

**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP.HCM**

**KHOA CÔNG NGHỆ THÔNG TIN**

**🙡🙞🕮🙜🙣**



**KHÓA LUẬN TỐT NGHIỆP**

**DỰ BÁO DỮ LIỆU CHUỖI THỜI GIAN BẰNG LAI GHÉP 2 MÔ HÌNH HỌC SÂU**

**GVHD:** TS. Nguyễn Thành Sơn

**GVPB:**

**Môn học:** Khóa Luận Tốt Nghiệp

**Nhóm thực hiện:** Nhóm 06

**Thành viên:**

Lê Tuấn Nghĩa 20133072

Vũ Xuân Huy 20133048

***TP. Hồ Chí Minh, Tháng 06 năm 2024***

**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP.HCM**

**KHOA CÔNG NGHỆ THÔNG TIN**

**🙡🙞🕮🙜🙣**



**KHÓA LUẬN TỐT NGHIỆP**

**DỰ BÁO DỮ LIỆU CHUỖI THỜI GIAN BẰNG LAI GHÉP 2 MÔ HÌNH HỌC SÂU**

**GVHD:** TS. Nguyễn Thành Sơn

**GVPB:**

**Môn học:** Khóa Luận Tốt Nghiệp

**Nhóm thực hiện:** Nhóm 06

**Thành viên:**

Lê Tuấn Nghĩa 20133072

Vũ Xuân Huy 20133048

***TP. Hồ Chí Minh, Tháng 06 năm 2024***

|  |  |
| --- | --- |
| **ĐH SƯ PHẠM KỸ THUẬT TP.HCM** | **CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM** |
| **KHOA CNTT** | **Độc lập – Tự do – Hạnh phúc** |
| **\*\*\*\*\*\*\*** | **\*\*\*\*\*\*\*\*\*** |

**PHIẾU NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN**

|  |  |
| --- | --- |
| Họ và tên Sinh viên 1: Lê Tuấn Nghĩa | MSSV: 20133072 |
| Họ và tên Sinh viên 2: Vũ Xuân Huy | MSSV: 20133048 |

Ngành: Kỹ thuật dữ liệu.

Tên đề tài: DỰ BÁO DỮ LIỆU CHUỖI THỜI GIAN BẰNG LAI GHÉP 2 MÔ HÌNH HỌC SÂU.

Họ và tên Giáo viên hướng dẫn: TS.Nguyễn Thành Sơn.

**NHẬN XÉT:**

1. Về nội dung thực hiện: ………………………………………………………………………………………… ………………………………………………………………………………………… …………

2. Ưu điểm: ………………………………………………………………………………………… ………………………………………………………………………………………… ………………………………………………………………………………………… ………………

3. Khuyết điểm: ………………………………………………………………………………………… ………………………………………………………………………………………… ………………………………………………………………………………………… ………………

4. Đề nghị cho bảo vệ hay không?

5. Đánh giá loại:

|  |
| --- |
| *Tp. Hồ Chí Minh, ngày … tháng … năm 2023* |
| Giáo viên hướng dẫn  (Ký & ghi rõ họ tên) |

6. Điểm:

|  |  |
| --- | --- |
| **ĐH SƯ PHẠM KỸ THUẬT TP.HCM** | **CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM** |
| **KHOA CNTT** | **Độc lập – Tự do – Hạnh phúc** |
| **\*\*\*\*\*\*\*** | **\*\*\*\*\*\*\*\*\*** |

**PHIẾU NHẬN XÉT CỦA GIÁO VIÊN PHẢN BIỆN**

|  |  |
| --- | --- |
| Họ và tên Sinh viên 1: Lê Tuấn Nghĩa | MSSV: 20133072 |
| Họ và tên Sinh viên 2: Vũ Xuân Huy | MSSV: 20133048 |

Ngành: Kỹ thuật dữ liệu

Tên đề tài: DỰ BÁO DỮ LIỆU CHUỖI THỜI GIAN BẰNG LAI GHÉP 2 MÔ HÌNH HỌC SÂU.

Họ và tên Giáo viên hướng dẫn:

**NHẬN XÉT:**

1. Về nội dung thực hiện: ………………………………………………………………………………………… ………………………………………………………………………………………… …………

2. Ưu điểm: ………………………………………………………………………………………… ………………………………………………………………………………………… ………………………………………………………………………………………… ………………

3. Khuyết điểm: ………………………………………………………………………………………… ………………………………………………………………………………………… ………………………………………………………………………………………… ………………

4. Đề nghị cho bảo vệ hay không?

5. Đánh giá loại:

|  |
| --- |
| *Tp. Hồ Chí Minh, ngày … tháng … năm 2023* |
| Giáo viên hướng dẫn  (Ký & ghi rõ họ tên) |

6. Điểm:

**LỜI CẢM ƠN**

Nhóm chúng em xin chân thành biết ơn quý Thầy Cô đã luôn giúp đỡ, theo dõi và hướng dẫn nhóm trong suốt quá trình thực hiện đề tài.

Đầu tiên, chúng em xin gửi lời cảm ơn đến Ban giám hiệu trường Đại học Sư phạm Kỹ thuật Thành phố Hồ Chí Minh. Nhờ nhà trường mà nhóm chúng em có được một môi trường học tập chất lượng và hiệu quả tốt nhất trong thời gian thực hiện đề tài nghiên cứu cũng như là xuyên suốt quá trình học tập tại trường.

Ngoài ra, chúng em xin gửi lời cảm ơn đến Ban chủ nhiệm khoa Công nghệ Thông tin và các quý Thầy Cô trong khoa đã luôn tận tâm dạy bảo chúng em trong suốt những năm học đại học vừa qua. Thầy Cô đã dạy cho chúng em tất cả những kiến thức cũng như là nền tảng để nhóm chúng em có thể thực hiện đề tài một cách tốt nhất sau quá trình học tập và làm việc tại trường.

Đặc biệt, chúng em xin gửi lời cảm ơn sâu sắc đến thầy **Nguyễn Thành Sơn** đã dạy dỗ, truyền đạt những kiến thức quý báu cho chúng em trong suốt thời gian học tập vừa qua. Trong thời gian thực hiện đề tài nghiên cứu, nhóm em đã luôn được thầy quan tâm, hướng dẫn và góp ý tận tâm. Mặc dù, thời gian thầy hướng dẫn không thể nhiều hơn được nữa, nhưng những gì thầy chỉ dạy chắc chắn sẽ là những kiến thức quý báu, là hành trang để chúng em có thể vững bước sau này.

Mặc dù chúng em đã cố gắng hết sức thực hiện đề tài nhưng chắc chắn đề tài này khó có thể tránh khỏi những thiếu sót và nhiều chỗ còn chưa chính xác, kính mong quý Thầy Cô xem xét và góp ý để bài đồ án của chúng em được hoàn thiện hơn.

Nhóm chúng em xin chân thành cảm ơn!

**MỤC LỤC**

[PHẦN 1: MỞ ĐẦU 1](#_Toc170129037)

[**1.** **Tính cấp thiết của đề tài** 1](#_Toc170129038)

[**2.** **Lý do chọn đề tài** 1](#_Toc170129039)

[**3.** **Mục tiêu đề tài** 2](#_Toc170129040)

[**4.** **Nhiệm vụ thực hiện đề tài** 2](#_Toc170129041)

[**5.** **Giới hạn đề tài** 3](#_Toc170129042)

[**6.** **Bố cục** 3](#_Toc170129043)

[PHẦN 2: NỘI DUNG 5](#_Toc170129044)

[**CHƯƠNG 1: KIẾN THỨC CƠ BẢN VÀ CÁC CÔNG TRÌNH LIÊN QUAN** 5](#_Toc170129045)

[**1.1.** **CÁC KHÁI NIỆM CƠ BẢN** 5](#_Toc170129046)

[**1.1.1.** **Học máy** 5](#_Toc170129047)

[**1.1.2.** **Học sâu** 6](#_Toc170129048)

[**1.1.3.** **Dữ liệu chuỗi thời gian** 7](#_Toc170129049)

[**1.1.4.** **Dự báo dữ liệu chuỗi thời gian** 7](#_Toc170129050)

[**1.2.** **ĐỊNH NGHĨA BÀI TOÁN** 7](#_Toc170129051)

[**1.3.** **CÁC MÔ HÌNH VÀ ƯU, NHƯỢC ĐIỂM** 8](#_Toc170129052)

[**1.3.1.** **CNN - Convolutional Neural Network** 8](#_Toc170129053)

[**1.3.1.1.** **Khái niệm** 8](#_Toc170129054)

[**1.3.1.2.** **Ưu điểm và nhược điểm** 8](#_Toc170129055)

[**1.3.2.** **LSTM – Long Short Term Memory** 9](#_Toc170129056)

[**1.3.2.1.** **Khái niệm** 9](#_Toc170129057)

[**1.3.2.2.** **Ưu điểm và nhược điểm** 9](#_Toc170129058)

[**1.3.3.** **RNN – Recurrent Neural Networks** 10](#_Toc170129059)

[**1.3.3.1.** **Khái niệm** 10](#_Toc170129060)

[**1.3.3.2.** **Ưu điểm và nhược điểm** 10](#_Toc170129061)

[**1.3.4.** **GRU - Gated Recurrent Unit** 10](#_Toc170129062)

[**1.3.4.1.** **Khái niệm** 10](#_Toc170129063)

[**1.3.4.2.** **Ưu điểm và nhược điểm** 11](#_Toc170129064)

[**1.4.** **CÁC CÔNG TRÌNH LIÊN QUAN** 11](#_Toc170129065)

[**CHƯƠNG 2: CÁC MÔ HÌNH SỬ DỤNG** 17](#_Toc170129066)

[**2.1.** **Mô hình CNN** 17](#_Toc170129067)

[**2.2.** **Mô hình RNN** 18](#_Toc170129068)

[2.2.1. Cấu trúc 18](#_Toc170129069)

[2.2.2. Cơ chế hoạt động 19](#_Toc170129070)

[**2.3.** **Mô hình LSTM** 21](#_Toc170129071)

[2.3.1. Cấu trúc 21](#_Toc170129072)

[2.3.1. Cơ chế hoạt động 22](#_Toc170129073)

[**2.4.** **Mô hình GRU** 27](#_Toc170129074)

[2.4.1. Cấu trúc 27](#_Toc170129075)

[2.4.2. Cơ chế hoạt động 28](#_Toc170129076)

[**2.5.** **Mô hình tuần tự** 29](#_Toc170129077)

[2.5.1. Cấu trúc 29](#_Toc170129078)

[2.5.2. Cơ chế hoạt động 29](#_Toc170129079)

[**2.6.** **Mô hình tuần tự cộng** 30](#_Toc170129080)

[2.6.1. Cấu trúc 30](#_Toc170129081)

[2.6.2. Cơ chế hoạt động 30](#_Toc170129082)

[**2.7.** **Mô hình tuần tự nhân** 31](#_Toc170129083)

[**2.8.** **Mô hình song song** 31](#_Toc170129084)

[2.8.1. Cấu trúc 31](#_Toc170129085)

[2.8.2. Cơ chế hoạt động 32](#_Toc170129086)

[**CHƯƠNG 3: CÀI ĐẶT MÔ HÌNH** 33](#_Toc170129087)

[**3.1.** **CHUẨN BỊ DỮ LIỆU VÀ TRỰC QUAN HÓA DỮ LIỆU** 33](#_Toc170129088)

[**3.1.1.** **Chuẩn hóa dữ liệu** 33](#_Toc170129089)

[**3.1.2.** **Chia tập dữ liệu x-y** 34](#_Toc170129090)

[**3.1.3.** **Chia tập dữ liệu train/validation/test** 35](#_Toc170129091)

[**3.1.4.** **Chia giá trị Index (ngày) cho tập train/validation/test** 35](#_Toc170129092)

[**3.1.5.** **Làm sạch dữ liệu và hoàn thành việc chuẩn bị dữ liệu** 36](#_Toc170129093)

[**3.1.6.** **Trực quan hóa dữ liệu** 38](#_Toc170129094)

[**3.2.** **XÂY DỰNG MÔ HÌNH VÀ CÁC HÀM LIÊN QUAN** 39](#_Toc170129095)

[**3.2.1.** **Xây dựng mô hình đơn** 39](#_Toc170129096)

[3.2.1.1. Mô hình CNN 39](#_Toc170129097)

[3.2.1.2. Mô hình LSTM 40](#_Toc170129098)

[3.2.1.3. Mô hình RNN 41](#_Toc170129099)

[3.2.1.4. Mô hình GRU 42](#_Toc170129100)

[**3.2.2.** **Xây dựng mô hình tuần tự** 43](#_Toc170129101)

[3.2.2.1. Mô hình CNN – LSTM 43](#_Toc170129102)

[3.2.2.2. Mô hình LSTM – CNN 44](#_Toc170129103)

[3.2.2.3. Mô hình CNN – RNN 46](#_Toc170129104)

[3.2.2.4. Mô hình RNN – CNN 47](#_Toc170129105)

[3.2.2.5. Mô hình CNN – GRU 48](#_Toc170129106)

[3.2.2.6. Mô hình GRU – CNN 49](#_Toc170129107)

[3.2.2.7. Mô hình LSTM – RNN 49](#_Toc170129108)

[3.2.2.8. Mô hình RNN – LSTM 50](#_Toc170129109)

[3.2.2.9. Mô hình LSTM – GRU 51](#_Toc170129110)

[3.2.2.10. Mô hình GRU – LSTM 52](#_Toc170129111)

[3.2.2.11. Mô hình RNN – GRU 53](#_Toc170129112)

[3.2.2.12. Mô hình GRU – RNN 54](#_Toc170129113)

[3.2.2.13. Huấn luyện mô hình 54](#_Toc170129114)

[**3.2.3.** **Xây dựng mô hình tuần tự nhân** 55](#_Toc170129115)

[3.2.3.1. Lựa chọn mô hình 55](#_Toc170129116)

[3.2.3.2. Huấn luyện mô hình 56](#_Toc170129117)

[**3.2.4.** **Xây dựng mô hình tuần tự cộng** 57](#_Toc170129118)

[3.2.4.1. Lựa chọn mô hình 57](#_Toc170129119)

[3.2.4.2. Huấn luyện mô hình 57](#_Toc170129120)

[**3.2.5.** **Xây dựng mô hình song song** 57](#_Toc170129121)

[3.2.5.1. Lựa chọn mô hình 57](#_Toc170129122)

[3.2.5.2. Xây dựng hàm tính toán 58](#_Toc170129123)

[**3.2.6.** **Xây dựng các hàm tối ưu hóa:** 58](#_Toc170129124)

[**CHƯƠNG 4: THỰC NGHIỆM VÀ ĐÁNH GIÁ** 60](#_Toc170129125)

[**4.1.** **THỰC NGHIỆM** 60](#_Toc170129126)

[**4.1.1.** **Môi trường thực nghiệm và các thư viện liên quan** 60](#_Toc170129127)

[4.1.1.1. Môi trường thực nghiệm 60](#_Toc170129128)

[4.1.1.2. Các thư viện liên quan 60](#_Toc170129129)

[**4.1.2.** **Dữ liệu thực nghiệm** 61](#_Toc170129130)

[4.1.2.1. Dữ liệu thời tiết 61](#_Toc170129131)

[4.1.2.2. Dữ liệu giá vàng 62](#_Toc170129132)

[4.1.2.3. Dữ liệu dầu thô 63](#_Toc170129133)

[4.1.2.4. Dữ liệu giá bia 64](#_Toc170129134)

[4.1.2.5. Dữ liệu Train/ Validation/Test được chia như thế nào? 64](#_Toc170129135)

[**4.2** **CÁC TIÊU CHÍ ĐÁNH GIÁ** 65](#_Toc170129136)

[**4.2.1.** **Đánh giá dựa trên độ chính xác (lỗi) mô hình** 65](#_Toc170129137)

[4.2.1.1. MSE 65](#_Toc170129138)

[4.2.1.2. RMSE 65](#_Toc170129139)

[4.2.1.3. MAE 65](#_Toc170129140)

[4.2.1.4. CV\_RMSE 66](#_Toc170129141)

[**4.2.2.** **Đánh giá theo thời gian thực thi** 66](#_Toc170129142)

[**4.3.** **CÁC TRƯỜNG HỢP THỰC NGHIỆM** 66](#_Toc170129143)

[4.3.1 So sánh quá trình tối ưu hóa siêu tham số của RandomSearchCV, GridSearhCV và BayesSearchCV 67](#_Toc170129144)

[4.3.2 Đánh giá kết quả giữa các mô hình đơn 67](#_Toc170129145)

[4.3.3 Đánh giá kết quả giữa các mô hình tuần tự 67](#_Toc170129146)

[4.3.4 Đánh giá kết quả giữa các mô hình tuần tự nhân 68](#_Toc170129147)

[4.3.5 Đánh giá kết quả giữa các mô hình tuần tự cộng 69](#_Toc170129148)

[4.3.6 Đánh giá kết quả giữa các mô hình song song 70](#_Toc170129149)

[4.3.7 Đánh giá kết quả giữa các mô hình đơn và các mô hình lai ghép liên quan 70](#_Toc170129150)

[**4.4.** **KẾT QUẢ THỰC NGHIỆM** 73](#_Toc170129151)

[4.4.1. Kết quả thực nghiệm của quá trình tối ưu hóa: 73](#_Toc170129152)

[4.4.2. Kết quả thực nghiệm và đánh giá các mô hình đơn 75](#_Toc170129153)

[4.4.3. Kết quả thực nghiệm và đánh giá các mô hình tuần tự 75](#_Toc170129154)

[4.4.4. Kết quả thực nghiệm và đánh giá các mô hình tuần tự nhân 76](#_Toc170129155)

[4.4.5. Kết quả thực nghiệm và đánh giá các mô hình tuần tự cộng 77](#_Toc170129156)

[4.4.6. Kết quả thực nghiệm và đánh giá các mô hình song song 78](#_Toc170129157)

[4.4.7. Kết quả thực nghiệm và đánh giá giữa mô hình đơn và các mô hình lai ghép liên quan 79](#_Toc170129158)

[4.4.8. Đánh giá chung và đề xuất mô hình 82](#_Toc170129159)

[PHẦN 3: KẾT LUẬN 84](#_Toc170129160)

[**1.** **Kết quả đạt được** 84](#_Toc170129161)

[**2.** **Hạn chế khi thực hiện đề tài** 84](#_Toc170129162)

[**3.** **Hướng phát triển đề tài** 84](#_Toc170129163)

[Tài liệu tham khảo 85](#_Toc170129164)

**DANH MỤC HÌNH ẢNH**

[Hình 1. 1 Sự hình thành và phát triển của Học máy, Học sâu [11] 4](#_Toc168436361)

[Hình 2. 1 Cấu trúc của một Mạng nơ-ron tích chập [12] 16](#_Toc168436376)

[Hình 2. 2 Tổng quan về Kiến trúc RNN [9] 17](#_Toc168436377)

[Hình 2. 3 Cơ chế hoạt động của RNN [8] 18](#_Toc168436378)

[Hình 2. 4 Cấu trúc của mạng LSTM [9] 20](#_Toc168436379)

[Hình 2. 5 Bước 1 - Quyết định loại bỏ hoặc giữ lại thông tin [9] 21](#_Toc168436380)

[Hình 2. 6 Bước 2 - Quyết định thông tin có được lưu trạng thái không [9] 22](#_Toc168436381)

[Hình 2. 7 Bước 3 - Thực hiện tính Ct từ Ct-1 và 2 tỷ lệ trước đó [9] 23](#_Toc168436382)

[Hình 2. 8 Bước 4 - Thực hiện quyết định đầu ra mà ta muốn dự đoán[9] 24](#_Toc168436383)

[Hình 2. 9 Ví dụ về quá trình hoạt động của 1 mô hình LSTM [10] 25](#_Toc168436384)

[Hình 2. 10 So sánh cấu trúc LSTM và GRU [22] 26](#_Toc168436385)

[Hình 2. 11 Cấu trúc tổng quan của mô hình GRU [22] 26](#_Toc168436386)

[Hình 2. 12 Cấu trúc của mô hình tuần tự 28](#_Toc168436387)

[Hình 2. 13 Cấu trúc mô hình tuần tự cộng 29](#_Toc168436388)

[Hình 2. 14 Cấu trúc mô hình tuần tự nhân 30](#_Toc168436389)

[Hình 2. 15 Cấu trúc mô hình song song 30](#_Toc168436390)

[Hình 4. 1 Trực quan hóa tập dữ liệu thời tiết 59](#_Toc168436391)

[Hình 4. 2 Trực quan hóa tập dữ liệu giá vàng 60](#_Toc168436392)

[Hình 4. 3 Trực quan hóa tập dữ liệu dầu thô 61](#_Toc168436393)

[Hình 4. 4 Trực quan hóa tập dữ liệu giá bia 62](#_Toc168436394)

**DANH MỤC BẢNG BIỂU**

[Bảng 4. 1 Các bộ tham số sau khi tối ưu hóa sử dụng cho các mô hình đơn 66](#_Toc169558711)

[Bảng 4. 2 Các bộ tham số sau quá trình tối ưu hóa sử dụng cho mô hình tuần tự 66](#_Toc169558712)

[Bảng 4. 3 Các bộ tham số sau quá trình tối ưu hóa sử dụng cho mô hình tuần tự nhân 67](#_Toc169558713)

[Bảng 4. 4 Các bộ tham số sau quá trình tối ưu hóa sử dụng cho mô hình tuần tự cộng 68](#_Toc169558714)

[Bảng 4. 5 Mô hình CNN và một số mô hình có liên quan 69](#_Toc169558715)

[Bảng 4. 6 Mô hình LSTM và một số mô hình có liên quan 70](#_Toc169558716)

[Bảng 4. 7 Mô hình RNN và một số mô hình liên quan 70](#_Toc169558717)

[Bảng 4. 8 Mô hình GRU và một số mô hình liên quan 71](#_Toc169558718)

[Bảng 4. 9 Kết quả thực nghiệm quá trình tối ưu hóa Mô hình CNN](#_Toc169558718) 73

[Bảng 4. 10 Kết quả thực nghiệm quá trình tối ưu hóa Mô hình RNN 7](#_Toc169558718)3

[Bảng 4. 11 Kết quả thực nghiệm quá trình tối ưu hóa Mô hình LSTM 7](#_Toc169558718)4

[Bảng 4. 12 Kết quả thực nghiệm quá trình tối ưu hóa Mô hình GRU 7](#_Toc169558718)4

[Bảng 4. 13 Kết quả thực nghiệm, đánh giá các mô hình đơn 7](#_Toc169558719)5

[Bảng 4. 14 Kết quả thực nghiệm, đánh giá các mô hình tuần tự 7](#_Toc169558720)5

[Bảng 4. 15 Kết quả thực nghiệm, đánh giá các mô hình tuần tự nhân 7](#_Toc169558721)6

[Bảng 4. 16 Kết quả thực nghiệm, đánh giá các mô hình tuần tự cộng 7](#_Toc169558722)7

[Bảng 4. 17 Kết quả thực nghiệm, đánh giá các mô hình song song 7](#_Toc169558723)8

[Bảng 4. 18 Kết quả thực nghiệm và đánh giá giữa mô hình CNN và các mô hình lai ghép liên quan](#_Toc169558724) 79

[Bảng 4. 19 Kết quả thực nghiệm và đánh giá giữa mô hình LSTM và các mô hình lai ghép liên quan](#_Toc169558725) 80

[Bảng 4. 20 Kết quả thực nghiệm và đánh giá giữa mô hình RNN và các mô hình lai ghép liên quan 8](#_Toc169558726)1

[Bảng 4. 21 Kết quả thực nghiệm và đánh giá giữa mô hình GRU và các mô hình lai ghép liên quan 82](#_Toc169558727)

# **PHẦN 1: MỞ ĐẦU**

1. **Tính cấp thiết của đề tài**

* Trong bối cảnh phát triển nhanh chóng của công nghệ và dữ liệu lớn, việc dự đoán chuỗi thời gian trở thành một nhiệm vụ quan trọng trong nhiều lĩnh vực như tài chính, y tế, khí tượng học và năng lượng. Những dự đoán chính xác có thể giúp các tổ chức đưa ra quyết định thông minh, tối ưu hóa quy trình và tăng cường hiệu quả hoạt động. Với sự tiến bộ trong học sâu, các mô hình như mạng nơ-ron hồi quy (RNN), LSTM (Long Short-Term Memory) và GRU (Gated Recurrent Unit) đã chứng minh hiệu quả vượt trội trong việc xử lý và dự đoán dữ liệu chuỗi thời gian.
* Tuy nhiên, mỗi mô hình học sâu đều có những ưu và nhược điểm riêng. RNN đơn giản nhưng dễ gặp vấn đề về gradient bị tiêu biến; LSTM và GRU khắc phục được vấn đề này nhưng lại có cấu trúc phức tạp và yêu cầu nhiều tài nguyên tính toán. Việc lai ghép các mô hình học sâu nhằm tận dụng các điểm mạnh và giảm thiểu các điểm yếu của từng mô hình riêng lẻ đang trở thành một hướng nghiên cứu triển vọng. Bằng cách kết hợp các mô hình khác nhau, chúng ta có thể xây dựng một hệ thống dự đoán mạnh mẽ hơn, cải thiện độ chính xác và khả năng tổng quát hóa của mô hình.
* Đề tài "Lai ghép các mô hình học sâu dự đoán chuỗi thời gian" sẽ tập trung vào việc nghiên cứu và phát triển các mô hình lai ghép, bao gồm sự kết hợp giữa RNN, LSTM, GRU và các mô hình học sâu khác. Chúng ta sẽ khảo sát các phương pháp lai ghép hiện có, đề xuất những phương pháp mới và tiến hành thực nghiệm để đánh giá hiệu quả của chúng trên các tập dữ liệu thực tế. Mục tiêu là tìm ra các phương pháp lai ghép tối ưu, giúp nâng cao hiệu suất dự đoán và mở ra những ứng dụng mới trong nhiều lĩnh vực.

1. **Lý do chọn đề tài**

* Dự đoán chuỗi thời gian là một vấn đề quan trọng và có ứng dụng rộng rãi trong nhiều lĩnh vực, từ tài chính đến y tế, từ dự báo thời tiết đến quản lý năng lượng. Việc nghiên cứu và phát triển các phương pháp hiệu quả để dự đoán time series có thể mang lại nhiều lợi ích thiết thực.
* Các mô hình học sâu như RNN, LSTM, GRU đã chứng minh được khả năng mạnh mẽ trong việc xử lý dữ liệu chuỗi thời gian. Sự phát triển của công nghệ học sâu đã mở ra cơ hội cho việc tạo ra các mô hình dự đoán chính xác hơn và linh hoạt hơn.
* Kỹ thuật lai ghép (ensemble learning) là một phương pháp hiệu quả để kết hợp các mô hình khác nhau và tạo ra một mô hình tổng hợp có hiệu suất cao hơn. Việc áp dụng kỹ thuật này vào dự đoán chuỗi thời gian có thể giúp cải thiện độ chính xác và ổn định của mô hình.
* Dự đoán chuỗi thời gian đôi khi đối diện với những thách thức như biến động không đều, dữ liệu thiếu sót, hoặc ảnh hưởng của yếu tố ngẫu nhiên. Việc nghiên cứu và phát triển các phương pháp và mô hình để vượt qua những thách thức này là rất cần thiết.
* Với những lý do trên, việc chọn đề tài này không chỉ mang tính thực tiễn mà còn có tiềm năng đóng góp vào sự phát triển của ngành khoa học máy tính và ứng dụng công nghệ trong nhiều lĩnh vực khác nhau.

1. **Mục tiêu đề tài**

* Sử dụng các phương pháp lai ghép để xây dựng các mô hình kết hợp nhằm cải thiện độ chính xác của mô hình.

1. **Nhiệm vụ thực hiện đề tài**

* Xây dựng các mô hình đơn lẻ.
* Sử dụng các phương pháp lai ghép để xây dựng các mô hình kết hợp:
* Mô hình tuần tự.
* Mô hình tuần tự cộng.
* Mô hình tuần tự nhân.
* Mô hình song song.
* Xây dựng các phương pháp tối ưu hóa tham số.
* Thực nghiệm trên các tập dữ liệu khác nhau để đưa ra cái nhìn khách quan nhất trong việc nhìn nhận kết quả của các mô hình.
* Đánh giá thực nghiệm đạt được và đề xuất các mô hình có khả năng dự đoán tốt nhất.

1. **Giới hạn đề tài**

* Các nguồn dữ liệu sẽ tập trung vào các bộ dữ liệu có yếu tố chuỗi thời gian và dữ liệu số, có tính liên tục, và dự báo duy nhất một biến, chẳng hạn như: Giá vàng, giá cổ phiếu, giá đất, nhiệt độ, lượng mưa,… Điều này có thể giảm thiểu sự ảnh hưởng của các yếu tố khác đến quá trình học của máy, và giúp máy có thể học trong thời gian ngắn và đưa ra kết quả chính xác.
* Đề tài sẽ tập trung vào việc phát triển các mô hình học sâu: mô hình CNN, mô hình LSTM, mô hình RNN, mô hình GRU. Đồng thời, xây dựng một số phương pháp lai ghép, kết hợp mô hình để tạo ra các mô hình lai ghép có hiệu suất làm việc tốt hơn so với mô hình đơn.

1. **Bố cục**

* PHẦN MỞ ĐẦU:
  + Tính cấp thiết của đề tài
  + Lý do chọn đề tài
  + Mục tiêu đề tài
  + Nhiệm vụ đề tài
  + Giới hạn đề tài
* PHẦN NỘI DUNG:
  + Kiến thức cơ bản và các công trình liên quan
    - Các khái niệm cơ bản
    - Định nghĩa bài toán
    - Các mô hình và ưu, nhược điểm:
    - Các công trình liên quan
  + Các mô hình sử dụng
    - Mô hình CNN
    - Mô hình RNN
    - Mô hình LSTM
    - Mô hình GRU
    - Mô hình tuần tự
    - Mô hình tuần tự cộng
    - Mô hình tuần tự nhân
    - Mô hình song song
  + Cài đặt mô hình
    - Chuẩn bị dữ liệu và trực quan hóa dữ liệu
    - Xây dựng mô hình và các hàm liên quan
  + Thực nghiệm và đánh giá
    - Thực nghiệm
    - Các tiêu chí đánh giá
    - Các trường hợp thực nghiệm
    - Kết quả thực nghiệm
* PHẦN KẾT LUẬN:
  + Kết quả đạt được
  + Hạn chế
  + Hướng phát triển

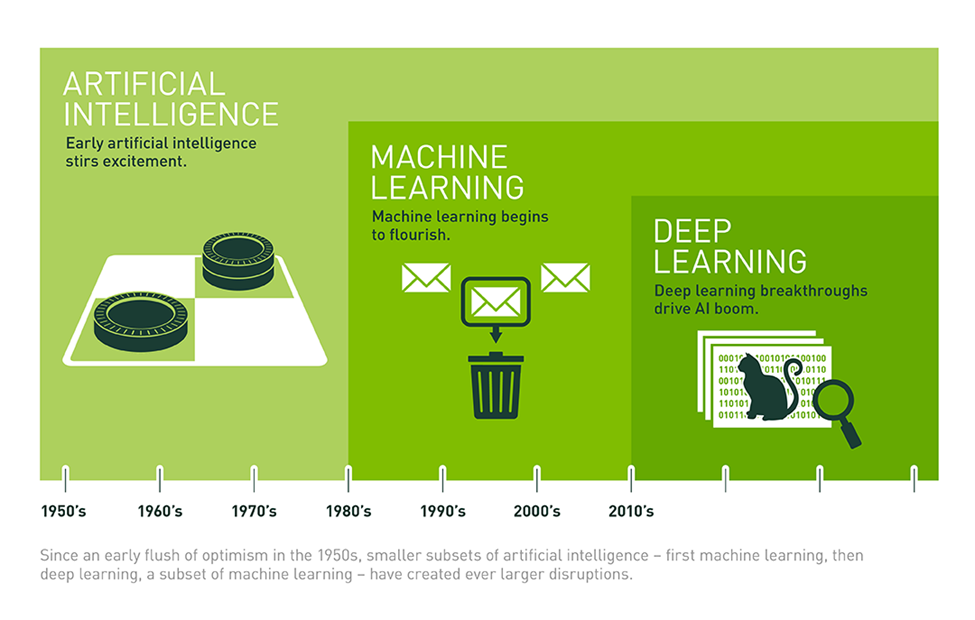
Tài liệu tham khảo

# **PHẦN 2: NỘI DUNG**

## **CHƯƠNG 1: KIẾN THỨC CƠ BẢN VÀ CÁC CÔNG TRÌNH LIÊN QUAN**

### **CÁC KHÁI NIỆM CƠ BẢN**

#### **Học máy**



Hình 1. 1 Sự hình thành và phát triển của Học máy, Học sâu [11]

* Máy học, hay Machine Learning, là một lĩnh vực của trí tuệ nhân tạo tập trung vào việc phát triển các thuật toán và mô hình máy tính có khả năng tự học và cải thiện theo thời gian thông qua kinh nghiệm, quan sát hoặc dữ liệu quá khứ. Mục tiêu chính của máy học là cho phép hệ thống máy tính hiểu và áp dụng tri thức từ dữ liệu mà không cần phải lập trình cụ thể cho từng nhiệm vụ. [2]
* **Phân loại học máy:**
  + *Học máy có giám sát* là một mô hình được huấn luyện dựa trên dữ liệu đã được gán nhãn, nghĩa là dữ liệu đầu vào đã được liên kết với kết quả mong muốn. Nói cách khác, dữ liệu huấn luyện bao gồm các cặp đầu vào và đầu ra mong đợi. Ví dụ: Hình ảnh những chữ số viết tay được chú thích để chỉ ra số tương ứng với hình ảnh đó,...
  + *Học máy không giám sát* là một thuật toán mà chúng ta không biết trước đầu ra hoặc nhãn của dữ liệu, chỉ có dữ liệu đầu vào. Máy học không giám sát sẽ dựa vào cấu trúc của dữ liệu để thực hiện các nhiệm vụ như phân nhóm (Clustering) hoặc giảm số chiều của dữ liệu (Dimension Reduction).
  + *Học máy bán giám sát* là các bài toán mà chúng ta có một lượng lớn dữ liệu nhưng chỉ một phần được gán nhãn được gọi là máy học bán giám sát. Những bài toán này nằm ở giữa máy học có giám sát và máy học không có giám sát. [2]
  + *Học máy tăng cường* là các bài toán giúp hệ thống tự động xác định hành vi dựa trên môi trường và hoàn cảnh, đồng thời nhận phản hồi dưới dạng phần thưởng và hình phạt. Thuật toán sẽ cố gắng tối đa hóa phần thưởng nhận được bằng cách điều chỉnh hành vi sao cho phù hợp. [3]

#### **Học sâu**

* Học Sâu (Deep Learning) là một nhánh của Học Máy (Machine Learning) tập trung vào việc sử dụng các mạng nơ-ron nhân tạo có cấu trúc sâu và phức tạp để học và biểu diễn dữ liệu. Deep Learning thường được áp dụng cho các tác vụ yêu cầu sự hiểu biết về cấu trúc và đặc trưng phức tạp của dữ liệu, chẳng hạn như nhận dạng hình ảnh, xử lý ngôn ngữ tự nhiên và dự đoán chuỗi thời gian. [4]
* Một số khái niệm khác:
  + **Overfitting (Quá Khớp):** Quá khớp xảy ra khi mô hình học rất tốt trên tập dữ liệu huấn luyện nhưng không thể tổng quát hóa cho dữ liệu mới. Điều này thường xảy ra khi mô hình quá phức tạp hoặc khi tập dữ liệu huấn luyện quá nhỏ.
  + **Underfitting (Quá Thấp):** Quá thấp xảy ra khi mô hình không học đủ tốt trên tập dữ liệu huấn luyện và không thể tổng quát hóa cho dữ liệu mới. Điều này thường xảy ra khi mô hình quá đơn giản hoặc khi tập dữ liệu huấn luyện quá lớn.
  + **Dropout:** Là một kỹ thuật chính để ngăn ngừa quá khớp trong mạng nơ-ron. Nó ngẫu nhiên tắt một số nơ-ron trong quá trình huấn luyện để tạo ra tính đa dạng trong mô hình. [13]
  + **Transfer Learning (Học Chuyển Giao):** Là một phương pháp sử dụng mô hình đã được huấn luyện sẵn (pre-trained) cho một nhiệm vụ cụ thể, sau đó điều chỉnh lại mô hình này cho nhiệm vụ mới. Điều này thường giúp tăng tốc quá trình huấn luyện và cải thiện hiệu suất.

#### **Dữ liệu chuỗi thời gian**

* Chuỗi thời gian (time series) là một dãy các sự kiện dữ liệu diễn ra liên tục trong một khung thời gian nhất định. Chuỗi thời gian đo lường sự biến động của các sự kiện dữ liệu đã chọn (chẳng hạn như giá cả thị trường chứng khoán) theo khung thời gian cụ thể.
* Dữ liệu chuỗi thời gian là kết quả của những phép đo lặp đi lặp lại theo thời gian. Dữ liệu này có mặt ở nhiều lĩnh vực khác nhau, bởi vì thời gian là yếu tố cơ bản của nhiều hiện tượng mà con người có thể nhận biết. [1]

#### **Dự báo dữ liệu chuỗi thời gian**

* Dự báo chuỗi thời gian là quá trình phân tích dữ liệu chuỗi thời gian bằng cách sử dụng các số liệu thống kê thu thập được và kết hợp với mô hình hóa dữ liệu để đưa ra dự đoán và đề xuất giải pháp tiếp theo. Dự báo chuỗi thời gian không phải lúc nào cũng chính xác, và độ chính xác của dự báo có thể thay đổi, đặc biệt khi phải xử lý các biến động bất thường trong chuỗi dữ liệu. [1]
* Ví dụ: Dự báo thời tiết, dự báo giá cổ phiếu, dự báo giá xăng dầu, vật tư,...

### **ĐỊNH NGHĨA BÀI TOÁN**

* Bài toán lai ghép các mô hình học sâu phục vụ cho dự đoán dữ liệu chuỗi thời gian là việc kết hợp nhiều mô hình học sâu khác nhau nhằm tận dụng ưu điểm của từng mô hình để cải thiện độ chính xác và hiệu quả dự đoán dữ liệu chuỗi thời gian. Quá trình này bao gồm việc thiết kế, huấn luyện và tối ưu hóa các mô hình học sâu, chẳng hạn như mạng nơ-ron hồi quy (RNN), mạng nơ-ron tích chập (CNN), và mạng nơ-ron dài ngắn hạn (LSTM), để dự đoán các giá trị tương lai của chuỗi thời gian. Bằng cách kết hợp các mô hình, hệ thống có thể xử lý tốt hơn các đặc điểm phức tạp và phi tuyến tính của dữ liệu, cũng như các yếu tố bất thường và biến động mạnh trong chuỗi thời gian.

### **CÁC MÔ HÌNH VÀ ƯU, NHƯỢC ĐIỂM**

#### **CNN - Convolutional Neural Network**

##### **Khái niệm**

* Mạng nơ-ron tích chập (Convolutional Neural Network - CNN) là một mô hình học sâu (Deep Learning) phổ biến với kiến trúc được thiết kế đặc biệt để xử lý các loại dữ liệu như hình ảnh và video. CNN được áp dụng rộng rãi trong các bài toán như nhận dạng hình ảnh, phân loại, phát hiện đối tượng, và nhiều nhiệm vụ khác liên quan đến xử lý hình ảnh. Ngoài ra, CNN cũng được sử dụng để phân tích dữ liệu chuỗi thời gian.
* Trong CNN dùng để phân tích chuỗi thời gian, dữ liệu đầu vào thường ở dạng chuỗi giá trị một chiều (1D). Các lớp tích chập có thể được sử dụng để xác định các mẫu trong chuỗi, như các xu hướng cục bộ hoặc các mẫu lặp lại. Kết quả từ các lớp tích chập này sau đó được sử dụng cho các nhiệm vụ như phân loại, hồi quy hoặc các loại phân tích khác. [7]

##### **Ưu điểm và nhược điểm**

* **Ưu điểm:**
* Tự động trích xuất đặc trưng: CNN có khả năng tự động học các đặc trưng quan trọng từ dữ liệu đầu vào mà không cần phải thiết kế thủ công​. [15]
* Khả năng nhận dạng đối tượng mạnh mẽ: CNN hoạt động rất tốt trong các bài toán nhận dạng hình ảnh, nhận dạng đối tượng, và phân loại hình ảnh​. [16],...
* **Nhược điểm:**
  + Yêu cầu tài nguyên tính toán lớn: Việc huấn luyện các mạng CNN thường đòi hỏi nhiều tài nguyên tính toán, bao gồm cả CPU và GPU mạnh​. [15]
  + Cần nhiều dữ liệu: Để huấn luyện một mô hình CNN tốt, cần có một lượng lớn dữ liệu huấn luyện để tránh tình trạng quá khớp​. [15]

#### **LSTM – Long Short Term Memory**

##### **Khái niệm**

* Mô hình LSTM là một biến thể đặc biệt của mạng nơ-ron hồi quy (RNN). Trong RNN truyền thống, việc dự đoán kết quả đầu ra gặp khó khăn khi thông tin quan trọng nằm xa trong chuỗi dữ liệu, dẫn đến giảm khả năng hiểu và dự đoán các chuỗi dài.
* Tuy nhiên, LSTM được thiết kế để khắc phục vấn đề phụ thuộc dài hạn (long-term dependency). Cấu trúc của LSTM cho phép nó duy trì thông tin quan trọng từ các bước thời gian xa hơn, giúp mô hình xử lý và phản hồi hiệu quả hơn đối với các chuỗi dữ liệu dài. [9]

##### **Ưu điểm và nhược điểm**

* **Ưu điểm:**
  + Giải quyết vấn đề phụ thuộc dài hạn: LSTM có khả năng lưu giữ thông tin trong thời gian dài nhờ cơ chế cửa (gates), giúp nó khắc phục vấn đề phụ thuộc xa mà các RNN thông thường gặp phải. [17]
  + Ứng dụng rộng rãi: LSTM được sử dụng trong nhiều lĩnh vực như nhận dạng giọng nói, dịch máy, phân tích chuỗi thời gian, phân loại văn bản, và nhiều hơn nữa. Khả năng xử lý chuỗi dài và phức tạp làm cho LSTM trở nên lý tưởng cho các ứng dụng này​.
* **Nhược điểm:**
  + Yêu cầu tài nguyên tính toán cao: Việc đào tạo mô hình LSTM đòi hỏi nhiều thời gian và tài nguyên tính toán, do tính phức tạp và số lượng lớn các tham số cần tối ưu​. [17]
  + Khó khăn trong việc điều chỉnh: LSTM có nhiều tham số và kiến trúc phức tạp, làm cho việc điều chỉnh và tối ưu mô hình trở nên khó khăn hơn so với các mô hình đơn giản hơn​.

#### **RNN – Recurrent Neural Networks**

##### **Khái niệm**

* Mạng nơ-ron hồi quy (Recurrent Neural Network - RNN) là một loại mạng nơ-ron được thiết kế để xử lý dữ liệu tuần tự như chuỗi thời gian hoặc ngôn ngữ tự nhiên. RNN có cấu trúc đặc biệt cho phép nó xử lý các đầu vào theo thứ tự tuần tự và duy trì trạng thái ẩn (hidden state) để lưu trữ thông tin từ các bước trước đó trong chuỗi. [14]

##### **Ưu điểm và nhược điểm**

* **Ưu điểm:**
  + Xử lý dữ liệu tuần tự: RNN được thiết kế để làm việc với các dữ liệu có tính tuần tự như chuỗi thời gian và ngôn ngữ tự nhiên, giúp nó có khả năng ghi nhớ và sử dụng thông tin từ các bước thời gian trước đó. [18]
  + Khả năng học các phụ thuộc dài hạn: RNN có khả năng học và nắm bắt các phụ thuộc dài hạn trong dữ liệu, điều này rất hữu ích cho các nhiệm vụ như dịch máy và phân tích cảm xúc​.[19]
* **Nhược điểm:**
  + Vấn đề về gradient biến mất: Khi độ dài của chuỗi dữ liệu tăng, các RNN truyền thống gặp khó khăn trong việc duy trì và học các phụ thuộc dài hạn do vấn đề gradient biến mất.
  + Đào tạo khó khăn: Việc huấn luyện RNN phức tạp và tốn nhiều tài nguyên tính toán. Việc tối ưu hóa RNN đòi hỏi nhiều thời gian và công sức để điều chỉnh các siêu tham số và kiến trúc mạng​. [18]
  + Khả năng xử lý chuỗi dài hạn hạn chế: Các RNN truyền thống gặp khó khăn trong việc xử lý các chuỗi rất dài do khả năng nhớ ngắn hạn, điều này dẫn đến mất thông tin quan trọng từ các bước thời gian trước đó​. [19]

#### **GRU - Gated Recurrent Unit**

##### **Khái niệm**

* Mô hình GRU (Gated Recurrent Unit) là một loại mạng nơ-ron hồi quy (RNN) được áp dụng rộng rãi trong xử lý ngôn ngữ tự nhiên và dự đoán chuỗi thời gian. Nó được giới thiệu bởi Kyunghyun Cho vào năm 2014 như một biến thể cải tiến của mô hình RNN, tương tự như mô hình LSTM.
* Tuy nhiên, so với cấu trúc phức tạp của LSTM, mô hình GRU có cấu trúc đơn giản hơn. Trong khi LSTM sử dụng cả hai trạng thái là Cell State (Ct) và Hidden State (Ht), GRU chỉ sử dụng một Hidden State (Ht). [20]

##### **Ưu điểm và nhược điểm**

* **Ưu điểm:**
* Tính hiệu quả tính toán: GRU có cấu trúc đơn giản hơn so với LSTM, giảm độ phức tạp tính toán và số lượng tham số cần huấn luyện. Điều này làm cho GRU thích hợp cho các ứng dụng có yêu cầu tính toán cao. [20]
* Khả năng học các phụ thuộc xa: Nhờ vào cơ chế cổng (gate mechanism), GRU có khả năng học và giữ lại thông tin từ quá khứ xa hơn, giúp cải thiện khả năng mô hình hóa các mối quan hệ dài hạn trong dữ liệu chuỗi. [21]
* **Nhược điểm:**
* Khả năng mô hình hóa phức tạp hơn của LSTM: Trong một số trường hợp, đặc biệt là khi đối mặt với dữ liệu có cấu trúc phức tạp và mối quan hệ xa, LSTM có thể vượt trội hơn so với GRU.
* Có thể gặp hiện tượng vanishing gradient: Mặc dù đã giảm thiểu vấn đề gradient bị tiêu biến (vanishing gradient) so với RNN tiêu chuẩn, nhưng GRU vẫn có thể gặp phải vấn đề này trong quá trình huấn luyện, đặc biệt là trên các chuỗi dài. [20]
  1. **CÁC CÔNG TRÌNH LIÊN QUAN**
* Tên công trình: “*Combining k-nearest neighbors-based method and neural network for time series prediction*”, Nguyen Thanh Son và Duong Tuan Anh
* Đường dẫn: <https://docs.google.com/document/d/1flyqFtA7Eqs21gsZirIi0a_GOsDdM6_v/edit?usp=drive_link&ouid=114970494274055372146&rtpof=true&sd=true>
* Mục tiêu: Đề xuất phương pháp kết hợp trong dự báo chuỗi thời gian. Phương pháp này kết hợp khớp mẫu và mạng nơ-ron để kết hợp ưu điểm của hai phương pháp này.
* Công nghệ sử dụng: Nghiên cứu và xây dựng phương pháp k-NN và ANN. Đồng thời, sử dụng phương pháp lai ghép song song để kết hợp 2 phương pháp.
* Kết quả: Kết quả thử nghiệm thu được cho thấy phương pháp lai hoạt động tốt hơn phương pháp ANN và phương pháp dựa trên k-NN về độ chính xác dự đoán.
* Tên công trình: “*Ứng Dụng Dự Báo Giá Chứng Khoán Bằng Mô Hình Lai Ghép Giữa Bộ Phân Lớp Đa Thức Và Mô Hình Học Sâu*”, Trần Văn Duy và Trần Công Trường, 5/2023.
* Đường dẫn: <https://drive.google.com/file/d/1A6U-fnVM9EuKKl02eOhhwFa1cmWQf1-e/view?usp=drive_link>
* Mục tiêu: Xây dựng mô hình dự báo giá chứng khoán với độ chính xác cao và đáng tin cậy bằng phương pháp lai ghép giữa phân lớp đa thức và mô hình học sâu.
* Công nghệ sử dụng: Tìm hiểu và xây dựng mô hình phân lớp đa thức và các mô hình học sâu (RNN, LSTM, GRU), đồng thời áp dụng phương pháp lai ghép (Mô hình cộng và mô hình song song).
* Kết quả: Xây dựng thành công mô hình lai ghép với độ chính xác khá cao. Các sai số tổng quan giữa các mô hình thể hiện rõ độ cải thiện của các phương pháp lai ghép.
* Tên công trình: “*CNN-LSTM vs. LSTM-CNN to Predict Power Flow Direction: A Case Study of the High-Voltage Subnet of Northeast Germany*”, Fachrizal Aksan và Yang Li và Vishnu Suresh và Przemysław Janik, 12/1/2023.
* Đường dẫn: <https://www.mdpi.com/1424-8220/23/2/901>
* Mục tiêu: Xây dựng mô hình dự đoán dựa trên sự kết hợp giữa mạng thần kinh tích chập (CNN) và bộ nhớ dài hạn (LSTM) để dự đoán luồng năng lượng trong cụm mạng được điều tra. Đánh giá và kiểm tra các yếu tố ảnh hưởng đến độ chính xác của mô hình kết hợp trong các trường hợp nhất định.
* Công nghệ sử dụng: Xây dựng 2 mô hình học sâu đơn lẻ là mô hình CNN và mô hình LSTM. Đồng thời xây dựng mô hình kết hợp tuần tự CNN-LSTM và LSTM-CNN.
* Kết quả: Kích thước của tập dữ liệu đầu vào có thể ảnh hưởng đến thời gian huấn luyện của mô hình. Mô hình kết hợp có thời gian chậm hơn các mô hình đơn. Tuy nhiên, dựa trên lỗi dự báo, mô hình kết hợp tốt hơn so với các mô hình cơ sở.
* Tên công trình: “*Random Search for Hyper-Parameter Optimization*”, James Bergstra và Yoshua Bengio.
* Đường dẫn: <https://jmlr.org/papers/volume13/bergstra12a/bergstra12a.pdf>
* Mục tiêu: Đánh giá cách hoạt động và mức độ hoạt động của RandomSearch so với GridSearch và tối ưu thủ công.
* Công nghệ sử dụng: GridSearchCV, RandomSearchCV.
* Kết quả: Hiệu suất của phương pháp tối ưu RandomSearchCV dường như không hoạt động tốt hơn phương pháp tối ưu GridSearchCV. Tuy nhiên, các đánh giá cho thấy rằng, GridSearchCV sẽ được ưu tiên dùng trong các trường hợp khi ta đã có 1 bộ tham số mà ta có thể chắc chắn mô hình hoạt động tốt. Ngược lại, phương pháp RandomSearchCV không hoạt động tốt như GridSearchCV, tuy nhiên, phương pháp này có thể nên được ưu tiên hơn khi ta có 1 bộ tham số mà ta không thể chắc chắn rằng các tham số đó hoạt động không tốt trong quá trình huấn luyện mô hình.
* Tên công trình: “*Introduction to Time Series and Forecasting - Second Edition*”, Peter J. Brockwell và Richard A. Davis.
* Đường dẫn: <https://ccs.fau.edu/~bressler/EDU/NTSA/References/B_D_In.pdf>
* Mục tiêu: Phân tích, đánh giá các yếu tố của chuỗi thời gian. Xây dựng mô hình để phân tích và dự báo chuỗi thời gian trong tương lai.
* Công nghệ sử dụng: Thuật toán ARAR, thuật toán Holt-Winters, thuật thoán Holt–Winters Seasonal, mô hình ARMA.
* Kết quả: Đánh giá các yếu tố ảnh hưởng đến chuỗi thời gian và kết hợp các thuật toán với mô hình ARMA và đưa ra kết luận. Tất cả các thử nghiệm đều phát hiện ra sai lệch đáng kể so với thực tế khi có các dấu hiện bất thường. Khi ta tăng số lượng mẫu đến 1 mức nhất định, sẽ có ít nhất một yếu tố chống lại kết quả dự đoán.
* Tên công trình: “*Time-series forecasting with deep learning: a survey*”, Bryan Lim và Stefan Zohren, 15/2/2021.
* Đường dẫn: <https://royalsocietypublishing.org/doi/10.1098/rsta.2020.0209>
* Mục tiêu: Khảo sát và thiết kế bộ mã hóa và bộ giải mã phổ biến được sử dụng trong việc dữ báo thời tiết. Nêu bật và phát triển các mô hình học sâu kết hợp, nghiên cứu kỹ lưỡng các thành phần mạng thần kinh để cải thiện các phương pháp.
* Công nghệ sử dụng: Deep learning, mô hình CNN, mô hình RNN, mô hình LSTM.
* Kết quả: Việc ứng dụng mạng Neural Network vào việc dự báo chuỗi thời gian là việc hoàn toàn có thể áp dụng. Tuy nhiên, cũng mang một số hạn chế như mạng lưới thần kinh thường yêu cầu chuỗi thời gian phải được rời rạc hóa theo các khoảng thời gian đều đặn, gây khó khăn cho việc dự báo các tập dữ liệu mà các quan sát có thể bị thiếu hoặc đến vào các khoảng thời gian ngẫu nhiên. Vì vậy, việc phát triển các kiến ​​trúc của các mô hình cần giải thích rõ ràng cho các hệ thống phân cấp có khả năng cải thiện hiệu suất dự báo so với các mô hình đơn biến hoặc đa biến hiện có.
* Tên công trình: “*Loss Functions And Metrics In Deep Learning*”, Edgar A. Chavez-Urbiola và Juan R. Terven và Diana M. Cordova-Esparza và Alfonzo Ramirez-Pedraza, 7/7/2023.
* Đường dẫn: <https://www.researchgate.net/publication/372163006_Loss_Functions_and_Metrics_in_Deep_Learning_A_Review>
* Mục tiêu: Xem xét các hàm mất mát phổ biến nhất và đo lường hiệu suất trong học sâu. Xem xét lợi ích và giới hạn của từng kỹ thuật cho các vấn đề học sâu khác nhau.
* Công nghệ sử dụng: Deep learning, Loss Functions, Regression, Classification.
* Kết quả: Việc lựa chọn hàm mất mát sẽ đánh giá thích hợp cho mô hình của bạn, nó thể hiện một cách khách quan độ chính xác của mô hình mà bạn đã xây dụng. Ngoài ra, việc xem xét tập dữ liệu các đặc điểm, đặc biệt là sự cân bằng giữa các lớp và các ngoại lệ trong khi thiết kế cũng sẽ giúp cho mô hình hoạt động ổn định hơn.
* Tên công trình: “*A Hybrid Approach For Crop Yield Prediction Using Machine Learning And Deep Learning Algorithms*”, Sonal Agarwal và Sandhya Tarar.
* Đường dẫn: <https://drive.google.com/file/d/1tdU9H7WGjug9siAEhf2pL_dso6OBH31E/view?usp=drive_link>
* Mục tiêu: Mô hình đề xuất được nâng cao bằng cách áp dụng các kỹ thuật học sâu và cùng với dự đoán về cây trồng, sẽ có được thông tin rõ ràng về lượng đất thành phần cần thiết với chi phí của họ. Xây dựng mô hình cung cấp với độ chính xác cao hơn so với mô hình hiện có. Phân tích điều kiện khí hậu và thổ nhưỡng của đất được xem xét để dự đoán một sản lượng thích hợp.
* Công nghệ sử dụng: mô hình SVM, mô hình LSTM, mô hình RNN.
* Kết quả: Mô hình đề xuất đã giảm bớt vấn đề của nông dân trong việc thất thoát, giảm thiệt hại do thiếu kiến ​​thức canh tác ở các điều kiện đất đai và thời tiết khác nhau. Mô hình dự đoán những loại cây trồng tốt nhất nên được trồng trên đất với chi phí thấp hơn trong số loại cây trồng. Với điều này, ngay cả khi chủ trang trại ở nhà, công việc vẫn có thể được quản lý ngay lập tức. Việc đề xuất và xây dựng mô hình này sẽ giúp ích nhiều hơn cho nông dân trong sản xuất cây trồng.
* Tên công trình: “*Universal Activation Function For Machine Learning*”, Brosnan Yuen, MinhTu Hoang, Xiaodai Dong và Tao Lu.
* Đường dẫn: <https://drive.google.com/file/d/1LCpApJczFP-YdC4QUREnsA1OQHx07fpT/view?usp=drive_link>
* Mục tiêu: Đánh giá, so sánh và xem xét mức độ hoạt động của các hàm kích hoạt trong ML. Đề xuất các hàm kích hoạt hoạt động tốt và phổ biến cho người xây dựng mô hình học máy.
* Kết quả: Việc kết hợp các lớp kích hoạt trong mạng nơ-ron sẽ mang lại hiệu suất tốt hơn. Nó giúp mạng nơ-ron có thể mô hình hóa nhiều quy trình phi tuyến tính hơn và giải quyết các vấn đề khó khăn hơn, ngoài ra, nó còn giúp mạng nơ-ron học hỏi lẫn nhau và hội tụ nhanh hơn.
* Tên công trình: “*A Study of the Optimization Algorithms in Deep Learning*”, Raniah Zaheer và Humera Shaziya.
* Đường dẫn: <https://www.researchgate.net/publication/339973030_A_Study_of_the_Optimization_Algorithms_in_Deep_Learning>
* Mục tiêu: So sánh và đánh giá hiệu suất giữa các phương pháp tối ưu hóa trong quá trình huấn luyện mô hình. Xem xét chu kỳ của các phương pháp tối ưu trong việc giải quyết các vấn đề thách thức đến quá trình học tập.
* Công nghệ sử dụng: Các phương pháp tối ưu hóa (SGD, NM, Adam,…).
* Kết quả: Các thuật toán tối ưu hóa có thể được xem xét theo nhiều cách khác nhau. Mỗi thuật toán tối ưu hóa phù hợp với các bộ dữ liệu khác nhau, tùy vào độ phức tạp của tập dữ liệu mà lựa chọn phương pháp tối ưu hóa sẽ giúp cho mô hình đạt hiệu suất tốt nhất.

## **CHƯƠNG 2: CÁC MÔ HÌNH SỬ DỤNG**

### **Mô hình CNN**

**A diagram of a layer of layers

Description automatically generated**

Hình 2. 1 Cấu trúc của một Mạng nơ-ron tích chập [12]

Mạng nơ-ron tích chập gồm 3 phần:

* **Phần đầu vào** (Input): Đây là nơi dữ liệu được truyền vào, có thể là hình ảnh hoặc dữ liệu chuỗi thời gian (Time Series).
* **Phần lớp ẩn** (Hidden Layers):
* Lớp tích chập (Convolutional Layer): Đây là khối xây dựng cốt lõi của CNN, nơi diễn ra phần lớn các hoạt động tính toán. Đầu vào thường là hình ảnh màu, được biểu diễn dưới dạng ma trận pixel 3D với chiều cao, chiều rộng và chiều sâu. Bộ lọc sẽ di chuyển qua các trường tiếp nhận của hình ảnh để kiểm tra sự xuất hiện của các đặc trưng.
* Lớp gộp (Pooling Layer): Lớp này được sử dụng để giảm kích thước dữ liệu, với hai loại chính là gộp trung bình (Average Pooling) và gộp tối đa (Max Pooling).
* Lớp làm phẳng (Flatten Layer): Chuyển dữ liệu từ dạng ma trận thành một vector, sau đó đưa vào các lớp kết nối đầy đủ (Fully Connected) để đưa ra kết quả dự đoán cuối cùng.
* Hàm kích hoạt (Activation Functions): Các hàm phi tuyến phổ biến bao gồm sigmoid, ReLU, tanh và LeakyReLU, giúp mô hình hóa các mối quan hệ phức tạp.
* Lớp loại bỏ ngẫu nhiên (Dropout): Giảm thiểu tình trạng quá khớp (overfitting) bằng cách loại bỏ ngẫu nhiên một số nơ-ron trong quá trình huấn luyện.
* Lớp kết nối đầy đủ (Fully Connected Layer): Mỗi nơ-ron được kết nối với tất cả các nơ-ron trong lớp trước đó, với các trọng số cần được học để tối ưu hóa hiệu suất của mô hình.
* **Phần đầu ra** (Output): Trả về dữ liệu dự đoán cuối cùng.[7]

### **Mô hình RNN**

#### Cấu trúc

**A screenshot of a computer

Description automatically generated**

Hình 2. 2 Tổng quan về Kiến trúc RNN [9]

Cấu trúc cơ bản của một mạng RNN bao gồm một chuỗi các đơn vị nơ-ron được gọi là cell (tế bào) được kết nối với nhau theo thứ tự chuỗi. Mỗi cell bao gồm:

* Input (Đầu vào): Mạng RNN nhận đầu vào một cách tuần tự theo thời gian, thường là một vector có số chiều nhất định. Mỗi cell có hai đầu vào: một từ phần tử hiện tại của chuỗi và một từ trạng thái ẩn của cell trước đó.
* Hidden State (Trạng thái ẩn): Ở mỗi bước thời gian, mạng RNN duy trì một trạng thái ẩn, là một vector số học dùng để lưu trữ và tái sử dụng thông tin từ quá khứ. Trạng thái ẩn này rất quan trọng trong việc duy trì thông tin liên quan tới dữ liệu đã được xử lý trước đó trong chuỗi thời gian.
* Activation Function (Hàm kích hoạt): Hàm kích hoạt thường được sử dụng ở mỗi bước thời gian là hàm tanh, để biểu diễn sự biến đổi phi tuyến của đầu vào và trạng thái ẩn.
* Time Step (Bước thời gian): Mạng RNN hoạt động dựa trên các bước thời gian liên tiếp nhau. Trạng thái ẩn trước đó (h\_{t-1}) được dùng để tính toán trạng thái ẩn hiện tại (h\_t).
* Output (Đầu ra): Tại mỗi bước thời gian, mạng RNN có thể tạo ra đầu ra (y\_t) dựa trên trạng thái ẩn tại thời điểm đó. Đầu ra này có thể được sử dụng để dự đoán chuỗi thời gian hoặc phân loại từng bước thời gian.
* Weights (Trọng số): Trong mạng RNN có hai bộ trọng số chính: trọng số đầu vào (W\_{in}) và trọng số cho trạng thái ẩn trước đó (W\_{rec}). Trong một số trường hợp, có thể có thêm trọng số đầu ra (W\_{out}) để thực hiện dự đoán đầu ra. [14]

#### Cơ chế hoạt động

A diagram of a network

Description automatically generated

Hình 2. 3 Cơ chế hoạt động của RNN [8]

**Bước 1: Đầu vào và Trạng thái ẩn ban đầu:** Ở bước thời gian đầu tiên ( = 0), mạng RNN nhận đầu vào bạn đầu () và trạng thái ẩn ban đầu () thường được khởi tạo bằng vector không hoặc một vector ngẫu nhiên.

**Bước 2: Tính toán trạng thái ẩn:** Tại mỗi bước thời gian , mạng RNN tính toán trạng thái ẩn () bằng công thức sau:

**=**

Trong đó:

* là trạng thái ẩn tại bước thời gian .
* là đầu vào tại bước thời gian .
* và là ma trậntrọng số cho đầu vào và trạng thái ẩn.
* là bias cho trạng thái ẩn, dùng để tinh chỉnh .
* là hàm kích hoạt với mạng RNN thì thường là hàm tanh hoặc hàm sigmoid.

**Bước 3: Tính toán đầu ra:** Tại mỗi bước thời gian , mạng RNN tính toán đầu ra () bằng công thức sau:

**= . +**

Trong đó:

* là vector đầu ra tại bước thời gian t.
* là ma trận trọng số giữa trạng thái ẩn và output.
* là trạng thái ẩn tại bước thời gian .
* là bias cho đầu ra.

**Bước 4: Lan truyền ngược (Backpropagation):** Sau khi tính toán đầu ra ở mỗi bước thời gian, mạng RNN sử dụng thuật toán lan truyền ngược để tính gradient của hàm mất mát đối với các tham số, bao gồm trọng số và bias.

**Bước 5: Cập nhật trọng số và Bias:** Gradient tính được được sử dụng để cập nhật các trọng số (, , ) và Bias (, ) thông qua các thuật toán tối ưu hóa như Gradient Descent hoặc biến thể của nó.

**Bước 6: Lặp lại quá trình:** Quá trình tính toán và cập nhật bias và trọng số được lặp đi lặp lại qua từng bước thời gian cho đến khi mô hình đạt được hiệu suất tốt trên tập dữ liệu đào tạo. [9]

### **Mô hình LSTM**

#### Cấu trúc

* Mô hình LSTM, bao gồm 3 cổng chính: Cổng quên, cổng đầu vào và cổng đầu ra. LSTM được cấu trúc cụ thể như sau:

**A screenshot of a computer

Description automatically generated**

Hình 2. 4 Cấu trúc của mạng LSTM [9]

* **Cổng quên (Forget Gate):** Quyết định thông tin nào sẽ bị loại bỏ từ trạng thái Cell State đã được tính trước đó (bộ nhớ dài hạn).
* **Cổng đầu vào (Input Gate):** Quyết định thông tin nào sẽ được đưa vào quá trình tính toán, tức là dữ liệu nào sẽ được thêm vào bộ nhớ từ trạng thái ẩn trước đó.
* **Cổng đầu ra (Output Gate):** Quyết định thông tin nào sẽ được sử dụng để đưa ra kết quả dự đoán thông qua các trạng thái của cell. [9]

#### Cơ chế hoạt động

* Bước 1: Quyết định các thông tin loại bỏ hay giữ lại. Thông tin sẽ được tính thông qua các trọng số nhân với ht-1 và xt, sau đó cộng với bf và hàm sigmoid. Kết quả thu được của quá trình sẽ nằm trong đoạn [0,1]

A diagram of a flowchart

Description automatically generated

Hình 2. 5 Bước 1 - Quyết định loại bỏ hoặc giữ lại thông tin [9]

* Trong đó:
  + 𝑓𝑡 : Tỷ lệ thông tin của bộ nhớ dài hạn được giữ lại.
  + 𝑊𝑓 : trọng số, trọng lượng.
  + x𝑡: dữ liệu đầu vào tại thời điểm t.
  + h𝑡-1: bộ nhớ ngắn hạn tại thời điểm trước.
  + b𝑓: bias điều chỉnh ft.
* Bước 2: Quyết định xem thông tin sẽ được lưu vào trạng thái. Ở bước này gồm 2 phần. Phần 1 tương tự như bước 1, và sau ham sigmoid đó ta cũng thu được 1 giá trị it trong đoạn [0,1]. Tiếp theo là một tầng tanh tạo ra một vectơ Ct nhận giá trị trong đoạn [-1,1]. Trong bước tiếp theo, ta sẽ kết hợp 2 giá trị đó lại để tạo ra một cập nhập cho trạng thái.

A diagram of a flowchart

Description automatically generated

Hình 2. 6 Bước 2 - Quyết định thông tin có được lưu trạng thái không [9]

* Trong đó:
  + i𝑡: tỷ lệ dữ liệu tiềm năng.
  + Ct: Cổng điều chế đầu vào.
  + Wi, Wc : ma trận trong lượng.
  + bi, bc: bias điều chỉnh .
* Bước 3: Từ 2 bước trước, ta đã tính được tỷ lệ của dữ liệu bị loại bỏ và dữ liệu sẽ được lưu vào trạng thái tế bào. Tiếp theo ta thực hiện tính Ct từ Ct-1 và 2 tỷ lệ trước đó.Trạng thái mới thu được này sẽ phụ thuộc vào việc ta quyết định cập nhập mỗi giá trị trạng thái ra sao.

A diagram of a flowchart

Description automatically generated

Hình 2. 7 Bước 3 - Thực hiện tính Ct từ Ct-1 và 2 tỷ lệ trước đó [9]

* Trong đó:
  + Ct: Trạng thái Cell State tại thời điểm t
  + Ct-1: Trạng thái Cell State tại thời điểm t-1
* Cuối cùng, ta thực hiện quyết định đầu ra mà ta muốn dự đoán là gì. Giá trị đầu ra sẽ được dựa trên trạng thái mới được cập nhật trên Cell State tại thời điểm t. Đầu tiên, ta chạy một tầng sigmoid để quyết định phần nào của trạng thái tế bào ta muốn xuất ra ta thu được ot. Sau đó, ta đưa trạng thái mới qua một hàm tanh để đưa giá trị nó về khoảng [-1,1], và nhân nó với đầu ra của cổng sigmoid để được giá trị đầu ra ta mong muốn.

A diagram of a flowchart

Description automatically generated

Hình 2. 8 Bước 4 - Thực hiện quyết định đầu ra mà ta muốn dự đoán[9]

* Trong đó:
  + ot: cổng đầu ra
  + bo: Bias điều chế đầu ra
  + Wo: ma trận trọng lượng
  + ht: vecto đầu ra tại thời điểm t.
* Để dễ hiểu hơn, ta sẽ cho 1 ví dụ đơn giản như sau: [10]

A diagram of a graph

Description automatically generated

Hình 2. 9 Ví dụ về quá trình hoạt động của 1 mô hình LSTM [10]

* Đầu tiên, tại cổng quên ta thực hiện xác định tỷ lệ input bị loại bỏ:

ft = σ(0.5 \* 2.7 + 1 \*1.63 +1.62) = 0,99

* Bước 2: Quyết định dữ liệu được lưu vào trạng thái:

it = σ(0.5 \* 2 + 1 \* 1.65 + 0.62) = 0.96

gt = tanh(0.5 \* 1.41 + 1 \* 0.94 - 0.32) = 0.87

* Bước 3: Thực hiện cập nhật lại trạng thái cho thời điểm t:

Ct = 0.99 \* 2 + 0.96 \* 0.87 = 2.82

* Cuối cùng, ta thực hiện tính toán và quyết định đầu ra mà ta muốn dự đoán:

ot = σ(0.5 \* 4.38 + 1 \* (-0.19) + 0.59) = 0.93

→ ht = 0.93 \* tanh (2.82) = 0.92

### **Mô hình GRU**

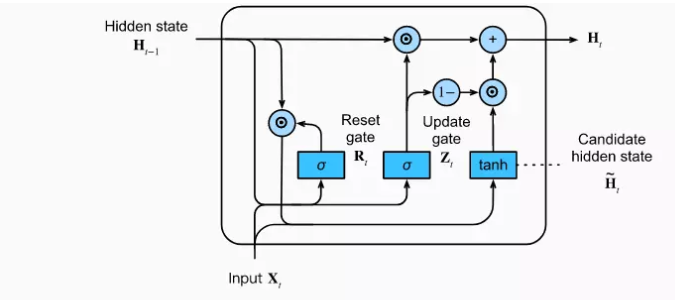
#### Cấu trúc

* So với cấu trúc của LSTM thì mô hình GRU có cấu trúc đơn giản hơn. Đối với mô hình LSTM có cả 2 Cell State (Ct) và Hidden State (Ht) thì GRU chỉ đơn giản có Hidden State (Ht). [22]

A chemical formula with black letters

Description automatically generated

Hình 2. 10 So sánh cấu trúc LSTM và GRU [22]



Hình 2. 11 Cấu trúc tổng quan của mô hình GRU [22]

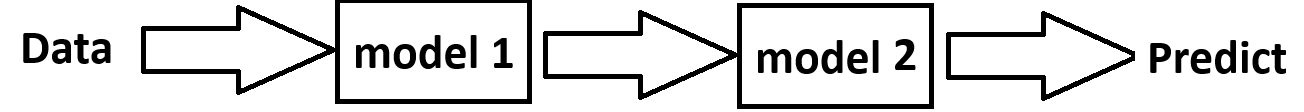
* Cấu trúc của GRU bao gồm:
  + Reset gate: Quyết định phần nào của thông tin cũ sẽ được bỏ qua.
  + Update gate: Quyết định phần nào của thông tin cũ sẽ được giữ lại và phần nào sẽ được cập nhật với thông tin mới
  + Candidate hidden state: ước lượng trạng thái mới mà có thể được cập nhật vào trạng thái hiện tại. Chứa thông tin từ trạng thái ẩn trước đó và đầu vào hiện tại, và có thể được cập nhật vào trạng thái hiện tại thông qua Update gate.[23]

#### Cơ chế hoạt động

* Cơ chế hoạt động của mô hình GRU (Gated Recurrent Unit) được thực hiện thông qua các bước sau:
  + Cổng cập nhật (Update gate):
  + Cổng cập nhật quyết định phần nào của thông tin từ quá khứ sẽ được giữ lại và phần nào sẽ được cập nhật với thông tin mới.
  + Cổng cập nhật được tính toán bằng cách sử dụng trạng thái ẩn trước đó (**H***t*−1​) và đầu vào hiện tại (**X***t*​).
  + Công thức tính toán: **Z***t*​ = *σ*(**X***t*​**W***xz*​+**H***t*−1​**W***hz*​+**b***z*​), trong đó **W***\_z*​ là ma trận trọng số.
  + Cổng đặc quyền (Reset gate):
  + Cổng đặc quyền quyết định phần nào của thông tin từ quá khứ sẽ được bỏ qua.
  + Cổng đặc quyền được tính toán bằng cách sử dụng trạng thái ẩn trước đó (**H***t*−1​) và đầu vào hiện tại (**X***t*​).
  + Công thức tính toán: **R***t*​ = *σ*(**X***t*​**W***xr*​+**H***t*−1​**W***hr*​+**b***r*​), trong đó **W**\_r là ma trận trọng số.
  + Trạng thái ẩn ứng cử viên (Candidate hidden state):
  + Trạng thái ẩn ứng cử viên (**H***t’*) là một ước lượng trạng thái mới mà có thể được cập nhật vào trạng thái ẩn hiện tại.
  + Trạng thái ẩn ứng cử viên được tính toán bằng cách kết hợp trạng thái ẩn trước đó (**H***t*−1) và đầu vào hiện tại (**X***t*​) sau khi đã được điều chỉnh bởi cổng đặc quyền (**R***t*​).
  + Công thức tính toán: **H***t’* ​= tanh(**X***t*​**W***xh*​+(**R***t*​⊙**H***t*−1​)**W***hh*​+**b***h*​), trong đó ⊙ đại diện cho phép nhân Hadamard (element-wise multiplication) và W\_h là ma trận trọng số.
  + Trạng thái ẩn mới (Hidden state):
  + Trạng thái ẩn mới (**H**t) được tính toán bằng cách kết hợp trạng thái ẩn trước đó (**H**t-1) và trạng thái ẩn ứng cử viên (**H**t’) thông qua cổng cập nhật (**Z**t).
  + Công thức tính toán: **H***t*​=**Z***t*​⊙**H***t*−1​ +(1−**Z***t*​)⊙**H**t’.

### **Mô hình tuần tự**

#### Cấu trúc



Hình 2. 12 Cấu trúc của mô hình tuần tự

* Khác với cấu trúc của các mô hình đơn, mô hình kết hợp tuần tự là một kiến trúc mạng nơ-ron kết hợp hai hoặc nhiều loại mạng nơ-ron khác nhau trong một hệ thống duy nhất. Thông thường, các loại mạng nơ-ron khác nhau có ưu điểm riêng và được kết hợp để tận dụng sức mạnh của mỗi loại mạng và cải thiện hiệu suất của mô hình. Việc kết hợp các loại mạng nơ-ron khác nhau trong một mô hình học sâu giúp tăng cường khả năng xử lý và hiểu dữ liệu phức tạp hơn.

#### Cơ chế hoạt động

* Khi nhận dữ liệu đầu vào, các mô hình ở lớp đầu tiên thường được dùng cho mục đích trích xuất các đặc trưng hoặc xử lý các thông tin của dữ liệu nhằm thuận tiện hơn trong quá trình dự đoán.
* Ở mô hình tiếp theo, những dữ liệu đã được sàn lọc và xử lý ở mô hình đầu tiên sẽ được đưa vào tính toán và đưa ra kết quả dự đoán.
* Việc tận dụng phương pháp này, ta có thể đồng thời kết hợp những điểm mạnh của các mô hình với nhau và tạo ra một mô hình kết hợp với hiệu quả tốt hơn. [24]

### **Mô hình tuần tự cộng**

#### Cấu trúc

**A diagram of a model

Description automatically generated**

Hình 2. 13 Cấu trúc mô hình tuần tự cộng

* Cấu trúc của mô hình tuần tự cộng cũng được cấu tạo bởi 2 mô hình khác nhau:
* Mô hình 1: thực hiện xử lý dữ liệu đầu vào và đưa ra dự đoán
* Mô hình 2: Tính toán lỗi dự đoán

#### Cơ chế hoạt động

* Khi nhận dữ liệu đầu vào X, mô hình 1 sẽ có nhiệm vụ tính toán và đưa ra dự đoán Y.
* Sau quá trình dự đoán của mô hình 1 kết thúc, mô hình tính toán độ sai lệch của quá trình dự đoán (X – Y).
* Với độ sai lệch vừa tính toán, mô hình 2 thực hiện xử lý dữ liệu đó và trả về kết quả N.
* Cuối cùng là cập nhật lại kết quả dự đoán bằng cách tính tổng kết quả dự đoán Y của mô hình 1 và kết quả N của mô hình 2 (Y + N). [25]

### **Mô hình tuần tự nhân**

A diagram of a model

Description automatically generated

Hình 2. 14 Cấu trúc mô hình tuần tự nhân

* Cấu trúc và cơ chế hoạt động của mô hình tuần tự nhân tương tự như mô hình tuần tự cộng.
* Tuy nhiên, trong qua trình tính độ lệch của mô hình 1 và kết quả thực tế sẽ được tính bằng cách lấy dữ liệu đầu vào chia lại cho kết quả dự đoán Y của mô hình 1.
* Và kết quả cuối cùng sẽ được tính bằng tính của kết quả dự đoán Y của mô hình 1 và kết quả N của mô hình 2 (Y \* N). [25]

### **Mô hình song song**

#### Cấu trúc

A diagram of a model

Description automatically generated

Hình 2. 15 Cấu trúc mô hình song song

* Cấu trúc của mô hình song song được cấu tạo bởi 2 mô hình đơn hoạt động cùng lúc. Việc xây dựng mô hình như trên có thể giúp ta tính toán lại và đánh giá mô hình để đưa ra kết quả có độ chính xác cao hơn so với mô hình đơn.
* Kết quả cuối cùng sẽ được tính toán dựa trên kết quả của 2 mô hình được sử dụng.

#### Cơ chế hoạt động

* Đầu tiên, dữ liệu đầu vào sẽ được xử lý đồng thời và cùng kích thước ở cả mô hình 1 và mô hình 2. Sau quá trình xử lý, ta lần lượt thu được 2 kết quả dự đoán của mô hình 1 là A và kết quả dự đoán của mô hình 2 là B.
* Kết quả dự đoán cuối cùng sẽ được tính với công thức sau:[26]

Predict = \* A + (1 - ) \* B

* Trong đó:

=

## **CHƯƠNG 3: CÀI ĐẶT MÔ HÌNH**

### **CHUẨN BỊ DỮ LIỆU VÀ TRỰC QUAN HÓA DỮ LIỆU**

#### **Chuẩn hóa dữ liệu**

|  |
| --- |
| #Chuan hoa du lieu      def normalize\_data(self, X\_value, y\_value, scaler = 'Min-Max'):          #chuan hoa min-max          if scaler == 'Min-Max':              X\_scaler = MinMaxScaler()              y\_scaler = MinMaxScaler()          # Chuan hoa zero-mean          elif scaler == 'Zero-Mean':              X\_scaler = StandardScaler()              y\_scaler = StandardScaler()          # du lieu goc          else:              return X\_value.to\_numpy(), y\_value.to\_numpy()          X\_scaler.fit(X\_value)          y\_scaler.fit(y\_value)            dump(X\_scaler, open('./static/X\_scaler.pkl', 'wb'))          dump(y\_scaler, open('./static/y\_scaler.pkl', 'wb'))          X\_scale\_dataset = X\_scaler.fit\_transform(X\_value)          y\_scale\_dataset = y\_scaler.fit\_transform(y\_value)          return X\_scale\_dataset, y\_scale\_dataset |

* Chuẩn hóa dữ liệu với 2 dạng chuẩn hóa được hỗ trợ từ viện của **sklearn** hoặc trả về dữ liệu gốc:
* Chuẩn hóa Min-Max
* Chuẩn hóa Zero-Mean
* **X\_scaler.fit(X\_value)** và **y\_scaler.fit(y\_value)** được sử dụng để tính toán các tham số chuẩn hóa cho dữ liệu đầu vào **X\_value** và **y\_value**. Từ đó, **dump(X\_scaler, open('./static/X\_scaler.pkl', 'wb'))** và **dump(y\_scaler, open('./static/y\_scaler.pkl', 'wb'))** được sử dụng để lưu trữ các tham số đã chuẩn hóa vào các tệp **X\_scaler.pkl** và **y\_scaler.pkl**. Cuối cùng, **X\_scaler.fit\_transform(X\_value)** và **y\_scaler.fit\_transform(y\_value)** được sử dụng để chuẩn hóa dữ liệu đầu vào **X\_value** và **y\_value** và lưu trữ kết quả vào **X\_scale\_dataset** và **y\_scale\_datase**t tương ứng.

#### **Chia tập dữ liệu x-y**

|  |
| --- |
| def get\_X\_y(self, X\_data, y\_data, n\_steps\_in, n\_steps\_out):          X = list()          y = list()          length = len(X\_data)          # chia dữ liệu vào các mẫu dữ liệu có độ dài nhất định cho việc huấn luyện mô hình          for i in range(0, length, 1):              X\_value = X\_data[i: i + n\_steps\_in][:, :]              y\_value = y\_data[i + n\_steps\_in: i + (n\_steps\_in + n\_steps\_out)][:, 0]              if len(X\_value) == n\_steps\_in and len(y\_value) == n\_steps\_out:                  X.append(X\_value)                  y.append(y\_value)          return np.array(X), np.array(y) |

* Hàm lấy dữ liệu đầu vào **X\_data** và **y\_data**, cùng với số ngày thời gian đầu vào **n\_steps\_in** và số ngày thời gian đầu ra **n\_steps\_ou**t. Hàm này trả về hai mảng numpy **X** và **y**, chứa các mẫu dữ liệu được chuẩn bị để huấn luyện mô hình dự đoán chuỗi thời gian.
* **X\_value** và **y\_value** được sử dụng để lấy các giá trị đầu vào và đầu ra cho mỗi mẫu dữ liệu. Sau đó, **X\_value** và **y\_value** được kiểm tra để đảm bảo rằng chúng có độ dài phù hợp với **n\_steps\_in** và **n\_steps\_out**. Nếu độ dài của **X\_value** và **y\_value** phù hợp, chúng được thêm vào các mảng **X** và **y** tương ứng.

#### **Chia tập dữ liệu train/validation/test**

|  |
| --- |
| #Chia tap du lieu train/validation/test      def split\_train\_test(self, data, train\_ratio = 0.7, valid\_ratio = 0.2):          # tính Độ dài của tập train/valid/test  total\_size = len(data)          train\_size = int(total\_size \* train\_ratio)          valid\_size = int(total\_size \* valid\_ratio)          test\_size = total\_size - train\_size - valid\_size          # Chia tap du lieu dua tren do dai cua tung tap          data\_train, data\_valid, data\_test = (          data[:train\_size, :],          data[train\_size:train\_size + valid\_size, :],          data[train\_size + valid\_size: train\_size + valid\_size + test\_size, :])          return data\_train, data\_valid, data\_test |

* Sau khi tính tổng độ dài **total\_size** = len(data), kích thước của tập dữ liệu **train/ validation/ test** được tính bằng tích của tỉ lệ từng tập với tổng độ dài **total\_size.**
* Sau khi tính kích thước của từng tập ta thực hiện chia dữ liệu cho từng tập tương ứng với kích thước của chúng. Kết quả, ta thu được 3 tập dữ liệu tương ứng **data\_train, data\_valid, data\_test**.

#### **Chia giá trị Index (ngày) cho tập train/validation/test**

|  |
| --- |
| def predict\_index(self, dataset, X\_train, X\_valid, n\_steps\_in, n\_steps\_out):          #Gia tri index train          train\_predict\_index = dataset.iloc[n\_steps\_in: X\_train.shape[0] + n\_steps\_in + n\_steps\_out - 1, :].index          #Gia tri index valid          valid\_start\_index = X\_train.shape[0] + n\_steps\_in + n\_steps\_out # Define the starting index for the validation set          valid\_end\_index = valid\_start\_index + X\_valid.shape[0] -1  # Define the ending index for the validation set          valid\_predict\_index = dataset.iloc[valid\_start\_index:valid\_end\_index, :].index            #Gia tri index test          test\_start\_index = valid\_end\_index  # Define the starting index for the test set          test\_predict\_index = dataset.iloc[test\_start\_index:, :].index          return train\_predict\_index, valid\_predict\_index, test\_predict\_index |

* Sau khi thực hiện chia dữ liệu cho 3 tập Train/ Validation/ Test, ta tiếp tục thực hiện chia các giá trị **Index**(ngày) tương ứng cho 3 tập trên. Hàm này trả về giá trị **train\_predict\_index, valid\_predict\_index, test\_predict\_index,** được sử dụng để lấy các chỉ mục dự đoán cho mỗi tập dữ liệu và chúng được tính toán dựa trên kích thước của **X\_train, X\_valid, n\_steps\_in, n\_steps\_out.**

#### **Làm sạch dữ liệu và hoàn thành việc chuẩn bị dữ liệu**

|  |
| --- |
| def CleanData(self, data, n\_steps\_in = 10, n\_steps\_out = 1, feature = 'Price',\                     train\_ratio = 0.7, valid\_ratio = 0.2, scaler = 'Min-Max'):            # loai bo cac ki tu va dau tren cac bien du lieu          data = data.dropna()          column\_names = tuple(data.drop(data.columns[0], axis=1).columns.values)          for column in column\_names:              data[column] = data[column].fillna('0').astype(str).str.replace(',', '')\                  .str.replace('K', 'e3').str.replace('M', 'e6')\                      .str.replace('%', 'e-2').map(lambda x: pd.eval(x) if x != 'nan' else np.nan).astype(float)          # Dinh dang cot dau tien theo dang ngay gio va sap xep du lieu          data[data.columns[0]] = pd.to\_datetime(data[data.columns[0]])          data = data.sort\_values(by=data.columns[0])          data.set\_index(data.columns[0], inplace=True)          # sao luu data truoc khi chia tap du lieu          data\_old = data          X\_value = data[[feature]]          y\_value = data[[feature]]          # Chuan hoa du lieu          X\_scale\_dataset, y\_scale\_dataset = self.normalize\_data(X\_value, y\_value, scaler)          # Chia tap X-y          X, y = self.get\_X\_y(X\_scale\_dataset, y\_scale\_dataset, n\_steps\_in, n\_steps\_out)          # Chia tap train/validation/test          X\_train, X\_valid, X\_test, = self.split\_train\_test(X, train\_ratio, valid\_ratio)          y\_train, y\_valid, y\_test, = self.split\_train\_test(y, train\_ratio, valid\_ratio)          # Chia index(ngay) train/validation/test          index\_train, index\_valid, index\_test, = self.predict\_index(data, X\_train, X\_valid, n\_steps\_in, n\_steps\_out)          return data, data\_old, X\_train, X\_test, X\_valid, y\_train, y\_test, \              y\_valid, index\_train, index\_test, index\_valid |

* Lấy tên các cột có trong file input, trừ cột đầu tiên (cột đầu tiên là cột thời gian của dòng dữ liệu). Tiếp theo, chúng ta bắt đầu biến đổi các cột dữ liệu cho đồng nhất. Ví dụ: Ở đây, các cột có xuất hiện giá trị “K” sẽ tương ứng e^3, chẳng hạn như giá trị 3.56K sẽ tương đương với 3.56 \* e^3.
* Tiếp theo, ta chuẩn hóa cột đầu tiên (cột thời gian của tập dữ liệu) trong data đầu vào theo dạng DateTime và tiến hành sort theo cột ngày.
* Sau đó, ta tiến hành chuẩn hóa dữ liệu, đồng thời chia tập dữ liệu thành các tập train, validation, test và các giá trị index tương ứng với các tập trên.

#### **Trực quan hóa dữ liệu**

|  |
| --- |
| def OneLine(data, selected\_column\_name):          trace = go.Scatter(x=data.data\_old.index ,y=data.data\_old[selected\_column\_name], mode='lines')          layout = go.Layout(              title='Biểu đồ trực quan hóa',              xaxis=dict(title='Ngày'),              yaxis=dict(title=''),              hovermode='closest'          )          fig = go.Figure(data=[trace], layout=layout)          fig.update\_layout()  # Kích thước tùy chỉnh 800x400          return fig |

* Sau khi thực hiện xử lý dữ liệu, cuối cùng ta thực hiện trực quan hóa dữ liệu lên biểu đồ với cột mà người dùng chọn.

### **XÂY DỰNG MÔ HÌNH VÀ CÁC HÀM LIÊN QUAN**

#### **Xây dựng mô hình đơn**

##### Mô hình CNN

|  |
| --- |
| def CNN\_Model(input\_dim=10, output\_dim=1, units = 32, learning\_rate = 0.0001, activation = 'relu'):      model = Sequential()      # Thêm lớp Convolutional 1D đầu tiên      model.add(Conv1D(units, input\_shape=(input\_dim, 1), \                       kernel\_size=3, strides=1, padding='same', activation=activation))      model.add(Conv1D(units, kernel\_size=3, strides=1, padding='same', activation=activation))      model.add(MaxPooling1D(pool\_size=2,strides=2, padding='same'))      # Hoàn thiện mô hình      model.add(Flatten())      model.add(Dense(256, activation=activation))      model.add(Dense(units=output\_dim))      # Thiết lập cấu hình cho mô hình để sẵn sàng cho quá trình huấn luyện.      model.compile(optimizer=Adam(learning\_rate=learning\_rate), loss='mse')      return model |

* Khởi tạo một mô hình tuần tự **Sequential()**. Sau đó, ta tạo lớp đầu tiên của mô hình với lớp 1D Convolution (**Conv1D**) với số filters = số lượng filter (số filter sẽ được set lại sau quá trình Optimize), **input\_shape = (input\_dim, 1)** với **input\_dim** là số điểm dữ liệu đầu vào (**n\_step\_in**) do người dùng chọn, kích thước **kernel** là 3, với bước nhảy **strides** =1. Và **padding** = ‘same’ sẽ tự động thêm các giá trị đệm vào đầu vào sao cho đầu ra có kích thước bằng với đầu vào. Và cuối cùng là hàm kích hoạt sẽ được thêm người dùng chọn (**ReLU, LeakyReLU, tanh**).
* Ở lớp tiếp theo, ta thêm một lớp **MaxPooling1D**, với kích thước là 2 và bước nhảy = 2, lớp này giúp ta có thể gộp các phần tử và loại bỏ bớt thông tin không cần thiết giúp quá trình học được tốt hơn. Sau hàm này, dữ liệu sau các lớp tích chập sẽ được giảm đi một nửa.
* Tiếp theo, ta xây dựng một mô hình mạng nơ-ron với các lớp **Flatten** và **Dense**. Lớp **Flatten** được sử dụng để chuyển đổi đầu vào thành một vector 1 chiều. Lớp **Dense** đầu tiên được sử dụng để thêm một lớp **Fully Connection** với 32 đơn vị đầu ra. Cuối cùng, lớp **Dense** được sử dụng để thêm một **Fully Connection** với output\_dim (số output đầu ra do người dùng chỉ định) đơn vị đầu ra và hàm kích hoạt được chỉ định.
* Cuối cùng là ta tiến hành **Compile** để cấu hình quá trình học của mô hình, sử dụng hàm tối ưu **Adam** với số **learning\_rate** được chỉ định và hàm mất mát mse.

##### Mô hình LSTM

|  |
| --- |
| def LSTM\_Model(input\_dim=10, output\_dim=1, units =32, learning\_rate=0.0001, activation = 'relu') -> tf.keras.models.Model:      model = Sequential()      model.add(LSTM(units=units, return\_sequences=True, input\_shape=(input\_dim, 1), activation=activation))      model.add(LSTM(units=units, activation=activation))      model.add(Dense(32, activation=activation))      model.add(Dense(units=output\_dim))      model.compile(optimizer=Adam(learning\_rate=learning\_rate), loss='mse')      return model |

* Tương tự như mô hình CNN, ta thực hiện xây dựng mô hình LSTM.
* Mô hình được xây dựng theo thứ tự sau:
* **model = Sequential():** Khởi tạo một mô hình tuần tự.
* **model.add(LSTM(units=units,return\_sequences=True, input\_shape=(input\_dim, 1), activation=activation)):** Thêm một layer **LSTM** vào mô hình với **units** là số lượng **units** trong layer này, **return\_sequences=True** cho phép layer trả về output cho mỗi sequence input, **input\_shape=(input\_dim, 1)** định nghĩa kích thước đầu vào của layer là **(số ngày đầu vào, 1)**, và **activation=activation** là hàm kích hoạt được sử dụng.
* **model.add(LSTM(units=units, activation=activation)):** Thêm một layer LSTM không trả về sequences. Điều này chỉ có ý nghĩa nếu layer trước đó đã trả về sequences. Các tham số khác giống như layer trước.
* **model.add(Dense(32, activation=activation)):** Thêm một layer Dense với 32 units và hàm kích hoạt activation.
* **model.add(Dense(units=output\_dim)):** Thêm một layer Dense với số units bằng output\_dim, không sử dụng hàm kích hoạt.
* **model.compile(optimizer=Adam(learning\_rate=learning\_rate), loss='mse'):** Compile mô hình với thuật toán tối ưu hóa Adam và hàm mất mát là 'mse' (mean squared error).

##### Mô hình RNN

|  |
| --- |
| def RNN\_Model(input\_dim=10, output\_dim=1, units =32, learning\_rate=0.0001, activation = 'relu') -> tf.keras.models.Model:      model = Sequential()      model.add(SimpleRNN(units=units, return\_sequences=True, \                          input\_shape=(input\_dim, 1), activation=activation))      model.add(SimpleRNN(units=units, activation=activation))      model.add(Dense(32, activation=activation))      model.add(Dense(units=output\_dim))      model.compile(optimizer=Adam(learning\_rate=learning\_rate), loss='mse')      return model |

* Mô hình được xây dựng theo thứ tự sau:
  + **model = Sequential()**: Khởi tạo một mô hình tuần tự.
  + **model.add(SimpleRNN(units=units, return\_sequences = True , input\_shape = (input\_dim, 1), activation=activation))**: Thêm một layer **SimpleRNN** vào mô hình với **units** là số lượng units trong layer này, **return\_sequences=True** cho phép layer trả về output cho mỗi sequence input, **input\_shape=(input\_dim, 1)** định nghĩa kích thước đầu vào của layer là (input\_dim, 1), và **activation=activation** là hàm kích hoạt được sử dụng.
  + **model.add(SimpleRNN(units=units, activation=activation))**: Thêm một layer SimpleRNN không trả về sequences. Điều này chỉ có ý nghĩa nếu layer trước đó đã trả về sequences. Các tham số khác giống như layer trước.
  + **model.add(Dense(32, activation=activation))**: Thêm một layer Dense với **32 units** và hàm kích hoạt activation.
  + **model.add(Dense(units=output\_dim))**: Thêm một layer Dense với số units bằng output\_dim, không sử dụng hàm kích hoạt.
  + **model.compile(optimizer=Adam(learning\_rate=learning\_rate), loss='mse')**: Compile mô hình với thuật toán tối ưu hóa Adam và hàm mất mát là 'mse' (mean squared error).

##### Mô hình GRU

|  |
| --- |
| def GRU\_Model(input\_dim=10, output\_dim=1, units =32, learning\_rate=0.0001, activation = 'relu') -> tf.keras.models.Model:      model = Sequential()      model.add(GRU(units=units, return\_sequences=True, input\_shape=(input\_dim, 1), activation=activation))      model.add(GRU(units=units, activation=activation))      model.add(Dense(32, activation=activation))      model.add(Dense(units=output\_dim))      model.compile(optimizer=Adam(learning\_rate=learning\_rate), loss='mse')      return model |

* Tương tự mô hình LSTM và RNN.

#### **Xây dựng mô hình tuần tự**

##### Mô hình CNN – LSTM

|  |
| --- |
| def CNN\_LSTM\_Model(input\_dim=10, output\_dim=1, units = 32, learning\_rate = 0.0001, activation = 'relu') -> tf.keras.models.Model:      model = Sequential()      #CNN      model.add(Conv1D(units, input\_shape=(input\_dim, 1), \                       kernel\_size=3, strides=1, padding='same', activation=activation))      model.add(Conv1D(units, kernel\_size=3, strides=1, padding='same', activation=activation))      model.add(MaxPooling1D(pool\_size=2,strides=2, padding='same'))      model.add(Flatten())      #tạo ra một tensor mới có hình dạng (None, out, units/filter)      model.add(RepeatVector(output\_dim))      #LSTM      model.add(LSTM(units=units, return\_sequences=True, activation=activation))      model.add(LSTM(units=units, activation=activation))      # fully connection      model.add(Dense(32, activation=activation))      model.add(Dense(units=output\_dim))      model.compile(optimizer=Adam(learning\_rate=learning\_rate), loss='mse')      return model |

* Trong mô hình trên, ta khởi tạo một mô hình tuần tự **Sequential()**.
* Sau đó ta thực hiện xây dựng mô hình CNN đơn giản với lớp đầu tiên là lớp tích chập 1D (**Conv1D**) với số lớp filter = số units (sẽ được chọn lại sau quá trình tối ưu hóa). Với **input\_shape=(input\_dim, 1),** với **input\_dim** là số điểm dữ liệu đầu vào (**n\_step\_in**) do người dùng chọn, kích thước **kernel** = 3, với bước nhảy **strides** =1. Và **padding** = ‘same’ sẽ tự động thêm các giá trị đệm vào đầu vào sao cho đầu ra có kích thước bằng với đầu vào. Và cuối cùng là hàm kích hoạt sẽ được thêm người dùng chọn (**ReLU, LeakyReLU, tanh**). Tương tự, ta xây dụng lớp tích chập thứ 2 với các thông số tương tự.
* Ở lớp tiếp theo, ta thêm một lớp **MaxPooling1D**, với kích thước là 2 và bước nhảy = 2, lớp này giúp ta có thể gộp các phần tử và loại bỏ bớt thông tin không cần thiết giúp quá trình học được tốt hơn. Sau hàm này, dữ liệu sau các lớp tích chập sẽ được giảm đi một nửa. Cuối cùng, ta thêm một lớp **Flatten()**, để biến đổi dữ liệu thành dạng vector 1 chiều.
* Sau khi xây dựng lớp đầu tiên là CNN, và kết quả đầu ra ở dạng vector 1 chiều. Ta thêm một lớp **RepeatVector()**, để biến đổi vector đầu ra ở mô hình đầu tiên phù hợp với đầu vào của mô hình thứ 2.
* Sau khi biến đổi dạng của mô hình thứ nhất, ta tiếp tục xây dựng mô hình thứ 2 với lớp đầu tiên là **LSTM(units=units, return\_sequences=True, activation = activation))**, với số lượng units = số units sau khi tối ưu hóa,  **return\_sequences = True** cho phép layer trả về output cho mỗi sequence input **và activation=activation** là hàm kích hoạt được sử dụng.
* **model.add(LSTM(units=units, activation=activation))**: Thêm một layer LSTM không trả về sequences. Điều này chỉ có ý nghĩa nếu layer trước đó đã trả về sequences. Các tham số khác giống như layer trước.
* **model.add(Dense(32, activation=activation))**: Thêm một layer Dense với 32 units và hàm kích hoạt activation.
* **model.add(Dense(units=output\_dim))**: Thêm một layer Dense với số units bằng output\_dim, không sử dụng hàm kích hoạt.
* **model.compile(optimizer=Adam(learning\_rate=learning\_rate), loss='mse')**: Compile mô hình với thuật toán tối ưu hóa Adam và hàm mất mát là 'mse' (mean squared error).

##### Mô hình LSTM – CNN

|  |
| --- |
| def LSTM\_CNN\_Model(input\_dim=10, output\_dim=1, units = 32, learning\_rate = 0.0001, activation = 'relu') -> tf.keras.models.Model:      model = Sequential()      #LSTM      model.add(LSTM(units=units, return\_sequences=True, input\_shape=(input\_dim, 1), activation=activation))      model.add(LSTM(units=units, activation=activation))        #tạo ra một tensor mới có hình dạng (None, out, units/filter)      model.add(RepeatVector(output\_dim))      #CNN      model.add(Conv1D(units, kernel\_size=3, strides=1, padding='same', activation=activation))      model.add(MaxPooling1D(pool\_size=2,strides=2, padding='same'))      model.add(Flatten())      # fully connection      model.add(Dense(32, activation=activation))      model.add(Dense(units=output\_dim))      model.compile(optimizer=Adam(learning\_rate=learning\_rate), loss='mse')      return model |

* Ngược lại với mô hình CNN - LSTM, mô hình LSTM – CNN xây dựng với mô hình đầu tiên là mô hình LSTM.
* Mô hình được xây dựng như sau:
* **model = Sequential()**: Khai báo mô hình tuần tự
* **model.add(LSTM(units=units, return\_sequences=True, input\_shape = (input\_dim, 1) , activation=activation))**: Thêm một layer **LSTM** vào mô hình với **units** là số lượng **units** trong layer này, **return\_sequences=True** cho phép layer trả về output cho mỗi sequence input, **input\_shape=(input\_dim, 1)** định nghĩa kích thước đầu vào của layer là **(số ngày đầu vào, 1)**, và **activation=activation** là hàm kích hoạt được sử dụng.
* **model.add(LSTM(units=units, activation=activation)):** Thêm một layer LSTM không trả về sequences. Điều này chỉ có ý nghĩa nếu layer trước đó đã trả về sequences. Các tham số khác giống như layer trước.
* **model.add(RepeatVector(output\_dim))**: lớp này biến đổi dạng tensor phù hợp dữ liệu đầu vào ở mô hình 2 (None, out, units/filter).
* **model.add(Conv1D(units, kernel\_size=3, strides=1, padding='same', activation=activation))**: Xây dựng lớp đầu tiên của mô hình thứ 2. Mô hình CNN, với lớp tích chập 1D (**Conv1D**) với số lớp filter= units, kích thước kernel = 3, bước nhảy strides = 1. Và **padding** = ‘same’ sẽ tự động thêm các giá trị đệm vào đầu vào sao cho đầu ra có kích thước bằng với đầu vào. Và cuối cùng là hàm kích hoạt sẽ được thêm người dùng chọn (**ReLU, LeakyReLU, tanh**).
* **model.add(MaxPooling1D(pool\_size=2,strides=2, padding='same'))**: Thêm một hàm gộp với kích thước = 2 và bước nhảy = 2.
* **model.add(Flatten())**: cuối cùng ta biến đổi dữ liệu đầu ra thành dạng vector 1 chiều.
* **model.add(Dense(32, activation=activation)):** Thêm một layer Dense với 32 units và hàm kích hoạt activation.
* **model.add(Dense(units=output\_dim)):** Thêm một layer Dense với số units bằng output\_dim, không sử dụng hàm kích hoạt.
* **model.compile(optimizer=Adam(learning\_rate=learning\_rate), loss='mse'):** Compile mô hình với thuật toán tối ưu hóa Adam và hàm mất mát là 'mse' (mean squared error).

##### Mô hình CNN – RNN

* Với ý tưởng xây dựng mô hình CNN – LSTM và LSTM – CNN, ta xây dựng các mô hình tuần tự còn lại tương tự.

|  |
| --- |
| def CNN\_RNN\_Model(input\_dim=10, output\_dim=1, units = 32, learning\_rate = 0.0001, activation = 'relu') -> tf.keras.models.Model:      model = Sequential()      #CNN      model.add(Conv1D(units, input\_shape=(input\_dim, 1), \                       kernel\_size=3, strides=1, padding='same', activation=activation))      model.add(Conv1D(units, kernel\_size=3, strides=1, padding='same', activation=activation))      model.add(MaxPooling1D(pool\_size=2,strides=2, padding='same'))      model.add(Flatten())      #tạo ra một tensor mới có hình dạng (None, out, units/filter)      model.add(RepeatVector(output\_dim))      #RNN      model.add(SimpleRNN(units=units, return\_sequences=True, activation=activation))      model.add(SimpleRNN(units=units, activation=activation))      # fully connection      model.add(Dense(32, activation=activation))      model.add(Dense(units=output\_dim))      model.compile(optimizer=Adam(learning\_rate=learning\_rate), loss='mse')      return model |

##### Mô hình RNN – CNN

|  |
| --- |
| def RNN\_CNN\_Model(input\_dim=10, output\_dim=1, units = 32, learning\_rate = 0.0001, activation = 'relu') -> tf.keras.models.Model:      model = Sequential()      #RNN      model.add(SimpleRNN(units=units, return\_sequences=True,\                      input\_shape=(input\_dim, 1), activation=activation))      model.add(SimpleRNN(units=units, activation=activation))      #tạo ra một tensor mới có hình dạng (None, out, units/filter)      model.add(RepeatVector(output\_dim))      #CNN      model.add(Conv1D(units, kernel\_size=3, strides=1, padding='same', activation=activation))      model.add(MaxPooling1D(pool\_size=2,strides=2, padding='same'))      model.add(Flatten())      # fully connection      model.add(Dense(32, activation=activation))      model.add(Dense(units=output\_dim))      model.compile(optimizer=Adam(learning\_rate=learning\_rate), loss='mse')  return model |

##### Mô hình CNN – GRU

|  |
| --- |
| def CNN\_GRU\_Model(input\_dim=10, output\_dim=1, units = 32, learning\_rate = 0.0001, activation = 'relu') -> tf.keras.models.Model:      model = Sequential()      #CNN      model.add(Conv1D(units, input\_shape=(input\_dim, 1), \                       kernel\_size=3, strides=1, padding='same', activation=activation))      model.add(Conv1D(units, kernel\_size=3, strides=1, padding='same', activation=activation))      model.add(MaxPooling1D(pool\_size=2,strides=2, padding='same'))      model.add(Flatten())      #tạo ra một tensor mới có hình dạng (None, out, units/filter)      model.add(RepeatVector(output\_dim))      #GRU      model.add(GRU(units=units, return\_sequences=True, activation=activation))      model.add(GRU(units=units, activation=activation))      # fully connection      model.add(Dense(32, activation=activation))      model.add(Dense(units=output\_dim))      model.compile(optimizer=Adam(learning\_rate=learning\_rate), loss='mse')  return model |

##### Mô hình GRU – CNN

|  |
| --- |
| def GRU\_CNN\_Model(input\_dim=10, output\_dim=1, units = 32, learning\_rate = 0.0001, activation = 'relu') -> tf.keras.models.Model:      model = Sequential()      #GRU      model.add(GRU(units=units, return\_sequences=True,\                      input\_shape=(input\_dim, 1), activation=activation))      model.add(GRU(units=units, activation=activation))      #tạo ra một tensor mới có hình dạng (None, out, units/filter)      model.add(RepeatVector(output\_dim))      #CNN      model.add(Conv1D(units, kernel\_size=3, strides=1, padding='same', activation=activation))      model.add(MaxPooling1D(pool\_size=2,strides=2, padding='same'))      model.add(Flatten())      # fully connection      model.add(Dense(32, activation=activation))      model.add(Dense(units=output\_dim))      model.compile(optimizer=Adam(learning\_rate=learning\_rate), loss='mse')  return model |

##### Mô hình LSTM – RNN

|  |
| --- |
| def LSTM\_RNN\_Model(input\_dim=10, output\_dim=1, units = 32, learning\_rate = 0.0001, activation = 'relu') -> tf.keras.models.Model:      model = Sequential()      #LSTM      model.add(LSTM(units=units, return\_sequences=True, input\_shape=(input\_dim, 1), activation=activation))      model.add(LSTM(units=units, activation=activation))        #tạo ra một tensor mới có hình dạng (None, out, units/filter)      model.add(RepeatVector(output\_dim))      #RNN      model.add(SimpleRNN(units=units, return\_sequences=True, activation=activation))      model.add(SimpleRNN(units=units, activation=activation))      # fully connection      model.add(Dense(32, activation=activation))      model.add(Dense(units=output\_dim))      model.compile(optimizer=Adam(learning\_rate=learning\_rate), loss='mse')  return model |

##### Mô hình RNN – LSTM

|  |
| --- |
| def RNN\_LSTM\_Model(input\_dim=10, output\_dim=1, units = 32, learning\_rate = 0.0001, activation = 'relu') -> tf.keras.models.Model:      model = Sequential()      #RNN      model.add(SimpleRNN(units=units, return\_sequences=True, input\_shape=(input\_dim, 1), activation=activation))      model.add(SimpleRNN(units=units, activation=activation))        #tạo ra một tensor mới có hình dạng (None, out, units/filter)      model.add(RepeatVector(output\_dim))      #LSTM      model.add(LSTM(units=units, return\_sequences=True, activation=activation))      model.add(LSTM(units=units, activation=activation))      # fully connection      model.add(Dense(32, activation=activation))      model.add(Dense(units=output\_dim))      model.compile(optimizer=Adam(learning\_rate=learning\_rate), loss='mse')  return model |

##### Mô hình LSTM – GRU

|  |
| --- |
| def LSTM\_GRU\_Model(input\_dim=10, output\_dim=1, units = 32, learning\_rate = 0.0001, activation = 'relu') -> tf.keras.models.Model:      model = Sequential()      #LSTM      model.add(LSTM(units=units, return\_sequences=True, input\_shape=(input\_dim, 1), activation=activation))      model.add(LSTM(units=units, activation=activation))        #tạo ra một tensor mới có hình dạng (None, out, units/filter)      model.add(RepeatVector(output\_dim))      #GRU      model.add(GRU(units=units, return\_sequences=True, activation=activation))      model.add(GRU(units=units, activation=activation))      # fully connection      model.add(Dense(32, activation=activation))      model.add(Dense(units=output\_dim))      model.compile(optimizer=Adam(learning\_rate=learning\_rate), loss='mse')  return model |

##### Mô hình GRU – LSTM

|  |
| --- |
| def LSTM\_GRU\_Model(input\_dim=10, output\_dim=1, units = 32, learning\_rate = 0.0001, activation = 'relu') -> tf.keras.models.Model:      model = Sequential()      #LSTM      model.add(LSTM(units=units, return\_sequences=True, input\_shape=(input\_dim, 1), activation=activation))      model.add(LSTM(units=units, activation=activation))        #tạo ra một tensor mới có hình dạng (None, out, units/filter)      model.add(RepeatVector(output\_dim))      #GRU      model.add(GRU(units=units, return\_sequences=True, activation=activation))      model.add(GRU(units=units, activation=activation))      # fully connection      model.add(Dense(32, activation=activation))      model.add(Dense(units=output\_dim))      model.compile(optimizer=Adam(learning\_rate=learning\_rate), loss='mse')  return model |

##### Mô hình RNN – GRU

|  |
| --- |
| def RNN\_GRU\_Model(input\_dim=10, output\_dim=1, units = 32, learning\_rate = 0.0001, activation = 'relu') -> tf.keras.models.Model:      model = Sequential()      #RNN      model.add(SimpleRNN(units=units, return\_sequences=True, input\_shape=(input\_dim, 1), activation=activation))      model.add(SimpleRNN(units=units, activation=activation))        #tạo ra một tensor mới có hình dạng (None, out, units/filter)      model.add(RepeatVector(output\_dim))      #GRU      model.add(GRU(units=units, return\_sequences=True, activation=activation))      model.add(GRU(units=units, activation=activation))      # fully connection      model.add(Dense(32, activation=activation))      model.add(Dense(units=output\_dim))      model.compile(optimizer=Adam(learning\_rate=learning\_rate), loss='mse')  return model |

##### Mô hình GRU – RNN

|  |
| --- |
| def GRU\_RNN\_Model(input\_dim=10, output\_dim=1, units = 32, learning\_rate = 0.0001, activation = 'relu') -> tf.keras.models.Model:      model = Sequential()      #GRU      model.add(GRU(units=units, return\_sequences=True, input\_shape=(input\_dim, 1), activation=activation))      model.add(GRU(units=units, activation=activation))        #tạo ra một tensor mới có hình dạng (None, out, units/filter)      model.add(RepeatVector(output\_dim))      #RNN      model.add(SimpleRNN(units=units, return\_sequences=True, activation=activation))      model.add(SimpleRNN(units=units, activation=activation))      # fully connection      model.add(Dense(32, activation=activation))      model.add(Dense(units=output\_dim))      model.compile(optimizer=Adam(learning\_rate=learning\_rate), loss='mse')  return model |

##### Huấn luyện mô hình

|  |
| --- |
| model\_training = eda.train\_model (m1 , epochs = best\_params['epochs'] , batch\_size=best\_params['batch\_size'])  def train\_model(self, model , epochs, batch\_size):          model.fit(self.X\_train, self.y\_train, epochs=epochs, \                    batch\_size=batch\_size, validation\_data=(self.X\_valid, self.y\_valid), verbose=2, shuffle=False)          return model |

* Thực hiện huấn luyện mô hình với tập dữ liệu train và validation. Trong đó batch\_size và epochs là các siêu tham số thu được sau quá trình Optimizer.

#### **Xây dựng mô hình tuần tự nhân**

##### Lựa chọn mô hình

|  |
| --- |
| match mod:     case 'CNN':  m1 = CNN\_Model(input\_dim=input\_dim, output\_dim=output\_dim, units = best\_params['units'], learning\_rate = best\_params['learning\_rate'], activation= activation)  case 'RNN':  m1 = RNN\_Model(input\_dim=input\_dim, output\_dim=output\_dim, units = best\_params['units'], learning\_rate = best\_params['learning\_rate'], activation= activation)  case 'GRU':  m1 = GRU\_Model(input\_dim=input\_dim, output\_dim=output\_dim, units = best\_params['units'], learning\_rate = best\_params['learning\_rate'], activation= activation)  case 'LSTM':  m1 = LSTM\_Model(input\_dim=input\_dim, output\_dim=output\_dim, units = best\_params['units'], learning\_rate = best\_params['learning\_rate'], activation= activation)  match mod1:  case 'CNN':  m2 = CNN\_Model(input\_dim=input\_dim, output\_dim=output\_dim, units = best\_params['units'], learning\_rate = best\_params['learning\_rate'], activation= activation)  case 'RNN':  m2 = RNN\_Model(input\_dim=input\_dim, output\_dim=output\_dim, units = best\_params['units'], learning\_rate = best\_params['learning\_rate'], activation= activation)  case 'GRU':  m2 = GRU\_Model(input\_dim=input\_dim, output\_dim=output\_dim, units = best\_params['units'], learning\_rate = best\_params['learning\_rate'], activation= activation)  case 'LSTM':  m2 = LSTM\_Model(input\_dim=input\_dim, output\_dim=output\_dim, units = best\_params['units'], learning\_rate = best\_params['learning\_rate'], activation= activation) |

* Khi lựa chọn mô hình sử dụng là mô hình tuần tự nhân, hệ thống yêu cầu người dùng lựa chọn mô hình sử dụng cho cả mô hình 1 và mô hình 2.
* Các tham số đầu vào **input\_dim, output\_dim** và **activation** được người dùng chọn khi Optimze/ Train. Các tham số **units**, **learning\_rate** được điều chỉnh sau quá trình **Optimizer**.

##### Huấn luyện mô hình

|  |
| --- |
| model\_training = eda.train\_model(m1,epochs=best\_params['epochs'], batch\_size=best\_params['batch\_size'])  predict\_train, actual\_train, index\_train, predict\_scale\_1\_train, actua\_scale\_train = eda.TestingModel(model\_training)  model\_training\_2 = eda.train\_model\_seq(m2, (actual\_train/predict\_train),epochs=best\_params['epochs'], batch\_size=best\_params['batch\_size'])  def train\_model(self, model , epochs, batch\_size):          model.fit(self.X\_train, self.y\_train, epochs=epochs, \  batch\_size=batch\_size, validation\_data=(self.X\_valid, self.y\_valid), verbose=2, shuffle=False)          return model    def train\_model\_seq(self, model, predict\_1, epochs, batch\_size):          model.fit(predict\_1, predict\_1, epochs=epochs, \                    batch\_size=batch\_size, verbose=2, shuffle=False)          return model |

* Tương tự mô hình đơn và mô hình tuần tự. Tuy nhiên, dữ liệu trong quá trình huấn luyện mô hình 2 sẽ được thay bằng **actual\_train/ predict\_train.**

#### **Xây dựng mô hình tuần tự cộng**

##### Lựa chọn mô hình

* Tương tự mô hình tuần tự nhân.

##### Huấn luyện mô hình

|  |
| --- |
| model\_training = eda.train\_model(m1,epochs=best\_params['epochs'], batch\_size=best\_params['batch\_size'])  predict\_train, actual\_train, index\_train, predict\_scale\_1\_train, actua\_scale\_train = eda.TestingModel(model\_training)  model\_training\_2 = eda.train\_model\_seq(m2, (actual\_train-predict\_train),epochs=best\_params['epochs'], batch\_size=best\_params['batch\_size']) |

* Khác với tuần tự nhân, mô hình 2 được huấn luyện với dữ liệu đầu vào là **actual\_train - predict\_train.**
* Các tham số đầu vào **input\_dim, output\_dim** và **activation** được người dùng chọn khi Optimze/ Train. Các tham số **units**, **learning\_rate** được điều chỉnh sau quá trình **Optimizer**.

#### **Xây dựng mô hình song song**

##### Lựa chọn mô hình

|  |
| --- |
| checkpoint = torch.load("./model/"+ mod +".pth")  model\_training = checkpoint["model"]  unit\_training = checkpoint["units"]  epochs\_traning = checkpoint['epochs']  batch\_size\_training = checkpoint['batch\_size']  LR\_training = checkpoint['learning\_rate']  time\_traing = checkpoint['time\_train']  checkpoint\_2 = torch.load("./model/"+ mod1 +".pth")  model\_training\_2 = checkpoint\_2["model"]  unit\_training\_2 = checkpoint\_2["units"]  epochs\_traning\_2 = checkpoint\_2['epochs']  batch\_size\_training\_2 = checkpoint\_2['batch\_size']  LR\_training\_2 = checkpoint\_2['learning\_rate']  time\_traing\_2 = checkpoint\_2['time\_train'] |

* Tương tự tuần tự nhân và tuần tự cộng. Sau khi lựa chọn mô hình sử dụng, thực hiện load các mô hình đơn từ các mục .pth tương ứng.

##### Xây dựng hàm tính toán

|  |
| --- |
| def calculate\_omega(y, y1, y2):      numerator = np.sum((y2 - y1) \* (y - y1))      denominator = np.sum((y2 - y1) \*\* 2)      omega = numerator / denominator if denominator != 0 else 0      # Ensure w\_parallel is between 0 and 1      omega = max(0, min(omega, 1))      return omega  predict\_1, actual\_1, index\_1, predict\_scale\_1, actua\_scale\_1 = eda.TestingModel(model\_train)  predict\_2, actual\_2, index\_2, predict\_scale\_2, actua\_scale\_2 = eda.TestingModel(model\_train\_2)  actual, index, actua\_scale = actual\_1, index\_1, actua\_scale\_1  omega = calculate\_omega(actual, predict\_1, predict\_2)  predict = omega \* predict\_1 + (1 - omega) \* predict\_2  predict\_scale = omega \* predict\_scale\_1 + (1 - omega) \* predict\_scale\_2 |

* Thực hiện tính toán Omega và tính kết quả cuối cùng.

#### **Xây dựng các hàm tối ưu hóa:**

* Bài toán được xây dựng với 3 phương pháp tối ưu hóa các tham số là RandomSearchCV, GridSearchCV, BayesSearchCV.
* RandomSearchCV: Khai báo các tham số cần truyền cho quá trình tối ưu. Sau đó thực hiện quá trình chọn ngẫu nhiên các tham số để thực hiện tối ưu và chọn ra bộ tham số có mức độ **neg\_mean\_squared\_error** cao nhất.

|  |
| --- |
| param\_ran = {                                  'units': [16, 32, 64, 128, 256],                                  'epochs': range(1, 101),                                  'batch\_size': [16, 32, 64, 128, 256],                                  'learning\_rate': [0.0001, 0.001]                                  }                              opti\_search = RandomizedSearchCV(m, param\_distributions=param\_ran, cv=3, n\_iter=10, n\_jobs=-1, scoring='neg\_mean\_squared\_error')                              opti\_search.fit(eda.X\_valid, eda.y\_valid) |

* GridSearchCV: Khai báo các tham số cần truyền cho quá trình tối ưu. Sau đó thực hiện quá trình tối ưu các tham số và cuối cùng chọn ra bộ tham số có mức độ **neg\_mean\_squared\_error** cao nhất.

|  |
| --- |
| param\_grid = {                                  'units': [32, 64, 128, 256],                                  'epochs': [20, 40, 60, 80, 100],                                  'batch\_size': [16, 32, 64, 128],                                  'learning\_rate': [0.0001, 0.001]                                  }                              opti\_search = GridSearchCV(m, param\_grid=param\_grid, n\_jobs=-1, cv=3 ,scoring='neg\_mean\_squared\_error', error\_score='raise')                                opti\_search.fit(eda.X\_valid, eda.y\_valid) |

* + - * BayesSearchCV:

|  |
| --- |
| param\_bay = {                                  'units': Categorical([16, 32, 64, 128, 256]),                                  'epochs': Integer(1, 100),                                  'batch\_size': Categorical([16, 32, 64, 128, 256]),                                  'learning\_rate': Categorical([0.0001, 0.001])                                  }                              opti\_search =BayesSearchCV(m, search\_spaces=param\_bay, cv=3, n\_iter=10, n\_jobs=-1, scoring='neg\_mean\_squared\_error') |

## **CHƯƠNG 4: THỰC NGHIỆM VÀ ĐÁNH GIÁ**

### **THỰC NGHIỆM**

#### **Môi trường thực nghiệm và các thư viện liên quan**

##### Môi trường thực nghiệm

* **Visual Studio Code** là phần mềm lập trình được phát triển bởi Microsoft trên ba nền tảng khác nhau và hỗ trợ đa ngôn ngữ. Ngoài ra, Visual Studio Code còn có chức năng tự hoàn thành lệnh thông minh và cải tiến mã nguồn.
* **Streamlit** là công cụ được xây dựng với mục đích dành cho Machine Learning Engineer, tạo ra giao diện web.

##### Các thư viện liên quan

* **TensorFlow** là một trong những Framework Deep Learning phổ biến nhất và cung cấp một số lượng lớn các lớp và công cụ để xây dựng mô hình CNN. TensorFlow cung cấp TensorFlow 2.0, một phiên bản cải tiến và dễ sử dụng hơn cho việc xây dựng mô hình CNN. Ở phiên bản này Keras đã trở thành một phần của TensorFlow giúp người dùng có thể sử dụng cả hai thư viện này dễ dàng và hiệu quả hơn.
* Keras là một thư viện phổ biến và rất dễ dàng sử dụng. Keras được tích hợp vào TensorFlow và được TensorFlow hỗ trợ chính thức. Thư viện này hỗ trợ xây dựng nhiều mô hình Deep Learning như: mạng recurrent neural network (RNN), mạng LSTM, mạng GRU, và nhiều kiến trúc mô hình khác.

#### **Dữ liệu thực nghiệm**

##### Dữ liệu thời tiết

A graph of blue lines

Description automatically generated

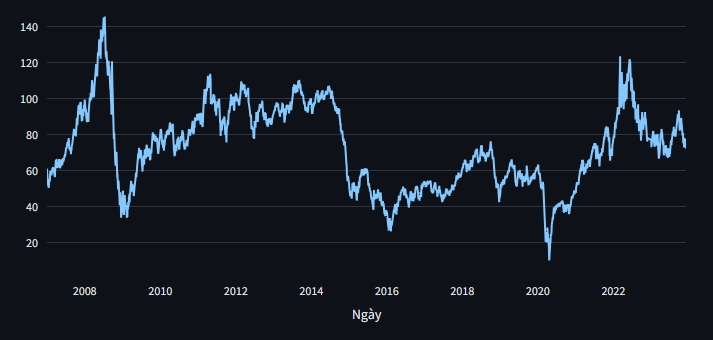
Hình 4. 1 Trực quan hóa tập dữ liệu thời tiết

* **Đặc tính:** mùa vụ.
* **Nguồn dữ liệu:** Dữ liệu được lấy từ website kaggle:

<https://www.kaggle.com/datasets/sumanthvrao/daily-climate-time-series-data>

* Bộ dữ liệu chứa thông tin dữ liệu thời tiết từ 1/1/2013 đến ngày 24/4/2017 tại thành phố Delhi, Ấn Độ.
* Dữ liệu bao gồm: 1575 hàng  và 5 cột.
* Ý nghĩa 5 thuộc tính trong tập dữ liệu như sau:
  + Meantemp: Nhiệt độ trung bình theo ngày.
  + Wind\_speed: Tốc độ gió của khu vực theo ngày.
  + Humidity: Độ ẩm của không khí theo từng khu vực, ngày.
  + Meanpressure: Áp suất không khí theo ngày.
  + date: Ngày tháng năm của khu vực.

##### Dữ liệu giá vàng



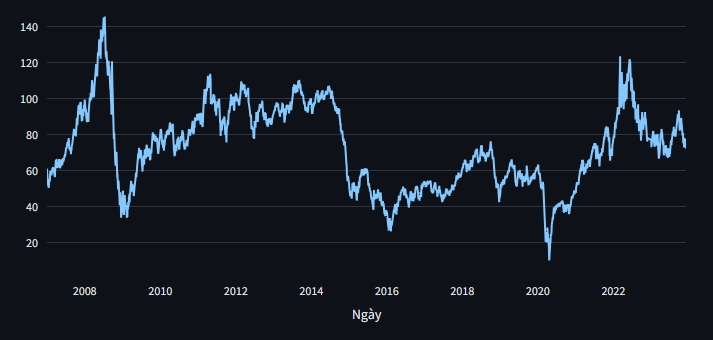
Hình 4. 2 Trực quan hóa tập dữ liệu giá vàng

* **Đặc tính:** xu hướng tăng + mùa vụ.
* **Nguồn dữ liệu:** Dữ liệu được lấy từ website kaggle:

<https://www.kaggle.com/datasets/kapturovalexander/gold-and-silver-prices-2013-2023>

* Tập dữ liệu chứa thông tin về giá vàng được thu nhập hàng ngày từ năm 2013 đến năm 2023.
* Độ lớn của dữ liệu: 2539 dòng tương ứng với 2539 ngày giao dịch từ 08/19/2013 tới 08/17/2023 và 6 thuộc tính.
* Ý nghĩa 6 thuộc tính trong tập dữ liệu như sau:
  + Date: Ngày giao dịch.
  + Close/Last: Giá thực hiện tại lần khớp lệnh cuối cùng trong ngày giao dịch.
  + Volume: Khối lượng giao dịch.
  + Open: Giá đóng cửa của phiên giao dịch hôm trước.
  + High: Mức giá cao nhất trong 1 chu kỳ theo dõi biến động.
  + Low: Mức giá thấp nhất trong 1 chu kỳ theo dõi biến động.

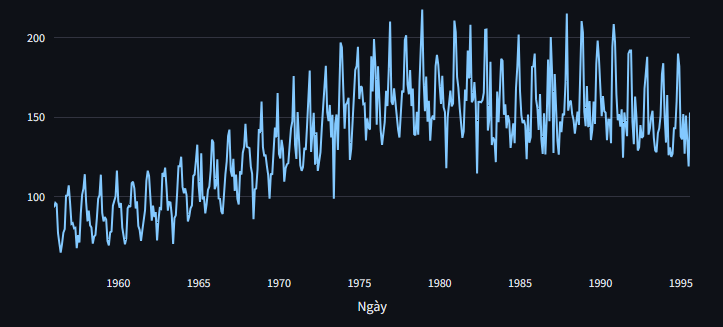
##### Dữ liệu dầu thô



Hình 4. 3 Trực quan hóa tập dữ liệu dầu thô

* **Đặc tính:** mùa vụ.
* **Nguồn dữ liệu:** Tập dữ liệu giá Dầu thô thế giới được thu thập từ: [https://vn.investing.com/commodities/crude-oil](https://vn.investing.com/commodities/crude-oil%20)
* Tập dữ liệu thống kê biến động của giá Dầu thô trên thế giới từng ngày. Dữ liệu được thu thập từ ngày 02/01/2007 đến 05/12/2023.
* Tập dữ liệu gồm 4242 dòng và 7 cột:
  + Date: ngày giao dịch cổ phiếu.
  + Open: giá mở cửa của dầu thô vào ngày giao dịch
  + High: giá cao nhất của dầu thô vào ngày giao dịch
  + Low: giá thấp nhất của dầu thô vào ngày giao dịch.
  + Close: giá đóng cửa của dầu thô vào ngày giao dịch.
  + Volume: khối lượng giao dịch của dầu thô trong ngày giao dịch.
  + Change %: Mức độ thay đổi trung bình của dầu thô vào ngày giao dịch.

##### Dữ liệu giá bia



Hình 4. 4 Trực quan hóa tập dữ liệu giá bia

* **Đặc tính:** Xu hướng tăng + mùa vụ.
* **Nguồn dữ liệu:** Tập dữ liệu từ kaggle :

<https://www.kaggle.com/code/mpwolke/australian-monthly-beer-production>

* Tập dữ liệu là số lượng bán ra theo từng tháng của Australia từ 1956 đến 1995
* Tập dữ liệu bao: 477 hàng gồm 2 cột.
* Month: Tháng từ năm 1956 đến 1995.
* Monthly beer production: Số lượng bán ra theo tháng.

##### Dữ liệu Train/ Validation/Test được chia như thế nào?

* Đối với đề tài của nhóm, tỷ lệ các tập dữ liệu Train/ Validation được chọn từ đầu là 70%/ 20%. Lý do mà nhóm chọn tỷ lệ trên, vì đây là tỷ lệ được dùng nhiều trong các trường hợp thực nghiệm và đạt kết quả tốt. Tỷ lệ Test sẽ được người dùng chọn lại tập dữ liệu đã huấn luyện với tỷ lệ 100% để nhóm có thể đánh giá khả năng dự đoán của mô hình trên dữ liệu lớn.
* Ngoài ra, trong quá trình huấn luyện, người dùng có thể thực hiện tùy ý điều chỉnh tỷ lệ tập Train/ Validation. Tuy nhiên, việc điều chỉnh tỷ lệ trên có thể sẽ gây ảnh hưởng đến độ chính xác của mô hình.

### **CÁC TIÊU CHÍ ĐÁNH GIÁ**

#### **Đánh giá dựa trên độ chính xác (lỗi) mô hình**

##### MSE

* Lỗi MSE (Mean Squared Error) là một phép đo để đánh giá sự khác biệt giữa giá trị dự đoán và giá trị thực tế trong các bài toán hồi quy trong lĩnh vực machine learning. [5]
* MSE được tính bằng cách lấy trung bình của bình phương sai số giữa giá trị dự đoán và giá trị thực tế trên tập dữ liệu huấn luyện. Công thức tính MSE như sau:

**MSE =**

* Trong đó:
* n: số phần tử của tập dữ liệu.
* yi: Giá trị thực tế thứ i.
* i: Giá trị dự đoán tại i.

##### RMSE

* Lỗi RMSE (Root Mean Squared Error) là một phép đo để đánh giá sự khác biệt giữa giá trị dự đoán và giá trị thực tế trong các bài toán hồi quy. Nó tương tự như MSE, nhưng khác biệt ở việc lấy căn bậc hai của MSE để đưa ra giá trị cuối cùng.
* RMSE được tính bằng cách lấy căn bậc hai của giá trị MSE. Công thức tính RMSE như sau:

**RMSE =**

##### MAE

* Tương tự, ta cũng có một cách khác để tính lỗi đó là tính lỗi MAE. MAE được tính bằng cách lấy trung bình của giá trị tuyệt đối của sai số giữa giá trị dự đoán và giá trị thực tế trên tập dữ liệu huấn luyện. Công thức tính MAE như sau: [6]

**MAE =**

* Trong đó:
* n: số phần tử của tập dữ liệu.
* yi: Giá trị thực tế thứ i.
* i: Giá trị dự đoán tại i.

##### CV\_RMSE

* Đây là một chỉ số đánh giá hiệu suất thường được sử dụng trong máy học và thống kê để đánh giá độ chính xác của một mô hình dự đoán. Nó được tính bằng cách lấy căn bậc hai của trung bình của bình phương của các sai số dự đoán.

**CV(RMSE) =**

* Trong đó:
* n: số phần tử của tập dữ liệu.
* yi: Giá trị thực tế thứ i.
* i: Giá trị dự đoán tại i.

#### **Đánh giá theo thời gian thực thi**

* Đánh giá mô hình dựa trên thời gian thực thi cũng là một yếu tố quan trọng để đánh giá hiệu suất của một mô hình. Thời gian thực thi có thể đo từ khi mô hình nhận dữ liệu đầu vào cho đến khi mô hình hoàn thành quá trình xử lý và trả về kết quả.

### **CÁC TRƯỜNG HỢP THỰC NGHIỆM**

* + - Trường hợp 1: So sánh quá trình tối ưu hóa siêu tham số của RandomSearchCV, GridSearhCV và BayesSearchCV.
    - Trường hợp 2: Đánh giá kết quả giữa các mô hình đơn
    - Trường hợp 3: Đánh giá kết quả giữa các mô hình tuần tự
    - Trường hợp 4: Đánh giá kết quả giữa các mô hình tuần tự nhân
    - Trường hợp 5: Đánh giá kết quả giữa các mô hình tuần tự cộng
    - Trường hơp 6: Đánh giá kết quả giữa các mô hình song song
    - Trường hợp 7: Đánh giá kết quả giữa các mô hình đơn và các mô hình lai ghép liên quan

#### So sánh quá trình tối ưu hóa siêu tham số của RandomSearchCV, GridSearhCV và BayesSearchCV

* + - * Trong trường hợp đầu tiên, ta sẽ xem xét, đánh giá và so sánh mức độ tối ưu của 2 phương pháp tối ưu đã xây dựng là RandomSearchCV và GridSearchCV đối với các mô hình đơn và tập dữ liệu được sử dụng là “**Delhi\_Climate.csv**”. Và cuối cùng, ta sẽ chọn ra phương pháp tối ưu hóa phù hợp với bài toán.

#### Đánh giá kết quả giữa các mô hình đơn

* Trong trường hợp thực nghiệm này, ta thực hiện so sánh và đánh giá mức độ chính xác của các mô hình đơn đối với tập dữ liệu “**3\_Gold\_Price.csv**”. Sau quá trình tối ưu hóa, các bộ tham số được sử dụng cho các mô hình như sau:

Bảng 4. 1 Các bộ tham số sau khi tối ưu hóa sử dụng cho các mô hình đơn

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **units** | **epochs** | **batch\_size** | **learning\_rate** |
| **CNN** | 16 | 82 | 16 | 0,001 |
| **LSTM** | 16 | 74 | 32 | 0,001 |
| **RNN** | 16 | 48 | 16 | 0,001 |
| **GRU** | 32 | 57 | 16 | 0,001 |

#### Đánh giá kết quả giữa các mô hình tuần tự

* Tương tự như trường hợp 4.3.1., ta thực hiện so sánh và đánh giá mức độ chính xác của các mô hình tuần tự trên tập dữ liệu “**3\_Gold\_Price.csv**”. Xem xét các mô hình tuần tự có thật sự hoạt động tốt trong việc dự báo hay không? Và mô hình nào sẽ đưa ra mô hình tốt hơn? Bộ tham số sau quá trình tối ưu hóa của các mô hình như sau:

Bảng 4. 2 Các bộ tham số sau quá trình tối ưu hóa sử dụng cho mô hình tuần tự

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **units** | **epochs** | **batch\_size** | **learning\_rate** |
| **CNN-LSTM** | 64 | 56 | 16 | 0,001 |
| **CNN-RNN** | 256 | 85 | 16 | 0,0001 |
| **CNN-GRU** | 128 | 92 | 32 | 0,001 |
| **LSTM-CNN** | 128 | 98 | 16 | 0,001 |
| **LSTM-RNN** | 128 | 94 | 16 | 0,001 |
| **LSTM-GRU** | 256 | 87 | 16 | 0,001 |
| **RNN-CNN** | 256 | 79 | 16 | 0,0001 |
| **RNN-LSTM** | 64 | 95 | 32 | 0,001 |
| **RNN-GRU** | 128 | 84 | 16 | 0,0001 |
| **GRU-CNN** | 256 | 73 | 32 | 0,0001 |
| **GRU-LSTM** | 256 | 45 | 16 | 0,001 |
| **GRU-RNN** | 256 | 91 | 16 | 0,0001 |

#### Đánh giá kết quả giữa các mô hình tuần tự nhân

* Trong trường hợp này, ta sẽ xem xét và đánh giá kết quả giữa các mô hình tuần tự nhân trên tập dữ liệu “**8\_DauTho**\_**Historical Data.csv**”. Bộ tham số của các mô hình tuần tự nhân sau quá trình tối ưu hóa như sau:

Bảng 4. 3 Các bộ tham số sau \*quá trình tối ưu hóa sử dụng cho mô hình tuần tự nhân

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **units** | **epochs** | **batch\_size** | **learning\_rate** |
| **CNN-LSTM\_mul** | 256 | 93 | 32 | 0,001 |
| **CNN-RNN\_mul** | 256 | 76 | 16 | 0,0001 |
| **CNN-GRU\_mul** | 128 | 89 | 32 | 0,0001 |
| **LSTM-LSTM\_mul** | 256 | 71 | 128 | 0,001 |
| **LSTM-RNN\_mul** | 64 | 100 | 32 | 0,001 |
| **LSTM-GRU\_mul** | 256 | 62 | 16 | 0,0001 |
| **RNN-LSTM\_mul** | 256 | 100 | 32 | 0,001 |
| **RNN-RNN\_mul** | 256 | 90 | 64 | 0,001 |
| **RNN-GRU\_mul** | 256 | 84 | 256 | 0,001 |
| **GRU-LSTM\_mul** | 256 | 50 | 32 | 0,001 |
| **GRU-RNN\_mul** | 64 | 30 | 64 | 0,001 |
| **GRU-GRU\_mul** | 256 | 94 | 256 | 0,001 |

#### Đánh giá kết quả giữa các mô hình tuần tự cộng

* Tương tự như trường hợp trước (4.3.3.), ta đánh giá độ chính xác của các mô hình tuần tự cộng trên tập dữ liệu “**8\_DauTho**\_**Historical Data.csv**”. Bộ tham số của các mô hình tuần tự cộng sau quá trình tối ưu hóa như sau:

Bảng 4. 4 Các bộ tham số sau quá trình tối ưu hóa sử dụng cho mô hình tuần tự cộng

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **units** | **epochs** | **batch\_size** | **learning\_rate** |
| **CNN-LSTM\_add** | 16 | 55 | 16 | 0,001 |
| **CNN-RNN\_add** | 256 | 58 | 16 | 0,001 |
| **CNN-GRU\_add** | 64 | 84 | 16 | 0,001 |
| **LSTM-LSTM\_add** | 256 | 89 | 32 | 0,0001 |
| **LSTM-RNN\_add** | 32 | 49 | 64 | 0,001 |
| **LSTM-GRU\_add** | 128 | 30 | 32 | 0,001 |
| **RNN-LSTM\_add** | 256 | 92 | 32 | 0,0001 |
| **RNN-RNN\_add** | 256 | 63 | 16 | 0,0001 |
| **RNN-GRU\_add** | 64 | 45 | 64 | 0,001 |
| **GRU-LSTM\_add** | 256 | 100 | 256 | 0,001 |
| **GRU-RNN\_add** | 128 | 78 | 256 | 0,001 |
| **GRU-GRU\_add** | 128 | 58 | 32 | 0,001 |

#### Đánh giá kết quả giữa các mô hình song song

* Sau khi thực nghiệm và đánh giá các mô hình đơn ở trường hợp 1 (4.3.1.) trên tập dữ liệu “**3\_Gold\_Price.csv**”, ta lai ghép các mô hình đơn đã huấn luyện thành mô hình song song. Sau khi thu được kết quả, ta tiến hành đánh giá mức độ cải thiện của mô hình song song đối với mô hình đơn trong việc dự đoán chuỗi thời gian.

#### Đánh giá kết quả giữa các mô hình đơn và các mô hình lai ghép liên quan

* Trong trường hợp này, ta tiến hành đánh giá độ chính xác của một mô hình đơn và các mô hình liên quan trên tập dữ liệu “**9\_monthly-beer-production.csv**”. Nhóm sẽ lựa chọn 4 trường hợp như sau:
* Trường hợp 1: Mô hình CNN và một số mô hình liên quan

Bảng 4. 5 Mô hình CNN và một số mô hình có liên quan

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | | **units** | **epochs** | **batch\_size** | **learning\_rate** |
| **CNN** | | 128 | 87 | 16 | 0,001 |
| **CNN - LSTM** | | 128 | 56 | 16 | 0,001 |
| **LSTM - CNN** | | 64 | 84 | 16 | 0,001 |
| **CNN-LSTM\_add** | | 64 | 94 | 32 | 0,001 |
| **CNN-RNN\_add** | | 128 | 99 | 16 | 0,001 |
| **CNN-RNN\_mul** | | 64 | 91 | 16 | 0,001 |
| **CNN-GRU\_mul** | | 256 | 60 | 16 | 0,001 |
| **CNN-LSTM\_para** | **CNN** | 128 | 87 | 16 | 0,001 |
| **LSTM** | 32 | 99 | 32 | 0,001 |
| **CNN-GRU\_para** | **CNN** | 128 | 87 | 16 | 0,001 |
| **GRU** | 16 | 91 | 32 | 0,001 |

* Trường hợp 2: Mô hình LSTM và một số mô hình liên quan

Bảng 4. 6 Mô hình LSTM và một số mô hình có liên quan

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | | **units** | **epochs** | **batch\_size** | **learning\_rate** |
| **LSTM** | | 32 | 99 | 32 | 0,001 |
| **LSTM - RNN** | | 32 | 56 | 16 | 0,001 |
| **RNN - LSTM** | | 64 | 68 | 32 | 0,001 |
| **LSTM-RNN\_add** | | 16 | 100 | 32 | 0,001 |
| **RNN-LSTM\_add** | | 64 | 95 | 128 | 0,001 |
| **LSTM-RNN\_mul** | | 64 | 65 | 32 | 0,001 |
| **RNN-LSTM\_mul** | | 256 | 96 | 128 | 0,001 |
| **LSTM-RNN\_para** | **LSTM** | 32 | 99 | 32 | 0,001 |
| **RNN** | 128 | 66 | 16 | 0,0001 |
| **LSTM-GRU\_para** | **LSTM** | 32 | 99 | 32 | 0,001 |
| **GRU** | 16 | 91 | 32 | 0,001 |

* Trường hợp 3: Mô hình RNN và một số mô hình liên quan

Bảng 4. 7 Mô hình RNN và một số mô hình liên quan

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | | **units** | **epochs** | **batch\_size** | **learning\_rate** |
| **RNN** | | 128 | 66 | 16 | 0,0001 |
| **RNN - GRU** | | 256 | 27 | 16 | 0,001 |
| **GRU - RNN** | | 256 | 29 | 32 | 0,001 |
| **RNN-GRU\_add** | | 32 | 82 | 16 | 0,001 |
| **GRU-RNN\_add** | | 16 | 99 | 16 | 0,001 |
| **RNN-GRU\_mul** | | 64 | 75 | 16 | 0,001 |
| **GRU-RNN\_mul** | | 64 | 72 | 32 | 0,001 |
| **RNN-GRU\_para** | **RNN** | 128 | 66 | 16 | 0,0001 |
| **GRU** | 16 | 91 | 32 | 0,001 |
| **CNN-RNN\_para** | **CNN** | 128 | 87 | 16 | 0,001 |
| **RNN** | 128 | 66 | 16 | 0,0001 |

* Trường hợp 4: Mô hình GRU và một số mô hình liên quan

Bảng 4. 8 Mô hình GRU và một số mô hình liên quan

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | | **units** | **epochs** | **batch\_size** | **learning\_rate** |
| **GRU** | | 16 | 91 | 32 | 0,001 |
| **GRU - CNN** | | 64 | 89 | 16 | 0,001 |
| **CNN - GRU** | | 32 | 98 | 32 | 0,001 |
| **GRU-LSTM\_add** | | 64 | 82 | 16 | 0,001 |
| **LSTM-GRU\_add** | | 64 | 77 | 256 | 0,001 |
| **GRU-LSTM\_mul** | | 32 | 66 | 32 | 0,001 |
| **LSTM-GRU\_mul** | | 64 | 65 | 16 | 0,001 |
| **CNN-GRU\_para** | **LSTM** | 32 | 99 | 32 | 0,001 |
| **RNN** | 128 | 66 | 16 | 0,0001 |
| **LSTM-GRU\_para** | **LSTM** | 32 | 99 | 32 | 0,001 |
| **GRU** | 16 | 91 | 32 | 0,001 |

### **KẾT QUẢ THỰC NGHIỆM**

#### Kết quả thực nghiệm của quá trình tối ưu hóa:

* + - Mô hình CNN:

Bảng 4. 9 Kết quả thực nghiệm quá trình tối ưu hóa Mô hình CNN

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Unit | Epoch | Batch size | Learning rate | MSE | time |
| GridSearchCV | 128 | 100 | 64 | 0,001 | 0,0026 | 184,562s |
| RandomSearhCV | 256 | 68 | 64 | 0,0001 | 0,0024 | 29,561s |
| BayesSearchCV | 64 | 68 | 16 | 0,001 | 0,0028 | 21.639s |

* + - Mô hình RNN:

Bảng 4. 10 Kết quả thực nghiệm quá trình tối ưu hóa Mô hình RNN

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Unit | Epoch | Batch size | Learning rate | MSE | time |
| GridSearchCV | 256 | 40 | 16 | 0,001 | 0,0036 | 193,267s |
| RandomSearhCV | 128 | 16 | 16 | 0,001 | 0,0032 | 69,562s |
| BayesSearchCV | 64 | 37 | 32 | 0.0010 | 0,0034 | 78,759s |

* + - Mô hình LSTM:

Bảng 4. 11 Kết quả thực nghiệm quá trình tối ưu hóa Mô hình LSTM

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Unit | Epoch | Batch size | Learning rate | MSE | time |
| GridSearchCV | 32 | 100 | 16 | 0,001 | 0,0026 | 236,826s |
| RandomSearhCV | 128 | 71 | 32 | 0,0001 | 0,0027 | 104,671s |
| BayesSearchCV | 128 | 93 | 64 | 0.0010 | 0.0033 | 84,147s |

* + - Mô hình GRU:

Bảng 4. 12 Kết quả thực nghiệm quá trình tối ưu hóa Mô hình GRU

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Unit | Epoch | Batch size | Learning rate | MSE | time |
| GridSearchCV | 256 | 100 | 16 | 0,001 | 0,0027 | 248,957s |
| RandomSearhCV | 64 | 49 | 16 | 0,001 | 0,003 | 98,135s |
| BayesSearchCV | 32 | 84 | 16 | 0,001 | 0,0027 | 114,979s |

* + - Đánh giá
      * Quá trình tối ưu hóa của 3 phương pháp có mức độ lỗi chênh lệch nhau không lớn. Tuy nhiên phương pháp GridSearchCV cho ta mức độ lỗi ổn định hơn so với phương pháp RandomSearchCV và BayesSearchCV.
      * Mặt khác, RandomSearhCV chỉ chọn ngẫu nhiên các bộ tham số trong danh sách các tham số ta đã khai báo, còn GridSearchCV chạy tất cả các trường hợp trong danh sách các tham số nên quá trình tối ưu hóa mất rất nhiều thời gian. Do vậy, nhóm đã rút bớt số lượng các phần tử của tham số để có thể thực hiện quá trình tối ưu hóa bằng GridSearchCV nhanh chóng hơn. Còn lại, phương pháp BayesSearchCV lại có thời gian tương đối với RandomSearchCV nhưng độ chính xác thấp hơn RandomSearchCV trong hầu hết các mô hình đơn.
    - Kết luận:
      * Mặc dù phương pháp GridSearchCV cho ta độ chính xác ổn định hơn 2 phương pháp còn lại nhưng việc tối ưu hóa với GridSearchCV lại mất rất nhiều thời gian và cùng với đó là ta phải giới hạn các phần tử trong danh sách tham số. Điều này, có thể dẫn dến việc quá trình huấn luyện sau khi tối ưu với GridSearchCV sẽ đưa ra độ chính xác thấp hơn so với RandomSearchCV.
      * GridSearchCV sẽ hoạt động tốt hơn khi ta đã có sẵn một bộ tham số mà ta chắc chắn rằng các bộ tham số ấy hoạt động tốt trên các mô hình và tập dữ liệu. Điều này sẽ giúp ta giảm được thời gian trong quá trình tối ưu hóa.
      * BayesSearchCV ước lượng hàm mục tiêu một cách liên tục trong không gian tham số phức tạp, giúp xác định các khu vực có tiềm năng cao để tìm tham số tốt hơn mà không cần đánh giá trực tiếp hàm mục tiêu tại mọi điểm. Vì vậy, BayesSearchCV có thể gây phức tạp hơn trong việc đánh giá và tối ưu mô hình, điều đó làm các mô hình đơn giản không đạt hiệu suất tốt nhất.
      * Vậy, phương pháp tối ưu bằng phương pháp RandomSearchCV sẽ được nhóm sử dụng trong bài toán trên. Mặc dù RandomSearchCV không ổn định như GridSearchCV nhưng phương pháp này lại có thể hoạt động tốt trên bộ tham số mà ta không chắc chắn và thời gian tối ưu cũng ngắn hơn nhiều so với phương pháp GridSearchCV.

#### Kết quả thực nghiệm và đánh giá các mô hình đơn

Bảng 4. 13 Kết quả thực nghiệm, đánh giá các mô hình đơn

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **MAE** | **MSE** | **RMSE** | **CV\_RMSE** | **train\_time (s)** | **test\_time (s)** |
| **CNN** | 0,03605262 | 0,00152514 | 0,03905316 | 0,03887611 | 19.8825 | 1.14 |
| **LSTM** | 0,01804680 | 0,00059378 | 0,02436778 | 0,02366043 | 25.9400 | 3.72 |
| **RNN** | 0,02688254 | 0,00091124 | 0,03018682 | 0,03010674 | 21.0611 | 2.18 |
| **GRU** | 0,02991669 | 0,00112480 | 0,03353810 | 0,03279277 | 45.1421 | 2.29 |

* Đánh giá: Trong trường hợp thực nghiệm này, mô hình đưa ra dự đoán tốt nhất là mô hình LSTM. Nhưng so về thời gian huấn luyện và thời gian thực nghiệm thì LSTM là mô hình có thời gian lớn hơn so với các mô hình khác.

#### Kết quả thực nghiệm và đánh giá các mô hình tuần tự

Bảng 4. 14 Kết quả thực nghiệm, đánh giá các mô hình tuần tự

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **MAE** | **MSE** | **RMSE** | **CV\_RMSE** | **train\_time (s)** | **test\_time (s)** |
| **CNN-LSTM** | 0,031002 | 0,00118475 | 0,0344202 | 0,0343941 | 27.0772 | 2.59 |
| **CNN-RNN** | 0,022864 | 0,00069624 | 0,0263864 | 0,0262038 | 43.5008 | 2.74 |
| **CNN-GRU** | 0,037254 | 0,00172414 | 0,0415227 | 0,0413765 | 27.5016 | 3.42 |
| **LSTM-CNN** | 0,030440 | 0,00113607 | 0,0337057 | 0,0336626 | 73.7551 | 2.71 |
| **LSTM-RNN** | 0,053200 | 0,00335445 | 0,0579176 | 0,0577207 | 73.7955 | 4.13 |
| **LSTM-GRU** | 0,062051 | 0,00442187 | 0,0664972 | 0,0655373 | 167.9765 | 4.77 |
| **RNN-CNN** | 0,019995 | 0,00055699 | 0,02360064 | 0,02345617 | 56.1029 | 2.46 |
| **RNN-LSTM** | 0,038956 | 0,00220303 | 0,04693651 | 0,04472833 | 22.8319 | 3.97 |
| **RNN-GRU** | 0,026218 | 0,00088946 | 0,02982394 | 0,02964667 | 37.8811 | 4.86 |
| **GRU-CNN** | 0,014913 | 0,00037531 | 0,01937303 | 0,01909228 | 88.4059 | 3.38 |
| **GRU-LSTM** | 0,026211 | 0,00099485 | 0,03154134 | 0,03142469 | 78.1468 | 5.23 |
| **GRU-RNN** | 0,028168 | 0,000970675 | 0,03115565 | 0,03111635 | 132.7020 | 4.37 |

* Đánh giá: Thông qua thực nghiệm, ta có thể thấy rằng, các mô hình tuần tự với mô hình thứ 2 là CNN thì các lỗi dự báo thấp hơn các mô hình khác. Với lợi thế, mô hình 1 trích xuất đặc trưng dữ liệu và mô hình 2 thực hiện dự đoán, thì mô CNN có thể kết hợp tốt với các mô hình khác trong việc đưa ra dự đoán, đặc biệt là mô hình GRU và RNN.

#### Kết quả thực nghiệm và đánh giá các mô hình tuần tự nhân

Bảng 4. 15 Kết quả thực nghiệm, đánh giá các mô hình tuần tự nhân

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **MAE** | **MSE** | **RMSE** | **CV\_RMSE** | **train\_time (s)** | **test\_time (s)** |
| **CNN-LSTM\_mul** | 0,01855 | 0,000469 | 0,021678 | 0,0207212 | 44.7094 | 2.93 |
| **CNN-RNN\_mul** | 0,00492 | 3,40501E-05 | 0,005835 | 0,0056395 | 48.8738 | 2.49 |
| **CNN-GRU\_mul** | 0,01174 | 0,0001952 | 0,013972 | 0,0135862 | 27.7403 | 4.24 |
| **LSTM-LSTM\_mul** | 0,02492 | 0,000789 | 0,028105 | 0,0274002 | 51.3557 | 6.13 |
| **LSTM-RNN\_mul** | 0,01015 | 0,0001476 | 0,012149 | 0,0115654 | 40.8618 | 3.76 |
| **LSTM-GRU\_mul** | 0,01202 | 0,00024031 | 0,015502 | 0,0147546 | 171.666 | 6.35 |
| **RNN-LSTM\_mul** | 0,02136 | 0,00062672 | 0,025034 | 0,0239493 | 71.2503 | 5.47 |
| **RNN-RNN\_mul** | 0,01016 | 0,000117 | 0,010833 | 0,01067 | 42.9683 | 3.45 |
| **RNN-GRU\_mul** | 0,01677 | 0,000344 | 0,018554 | 0,01825 | 24.6702 | 5.49 |
| **GRU-LSTM\_mul** | 0,03955 | 0,001698 | 0,041215 | 0,0408425 | 104.5530 | 4.46 |
| **GRU-RNN\_mul** | 0,0070 | 6,81124E-05 | 0,008253 | 0,00802 | 13.3726 | 3.64 |
| **GRU-GRU\_mul** | 0,02393 | 0,000722 | 0,026881 | 0,0262565 | 63.4023 | 6.13 |

* Đánh giá: Trong trường hợp thực nghiệm này, các mô hình có mức độ lỗi chênh lệch nhau không quá lớn. Tuy nhiên trong các mô hình trên, mô hình GRU\_RNN và mô hình CNN\_RNN đưa ra mức độ chính xác cao hơn các mô hình tuần tự nhân khác. Về thời gian thực thi, 2 trong 3 mô hình tuần tự nhân của CNN có thời gian thực thi ngắn hơn các mô hình khác và độ chính xác cũng khá cao.

#### Kết quả thực nghiệm và đánh giá các mô hình tuần tự cộng

Bảng 4. 16 Kết quả thực nghiệm, đánh giá các mô hình tuần tự cộng

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **MAE** | **MSE** | **RMSE** | **CV\_RMSE** | **train\_time (s)** | **test\_time (s)** |
| **CNN-LSTM\_add** | 0,012576 | 0,000171 | 0,013102 | 0,012948 | 19.1256 | 2.56 |
| **CNN-RNN\_add** | 0,00194 | 4,211E-06 | 0,002052 | 0,002039 | 46.7240 | 4.00 |
| **CNN-GRU\_add** | 0,004583 | 2,302E-05 | 0,004801 | 0,004732 | 30.0538 | 2.70 |
| **LSTM-LSTM\_add** | 0,030536 | 0,001067 | 0,032665 | 0,032011 | 213.920 | 7.07 |
| **LSTM-RNN\_add** | 7,42E-05 | 7,768E-09 | 8,81E-05 | 8,658E-05 | 15.9487 | 5.21 |
| **LSTM-GRU\_add** | 0,006113 | 4,081E-05 | 0,006388 | 0,006355 | 34.2513 | 4.11 |
| **RNN-LSTM\_add** | 0,003028 | 1,531E-05 | 0,003913 | 0,003781 | 56.5343 | 5.58 |
| **RNN-RNN\_add** | 0,000292 | 1,089E-07 | 0,000330 | 0,000317 | 74.1096 | 5.00 |
| **RNN-GRU\_add** | 0,003822 | 1,676E-05 | 0,004095 | 0,004015 | 11.4290 | 3.71 |
| **GRU-LSTM\_add** | 0,012397 | 0,000167 | 0,012939 | 0,012782 | 51.1670 | 6.24 |
| **GRU-RNN\_add** | 0,000682 | 5,101E-07 | 0,000714 | 0,000704 | 26.6387 | 5.32 |
| **GRU-GRU\_add** | 0,006153 | 4,214E-05 | 0,006492 | 0,006384 | 48.7546 | 4.11 |

* Đánh giá: Sau khi thực nghiệm, các mô hình tuần tự cộng đưa ra dự đoán với lỗi dự đoán tương đối thấp. Các mô hình tuần tự cộng có thể sẽ là các mô hình tốt nhất trong việc đưa ra dự đoán với thông số lỗi trên. Đặc biệt là mô hình LSTM\_RNN có mức độ lỗi nhỏ nhất trong các mô hình tuần tự cộng.

#### Kết quả thực nghiệm và đánh giá các mô hình song song

Bảng 4. 17 Kết quả thực nghiệm, đánh giá các mô hình song song

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **MAE** | **MSE** | **RMSE** | **CV\_RMSE** | **train\_time (s)** | **test\_time (s)** |
| **CNN-LSTM\_para** | 0,028716 | 0,001016 | 0,031888 | 0,031825 | 3.8685 | 2.99 |
| **CNN-RNN\_para** | 0,036052 | 0,001525 | 0,039053 | 0,038876 | 3.4099 | 2.71 |
| **CNN-GRU\_para** | 0,036052 | 0,001525 | 0,039053 | 0,038876 | 3.9334 | 4.73 |
| **LSTM-RNN\_para** | 0,019662 | 0,000554 | 0,023539 | 0,023486 | 5.7818 | 6.94 |
| **LSTM-GRU\_para** | 0,023369 | 0,000738 | 0,027174 | 0,026905 | 6.0322 | 7.22 |
| **RNN-GRU\_para** | 0,029916 | 0,001124 | 0,033538 | 0,032792 | 3.5885 | 6.59 |

* Đánh giá: Kết quả thu được cho thấy rằng, các mô hình song song đưa ra dự báo khá tốt. Tuy nhiên, trong quá trình thực nghiệm cho thấy rằng mô hình có thể đã hoạt động không tốt hoặc trong quá trình tính Omega kết quả thu được luôn là 0 hoặc 1, điều này sẽ ảnh hưởng đến kết quả cuối cùng luôn là kết quả của một mô hình đơn.

#### Kết quả thực nghiệm và đánh giá giữa mô hình đơn và các mô hình lai ghép liên quan

* Trường hợp 1: Mô hình CNN và một số mô hình liên quan

Bảng 4. 18 Kết quả thực nghiệm và đánh giá giữa mô hình CNN và các mô hình lai ghép liên quan

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **MAE** | **MSE** | **RMSE** | **CV\_RMSE** | **train\_time (s)** | **test\_time (s)** |
| **CNN** | 0,065322 | 0,007603 | 0,08719 | 0,082963 | 8.4939 | 0.95 |
| **CNN - LSTM** | 0,067282 | 0,007955 | 0,089194 | 0,085569 | 13.5507 | 3.65 |
| **LSTM - CNN** | 0,091213 | 0,013614 | 0,116679 | 0,114790 | 15.5037 | 3.43 |
| **CNN-LSTM\_add** | 0,012725 | 0,000196 | 0,014003 | 0,013468 | 11.2491 | 3.99 |
| **CNN-RNN\_add** | 0,009718 | 0,000109 | 0,010485 | 0,010043 | 13.5653 | 6.51 |
| **CNN-RNN\_mul** | 0,014384 | 0,000309 | 0,017585 | 0,017328 | 7.4770 | 3.51 |
| **CNN-GRU\_mul** | 0,029266 | 0,001359 | 0,036874 | 0,036276 | 8.7294 | 4.24 |
| **CNN-LSTM\_para** | 0,090279 | 0,013113 | 0,114512 | 0,111658 | 3.5932 | 7.26 |
| **CNN-GRU\_para** | 0,083501 | 0,011061 | 0,105175 | 0,102073 | 2.1003 | 4.08 |

* Đánh giá: Mô hình tuần tự cộng và mô hình tuần tự nhân cho kết quả tốt hơn các mô hình khác.
* Trường hợp 2: Mô hình LSTM và một số mô hình liên quan

Bảng 4. 19 Kết quả thực nghiệm và đánh giá giữa mô hình LSTM và các mô hình lai ghép liên quan

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **MAE** | **MSE** | **RMSE** | **CV\_RMSE** | **train\_time (s)** | **test\_time (s)** |
| **LSTM** | 0,018046 | 0,000593 | 0,024367 | 0,023660 | 25.9400 | 3.72 |
| **LSTM - RNN** | 0,05320 | 0,003354 | 0,057917 | 0,057720 | 73.7955 | 4.13 |
| **RNN - LSTM** | 0,038956 | 0,002203 | 0,046936 | 0,044728 | 22.8319 | 3.97 |
| **LSTM-RNN\_add** | 0,000273 | 1,074E-07 | 0,000327 | 0,000322 | 14.6875 | 4.50 |
| **RNN-LSTM\_add** | 0,000277 | 4,13E-07 | 0,000403 | 0,000397 | 16.1902 | 4.96 |
| **LSTM-RNN\_mul** | 0,007138 | 9,190E-05 | 0,009586 | 0,008544 | 19.0519 | 6.64 |
| **RNN-LSTM\_mul** | 0,006642 | 8,690E-05 | 0,009322 | 0,008258 | 27.9316 | 7.12 |
| **LSTM-RNN\_para** | 0,019662 | 0,000554 | 0,023539 | 0,023486 | 5.7818 | 6.94 |
| **LSTM-GRU\_para** | 0,023369 | 0,000738 | 0,027174 | 0,026905 | 6.0322 | 7.22 |

* Đánh giá: Tương tự như trường hợp 1, mô hình tuần tự cộng và mô hình tuần tự nhân cho kết quả tốt hơn các mô hình khác.
* Trường hợp 3: Mô hình RNN và một số mô hình liên quan:

Bảng 4. 20 Kết quả thực nghiệm và đánh giá giữa mô hình RNN và các mô hình lai ghép liên quan

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **MAE** | **MSE** | **RMSE** | **CV\_RMSE** | **train\_time (s)** | **test\_time (s)** |
| **RNN** | 0,026882 | 0,000911 | 0,030186 | 0,030106 | 21.0611 | 2.18 |
| **RNN - GRU** | 0,026218 | 0,000889 | 0,029823 | 0,029646 | 37.8811 | 4.86 |
| **GRU - RNN** | 0,028168 | 0,000970 | 0,031155 | 0,031116 | 132.7020 | 4.37 |
| **RNN-GRU\_add** | 0,000304 | 5,24E-07 | 0,000428 | 0,000324 | 31.7095 | 4.71 |
| **GRU-RNN\_add** | 0,001084 | 2,027E-06 | 0,001423 | 0,001162 | 52.0278 | 4.36 |
| **RNN-GRU\_mul** | 0,018776 | 0,000437 | 0,020927 | 0,020509 | 15.8319 | 3.21 |
| **GRU-RNN\_mul** | 0,003396 | 2,068E-05 | 0,004548 | 0,004045 | 63.1556 | 7.56 |
| **RNN-GRU\_para** | 0,029916 | 0,001124 | 0,033538 | 0,032792 | 3.5885 | 6.59 |
| **CNN-RNN\_para** | 0,036052 | 0,001525 | 0,039053 | 0,038876 | 3.4099 | 2.71 |

* Trường hợp 4: Mô hình GRU và một số mô hình liên quan:

Bảng 4. 21 Kết quả thực nghiệm và đánh giá giữa mô hình GRU và các mô hình lai ghép liên quan

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **MAE** | **MSE** | **RMSE** | **CV\_RMSE** | **train\_time (s)** | **test\_time (s)** |
| **GRU** | 0,093662 | 0,013657 | 0,116865 | 0,114134 | 11.4182 | 3.25 |
| **GRU - CNN** | 0,084710 | 0,012199 | 0,110450 | 0,106974 | 17.1469 | 3.50 |
| **CNN - GRU** | 0,063838 | 0,007383 | 0,085929 | 0,082163 | 11.0661 | 6.70 |
| **GRU-LSTM\_add** | 0,010148 | 0,000145 | 0,012071 | 0,011304 | 21.2914 | 6.41 |
| **LSTM-GRU\_add** | 0,007649 | 9,513E-05 | 0,009753 | 0,009359 | 15.6187 | 9.49 |
| **GRU-LSTM\_mul** | 0,073715 | 0,008452 | 0,091937 | 0,08974 | 16.6835 | 9.46 |
| **LSTM-GRU\_mul** | 0,043110 | 0,002942 | 0,054240 | 0,053823 | 16.8192 | 6.35 |
| **CNN-GRU\_para** | 0,083501 | 0,011061 | 0,105175 | 0,102073 | 2.1003 | 4.08 |
| **LSTM-GRU\_para** | 0,093771 | 0,01372 | 0,117163 | 0,114263 | 5.5389 | 9.54 |

* Đánh giá chung: Trong cả 4 trường hợp thực nghiệm trên, mô hình tuần tự cộng và tuần tự nhân luôn cho kết quả tốt nhất so các mô hình đơn, mô hình tuần tự và mô hình song song. Đặc biệt là mô hình tuần tự cộng, trong cả 4 trường hợp trên luôn cho kết quả với mức độ lỗi thấp nhất.

#### Đánh giá chung và đề xuất mô hình

* Thông qua các trường nghiệm trên, ta có thể thấy rằng, việc xây dựng các mô hình lai ghép đã giúp cho việc dự đoán có thể chính xác hơn so với các mô hình đơn. Tuy nhiên, phần lớn các mô hình lai ghép có thời gian huấn luyện cao hơn so với các mô học đơn lẻ.
* Mô hình tuần tự với việc sử dụng mô hình 1 làm mô hình trích xuất đặc trưng của tập dữ liệu và mô hình 2 thực hiện dự đoán sau khi phân tích đặc trưng ấy. Tuy nhiên, mức độ cải thiện của mô hình tuần tự không tốt hơn nhiều so với các mô hình đơn.
* Mô hình song song đưa ra dự đoán với độ chính xác không ổn định, trong quá trình thực nghiệm, đôi khi việc tính Omega đưa ra kết quả là 0 hoặc 1, điều này sẽ dẫn đến kết quả cuối cùng là kết quả của mô hình đơn.
* Mô hình tuần tự cộng và mô hình tuần tự nhân, bằng cách sử dụng mô hình 1 đưa ra dự đoán và mô hình 2 cải thiện lỗi dự đoán. Việc xây dựng như trên đã giúp cho mô hình đơn cải thiện và giảm thiểu phần lớn lỗi dự báo trong quá trình thực nghiệm. Người dùng có thể sử dụng mô hình tuần tự cộng và tuần tự nhân trong việc dự báo chuỗi thời gian. Đặc biệt, **mô hình tuần tự cộng LSTM\_RNN** cho kết quả tốt nhất trong tất cả các mô hình và trên các tập dữ liệu thực nghiệm.

# **PHẦN 3: KẾT LUẬN**

## **Kết quả đạt được**

* Xây dựng thành công các mô hình đơn và các mô hình lai ghép.
* Bằng các phương pháp lai ghép, nhóm đã tăng độ chính xác của các mô hình trong việc dự báo chuỗi thời gian. Đánh giá mức độ chính xác giữa các mô hình lai ghép và mô hình đơn, đồng thời đánh giá giữa các mô hình lai ghép với nhau.
* Đánh giá kết quả quá trình xây dựng các mô hình lai ghép và đề xuất người dùng sử dụng mô hình lai ghép có độ chính xác tốt nhất.

## **Hạn chế khi thực hiện đề tài**

* Tập dữ liệu chưa đủ lớn để có thể khách quan hóa kết quả của các mô hình lai ghép trong quá trình thực nghiệm.
* Nhóm đã cố gắng hết sức để có thể đưa ra các hướng xây dựng, tuy nhiên, thời gian có hạn nên nhóm vẫn chưa thể hiện thực hóa đề tài.
* Trong quá trình xây dựng các mô hình tuần tự cộng và tuần tự nhân, mô hình CNN không thể nhận được shape từ mô hình đầu, mặc dù nhóm đã cố gắng tìm hiểu và khắc phục.

## **Hướng phát triển đề tài**

* Xây dựng một trang web hoặc một ứng dụng mà người dùng có thể trực tiếp chạy và thực thi các mô hình để đưa ra dự đoán trên luồng dữ liệu (streaming data).
* Tìm các nguồn dữ liệu lớn, dữ liệu có các yếu tố bất thường trong tập để có thể cải thiện hơn trong quá trình huấn luyện, nhận biết và xử lý lỗi của các mô hình và đưa ra kết quả chính xác nhất.

# **Tài liệu tham khảo**

[1] - Thủy Nhi Lê Nguyễn, “Chuỗi thời gian là gì? What is time series?”, 21/12/1022, <https://viblo.asia/p/chuoi-thoi-gian-la-gi-what-is-time-series-pgjLNd6w432>

[2] - SOM AIT, “Machine Learning là gì? Khái niệm, phân loại và ứng dụng của Machine Learning”, 22/11/2022, <https://som.edu.vn/machine-learning-la-gi-phan-loai-quy-trinh-ung-dung/>

[3] – Amazon, “Học tăng cường là gì?”, <https://aws.amazon.com/vi/what-is/reinforcement-learning/>

[4] – Hưng Nguyễn, “Deep Learning là gì? Tổng quan về Deep Learning từ A-Z”, 21/4/2022, <https://vietnix.vn/deep-learning-la-gi/>

[5] - LamTien, “MSE và RMSE là gì và cách tính trên STATA”, 29/7/2020, <https://solieu.vip/mse-va-rmse-la-gi-va-cach-tinh-tren-stata/#Cong_thuc_tinh_MSE>

[6] - Nguyễn Ngọc Bình, “Đánh giá mô hình hồi quy, machine learning (performance and predict)”, 12/6/2018, <https://rpubs.com/nguyenngocbinhneu/performance_predict>

[7] – Topdev, “Thuật toán CNN là gì? Cấu trúc mạng Convolutional Neural Network”, <https://topdev.vn/blog/thuat-toan-cnn-convolutional-neural-network/>

[8] - Do Duong, “Recurrent Neural Network(Phần 1): Tổng quan và ứng dụng”, 30/7/2018, <https://viblo.asia/p/recurrent-neural-networkphan-1-tong-quan-va-ung-dung-jvElaB4m5kw>

[9] – Đỗ Minh Hải, “[RNN] LSTM là gì?”, 20/10/2017, <https://dominhhai.github.io/vi/2017/10/what-is-lstm/>

[10] – StatQuest, “Long Short-Term Memory (LSTM), Clearly Explained”, 7/11/2022, <https://www.youtube.com/watch?v=YCzL96nL7j0>

[11] – Vũ Hữu Tiệp, “Bài 35: Lược sử Deep Learning”, 22/9/2018, <https://machinelearningcoban.com/2018/06/22/deeplearning/>

[12] – Eryk Lewinson, “ Convolutional neural networks for time series forecasting”, <https://subscription.packtpub.com/book/data/9781789618518/10/ch10lvl1sec63/convolutional-neural-networks-for-time-series-forecasting>

[13] – Chung Pham Van, “Kỹ thuật Dropout (Bỏ học) trong Deep Learning”, 23/5/2021, <https://viblo.asia/p/ky-thuat-dropout-bo-hoc-trong-deep-learning-XL6lAd8BZek>

[14] - Nguyen Thanh Huyen, “Recurrent Neural Network: Từ RNN đến LSTM”, 24/6/2021, <https://viblo.asia/p/recurrent-neural-network-tu-rnn-den-lstm-gGJ597z1ZX2>

[15] – “Advantages and Disadvantages of Convolutional Neural Network”, <https://aspiringyouths.com/advantages-disadvantages/convolutional-neural-network/#google_vignette>

[16] – Zoumana Keita, “An Introduction to Convolutional Neural Networks (CNNs)”, <https://www.datacamp.com/tutorial/introduction-to-convolutional-neural-networks-cnns>

[17] – Debasish Kalita, “An Overview on Long Short Term Memory (LSTM)”, 24 Mar, 2022, <https://www.analyticsvidhya.com/blog/2022/03/an-overview-on-long-short-term-memory-lstm/>

[18] – Kaushik Das, “HOW RECURRENT NEURAL NETWORK (RNN) WORKS”, <https://dataaspirant.com/how-recurrent-neural-network-rnn-works/>

[19] – Debasish Kalita,“ What is Recurrent Neural Networks (RNN)?”, 29 May, 2024, <https://www.analyticsvidhya.com/blog/2022/03/a-brief-overview-of-recurrent-neural-networks-rnn/>

[20] – Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y., “Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation”, 3 Sep 2014, <https://arxiv.org/abs/1406.1078>

[21] – Junyoung Chung, “Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling”, 11 Dec 2014, <https://arxiv.org/abs/1412.3555>

[22] – Shipra Saxena,“Introduction to Gated Recurrent Unit (GRU)”, 02 Jun, 2024, <https://www.analyticsvidhya.com/blog/2021/03/introduction-to-gated-recurrent-unit-gru/>

[23] – Chung Pham Van, “GRU - Mạng Neural hồi tiếp với nút có cổng”, 05/08/2021, <https://viblo.asia/p/gru-mang-neural-hoi-tiep-voi-nut-co-cong-3P0lPGevZox>

[24] – Fachrizal Aksan và Yang Li và Vishnu Suresh và Przemysław Janik , “CNN-LSTM vs. LSTM-CNN to Predict Power Flow Direction: A Case Study of the High-Voltage Subnet of Northeast Germany”, 12/1/2023, <https://www.mdpi.com/1424-8220/23/2/901>

[25] – Trần Văn Duy và Trần Công Trường, “Ứng Dụng Dự Báo Giá Chứng Khoán Bằng Mô Hình Lai Ghép Giữa Bộ Phân Lớp Đa Thức Và Mô Hình Học Sâu”, 5/2023.

[26] – “Combining k-nearest neighbors-based method and neural network for time series prediction”, Nguyen Thanh Son và Duong Tuan Anh