TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

ĐẠI HỌC TÔN ĐỨC THẮNG

Công trình nghiên cứu khoa học sinh viên năm 2020-2021

Khảo sát và đánh giá hoạt động của hệ thống nguồn điện năng lượng mặt trời (PVS) nối lưới dùng các giải thuật Machine learning

GIẢNG VIÊN HƯỚNG DẪN:

Tiến Sĩ Đinh Hoàng Bách

Nhóm SV THỰC HIỆN:

Trần Gia Chí Bảo 41701055

Huỳnh Công Ân 41701041

Ninh Thế Vĩnh Cường 418H0363

Thành Phố Hồ Chí Minh, ngày 5 tháng 4 năm 2021.

Mục Lục

Danh mục hình ảnh:

Danh mục Bảng Biểu

LỜI MỞ ĐẦU

Lời nói đầu tiên trước khi bước vào đề tài, em xin phép cảm ơn Thầy.TS Đinh Hoàng Bách. Người đã hướng dẫn nhóm em trong suốt quá trình học tập và nghiên cứu khoa học tại Trường Đại Học Tôn Đức Thắng. Là người Thầy tận tâm luôn giúp đỡ, chỉ dẫn từng bước thực hiện trong suốt sáu tháng thực hiện nghiên cứu khoa học. Xin cảm ơn EVNHCMC đã tạo điều kiện cho nhóm em thực hiện các dự án liên quan đến Machine learning cũng như vấn đề thu thập dữ liệu được diễn ra suôn sẻ hơn. Và có lẽ nơi giúp đỡ cả nhóm và tạo điều kiện cho phép nghiên cứu đạt hiệu quả tốt nhất đó chính là Khoa Điện- Điện tử Trường Đại Học Tôn Tôn Đức Thắng, không còn lời nào hoa mỹ hơn là xin chân thành cảm ơn sự cống hiến của các thầy- cô đã giúp đỡ chúng em trong thời gian vừa qua để hoàn thành tốt nhiệm vụ được giao.

Tóm tắt đề tài:

Photovoltaic (PV) là một trong những nguồn năng lượng tái tạo được kì vọng sẽ giúp cho trái đất có sự sống bền vững nhất. Để đảm bảo hoạt động an toàn và có lợi về mặt kinh tế của hệ thống PV trong lưới điện thông minh, việc dự đoán chính xác sản lượng điện sản sinh là điều cực kì quan trọng và đáng được quan tâm. Trong đề tài này, chúng ta đề xuất việc sử dụng Long Short-Term Memory Recurrent Neural Network (LSTM-RNN) để dự đoán chính xác hơn về sản lượng của hệ thống PV. Mạng LSTM có thể thiết lập mô hình thay đổi theo thời gian trong sản lượng sinh ra của PV điều đó thực hiện được đều nhờ vào cấu trúc lặp lại và các nút bộ nhớ của chúng. Phương pháp đề xuất được đánh giá mà cách đánh giá đó sử dụng tập dữ liệu theo giờ của cùng một địa điểm trong một tháng. Ta so sánh phương pháp đề xuất với hai phương pháp dự đoán khác. Sử dụng LSTM cho ta kết quả lỗi dự đoán giảm hơn khi so sánh với các phương pháp khác. Mục đích của phương pháp dự đoán này có thể là một công cụ hữu ích cho việc lên kế hoạch vận hành và điều khiển hệ thống điện trong tương lai.

Lý do lựa chọn đề tài:

Hiện nay, do nhu cầu sử dụng điện vào mùa nắng ở địa bàn TP.HCM tăng cao. Nên sản lượng điện sinh ra từ các nhà máy điện vào giờ cao điểm rất lớn để đảm bảo đáp ứng đủ nhu cầu sử dụng điện của khách hàng trong khu vực. Chính phủ đã ban hành nghị định số 13/2020/QĐ-ttg về cơ chế khuyến khích phát triển điện mặt trời tại việt nam nói chung và TP.HCM nói riêng. Sau hơn một năm rưỡi (từ 1/07/2019 đến 31/12/2020), trong khoảng thời gian đó có nảy sinh một số vấn đề liên quan đến việc vận hành các trạm điện. Người dân ồ ạt lắp điện mặt trời áp mái, công suất điện mặt trời tính đến 21/08/2020 có đến 9267 công trình lắp điện mặt trời với công suất lắp là 127 triệu kWp.

Với sản lượng điện mặt trời lớn như vậy sẽ sinh ra một vài vấn đề về việc vận hành lưới điện vào các khung giờ khác nhau trong ngày. Nguyên nhân thứ nhất là quá tải cục bộ hệ thống điện vào các giờ nắng cao điểm tại một vài nơi trên địa bàn TP.HCM. Nguyên nhân thứ hai là khó khăn trong việc liên hệ các công trình cắt giảm công suất PV vào các giờ thấp điểm của phụ tải. Đối với mùa nắng nóng kéo dài, sản lượng PV sinh ra lớn cùng thời điểm với việc phụ tải thấp sẽ gây nguy hại cho lưới điện.

Để giải quyết vấn đề đó, machine learning chính là chìa khóa. Vậy áp dụng như thế nào, và mục tiêu của việc áp dụng đó là gì?

Machine learning là một công cụ giúp cho việc tính toán trở nên hiệu quả, đồng thời nó cũng giúp việc dự báo sản lượng điện mặt trời sinh ra tại các thời điểm trong ngày một cách chính xác và hiệu quả dựa vào dữ liệu quá khứ mà ta đã thu thập được.
Hiện tại Trường Đại Học Tôn Đức Thắng đã lắp đặt và đi vào vận hành hệ thống điện mặt trời có công suất tổng là 500kWp, vì thế có thể xem trường là một trạm phát PV vào các giờ thấp điểm của phụ tải và những ngày lễ.
Phương pháp thực hiện đề tài sẽ được tiếp tục nói đến ở các phần sau.

Mục đích nghiên cứu:

Mục tiêu của đề tài nghiên cứu là đề xuất các phương pháp dự báo công suất phát PV tại từng thời điểm trong ngày. Ước lượng trước khả năng phát điện của trạm PV từ đó đưa ra các giả thuyết về những trường hợp có thể xảy ra trong tương lai đối với lướng điện truyền tải và phân phối. Phương pháp dự báo sử dụng mạng Neural nhân tạo kết hợp với mô hình chuỗi thời gian, các tính năng làm việc của mô hình được thiết lập trong không gian ngôn ngữ lập trình python 3.8.

Khảo sát đồ thị phát điện của các hệ thống Pin Mặt Trời (Photo-voltaic sources- PVS) nối lưới khu vực Tp.HCM. Ảnh hưởng yếu tố thời tiết đến khả năng phát điện của PVS lên lưới.

Dự báo khả năng phát điện của PVS trong các thời điểm ngắn hạn (tương lai).

Đánh giá được nhu cầu sử dụng điện của hệ thống lưới điện theo thời gian, cảnh báo khả năng quá tải cục bộ của một số tuyến dây.

Xây Dựng đồ thị phát của PSV trong ngắn hạn.

Đối tượng và phạm vi nghiên cứu:

- Đối tượng nghiên cứu đề tài

Các mô hình và phương pháp dự báo.

Các yếu tố ảnh hưởng đến phụ tải điện (ngắn hạn)

Phương pháp dự báo công suất phát điện dựa trên các kỹ thuật của mạng neural nhân tạo.

Nghiên cứu đồ thị phát điện của hệ thống PV tại trường đại học Tôn Đức Thắng, xây dựng mô hình dự báo cho hệ thống.

Sử dụng machine learning khảo sát hoạt động phát điện của hệ thống PV

- Phạm vi nghiên cứu

Ảnh hưởng của hệ thống PV đối với lưới phân phối.

Dự đoán khả năng phát điện tại một số thời điểm ngắn hạn.

Xây dựng đồ thị phát của PSV.

CHƯƠNG I: TỔNG QUAN VỀ PHƯƠNG PHÁP DỰ BÁO CÔNG SUẤT PHÁT PV

1.1 Giới thiệu:

Việc áp dụng cộng nghệ AI vào lĩnh vực điện đã có từ lâu ở các nước khác, do điều kiện kinh tế và sự phụ thuộc công nghệ quá nhiều từ nước ngoài. Từ năm 2014 cho đến những năm gần đây Việt Nam chính thức bước chân vào sàn đấu AI của quốc tế với những cái cái tên lớn như VinAI, Viettel AI open platform…

Hiện nay vấn đề này còn khá mới mẻ ở Việt Nam, cơ sở dữ liệu, hạ tầng kỹ thuật còn khá sơ xài không đủ để đáp ứng nhu cầu thu thập dữ liệu cho ngành khoa học dữ liệu ở Việt nam. Đối với AI thì dữ liệu được xem phần thiết yếu trong các mô hình máy học. Việc thiếu hụt nguồn thông tin, dữ liệu mang đến nhiều thách thức cho ngành AI. Bên cạnh sự quan trọng của dữ liệu thì việc đào tạo về mảng khoa học dữ liệu cũng hết sức khó khắn và tốn kém. Thông thường mất từ 1 – 2 năm cho việc làm quen về kỹ thuật lập trình và xử lý các dữ liệu số.

Công cụ chính trong đề tài này là machine learning, áp dụng những kỹ thuật lập trình, toán học, kiến thức về điện, …. Ngôn ngữ chính sử dụng để lập trình là PYTHON 3.8 được chạy trên ứng dụng Pycharm. Những vấn đề liên quan đến machine learning có thể nói đến là phân loại và dự đoán phụ tải, dự báo khả năng phát PV, dự báo thời tiết, chứng khoáng, …

Trong bài viết này chúng ta sẽ tìm hiểu về cách sử dụng mạng Long Short-term Memory Recurrent Neural Netwwork (LSTM-RNN) để dự đoán cho công suất phát của hệ thống PV theo từng khoảng thời gian trong ngày, từ đó đưa các khuyến cáo về việc giảm bớt công suất phát của các trạm cho ngày hôm sau.

1.2 Các loại dự báo công suất

Liên quan đến kiến thức về chuối thời gian, quy mô dự báo thường được định nghĩa như các loại sau [2]:

- Dự đoán trung hạn và dài hạn được chấp nhận cho nhà máy phát điện PV và các thông số về thời gian được chọn là từ tuần đến năm.

- Dự báo ngắn hạn được biết là việc chọn khoảng thời gian từ một vài giờ đến một tuần. Quy mô của việc dự đoán này là tập trung vào sản lượng của hệ thống PV lập kế hoạch và điều độ lưới điện một cách kinh tế.
- Dự báo cực ngắn hạn công suất PV là việc dự báo công suất từ vài phút đến vài giờ.

Quy mô việc dự đoán này được áp dụng cho thị trường cung cấp điện. nó cũng được sử dụng cho việc đối phó với vấn đề khẩn cấp ở thời điểm thực tế và giám sát tình hình hệ thống PV. Với sự cải thiện công nghệ tính toán và độ phủ sóng 5 G hiện nay thì việc con người tham gia vào việc vận hành của con người trong hệ thống điện cũng được cải thiện.

Giữa cả hai bên sử dụng năng lượng và cung cấp đều hy vọng độ chính xác của dữ liệu thực và sự cải thiện trong tương lai. Nhu cầu này thúc đẩy sự phát triển của công nghệ dự báo ngắn hạn PV, và có thể đảm bảo hệ thống PV hoạt động đáng tin cậy và hiều quả hơn.

CHƯƠNG II: SƠ LƯỢC VỀ MACHINE LEARNING

2.1 Machine learning là gì

Theo như [1] thì Machine Learning là một tập con của AI. Theo định nghĩa của Wikipedia, *Machine learning is the subfield of computer science that "gives computers the ability to learn without being explicitly programmed"*. Nói đơn giản, Machine Learning là một lĩnh vực nhỏ của Khoa Học Máy Tính, nó có khả năng tự học hỏi dựa trên dữ liệu đưa vào mà không cần phải được lập trình cụ thể.

Những năm gần đây, khi mà khả năng tính toán của các máy tính được nâng lên một tầm cao mới và lượng dữ liệu khổng lồ được thu thập bởi các hãng công nghệ lớn, Machine Learning đã tiến thêm một bước dài và một lĩnh vực mới được ra đời gọi là Deep Learning

Deep Learning đã giúp máy tính thực thi những việc tưởng chừng như không thể vào 10 năm trước: phân loại cả ngàn vật thể khác nhau trong các bức ảnh, tự tạo chú thích cho ảnh, bắt chước giọng nói và chữ viết của con người, giao tiếp với con người, hay thậm chí cả sáng tác văn hay âm nhạc

2.2 Feature vector

Theo như tác giả của [1] thì mỗi điểm dữ liệu trong các bài toán machine learning thường được biểu diễn bằng một vector được gọi là vector đặc trưng (feature vector). Hơn nữa, trong cùng một bài toán, các feature vector của tất cả các điểm thường có kích thước như nhau. Điều này là cần thiết vì các phép toán trong mô hình (cộng, nhân ma trận, vector) yêu cầu đầu vào có cùng kích thước. Khi đó, toàn bộ dữ liệu có thể được lưu trong một ma trận mà mỗi hàng hoặc mỗi cột là feature vector của một điểm dữ liệu. Tuy nhiên, trên thực tế, dữ liệu thường ở dạng thô (raw data) với kích thước khác nhau. Hoặc thậm chí khi kích thước của các điểm là như nhau, việc lựa chọn, tính toán đặc trưng nào phù hợp cho mỗi bài toán là nhiệm vụ quan trọng trước tiên cần được giải quyết.

2.3 Mô hình chung cho các bài toán machien learning

Phần lớn các mô hình machine learning có thể được minh họa trong Hình 2.2. Có hai bước quan trọng (phase) lớn trong mỗi bài toán machine learning là bước huấn luyện

(trainning phase) và bước kiểm thử (test phase). Bước huấn luyện sẽ chỉ dùng dữ liệu huấn luyện, bước kiểm thử sẽ chỉ dùng dữ liệu trong tập kiểm thử

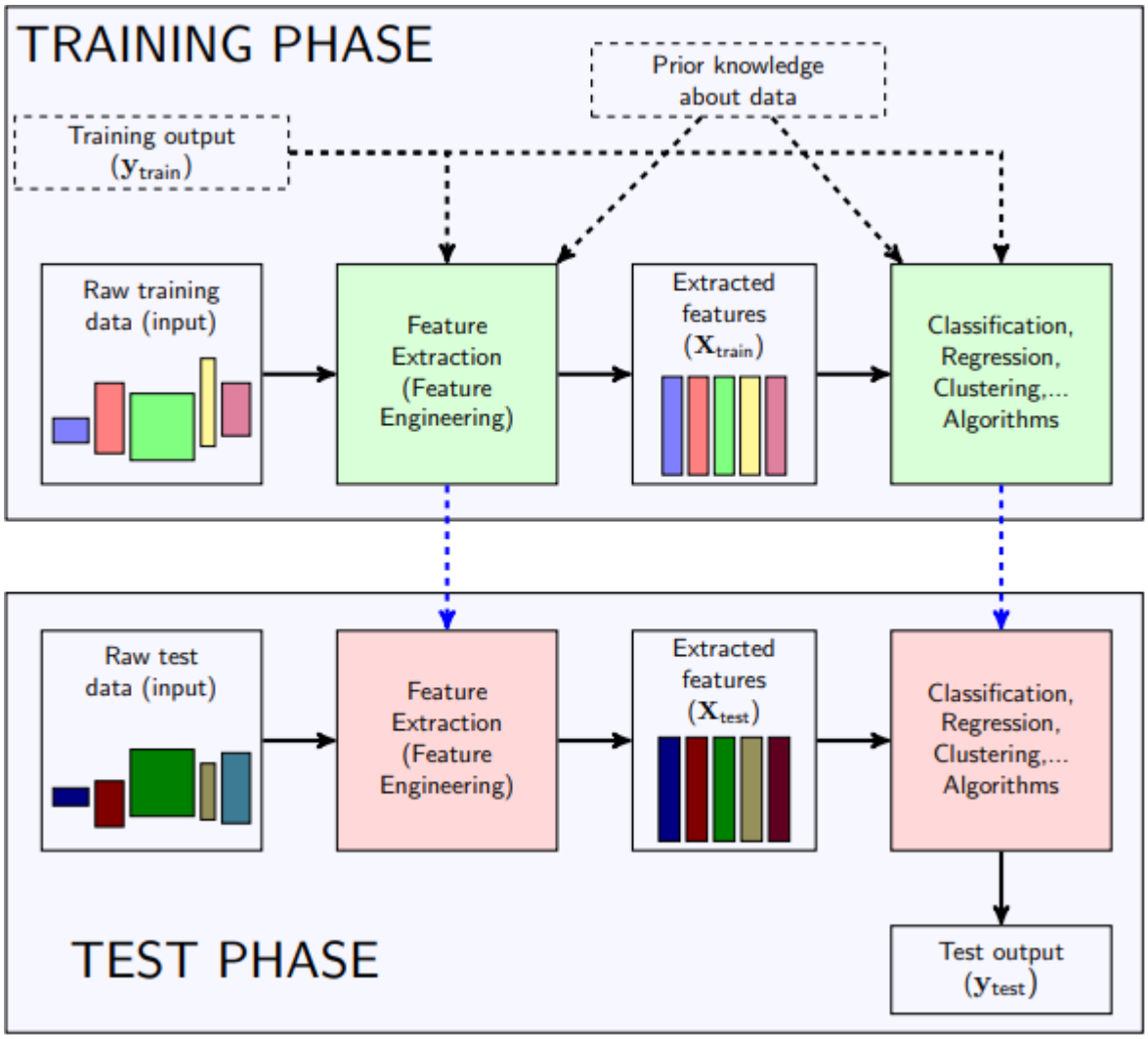


Figure 2.3.1: Mô hình chung cho các bài toán machien learning

2.3.1 Training phase

Có hai khối có nền màu lục cần được thiết kế: Khối thứ nhất, Feature Extraction, có nhiệm vụ tạo ra một vector đặc trưng cho mỗi điểm dữ liệu đầu vào. Vector đặc trưng này thường có kích thước như nhau, bất kể dữ liệu đầu vào có kích thước như thế nào. Đầu vào của khối Feature Extraction có thể là các yếu tố sau:

- Dữ liệu thô ban đầu (raw training input). Dữ liệu thô bao gồm tất cả các thông tin ta biết về dữ liệu. Ví dụ: dữ liệu thô của một ảnh là giá trị của từng pixel; của một văn bản là từng từ, từng câu; của một file âm thanh là một đoạn tín hiệu; với bài toán dự báo thời tiết, dữ liệu thô là thông tin về hướng gió, nhiệt độ, độ ẩm, v.v… Dữ liệu thô này thường không ở dạng vector và không có số chiều như nhau. Thậm chí có thể có số chiều như nhau nhưng số chiều quá lớn,

chẳng hạn một bức ảnh màu 1000 × 1000 pixel sẽ có số pixel là đã là 3 × 106 (ảnh màu thường có ba channel: red, green, blue–RGB). Đây là một con số quá lớn, không lợi cho lưu trữ và tính toán.

- output của training set. Trong các bài toán unsupervised learning, ta không biết output nên hiển nhiên sẽ không có giá trị này. Trong các bài toán supervised learning, có khi dữ liệu này cũng không được sử dụng. Ví dụ, nếu raw input đã có cùng số chiều rồi nhưng số chiều quá lớn, ta muốn giảm số chiều của nó thì cách đơn giản nhất là chiếu vector đó xuống một không gian có số chiều nhỏ hơn bằng cách lấy một ma trận ngẫu nhiên nhân với nó vào bên trái. Ma trận này thường là ma trận béo, tức có số hàng ít hơn số cột, để đảm bảo số chiều thu được nhỏ hơn số chiều ban đầu. Việc làm này mặc dù làm mất đi thông tin, trong nhiều trường hợp vẫn mang lại hiệu quả vì đã giảm được lượng tính toán ở phần sau. Đôi khi ma trận chiếu không phải là ngẫu nhiên mà có thể được học dựa trên toàn bộ dữ liệu thô ban đầu. Trong nhiều trường hợp khác, dữ liệu output của tập huấn luyện cũng được sử dụng để tạo ra bộ trích chọn đặc trưng. Trong bài toán classification, việc giữ lại nhiều thông tin không quan trọng bằng việc giữ lại các thông tin có ích cho bài toán. Ví dụ, giả sử dữ liệu thô là các hình vuông và hình tam giác có màu đỏ và xanh. Trong bài toán phân loại đa giác, nếu các nhãn là tam giác và vuông, ta không quan tâm tới màu sắc mà chỉ quan tâm tới số cạnh của đa giác. Ngược lại, trong bài toán phân loại màu, các nhãn là xanh và đỏ, ta không quan tâm tới số cạnh mà chỉ quan tâm đến màu sắc.
- Prior knowledge about data: Các thông tin khác đã biết về loại dữ liệu (ngoài những thông tin về raw input và output).

Sau khi các tham số mô hình của bộ feature extraction được thiết kế, dữ liệu thô ban đầu được đưa qua và tạo ra các vector đặc trưng tương ứng được gọi là extracted feature. Những extracted feature này sẽ được đưa vào huấn luyện các thuật toán chính như classification, regression, clustering, v.v. trong khối màu lục phía sau.

2.3.2 Testing phase

Khi có dữ liệu thô mới, ta sử dụng bộ trích chọn đặc trưng đã tìm được ở trên để tạo ra vector đặc trưng ứng với dữ liệu thô đó. Vector đặc trưng này được đưa vào thuật toán chính đã tìm được để đưa ra quyết định.

2.4 Overfitting

Overfitting là một hiện tượng không mong muốn thường gặp, người xây dựng mô hình machine learning cần nắm được các kỹ thuật để tránh hiện tượng này.

2.4.1 Giới thiệu

Cũng như các phần trước đó tác giả của [1] đã miêu tả trong các bài toán supervised learning, chúng ta thường phải đi tìm một mô hình ánh xạ các vector đặc trưng thành các kết quả tương ứng trong training set. Tức là đi tìm hàm số f sao cho $y_i \approx f(x_i),\ \forall i = 1, 2, \ldots, N$. Một cách tự nhiên, ta sẽ đi tìm các tham số mô hình của f sao cho việc xấp xỉ có sai số càng nhỏ càng tốt. Nói cách khác, mô hình càng khớp (fit) với dữ liệu càng tốt. Tuy nhiên, sự thật là nếu một mô hình quá fit với dữ liệu thì nó sẽ gây phản tác dụng. Hiện tượng quá fit này trong machine learning được gọi là overfitting. Đây là một hiện tượng xấu cần tránh. Vì có thể mô hình rất fit với training set nhưng lại không biểu diễn tốt dữ liệu không được nhìn thấy khi huấn luyện. Một mô hình chỉ mô tả tốt training set là mô hình không có tính tổng quát (generalization). Một mô hình tốt là mô hình có tính tổng quát.

Để có cái nhìn tổng quát về overfitting, chúng ta chuẩn bị 50 điểm dữ liệu. Đầu ra bằng đa thức bậc ba của đầu vào cộng thêm nhiễu. Tập dữ liệu được chia làm hai phần: 35 dữ liệu màu đỏ là tập training, 15 điểm dữ liệu màu vàng là dữ liệu kiểm thử. Đồ thị của đa thức bậc ba này được cho bởi đường nét đứt màu xanh lục. Bài toán đặt ra là giả sử ta không biết mô hình ban đầu mà chỉ biết các điểm dữ liệu, hãy tìm một mô hình tốt để mô tả quan hệ giữa đầu vào và đầu ra của dữ liệu đã cho. Giả sử biết thêm rằng mô hình được mô tả bởi một đa thức.

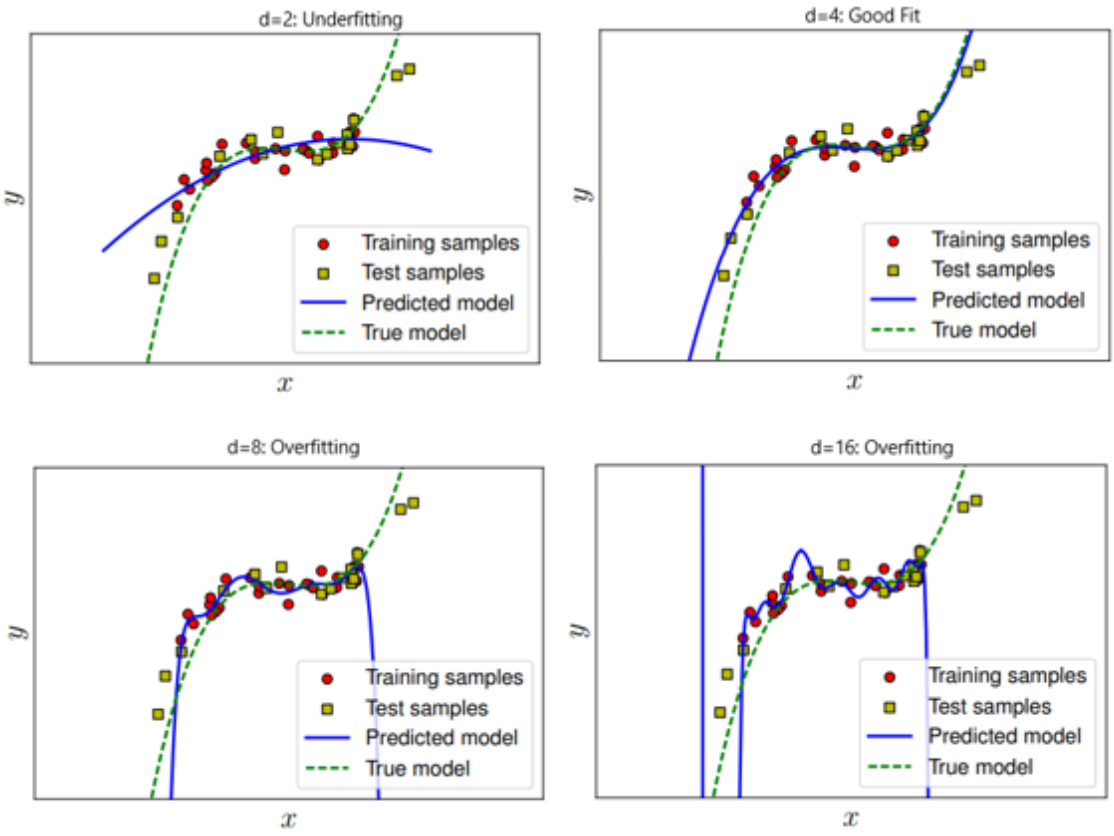


Figure 2.4.1: Underfitting và Overfitting với polynomial regression

Rõ ràng là một đa thức bậc không vượt quá 29 có thể fit được hoàn toàn với 30 điểm trong tập training set. Xét một vài giá trị d = 2, 4, 8, 16. Với d = 2, mô hình không thực sự tốt vì mô hình dự đoán (predicted model) quá khác so với mô hình thực (true model); thậm chí nó có xu hướng đi xuống khi mà dữ liệu đang có hướng đi lên. Trong trường hợp này, ta nói mô hình bị underfitting.

Với d= 8, với các điểm dữ liệu trong training set, mô hình dự đoán và mô hình thực là khá giống nhau. Tuy nhiên, về phía phải, đa thức bậc 8 cho kết quả hoàn toàn ngược với xu hướng của dữ liệu

Với d= 16 thì đa thức bậc 16 này quá fit training set. Việc quá fit trong trường hợp bậc 16 là không tốt vì mô hình có thể đang cố gắng mô tả nhiễu hơn là dữ liệu. Hiện tượng xảy ra ở hai trường hợp đa thức bậc cao này được gọi là overfitting. Độ phức tạp của đồ thị trong hai trường hợp này cũng khá lớn, dẫn đến các đường dự đoán không được tự nhiên.

Với d= 4, mô hình dự đoán khá giống với mô hình thực. Hệ số bậc cao nhất tìm được rất gần với 01, vì vậy đa thưc bậc bốn này khá gần với đa thức bậc ba ban đầu. Đây chính là một mô hình tốt.

Overfitting là hiện tượng mô hình tìm được quá khớp với dữ liệu huấn luyện. Việc này sẽ gây ra hậu quả lớn nếu trong training set có nhiễu. Khi đó, mô hình quá chú trọng vào việc xấp xỉ training set mà quên đi việc quan trọng hơn là tính tổng quát, khiến cho mô hình không thực sự mô tả tốt dữ liệu ngoài training set. Overfitting đặc biệt xảy ra khi lượng dữ liệu huấn luyện quá nhỏ hoặc độ phức tạp của mô hình quá cao. Trong ví dụ trên đây, độ phức tạp của mô hình có thể được coi là bậc của đa thức cần tìm.

2.4.2 Đại lượng đánh giá chất lượng của mô hình

2.4.2.1 Training Error

Đại lượng này là mức độ sai khác giữa đầu ra thực và đầu ra dự đoán của mô hình, thường là giá trị của hàm mất mát áp dụng lên training data. Hàm mất mát này cần có một thừa số $\frac{1}{N_{train}}$ để tính giá trị trung bình, tức mất mát trung bình trên mỗi điểm dữ liệu. Với các bài toán regression, đại lượng này thường được xác định bởi mean squared error (MSE)

$$training\_error = \frac{1}{2N_{train}} \sum_{training\_set} \| y - \hat{y} \|_2^2$$

2.4.2.2 Test Error

Tương tự như trên, nhưng mô hình tìm được được áp dụng vào test data. Chú ý rằng, khi xây dựng mô hình, ta không được sử dụng thông tin trong tập dữ liệu này. Với regression, đại lượng này thường được định nghĩa bởi

$$test\_error = \frac{1}{2N_{test}} \sum_{test\_set} \| y - \hat{y} \|_2^2$$

Việc lấy trung bình là quan trọng vì lượng dữ liệu trong tập huấn luyện và tập kiểm thử có thể chênh lệch rất nhiều.

Một mô hình được coi là tốt (fit) nếu cả training error và test error đều thấp. Nếu training error thấp nhưng test error cao, ta nói mô hình bị overfitting. Nếu training error cao và test error cao, ta nói mô hình bị underfitting. Xác suất để xảy ra việc training error cao nhưng test error thấp là rất nhỏ.

2.5 Hàm mất mát

Mỗi mô hình machine learning được mô tả bởi các tham số mô hình (model parameters). Công việc của một thuật toán machine learning là đi tìm các tham số mô hình phù hợp với mỗi bài toán. Việc đi tìm các tham số mô hình có liên quan mật thiết đến các phép đánh giá. Mục đích của chúng ta là đi tìm các tham số mô hình sao cho các phép đánh giá cho kết quả tốt nhất. Trong bài toán classification, kết quả tốt có thể được hiểu là ít điểm dữ liệu bị phân lớp sai nhất. Trong bài toán regression, kết quả tốt là khi sự sai lệch giữa đầu ra dự đoán và đầu ra thực sự là ít nhất.

Quan hệ giữa một phép đánh giá và các tham số mô hình thường được mô tả thông qua một hàm số được gọi là hàm mất mát (loss function, hay cost function). Hàm mất mát này thường có giá trị nhỏ khi phép đánh giá cho kết quả tốt và ngược lại. Việc đi tìm các tham số mô hình sao cho phép đánh giá trả về kết quả tốt tương đương với việc tối thiểu hàm mất mát. Như vậy, việc xây dựng một mô hình machine learning chính là việc đi giải một bài toán tối ưu. Quá trình đó có thể được coi là quá trình learning của machine.

Tập hợp các tham số mô hình thường được ký hiệu bằng θ, hàm mất mát của mô hình thường được ký hiệu là L(θ) hoặc J(θ). Bài toán tối thiểu hàm mất mát để tìm tham số mô hình thường được viết dưới dạng:

$$\theta^* = \underset{\theta}{argmin}\ \mathrm{L}(\theta)$$

$\underset{\theta}{argmin}\ \mathrm{L}(\theta)$: được hiểu là giá trị của θ để hàm số L(θ) đạt giá trị nhỏ nhất.

2.6 Hàm kích hoạt

2.6.1 Sigmoid

Hàm sigmoid có công thức như sau:

$$\sigma(x) = \frac{1}{1+e^{-x}}$$

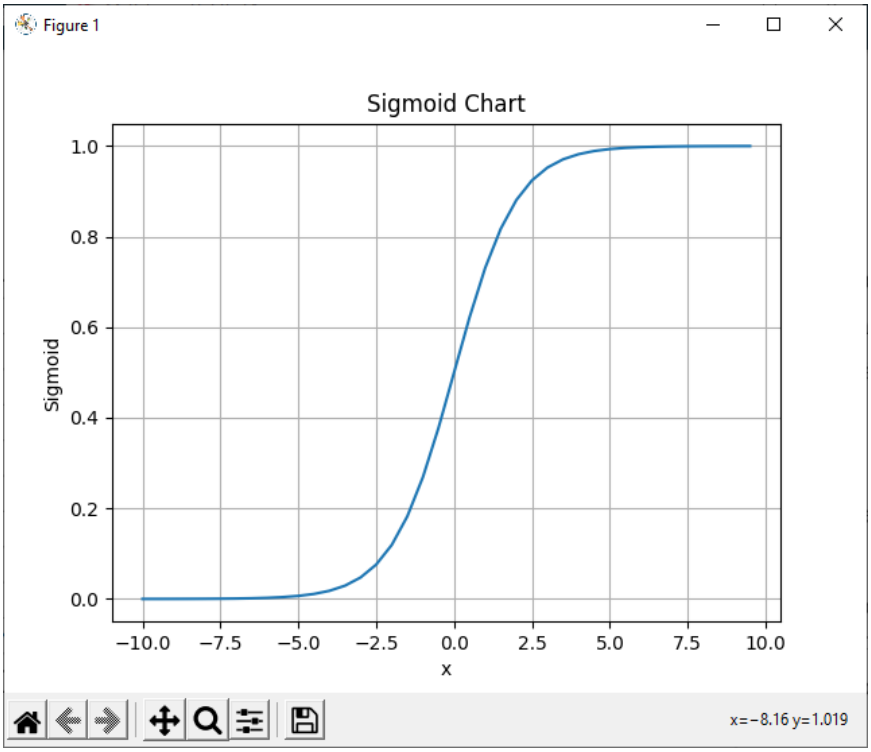


Figure 2.6.1: Đồ thị hàm số Sigmoid

Hàm Sigmoid nhận đầu vào là một số thực và chuyển thành một giá trị trong khoảng (0;1) (xem đồ thị phía trên). Đầu vào là số thực âm rất nhỏ sẽ cho đầu ra tiệm cận với 0, ngược lại, nếu đầu vào là một số thực dương lớn sẽ cho đầu ra là một số tiệm cận với 1. Trong quá khứ hàm Sigmoid hay được dùng vì có đạo hàm rất đẹp. Tuy nhiên hiện nay hàm Sigmoid rất ít được dùng vì những nhược điểm sau:

1. Hàm Sigmoid bão hào và triệt tiêu gradient: Một nhược điểm dễ nhận thấy là khi đầu vào có trị tuyệt đối lớn (rất âm hoặc rất dương), gradient của hàm số này sẽ rất gần với 0. Điều này đồng nghĩa với việc các hệ số tương ứng với unit đang xét sẽ gần như không được cập nhật (còn được gọi là *vanishing gradient*).

Hàm Sigmoid không có trung tâm là 0 gây khó khăn cho việc hội tụ

2.6.2 Tanh

Hàm tanh có công thức như sau:

$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

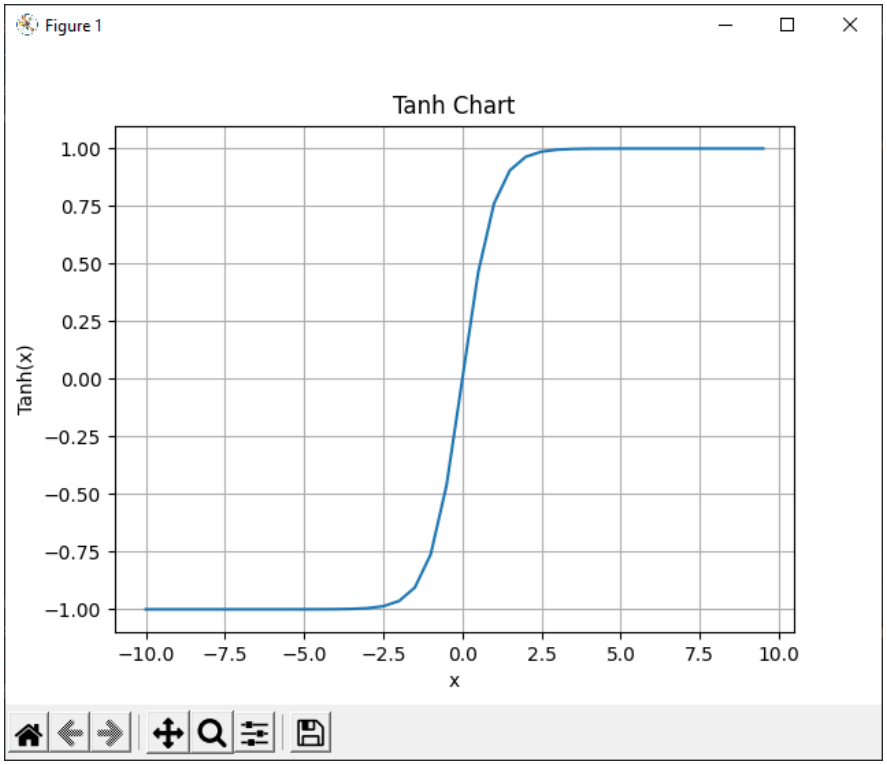


Figure 2.6.2: Đồ thị hàm số Tanh

Hàm tanh nhận đầu vào là một số thực và chuyển thành một giá trị trong khoảng (-1; 1). Cũng như Sigmoid, hàm Tanh bị bão hoà ở 2 đầu (gradient thay đổi rất ít ở 2 đầu). Tuy nhiên hàm Tanh lại đối xứng qua 0 nên khắc phục được một nhược điểm của Sigmoid.

2.6.3 ReLu

ReLu có công thức như sau:

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

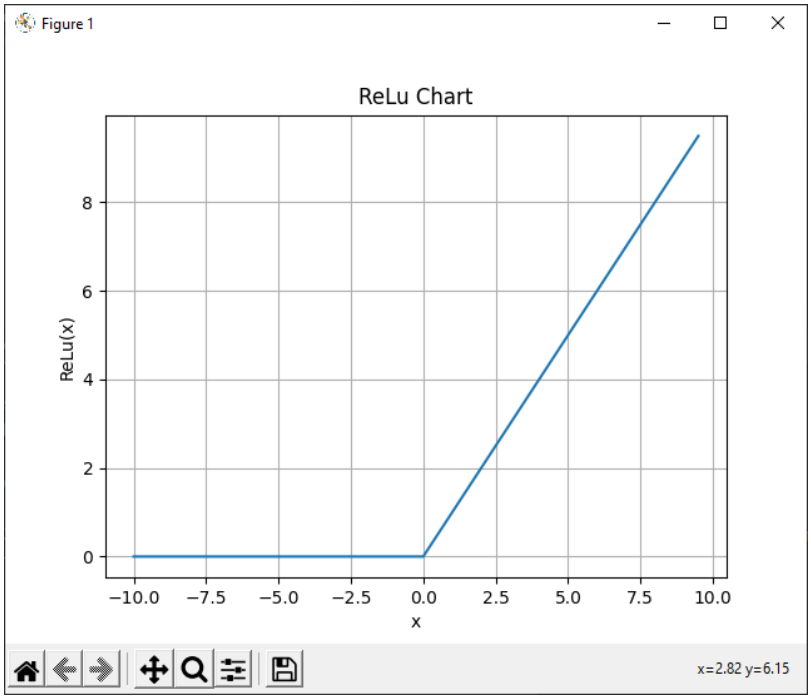


Figure 2.6.3: Đồ thị hàm số ReLu

Hàm ReLU đang được sử dụng khá nhiều trong những năm gần đây khi huấn luyện các mạng neuron. ReLU đơn giản lọc các giá trị < 0. Nhìn vào công thức chúng ta dễ dàng hiểu được cách hoạt động của nó. Một số ưu điểm khá vượt trội của nó so với Sigmoid và Tanh:

- (+) Tốc độ hội tụ nhanh hơn hẳn. ReLU có tốc độ hội tụ nhanh gấp 6 lần Tanh (Krizhevsky et al.). Điều này có thể do ReLU không bị bão hoà ở 2 đầu như Sigmoid và Tanh.
- (+) Tính toán nhanh hơn. Tanh và Sigmoid sử dụng hàm exp và công thức phức tạp hơn ReLU rất nhiều do vậy sẽ tốn nhiều chi phí hơn để tính toán.
- (-) Tuy nhiên ReLU cũng có một nhược điểm: Với các node có giá trị nhỏ hơn 0, qua ReLU activation sẽ thành 0, hiện tượng đấy gọi là "Dying ReLU". Nếu các node bị chuyển thành 0 thì sẽ không có ý nghĩa với bước linear activation ở lớp tiếp theo và các hệ số tương ứng từ node đấy cũng không được cập nhật với gradient descent. => Leaky ReLU ra đời.
- (-) Khi learning rate lớn, các trọng số (weights) có thể thay đổi theo cách làm tất cả neuron dừng việc cập nhật.

2.6.4 Leaky ReLu

Leaky ReLu có công thức như sau:

$$f(x) = 1(x < 0)(\alpha x) + 1(x >= 0)(x)$$

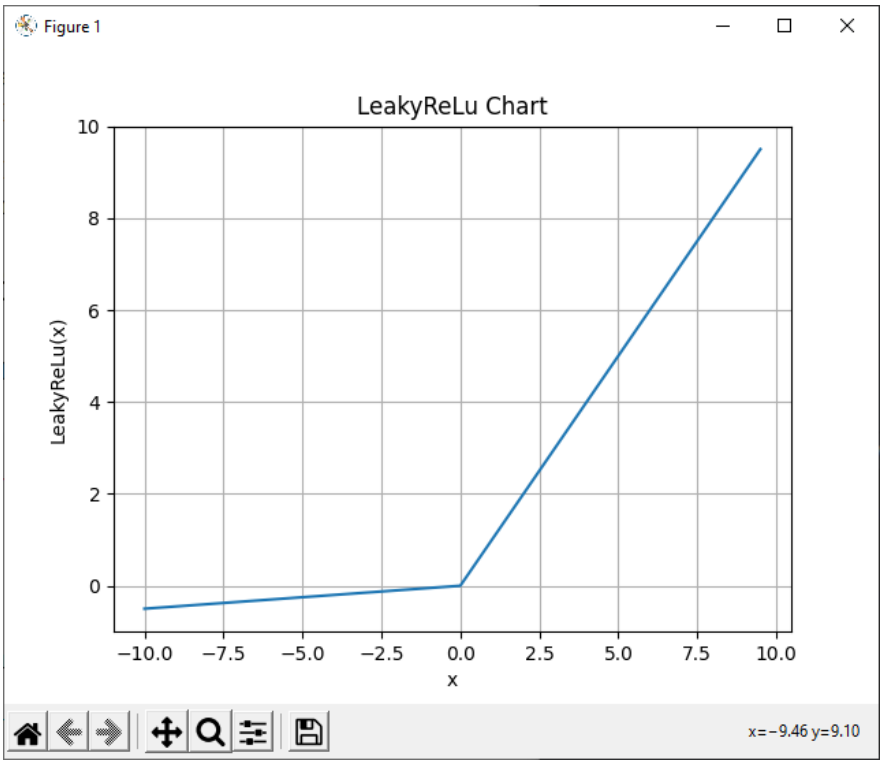


Figure 2.6.4: đồ thị hàm số LeakyReLu

Leaky ReLU là một cố gắng trong việc loại bỏ "dying ReLU". Thay vì trả về giá trị 0 với các đầu vào <0 thì Leaky ReLU tạo ra một đường xiên có độ dốc nhỏ (xem đồ thị). Có nhiều báo cáo về việc hiệu Leaky ReLU có hiệu quả tốt hơn ReLU, nhưng hiệu quả này vẫn chưa rõ ràng và nhất quán.

Ngoài Leaky ReLU có một biến thể cũng khá nổi tiếng của ReLU là PReLU. PReLU tương tự Leaky ReLU nhưng cho phép neuron tự động chọn hệ số α tốt nhất.

CHƯƠNG III: CÔNG VIỆC LIÊN QUAN

3.1 Khai phá dữ liệu:

khai phá dữ liệu là quá trình phân loại, sắp xếp các tập hợp dữ liệu lớn để xác định các mẫu và thiết lập các mối liên hệ nhằm giải quyết các vấn đề nhờ phân tích dữ liệu. Quá trình khai phá dữ liệu là một quá trình phức tạp bao gồm kho dữ liệu chuyên sâu cũng như các công nghệ tính toán. Hơn nữa, Data Mining không chỉ giới hạn trong việc trích xuất dữ liệu mà còn được sử dụng để chuyển đổi, làm sạch, tích hợp dữ liệu và phân tích mẫu.

Có nhiều tham số quan trọng khác nhau trong Data Mining, chẳng hạn như quy tắc kết hợp, phân loại, phân cụm và dự báo. Một số tính năng chính của Data Mining:

- Dự đoán các mẫu dựa trên xu hướng trong dữ liệu.
- Tính toán dự đoán kết quả
- Tạo thông tin phản hồi để phân tích
- Tập trung vào cơ sở dữ liệu lớn hơn.
- Phân cụm dữ liệu trực quan

3.2 Xử lý dữ liệu:

Dữ liệu trước khi đưa vào máy học phải qua bước xử lý thô, đây được xem là bước quan trọng trong quá trình làm việc. Vì độ chính xác của mô hình phụ thuộc vào độ chính xác của dữ liệu và cũng như các tính chất liên quan đến bài toán của dữ liệu. Chuẩn hóa dữ liệu là đưa dữ liệu từ các giá trị rất lớn khác nhau về chung một dạng tổng quát mà máy có thể học được. Nhằm mục đích giảm thời gian học của máy và tối ưu hóa việc tiêu tốn dung lượng trong máy, nên việc xử lý dữ liệu được xem là quá trình tối ưu trong các mô hình máy học hiện nay. Thường thì biên độ giá trị của các vecto khi được chuẩn hóa sẽ giao động trong khoảng [-1, 1], [0, 1] hoặc [-0.5, 0.5] … Những giá trị khi đưa vào chuẩn hóa thì các giá trị đầu ra sẽ không bị mất đi thuộc tính ban đầu.

Đối với dữ liệu này ta sẽ sử dụng một loại chuẩn hóa như sau.

Chuẩn hóa Min-Max: đây là loại chuẩn hóa đơn giản, giá trị được chuẩn hóa nằm trong khoảng [0; 1].

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

Trong đó:

x’ là giá trị đã được chuẩn hóa.

x: là giá trị tham chiếu trong chuỗi giá trị hiện hữu.

Min(x): giá trị nhỏ nhất trong chuỗi giá trị đang tham chiếu.

Max(x): giá trị lớn nhất trong chuối giá trị đang tham chiếu.

CHƯƠNG IV: CÁC PHƯƠNG PHÁP DỰ BÁO

4.1 Phương pháp trí tuệ nhân tạo

Phương pháp trí tuệ nhân tạo có thể ứng dụng được trong mọi lĩnh vực khác nhau, dự báo là một trong những công việc được biết đến nhất. Một mô hình nơ ron nhân tạo nghiên cứu về khả năng dự đoán tổng lượng bức xạ mặt trời tại Uganda dựa trên dữ liệu địa lý và khí tượng: vĩ độ, kinh độ, độ cao, thời gian nắng, độ ẩm tương đối và nhiệt độ tối đa với độ sai số 0,1% sai số tuyệt đối trung bình [8]. Do tính gián đoạn và ngẫu nhiên của năng lượng quang điện mặt trời nên các nhà điều hành hệ thống rất khó để điều động các trạm phát điện năng lượng mặt trời. Để tìm ra kỳ vọng cho việc sản xuất năng lượng điện mặt trời trước mắt, các mô hình thông thường đã xem xét dữ liệu nhiệt độ, độ ẩm và tốc độ gió để dự báo, nhưng những dự đoán này luôn không đủ chính xác trong điều kiện thời tiết khắc nghiệt. Chỉ số Aerosol (AI), cho biết các hạt vật chất trong khí quyển, được phát hiện có mối tương quan tuyến tính chặt chẽ với sự suy giảm bức xạ mặt trời và có thể có ảnh hưởng tiềm tàng đến năng lượng do các tấm PV tạo ra. Một mô hình dự báo điện năng PV mới được đề xuất trong bài báo này, coi dữ liệu AI là một tham số đầu vào bổ sung để dự báo sản lượng điện PV trong 24 giờ tiếp theo [9].

4.2 Phương pháp hồi quy tuyến tính

Linear Regression là thuật toán hồi quy tuyến tính, ở đó quan hệ giữa đầu vào và đầu ra được mô tả bởi một hàm tuyến tính. Thuật toán này còn được gọi là linear fitting hoặc linear least square

Theo cuốn sách [1] giá trị hồi quy phải có hàm mất mát nhỏ nhất so với giá trị thực thì được xem là chính xác. Để làm điều đó, ta phải đi đạo hàm để tìm giá trị đạo hàm tại vị trí hàm số bằng không. Tuy nhiên, nếu một hàm mất mát có đạo hàm không quá phức tạp, ta không thể đạo hàm chúng theo cách tương tự, đó cũng là khuyết điểm của linear regression.

Một khuyết điểm nữa của mô hình này là sự nhạy cảm với các yếu tố gây nhiễu. Vì bản chất là một hàm tuyến tính, khi những giá trị có tính chất khác nhau sẽ làm cho mô hình không còn yếu tố tuyến tính cũng như là độ chính xác cao được.

Đối với dữ liệu của mô hình chúng ta, việc thu thập PV theo thời gian trong ngày thay đổi ngẫu nhiên một cách liên tục. Vì thế đối với dữ liệu như vậy thì mô hình này không được xem là phương pháp tối ưu.

Mô hình nghiên cứu mạng nơ ron Bayesian được giới thiệu để ước tính lượng bức xạ mặt trời toàn cầu hàng ngày dựa trên bốn thông số đầu vào: nhiệt độ không khí, độ ẩm, thời gian nắng và chiếu xạ ngoài trái đất cho kết quả tốt hơn các phương pháp khác như mạng nơ ron cổ điển và các mô hình thực nghiệm [10]. Để đạt được độ chính xác của dự báo cao hơn, một mô hình được hoàn thành dựa trên cơ sở nguyên tắc kết hợp giữa mạng nơ-ron nhân tạo và phân tích Wavelet để phát triển một phương pháp dự báo bức xạ mặt trời mới [11]. Dự báo dài hạn về công suất đầu ra PV được thực hiện bằng cách sử dụng dữ liệu lịch sử, lý thuyết mờ và mạng nơ-ron [12]. Tính hợp lệ của phương pháp đề xuất được xác nhận bằng cách so sánh các khả năng dự báo trên các mô phỏng máy tính. Một mô hình mạng nơron nhân tạo đã được sử dụng để dự báo công suất PV và xác định khoảng thời gian đủ để biểu diễn dữ liệu PV một cách chính xác [13].

4.3 Phương pháp vật lý

Các phương pháp vật lý phải có mô hình cụ thể và phương pháp thu thập dữ liệu. Vì bức xạ bề mặt mặt trời có thể được suy ra từ các phép đo vệ tinh với độ phân giải không gian và thời gian cao, chúng tôi sử dụng ảnh vệ tinh làm nguồn dữ liệu để dự báo. Dữ liệu vệ tinh cung cấp thông tin về độ mây, thông số khí quyển quan trọng nhất đối với bức xạ bề mặt. Các vệ tinh, với khả năng giám sát chuyển động của đám mây trên các khu vực rộng, đã được sử dụng để dự báo bức xạ mặt trời [14] [15] [16]. Một mô hình tiên tiến đã được đề xuất để ước tính bức xạ mặt trời với việc giới thiệu các cảm biến mới giúp cải thiện đáng kể độ chính xác của dự báo [17]. Máy chụp ảnh bầu trời trên mặt đất được sử dụng để dự báo đám mây và bức xạ mặt trời, trong đó hình ảnh bầu trời được ghi lại sau mỗi nửa phút [18].

Vệ tinh là một cách hiệu quả để dự báo ngắn hạn lên đến 5 giờ. Trong trường hợp dự báo dài hạn về bức xạ mặt trời, các mô hình dự báo thời tiết bằng số được chứng minh là hiệu quả hơn vệ tinh [19]. Một số mô hình dự báo thời tiết bằng số tại một số địa điểm ở Hoa Kỳ, Châu Âu và Canada cho kết quả xác thực trên mô phỏng bức xạ mặt trời với độ phân giải ngang 500 m cho thấy độ lệch dương đối với bức xạ bình thường trực tiếp chiếm ưu thế trong khoảng thời gian từ 09 JST (Giờ chuẩn Nhật Bản) đến 15 JST. Dự báo thời tiết số (NWP) là công cụ chính xác nhất sử dụng dữ liệu đo đạc trên mặt đất của SURFRAD để dự báo sự chiếu xạ mặt trời trước vài giờ. Một nghiên cứu

toàn diện nhằm xác nhận và kiểm tra độ chính xác của một số mô hình dự báo thời tiết bằng số và hệ thống dự báo ở Hoa Kỳ [21].

4.4 Phương pháp lai (Hybrid Method)

Phương pháp ARIMA là phương pháp dự báo yếu tố nghiên cứu một cách độc lập (dự báo theo chuỗi thời gian). Bằng các thuật toán sử dụng độ trễ sẽ đưa ra mô hình dự báo thích hợp. Số quan sát tối thiểu để dùng được ARIMA là 50, môi trường dự báo trong tương lai ít có sự biến động.

Sự hiệu quả và chính xác của phương pháp lai có thể được xây dựng bằng cách kết hợp nhiều phương pháp dự báo khác nhau. ARMA và mô hình hồi quy phi tuyến được kết hợp để đạt được sự chính xác trong kết quả dự đoán [21]. Tác giả của [22] đã chứng minh rằng sự kết kết hợp ARMA và thời gian trễ của mang nơ ron tạo ra một phương pháp lai hiệu quả cho việc dự đoán bức xạ mặt trời. Tác gải của bài [23] đã kết hợp hai phương pháp truyền thống, ARIMA và support vector machines để dự đoán công suất PV. Tác giả của bài số [24] đã đề xuất hai phương pháp kết hợp mô hình tự động hồi quy và tự động hồi quy với mô hình giá trị đầu vào ngoại sinh cho việc dự đoán công suất PY ngắn hạn. cả hai đều làm mịn mô hình không gian trạng thái theo cấp số nhân và mạng nơ ron nhân tạo được để xuất trong bài [25].

Các phương pháp đã nêu ở trên không xem xét đến vấn đề thay đổi thời gian trong dữ liệu quá khứ của PV khi ta xây dựng mô hình. Do đó các tác giả đã loại bỏ các thông tin về sự biến đổi cảu dữ liệu theo thời gian.

Trong bài viết này, ta đề xuất việc sử dụng LSTM-RNN để xây dụng cấu trúc dự đoán công xuất sản lượng PV trong ngắn hạn. LSTM-RNN bao gồm sự thay đổi công suất theo thời gian, bằng cách đấy ta sẽ làm các mô hình đáng tin cậy hơn.

CHƯƠNG V: PHƯƠNG PHÁP ĐỀ XUẤT

5.1 LSTM (Long Short Term Memory networks):

Mạng bộ nhớ dài-ngắn (Long Short Term Memory networks), thường được gọi là LSTM - là một dạng đặc biệt của RNN, nó có khả năng học được các phụ thuộc xa. LSTM được giới thiệu bởi Hochreiter & Schmidhuber (1997), và sau đó đã được cải tiến và phổ biến bởi rất nhiều người trong ngành. Chúng hoạt động cực kì hiệu quả trên nhiều bài toán khác nhau nên dần đã trở nên phổ biến như hiện nay.

LSTM được thiết kế để tránh được vấn đề phụ thuộc xa (long-term dependency). Việc nhớ thông tin trong suốt thời gian dài là đặc tính mặc định của chúng, chứ ta không cần phải huấn luyện nó để có thể nhớ được. Tức là ngay nội tại của nó đã có thể ghi nhớ được mà không cần bất kì can thiệp nào.

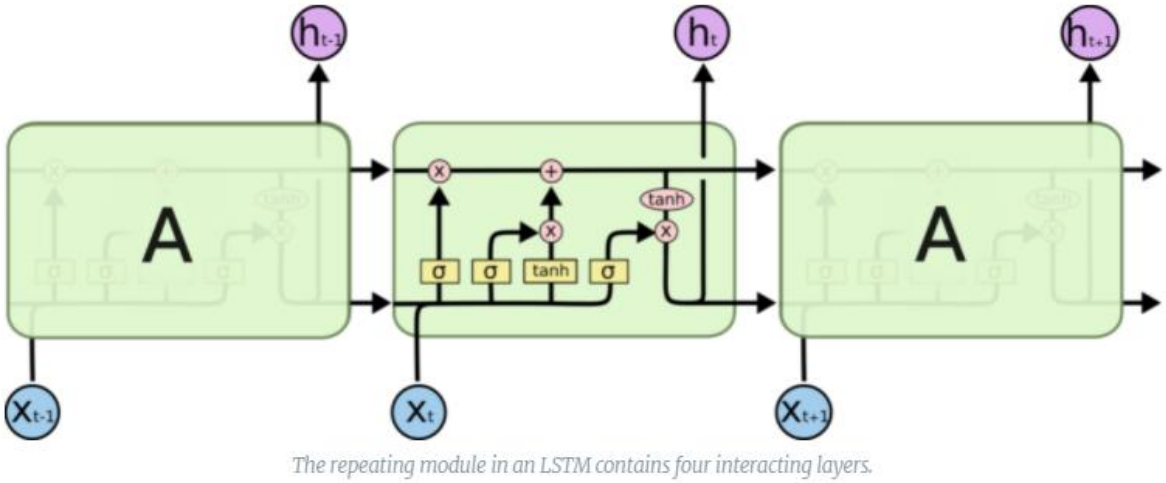
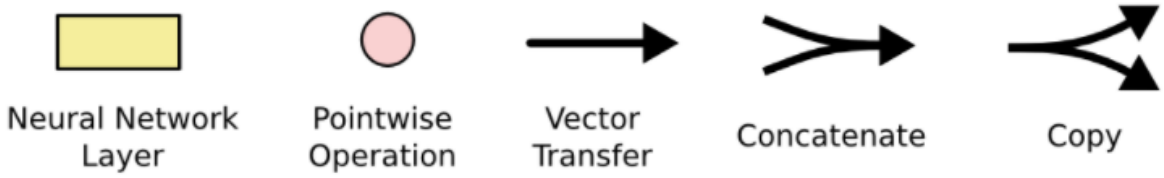


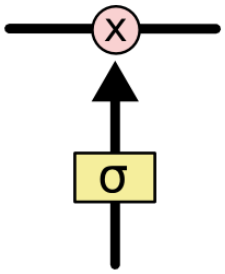
Figure 5.1.1: Mạng LSTM

Các ký hiệu bên trong mô hình LSTM:



Ở sơ đồ trên, mỗi một đường mang một véc-tơ từ đầu ra của một nút tới đầu vào của một nút khác. Các hình trong màu hồng biểu diễn các phép toán như phép cộng véc-tơ chẳng hạn, còn các ô màu vàng được sử dụng để học trong các từng mạng nơ-ron. Các đường hợp nhau kí hiệu việc kết hợp, còn các đường rẽ nhánh ám chỉ nội dung của nó được sao chép và chuyển tới các nơi khác nhau.

Cổng gate: là nơi sàng lọc thông tin đi qua nó, chúng được kết hợp bởi một tầng mạng sigmoid và một phép nhân.



5.1.1 Cổng quên - Forget Gate:

Cổng đầu tiên của LSTM giúp nó quyết định thông tin nào sẽ được rút ra từ cell. Bằng việc sử dụng **sigmoid layer** sẽ đưa ra kết quả trong khoảng từ **0** đến **1**. Điều này có nghĩa rằng nếu kết quả càng gần 0 thì có nghĩa là **đừng nhớ gì** và ngược lại **nhớ hoàn toàn** kết quả trạng thái phía trước. Đầu vào của cổng là kết quả đầu ra trước đó $\mathbf{h_{t-1}}$ và dữ liệu vào $\mathbf{x_t}$

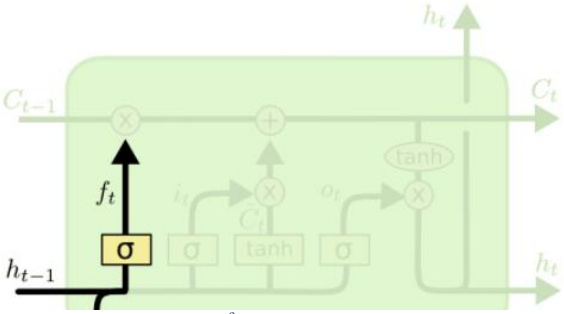


Figure 5.1.2: Cổng Quên – Forget Gate

$$f_t = \sigma(W_f.[h_{t-1}, x_t] + b_f)$$

5.1.2 Cổng vào - Input Gate:

Cổng tiếp theo của LSTM giúp nó quyết định thông tin nào sẽ được lưu trữ vào trong cell. Việc này được thực hiện nhờ 2 phần chính.

- ✓ **Input gate layer**: Dùng hàm sigmoid để quyết định thông tin nào chúng ta sẽ cập nhật, thêm mới.
- ✓ **Cell candidate**: Dùng hàm tanh để quyết định xem sẽ cập nhật như thế nào, bao nhiêu thông tin.

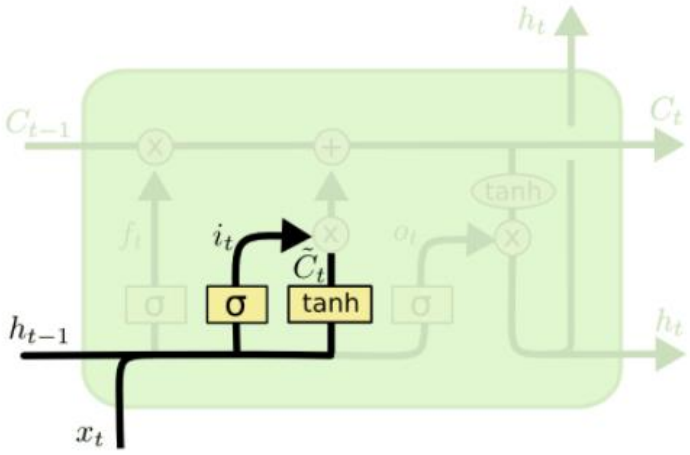
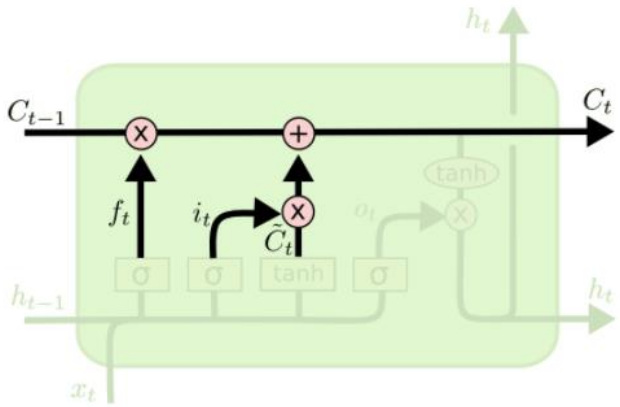


Figure 5.1.3: Cổng vào - InputGate

$$i_t = \sigma(W_i.[h_{t-1}, x_t] + b_i)$$

$$\tilde{C}_t = \tanh(W_C.[h_{t-1}, x_t] + b_C)$$

sau khi đã tính toán, việc tiếp theo cần làm là cập nhật giá trị của cell – **C**



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

5.1.3 Cổng ra - Output gate:

Cuối cùng, LSTM tính toán xem kết quả đầu ra là gì. Kết quả sẽ được tính toán dựa trên trạng thái tế bào đã được lọc.

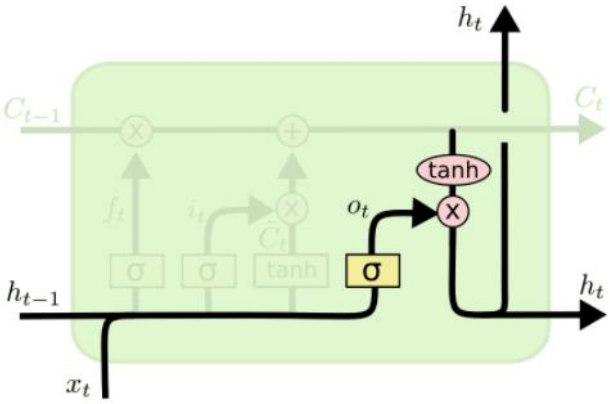


Figure 5.1.4: Cổng Ra - OutputGate

$$o_t = \sigma(W_o.[h_{t-1}, x_t] + b_o)$$

$$h_t = o_t * \tanh(C_t)$$

5.2 LSTM with Keras:

```
tf.keras.layers.LSTM(
    units,
    activation="tanh",
    recurrent_activation="sigmoid",
    use_bias=True,
    kernel_initializer="glorot_uniform",
    recurrent_initializer="orthogonal",
    bias_initializer="zeros",
    unit_forget_bias=True,
    kernel_regularizer=None,
    recurrent_regularizer=None,
    bias_regularizer=None,
    activity_regularizer=None,
    kernel_constraint=None,
    recurrent_constraint=None,
    bias_constraint=None,
    dropout=0.0,
    recurrent_dropout=0.0,
```

```
    return_sequences=False,
    return_state=False,
    go_backwards=False,
    stateful=False,
    time_major=False,
    unroll=False,
    **kwargs
)
```

Giải thích các thành phần trong phân lớp LSTM trong thư viện Keras

- **units**: Số chiều của output - Chính là độ lớn của vector $\mathbf{h}_t$ đầu ra, gần giống với lớp fully-connected.
- **activation**: Hàm kích hoạt sử dụng ở **cell candidate layer** và cổng ra sử dụng để tính toán đầu ra từ giá trị của cell. Như ở trong mô hình cơ bản thì mặc định của nó là hàm Tanh.
- **recurrent_activation**: Hàm kích hoạt ở các cổng quên, vào, ra - ở đây mặc định là hàm sigmoid
- **return_sequences**: Boolean - Quyết định xem sẽ trả về đầu ra cuối cùng của mạng hoặc tất cả các đầu ra.
- **unroll**: Boolean - Nếu True thì mạng sẽ không đổi, vòng lặp sẽ được sử dụng để tính toán các giá trị, Unroll có thể tăng tốc độ xử lí của RNN tuy nhiên sẽ chiếm dụng bộ nhớ lớn, nên option này chỉ phù hợp với các dữ liệu ngắn.
- **Return_state**: Kiểu Boolean. Liệu nó có trả lại trạng thái cuối cùng của đầu ra hay không. Mặc định là False

5.3 Dự đoán công suất PV sử dụng nhiều mô hình LSTM khác nhau:

Model 1: basic LSTM network for regression

Trong cấu trúc mạng này, ta xem việc dự báo năng lượng PV là một vấn đề liên quan đến việc hồi quy. Với công suất PV trong giờ này, ta sẽ dự đoán công suấ PV đầu ra trong giờ tiếp theo. Mạng LSTM được thiết kế như sau. Các mạng có một lớp hiển thị với một đầu vào, một lớp ẩn với bốn khối LSTM (tế bào thần kinh) và một lớp đầu ra đưa ra công suất dự đoán. Ta đã sử dụng hàm sigmoid kích hoạt chức năng cho các khối LSTM. Ta có quá trình huấn luyện là 30, 50 và 100 lần học với batch_size là 1.

Model 2: LSTM for regression using the window technique

Trong mô hình này, ta sử dụng nhiều bước thời gian gần đây để dự đoán công suất đầu ra PV ở bước thời gian tiếp theo (một kỹ thuật cửa sổ). Trong kỹ thuật này, ta có thể điều chỉnh kích thước của cửa sổ cho vấn đề dự báo công suất PV. Ví dụ, với thời gian hiện tại (t), ta hướng tới việc dự đoán công suất PV tại thời điểm tiếp theo (t + 1). Để làm như vậy, ta sử dụng công suất PV của thời điểm hiện tại t và công suất của hai thời điểm trước (t – 1) và (t - 2) làm biến đầu vào cho các nút của LSTM. Trong trường hợp này, các biến đầu vào của nút LSTM là công suất PV tại thời điểm t, (t – 1) và (t – 2) trong khi biến đầu ra là công suất PV tại thời điểm tương lai (t+1).

Model 3: LSTM for regression with time steps

Giống như mô hình trước đó (model 2), ta lấy các bước thời gian trước trong chuỗi thời gian của công suất PV như là đầu vào để dự đoán công suất PV đầu ra ở thời điểm tiếp theo. Trong mô hình này, thay vì sử dụng các quan sát trước đây làm đầu vào riêng biệt, Ta sử dụng chúng như các bước thời gian của một tính năng đầu vào, là tính năng tạo khung chính xác hơn cho vấn đề dự báo công suất PV. Ví dụ, nếu thời gian bước bằng 3 (Lookback =3), đơn vị LSTM tạo ra công suất PV tại thời gian t sau khi nó xử lý công suất PV tại (t – 3), (t – 2) và (t – 1).

Model 4: LSTM with memory between batches

Mạng LSTM có một bộ nhớ cho phép nó nhớ qua các chuỗi dài. Khi thiết lặp mô hình vào cấu trúc bình thường, ta đặt lại trạng thái (reset_state) trong mạng sau mỗi đợt đào tạo. Ta có thể làm cho tốt hơn khi trạng thái nội bộ của mạng LSTM được xóa bằng cách làm cho lớp LSTM có trạng thái. Nói cách khác, LSTM có thể xây dựng trạng thái trong toàn bộ quá trình đào tạo trình tự và thậm chí duy trì trạng thái đó nếu cần để dự đoán công suất đầu ra PV. Nó yêu cầu dữ liệu đào tạo không được xáo trộn khi lắp mạng LSTM.

Model 5: Stacked LSTMs with memory between batches

LSTM xếp chồng thêm dung lượng bằng cách xếp chồng các lớp LSTM lên đầu nhau [20, 50]. Các mạng LSTM có thể được xếp chồng lên nhau theo cùng một cách mà các

loại lớp khác có thể được xếp chồng lên nhau (ví dụ: các lớp của mạng nơ-ron). Hình 2 trình bày cách Các lớp LSTM có thể được xếp chồng lên nhau. Các khối màu xanh thuộc về layer1, trong khi các khối màu đỏ thuộc về layer 2. Các đầu vào cho lớp 1 là công suất PV x_t; x_{t+1}; ... x_N trong khi các đầu vào cho lớp2 là h_t; h_{t+1}; ... h_N.

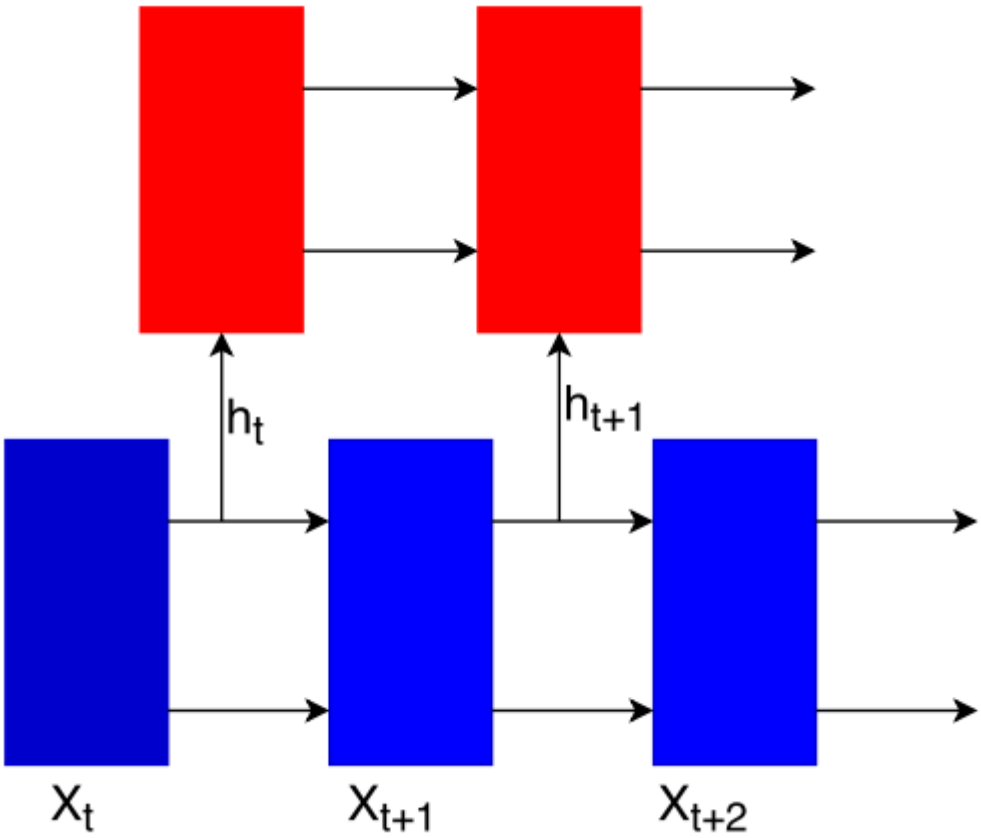


Figure 5.3.1: LSTM xếp chồng

CHƯƠNG VI: KẾT QUẢ MÔ HÌNH

6.1 Dữ liệu:

Dữ liệu được thu thập tại tòa D trường đại học Tôn Đức Thắng, phường Tân phong, Quận 7. Hệ thống PV gồm có 3 inverter có công suất 60kWp, có hiệu suất tối ta đặt dưới 90%. Việc thu thập dữ liệu bắt đầu từ 01/04/2021 đến 30/04/2021, vì một vài vấn đề liên quan đến thủ tục nên công việc thu thập dữ liệu được thực hiện thủ công thông qua việc chụp lại các thông số hiển thị trên inverter đều đặng hằng ngày.

Dựa vào hình 5 ta thấy rõ hiện tại Tp. Hồ Chí Minh đang vào mùa mưa, nên giá trị kWp theo ngày có giá trị thấp cao không đồng đều vào các ngày khác nhau. Số giờ nắng cực đại cũng giảm dần. Cho ta thấy một cách tổng quan về sự ảnh hưởng của yêu tố thời tiết đến khả năng sản sinh năng lượng của hệ thống PV.

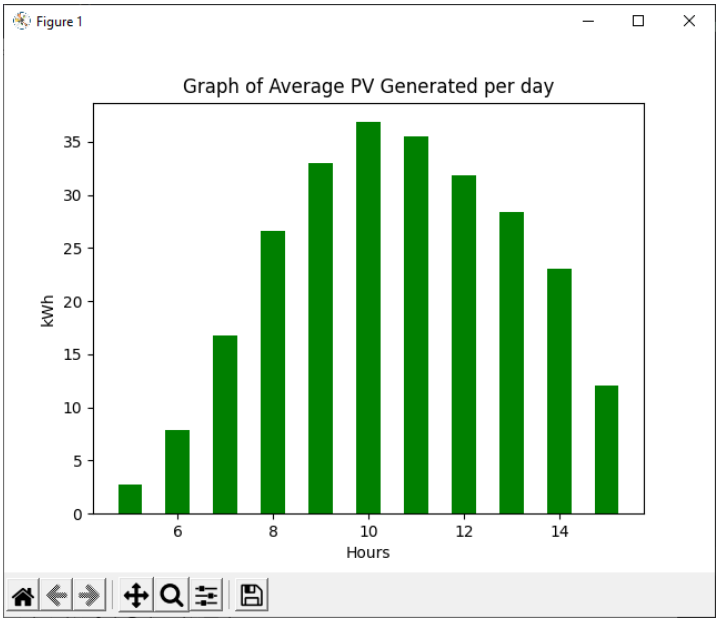


Figure 6.1.2: (a) Hour of day

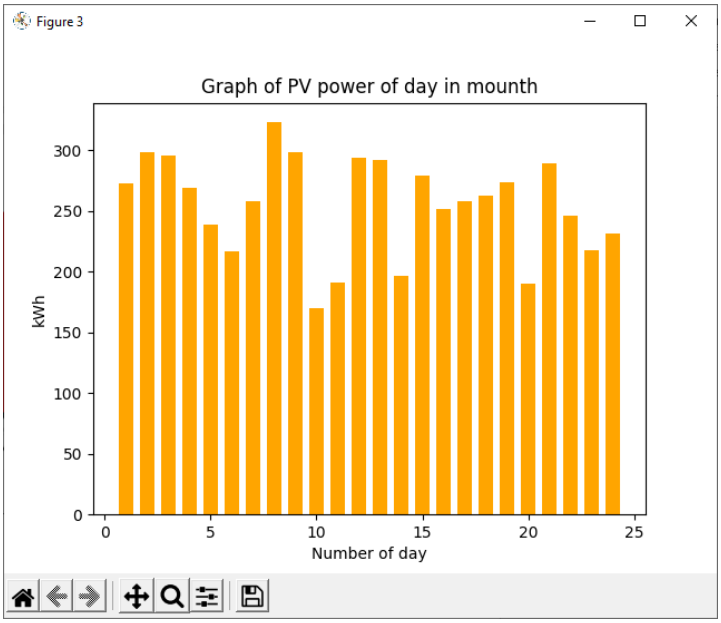


Figure 6.1.1: (b) PV of month

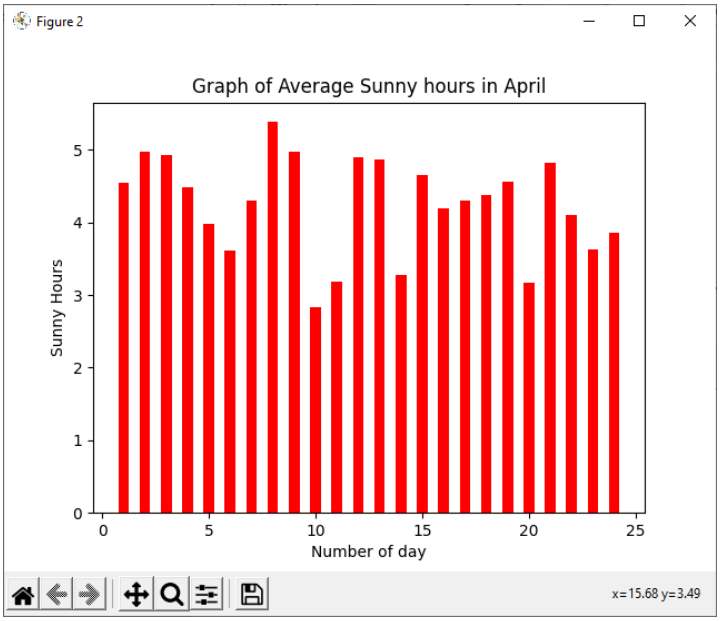


Figure 6.1.3: (c) sunny hours in month

6.2 Kết quả:

Chúng ta chia dữ liệu gốc thành 80% sử dụng làm tập huấn luyện và 20% còn lại sử dụng là tập kiểm thử. Sau khi dự đoán ta sẽ lấy giá trị dự đoán của tập kiểm thử để so sánh với dữ liệu gốc. ta sử dụng root-mean-square error (RMSE) làm thang đo cho sự chính xác của các giá trị dự đoán, vì hiện tại những giá trị dự đoán chưa có một quy chuẩn cụ thể làm thang đo độ chính. Nếu RMSE gần tiền về 0 thì mô hình dự đoán hoàn toàn chính xác.

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\hat{X}_i - X_i)^2}$$

Trong đó, $\hat{X}_i$ và X_i lần lượt là giá trị dự đoán và thực tế thứ i. N là số lượng mẫu kiểm thử.

Hàm mất mát trong LSTM là mean-square error, hàm tối ưu ‘adam’. Mô hình LSTM được thực hiện bằng thư viện keras

Trong các mô hình dùng để so sánh được cấu hình như nhau. Gồm có 5 nút của lớp ẩn đầu tiên (LSTM), 3 nút cho lớp ẩn thứ 2 và 1 nút ứng với một giá trị đầu ra cho lớp ẩn cuối cùng.

Sau đây là hai bảng so sánh giá trị RMSE (root mean square error) của hai tập giá trị train và test sau khi mô hình xuất kết quả dự đoán.

RMSE_Test Value			
Model| Epochs	**30**	**50**	**100**
1	**10.03**	**9.92**	**9.89**
2	**8.1**	**8.05**	**8.06**
3	**8.18**	**7.83**	**7.74**
4	**7.62**	**7.36**	**7.46**
5	**7.61**	**7.39**	**8.7**

Table 6.2.1: RMSE_Test Value

RMSE_Train Value			
Model| Epochs	**30**	**50**	**100**
1	**9.08**	**9.3**	**9.04**
2	**7.68**	**7.65**	**7.6**
3	**7.48**	**7.28**	**7.21**
4	**6.87**	**5.89**	**6.57**
5	**6.34**	**5.89**	**5.86**

Table 6.2.2: RMSE_Train Value

Ta thấy:

- Table 5.2.1 là giá trị RMSE của tập dữ liệu kiểm thử sau khi ta so giữa giá trị dự đoán và giá trị thực tế của tập dữ liệu.
- Table 5.2.2 là giá trị RMSE của tập dữ liệu huấn luyện khi ta so giữa giá trị dự đoán và giá trị thực tế của tập dữ liệu

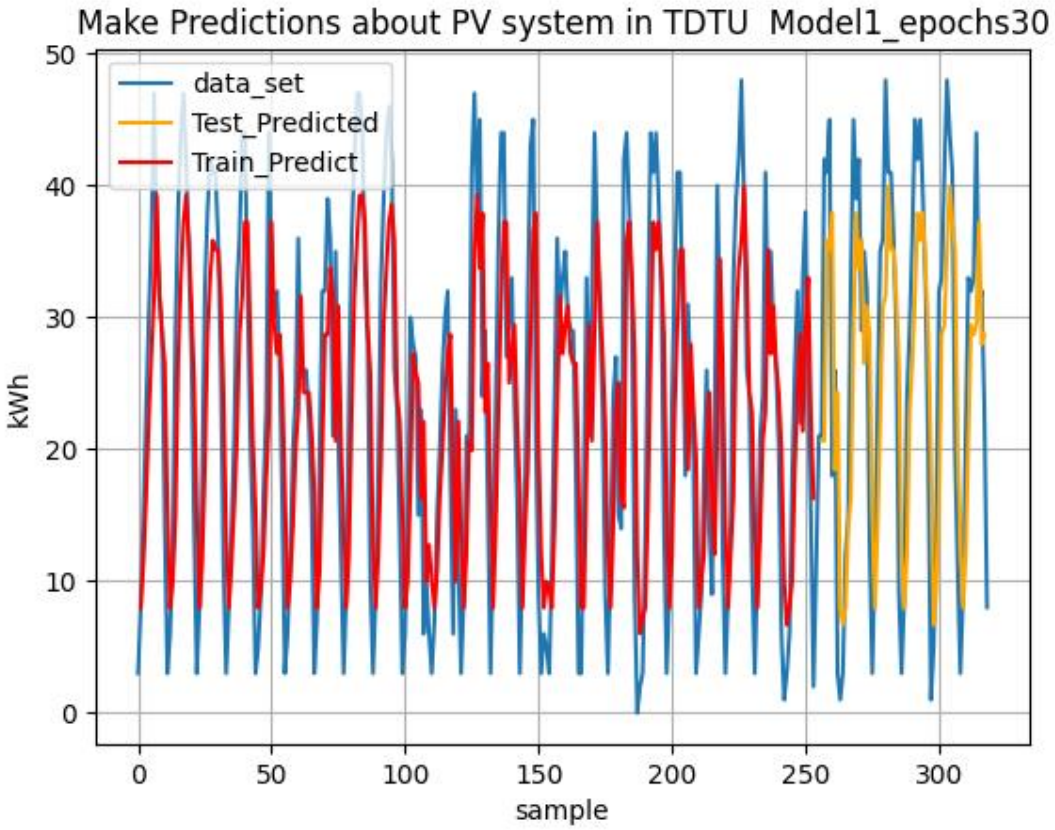
Ta thấy nếu giá trị RMSE gần bằng 0 nghĩa là mô hình được huấn luyện rất tốt đồng thời sai số giữa giá trị thực và giá trị dự đoán rất nhỏ.

So sánh giá trị từ Table 5.2.1 và Table 5.2.2

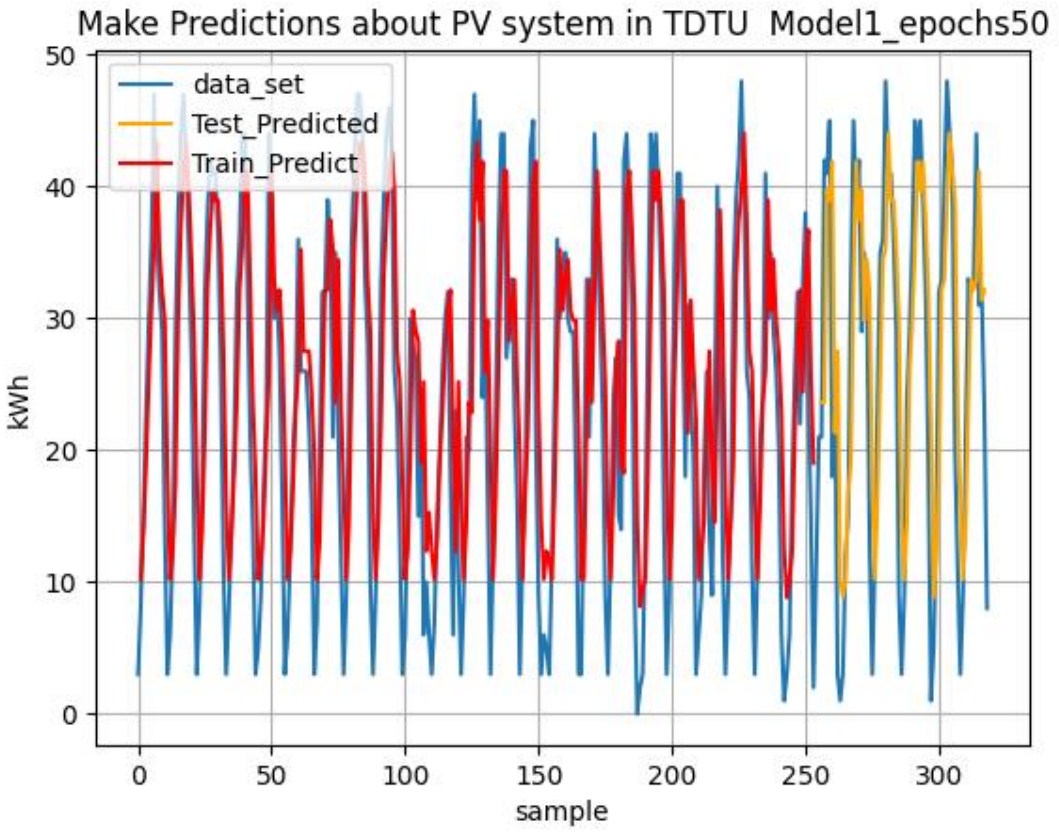
- RMSE_Test nhỏ nhất thuộc mô hình thứ 4, số lần học 50 có giá trị là 7.36.
- RMSE_Train nhỏ nhất thuộc mô hình 5, số lần học là 100 có giá trị là 5.86.

Sau đây là một vài hình ảnh về giá trị dự đoán và giá trị thực tế của từng mô hình qua từng lần huấn luyện ứng với Table 5.2.1 và Table 5.2.2.

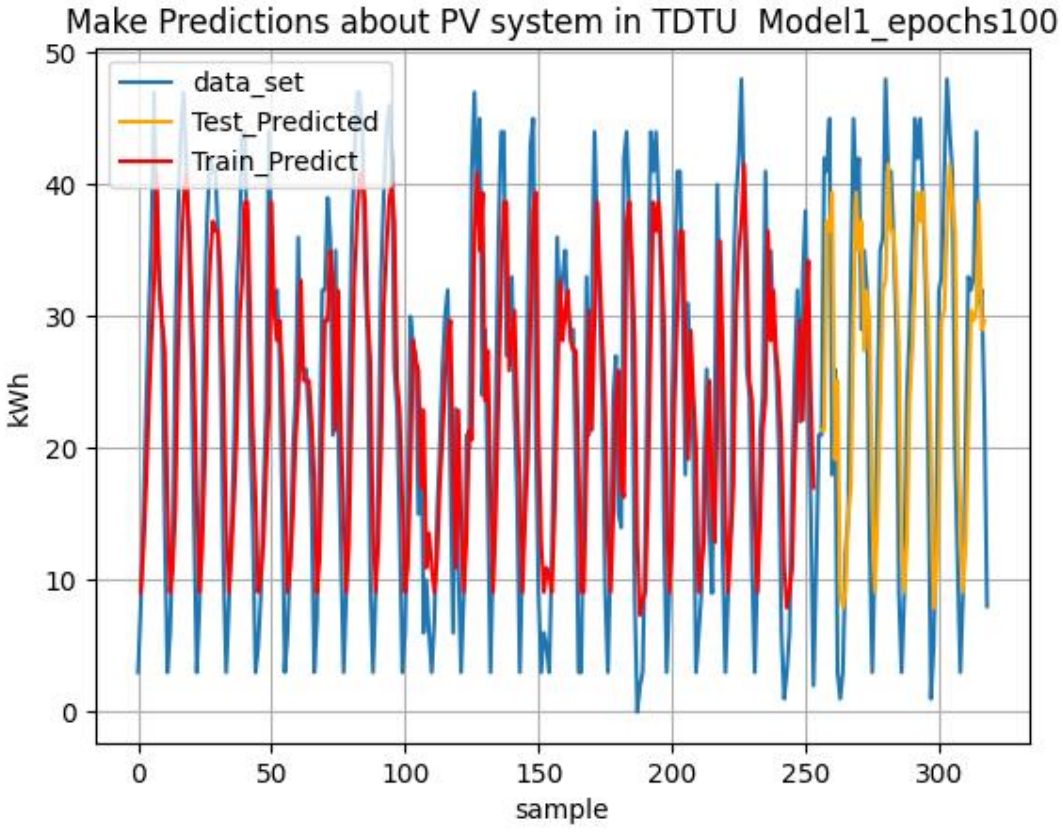
Trục đứng là giá trị PV theo kWh, trục ngang là số lượng dữ liệu tính theo giờ



(a)

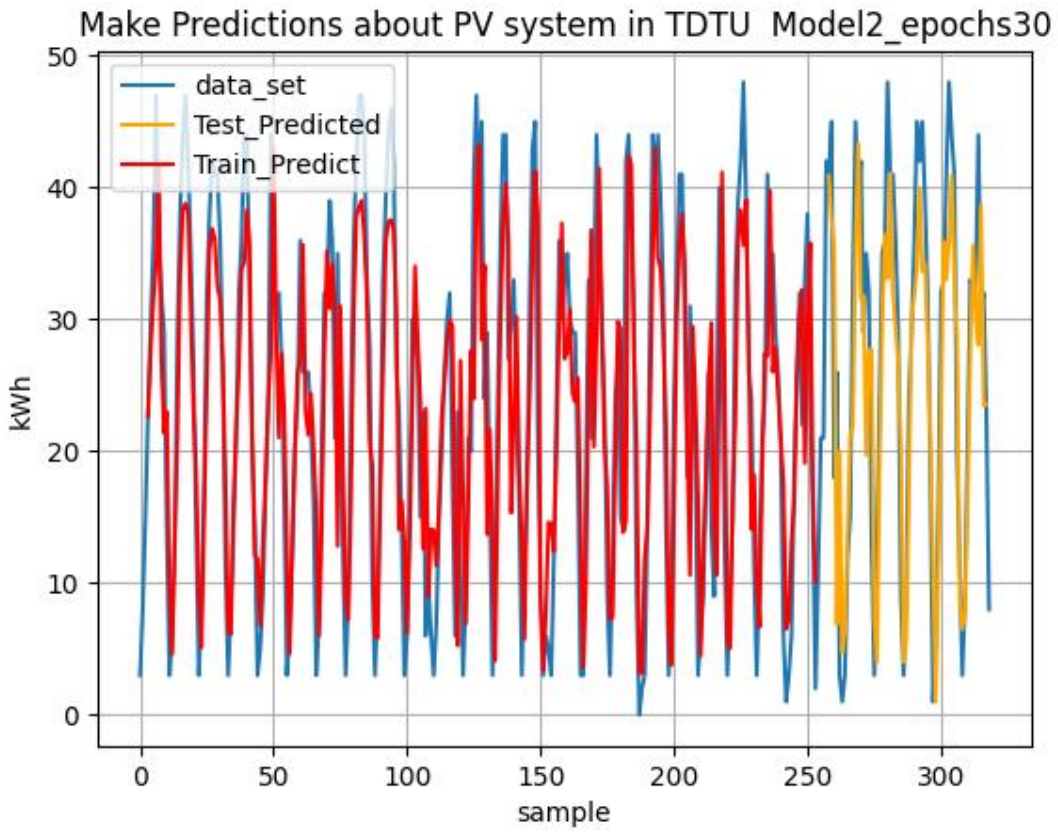

Make Predictions about PV system in TDTU Model1_epochs50
data_set
Test_Predicted
Train_Predict
kWh
sample
0
10
20
30
40
50
0
50
100
150
200
250
300

(b)

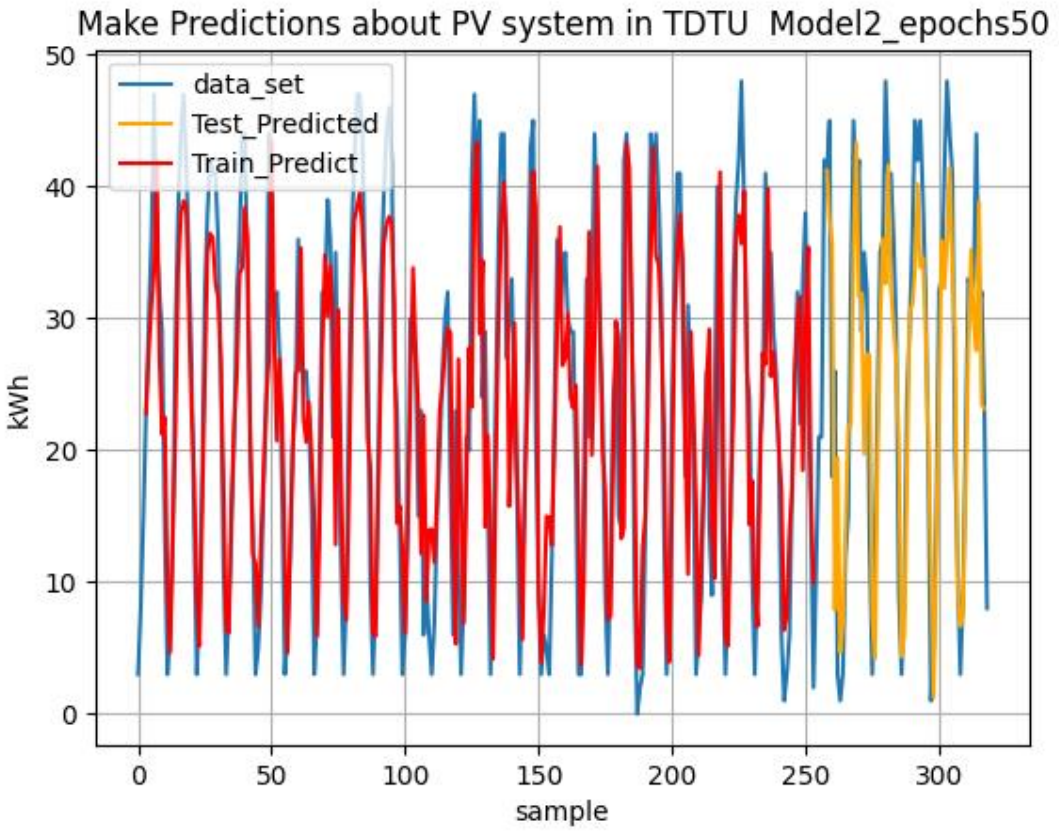

Make Predictions about PV system in TDTU Model1_epochs100
data_set
Test_Predicted
Train_Predict
kWh
sample
0
10
20
30
40
50
0
50
100
150
200
250
300

(c)

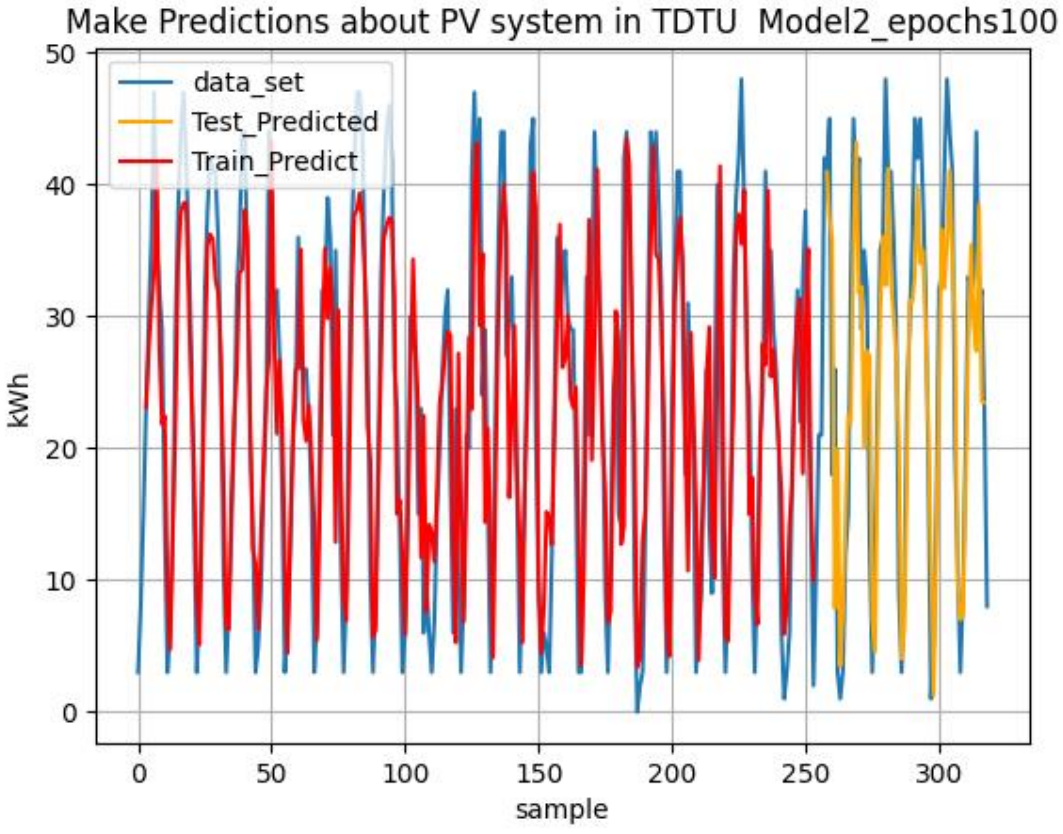
Model 1 basic LSTM network for regression, (a) epochs 30, (b) epochs 50, (c) epoch 100.



(a)

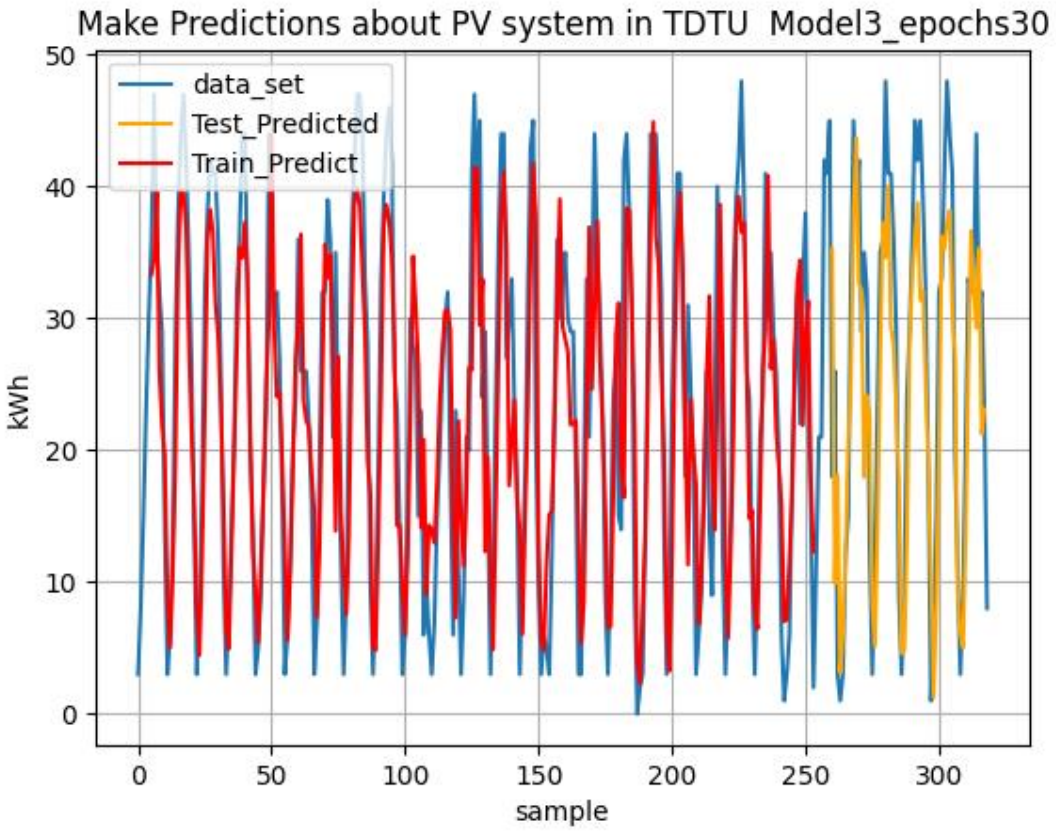

Make Predictions about PV system in TDTU Model2_epochs50
data_set
Test_Predicted
Train_Predict
kWh
sample
0
10
20
30
40
50
0
50
100
150
200
250
300

(b)

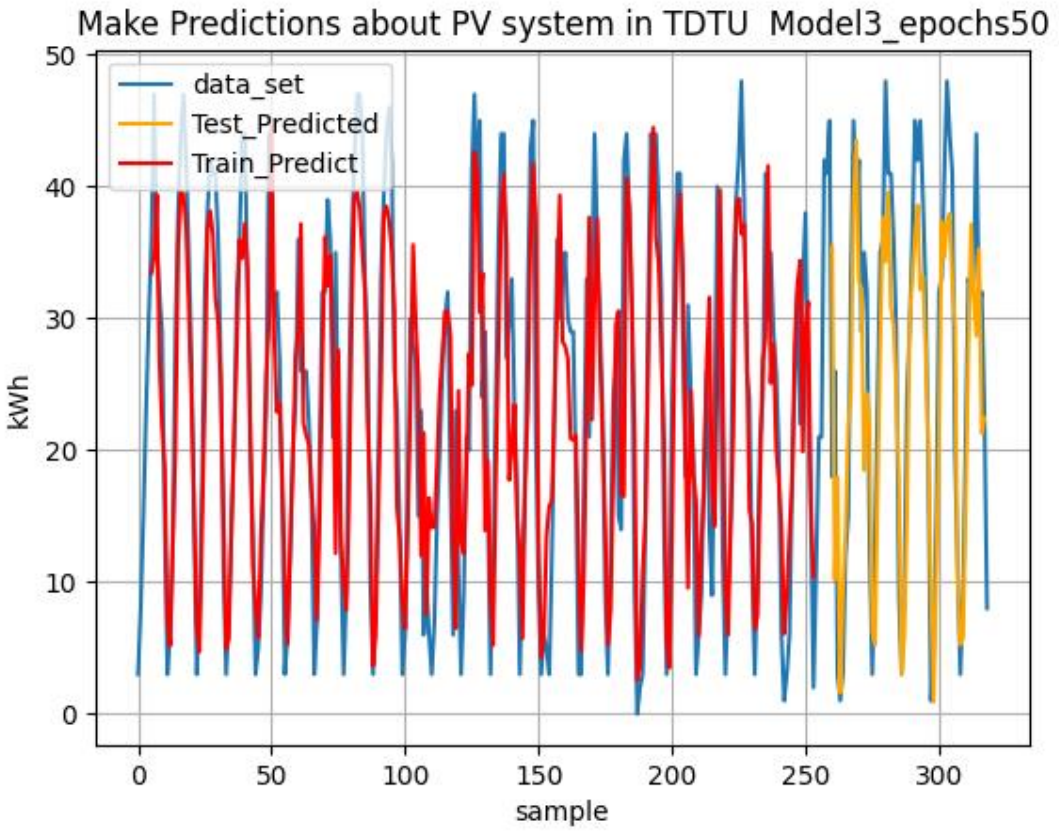

Make Predictions about PV system in TDTU Model2_epochs100
data_set
Test_Predicted
Train_Predict
kWh
sample
0
10
20
30
40
50
0
50
100
150
200
250
300

(c)

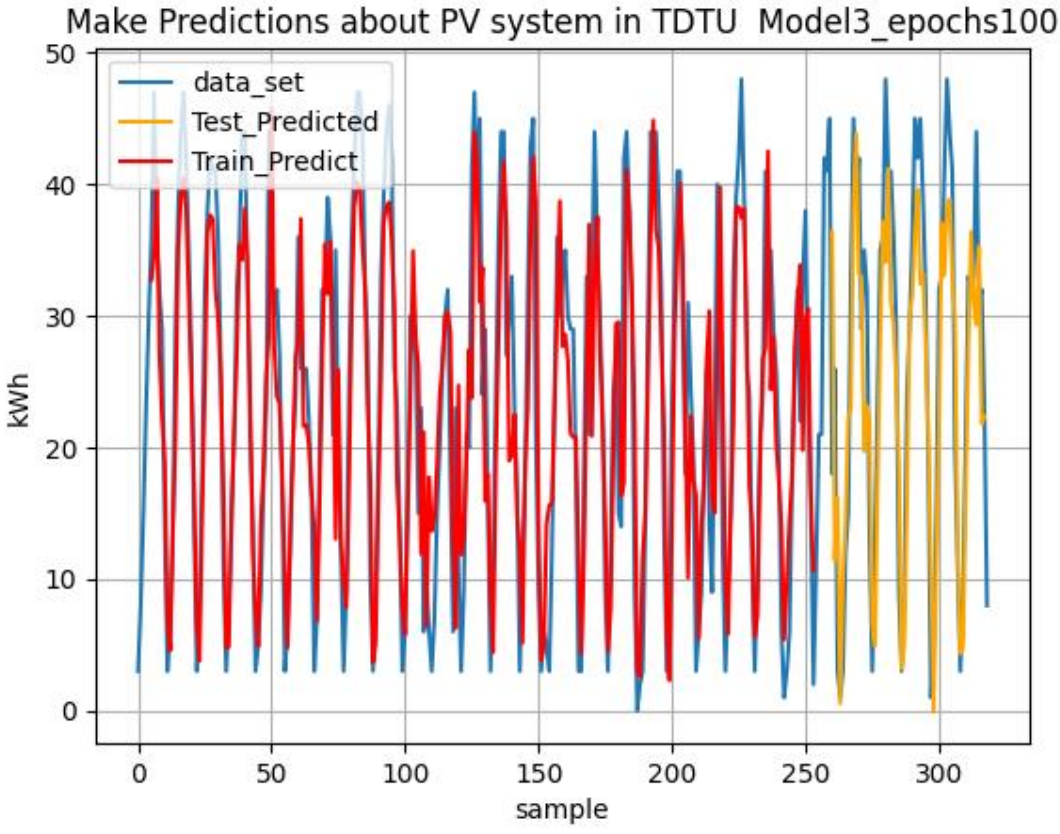
Model 2 LSTM for regression using the window technique, (a) epochs 30, (b) epochs 50, (c) epoch 100.



(a)

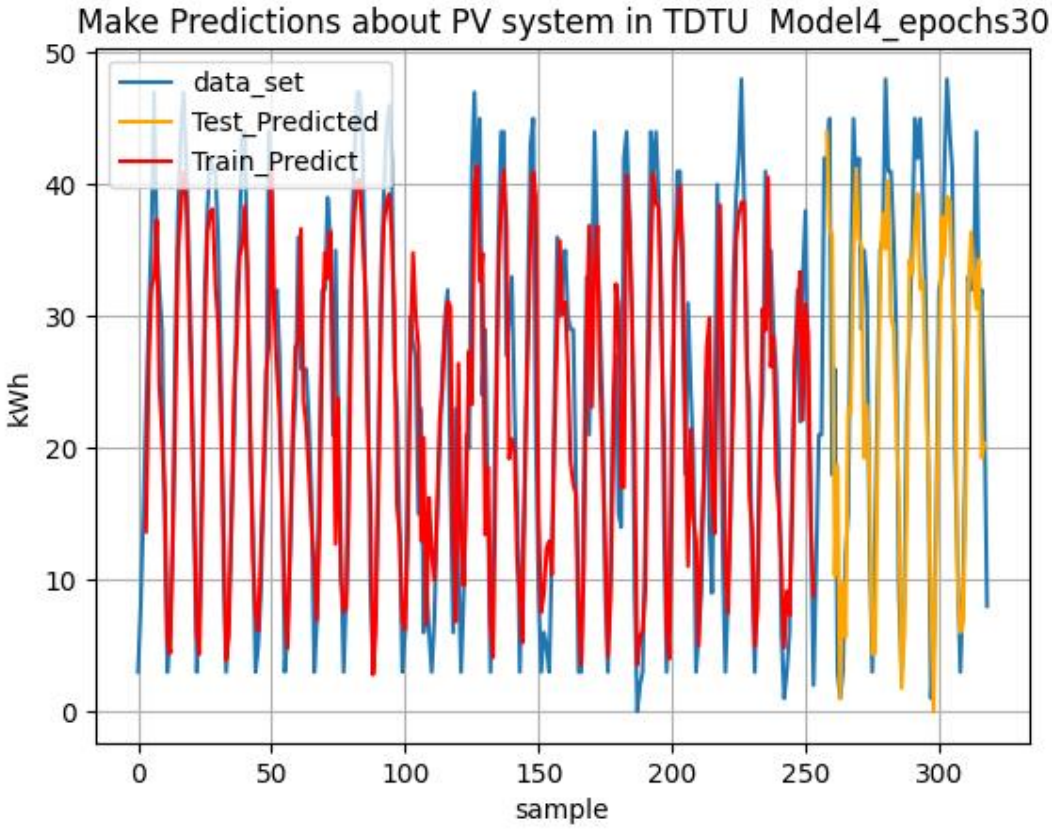

Make Predictions about PV system in TDTU Model3_epochs50
data_set
Test_Predicted
Train_Predict
kWh
sample
0
10
20
30
40
50
0
50
100
150
200
250
300

(b)

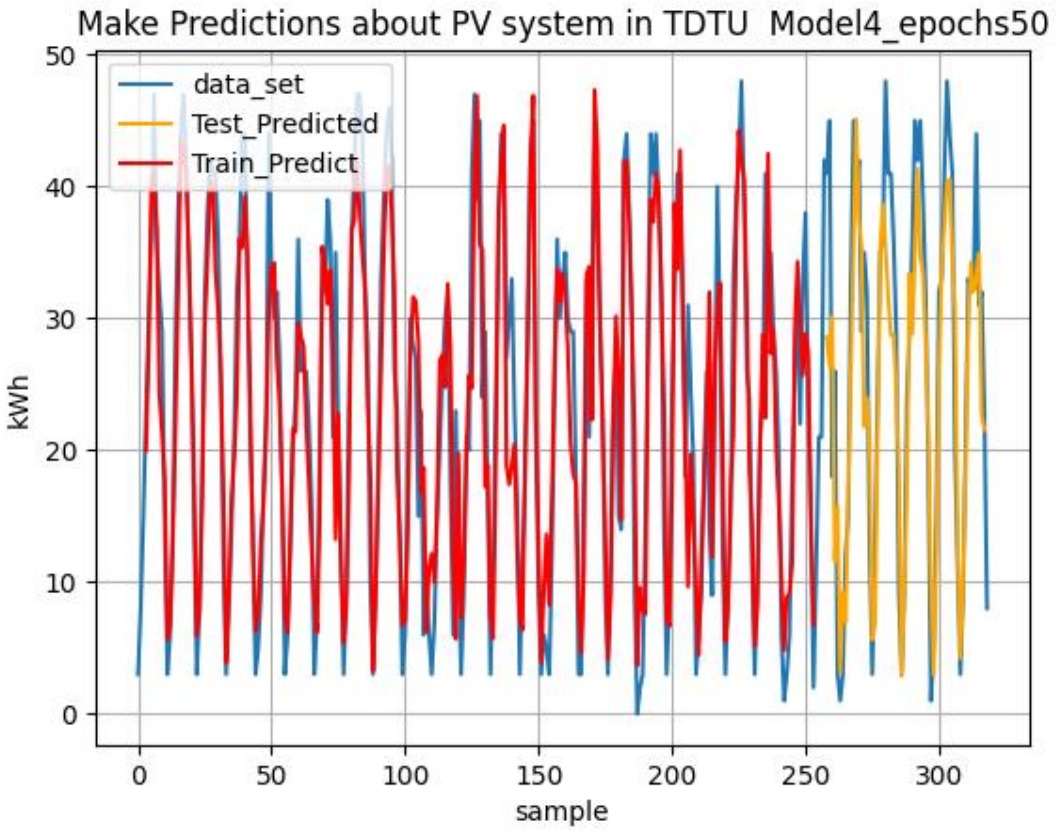

Make Predictions about PV system in TDTU Model3_epochs100
data_set
Test_Predicted
Train_Predict
kWh
sample
0
10
20
30
40
50
0
50
100
150
200
250
300

(c)

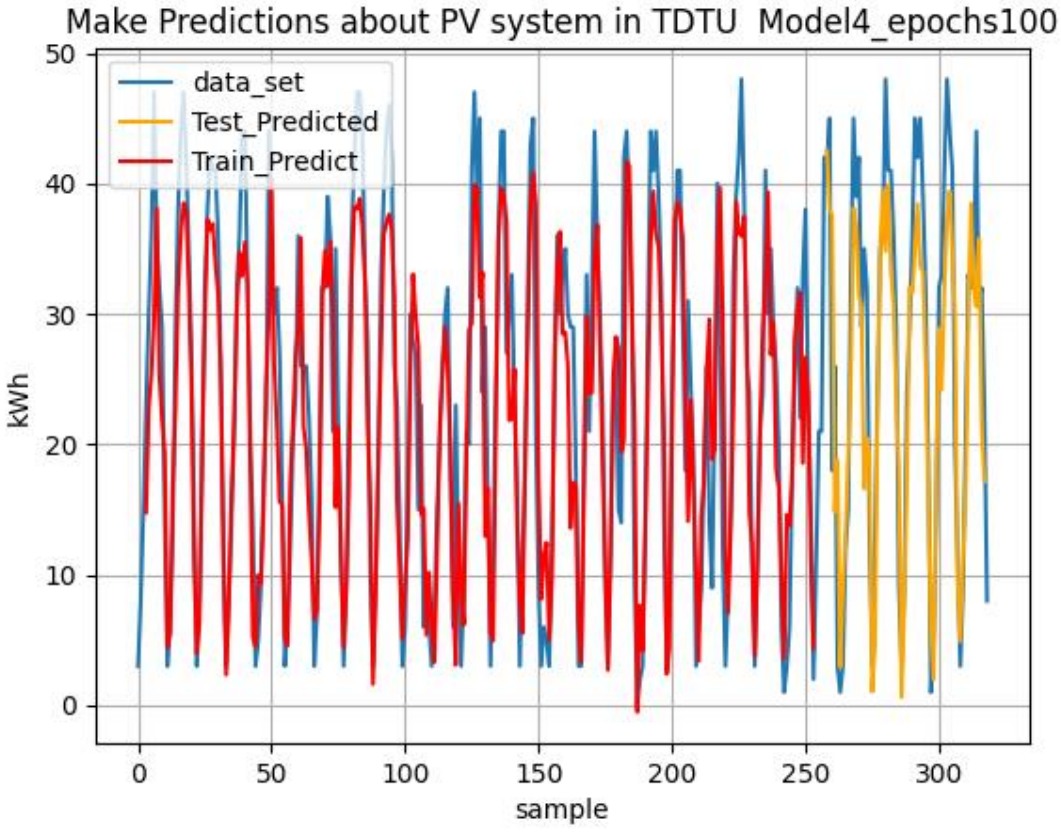
Model 3 LSTM for Regression with Time Steps, (a) epochs 30, (b) epochs 50, (c) epoch 100.



(a)

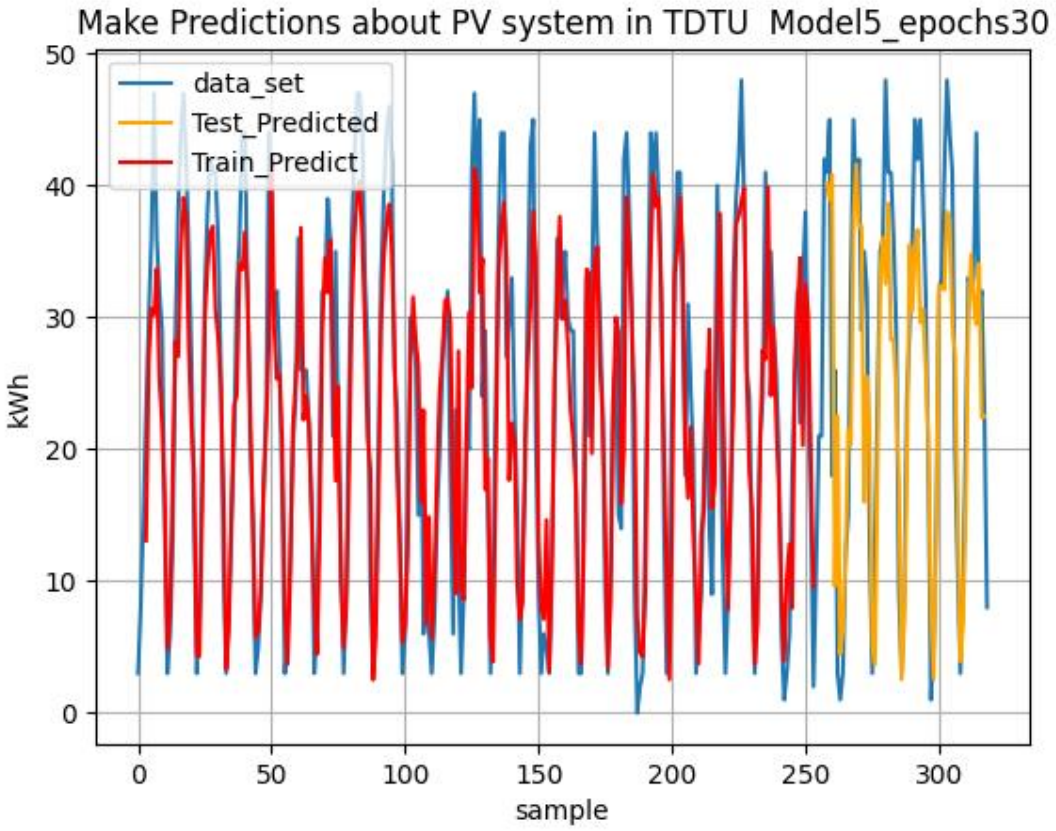

Make Predictions about PV system in TDTU Model4_epochs50
data_set
Test_Predicted
Train_Predict
kWh
sample
0
10
20
30
40
50
0
50
100
150
200
250
300

(b)

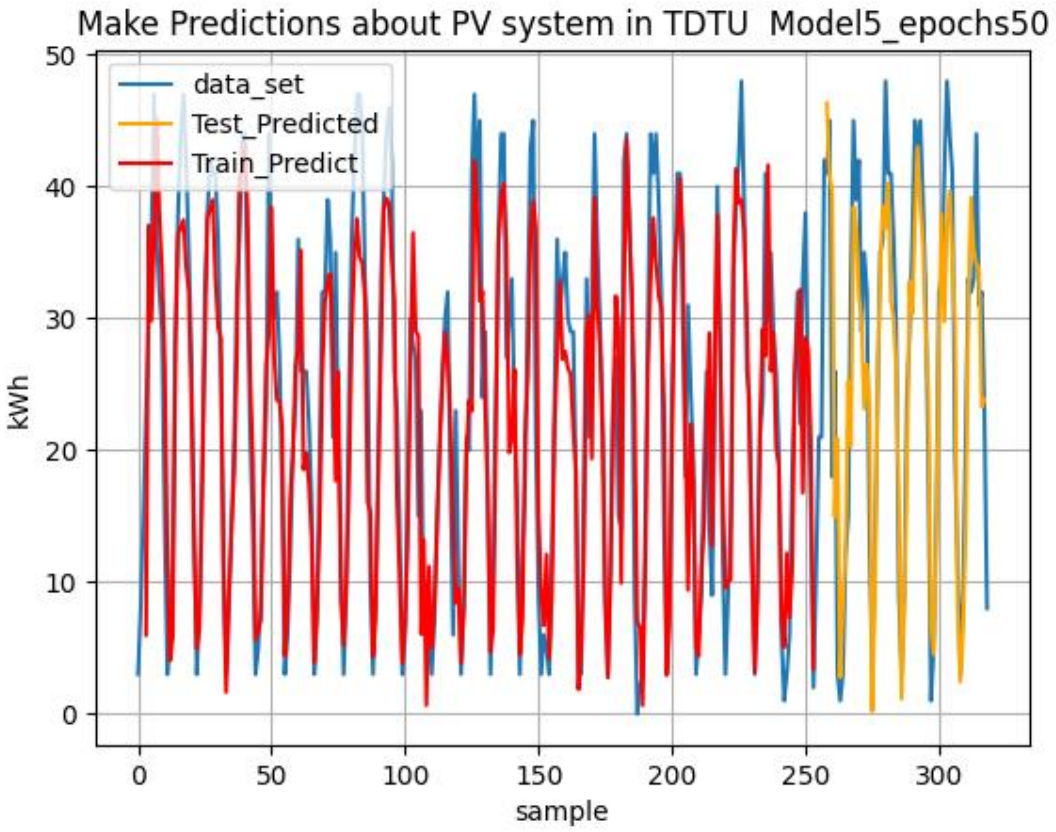

Make Predictions about PV system in TDTU Model4_epochs100
data_set
Test_Predicted
Train_Predict
kWh
sample
0
10
20
30
40
50
0
50
100
150
200
250
300

(c)

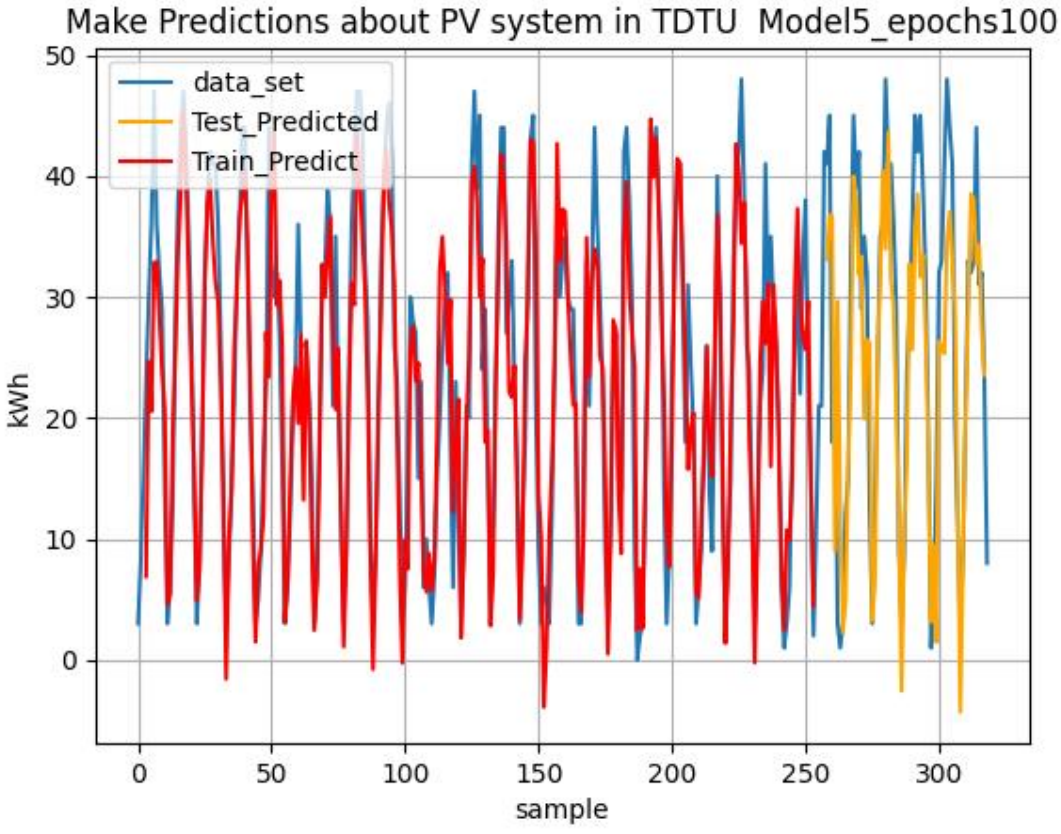
Model 4: LSTM with Memory Between Batches, (a) epochs 30, (b) epochs 50, (c) epoch 100.



(a)


Make Predictions about PV system in TDTU Model5_epochs50
data_set
Test_Predicted
Train_Predict
kWh
sample
0
10
20
30
40
50
0
50
100
150
200
250
300

(b)


Make Predictions about PV system in TDTU Model5_epochs100
data_set
Test_Predicted
Train_Predict
kWh
sample
0
10
20
30
40
50
0
50
100
150
200
250
300

(c)

Model 5: Stacked LSTMs with Memory Between Batches, (a) epochs 30, (b) epochs 50, (c) epoch 100.

Sau đây là giá trị dự đoán theo từng khung giờ của model 5. Ta sẽ dự báo dựa vào dữ liệu có sẵn xem theo từng khung giờ.

6.3 So sánh với những phương pháp khác

Trong phần này ta sẽ so sánh tính hiệu quả của phương pháp đề xuất (mô hình 5) với hai phương pháp dự đoán công suất PV như sao: Linear Regression, NeuralNetwork. Chúng ta sẽ xét giá trị RMSE trên tập kiểm thử của tất cả các mô hình xem mô hình đề xuất có thật sự là tốt khi so sánh với các mô hình khác hay không.

So sánh độ chính xác của các mô hình

	RMSE_test	Memory_based?	Recurrent?
LinearRegression	14.19	No	No
RegressionNeuralNetwork	20.59	No	No
Model 4	7.36	yes	Yes

Table 6.3.1: So sánh độ chính xác của các mô hình

Ta thấy độ chính xác của mô hình 4 vượt trội hơn hẳn so với các mô hình khác, vì LSTM hỗ trợ cho ta khả năng nhớ một cách tuần tự. Từ đó giải quyết các vấn đề liên quan đến tính chất phức tạp của bài toán, ở đây là yếu tố ảnh hưởng của thời tiết đối với hệ thống PV. Không giống như LSTM thì LinearRegression và RNN không hỗ trợ nhớ cho dữ liệu của chúng ta mà thay vào đó chúng chỉ dựa vào dữ liệu đầu vào và suy ra dữ liệu đầu ra một cách tuần tự. Vì vậy chúng không thể dự đoán chính xác được giá trị PV.

Phương pháp RNN nó mô phỏng lại cấu trúc của LSTM nhưng nó không có khả năng nhớ những dữ liệu trước đã học, học trước quên sau. Còn đối với LSTM, dữ liệu được đưa vào có dạng chuỗi thời gian quá khứ để dự báo cho những giá trị ở hiện tại.

6.4 Kết quả dự đoán công suất PV ngày 29/04/2020

Dựa vào model có độ chính xác RMSE_test cao nhất để thực hiện việc dự đoán công suất PV cho ngày hôm sau:

Cụ thể dữ liệu được đưa vào huấn luyện gồm có từ ngày 30/03/2020 đến ngày 27/04/2020. Không đưa giá trị ngày 28/04 vào huấn luyện, mục đích là để tăng độ khách quan cho mô hình. Giá trị dự đoán có thể khác nhau đối với các lần huấn luyện khác nhau. Ta cho giá trị đầu vào LSTM là t, (t-1), (t-2), (t-3) ứng với giá trị của các ngày trước đó và giá trị đầu ra là (t+1) ứng với giá trị của ngày 29/04-giá trị cần được dự đoán.

Giá trị dự đoán có thể sẽ thay đổi trong những lần huấn luyện khác nhau. Về mặt kỹ thuật, việc sử dụng LSTM mang lại kết quả đáng mong đợi cho vai trò dự đoán những giá trị có sự thay đổi ngẫu nhiên theo chuỗi thời gian. Việc dự đoán có thể sẽ mất thời gian cho việc xử lý dữ liệu và chạy mô hình.

Bảng So sánh giá trị dự đoán và giá trị thực của ngày 28/04/2020

Hours (h)	Data_28_04_2020	Data_Predict	Đơn vị
5	3.00	0.51	kWh
6	9.00	7.05	
7	21.00	15.4	
8	33.00	18.3	
9	32.00	27.9	
10	33.00	32.10	
11	44.00	30.35	
12	31.00	26.14	
13	32.00	20.05	
14	23.00	12.04	
15	8.00	5.15	

Table 6.4.1: Bảng So sánh giá trị dự đoán và giá trị thực của ngày 28/04/2020

Đồ thị dự đoán công suất PV:

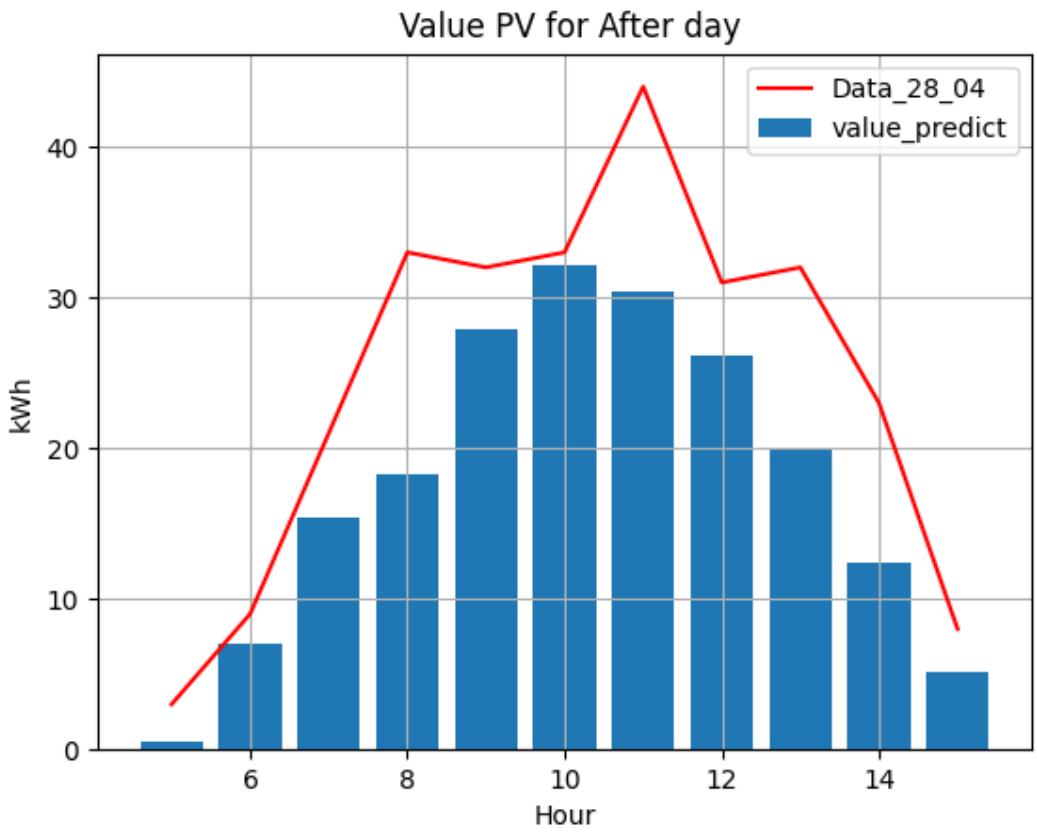


Figure 6.4.1: Biểu đồ sản lượng dự đoán công suất PV

Giá trị được mô hình LSTM dự đoán cho ngày hôm sau (28/04/2020) từ 5h đến 15h.

CHƯƠNG VII: TỔNG KẾT

Ở các chương trước, ta đã đề xuất một phương pháp mới cho việc dự đoán công suất sản sinh của PV trong thời gian ngắn hạn bằng cách sử dụng LSTM. Không giống như các phương pháp dự đoán truyền thống trước đây chúng ta đã biết, phương pháp của chúng ta dựa vào LSTM có thể nắm bắt được các mối liên hệ với nhau về công suất phát PV mà cụ thể ở đây là tính chất về thời tiết. Vì vậy LSTM có thể là mô hình dùng cho việc dự đoán chuỗi thời thời gian sản lượng PV trong ngắn hạn vì cấu trúc lặp lại của nó và các đơn vị bộ nhớ (memory units). Ta đã đánh giá hiệu suất của năm mô hình LSTM ứng với năm cấu trúc khác nhau cho việc dự đoán công suất PV. Mô hình tốt nhất được đề xuất là mô hình 4 (LSTM with Memory Between Batches) cho kết quả tối ưu hơn các mô hình trước đó. Vì vậy nó được đưa vào áp dụng cho việc dự đoán công suất sản sinh của hệ thống PV trong ngày tiếp theo.

Ta cũng đã so sánh mô hình 4 với các phương pháp LinearRegression, RNN. Mô hình 4 cho kết quả sai số dựa vào bộ dữ liệu kiểm thử (RMSE_test) là tối ưu nhất.

Trong tương lai mô hình 4 sẽ được phát triển hơn nữa, bằng cách sử dụng cấu trúc mô hình lặp lại (Recurrent Neural Network) khác nhau và cải thiện hàm mất mát nhằm mục đích tăng độ chính xác cho mô hình dự báo.

Tài Liệu Tham Khảo

[1] Vũ Hữu tiệp. Machine Learning Cơ Bản. June 15, 2019.

[2] Biaowei Chen, Peijie Lin, Yunfeng Lai, Shuying Cheng, Zhicong Chen, and Lijun Wu. Very-Short-Term Power Prediction for PV Power Plants Using a Simple and Effective RCC-LSTM Model Based on Short Term Multivariate Historical Datasets. 8 February 2020.

[3] Mohamed Abdel-Nasser, Karar Mahmoud. Accurate photovoltaic power forecasting models using deep LSTM-RNN. 4 October 2017.

[4] Can Wan, Jian Zhao, Yonghua Song, Zhao Xu, Jin Lin and Zechun Hu. Photovoltaic and Solar Power Forecasting for Smart Grid Energy Management. DECEMBER 2015.

[5] Mingming Gao, Jianjing Li, Feng Hong, and Dongteng Long. Short-Term Forecasting of Power Production in a Large-Scale Photovoltaic Plant Based on LSTM. 5 August 2019.

[6] Abdelhakim El hendouzi, Abdennaser Bourouhou. Solar Photovoltaic Power Forecasting. 31 December 2020

[7] Sean D. CAMPBELL, Francis X. DIEBOLD. Weather Forecasting for Weather Derivatives.

[8] Mubiru J (2008) Predicting total solar irradiation values using artificial neural networks. Liu J, Fang W, Zhang X, Yang C (2015) An improved photo voltaic power forecasting model with the assistance of aerosol index data.

[9] Yacef R, Benghanem M, Mellit A (2012) Prediction of daily global solar irradiation data using Bayesian neural network: a comparative study.

[10] Cao JC, Cao S (2006) Study of forecasting solar irradiance using neural networks with preprocessing sample data by wavelet analysis.

[11] Yona A, Senjyu T, Funabashi T, Kim CH (2013) Determination method of insolation prediction with fuzzy and applying neural network for long-term ahead PV power output correction.

[12] Izgi E, Oztopal A, Yerli B, Kaymak MK, Sahin AD (2012) Short-mid-term solar power prediction by using artificial neural networks.

[13] Hammer A, Heinemann D, Lorenz E, Luckehe B (1999) Short-term forecasting of solar radiation: a statistical approach using satellite data.

[14] Lorenz E, Hurka J, Heinemann D, Beyer HG (2009) Irradiance forecasting for the power prediction of grid-connected photo-voltaic systems.
[15] Perez R, Kivalov S, Schlemmer J, Hemker K, Renne D, Hoff TE (2010) Validation of short and medium term operational solar radiation forecasts in the US.
[16] Geraldi E, Romano F, Ricciardelli E (2012) An advanced model for the estimation of the surface solar irradiance under all atmospheric conditions using MSG/SEVIRI data.
[17] Chow CW, Urquhart B, Lave M, Dominguez A, Kleissl J, Shields J, Washom B (2011) Intra-hour forecasting with a total sky imager at the UC San Diego solar energy testbed.
[18] Lorenz E, Hurka J, Heinemann D, Beyer HG (2009) Irradiance forecasting for the power prediction of grid-connected photo-voltaic systems.
[19] Perez R, Lorenz E, Pelland S, Beauharnois M, Van Knowe G, Hemker K, Heinemann D, Remund J, Mu¨ller SC, Traunmu¨ller W et al (2013) Comparison of numerical weather prediction solar irradiance forecasts in the US, Canada and Europe.
[20] Mathiesen P, Kleissl J (2011) Evaluation of numerical weather prediction for intra-day solar forecasting in the continental united states.
[21] Benmouiza K, Cheknane A (2016) Small-scale solar radiation forecasting using ARMA and nonlinear autoregressive neural network models.
[22] Ji W, Chee KC (2011) Prediction of hourly solar radiation using a novel hybrid model of ARMA and TDNN.
[23] Bouzerdoum M, Mellit A, Pavan AM (2013) A hybrid model (SARIMA–SVM) for short-term power forecasting of a smallscale grid-connected photovoltaic plant.
[24] Bacher P, Madsen H, Nielsen HA (2009) Online short-term solar power forecasting
[25] Dong Z, Yang D, Reindl T, Walsh WM (2014) Satellite image analysis and a hybrid ESSS/ANN model to forecast solar irradiance in the tropics

Phụ Lục

Import Thư viện

```
from keras.models import Sequential
from keras.layers import Dense
from sklearn.datasets import make_blobs
import numpy
import math
import pandas
from sklearn.preprocessing import MinMaxScaler
from keras.layers import LSTM,Dropout
from sklearn.metrics import mean_squared_error, mean_absolute_percentage_error,
mean_absolute_error
import math
import matplotlib.pyplot as plt
from keras.models import load_model
from keras.models import Sequential
from keras.optimizers import SGD
```

Code tham khảo cho việc huấn luyện LSTM

Khởi tạo dữ liệu:

```
file=pandas.read_excel(r'C:\Users\Bao\Desktop\file so lieu dien mat troi TDTU.xlsx')
data=numpy.array(file)
data=numpy.array(file)
data_set=data[:,2:3]
a=[]
for i in range(len(data_set)-1):
    if data_set[i,:] or data_set[i+1,:] !=0:
        a.append(data_set[i,:])
data_set=numpy.array(a)

#
scaler=MinMaxScaler()
```

```
data_set=scaler.fit_transform(data_set)

#
train_size = int(len(data_set)*0.8)
test_size = len(data_set) - train_size
train, test = data_set[0:train_size,:], data_set[train_size:len(data_set),:]
print('Train_size=', len(train))
print('Test_size=', len(test))
```

Model1:

```
# convert an array of values into a dataset matrix
def create_dataset(dataset, look_back=1):
  dataX, dataY = [], []
  for i in range(len(dataset)-look_back-1):
    a = dataset[i:(i+look_back), 0]
    dataX.append(a)
    dataY.append(dataset[i + look_back, 0])
  return numpy.array(dataX), numpy.array(dataY)

look_back=1
trainX, trainY= create_dataset(train,look_back)
testX, testY=create_dataset(test,look_back)



# reshape input to be [samples, time steps, features]
trainX = numpy.reshape(trainX, (trainX.shape[0], 1, trainX.shape[1]))
testX = numpy.reshape(testX,(len(testX),1,testX.shape[1]))

#
epochs=100
batch_size=1
```

```
# create and fit the LSTM network
model = Sequential()
model.add(LSTM(5, input_shape=(1, look_back)))
model.add(Dropout(0.01))
model.add(Dense(3,activation='elu'))
model.add(Dense(1))
model.compile(loss='mean_squared_error', optimizer='adam',metrics=['mae','mse'])
history=model.fit(trainX, trainY, epochs=epochs, batch_size=batch_size,
verbose=2,validation_split=0.1)


#model.save('LSTM_Make_predict_final.h5') # saving best models after training
#old_model=load_model('LSTM_Make_predict_final.h5') #call models name

testPredict = model.predict(testX)
testPredict= scaler.inverse_transform(testPredict)
trainPredict = model.predict(trainX)
trainPredict=scaler.inverse_transform(trainPredict)


data_set=scaler.inverse_transform(data_set)
testPredictPlot= numpy.empty_like(data_set)
testPredictPlot[:,:]= numpy.nan
testPredictPlot[len(trainPredict)+(look_back*2)+1:len(data_set)-1,:] = testPredict


trainPredictPlot= numpy.empty_like(data_set)
trainPredictPlot[:,:]=numpy.nan
trainPredictPlot[look_back:len(trainPredict)+look_back,:] = trainPredict

#
nameFig='  Model1_epochs'+str(epochs)
```

```
# plot predicts data with datasets
plt.plot(data_set)
plt.plot(testPredictPlot, color='orange')
plt.plot(trainPredictPlot, color='red')
plt.xlabel('sample')
plt.ylabel('kWh')
plt.grid()
plt.legend(['data_set','Test_Predicted','Train_Predict'],loc='upper left')
plt.title('Make Predictions about PV system in TDTU' + nameFig)
plt.show()
plt.savefig(nameFig)

#
testY=scaler.inverse_transform(testY.reshape(-1,1))
RMSE_test=numpy.sqrt(mean_squared_error(testY,testPredict))
print('RMSE_test'+ nameFig,RMSE_test)
#
trainY=scaler.inverse_transform(trainY.reshape(-1,1))
RMSE_train=numpy.sqrt(mean_squared_error(trainY,trainPredict))
print('RMSE_train'+ nameFig, RMSE_train)
```

Model2:

```
# convert an array of values into a dataset matrix
def create_dataset(dataset, look_back=1):
  dataX, dataY = [], []
  for i in range(len(dataset)-look_back-1):
    a = dataset[i:(i+look_back), 0]
    dataX.append(a)
    dataY.append(dataset[i + look_back, 0])
  return numpy.array(dataX), numpy.array(dataY)

look_back=3
```

```
trainX, trainY= create_dataset(train,look_back)
testX, testY = create_dataset(test,look_back)


# reshape input to be [samples, time steps, features]
trainX = numpy.reshape(trainX, (trainX.shape[0], 1, trainX.shape[1]))
testX = numpy.reshape(testX, (testX.shape[0],1,testX.shape[1]))

#
epochs=100
batch_size=1

# create and fit the LSTM network
model = Sequential()
model.add(LSTM(5, input_shape=(1, look_back)))
model.add(Dense(3,activation='elu'))
model.add(Dense(1,activation='linear'))
model.compile(loss='mean_squared_error', optimizer='adam',metrics=['mae','mse'])
history=model.fit(trainX, trainY, epochs=epochs, batch_size=batch_size,
verbose=2,validation_split=0.1, shuffle=False)


#model.save('LSTM_Make_predict_final.h5') # saving best models after training
#old_model=load_model('LSTM_Make_predict_final.h5') #call models name


testPredict = model.predict(testX)
testPredict= scaler.inverse_transform(testPredict)
trainPredict = model.predict(trainX)
trainPredict=scaler.inverse_transform(trainPredict)
```

```
data_set=scaler.inverse_transform(data_set)
testPredictPlot= numpy.empty_like(data_set)
testPredictPlot[:,:]= numpy.nan
testPredictPlot[len(trainPredict)+(look_back*2)+1:len(data_set)-1,:] = testPredict


trainPredictPlot= numpy.empty_like(data_set)
trainPredictPlot[:,:]=numpy.nan
trainPredictPlot[look_back:len(trainPredict)+look_back,:] = trainPredict

#
nameFig='  Model2_epochs'+str(epochs)
# plot predicts data with datasets
plt.plot(data_set)
plt.plot(testPredictPlot, color='orange')
plt.plot(trainPredictPlot, color='red')
plt.xlabel('sample')
plt.ylabel('kWh')
plt.grid()
plt.legend(['data_set','Test_Predicted','Train_Predict'],loc='upper left')
plt.title('Make Predictions about PV system in TDTU' + nameFig)
plt.show()
plt.savefig(nameFig)

#
testY=scaler.inverse_transform(testY.reshape(-1,1))
RMSE_test=numpy.sqrt(mean_squared_error(testY,testPredict))
print('RMSE_test'+ nameFig,RMSE_test)
#
trainY=scaler.inverse_transform(trainY.reshape(-1,1))
RMSE_train=numpy.sqrt(mean_squared_error(trainY,trainPredict))
print('RMSE_train'+nameFig, RMSE_train)
```

Model3:

```
# convert an array of values into a dataset matrix
def create_dataset(dataset, look_back=1):
  dataX, dataY = [], []
  for i in range(len(dataset)-look_back-1):
    a = dataset[i:(i+look_back), 0]
    dataX.append(a)
    dataY.append(dataset[i + look_back, 0])
  return numpy.array(dataX), numpy.array(dataY)

look_back=5
trainX, trainY= create_dataset(train,look_back)
testX, testY = create_dataset(test,look_back)



# reshape input to be [samples, time steps, features]
trainX = numpy.reshape(trainX, (trainX.shape[0], trainX.shape[1], 1))
testX = numpy.reshape(testX, (testX.shape[0],testX.shape[1],1))
#
epochs=100
batch_size=1

# create and fit the LSTM network

model = Sequential()
model.add(LSTM(5, input_shape=(look_back,1)))
#opt = SGD(learning_rate=0.01, momentum=0.9)
model.add(Dense(3,activation='elu'))
model.add(Dense(1,activation='linear'))
model.compile(loss='mean_squared_error', optimizer='adam',metrics=['mae','mse'])
history=model.fit(trainX, trainY, epochs=epochs, batch_size=batch_size,
```

```
verbose=2,validation_split=0.1, shuffle=False)


#model.save('LSTM_Make_predict_final.h5') # saving best models after training
#old_model=load_model('LSTM_Make_predict_final.h5') #call models name

testPredict = model.predict(testX)
testPredict= scaler.inverse_transform(testPredict)
trainPredict = model.predict(trainX)
trainPredict=scaler.inverse_transform(trainPredict)


data_set=scaler.inverse_transform(data_set)
testPredictPlot= numpy.empty_like(data_set)
testPredictPlot[:,:]= numpy.nan
testPredictPlot[len(trainPredict)+(look_back*2)+1:len(data_set)-1,:] = testPredict


trainPredictPlot= numpy.empty_like(data_set)
trainPredictPlot[:,:]=numpy.nan
trainPredictPlot[look_back:len(trainPredict)+look_back,:] = trainPredict

#
nameFig='  Model3_epochs'+str(epochs)
# plot predicts data with datasets
plt.plot(data_set)
plt.plot(testPredictPlot, color='orange')
plt.plot(trainPredictPlot, color='red')
plt.xlabel('sample')
plt.ylabel('kWh')
plt.grid()
plt.legend(['data_set','Test_Predicted','Train_Predict'],loc='upper left')
```

```
plt.title('Make Predictions about PV system in TDTU' + nameFig)
plt.show()
plt.savefig(nameFig)

#
testY=scaler.inverse_transform(testY.reshape(-1,1))
RMSE_test=numpy.sqrt(mean_squared_error(testY,testPredict))
print('RMSE_test'+ nameFig,RMSE_test)
#
trainY=scaler.inverse_transform(trainY.reshape(-1,1))
RMSE_train=numpy.sqrt(mean_squared_error(trainY,trainPredict))
print('RMSE_train'+nameFig, RMSE_train)
```

Model4:

```
# convert an array of values into a dataset matrix
def create_dataset(dataset, look_back=1):
  dataX, dataY = [], []
  for i in range(len(dataset)-look_back-1):
    a = dataset[i:(i+look_back), 0]
    dataX.append(a)
    dataY.append(dataset[i + look_back, 0])
  return numpy.array(dataX), numpy.array(dataY)

look_back=3
trainX, trainY= create_dataset(train,look_back)
testX, testY = create_dataset(test,look_back)


# reshape input to be [samples, time steps, features]
trainX = numpy.reshape(trainX, (trainX.shape[0], trainX.shape[1], 1))
testX = numpy.reshape(testX, (testX.shape[0],testX.shape[1],1))
#
```

```
epochs = 100
batch_size = 1

# create and fit the LSTM network
model = Sequential()
model.add(LSTM(5, batch_input_shape=(batch_size,look_back,1),stateful=True))
model.add(Dense(3,activation='elu'))
model.add(Dense(1,activation='linear'))
model.compile(loss='mean_squared_error', optimizer='adam',metrics=['mae','mse'])
for i in range(10):
  history=model.fit(trainX, trainY, epochs=epochs, batch_size=batch_size,
verbose=2,validation_split=0.1, shuffle=False)
  model.reset_states()

#model.save('LSTM_Make_predict_final.h5') # saving best models after training
#old_model=load_model('LSTM_Make_predict_final.h5') #call models name


testPredict = model.predict(testX, batch_size=batch_size)
testPredict= scaler.inverse_transform(testPredict)
trainPredict = model.predict(trainX, batch_size=batch_size)
trainPredict=scaler.inverse_transform(trainPredict)


data_set=scaler.inverse_transform(data_set)
testPredictPlot= numpy.empty_like(data_set)
testPredictPlot[:,:]= numpy.nan
testPredictPlot[len(trainPredict)+(look_back*2)+1:len(data_set)-1,:] = testPredict


trainPredictPlot= numpy.empty_like(data_set)
trainPredictPlot[:,:]=numpy.nan
```

```
trainPredictPlot[look_back:len(trainPredict)+look_back,:] = trainPredict

#
nameFig='  Model4_epochs'+str(epochs)
# plot predicts data with datasets
plt.plot(data_set)
plt.plot(testPredictPlot, color='orange')
plt.plot(trainPredictPlot, color='red')
plt.xlabel('sample')
plt.ylabel('kWh')
plt.grid()
plt.legend(['data_set','Test_Predicted','Train_Predict'],loc='upper left')
plt.title('Make Predictions about PV system in TDTU' + nameFig)
plt.show()
plt.savefig(nameFig)

#
testY=scaler.inverse_transform(testY.reshape(-1,1))
RMSE_test=numpy.sqrt(mean_squared_error(testY,testPredict))
print('RMSE_test'+ nameFig,RMSE_test)
#
trainY=scaler.inverse_transform(trainY.reshape(-1,1))
RMSE_train=numpy.sqrt(mean_squared_error(trainY,trainPredict))
print('RMSE_train'+ nameFig, RMSE_train)
```

Model5:

```
data=numpy.array(file)
data_set=data[:,2:3]
a=[]
for i in range(len(data_set)-1):
    if data_set[i,:] or data_set[i+1,:] !=0:
        a.append(data_set[i,:])
```

```
data_set=numpy.array(a)

#
scaler=MinMaxScaler()
data_set=scaler.fit_transform(data_set)

#
train_size = round(len(data_set)*0.8)
test_size = len(data_set) - train_size
train, test = data_set[0:train_size,:], data_set[train_size:len(data_set),:]
print('Train_size=', len(train))
print('Test_size=', len(test))

#create new data test
X_test_new=numpy.reshape(test,(len(test),1,1))



# convert an array of values into a dataset matrix
def create_dataset(dataset, look_back=1):
  dataX, dataY = [], []
  for i in range(len(dataset)-look_back-1):
    a = dataset[i:(i+look_back), 0]
    dataX.append(a)
    dataY.append(dataset[i + look_back, 0])
  return numpy.array(dataX), numpy.array(dataY)

look_back=3
trainX, trainY= create_dataset(train,look_back)
testX, testY = create_dataset(test,look_back)



# reshape input to be [samples, time steps, features]
```

```
trainX = numpy.reshape(trainX, (trainX.shape[0], trainX.shape[1], 1))
testX = numpy.reshape(testX, (testX.shape[0],testX.shape[1],1))
#
epochs=100
batch_size=1
# create and fit the LSTM network

model = Sequential()
model.add(LSTM(5, batch_input_shape=(batch_size, look_back, 1), stateful=True,
return_sequences=True))
model.add(LSTM(5,batch_input_shape=(batch_size, look_back, 1), stateful=True))
model.add(Dense(3,activation='elu'))
model.add(Dense(1,activation='linear'))
model.compile(loss='mean_squared_error', optimizer='adam',metrics=['mae','mse'])
for i in range(10):
  history=model.fit(trainX, trainY, epochs=epochs, batch_size=batch_size, verbose=2
,validation_split=0.1, shuffle=False)
  model.reset_states()

#model.save('LSTM_Make_predict_final.h5') # saving best models after training
#old_model=load_model('LSTM_Make_predict_final.h5') #call models name


testPredict = model.predict(testX, batch_size=batch_size)
testPredict= scaler.inverse_transform(testPredict)
trainPredict = model.predict(trainX, batch_size=batch_size)
trainPredict=scaler.inverse_transform(trainPredict)


data_set=scaler.inverse_transform(data_set)
testPredictPlot= numpy.empty_like(data_set)
testPredictPlot[:,:]= numpy.nan
```

```
testPredictPlot[len(trainPredict)+(look_back*2)+1:len(data_set)-1,:] = testPredict


trainPredictPlot= numpy.empty_like(data_set)
trainPredictPlot[:,:]=numpy.nan
trainPredictPlot[look_back:len(trainPredict)+look_back,:] = trainPredict

#
nameFig='  Model5_epochs'+str(epochs)
# plot predicts data with datasets
plt.plot(data_set)
plt.plot(testPredictPlot, color='orange')
plt.plot(trainPredictPlot, color='red')
plt.xlabel('sample')
plt.ylabel('kWh')
plt.grid()
plt.legend(['data_set','Test_Predicted','Train_Predict'],loc='upper left')
plt.title('Make Predictions about PV system in TDTU' + nameFig)
plt.show()
plt.savefig(nameFig)

#
testY=scaler.inverse_transform(testY.reshape(-1,1))
RMSE_test=numpy.sqrt(mean_squared_error(testY,testPredict))
print('RMSE_test'+ nameFig,RMSE_test)
#
trainY=scaler.inverse_transform(trainY.reshape(-1,1))
RMSE_train=numpy.sqrt(mean_squared_error(trainY,trainPredict))
print('RMSE_train'+ nameFig, RMSE_train)
```

Code dự đoán sản lượng PV ngày tiếp theo:

```
file=pandas.read_excel (r'C:\Users\Bao\Desktop\file so lieu dien mat troi TDTU.xlsx')
data=numpy.array(file)
data=numpy.array(file)
data=numpy.delete(data,len(data)-1,0)
data_set=data[:,2:3]



data_28_04=data_set[len(data_set)-11:len(data_set)]
data_set=data_set[0:len(data_set)-11,:]

#
scaler=MinMaxScaler()
data_set=scaler.fit_transform(data_set)

#

data_for_train=data_set

train = data_for_train
print('Train_size=', len(train))



# convert an array of values into a dataset matrix
def create_dataset(dataset, look_back=1):
  dataX, dataY = [], []
  for i in range(len(dataset)-look_back-1):
    a = dataset[i:(i+look_back), 0]
    dataX.append(a)
    dataY.append(dataset[i + look_back, 0])
  return numpy.array(dataX), numpy.array(dataY)

look_back=3
```

```
trainX, trainY= create_dataset(train,look_back)

#testX, testY = create_dataset(test,look_back)


# reshape input to be [samples, time steps, features]
trainX = numpy.reshape(trainX, (trainX.shape[0], trainX.shape[1], 1))


#
epochs = 100
batch_size = 1

# create and fit the LSTM network
model = Sequential()
model.add(LSTM(5,activation='relu'
,batch_input_shape=(batch_size,look_back,1),stateful=True))
model.add(Dense(3,activation='elu'))
model.add(Dense(1,activation='linear'))
model.compile(loss='mean_squared_error', optimizer='adam',metrics=['mae','mse'])
for i in range(10):
  history=model.fit(trainX, trainY, epochs=epochs, batch_size=batch_size,
verbose=2,validation_split=0.1, shuffle=False)
  model.reset_states()



#testPredict = model.predict(testX, batch_size=batch_size)
#testPredict= scaler.inverse_transform(testPredict)

#model.save('PredictPVAfterday.h5')
```

```
model=load_model('PredictPVAfterday.h5')
# Predict For 11 Hours later
sample, time_step, feature= [1, 3, 1]
x_input=data_set[len(data_set)-3:,:].reshape(1,-1)
x_input=numpy.mean(x_input,axis=0) #return 1D array
temp_x=list(x_input)
print(len(temp_x))
list_predict=[]
for i in range(11):
  if len(temp_x) < 4:
    i=i+1
    x_input=numpy.reshape(x_input,(sample, time_step, feature))
    y_predict=model.predict(x_input, batch_size=batch_size)
    list_predict.append(y_predict[0][0])
    temp_x.append(y_predict[0][0])
    print('Hour {} ----> Input {}'.format(i, temp_x ))
    print('Hour {} ----> Output {}'. format(i, y_predict))
  else:
    i=i+1
    x_input=numpy.array(temp_x[1:])
    print('Hour {} ----> Input {}'.format(i,x_input))
    x_input=numpy.reshape(x_input,(sample, time_step, feature))
    y_predict=model.predict(x_input, batch_size=batch_size)
    print('Hour {} ----> Output {}'. format(i, y_predict))
    list_predict.append(y_predict[0][0])
    temp_x.append(y_predict[0][0])
    temp_x=temp_x[1:]
list_predict=numpy.array(list_predict)
value_predict=scaler.inverse_transform(list_predict.reshape(-1,1)) # value Predict for
11 hour

for i in range(11):
```

```
  if value_predict[i,:]<0:
     value_predict[i,:]=numpy.abs(value_predict[i,:])

time=numpy.arange(1,12,1)
plt.bar(time,numpy.mean(value_predict,axis=1))
plt.plot(time,data_28_04, color='red')
plt.xlabel('Hour')
plt.ylabel('kWh')
plt.legend(['Data_28_04','value_predict'])
plt.title(' Value PV for After day')
plt.grid()
plt.savefig('Value PV for After day.PNG')

table=pandas.DataFrame({'Data_predict': numpy.mean(value_predict,axis=1),
'data_28_04': numpy.mean(data_28_04,axis=1)})
print(table)
```