

TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN TOÁN ỨNG DỤNG VÀ TIN HỌC



Deep learning for classification of time series

MỘT SỐ PHƯƠNG PHÁP SINH DỮ LIỆU
CHO BÀI TOÁN PHÂN LỐP CHUỖI THỜI GIAN

Chuyên ngành: TOÁN TIN

Chuyên sâu: Chuỗi thời gian

Giảng viên hướng dẫn: TS. NGUYỄN THỊ NGỌC ANH

Sinh viên thực hiện: NHÓM 03

Lớp: ATS - 129873

HÀ NỘI - 2022

NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN

1. Mục đích và nội dung của đồ án:

2. Kết quả đạt được:

3. Ý thức làm việc của sinh viên:

Hà Nội, ngày 06 tháng 02 năm 2022

Giảng viên hướng dẫn

TS. Nguyễn Thị Ngọc Anh

Lời cảm ơn

Đồ án này được hoàn thành tại trường Đại học Bách khoa Hà Nội dưới sự hướng dẫn của **TS.Nguyễn Thị Ngọc Anh**. Em xin được gửi lời cảm ơn sâu sắc tới TS. Nguyễn Thị Ngọc Anh đã tận tình hướng dẫn trong quá trình học tập và nghiên cứu.

Em trân trọng gửi lời cảm ơn đến Ban Lãnh đạo trường Đại học Bách khoa Hà Nội. Viện Toán ứng dụng và Tin học, đặc biệt là các thầy cô trong Viện Toán ứng dụng và Tin học đã luôn động viên, giúp đỡ em trong suốt thời gian học tập tại Viện và Trường.

Mặc dù đã rất cố gắng, nhưng vì điều kiện thời gian cũng như kiến thức của bản thân còn hạn chế nên đồ án không tránh khỏi những sai sót. Em rất mong muôn nhận được sự đóng góp của thầy cô để em được bổ sung kiến thức và rút kinh nghiệm ở các đồ án sau.

Tóm tắt nội dung đồ án

Bài báo cáo gồm 04 phần chính là:

- i Tổng quan về đề tài;
- ii Một số lý thuyết chuẩn bị;
- iii Các phương pháp sinh dữ liệu;
- iv Thực nghiệm và kết luận.

Hà Nội, ngày 06 tháng 02 năm 2022

Tác giả đồ án

Nhóm 03

Keyword: *Data augmentation, Deep Learning, time series classification, machine learning*

Mục lục

Danh mục thuật ngữ	5
Danh sách bảng	6
Danh sách hình vẽ	7
1 Tổng quan	8
1.1 Giới thiệu đề tài	8
1.2 Thực trạng nghiên cứu của đề tài	8
1.3 Lý do chọn đề tài	9
1.4 Đối tượng nghiên cứu	9
1.5 Phạm vi nghiên cứu đề tài	9
1.6 Mục đích nghiên cứu	9
1.7 Phương pháp nghiên cứu	9
1.8 Kết quả, ý nghĩa của việc nghiên cứu	9
2 Cơ sở lý thuyết	10
2.1 Chuỗi thời gian	10
2.2 Lý thuyết DTW	10
2.3 Nền tảng máy học	11
2.3.1 Khái niệm	11
2.3.2 Vai trò của Trí tuệ nhân tạo	12
2.3.3 Một số ứng dụng Trí tuệ nhân tạo tại Việt Nam	12
2.3.4 Phương pháp đánh giá một hệ thống phân lớp	14
2.4 Giới thiệu một số Mô hình	14
2.4.1 K - Nearest Neighbours - KNN	14
2.4.2 Logistic Regression	15
2.4.3 Gaussian Naive Bayes	15
2.4.4 Support Vector Machine - SVM	16
2.4.5 Decision Tree	17
2.4.6 Random Forest	18
2.4.7 Extra Trees Classifier	19
2.4.8 Adaptive Boosting	19
2.4.9 Gradient Boosting Classifier	20
2.4.10 eXtreme Gradient Boosting	20
2.4.11 LightGBM	21
2.4.12 Multi-Layer Perceptron Neural Networks	22
2.4.13 Visual Geometry Group	22
2.4.14 Long Short Term Memory	23
2.4.15 Long Short Term Memory with Fully Convolutional Network	24
2.4.16 Residual Network	25
3 Các phương pháp sinh dữ liệu	26
3.1 Jittering	26
3.2 Rotation	26
3.3 Scaling	27

3.4	Magnitude Warping	28
3.5	Permutation	28
3.6	Window Slice	29
3.7	Time Warping	30
3.8	Window Warping	30
3.9	SuboPtimAl Warped time series geNERatoR (SPAWNer)	31
3.10	Weighted DTW Barycentric Averaging (wDBA)	32
3.11	Random Guided Warping (RGW)	33
3.12	Discriminative Guided Warping (DGW)	34
4	Thực nghiệm và kết luận	35
4.1	Dữ liệu	35
4.2	Phương pháp thực nghiệm	37
4.3	Kết quả thực nghiệm	37
4.4	Kết luận	39
Tài liệu tham khảo		41

Danh mục thuật ngữ

STT	THUẬT NGỮ TIẾNG ANH	THUẬT NGỮ TIẾNG VIỆT	TỪ VIẾT TẮT
1	Artificial Intelligence	Trí tuệ nhân tạo	AI
2	Machine Learning	Máy học	ML
3	Accuracy	Dộ chính xác	
4	Classification	Phân lớp	
5	Label	Nhãn	
6	K - Nearest Neighbours	K - hàng xóm gần nhất	KNN
7	Logistic Regression	Hồi quy Logistic	
8	Support Vector Machine	Máy vecto hỗ trợ	SVM
9	Decision Tree	Cây quyết định	
10	Random Forest	Rừng ngẫu nhiên	
11	Deep Learning	Học sâu	
12	Neuron	Nơ-ron	
13	Long Short Term Memory	Mạng bộ nhớ dài ngắn	LSTM
14	Recurrent neural network	Mạng nơ-ron hồi tiếp	RNN
15	Window Slice	Cửa sổ trượt	
16	Suboptimal Warped time series geNeratoR		SPAWNER
17	Dynamic Time Warping		DTW
18	weighted DTW Barycentric Averaging		wDBA
19	Random Guided Warping		RGW
20	Discriminative Guided Warping		DGW

Danh sách bảng

4.3.1 Kết quả thực nghiệm các phương pháp sinh dữ liệu (phần 1)	38
4.3.2 Kết quả thực nghiệm các phương pháp sinh dữ liệu (phần 2)	39

Danh sách hình vẽ

2.2.1 Khoảng cách Euclid và khoảng cách DTW	10
2.2.2 Đường căn chỉnh của hai chuỗi thời gian	11
2.3.1 Trí tuệ nhân tạo, Máy học và Học sâu	12
2.3.2 Ô tô điện của VinFast - VF e35 được trang bị tính năng tự lái	13
2.3.3 Nền tảng chuẩn đoán hình ảnh VinDr	13
2.3.4 Chuyển văn bản thành giọng đọc tự nhiên của FPT	14
2.4.1 K - Nearest Neighbours	15
2.4.2 Margin trong bài toán phân lớp	16
2.4.3 Cây quyết định	17
2.4.4 Random Forest	18
2.4.5 Decision Stumps	19
2.4.6 Quá trình đào tạo của mô hình AdaBoost	20
2.4.7 eXtreme Gradient Boosting	21
2.4.8 Cơ chế phát triển cây của LightGBM	22
2.4.9 Cơ chế phát triển cây của các mô hình cây khác	22
2.4.10 Cấu trúc mạng MLP	23
2.4.11 Mạng VGG	23
2.4.12 Cấu trúc tế bào nhớ của LSTM	24
2.4.13 Cấu trúc mạng LSTM-FCN	25
2.4.14 Cấu trúc mạng ResNet	25
3.1.1 Chuỗi thời gian ban đầu (đường màu xanh) và chuỗi thời gian đã qua jittering (đường màu cam)	26
3.2.1 Chuỗi thời gian ban đầu (đường màu xanh) và chuỗi thời gian đã qua rotation (đường màu cam)	27
3.3.1 Chuỗi thời gian ban đầu (đường màu xanh) và chuỗi thời gian đã qua scaling (đường màu cam)	27
3.4.1 Chuỗi thời gian ban đầu (đường màu xanh) và chuỗi thời gian đã qua magnitude warping (đường màu cam)	28
3.5.1 Chuỗi thời gian ban đầu (đường màu xanh) và chuỗi thời gian đã qua permutation (đường màu cam)	29
3.6.1 Chuỗi thời gian ban đầu (đường màu xanh) và chuỗi thời gian đã qua window slice (đường màu cam)	29
3.7.1 Chuỗi thời gian ban đầu (đường màu xanh) và chuỗi thời gian đã qua time warping (đường màu cam)	30
3.8.1 Chuỗi thời gian ban đầu (đường màu xanh) và chuỗi thời gian đã qua window warping (đường màu cam)	31
3.9.1 Chuỗi thời gian ban đầu (đường màu xanh) và chuỗi thời gian đã qua spawner (đường màu cam)	32
3.10.1 Chuỗi thời gian ban đầu (đường màu xanh) và chuỗi thời gian đã qua wdba (đường màu cam)	33
3.11.1 Chuỗi thời gian ban đầu (đường màu xanh) và chuỗi thời gian đã qua rgw (đường màu cam)	34
3.12.1 Chuỗi thời gian ban đầu (đường màu xanh) và chuỗi thời gian đã qua dgw (đường màu cam)	35
4.1.1 Các class của 1 bộ dữ liệu	36
4.1.2 Thành phần của 1 bộ dữ liệu	36

1 Tổng quan

1.1 Giới thiệu đề tài

Trước khi chọn đề tài này, Nhóm đã tìm hiểu và thử nghiệm một số đề tài khác như: dự báo thời tiết, dự báo tinh hình Covid-19 ở Việt Nam hay dự đoán giá chứng khoán. Nhưng sau khi đi sâu hơn vào thực hiện, Nhóm nhận thấy được sự khó khăn, khan hiếm trong việc tìm hiểu, thu thập dữ liệu, mà đặc biệt là dữ liệu chất lượng cao. Tuy vậy, nhóm nhận thấy các dữ liệu này đều ở dạng *Chuỗi thời gian*. Do đó, Nhóm đã nảy ra ý tưởng và đi tìm hiểu về đề tài **Sinh dữ liệu** đối với chuỗi thời gian.

Đề tài của Nhóm tận dụng những đặc trưng rất độc đáo của dữ liệu ở dạng Chuỗi thời gian, rồi từ đó sử dụng các phương pháp Sinh dữ liệu đã tìm hiểu, tổng hợp được để sinh ra những bộ dữ liệu có kích thước như mong muốn với chất lượng cao hơn so với bộ dữ liệu ban đầu. Việc kiểm tra chất lượng dữ liệu được Nhóm thực hiện bằng cách so sánh kết quả của các mô hình dự đoán đối với từng bộ dữ liệu.

1.2 Thực trạng nghiên cứu của đề tài

- Các kỹ thuật sinh dữ liệu đã được tìm thấy, hữu ích trong các lĩnh vực như NLP- xử lý ngôn ngữ tự nhiên và Computer Vision-thị giác máy tính:

- + Trong thị giác máy tính, các phép biến đổi như cắt, lật và xoay được sử dụng.

- + Trong NLP, các kỹ thuật gia sinh dữ liệu có thể bao gồm hoán vị, xóa, chèn ngẫu nhiên, và một số các kỹ thuật khác.

- Người ta có đang nghiên cứu về sinh dữ liệu nhiều hay không ?**

- + Là một vấn đề đã và đang được nghiên cứu trong khoảng 20 năm trở lại đây;

- + Ngoài mục đích gia tăng sự đa dạng cho tập dữ liệu đào tạo cho mô hình học máy, còn nhằm giảm thiểu vấn đề overfitting cho các mô hình.

- Nghiên cứu thì nghiên cứu bằng những phương pháp nào ?**

- + Một số phương pháp thêm dữ liệu:

- i. *Collect more data*: Đúng nghĩa là lấy thêm dữ liệu. Trả tiền, lấy dữ liệu trên mạng, .v.v.

- ii. *Data synthesis*: Tạo dữ liệu giả. Đối với một số bài toán dữ liệu có thể được mô phỏng qua computer graphic. Như ảnh depth, ảnh ở chiều góc nhìn khác nhau, .v.v.

- iii. *Data Augmentation*: Là kỹ thuật đơn giản nhất bằng việc xử lý đơn giản dữ liệu sẵn có bằng các phép tuyến tính hay phi tuyến;

- + Sinh dữ liệu được chia làm 2 hướng chính là cơ bản và nâng cao:

- i. Cơ bản sử dụng các phương pháp biến đổi: thay đổi miền độ lớn, thay đổi miền tần số, thay đổi miền thời gian.

- ii. Nâng cao sử dụng các mô hình học máy và deep learning: GAN, CNN, ...

- Nhược điểm của các phương pháp vừa liệt kê:**

- + *Phương pháp 1*: thì quá tốt nếu thực hiện được, tuy nhiên vì nhiều lý do và điều kiện ta không thể thu thập thêm được dữ liệu vì việc này tốn thời gian, công sức và cả tiền nữa.

- + *Phương pháp số 2*: thì khó có thể áp dụng được, vì là dữ liệu giả không gắn liền với thực tế nên khi đưa vào mô hình sẽ xảy ra vấn đề đó là một số thông số đánh giá bị sai lệch khá

nhiều.

+ *Phương pháp số 3* Tăng cường dữ liệu đã được chứng minh là cải thiện khả năng tổng quát hóa khả năng của các mô hình và đặc biệt phổ biến trong lĩnh vực thị giác máy tính, đó là xử lý dữ liệu hình ảnh. Ngược lại, tăng cường dữ liệu ít được sử dụng rộng rãi trong lĩnh vực dữ liệu chuỗi thời gian, chẳng hạn như phân loại chuỗi thời gian, hơn là trong thị giác máy tính. Đây là bởi vì dữ liệu chuỗi thời gian đặc biệt dễ bị ảnh hưởng bởi sự biến đổi dữ liệu xảy ra trong khi thực hiện tăng dữ liệu.

Ví dụ: việc lật hoặc xoay hình ảnh để tăng thêm dữ liệu hình ảnh không làm giảm đáng kể ý nghĩa của dữ liệu gốc. Tuy nhiên, trong trường hợp dữ liệu chuỗi thời gian, nó có khả năng làm sai lệch ý nghĩa của dữ liệu và rất khó để xác định liệu nó có đã thay đổi. Các phương pháp tăng dữ liệu chuỗi thời gian được đề xuất trước đây hoạt động tốt trong nhiều lĩnh vực, nhưng thường không xem xét thông tin xu hướng của dữ liệu chuỗi thời gian chẳng hạn như cắt hoặc sắp xếp lại chuỗi thời gian ban đầu.

1.3 Lý do chọn đề tài

- Dữ liệu về chuỗi thời gian là rất phổ biến trong đời sống hiện nay. Nhưng dữ liệu được phân lớp, gán nhãn thì còn hạn chế do chi phí của việc gán nhãn dữ liệu là không hề nhỏ.
- Do đó, chúng ta cần tận dụng những tập dữ liệu đã phân lớp, gán nhãn để đưa ra những bộ dữ liệu mới lớn hơn, đem lại hiệu quả cao hơn trong những mô hình dự báo.

1.4 Đối tượng nghiên cứu

Các phương pháp sinh dữ liệu.

1.5 Phạm vi nghiên cứu đề tài

- Tìm hiểu/ khảo sát 12 phương pháp sinh dữ liệu phổ biến đối với bài toán phân lớp chuỗi thời gian, đánh giá thông qua 1 số mô hình deep learning cho việc phân lớp chuỗi thời gian.

1.6 Mục đích nghiên cứu

- Tìm hiểu các phương pháp sinh dữ liệu.
- Thực nghiệm các mô hình đã nêu trên các tập dữ liệu mới sinh để đưa ra phương pháp sinh dữ liệu tối ưu nhất đối với từng mô hình.

1.7 Phương pháp nghiên cứu

- Phương pháp thực nghiệm khoa học.
- Phương pháp phân loại và hệ thống hóa lý thuyết.
- Phương pháp quan sát khoa học.

1.8 Kết quả, ý nghĩa của việc nghiên cứu

- Kết quả:

- + Nắm được 12 phương pháp cơ bản đã đề xuất.
- + Thực nghiệm trên những bộ dữ liệu cụ thể với một số mô hình phổ biến.

- **Ý nghĩa:** Mang lại một giải pháp về vấn đề dữ liệu cho những bài toán học sâu.

2 Cơ sở lý thuyết

2.1 Chuỗi thời gian

- **Khái niệm chuỗi thời gian:** Chuỗi thời gian là 1 tập quan sát X_t , mỗi quan sát được ghi lại tại 1 thời điểm cụ thể.

- **White Noise:** Chuỗi X_t là chuỗi các biến ngẫu nhiên không tương quan là 1 chuỗi nhiễu trắng với kỳ vọng là 0 và phương sai là σ^2 .

Ký hiệu: $X_t \sim WN(0, \sigma^2)$

- **Thành phần chuỗi thời gian:** Bao gồm 4 thành phần:

- + Thành phần xu hướng (trend)
- + Thành phần mùa (seasonal)
- + Thành phần dư (residual)
- + Thành phần chu kỳ (cycle)

- **Tính dừng:**

+ Dừng chặt: X_t là chuỗi dừng chặt nếu: tồn tại thời gian dịch chuyển h sao cho: X_{t+h} có cùng phân phối với X_t

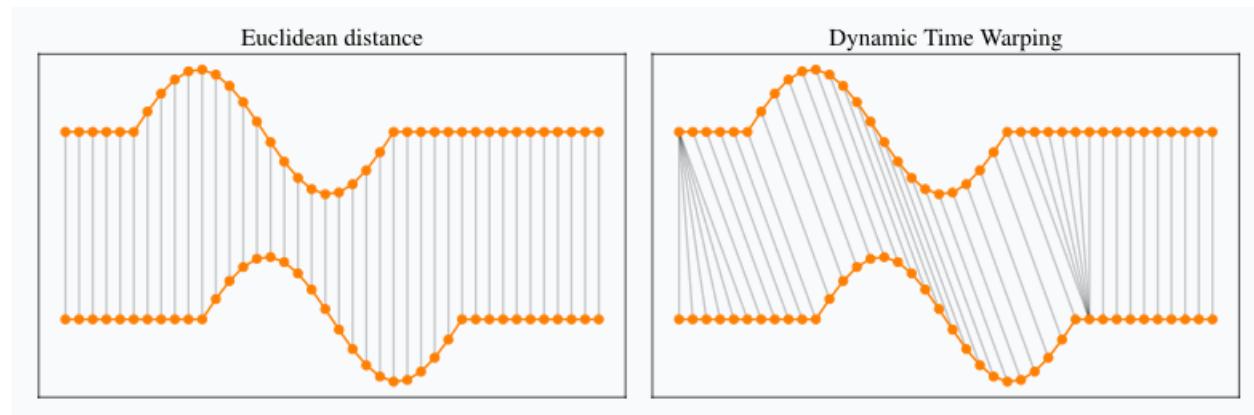
+ Dừng yếu: X_t là chuỗi dừng yếu nếu:

- i. $\mu_X(r)$ độc lập với t
- ii. $\gamma_X(t+h, t)$ độc lập với t với mọi h

2.2 Lý thuyết DTW

- **Định nghĩa:** Dynamic Time Warping là cách để so sánh hai chuỗi thời gian không đồng bộ với nhau. Nó chỉ ra sự phù hợp của hai chuỗi, sử dụng để tìm ra “khoảng cách” giữa hai chuỗi thời gian.

- Để thực hiện được điều này, DTW sử dụng khoảng cách căn chỉnh thay vì khoảng cách Euclid.



Hình 2.2.1: Khoảng cách Euclid và khoảng cách DTW

Giả sử ta có hai chuỗi thời gian như sau: $X = x_0, x_1, \dots, x_n$ và $X' = x'_0, x'_1, \dots, x'_m$
Công thức khoảng cách Euclid của hai chuỗi thời gian:

$$\sum_{i < T} d(x_i, x'_i)$$

Công thức khoảng cách theo DTW:

$$DTW_q(x, x') = \min_{\pi \in \mathcal{A}(x, x')} \left(\sum_{(i,j) \in \pi} d(x_i, x'_j)^q \right)^{\frac{1}{q}}$$

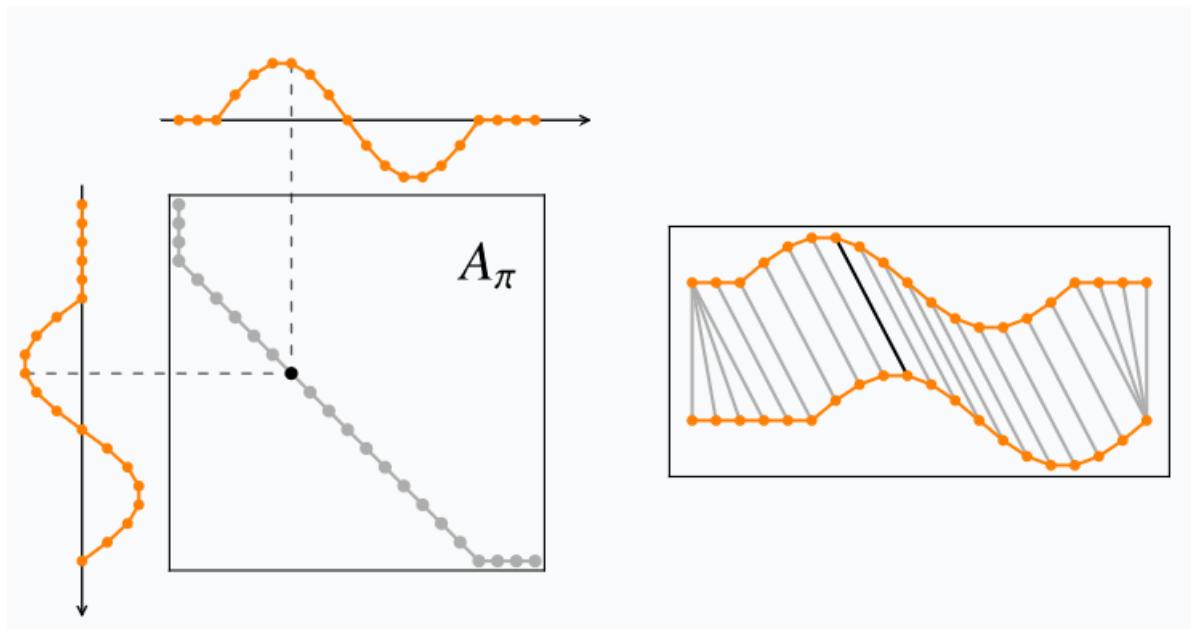
Đường căn chỉnh π độ dài K là dãy các cặp index $((i_0, j_0), (i_{K-1}, j_{K-1}))$ và $A(x, x')$ là tập hợp các đường có thể chấp nhận. Để được chấp nhận, đường đó phải thỏa mãn các điều kiện sau:

+ Điểm bắt đầu và kết thúc của hai chuỗi thời gian trùng nhau:

- i. $\pi_0 = (0,0)$
- ii. $\pi_{K-1} = (n-1)(m-1)$

+ Chuỗi tăng đơn điệu với ở cả i và j, tất cả các index phải xuất hiện ít nhất một lần:

- i. $i_{k-1} \leq i_k \leq i_{k-1} + 1$
- ii. $j_{k-1} \leq j_k \leq j_{k-1} + 1$



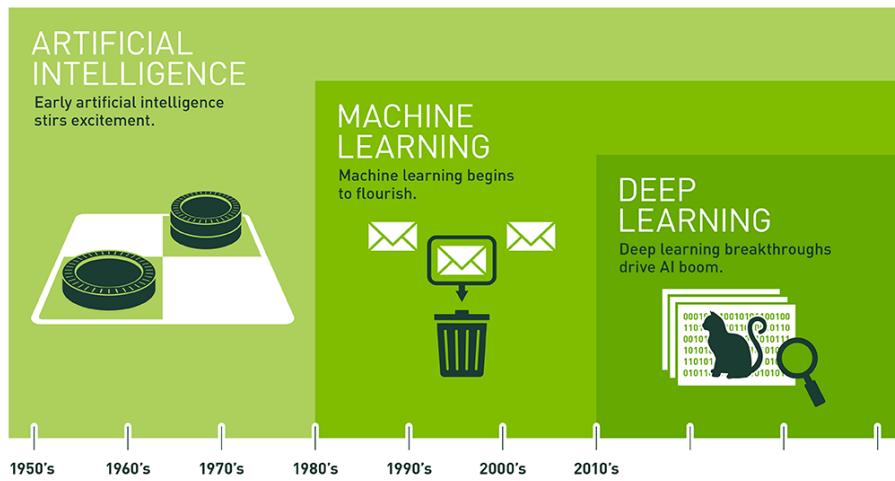
Hình 2.2.2: Đường căn chỉnh của hai chuỗi thời gian

2.3 Nền tảng máy học

2.3.1 Khái niệm

- **Trí tuệ nhân tạo (Artificial Intelligence):** Là trí tuệ do con người lập trình tạo nên với mục tiêu giúp máy tính có thể tự động hóa các hành vi thông minh như con người.
- **Máy học (Machine Learning):** là một lĩnh vực của trí tuệ nhân tạo liên quan đến việc nghiên cứu và xây dựng các kỹ thuật cho phép các hệ thống "học" tự động từ dữ liệu để giải quyết những vấn đề cụ thể.

- **Học sâu (Deep Learning):** là một tập con của Machine Learning dựa trên một tập hợp các thuật toán để cố gắng mô hình dữ liệu trừu tượng hóa ở mức cao bằng cách sử dụng nhiều lớp xử lý với cấu trúc phức tạp, hoặc bằng cách khác bao gồm nhiều biến đổi phi tuyến.



Hình 2.3.1: Trí tuệ nhân tạo, Máy học và Học sâu

2.3.2 Vai trò của Trí tuệ nhân tạo

Kể từ năm 1955, khi Jhon McCarthy đưa ra những khái niệm đầu tiên về Trí tuệ nhân tạo, nhân loại đã nhanh chóng phát triển và nhận thấy đây là một lĩnh vực khoa học rất có tiềm năng trong một tương lai không xa. Và thực tế đã kiểm nghiệm điều đó, "kỷ nguyên AI" đã được đánh dấu thông qua sự kiện máy tính **Deep Blue** của IBM đã đánh bại Garry Kasparov - nhà vô địch cờ vua thế giới. Siêu máy tính này có khả năng tính toán 200 triệu nước đi mỗi giây và được cải thiện với việc phân tích hàng ngàn trận đấu cờ vua khác nhau.

Ngày nay, Trí tuệ nhân tạo đang len lỏi vào mọi lĩnh vực của đời sống, từ những thiết bị di động ta sử dụng hàng ngày tới những công cụ trong sản xuất, thậm chí nó còn đang thay đổi nhiều ngành nghề như giáo dục, y tế,... cũng như làm một số nghề sử dụng nhiều nhân lực hiện nay như tiếp thị, công nhân đóng dây chuyền.. có thể biến mất trong tương lai.

Như vậy, ta có thể thấy tầm ảnh hưởng của trí tuệ nhân tạo trong cuộc cách mạng 4.0 là rất lớn.

2.3.3 Một số ứng dụng Trí tuệ nhân tạo tại Việt Nam

Xe điện tự lái của VinFast

Vào cuối năm 2021, hàng triệu người dân trên cả nước đã dõi theo sự kiện ra mắt xe điện của VinFast tại Triển lãm ô tô Los Angeles 2021 (Mỹ). Đây thực sự là một bước ngoặt lớn

không chỉ với ngành ô tô nước nhà mà còn là sự kiện đánh dấu việc VinGroup - tập đoàn tư nhân lớn nhất Việt Nam cạnh tranh thị phần đầy tiềm năng của xe điện tự hành trên toàn cầu với những hãng xe "sừng sỏ" nhất hiện nay như Tesla hay Blue Origin.



Hình 2.3.2: Ô tô điện của VinFast - VF e35 được trang bị tính năng tự lái

Nền tảng AI toàn diện cho chẩn đoán hình ảnh y tế - VinBigdata

Nền tảng trí tuệ nhân tạo tích hợp trên hệ thống lưu trữ và truyền tải hình ảnh y tế (PACS) nhằm hỗ trợ bác sĩ chẩn đoán hình ảnh đưa ra quyết định chính xác, nhanh chóng và giảm thiểu sai sót. Hiện nay, nền tảng đã được ứng dụng trong thực tế và có một số tính năng nổi bật như:

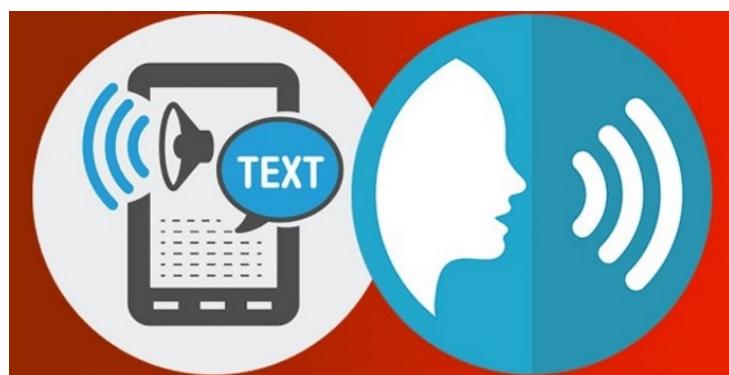
- Đưa ra gợi ý chẩn đoán bệnh và khoanh vùng tổn thương trên ảnh;
- Tự động chẩn đoán đồng thời nhiều ca chụp theo thời gian thực;
- Chẩn đoán X-quang lồng ngực, X-quang tuyến vú, X-quang cột sống, CT lồng ngực và CT gan mật.



Hình 2.3.3: Nền tảng chẩn đoán hình ảnh VinDr

Giọng đọc tự nhiên tiếng Việt - FPT: FPT.AI Text To Speech (TTS) cung cấp giọng đọc tự nhiên, tự động ngắt nghỉ, biểu cảm ngữ điệu được phát triển dành cho thị trường Việt

Nam với giọng đọc đến từ nhiều khu vực khác nhau của Việt Nam.



Hình 2.3.4: Chuyển văn bản thành giọng đọc tự nhiên của FPT

2.3.4 Phương pháp đánh giá một hệ thống phân lớp

Trong bài toán phân lớp, ta có khá nhiều phương pháp để đánh giá mức độ hiệu quả của một mô hình. Tuy vậy, tùy thuộc vào tính chất của bài toán mà ta cần chọn một (một vài) phương pháp phù hợp, rồi từ đó so sánh một cách tương đối mức độ hiệu quả của các mô hình với nhau. Trong bài toán này, ta sẽ sử dụng độ đo là **độ chính xác** để kiểm tra mức độ hiệu quả của Phép tách phân lớp, mô hình.

Độ chính xác hay *Accuracy* là cách đơn giản và phổ biến nhất khi nói đến bài toán phân lớp. Cách đánh giá này đơn giản là tính tỉ lệ giữa số điểm dự đoán đúng và số điểm trong tập dữ liệu kiểm thử. Độ chính xác càng cao thì mô hình của chúng ta càng chính xác.

Ví dụ, ta có giá trị kiểm định là [A, B, A, C] và giá trị dự đoán là [A, B, A, B], tức mô hình đã dự đoán đúng 3 trên tổng số 4 điểm dữ liệu, hay độ chính xác là 75%.

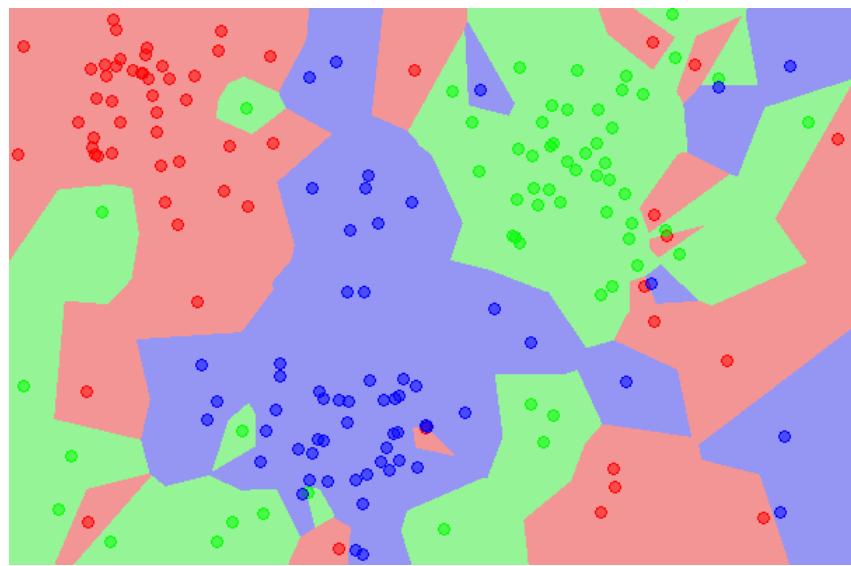
2.4 Giới thiệu một số Mô hình

2.4.1 K - Nearest Neighbours - KNN

KNN là một trong những thuật toán học có giám sát đơn giản nhất trong Máy học. Mặc dù vậy, nó tỏ ra khá hiệu quả trong một số trường hợp. Điều đặc biệt của mô hình này chính là việc nó *không học* bất cứ một điều gì từ dữ liệu huấn luyện (thuộc nhóm "lười học" hay *lazy learning*), mọi tính toán chỉ được thực hiện khi nó cần dự báo kết quả của dữ liệu mới [6].

Ý tưởng của nó cũng rất đơn giản, nhãn của một điểm dữ liệu được suy ra từ việc xét **K** điểm có khoảng cách gần nó nhất trong tập huấn luyện. Ở đây, nó có thể quyết định bằng phương thức bầu chọn hoặc có thể đánh trọng số khác nhau ở mỗi điểm, rồi từ đó suy ra nhãn.

Tóm lại, chỉ dựa trên **K** điểm lân cận mà một điểm dữ liệu mới có thể xác định được nhãn. Nó rất đơn giản, nhưng cũng rất nguy hiểm khi tập dữ liệu chưa được lọc *nhiều*.



Hình 2.4.1: K - Nearest Neighbours

2.4.2 Logistic Regression

Logistic Regression hay *Hồi quy Logistic* được sử dụng phổ biến để ước lượng xác suất một mẫu dữ liệu thuộc về một lớp cụ thể nào đó (ví dụ như xác suất một email là thư rác). Nếu xác suất ước lượng cho một lớp lớn hơn 50% thì mô hình dự đoán mẫu thuộc về lớp đó. Vì vậy, đây là một bộ phân loại nhị phân.

Ước lượng Xác suất

Giống như Hồi quy Tuyến tính, mô hình Hồi quy Logistic tính tổng trọng số các đặc trưng đầu vào (cộng với hệ số điều chỉnh). Nhưng thay vì cho ra kết quả trực tiếp như Hồi quy tuyến tính, nó cho ra *Logistic* của tổng này [6].

Xác suất ước lượng của mô hình Hồi quy Logistic:

$$\hat{p} = h_{\theta}(x) = \sigma(x^T \theta) \quad (2.4.1)$$

Ở đây, logistic - được ký hiệu là $\sigma(\cdot)$ - là một *hàm sigmoid* hay *sigmoid function* cho đầu ra từ 0 đến 1 (Hàm sigmoid sẽ được đề cập cụ thể ở phần 4.2.2)

Khi mô hình Hồi quy Logistic đã ước lượng xác suất mẫu x thuộc lớp dương $\hat{p} = h_{\theta}(x)$, nó có thể đưa ra dự đoán một cách dễ dàng.

2.4.3 Gaussian Naive Bayes

Mô hình này được sử dụng chủ yếu trong loại dữ liệu mà các thành phần là liên tục.

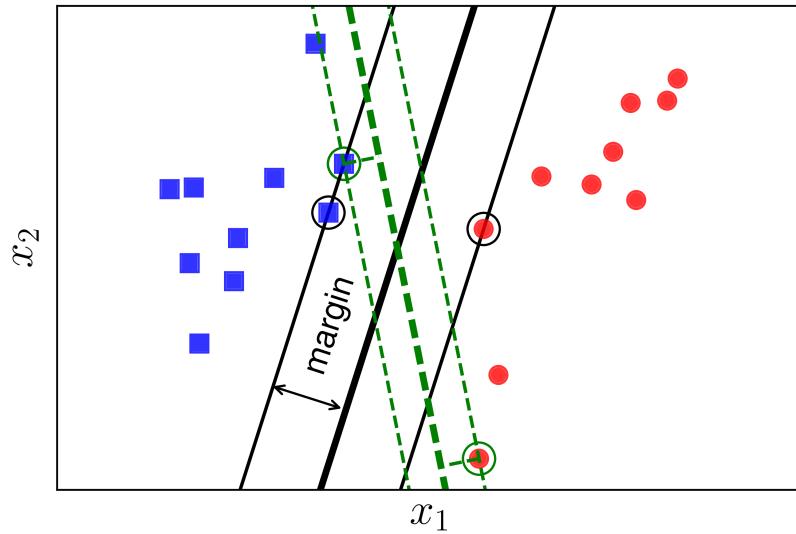
Với mỗi chiều dữ liệu i và một class c , x_i tuân theo một phân phối chuẩn có kỳ vọng μ_{ci} và phương sai σ_{ci}^2

$$p(x_i | c) = p(x_i | \mu_{ci}, \sigma_{ci}^2) = \frac{1}{\sqrt{2\pi\sigma_{ci}^2}} \exp\left(-\frac{(x_i - \mu_{ci})^2}{2\sigma_{ci}^2}\right) \quad (2.4.2)$$

Trong đó, bộ tham số $\theta = \{\mu_{ci}, \sigma_{ci}^2\}$ được xác định bằng Maximum Likelihood.

2.4.4 Support Vector Machine - SVM

Support Vector Machine hay *Máy vecto hỗ trợ* là một trong những thuật toán phân loại phổ biến, hiệu quả với ý tưởng đúng sau khá đơn giản.



Hình 2.4.2: Margin trong bài toán phân lớp

Ta có định nghĩa, *Margin* là khoảng cách từ điểm gần nhất của mỗi lớp tới đường phân chia sao cho các khoảng cách này bằng nhau. Như vậy, nếu margin càng rộng thì hiệu quả của việc phân lớp sẽ càng cao.

Như vậy, ý tưởng của Máy vecto hỗ trợ chính là bài toán đi tìm đường phân chia sao cho *margin* lớn nhất.

Cách tính Margin

Giả sử, mặt $w^T x + b = w_1 x_1 + w_2 x_2 + b = 0$ chia 2 lớp tương ứng với hai màu xanh và đỏ như hình. Khi đó, với cặp dữ liệu (x_n, y_n) bất kỳ, khoảng cách từ điểm đó tới mặt phân chia là:

$$\frac{y_n(w^T x_n + b)}{\|w\|_2} \quad (2.4.3)$$

Do margin là khoảng cách gần nhất từ một điểm tới mặt phân cách, nên ta có:

$$\text{margin} = \min_n \frac{y_n(w^T x_n + b)}{\|w\|_2} \quad (2.4.4)$$

Do vậy, việc tối ưu trong Máy vecto hỗ trợ chính là việc tìm w và b sao cho margin đạt giá trị lớn nhất [6].

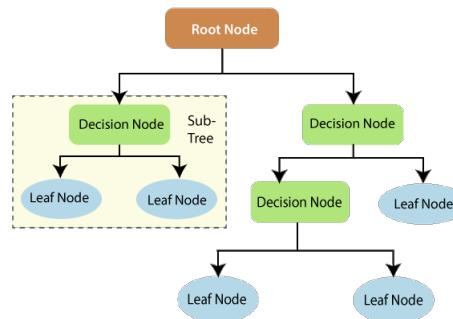
2.4.5 Decision Tree

Các mô hình như Hồi quy tuyến tính hay Hồi quy Logistic sẽ gặp khó khăn trong những tình huống như mối quan hệ giữa các đặc trưng (features) và nhãn (label) là phi tuyến tính, hay giữa các đặc trưng có mối quan hệ lẫn nhau. Nhưng *Decision Tree* hay *Cây quyết định* có thể làm được điều đó.

Qua quá trình tách, những tập dữ liệu con phân biệt sẽ được tạo, với mỗi ví dụ thuộc về một tập con. Tập con cuối cùng được gọi là phần cuối hay nút lá (*leaf*) và các tập con phía trên lá được gọi là các *nút không phải là lá* (*non-leaf node*) (gọi là **nút**). Các nút có thể là *nút gốc* (*root node*), *nút con* (*child node*) hay *nút anh em* (*sibling node*) [6].

Trong đó,

- Nút gốc thể hiện câu hỏi đầu tiên
- Các nút có thể có hai hoặc nhiều nút con. Các nút con này có thể là một nút lá hoặc một nút khác.



Hình 2.4.3: Cây quyết định

Việc xây dựng một cây quyết định trên tập dữ liệu huấn luyện về bản chất là việc *đi tìm câu hỏi* của bài toán cũng như *thứ tự* để giải bài toán đó.

Vậy làm thế nào để xây dựng một cây quyết định? Một trong những cách phổ biến thường được sử dụng trong bài toán Phân lớp (Classification) với tất cả các thuộc tính đều ở dạng phân loại (categorical) chính là Iterative Dichotomiser 3 (ID3).

Ý tưởng thuật toán ID3

Cây quyết định cần xác định thứ tự của các thuộc tính tại mỗi nút. Tuy nhiên, với những bài toán có một lượng thuộc tính lớn, cũng như các giá trị có trong mỗi thuộc tính lớn thì việc lựa chọn một các thủ công sẽ rất khó khăn. Do vậy, ID3 sẽ:

- Tại mỗi bước, ta sẽ tìm ra một thuộc tính *tốt nhất* dựa trên một quy tắc nào đó.
- Quy tắc này dựa trên tầm quan trọng trong việc phân chia các lớp.
- Rồi từ đó, chia dữ liệu vào các tập con tương ứng.

Cách chọn này có thể không phải là tối ưu nhất, nhưng nó giúp cho bài toán trở lên đơn giản hơn, đồng thời nó cho kết quả tương đối tốt.

2.4.6 Random Forest

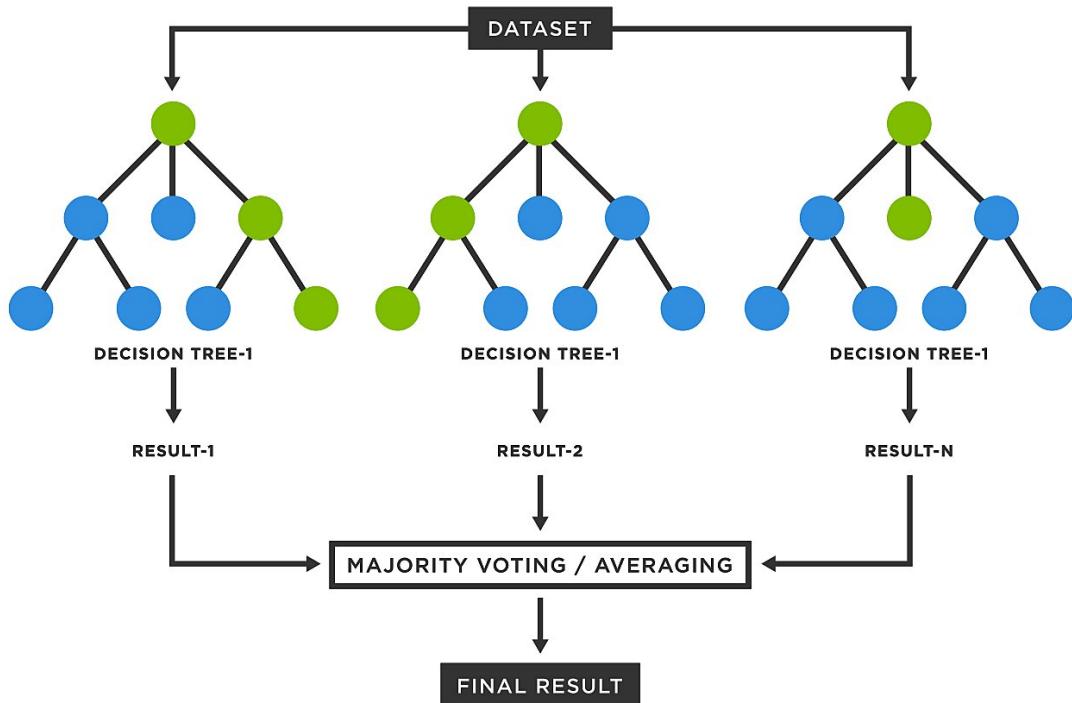
Random Forest hay *Rừng ngẫu nhiên* là một mô hình học có giám sát được xây dựng từ *cây quyết định*. Nó sử dụng phương pháp học theo nhóm (ensemble learning), phương pháp cho phép kết hợp nhiều bộ phận phân loại đều giải quyết những vấn đề phức tạp.

Mô hình Rừng ngẫu nhiên là sự kết hợp của rất nhiều cây quyết định. Ở đây, "Rừng" được tạo ra thông qua *bagging* hay *bootstrap*. Bagging là một thuật toán tổng hợp giúp cải thiện độ chính xác của các mô hình máy học.

Cách thức vận hành

Mỗi cây quyết định thuộc "rừng" sẽ bao gồm một mẫu dữ liệu được lấy ngẫu nhiên từ tập huấn luyện (dữ liệu được phép lặp lại) - được gọi là *mẫu bootstrap*. Trong mẫu dữ liệu đó, một phần ba được dành làm dữ liệu thử nghiệm. Số còn lại sẽ được đưa vào mô hình dự đoán, tùy từng bài toán mà ta sẽ có các cách dự đoán khác nhau. Và cuối cùng, tập dữ liệu thử nghiệm sẽ được sử dụng để xác thực chéo, từ đó hoàn thiện dự báo.

Như vậy, trong khi cây quyết định một mình có kết quả và phạm vi hẹp thì Rừng ngẫu nhiên đảm bảo kết quả chính xác hơn với số lượng nhóm và quyết định lớn hơn. Không chỉ vậy, tính ngẫu nhiên - "Random" giúp mô hình có thể tìm ra các tính năng tốt nhất trong số một tập hợp các tính năng.



Hình 2.4.4: Random Forest

Nhờ những cải tiến trên mà hiện nay nó được sử dụng một cách khá rộng rãi và được ứng dụng trong nhiều bài toán với những lĩnh vực khác nhau.

2.4.7 Extra Trees Classifier

Extremely Randomized Trees Classifier hay *Extra Trees Classifier* là mô hình có rất nhiều điểm giống với *Random Forest*. Nó cũng sử dụng phương pháp học theo nhóm, nó tổng hợp kết quả của nhiều cây quyết định không tương quan được thu thập trong một rừng để đưa ra kết quả phân loại.

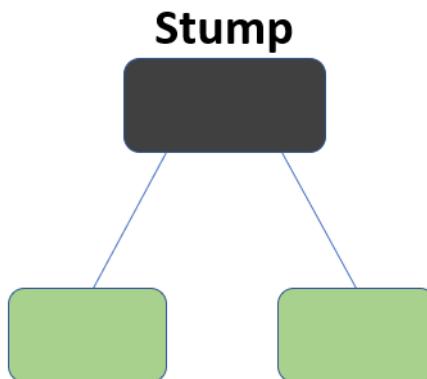
Điểm khác biệt chính của 2 mô hình trên chính là ở cách xây dựng các cây quyết định. Mỗi cây quyết định trong *Extra Trees Classifier* được xây dựng từ mẫu huấn luyện ban đầu. Sau đó, tại mỗi nút kiểm tra, mỗi cây được cung cấp một mẫu ngẫu nhiên gồm k đặc trưng, rồi từ đó mỗi cây quyết định phải chọn ra đặc trưng tốt nhất để tách dữ liệu dựa trên một số tiêu chí toán học (diễn hình là chỉ số *Gini*). Những tập đặc trưng ngẫu nhiên này dẫn đến việc tạo ra những cây quyết định không tương quan.

2.4.8 Adaptive Boosting

Năm 1997, lần đầu tiên kỹ thuật **Boosting** được công bố bởi Freund và Schapire, kể từ đó Boosting đã là một kỹ thuật phổ biến để giải quyết các bài toán phân lớp nhị phân. Các thuật toán sử dụng kỹ thuật này cải thiện khả năng dự báo bằng cách chuyển đổi một số mô hình có khả năng dự báo yếu thành mô hình có khả năng dự báo mạnh.

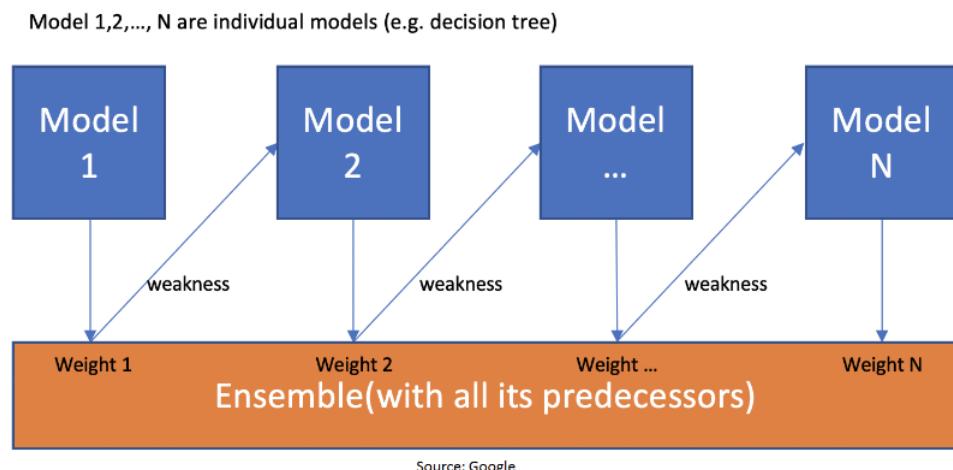
Nguyên tắc đằng sau các mô hình thúc đẩy là đầu tiên chúng xây dựng một mô hình trên tập dữ liệu đào tạo, sau đó mô hình thứ hai được xây dựng để sửa các lỗi có trong mô hình đầu tiên. Quá trình này được tiếp tục cho đến khi các lỗi được giảm thiểu và tập dữ liệu được dự đoán chính xác hơn.

Adaptive Boosting hay *AdaBoost* là một kỹ thuật trong Máy học được sử dụng như một phương pháp gộp (*Ensemble Method*). Mô hình phổ biến nhất được sử dụng với AdaBoost là cây quyết định có *một mức nghĩa*, tức là cây quyết định chỉ có 1 lần phân chia. Những cây này còn được gọi là **Decision Stumps**.



Hình 2.4.5: Decision Stumps

Nguyên tắc của mô hình này làm là nó xây dựng một mô hình và đưa ra trọng số bằng nhau cho tất cả các điểm dữ liệu. Sau đó, nó chỉ định các trọng số cao hơn cho các điểm được phân loại sai. Bây giờ tất cả các điểm có trọng số cao hơn được coi trọng hơn trong lần đào tạo mô hình tiếp theo. Nó sẽ tiếp tục quá trình đào tạo cho đến khi và chỉ khi nhận được lỗi.



Hình 2.4.6: Quá trình đào tạo của mô hình AdaBoost

2.4.9 Gradient Boosting Classifier

Ta sẽ đến với một thuật toán sử dụng kỹ thuật Boosting, chính là *Gradient Boosting*. *Gradient Boosting* tạo ra một mô hình dự đoán dưới dạng một tập hợp các mô hình dự đoán yếu, mà điển hình là *cây quyết định*.

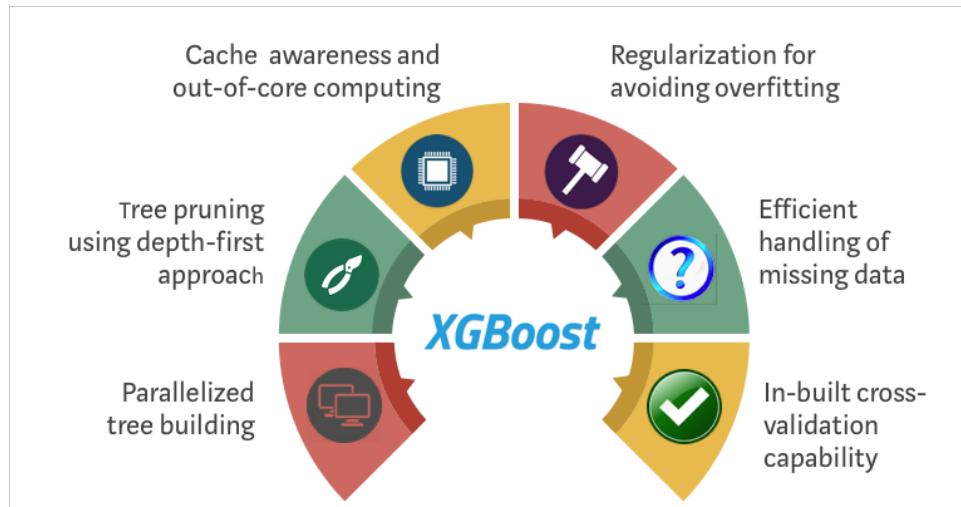
Ta có thể thấy 2 mô hình *Adaptive Boosting* và *Gradient Boosting* dường như khá giống nhau. Tuy vậy, điểm khác nhau dẫn đến việc không gộp 2 mô hình này là một chính là các hai thuật toán xác định những thiếu sót của các mô hình dự đoán yếu. Trong khi Adaptive Boosting xác định những thiếu sót của mô hình yếu bằng cách sử dụng các điểm dữ liệu có trọng số cao, Gradient Boosting cũng thực hiện tương tự nhưng bằng cách sử dụng gradient trong hàm mất mát.

2.4.10 eXtreme Gradient Boosting

Cách đây vài năm, *eXtreme Gradient Boosting* hay *XGBoost* là một mô hình thống trị trong các cuộc thi ứng dụng máy học cũng như trên nền tảng Kaggle với dữ liệu có cấu trúc. Ta thấy rằng, trong lĩnh vực dự báo dữ liệu phi cấu trúc thì mạng nơ-ron có xu hướng hoạt động tốt hơn cả. Tuy nhiên, khi đề cập đến dữ liệu có cấu trúc thì các thuật toán dựa trên cây quyết định đang là tốt nhất hiện nay.

XGBoost được phát triển như một dự án nghiên cứu tại Đại học Washington. Và nó đã được công bố vào năm 2016 tại hội nghị SIGKDD và khiến cho giới quan tâm đến Máy học chú ý tới. Kể từ đó, nó thường xuyên là phương án trong các bài toán Máy học, cũng như đã được áp dụng vào các ngành khoa học tiên tiến khác.

Vậy XGBoost là gì? XGBoost là một mô hình Máy học được tổng hợp dựa trên cây quyết định sử dụng kỹ thuật gradient boosting.



Hình 2.4.7: eXtreme Gradient Boosting

Những điểm khiến XGBoost nổi bật:

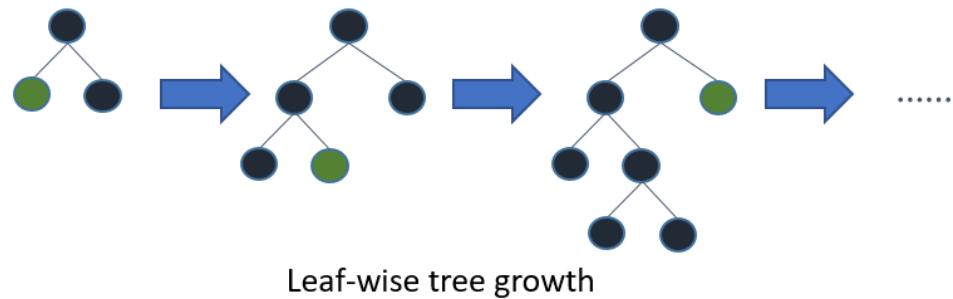
1. Khả năng song song: nó xây dựng cây bằng việc thực hiện song song hoàn toàn. Điều này có thể thực hiện được là do tính chất có thể hoán đổi được của các vòng lặp được sử dụng để xây dựng các trình học cơ sở. Vòng lặp bên ngoài liệt kê các nút lá của cây và vòng lặp bên trong thứ hai tính toán các đặc trưng.
2. Phương pháp "tỉa cây" (*Tree Pruning*): nó sử dụng siêu tham số *max_depth* để người dùng có thể lựa chọn được độ sâu tối đa mà cây có thể phát triển. Phương pháp này giúp tăng đáng kể hiệu suất tính toán.
3. Tối ưu hóa phần cứng: mô hình này được thiết kế để sử dụng hiệu quả tài nguyên phần cứng. Điều này được thực hiện thông qua việc bộ nhớ cache phân phối bộ đệm tới mỗi luồng để lưu trữ và thống kê gradient.

2.4.11 LightGBM

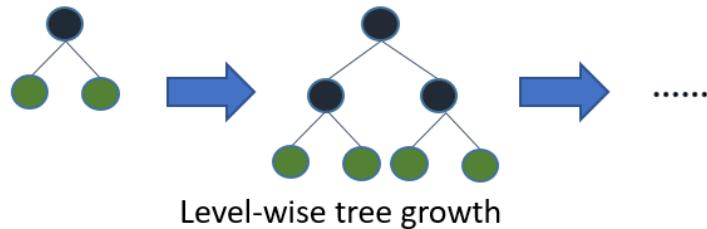
Khi trình bày về XGBoost, sở dĩ ta nói nói là "vua" của các cuộc thi về Máy học với dữ liệu ở dạng có cấu trúc **cách đây vài năm** là bởi vì sự xuất hiện của LightGBM. Nó đã thay thế XGBoost để trở thành mô hình có độ chính xác cao nhất, mang lại hiệu quả tốt nhất trong các cuộc thi, cũng như trong rất nhiều ứng dụng.

LightGBM được công bố lần đầu năm 2016 bởi các nhà phát triển đến từ Microsoft. Nó dựa trên kỹ thuật tăng cường độ dốc (gradient boosting) sử dụng các thuật toán học tập dựa trên cây.

Một điểm rất khác của LightGBM so với các mô hình cây khác chính là việc Light GBM phát triển cây theo chiều sâu, trong khi các mô hình cây khác phát triển theo chiều ngang.



Hình 2.4.8: Cơ chế phát triển cây của LightGBM



Hình 2.4.9: Cơ chế phát triển cây của các mô hình cây khác

Hay nói cách khác, LightGBM phát triển cây *lá khôn ngoan* (*leaf-wise*) còn các thuật toán khác phát triển cây theo *mức độ khôn ngoan* (*level-wise*).

LightGBM sẽ chọn lá bị mất denta tối đa để phát triển cây. Các nhà khoa học đã nhận thấy rằng việc sử dụng *lá khôn ngoan* có thể giảm tổn thất nhiều hơn *cấp khôn ngoan*.

Ngoài ra, có một số lý do khác mà nó đang trở lên cực kỳ phổ biến. Ví như việc nó có thể xử lý tập dữ liệu có **kích thước lớn** với **tốc độ cao** nhưng lại **chiếm ít bộ nhớ** để chạy. Ngoài ra, LightGBM được ưa chuộng vì nó tập trung để tăng **độ chính xác** của mô hình, cũng như nó có thể chạy được trên **GPU**.

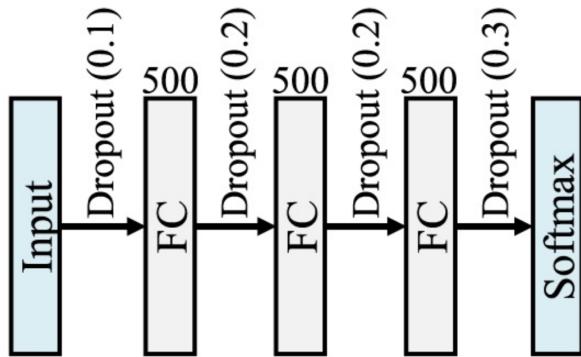
2.4.12 Multi-Layer Perceptron Neural Networks

Multi-Layer Perceptron hay *MLP* là một trong những mô hình cơ bản nhất của mạng nơ-ron. Nó được cấu tạo từ các tầng bao gồm: tầng đầu vào, tầng ẩn và tầng đầu ra.

Trong các tầng có các nút, các nút ở tầng trước sẽ liên kết với tầng sau để tạo ra một cấu trúc mạng hoàn chỉnh.

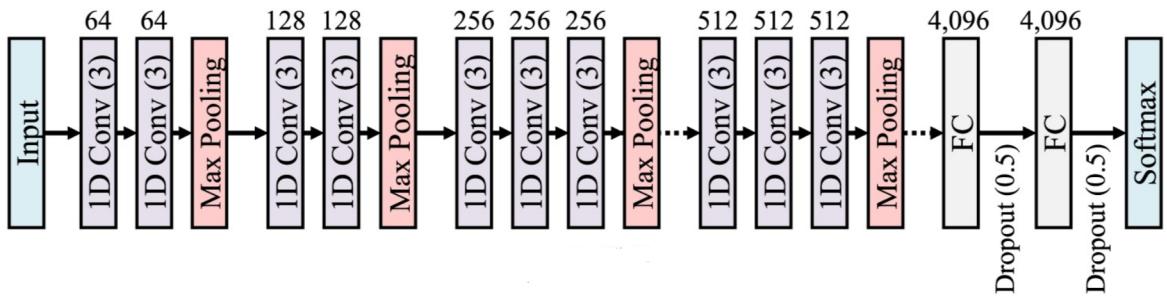
2.4.13 Visual Geometry Group

Visual Geometry Group hay *VGG* là một kiến trúc của *Convolutional Neural Network (CNN)* với nhiều lớp. Chúng ta có thể thiết kế mạng VGG với số lớp bất kỳ. Nhưng trên thực tế, người ta thường sử dụng hai loại mạng là *VGG-16* và *VGG-19* tương ứng với mạng bao gồm



Hình 2.4.10: Cấu trúc mạng MLP

16 và 19 lớp chập (convolutional layers).



Hình 2.4.11: Mạng VGG

Mạng VGG có thể được phân chia thành hai phần: phần đầu tiên bao gồm chủ yếu các tầng tích chập và tầng gộp, còn phần thứ hai bao gồm các tầng kết nối đầy đủ. Phần tích chập của mạng gồm các mô-đun **vgg_block** kết nối liên tiếp với nhau [4].

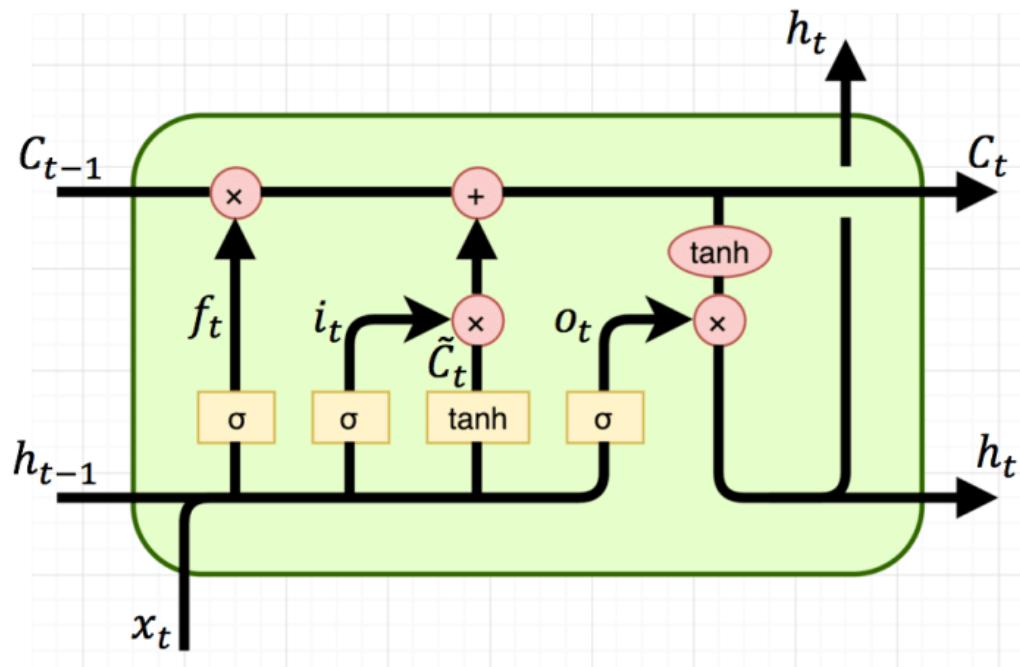
Việc sử dụng các khối giúp ta định nghĩa mạng bằng các đoạn mã nguồn ngắn gọn và thiết kế các mạng phức tạp một cách hiệu quả hơn.

2.4.14 Long Short Term Memory

Long Short Term Memory - LSTM hay *Mạng bộ nhớ dài ngắn* là một nâng cấp của *Recurrent neural network - RNN* hay *Mạng nơ-ron hồi tiếp*. LSTM khắc phục được hiện tượng **vanishing gradient** của RNN, đồng thời đặc điểm trong thiết kế các nơ-ron của LSTM khiến nó có thể "nhớ" được lâu hơn, làm cho quá trình học được hiệu quả hơn.

Cấu trúc tế bào nhớ của LSTM (Hình 2.4.12):

- x_t : đầu vào ở thời điểm t;
- C_t : trạng thái bộ nhớ ở thời điểm t;
- h_t : đầu ra ở thời điểm t;



Hình 2.4.12: Cấu trúc tế bào nhớ của LSTM

- \tilde{C}_t : trạng thái phụ của tế bào nhớ;
- f_t : cổng quên;
- i_t : cổng đầu vào;
- o_t : cổng đầu ra;
- σ, \tanh : hàm kích hoạt.

Quá trình lan truyền thông tin của tế bào nhớ:

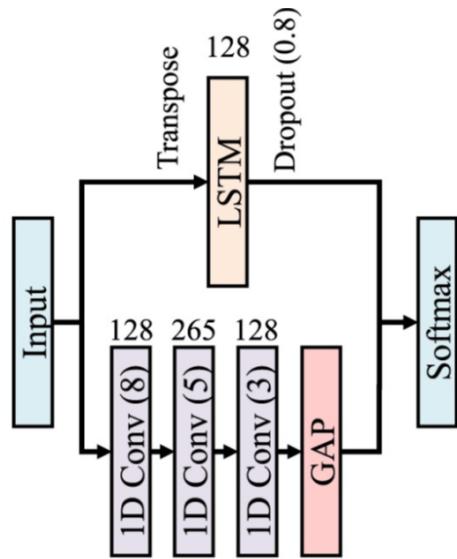
Về cơ bản, quá trình lan truyền này gồm 04 bước là:

- + Bước 1: mạng quyết định những thông tin nào cần bị loại khỏi trạng thái tế bào.
- + Bước 2: mạng quyết định xem những thông tin mới nào sẽ được lưu vào trạng thái tế bào.
- + Bước 3: cập nhật trạng thái tế bào cũ thành trạng thái mới.
- + Bước 4: quyết định những gì sẽ xuất ra dựa vào trạng thái của tế bào.

2.4.15 Long Short Term Memory with Fully Convolutional Network

Long Short Term Memory with Fully Convolutional Network hay *LSTM-FCN* là một cải tiến của FCN, nó mang lại hiệu suất cao hơn trong nhiệm vụ phân lớp chuỗi thời gian tuần tự.

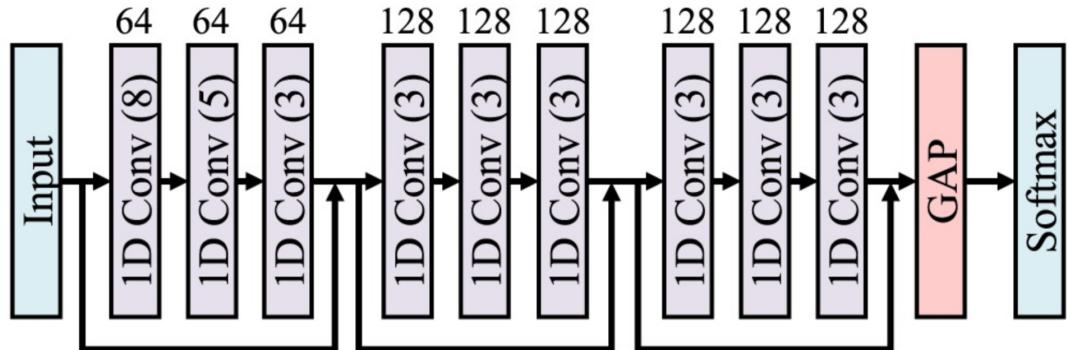
Cấu trúc mạng LSTM-FCN gồm hai phần là một khối LSTM và 3 khối FCN với 3 tầng chập.



Hình 2.4.13: Cấu trúc mạng LSTM-FCN

2.4.16 Residual Network

Residual Network hay *ResNet* là một kiến trúc của *CNN* được thiết kế hàng trăm hoặc hàng nghìn lớp chập. Tuy vậy, do có quá nhiều lớp chập trong mạng nên vấn đề *Vanishing Gradient* rất dễ xảy ra.



Hình 2.4.14: Cấu trúc mạng ResNet

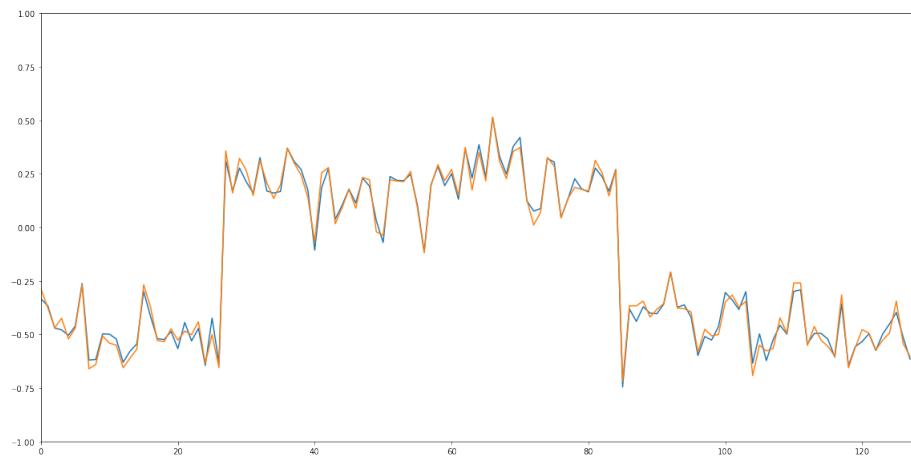
3 Các phương pháp sinh dữ liệu

3.1 Jittering

Định nghĩa: Jittering là việc cộng ngẫu nhiên một đại lượng nhỏ(nhiều) tại mỗi thời điểm của chuỗi thời gian:

$$x' = x_1 + \epsilon_1, \dots, x_t + \epsilon_t, \dots, x_T + \epsilon_T,$$

Trong đó, ϵ là thành phần nhiễu được thêm vào ở mỗi bước thời gian t , $\epsilon \sim N(0, \sigma^2)$. Thêm nhiễu ở đầu vào là một phương pháp phổ biến để tăng tính tổng quát của mạng nơ-ron. Có thể làm điều này một cách hiệu quả bằng cách tạo các mẫu mới với giả định rằng các mẫu test chỉ khác với các mẫu train bởi một yếu tố nhiễu.



Hình 3.1.1: Chuỗi thời gian ban đầu (đường màu xanh) và chuỗi thời gian đã qua jittering (đường màu cam)

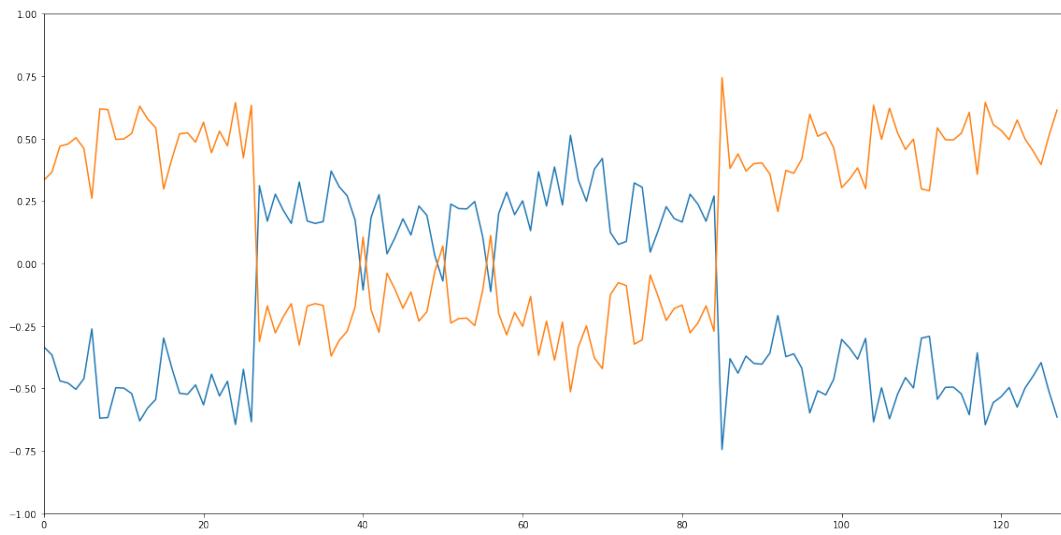
3.2 Rotation

Định nghĩa: Rotation là phương pháp “xoay” dữ liệu theo một góc bất kì để tạo một bộ dữ liệu mới, công thức của chuỗi thời gian mới được định nghĩa như sau:

$$x' = Rx_1, \dots, Rx_t, \dots, Rx_T$$

Trong đó, R là ma trận xoay ngẫu nhiên các phần tử một góc $\theta \sim N(0, \sigma^2)$ đối với chuỗi thời gian nhiều chiều và lật ngược đối với chuỗi thời gian một chiều.[3]

Phương pháp sinh dữ liệu này phù hợp cho việc phân loại, nhận dạng ảnh nhưng có thể sẽ không phù hợp cho chuỗi thời gian vì xoay dữ liệu có thể ảnh hưởng tới lớp/nhân của dữ liệu gốc.



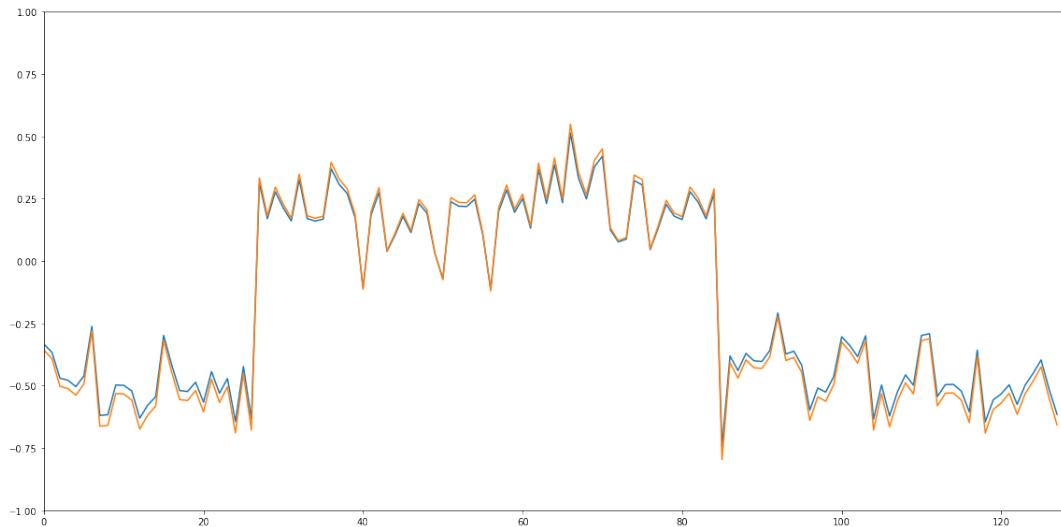
Hình 3.2.1: Chuỗi thời gian ban đầu (đường màu xanh) và chuỗi thời gian đã qua rotation (đường màu cam)

3.3 Scaling

Định nghĩa: Scaling là phương pháp thêm dữ liệu bằng cách nhân một đại lượng α tại mỗi thời điểm của chuỗi thời gian:

$$\mathbf{x}' = \alpha x_1, \dots, \alpha x_t, \dots, \alpha x_T,$$

Trong đó, α được xác định bởi phân phối Gauss: $\alpha \sim N(1, \sigma^2)$ với σ là siêu tham số (hyperparameter). [3]



Hình 3.3.1: Chuỗi thời gian ban đầu (đường màu xanh) và chuỗi thời gian đã qua scaling (đường màu cam)

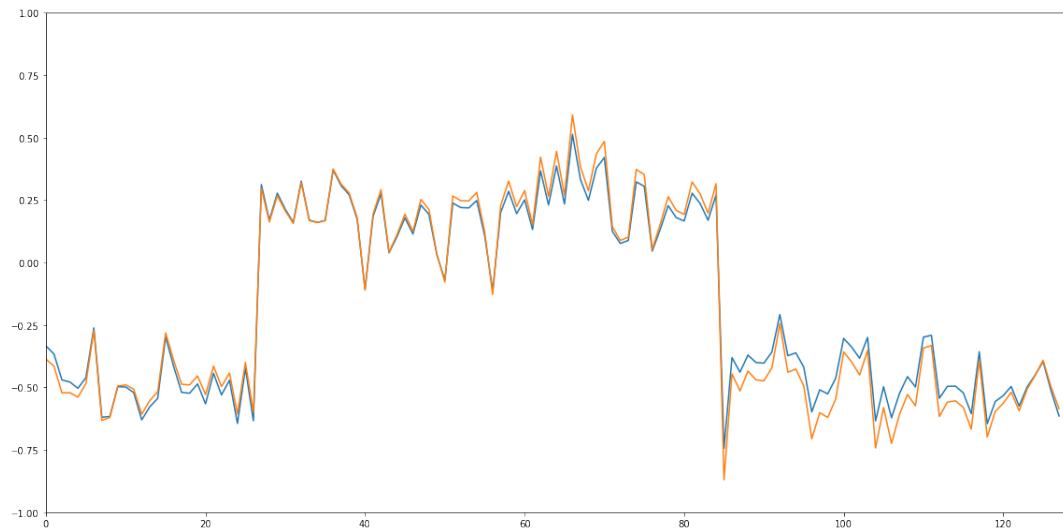
3.4 Magnitude Warping

Định nghĩa: Magnitude Warping[3] là phương pháp tăng cường dữ liệu cho chuỗi thời gian bằng cách làm cong giá trị của tín hiệu bằng đường cong trơn:

$$x' = \alpha_1 x_1, \dots, \alpha_t x_t, \dots, \alpha_T x_T,$$

$\alpha_1, \dots, \alpha_t, \dots, \alpha_T$ là dãy được tạo bằng cách sử dụng hàm nội suy cubic spline $S(u)$ với các nút chia $u = u_1, \dots, u_i, \dots, u_I$. Mỗi nút u_i được lấy ngẫu nhiên theo phân phối $N(l, \sigma^2)$, số nút l và σ là các siêu tham số.

Ý tưởng của phương pháp này là việc thêm giao động nhỏ vào dữ liệu bằng cách tăng hoặc giảm các vùng của chuỗi thời gian một cách ngẫu nhiên. Một nhược điểm của phương pháp này là việc phải thừa nhận phép biến đổi ngẫu nhiên là thực tế và phụ thuộc vào hai siêu tham số (số khoảng chia l và độ lệch chuẩn của mỗi khoảng σ) thay vì chỉ một như nhiều phương pháp biến đổi dữ liệu khác.



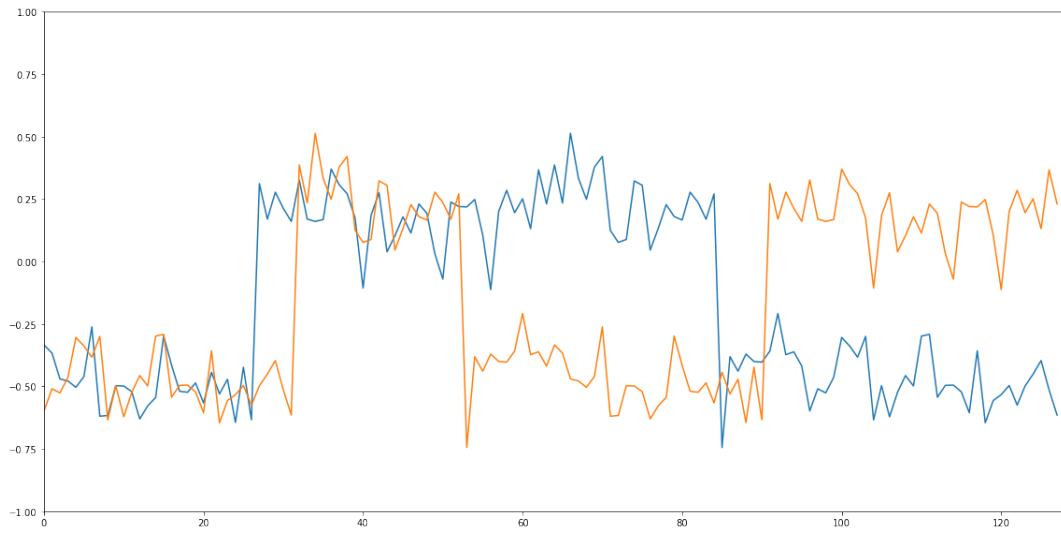
Hình 3.4.1: Chuỗi thời gian ban đầu (đường màu xanh) và chuỗi thời gian đã qua magnitute warping (đường màu cam)

3.5 Permutation

Định nghĩa: Permutation được đề xuất bởi Um và cộng sự[3] là phương pháp tăng cường dữ liệu bằng cách hoán đổi vị trí các đoạn của chuỗi thời gian nhằm tạo ra một mẫu mới.

Các đoạn chia có thể có độ dài bằng hoặc khác nhau. Sử dụng các đoạn chia có độ dài bằng nhau chia chuỗi thời gian thành N đoạn với độ dài $\frac{T}{N}$ (T là độ dài chuỗi thời gian, N là số đoạn chia) và hoán đổi vị trí giữa chúng.

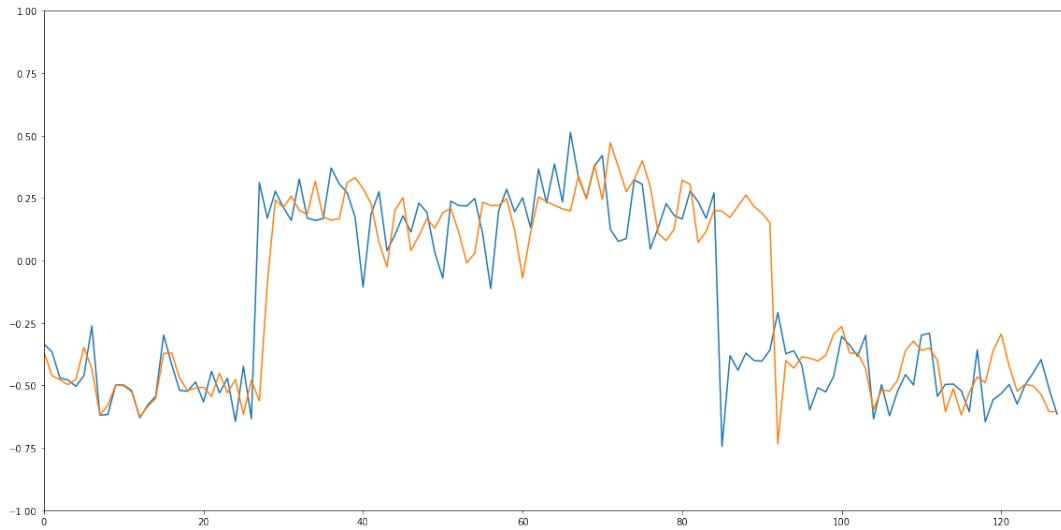
Một trường hợp của Permutation là Random Shuffle – hoán đổi các phần tử thay vì cả đoạn ($N = T$).



Hình 3.5.1: Chuỗi thời gian ban đầu (đường màu xanh) và chuỗi thời gian đã qua permutation (đường màu cam)

3.6 Window Slice

Định nghĩa: Window Slice[1] là phương pháp tăng cường dữ liệu bằng cách cắt ra 90% của chuỗi thời gian ban đầu, điểm bắt đầu được chọn một cách ngẫu nhiên. Để độ dài của chuỗi thời gian không đổi, ta dùng nội suy lên chuỗi thời gian vừa cắt để thu được chuỗi thời gian mới với độ dài như chuỗi ban đầu.



Hình 3.6.1: Chuỗi thời gian ban đầu (đường màu xanh) và chuỗi thời gian đã qua window slice (đường màu cam)

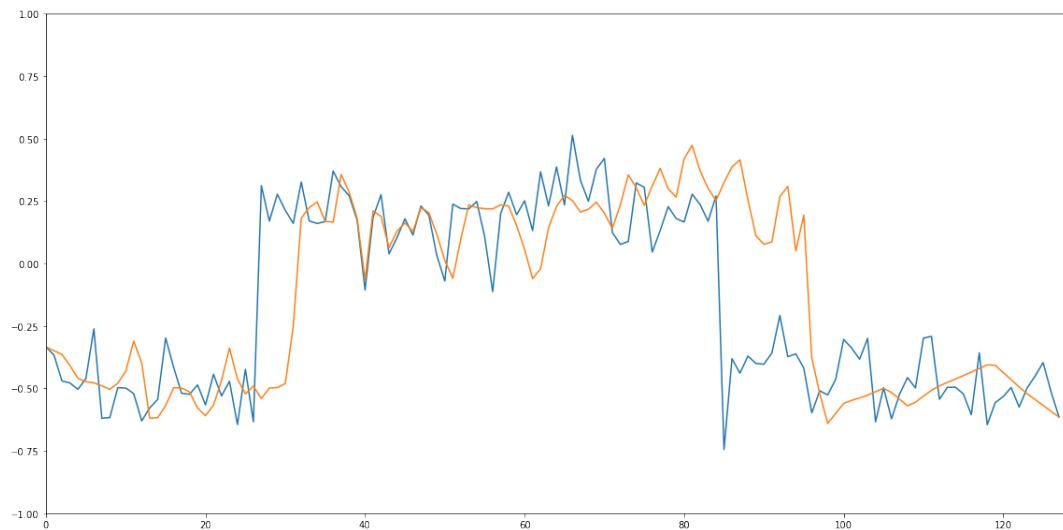
3.7 Time Warping

Định nghĩa: Time warping là phương pháp làm xáo trộn chuỗi thời gian theo trục thời gian. Điều này được thực hiện qua đường cong mịn[3], dữ liệu được tăng có dạng như sau:

$$x' = x_{\tau(1)}, \dots, x_{\tau(t)}, \dots, x_{\tau(T)},$$

Trong đó, $\tau(\cdot)$ là hàm làm cong vênh các bước thời gian dựa trên đường cong trơn. Đường cong trơn này được định nghĩa bởi cubic spline $S(u)$ với các nút thắt $u_1, \dots, u_i, \dots, u_l$, $u_i \sim N(1, \sigma^2)$.

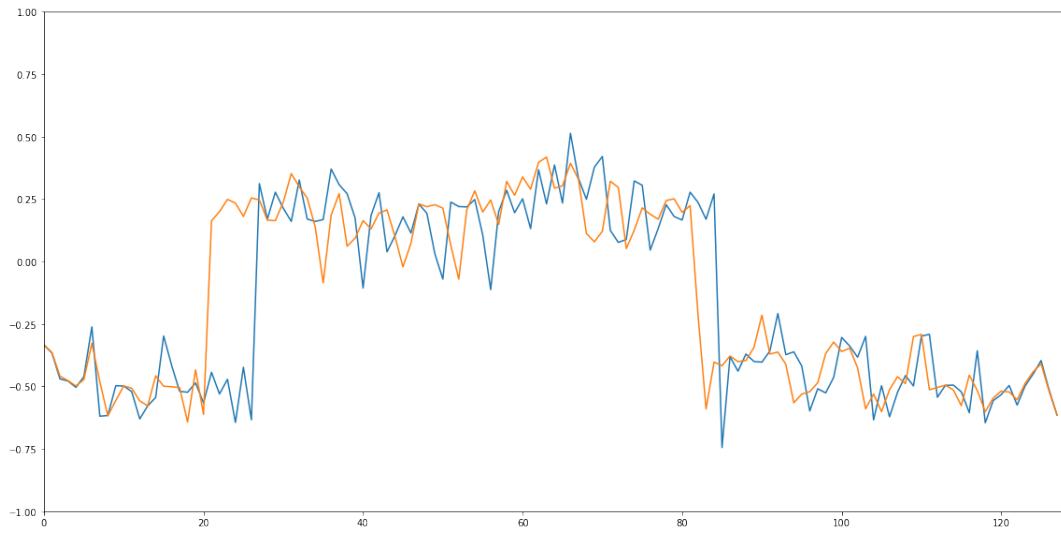
Bằng cách này, các bước thời gian trong chuỗi thời gian có sự thay đổi mượt mà giữa các đoạn co và giãn.



Hình 3.7.1: Chuỗi thời gian ban đầu (đường màu xanh) và chuỗi thời gian đã qua time warping (đường màu cam)

3.8 Window Warping

Định nghĩa: Là phương pháp sinh dữ liệu dựa trên Time Warping được đề xuất bởi Le Guennec và cộng sự[1], lấy một khung thời gian bất kì để kéo giãn ra gấp 2 hoặc co lại với tỉ lệ $\frac{1}{2}$. Các tỉ lệ này có thể được thay bằng giá trị khác để tối ưu hơn.



Hình 3.8.1: Chuỗi thời gian ban đầu (đường màu xanh) và chuỗi thời gian đã qua window warping (đường màu cam)

3.9 SubOptimal Warped time series geNEratoR (SPAWNER)

Định nghĩa: SubOptimal Warped time-series geNEratoR (SPAWNER)[5] là phương pháp tăng cường dữ liệu dựa trên việc hợp nhất các chuỗi thời gian theo đường căn chỉnh của Dynamic Time Warping (DTW).

Giả sử ta có bộ dữ liệu $U = (X_1, C_1), (X_2, C_2), \dots, (X_N, C_N)$ trong đó X_1, X_2, \dots, X_N là các chuỗi thời gian và C_1, C_2, \dots, C_N là nhãn tương ứng của chúng.

Để thực hiện SPAWNER, ta cần hai chuỗi thời gian cùng nhãn. Để làm điều này ta chọn một chuỗi thời gian từ bộ dữ liệu và chọn ngẫu nhiên một chuỗi thời gian khác cùng lớp không trùng với nó. Không giảm tổng quát, hai chuỗi thời gian vừa chọn là X_1 và X_2 .

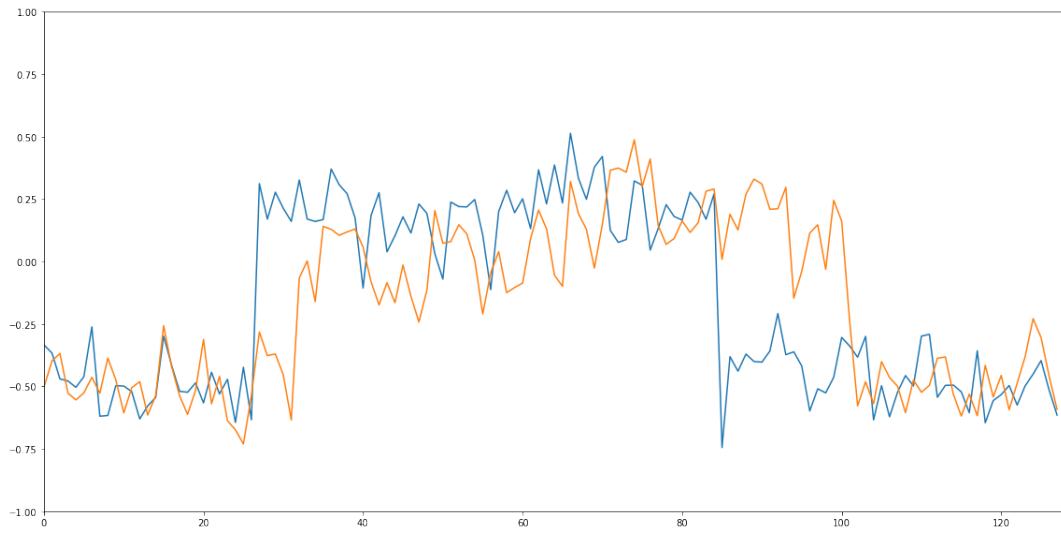
$$X_1 = \{x_1^0, x_1^1, \dots, x_1^{L_1-1}\}, X_2 = \{x_2^0, x_2^1, \dots, x_2^{L_2-1}\}$$

Sử dụng DTW với hai chuỗi thời gian thu được đường căn chỉnh $W = \{w_1, w_2, \dots, w_P\}$. Hợp nhất hai chuỗi thời gian trên bằng cách lấy giá trị trung bình các phần tử tương ứng của hai chuỗi $\rightarrow X^*$.

X^* là chuỗi thời gian mới có độ dài P. Trong bài này, các chuỗi thời gian có độ dài như nhau: $L_1 = L_2 = L$. Ta có thể sử dụng hàm nội suy để biến đổi X^* có độ dài P thành X^{**} có độ dài L.

X^* cũng có thể lấy sao cho $x^* \in X^*$:

$$x^* \sim N(\mu, \sigma^2), \mu = 0.5(x_1^* + x_2^*), \sigma = 0.05\|x_1^* - x_2^*\|$$



Hình 3.9.1: Chuỗi thời gian ban đầu (đường màu xanh) và chuỗi thời gian đã qua spawner (đường màu cam)

3.10 Weighted DTW Barycentric Averaging (wDBA)

Định nghĩa: wDBA[2] là phiên bản phát triển hơn của DBA bao gồm trọng số. Chuỗi thời gian tìm được tối thiểu hóa hàm sau đây:

$$\operatorname{argmin} \bar{T} \in E \sum_{i=1}^N w_i \cdot DTW^2(\bar{T}, T_i)$$

T là chuỗi thời gian rung bình, N là số chuỗi thời gian trong bộ dữ liệu, w_i là trọng số của chuỗi thời gian thứ i . $DTW^2(T, T_i)$ là khoảng cách căn chỉnh.

Có nhiều phương pháp khác nhau để cài đặt wDBA như AA, AS, ASD.

AA hay Average All tính trung bình tất cả chuỗi thời gian trong bộ dữ liệu, gán cho mỗi chuỗi trọng số w_i tuân theo phân phối Dirichlet.

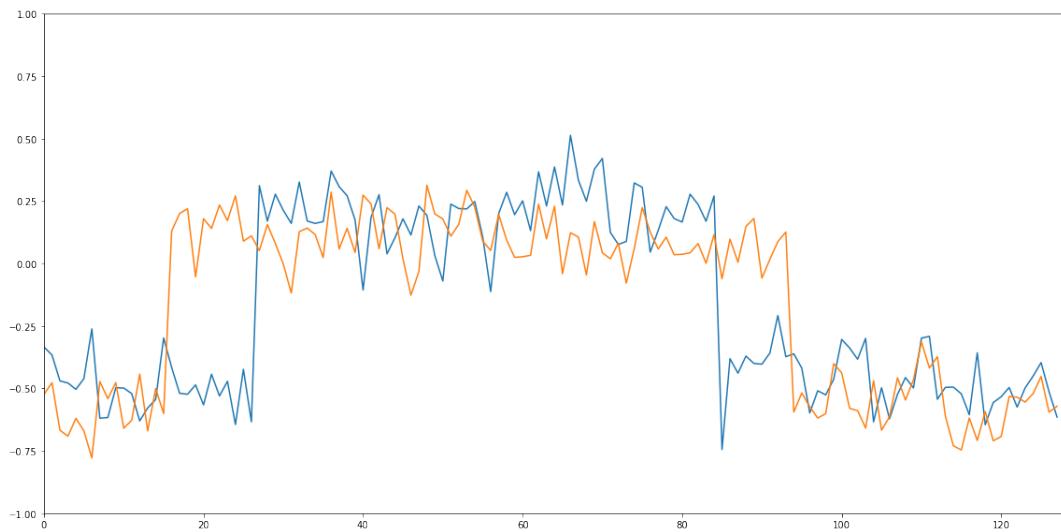
AS hay Average Selected lại chọn một chuỗi thời gian bất kỳ, gán trọng số 0.5 cho nó. Sau đó chọn 5 chuỗi thời gian gần nhất rồi lại chọn ngẫu nhiên 2 chuỗi thời gian trong 5 chuỗi đó để gán trọng số 0.15. Tổng ba chuỗi thời gian vừa chọn có trọng số 0.8. Cuối cùng, ta chia đều 0.2 trong số cho các chuỗi thời gian còn lại.

ASD hay Average Selected with Distance cũng chung tinh thần với AS nhưng trọng số được tính toán dựa theo khoảng cách với các hàng xóm gần nhất của chuỗi thời gian được chọn ban đầu. Đầu tiên, ta chọn chuỗi thời gian T^* và gán trọng số 1, tìm hàng xóm gần nhất và gán cho nó trọng số 0.5. Sau đó gán trọng số cho các chuỗi còn lại theo công thức sau:

$$w_i = e^{\ln(0.5) * \frac{DTW(T_i, T^*)}{d_{NN}^*}}$$

Trong đó, d_{NN}^* là khoảng cách giữa T^* và hàng xóm gần nhất. Với cách đánh trọng số này, những chuỗi thời gian gần T^* sẽ có tác động mạnh hơn các chuỗi thời gian ở xa.

Ưu thế của wDBA so với DBA là do DBA là thuật toán tất định (deterministic), nghĩa là với một tham số đầu vào sẽ luôn cho ra một kết quả duy nhất. Với việc cho thêm trọng số ngẫu nhiên, ta có thể tạo ra vô số các chuỗi thời gian khác nhau chỉ từ hai chuỗi thời gian.

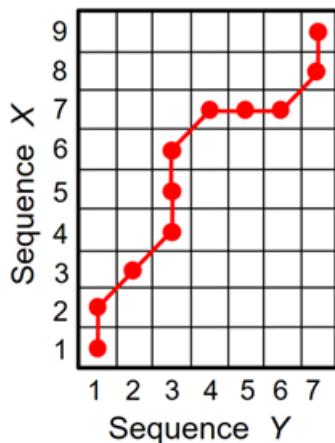


Hình 3.10.1: Chuỗi thời gian ban đầu (đường màu xanh) và chuỗi thời gian đã qua wdba (đường màu cam)

3.11 Random Guided Warping (RGW)

Định nghĩa: RGW hay Random Guided Warping[7] là phương pháp tăng cường dữ liệu thực hiện bằng cách chọn hai chuỗi thời gian cùng lớp ngẫu nhiên, sử dụng dtw tìm được đường cǎn chỉnh và dựa vào nó để cǎn chỉnh chuỗi thời gian 1 theo trực thời gian cǎn chỉnh của chuỗi thời gian 2.

Ví dụ, ta tìm được đường cǎn chỉnh như sau của hai chuỗi thời gian X, Y:



$$X = [x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9]$$

$$Y = [y_1, y_2, y_3, y_4, y_5, y_6, y_7]$$

- Trục thời gian cǎn chỉnh của X là:

$$[12, 3, 4, 5, 6, 7, 7, 7, 8, 9]$$

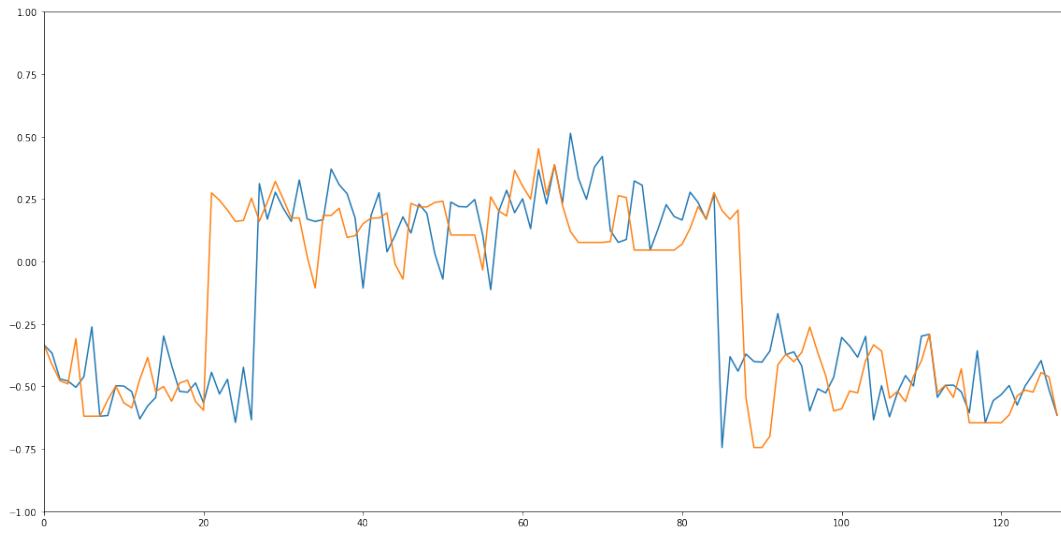
- Trục thời gian cǎn chỉnh của Y là:

$$[1, 1, 2, 3, 3, 3, 4, 5, 6, 7, 7]$$

- Ta lấy giá trị chuỗi thời gian X ứng với các mốc thời gian của chuỗi Y => chuỗi mới được sinh ra là:

$$[x_1, x_1, x_2, x_3, x_3, x_3, x_4, x_5, x_6, x_7, x_7]$$

Lợi thế của phương pháp này là các giá trị của chuỗi thời gian tồn tại trong bộ dữ liệu gốc. Các phương pháp sinh ngẫu nhiên khác thì không chắc chắn dữ liệu là thực tế.



Hình 3.11.1: Chuỗi thời gian ban đầu (đường màu xanh) và chuỗi thời gian đã qua rgw (đường màu cam)

3.12 Discriminative Guided Warping (DGW)

Định nghĩa: Discriminative Guided Warping[7] là phương pháp tăng cường dữ liệu dựa trên nguyên lý của Random Guided Warping nhưng khác biệt ở việc chọn chuỗi thời gian ban đầu phân biệt nhất.

Để chọn chuỗi thời gian ban đầu này, đầu tiên ta lấy ngẫu nhiên B chuỗi thời gian của bộ dữ liệu ban đầu, $B \gg N$ với N là số chuỗi thời gian trong bộ dữ liệu. Gọi tập con này là bootstrap.

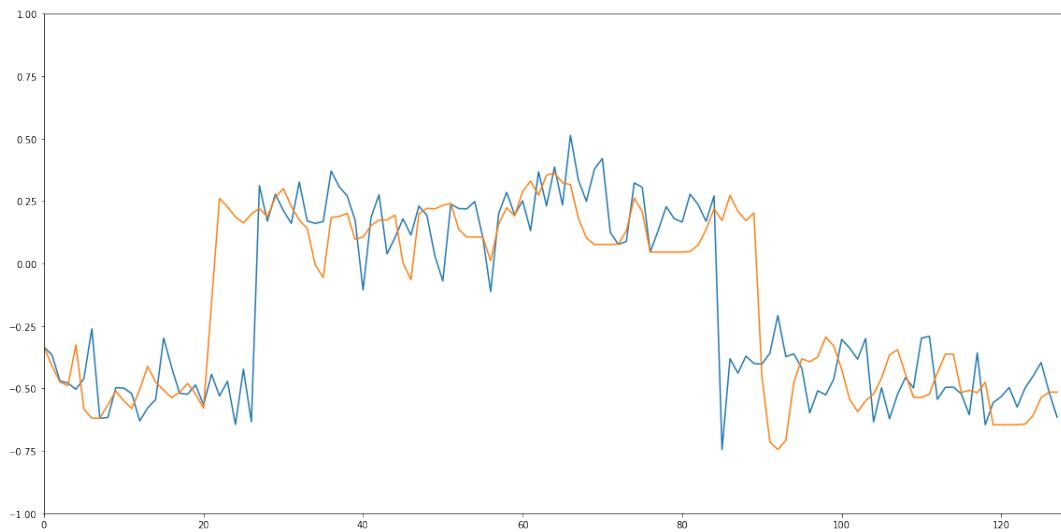
Duyệt lần lượt các chuỗi thời gian trong bootstrap, với mỗi lần duyệt ta chia các chuỗi còn lại trong bootstrap thành hai phần là Positive và Negative. Positive chứa các chuỗi thời gian cùng lớp với chuỗi đang duyệt và Negative chứa các chuỗi thời gian khác lớp với chuỗi đang duyệt. Rồi xét giá trị sau:

$$h(b_m) = \frac{1}{\sum_{m'} [l_{m'} \neq l_m]} \sum_m D(b_{m'}, b_m) |[l_{m'} \neq l_m] - \frac{1}{\sum_{m'} [l_{m'} = l_m]} \sum_m D(b_{m'}, b_m) |[l_{m'} = l_m],$$

b_m là chuỗi thời gian đang duyệt, $b_{m'}$ là các chuỗi thời gian khác. Sau khi duyệt hết các chuỗi thời gian trong bootstrap, chọn chuỗi b_{disc} sao cho:

$$b_{disc} = \underset{\{m=1, \dots, M\}}{\operatorname{argmax}} h(b_m),$$

Sau khi chọn xong chuỗi thời gian ban đầu, ta tiến hành chọn ngẫu nhiên một chuỗi thời gian từ tập dữ liệu ban đầu và thực hiện biến đổi như với RGW. Lấy giá trị của chuỗi được chọn ban đầu và trực thời gian căn chỉnh của chuỗi thời gian thứ hai được chọn ngẫu nhiên cho ta chuỗi thời gian mới để thêm vào bộ dữ liệu.



Hình 3.12.1: Chuỗi thời gian ban đầu (đường màu xanh) và chuỗi thời gian đã qua dgw (đường màu cam)

4 Thực nghiệm và kết luận

4.1 Dữ liệu

Thông tin chung về kho lưu trữ những bộ dữ liệu được sử dụng:

- UCR Suite được phát triển bởi:

+ UC Riverside: Thanawin Rakthanmanon, Bilson Campana, Abdullah Mueen, Qiang Zhu, Jesin Zakaria, Eamonn Keogh

+ University of São Paulo: Gustavo Batista

+ Brigham and Women's Hospital: Brandon Westover

+ Các tác giả Rakthanmanon, Campana, Mueen và Batista đóng góp như nhau, và nên được coi là tác giả đầu tiên chung.

- The UCR Suite: được tài trợ bởi NSF IIS - 1161997 II.

+ NSF (National Science Foundation):

+ IIS (Information and intelligent System): nghiên cứu các vai trò liên quan lẫn nhau của con người, máy tính và thông tin để tăng khả năng hiểu dữ liệu, cũng như bắt chước các dấu hiệu của trí thông minh trong các hệ thống tính toán.

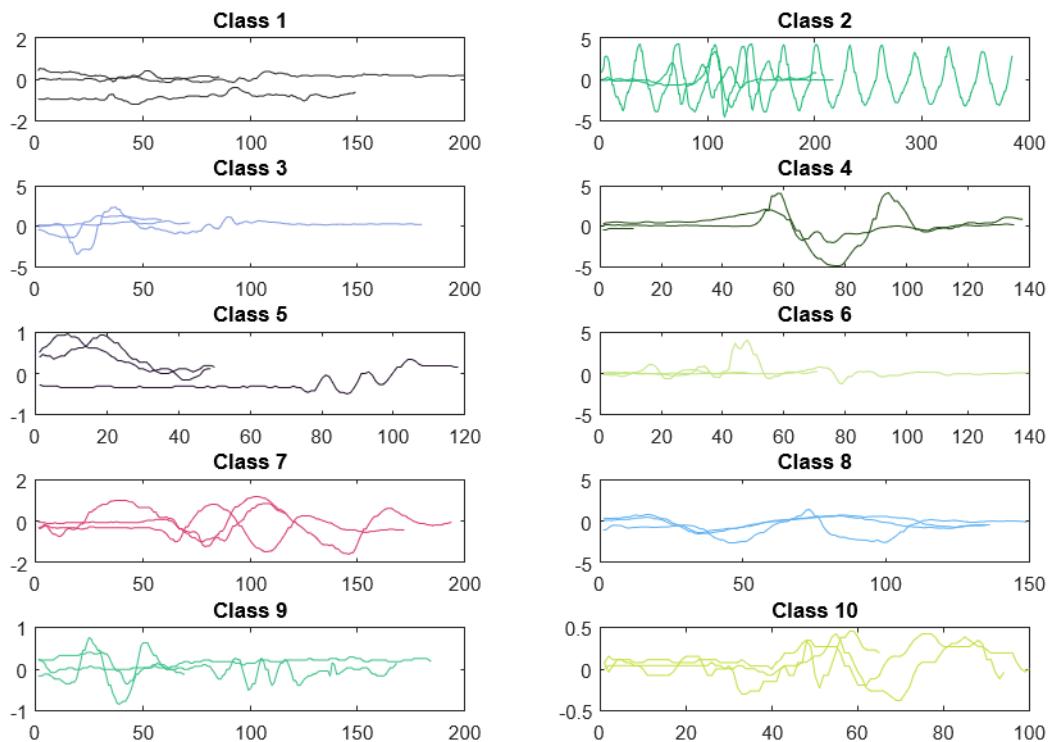
- Các bộ dữ liệu trong UCR Suite đều đã được phân lớp.

Các bộ dữ liệu được sử dụng trong paper:

Cách đọc: Có một ví dụ về chuỗi thời gian cho mỗi hàng. Giá trị đầu tiên trong hàng là nhãn lớp (một số nguyên từ 1 đến số lớp). Phần còn lại của hàng là các giá trị mẫu dữ liệu. Thứ tự của chuỗi thời gian mẫu không mang ý nghĩa đặc biệt và trong hầu hết các trường hợp là ngẫu nhiên. Một số ít tập dữ liệu có nhãn lớp bắt đầu từ 0 hoặc -1 theo kế thừa.

Bộ dữ liệu được sử dụng để đánh giá độ hiệu quả của các phương pháp:

+ CBF(Cylinder-Bell-Funnel): là một tập dữ liệu mô phỏng được Naoki Saiko xác định trong luận án Tiến sĩ của mình. Dữ liệu từ mỗi lớp là tiếng ồn thông thường tiêu chuẩn cộng với

AllGestureWiimoteXThree exemplars per class,
without z-normalization

Hình 4.1.1: Các class của 1 bộ dữ liệu

This instance
is in class 1

This instance
is in class 2

1	1.000000e+00	1.1806977e+00	1.0871205e+00	9.2358964e-01	7.0391720e-01	6.0103560e-01	
2	1.000000e+00	-3.3690280e-03	1.3979743e-01	2.4634900e-01	2.8267125e-01	2.4853416e-01	
3	1.000000e+00	2.9279457e-01	3.5135336e-01	4.1774429e-01	4.5661213e-01	4.5986320e-01	
4	1.000000e+00	4.1583091e-01	3.1032058e-01	2.6886606e-01	3.7778743e-01	5.3290599e-01	
5	1.000000e+00	2.8197346e-01	2.3803050e-01	2.1128170e-01	2.2135987e-01	2.2535374e-01	
6	1.000000e+00	-9.0846439e-02	-1.6569281e-02	-3.6058030e-03	-6.6683031e-02	-3.4445864e-02	
7	1.000000e+00	4.8517572e-01	5.4267337e-01	6.7034509e-01	8.5107895e-01	9.1213376e-01	
8	2.000000e+00	1.2023614e+00	1.3947892e+00	1.4275099e+00	1.4241362e+00	1.4028836e+00	
9	2.000000e+00	5.1106447e-01	7.6259013e-01	9.6319197e-01	1.1224429e+00	1.2362984e+00	
10	2.000000e+00	7.5517824e-01	7.9050020e-01	8.9504039e-01	1.0803650e+00	1.1627584e+00	
11	2.000000e+00	1.9840994e-01	3.3180604e-01	5.1876476e-01	7.4746243e-01	9.3133439e-01	
12	2.000000e+00	9.2186744e-01	1.0271740e+00	1.1180091e+00	1.1847472e+00	1.1496392e+00	
13	2.000000e+00	8.8216014e-01	1.0909846e+00	1.2980519e+00	1.3590758e+00	1.2844307e+00	
14	2.300000e+00	5.2002435e-01	6.0897439e-01	7.2557887e-01	8.8508781e-01	1.0142352e+00	
15	2.000000e+00	9.7588532e-01	1.1256302e+00	1.2858246e+00	1.4679852e+00	1.6140577e+00	
16	2.000000e+00	1.6684440e-01	2.8908015e-01	4.6257987e-01	5.4681370e-01	5.7076904e-01	
17	2.000000e+00	9.8064177e-01	9.9478049e-01	1.0076349e+00	1.0515163e+00	1.0459247e+00	
18	2.000000e+00	8.3017593e-01	1.0461612e+00	1.2449432e+00	1.3195458e+00	1.3018360e+00	
19	3.000000e+00	2.7181028e-01	4.6021806e-01	6.5528037e-01	8.4635225e-01	9.5545865e-01	
20	3.000000e+00	5.3942622e-01	7.2130396e-01	9.5383293e-01	1.0628592e+00	1.0408570e+00	
21	3.000000e+00	3.2801034e-01	2.9006955e-01	2.6270711e-01	3.3797013e-01	4.3132201e-01	

Hình 4.1.2: Thành phần của 1 bộ dữ liệu

một thuật ngữ bù đắp khác nhau cho mỗi lớp.

i. Cố tập đào tạo : 30

- ii. Cỡ tập huấn luyện: 900
- iii. Giá trị bị thiếu: 0
- iv. Số nhãn: 3
- v. Độ dài chuỗi thời gian: 128

4.2 Phương pháp thực nghiệm

Trong bài báo cáo này, Nhóm chỉ mong muốn đưa ra cách thức thực hiện các phương pháp trong thực tiễn mà không đề xuất phương pháp nào là tốt nhất trong thực tế. Lý do là mỗi tập dữ liệu có những đặc trưng riêng, mà ta không thể khái quát hết những đặc trưng đó trong một tập dữ liệu để thực nghiệm.

Do vậy, Nhóm đã sử dụng một tập dữ liệu đại diện để đề ra cách thức thực hiện các phương pháp sinh dữ liệu.

Ban đầu, chúng ta có một tập dữ liệu gốc (đã được đề cập ở phần 4.1) với kích thước tập train và test tương ứng là 30 và 900 bộ dữ liệu. Có thể sẽ có nhiều người sẽ nghĩ rằng Nhóm đã "viết nhầm" kích thước các tập, nhưng thực tế là không.

Trong bài báo cáo này, Nhóm mong muốn sử dụng một tập dữ liệu đủ nhỏ để sinh ra một bộ dữ liệu có kích thước lớn hơn nhiều lần so với bộ dữ liệu ban đầu.

Cách thức thực hiện của Nhóm là tương đối đơn giản:

- i Nhóm đưa bộ dữ liệu gốc lần lượt đi qua 12 phương pháp sinh dữ liệu, từ đó thu được 13 bộ dữ liệu bao gồm cả bộ dữ liệu gốc. Ở đây ta chú ý rằng, kích thước tập huấn luyện sau khi sinh dữ liệu là **3000** mới mong muốn tỉ lệ tập train-test trở về "*tỉ lệ vàng*" trong các bài toán Máy học.
- ii Tiếp theo, 13 bộ dữ liệu sẽ được đi qua các mô hình máy học và học sâu để thu được kết quả là độ chính xác dự đoán của các mô hình với từng bộ dữ liệu.
- iii Từ đó, Nhóm đưa ra kết quả, kết luận và tổng kế.

4.3 Kết quả thực nghiệm

Sau khi tiến hành thực hiện các bước trên, ta thu được kết quả như bảng 4.3.1 và 4.3.2. Chúng ta sẽ phân tích đôi chút về kết quả thu được.

Đầu tiên ta xét nhóm các mô hình máy học, điều đầu tiên ta thấy khá nổi bật chính là việc kết quả thu được của tập dữ liệu gốc (với chỉ 30 bản ghi) nhưng thu được kết quả tốt không ngờ. Độ chính xác của chúng hầu hết đều giao động ở mức xấp xỉ 80%, và cao nhất lên tới gần 90%.

Cũng trong nhóm này, chúng ta thấy rằng không có bất cứ một phương pháp nào cho được kết quả tốt nhất ở tất cả mô hình. Ví như mô hình *Logistic Regression* đem lại độ chính xác trong dữ đoán của tập dữ liệu sử dụng phương pháp **Slicing** là 95% - cao nhất trong các phương pháp, nhưng đối với mô hình *Linear Discriminant Analysis* thì "quán quân" về hiệu quả lại là **Rotation**.

Tuy nhiên, có một điều mà ta dễ dàng nhận thấy chính là việc sẽ luôn có **ít nhất** một phương pháp đem lại độ chính xác **cao hơn** so với việc sử dụng bộ dữ liệu ban đầu. Điều này phần nào đã chứng minh được mức độ hiệu quả của các phương pháp.

Tiếp theo, ta sẽ xét tới nhóm các mô hình học sâu, chúng ta cũng thấy được ngay rằng độ chính xác nếu như dùng dữ liệu gốc là rất nhỏ. Lý do ở đây chính là việc với tập dữ liệu nhỏ thì việc tối ưu các tham số của mạng nơ-ron là hoàn toàn bất khả thi do một mạng nơ-ron thông thường có thể có tới hàng triệu tham số.

Ta cũng thấy được rằng bên cạnh những phương pháp cho ra độ chính xác thấp thì có những phương pháp mang lại kết quả "không tưởng", thậm chí độ chính xác còn xấp xỉ tới *ngưỡng tuyệt đối*. Đây là một tín hiệu đáng mừng cho thấy rằng tập dữ liệu được sinh ra có thể áp dụng một cách hiệu quả trong quá trình đào tạo mô hình mạng nơ-ron.

	None	Jittering	Scaling	Rotation	Permutation	Magnitude Warping
Logistic Regression	0.85	0.85	0.84	0.46	0.34	0.86
Linear Discriminant Analysis	0.84	0.87	0.79	0.96	0.36	0.41
Gaussian Naive Bayes	0.89	0.90	0.90	0.58	0.86	0.91
Decision Tree Classifier	0.76	0.80	0.72	0.47	0.54	0.73
Random Forest Classifier	0.87	0.93	0.93	0.78	0.84	0.95
KNN	0.77	0.85	0.85	0.85	0.84	0.85
Ada Boost	0.81	0.81	0.73	0.49	0.49	0.79
XGBoost	0.77	0.82	0.85	0.54	0.73	0.8
Light GBM	0.33	0.72	0.81	0.60	0.68	0.75
VGG	0.41	1.0	1.0	0.84	0.98	0.99
LSTM-1	0.33	0.43	0.53	0.33	0.33	0.44
LSTM	0.51	0.33	0.53	0.52	0.33	0.55
LSTM-2	0.33	0.33	0.52	0.51	0.44	0.40
BLSTM	0.33	0.33	0.35	0.52	0.34	0.44
BLSTM-2	0.42	0.48	0.76	0.81	0.77	0.94
LSTM-FCN	0.33	0.36	0.61	0.95	0.70	0.57
Res-Net	0.35	0.33	0.33	0.33	0.33	0.33
MLP	0.5	0.90	0.92	0.87	0.88	0.89
Le-Net	0.36	0.89	0.90	0.84	0.88	0.90

Bảng 4.3.1: Kết quả thực nghiệm các phương pháp sinh dữ liệu (phần 1)

	Time Warping	Slicing	Window Warping	SPAWNER	wDBA	RGW	DGW
Logistic Regression	0.75	0.95	0.89	0.84	0.83	0.89	0.91
Linear Discriminant Analysis	0.78	0.94	0.78	0.84	0.84	0.78	0.89
Gaussian Naive Bayes	0.82	0.91	0.88	0.90	0.78	0.90	0.90
Decision Tree Classifier	0.73	0.83	0.83	0.78	0.72	0.77	0.79
Random Forest Classifier	0.95	0.99	0.95	0.95	0.91	0.97	0.99
KNN	1.0	0.97	0.97	0.91	0.84	0.93	0.98
Ada Boost	0.66	0.81	0.82	0.90	0.70	0.80	0.89
XGBoost	0.75	0.94	0.88	0.94	0.78	0.88	0.95
Light GBM	0.56	0.91	0.85	0.91	0.74	0.88	0.92
VGG	0.99	0.99	0.98	0.96	0.97	0.98	0.97
LSTM-1	0.61	0.69	0.47	0.47	0.48	0.55	0.53
LSTM	0.57	0.59	0.78	0.82	0.36	0.48	0.51
LSTM-2	0.55	0.62	0.67	0.75	0.82	0.76	0.87
BLSTM	0.51	0.49	0.72	0.70	0.77	0.75	0.81
BLSTM-2	0.97	0.93	0.97	0.99	0.98	0.97	0.96
LSTM-FCN	0.95	0.97	0.95	0.60	0.99	0.95	0.91
Res-Net	0.33	0.38	0.61	0.59	0.98	0.99	0.96
MLP	0.89	0.96	0.95	0.95	0.93	0.95	0.95
Le-Net	0.91	0.95	0.95	0.97	0.97	0.98	0.97

Bảng 4.3.2: Kết quả thực nghiệm các phương pháp sinh dữ liệu (phần 2)

4.4 Kết luận

Qua thực nghiệm trên, một lần nữa ta thấy được tầm quan trọng của quy mô dữ liệu và chất lượng của dữ liệu đối với một mô hình học sâu. Mô hình học sâu có thể đem lại những kết quả rất tốt nếu tập dữ liệu là đủ tốt, nhưng cũng có thể đem lại kết quả tồi tệ nếu tập dữ liệu chưa đủ đáp ứng của mô hình. Do vậy, trong trường hợp quy mô dữ liệu ít thì ta nên cân nhắc việc sử dụng các mô hình máy học.

Ta cũng thấy rằng các tập dữ liệu sinh ra từ phương pháp sinh dữ liệu đều đem lại hiệu quả tích cực so với tập dữ liệu. Điều này thực sự là một tín hiệu đáng mừng vì hiện nay mặc dù

kho dữ liệu của nhân loại là rất lớn, nhưng lượng dữ liệu chúng ta có thể thu thập với một mục đích cụ thể là không quá lớn, và chi phí thu thập, xử lý dữ liệu cũng là một vấn đề để thúc đẩy sự phát triển của các phương pháp sinh dữ liệu.

Tài liệu tham khảo

- [1] Arthur Le Guennec, Simon Malinowski, and Romain Tavenard. “Data Augmentation for Time Series Classification using Convolutional Neural Networks”. In: 2016.
- [2] Germain Forestier et al. “Generating synthetic time series to augment sparse datasets”. In: *2017 IEEE international conference on data mining (ICDM)*. IEEE. 2017, pp. 865–870.
- [3] Terry T. Um et al. “Data augmentation of wearable sensor data for parkinson’s disease monitoring using convolutional neural networks”. In: *Proceedings of the 19th ACM International Conference on Multimodal Interaction* (Nov. 2017). DOI: 10.1145/3136755.3136817. URL: <http://dx.doi.org/10.1145/3136755.3136817>.
- [4] Aston Zhang; Mu Li; Zachary C Lipton and Alexander J Smola. *Dive into Deep Learning*. 1st ed. Corwin, 2019. ISBN: 9781544361376.
- [5] Krzysztof Kamycki, Tomasz Kapuscinski, and Mariusz Oszust. “Data augmentation with suboptimal warping for time-series classification”. In: *Sensors* 20.1 (2020), p. 98.
- [6] Vũ Hữu Tiệp. *Machine Learning cơ bản*. 2020.
- [7] Brian Kenji Iwana and Seiichi Uchida. “Time series data augmentation for neural networks by time warping with a discriminative teacher”. In: *2020 25th International Conference on Pattern Recognition (ICPR)*. IEEE. 2021, pp. 3558–3565.