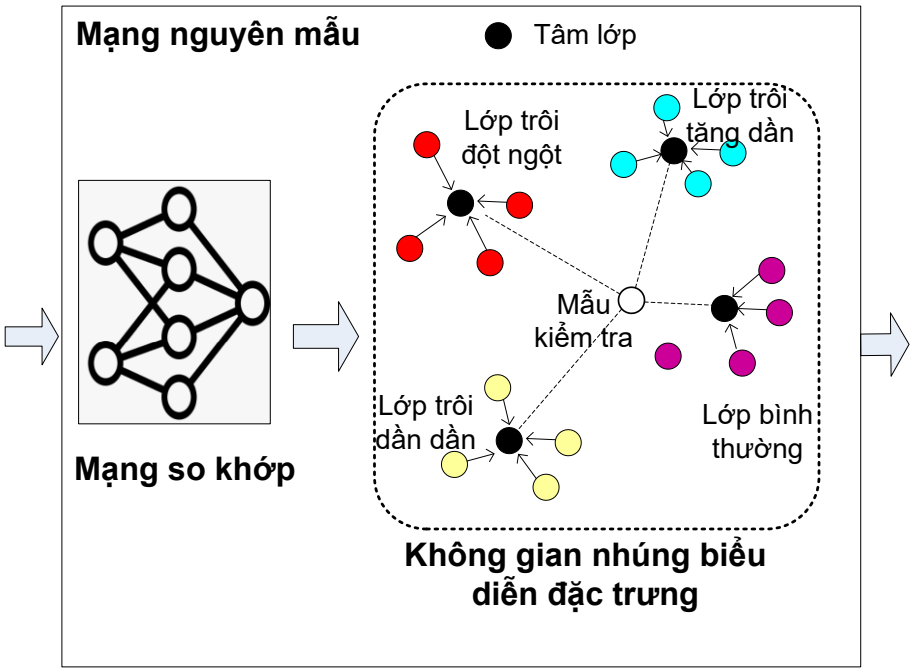
Phần thứ hai đi sâu phân tích nguyên nhân gây ra sự biến thiên trong lớp và đề xuất hai kỹ thuật để khắc phục vấn đề này: (1) thay thế tích vô hướng trong bộ phân lớp Softmax bằng độ tương đồng Cosine, và (2) áp dụng hàm mất mát trung tâm (center loss).

Phần thứ ba tập trung vào hiện tượng chồng lấn giữa các lớp, từ đó đề xuất bổ sung biên góc vào bộ phân lớp Softmax nhằm tăng cường khả năng phân tách giữa các lớp trong không gian đặc trưng.

Cuối cùng, các kỹ thuật trên được tích hợp thành mô hình đề xuất CS&AM_SC. Mô hình này sử dụng chiến lược huấn luyện với hàm mất mát tổng hợp, đồng thời cập nhật cả mạng nguyên mẫu và bộ phân lớp Softmax nhằm mục tiêu là tối ưu hóa không gian biểu diễn đặc trưng theo hai hướng đồng thời: giảm thiểu biến thiên trong lớp và tăng cường sự phân biệt giữa các lớp.

4.1 Không gian biểu diễn đặc trưng

Trong các hệ thống học máy áp dụng cho luồng dữ liệu động, đặc biệt là trong bài toán phát hiện và phân lớp trôi khái niệm, việc xây dựng một không gian biểu diễn đặc trưng hiệu quả là yếu tố rất quan trọng để đảm bảo hiệu năng phân lớp lâu dài và ổn định [55]. Không giống như các bài toán phân lớp thông thường, nơi các lớp dữ liệu thường được giả định là ổn định và có phân phối rõ ràng, dữ liệu luồng trong môi trường trôi khái niệm thường có sự thay đổi liên tục về phân phối xác suất, ngữ nghĩa của nhãn, cũng như tính đại diện của các thuộc tính đầu vào theo thời gian. Chính sự thay đổi này khiến cho việc học một bộ phân lớp tĩnh trở nên không còn hiệu quả, từ đó đòi hỏi các kỹ thuật học biểu diễn phải thích nghi tốt với sự biến thiên của dữ liệu.



Hình 4.1 Minh họa không gian biểu diễn đặc trưng

Một trong những hướng tiếp cận mang tính nền tảng để giải quyết vấn đề trên là học một không gian biểu diễn đặc trưng sao cho các mẫu dữ liệu thuộc cùng một lớp nằm gần nhau, đồng thời mẫu dữ liệu tương ứng với các lớp khác nhau phải được phân tách rõ ràng, được minh họa trên Hình 4.1. Trong không gian đặc trưng lý tưởng, mỗi kiểu trôi khái niệm nên được biểu diễn dưới dạng một lớp đặc trưng rõ ràng, với mức độ biến thiên thấp trong lớp và khoảng cách đủ lớn giữa các lớp khác nhau. Tuy nhiên, trong thực tế, điều này rất khó đạt được do các lý do như số lượng mẫu ít, phân phối dữ liệu không đồng đều, và bản chất giao thoa giữa các kiểu trôi.

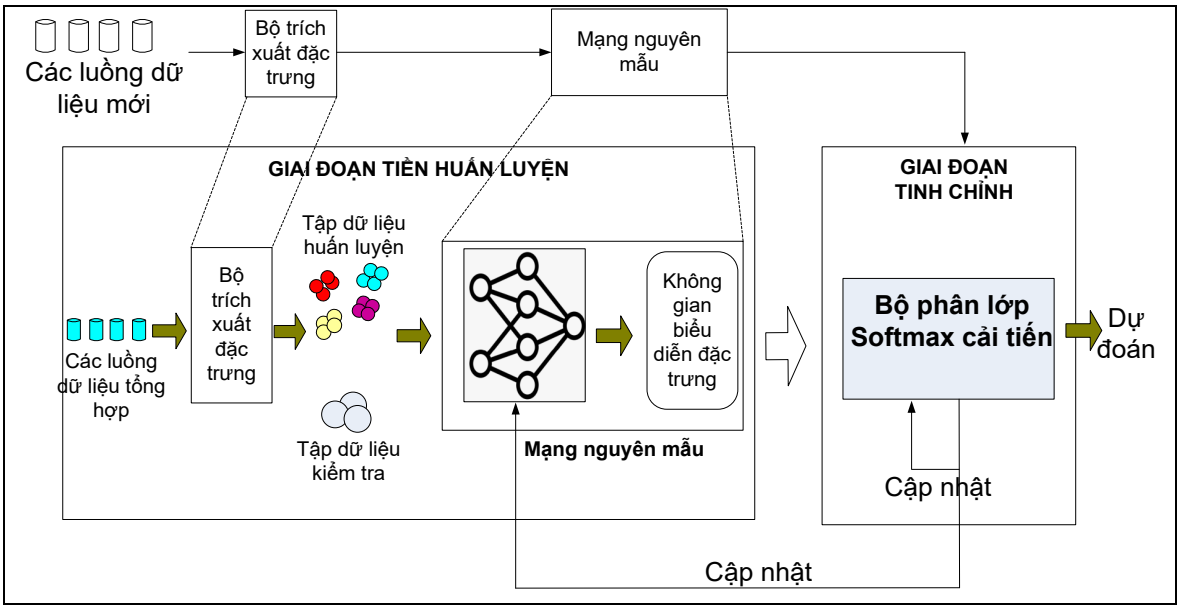
Mô hình MetaLDD-Finetune tập trung vào việc tinh chỉnh bộ phân lớp Softmax mà *chưa* đề cập đến việc *tối ưu không gian biểu diễn đặc trưng*, tức là làm thế nào để tối ưu cả mạng nhúng và cả bộ phân lớp Softmax trong giai đoạn tinh chỉnh.

Vì vậy chương này tập trung làm rõ hai vấn đề then chốt có ảnh hưởng trực tiếp đến chất lượng của không gian biểu diễn đặc trưng trong bài toán phân lớp trôi khái niệm:

i. **Độ biến thiên trong lớp**: phản ánh mức độ phân tán của các mẫu thuộc cùng một lớp trong không gian đặc trưng. Biến thiên trong lớp cao khiến

mô hình khó xác định được ranh giới lớp chính xác, làm tăng xác suất dự đoán sai và giảm độ tin cậy [10].

ii. **Mức độ phân tách giữa các lớp**: thể hiện khả năng các lớp khác nhau được đặt cách xa nhau trong không gian đặc trưng [13]. Việc tăng khoảng cách giữa các lớp giúp giảm sự chồng lấn ranh giới và từ đó tăng khả năng phân biệt giữa các kiểu trôi, đặc biệt trong các tình huống phức tạp khi biểu hiện giữa các lớp có thể gần tương tự nhau như trôi dần dần và trôi gia tăng.



Hình 4.2 Mô hình CS&AM_SC đề xuất

Để giải quyết đồng thời hai vấn đề trên, chương đề xuất mô hình cải tiến CS&AM_SC như trong Hình 4.2. Mô hình này sử dụng chiến lược học biểu diễn đặc trưng với ba thành phần chính:

i. *Bộ phân lớp Softmax cải tiến:* thay thế tích vô hướng trong Softmax bằng độ tương đồng Cosine, kết hợp với chuẩn hóa vector đặc trưng và vector trọng số để loại bỏ ảnh hưởng của độ lớn, từ đó tập trung tối ưu hóa theo hướng vector. Cách tiếp cận này giúp giảm biến thiên trong lớp [11].

ii. *Hàm mất mát trung tâm (center loss):* thêm một ràng buộc vào quá trình tinh chỉnh nhằm tối thiểu hóa khoảng cách giữa các vector đặc trưng và tâm lớp tương ứng trong không gian biểu diễn. Việc này giúp thu hẹp sự phân tán trong lớp, từ đó tăng cường tính đồng nhất của các mẫu cùng lớp và hỗ trợ quá trình phân lớp hiệu quả hơn [50].

iii. *Biên góc:* điều chỉnh ranh giới phân lớp bằng cách áp dụng một biên trên góc giữa vector đặc trưng và vector trọng số của lớp tương ứng, nhằm tăng cường khả năng phân tách giữa các lớp trong không gian biểu diễn [47], [13].

Trong mô hình CS&AM_SC đề xuất, như minh họa ở trên, quá trình huấn luyện được chia làm hai giai đoạn:

(1) Giai đoạn tiền huấn luyện:

Ở giai đoạn này, mạng so khớp được huấn luyện trên tập dữ liệu có nhãn (gồm các luồng dữ liệu được gán kiểu trôi). Chiến lược huấn luyện theo tập nhiệm vụ được sử dụng để học biểu diễn đặc trưng phân biệt giữa các kiểu trôi. Tại thời điểm này, mô hình *chưa áp dụng* các kỹ thuật độ tương đồng Cosin, hàm mất mát trung tâm, biên góc. Mục tiêu giai đoạn này là huấn luyện một mạng so khớp có khả năng khái quát tốt, tạo tiền đề cho giai đoạn tinh chỉnh.

(2) Giai đoạn tinh chỉnh (Fine-tuning):

Sau khi kết thúc tiền huấn luyện, mạng so khớp được giữ nguyên tham số và được sử dụng làm khởi tạo ban đầu cho giai đoạn tinh chỉnh. Giai đoạn này được thực hiện trên các tập nhiệm vụ mới đại diện cho các luồng dữ liệu chưa từng thấy. Khác với MetaLDD-Finetune – vốn chỉ cập nhật bộ phân lớp Softmax – mô hình CS&AM_SC thực hiện cập nhật đồng thời cả mạng so khớp và bộ phân lớp Softmax bằng một hàm mất mát tích hợp. Cụ thể, hàm mất mát tổng được xây dựng từ ba thành phần:

i. Hàm Softmax dựa trên độ tương đồng Cosine và biên góc.
ii. Hàm mất mát trung tâm.
iii. Giữ lại thành phần điều chuẩn độ bất định trong MetaLDD-Finetune để tăng độ chắc chắn của dự đoán.

4.2 Biến thiên trong lớp và giải pháp

Trong quá trình học biểu diễn cho bài toán phân lớp kiểu trôi, một trong những thách thức lớn là làm sao đảm bảo các mẫu thuộc cùng một lớp (ví dụ: cùng là trôi đột ngột) được biểu diễn sao cho có sự đồng nhất cao trong không gian đặc trưng. Khi các mẫu trong cùng một lớp có đặc trưng không ổn định, phân tán hoặc hướng khác nhau, mô hình sẽ gặp khó khăn trong việc xác định ranh giới phân lớp rõ ràng. Hiện tượng này được gọi là biến thiên trong lớp [11].

Giả sử ta có một tập dữ liệu $D = \{(x_i, y_i)\}_{i=1}^{N}$ trong đó $\boldsymbol{x}_i$ là mẫu đầu vào và $y_i \in \{1,2,\dots,K\}$ là nhãn lớp tương ứng. Một hàm f_ϕ ánh xạ $\boldsymbol{x}_i$ thành một vector trong không gian đặc trưng $f_\phi(\boldsymbol{x}_i) \in R^d$ (gọi là vector đặc trưng), với d là số chiều của không gian biểu diễn, ϕ là tham số.

Độ biến thiên trong lớp của lớp k được định nghĩa là trung bình khoảng cách bình phương (theo chuẩn Euclid) giữa các vector đặc trưng trong lớp đó với tâm lớp tương ứng ($\boldsymbol{c}_k$), được tính như sau:

$$\mathrm{Var}_{\mathrm{intra}}(k) = \frac{1}{|S_k|}\sum_{\boldsymbol{x}_i \in S_k} ||f_\phi(\boldsymbol{x}_i) - \boldsymbol{c}_k||_2^2 \qquad (4.1)$$

Trong đó:

- $S_k \subset D$ là tập các mẫu có nhãn $y_i = k$,
- $\boldsymbol{c}_k$ tâm của lớp k, là trung bình các vector đặc trưng lớp k.
- $||.||_2$ là chuẩn Euclid bậc hai

Nếu $\mathrm{Var}_{\mathrm{intra}}(k)$ lớn, điều đó cho thấy các mẫu trong lớp k phân tán mạnh trong không gian đặc trưng, dẫn đến việc mô hình khó học được ranh giới ổn định và tổng quát cho lớp đó. Ngược lại, nếu độ biến thiên thấp, các mẫu có xu hướng hội tụ gần nhau, từ đó cải thiện khả năng phân lớp chính xác.

Trong môi trường trôi khái niệm, các lớp không chỉ mang thông tin về nhãn đầu ra, mà còn mang ý nghĩa ngữ nghĩa — ví dụ: "trôi đột ngột", "trôi dần dần", hay "trôi gia tăng". Đặc điểm của các kiểu trôi này là có thể xuất hiện dưới nhiều dạng biểu hiện khác nhau theo thời gian, có sự chuyển tiếp không rạch ròi giữa các kiểu trôi và số lượng mẫu mỗi loại thường rất hạn chế. Chính vì vậy, các mẫu thuộc cùng một kiểu trôi không đảm bảo có cùng phân phối đầu vào. Điều này dẫn đến tình trạng mô hình không thể học được một biểu diễn thống nhất cho lớp đó. Ví dụ: hai mẫu dữ liệu biểu hiện “trôi gia tăng” ở hai thời điểm khác nhau có thể có hình dạng hoặc thuộc tính khác nhau, dẫn đến vector đặc trưng của chúng bị lệch hướng hoặc nằm xa nhau trong không gian đặc trưng.

Việc không kiểm soát được độ biến thiên trong lớp có thể gây ra nhiều hệ quả bất lợi cho quá trình phân lớp. Trước hết, độ chính xác phân lớp bị suy giảm do các mẫu cùng lớp có thể bị ánh xạ gần với tâm của lớp khác, dẫn đến sai nhãn. Đồng thời, mô hình cũng dễ rơi vào tình trạng bất định, với phân phối xác suất dự đoán bị phân tán và thiếu sự tự tin. Ngoài ra, khi các lớp có biểu diễn gần nhau

trong không gian đặc trưng – chẳng hạn như trôi dần dần và trôi gia tăng – thì việc phân biệt giữa chúng trở nên khó khăn hơn. Cuối cùng, độ biến thiên trong lớp cao còn làm méo mó không gian biểu diễn và gây ra hiện tượng chồng lấn giữa các lớp, làm giảm khả năng khái quát hóa của mô hình trong các kịch bản học ít mẫu, nơi mà dữ liệu huấn luyện rất hạn chế.

Do đó, cần có các kỹ thuật cụ thể để khắc phục vấn đề này, nhằm giảm độ biến thiên trong lớp một cách trực tiếp hoặc gián tiếp. Hai kỹ thuật chủ đạo sẽ được trình bày trong các mục tiếp theo là:

(i) Thay thế tích vô hướng trong Softmax bằng độ tương đồng Cosine: giúp giảm biến thiên theo *hướng* nhờ chuẩn hóa vector.

(ii) Sử dụng hàm mất mát trung tâm: giúp giảm khoảng cách giữa các mẫu và tâm lớp.

4.2.1 Thay thế tích vô hướng bằng độ tương đồng Cosine

Như đã phân tích ở phần trước, một trong những nguyên nhân chính dẫn đến độ biến thiên trong lớp cao là do các mẫu trong cùng một lớp có thể được biểu diễn bằng các vector có hướng khác nhau trong không gian đặc trưng. Điều này thường xảy ra khi mô hình được huấn luyện với hàm mất mát truyền thống như Softmax với tích vô hướng. Trong trường hợp này, mô hình có thể tăng độ tự tin dự đoán chỉ bằng cách tăng độ lớn của vector đặc trưng, mà không đảm bảo rằng các mẫu cùng lớp thực sự hội tụ về cùng hướng - dẫn đến biểu diễn không ổn định và thiếu gắn kết. Điều này đã được chỉ ra trong nghiên cứu của W.Y Chen và cộng sự tại [11].

Độ tương đồng Cosin là một chỉ số đo mức độ tương đồng về hướng giữa hai vector, độc lập với độ lớn của chúng. Với hai vector $\boldsymbol{f}_{\phi}(\boldsymbol{x})$, $\boldsymbol{c}_k \in R^d$ độ tương đồng Cosin được định nghĩa như sau:

$$\cos(\theta_k) = \frac{\boldsymbol{f}_{\phi}(\boldsymbol{x}) \cdot \boldsymbol{c}_k}{||\boldsymbol{f}_{\phi}(\boldsymbol{x})|| \cdot ||\boldsymbol{c}_k||} \tag{4.2}$$

Trong đó:

- $\boldsymbol{f}_{\phi}(\boldsymbol{x}) \cdot \boldsymbol{c}_k$ là tích vô hướng giữa hai vector,
- $\|\cdot\|$ là chuẩn Euclid bậc hai,
- θ_k: là góc giữa hai vector $\boldsymbol{f}_{\phi}(\boldsymbol{x})$ và $\boldsymbol{c}_k$,

- $\cos(\theta_k)$: độ tương đồng Cosin giữa $\boldsymbol{f}_{\phi}(\boldsymbol{x})$ và $\boldsymbol{c}_k$.

Giá trị tương đồng Cosin nằm trong khoảng [−1,1], trong đó giá trị 1 biểu thị sự đồng hướng hoàn toàn giữa hai vector. Khi sử dụng độ tương đồng Cosin làm cơ sở phân lớp thay vì tích vô hướng, mục tiêu tối ưu của mô hình là tăng độ tương đồng về hướng giữa vector đặc trưng và tâm lớp tương ứng. Điều này đồng nghĩa với việc các mẫu cùng lớp được học để có hướng gần nhau trong không gian đặc trưng, bất kể độ lớn tuyệt đối của chúng.

Khi tích hợp độ tương đồng Cosin vào bộ phân lớp Softmax, ta thay thế tích vô hướng trong hàm Softmax bằng độ tương đồng Cosin. Khi đó, xác suất phân lớp vào lớp k được tính theo công thức:

$$p_k = \frac{\exp(\tau \cdot \cos(\theta_k))}{\sum_{j=1}^{K} \exp(\tau \cdot \cos(\theta_j))} \quad (4.3)$$

Trong đó:

- $\tau \in R^+$ là hệ số điều chỉnh nhằm khuếch đại sự phân biệt giữa các lớp,
- K là số lượng lớp.

Việc thêm hệ số τ là cần thiết vì độ tương đồng Cosin có giá trị nằm trong khoảng hẹp [−1,1], nên nếu không có τ, hàm Softmax sẽ tạo ra phân phối xác suất không rõ ràng, làm giảm hiệu quả phân biệt giữa các lớp.

Khi sử dụng độ tương đồng Cosin trong phân lớp, mô hình được huấn luyện để các đặc trưng cùng lớp có cùng hướng, bất kể độ lớn ban đầu. Điều này có một số tác động tích cực đến việc học biểu diễn [10]:

- Loại bỏ phụ thuộc vào độ lớn: vì độ tương đồng Cosin không bị ảnh hưởng bởi chuẩn của vector, các mẫu có độ lớn khác nhau (ví dụ do chênh lệch về chuẩn hóa dữ liệu đầu vào) vẫn có thể được học để nằm cùng hướng.
- Khuyến khích hướng hội tụ: mô hình tối ưu để giảm góc giữa vector đặc trưng và tâm lớp dẫn đến các mẫu cùng lớp dần hội tụ về cùng hướng.
- Tăng khả năng khái quát: vì các hướng vector là đặc trưng ổn định hơn độ lớn, mô hình sẽ học được các đặc trưng có khả năng tổng quát hóa cao hơn cho các mẫu mới.

Điều quan trọng là, dù độ tương đồng Cosin không trực tiếp giảm khoảng cách giữa các mẫu (vì không kiểm soát độ lớn vector), nhưng việc điều chỉnh hướng

biểu diễn đã tạo ra một ràng buộc ngầm mạnh, góp phần làm giảm độ biến thiên trong lớp — đặc biệt là trong biểu diễn theo góc.

Việc chuyển từ tích vô hướng sang độ tương đồng Cosin không chỉ đơn thuần là một thay đổi kỹ thuật nhỏ, mà là một thay đổi từ việc tối đa hóa giá trị đầu ra sang việc tối đa hóa sự đồng nhất về hướng trong không gian đặc trưng.

Tuy độ tương đồng Cosin giúp giảm biến thiên về hướng, nhưng không kiểm soát khoảng cách tuyệt đối giữa các mẫu trong lớp. Điều này dẫn đến tình trạng có thể có những mẫu rất gần về hướng nhưng lại cách xa nhau trong không gian Euclid. Để khắc phục điều này, cần tích hợp thêm các hàm mất mát bổ sung như hàm mất mát trung tâm, sẽ được trình bày trong mục kế tiếp.

4.2.2 Sử dụng hàm mất mát trung tâm

Như đã trình bày trong Mục 4.2.1, việc sử dụng độ tương đồng Cosin là một chiến lược hiệu quả để giảm độ biến thiên trong lớp bằng cách tối ưu hướng biểu diễn giữa các vector đặc trưng và tâm lớp. Tuy nhiên, độ tương đồng Cosin chỉ điều chỉnh góc giữa các vector đặc trưng, trong khi lại không trực tiếp kiểm soát khoảng cách tuyệt đối giữa các vector trong cùng một lớp. Điều này có thể dẫn đến một hiện tượng không mong muốn: các vector có thể cùng hướng nhưng lại nằm cách xa nhau trong không gian, gây ra khó khăn cho quá trình phân lớp – đặc biệt là khi biểu diễn được sử dụng để phân biệt những lớp gần nhau.

Để giải quyết vấn đề trên, một giải pháp bổ sung có tính trực tiếp và rõ ràng là sử dụng một hàm mất mát được thiết kế để tối thiểu hóa khoảng cách giữa các vector đặc trưng và tâm lớp tương ứng, dựa trên ý tưởng từ nghiên cứu của Y. Wen và cộng sự [50]. Đây là một trong những kỹ thuật quan trọng giúp tăng cường tính gắn kết của biểu diễn lớp trong không gian đặc trưng.

Do vậy phần này đề xuất một phương pháp kết hợp giữa mạng nguyên mẫu và hàm mất mát trung tâm nhằm giảm biến thiên trong lớp trong bối cảnh phân loại ít mẫu. Ý tưởng chính là tận dụng khả năng phân loại theo hướng của mạng nguyên mẫu, đồng thời bổ sung một ràng buộc thông qua hàm mất mát trung tâm để thúc đẩy các biểu diễn cùng lớp hội tụ về một vị trí trung tâm ổn định trong không gian đặc trưng. Phương pháp được triển khai trong các vòng huấn luyện theo tập nhiệm vụ.

Trong bối cảnh huấn luyện theo tập nhiệm vụ của mạng nguyên mẫu, không gian đặc trưng đề cập đến không gian nhúng R^d do hàm nhúng $f_\phi(.)$ tạo ra. Với mỗi lớp $k \in 1,2,\dots,K$, ta định nghĩa một *vector trung tâm lớp* k là $\boldsymbol{\mu}_k \in R^d$ là tham số có thể học được, và được cập nhật sau mỗi vòng huấn luyện thông qua cơ chế trung bình động:

$$\boldsymbol{\mu}_k^{(t+1)} = \boldsymbol{\mu}_k^{(t)} - \alpha \cdot \Delta\boldsymbol{\mu}_k \tag{4.4}$$

Với:

$$\Delta\boldsymbol{\mu}_k = \frac{1}{1+N_k}\left(\boldsymbol{\mu}_k - \overline{\boldsymbol{f}_k}\right) \tag{4.5}$$

$$\overline{\boldsymbol{f}_k} = \frac{1}{N_k}\sum_{i:\, y_i=k} \boldsymbol{f}_\phi(\boldsymbol{x}_i) \tag{4.6}$$

- $\boldsymbol{\mu}_k^{(t)}$: vector trung tâm lớp ở vòng huấn luyện thứ t (hiện tại)
- $\boldsymbol{\mu}_k^{(t+1)}$: vector trung tâm lớp ở vòng huấn luyện thứ $t+1$ (kế tiếp)
- $\Delta\boldsymbol{\mu}_k$ là trung bình các giá trị nhúng thuộc lớp *k* trong cả tập tập huấn luyện và tập kiểm tra,
- $\overline{\boldsymbol{f}_k}$ là trung bình các vector nhúng của các mẫu thuộc lớp k trong cả tập tập huấn luyện và tập kiểm tra,
- N_k là số lượng mẫu thuộc lớp k trong cả tập huấn luyện và tập kiểm tra,
- α là hệ số tốc độ học riêng cho vector trung tâm lớp,

Cơ chế cập nhật này đảm bảo rằng vector trung tâm lớp không bị cập nhật đột ngột mà được điều chỉnh dần dần, từ đó tạo ra một vector trung tâm lớp ổn định, đại diện tốt cho toàn bộ phân phối lớp trong không gian biểu diễn.

Lưu ý:

- Tâm lớp $\boldsymbol{c}_k$ được tính bằng trung bình của các vector đặc trưng của các mẫu trong tập huấn luyện thuộc lớp *k* trong mỗi vòng huấn luyện theo tập nhiệm vụ. Giá trị này được dùng trong phân loại dựa trên độ tương đồng Cosin, vì nó phụ thuộc vào dữ liệu trong từng vòng nên được tính lại liên tục và không phải là tham số được huấn luyện.
- Ngược lại, vector trung tâm của lớp ($\boldsymbol{\mu}_k$) là một tham số học được, đại diện cho vị trí trung tâm dài hạn của lớp *k* trong không gian nhúng [50]. Nó được cập nhật dần bằng chiến lược trung bình động (*Moving Average*), và được

sử dụng riêng trong hàm mất mát trung tâm nhằm giảm sự biến thiên trong lớp. Mặc dù cả hai đều được xem là "trung tâm" của một lớp, nhưng chúng khác nhau về mục đích sử dụng, cơ chế cập nhật và vai trò trong quá trình huấn luyện.

Hàm mất mát trung tâm được định nghĩa như sau:

$$\mathcal{L}_{\text{center-loss}} = \frac{1}{2Z}\sum_{i=1}^{Z}||\boldsymbol{f}_{\phi}(\boldsymbol{x}_i) - \boldsymbol{\mu}_{y_i}||_2^2 \quad (4.7)$$

Trong đó:

- Z là số lượng mẫu trong tập tập huấn luyện và tập kiểm tra,
- $\boldsymbol{x}_i$ là mẫu đầu vào thứ i, y_i là nhãn lớp tương ứng,
- $\boldsymbol{\mu}_{yi}$ là vector trung tâm lớp tương ứng với lớp y_i,

Hàm mất mát này phạt các biểu diễn đặc trưng nằm xa vector trung tâm lớp $\boldsymbol{\mu}$ tương ứng, đồng thời thúc đẩy tất cả các vector trong cùng một lớp hội tụ về một vị trí trung tâm trong không gian biểu diễn. Khác với độ tương đồng Cosin chỉ điều chỉnh hướng của vector, hàm mất mát trung tâm tập trung vào khoảng cách tuyệt đối, giúp đảm bảo rằng các vector cùng lớp không chỉ có hướng đồng nhất mà còn gần nhau.

Như vậy, hàm mất mát trung tâm có hai vai trò then chốt:

- Giảm độ phân tán trong lớp: bằng cách trực tiếp tối thiểu hóa khoảng cách giữa các giá trị nhúng và vector trung tâm lớp, mất mát trung tâm giới hạn độ lệch của mẫu cùng lớp. Điều này làm giảm rõ rệt độ biến thiên trong lớp – vốn là một yếu tố gây nhiễu trong phân lớp.
- Tăng tính gắn kết: các giá trị nhúng không chỉ được điều chỉnh về hướng mà còn được hội tụ về vị trí — các vector đặc trưng hội tụ về một vị trí, dễ dàng nhận biết trong không gian.

Khi kết hợp với độ tương đồng Cosin (điều chỉnh hướng) hàm mất mát trung tâm đóng vai trò kiểm soát cấu trúc cục bộ của lớp, đảm bảo vector mỗi lớp không chỉ hội tụ về cùng hướng mà còn nằm gần về vị trí.

4.3 Chồng lấn giữa các lớp và giải pháp

Trong học máy, khả năng phân biệt giữa các lớp phụ thuộc chặt chẽ vào cách mà các lớp được tổ chức trong không gian biểu diễn [47]. Một mô hình phân loại hiệu quả không chỉ cần đảm bảo các mẫu trong cùng một lớp được biểu diễn nhất

quán mà còn phải phân tách rõ ràng các lớp khác nhau để tránh chồng lấn và sai nhãn, đặc biệt ở những khu vực biên. Lấy cảm hứng từ các nghiên cứu của J. Deng và cộng sự [13], H. Wang và cộng sự [47] về biên góc, phần này phân tích sâu về cơ chế hình thành và tối ưu hóa sự phân tách giữa các lớp trong không gian biểu diễn.

Giả sử ta có hai lớp k và l với các mẫu gần vùng biên.

Khoảng cách giữa hai lớp k và l (dựa trên tâm lớp):

$$d_{kl} = ||c_k - c_l||_2 \tag{4.8}$$

Trung bình độ phân tách giữa các lớp:

$$\text{Sep}_{\text{inter}} = \frac{2}{K(K-1)} \sum_{k<l} d_{kl} \tag{4.9}$$

Giá trị Sep_{inter} càng lớn thì các lớp càng được phân tách rõ ràng và mô hình càng dễ học ranh giới phân loại.

Có hai loại phân tách chính trong không gian đặc trưng là phân tách theo khoảng cách Euclid và phân tách theo góc. Phân tách theo khoảng cách dựa trên vị trí tuyệt đối của các lớp và được tối ưu thông qua hàm mất mát trung tâm. Phân tách theo góc dựa trên hướng giữa vector đặc trưng và tâm lớp và được tối ưu thông qua độ tương đồng Cosin và biên góc. Dạng phân tách này phù hợp với không gian biểu diễn được chuẩn hóa.

Trong mô hình đề xuất của luận án, hai loại phân tách này được tối ưu song song: hàm mất mát trung tâm đảm bảo các tâm lớp không chồng lấn trong không gian biểu diễn, trong khi biên góc đảm bảo rằng các lớp được phân biệt rõ.

Biên góc được đưa vào công thức (4.3) bằng cách thêm một giá trị m vào góc θ như sau:

$$p_k = \frac{\exp(\tau \cdot \cos(\theta_k + m))}{\sum_{j=1}^{K} \exp(\tau \cdot \cos(\theta_j))} \tag{4.10}$$

Với $m > 0$, giá trị cosine sẽ giảm đi vì $cos(\theta + m) < cos(\theta)$. Điều này buộc mô hình phải giảm góc giữa $\boldsymbol{f}_{\phi}(\boldsymbol{x})$ và $\mathbf{c}_k$ xuống hơn nữa để yêu cầu mô hình phải hội tụ chặt hơn mới đạt được xác suất cao.

Biên góc là một kỹ thuật được áp dụng trong không gian đặc trưng đã được chuẩn hóa để tăng cường khả năng phân tách giữa các lớp. Trong Softmax thông thường, một mẫu được gán nhãn dựa trên việc có góc nhỏ nhất (tức là độ tương

đồng Cosine lớn nhất) với trọng số của lớp tương ứng. Tuy nhiên, khi áp dụng biên góc, mẫu đó không chỉ cần có góc nhỏ hơn các lớp còn lại, mà còn phải đủ nhỏ hơn một ngưỡng cụ thể – tức là gần hơn về mặt góc với trọng số của lớp đúng, nhằm tạo ra khoảng cách an toàn giữa các lớp. Điều này giúp cải thiện rõ rệt khả năng phân biệt trong các không gian biểu diễn có lớp chồng lấn [47], [13].

Khi áp dụng vào phân lớp kiểu trôi, biên góc tạo ra sự hiệu quả vì các kiểu trôi như trôi dần dần và gia tăng thường có biểu hiện rất giống nhau và dễ chồng lấn ở ranh giới biểu diễn. Đặc biệt, biên góc còn hỗ trợ tốt cho tính tổng quát hóa: vì mô hình học được các lớp có ranh giới rõ ràng và đồng nhất, nên khi gặp các mẫu mới thuộc cùng lớp (ví dụ: trôi kiểu mới nhưng vẫn là trôi dần dần), mô hình vẫn có thể phân lớp đúng nhờ cấu trúc lớp đã học rõ ràng.

Tóm lại, biên góc là một kỹ thuật hiệu quả nhằm tăng cường khả năng phân tách giữa các lớp trong không gian biểu diễn bằng cách mở rộng biên góc giữa các vector đặc trưng và trọng số lớp. Khi được tích hợp vào bộ phân lớp Softmax, biên góc thu hẹp vùng quyết định của từng lớp, làm rõ ràng ranh giới phân lớp và giảm thiểu hiện tượng chồng lấn giữa các lớp – điều này đặc biệt quan trọng trong các bài toán phân lớp kiểu trôi, nơi dữ liệu các lớp dễ có biểu hiện gần nhau về mặt hình dạng. Khi được kết hợp với các kỹ thuật như hàm mất mát trung tâm – vốn kiểm soát sự phân tán nội lớp theo khoảng cách tuyệt đối – biên góc đóng vai trò bổ sung giúp nâng cao khả năng phân biệt giữa các lớp và cải thiện độ tổng quát của mô hình trong môi trường trôi khái niệm phức tạp.

4.4. Mô hình CS&AM_SC đề xuất

4.4.1 Hàm mất mát kết hợp

Trong MetaLDD-Finetune, giai đoạn tinh chỉnh chỉ tối ưu bộ phân lớp trong khi giữ cố định mạng so khớp f_ϕ. Mặc dù điều này giúp giảm chi phí huấn luyện và tránh quá khớp, nhưng lại hạn chế khả năng điều chỉnh không gian biểu diễn đặc trưng theo các yêu cầu phân loại mới, tức là phải giảm biến thiên trong cùng một lớp và tăng được sự phân tách giữa các lớp.

Để khắc phục hạn chế này và thực sự tối ưu hóa không gian biểu diễn đặc trưng, luận án đề xuất một phương pháp tinh chỉnh: đồng thời cập nhật cả mạng so khớp f_ϕ và bộ phân lớp Softmax. Cụ thể:

- Mạng so khớp f_ϕ sẽ tiếp tục được cập nhật tham số trong quá trình tinh chỉnh.
- Bộ phân lớp Softmax sử dụng độ tương đồng Cosin và biên góc cũng được tối ưu đồng thời.

Hàm mất mát tổng hợp được thiết kế để tối ưu đồng thời cả hai mục tiêu:

$$\mathcal{L}_{total} = \mathcal{L}_{cos-softmax+margin} + \lambda \cdot \mathcal{L}_{center\text{-}loss} \quad (4.11)$$

Trong đó:

- $\mathcal{L}_{cos-softmax+margin}$: hàm mất mát sử dụng độ tương đồng Cosin và biên góc nhằm mở rộng phân tách lớp.

$$\mathcal{L}_{cos-softmax+margin} = -\log\left(\frac{\exp\left(\tau\cdot\cos\left(\theta_{y_i}+m\right)\right)}{\sum_{j=1}^{K}\exp(\tau\cdot\cos(\theta_j))}\right) \quad (4.12)$$

- $\mathcal{L}_{center\text{-}loss}$: thành phần giảm khoảng cách giữa các mẫu và tâm lớp, từ đó làm giảm biến thiên trong lớp và tăng tính gắn kết biểu diễn, được mô tả ở công thức 4.7
- λ: hệ số điều chỉnh ảnh hưởng của mất mát trung tâm.

Hàm mất mát tổng hợp chỉ được áp dụng trong giai đoạn tinh chỉnh để cập nhật đồng thời cả mạng so khớp và bộ phân lớp.

4.4.2 Chiến lược huấn luyện

Phần này mô tả chi tiết quá trình tinh chỉnh, được thực hiện sau khi hoàn tất tiền huấn luyện mạng so khớp.

Đầu vào: • Tập huấn luyện *D_train* gồm nhiều kiểu trôi có nhãn. • Số lượng vòng huấn luyện T. • Các siêu tham số: τ, m, λ, α
Đầu ra: • Mạng so khớp đã tinh chỉnh f_ϕ • Bộ phân lớp Cosine Softmax đã huấn luyện
Begin 1. Khởi tạo tham số mạng so khớp f_ϕ thông qua tiền huấn luyện trên *D_train*

2. Khởi tạo vector trung tâm lớp:

3. $\forall k \in \{1, \ldots, K\}$, đặt $\boldsymbol{\mu}_k \leftarrow 0$

3. **for** t = 1 → T **do**

4. Sinh một tập huấn luyện gồm:

- Tập huấn luyện S_k
- Tập kiểm tra Q_k
- $D\_batch \leftarrow S_k \cup Q_k$

5. Tính vector đặc trưng cho tất cả mẫu:

$\forall x_i \in D\_batch$: Chuẩn hóa $f_\phi(\boldsymbol{x}_i)$

6. Tính tâm $\boldsymbol{c}_k$ của mỗi lớp:

$$\forall k, \ \mathbf{c}_k \leftarrow \frac{1}{|S_k|}\sum_{x_i \in S_k} f_\phi(\boldsymbol{x}_i)$$

7. Khởi tạo trọng số Softmax:

$$\forall k, \ \boldsymbol{w}_k \leftarrow \boldsymbol{c}_k$$

8. Tính loss cosine softmax có margin: $\mathcal{L}_{cos-softmax+margin}$

$\forall x_i \in D\_batch$:

$$\theta_j = góc\ giữa\ (f_\phi(\boldsymbol{x}_i), \boldsymbol{w}_j)$$

$$\mathcal{L}_{cos-softmax+margin} \leftarrow -\log\left(\frac{\exp\left(\tau\cdot\cos\left(\theta_{y_i}+m\right)\right)}{\sum_{j=1}^{K}\exp(\tau\cdot\cos(\theta_j))}\right)$$

9. Tính loss trung tâm:

$$\mathcal{L}_{\text{center-loss}} \leftarrow \frac{1}{2Z}\sum_{i=1}^{Z}||f_\phi(\boldsymbol{x}_i) - \boldsymbol{\mu}_{y_i}||_2^2$$

10. Tổng hợp hàm mất mát:

$$\mathcal{L}_{total} \leftarrow \mathcal{L}_{cos-softmax+margin} + \lambda \cdot \mathcal{L}_{\text{center-loss}}$$

11. Lan truyền ngược:

- Cập nhật tham số của mạng so khớp f_ϕ theo $\nabla_\phi \mathcal{L}_{total}$
- Cập nhật trọng số w_k của bộ phân lớp Softmax

12. Cập nhật vector trung tâm lớp:

$\forall k$:

$$\overline{f_k} \leftarrow \frac{1}{N_k}\sum_{i:\, y_i=k} f_\phi(\boldsymbol{x}_i)$$

$$\Delta\mu_k \leftarrow \frac{1}{1+N_k}\left(\boldsymbol{\mu}_k - \overline{\boldsymbol{f}_k}\right)$$

$\boldsymbol{\mu}_k \leftarrow \boldsymbol{\mu}_k - \alpha \cdot \Delta\boldsymbol{\mu}_k$

14. **end for**

15. Trả về f_ϕ, bộ phân lớp Softmax

End

Bảng 4.1 là các siêu tham số của mô hình. Việc lựa chọn các giá trị này cần dựa vào tập kiểm thử, kết hợp phân tích độ chính xác, độ chắc chắn của mô hình.

Bảng 4.1. Vai trò các siêu tham số trong mô hình CS&AM_SC

Siêu tham số	Vai trò
λ	Cân bằng giữa mất mát hàm softmax và mất mát trung tâm
m	Mức độ thu hẹp vùng chấp nhận lớp
τ	Làm rõ biên phân lớp
α	Tốc độ học cho vector trung tâm lớp

Việc kết hợp các thành phần huấn luyện một cách có hệ thống cho phép mô hình không chỉ học được biểu diễn chính xác cho từng lớp, mà còn tối ưu hóa toàn bộ cấu trúc không gian đặc trưng theo hai hướng: giảm biến thiên trong lớp, tăng và mở rộng phân tách giữa các lớp. Đây là nền tảng vững chắc cho việc phát triển các hệ thống nhận diện kiểu trôi có khả năng thích ứng cao, chính xác và ổn định trong các môi trường dữ liệu động.

4.5 Thực nghiệm và đánh giá

4.5.1 Mục tiêu và kịch bản thực nghiệm

Mục tiêu của thực nghiệm: đánh giá hiệu năng giữa mô hình MetaLDD-Finetune và mô hình CS&AM_SC.

Kịch bản thực nghiệm: luận án sử dụng hai bộ dữ liệu để tiến hành thực nghiệm. Trên mỗi bộ dữ liệu, tiến hành chạy thử nghiệm cả hai mô hình với các kiến trúc mạng so khớp khác nhau (FCN, FAN, FNN) để kiểm chứng khả năng tối ưu hóa không gian biểu diễn đặc trưng trong CS&AM_SC, so sánh với MetaLDD-Finetune.

4.5.2 Thiết lập thực nghiệm

- Phần cứng: máy tính Dell PowerEdge T40, CPU Intel Xeon E-2224G 3.5 Ghz, 32GB RAM, 2TB HDD.
- Phần mềm: hệ điều hành Windows 10, Pycharm IDE Community, Python 3.11.
- **Bộ dữ liệu**
    - Bộ dữ liệu thứ nhất: Dataset_200-25-3800
    - Bộ dữ liệu thứ hai: Dataset_50-50-4800

  Mỗi bộ dữ liệu được chia thành ba phần: tập huấn luyện, tập tinh chỉnh, và tập kiểm tra theo tỷ lệ 60–20–20.
- **Cấu hình mô hình**

Trong luận án này, ba kiến trúc mạng so khớp gồm FNN (Feedforward Neural Network), FCN (Fully Convolutional Network) và FAN (Feature Attention Network) được lựa chọn cho quá trình đánh giá, bởi các lý do sau:

Trước hết, việc lựa chọn ba kiến trúc này không phải ngẫu nhiên, mà kế thừa trực tiếp từ các công trình nghiên cứu có liên quan, điển hình là hai mô hình phân lớp trôi khái niệm là Meta-ADD và Type-LDD. Cả hai mô hình này đều sử dụng FNN, FCN và FAN như là ba kiến trúc nền tảng để trích xuất đặc trưng, từ đó huấn luyện mạng nguyên mẫu nhằm phân lớp các kiểu trôi khác nhau.

Thứ hai, việc sử dụng đồng thời nhiều kiến trúc còn có mục tiêu kiểm định độ ổn định và mức độ tương thích của mô hình đề xuất CS&AM_SC với các chiến lược biểu diễn đặc trưng khác nhau:

- FNN đại diện cho mô hình đơn giản, có khả năng tổng quát cao nhưng dễ bị ảnh hưởng bởi nhiễu.
- FCN có ưu thế trong việc xử lý dữ liệu có cấu trúc chuỗi nhờ khả năng trích xuất đặc trưng cục bộ qua tích chập.
- FAN có khả năng tập trung vào các đặc trưng quan trọng thông qua cơ chế chú ý, đặc biệt hiệu quả trong môi trường có biến đổi phức tạp.

Do vậy, việc sử dụng cả ba kiến trúc này không chỉ giúp duy trì tính nhất quán với các mô hình nghiên cứu trước, mà còn tạo điều kiện để phân tích hiệu năng của mô hình đề xuất trên nhiều điều kiện khác nhau.

Hệ số điều chỉnh τ được khởi tạo bằng 1 và biên góc m được đặt bằng 0.1.

- **Chiến lược huấn luyện và tinh chỉnh**
  - Trong giai đoạn tiền huấn luyện: thực hiện huấn luyện theo phương pháp huấn luyện theo tập nhiệm vụ, trên tập dữ liệu huấn luyện. Trong giai đoạn này *không* áp dụng các kỹ thuật thay thế độ tương đồng Cosin trong hàm Softmax, hàm mất mát trung tâm, biên góc.
  - Trong giai đoạn tinh chỉnh: thực hiện tinh chỉnh cả mạng so khớp và bộ phân lớp Softmax, áp dụng các kỹ thuật để làm giảm biến thiên trong lớp và phân tách giữa các lớp, với hàm mất mát kết hợp.

4.5.3 Các độ đo hiệu năng

Luận án sử dụng các độ đo hiệu năng là độ chính xác, điểm F1 và thời gian dự đoán.

4.5.4 Trình bày và phân tích kết quả

Bảng 4.2 so sánh hiệu năng giữa hai mô hình MetaLDD-Finetune và CS&AM_SC trên hai tập dữ liệu (Dataset_200-25-3800 và Dataset_50-50-4800) với 4 kiểu trôi khái niệm và 3 cấu trúc mạng so khớp (FCN, FAN, FNN). Các Hình 4.3 và 4.4 biểu diễn Bảng 4.3 dưới dạng biểu đồ cột.

Mô hình đề xuất CS&AM-SC cho độ chính xác cao hơn do với mô hình gốc MetaLDD-Finetune ở **22/24** trường hợp, ngoại trừ hai trường hợp phân lớp trôi đột ngột FAN và trôi dần dần với FCN.

Bảng 4.2. Độ chính xác phân lớp theo từng kiểu trôi

Kiểu trôi	Cấu trúc mạng so khớp	Dataset_200-25-3800		Dataset_50-50-4800	
		MetaLDD-Finetune	CS&AM_SC	MetaLDD-Finetune	CS&AM_SC
Đột ngột	FCN	0.6233	**0.6547**	0.9510	**0.9513**
	FAN	0.6819	**0.6929**	**0.9504**	0.9213
	FNN	0.6833	**0.6858**	0.9221	**0.9239**
Dần dần	FCN	0.9571	**0.9588**	**0.6808**	0.6643
	FAN	0.9345	**0.9448**	0.9102	**0.9215**

	FNN	0.9242	**0.9455**	0.9243	**0.9334**
Gia tăng	FCN	0.8987	**0.9044**	0.2330	**0.4431**
	FAN	0.9173	**0.9256**	0.8765	**0.9063**
	FNN	0.9402	**0.9534**	0.8812	**0.9067**
Bình thường	FCN	0.9723	**0.9769**	0.9523	**0.9549**
	FAN	0.9514	**0.9656**	0.9475	**0.9488**
	FNN	0.9678	**0.9690**	0.9625	**0.9653**

Trôi đột ngột

Trên cả hai tập dữ liệu, mô hình CS&AM_SC đạt kết quả tương đương hoặc cao hơn một chút so với MetaLDD-Finetune với hầu hết các cấu trúc mạng so khớp:

- Trên Dataset_200-25-3800, CS&AM_SC vượt trội hơn rõ rệt với mạng FCN và FAN, tuy nhiên gần như tương đương với mạng FNN (0.6858 so với 0.6833).
- Trên Dataset_50-50-4800, hai mô hình cho kết quả gần như ngang bằng với mạng FCN (0.9513 so với 0.9510), mạng FNN (0.9239 so với 0.9221). Riêng với mạng FAN, CS&AM_SC thấp hơn 0.9213 so với 0.9504.

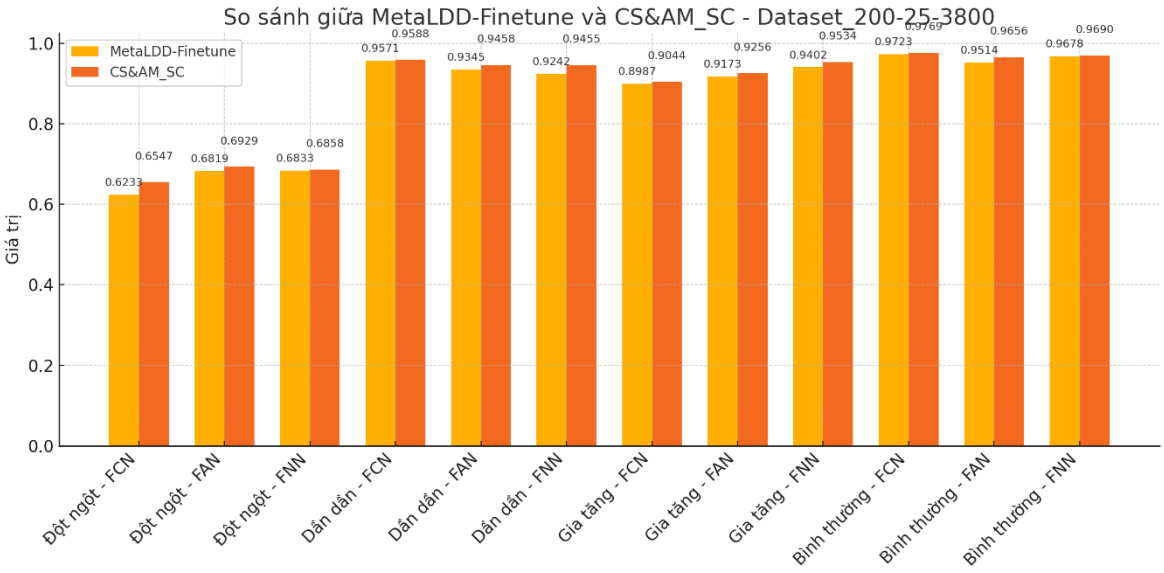
Điều này cho thấy mô hình đề xuất có khả năng duy trì độ chính xác tốt đối với các thay đổi đột ngột trong luồng dữ liệu. Tuy nhiên CS&AM_SC có cải thiện khi dùng mạng FCN, và FAN nhưng chưa ổn định khi dùng mạng FNN trong môi trường có trôi đột ngột.

Trôi dần dần

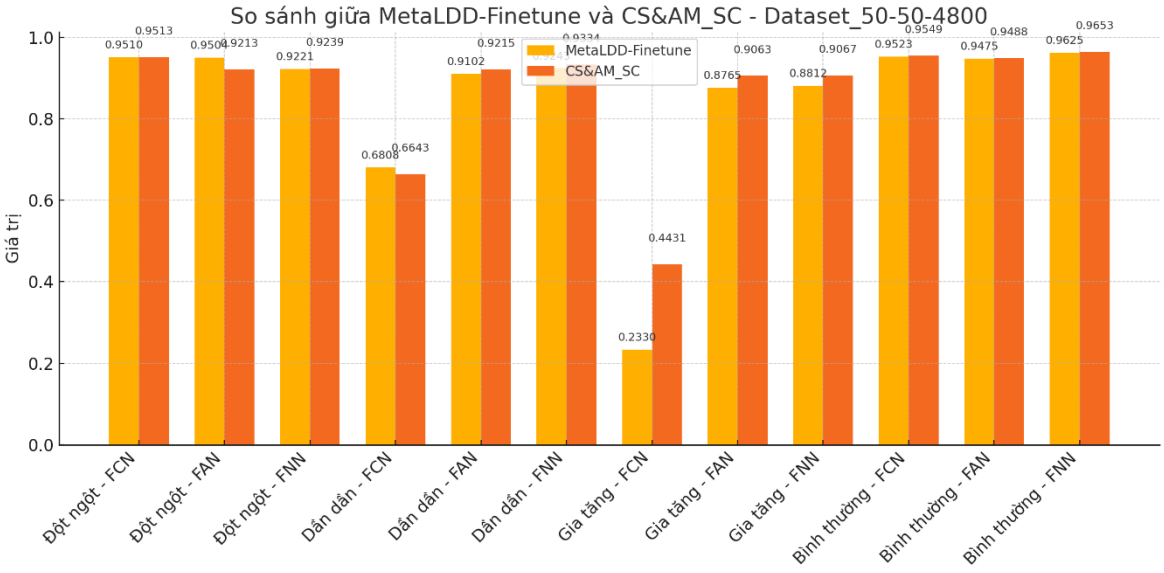
Trôi dần dần thường khó phát hiện hơn do sự thay đổi mượt của phân phối dữ liệu. Mô hình CS&AM_SC cho thấy sự vượt trội rõ rệt trong các tình huống này:

- Với mạng FAN và FNN, độ chính xác trên Dataset_200-25-3800 tăng tương ứng từ 0.9345 lên 0.9458 và 0.9242 lên 0.9455.
- Trên Dataset_50-50-4800, với mạng FNN, kết quả tăng từ 0.9243 lên 0.9334. Tuy nhiên với mạng FCN, CS&AM_SC lại cho kết quả thấp hơn: $0.6643 < 0.6808$.

Điều này phản ánh khả năng nắm bắt các thay đổi mờ dần trong khái niệm của CS&AM_SC hiệu quả hơn. Hình 4.3 và 4.4 là biểu đồ so sánh độ chính xác phân lớp giữa hai mô hình.



Hình 4.3. So sánh MetaLDD-Finetune và CS&AM_SC trên bộ 200-25-3800



Hình 4.4. So sánh MetaLDD-Finetune và CS&AM_SC trên bộ 50-50-4800

Trôi gia tăng là kiểu trôi khó nắm bắt vì các khái niệm trung gian mới xuất hiện dần vào cho đến khi khái niệm mới ổn định hoàn toàn. Đây cũng là kiểu trôi mà CS&AM_SC thể hiện ưu thế phân lớp rõ rệt nhất:

- Với mạng FCN trên Dataset_50-50-4800, MetaLDD-Finetune chỉ đạt 0.2330 trong khi CS&AM_SC đạt 0.4431 – gấp gần 2 lần.
- Với mạng FNN và FAN, mức cải thiện cũng từ 0.8765 lên 0.9063 và 0.8812 lên 0.9067.

Mô hình CS&AM_SC tỏ ra thích nghi tốt với hình thức có các khái niệm trung gian xuất hiện.

Trường hợp bình thường (không trôi)

Trong cả 6 cấu hình thử nghiệm (3 mạng × 2 tập), CS&AM_SC đều đạt kết quả nhỉnh hơn một chút so với MetaLDD-Finetune, ví dụ:

- Với FAN: 0.9514 → 0.9656 trên Dataset_200-25-3800 và 0.9475 → 0.9488 trên Dataset_50-50-4800.

Điều này cho thấy mô hình CS&AM_SC không làm giảm hiệu năng trong điều kiện không có trôi, tức là có tính ổn định cao.

CS&AM_SC mặc dù không vượt trội hoàn toàn trong tất cả tình huống ở Dataset-50_50_4800, nhưng đã cho thấy sự cải thiện ổn định trong đa số các trường hợp thử nghiệm khác, đặc biệt là đối với kiểu trôi gia tăng và dần dần, cũng như khi sử dụng các kiến trúc mạng giàu khả năng biểu diễn như FNN. Những quan sát này củng cố tính hiệu quả và thực tiễn của mô hình đề xuất, đồng thời làm nổi bật sự cân bằng giữa độ chính xác và khả năng tổng quát hóa trên các tình huống trôi khác nhau.

Bảng 4.3 Độ chính xác trung bình và thời gian dự đoán trên bộ 200-25-3800

Cấu trúc	Trung bình của độ chính xác phân lớp		Điểm F1		Thời gian dự đoán (ms)	
	MetaLDD-Finetune	CS&AM_SC	MetaLDD-Finetune	CS&AM_SC	MetaLDD-Finetune	CS&AM_SC
FCN	0.8511	**0.8623**	0.8512	**0.8619**	1510	**1014**
FAN	0.8713	**0.8737**	0.8766	**0.8843**	793	873
FNN	0.8742	**0.8869**	0.8781	**0.8894**	1120	**726**

Bảng 4.4. Độ chính xác trung bình và thời gian dự đoán trên bộ 50-50-4800

Cấu trúc	Trung bình của độ chính xác phân lớp		Điểm F1		Thời gian dự đoán (ms)	
	MetaLDD-Finetune	CS&AM_SC	MetaLDD-Finetune	CS&AM_SC	MetaLDD-Finetune	CS&AM_SC
FCN	0.7012	**0.7367**	0.6928	**0.7332**	**597**	739
FAN	0.9159	**0.9189**	0.9198	**0.9205**	**182**	920
FNN	0.9223	**0.9276**	0.9223	**0.9251**	508	**436**

Bảng 4.3 và Bảng 4.4 trình bày kết quả trung bình độ chính xác phân lớp, điểm F1 và thời gian dự đoán của hai mô hình MetaLDD-Finetune và CS&AM_SC trên ba kiến trúc mạng so khớp khác nhau. Trên cả hai bộ Dataset_20-25-3800 và Dataset_50-50-4800, CS&AM_SC đều cho thấy sự cải thiện nhẹ về độ chính xác phân lớp và điểm F1 so với MetaLDD-Finetune, trong đó cấu trúc FNN đạt hiệu quả nổi bật nhất, với độ chính xác trung bình là 0.8869 và 0.9276, cùng điểm F1 tương ứng 0.8894 và 0.9251. Đối với cấu trúc FCN, mặc dù độ chính xác có cải thiện (0.8511 → 0.8623 trên bộ 20-25-3800) nhưng thời gian dự đoán giảm mạnh (từ 1510 ms xuống 1014 ms), cho thấy hiệu quả về mặt tính toán được tối ưu tốt hơn. Ngược lại, đối với cấu trúc FAN và FNN, thời gian dự đoán có xu hướng tăng nhẹ hoặc giữ nguyên, phản ánh chi phí tính toán bổ sung khi áp dụng các kỹ thuật như biên góc và mất mát trung tâm.

Hạn chế của CS&AM_SC

So với MetaLDD-Finetune, CS&AM_SC có thêm một số thành phần tính toán trong quá trình huấn luyện bao gồm: chuẩn hóa vector đặc trưng, tính độ tương đồng Cosine giữa các mẫu và trọng số phân lớp, áp dụng biên góc và sử dụng hàm mất mát trung tâm. Ngoài ra, mô hình còn thực hiện cập nhật động vector trung tâm lớp trong không gian đặc trưng sau mỗi vòng huấn luyện. Các phép tính này làm tăng độ phức tạp tính toán của mỗi vòng huấn luyện so với MetaLDD-Finetune trong trường hợp sử dụng số lớp và số mẫu mỗi lô tương đương. Tuy nhiên, chi phí này là chấp nhận được khi xét đến sự cải thiện rõ rệt về hiệu năng phân loại, đặc biệt với các kiểu trôi khó như trôi dần và trôi gia tăng. Cụ thể so sánh độ phức tạp tính toán được mô tả trong Bảng 4.6.

Bảng 4.5. So sánh chi phí tính toán giữa MetaLDD-Finetune và CS&AM_SC

Thành phần huấn luyện	MetaLDD -Finetune	CS&AM_SC	Độ phức tạp tăng thêm	Ghi chú
Áp dụng biên góc	Không	Có thực hiện	O(Z)	Một phép toán độ tương đồng Cosin + biên góc cho từng mẫu đúng lớp
Hàm mất mát Cosine Softmax	Có	Có thực hiện và mở rộng thêm thành phần biên góc	O(ZKd)	CS&AM dùng phiên bản mở rộng có biên góc
Hàm mất mát trung tâm	Không	Có thực hiện	O(Zd)	Khoảng cách L2 giữa đặc trưng và tâm lớp
Cập nhật vector trung tâm lớp	Không	Có thực hiện	O(KNd)	Trung bình đặc trưng từng lớp rồi cập nhật vector trung tâm lớp
Tổng chi phí mỗi vòng lặp huấn luyện (D_batch)	**O(ZKd)**	**O(ZKd + KNd)**		**Phụ thuộc K, d, kích thước D_batch**

Ký hiệu trong bảng

- Z: số mẫu trong một vòng huấn luyện (theo tập nhiệm vụ)
- K: số lớp trôi
- N: số mẫu mỗi lớp

- d: số chiều không gian nhúng
- T: số vòng huấn luyện
- D_batch là tập gồm các mẫu hỗ trợ và các mẫu kiểm tra của một vòng huấn luyện (mô tả trong bước 4 của chiến lược huấn luyện Mục 4.4.2)

Qua phân tích ở Bảng 4.6, CS&AM_SC có độ phức tạp tính toán cao hơn $O(ZKd + KNd)$ so với MetaLDD-Finetune $O(ZKd)$. Tuy nhiên, chi phí tăng thêm này là có kiểm soát và được bù đắp bằng hiệu quả phân loại cao hơn trong hầu hết các tình huống thử nghiệm. Tuy nhiên, do phương pháp chỉ thực hiện trên số lượng mẫu ít trong mỗi vòng lặp như đã đề cập trong Phần chiến lược huấn luyện mục 4.4.2, bước số 4, nên chi phí tinh chỉnh thực tế vẫn ở mức chấp nhận được.

Bên cạnh đó, các kỹ thuật bổ sung không làm thay đổi kiến trúc mạng so khớp, mà chủ yếu ảnh hưởng đến bộ phân lớp, vốn có quy mô nhỏ. Do đó, yêu cầu tài nguyên (bộ nhớ, CPU) không tăng đáng kể, vẫn phù hợp với môi trường tính toán tiêu chuẩn (máy trạm CPU phổ thông với cấu hình đã nêu ở Phần 4.5.2).

Tổng thể, mô hình CS&AM_SC mang lại sự cải thiện ổn định về hiệu năng phân lớp mà không làm gia tăng thời gian đáng kể, chứng minh hiệu quả của việc tối ưu không gian biểu diễn đặc trưng thông qua giảm biến thiên trong lớp và tăng cường phân tách giữa các lớp.

4.6 Kết luận chương 4

Chương này trình bày đề xuất cải tiến mô hình MetaLDD-Finetune để có mô hình CS&AM_SC (Cosine Similarity and Angular Margin based Softmax Classifier), trong đó phân tích và đưa ra giải pháp tối ưu không gian biểu diễn đặc trưng trong mạng nguyên mẫu, sử dụng: (i) độ tương đồng Cosine thay cho sử dụng tích vô hướng; (ii) biên góc và (iii) mất mát trung tâm, nhằm giảm biến thiên trong lớp và phân tách giữa các lớp rõ ràng. Kết quả nghiên cứu theo nhóm đóng góp này của luận án được công bố trong [TungNK4].

Kết luận

Luận án này hướng tới việc phát triển những phương pháp nhằm nâng cao độ tin cậy trong việc phát hiện và phân lớp trôi khái niệm trên luồng dữ liệu. Các kết quả thực nghiệm và phân tích cho thấy tính hiệu quả của các phương pháp đề xuất.

Các kết quả đạt được

Luận án đã đóng góp ba hướng chính cho lĩnh vực phát hiện và phân lớp trôi khái niệm trong dữ liệu luồng như được mô tả sau đây:

Về phát hiện trôi khái niệm

Mô hình học kết hợp **E-ERICS** tận dụng ưu điểm của các mô hình đơn lẻ trong việc giảm các trường hợp dương tính giả, đồng thời bổ sung thêm các điểm dương tính thực. Nhờ đó, mô hình này cải thiện điểm F1 trung bình khoảng **4%** trên các bộ dữ liệu SEA và KDD, và đạt kết quả tốt hơn trong hầu hết các tiêu chí đánh giá so với chỉ sử dụng pha BO, đặc biệt trong việc phát hiện trôi khái niệm đột ngột.

Mô hình **ERICS+3** cũng cho thấy sự cải thiện đáng kể, với mức tăng trung bình khoảng **6,25% về độ hồi tưởng**, **2,4% về điểm F1**, và cải thiện nhẹ về độ chính xác trong việc phát hiện trôi dần dần. Trong một số trường hợp cụ thể, ERICS+3 vượt trội hơn cả E-ERICS và ERICS.

Về phân lớp kiểu trôi khái niệm

Mô hình **VAR-WIND** cho thấy chiến lược cửa sổ linh hoạt luôn đạt hiệu năng cao hơn so với chiến lược cửa sổ rời rạc, khẳng định khả năng thích ứng và phản ứng nhanh với sự thay đổi của luồng dữ liệu. Điều này góp phần nâng cao độ chính xác trong cả phân lớp và phát hiện trôi khái niệm. Mô hình **MetaLDD-Finetune**, bổ sung kỹ thuật tinh chỉnh, không chỉ duy trì hoặc nâng cao độ chính xác trong phát hiện trôi, mà còn giúp giảm đáng kể thời gian dự đoán. Bên cạnh đó, mô hình **CS_AM&SC** cho thấy hiệu năng vượt trội về độ chính xác và điểm F1 trên tất cả các cấu hình mạng so khớp. Mặc dù thời gian dự đoán có dao động tuỳ theo kiến trúc mạng, phương pháp vẫn duy trì hiệu năng cạnh tranh hoặc cao hơn so với MetaLDD-Finetune, cho thấy tính hiệu quả trong việc nâng cao phân loại kiểu trôi đồng thời tối ưu tốc độ.

Hạn chế

Luận án đã đạt được một số kết quả nghiên cứu bước đầu trong phát hiện và phân lớp trôi khái niệm. Tuy nhiên, luận án còn một số hạn chế như sau:

i) Thứ nhất, trong mô hình E-ERICS và ERICS+3, mặc dù luận án đã sử dụng nhiều bộ dữ liệu khác nhau để kiểm chứng phương pháp, các thử nghiệm vẫn chưa bao phủ đầy đủ các tình huống khó, đặc biệt là trong các môi trường có nhiễu cao (ví dụ: dữ liệu chứa nhiều ngoại lai, hoặc các hiện tượng trôi mờ nhạt và không rõ ràng). Đây là những điều kiện thường gặp trong thực tế và có thể ảnh hưởng đáng kể đến hiệu năng của hệ thống, do đó cần được đưa vào phạm vi đánh giá trong các nghiên cứu tiếp theo.

ii) Thứ hai, phương pháp ở mô hình CS&AM_SC sử dụng kiến trúc mạng nguyên mẫu để thực hiện phân lớp kiểu trôi, mà chưa tiến hành đánh giá so sánh với các kiến trúc học từ ít mẫu phổ biến khác như Matching Networks, Siamese Networks hoặc Relation Networks. Việc mở rộng so sánh thực nghiệm với các kiến trúc này có thể giúp làm rõ hiệu quả tương đối của mô hình được đề xuất.

iii) Các mô hình đề xuất trong luận án đều là các đề xuất mang tính kỹ thuật, kết thừa từ các mô hình sẵn có mà chưa đề xuất được một mô hình khái quát dựa trên một khung kỹ thuật chung với các lý giải chặt chẽ về mặt lý thuyết để áp dụng vào các bài toán phát hiện và phân lớp trôi khái niệm.

Định hướng nghiên cứu tiếp theo

Nghiên cứu sinh đang tiến hành các nghiên cứu mở rộng và hoàn thiện các mô hình đã được đề xuất trong luận án, với trọng tâm là khắc phục các hạn chế đã chỉ ra. Cụ thể, hai hướng nghiên cứu đang được triển khai như sau:

Thứ nhất, nhằm nâng cao hiệu quả phát hiện trôi khái niệm trong các điều kiện nhiễu cao và không rõ ràng – một thách thức đối với mô hình E-ERICS và ERICS+3 – nghiên cứu sinh đang phân tích phương pháp DriftLens của Greco và cộng sự. Phương pháp này sử dụng biểu diễn học sâu kết hợp với kỹ thuật phát hiện trôi không giám sát, cho phép nhận diện các thay đổi nhỏ trong phân phối dữ liệu mà không cần nhãn. Dựa trên phương pháp này, nghiên cứu sinh dự kiến áp dụng vào quy trình phát hiện trôi trong E-ERICS/ERICS+3, đồng thời tiến hành thử nghiệm trên các tập dữ liệu có chứa nhiễu mạnh và các dạng trôi mờ nhạt. Kết quả dự kiến sẽ giúp nâng cao độ nhạy và tính ổn định của mô hình trong các môi trường dữ liệu phức tạp hơn so với các thử nghiệm trước đây.

Thứ hai, để cải thiện hiệu năng phân lớp kiểu trôi khái niệm trong điều kiện học từ ít mẫu, nghiên cứu sinh đang tìm hiểu các kiến trúc học từ ít mẫu nền tảng, cụ thể là Matching Networks, Siamese Networks và Relation Networks. Nghiên cứu sinh hiện đang triển khai thực nghiệm so sánh trực tiếp với mạng nguyên mẫu trong cùng điều kiện và tập dữ liệu, từ đó đánh giá hiệu quả tương đối của từng kiến trúc. Kết quả dự kiến sẽ giúp xác định mô hình tối ưu hơn cho bài toán phân lớp kiểu trôi, hoặc gợi mở các hướng kết hợp kiến trúc để nâng cao hiệu năng phân lớp trong điều kiện dữ liệu thiếu nhãn và phân bố thay đổi.

Hai hướng nghiên cứu này sẽ tiếp tục được hoàn thiện với hy vọng đóng góp vào việc phát triển một hệ thống phát hiện và phân lớp trôi khái niệm có độ chính xác cao, khả năng thích nghi tốt và phù hợp với môi trường ứng dụng thực tiễn.

Danh mục công trình khoa học của tác giả liên quan tới luận án

1. [TungNK1]. Nguyen, K. T., Tran, T., Nguyen, A. D., Phan, X. H., & Ha, Q. T. (2022, November). *Parameter distribution ensemble learning for sudden concept drift detection.* In Asian Conference on Intelligent Information and Database Systems, pp. 192-203. Cham: Springer Nature Switzerland. **Scopus**, **DBLP**, *01 trích dẫn Scopus*.
2. [TungNK2]. Nguyen, K. T., Tran, T., Nguyen, A. D., Phan, X. H., & Ha, Q. T. (2023, November). *Concept Drift Detection in Data Stream: Ensemble Learning Method for Detecting Gradual Instances*. In 2023 Asia Meeting on Environment and Electrical Engineering (EEE-AM), pp. 01-05. **IEEE**[12] .
3. [TungNK3]. K. -T. Nguyen, Q. -T. Ha, X. -H. Phan and Q. -N. N. Han. *A Fine-Tuning Approach to Improve Concept Drift Type Classification Accuracy*, 2024 16th International Conference on Knowledge and System Engineering (KSE), Kuala Lumpur, Malaysia, 2024, pp. 01-05, doi: 10.1109/KSE63888.2024.11063551. **Scopus**, **DBLP**, **IEEE**.
4. [TungNK4]. Nguyen, K. T., Ha, Q. T., & Phan, X. H. (2024, December). *Enhancing Drift Type Classification Through Intra-class Variation Reduction.* In International Conference on Applied Mathematics and Computer Science, pp. 168-177. Cham: Springer Nature Switzerland. (First Online: 27 August 2025). **Scopus**, **DBLP**.
5. [TungNK5]. Nguyen, K. T., Ha, Q. T., & Phan, X. H. (2024, December) *Dynamic Windowing Strategies for Concept Drift Type Classification in Data Streams*. In 2024 IEEE International Conference on Progress in Informatics and Computing (PIC) (pp. 70-74). **Scopus**, **IEEE**, *01 trích dẫn Scopus*.

[12] https://ieeexplore.ieee.org/author/697470114198698

Tài liệu tham khảo

[1]. Ahmad Abbasi, Abdul Rehman Javed, Chinmay Chakraborty, Jamel Nebhen, Wisha Zehra, Zunera Jalil. *ElStream: An Ensemble Learning Approach for Concept Drift Detection in Dynamic Social Big Data Stream Learning*. IEEE Access 9: 66408-66419 (2021).

[2]. Supriya Agrahari, Anil Kumar Singh. *Concept Drift Detection in Data Stream Mining : A literature review*. J. King Saud Univ. Comput. Inf. Sci. 34(10 Part B): 9523-9540 (2022)–.

[3]. Cesare Alippi, Giacomo Boracchi, Manuel Roveri. *Hierarchical Change-Detection Tests*. IEEE Trans. Neural Networks Learn. Syst. 28(2): 246-258 (2017).

[4]. Ethem Alpaydin. *Introduction to Machine Learning (4th edition)*. The MIT Press, 2020.

[5]. Robert Anderson, Yun Sing Koh, Gillian Dobbie, Albert Bifet. *Recurring concept meta-learning for evolving data streams*. Expert Syst. Appl. 138 (2019).

[6]. Manuel Baena-Garcıa, Jose del Campo-Avila, Raul Fidalgo, Albert Bifet, Ricard Gavalda, Rafael Morales-Bueno. *Early drift detection method*. In Proc. 4th Int. Workshop Knowledge Discovery from Data Streams, 2006.

[7]. Albert Bifet, Ricard Gavaldà. *Learning from Time-Changing Data with Adaptive Windowing*. SDM 2007: 443-448.

[8]. Albert Bifet, Geoff Holmes, Bernhard Pfahringer, Philipp Kranen, Hardy Kremer, Timm Jansen, Thomas Seidl. *MOA: Massive Online Analysis, a Framework for Stream Classification and Clustering*. WAPA 2010: 44-50.

[9]. Andrés L. Suárez-Cetrulo, David Quintana, Alejandro Cervantes. *A survey on machine learning for recurring concept drifting data streams*. Expert Syst. Appl. 213(Part): 118934 (2023).

[10]. Xu Luo, Hao Wu, Ji Zhang, Lianli Gao, Jing Xu, Jingkuan Song. *A Closer Look at Few-shot Classification Again*. ICML 2023: 23103-23123.

[11]. Yinbo Chen, Zhuang Liu, Huijuan Xu, Trevor Darrell, Xiaolong Wang. *Meta-Baseline: Exploring Simple Meta-Learning for Few-Shot Learning*. ICCV 2021: 9042-9051.

[12]. Wai Chiu. *Online data stream classification in the presence of Concept Drift and Class Imbalance*. PhD thesis, University of Birmingham, UK, 2022.

[13]. Jiankang Deng, Jia Guo, Niannan Xue, Stefanos Zafeiriou. *ArcFace: Additive Angular Margin Loss for Deep Face Recognition*. CVPR 2019: 4690-4699.

[14]. Thomas G. Dietterich. *Ensemble Methods in Machine Learning*. Multiple Classifier Systems 2000: 1-15.

[15]. Lei Du, Qinbao Song, Lei Zhu, Xiaoyan Zhu. *A Selective Detector Ensemble for Concept Drift Detection*. Comput. J. 58(3): 457-471 (2015).

[16]. Gu Feng. *Concept drift detection for machine learning with stream data*. PhD Thesis, University of Technology Sydney (Australia)), 2019.

[17]. Par Maxime Fuccellaro. *Concept Drift: detection, update and correction*. PhD Thesis, Université de Bordeaux, 2024.

[18]. João Gama, Pedro Medas, Gladys Castillo, Pedro Pereira Rodrigues. *Learning with Drift Detection*. SBIA 2004: 286-295.

[19]. João Gama, Indre Zliobaite, Albert Bifet, Mykola Pechenizkiy, Abdelhamid Bouchachia. *A survey on concept drift adaptation*. ACM Comput. Surv. 46(4): 44:1-44:37 (2014).

[20]. Salvatore Greco, Bartolomeo Vacchetti, Daniele Apiletti, Tania Cerquitelli. *Unsupervised Concept Drift Detection from Deep Learning Representations in Real-time*. CoRR abs/2406.17813 (2024).

[21]. Guneet Singh Dhillon, Pratik Chaudhari, Avinash Ravichandran, Stefano Soatto. *A Baseline for Few-Shot Image Classification*. ICLR 2020.

[22]. Husheng Guo, Hai Li, Qiaoyan Ren, Wenjian Wang. *Concept drift type identification based on multi-sliding windows*. Inf. Sci. 585: 1-23 (2022).

[23]. Johannes Haug, Gjergji Kasneci. *Learning Parameter Distributions to Detect Concept Drift in Data Streams*. ICPR 2020: 9452-9459.

[24]. Johannes Haug, Martin Pawelczyk, Klaus Broelemann, Gjergji Kasneci. *Leveraging Model Inherent Variable Importance for Stable Online Feature Selection*. KDD 2020: 1478-1502.

[25]. Johannes Haug. *Towards Reliable Machine Learning in Evolving Data Streams*. PhD Thesis, University of Tübingen, Germany, 2022.

[26]. Jinpeng Li, Hang Yu, Zhenyu Zhang, Xiangfeng Luo, Shaorong Xie. *Concept Drift Adaptation by Exploiting Drift Type*. ACM Trans. Knowl. Discov. Data 18(4): 96:1-96:22 (2024).

[27]. Daniel Kifer, Shai Ben-David, Johannes Gehrke. *Detecting Change in Data Streams*. VLDB 2004: 180-191.

[28]. Jie Lu, Anjin Liu, Fan Dong, Feng Gu, João Gama, Guangquan Zhang. *Learning under Concept Drift: A Review*. CoRR abs/2004.05785 (2020).

[29]. Daniel Lukats, Oliver Zielinski, Axel Hahn, Frederic T. Stahl. *A benchmark and survey of fully unsupervised concept drift detectors on real-world data streams*. Int. J. Data Sci. Anal. 19(1): 1-31 (2025).

[30]. Stratos Mansalis, Eirini Ntoutsi, Nikos Pelekis, Yannis Theodoridis. *An evaluation of data stream clustering algorithms*. Stat. Anal. Data Min. 11(4): 167-187 (2018).

[31]. Helen McKay. *Online transfer learning for concept drifting data streams*. PhD thesis, University of Warwick, Coventry, UK, 2022.

[32]. Hassan Mehmood. *Concept drift in smart city scenarios*. PhD thesis, University of Oulu, Finland, 2023.

[33]. Alireza Ostovar. *Business Process Drift Detection and Characterization*. PhD Thesis, Queensland University of Technology, 2019.

[34]. Carl Edward Rasmussen, Christopher K. I. Williams. *Gaussian processes for machine learning*. MIT Press 2006..

[35]. Adam Santoro, Sergey Bartunov, Matthew M. Botvinick, Daan Wierstra, Timothy P. Lillicrap. *Meta-Learning with Memory-Augmented Neural Networks*. ICML 2016: 1842-1850.

[36]. Denise Maria Vecino Sato, Sheila Cristiana De Freitas, Jean Paul Barddal, Edson Emílio Scalabrin. *A Survey on Concept Drift in Process Mining*. ACM Comput. Surv. 54(9), 2022.

[37]. Denise Maria Vecino Sato. *Concept drift detection in process models*. PhD Thesis, Pontifícia Universidade Católica do Paraná, Curitiba, 2022.

[38]. Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P. Adams, Nando de Freitas. *Taking the Human Out of the Loop: A Review of Bayesian Optimization*. Proc. IEEE 104(1): 148-175 (2016).

[39]. Jake Snell, Kevin Swersky, Richard S. Zemel. *Prototypical Networks for Few-shot Learning*. NIPS 2017: 4077-4087.

[40]. Jasper Snoek, Hugo Larochelle, Ryan P. Adams. Practical *Bayesian Optimization of Machine Learning Algorithms*. NIPS 2012: 2960-2968.

[41]. Chang How Tan. *A Unified Framework for Anomaly and Concept Drift Detection with Explainability*. PhD Thesis, Monash University, 2022.

[42]. Jivka Ovtcharova Martin Trat. *Designing Concept Drift Detection Ensembles: A Survey*. DSAA 2023: 1-10.

[43]. Alexey Tsymbal. *The problem of concept drift: definitions and related work*. Computer Science Department, Trinity College Dublin, 106.2: 58, 2004.

[44]. Juliane Verwiebe, Philipp M. Grulich, Jonas Traub, Volker Markl. *Survey of window types for aggregation in stream processing systems*. VLDB J. 32(5): 985-1011 (2023).

[45]. Oriol Vinyals, Charles Blundell, Tim Lillicrap, Koray Kavukcuoglu, Daan Wierstra. *Matching Networks for One Shot Learning*. NIPS: 3630-3638, 2016.

[46]. Pingfan Wang. *Concept drift detection using machine learning in data stream*. PhD Thesis, Northumbria University, 2024.

[47]. Hao Wang, Yitong Wang, Zheng Zhou, Xing Ji, Dihong Gong, Jingchao Zhou, Zhifeng Li, Wei Liu. *CosFace: Large Margin Cosine Loss for Deep Face Recognition*. CVPR 2018: 5265-5274.

[48]. Pingfan Wang, Hang Yu, Nanlin Jin, Duncan Davies, Wai Lok Woo. *QuadCDD: A Quadruple-based Approach for Understanding Concept Drift in Data Streams*. Expert Syst. Appl. 238(Part E): 122114, 2024.

[49]. Geoffrey I. Webb, Roy Hyde, Hong Cao, Hai-Long Nguyen, François Petitjean. *Characterizing concept drift*. Data Min. Knowl. Discov. 30(4): 964-994, 2016.

[50]. Yandong Wen, Kaipeng Zhang, Zhifeng Li, Yu Qiao. *A Discriminative Feature Learning Approach for Deep Face Recognition*. ECCV (7) 2016: 499-515.

[51]. Gerhard Widmer, Miroslav Kubat. *Learning in the Presence of Concept Drift and Hidden Contexts*. Mach. Learn. 23(1): 69-101 (1996).

[52]. Qiuyan Xiang, Lingling Zi, Xin Cong and Yan Wang. *Concept Drift Adaptation Methods under the Deep Learning Framework: A Literature Review*. Appl. Sci, 13(11), 6515, 2023.

[53]. Li Yang, Abdallah Shami. *A Lightweight Concept Drift Detection and Adaptation Framework for IoT Data Streams*. IEEE Internet Things Mag. 4(2): 96-101 (2021).

[54]. Shujian Yu, Zubin Abraham. *Concept Drift Detection with Hierarchical Hypothesis Testing*. SDM 2017: 768-776.

[55]. Hang Yu, Qingyong Zhang, Tianyu Liu, Jie Lu, Yimin Wen, Guangquan Zhang. *Meta-ADD: A meta-learning based pre-trained model for concept drift active detection*. Inf. Sci. 608: 996-1009 (2022).

[56]. Hang Yu, Jinpeng Li, Jie Lu, Yiliao Song, Shaorong Xie, Guangquan Zhang. *Type-LDD: A Type-Driven Lite Concept Drift Detector for Data Streams*. IEEE Trans. Knowl. Data Eng. 36(12): 9476-9489 (2024).

[57]. En Yu. *Adaptive learning under concept drift for multiple data streams*. PhD Thesis, University of Technology Sydney, 2024.

[58]. Tong Yu, Hong Zhu. *Hyper-Parameter Optimization: A Review of Algorithms and Applications*. CoRR abs/2003.05689 (2020).

[59]. Indre Zliobaite. *Learning under Concept Drift: an Overview*. CoRR abs/1010.4784, 2010.

[60]. Zhaoyang Zhu, Zhaoyang Zhu, Weiqi Chen, YiFan Zhang, Qingsong Wen, Liang Sun. *Learning to Extrapolate and Adjust: Two-Stage Meta-Learning for Concept Drift in Online Time Series Forecasting*. In International Conference on Learning Representations (ICLR) Workshop, 2023.

[61] Jan Niklas Adams, Sebastiaan J. van Zelst, Thomas Rose, Wil M. P. van der Aalst. *Explainable concept drift in process mining*. Inf. Syst. 114: 102177 (2023).