

TRƯỜNG ĐẠI HỌC MỎ ĐỊA CHẤT
KHOA TRẮC ĐỊA BẢN ĐỒ VÀ QUẢN LÝ ĐẤT ĐAI
BỘ MÔN TRẮC ĐỊA CAO CẤP

BÁO CÁO HỌC THUẬT

GIẢI PHÁP DỰ ĐOÁN SỬ DỤNG MÔ HÌNH TRÍ TUỆ NHÂN TẠO
VỚI GIẢI PHÁP CHUẨN HÓA DỮ LIỆU ĐẦU VÀO KHÁC NHAU

Hà Nội, 2025

TRƯỜNG ĐẠI HỌC MỎ ĐỊA CHẤT
KHOA TRẮC ĐỊA BẢN ĐỒ VÀ QUẢN LÝ ĐẤT ĐAI
BỘ MÔN TRẮC ĐỊA CAO CẤP

BÁO CÁO HỌC THUẬT
GIẢI PHÁP DỰ ĐOÁN SỬ DỤNG MÔ HÌNH TRÍ TUỆ NHÂN TẠO
VỚI GIẢI PHÁP CHUẨN HÓA DỮ LIỆU ĐẦU VÀO KHÁC NHAU

Báo cáo viên: TS. Nguyễn Gia Trọng

Hà Nội, 2025

MỤC LỤC

1. Mở đầu	4
2. Giới thiệu về mô hình trí tuệ nhân tạo được đề xuất	6
2.1 Mô hình CNN(Conv1D)	6
2.2 Các thành phần khi xây dựng mô hình CNN(conv1D)	8
2.3 Các giải pháp chuẩn hóa dữ liệu đầu vào	10
3. Giải pháp xử lý dữ liệu	12
3.1 Anaconda	12
3.2 Spyder	13
3.3 Python	14
4. Dữ liệu sử dụng	15
5. Kết quả dự đoán	16
5.1 <i>Kết quả chuẩn hóa với Min-Max Scaling</i>	16
5.2 <i>Kết quả chuẩn hóa với Z-Score</i>	17
5.3 <i>Kết quả chuẩn hóa với Robust Scaling</i>	18
Tài liệu tham khảo	19

GIẢI PHÁP DỰ ĐOÁN SỬ DỤNG MÔ HÌNH TRÍ TUỆ NHÂN TẠO VỚI GIẢI PHÁP CHUẨN HÓA DỮ LIỆU ĐẦU VÀO

Nguyễn Gia Trọng, Bộ môn Trắc địa cao cấp

Email: nguyengiatrong@humg.edu.vn

Điện thoại: 0963124980

1. Mở đầu

Mô hình hóa chuỗi thời gian từ lâu đã là một lĩnh vực nghiên cứu quan trọng, đóng vai trò thiết yếu trong nhiều ứng dụng như mô hình khí hậu, khoa học sinh học, y học, cũng như trong các quyết định kinh doanh trong lĩnh vực bán lẻ và tài chính. Trong khi các phương pháp truyền thống tập trung vào các mô hình tham số dựa trên hiểu biết chuyên môn, chẳng hạn như mô hình tự hồi quy (AR), làm mượt hàm mũ hoặc mô hình chuỗi thời gian cấu trúc, thì các phương pháp học máy hiện đại cho phép học trực tiếp động học theo thời gian theo hướng hoàn toàn dựa trên dữ liệu. Với sự gia tăng mạnh mẽ của dữ liệu và năng lực tính toán trong những năm gần đây, học máy đã trở thành một thành phần quan trọng trong thể hệ mô hình dự báo chuỗi thời gian tiếp theo [1].

Việc dự báo chuỗi thời gian là một nhiệm vụ khó. Khác với các bài toán phân loại hay hồi quy vốn đơn giản hơn, chuỗi thời gian đòi hỏi phải xử lý thêm yếu tố thứ tự và sự phụ thuộc theo thời gian giữa các quan sát. Điều này gây khó khăn vì dữ liệu cần được xử lý theo những cách chuyên biệt khi xây dựng và đánh giá mô hình. Tuy vậy, cấu trúc thời gian này cũng mang lại lợi thế, chẳng hạn như xu hướng và tính mùa vụ, giúp cải thiện chất lượng dự báo nếu được khai thác đúng cách.

Truyền thống dự báo chuỗi thời gian lâu nay chủ yếu dựa vào các phương pháp tuyến tính như ARIMA vì chúng dễ hiểu và hiệu quả với nhiều bài toán. Tuy nhiên, các phương pháp kinh điển này tồn tại nhiều hạn chế, bao gồm:

- + Yêu cầu dữ liệu đầy đủ: dữ liệu bị thiếu hoặc hỏng thường không được hỗ trợ tốt.
- + Giả định mối quan hệ tuyến tính: điều này khiến mô hình không thể nắm bắt được các phân phối phức tạp hơn.
- + Phụ thuộc vào cấu trúc thời gian cố định: mối quan hệ giữa các quan sát ở các thời điểm khác nhau và số lượng độ trễ cần sử dụng phải được xác định thủ công.
- + Chỉ xử lý dữ liệu đơn biến: trong khi nhiều bài toán thực tế có nhiều biến đầu vào.
- + Chỉ dự báo một bước: nhiều ứng dụng thực tế yêu cầu dự báo nhiều bước với tầm nhìn dài hơn.

Các phương pháp học máy có thể hiệu quả đối với những bài toán dự báo chuỗi thời gian phức tạp, có nhiều biến đầu vào, quan hệ phi tuyến và dữ liệu bị thiếu. Tuy nhiên, để đạt hiệu quả cao, các phương pháp này thường cần những đặc trưng được thiết

kể thủ công bởi chuyên gia lĩnh vực hoặc những người có kiến thức về xử lý tín hiệu [2].

Các mạng nơ-ron đơn giản hơn, như MLP (Multilayer Perceptron), hoạt động bằng cách xấp xỉ một hàm ánh xạ từ các biến đầu vào sang các biến đầu ra. Khả năng tổng quát này rất hữu ích đối với chuỗi thời gian vì nhiều lý do:

- + Khả năng chịu nhiễu tốt: Mạng nơ-ron có độ bền vững trước nhiễu trong dữ liệu đầu vào và trong chính hàm ánh xạ, thậm chí vẫn có thể học và dự báo khi tồn tại giá trị bị thiếu.

- + Phi tuyến: Mạng nơ-ron không đặt ra những giả định mạnh về dạng của hàm ánh xạ và có thể dễ dàng học cả quan hệ tuyến tính lẫn phi tuyến.

Cụ thể hơn, mạng nơ-ron có thể được cấu hình để hỗ trợ một số lượng đầu vào và đầu ra bất kỳ nhưng cố định trong hàm ánh xạ. Điều này có nghĩa là mạng nơ-ron có thể trực tiếp hỗ trợ:

- + Dữ liệu đa biến (Multivariate Inputs): Có thể chỉ định số lượng đặc trưng đầu vào tùy ý, từ đó hỗ trợ trực tiếp cho các mô hình dự báo đa biến.

- + Dự báo nhiều bước (Multi-step Forecasts): Có thể chỉ định số lượng giá trị đầu ra tùy ý, từ đó hỗ trợ trực tiếp cho dự báo nhiều bước và thậm chí dự báo đa biến.

Chỉ với những khả năng này thôi, các mạng nơ-ron truyền thẳng (feedforward neural networks) đã có thể hữu ích cho dự báo chuỗi thời gian. Tuy nhiên, việc sử dụng mạng nơ-ron cũng hàm ý rằng phải tồn tại một hàm ánh xạ có ý nghĩa để học được. Nếu chuỗi thời gian thực chất là một quá trình ngẫu nhiên dạng "random walk", thì mô hình sẽ không thể dự báo tốt hơn mô hình kéo dài giá trị cuối cùng (persistence model).

Kỳ vọng rằng tồn tại một hàm ánh xạ có thể học được cũng làm lộ rõ một số hạn chế của mạng nơ-ron truyền thống:

- + Đầu vào cố định: Số lượng biến trễ (lag) được đưa vào phải cố định, tương tự như các phương pháp dự báo chuỗi thời gian truyền thống.

- + Đầu ra cố định: Số lượng biến đầu ra cũng phải cố định; mặc dù đây là hạn chế tinh tế hơn, nó có nghĩa là với mỗi mẫu đầu vào, mô hình buộc phải tạo ra một đầu ra tương ứng [2].

Mạng Nơ-ron Tích chập (Convolutional Neural Networks – CNN) là một loại mạng nơ-ron được thiết kế để xử lý dữ liệu hình ảnh một cách hiệu quả. Chúng đã chứng minh năng lực mạnh mẽ trong nhiều bài toán thị giác máy tính khó, đạt kết quả vượt trội ở các nhiệm vụ như phân loại ảnh, đồng thời đóng vai trò như một thành phần quan trọng trong các mô hình lai cho các bài toán mới hơn, chẳng hạn như định vị đối tượng, tạo mô tả ảnh và nhiều ứng dụng khác. CNN hoạt động trực tiếp trên dữ liệu gốc - ví dụ như các giá trị điểm ảnh - thay vì sử dụng các đặc trưng được rút trích thủ công theo từng lĩnh vực. Nhờ đó, mô hình học được cách tự động trích xuất những đặc trưng hữu

ích nhất phục vụ bài toán cần giải quyết. Quá trình này được gọi là học biểu diễn (representation learning). Điểm đặc biệt là các đặc trưng được CNN học sẽ không phụ thuộc vào vị trí hay hình dạng xuất hiện trong dữ liệu, thể hiện tính bất biến đối với biến đổi hoặc méo dạng.

Khả năng học và tự động trích xuất đặc trưng từ dữ liệu thô này có thể được áp dụng rất tốt vào các bài toán dự báo chuỗi thời gian. Một chuỗi quan sát có thể được xem như một “ảnh một chiều”, mà CNN có thể đọc và chất lọc ra những thành phần quan trọng nhất. Thuộc tính cốt lõi của CNN là sử dụng các đơn vị xử lý khác nhau trong các lớp tích chập, mỗi đơn vị chịu trách nhiệm nhận dạng những đặc trưng cục bộ trong dữ liệu tín hiệu. Sự đa dạng này giúp mô hình tạo ra các biểu diễn hiệu quả về những điểm nổi bật của tín hiệu. Khi được xếp chồng nhiều lớp, kiến trúc sâu của CNN cho phép mô hình ghi nhận sự nổi bật của tín hiệu ở nhiều cấp độ khác nhau.

Năng lực đó đã được chứng minh rõ ràng trong các nhiệm vụ phân loại chuỗi thời gian, chẳng hạn như tự động nhận diện hoạt động của con người dựa trên dữ liệu cảm biến gia tốc thu từ thiết bị thể thao hoặc điện thoại thông minh. Ưu điểm chính của CNN trong các bài toán nhận diện hoạt động bao gồm: (i) mô hình tự động trích xuất đặc trưng theo nhiệm vụ, không phụ thuộc đặc trưng dự định; (ii) đặc trưng được rút trích có khả năng phân biệt tốt hơn giữa các loại hoạt động; (iii) quá trình trích xuất đặc trưng và phân loại được gộp chung trong một mô hình duy nhất, giúp hiệu quả của hai phần được tăng cường lẫn nhau.

CNN thừa hưởng các lợi thế của mạng Perceptron nhiều lớp (MLP) trong dự báo chuỗi thời gian - như hỗ trợ dữ liệu đầu vào đa biến, đầu ra đa biến và khả năng học các mối quan hệ hàm phức tạp - nhưng không đòi hỏi mô hình phải học trực tiếp từ các quan sát trễ (lag). Thay vào đó, CNN có thể học một biểu diễn từ một chuỗi đầu vào dài và tự chọn ra những phần quan trọng nhất phục vụ cho nhiệm vụ dự báo.

2. Giới thiệu về mô hình trí tuệ nhân tạo được đề xuất

Với những ưu điểm như đã giới thiệu ở trên và đã được khẳng định qua các công trình đã công bố [3, 4] ... Nên báo cáo này đã lựa chọn mô hình CNN(conv1D) làm ví dụ minh họa.

2.1 Mô hình CNN(Conv1D)

Mạng nơ-ron tích chập một chiều (Convolutional Neural Network - Conv1D) là một trong những mô hình quan trọng nhất khi xử lý dữ liệu chuỗi thời gian. Ý tưởng cốt lõi của Conv1D là sử dụng các bộ lọc (kernel) trượt qua trục thời gian để phát hiện những mẫu cục bộ có ý nghĩa, chẳng hạn như xu hướng ngắn hạn, sự tăng giảm đột ngột hay tín hiệu dao động đặc trưng. Khi làm việc với chuỗi dữ liệu, mỗi quan sát trong chuỗi có thể được biểu diễn như một vector có nhiều đặc trưng (trong trường hợp đa biến) hoặc chỉ là một giá trị đơn (trong trường hợp đơn biến). Nếu chuỗi có chiều dài T

và có d đặc trưng, dữ liệu đầu vào có thể được coi như một ma trận $T \times d$. Mục tiêu của Conv1D là từ cấu trúc này, mạng học được các biểu diễn mới nhằm mô tả chính xác nhất đặc tính của chuỗi trong phạm vi cục bộ.

Ở cấp độ toán học, phép tích chập một chiều được mô tả qua công thức:

$$y_i = \sum_{j=0}^{k-1} w_j x_{i+j} + b \quad (1)$$

Trong đó, x_{i+j} là giá trị của chuỗi tại vị trí thứ $i + j$; tập $w_0, w_1, \dots, w_{k-1}$ chính là các trọng số của kernel có kích thước k ; còn b là hệ số điều chỉnh (bias) nhằm tăng tính linh hoạt cho mô hình. Công thức này mô tả cách mà kernel được “trượt” từ trái sang phải, mỗi lần ghi nhận một đoạn nhỏ chiều dài k của chuỗi. Kết quả tại từng vị trí y_i là một phép tổng có trọng số của các phần tử trong đoạn đó. Ý nghĩa của phép cộng tích trọng số này là để cho mô hình nhận diện các đặc trưng: nếu một mẫu trong chuỗi khớp với hình dạng mà kernel đã học, giá trị y_i sẽ cao; ngược lại, khi mẫu không tương đồng, giá trị tích chập sẽ nhỏ. Đây chính là cơ chế nhận dạng đặc trưng của CNN.

Khi dữ liệu có nhiều biến đầu vào, phép tích chập được mở rộng thêm một chiều. Thay vì chỉ làm việc với giá trị đơn, mỗi phần tử x_t sẽ trở thành một vector có C_{in} kênh (features). Kernel lúc này cũng được mở rộng thành tensor ba chiều, và công thức tích chập được viết thành:

$$y_{i,cout} = \sum_{c=1}^{C_{in}} \sum_{j=0}^{k-1} W_{j,c,Cout} x_{i+j,c} + b_{Cout} \quad (2)$$

Trong (2):

C_{in} là số kênh của dữ liệu đầu vào;

C_{out} là số kênh đầu ra mà Conv1D muốn học;

$W_{j,c,Cout}$ là trọng số của kernel tương ứng với bước thời gian j , kênh đầu vào c và kênh đầu ra C_{out} .

Công thức trên cho thấy quá trình tích chập thực chất là phép kết hợp có trọng số giữa các biến đầu vào tại nhiều bước thời gian khác nhau, từ đó tạo ra một biểu diễn mới giàu thông tin hơn. Đây là lý do Conv1D đặc biệt mạnh trong phân tích chuỗi thời gian đa biến: nó có thể tự động học được cách các biến liên quan với nhau trong một cửa sổ thời gian cục bộ.

Sau lớp tích chập, mô hình thường áp dụng một hàm kích hoạt phi tuyến để tăng sức biểu diễn. Một trong những hàm phổ biến nhất là ReLU, được định nghĩa bởi:

$$f(x) = \max(0, x) \quad (3)$$

Hàm này có tác dụng giữ lại các giá trị dương - vốn thường đại diện cho những đặc trưng mạnh - và loại bỏ nhiều hoặc các tín hiệu không hữu ích bằng cách đặt các giá trị âm về 0. Việc áp dụng phi tuyến là bước quan trọng, giúp Conv1D không còn giới

hạn ở việc mô phỏng các quan hệ tuyến tính đơn giản mà có thể học các mẫu phức tạp hơn nhiều.

Bên cạnh lớp tích chập và hàm kích hoạt, một thành phần quan trọng khác là lớp pooling. MaxPooling1D hoạt động bằng cách chọn giá trị cực đại trong một cửa sổ:

$$y_i = \max(x_i, x_{i+1}, \dots, x_{i+k-1}) \quad (4)$$

Cách làm này nén thông tin, giúp mô hình bất biến hơn trước nhiễu nhỏ hoặc những thay đổi không quan trọng. Ngược lại, AveragePooling1D thay thế cực đại bằng trung bình, phù hợp cho dữ liệu có nhiễu mạnh, nơi việc giữ lại “độ lớn” không quan trọng bằng việc nắm bắt xu hướng chung.

Khi các lớp Conv1D được xếp chồng nhiều tầng, mô hình dần học được các đặc trưng ngày càng trừu tượng: lớp đầu tiên nhận diện các biến thiên nhỏ, lớp thứ hai có thể nhận ra mẫu dao động phức tạp hơn, và các lớp tiếp theo mô hình hóa các quan hệ rộng theo thời gian. Sau nhiều tầng tích chập và pooling, dữ liệu được rút gọn thành một biểu diễn đa chiều nhưng giàu thông tin. Giai đoạn cuối cùng là làm phẳng tensor thành vector và đưa qua các lớp Dense để dự báo:

$$\hat{y} = Wx + b \quad (5)$$

Công thức này biểu diễn quá trình tuyến tính cuối cùng, nơi vector đặc trưng được ánh xạ thành đầu ra dự báo. Trọng số W và bias b được học tối ưu theo hàm mất mát, ví dụ như MSE đối với dự báo giá trị tương lai.

Điểm mạnh đặc biệt của Conv1D trong chuỗi thời gian nằm ở cơ chế học sâu biểu diễn. Trong khi các mô hình truyền thống như ARIMA yêu cầu thiết kế thủ công các độ trễ (lag), Conv1D có thể học trực tiếp các mẫu quan trọng từ một chuỗi dài mà không cần quy định trước độ trễ nào quan trọng hơn. Điều này giúp mô hình linh hoạt hơn trong các bài toán dự báo đa biến, dự báo nhiều bước, hoặc khi tín hiệu chứa nhiều cấu trúc cục bộ phức tạp [2].

2.2 Các thành phần khi xây dựng mô hình CNN(conv1D)

Khi xây dựng một mô hình CNN sử dụng lớp tích chập một chiều (Conv1D), điều quan trọng là hiểu rõ các thành phần có trong mô hình và vai trò của chúng trong toàn bộ quá trình học. Mô hình mô tả trong ví dụ được tổ chức theo dạng Sequential, nghĩa là các lớp được xếp nối tiếp nhau từ đầu vào đến đầu ra. Mỗi lớp đảm nhiệm một chức năng riêng biệt, cùng nhau tạo thành một kiến trúc trích xuất đặc trưng và suy luận. Lớp đầu vào xác định cấu trúc dữ liệu mà mô hình sẽ tiếp nhận, thường bao gồm số bước thời gian và số đặc trưng. Việc định nghĩa rõ hình dạng đầu vào giúp Conv1D xử lý chuỗi dữ liệu như một tập tín hiệu có cấu trúc, đảm bảo tính nhất quán khi huấn luyện và dự đoán.

Trọng tâm của kiến trúc CNN trong ví dụ là lớp Conv1D, lớp chịu trách nhiệm thực hiện phép tích chập trên trục thời gian. Lớp Conv1D bao gồm nhiều bộ lọc (filters),

mỗi bộ lọc học một mẫu cục bộ khác nhau trong chuỗi dữ liệu. Số lượng bộ lọc quyết định độ phong phú của các đặc trưng mà mô hình có thể trích xuất. Bên cạnh đó, kích thước kernel (`kernel_size`) là yếu tố quyết định độ rộng của “cửa sổ quan sát” mà bộ lọc dùng để quét chuỗi. Kernel càng lớn thì đặc trưng càng mang tính khái quát; kernel nhỏ giúp mô hình nhạy với tín hiệu ngắn và sắc nét. Lớp `Conv1D` còn có thể đi kèm một hàm kích hoạt (activation) nhằm đưa phi tuyến vào mô hình, cho phép mạng học các quan hệ phức tạp thay vì bị giới hạn trong miền tuyến tính. Một lựa chọn quan trọng khác là padding, tham số quy định mô hình có giữ nguyên chiều dài chuỗi sau tích chập hay không. Padding “same” thường được sử dụng trong các trường hợp cần duy trì cấu trúc chuỗi, giúp mô hình dễ xây dựng và ổn định hơn qua nhiều lớp liên tiếp.

Sau khi dữ liệu được xử lý bởi lớp `Conv1D` và các hàm kích hoạt tương ứng, đầu ra của lớp tích chập có dạng một tensor ba chiều gồm số mẫu, chiều dài chuỗi mới và số lượng kênh đặc trưng. Để chuyển dữ liệu này vào các lớp kết nối đầy đủ (`Dense`), mô hình sử dụng lớp `Flatten`. Lớp `Flatten` có nhiệm vụ “trải phẳng” toàn bộ tensor thành một vector một chiều, giúp dữ liệu có thể được đưa vào các lớp cuối cùng sử dụng cấu trúc ma trận đơn giản hơn. Đây là bước quan trọng trong kiến trúc CNN, vì nó đánh dấu sự chuyển đổi từ không gian đặc trưng có cấu trúc (chuỗi) sang không gian vector của các mô hình hồi quy hoặc phân loại truyền thống. Lớp tiếp theo là lớp `Dense`, lớp này đóng vai trò ánh xạ biểu diễn đặc trưng đã học thành đầu ra mong muốn. `Dense` có thể có một hoặc nhiều nút tùy theo số lượng biến mục tiêu; trong các bài toán hồi quy đa đầu ra, lớp `Dense` thường có số nút bằng đúng số giá trị cần dự báo.

Sau khi kiến trúc của mô hình được xác định, bước tiếp theo là cấu hình quá trình học thông qua việc lựa chọn optimizer và loss function. Bộ tối ưu hóa (optimizer) là thuật toán điều chỉnh trọng số của mạng dựa trên đạo hàm của hàm mất mát; các bộ tối ưu hiện đại như Adam thường được ưa chuộng nhờ khả năng tự điều chỉnh tốc độ học của từng trọng số. Còn hàm mất mát (loss) đóng vai trò đo lường mức độ sai lệch giữa dự đoán của mô hình và giá trị thực tế. Trong bài toán hồi quy, hàm mất mát có thể là MSE, MAE, hoặc các biến thể như Huber Loss nhằm cân bằng giữa độ nhạy và khả năng chống nhiễu. Cuối cùng, mô hình được huấn luyện bằng lệnh `fit()`, trong đó ta chỉ định số vòng lặp (epochs), kích thước lô dữ liệu (`batch_size`), và tập dữ liệu huấn luyện. Quá trình này giúp mô hình dần điều chỉnh trọng số sao cho hàm mất mát được tối thiểu hóa.

Như vậy, một mô hình `Conv1D` hoàn chỉnh được hình thành từ các thành phần chính gồm định nghĩa đầu vào, lớp tích chập với các tham số như filters, `kernel_size`, activation và padding, lớp `Flatten` để chuyển đổi không gian đặc trưng, lớp `Dense` để đưa ra dự đoán, cùng các cấu hình tối ưu hóa như optimizer và loss. Sự phối hợp giữa

các thành phần này tạo nên một mô hình CNN có khả năng học đặc trưng mạnh mẽ, phù hợp cho các bài toán dự báo chuỗi thời gian hoặc hồi quy nhiều biến.

Sau đây là ví dụ về 1 phương án xây dựng mô hình CNN(Conv1D):

```
model = Sequential([
    tf.keras.Input(shape=(1, X_train_scaled.shape[1])),
    Conv1D(256, kernel_size=2, activation='relu', padding='same'),
    Flatten(),
    Dense(2) # predict latitude và longitude
])
model.compile(optimizer='Adam', loss='huber')
history = model.fit(X_train_resaped, y_train_scaled, epochs=200, batch_size=16,
verbose=1)
```

2.3 Các giải pháp chuẩn hóa dữ liệu đầu vào

Chuẩn hóa dữ liệu là một bước quan trọng trong quá trình xây dựng mô hình học máy, đặc biệt khi làm việc với các thuật toán nhạy cảm với độ lớn của đặc trưng như mạng nơ-ron, SVM, KNN hoặc khi huấn luyện các mô hình dự báo chuỗi thời gian. Một trong những phương pháp phổ biến nhất là chuẩn hóa theo z-score, trong đó mỗi đặc trưng được đưa về phân phối có trung bình bằng 0 và độ lệch chuẩn bằng 1. Công thức chuẩn hóa z-score được viết như sau:

$$x' = \frac{x - \mu}{\sigma} \quad (6)$$

với μ là giá trị trung bình và σ là độ lệch chuẩn của đặc trưng. Cách làm này giúp các giá trị ở nhiều thang đo khác nhau có thể được mô hình xử lý đồng đều, giảm ảnh hưởng của các thuộc tính có biên độ lớn lẫn át các đặc trưng nhỏ hơn. Khi dữ liệu có nhiều biến với mức độ phân tán khác nhau, chuẩn hóa z-score giúp mô hình học nhanh hơn và ổn định hơn trong quá trình tối ưu hóa.

Một kỹ thuật khác thường được sử dụng là Min–Max Scaling, trong đó giá trị của mỗi đặc trưng được ánh xạ về một khoảng cố định, phổ biến nhất là $[0, 1]$ hoặc $[-1, 1]$. Phương pháp này được mô tả bằng công thức:

$$x' = \frac{x - x_{\min}}{x_{\max} - x_{\min}} \quad (7)$$

với $x_{\min}$ và $x_{\max}$ lần lượt là giá trị nhỏ nhất và lớn nhất của đặc trưng. Min - Max Scaling bảo toàn cấu trúc tương đối của dữ liệu nhưng thu nhỏ toàn bộ phạm vi, khiến cho thuật toán gradient descent hoạt động hiệu quả hơn. Tuy nhiên, kỹ thuật này nhạy

cảm với ngoại lệ (outliers), do các giá trị cực lớn hoặc cực nhỏ có thể làm thay đổi toàn bộ thang đo của tập dữ liệu.

Trong nhiều trường hợp, dữ liệu có xu hướng lệch (skewed distribution), không tuân theo phân phối chuẩn và chứa nhiều giá trị lớn dồn về một phía. Khi đó, các biến đổi phi tuyến như log-transform $x' = \log(x + 1)$, square-root transform $x' = \sqrt{x}$ hoặc Box-Cox transform được sử dụng để làm giảm mức độ lệch của phân phối. Biến đổi Box-Cox được định nghĩa bởi:

$$x' = \begin{cases} \frac{x^\lambda - 1}{\lambda}, & \lambda \neq 0 \\ \log(x), & \lambda = 0 \end{cases} \quad (8)$$

Những kỹ thuật này đặc biệt hữu ích trong các bài toán có dữ liệu dạng tỷ lệ hoặc chỉ nhận giá trị dương. Việc thay đổi hình dạng phân phối giúp mô hình học được quan hệ giữa các biến một cách tự nhiên hơn, tránh hiện tượng gradient lớn gây bất ổn hoặc mất cân bằng trong quá trình huấn luyện.

Ngoài ra, đối với dữ liệu có sự khác biệt giữa các đơn vị đo hoặc các biến quan sát có quy mô khác nhau theo bản chất, các kỹ thuật chuẩn hóa theo robust scaling trở nên hữu ích hơn. Kỹ thuật này sử dụng các thống kê bền vững như median và interquartile range (IQR) thay vì mean và standard deviation. Công thức chuẩn hóa robust được viết như sau:

$$x' = \frac{x - \text{median}(x)}{\text{IQR}(x)} \quad (9)$$

với $\text{IQR}(x) = Q_3 - Q_1$ (hiệu giữa tứ phân vị thứ ba và thứ nhất). Cách làm này giảm mạnh ảnh hưởng của ngoại lệ và phù hợp với dữ liệu môi trường, dữ liệu cảm biến hoặc hành vi người dùng vốn chứa nhiều nhiễu.

Trong một số bài toán chuỗi thời gian, thay vì chuẩn hóa từng đặc trưng độc lập, có thể áp dụng chuẩn hóa mang tính động như standardization theo từng cửa sổ thời gian, hoặc các kỹ thuật như differencing để loại bỏ xu hướng dài hạn. Differencing được biểu diễn bằng công thức:

$$x'_t = x_t - x_{t-1} \quad (10)$$

Biến đổi này giúp làm cho chuỗi trở nên tĩnh (stationary), phù hợp cho các mô hình cần tính ổn định như ARIMA hoặc một số kiến trúc học sâu. Khi dữ liệu được chuẩn hóa theo từng cửa sổ hoặc bằng các biến đổi tĩnh hóa chuỗi, mô hình CNN hoặc các thuật toán dự báo khác có thể tập trung vào biến động thực sự thay vì các xu thế dài hạn không liên quan.

Nhìn chung, chuẩn hóa dữ liệu không chỉ giúp tăng tốc độ hội tụ của mô hình mà còn cải thiện đáng kể chất lượng dự đoán. Tùy thuộc vào tính chất của tập dữ liệu và mục tiêu phân tích, ta có thể lựa chọn giữa các kỹ thuật như z-score, Min-Max Scaling,

log-transform, robust scaling hay chuẩn hóa theo thời gian, đảm bảo dữ liệu được đưa vào mô hình ở trạng thái cân bằng và thích hợp nhất.

3. Giải pháp xử lý dữ liệu

Để xử lý dữ liệu, tác giả đã sử dụng ngôn ngữ lập trình Python trên trình biên dịch Spyder thuộc khung chương trình Anaconda với các thư viện phục vụ cho lập trình trí tuệ nhân tạo.

3.1 Anaconda

Anaconda được nhiều người biết đến như một nền tảng phân phối Python thuận tiện cho khoa học dữ liệu, nhưng thực chất giá trị của nó vượt xa khuôn khổ của một bộ cài đặt thông thường. Anaconda hoạt động như một hệ sinh thái tích hợp, nơi người dùng có thể xây dựng, điều phối và kiểm soát toàn bộ môi trường phân tích dữ liệu của mình mà không cần quá am hiểu về cấu hình hệ thống. Điểm mạnh đặc biệt là cơ chế tách biệt môi trường, cho phép mỗi dự án vận hành trên phiên bản thư viện riêng, tránh hoàn toàn sự xung đột mà các lập trình viên thường gặp khi dùng Python thuần. Thay vì phải cài đặt thủ công từng gói và lo lắng về sự phụ thuộc phức tạp, người dùng chỉ cần vài lệnh đơn giản để thiết lập môi trường theo mong muốn. Khi quy mô dự án tăng lên, sự linh hoạt này tạo ra cảm giác “ổn định” hiếm thấy, nhất là đối với những công việc yêu cầu thử nghiệm nhiều mô hình học máy hoặc xử lý dữ liệu từ nhiều nguồn khác nhau [5].

Một trong những điểm khiến Anaconda trở nên quen thuộc với cộng đồng nghiên cứu là khả năng đóng vai trò như “cổng trung gian” giữa các công cụ lập trình, tính toán và trình bày khoa học. Thay vì phải mở từng ứng dụng rời rạc, người dùng có thể khởi động Anaconda Navigator để quản lý notebook, trình chỉnh sửa, thư viện và môi trường chỉ trong một giao diện trực quan. Công cụ này đặc biệt hữu ích với những người mới bắt đầu, vốn thường bối rối trước cách cài đặt Jupyter Notebook, Spyder hay các IDE khác. Anaconda biến quá trình đó thành chuỗi thao tác trực quan và không yêu cầu kiến thức hệ thống sâu. Ở các nhóm nghiên cứu hoặc đơn vị phân tích dữ liệu, Navigator còn đóng vai trò “tủ điều khiển” chung, nơi mọi người có thể chia sẻ cách cấu hình môi trường hoặc cung cấp các kịch bản đã tối ưu để đồng nghiệp tái sử dụng, giúp rút ngắn thời gian làm quen và tránh lỗi phát sinh khi thiết lập thủ công [5].

Đi sâu vào cách hoạt động, conda - trình quản lý gói và môi trường của Anaconda - là yếu tố khiến nền tảng này có sức lan tỏa rộng rãi. Khác với pip vốn chỉ tập trung vào Python, conda làm việc ở cấp độ hệ thống, cho phép quản lý cả thư viện C, R hay các gói biên dịch phức tạp thường dùng trong mô phỏng khoa học và xử lý ảnh. Điều này khiến conda trở thành lựa chọn hàng đầu cho các dự án yêu cầu nhiều thành phần phụ thuộc khó cài đặt, ví dụ như TensorFlow kết hợp CUDA, thư viện GIS, hoặc các mô-đun tối ưu hóa. Nhờ conda, người dùng có thể tái tạo môi trường một cách chính xác chỉ bằng tệp cấu hình, điều mà trước đây đòi hỏi nhiều bước thiết lập thủ công và dễ gây sai

lệch giữa máy phát triển và máy triển khai. Khả năng này mang lại lợi thế rõ rệt trong triển khai thực nghiệm học máy, khi việc tái hiện điều kiện cài đặt đóng vai trò quan trọng trong tính lặp lại và độ tin cậy của kết quả nghiên cứu.

Mặc dù được đánh giá cao, Anaconda không phải là công cụ chỉ dành cho người làm dữ liệu chuyên nghiệp; nó còn hữu ích trong giáo dục, doanh nghiệp và cả các dự án cá nhân. Sinh viên có thể dùng Anaconda để tạo môi trường học tập cho môn lập trình, tránh xung đột khi thử nghiệm nhiều phiên bản Python khác nhau. Doanh nghiệp có thể áp dụng Anaconda trong quá trình chuẩn hóa công cụ nội bộ, bảo đảm mỗi nhóm phát triển đều sử dụng đúng phiên bản thư viện theo yêu cầu sản phẩm. Ngay cả những người làm dự án nhỏ, ví dụ xây dựng dashboard, mô phỏng tài chính hay xử lý ảnh cơ bản, cũng được hưởng lợi nhờ khả năng quản lý gói gọn nhẹ và dễ hiểu của conda. Điểm quan trọng là Anaconda tạo ra sự đồng nhất trong cách quản lý môi trường, biến công việc cài đặt và duy trì hệ thống - vốn là trở ngại của nhiều người - thành một phần mượt mà của quy trình làm việc. Chính sự kết hợp giữa tính ổn định, tính linh hoạt và khả năng tiếp cận đã giúp Anaconda trở thành một trong những nền tảng khoa học dữ liệu phổ biến nhất hiện nay, bất kể người dùng là chuyên gia, người học hay người đam mê khám phá công nghệ.

3.2 Spyder

Spyder là một môi trường phát triển tích hợp (IDE) được thiết kế chuyên biệt cho khoa học dữ liệu và tính toán khoa học bằng Python. Khác với những IDE lập trình phổ biến như VS Code hay PyCharm, Spyder chú trọng vào việc cung cấp không gian làm việc trực quan, nơi người dùng có thể viết mã, xem kết quả, theo dõi biến và phân tích dữ liệu trong cùng một cửa sổ. Giao diện của Spyder được tổ chức theo dạng mô-đun: trình soạn thảo mã, bảng điều khiển Python, bảng theo dõi biến và khu vực hiển thị đồ thị. Cách bố trí này khiến người dùng cảm nhận môi trường làm việc gần giống các phần mềm phân tích như MATLAB hoặc RStudio, giúp việc chuyển đổi sang Python diễn ra dễ dàng hơn đối với những người quen với phong cách làm việc kiểu “tính toán - xem kết quả - điều chỉnh”.

Điểm nổi bật của Spyder nằm ở khả năng quản lý và giám sát biến trực quan thông qua công cụ Variable Explorer. Thành phần này cho phép người dùng quan sát toàn bộ biến có trong bộ nhớ tại thời điểm chạy, xem cấu trúc mảng NumPy, nội dung DataFrame của Pandas hoặc trực tiếp thao tác dữ liệu bằng giao diện bảng. Không cần phải dùng lệnh in dữ liệu thủ công, người dùng có thể mở, lọc và kiểm tra từng phần của dữ liệu giống như làm việc với bảng tính. Ngoài ra, Spyder tích hợp mạnh với Matplotlib, SciPy và nhiều thư viện khoa học khác, giúp quy trình thử nghiệm mô hình hoặc kiểm tra giả thuyết diễn ra liền mạch. Việc kết hợp giữa lập trình và trực quan hóa

ngay trong cùng một giao diện giúp giảm đáng kể thời gian thao tác, đặc biệt hữu ích trong những giai đoạn khám phá dữ liệu.

Bên cạnh các tính năng mạnh mẽ có sẵn, Spyder còn dễ dàng mở rộng nhờ hệ thống plugin. Người dùng có thể bổ sung các tính năng như phân tích hiệu suất, kiểm tra chất lượng mã, gợi ý tự động nâng cao hoặc liên kết với các môi trường conda khác nhau. Đây cũng là lý do Spyder phổ biến trong cộng đồng nghiên cứu và giáo dục: nó cung cấp các công cụ cần thiết để tổ chức thí nghiệm lập trình, nhưng không quá nặng nề trong cấu hình. Khi kết hợp cùng Anaconda, Spyder trở thành một giải pháp hoàn chỉnh phục vụ học tập, phân tích dữ liệu và xây dựng mô hình tính toán. Sự cân bằng giữa đơn giản và chức năng khiến Spyder phù hợp cả với người mới học lẫn các nhà nghiên cứu muốn có một môi trường Python ổn định, rõ ràng và dễ theo dõi các bước xử lý.

3.3 Python

Python được xem là ngôn ngữ lập trình cốt lõi trong phần lớn các hoạt động tính toán khoa học và phân tích dữ liệu hiện đại. Điểm mạnh lớn nhất của Python nằm ở triết lý thiết kế “dễ đọc - dễ viết”, giúp người dùng tập trung vào bản chất của bài toán thay vì phải loay hoay với cú pháp. Nhờ vậy, Python trở thành ngôn ngữ phù hợp cho cả người mới bắt đầu lẫn các nhà nghiên cứu chuyên sâu. Bên cạnh đó, Python hỗ trợ phong cách lập trình linh hoạt từ thủ tục, hướng đối tượng cho đến lập trình hàm, tạo điều kiện cho người dùng xây dựng các mô hình phân tích hoặc thuật toán tính toán theo cách tự nhiên nhất. Khi làm việc trong môi trường học thuật hoặc doanh nghiệp, khả năng mở rộng và kết nối của Python với các thư viện hệ thống, cơ sở dữ liệu và công cụ trực quan hóa cũng giúp các nhóm dễ dàng tích hợp nó vào quy trình làm việc chung.

Sức mạnh của Python trong khoa học dữ liệu đến từ hệ sinh thái thư viện rộng lớn. Các gói như NumPy, Pandas, Matplotlib hay SciPy cung cấp nền tảng xử lý dữ liệu, tính toán đại số và phân tích thống kê ở mức độ rất chuyên sâu. Điều này cho phép Python cạnh tranh trực tiếp với các công cụ truyền thống như MATLAB hoặc R, đồng thời vượt trội về tính linh hoạt nhờ khả năng kết hợp với web, tự động hóa và trí tuệ nhân tạo. Đối với những ứng dụng cần mô hình học máy hoặc phân tích dự báo, các thư viện như Scikit-learn, TensorFlow hay PyTorch cung cấp các thuật toán và mô hình hiện đại, giúp người dùng triển khai từ bài toán đơn giản cho đến các mô hình phức tạp dựa trên mạng nơ-ron. Chính sự phong phú của hệ sinh thái này đã khiến Python trở thành lựa chọn mặc định trong các khóa học và dự án liên quan đến dữ liệu.

Bên cạnh vai trò trong phân tích và mô hình hóa, Python còn đóng góp quan trọng vào việc chuẩn hóa quy trình làm việc trong nghiên cứu và triển khai sản phẩm. Các công cụ như Jupyter Notebook, JupyterLab hoặc các IDE chuyên dụng giúp người dùng lưu trữ toàn bộ mã nguồn, kết quả phân tích và mô tả phương pháp trong cùng một tài

liệu. Điều này không chỉ thuận lợi cho việc chia sẻ, tái lập kết quả mà còn hỗ trợ xây dựng báo cáo khoa học linh hoạt và dễ truy cập. Python cũng được ưa chuộng trong các nhóm kỹ thuật vì khả năng tự động hóa, ví dụ xử lý dữ liệu hàng loạt, điều phối pipeline hoặc kết hợp các dịch vụ web. Chính nhờ sự kết hợp giữa đơn giản, mạnh mẽ và khả năng thích nghi với nhiều lĩnh vực mà Python giữ vị trí trung tâm trong các nền tảng như Anaconda, đóng vai trò là “ngôn ngữ chung” cho cộng đồng khoa học dữ liệu và lập trình phân tích hiện đại.

4. Dữ liệu sử dụng

Dữ liệu sử dụng trong báo cáo này là dữ liệu phao trôi được cung cấp bởi các tổ chức nghiên cứu biển như NASA [6].

Phao trôi là một thiết bị quan trắc hải dương học được thiết kế để ghi nhận các biến số môi trường từ lớp nước mặt của đại dương theo phương thức di chuyển tự do cùng dòng chảy. Khác với các hệ thống phao được neo cố định, phao trôi không gắn với đáy biển mà dịch chuyển theo dòng chảy bề mặt, cho phép ghi nhận chuỗi dữ liệu mang tính chất Lagrangian phản ánh trực tiếp động lực của đại dương. Cấu trúc cơ bản của phao trôi bao gồm một phao nổi chứa thiết bị thu thập và truyền dữ liệu, một bộ neo nước (drogue hoặc sea-anchor) được thả ở độ sâu tiêu chuẩn nhằm giảm thiểu ảnh hưởng của gió và sóng, cùng hệ thống định vị vệ tinh để xác định vị trí tức thời. Nhờ đặc tính này, phao trôi trở thành một phần quan trọng của hệ thống quan trắc đại dương toàn cầu, cung cấp dữ liệu liên tục và không gian mở rộng trên nhiều vùng biển.

Dữ liệu thu được từ phao trôi chủ yếu bao gồm thông tin vị trí (vĩ độ, kinh độ) theo thời gian, được ghi nhận định kỳ nhằm mô tả chuyển động của phao và từ đó suy ra vận tốc dòng chảy mặt biển. Bên cạnh thông tin không gian - thời gian, các phao hiện đại còn tích hợp cảm biến đo nhiệt độ mặt biển (sea surface temperature - SST), áp suất khí quyển, độ dẫn điện của nước biển (cho phép suy ra độ mặn), cũng như một số tham số bổ sung tùy thuộc cấu hình thiết bị. Việc thu thập đồng thời nhiều biến số này tạo ra một bộ dữ liệu hải khí tổng hợp, phù hợp cho các nghiên cứu về tương tác đại dương - khí quyển, năng lượng bề mặt, biến đổi nhiệt, và các quá trình lan truyền vật chất trong đại dương. Dữ liệu vị trí được truyền qua vệ tinh trong thời gian thực hoặc gần thời gian thực, cho phép theo dõi liên tục quỹ đạo của phao và các biến thiên môi trường liên quan.

Nhiều chương trình quan trắc quốc tế vận hành, quản lý và cung cấp dữ liệu phao trôi theo chuẩn mở. Một trong những hệ thống quan trọng nhất là Global Drifter Program (GDP), trực thuộc NOAA, sử dụng các phao thuộc Surface Velocity Program (SVP) để thu thập chuỗi quan trắc lâu dài về nhiệt độ mặt biển, áp suất khí quyển và dòng chảy bề mặt. Dữ liệu từ chương trình này được truyền qua vệ tinh và lưu trữ trong các trung tâm dữ liệu đại dương toàn cầu, trong đó có các kho lưu trữ được NASA Earthdata

Catalog chỉ mục và phân phối. Các trang dữ liệu như Earthdata CMR cung cấp mô tả chi tiết về cấu hình thiết bị, phương thức thu thập, định dạng tệp và các mức xử lý chất lượng dữ liệu.

Với tính chất trực tiếp, rộng không gian và có chuỗi thời gian dài, dữ liệu phao trôi đóng vai trò cốt lõi trong nhiều lĩnh vực nghiên cứu khoa học biển và khí hậu. Trong hải dương học vật lý, dữ liệu này được dùng để xác định trường dòng chảy mặt biển, khảo sát các xoáy đại dương, nghiên cứu quá trình lan truyền chất và mô phỏng động lực biển. Trong khí hậu học, dữ liệu nhiệt độ mặt biển và áp suất từ phao trôi hỗ trợ các mô hình dự báo khí hậu theo mùa, phân tích ENSO, và kiểm chứng dữ liệu viễn thám. Ngoài ra, dữ liệu phao trôi còn được sử dụng để hiệu chỉnh và đánh giá độ chính xác của các sản phẩm đo đạc bằng vệ tinh, đặc biệt đối với SST và dòng chảy bề mặt. Nhờ tính nhất quán và độ tin cậy cao, phao trôi hiện là một trong những nguồn dữ liệu in-situ quan trọng nhất trong hệ thống quan trắc đại dương toàn cầu.

5. Kết quả dự đoán

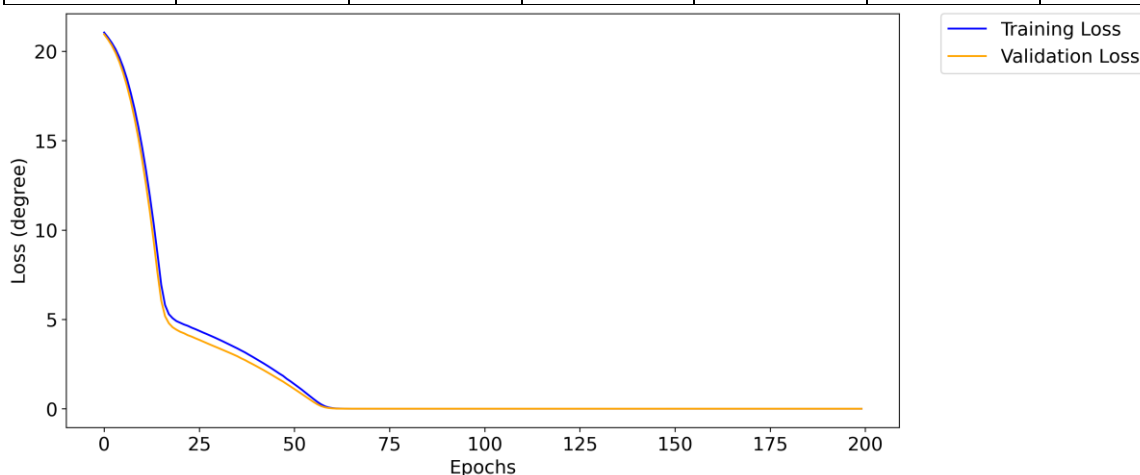
Dữ liệu dùng để dự đoán là dữ liệu thu nhận bởi phao trôi có số hiệu 56644 với thời gian thu dữ liệu từ ngày 20/11/2005 đến ngày 30/11/2005, giãn cách thu tín hiệu là 1 giờ.

Báo cáo sử dụng 03 phương pháp chuẩn hóa dữ liệu bao gồm Min-Max Scaling, Z-Score và Robust Scaling.

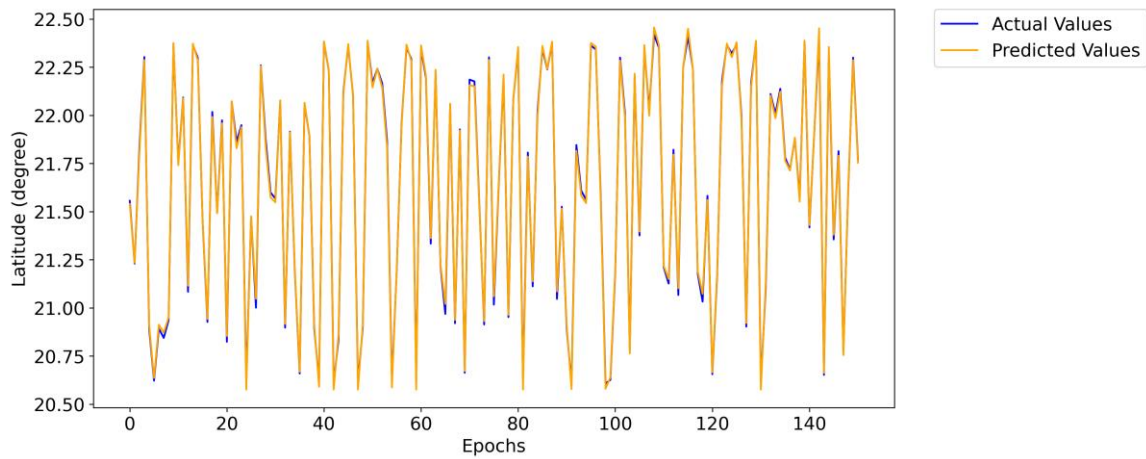
5.1 Kết quả chuẩn hóa với Min-Max Scaling

Bảng 1. Thống kê các đặc trưng độ chính xác của mô hình

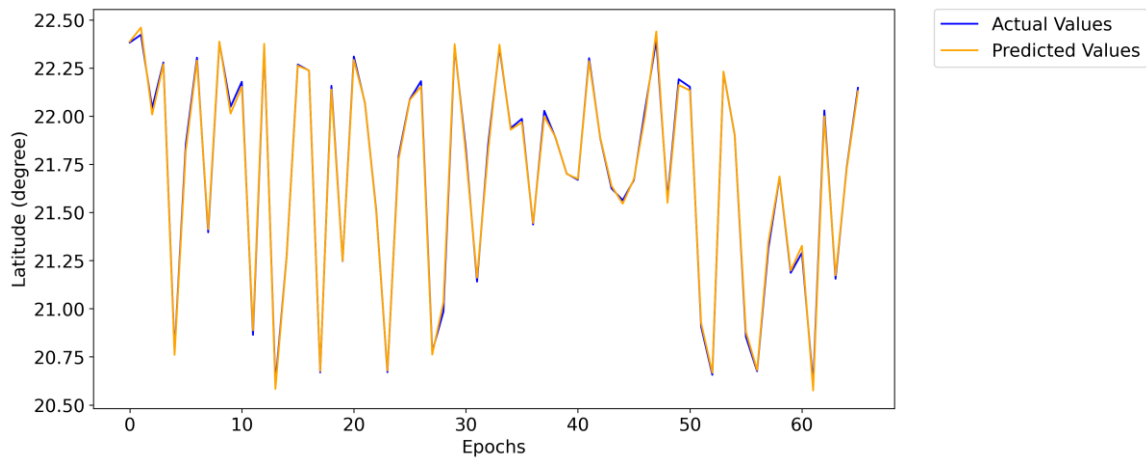
	MSE	RMSE	MAE	R^2	F1-Score	Kappa
	0.000424	0.0206	0.01677	0.9987	1	1



Hình 1. Giá trị của hàm mất mát trong trường hợp chuẩn hóa Min-Max



Hình 2. Giá trị dự đoán với tập dữ liệu huấn luyện sử dụng chuẩn hóa Min-Max

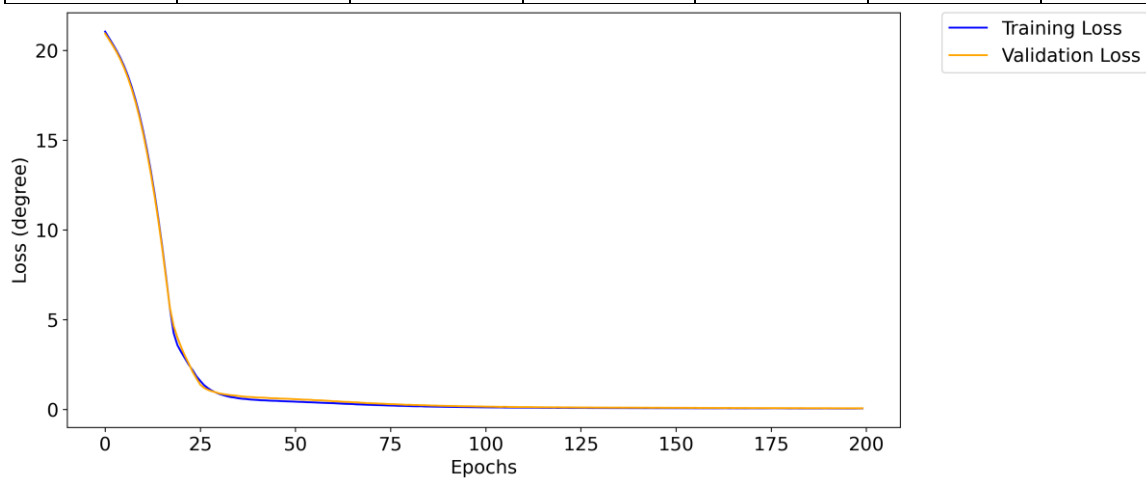


Hình 3. Giá trị dự đoán với tập dữ liệu kiểm tra sử dụng chuẩn hóa Min-Max

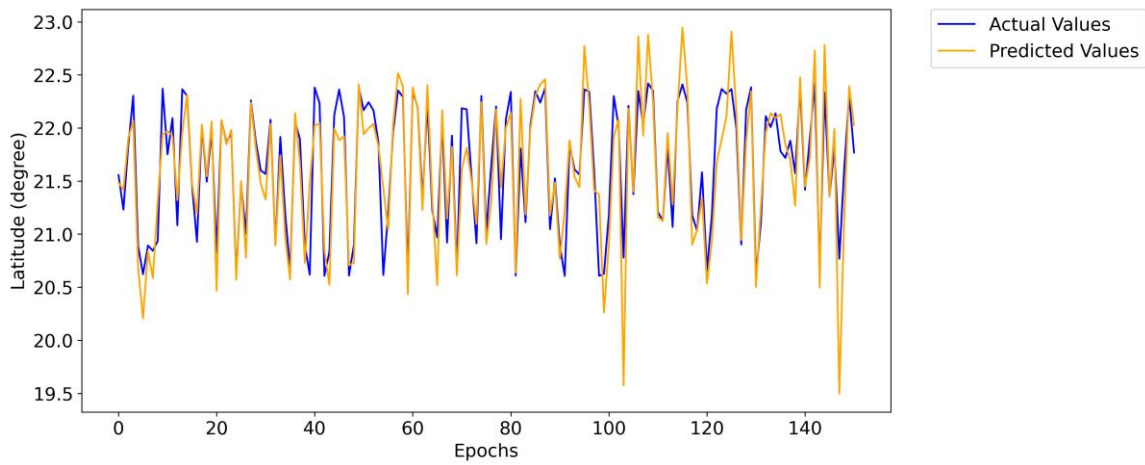
5.2 Kết quả chuẩn hóa với Z-Score

Bảng 2. Thống kê các đặc trưng độ chính xác của mô hình sử dụng chuẩn hóa Z-Score

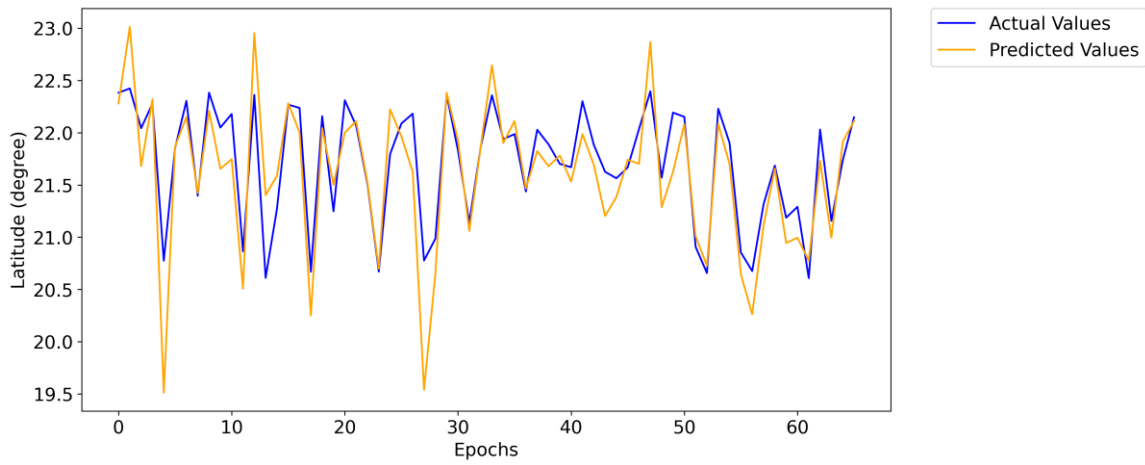
	MSE	RMSE	MAE	R^2	F1-Score	Kappa
	0.124606	0.35300	0.25281	0.60447	0.8182	0.6364



Hình 4. Giá trị hàm mất mát trong trường hợp sử dụng chuẩn hóa Z-Score



Hình 5. Giá trị dự báo trên tập dữ liệu huấn luyện sử dụng chuẩn hóa Z-Score

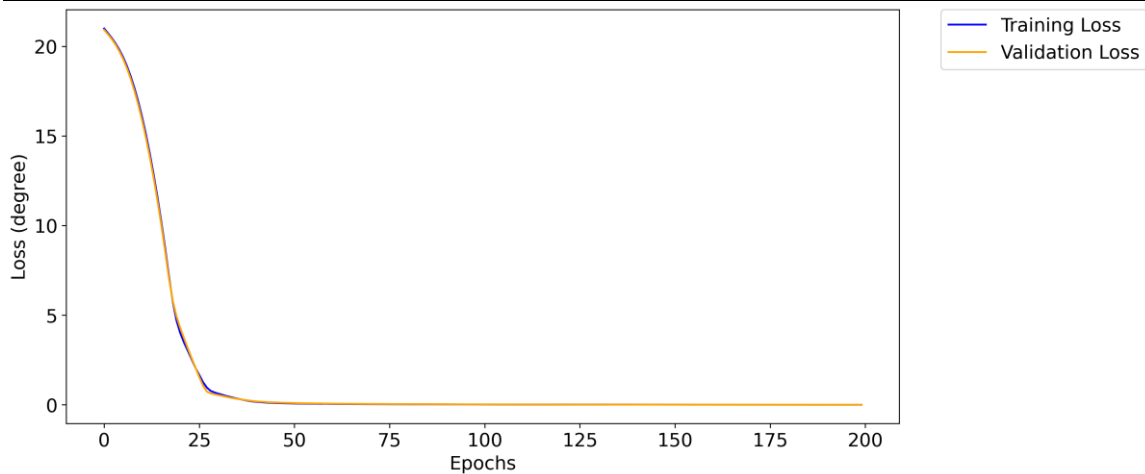


Hình 6. Giá trị dự báo trên tập dữ liệu kiểm tra sử dụng chuẩn hóa Z-Score

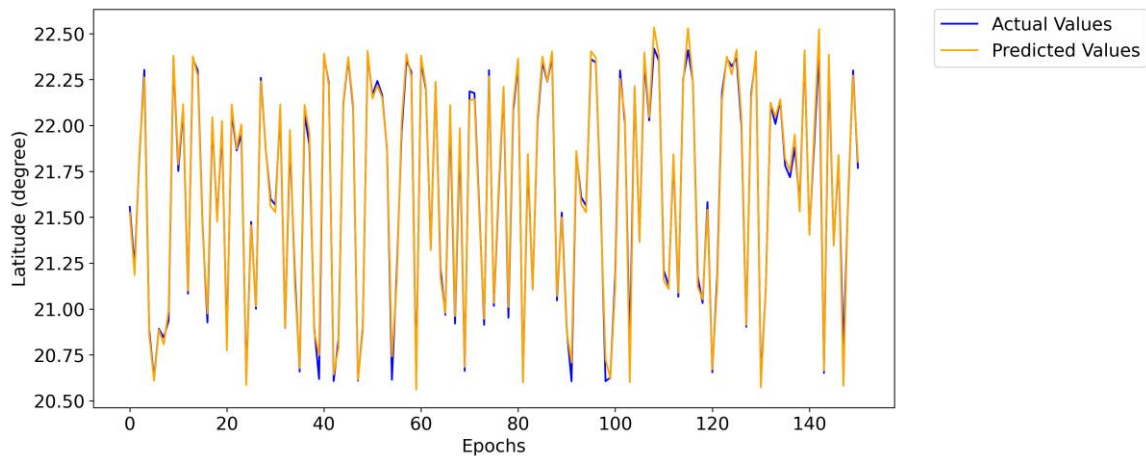
5.3 Kết quả chuẩn hóa với Robust Scaling

Bảng 3. Thống kê các đặc trưng độ chính xác của mô hình sử dụng chuẩn hóa Robust Scaling

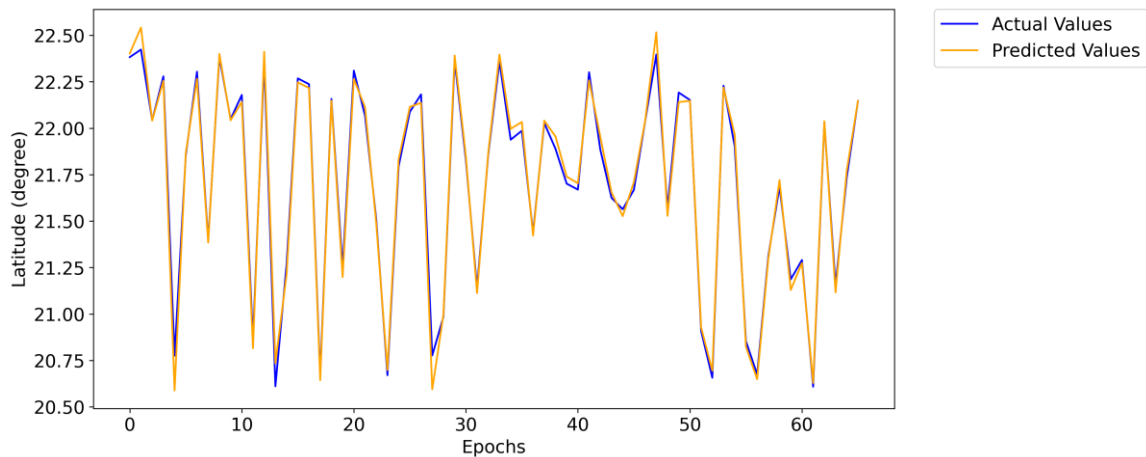
	MSE	RMSE	MAE	R^2	F1-Score	Kappa
	0.002816	0.05306	0.03910	0.9911	1	1



Hình 7. Giá trị hàm mất mát sử dụng chuẩn hóa Robust Scaling



Hình 8. Giá trị dự báo trên tập dữ liệu huấn luyện sử dụng chuẩn hóa Robust Scaling



Hình 9. Giá trị dự báo trên tập dữ liệu kiểm tra sử dụng chuẩn hóa Robust Scaling

Tài liệu tham khảo

1. Lim, B. and S. Zohren, *Time-series forecasting with deep learning: a survey*. Philosophical Transactions of the Royal Society A, 2021. **379**(2194): p. 20200209.
2. Brownlee, J., *Deep learning for time series forecasting: predict the future with MLPs, CNNs and LSTMs in Python*. 2018: Machine Learning Mastery.
3. Trong, N.G., et al., *Spatial prediction of fluvial flood in high-frequency tropical cyclone area using TensorFlow 1D-convolution neural networks and geospatial data*. Remote Sensing, 2023. **15**(22): p. 5429.
4. Trong, N., et al., *Comparative Analysis of Prediction Accuracy for Drifting Buoy Data Using CNN (Conv1D) and GRU Deep Learning Models with Varying Data Volumes*. International Journal of Geoinformatics, 2025. **21**(4): p. 115-130.
5. Vaughan, L., *Python Tools for Scientists: An Introduction to Using Anaconda, JupyterLab, and Python's Scientific Libraries*. 2023: No Starch Press.
6. NASA. *EarthData CMR Search*. Available from: <https://cmr.earthdata.nasa.gov/search/concepts/C1386250149-NSIDCV0.html>.