ĐẠI HỌC ĐÀ NẴNG
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA ĐIỆN

ĐỒ ÁN TỐT NGHIỆP
CAPSTONE PROJECT

NGÀNH: KỸ THUẬT ĐIỀU KHIỂN VÀ TỰ ĐỘNG HÓA

ĐỀ TÀI:

DỰ ĐOÁN VÀ TỐI ƯU HOÁ CÔNG SUẤT ĐIỆN TIÊU THỤ CHO CÔNG ĐOẠN NGHIỀN XI MĂNG

Người hướng dẫn: **TS. NGUYỄN KIM ÁNH**

KS. VƯƠNG HOÀNG LINH

Sinh viên thực hiện:

1. NGUYỄN PHI HÙNG – MSSV: 105200493 – LỚP: 20TDHCLC4

2. NGUYỄN PHÚC NGUYÊN – MSSV: 105200504 – LỚP: 20TDHCLC4

Đà Nẵng, 6/2025

TÓM TẮT

Tên đề tài: Dự đoán và tối ưu hoá công suất điện tiêu thụ cho công đoạn nghiền xi măng

Sinh viên thực hiện: Nguyễn Phi Hùng

Số thẻ SV: 105200493 Lớp: 20TDHCLC4

Sinh viên thực hiện: Nguyễn Phúc Nguyên

Số thẻ SV: 105200504 Lớp: 20TDHCLC4

Trong bối cảnh ngành công nghiệp hiện đại ngày càng đề cao việc sử dụng hiệu quả tài nguyên, đặc biệt là năng lượng, tối ưu hóa điện năng tiêu thụ đang trở thành một trong những yêu cầu trọng tâm nhằm nâng cao hiệu quả vận hành, giảm thiểu chi phí và đáp ứng các tiêu chuẩn về phát triển bền vững. Ngành xi măng, với đặc thù tiêu hao năng lượng lớn trong quá trình sản xuất, đặc biệt ở công đoạn nghiền, là một trong những lĩnh vực cần được chú trọng nghiên cứu và cải tiến. Đây là công đoạn không chỉ ảnh hưởng đến chất lượng sản phẩm đầu ra mà còn đóng vai trò quyết định trong tổng mức tiêu thụ năng lượng của toàn nhà máy. Việc triển khai các giải pháp công nghệ hỗ trợ giám sát, dự đoán và tối ưu hóa điện năng trong công đoạn này không chỉ mang lại lợi ích kinh tế rõ rệt mà còn góp phần giảm áp lực lên hệ thống điện quốc gia và giảm thiểu tác động tiêu cực đến môi trường.

Để giảm thiểu lãng phí năng lượng và góp phần bảo vệ môi trường, đề tài này được triển khai với mục tiêu xây dựng một mô hình thông minh kết hợp giữa mạng nơ-ron tích chập, mạng bộ nhớ dài-ngắn hạn và thuật toán lấy mẫu Latin Hypercube . Mô hình kết hợp này không chỉ có khả năng dự đoán chính xác điện năng tiêu thụ theo chuỗi thời gian, mà còn tìm ra các thông số tối ưu vận hành của hệ thống, từ đó nâng cao hiệu quả điều hành công đoạn nghiền xi măng. Kết quả từ mô hình mang lại tiềm năng ứng dụng thực tiễn cao trong việc tiết kiệm điện năng một cách chủ động, đồng thời góp phần thúc đẩy quá trình chuyển đổi xanh, phát triển bền vững và thân thiện với môi trường cho ngành xi măng nói riêng và toàn bộ lĩnh vực sản xuất công nghiệp nói chung.

ĐẠI HỌC ĐÀ NẴNG
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA ĐIỆN

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc lập - Tự do - Hạnh phúc

NHIỆM VỤ ĐỒ ÁN TỐT NGHIỆP

TT	Họ tên sinh viên	Số thẻ SV	Lớp	Ngành
1	Nguyễn Phi Hùng	105200493	20TDHCLC4	Kỹ thuật điều khiển và tự động hóa
2	Nguyễn Phúc Nguyên	105200504	20TDHCLC4	Kỹ thuật điều khiển và tự động hóa

1. *Tên đề tài đồ án:*

Dự đoán và tối ưu hoá công suất điện tiêu thụ cho công đoạn nghiền xi măng

2. *Đề tài thuộc diện:* ☐ *Có ký kết thỏa thuận sở hữu trí tuệ đối với kết quả thực hiện*

3. *Các số liệu và dữ liệu ban đầu:*

Dữ liệu được trích xuất từ cơ sở dữ liệu của hệ thống quản lý sản xuất thông minh của nhà máy xi măng ở miền Trung Viện Nam.

4. *Nội dung các phần thuyết minh và tính toán:*

a. *Phần chung:*

TT	Họ tên sinh viên	Nội dung
1	Nguyễn Phi Hùng	Thu thập và tính toán dữ liệu từ hệ thống XHQ của các thiết bị ở công đoạn nghiền xi măng. Đưa ra giải pháp dự đoán và tối ưu lượng điện tiêu thụ
2	Nguyễn Phúc Nguyên	

b. *Phần riêng:*

TT	Họ tên sinh viên	Nội dung
1	Nguyễn Phi Hùng	Ứng dụng thuật toán Long Short-Term Memory để dự đoán chuỗi thời gian cho điện năng tiêu thụ
2	Nguyễn Phúc Nguyên	Ứng dụng thuật toán lấy mẫu Latin Hypercube tìm ra các thông số giúp tối ưu hoá công suất điện tiêu thụ

5. *Các bản vẽ, đồ thị (ghi rõ các loại và kích thước bản vẽ):*

Bản vẽ nguyên lý hoạt động của công đoạn nghiền xi măng (A1: 594 mm × 841 mm).

6. Người hướng dẫn

Họ tên người hướng dẫn:	*Phần/ Nội dung:*
TS. Nguyễn Kim Ánh	Toàn bộ đồ án
KS. Vương Hoàng Linh	

7. Ngày giao nhiệm vụ đồ án: 24/2/2025

8. Ngày hoàn thành đồ án: 15/6/2025

Đà Nẵng, ngày 15 tháng 06 năm 2025

Trưởng Bộ môn Tự động hóa **Người hướng dẫn 1** **Người hướng dẫn 2**

TS. Giáp Quang Huy **TS. Nguyễn Kim Ánh** **KS. Vương Hoàng Linh**

ĐẠI HỌC ĐÀ NẴNG
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA ĐIỆN

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc lập - Tự do - Hạnh phúc

PHIẾU KIỂM SOÁT TIẾN ĐỘ LÀM ĐỒ ÁN TỐT NGHIỆP

(Phiếu dành cho người hướng dẫn/sinh viên)

Họ tên sinh viên 1: Nguyễn Phi Hùng Số thẻ SV: 105200493

Họ tên sinh viên 2: Nguyễn Phúc Nguyên Số thẻ SV: 105200504

Tên đề tài ĐATN: **Dự đoán và tối ưu hoá công suất điện tiêu thụ cho công đoạn nghiền xi măng**

Họ tên người HD1: TS. Nguyễn Kim Ánh Đơn vị: Khoa Điện, ĐHBK Đà Nẵng

Họ tên người HD2: KS. Vương Hoàng Linh Đơn vị: Công ty TNHH Kỹ thuật Công Nghệ Điện Tự Động Biển Đông

Tuần	**Ngày**	**Khối lượng**		**GVHD ký tên**
		Đã thực hiện (%)	**Tiếp tục thực hiện (%)**	
1	24/2- 2/3	Lựa chọn và nghiên cứu đề tài (100%).	(0%)	
2	3/3-9/3	Tìm hiểu tổng quan hệ thống nghiền xi măng (100%).	(0%)	
3	10/3-16/3	Tìm hiểu về các yếu tố ảnh hưởng tới tiêu thụ điện trong công đoạn nghiền (100%).	(0%)	
4	17/3-23/3	Duyệt lần 1: Đánh giá khối lượng hoàn thành _____ % : Được tiếp tục làm ĐATN □ Không tiếp tục thực hiện ĐATN □		
5	24/3-30/3	Tìm hiểu về các phương pháp dự đoán và tối ưu điện năng (100%).	(0%)	

6	31/3-6/4	Xây dựng mô hình kết hợp các thuật toán và quy trình công đoạn nghiền xi măng (100%).	(0%)	
7	7/4-13/4	Lập trình trên phần mềm Jupyter notebook (50%).	(50%)	
8	14/4-20/4	Duyệt lần 2: Đánh giá khối lượng hoàn thành _____ % : Được tiếp tục làm ĐATN □ Không tiếp tục thực hiện ĐATN □		
9	21/4-27/4	Mô phỏng với tập dữ liệu tại nhà máy ở Trung Quốc (80%).	(20%)	
10	28/4-4/5	Nhận dữ liệu từ nhà máy xi măng ở Việt Nam và xử lí dữ liệu (100%).	(0%)	
11	5/5-11/5	Hiệu chỉnh mô hình phù hợp với số liệu thực tế nhận được (50%).	(50%)	
12	12/5-18/5	Duyệt lần 3: Đánh giá khối lượng hoàn thành _____ % : Được tiếp tục làm ĐATN □ Không tiếp tục thực hiện ĐATN □		
13	19/5-25/5	Hiệu chỉnh mô hình phù hợp với số liệu thực tế nhận được (90%).	(10%)	
14	26/5-1/6	Tiến hành mô phỏng kiểm nghiệm, đánh giá kết quả (90%).	(10%)	
15	2/6-8/6	Hoàn thiện đồ án, báo cáo (70%).	(30%)	

LỜI NÓI ĐẦU VÀ CẢM ƠN

Sau năm năm học tập và rèn luyện tại Trường Đại học Bách Khoa – Đại học Đà Nẵng, đồ án tốt nghiệp này là một cột mốc quan trọng, đánh dấu sự trưởng thành và hoàn thành chặng đường học tập của một sinh viên đại học. Đây không chỉ là sản phẩm học thuật mà còn là minh chứng cho quá trình tích lũy kiến thức, rèn luyện tư duy và khả năng áp dụng lý thuyết vào thực tiễn.

Trong suốt thời gian thực hiện đồ án với đề tài "Dự đoán và tối ưu hóa công suất điện tiêu thụ cho công đoạn nghiền xi măng", nhóm em đã nhận được sự hỗ trợ tận tình, những lời chỉ dẫn quý báu cũng như động viên kịp thời từ quý thầy cô, bạn bè và gia đình. Đặc biệt, sự hướng dẫn sâu sát của thầy Nguyễn Kim Ánh đã giúp nhóm định hướng rõ ràng và vượt qua nhiều khó khăn trong quá trình thực hiện.

Đồ án này không chỉ là cơ hội để nhóm em tổng hợp, kiểm chứng những kiến thức đã học mà còn là dịp rèn luyện năng lực tư duy phản biện, phân tích và giải quyết vấn đề trong bối cảnh một dự án thực tế. Qua đó, nhóm nhận thức rõ hơn về tầm quan trọng của việc liên kết giữa lý thuyết và ứng dụng, giữa học thuật và thực tiễn sản xuất công nghiệp.

Nhóm xin bày tỏ lòng biết ơn sâu sắc đến TS. Nguyễn Kim Ánh, giảng viên Bộ môn Kỹ thuật điều khiển và Tự động hóa, cùng KS. Vương Hoàng Linh, người đã tận tình hướng dẫn, định hướng và hỗ trợ nhóm trong suốt quá trình thực hiện khóa luận.

Chúng em cũng xin gửi lời cảm ơn chân thành đến toàn thể quý thầy cô Trường Đại học Bách Khoa, đặc biệt là các thầy cô trong Bộ môn Tự động hóa, những người đã truyền đạt kiến thức nền tảng lẫn chuyên ngành, tạo điều kiện tốt nhất để nhóm có thể học tập và trưởng thành về mọi mặt.

Cuối cùng, nhóm xin tri ân đến gia đình và bạn bè thân thiết – những người luôn đồng hành, động viên và tiếp thêm động lực để nhóm vượt qua những áp lực trong quá trình học tập và thực hiện đồ án.

Do thời gian có hạn và kinh nghiệm thực tiễn còn hạn chế, chắc chắn đồ án vẫn còn tồn tại những thiếu sót nhất định. Nhóm em rất mong nhận được sự góp ý, chỉ bảo từ quý thầy cô để có thể hoàn thiện hơn, phục vụ tốt hơn cho công việc và học tập trong tương lai.

LỜI CAM ĐOAN LIÊM CHÍNH HỌC THUẬT

Nhóm xin cam đoan báo cáo Đồ Án Tốt Nghiệp với đề tài là "**Dự đoán và tối ưu hoá công suất điện tiêu thụ cho công đoạn nghiền xi măng**" là kết quả tìm hiểu, nghiên cứu của nhóm dưới sự dẫn dắt của TS. Nguyễn Kim Ánh và KS. Vương Hoàng Linh (Công ty TNHH Kỹ thuật Công Nghệ Điện Tự Động Biển Đông).

Nghiên cứu đề tài và nội dung đồ án là sản phẩm mà nhóm đã được học tập tại Trường cũng như tự tìm hiểu nghiên cứu và thời gian đi thực tập tại công ty. Các số liệu và kết quả là hoàn toàn trung thực. Nếu có vấn đề xảy ra, nhóm xin chịu trách nhiệm và nhận mọi kỷ luật.

Sinh viên thực hiện 1	**Sinh viên thực hiện 2**
Nguyễn Phi Hùng	**Nguyễn Phúc Nguyên**

MỤC LỤC

DANH SÁCH CÁC BẢNG, HÌNH VẼ

DANH SÁCH CÁC KÝ HIỆU, CHỮ VIẾT TẮT

Chữ viết tắt	Viết đầy đủ	Ý nghĩa
AI	Artificial Intelligence	Trí tuệ nhân tạo
ANN	Artificial Neural Network	Mạng nơ-ron nhân tạo
ARIMA	Autoregressive Integrated Moving Average	Mô hình chuỗi thời gian tự hồi quy, sai phân & trung bình trượt
BGD	Batch Gradient Descent	Tối ưu dùng toàn bộ tập dữ liệu mỗi lượt
BO	Bayesian Optimization	Tối ưu dựa trên mô hình xác suất
CKP	Cement Kiln Pre-grinder	Máy nghiền đứng CKP
CNN	Convolutional Neural Network	Mạng nơ-ron tích chập
DCS	Distributed Control System	Hệ thống điều khiển phân tán
DE	Differential Evolution	Thuật toán tiến hóa vi phân
GD	Gradient Descent	Thuật toán giảm gradient
IoT	Internet of Things	Internet kết nối vạn vật
KPI	Key Performance Indicator	Chỉ số đánh giá hiệu suất hoạt động hoặc mục tiêu
LHS	Latin Hypercube Sampling	Thuật toán lấy mẫu Latin Hypercube
LSTM	Long Short-Term Memory	Bộ nhớ dài-ngắn hạn
MAE	Mean Absolute Error	Sai số tuyệt đối trung bình – đo mức lệch bình quân
MAPE	Mean Absolute Percentage Error	Sai số phần trăm tuyệt đối trung bình
ME	Mean Error	Trung bình sai số có dấu

NARX	Nonlinear AutoRegressive model with eXogenous inputs	Mô hình phi tuyến dùng giá trị quá khứ & biến ngoại sinh
OFR	Ore Feed Rate	Tốc độ cấp liệu
R^2	Coefficient of Determination	Hệ số xác định
RMSE	Root Mean Square Error	Sai số căn bình phương trung bình
RNN	Recurrent Neural Network	Mạng nơ-ron hồi tiếp
SCADA	Supervisory Control & Data Acquisition	Hệ thống giám sát, thu thập dữ liệu & điều khiển từ xa
VRM	Vertical Roller Mill	Máy nghiền con lăn đứng
WFR	Water Feed Rate	Tốc độ cấp nước

MỞ ĐẦU

1. Lý do chọn đề tài:

Ngành công nghiệp xi măng là một trong những lĩnh vực tiêu thụ năng lượng điện lớn nhất tại Việt Nam. Trong đó, công đoạn nghiền đóng vai trò quan trọng và chiếm tỷ trọng điện năng tiêu thụ lớn nhất trong toàn bộ dây chuyền sản xuất. Tuy nhiên, hiện nay phần lớn các nhà máy xi măng trong nước vẫn vận hành theo kinh nghiệm hoặc theo các thông số thiết lập sẵn từ nhà sản xuất, thiếu các giải pháp dự đoán và tối ưu hiệu quả năng lượng một cách có hệ thống.

Trong bối cảnh nguồn năng lượng truyền thống ngày càng cạn kiệt, giá điện tăng cao và áp lực từ các cam kết giảm phát thải khí nhà kính, yêu cầu tiết kiệm và sử dụng năng lượng hiệu quả ngày càng trở nên cấp thiết. Đặc biệt, các nghiên cứu gần đây cho thấy việc áp dụng các mô hình học máy và thuật toán tối ưu hiện đại có thể giúp giảm đáng kể lượng điện tiêu thụ trong công đoạn nghiền mà không làm ảnh hưởng đến chất lượng sản phẩm.

Từ thực tiễn đó, nhóm thực hiện đề tài "Dự đoán và tối ưu hoá công suất điện tiêu thụ cho công đoạn nghiền xi măng" với mong muốn xây dựng một giải pháp có khả năng dự đoán mức điện tiêu thụ và tự động đề xuất các thông số vận hành tối ưu. Đây là hướng tiếp cận phù hợp với xu thế chuyển đổi số trong công nghiệp, giúp tiết kiệm năng lượng, nâng cao hiệu quả sản xuất và góp phần bảo vệ môi trường.

2. Mục tiêu đề tài

Đề tài hướng đến mục tiêu nghiên cứu quy trình nghiền trong sản xuất xi măng, đồng thời phân tích các yếu tố ảnh hưởng đến mức tiêu thụ điện năng trong công đoạn này. Trên cơ sở đó, nhóm xây dựng một mô hình dự đoán chính xác lượng điện tiêu thụ dựa vào dữ liệu vận hành và đặc tính nguyên liệu đầu vào. Song song với đó, một thuật toán tối ưu hóa sẽ được phát triển nhằm xác định các tổ hợp thông số vận hành giúp giảm thiểu điện năng mà vẫn duy trì chất lượng sản phẩm đầu ra. Mô hình sẽ được kiểm nghiệm bằng dữ liệu thực tế tại nhà máy xi măng để đánh giá hiệu quả và khả năng ứng dụng. Ngoài ra, đề tài còn hướng đến việc xây dựng một hệ thống hỗ trợ ra quyết định cho đội ngũ kỹ thuật trong quá trình điều chỉnh và kiểm soát thông số vận hành, từ đó góp phần nâng cao hiệu quả sử dụng năng lượng và tối ưu hóa toàn bộ quy trình sản xuất.

3. Phạm vi đề tài

Đề tài tập trung nghiên cứu quy trình nghiền clinker và phụ gia trong các dây chuyền sản xuất xi măng tại Việt Nam, nơi tiêu thụ điện năng lớn và có nhiều tiềm năng tối ưu hóa. Đối tượng nghiên cứu là các thông số vận hành ảnh hưởng đến điện năng tiêu thụ như tốc độ cấp liệu, tải máy nghiền, tốc độ cấp nước v.v., kết hợp với các mô hình dự báo và tối ưu hóa hiện đại như mạng nơ-ron nhân tạo, Long Short-Term Memory và Latin Hypercube Sampling. Về phạm vi thời gian, đề tài được triển khai từ tháng 2 đến tháng 6 năm 2025, bao gồm các giai đoạn khảo sát, thu thập dữ liệu, xây dựng mô hình, mô phỏng, đánh giá kết quả và đề xuất giải pháp ứng dụng thực tiễn.

4. Nội dung thực hiện

Nội dung của đề tài được chia thành 5 phần chính tương ứng với 4 chương và phần kết luận trong báo cáo:

Chương 1: Tổng quan về quy trình nghiền xi măng

Chương 2: Phương pháp luận

Chương 3: Mô hình dự đoán và tối ưu năng lượng tiêu thụ của công đoạn nghiền xi măng

Chương 4: Thử nghiệm và đánh giá mô hình

Chương 1: TỔNG QUAN VỀ HỆ THỐNG NGHIỀN XI MĂNG

1.1. Giới thiệu

Ngành công nghiệp xi măng đóng vai trò then chốt trong sự phát triển kinh tế – xã hội của mỗi quốc gia. Xi măng là vật liệu kết dính thủy lực cơ bản, không thể thiếu trong xây dựng nhà ở, hạ tầng giao thông, công trình công nghiệp và quốc phòng. Sự phát triển mạnh mẽ của ngành này phản ánh nhu cầu ngày càng tăng về cơ sở hạ tầng, đặc biệt tại các quốc gia đang phát triển và mới nổi.

Hình 1.1 Nhà máy xi măng

Một nhà máy xi măng hiện đại vận hành theo chuỗi công đoạn khép kín, bắt đầu từ khai thác nguyên liệu (đá vôi, đất sét), tiếp theo là nghiền, đồng nhất hóa, tiền nung, nung clinker ở nhiệt độ khoảng 1450°C, làm nguội, nghiền mịn và đóng bao thành phẩm. Trong toàn bộ quy trình, năng lượng tiêu thụ chủ yếu dưới dạng nhiệt (cho lò nung) và điện (cho các thiết bị quay, bơm, máy nghiền...).

Trong đó, công đoạn nghiền xi măng đóng vai trò quan trọng trong việc quyết định chất lượng sản phẩm cuối cùng, đồng thời cũng là một trong những khâu tiêu thụ điện năng lớn nhất trong toàn bộ dây chuyền. Máy nghiền xi măng thường sử dụng dạng nghiền bi, làm việc với vật liệu có độ cứng cao và yêu cầu độ mịn lớn, dẫn đến mức tiêu thụ điện rất đáng kể. Theo thống kê, công đoạn này có thể chiếm tới 30–40% tổng lượng điện sử dụng trong toàn nhà máy theo nghiên cứu của Worrell và các cộng sự (2008)

[1]. Ngoài ra, hiệu suất nghiền còn bị ảnh hưởng bởi tổn hao cơ học, rung động và chế độ vận hành chưa tối ưu được nêu lên nghiên cứu của Madlool và các cộng sự (2011) [2].

Trong bối cảnh giá điện ngày càng tăng, áp lực cắt giảm phát thải CO_2 theo các cam kết khí hậu toàn cầu như COP26, và xu thế chuyển đổi sang mô hình phát triển xanh thế nên việc tối ưu hóa vận hành máy nghiền xi măng để tiết kiệm điện trở thành yêu cầu cấp thiết. Không chỉ nhằm giảm chi phí sản xuất, đây còn là yếu tố sống còn để đảm bảo cạnh tranh và phát triển bền vững cho doanh nghiệp.

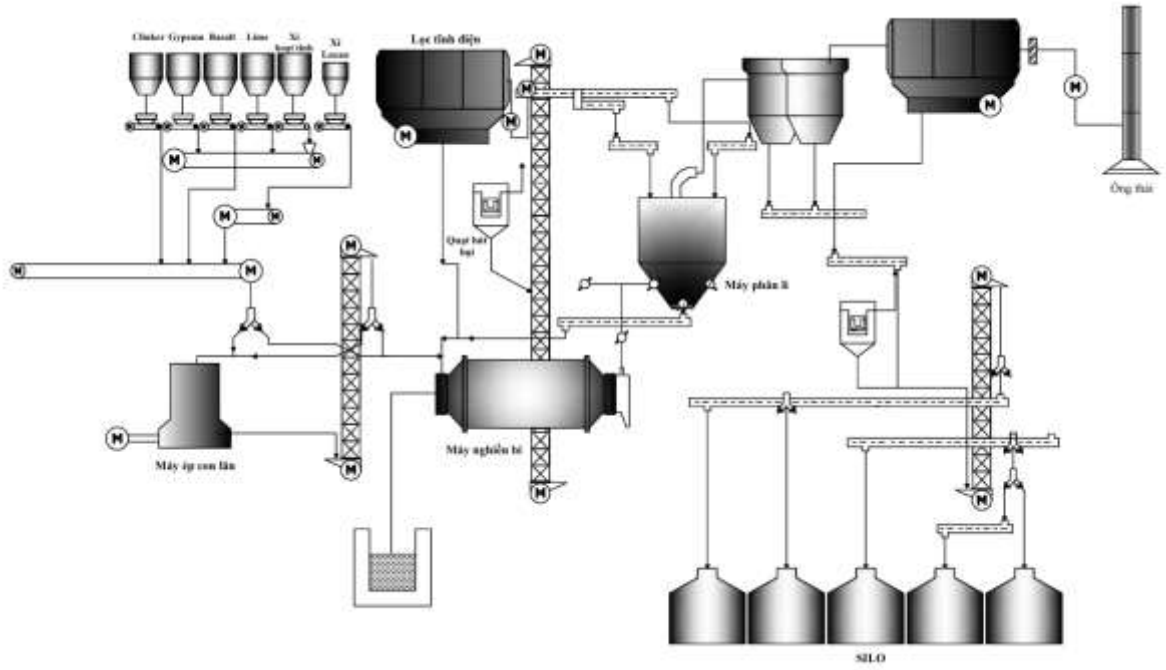
Gần đây, nhiều nghiên cứu đã chỉ ra tiềm năng lớn của việc ứng dụng các phương pháp hiện đại như trí tuệ nhân tạo (AI), học máy, học sâu (deep learning), kết hợp với các thuật toán tối ưu hóa (tiến hóa, bầy đàn, tối ưu hóa Bayes...) để nâng cao hiệu suất vận hành máy nghiền.

Chẳng hạn, nghiên cứu của Sivanandam và các cộng sự (2017) [3] đã chứng minh rằng việc ứng dụng mạng nơ-ron nhân tạo kết hợp với các thuật toán tối ưu trong mô hình hóa và điều khiển quá trình nghiền xi măng có thể giúp giảm đến 5–10% mức tiêu thụ điện năng mà vẫn đảm bảo chất lượng sản phẩm đầu ra. Tương tự, trong tổng quan hệ thống nhà máy xi măng của Oguntola và các cộng sự (2024) [4], các tác giả đã chỉ ra rằng việc triển khai các hệ thống điều khiển thông minh ứng dụng trí tuệ nhân tạo tại các nhà máy xi măng không chỉ giúp tiết kiệm hàng triệu kWh điện mỗi năm mà còn nâng cao độ ổn định vận hành và giảm phát thải khí nhà kính. Những kết quả này cho thấy rõ vai trò quan trọng và tính khả thi cao của việc tối ưu hóa tiêu thụ năng lượng tại công đoạn nghiền xi măng thông qua các giải pháp công nghệ tiên tiến.

1.2. Cấu trúc và nguyên lí hoạt động của công đoạn nghiền xi măng

Trong chuỗi sản xuất xi măng, công đoạn nghiền đóng vai trò then chốt, ảnh hưởng trực tiếp đến chất lượng sản phẩm cuối cùng cũng như hiệu quả sử dụng năng lượng của toàn bộ dây chuyền. Như đã trình bày ở mục 1.1, công đoạn này tiêu thụ lượng điện năng lớn nhất và cũng là mắt xích quyết định đến độ mịn, tính chất cơ lý và độ ổn định của xi măng thành phẩm.

Sau khi clinker được hình thành từ quá trình nung và làm nguội, chúng được đưa đến hệ thống nghiền – nơi diễn ra quá trình nghiền mịn kết hợp với thạch cao và các phụ gia khoáng như đá vôi, xỉ lò cao,... . Hệ thống nghiền hiện đại thường bao gồm các thiết bị chính như: phễu chứa định lượng, máy nghiền đứng CKP, máy nghiền bi, bộ phân ly, quạt hút, hệ thống lọc bụi và các silo chứa sản phẩm.



Hình 1.2 Nguyên lý hoạt động của công đoạn nghiền xi măng

Quy trình bắt đầu từ các phễu chứa Clinker, Gypsum, Basalt, Lime, Xi hoạt tính, Xi Locao mỗi phễu được trang bị băng tải định lượng giúp kiểm soát chính xác tỷ lệ phối trộn nguyên liệu. Hỗn hợp sau định lượng được vận chuyển đến cụm cấp liệu, một phần đi qua máy nghiền đứng CKP để giảm kích thước hạt, phần còn lại có thể đi thẳng vào máy nghiền bi.

Máy nghiền bi là thiết bị trung tâm trong hệ thống, nơi nguyên liệu được nghiền mịn dưới tác động quay của tang trống chứa các viên bi thép. Trong quá trình này, hệ thống tuần hoàn khí nóng được tích hợp nhằm kiểm soát nhiệt độ, cùng với hệ thống bôi trơn đảm bảo vận hành ổn định và kéo dài tuổi thọ thiết bị.

Sau khi ra khỏi máy nghiền bi, hỗn hợp được đưa đến bộ phân ly. Tại đây, sản phẩm được phân loại: phần đạt độ mịn sẽ được thu nhận, còn phần thô sẽ được hoàn lưu về máy nghiền để tiếp tục xử lý. Quá trình này được hỗ trợ bởi hệ thống quạt hút ID Fan và các van điều tiết nhằm tối ưu hóa dòng khí và hiệu suất phân ly.

Khí mang theo bụi mịn sẽ đi qua hệ thống lọc bụi (lọc túi hoặc lọc tĩnh điện) để giữ lại các hạt xi măng, đảm bảo khí thải ra môi trường đạt tiêu chuẩn. Xi măng thành phẩm sau đó được vận chuyển bằng hệ thống khí nén hoặc gầu nâng đến các silo chứa, sẵn sàng cho quá trình đóng bao hoặc xuất hàng.

Vai trò và chức năng của từng thiết bị chính:

- Phễu chứa và băng tải định lượng: Chứa và cấp liệu chính xác theo tỷ lệ phối trộn thiết kế.
- Máy nghiền đứng CKP: Thiết bị nghiền sơ bộ hiệu quả, hoạt động theo nguyên lý ép nghiền giữa con lăn và bàn quay, giúp giảm tải cho máy nghiền bi và tiết kiệm năng lượng.
- Máy nghiền bi: Thiết bị nghiền mịn chủ lực, hoạt động theo nguyên lý va đập và ma sát của bi thép trong tang trống quay, đảm bảo độ mịn và phân bố kích thước hạt đồng đều.
- Bộ phân ly: Tách sản phẩm đạt chuẩn khỏi phần thô, có thể điều chỉnh để kiểm soát độ mịn sản phẩm.
- Quạt hút ID Fan và van điều tiết: Duy trì lưu thông khí và hỗ trợ quá trình phân ly hiệu quả.
- Hệ thống lọc bụi (tĩnh điện hoặc túi lọc): Loại bỏ bụi xi măng khỏi khí thải, đảm bảo yêu cầu về môi trường.
- Silo chứa: Lưu trữ xi măng thành phẩm trước khi đóng gói hoặc vận chuyển.

Có thể thấy, công đoạn nghiền không chỉ đảm nhiệm chức năng hoàn thiện sản phẩm mà còn đóng vai trò trung tâm trong việc đảm bảo chất lượng, đồng thời là cơ hội lớn để tối ưu hóa tiêu thụ năng lượng trong sản xuất xi măng.

1.3. Tiêu thụ điện và các yếu tố ảnh hưởng đến tiêu thụ điện trong công đoạn nghiền xi măng

Trong bối cảnh ngành công nghiệp xi măng ngày càng chú trọng tới hiệu quả sử dụng năng lượng và phát triển bền vững, việc kiểm soát tiêu thụ điện năng trong công đoạn nghiền xi măng trở nên đặc biệt quan trọng. Như đã đề cập ở các mục trước, đây là công đoạn tiêu tốn nhiều điện năng nhất trong toàn bộ dây chuyền sản xuất, chiếm rất lớn tổng điện năng tiêu thụ của nhà máy theo nghiên cứu của Genç và các cộng sự (2016) [5]. Vì vậy, việc hiểu rõ các yếu tố ảnh hưởng đến tiêu thụ điện không chỉ giúp doanh nghiệp tiết kiệm chi phí vận hành mà còn góp phần giảm phát thải khí nhà kính và nâng cao năng lực cạnh tranh.

Có thể thấy rằng, hiệu quả hoạt động của từng thiết bị cũng như sự phối hợp giữa chúng đóng vai trò then chốt đối với mức tiêu thụ điện của toàn hệ thống. Bên cạnh các đặc tính và thông số kỹ thuật ban đầu của thiết bị, nhiều yếu tố vận hành và điều kiện thực tế tại hiện trường cũng tác động trực tiếp hoặc gián tiếp đến lượng điện năng sử

dụng. Việc nhận diện và kiểm soát những yếu tố này một cách hợp lý sẽ giúp tối ưu hóa hiệu suất vận hành, duy trì chất lượng sản phẩm và giảm thiểu lãng phí năng lượng.

Dưới đây là các yếu tố chính ảnh hưởng đến tiêu thụ điện trong công đoạn nghiền xi măng:

- Tốc độ cấp liệu: Là yếu tố quan trọng ảnh hưởng đến công suất hoạt động của máy nghiền. Tốc độ cấp liệu cao khiến máy phải làm việc mạnh hơn để xử lý khối lượng vật liệu lớn, từ đó làm tăng điện năng tiêu thụ. Ngược lại, tốc độ quá thấp dẫn đến máy vận hành dưới tải, gây lãng phí năng lượng.
- Lưu lượng hồi về nghiền bi: Phản ánh tỷ lệ vật liệu chưa đạt độ mịn quay lại máy nghiền. Lưu lượng hồi cao kéo dài thời gian nghiền và chu trình tuần hoàn, làm tăng số lần vật liệu tiếp xúc với bi nghiền và tiêu thụ nhiều điện năng hơn để đạt chất lượng đầu ra mong muốn.
- Tải của máy nghiền bi: Là chỉ số trực tiếp thể hiện mức tiêu hao điện năng của máy nghiền. Khi tải tăng, máy phải vận hành ở công suất cao, dẫn đến điện năng tiêu thụ cũng tăng lên tương ứng. Tải không ổn định còn có thể ảnh hưởng đến chất lượng nghiền.
- Độ ồn của ngăn 1 máy nghiền bi: Độ ồn trong quá trình nghiền thường là dấu hiệu cho thấy mức độ lấp đầy nguyên liệu trong máy nghiền. Nếu độ ồn quá cao, mức độ lắp đầy nguyên liệu đang thấp và gây ra mày nghiền hoạt động không ổn định, công suất tiêu thụ điện tăng lên.
- Áp suất sau máy nghiền bi: Phản ánh khả năng thông thoáng của hệ thống nghiền. Áp suất quá cao hoặc quá thấp đều có thể gây cản trở lưu thông vật liệu, ảnh hưởng đến hiệu suất phân ly và làm tăng điện năng tiêu thụ do sự quá tải hoặc tắc nghẽn cục bộ.
- Tải của máy nghiền đứng CKP: Tương tự như máy nghiền bi, tải trọng của CKP càng lớn thì điện năng tiêu thụ càng cao. Việc kiểm soát tải máy hợp lý giúp duy trì hiệu suất nghiền mà không gây lãng phí điện năng.
- Độ rung của máy nghiền đứng CKP: Rung động lớn thường là dấu hiệu của sự mất cân bằng hoặc vận hành không ổn định, dẫn đến tăng tổn hao năng lượng do ma sát và ảnh hưởng đến tuổi thọ thiết bị.
- Tốc độ cấp nước: Nước đóng vai trò làm mát và điều hòa quá trình nghiền. Nếu cấp nước không hợp lý (quá nhiều hoặc quá ít) có thể gây ra hiện tượng quá nhiệt,

kết dính vật liệu hoặc tắc nghẽn, dẫn đến tăng điện năng tiêu thụ và giảm hiệu suất nghiền.

- Nhiệt độ bệ động cơ: Là một chỉ số quan trọng trong giám sát tình trạng làm việc của máy nghiền. Nhiệt độ tăng cao bất thường có thể cho thấy động cơ đang vận hành quá tải hoặc hệ thống làm mát không hiệu quả, dẫn đến tiêu thụ điện vượt mức bình thường.
- Sản lượng đầu ra: Là thước đo tổng thể về hiệu suất của hệ thống nghiền. Khi sản lượng tăng, máy nghiền phải làm việc nhiều hơn, kéo theo mức tiêu thụ điện tăng. Tuy nhiên, cần tối ưu hóa để đạt được sản lượng cao nhất với lượng điện năng tiêu thụ thấp nhất.

Việc giám sát và điều chỉnh kịp thời các yếu tố trên sẽ giúp nâng cao hiệu quả vận hành của hệ thống nghiền, đồng thời đóng góp quan trọng vào mục tiêu tiết kiệm năng lượng và phát triển bền vững của nhà máy xi măng.

1.4. Kết luận

Trong chương này, báo cáo đã trình bày tổng quan về vai trò, cấu trúc và nguyên lý hoạt động của công đoạn nghiền trong quá trình sản xuất xi măng – một trong những công đoạn tiêu thụ điện năng lớn nhất trong toàn bộ dây chuyền. Việc hiểu rõ cấu trúc dây chuyền, chức năng của từng thiết bị (như máy nghiền bi, máy nghiền đứng CKP, máy phân ly…) cũng như cơ chế phối hợp giữa chúng là cơ sở quan trọng để phân tích và đánh giá hiệu quả năng lượng của toàn hệ thống.

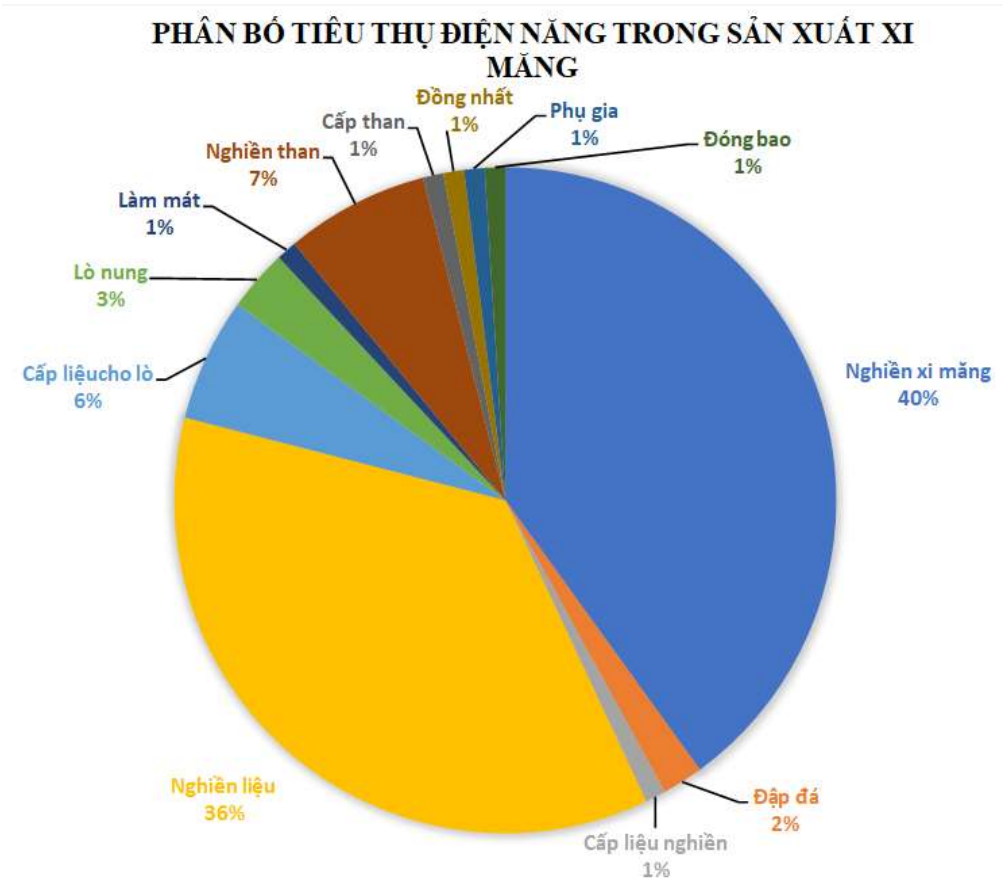
Bên cạnh đó, chương cũng đã làm rõ các yếu tố ảnh hưởng đến tiêu thụ điện trong công đoạn nghiền xi măng. Không chỉ các thông số kỹ thuật của thiết bị mà còn nhiều yếu tố vận hành như tốc độ cấp liệu, lưu lượng hồi về, độ rung, nhiệt độ, áp suất hay tải trọng… đều có tác động trực tiếp đến lượng điện năng tiêu thụ. Việc kiểm soát tốt các yếu tố này sẽ góp phần nâng cao hiệu quả sử dụng năng lượng, giảm chi phí vận hành và hạn chế phát thải khí nhà kính những mục tiêu cốt lõi trong định hướng phát triển bền vững của ngành công nghiệp xi măng hiện nay.

Từ những cơ sở lý thuyết và thực tiễn nêu trên, chương sau sẽ tập trung vào việc ứng dụng các mô hình học máy nhằm dự đoán và tối ưu hóa tiêu thụ điện năng trong công đoạn nghiền xi măng, hướng tới một giải pháp thông minh, hiệu quả và thân thiện với môi trường hơn trong sản xuất công nghiệp.

Chương 2: PHƯƠNG PHÁP LUẬN

2.1. Quá trình tiêu thụ điện năng của công đoạn nghiền xi măng

Ngành công nghiệp xi măng là một trong những ngành sử dụng nhiều năng lượng nhất trong lĩnh vực công nghiệp nặng, với điện năng đóng vai trò chủ chốt trong toàn bộ quá trình sản xuất. Trong tổng mức tiêu thụ năng lượng, điện chiếm khoảng 20-30%, ảnh hưởng trực tiếp đến chi phí sản xuất và hiệu quả kinh tế của doanh nghiệp. Đặc biệt, công đoạn nghiền clinker được ghi nhận là công đoạn tiêu thụ điện năng lớn nhất, chiếm tới 30–40% tổng lượng điện sử dụng trong toàn nhà máy. Đây là quá trình nghiền nhỏ clinker nóng chảy thành bột mịn trước khi phối trộn với phụ gia để tạo thành xi măng thành phẩm. Do clinker có độ cứng và tính chất cơ lý đặc thù, nên yêu cầu công suất nghiền lớn, hoạt động liên tục và tiêu tốn rất nhiều năng lượng.



Hình 2.1 Phân bố tiêu thụ điện năng trong sản xuất xi măng

Theo thống kê của Bộ Công Thương và nhiều nguồn báo chí chuyên ngành [6], mức tiêu thụ điện trung bình cho công đoạn nghiền clinker tại các nhà máy xi măng Việt

Nam hiện dao động trong khoảng 50–70 kWh/tấn clinker, cao hơn so với một số nước trong khu vực như Thái Lan hoặc Indonesia, nơi đã áp dụng rộng rãi công nghệ nghiền con lăn đứng (VRM) thay cho máy nghiền bi truyền thống. Trong khi đó, tại nhiều nhà máy ở Việt Nam, máy nghiền bi vẫn được sử dụng phổ biến, dù hiệu suất thấp hơn và tiêu tốn điện nhiều hơn khoảng 10–20%. Ngoài ra, hệ thống quạt gió, băng tải, thiết bị phân ly và hệ thống hồi nhiệt cũng góp phần không nhỏ vào tổng mức tiêu thụ điện của công đoạn này.

Tăng hiệu quả sử dụng điện trong quá trình nghiền clinker hiện đang là một trong những ưu tiên hàng đầu của nhiều doanh nghiệp sản xuất xi măng. Các giải pháp được nghiên cứu và áp dụng bao gồm: nâng cấp thiết bị nghiền (chuyển đổi từ máy nghiền bi sang VRM), lắp đặt biến tần điều khiển tốc độ cho quạt và bơm, cải tiến quy trình phối liệu, và đặc biệt là ứng dụng các hệ thống điều khiển thông minh (DCS/SCADA) tích hợp trí tuệ nhân tạo (AI) để giám, dự đoán và tối ưu tiêu thụ điện theo thời gian thực. Những cải tiến này không chỉ giúp tiết kiệm từ 10–30% điện năng cho mỗi tấn clinker, mà còn góp phần nâng cao năng suất, giảm chi phí vận hành và giảm phát thải khí CO_2 gián tiếp do tiêu thụ điện.

Trong bối cảnh xu hướng phát triển xanh và yêu cầu giảm phát thải carbon ngày càng cao, việc kiểm soát, dự đoán và tối ưu tiêu thụ điện trong công đoạn nghiền clinker là chìa khóa để các nhà máy xi măng đáp ứng các tiêu chuẩn môi trường quốc tế, giảm chi phí sản xuất và nâng cao năng lực cạnh tranh trong dài hạn.

2.2. Vấn đề tối ưu điện năng cho công đoạn nghiền

Trong bối cảnh giá điện ngày càng tăng và áp lực giảm phát thải carbon trở nên cấp thiết, việc kiểm soát và tối ưu hóa tiêu thụ điện năng trong các nhà máy xi măng không chỉ là vấn đề chi phí mà còn là yếu tố sống còn để đảm bảo năng lực cạnh tranh và phát triển bền vững. Đặc biệt ở công đoạn nghiền clinker – nơi tiêu thụ điện lớn nhất – bất kỳ cải tiến nào trong quản lý điện năng đều có thể mang lại lợi ích đáng kể về mặt kinh tế và môi trường.

Hiện nay, phần lớn các hệ thống điều khiển trong nhà máy xi măng vẫn hoạt động theo logic cố định hoặc bán tự động, chưa khai thác hiệu quả tiềm năng của dữ liệu sản xuất thời gian thực và chưa thích ứng linh hoạt với biến động vận hành như thay đổi đặc tính nguyên liệu, độ ẩm clinker, hoặc điều kiện môi trường. Trong khi đó, sản lượng sản xuất xi măng thường dao động theo mùa vụ, nhu cầu thị trường và kế hoạch vận hành, đòi hỏi hệ thống tiêu thụ điện phải được điều chỉnh hợp lý theo thời gian thực để đảm bảo hiệu quả tối ưu.

Tuy nhiên, để triển khai hiệu quả các giải pháp tiết kiệm điện năng, doanh nghiệp cần nắm bắt chính xác các quy luật biến động tiêu thụ điện theo thời gian, sản lượng, điều kiện vận hành và yếu tố môi trường. Do đó, việc dự báo chính xác mức tiêu thụ điện năng trở thành bước đầu tiên quan trọng trong quá trình quản lý năng lượng chủ động. Dự báo giúp các nhà máy đưa ra các quyết định vận hành phù hợp, lập kế hoạch sản xuất hợp lý, đồng thời phát hiện sớm các xu hướng bất thường dẫn đến lãng phí điện.

Chính vì vậy, việc dự báo tiêu thụ điện năng theo thời gian (time-series forecasting) trở nên đặc biệt quan trọng. Các mô hình học máy và học sâu như Linear Regression, ARIMA, ANN, LSTM đang được áp dụng để xây dựng các hệ thống dự đoán lượng điện tiêu thụ trong tương lai với độ chính xác cao. Dự báo tiêu thụ điện năng giúp nhà máy:

- Lập kế hoạch vận hành hợp lý: Dự đoán được giai đoạn tiêu thụ đỉnh điểm để phân bổ nguồn lực hoặc chuyển tải công suất.
- Tối ưu hóa hiệu suất nghiền clinker: Điều chỉnh tốc độ nghiền, tải hệ thống quạt, và phân ly theo dữ liệu dự báo nhằm giảm tiêu hao điện mà vẫn đảm bảo chất lượng sản phẩm.
- Ngăn ngừa các tình huống quá tải hoặc lãng phí năng lượng: Dựa trên mô hình dự báo, hệ thống có thể phát hiện các xu hướng bất thường và cảnh báo sớm để điều chỉnh vận hành.
- Tích hợp với các thuật toán tối ưu hóa: Từ kết quả dự báo, hệ thống có thể phối hợp với các thuật toán như Bayesian Optimization, Differential Evolution hay Latin Hypercube Sampling để tìm ra tổ hợp thông số vận hành tối ưu nhất nhằm giảm mức tiêu thụ điện năng trên mỗi tấn clinker.

Không dừng lại ở dự báo, các nhà máy còn cần đi xa hơn: tối ưu hóa điện năng tiêu thụ thông qua điều chỉnh các tham số vận hành như tốc độ cấp liệu, tốc độ cấp nước, áp suất, v.v., để đạt được hiệu quả năng lượng cao nhất. Quá trình này đòi hỏi sự kết hợp giữa mô hình dự đoán thông minh và các thuật toán tối ưu mạnh mẽ có khả năng xử lý bài toán nhiều biến và ràng buộc phức tạp.

2.3. Các phương pháp áp dụng để dự báo lượng điện năng tiêu thụ

Dự báo chính xác mức tiêu thụ điện năng là một bước quan trọng để tối ưu hóa vận hành, tiết kiệm chi phí và giảm phát thải trong các nhà máy xi măng. Trải qua nhiều giai đoạn phát triển, các phương pháp dự báo đã mở rộng từ những mô hình thống kê

đơn giản đến các mô hình học sâu hiện đại, với khả năng xử lý cả dữ liệu tuyến tính và phi tuyến, ngắn hạn và dài hạn. Việc lựa chọn phương pháp phù hợp phụ thuộc vào đặc điểm dữ liệu và mục tiêu ứng dụng trong từng nhà máy cụ thể.

2.3.1. Hồi quy tuyến tính

Hồi quy tuyến tính (Linear Regression) là một trong những phương pháp dự báo cơ bản và phổ biến nhất. Bianco và các cộng sự (2009) [7] đã tìm cách mô hình hóa mối quan hệ tuyến tính giữa biến phụ thuộc (mức tiêu thụ điện) tại một nhà máy ở Ý với một hoặc nhiều biến độc lập (ví dụ: sản lượng xi măng, nhiệt độ môi trường, giờ trong ngày, v.v.).

Cách hoạt động: Phương pháp này xây dựng một đường thẳng (hoặc mặt phẳng trong trường hợp đa biến) phù hợp nhất với dữ liệu, nhằm mục đích giảm thiểu tổng bình phương khoảng cách từ các điểm dữ liệu đến đường thẳng đó. Công thức tổng quát có thể là $y = ax + b$ (đơn biến) hoặc $y = b$ là các hệ số cần tìm.

Ưu điểm: Đơn giản, dễ hiểu, dễ triển khai và tính toán nhanh. Cung cấp một cái nhìn rõ ràng về mối quan hệ giữa các biến.

Nhược điểm: Giả định mối quan hệ tuyến tính, có thể không chính xác đối với các dữ liệu có quan hệ phi tuyến phức tạp. Dễ bị ảnh hưởng bởi các giá trị ngoại lai (outliers).

2.3.2. Autoregressive Integrated Moving Average

Autoregressive Integrated Moving Average (ARIMA) là một mô hình thống kê được sử dụng rộng rãi cho dữ liệu chuỗi thời gian. Nó đặc biệt hiệu quả khi dữ liệu có tính xu hướng và tính mùa vụ. Nhờ vào các sử dụng dữ liệu cho chuỗi thời gian bài báo của Parag Sen, Mousumi Roy và Parimal Pal (2016) [8] trình bày việc ứng dụng mô hình ARIMA để dự báo tiêu thụ năng lượng và lượng phát thải khí nhà kính (GHG) ở Ấn Độ, giúp hỗ trợ hoạch định chính sách năng lượng bền vững.

Cách hoạt động: Mô hình ARIMA được đặc trưng bởi ba tham số: (p, d, q).

- AR (p): Tự hồi quy (Autoregressive) - sử dụng các giá trị quan sát trong quá khứ của chính chuỗi thời gian để dự báo giá trị hiện tại.
- I (d): Tích hợp (Integrated) - số lần khác biệt hóa cần thiết để làm cho chuỗi thời gian trở nên dừng (stationary - tức là trung bình và phương sai không đổi theo thời gian).
- MA (q): Trung bình trượt (Moving Average) - sử dụng các sai số dự báo trong quá khứ để dự báo giá trị hiện tại.

Ưu điểm: Xử lý tốt các dữ liệu chuỗi thời gian có xu hướng và tính mùa vụ. Có cơ sở lý thuyết vững chắc.

Nhược điểm: Yêu cầu dữ liệu phải là chuỗi thời gian dừng hoặc có thể chuyển đổi thành dừng. Khó khăn trong việc xác định các tham số p, d, q tối ưu. Không hiệu quả với các mối quan hệ phi tuyến phức tạp.

2.3.3. Mạng nơ-ron nhân tạo

Mạng Nơ-ron Nhân Tạo (Artificial Neural Network - ANN) là một mô hình tính toán lấy cảm hứng từ cấu trúc và chức năng của bộ não con người. Nó bao gồm nhiều "nơ-ron" nhân tạo được kết nối với nhau thành các lớp.

Cách hoạt động: ANN học thông qua việc điều chỉnh trọng số của các kết nối giữa các nơ-ron. Dữ liệu đầu vào được truyền qua các lớp nơ-ron, mỗi nơ-ron thực hiện một phép tính và truyền kết quả đến các nơ-ron ở lớp tiếp theo. Quá trình này được lặp lại cho đến khi đạt được đầu ra. Việc học được thực hiện bằng cách so sánh đầu ra dự đoán với đầu ra thực tế và điều chỉnh trọng số thông qua thuật toán lan truyền ngược để giảm thiểu sai số. Trong nghiên cứu của Gifalli và các cộng sự (2024) [9], phương pháp dự báo tiêu thụ điện năng ngắn hạn được đề xuất bằng cách kết hợp mạng nơ-ron nhân tạo (MLP) với các mô hình hồi quy đa thức. Mô hình MLP đạt độ chính xác rất cao ($R^2 \approx 0.99$), vượt trội so với các hồi quy bậc 1–3 trong cả giai đoạn huấn luyện và kiểm chứng. Dựa trên dữ liệu thực tế từ công tơ điện thông minh, nghiên cứu này cho thấy ANN có tiềm năng ứng dụng mạnh mẽ trong quản lý năng lượng hộ gia đình và các hệ thống lưới điện thông minh.

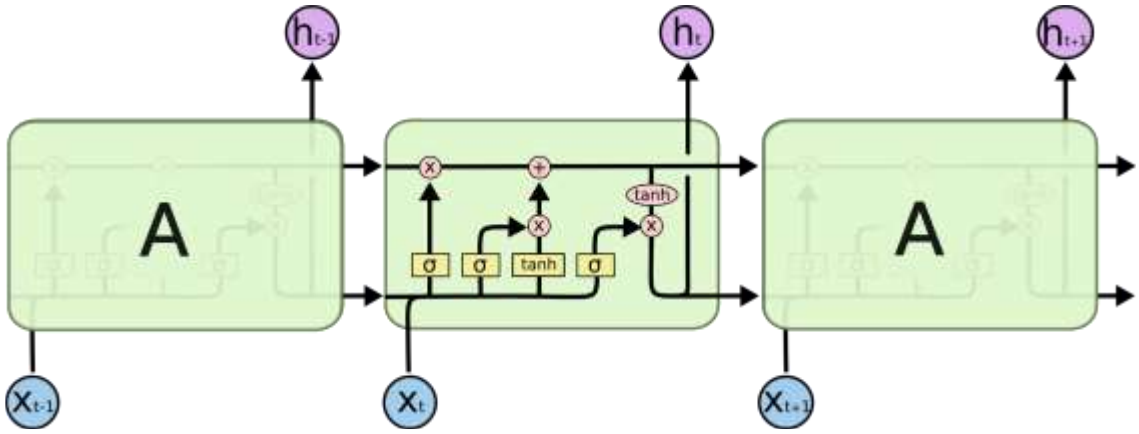
Ưu điểm: Có khả năng mô hình hóa các mối quan hệ phi tuyến phức tạp mà các phương pháp truyền thống khó xử lý. Hiệu quả với lượng lớn dữ liệu. Có khả năng tự học và thích nghi.

Nhược điểm: Cần lượng lớn dữ liệu để huấn luyện hiệu quả. "Hộp đen" (black box) - khó giải thích cách mô hình đưa ra dự đoán. Dễ bị quá khớp (overfitting) nếu không được kiểm soát.

2.3.4. Long Short-Term Memory

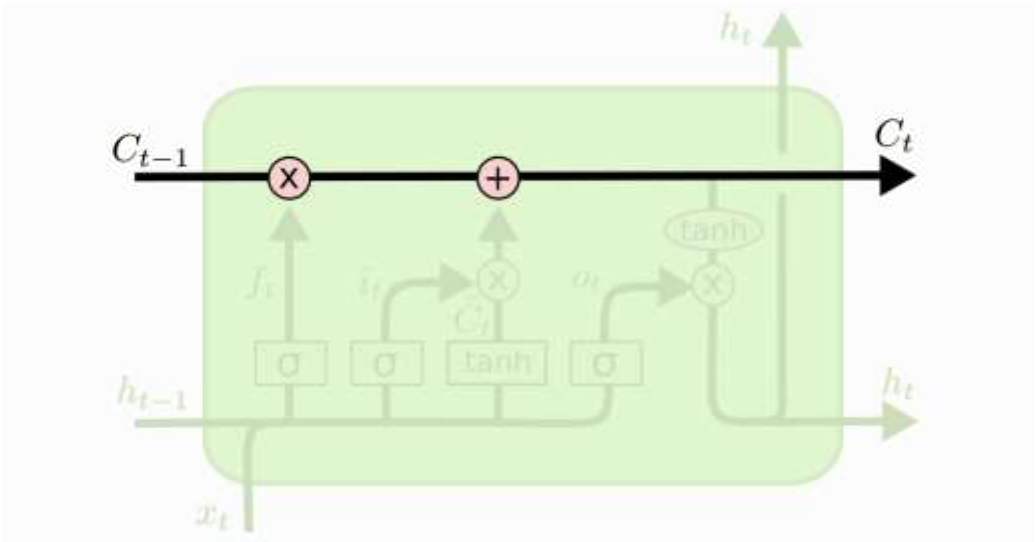
Mạng LSTM (Long Short-Term Memory) là một kiến trúc mạng nơ-ron hồi tiếp (Recurrent Neural Network – RNN) được thiết kế nhằm khắc phục nhược điểm của RNN truyền thống trong việc học các phụ thuộc dài hạn trong chuỗi dữ liệu. RNN thông thường thường gặp hiện tượng tiêu biến gradient (vanishing gradient), khiến cho mô hình không thể ghi nhớ được các thông tin từ xa trong chuỗi. LSTM được đề xuất lần

đầu bởi Hochreiter và Schmidhuber (1997) [10], đã trở thành một trong những kiến trúc nền tảng trong lĩnh vực học sâu liên quan đến chuỗi thời gian và ngôn ngữ tự nhiên.



Hình 2.2 Các mô-đun lặp của mạng LSTM chứa bốn layer

Khác với RNN truyền thống, LSTM sử dụng một cơ chế ô nhớ (cell state) đóng vai trò như một "đường truyền" thông tin chính được đề xuất bởi Sang và các cộng sự (2024) [11], cho phép lưu giữ hoặc loại bỏ thông tin thông qua ba cổng điều khiển: cổng quên (forget gate), cổng đầu vào (input gate), và cổng đầu ra (output gate). Cấu trúc này cho phép mô hình vừa duy trì thông tin dài hạn, vừa cập nhật linh hoạt thông tin mới từ chuỗi đầu vào.



Hình 2.3 Tế bào trạng thái LSTM

Cụ thể, tại thời điểm t, với đầu vào x_t, trạng thái ẩn từ bước trước h_{t-1}, và trạng thái ô nhớ trước đó C_{t-1}, các thành phần trong LSTM được tính như sau:

$$f_t = \sigma(\mathrm{w}_f \cdot [h_{t-1,x_t}] + b_f) \tag{2.1}$$

$$i_t = \sigma(\mathrm{w}_i \cdot [h_{t-1,x_t}] + b_i) \tag{2.2}$$

$$\hat{C}_t = \tanh(\mathrm{W}_c \cdot [h_{t-1,x_t}] + \mathrm{b}_c) \quad (2.3)$$

$$C_t = f_t * C_{t-1} + i_t * \hat{C}_t \quad (2.4)$$

$$o_t = \sigma(\mathrm{w}_o \cdot [h_{t-1,x_t}] + b_o) \quad (2.5)$$

$$h_t = o_t * \tanh(C_t) \quad (2.6)$$

Các phương trình (2.1)–(2.6) biểu diễn quá trình hoạt động của LSTM, được mô tả như sau: trong đó W_x và b_x , ($x \in (f, i, c, o)$) là các ma trận trọng số và độ lệch liên quan đến các cổng điều khiển, đầu ra h_{t-1} từ bước thời gian trước đó và đầu vào hiện tại x_t được sử dụng làm đầu vào cho LSTM, C_{t-1} và C_t biểu diễn trạng thái ô tại các bước thời gian trước đó và hiện tại, tương ứng, σ biểu thị hàm kích hoạt hình chữ S và tanh biểu diễn hàm kích hoạt tiếp tuyến hypebolic.

Với cơ chế kiểm soát dòng thông tin một cách chi tiết và chặt chẽ, LSTM cho phép mô hình học được các mối liên hệ dài hạn trong chuỗi thời gian mà vẫn giữ được tính ổn định trong quá trình huấn luyện. Điều này đặc biệt quan trọng trong các bài toán dự báo tiêu thụ điện năng, nơi mà các yếu tố đầu vào có thể có mối liên hệ không tuyến tính và phụ thuộc mạnh vào ngữ cảnh lịch sử. Dựa vào đó nghiên cứu của Palaniyappan và các cộng sự (2025) [12] đã đề xuất một mô hình LSTM được tối ưu hóa để dự báo ngắn hạn mức tiêu thụ điện năng tại trạm phân phối thông minh, nhằm phục vụ cho chính sách giá điện động theo ngày trong chương trình phản hồi phụ tải. Bằng cách điều chỉnh có hệ thống các siêu tham số (batch size, learning rate, epoch, optimizer), mô hình đã đạt độ chính xác cao với RMSE = 0.4454 và hệ số $R^2 = 0.9677$. Kết quả cho thấy phương pháp này giúp tối ưu cân bằng cung-cầu, giảm chi phí điện và cải thiện hiệu quả hệ thống lưới điện thông minh.

Ưu điểm: Cực kỳ hiệu quả trong việc xử lý dữ liệu chuỗi thời gian, như dự báo tiêu thụ điện theo giờ, ngày, tuần. Có khả năng nắm bắt các mẫu hình phức tạp và phụ thuộc dài hạn trong dữ liệu.

Nhược điểm: Phức tạp hơn ANN, cần nhiều tài nguyên tính toán hơn để huấn luyện. Cũng là một mô hình "hộp đen" và khó giải thích. Dễ bị quá khớp nếu dữ liệu không đủ hoặc mô hình không được điều chỉnh đúng cách.

2.3.5. Hiệu quả đánh giá

Sau khi phân tích các mô hình dự báo tiêu biểu như hồi quy tuyến tính, ARIMA, ANN và mạng LSTM, có thể nhận thấy rằng việc lựa chọn phương pháp dự báo phù hợp cần phải dựa trên đặc điểm cụ thể của tập dữ liệu và mục tiêu ứng dụng trong thực tế. Trong những trường hợp dữ liệu đầu vào đơn giản, số lượng biến ít và mối quan hệ giữa các biến có tính tuyến tính rõ ràng (ví dụ như giữa sản lượng xi măng và điện năng tiêu thụ), các phương pháp cổ điển như hồi quy tuyến tính hoặc ARIMA thường là lựa chọn

hợp lý. Hồi quy tuyến tính mang lại lợi thế về tính đơn giản, dễ triển khai và giải thích, trong khi ARIMA đặc biệt phù hợp cho các chuỗi dữ liệu có tính xu hướng hoặc tính mùa vụ rõ rệt.

Tuy nhiên, trong bối cảnh dữ liệu ngày càng trở nên phức tạp, chứa nhiều biến đầu vào và các mối quan hệ phi tuyến khó xác định bằng mô hình thống kê truyền thống, các phương pháp học sâu như ANN tỏ ra vượt trội hơn. ANN có khả năng học các mối liên hệ phi tuyến và mô hình hóa sự tương tác giữa nhiều yếu tố ảnh hưởng đến tiêu thụ điện năng, từ đó cho phép dự báo với độ chính xác cao hơn trong các hệ thống sản xuất đa biến. Đặc biệt, khi dữ liệu có dạng chuỗi thời gian và chứa các quan hệ phụ thuộc dài hạn như tiêu thụ điện theo giờ, theo ca hoặc theo ngày thì mạng LSTM là lựa chọn tối ưu. Với cấu trúc bộ nhớ và các cơ chế cổng thông minh, LSTM có thể ghi nhớ thông tin dài hạn và nắm bắt các xu hướng động trong dữ liệu mà các mô hình khác không xử lý hiệu quả.

Để minh họa rõ hơn các đặc điểm, ưu nhược điểm cũng như phạm vi áp dụng của từng phương pháp dự báo đã phân tích, bảng 2.1 trình bày sự so sánh tổng quan giữa các mô hình tiêu biểu bao gồm: hồi quy tuyến tính, ARIMA, mạng nơ-ron nhân tạo và mạng LSTM. Việc tổng hợp này nhằm cung cấp cái nhìn trực quan giúp lựa chọn phương pháp phù hợp với đặc điểm dữ liệu và yêu cầu thực tiễn của bài toán dự báo điện năng tiêu thụ trong công nghiệp.

Bảng 2.1 So sánh các phương pháp dự đoán

Mô hình	**Đặc điểm**	**Ưu điểm**	**Hạn chế**	**Dữ liệu**
Hồi quy tuyến tính	Mô hình thống kê tuyến tính	Đơn giản, dễ hiểu, dễ triển khai	Không xử lý tốt quan hệ phi tuyến, nhạy với ngoại lai	Dữ liệu đơn giản, quan hệ tuyến tính rõ ràng
ARIMA	Mô hình thống kê chuỗi thời gian	Hiệu quả với dữ liệu có xu hướng và mùa vụ rõ rệt	Cần chuỗi thời gian dừng, khó chọn tham số	Dữ liệu chuỗi thời gian có xu hướng hoặc mùa vụ
ANN	Mô hình học sâu phi tuyến	Mô hình hóa quan hệ phi tuyến phức tạp,	Cần dữ liệu lớn, khó giải thích kết quả, dễ quá khớp	Dữ liệu nhiều biến, mối quan hệ phức tạp

		hiệu quả với dữ liệu lớn		
LSTM	Mạng nơ-ron hồi tiếp cho chuỗi thời gian dài hạn	Mạng nơ-ron hồi tiếp cho chuỗi thời gian dài hạn	Mạng nơ-ron hồi tiếp cho chuỗi thời gian dài hạn	Dữ liệu chuỗi thời gian có phụ thuộc dài, biến động theo thời gian

2.4. Các phương pháp tối ưu hóa năng lượng điện năng tiêu thụ

2.4.1. Gradient Descent

Gradient Descent là một thuật toán tối ưu hóa cổ điển, được sử dụng rộng rãi để tìm ra giá trị nhỏ nhất của một hàm mục tiêu (thường là hàm chi phí hoặc hàm mất mát). Phương pháp này đặc biệt hữu ích trong các bài toán tối ưu hóa năng lượng tiêu thụ, khi cần điều chỉnh các tham số của hệ thống sao cho tổng năng lượng sử dụng là thấp nhất. Nghiên cứu của Haji và các cộng sự (2021) [13] đã tổng quan và so sánh các kỹ thuật tối ưu hóa dựa trên thuật toán Gradient Descent như GD, BGD, SGD, và Mini-batch GD, cùng các biến thể tiên tiến như Adam, RMSProp, và Nadam. Mục tiêu là đánh giá hiệu suất, tốc độ hội tụ, ưu nhược điểm trong huấn luyện mô hình học máy và học sâu.

Gradient Descent dựa trên ý tưởng dịch chuyển dần các tham số theo hướng ngược lại với gradient (đạo hàm bậc nhất) của hàm mục tiêu. Gradient biểu thị hướng tăng nhanh nhất của hàm số, do đó, đi ngược lại gradient giúp đưa giá trị hàm số về điểm cực tiểu. Thuật toán này được đề xuất lần đầu bởi nhà toán học người Pháp Augustin-Louis Cauchy vào năm 1847 và hiện nay là nền tảng cho rất nhiều kỹ thuật tối ưu hóa hiện đại.

Thuật toán Gradient Descent dựa trên quan sát rằng nếu một hàm nhiều biến F(x) có đạo hàm và khả vi trong một vùng xung quanh điểm a, thì F(x) giảm nhanh nhất nếu ta đi theo hướng ngược lại với gradient của hàm tại điểm đó, tức là theo hướng $a_n - \nabla F(a)$.

Cập nhật giá trị của tham số a_{n+1} theo công thức:

$$a_{n+1} = a_n - \gamma \nabla F(a_n) \tag{2.7}$$

Trong đó:

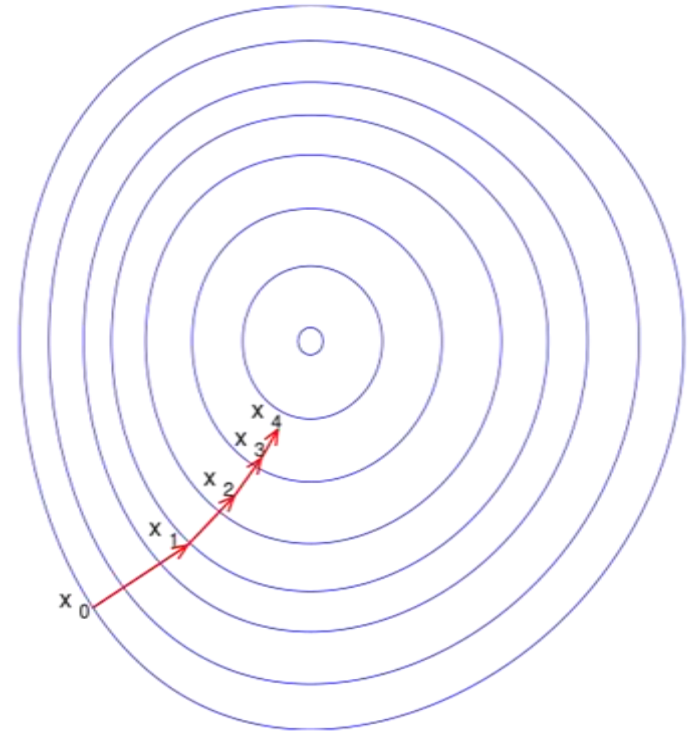
- γ là learning rate (tốc độ học) – chỉ ra bước đi mỗi lần cập nhật.
- $\nabla F(a_n)$ là gradient của hàm F tại a_n .

- a_n là giá trị tham số tại bước n, và a_{n+1} là giá trị tham số cập nhật sau bước đi.

Thuật toán Gradient Descent tạo ra một dãy điểm $x_0, x_1, x_2, \ldots$ trong đó mỗi điểm x_n là kết quả sau khi cập nhật tham số theo hướng ngược lại với gradient của hàm mất mát tại bước trước. Dãy này có tính chất đơn điệu, tức là giá trị hàm mất mát giảm dần theo từng bước:

$$F(x_0) \geq F(x_1) \geq F(x_1) \geq \ldots$$

Điều này có nghĩa là thuật toán luôn tiến dần đến điểm có giá trị hàm mất mát nhỏ hơn hoặc bằng, đảm bảo quá trình tối ưu diễn ra đúng hướng.



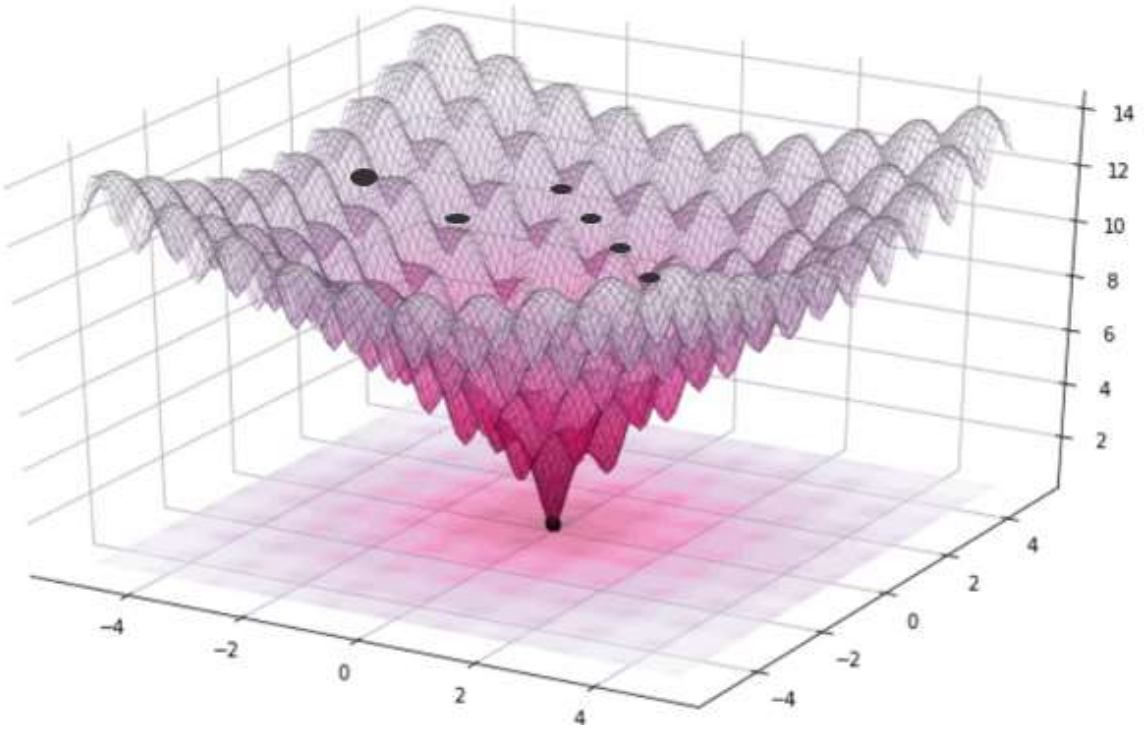
Hình 2.4 Thuật toán Gradient Descent

2.4.2. Differential Evolution

Differential Evolution (DE) là một thuật toán tiến hóa được sử dụng để giải quyết các bài toán tối ưu hóa toàn cục. Đây là một phương pháp tối ưu hóa dựa trên quần thể nhằm tìm kiếm giá trị tối ưu bằng cách liên tục cải thiện các giải pháp ứng viên thông qua các bước đột biến, lai ghép và chọn lọc.

Thuật toán này được giới thiệu bởi Storn và Price (2005) [14], những người đã mô tả DE như một phương pháp heuristic mới để tối thiểu hóa các hàm trong không gian liên tục, có thể phi tuyến tính và không khả vi. Lakshminarasimman và Subramanian

(2008) [15] đã nghiên cứu việc ứng dụng thuật toán tiến hóa vi sai và các biến thể của nó vào tối ưu hóa hệ thống điện, bao gồm bài toán phân phối kinh tế, phân phối kinh tế động và phân phối tổ máy. Các kỹ thuật DE được cải tiến nhằm đạt hiệu quả tính toán cao, khả năng hội tụ nhanh và xử lý tốt các ràng buộc phi tuyến phức tạp. Kết quả mô phỏng cho thấy phương pháp đề xuất vượt trội hơn so với các thuật toán tiến hóa truyền thống khác về cả chi phí và thời gian tính toán.



Hình 2.5 Thuật toán Differential Evolution

DE hoạt động bằng cách duy trì một quần thể gồm nhiều nghiệm ứng viên (gọi là agents hoặc cá thể) trong không gian tìm kiếm. Qua từng vòng lặp (thế hệ), các cá thể này được cập nhật thông qua quá trình kết hợp, lai ghép và chọn lọc, nhằm tìm ra nghiệm tối ưu (cực tiểu toàn cục) của hàm mục tiêu.

Thuật toán DE nhằm tìm cực tiểu của một hàm mục tiêu $f : R^n \to R$, thông qua quá trình lặp đi lặp lại việc cải thiện quần thể các nghiệm ứng viên. Các bước thực hiện của thuật toán được mô tả như sau:

Bước 1: Khởi tạo tham số

Trước tiên, cần xác định các tham số chính của thuật toán gồm: kích thước quần thể NP, hệ số lai ghép $CR \in [0,1]$, và hệ số khuếch đại sai phân $F \in [0,2]$. Trong thực tế, các giá trị khuyến nghị thường được chọn là $NP = 10n$ (với n là số chiều của bài

toán), $CR = 0.9$ và $F = 0.8$. Các tham số này ảnh hưởng trực tiếp đến tốc độ hội tụ và khả năng khám phá không gian tìm kiếm của thuật toán.

Bước 2: Khởi tạo quần thể ban đầu

Quần thể ban đầu gồm NP cá thể (vector) được khởi tạo ngẫu nhiên trong không gian tìm kiếm. Mỗi cá thể $x \in R^n$ là một nghiệm ứng viên cho bài toán tối ưu.

Bước 3: Tiến hành vòng lặp tối ưu

Thuật toán lặp lại quá trình tiến hóa cho đến khi thỏa mãn điều kiện dừng, chẳng hạn như đạt số vòng lặp tối đa hoặc hàm mục tiêu đạt giá trị đủ nhỏ. Trong mỗi vòng lặp, thuật toán thực hiện các bước sau đối với từng cá thể x trong quần thể:

- Chọn ngẫu nhiên ba cá thể khác nhau a, b, c từ quần thể, đảm bảo không trùng với cá thể x đang xét.
- Chọn một chỉ số ngẫu nhiên $R \in \{1, 2, ..., n\}$ đại diện cho vị trí bắt buộc áp dụng phép lai.
- Tạo một cá thể mới $y = (y_1, y_2, ..., y_n)$ thông qua tổ hợp từ các cá thể đã chọn. Với mỗi thành phần i, giá trị y_i được tính theo công thức:

$$y_i = \begin{cases} a_i + F \cdot (b_i - c_i), r_i < CR \text{ or } i = R \\ x_i \qquad\qquad\qquad ,otherwise \end{cases} \tag{2.8}$$

- Trong đó $r_i \sim U(0,1)$ là một số ngẫu nhiên phân bố đều trên khoảng [0,1]. Điều kiện này đảm bảo rằng ít nhất một thành phần trong vector y được lấy từ tổ hợp sai phân.

Bước 4: Chọn lọc

Nếu cá thể mới y có giá trị hàm mục tiêu tốt hơn hoặc bằng cá thể hiện tại x (tức là $f(y) \le f(x)$), thì x sẽ được thay thế bởi y trong quần thể. Quá trình này giúp quần thể dần cải thiện theo hướng tối ưu.

Bước 5: Kết thúc thuật toán

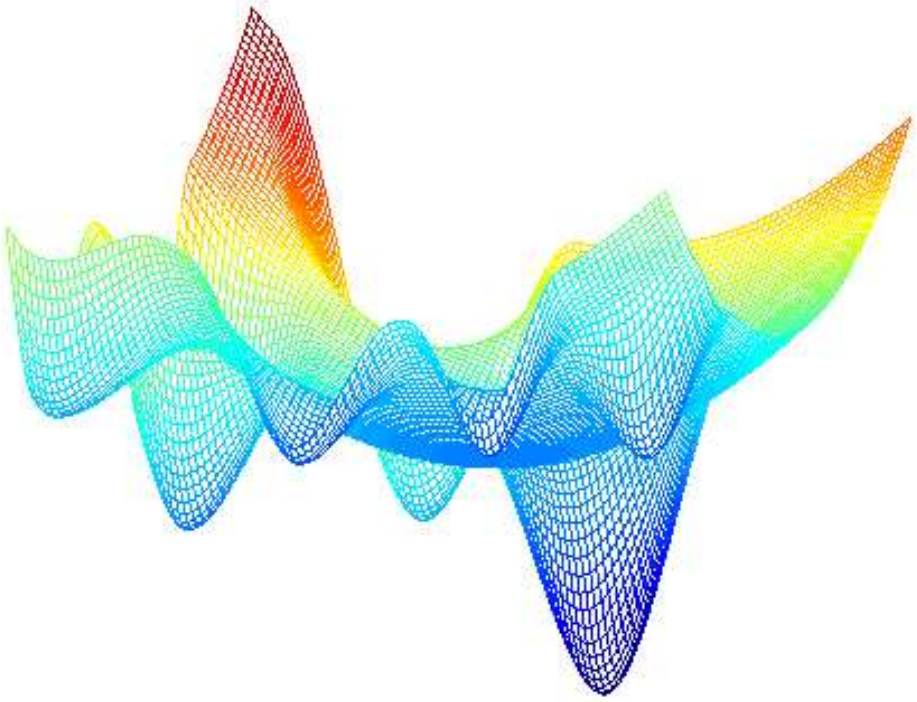
Sau khi quá trình lặp kết thúc, cá thể có giá trị hàm mục tiêu nhỏ nhất trong quần thể sẽ được chọn làm nghiệm tối ưu của bài toán.

2.4.3. Bayesian

Bayesian Optimization được phát triển từ các nghiên cứu về thống kê và lý thuyết xác suất vào những năm 1970–1980, nhưng phải đến khoảng đầu những năm 2010, thuật

toán này mới thực sự phổ biến trong cộng đồng học máy, đặc biệt qua các công trình của Jonas Mockus, Donald Jones, Eric Brochu và cộng sự.

Bayesian Optimization là một phương pháp tối ưu hóa toàn cục cho các hàm mục tiêu phức tạp, "hộp đen", không có công thức đạo hàm, và tốn kém khi đánh giá (ví dụ: tối ưu siêu tham số trong học máy, điều khiển quy trình công nghiệp,…). Nghiên cứu của Snoek và các cộng sự (2014) [16] đã đề xuất phương pháp biến đổi đầu vào sử dụng hàm phân phối tích lũy Beta nhằm cải thiện hiệu quả của tối ưu hóa Bayes khi xử lý các hàm không dừng, giúp mô hình Gaussian Process mô tả tốt hơn các biến đổi không đồng nhất trong không gian đầu vào. Phương pháp này cũng được mở rộng cho bài toán tối ưu đa tác vụ, mang lại kết quả vượt trội và đáng tin cậy hơn trên nhiều bài toán khác. Phương pháp này dựa trên xác suất để mô hình hóa hàm mục tiêu và chọn điểm đánh giá tiếp theo dựa trên thông tin đã biết, nhằm cân bằng giữa việc khai thác và khám phá các vùng chưa biết.



Hình 2.6 Thuật toán tối ưu Bayesian

Bước 1: Thử vài điểm ban đầu

Trên bề mặt "lượn sóng" nhiều đỉnh và đáy như hình, trước tiên ta chọn ngẫu nhiên một vài điểm trên mặt này để "thử nghiệm", tức là đo giá trị hàm mục tiêu tại các vị trí đó. Ví dụ, chấm vài điểm bất kỳ trên mặt để xem giá trị cao hay thấp.

Bước 2: Dự đoán toàn bộ bề mặt

Từ các điểm đã thử, Bayesian Optimization sẽ xây dựng một mô hình để "đoán" hình dạng toàn bộ bề mặt, kể cả những chỗ chưa từng thử. Mô hình này giống như một bản đồ dự đoán, chỗ nào càng chưa thử thì dự đoán càng "mờ" (không chắc chắn).

Bước 3: Quyết định thử tiếp ở đâu

Dựa vào bản đồ dự đoán này, thuật toán dùng một công thức (gọi là hàm thu nhận) để chọn ra điểm tiếp theo nên thử ở đâu. Có thể chọn những vùng dự đoán rất thấp (có thể là cực tiểu), hoặc vùng chưa từng thử (để khám phá thêm).

Bước 4: Thử điểm mới và cập nhật mô hình

Chọn điểm vừa tìm được để đo giá trị thực tế của hàm mục tiêu tại đó (tức là lại "hỏi" bề mặt này xem chỗ đó cao hay thấp). Sau đó, cập nhật lại bản đồ dự đoán với dữ liệu mới này, mô hình sẽ ngày càng chính xác hơn.

Bước 5: Lặp lại

Tiếp tục lặp lại các bước: chọn điểm mới, thử nghiệm, cập nhật mô hình. Càng thử nhiều điểm, mô hình càng biết rõ chỗ nào thấp nhất trên bề mặt.

Kết quả cuối cùng:

Sau nhiều lần lặp, thuật toán sẽ tìm được vị trí có giá trị thấp nhất trên bề mặt (tức là điểm tối ưu). Trên hình, đó sẽ là điểm nằm ở đáy sâu nhất của các "thung lũng" trên bề mặt lượn sóng.

Nhờ vào quy trình trên Sultana và các cộng sự (2022) [17] đã phát triển mô hình dự báo phụ tải điện ngắn hạn tại Ontario bằng cách kết hợp thuật toán tối ưu Bayesian (BOA) với các mô hình SARIMAX và NARX. Kết quả cho thấy mô hình BOA-NARX đạt hiệu quả dự báo ổn định hơn với sai số MAPE khoảng 3%, vượt trội so với BOA-SARIMAX vốn có độ biến động lớn theo mùa.

2.4.4. Thuật toán lấy mẫu Latin Hypercube

Thuật toán Lấy Mẫu Latin Hypercube (LHS) là một phương pháp lấy mẫu ngẫu nhiên được sử dụng rộng rãi trong mô phỏng thí nghiệm và tích phân số. Được phát triển từ những năm 1970, phương pháp này giúp tạo ra các tập dữ liệu đại diện tốt hơn so với lấy mẫu ngẫu nhiên đơn thuần, đặc biệt trong các bài toán yêu cầu phân tích độ nhạy hoặc tối ưu hóa tham số trong không gian nhiều chiều. Cùng với đó nghiên cứu của Shields và Zhang (2016) [18] đã tổng quát hóa phương pháp lấy mẫu Latin Hypercube thành một phổ thiết kế lấy mẫu phân tầng, gọi là lấy mẫu phân tầng từng phần (PSS),

đồng thời giới thiệu phương pháp lấy mẫu phân tầng Latin hóa (LPSS) nhằm giảm phương sai của cả hiệu chính và tương tác biến trong các bài toán bất định cao chiều.

Nghiên cứu của Guo và các cộng sự (2020) [19] đề xuất thuật toán đánh giá rủi ro an ninh hệ thống điện dựa trên phương pháp lấy mẫu Latin Hypercube kết hợp dự báo phụ tải cực đại hằng ngày bằng mạng nơ-ron GRU. Kết quả cho thấy phương pháp này đạt độ chính xác cao hơn với số mẫu ít hơn so với Monte Carlo, đồng thời phản ánh rõ mối liên hệ giữa biến động phụ tải và rủi ro hệ thống.

	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10
A1			X							
A2		X								
A3					X					
A4	X									
A5						X				
A6								X		
A7							X			
A8				X						
A9										X
A10									X	

Hình 2.7 Thuật toán lấy mẫu Latin Hypercube

Hình 2.7 là một ma trận 10x10, với các hàng (A1, A2, ..., A10) và cột (B1, B2, ..., B10). Mỗi ô có thể đại diện cho một tổ hợp các giá trị của hai biến (A là biến thứ nhất, B là biến thứ hai). Những dấu "X" chính là các điểm lấy mẫu được lựa chọn.

Quy trình lấy mẫu LHS gồm các bước sau:

Bước 1: Chia đều không gian tham số

Với mỗi biến (A và B), không gian giá trị của biến đó được chia thành 10 khoảng bằng nhau (nếu lấy 10 mẫu).

Như vậy, mỗi hàng và mỗi cột trong bảng ứng với một khoảng giá trị của từng biến.

Bước 2: Lấy mẫu không trùng lặp

Với mỗi hàng, chỉ chọn duy nhất một cột để đặt dấu "X", sao cho trong toàn bộ bảng, mỗi cột chỉ chứa một dấu "X" và mỗi hàng cũng chỉ chứa một dấu "X".

Điều này đảm bảo mỗi khoảng của mỗi biến đều được lấy mẫu một lần, không bị trùng.

Bước 3: Phân bố đồng đều

Kết quả là các điểm mẫu (vị trí của các dấu "X") được phân bố đều đặn trên toàn bộ không gian hai chiều, đảm bảo đa dạng và đại diện cho không gian tham số.

Không có hai điểm nào trùng hàng hoặc trùng cột, tránh việc các mẫu bị tập trung vào một vùng nào đó.

2.4.5. Hiệu quả đánh giá

Trong các hệ thống công nghiệp tiêu thụ nhiều điện năng như dây chuyền nghiền xi măng, việc tối ưu hóa tiêu thụ điện năng không chỉ giúp tiết kiệm chi phí vận hành mà còn góp phần giảm phát thải và nâng cao hiệu suất tổng thể. Để đạt được mục tiêu này, nhiều phương pháp tối ưu hóa đã được nghiên cứu và áp dụng. Các phương pháp này có thể dựa trên đạo hàm, tiến hóa theo quần thể, mô hình xác suất hoặc kỹ thuật lấy mẫu không gian tham số. Bảng 2.2 trình bày so sánh giữa một số thuật toán tối ưu hóa tiêu biểu thường được sử dụng trong lĩnh vực tối ưu tiêu thụ năng lượng.

Bảng 2.2 So sánh các thuật toán tối ưu

Thuật toán	**Đặc điểm**	**Ưu điểm**	**Hạn chế**	**Dữ liệu yêu cầu**
Gradient Descent	Tối ưu theo hướng đạo hàm âm, cập nhật từng bước nhỏ.	Nhanh, đơn giản, hiệu quả với hàm lồi.	Dễ mắc cực tiểu cục bộ, phụ thuộc learning rate, cần đạo hàm.	Cần đạo hàm hoặc gradient có thể tính được.
Differential Evolution	Tiến hóa theo quần thể, dùng tổ hợp và sai phân vector để tạo cá thể mới.	Không cần đạo hàm, tìm được cực trị toàn cục, dễ áp dụng.	Tốc độ chậm khi số chiều lớn, phụ thuộc tham số (CR, F, NP).	Chỉ cần đánh giá hàm mục tiêu tại từng điểm.

Bayesian Optimization	Dùng mô hình xác suất để dự đoán và chọn điểm đánh giá tiếp theo.	Hiệu quả khi số lần đánh giá hạn chế, xử lý bài toán phức tạp tốt.	Kém hiệu quả với bài toán nhiều chiều, cần xây mô hình xác suất phức tạp.	Cần đầu vào - đầu ra của hàm mục tiêu, không cần đạo hàm.
Latin Hypercube Sampling	Lấy mẫu đồng đều trên không gian tìm kiếm bằng cách chia đều từng chiều.	Phủ đều không gian, hiệu quả khi thiết kế thí nghiệm hoặc huấn luyện mô hình.	Cần không gian số liệu đủ lớn để tối ưu.	Cần khoảng giá trị các biến, dùng để sinh dữ liệu hoặc khảo sát không gian.

2.5. Kết luận và định hướng nghiên cứu

Chương 2 đã trình bày tổng quan về thực trạng tiêu thụ điện trong nhà máy xi măng, các yếu tố ảnh hưởng đến chi phí vận hành, cũng như các phương pháp dự báo và tối ưu hóa điện năng tiêu thụ. Qua quá trình phân tích, nhóm nghiên cứu nhận thấy rằng việc lựa chọn mô hình dự báo phù hợp kết hợp với chiến lược tối ưu hóa hiệu quả là yếu tố then chốt nhằm nâng cao độ chính xác trong dự báo và giảm thiểu chi phí năng lượng.

Về mô hình dự báo, nhóm lựa chọn sử dụng CNN-LSTM, một mô hình học sâu kết hợp giữa mạng tích chập và mạng hồi tiếp dài hạn. CNN có khả năng trích xuất đặc trưng tự động từ dữ liệu chuỗi thời gian, trong khi LSTM giúp học và ghi nhớ các mối quan hệ dài hạn trong dữ liệu. Nhờ đó, mô hình CNN-LSTM có thể đưa ra dự báo chính xác ngay cả khi dữ liệu biến động phức tạp, phi tuyến hoặc chứa nhiễu – đặc điểm thường thấy trong dữ liệu công nghiệp.

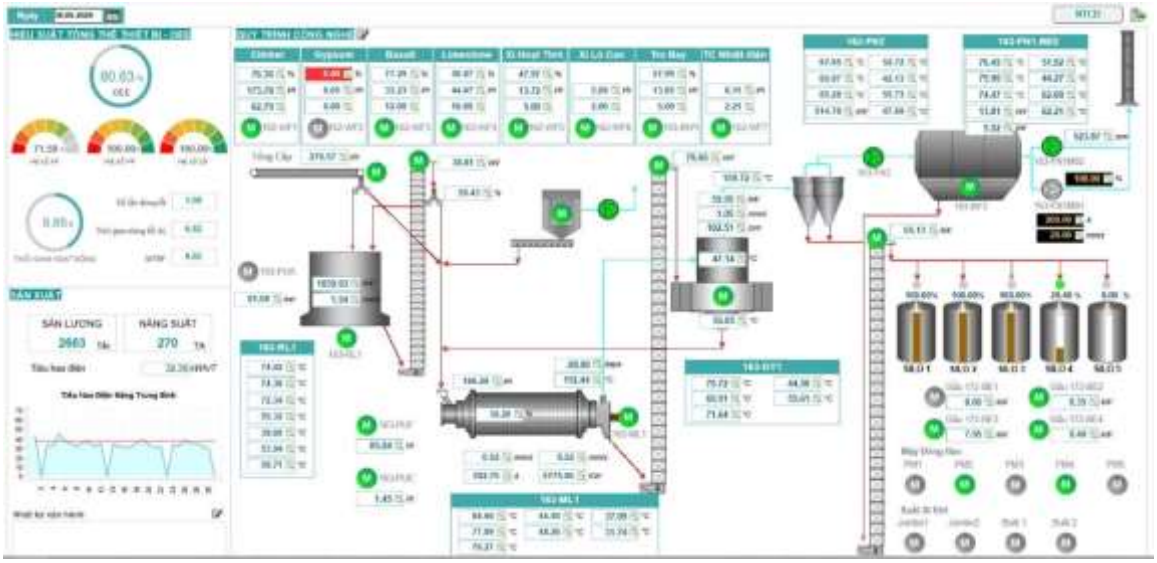
Để tối ưu hóa các đầu vào cho mô hình, nhóm sử dụng thuật toán Latin Hypercube Sampling. Đây là phương pháp lấy mẫu hiệu quả, đảm bảo phân bố đều và bao phủ toàn diện không gian tham số. So với các phương pháp dò quét toàn bộ, LHS giúp giảm thời gian tính toán và tăng khả năng tìm được tổ hợp tham số tốt mà không bỏ sót các vùng tiềm năng.

Nhìn chung, nội dung chương 2 đã xây dựng nền tảng lý thuyết và thực tiễn vững chắc, từ đó định hướng rõ ràng cho các bước triển khai mô hình, kiểm thử và đánh giá trong các chương tiếp theo.

Chương 3: MÔ HÌNH DỰ ĐOÁN VÀ TỐI ƯU NĂNG LƯỢNG TIÊU THỤ CỦA CÔNG ĐOẠN NGHIỀN XI MĂNG

3.1. Thu thập và xử lý dữ liệu thực nghiệm

Dữ liệu thực nghiệm được trích xuất từ cơ sở dữ liệu của hệ thống XHQ quản lý sản xuất thông minh mà công ty đã áp dụng cho một nhà máy xi măng ở Việt Nam, với tần suất theo thời gian thực và độ chính xác cao. Bộ dữ liệu bao gồm các thông số kỹ thuật quan trọng trong quá trình vận hành máy nghiền bi và máy nghiền đứng, cũng như các yếu tố môi trường và điều kiện hoạt động liên quan. Mục tiêu là dự báo chính xác mức tiêu thụ điện năng trên mỗi tấn sản phẩm.



Hình 3.1 Hệ thống XHQ

3.1.1. Các tham số vào/ra

Việc lựa chọn các tham số đầu vào và đầu ra (mục tiêu) là bước cơ bản trong quá trình dự báo và tối ưu công suất tiêu thụ điện cho giai đoạn nghiền xi măng. Trong nghiên cứu này, 11 thông số dữ liệu đầu vào và 1 thông số dữ liệu đầu ra đặc trưng cho tình trạng vận hành của hệ thống nghiền được mô tả ở bảng 3.1.

Bảng 3.1 Danh sách các biến đầu vào-đầu ra

STT	Tên biến	Đơn vị	Vai trò
1	Timestamp	Phút	Thời gian đo
2	Tốc độ cấp liệu	Tấn/giờ	Đầu vào

3	Lưu lượng hồi về nghiền bi	Tấn/giờ	Đầu vào
4	Tải của máy nghiền bi	kW	Đầu vào
5	Độ ồn của ngăn 1 máy nghiền bi	dB	Đầu vào
6	Áp suất sau máy nghiền bi	MPa	Đầu vào
7	Tải của máy nghiền đứng CKP	kW	Đầu vào
8	Độ rung của máy nghiền CKP	mm/s	Đầu vào
9	Tốc độ cấp nước	m^3/giờ	Đầu vào
10	Nhiệt độ bệ động cơ	℃	Đầu vào
11	Sản lượng	Tấn/giờ	Đầu vào
12	Tiêu hao điện	kWh/tấn	Đầu ra

3.1.2. Tiền xử lý dữ liệu

Trong thực tế, số liệu thu thập có chứa nhiều sai số và chuỗi số liệu có thể bị khuyết thiếu do hệ thống cảm biến, thời điểm bảo trì hoặc dừng máy nên việc xử lý trước khi đưa vào mô hình là rất cần thiết. Ngoài ra, các đặc trưng đầu vào trong bộ dữ liệu có đơn vị và dải giá trị rất khác nhau, ví dụ: tốc độ cấp liệu có thể từ vài chục đến vài trăm (T/h), trong khi độ rung chỉ dao động quanh vài đơn vị. Nếu không chuẩn hóa, các mô hình học máy, đặc biệt là mạng nơ-ron, sẽ dễ bị chi phối bởi các đặc trưng có giá trị tuyệt đối lớn hơn, dẫn đến kết quả học thiên lệch hoặc chậm hội tụ.

Trong nghiên cứu này, phương pháp Min-Max Scaler được sử dụng để chuẩn hóa toàn bộ các đặc trưng đầu vào và đầu ra (trừ cột thời gian) về khoảng [0, 1], theo công thức:

$$x' = \frac{x - x_{\min}}{x_{\max} - x_{\min}} \tag{3.1}$$

trong đó:

x : giá trị gốc của biến,

$x_{\max}$, $x_{\min}$: giá trị lớn nhất và nhỏ nhất của biến trong toàn bộ tập dữ liệu,

x' : giá trị sau chuẩn hóa.

Việc chuẩn hóa này giúp mô hình dễ dàng xử lý các đặc trưng có thang đo khác nhau, tăng hiệu quả huấn luyện và cải thiện khả năng hội tụ nhanh chóng của các thuật toán tối ưu. Ngoài ra, chuẩn hóa cũng giúp giảm rủi ro nổ gradient hoặc mất ổn định trong quá trình huấn luyện các mô hình phức tạp như ANN hoặc LSTM.

Sau khi chuẩn hóa, dữ liệu được phân chia theo thứ tự thời gian thành ba tập:

- Tập huấn luyện (train): dùng để huấn luyện mô hình (thường chiếm 70%).
- Tập kiểm tra (validation): dùng để điều chỉnh siêu tham số và tránh overfitting (khoảng 15%).
- Tập kiểm định (test): đánh giá khách quan độ chính xác của mô hình trên dữ liệu chưa từng thấy (khoảng 15%).

Việc phân chia theo trình tự thời gian (thay vì ngẫu nhiên) đặc biệt quan trọng trong bài toán chuỗi thời gian để đảm bảo tính thực tiễn khi triển khai mô hình dự báo trong tương lai.

3.2. Xây dựng mô hình dự đoán điện năng tiêu thụ

Sau khi hoàn tất quá trình thu thập và xử lý dữ liệu, bước tiếp theo trong chuỗi công việc là xây dựng một mô hình dự đoán điện năng tiêu thụ phù hợp, hiệu quả và có khả năng khái quát tốt trên dữ liệu thực tế. Mục tiêu của mô hình là dự đoán giá trị tiêu hao điện năng (kWh/T) dựa trên các thông số vận hành được đo lường liên tục từ hệ thống XHQ của nhà máy xi măng.

Do dữ liệu đầu vào có tính chất chuỗi thời gian (biến đổi theo từng thời điểm cụ thể) và đồng thời chứa nhiều đặc trưng kỹ thuật đa chiều như tốc độ, tải, độ rung, áp suất..., việc lựa chọn kiến trúc mô hình phải đảm bảo xử lý tốt cả hai yếu tố:

- Trích xuất được các đặc trưng không gian – kỹ thuật giữa các tín hiệu đầu vào ở mỗi thời điểm,
- Nắm bắt được mối quan hệ thời gian giữa các bước đo liên tiếp.

Để đáp ứng yêu cầu này, nghiên cứu đề xuất sử dụng mô hình kết hợp CNN-LSTM, trong đó:

- CNN được sử dụng để trích xuất các đặc trưng không gian từ dữ liệu vận hành đa chiều tại mỗi thời điểm. Trong công đoạn nghiền, dữ liệu đầu vào bao gồm các thông số như: tốc độ cấp liệu, lưu lượng hồi về, tải máy nghiền, áp suất, độ rung, nhiệt độ, v.v. Những biến này thường có mối liên hệ phức tạp và đồng thời ảnh hưởng đến mức tiêu thụ điện năng. CNN giúp tự động học được các mẫu đặc trưng

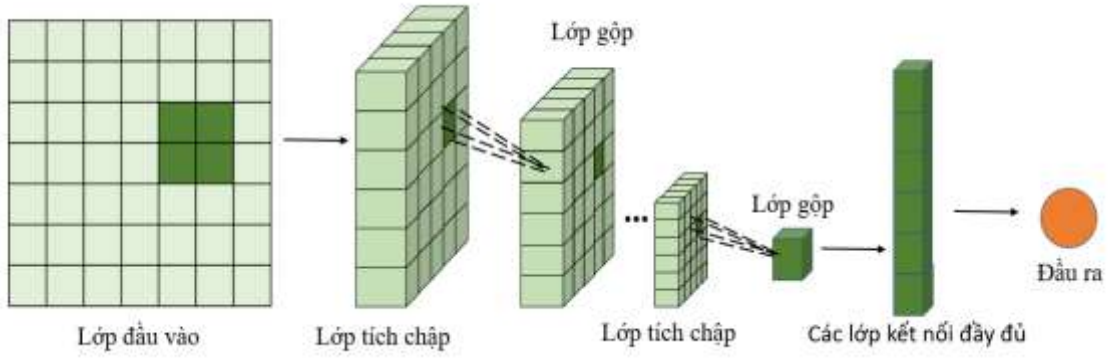
ẩn giữa các biến này, ví dụ: tổ hợp giữa độ ồn và tải máy nghiền có thể báo hiệu máy đang chạy non tải.

- LSTM được tích hợp tiếp theo để xử lý tính phụ thuộc theo thời gian của dữ liệu. Với bản chất là chuỗi thời gian, dữ liệu về tiêu thụ điện tại thời điểm hiện tại không chỉ phụ thuộc vào trạng thái tức thời mà còn bị ảnh hưởng bởi các điều kiện vận hành ở nhiều thời điểm trước đó (ví dụ: nếu tải máy tăng dần trong 30 phút, tiêu thụ điện có thể tăng mạnh ở thời điểm hiện tại). LSTM giúp mạng học được các xu hướng và mối liên hệ dài hạn, từ đó nâng cao độ chính xác của dự báo.

Việc kết hợp CNN và LSTM trong một kiến trúc thống nhất tận dụng được ưu điểm của cả hai mô hình, đồng thời cải thiện độ chính xác dự báo so với việc chỉ dùng LSTM hoặc ANN đơn thuần.

3.2.1. Mạng nơ-ron tích chập

Mạng nơ-ron tích chập (Convolutional Neural Network – CNN) được đề xuất bởi Alharkan và các cộng sự (2023) [20] là một loại mạng học sâu đặc biệt, được thiết kế để tự động trích xuất và học các đặc trưng từ dữ liệu đầu vào thông qua các phép tích chập với các bộ lọc. CNN có khả năng xử lý dữ liệu có cấu trúc không gian hoặc chuỗi, như hình ảnh, tín hiệu hoặc chuỗi thời gian. Các mạng CNN thường bao gồm ba thành phần chính: lớp tích chập, lớp gộp, và lớp liên kết đầy đủ được mô tả ở Hình 3.2. Các lớp này kết hợp với nhau để tự động học và tổng quát hóa đặc trưng đầu vào nhằm phục vụ mục tiêu dự đoán hoặc phân loại.



Hình 3.2 Cấu trúc mạng nơ-ron tích chập

Lớp tích chập là thành phần cốt lõi của CNN. Nó sử dụng các hạt nhân tích chập để thực hiện thao tác cửa sổ trượt trên dữ liệu đầu vào, cho phép mô hình học được các đặc điểm cục bộ như xu hướng biến thiên, mức độ dao động và cấu trúc của các vùng dữ liệu. Quá trình tích chập được thể hiện bởi phương trình đề xuất của Cheng và các cộng sự (2021) [21] (3.2):

$$C_n = f(F \bullet X_n + b) \tag{3.2}$$

Trong phương trình (3.2), hạt nhân tích chập là F, $F \in R^{l*d}$. l là chiều cao của hạt nhân tích chập và d là chiều rộng của hạt nhân tích chập. b là tham số độ lệch, f là hàm phi tuyến tính ReLU, X_n là tham số chính liên quan đến mức tiêu thụ điện năng của quá trình nghiền, $n \in [1, \infty)$ là số lượng tham số chính, $\bullet$ là phép toán tích chập và C_n là kết quả của phép tính tích chập. Cửa sổ tích chập trượt xuống với kích thước bước là 1 để thu được vectơ đặc trưng cục bộ $C = (c_1, c_2, \ldots, c_n)$ của tham số đầu vào.

Vai trò của lớp pooling là lấy mẫu kết quả tích chập, giảm kích thước của các vectơ tích chập và ngăn ngừa quá khớp. Tất cả các giá trị riêng được lấy mẫu được kết hợp vào đầu ra $M = (M_1, M_2, \ldots, M_n)$ của CNN. Nghiên cứu này áp dụng phương pháp MaxPooling và có công thức như sau:

$$M_i = Max(C_i) \tag{3.3}$$

Tại đây, M_i tham chiếu đến kết quả của việc áp dụng max pooling cho kết quả tích chập thứ C_i , i ∈ [1, n].

3.2.2. Ứng dụng Long Short-Term Memory

Trong mô hình CNN–LSTM được đề xuất để dự báo tiêu hao điện (kWh/T) trong công đoạn nghiền xi măng, dữ liệu vận hành bao gồm nhiều biến như tốc độ cấp liệu, áp suất, độ rung, tải máy nghiền… được thu thập liên tục theo thời gian. Với mục tiêu dự báo tiêu hao điện trong 20 phút kế tiếp, mô hình sử dụng 60 phút dữ liệu lịch sử gần nhất (window = 60) để làm cơ sở đầu vào.

Bước 1: Cổng quên – bỏ qua thông tin không còn giá trị từ 60 phút trước

Trong khoảng 60 phút gần nhất, không phải mọi thông tin đều cần giữ lại. Ví dụ, sự thay đổi nhỏ trong tải máy nghiền cách đây 50–60 phút có thể đã mất ảnh hưởng đối với tiêu hao điện ở thời điểm hiện tại. Cổng quên của LSTM đánh giá mức độ quan trọng của các thông tin này, và loại bỏ các phần ít ảnh hưởng, giúp mô hình tập trung vào những gì có giá trị nhất để dự báo.

Bước 2: Cổng đầu vào – chọn lọc thông tin mới từ hiện tại để ghi nhớ

Từ dữ liệu của 60 phút gần nhất, tại mỗi thời điểm, mô hình quan sát các biến vận hành như nhiệt độ tăng bất thường, tốc độ cấp liệu thay đổi mạnh… Những tín hiệu này có thể là dấu hiệu sắp xảy ra tăng/giảm tiêu hao điện. Cổng đầu vào của LSTM cho phép mô hình học cách xác định và ghi nhớ những tín hiệu mới có giá trị.

Bước 3: Cập nhật trạng thái bộ nhớ – duy trì "ngữ cảnh" vận hành 60 phút

LSTM sử dụng thông tin từ cổng quên và cổng đầu vào để cập nhật trạng thái ô (cell state) – bộ nhớ nội tại mô tả tổng quan quá trình vận hành trong 60 phút vừa qua. Trạng thái này cho phép mô hình nắm được "ngữ cảnh" toàn cục: thiết bị đang hoạt động ổn định, hay có dấu hiệu dao động thất thường.

Bước 4: Cổng đầu ra – tạo đặc trưng thời gian để dự báo 20 phút kế tiếp

Sau khi cập nhật trạng thái bộ nhớ, mô hình quyết định phần nào trong đó nên được sử dụng để đưa ra dự báo. Đầu ra tại bước này là một vector đặc trưng ngắn gọn chứa thông tin thời gian dài hạn, và được đưa vào các lớp Dense để tạo ra 20 giá trị đầu ra tương ứng với 20 phút tiếp theo.

3.2.3. Huấn luyện và đánh giá mô hình

3.2.3.1. Thuật toán Adam

Trong huấn luyện mạng nơron, các thuật toán lan truyền ngược như Gradient Descent truyền thống có thể dẫn đến tốc độ hội tụ chậm, đặc biệt khi gặp dữ liệu phức tạp hoặc gradient biến thiên mạnh. Thuật toán Adam (Adaptive Moment Estimation) là một phương pháp tối ưu hóa dựa trên gradient, giúp cải thiện đáng kể tốc độ hội tụ bằng cách điều chỉnh động tốc độ học (learning rate) cho từng tham số của mô hình. Adam là sự kết hợp giữa hai phương pháp trước đó là Momentum của Qian (1999) [22] và RMSProp trong tài liệu của Hinton (2012) [23], kế thừa ưu điểm từ cả hai thuật toán trên được xây dựng bởi Kingma và các cộng sự (2015) [24].

Thuật toán Adam sử dụng các trung bình động có trọng số để ước lượng moment bậc nhất (trung bình của gradient) và moment bậc hai (trung bình bình phương của gradient). Các phương trình cập nhật như sau:

- Cập nhật moment bậc nhất:

$$m_t = \beta_1 * m_{t-1} + (1-\beta_1) * g_t \tag{3.4}$$

- Cập nhật moment bậc hai:

$$v_t = \beta_2 * v_{t-1} + (1-\beta_2) * g_t^2 \tag{3.5}$$

Trong đó, g_t là gradient tại thời điểm t, β_1 và β_2 là các hệ số giảm trọng số thường được chọn là 0.9 và 0.999 tương ứng.

Do m_t và v_t ban đầu có xu hướng gần 0, cần hiệu chỉnh độ lệch như sau:

$$m_t = \frac{m_t}{1-\beta_1^t}, v_t = \frac{v_t}{1-\beta_2^t} \tag{3.6}$$

Cuối cùng, tham số w được cập nhật theo công thức:

$$w_{t+1} = w_t - \alpha \frac{m_t}{\sqrt{v_t} + \varepsilon} \tag{3.7}$$

Trong đó, α là tốc độ học ban đầu (learning rate), và ε là một giá trị nhỏ để tránh chia cho 0 (thường lấy 10^{-8}).

Thuật toán Adam có thể thích nghi với từng tham số riêng biệt và tự động điều chỉnh bước nhảy, giúp quá trình huấn luyện ổn định hơn ngay cả khi gradient rất nhỏ hoặc rất lớn. Adam cũng giúp tăng tốc quá trình hội tụ, đặc biệt hữu ích khi huấn luyện các mạng sâu như LSTM, CNN hay các mô hình học sâu có số lượng tham số lớn.

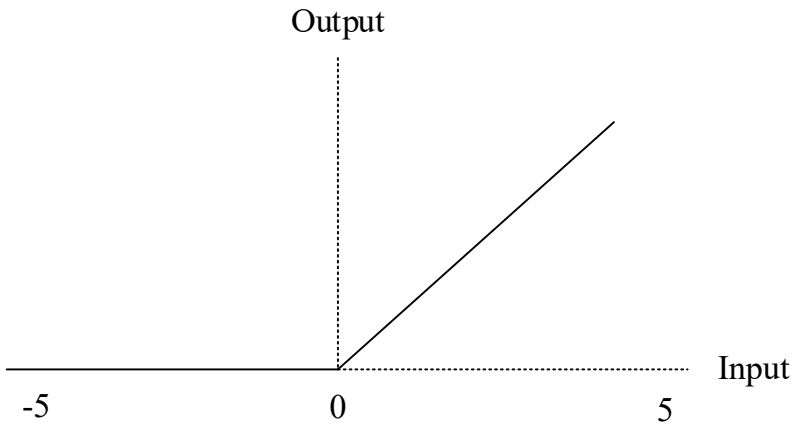
Tuy nhiên, tương tự như thuật toán Levenberg–Marquardt, Adam cũng có một số nhược điểm, chẳng hạn như dễ hội tụ tới cực tiểu không tối ưu trong một số bài toán Reddi và các cộng sự (2018) [25]. Ngoài ra, mặc dù Adam hội tụ nhanh, nhưng trong một số trường hợp cần hiệu chỉnh thêm các siêu tham số như α, β_1, β_2 để đạt hiệu suất tốt nhất.

Một số biến thể nâng cao của Adam như AMSGrad, AdaBound hay AdamW đã được đề xuất nhằm cải thiện độ tin cậy và khả năng tổng quát hóa của mô hình. Chi tiết về Adam và các ứng dụng thực tiễn có thể tham khảo trong các tài liệu hướng dẫn học sâu như TensorFlow, PyTorch hoặc các báo cáo nghiên cứu học thuật và Ruder các cộng sự (2016) [26].

3.2.3.2. Hàm kích hoạt ReLU

Một hàm kích hoạt ReLU được áp dụng sau mỗi phép toán tích chập. Hàm này giúp mạng học được mối quan hệ phi tuyến tính giữa các đặc điểm trong hình ảnh, do đó làm cho mạng mạnh mẽ hơn để xác định các mẫu khác nhau và cũng giúp giảm thiểu các vấn đề về độ dốc biến mất được đề cập bởi Nair và Hinton (2010) [27]. Công thức hàm Relu như sau:

$$F(x) = \max(0, x) \tag{3.8}$$



Hình 3.3 Hàm kích hoạt ReLU

Hàm này không chỉ giúp mô hình hóa các mối quan hệ phi tuyến tính trong dữ liệu mà còn có vai trò quan trọng trong việc giảm thiểu hiện tượng gradient biến mất trong quá trình lan truyền ngược được trích dẫn ở sách Deep Learning (2016) [28]. Nhờ đó, mạng có thể học được các đặc trưng phức tạp và sâu hơn, đồng thời tăng khả năng phân biệt giữa các mẫu đầu vào có tính chất khác nhau.

Việc sử dụng ReLU cũng giúp tăng tốc độ huấn luyện nhờ tính đơn giản về mặt tính toán và khả năng giữ lại các giá trị dương, loại bỏ các kích hoạt âm không có đóng góp hữu ích.

3.2.3.3. Hàm kích hoạt Tanh

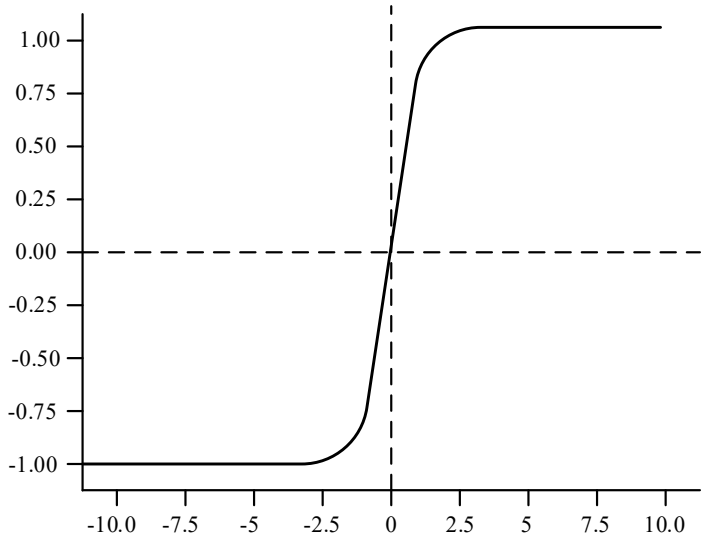
Hàm kích hoạt Tanh (hyperbolic tangent) là một trong những hàm phi tuyến được sử dụng phổ biến trong các mô hình mạng nơ-ron nhân tạo, đặc biệt là trong các lớp ẩn. Hàm được định nghĩa theo biểu thức toán học:

$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{3.9}$$

Hàm Tanh là một hàm liên tục, khả vi trên toàn miền xác định R, với giá trị đầu ra nằm trong khoảng $(-1,1)$ được mô tả ở Hình 3.4. Một trong những đặc điểm nổi bật của Tanh là tính chất lẻ, tức là $\tanh(-x) = -\tanh(x)$,cho phép đầu ra của nơ-ron có sự phân bố đối xứng quanh 0. Tính chất này giúp giảm hiện tượng dịch trung bình (mean shift) trong quá trình lan truyền tín hiệu, từ đó cải thiện hiệu quả huấn luyện của mô hình [28].

Trong thực tiễn, hàm Tanh thường được sử dụng trong các kiến trúc mạng như mạng nơ-ron truyền thẳng (feedforward neural networks), mạng hồi tiếp và đặc biệt là trong các khối bộ nhớ của LSTM. Khi dữ liệu đã được chuẩn hóa với trung bình gần 0,

việc sử dụng Tanh sẽ giúp mô hình hội tụ nhanh hơn và ổn định hơn trong quá trình huấn luyện được trích trong sách Neural Networks and Learning Machines (2009) [29].



Hình 3.4 Hàm kích hoạt Tanh

3.2.3.4. Các chỉ số đánh giá

Trong các công thức dưới đây, y_i giá trị thực tế, $\hat{y}_i$ là giá trị dự báo của mức tiêu hao điện và $\overline{y_i}$ biểu thị giá trị trung bình của các mẫu tiêu hao điện năng, i =1, 2, …, n với n là dung lượng mẫu.

Sai số tuyệt đối trung bình MAE (Mean Absolute Error)

$$MAE = \frac{1}{n}\sum_{i=1}^{n}\left|y_i - \hat{y}_i\right| \tag{3.10}$$

Giá trị MAE nằm trong khoảng (0,+∞). MAE biểu thị biên độ trung bình của sai số mô hình nhưng không nói lên xu hướng lệch của giá trị dự báo và thực tế. Khi MAE = 0, giá trị của mô hình hoàn toàn trùng khớp với giá trị quan trắc, mô hình được xem là "lý tưởng". Thông thường MAE được sử dụng cùng với ME để đánh giá độ tin cậy. Chẳng hạn, nếu MAE của sản phẩm khác biệt hẳn so với ME thì việc hiệu chỉnh là hết sức mạo hiểm. Trong trường hợp ngược lại, khi mà MAE và ME tương đối gần với nhau thì có thể dùng ME để hiệu chỉnh sản phẩm dự báo một cách đáng tin cậy.

Sai số bình phương trung bình MSE (Mean Square Error)

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \hat{y}_i\right)^2 \tag{3.11}$$

MSE là trung bình của tổng bình phương của hiệu giữa các giá trị mô hình và quan trắc, phản ánh mức độ dao động của sai số. Mô hình là "lí tưởng" nếu MSE=0.

Sai số phần trăm tuyệt đối trung bình MAPE (Mean Absolute Pecent Error)

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \tag{3.12}$$

Sai số bình phương trung bình MSE cho thấy độ lớn của sai số hay bình phương sai số dự báo, thường dùng trong so sánh độ chính xác của các mô hình dự báo khác nhau. Tuy nhiên, để đánh giá độ chính xác của một mô hình ta thường dùng MAPE. Sai số phần trăm tuyệt đối trung bình MAPE tính đến độ lớn tương đối của độ lệch dự báo so với độ lớn giá trị thực, mô hình có MAPE càng nhỏ thì dự báo càng chính xác.

Sai số bình phương trung bình quân phương (RMSE - Root mean square Error)

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{3.13}$$

Sai số bình phương trung bình là một trong những đại lượng cơ bản và thường được sử dụng phổ biến cho việc đánh giá kết quả của mô hình dự báo. Người ta thường hay sử dụng đại lượng sai số bình phương trung bình quân phương (RMSE) biểu thị độ lớn trung bình của sai số. Đặc biệt RMSE rất nhạy với những giá trị sai số lớn. Do đó nếu RMSE càng gần MAE sai số mô hình càng ổn định và có thể thực hiện việc hiệu chỉnh sản phẩm mô hình. Giống như MAE, RMSE không chỉ ra độ lệch giữa giá trị dự báo và giá trị thực tế. Giá trị của RMSE nằm trong khoảng (0, +∞). Khi so sánh MAE và RMSE ta thấy: RMSE ≥ MAE. Còn RMSE = MAE khi và chỉ khi tất cả các sai số có độ lớn như nhau: RMSE = MAE = 0.

Hệ số xác định (Coefficient of Determination)

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \overline{y_i})^2} \tag{3.14}$$

Hệ số xác định (R^2) cho phép đánh giá mức độ phù hợp của mô hình dự báo so với tập dữ liệu thực tế. Giá trị của R^2 nằm trong khoảng từ 0 đến 1, trong đó giá trị càng gần 1 cho thấy mô hình dự báo càng chính xác. R^2 phản ánh tỷ lệ phương sai của dữ liệu đầu ra được giải thích bởi mô hình. Khi $R^2 = 1$, mô hình hoàn toàn phù hợp với dữ liệu thực tế; ngược lại, $R^2 = 0$ thể hiện rằng mô hình không giải thích được bất kỳ

phương sai nào của dữ liệu. Giá trị R^2 cao chứng tỏ mối quan hệ mạnh giữa giá trị dự báo và giá trị thực tế, từ đó khẳng định khả năng dự báo hiệu quả của mô hình.

3.2.4. Tối ưu hóa LHS cho thông số vận hành

Phương pháp Lấy mẫu Latin Hypercube trích xuất từ nghiên cứu của Helton và Davis (2003) [30] hoạt động bằng cách chia miền giá trị của mỗi thành phần của vector mẫu (với $i = 1,2,3,...,N$) thành $M = n$ tập con (strata) không giao nhau có xác suất bằng nhau, ký hiệu là (với $i = 1,2,3,...,N$ và $k = 1,2,3,...,M$). Các mẫu của mỗi thành phần vector được lấy từ các tập con tương ứng theo công thức:

$$x_{ik} = D_X^{-1}(U_{ik}) \text{ với } i = 1,2,3,...,N \text{ và } k = 1,2,3,...,M \quad (3.15)$$

Trong đó, x_{ik} là các mẫu ngẫu nhiên độc lập (iid - independent and identically distributed) được phân phối đồng đều trên khoảng với:

Các mẫu được tập hợp lại bằng cách ghép ngẫu nhiên các thành phần của vector đã tạo. Cụ thể, một giá trị được chọn ngẫu nhiên từ mỗi thành phần vector (không trùng lặp), và các giá trị này được ghép nhóm để tạo thành một mẫu hoàn chỉnh. Quá trình này được lặp lại $M = n$ lần.

Do các thành phần mẫu được ghép cặp ngẫu nhiên, một tập hợp LHS không phải là duy nhất, với tổng cộng tổ hợp có thể xảy ra. Vì lý do này, các thuật toán LHS cải tiến thường được sử dụng để tối ưu hóa việc ghép cặp dựa trên các tiêu chí cụ thể, chẳng hạn như giảm tương quan giữa các biến hoặc tăng cường tính bao phủ không gian mẫu.

Quy trình lấy mẫu của thuật toán LHS:

Bước 1: Khởi tạo không gian tham số

Đầu tiên, không gian khảo sát được xác định dựa trên hai tham số đầu vào là tốc độ cấp liệu và tốc độ cấp nước. Đối với mỗi mẫu quan sát, khoảng giá trị của tốc độ cấp liệu được đặt là ±5 T/h so với giá trị tại thời điểm đang xét, trong khi tốc độ cấp nước được đặt là ±0.2 m³/h quanh giá trị tại thời điểm đang xét tương ứng.

Bước 2: Phân bố đều và ánh xạ về miền thực

Sau khi xác định được không gian khảo sát, bước tiếp theo là chia đều miền giá trị của mỗi tham số thành n=100 khoảng giá trị bằng nhau, hay còn gọi là các strata. Trong mỗi strata, một giá trị được chọn ngẫu nhiên theo phân phối đều. Các giá trị này ban đầu nằm trong khoảng [0, 1], sau đó được ánh xạ tuyến tính trở lại miền giá trị thực của từng biến thông qua hàm biến đổi tương ứng. Nhờ đó, mỗi mẫu sinh ra đều thuộc một khoảng

duy nhất, không bị trùng lặp, đồng thời đảm bảo phân bố đều trên toàn không gian khảo sát.

Bước 3: Áp dụng tổ hợp mẫu vào mô hình dự báo và đánh giá kết quả

Từng tổ hợp đầu vào được tạo thành từ cặp giá trị tốc độ cấp liệu và tốc độ cấp nước sẽ được tích hợp vào tập dữ liệu vận hành tại thời điểm hiện tại. Mô hình dự báo CNN-LSTM đã được huấn luyện sẽ sử dụng tổ hợp này để ước lượng mức tiêu hao điện năng tương ứng. Trong quá trình này, một điều kiện vận hành được áp dụng: sản lượng đầu ra không được vượt quá tốc độ cấp liệu đầu vào. Những tổ hợp không thỏa mãn điều kiện này sẽ bị loại bỏ nhằm đảm bảo tính khả thi trong thực tế.

Bước 4: Lựa chọn tổ hợp thông số tối ưu

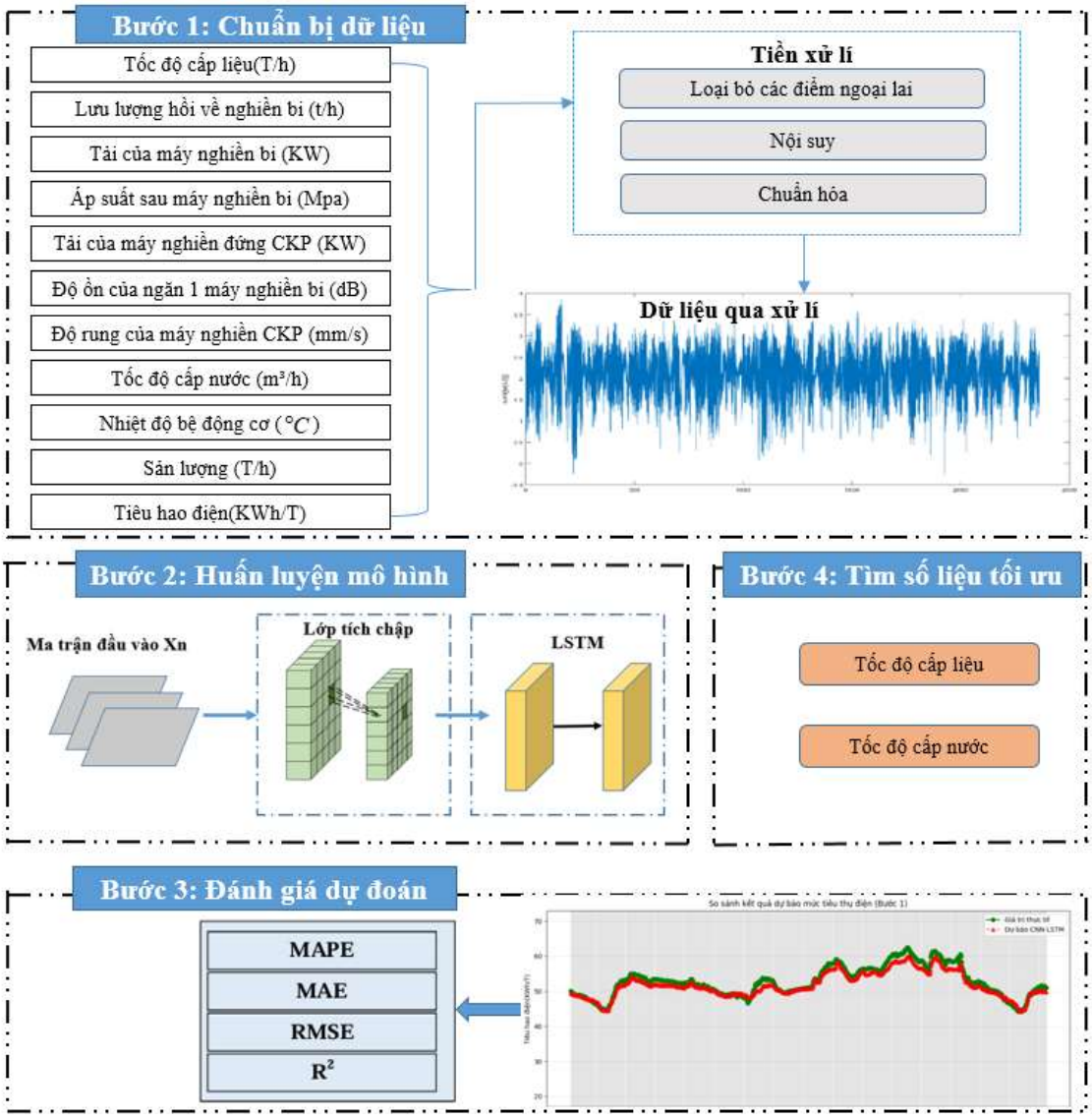
Sau khi đánh giá toàn bộ các tổ hợp hợp lệ, kết quả dự đoán tiêu hao điện sẽ được so sánh để tìm ra phương án có giá trị thấp nhất. Tổ hợp thông số tương ứng với mức tiêu hao điện nhỏ nhất sẽ được xem là phương án vận hành tối ưu tại thời điểm đó. Kết quả tối ưu bao gồm: tốc độ cấp liệu và tốc độ cấp nước tương ứng, giá trị tiêu hao điện sau tối ưu, cùng với mức cải thiện so với giá trị ban đầu. Việc lựa chọn này góp phần đề xuất cấu hình vận hành tiết kiệm điện năng hơn, qua đó nâng cao hiệu quả sử dụng năng lượng trong nhà máy xi măng.

3.3. Đề xuất mô hình kết hợp giữa dự báo và tối ưu hoá năng lượng

3.3.1. Xây dưng mô hình CNN-LSTM-LHS

Trong bối cảnh tối ưu hóa và dự báo tiêu hao điện năng cho công đoạn nghiền xi măng, không một thuật toán đơn lẻ nào có thể đáp ứng đầy đủ các yêu cầu về xử lý dữ liệu phức tạp, khả năng dự báo chính xác và tìm kiếm thông số vận hành tối ưu. Việc kết hợp ba thuật toán—xử lý dữ liệu, mô hình học sâu CNN-LSTM và LHS giúp tận dụng thế mạnh của từng phương pháp: xử lý dữ liệu đảm bảo chất lượng đầu vào, mô hình học sâu tăng độ chính xác dự báo, còn LHS hỗ trợ việc tìm kiếm và đánh giá các tổ hợp thông số tối ưu một cách toàn diện và khoa học. Nhờ đó, quy trình không chỉ nâng cao hiệu quả dự báo mà còn tối ưu hóa được hiệu suất vận hành của hệ thống nghiền xi măng.

Hình 3.5 minh họa trên trình bày một cách trực quan, logic và toàn diện về quy trình các bước thực hiện nghiên cứu xây dựng mô hình dự báo và tối ưu hóa điện năng tiêu thụ trong công đoạn nghiền xi măng.



Hình 3.5 Mô hình đề xuất kết hợp

Quy trình này được phân chia thành bốn bước chính, mỗi bước đều đóng vai trò quan trọng và bổ trợ lẫn nhau nhằm đảm bảo tính hiệu quả, chính xác của hệ thống dự báo và tối ưu hóa vận hành.

Bước 1: Chuẩn bị dữ liệu, hình vẽ liệt kê đầy đủ các biến đầu vào then chốt liên quan đến quá trình nghiền như: tốc độ cấp liệu, lưu lượng hồi về nghiền bi, tải của máy nghiền bi và CKP, áp suất, độ ồn, độ rung, tốc độ cấp nước, nhiệt độ động cơ, sản lượng và tiêu hao điện. Các biến này phản ánh toàn diện tình trạng vận hành thực tế của hệ thống và là cơ sở dữ liệu đầu vào cho các mô hình trí tuệ nhân tạo. Đặc biệt, phần tiền xử lý dữ liệu (loại bỏ ngoại lai, nội suy, chuẩn hóa) được chú trọng nhằm nâng cao độ tin cậy, giảm nhiễu, đảm bảo dữ liệu đầu vào sạch, nhất quán và phù hợp với các yêu cầu toán học của mô hình học máy. Biểu đồ minh họa dữ liệu sau xử lý cho thấy quy

mô, tính liên tục và chất lượng của tập dữ liệu, góp phần nâng cao độ chính xác của các bước tiếp theo.

Bước 2: Huấn luyện mô hình mô tả quá trình xây dựng mạng học sâu, kết hợp giữa lớp tích chập của CNN và LSTM. Đây là sự kết hợp ưu việt: CNN có thể tự động trích xuất các đặc trưng quan trọng từ dữ liệu nhiều chiều, trong khi LSTM lại đặc biệt hiệu quả khi xử lý chuỗi thời gian nhờ khả năng ghi nhớ các trạng thái trong quá khứ. Sự phối hợp này giúp mô hình vừa tận dụng các mối quan hệ phức tạp giữa các biến đầu vào, vừa nắm bắt được các xu hướng, biến động của quá trình nghiền qua thời gian. Điều này góp phần nâng cao đáng kể độ chính xác của dự báo tiêu thụ điện năng.

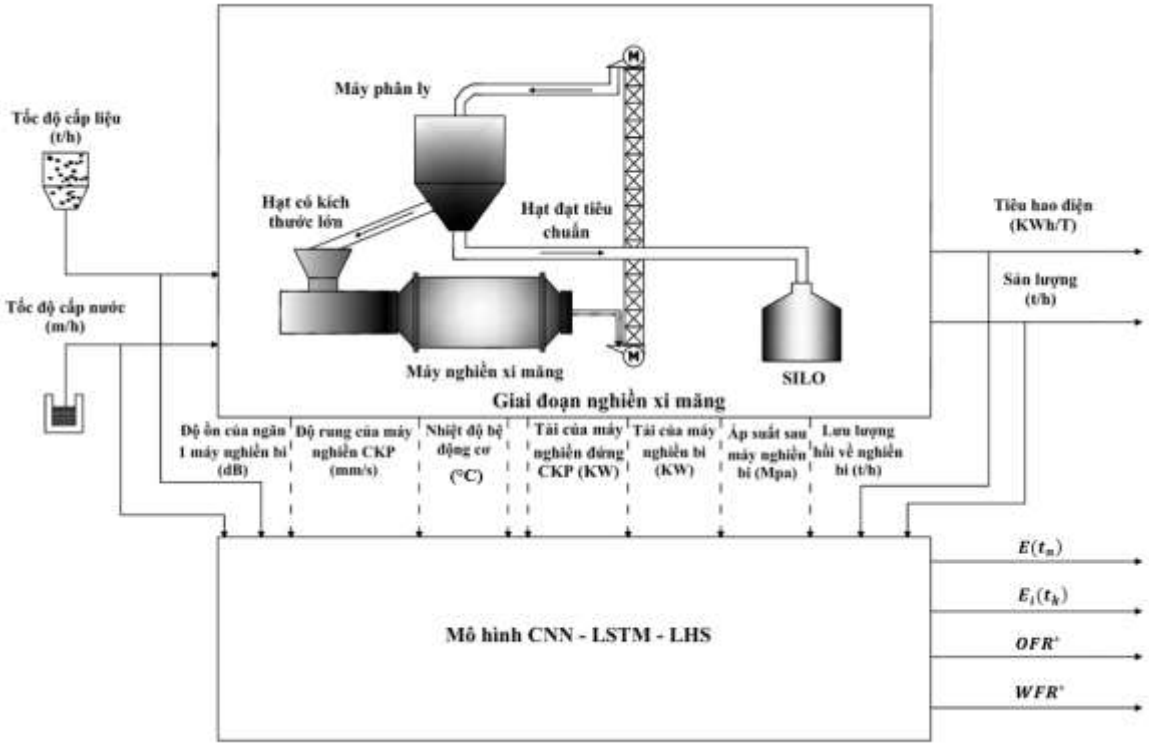
Bước 3: Đánh giá dự đoán sử dụng nhiều chỉ số đánh giá khách quan như MAPE, MAE, RMSE và hệ số xác định R^2. Việc sử dụng đa dạng các chỉ số giúp đánh giá toàn diện hiệu quả mô hình ở nhiều góc độ: sai số tuyệt đối, sai số bình phương trung bình và mức độ giải thích biến thiên của dữ liệu thực. Biểu đồ minh họa so sánh giữa giá trị dự đoán và thực tế cho thấy mức độ trùng khớp cao, chứng tỏ mô hình dự báo hoạt động hiệu quả và có khả năng ứng dụng thực tiễn cao.

Bước 4: Tìm số liệu tối ưu là điểm nhấn quan trọng về mặt ứng dụng, khi hai biến vận hành quan trọng nhất (tốc độ cấp liệu và tốc độ cấp nước) được lựa chọn để tối ưu hóa. Thông qua các thuật toán tối ưu, hệ thống có thể đề xuất các giá trị vận hành phù hợp nhất nhằm giảm thiểu điện năng tiêu thụ mà vẫn đảm bảo hiệu quả sản xuất. Đây là bước chuyển mạnh mẽ từ dự báo thụ động sang chủ động tối ưu hóa vận hành, thể hiện rõ giá trị ứng dụng thực tiễn của nghiên cứu.

3.3.2. Giải thuật để triển khai mô hình CNN-LSTM-LHS

Hệ thống được xây dựng dựa trên quy trình nghiền xi măng thực tế tại một nhà máy ở Việt Nam. Bộ dữ liệu sử dụng trong nghiên cứu bao gồm 44.638 mẫu quan sát liên tục, được ghi nhận trong điều kiện vận hành thực tế của nhà máy. Mỗi mẫu dữ liệu đại diện cho một chu kỳ vận hành, với thời gian lấy mẫu là 1 phút/lần và được thu thập đều đặn trong suốt 1 tháng, từ ngày 1/3/2025 đến ngày 31/3/2025. Việc thu thập dữ liệu theo phương pháp này giúp đảm bảo tính đại diện và độ tin cậy của bộ dữ liệu, phục vụ tốt cho quá trình nghiên cứu, phân tích.

Dựa trên các mẫu số liệu thu thập được, chúng tôi đã xây dựng hệ thống các biến đầu vào và đầu ra ở Hình 3.6, qua đó giúp làm rõ quy trình vận hành trong công đoạn nghiền xi măng và hỗ trợ cho việc phân tích, dự đoán cũng như tối ưu hóa hiệu quả tiêu thụ điện năng tại nhà máy.



Hình 3.6 Mô hình CNN–LSTM–LHS dự đoán tiêu thụ điện dựa trên dữ liệu vận hành nghiền xi măng

Hình trên mô tả mô hình kết hợp giữa hai phương pháp CNN-LSTM và LHS nhằm tối ưu hóa và dự báo tiêu hao điện năng trong công đoạn nghiền xi măng. Đầu vào của hệ thống bao gồm các thông số quan trọng như tốc độ cấp liệu (tấn/giờ), tốc độ cấp nước (m^3/h), cùng với các thông số vận hành khác như độ ồn của máy nghiền bi, độ rung của máy nghiền CKP, nhiệt độ ổ bi động cơ, tải máy nghiền đứng và nghiền bi, áp suất sau máy nghiền bi, lưu lượng hồi về nghiền bi và sản lượng (tấn/giờ). Tất cả các thông số này được đưa vào giai đoạn nghiền xi măng, bao gồm máy cấp liệu, máy nghiền, máy phân ly và silo chứa xi măng. Sau quá trình vận hành, hệ thống sẽ thu thập các chỉ số đầu ra như tiêu hao điện năng (kWh/tấn) và sản lượng (tấn/giờ).

Trong mô hình, các thông số vận hành sẽ được đưa vào mạng nơ-ron CNN-LSTM để dự báo lượng tiêu hao điện tương ứng với từng điều kiện làm việc tại các thời điểm khác nhau. CNN đảm nhiệm vai trò trích xuất đặc trưng từ dữ liệu đầu vào, trong khi LSTM giúp học và dự báo các chuỗi dữ liệu theo thời gian, từ đó nâng cao độ chính xác của kết quả dự báo. Song song với đó, phương pháp LHS được sử dụng để sinh mẫu và tối ưu hóa các thông số đầu vào như tốc độ cấp liệu và tốc độ cấp nước, dựa trên các giới hạn kỹ thuật và điều kiện vận hành thực tế. Mục tiêu của bước này là tìm ra các giá trị tối ưu giúp giảm tiêu hao điện năng mà vẫn đảm bảo chất lượng và sản lượng đầu ra.

Kết quả tối ưu hóa từ mô hình LHS sẽ tạo ra các tập thông số đầu vào khả thi. Những tập thông số này tiếp tục được đưa vào mô hình CNN-LSTM để dự báo hiệu quả về tiêu hao điện, từ đó lựa chọn ra tổ hợp thông số vận hành tối ưu nhất cho quá trình sản xuất thực tế. Như vậy, mô hình trên không chỉ giúp nhà máy xi măng kiểm soát tốt tiêu hao điện năng mà còn góp phần nâng cao hiệu quả sản xuất thông qua việc lựa chọn các tham số vận hành tối ưu một cách khoa học và tự động.

Quá trình xây dựng mô hình dự đoán bao gồm hai giai đoạn chính: trích xuất đặc trưng không gian bằng CNN, khai thác quan hệ chuỗi thời gian bằng LSTM và tối ưu hóa thông qua LHS

3.3.2.1. Biểu diễn dữ liệu đầu vào

Giả sử tại thời điểm t, ta có vector dữ liệu đầu vào gồm 11 biến đầu vào là: tốc độ cấp liệu (tấn/giờ), tốc độ cấp nước (m^3/h), độ ồn của máy nghiền bi, độ rung của máy nghiền CKP, nhiệt độ ổ bi động cơ, tải máy nghiền đứng và nghiền bi, áp suất sau máy nghiền bi, lưu lượng hồi về nghiền bi và sản lượng (tấn/giờ).

Xét chuỗi dữ liệu liên tiếp qua nhiều bước thời gian, đầu vào có thể mở rộng thành:

$$Z = \{z(t-h+1), z(t-h+2), ..., z(t)\} \tag{3.16}$$

với h là độ dài chuỗi thời gian đầu vào.

3.3.2.2. Trích xuất đặc trưng bởi CNN

Dữ liệu Z được đưa vào mạng CNN để trích xuất các đặc trưng không gian quan trọng. Hàm trích xuất đặc trưng được biểu diễn dưới dạng:

$$\varphi(t_j) = F_{CNN}(Z) \tag{3.17}$$

trong đó:

- $\varphi(t_j)$ là vector đặc trưng sau khi trích xuất.
- $F_{CNN}(Z)$ bao gồm các phép tích chập, kích hoạt, chuẩn hóa và gộp đặc trưng.

3.3.2.3. Dự đoán theo chuỗi thời gian bởi LSTM

Vector đặc trưng $\varphi(t_j)$ tiếp tục được đưa vào mạng LSTM để khai thác mối quan hệ theo thời gian, đặc biệt là những ảnh hưởng dài hạn đến mức tiêu thụ điện năng trong tương lai.

Phương trình dự đoán có dạng:

$$E(t_n) = M_{LSTM}(\varphi(t_j)) \tag{3.18}$$

Trong đó:

- M_{LSTM} hàm học được từ LSTM,
- $E(t_n)$ là mức tiêu thụ điện dự đoán tại thời điểm tương lai t_n.
- t_n được tính bằng: $t_n = n.\Delta t\ (t_n > t_j)$.

3.3.2.4. Tối ưu hóa thông qua LHS

Tiếp theo, thuật toán Latin Hypercube Sampling sẽ sinh ra các tổ hợp biến đầu vào: $\{a_i, b_i\}_{i=1}^{m}$ với a_j, b_j là giá trị tham số OFR (tốc độ cấp liệu), WFR (tốc độ cấp nước).

Đối với mỗi mẫu $x_i \in LHS_{samples}$, tính giá trị tiêu thụ điện tương ứng:

$$E_i(t_n) = M_{LSTM}(F_{CNN}(x_i)) \tag{3.19}$$

Mục tiêu của bài toán tối ưu là tìm tổ hợp x^* sao cho:

$$x^* = \arg\min_{x_i} E_i(t_n)$$

Kết quả cuối cùng là

$$E_i(t_k) = M_{LSTM}(F_{CNN}(x^*)) \tag{3.20}$$

trong đó:

- x^* là bộ tham số OFR, WFR tối ưu tại thời điểm t_k.
- Tại thời điểm t_k , t_k được tính bởi $t_k = k.\Delta t\ (t_n > t_k > t_j)$.

$E_i(t_k)$ là giá trị tiêu thụ điện năng tối ưu tương ứng.

Chương 4: THỬ NGHIỆM VÀ ĐÁNH GIÁ MÔ HÌNH

4.1. Quá trình thử nghiệm

Do tính ngẫu nhiên vốn có trong quá trình huấn luyện các mô hình học sâu, mỗi mô hình được đánh giá lặp lại nhiều lần để đảm bảo tính ổn định và độ tin cậy của kết quả. Trong nghiên cứu này, mô hình CNN-LSTM được đề xuất nhằm dự báo tiêu hao điện năng với mục tiêu tối ưu độ chính xác và khả năng khái quát hóa.

Tối ưu hóa siêu tham số bằng Grid Search

Trước khi xác định kiến trúc cuối cùng cho mô hình CNN-LSTM, chúng tôi đã tiến hành quá trình tối ưu siêu tham số bằng phương pháp tìm kiếm lưới (Grid Search). Các siêu tham số được lựa chọn để tối ưu bao gồm:

- Số đơn vị trong lớp LSTM (lstm_units)
- Số lớp LSTM chồng nhau (lstm_layers)
- Kích thước lô huấn luyện (batch_size)

Phạm vi khảo sát cho từng siêu tham số được thể hiện trong Bảng 4.1.

Bảng 4.1 Phạm vi siêu tham số khảo sát trong Grid Search

Tham số	**Giá trị khảo sát**
lstm_units	32, 64, 128
lstm_layers	1, 2, 3
batch_size	32, 64, 128

Chúng tôi sử dụng chỉ số R^2 (R-squared) trên tập xác thực để đánh giá hiệu suất của từng tổ hợp siêu tham số. Kết quả từ quá trình tìm kiếm cho thấy tổ hợp siêu tham số tối ưu là:

- lstm_units: 64
- lstm_layers: 1
- batch_size: 32

Tổ hợp này giúp mô hình đạt được điểm số R^2 cao nhất là 0.97, cho thấy khả năng dự báo chính xác và ổn định. Việc chỉ sử dụng một lớp LSTM giúp mô hình tránh hiện tượng quá khớp (overfitting), trong khi số lượng đơn vị 64 cung cấp độ biểu diễn đủ mạnh để học các phụ thuộc phức tạp trong chuỗi thời gian. Kích thước lô 32 cũng cho thấy sự cân bằng tốt giữa tốc độ hội tụ và độ ổn định trong huấn luyện.

Bảng 4.2 là kết quả sau khi xác định được cấu hình tham số tối ưu, chúng tôi xây dựng mô hình CNN-LSTM với kiến trúc gồm hai tầng Conv1D để trích xuất đặc trưng cục bộ từ chuỗi đầu vào, theo sau là một lớp LSTM có 64 đơn vị. Các lớp Dense ở cuối được sử dụng để dự báo tiêu thụ điện năng cho nhiều bước tiếp theo. Mỗi lớp chính đều được xen kẽ với Dropout để hạn chế quá khớp.

Bảng 4.2 Cấu trúc mô hình CNN-LSTM

Mô-đun	**Thành phần**	**Thông số**
CNN	Conv1D	64, 32
	Kích thước hạt nhân	3, 2
	Hàm kích hoạt	ReLU
LSTM	Units	64
	Batch size	32
	Cửa sổ thời gian	60
	Đầu ra	20
	Hàm kích hoạt	Tanh

Dữ liệu đầu vào được xử lý thành chuỗi thời gian có độ dài cửa sổ là 60 bước, đầu ra là 20 bước dự báo tiếp theo. Tất cả các đặc trưng đầu vào đều được chuẩn hóa bằng MinMaxScaler để đảm bảo ổn định khi huấn luyện.

Mô hình được huấn luyện với tối đa 50 epoch, sử dụng batch size = 32 như được xác định từ grid search. Để tránh hiện tượng quá khớp và rút ngắn thời gian huấn luyện, EarlyStopping được áp dụng với patience = 10, nghĩa là nếu giá trị val_loss không cải thiện trong 10 epoch liên tiếp, huấn luyện sẽ dừng lại. Đồng thời, ModelCheckpoint được sử dụng để lưu mô hình có hiệu suất tốt nhất trên tập xác thực.

Việc tích hợp tối ưu siêu tham số, kiến trúc CNN-LSTM kết hợp, và các chiến lược huấn luyện hiện đại đã giúp mô hình đạt được hiệu suất cao và ổn định trong bài toán

dự báo tiêu thụ điện năng nhiều bước. Các kết quả định lượng và trực quan hóa chi tiết sẽ được trình bày trong phần tiếp theo.

Sau khi xây dựng và huấn luyện thành công mô hình dự báo lượng điện tiêu thụ bằng mạng nơ-ron kết hợp CNN-LSTM, bước tiếp theo trong quá trình nghiên cứu là tiến hành tối ưu hóa các thông số vận hành đầu vào nhằm tìm ra tổ hợp điều kiện vận hành giúp giảm thiểu mức tiêu thụ điện năng, đồng thời duy trì ổn định sản lượng đầu ra.

Trong phạm vi nghiên cứu này, chúng tôi tập trung thay đổi hai thông số đầu vào chính, bao gồm: tốc độ cấp liệu (tấn/giờ) và tốc độ cấp nước (m^3/giờ)

Các thông số vận hành khác được giữ nguyên không thay đổi nhằm giới hạn phạm vi tối ưu trong không gian tham số quan trọng nhất và phù hợp với điều kiện thực tế vận hành tại nhà máy.

Để đảm bảo việc thay đổi hai thông số đầu vào không làm ảnh hưởng đến sản lượng sản xuất thực tế, chúng tôi đã tiến hành phân tích mối quan hệ giữa sản lượng (tấn/giờ) và tốc độ cấp liệu dựa trên bộ dữ liệu lịch sử được trích xuất và tổng hợp từ hệ thống XHQ. Cụ thể:

- Khi sản lượng đầu ra được duy trì ở mức 270 tấn/giờ, tốc độ cấp liệu tương ứng nằm trong khoảng 278 – 283 tấn/giờ.
- Khi sản lượng là 255 tấn/giờ, tốc độ cấp liệu được duy trì trong khoảng 263 – 268 tấn/giờ.

Từ đó, chúng tôi xây dựng thuật toán tìm kiếm vùng lân cận của các giá trị đầu vào theo nguyên tắc sau:

- Tốc độ cấp liệu tại mỗi thời điểm sẽ được mở rộng ±5 tấn/giờ so với giá trị gốc, tạo ra một dải giá trị tham khảo trong phạm vi an toàn sản xuất.
- Tốc độ cấp nước cũng được điều chỉnh trong khoảng ±0,2 m^3/giờ so với giá trị gốc tại cùng thời điểm. Phạm vi này được lựa chọn dựa trên việc tốc độ cấp nước từ 0,08 tới 2,272 m^3/giờ nên trong khoảng này lấy 10 % khoảng vận hành thực tế và nhằm đảm bảo không làm ảnh hưởng đến điều kiện nghiền hoặc gây mất ổn định cho thiết bị.

Sau khi xác định không gian tìm kiếm cho hai thông số đầu vào, chúng tôi tiến hành lấy mẫu ngẫu nhiên 100 điểm trong không gian hai chiều đó bằng phương pháp Latin Hypercube Sampling. Đây là kỹ thuật lấy mẫu hiệu quả, cho phép phân bố đồng đều các điểm mẫu trong toàn bộ không gian tìm kiếm mà không bị trùng lặp, giúp tăng xác suất tìm ra các tổ hợp thông số tối ưu.

Mỗi điểm lấy mẫu tương ứng với một cặp giá trị tốc độ cấp liệu – tốc độ cấp nước, sau đó được đưa vào mô hình CNN-LSTM đã huấn luyện trước để ước tính lượng điện tiêu thụ tương ứng. Kết quả mô phỏng từ 100 tổ hợp đầu vào sẽ được so sánh, và tổ hợp có mức tiêu thụ điện năng thấp nhất sẽ được lựa chọn làm giải pháp vận hành tối ưu cho thời điểm đang xét.

4.2. Kết quả và thảo luận

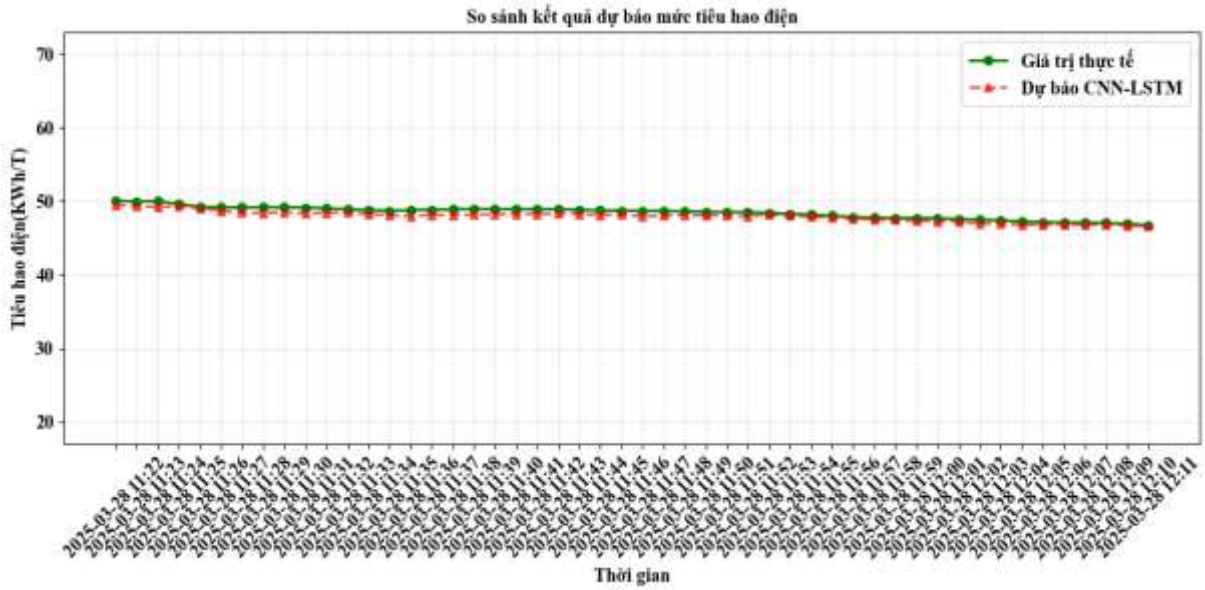
Các chỉ số đánh giá mô hình:

- MAE = 0.62
- MSE = 0.62
- RMSE = 0.79
- MAPE = 1.34%
- RSE = 0.028
- R^2 = 0.97

Kết quả thực nghiệm cho thấy mô hình CNN-LSTM đã đạt được hiệu suất dự báo xuất sắc trong việc dự đoán tiêu thụ điện năng. Với hệ số xác định $R^2 = 0.97$, mô hình giải thích được 97% biến động của dữ liệu thực tế, thể hiện khả năng dự báo có độ tin cậy cao. Điều này được củng cố thêm bởi giá trị MAPE chỉ 1.34%, cho thấy sai số tương đối trung bình rất thấp và đạt chuẩn "rất tốt" theo thang đánh giá quốc tế (MAPE < 5%).

Các chỉ số sai số tuyệt đối cũng khẳng định tính hiệu quả của mô hình. MAE = 0.62 và RMSE = 0.79 cho thấy sai số trung bình giữa giá trị dự báo và thực tế ở mức rất thấp. Đặc biệt, RSE = 0.028 (2.8%) thể hiện sai số tương đối chỉ chiếm một tỷ lệ nhỏ so với độ biến động tự nhiên của chuỗi dữ liệu, điều này chứng tỏ mô hình có khả năng dự báo chính xác và ổn định.

Hình 4.1 trình bày sự so sánh giữa giá trị thực tế và giá trị dự báo mức tiêu thụ điện năng (kWh/Tấn) trong khoảng thời gian từ 11 giờ 22 phút đến 12 giờ 11 phút ngày 28 tháng 3 năm 2025, sử dụng mô hình CNN-LSTM. Trục hoành thể hiện thời gian theo ngày, trục tung biểu diễn mức tiêu hao điện năng. Biểu đồ được xây dựng nhằm trực quan hóa khả năng dự báo của mô hình trên tập dữ liệu thời gian thực, đồng thời hỗ trợ đánh giá mức độ tương quan giữa kết quả dự báo và số liệu thực tế trong quá trình vận hành.

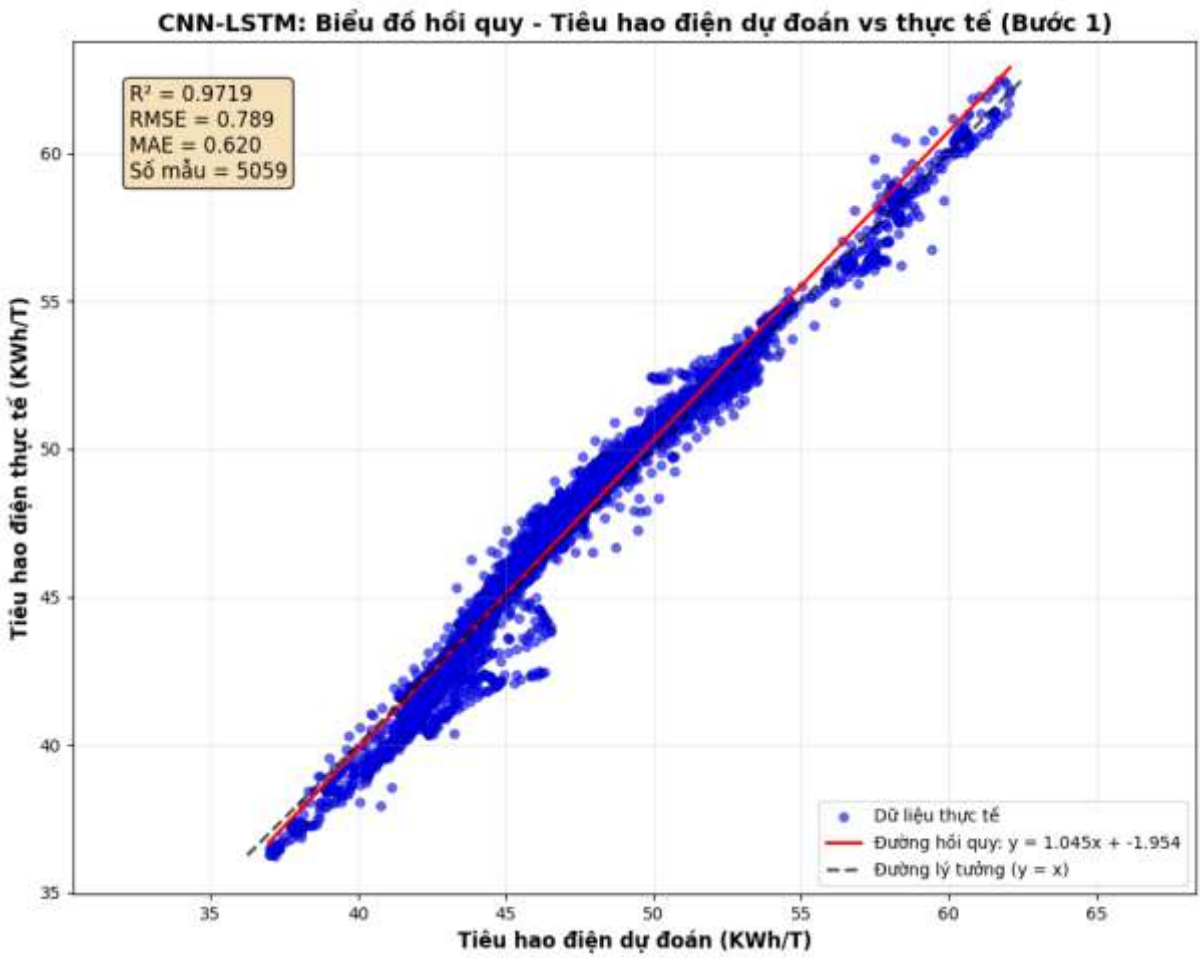


Hình 4.1 Biểu đồ so sánh kết quả dự đoán và thực tế

Phân tích từ biểu đồ so sánh cho thấy đường dự báo của mô hình CNN-LSTM bám sát hoàn toàn với đường giá trị thực tế trong suốt chuỗi thời gian quan sát. Mô hình không chỉ nắm bắt được xu hướng tổng thể mà còn có khả năng dự đoán chính xác các biến động ngắn hạn, đồng thời không xuất hiện hiện tượng trễ pha trong quá trình dự báo. Sự ổn định của sai số trong toàn bộ chu kỳ dự báo, không có các điểm bất thường lớn, chứng tỏ tính robustness của mô hình qua các giai đoạn khác nhau.

Hiệu suất vượt trội này có thể được giải thích bởi ưu điểm của kiến trúc lai CNN-LSTM. Lớp CNN đóng vai trò quan trọng trong việc trích xuất các đặc trưng cục bộ và mẫu lặp từ dữ liệu chuỗi thời gian, trong khi lớp LSTM xử lý hiệu quả các phụ thuộc thời gian dài hạn và thông tin ngữ cảnh. Sự kết hợp này tạo ra một mô hình mạnh mẽ, phù hợp với đặc tính phức tạp của dữ liệu tiêu thụ điện năng có tính chu kỳ và xu hướng biến đổi theo thời gian. Ngoài ra, CNN giúp giảm chiều dữ liệu đầu vào, làm tăng tốc quá trình huấn luyện và giảm hiện tượng quá khớp. LSTM bổ sung khả năng ghi nhớ dài hạn, giúp mô hình nhận diện được các mối liên kết xuyên suốt giữa các thời điểm cách xa nhau trong chuỗi dữ liệu.

Hình 4.2 trình bày biểu đồ hồi quy giữa giá trị tiêu hao điện năng dự đoán và giá trị thực tế khi áp dụng mô hình CNN-LSTM. Trục hoành thể hiện giá trị dự đoán (kWh/Tấn), trong khi trục tung biểu diễn giá trị thực tế tương ứng. Mỗi dấu chấm xanh trong biểu đồ đại diện cho một cặp dữ liệu (dự đoán, thực tế), qua đó phản ánh mức độ sai lệch giữa kết quả mô hình và số liệu thực tế. Biểu đồ đường lý tưởng và đường hồi quy tuyến tính thể hiện xu hướng phân bố dữ liệu thực tế.



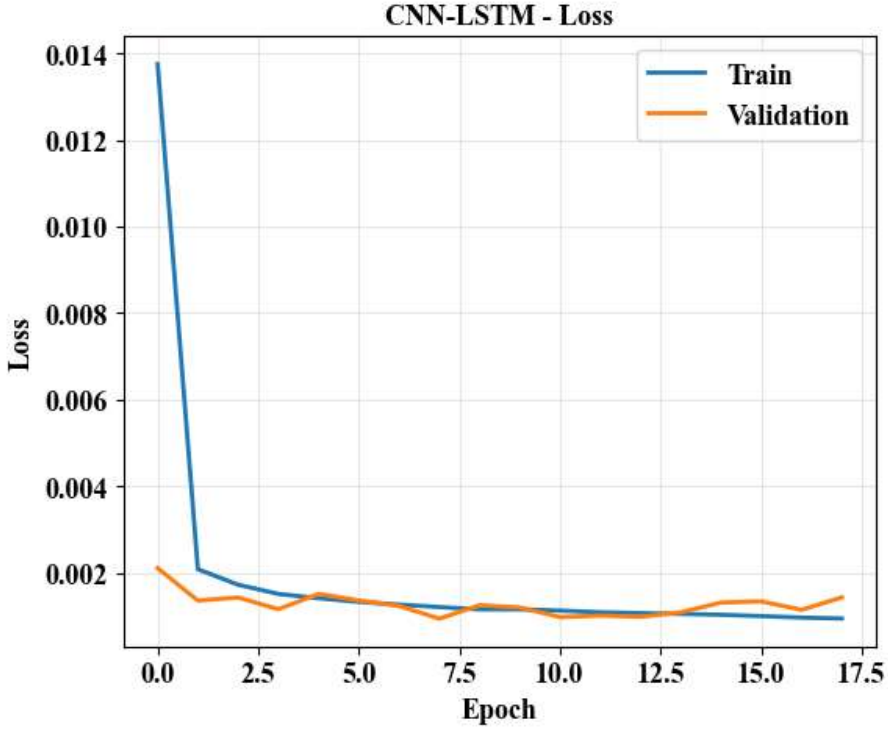
Hình 4.2 Biểu đồ hồi quy giữa dữ liệu thực tế và dự đoán

Đường hồi quy tuyến tính có phương trình y = 1.045x + (-1.954) với hệ số góc 1.045 gần bằng 1, điều này cho thấy mối tương quan gần như hoàn hảo giữa giá trị dự báo và thực tế. Hệ số góc lớn hơn 1 một chút (1.045) có thể cho thấy mô hình có xu hướng dự báo cao hơn giá trị thực tế một cách nhẹ nhàng và không ảnh hưởng đáng kể đến chất lượng dự báo tổng thể.

Phân tích trực quan từ biểu đồ cho thấy các điểm dữ liệu phân bố chặt chẽ xung quanh đường hồi quy màu đỏ và rất gần với đường lý tưởng y = x (đường đứt nét). Sự tập trung cao của các điểm xanh dương quanh đường hồi quy thể hiện tính nhất quán và ổn định của mô hình CNN-LSTM trong toàn bộ phạm vi dự báo. Mặc dù có một số điểm dữ liệu phân tán nhẹ, nhưng không quan sát thấy các điểm outlier rõ rệt hay các vùng có sai số lớn bất thường, chứng tỏ mô hình hoạt động ổn định trên toàn bộ miền giá trị.

Đặc biệt, sự phân bố đồng đều của các điểm dữ liệu từ vùng giá trị thấp (khoảng 37-40 kWh/T) đến vùng giá trị cao (khoảng 60-62 kWh/T) cho thấy mô hình có khả năng dự báo tốt ở tất cả các mức tiêu thụ điện khác nhau. Điều này rất quan trọng trong

ứng dụng thực tế khi hệ thống cần dự báo chính xác cho cả thời điểm tiêu thụ thấp và cao, đảm bảo tính tin cậy của mô hình trong các điều kiện vận hành đa dạng. Hình 4.3 dưới đây thể hiện quá trình thay đổi hàm mất mát (loss) của mô hình CNN-LSTM theo số lần lặp (epoch) trong quá trình huấn luyện và kiểm tra (validation).



Hình 4.3 Biểu đồ mất mát MSE

Biểu đồ mất mát trong quá trình huấn luyện mô hình CNN-LSTM cho thấy quá trình học diễn ra hiệu quả và hội tụ nhanh. Cụ thể, đường cong loss huấn luyện (màu xanh) bắt đầu từ khoảng 0.014 tại epoch đầu tiên và giảm mạnh trong 2–3 epoch đầu, sau đó tiếp tục giảm dần và ổn định quanh mức 0.001 từ epoch thứ 5 trở đi. Điều này cho thấy mô hình nhanh chóng học được đặc trưng của dữ liệu và đạt được trạng thái hội tụ chỉ sau vài vòng lặp đầu.

Tương tự, đường cong loss trên tập validation (màu cam) thể hiện xu hướng song hành với tập huấn luyện, khởi đầu ở mức khoảng 0.002 và ổn định ở mức ~0.0015 từ epoch thứ 3. Khoảng cách giữa hai đường loss là rất nhỏ và ổn định, chứng tỏ mô hình không có dấu hiệu overfitting và có khả năng tổng quát hóa tốt đối với dữ liệu chưa thấy.

Sự ổn định của cả hai đường cong loss từ epoch 5 đến epoch 17 và việc duy trì mức mất mát rất thấp (dưới 0.002) cho thấy mô hình đã được huấn luyện tối ưu. Không có hiện tượng dao động mạnh hay tăng loss trở lại, điều này phản ánh quá trình tối ưu diễn ra suôn sẻ, và mô hình đã đạt đến hiệu suất ổn định. Tóm lại, kết quả từ biểu đồ

loss khẳng định mô hình CNN-LSTM đã được huấn luyện đúng cách, đạt độ chính xác cao, và sẵn sàng áp dụng cho dự báo thực tế mà không cần huấn luyện thêm.

Sau khi hoàn thiện việc xây dựng và huấn luyện mô hình dự đoán kết hợp CNN-LSTM cho hệ thống nghiền xi măng, chúng tôi tiếp tục triển khai giai đoạn tối ưu hóa các tham số vận hành nhằm giảm thiểu điện năng tiêu thụ của toàn hệ thống. Bước này có ý nghĩa quyết định đối với việc nâng cao hiệu quả sử dụng năng lượng, giảm chi phí vận hành, đồng thời góp phần hướng tới mục tiêu sản xuất xanh và bền vững trong ngành công nghiệp xi măng – một ngành vốn tiêu tốn nhiều năng lượng. Việc tối ưu hóa không chỉ mang lại lợi ích kinh tế rõ rệt mà còn góp phần vào cam kết giảm phát thải khí nhà kính và bảo vệ môi trường.

Để đạt được mục tiêu tối ưu hóa, chúng tôi lựa chọn áp dụng thuật toán LHS một phương pháp lấy mẫu hiện đại, có khả năng bao phủ đều không gian tham số, giúp đảm bảo tính toàn diện trong quá trình khám phá các tổ hợp giá trị đầu vào. Khác với các phương pháp lấy mẫu ngẫu nhiên thông thường, LHS giúp tạo ra các mẫu phân bố đều trên toàn bộ miền giá trị, từ đó nâng cao xác suất tìm được điểm tối ưu với số lượng mẫu thử thấp hơn. Trong nghiên cứu này, hai biến vận hành quan trọng nhất được xác định là tốc độ cấp liệu và tốc độ cấp nước, vốn có ảnh hưởng trực tiếp đến hiệu suất nghiền và mức tiêu thụ năng lượng của thiết bị.

Thuật toán LHS được triển khai để tạo ra 100 mẫu thử nghiệm, mỗi mẫu tương ứng với một tổ hợp giá trị cụ thể của hai biến trên. Các giá trị này được chọn sao cho bao phủ toàn bộ khoảng giá trị khả thi, từ đó cung cấp một bức tranh đầy đủ về các trạng thái vận hành có thể của hệ thống nghiền. Nhờ phương pháp này, quá trình tìm kiếm điểm vận hành tối ưu trở nên hiệu quả hơn, tiết kiệm thời gian và công sức so với các phương pháp quét lưới toàn diện truyền thống, trong khi vẫn đảm bảo độ chính xác cao.

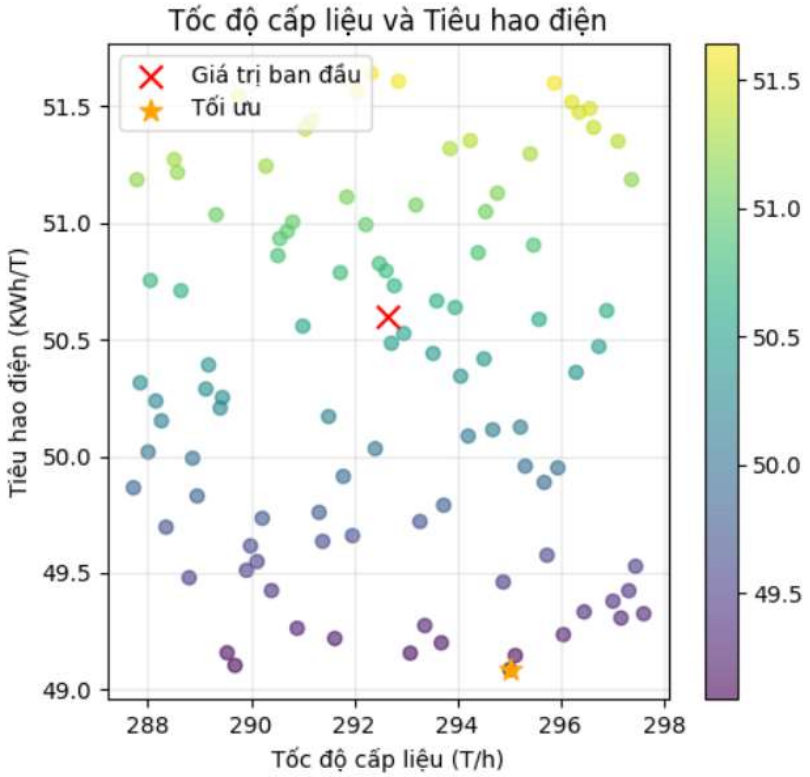
Đặc biệt, nhóm thực hiện đã tiến hành thử nghiệm tối ưu hóa tại một thời điểm vận hành cụ thể – 19 giờ 09 phút 01 giây, ngày 25 tháng 3 năm 2025. Tại thời điểm này, hệ thống đang vận hành ở trạng thái bình thường, cung cấp dữ liệu thực tế đầu vào cho mô hình dự đoán và tối ưu hóa. Mục tiêu là tìm ra tổ hợp tốc độ cấp liệu và tốc độ cấp nước sao cho điện năng tiêu thụ trong công đoạn nghiền là thấp nhất, nhưng vẫn đảm bảo sản lượng và chất lượng đầu ra. Đây là một bước đi chiến lược, thể hiện nỗ lực chuyển từ mô hình dự đoán sang ứng dụng thực tiễn, giúp các nhà máy có thể chủ động điều chỉnh thông số theo thời gian thực để đạt hiệu suất năng lượng tối ưu.

Hình 4.4 biểu diễn mối quan hệ giữa tốc độ cấp liệu (T/h) và mức tiêu thụ điện năng (kWh/T), trong đó các điểm màu thể hiện kết quả từ các tổ hợp thông số được mô

phỏng bởi mô hình dự đoán. Màu sắc của điểm biểu thị mức tiêu thụ điện, với màu tím tương ứng với mức tiêu thụ thấp, và màu vàng tương ứng với mức tiêu thụ cao.

- Dấu "X" màu đỏ biểu thị giá trị vận hành ban đầu: tốc độ cấp liệu là 292.633 T/h, ứng với mức tiêu thụ điện là 50.600 kWh/T.
- Dấu sao màu vàng biểu diễn giá trị sau khi tối ưu: tốc độ cấp liệu được điều chỉnh lên 295.019 T/h, qua đó mức tiêu thụ điện giảm xuống còn 49.087 kWh/T.

Từ đồ thị có thể thấy rõ điểm tối ưu nằm trong vùng tiêu thụ điện thấp nhất, cho thấy hiệu quả rõ rệt của thuật toán tối ưu trong việc tìm ra điều kiện vận hành hợp lý.



Hình 4.4 Mối quan hệ giữa tốc độ cấp liệu và tiêu thụ điện năng

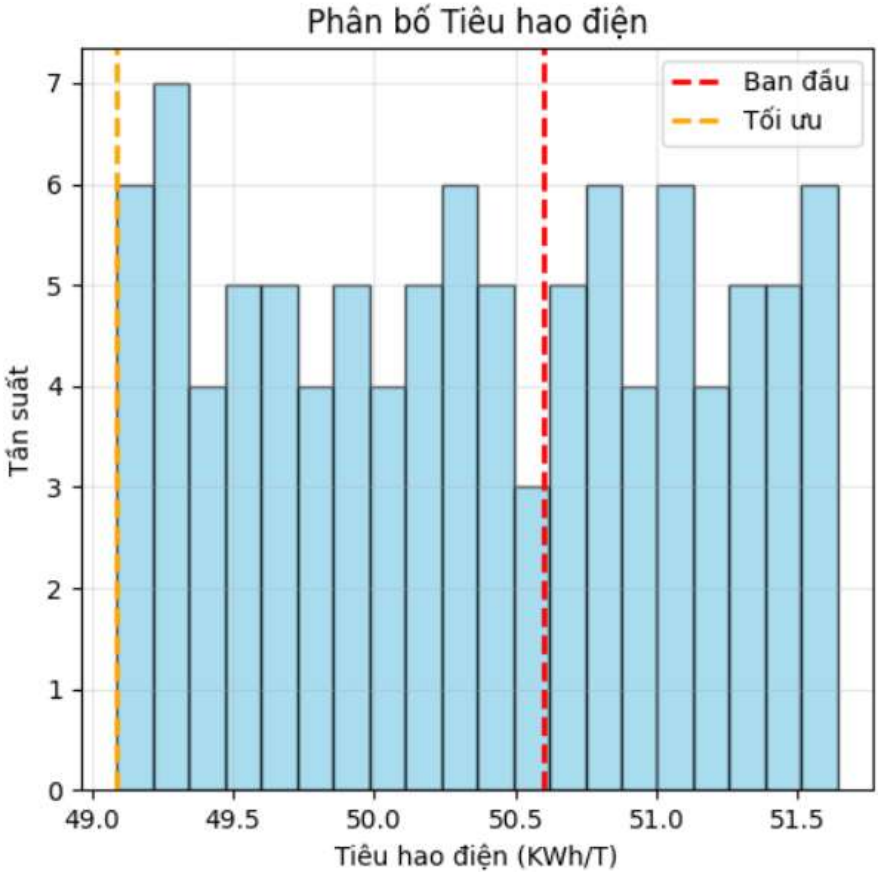
Hình 4.5 thể hiện phân bố tần suất mức tiêu thụ điện trên toàn bộ tập dữ liệu đầu vào. Hai đường kẻ dọc được sử dụng để so sánh:

- Đường đứt nét đỏ tương ứng với mức tiêu thụ điện ban đầu (50.600 kWh/T).
- Đường đứt nét cam biểu thị mức tiêu thụ điện sau tối ưu (49.087 kWh/T).

Sau khi tiến hành đánh giá sơ bộ hiệu quả của phương pháp tối ưu hóa trên một điểm dữ liệu cụ thể, nhóm nghiên cứu tiếp tục mở rộng phạm vi thí nghiệm bằng cách áp dụng mô hình tối ưu cho một tập hợp gồm 10 mẫu dữ liệu liên tiếp, thu thập tại thời điểm vận hành từ 19:09:01 đến 19:18:00 ngày 25 tháng 3 năm 2025. Mỗi mẫu tương

ứng với một thời điểm ghi nhận trong quá trình vận hành thực tế của công đoạn nghiền xi măng.

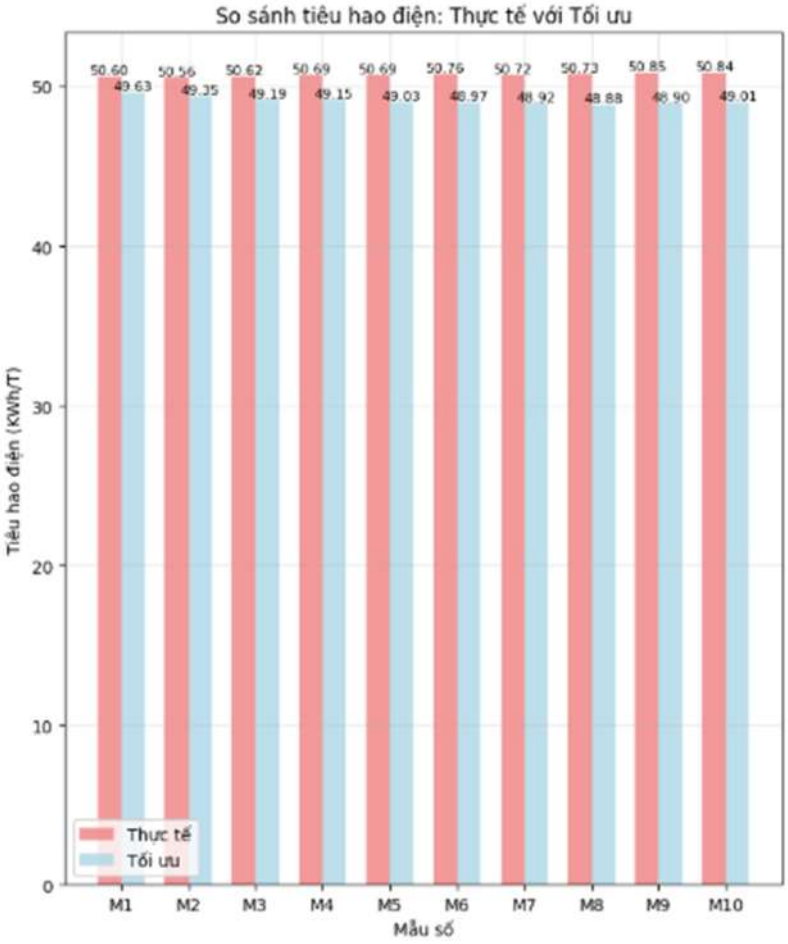
Trong từng mẫu, hai thông số vận hành chính – tốc độ cấp liệu (T/h) và tốc độ cấp nước (m^3/h) – được sử dụng làm biến đầu vào để tối ưu hóa, với mục tiêu là tìm ra tổ hợp giá trị giúp giảm thiểu mức tiêu thụ điện năng (kWh/tấn clinker) của thiết bị nghiền. Mô hình tối ưu sử dụng thuật toán học sâu kết hợp CNN–LSTM để dự đoán đầu ra, sau đó áp dụng thuật toán tìm kiếm tham số để xác định giá trị đầu vào tối ưu.



Hình 4.5 Phân bố tiêu hao điện sau tối ưu hóa

Hình 4.6 thể hiện so sánh mức tiêu thụ điện năng giữa giá trị thực tế và giá trị sau khi tối ưu cho từng mẫu dữ liệu từ M1 đến M10. Các cột màu đỏ thể hiện giá trị thực tế ghi nhận được tại nhà máy, trong khi cột màu xanh dương biểu thị kết quả dự đoán mức tiêu thụ điện tương ứng sau khi áp dụng bộ tham số tối ưu được mô hình đề xuất.

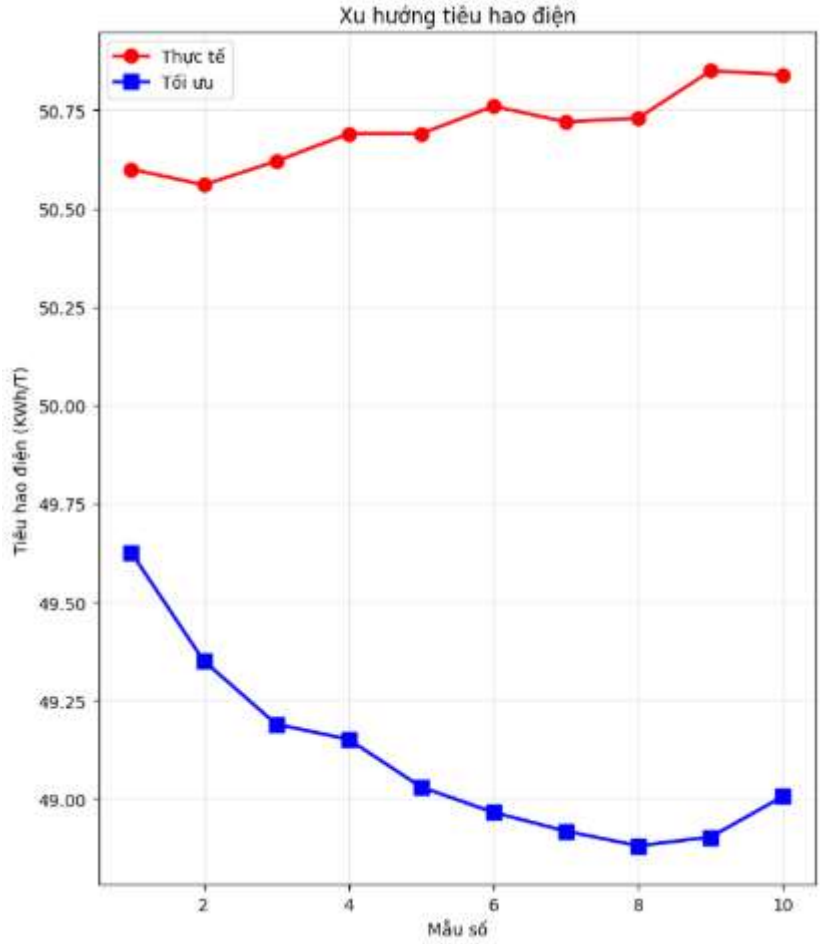
- Mức tiêu hao điện năng thực tế dao động trong khoảng 50.56 đến 50.84 kWh/tấn.
- Sau tối ưu, các giá trị giảm xuống còn khoảng 48.88 đến 49.63 kWh/tấn, tương ứng với mức tiết kiệm trung bình từ 0,975 đến 1.947 kWh/tấn clinker.



Hình 4.6 So sánh tiêu thụ điện giữa giá trị thực tế và giá trị tối ưu

Kết quả cho thấy rõ ràng mô hình tối ưu hóa không chỉ hiệu quả ở một vài trường hợp đơn lẻ, mà còn mang lại kết quả nhất quán trên toàn bộ chuỗi mẫu thử nghiệm. Điều này khẳng định khả năng tổng quát hóa và tính thực tiễn cao của phương pháp được đề xuất.

Hình 4.7 minh họa xu hướng tiêu hao điện năng (kWh/tấn) của hệ thống nghiền xi măng theo 10 mẫu dữ liệu khác nhau. Mỗi mẫu tương ứng với một tổ hợp tham số vận hành cụ thể, bao gồm tốc độ cấp liệu và tốc độ cấp nước. Đường màu đỏ biểu diễn mức tiêu thụ điện thực tế trong điều kiện vận hành hiện tại của nhà máy, trong khi đường màu xanh thể hiện mức tiêu thụ điện sau khi áp dụng thuật toán tối ưu hóa bằng Latin Hypercube Sampling (LHS) kết hợp với mô hình dự đoán CNN-LSTM. Dữ liệu được lấy tại thời điểm cụ thể (ngày 25/3/2025 lúc 19:09:01), nhằm đánh giá hiệu quả tiết kiệm năng lượng giữa hai phương án vận hành.

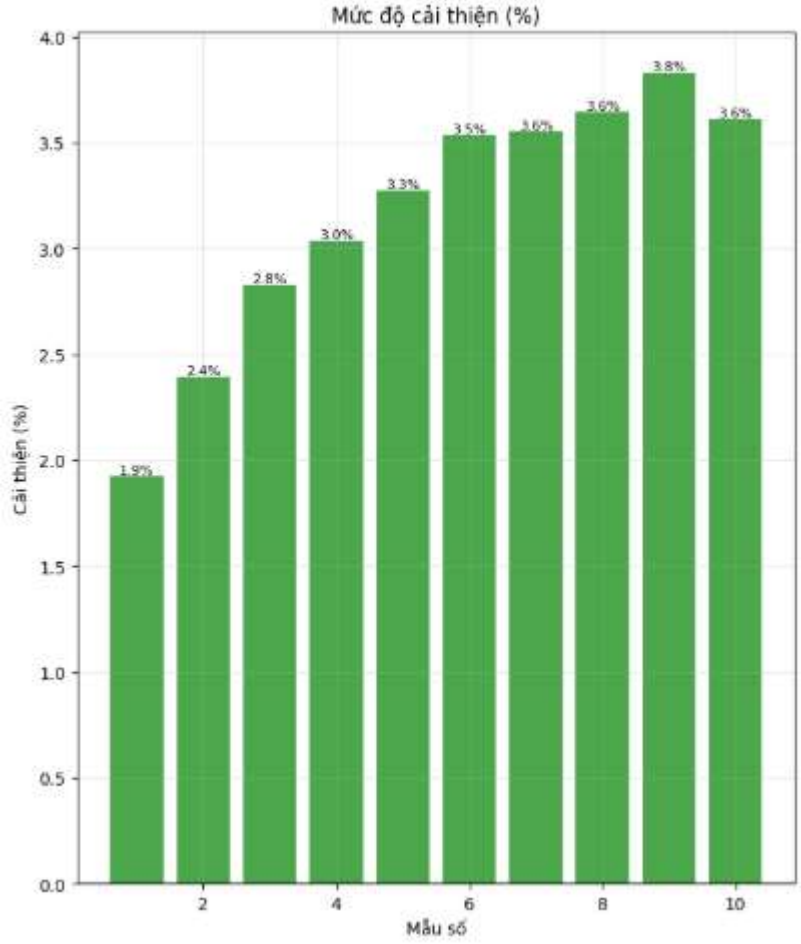


Hình 4.7 Phân tích xu hướng tiêu hao điện năng với 10 mẫu dữ liệu

Hình bên phải minh họa xu hướng biến thiên của mức tiêu thụ điện năng theo thời gian mẫu (M1 đến M10). Đường màu đỏ thể hiện xu hướng tiêu thụ điện thực tế, trong khi đường màu xanh thể hiện giá trị tiêu thụ điện được dự đoán sau khi tối ưu.

Từ biểu đồ, có thể thấy mức tiêu thụ điện sau tối ưu hóa luôn thấp hơn đáng kể so với mức tiêu thụ thực tế. Đặc biệt, từ mẫu số 1 đến mẫu số 8, đường tiêu hao năng lượng sau tối ưu có xu hướng giảm dần và đạt giá trị thấp nhất khoảng 48.85 kWh/tấn tại mẫu số 8, so với mức tiêu thụ thực tế dao động khoảng 50.5–50.8 kWh/tấn. Sự chênh lệch gần 2 kWh/tấn cho thấy tiềm năng rất lớn trong việc giảm chi phí vận hành nếu áp dụng các tổ hợp vận hành tối ưu được đề xuất. Điều này khẳng định tính hiệu quả và khả năng ứng dụng thực tiễn của mô hình CNN-LSTM kết hợp với LHS trong bài toán tối ưu hóa tiêu thụ năng lượng trong công nghiệp xi măng.

Hình 4.8 dưới đây thể hiện mức độ cải thiện tương đối (tính theo phần trăm) giữa giá trị tiêu thụ điện năng thực tế và giá trị dự đoán sau khi áp dụng tối ưu hóa cho từng mẫu dữ liệu cụ thể. Trong đó, các mẫu M1 đến M10 tương ứng với mười thời điểm liên tiếp trong giai đoạn từ 19:09:01 đến 19:18:00 ngày 25 tháng 3 năm 2025.



Hình 4.8 Mức độ cải thiện tiêu hao điện năng

Trên trục tung của biểu đồ là tỷ lệ phần trăm cải thiện, được tính toán dựa trên mức chênh lệch giữa điện năng tiêu thụ ban đầu và điện năng tiêu thụ sau tối ưu. Trục hoành biểu diễn số thứ tự của các mẫu dữ liệu. Kết quả từ biểu đồ cho thấy tất cả các mẫu đều đạt được mức cải thiện dương, nghĩa là phương pháp tối ưu hóa đã mang lại hiệu quả tiết kiệm năng lượng trên toàn bộ chuỗi dữ liệu.

Cụ thể, mẫu M1 đạt mức cải thiện thấp nhất là 1.9%, trong khi các mẫu từ M4 trở đi ghi nhận mức cải thiện lớn hơn 3%. Mức cải thiện cao nhất đạt 3.8% tại mẫu M9, tiếp đến là 3.6% ở mẫu M10. Trung bình, phương pháp tối ưu giúp tiết kiệm khoảng 3.2% điện năng so với thực tế, tương đương khoảng 1.5 KWh cho mỗi tấn clinker nghiền. Đây là một con số có ý nghĩa lớn trong bối cảnh chi phí điện năng chiếm tỷ trọng cao trong tổng chi phí sản xuất xi măng.

Kết quả được đưa ra ở Bảng 4.3 không chỉ khẳng định tính hiệu quả của mô hình tối ưu hóa được đề xuất, mà còn cho thấy khả năng ứng dụng thực tiễn trong điều kiện vận hành thực tế của nhà máy. Việc cải thiện đều đặn trên tất cả các mẫu chứng minh rằng mô hình có độ ổn định tốt, khả năng thích ứng cao và có thể được tích hợp vào hệ

thống điều khiển hiện tại nhằm hỗ trợ ra quyết định hoặc thực hiện điều khiển tự động theo thời gian thực. Đây là một bước tiến quan trọng hướng tới mục tiêu nâng cao hiệu quả năng lượng và phát triển bền vững trong ngành công nghiệp xi măng.

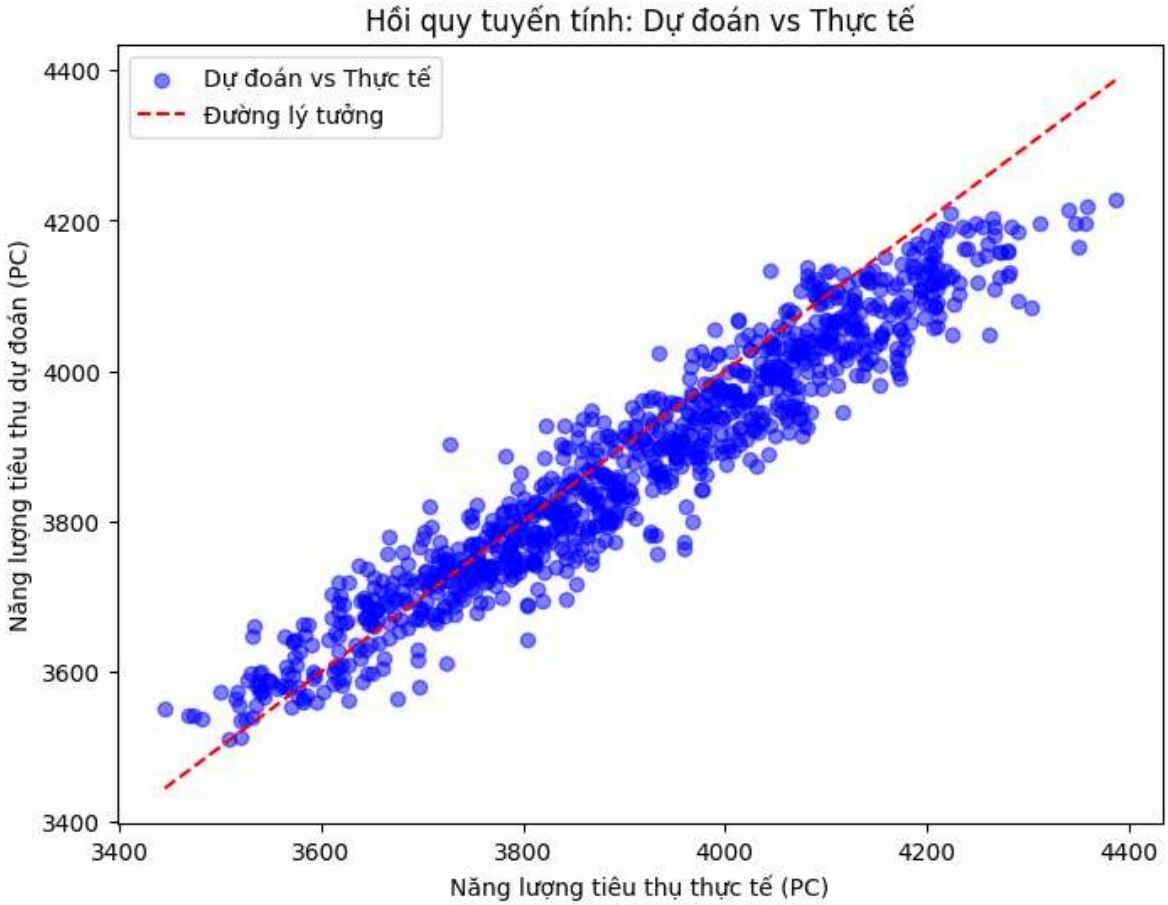
Bảng 4.3 Số liệu kết quả tối ưu hóa cho 10 mẫu dự đoán

Thời gian	**Sản lượng (T/h)**	**Cấp liệu trước (T/h)**	**Cấp liệu sau (T/h)**	**Cấp nước trước (m^3/h)**	**Cấp nước sau (m^3/h)**	**Tiêu hao điện thực tế (KWh/T)**	**Dự đoán tối ưu (KWh/T)**
2025/03/25 19:09:01	284.11	292.633	289.69	1.082	0.883	50.6	49.625
2025/03/25 19:10:00	284.07	292.592	289.648	1.081	0.882	50.56	49.35
2025/03/25 19:11:01	284.13	298.337	295.393	1.078	0.879	50.62	49.189
2025/03/25 19:12:00	284.2	298.41	295.466	1.076	0.877	50.69	49.151
2025/03/25 19:13:01	284.19	295.558	292.614	1.073	0.874	50.69	49.029
2025/03/25 19:14:00	284.25	295.62	292.676	1.071	0.872	50.76	48.965
2025/03/25 19:15:01	284.2	298.41	295.466	1.067	0.868	50.72	48.917
2025/03/25 19:16:00	284.21	292.736	289.793	1.064	0.865	50.73	48.88
2025/03/25 19:17:01	284.32	298.536	295.592	1.062	0.863	50.85	48.903
2025/03/25 19:18:00	284.31	298.526	295.582	1.064	0.865	50.84	49.006

Sau khi hoàn thiện và đánh giá mô hình tại một nhà máy xi măng ở Việt Nam, nhóm nghiên cứu tiếp tục kiểm tra khả năng tổng quát hóa của mô hình bằng cách áp

dụng nó cho dữ liệu từ một nhà máy xi măng tại Trung Quốc. Bộ dữ liệu được sử dụng trong thí nghiệm này được trích xuất từ bài báo tham khảo của Zhang và các cộng sự (2024) [31], với các đặc điểm vận hành và điều kiện công nghệ khác biệt so với môi trường huấn luyện ban đầu.

Việc thử nghiệm mô hình trên tập dữ liệu độc lập từ một nhà máy ở quốc gia khác không chỉ nhằm xác minh độ chính xác của mô hình trong điều kiện thực tế khác nhau, mà còn để đánh giá tính ứng dụng rộng rãi và mức độ tin cậy của mô hình trong việc dự đoán tiêu thụ điện năng trong công đoạn nghiền clinker tại các cơ sở sản xuất xi măng khác nhau. Kết quả mô phỏng và so sánh được trình bày cụ thể trong Hình 4.9.



Hình 4.9 Biểu đồ hồi quy của nhà máy xi măng ở Trung Quốc

Kết quả các chỉ số đánh giá:

RMSE: 67.7596

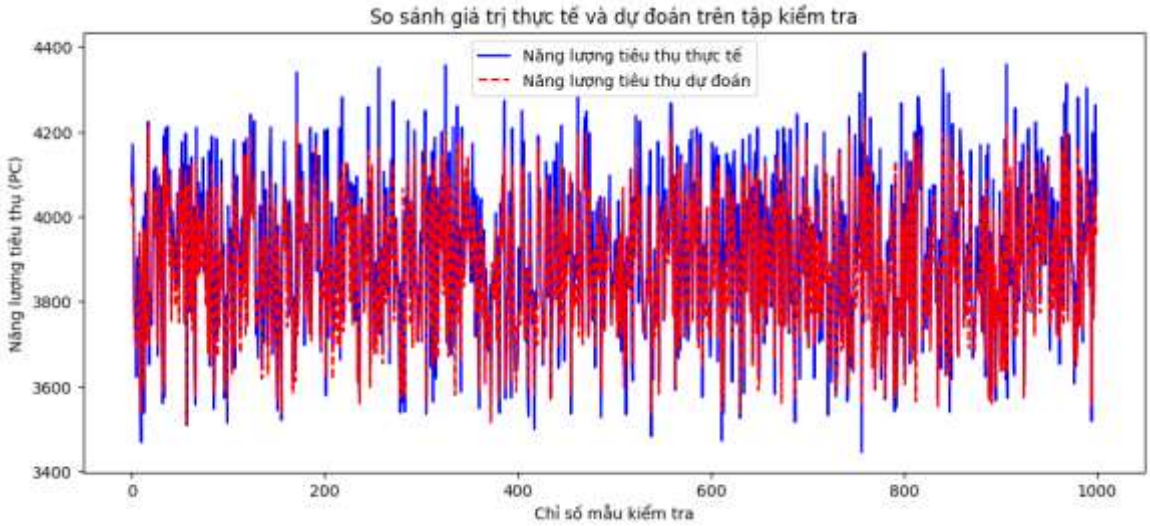
MAE: 54.0664

MAPE: 1.37%

R^2: 0.8777

Hình 4.9 thể hiện kết quả hồi quy tuyến tính giữa giá trị năng lượng tiêu thụ thực tế và giá trị dự đoán bởi mô hình trên bộ dữ liệu thu thập từ nhà máy tại Trung Quốc. Các điểm màu xanh đại diện cho mối quan hệ giữa năng lượng tiêu thụ dự đoán và năng lượng tiêu thụ thực tế, trong khi đường chấm đỏ là đường lý tưởng. Quan sát hình ảnh cho thấy phần lớn các điểm dữ liệu nằm gần đường lý tưởng, chứng tỏ mô hình có khả năng dự đoán chính xác mức tiêu thụ điện năng. Độ phân tán thấp quanh đường lý tưởng cho thấy sai số dự đoán nhỏ và mô hình có độ tin cậy cao. Điều này chứng minh rằng mô hình được xây dựng không chỉ hoạt động hiệu quả với dữ liệu tại nhà máy ở Việt Nam mà còn có khả năng tổng quát hóa và áp dụng thành công cho dữ liệu tại một nhà máy khác.

Hình 4.10 minh họa sự so sánh giữa giá trị năng lượng tiêu thụ thực tế (đường màu xanh) và giá trị năng lượng tiêu thụ được mô hình dự đoán (đường nét đứt màu đỏ) trên 1000 mẫu thuộc tập dữ liệu kiểm tra tại nhà máy ở Trung Quốc. Có thể thấy rằng hai đường biểu diễn bám sát nhau trong hầu hết các khoảng thời gian, cho thấy mô hình có khả năng dự đoán tương đối chính xác theo xu hướng thực tế.



Hình 4.10 Biểu đồ sánh giá trị thực tế và dự đoán với 1000 mẫu dữ liệu tại nhà máy Trung Quốc

Đặc biệt, các biến động mạnh của dữ liệu thực tế cũng được mô hình phản ánh tương đối tốt, chứng minh rằng mô hình không chỉ dự đoán giá trị trung bình mà còn theo sát được sự dao động của quá trình vận hành thực tế. Việc mô hình giữ được độ khớp cao với dữ liệu thực nghiệm trong một chuỗi thời gian dài và phức tạp càng khẳng định tính ổn định và hiệu quả dự báo của nó.

Từ kết quả này, có thể kết luận rằng mô hình được đề xuất không những hoạt động tốt trên dữ liệu huấn luyện mà còn duy trì hiệu suất cao khi áp dụng cho dữ liệu kiểm tra chưa từng thấy trước đó. Điều này làm nổi bật tính khả thi và ứng dụng thực tiễn của mô hình trong việc theo dõi, dự đoán và tối ưu hóa tiêu thụ điện năng cho các hệ thống nghiền xi măng ở quy mô công nghiệp.

KẾT LUẬN

❖ Kết luận chung

Trong khuôn khổ nghiên cứu này, chúng tôi đã xây dựng và triển khai thành công mô hình dự đoán kết hợp CNN-LSTM nhằm ước lượng mức tiêu thụ năng lượng tại công đoạn nghiền trong nhà máy xi măng. Mô hình được huấn luyện từ dữ liệu thực nghiệm thu thập từ hệ thống vận hành, với đầu vào là các thông số vận hành như tốc độ cấp liệu, tải máy nghiền, độ rung, nhiệt độ và áp suất… Sau quá trình huấn luyện, các giá trị tiêu thụ điện được mô phỏng từ mô hình đã được so sánh trực tiếp với số liệu thực tế, cho thấy độ sai lệch thấp và khả năng dự đoán cao.

Kết quả đánh giá mô hình bằng các chỉ số thống kê RMSE, MAE, MAPE, và hệ số tương quan R^2 đều đạt mức cao – với R^2 dao động từ 0,95 đến 0,98. Điều này phản ánh khả năng mô phỏng chính xác và đáng tin cậy của mô hình trong điều kiện dữ liệu thực tế. Mô hình không chỉ có khả năng mô tả tốt xu hướng tiêu thụ điện năng mà còn có tiềm năng trở thành một công cụ hỗ trợ quyết định cho đội ngũ kỹ thuật trong việc giám sát, điều chỉnh thông số vận hành theo thời gian thực một cách chủ động và hiệu quả.

Dựa trên kết quả của mô hình dự báo, nghiên cứu tiếp tục đề xuất và áp dụng phương pháp tối ưu hóa sử dụng thuật toán Latin Hypercube Sampling nhằm xác định tổ hợp các tham số đầu vào phù hợp nhất để giảm thiểu tiêu thụ năng lượng. LHS cho phép lấy mẫu đồng đều trên không gian tìm kiếm, từ đó phát hiện ra các điểm vận hành hiệu quả nhất. Các kết quả thực nghiệm cho thấy, phương pháp tối ưu hóa có thể giúp tiết kiệm trung bình từ 0,975 đến 1,947 kWh/tấn, tương đương giảm từ 2% đến 6% lượng điện tiêu thụ so với mức trung bình hiện tại. Đây là một con số đáng kể trong bối cảnh chi phí điện năng chiếm tỷ trọng lớn trong giá thành sản xuất xi măng.

Không những vậy, mô hình CNN-LSTM được áp dụng thử nghiệm tại một nhà máy xi măng tại Trung Quốc và đạt được hệ số tương quan R^2 dao động trong khoảng 0,87 đến 0,89. Kết quả này một lần nữa khẳng định tính khả chuyển và khả năng ứng dụng rộng rãi của mô hình đề xuất cho nhiều cơ sở sản xuất khác nhau với điều kiện vận hành tương tự.

Tuy nhiên, nghiên cứu cũng nhận diện được một số hạn chế. Đầu tiên, mô hình yêu cầu lượng dữ liệu đầu vào đa dạng và lớn để có thể huấn luyện và dự đoán với độ

chính xác cao. Trong khuôn khổ nghiên cứu này, dữ liệu thu thập vẫn còn tương đối hạn chế cả về độ phân giải (quy đổi từ phút sang giờ) lẫn mức độ chênh lệch giữa các mẫu. Do đó, độ phân giải thời gian thấp có thể ảnh hưởng đến khả năng mô hình hóa các biến động nhỏ trong hệ thống. Thứ hai, mô hình hiện mới chỉ dừng lại ở mức mô phỏng và thử nghiệm, chưa được tích hợp và triển khai trong hệ thống điều khiển vận hành thực tế (DCS/SCADA) tại nhà máy.

Tóm lại, nghiên cứu này đã chứng minh được hiệu quả của mô hình CNN-LSTM kết hợp tối ưu hóa LHS trong việc dự đoán và giảm thiểu tiêu thụ năng lượng cho công đoạn nghiền trong nhà máy xi măng. Mô hình đề xuất có tính khả thi, tiềm năng mở rộng và là một bước tiến quan trọng hướng tới mục tiêu sản xuất thông minh – tiết kiệm năng lượng – giảm phát thải – phát triển bền vững trong ngành công nghiệp xi măng hiện đại.

❖ Hướng phát triển

Mặc dù mô hình CNN-LSTM kết hợp thuật toán LHS trong nghiên cứu này đã chứng minh hiệu quả trong việc dự đoán và tối ưu hóa mức tiêu thụ năng lượng cho công đoạn nghiền xi măng, vẫn còn nhiều tiềm năng và không gian để tiếp tục phát triển, mở rộng và hoàn thiện mô hình nhằm áp dụng trong thực tiễn công nghiệp với hiệu suất cao hơn.

Trước hết, một hướng nghiên cứu quan trọng là tăng cường khả năng phản hồi thời gian thực và tích hợp hệ thống điều khiển đóng vòng. Hiện tại, mô hình của chúng tôi chủ yếu hoạt động theo cơ chế dự đoán và tối ưu hóa ngoại tuyến. Việc xây dựng một hệ thống AI Recommendation Module tương tự như trong mô hình điều khiển lò nung của Siemens do Balakrishnan và các cộng sự phát triển (2023) [32] với khả năng đưa ra khuyến nghị điều chỉnh trực tiếp hoặc bán tự động cho hệ thống vận hành thông qua giao diện kết nối (như OPC UA/DA) – sẽ giúp mô hình trở thành công cụ điều khiển thực sự. Đây là bước chuyển từ "hỗ trợ ra quyết định" sang "tự động điều chỉnh", giúp tăng tính linh hoạt và hiệu quả cho dây chuyền sản xuất.

Thứ hai, việc tích hợp mô hình học sâu với tri thức chuyên gia (hybrid AI) cũng là một hướng đi tiềm năng. Các mô hình học máy hiện tại chủ yếu dựa trên dữ liệu, trong khi các hệ thống điều khiển hiện hành tại nhà máy thường dựa vào kinh nghiệm của kỹ sư vận hành. Việc phát triển mô hình lai (neuro-fuzzy, rule-based + machine learning) cho phép mô hình vừa tận dụng được khả năng học từ dữ liệu, vừa khai thác hiệu quả tri thức chuyên môn đã tích lũy, đặc biệt hữu ích khi dữ liệu bị giới hạn. Đồng thời, mô hình lai còn giúp cải thiện khả năng diễn giải và minh bạch hóa quyết định của hệ thống

AI, từ đó tăng độ tin cậy và khả năng chấp nhận của người vận hành trong môi trường công nghiệp thực tế.

Thứ ba, để nâng cao độ chính xác và tính tổng quát của mô hình, cần mở rộng cơ sở dữ liệu thực tế về vận hành ở các khung thời gian nhỏ hơn (dữ liệu theo phút hoặc giây, thay vì giờ), cũng như từ nhiều nhà máy khác nhau để đảm bảo khả năng khái quát và tính phổ dụng của mô hình. Đồng thời, cần đánh giá khả năng áp dụng transfer learning (học chuyển tiếp) để rút ngắn thời gian huấn luyện khi triển khai ở môi trường mới.

Ngoài ra, hướng phát triển trong tương lai nên tập trung vào việc xây dựng nền tảng giám sát thông minh tích hợp toàn nhà máy xi măng, kết hợp dữ liệu từ nhiều nguồn (SCADA, cảm biến IoT, dữ liệu lịch sử vận hành) để đưa ra quyết định điều hành tổng thể. Điều này sẽ không chỉ giúp tối ưu hóa từng công đoạn riêng lẻ, mà còn cải thiện hiệu quả tổng thể và đáp ứng mục tiêu phát triển bền vững như giảm phát thải CO_2, tiết kiệm tài nguyên, và kéo dài tuổi thọ thiết bị.

Cuối cùng, từ góc độ vận hành và kinh tế, một hướng nghiên cứu thực tiễn là tích hợp mô hình với chiến lược vận hành theo giá điện thời gian thực (TOU pricing). Bằng cách dự báo tiêu thụ điện theo thời gian và điều chỉnh thông số vận hành theo khung giờ giá thấp, mô hình có thể hỗ trợ doanh nghiệp tối ưu hóa chi phí năng lượng một cách chủ động và linh hoạt hơn.

Tổng thể, các hướng nghiên cứu tiếp theo cần đi theo xu hướng AI hướng tới sản xuất tự động thông minh, vừa đảm bảo hiệu suất năng lượng, vừa tăng cường khả năng kiểm soát, và trên hết là tạo ra hệ thống sản xuất bền vững – thông minh – thích ứng trong bối cảnh chuyển đổi số ngành công nghiệp nặng hiện nay.

TÀI LIỆU THAM KHẢO

[1] Worrell, E., Galitsky, C., & Price, L. (2008). *Energy efficiency improvement and cost saving opportunities for cement making* (LBNL-54036-Revision). Berkeley, CA: Ernest Orlando Lawrence Berkeley National Laboratory, University of California.

[2] Madlool, N. A., Saidur, R., Hossain, M. S., & Rahim, N. A. (2011). A critical review on energy use and savings in the cement industries. *Renewable and Sustainable Energy Reviews*, *15*(4), 2042–2060. https://doi.org/10.1016/j.rser.2011.01.005

[3] Sivanandam, V., Kannan, R., Srinivasan, S., & Muralidharan, G. (2017). Data driven models for cement grinding circuit. *International Journal of Advanced Intelligence Paradigms*, *9*(4), 414–435.

[4] Oguntola, O., Boakye, K., & Simske, S. (2024). Towards leveraging artificial intelligence for sustainable cement manufacturing: A systematic review of AI applications in electrical energy consumption optimization. *Sustainability*, *16*(11), 4798. https://doi.org/10.3390/su16114798

[5] Genç, Ö., & Genç, Ö. (2016). Energy-efficient technologies in cement grinding. In F. Pacheco Torgal, S. Jalali, & N. De Belie (Eds.), *High Performance Concrete Technology and Applications* (pp. 115–139). Rijeka: InTech.

[6] Báo Công Thương, Tạp chí Năng lượng Việt Nam, Báo Lao Động, Báo Tuổi Trẻ. (2019–2023). *Các bài viết liên quan đến tiêu thụ điện và công nghệ nghiền clinker trong ngành xi măng*. Truy cập ngày 09/06/2025, từ https://moit.gov.vn, https://nangluongvietnam.vn, https://laodong.vn, https://tuoitre.vn

[7] Bianco, F., Manca, M., & Serra, M. (2009). Electricity consumption forecasting in Italy using linear regression analysis. *Energy*, *34*(9), 1413–1421. https://doi.org/10.1016/j.energy.2009.06.034

[8] Sen, P., Roy, M., & Pal, P. (2016). Application of ARIMA for forecasting energy consumption and GHG emission: A case study of an Indian pig iron manufacturing organization. *Energy*, *116*, 1031–1038.

[9] Gifalli, A., Amaral, H. L. M. D., Bonini Neto, A., de Souza, A. N., Frühauf Hublard, A. V., Carneiro, J. C., & Neto, F. T. (2024). Forecasting electricity consumption using function fitting artificial neural networks and regression methods. *Applied System Innovation*, *7*(5), 100. https://doi.org/10.3390/asi7050100

[10] Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, *9*(8), 1735-1780.

[11] Sang, S.; Li, L. A Stock Prediction Method Based on Heterogeneous Bidirectional LSTM. *Appl. Sci.* **2024**, *14*, 9158. [CrossRef]

[12] Palaniyappan, B., & Ramu, S. K. (2025). Optimized LSTM-based electric power consumption forecasting for dynamic electricity pricing in demand response scheme of smart grid. *Results in Engineering*, *2025*, Article 104356. https://doi.org/10.1016/j.rineng.2025.104356

[13] Haji, S. H., & Abdulazeez, A. M. (2021). Comparison of optimization techniques based on gradient descent algorithm: A review. *PalArch's Journal of Archaeology of Egypt/Egyptology*, *18*(4), 2715–2743.

[14] Price, K. V., Storn, R. M., & Lampinen, J. A. (2005). The differential evolution algorithm. *Differential evolution: a practical approach to global optimization*, 37-134.

[15] Lakshminarasimman, L., & Subramanian, S. (2008). Applications of differential evolution in power system optimization. In *Advances in Differential Evolution* (pp. 257-273). Berlin, Heidelberg: Springer Berlin Heidelberg.

[16] Snoek, J., Swersky, K., Zemel, R., & Adams, R. (2014, June). Input warping for Bayesian optimization of non-stationary functions. In *International conference on machine learning* (pp. 1674-1682). PMLR.

[17] Sultana, N., Hossain, S. Z., Almuhaini, S. H., & Düştegör, D. (2022). Bayesian optimization algorithm-based statistical and machine learning approaches for forecasting short-term electricity demand. *Energies*, *15*(9), 3425.

[18] Shields, Michael D., and Jiaxin Zhang. "The generalization of Latin hypercube sampling." Reliability Engineering & System Safety 148 (2016): 96-108.

[19] Guo, J., Feng, T., Cai, Z., Yu, Z., Gu, Y., Qian, T., ... & Tang, W. (2020, October). Security risk assessment of power system based on latin hypercube sampling and daily peak load forecasting. In *2020 IEEE 4th Conference on Energy Internet and Energy System Integration (EI2)* (pp. 2787-2792). IEEE.

[20] Alharkan, H., Habib, S., & Islam, M. (2023). Solar power prediction using dual stream CNN-LSTM architecture. *Sensors*, *23*(2), 945.

https://doi.org/10.3390/s23020945

[21] Cheng, W., Wang, Y., Peng, Z., Ren, X., Shuai, Y., Zang, S., ... & Wu, J. (2021). High-efficiency chaotic time series prediction based on time convolution neural network. *Chaos, Solitons & Fractals*, *152*, 111304.

[22] Qian, N. (1999). On the momentum term in gradient descent learning algorithms. *Neural networks*, *12*(1), 145-151.

[23] Hinton, G. (2012). *Neural networks for machine learning* [lecture notes]. University of Toronto. Chapter 6: RMSProp.

[24] Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization. *Proceedings of the International Conference on Learning Representations (ICLR), 2015*. Truy cập ngày 09/06/2025, từ https://arxiv.org/abs/1412.6980

[25] Reddi, S. J., Kale, S., & Kumar, S. (2018). On the convergence of Adam and beyond. *Proceedings of the International Conference on Learning Representations (ICLR), 2018*. Truy cập ngày 09/06/2025, từ https://openreview.net/forum?id=ryQu7f-RZ

[26] Ruder, S. (2016). *An overview of gradient descent optimization algorithms*. Truy cập ngày 09/06/2025, từ https://arxiv.org/abs/1609.04747

[27] V. Nair and G. E. Hinton, *"Rectified linear units improve restricted Boltzmann machines,"* ICML 2010.

[28] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning,* MIT Press, 2016.

[29] S. Haykin, *Neural Networks and Learning Machines*, 3rd ed. Upper Saddle River, NJ: Pearson Education, 2009.

[30] Helton, J. C., & Davis, F. J. (2003). Latin hypercube sampling and the propagation of uncertainty in analyses of complex systems. *Reliability Engineering & System Safety*, *81*(1), 23-69.

[31] Zhang, D., Xiong, X., Shao, C., Zeng, Y., & Ma, J. (2024). Semi-Autogenous Mill Power Consumption Prediction Based on CACN-LSTM. *Applied Sciences*, *15*(1), 2.

[32] Balakrishnan, A., Chandrashekara, R. S., & Sunny, S. (2023, November). *AI Kiln Solution For Optimized Control: How To Reduce Energy Consumption And Emissions In The Clinker Process*.

PHỤ LỤC

1. Chương trình chọn tham số LSTM

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow.keras.models import Sequential, load_model
from tensorflow.keras.layers import InputLayer, LSTM, Dense, Conv1D, Dropout
from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping
from tensorflow.keras.losses import MeanSquaredError
from tensorflow.keras.metrics import RootMeanSquaredError
from tensorflow.keras.optimizers import Adam
from sklearn.metrics import r2_score, mean_squared_error, mean_absolute_error
from sklearn.preprocessing import MinMaxScaler
import itertools
import time
import warnings
warnings.filterwarnings('ignore')

# Các hàm tiện ích từ code gốc
def load_and_prepare_data(file_path):
    df = pd.read_csv(file_path, encoding='utf-8')
    df['TimeStamp'] = pd.to_datetime(df['TimeStamp'], format='%d/%m/%Y %H:%M:%S', errors='coerce')
    df = df.reset_index(drop=True)
    return df

def scale_data(features, target):
```

```
    feature_scaler = MinMaxScaler()
    target_scaler = MinMaxScaler()
    scaled_features = feature_scaler.fit_transform(features)
    scaled_target = target_scaler.fit_transform(target.values.reshape(-1, 1))
    return scaled_features, scaled_target, feature_scaler, target_scaler

def df_to_X_y_multi_var(X, y, window_size=60, forecast_horizon=20):
    X_windows, y_windows = [], []
    for i in range(len(X) - window_size - forecast_horizon + 1):
        X_windows.append(X[i:i+window_size])
        y_windows.append(y[i+window_size:i+window_size+forecast_horizon])
    return np.array(X_windows), np.array(y_windows)

def calculate_metrics(y_true, y_pred):
    """Tính các metrics đánh giá"""
    def mape(y_true, y_pred):
        y_true, y_pred = np.array(y_true), np.array(y_pred)
        nonzero_idx = y_true != 0
        if np.sum(nonzero_idx) == 0:
            return np.nan
        return np.mean(np.abs((y_true[nonzero_idx] - y_pred[nonzero_idx]) / y_true[nonzero_idx])) * 100

    mae = mean_absolute_error(y_true, y_pred)
    mse = mean_squared_error(y_true, y_pred)
    rmse = np.sqrt(mse)
    mape_score = mape(y_true, y_pred)
    r2_value = r2_score(y_true, y_pred)
```

```
    return {
        "MAE": mae,
        "MSE": mse,
        "RMSE": rmse,
        "MAPE": mape_score,
        "R2": r2_value
    }

# Hàm xây dựng mô hình CNN-LSTM với tham số tùy chỉnh
def build_cnn_lstm_model_flexible(window_size, n_features, forecast_horizon=1,
                                  lstm_units=64, lstm_layers=1, dropout_rate=0.2):
    """
    Xây dựng mô hình CNN-LSTM với thông số linh hoạt

    Args:
        window_size: Kích thước window đầu vào
        n_features: Số lượng features
        forecast_horizon: Số bước dự báo
        lstm_units: Số units trong mỗi LSTM layer
        lstm_layers: Số lượng LSTM layers
        dropout_rate: Tỷ lệ dropout
    """
    model = Sequential()
    model.add(InputLayer((window_size, n_features)))

    # CNN layers
    model.add(Conv1D(filters=64, kernel_size=3, activation='relu', padding='same'))
    model.add(Dropout(dropout_rate))
    model.add(Conv1D(filters=32, kernel_size=2, activation='relu', padding='same'))
```

```
    model.add(Dropout(dropout_rate))

    # LSTM layers
    for i in range(lstm_layers):
        if i == lstm_layers - 1:  # Last LSTM layer
            model.add(LSTM(lstm_units))
        else:  # Intermediate LSTM layers
            model.add(LSTM(lstm_units, return_sequences=True))
        model.add(Dropout(dropout_rate + 0.1))  # Slightly higher dropout for LSTM

    # Dense layers
    model.add(Dense(32, activation='relu'))
    model.add(Dropout(dropout_rate))
    model.add(Dense(8, activation='relu'))
    model.add(Dense(forecast_horizon, activation='linear'))

    model.compile(
        loss=MeanSquaredError(),
        optimizer=Adam(learning_rate=0.001),
        metrics=[RootMeanSquaredError()]
    )
    return model

def train_and_evaluate_model(X_train, y_train, X_val, y_val, X_test, y_test,
                             target_scaler, window_size, n_features, forecast_horizon,
                             lstm_units, lstm_layers, batch_size, max_epochs=30):
    """
    Huấn luyện và đánh giá mô hình với tham số cho trước
```

```
    Returns:
        dict: Kết quả đánh giá và thông tin huấn luyện
    """
    print(f"Đang huấn luyện: LSTM_units={lstm_units}, LSTM_layers={lstm_layers}, Batch_size={batch_size}")

    # Xây dựng mô hình
    model = build_cnn_lstm_model_flexible(
        window_size=window_size,
        n_features=n_features,
        forecast_horizon=forecast_horizon,
        lstm_units=lstm_units,
        lstm_layers=lstm_layers
    )

    # Callbacks
    model_name = f'model_lstm{lstm_units}_layers{lstm_layers}_batch{batch_size}.keras'
    cp = ModelCheckpoint(model_name, save_best_only=True, verbose=0)
    early_stop = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True, verbose=0)

    # Huấn luyện
    start_time = time.time()
    try:
        history = model.fit(
            X_train, y_train,
            validation_data=(X_val, y_val),
            epochs=max_epochs,
            batch_size=batch_size,
            callbacks=[cp, early_stop],
```

```
    verbose=0  # Tắt output để giảm spam
)

training_time = time.time() - start_time

# Tải mô hình tốt nhất
best_model = load_model(model_name)

# Dự đoán
pred_test = best_model.predict(X_test, verbose=0)

# Xử lý shape cho multi-step prediction
if len(y_test.shape) == 3:
    y_test_2d = y_test.reshape(y_test.shape[0], y_test.shape[1])
else:
    y_test_2d = y_test
if len(pred_test.shape) == 3:
    pred_test_2d = pred_test.reshape(pred_test.shape[0], pred_test.shape[1])
else:
    pred_test_2d = pred_test

# Nghịch chuẩn hóa
y_test_original = target_scaler.inverse_transform(y_test_2d)
pred_test_original = target_scaler.inverse_transform(pred_test_2d)

# Tính metrics cho bước đầu tiên
metrics = calculate_metrics(y_test_original[:, 0], pred_test_original[:, 0])

# Lấy thông tin từ history
```

```
        best_epoch = len(history.history['loss'])
        final_train_loss = history.history['loss'][-1]
        final_val_loss = history.history['val_loss'][-1]
        best_val_loss = min(history.history['val_loss'])

        result = {
            'lstm_units': lstm_units,
            'lstm_layers': lstm_layers,
            'batch_size': batch_size,
            'training_time': training_time,
            'best_epoch': best_epoch,
            'final_train_loss': final_train_loss,
            'final_val_loss': final_val_loss,
            'best_val_loss': best_val_loss,
            'model_path': model_name,
            'metrics': metrics,
            'status': 'success'
        }

        # Cleanup
        del model, best_model
        tf.keras.backend.clear_session()

        return result

    except Exception as e:
        print(f"Lỗi khi huấn luyện: {str(e)}")
        return {
            'lstm_units': lstm_units,
```

```
            'lstm_layers': lstm_layers,
            'batch_size': batch_size,
            'status': 'failed',
            'error': str(e)
        }

def grid_search_cnn_lstm(X_train, y_train, X_val, y_val, X_test, y_test,
                         target_scaler, window_size, n_features, forecast_horizon):
    """
    Thực hiện Grid Search cho CNN-LSTM
    """
    # Định nghĩa grid search parameters
    param_grid = {
        'lstm_units': [32, 64, 128],
        'lstm_layers': [1, 2, 3],
        'batch_size': [32, 64, 128]
    }

    print("=== BẮT ĐẦU GRID SEARCH CNN-LSTM ===")
    print(f"Tổng số combinations: {len(param_grid['lstm_units']) * len(param_grid['lstm_layers']) * len(param_grid['batch_size'])}")

    # Tạo tất cả combinations
    param_combinations = list(itertools.product(
        param_grid['lstm_units'],
        param_grid['lstm_layers'],
        param_grid['batch_size']
    ))
```

```
    results = []

    for i, (lstm_units, lstm_layers, batch_size) in enumerate(param_combinations, 1):
        print(f"\n--- Combination {i}/{len(param_combinations)} ---")

        result = train_and_evaluate_model(
            X_train, y_train, X_val, y_val, X_test, y_test,
            target_scaler, window_size, n_features, forecast_horizon,
            lstm_units, lstm_layers, batch_size
        )

        results.append(result)

        # In kết quả ngay lập tức
        if result['status'] == 'success':
            metrics = result['metrics']
            print(f"R² = {metrics['R2']:.4f}, RMSE = {metrics['RMSE']:.4f}, "
                  f"Time = {result['training_time']:.1f}s, Epochs = {result['best_epoch']}")
        else:
            print(f"FAILED: {result.get('error', 'Unknown error')}")

    return results

def analyze_results(results):
    """
    Phân tích và hiển thị kết quả grid search
    """
    # Lọc các kết quả thành công
    successful_results = [r for r in results if r['status'] == 'success']
```

```
if not successful_results:
    print("Không có kết quả thành công nào!")
    return None

print(f"\n=== PHÂN TÍCH KẾT QUẢ GRID SEARCH ===")
print(f"Số lượng thành công: {len(successful_results)}/{len(results)}")

# Tạo DataFrame để dễ phân tích
df_results = pd.DataFrame([
    {
        'LSTM_Units': r['lstm_units'],
        'LSTM_Layers': r['lstm_layers'],
        'Batch_Size': r['batch_size'],
        'R2': r['metrics']['R2'],
        'RMSE': r['metrics']['RMSE'],
        'MAE': r['metrics']['MAE'],
        'MAPE': r['metrics']['MAPE'],
        'Training_Time': r['training_time'],
        'Best_Epoch': r['best_epoch'],
        'Best_Val_Loss': r['best_val_loss']
    }
    for r in successful_results
])

# Sắp xếp theo R²
df_results = df_results.sort_values('R2', ascending=False)

print("\n=== TOP 5 MÔ HÌNH TỐT NHẤT (theo R²) ===")
```

```
print(df_results.head().to_string(index=False, float_format='%.4f'))

# Tìm best model
best_result = successful_results[df_results.index[0]]

print(f"\n=== MÔ HÌNH TỐT NHẤT ===")
print(f"LSTM Units: {best_result['lstm_units']}")
print(f"LSTM Layers: {best_result['lstm_layers']}")
print(f"Batch Size: {best_result['batch_size']}")
print(f"R²: {best_result['metrics']['R2']:.4f}")
print(f"RMSE: {best_result['metrics']['RMSE']:.4f}")
print(f"MAE: {best_result['metrics']['MAE']:.4f}")
print(f"MAPE: {best_result['metrics']['MAPE']:.2f}%")
print(f"Training Time: {best_result['training_time']:.1f}s")
print(f"Best Epoch: {best_result['best_epoch']}")

# Phân tích theo từng tham số
print(f"\n=== PHÂN TÍCH THEO THAM SỐ ===")

# Theo LSTM Units
print("\nTheo LSTM Units:")
lstm_units_analysis = df_results.groupby('LSTM_Units').agg({
    'R2': ['mean', 'std', 'max'],
    'RMSE': ['mean', 'std', 'min'],
    'Training_Time': 'mean'
}).round(4)
print(lstm_units_analysis)

# Theo LSTM Layers
```

```
    print("\nTheo LSTM Layers:")
    lstm_layers_analysis = df_results.groupby('LSTM_Layers').agg({
        'R2': ['mean', 'std', 'max'],
        'RMSE': ['mean', 'std', 'min'],
        'Training_Time': 'mean'
    }).round(4)
    print(lstm_layers_analysis)

    # Theo Batch Size
    print("\nTheo Batch Size:")
    batch_size_analysis = df_results.groupby('Batch_Size').agg({
        'R2': ['mean', 'std', 'max'],
        'RMSE': ['mean', 'std', 'min'],
        'Training_Time': 'mean'
    }).round(4)
    print(batch_size_analysis)

    return df_results, best_result

def plot_grid_search_results(df_results):
    """
    Vẽ biểu đồ phân tích kết quả grid search
    """
    fig, axes = plt.subplots(2, 3, figsize=(18, 12))
    fig.suptitle('Grid Search Results Analysis', fontsize=16, fontweight='bold')

    # 1. R² theo LSTM Units
    lstm_units_r2 = df_results.groupby('LSTM_Units')['R2'].agg(['mean', 'std'])
    axes[0,0].bar(lstm_units_r2.index, lstm_units_r2['mean'],
```

```
        yerr=lstm_units_r2['std'], capsize=5, alpha=0.7, color='skyblue')
axes[0,0].set_title('R² theo LSTM Units')
axes[0,0].set_xlabel('LSTM Units')
axes[0,0].set_ylabel('R²')
axes[0,0].grid(True, alpha=0.3)

# 2. R² theo LSTM Layers
lstm_layers_r2 = df_results.groupby('LSTM_Layers')['R2'].agg(['mean', 'std'])
axes[0,1].bar(lstm_layers_r2.index, lstm_layers_r2['mean'],
        yerr=lstm_layers_r2['std'], capsize=5, alpha=0.7, color='lightgreen')
axes[0,1].set_title('R² theo LSTM Layers')
axes[0,1].set_xlabel('LSTM Layers')
axes[0,1].set_ylabel('R²')
axes[0,1].grid(True, alpha=0.3)

# 3. R² theo Batch Size
batch_size_r2 = df_results.groupby('Batch_Size')['R2'].agg(['mean', 'std'])
axes[0,2].bar(batch_size_r2.index, batch_size_r2['mean'],
        yerr=batch_size_r2['std'], capsize=5, alpha=0.7, color='salmon')
axes[0,2].set_title('R² theo Batch Size')
axes[0,2].set_xlabel('Batch Size')
axes[0,2].set_ylabel('R²')
axes[0,2].grid(True, alpha=0.3)

# 4. RMSE theo LSTM Units
lstm_units_rmse = df_results.groupby('LSTM_Units')['RMSE'].agg(['mean', 'std'])
axes[1,0].bar(lstm_units_rmse.index, lstm_units_rmse['mean'],
        yerr=lstm_units_rmse['std'], capsize=5, alpha=0.7, color='orange')
axes[1,0].set_title('RMSE theo LSTM Units')
```

```
axes[1,0].set_xlabel('LSTM Units')
axes[1,0].set_ylabel('RMSE')
axes[1,0].grid(True, alpha=0.3)

# 5. Training Time theo cấu hình
df_pivot = df_results.pivot_table(values='Training_Time',
                                  index='LSTM_Layers',
                                  columns='LSTM_Units',
                                  aggfunc='mean')
im = axes[1,1].imshow(df_pivot.values, cmap='YlOrRd', aspect='auto')
axes[1,1].set_xticks(range(len(df_pivot.columns)))
axes[1,1].set_yticks(range(len(df_pivot.index)))
axes[1,1].set_xticklabels(df_pivot.columns)
axes[1,1].set_yticklabels(df_pivot.index)
axes[1,1].set_xlabel('LSTM Units')
axes[1,1].set_ylabel('LSTM Layers')
axes[1,1].set_title('Training Time Heatmap')
plt.colorbar(im, ax=axes[1,1])

# 6. Scatter plot R² vs Training Time
scatter = axes[1,2].scatter(df_results['Training_Time'], df_results['R2'],
                            c=df_results['LSTM_Units'], cmap='viridis', alpha=0.7, s=50)
axes[1,2].set_xlabel('Training Time (seconds)')
axes[1,2].set_ylabel('R²')
axes[1,2].set_title('R² vs Training Time')
axes[1,2].grid(True, alpha=0.3)
plt.colorbar(scatter, ax=axes[1,2], label='LSTM Units')

plt.tight_layout()
```

```
    return plt

# Hàm chính để chạy grid search
def main_grid_search():
    """
    Hàm chính để chạy toàn bộ quá trình grid search
    """
    # Thiết lập tham số
    WINDOW_SIZE = 60
    FORECAST_HORIZON = 20
    FILE_PATH = 'D:/Download/solieu1234.csv'  # Thay đổi đường dẫn phù hợp

    print("=== CHUẨN BỊ DỮ LIỆU ===")

    # Đọc và chuẩn bị dữ liệu
    df = load_and_prepare_data(FILE_PATH)

    # Tách đặc trưng và mục tiêu
    feature_cols = [col for col in df.columns if col not in ['TimeStamp']]
    features = df[feature_cols]
    target = df['Tiêu hao điện(KWh/T)']

    # Chuẩn hóa dữ liệu
    scaled_features, scaled_target, feature_scaler, target_scaler = scale_data(features, target)

    # Tạo dữ liệu window
    X, y = df_to_X_y_multi_var(scaled_features, scaled_target,
                               window_size=WINDOW_SIZE,
                               forecast_horizon=FORECAST_HORIZON)
```

```
# Chia tập dữ liệu
train_size = 35000
val_size = 4500
X_train, y_train = X[:train_size], y[:train_size]
X_val, y_val = X[train_size:train_size+val_size], y[train_size:train_size+val_size]
X_test, y_test = X[train_size+val_size:], y[train_size+val_size:]

print(f"Kích thước dữ liệu - Train: {X_train.shape}, Val: {X_val.shape}, Test: {X_test.shape}")

# Chạy Grid Search
results = grid_search_cnn_lstm(
    X_train, y_train, X_val, y_val, X_test, y_test,
    target_scaler, WINDOW_SIZE, X.shape[2], FORECAST_HORIZON
)

# Phân tích kết quả
df_results, best_result = analyze_results(results)

if df_results is not None:
    # Vẽ biểu đồ
    plt_results = plot_grid_search_results(df_results)
    plt_results.savefig('grid_search_results_analysis.png', dpi=300, bbox_inches='tight')
    plt_results.show()

    # Lưu kết quả vào file
    df_results.to_csv('grid_search_results.csv', index=False)
    print(f"\nĐã lưu kết quả vào 'grid_search_results.csv'")
```

```
        return df_results, best_result

    return None, None

# Chạy grid search
if __name__ == "__main__":
    df_results, best_result = main_grid_search()
```

2. Chương trình dự đoán công suất điện tiêu thụ

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow.keras.models import Sequential, load_model
from tensorflow.keras.layers import InputLayer, LSTM, Dense, Conv1D, Dropout
from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping
from tensorflow.keras.losses import MeanSquaredError
from tensorflow.keras.metrics import RootMeanSquaredError
from tensorflow.keras.optimizers import Adam
from sklearn.metrics import r2_score
from sklearn.preprocessing import MinMaxScaler
from sklearn.linear_model import LinearRegression

# Thiết lập font Times New Roman cho tất cả các đồ thị với chữ in đậm
plt.rcParams['font.family'] = 'Times New Roman'
plt.rcParams['font.size'] = 13
plt.rcParams['font.weight'] = 'bold'
```

```
plt.rcParams['axes.labelweight'] = 'bold'
plt.rcParams['axes.titleweight'] = 'bold'
plt.rcParams['legend.fontsize'] = 13
plt.rcParams['xtick.labelsize'] = 13
plt.rcParams['ytick.labelsize'] = 13

# 1. Đọc dữ liệu và tiền xử lý
def load_and_prepare_data(file_path):
    df = pd.read_csv(file_path, encoding='utf-8')
    df['TimeStamp'] = pd.to_datetime(df['TimeStamp'], format='%d/%m/%Y %H:%M:%S', errors='coerce')
    df = df.reset_index(drop=True)
    return df

# 2. Chuẩn hóa dữ liệu
def scale_data(features, target):
    feature_scaler = MinMaxScaler()
    target_scaler = MinMaxScaler()
    scaled_features = feature_scaler.fit_transform(features)
    scaled_target = target_scaler.fit_transform(target.values.reshape(-1, 1))
    return scaled_features, scaled_target, feature_scaler, target_scaler

# 3. Hàm tạo dữ liệu window cho dự báo chuỗi thời gian
def df_to_X_y_multi_var(X, y, window_size=60, forecast_horizon=20):
    X_windows, y_windows = [], []
    for i in range(len(X) - window_size - forecast_horizon + 1):
        X_windows.append(X[i:i+window_size])
        y_windows.append(y[i+window_size:i+window_size+forecast_horizon])
    return np.array(X_windows), np.array(y_windows)
```

```
# 4. Xây dựng mô hình CNN-LSTM với Dropout
def build_cnn_lstm_model(window_size, n_features, forecast_horizon=1):
    model = Sequential()
    model.add(InputLayer((window_size, n_features)))
    model.add(Conv1D(filters=64, kernel_size=3, activation='relu', padding='same'))
    model.add(Dropout(0.2))
    model.add(Conv1D(filters=32, kernel_size=2, activation='relu', padding='same'))
    model.add(Dropout(0.2))
    model.add(LSTM(64))
    model.add(Dropout(0.3))
    model.add(Dense(32, activation='tanh'))
    model.add(Dropout(0.2))
    model.add(Dense(8, activation='tanh'))
    model.add(Dense(forecast_horizon, activation='tanh'))
    model.compile(
        loss=MeanSquaredError(),
        optimizer=Adam(learning_rate=0.001),
        metrics=[RootMeanSquaredError()]
    )
    return model

# 5. Hàm tính các chỉ số đánh giá
def calculate_metrics(y_true, y_pred):
    def mape(y_true, y_pred):
        y_true, y_pred = np.array(y_true), np.array(y_pred)
        nonzero_idx = y_true != 0
        if np.sum(nonzero_idx) == 0:
            return np.nan
```

```
    return np.mean(np.abs((y_true[nonzero_idx] - y_pred[nonzero_idx]) / y_true[nonzero_idx])) * 100


  mae = np.mean(np.abs(y_true - y_pred))
  mse = np.mean((y_true - y_pred) ** 2)
  rmse = np.sqrt(mse)
  mape_score = mape(y_true, y_pred)
  numerator = np.sum((y_true - y_pred) ** 2)
  denominator = np.sum((y_true - np.mean(y_true)) ** 2)
  rse = numerator / denominator
  r2_value = r2_score(y_true, y_pred)
  return {
    "MAE": mae,
    "MSE": mse,
    "RMSE": rmse,
    "MAPE": mape_score,
    "RSE": rse,
    "R2": r2_value
  }

# 6. Hàm trực quan hóa kết quả với y-axis min/max theo 'Muc tieu thu dien'
def visualize_results(actual, pred_cnnlstm, time_labels, y_min, y_max, title="So sánh kết quả dự báo"):
  plt.figure(figsize=(12, 6))
  plt.plot(time_labels, actual, 'o-', label='Giá trị thực tế', color='green', linewidth=2)
  plt.plot(time_labels, pred_cnnlstm, '^--', label='Dự báo CNN-LSTM', color='red', alpha=0.7, linewidth=2)
  plt.title(title, fontsize=13, fontweight='bold', fontfamily='Times New Roman')
  plt.xlabel("Thời gian", fontsize=13, fontweight='bold', fontfamily='Times New Roman')
```

```
    plt.ylabel("Tiêu hao điện(KWh/T)", fontsize=13, fontweight='bold', fontfamily='Times New Roman')
    plt.xticks(rotation=45, fontsize=13, fontweight='bold', fontfamily='Times New Roman')
    plt.yticks(fontsize=13, fontweight='bold', fontfamily='Times New Roman')
    plt.grid(True, alpha=0.3)
    plt.legend(fontsize=13, prop={'family': 'Times New Roman', 'weight': 'bold'})
    plt.ylim([y_min, y_max])
    plt.tight_layout()
    return plt

# 7. HÀM VẼ BIỂU ĐỒ HỒI QUY
def plot_regression_step1(actual_values, predicted_values, title="Biểu đồ hồi quy: Tiêu hao điện dự đoán vs thực tế "):
    """
    Vẽ biểu đồ hồi quy tuyến tính

    Parameters:
    actual_values: Giá trị thực tế (array)
    predicted_values: Giá trị dự đoán (array)
    title: Tiêu đề biểu đồ
    """

    plt.figure(figsize=(10, 8))

    # Vẽ scatter plot với màu xanh dương
    plt.scatter(predicted_values, actual_values, alpha=0.6, color='blue', s=30,
            edgecolors='darkblue', linewidth=0.5, label='Dữ liệu thực tế')

    # Tính toán đường hồi quy tuyến tính
    lr = LinearRegression()
```

```
predicted_reshaped = predicted_values.reshape(-1, 1)
lr.fit(predicted_reshaped, actual_values)

# Vẽ đường hồi quy màu đỏ
x_range = np.linspace(predicted_values.min(), predicted_values.max(), 100)
y_pred_line = lr.predict(x_range.reshape(-1, 1))
plt.plot(x_range, y_pred_line, color='red', linewidth=2,
        label=f'Đường hồi quy: y = {lr.coef_[0]:.3f}x + {lr.intercept_:.3f}')

# Tính toán trung tâm dữ liệu để đường y = x đi qua giữa
center_x = (predicted_values.min() + predicted_values.max()) / 2
center_y = (actual_values.min() + actual_values.max()) / 2

# Tính toán phạm vi cho đường y = x dựa trên trung tâm
data_range = max(predicted_values.max() - predicted_values.min(),
                actual_values.max() - actual_values.min())
half_range = data_range * 0.6  # Mở rộng một chút để đường line rõ ràng

# Vẽ đường y = x đi qua trung tâm dữ liệu
ideal_x_start = center_x - half_range
ideal_x_end = center_x + half_range
ideal_y_start = center_y - half_range
ideal_y_end = center_y + half_range

plt.plot([ideal_x_start, ideal_x_end], [ideal_y_start, ideal_y_end],
        'k--', alpha=0.7, linewidth=2, label='Đường lý tưởng (y = x)')

# Tính các chỉ số đánh giá
r2 = r2_score(actual_values, predicted_values)
```

```
    rmse = np.sqrt(np.mean((actual_values - predicted_values) ** 2))
    mae = np.mean(np.abs(actual_values - predicted_values))

    # Thêm hộp thông tin thống kê với font Times New Roman in đậm
    stats_text = f'R² = {r2:.4f}\nRMSE = {rmse:.3f}\nMAE = {mae:.3f}\nSố mẫu = {len(actual_values)}'
    plt.text(0.05, 0.95, stats_text, transform=plt.gca().transAxes, fontsize=13,
             fontfamily='Times New Roman', fontweight='bold', verticalalignment='top',
             bbox=dict(boxstyle='round', facecolor='wheat', alpha=0.9))

    # Thiết lập nhãn và tiêu đề với font Times New Roman in đậm
    plt.xlabel('Tiêu hao điện dự đoán (KWh/T)', fontsize=13, fontweight='bold', fontfamily='Times New Roman')
    plt.ylabel('Tiêu hao điện thực tế (KWh/T)', fontsize=13, fontweight='bold', fontfamily='Times New Roman')
    plt.title(title, fontsize=13, fontweight='bold', fontfamily='Times New Roman')
    plt.legend(fontsize=13, prop={'family': 'Times New Roman', 'weight': 'bold'})
    plt.grid(True, alpha=0.3)

    # Thiết lập font cho tick labels
    plt.xticks(fontsize=13, fontweight='bold', fontfamily='Times New Roman')
    plt.yticks(fontsize=13, fontweight='bold', fontfamily='Times New Roman')

    # Thiết lập giới hạn trục với buffer nhỏ
    min_val = min(predicted_values.min(), actual_values.min())
    max_val = max(predicted_values.max(), actual_values.max())
    buffer = (max_val - min_val) * 0.02
    plt.xlim(min_val - buffer, max_val + buffer)
    plt.ylim(min_val - buffer, max_val + buffer)
```

```
    # Đảm bảo tỷ lệ trục bằng nhau
    plt.gca().set_aspect('equal', adjustable='box')

    plt.tight_layout()
    return plt

# 8. Chương trình chính
def main():
    # Thiết lập tham số
    WINDOW_SIZE = 60
    FORECAST_HORIZON = 20
    FILE_PATH = 'D:/Download/solieu1234.csv'

    # Đọc và chuẩn bị dữ liệu
    print("Đang đọc và chuẩn bị dữ liệu...")
    df = load_and_prepare_data(FILE_PATH)

    # Tách đặc trưng và mục tiêu
    feature_cols = [col for col in df.columns if col not in ['TimeStamp']]
    features = df[feature_cols]
    target = df['Tiêu hao điện(KWh/T)']

    # Xác định min, max của mục tiêu để dùng cho y-axis
    y_min = target.min()
    y_max = target.max()

    # Chuẩn hóa dữ liệu
    print("Đang chuẩn hóa dữ liệu...")
    scaled_features, scaled_target, feature_scaler, target_scaler = scale_data(features, target)
```

```
# Tạo dữ liệu window
print("Đang tạo dữ liệu window cho dự báo...")
X, y = df_to_X_y_multi_var(scaled_features, scaled_target,
                           window_size=WINDOW_SIZE,
                           forecast_horizon=FORECAST_HORIZON)

# Chia tập dữ liệu
train_size = 35000
val_size = 4500
X_train, y_train = X[:train_size], y[:train_size]
X_val, y_val = X[train_size:train_size+val_size], y[train_size:train_size+val_size]
X_test, y_test = X[train_size+val_size:], y[train_size+val_size:]

print(f"Kích thước dữ liệu - Train: {X_train.shape}, Val: {X_val.shape}, Test: {X_test.shape}")

# Xây dựng và huấn luyện mô hình CNN-LSTM
print("\nĐang xây dựng và huấn luyện mô hình CNN-LSTM với Dropout và EarlyStopping...")
model_cnnlstm = build_cnn_lstm_model(WINDOW_SIZE, X.shape[2], FORECAST_HORIZON)
cp_cnnlstm = ModelCheckpoint('model_cnnlstm.keras', save_best_only=True)
early_stop = EarlyStopping(monitor='val_loss', patience=10, restore_best_weights=True)
history_cnnlstm = model_cnnlstm.fit(
    X_train, y_train,
    validation_data=(X_val, y_val),
    epochs=50,
    batch_size=32,
    callbacks=[cp_cnnlstm, early_stop],
```

```
    verbose=1
)

# Tải mô hình tốt nhất
print("\nĐang tải mô hình tốt nhất...")
best_cnnlstm = load_model('model_cnnlstm.keras')

# Dự đoán trên tập test
print("Đang dự báo trên tập test...")
pred_cnnlstm = best_cnnlstm.predict(X_test)

# Đảm bảo y_test và pred_cnnlstm là 2 chiều
if len(y_test.shape) == 3:
    y_test_2d = y_test.reshape(y_test.shape[0], y_test.shape[1])
else:
    y_test_2d = y_test
if len(pred_cnnlstm.shape) == 3:
    pred_cnnlstm_2d = pred_cnnlstm.reshape(pred_cnnlstm.shape[0],
pred_cnnlstm.shape[1])
else:
    pred_cnnlstm_2d = pred_cnnlstm

# Nghịch chuẩn hóa cho multi-step (shape: [num_samples, 20])
y_test_original = target_scaler.inverse_transform(y_test_2d)
pred_cnnlstm_original = target_scaler.inverse_transform(pred_cnnlstm_2d)

# Đánh giá mô hình cho bước đầu tiên
print("\n== Chỉ số đánh giá mô hình CNN-LSTM  ==")
metrics_cnnlstm = calculate_metrics(y_test_original[:, 0], pred_cnnlstm_original[:, 0])
```

```
for metric, value in metrics_cnnlstm.items():
    print(f"CNN-LSTM - {metric}: {value}")

# In ra 20 giá trị dự đoán đầu tiên cho bước 1
print("\n20 giá trị dự đoán và thực tế cho bước dự báo thứ nhất (CNN-LSTM):")
for i in range(20):
    print(f"CNN-LSTM - Sample {i+1}: Predicted = {pred_cnnlstm_original[i, 0]:.3f}, Actual = {y_test_original[i, 0]:.3f}")

# Trực quan hóa kết quả 50 mẫu đầu tiên cho bước dự báo thứ nhất
print("\nĐang trực quan hóa kết quả bước thứ nhất...")
start_idx = WINDOW_SIZE + train_size + val_size
time_indices = [start_idx + i for i in range(50)]  # Lấy 50 mẫu để trực quan hóa
time_labels = df['TimeStamp'].iloc[time_indices].dt.strftime('%Y-%m-%d %H:%M').to_list()
plt1 = visualize_results(
    y_test_original[:50, 0],
    pred_cnnlstm_original[:50, 0],
    time_labels, y_min, y_max,
    "So sánh kết quả dự báo mức tiêu thụ điện"
)
plt1.savefig('forecast_comparison_cnnlstm_step1.png')
plt1.show()

# VẼ BIỂU ĐỒ HỒI QUY
print("\nĐang vẽ biểu đồ hồi quy tuyến tính...")
plt_regression = plot_regression_step1(
    y_test_original[:, 0],
    pred_cnnlstm_original[:, 0],
    "CNN-LSTM: Biểu đồ hồi quy - Tiêu hao điện dự đoán vs thực tế"
```

```
    )
    plt_regression.savefig('regression_step1_cnnlstm.png', dpi=300, bbox_inches='tight')
    plt_regression.show()

    # Vẽ biểu đồ loss trong quá trình huấn luyện với font Times New Roman in đậm
    plt.figure(figsize=(6, 5))
    plt.plot(history_cnnlstm.history['loss'], label='Train', linewidth=2)
    plt.plot(history_cnnlstm.history['val_loss'], label='Validation', linewidth=2)
    plt.title('CNN-LSTM - Loss', fontsize=13, fontweight='bold', fontfamily='Times New Roman')
    plt.xlabel('Epoch', fontsize=13, fontweight='bold', fontfamily='Times New Roman')
    plt.ylabel('Loss', fontsize=13, fontweight='bold', fontfamily='Times New Roman')
    plt.xticks(fontsize=13, fontweight='bold', fontfamily='Times New Roman')
    plt.yticks(fontsize=13, fontweight='bold', fontfamily='Times New Roman')
    plt.legend(fontsize=13, prop={'family': 'Times New Roman', 'weight': 'bold'})
    plt.grid(True, alpha=0.3)
    plt.tight_layout()
    plt.savefig('training_loss_cnnlstm.png')
    plt.show()

    print("Quá trình hoàn tất!")
    print("Các file đã được lưu:")
    print("- forecast_comparison_cnnlstm_step1.png: So sánh dự báo theo thời gian")
    print("- regression_step1_cnnlstm.png: Biểu đồ hồi quy ")
    print("- training_loss_cnnlstm.png: Biểu đồ loss huấn luyện")

if __name__ == "__main__":
    main()
```

3. Chương trình tối ưu hoá điện năng tiêu thụ

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow.keras.models import Sequential, load_model
from tensorflow.keras.layers import InputLayer, LSTM, Dense, Conv1D, Dropout
from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping
from tensorflow.keras.losses import MeanSquaredError
from tensorflow.keras.metrics import RootMeanSquaredError
from tensorflow.keras.optimizers import Adam
from sklearn.metrics import r2_score
from sklearn.preprocessing import MinMaxScaler
from sklearn.linear_model import LinearRegression
from scipy.stats import qmc
import warnings
warnings.filterwarnings('ignore')


# 1. Đọc dữ liệu và tiền xử lý
def load_and_prepare_data(file_path):
    """Đọc dữ liệu từ file Excel và chuẩn bị"""
    df = pd.read_excel(file_path, engine='openpyxl')

    # Đổi tên cột theo mô tả
    column_mapping = {
        df.columns[0]: 'TimeStamp',  # Cột A
        df.columns[1]: 'Toc_do_cap_lieu',  # Cột B - Tốc độ cấp liệu(T/h)
        df.columns[8]: 'Toc_do_cap_nuoc',  # Cột I - Tốc độ cấp nước (m³/h)
```

```
        df.columns[10]: 'San_luong',  # Cột K - Sản lượng(T/h)
        df.columns[11]: 'Tieu_hao_dien'  # Cột L - Tiêu hao điện(KWh/T)
    }

    # Đổi tên các cột quan trọng
    for old_name, new_name in column_mapping.items():
        if old_name in df.columns:
            df = df.rename(columns={old_name: new_name})

    # Chuyển đổi TimeStamp
    df['TimeStamp'] = pd.to_datetime(df['TimeStamp'], errors='coerce')
    df = df.dropna(subset=['TimeStamp']).reset_index(drop=True)

    return df

# 2. Chuẩn hóa dữ liệu
def scale_data(features, target):
    feature_scaler = MinMaxScaler()
    target_scaler = MinMaxScaler()
    scaled_features = feature_scaler.fit_transform(features)
    scaled_target = target_scaler.fit_transform(target.values.reshape(-1, 1))
    return scaled_features, scaled_target, feature_scaler, target_scaler

# 3. Hàm tạo dữ liệu window cho dự báo chuỗi thời gian
def df_to_X_y_multi_var(X, y, window_size=60, forecast_horizon=20):
    X_windows, y_windows = [], []
    for i in range(len(X) - window_size - forecast_horizon + 1):
        X_windows.append(X[i:i+window_size])
        y_windows.append(y[i+window_size:i+window_size+forecast_horizon])
```

```
    return np.array(X_windows), np.array(y_windows)

# 4. Xây dựng mô hình CNN-LSTM với Dropout
def build_cnn_lstm_model(window_size, n_features, forecast_horizon=20):
    model = Sequential()
    model.add(InputLayer((window_size, n_features)))
    model.add(Conv1D(filters=64, kernel_size=3, activation='relu', padding='same'))
    model.add(Dropout(0.2))
    model.add(Conv1D(filters=32, kernel_size=2, activation='relu', padding='same'))
    model.add(Dropout(0.2))
    model.add(LSTM(64))
    model.add(Dropout(0.3))
    model.add(Dense(32, activation='tanh'))
    model.add(Dropout(0.2))
    model.add(Dense(8, activation='tanh'))
    model.add(Dense(forecast_horizon, activation='tanh'))
    model.compile(
        loss=MeanSquaredError(),
        optimizer=Adam(learning_rate=0.001),
        metrics=[RootMeanSquaredError()]
    )
    return model

# 5. Hàm tính các chỉ số đánh giá
def calculate_metrics(y_true, y_pred):
    def mape(y_true, y_pred):
        y_true, y_pred = np.array(y_true), np.array(y_pred)
        nonzero_idx = y_true != 0
        if np.sum(nonzero_idx) == 0:
```

```
        return np.nan
    return np.mean(np.abs((y_true[nonzero_idx] - y_pred[nonzero_idx]) / y_true[nonzero_idx])) * 100

  mae = np.mean(np.abs(y_true - y_pred))
  mse = np.mean((y_true - y_pred) ** 2)
  rmse = np.sqrt(mse)
  mape_score = mape(y_true, y_pred)
  numerator = np.sum((y_true - y_pred) ** 2)
  denominator = np.sum((y_true - np.mean(y_true)) ** 2)
  rse = numerator / denominator
  r2_value = r2_score(y_true, y_pred)
  return {
    "MAE": mae,
    "MSE": mse,
    "RMSE": rmse,
    "MAPE": mape_score,
    "RSE": rse,
    "R2": r2_value
  }

# 6. Hàm Latin Hypercube Sampling
def generate_lhs_samples(center_toc_do_cap_lieu, center_toc_do_cap_nuoc, n_samples=100):
  """
  Tạo các mẫu LHS cho 2 thông số
  center_toc_do_cap_lieu ± 5
  center_toc_do_cap_nuoc ± 0.2
  """
  sampler = qmc.LatinHypercube(d=2, seed=42)
```

```
    samples = sampler.random(n=n_samples)

    # Chuyển đổi từ [0,1] sang phạm vi mong muốn
    toc_do_cap_lieu_range = [center_toc_do_cap_lieu - 5, center_toc_do_cap_lieu + 5]
    toc_do_cap_nuoc_range = [center_toc_do_cap_nuoc - 0.2, center_toc_do_cap_nuoc + 0.2]

    # Scale samples
    samples[:, 0] = samples[:, 0] * (toc_do_cap_lieu_range[1] - toc_do_cap_lieu_range[0]) +
toc_do_cap_lieu_range[0]
    samples[:, 1] = samples[:, 1] * (toc_do_cap_nuoc_range[1] - toc_do_cap_nuoc_range[0]) +
toc_do_cap_nuoc_range[0]

    return samples

# 7. Hàm tối ưu hóa bằng LHS cho một mẫu
def optimize_single_sample(model, test_sample, feature_scaler, target_scaler,
                           original_toc_do_cap_lieu, original_toc_do_cap_nuoc,
                           san_luong, feature_cols, df_features):
    """Tối ưu hóa cho một mẫu duy nhất"""
    # Tạo các mẫu LHS
    lhs_samples = generate_lhs_samples(original_toc_do_cap_lieu, original_toc_do_cap_nuoc,
100)

    best_tieu_hao = float('inf')
    best_params = None

    for new_toc_do_cap_lieu, new_toc_do_cap_nuoc in lhs_samples:
        # Kiểm tra điều kiện: sản lượng < tốc độ cấp liệu
        if san_luong >= new_toc_do_cap_lieu:
            continue
```

```
        # Tạo sample mới với các thông số tối ưu
        new_sample = test_sample.copy()

        # Tìm index của các cột cần thay đổi
        toc_do_cap_lieu_idx = feature_cols.index('Toc_do_cap_lieu')
        toc_do_cap_nuoc_idx = feature_cols.index('Toc_do_cap_nuoc')

        # Cập nhật giá trị mới (chỉ cập nhật điểm cuối cùng của window)
        new_sample[-1, toc_do_cap_lieu_idx] = (new_toc_do_cap_lieu - df_features['Toc_do_cap_lieu'].min()) / (df_features['Toc_do_cap_lieu'].max() - df_features['Toc_do_cap_lieu'].min())
        new_sample[-1, toc_do_cap_nuoc_idx] = (new_toc_do_cap_nuoc - df_features['Toc_do_cap_nuoc'].min()) / (df_features['Toc_do_cap_nuoc'].max() - df_features['Toc_do_cap_nuoc'].min())

        # Dự đoán với sample mới
        pred_scaled = model.predict(new_sample.reshape(1, new_sample.shape[0], new_sample.shape[1]), verbose=0)
        pred_original = target_scaler.inverse_transform(pred_scaled.reshape(-1, 1)).flatten()

        # Lấy giá trị dự đoán cho bước đầu tiên
        predicted_tieu_hao = pred_original[0]

        # Cập nhật kết quả tốt nhất
        if predicted_tieu_hao < best_tieu_hao:
            best_tieu_hao = predicted_tieu_hao
            best_params = (new_toc_do_cap_lieu, new_toc_do_cap_nuoc)

    return best_params, best_tieu_hao
```

```
# 8. Hàm tối ưu hóa bằng LHS cho nhiều mẫu
def optimize_with_lhs_multiple(model, X_test, df, train_size, WINDOW_SIZE,
                               feature_scaler, target_scaler, feature_cols, df_features, n_samples=10):
    """Tối ưu hóa cho nhiều mẫu test"""
    print(f"\n=== BẮT ĐẦU TỐI ƯU HÓA {n_samples} MẪU BẰNG LHS ===")

    optimization_results = []

    for i in range(min(n_samples, len(X_test))):
        print(f"\nTối ưu hóa mẫu {i+1}/{n_samples}...")

        test_sample = X_test[i]
        original_idx = train_size + i + WINDOW_SIZE

        if original_idx >= len(df):
            continue

        original_data = df.iloc[original_idx]

        original_toc_do_cap_lieu = original_data['Toc_do_cap_lieu']
        original_toc_do_cap_nuoc = original_data['Toc_do_cap_nuoc']
        san_luong = original_data['San_luong']
        original_tieu_hao = original_data['Tieu_hao_dien']

        # Dự đoán ban đầu (không tối ưu)
        pred_original_scaled = model.predict(test_sample.reshape(1, test_sample.shape[0], test_sample.shape[1]), verbose=0)
        pred_original_value = target_scaler.inverse_transform(pred_original_scaled.reshape(-1, 1)).flatten()[0]
```

```
# Thực hiện tối ưu hóa
best_params, best_tieu_hao = optimize_single_sample(
    model, test_sample, feature_scaler, target_scaler,
    original_toc_do_cap_lieu, original_toc_do_cap_nuoc,
    san_luong, feature_cols, df_features
)

if best_params is not None:
    improvement = original_tieu_hao - best_tieu_hao
    improvement_percent = (improvement / original_tieu_hao) * 100

    result = {
        'sample_idx': i + 1,
        'timestamp': original_data['TimeStamp'],
        'original_toc_do_cap_lieu': original_toc_do_cap_lieu,
        'original_toc_do_cap_nuoc': original_toc_do_cap_nuoc,
        'optimized_toc_do_cap_lieu': best_params[0],
        'optimized_toc_do_cap_nuoc': best_params[1],
        'san_luong': san_luong,
        'actual_tieu_hao': original_tieu_hao,
        'predicted_original': pred_original_value,
        'predicted_optimized': best_tieu_hao,
        'improvement': improvement,
        'improvement_percent': improvement_percent
    }

    optimization_results.append(result)
```

```
            print(f"  - Tiêu hao thực tế: {original_tieu_hao:.3f} KWh/T")
            print(f"  - Dự đoán ban đầu: {pred_original_value:.3f} KWh/T")
            print(f"  - Dự đoán tối ưu: {best_tieu_hao:.3f} KWh/T")
            print(f"  - Cải thiện: {improvement:.3f} KWh/T ({improvement_percent:.2f}%)")

    return optimization_results

# 9. Hàm trực quan hóa kết quả tối ưu hóa cho nhiều mẫu
def visualize_optimization_comparison(optimization_results):
    """Trực quan hóa so sánh kết quả tối ưu hóa cho nhiều mẫu"""

    if not optimization_results:
        print("Không có dữ liệu để trực quan hóa!")
        return

    results_df = pd.DataFrame(optimization_results)

    # Tạo figure với nhiều subplot
    fig = plt.figure(figsize=(20, 15))

    # Subplot 1: So sánh tiêu hao điện trước và sau tối ưu (Bar chart)
    ax1 = plt.subplot(2, 3, 1)
    x_pos = np.arange(len(results_df))
    width = 0.35

    bars1 = ax1.bar(x_pos - width/2, results_df['actual_tieu_hao'], width,
                    label='Thực tế', color='lightcoral', alpha=0.8)
    bars2 = ax1.bar(x_pos + width/2, results_df['predicted_optimized'], width,
                    label='Tối ưu', color='lightblue', alpha=0.8)
```

```
ax1.set_xlabel('Mẫu số')
ax1.set_ylabel('Tiêu hao điện (KWh/T)')
ax1.set_title('So sánh tiêu hao điện: Thực tế với Tối ưu')
ax1.set_xticks(x_pos)
ax1.set_xticklabels([f'M{i+1}' for i in range(len(results_df))])
ax1.legend()
ax1.grid(True, alpha=0.3)

# Thêm giá trị lên bar
for i, (bar1, bar2) in enumerate(zip(bars1, bars2)):
    height1 = bar1.get_height()
    height2 = bar2.get_height()
    ax1.text(bar1.get_x() + bar1.get_width()/2., height1,
             f'{height1:.2f}', ha='center', va='bottom', fontsize=8)
    ax1.text(bar2.get_x() + bar2.get_width()/2., height2,
             f'{height2:.2f}', ha='center', va='bottom', fontsize=8)

# Subplot 2: Đường xu hướng tiêu hao điện
ax2 = plt.subplot(2, 3, 2)
ax2.plot(range(1, len(results_df)+1), results_df['actual_tieu_hao'],
         'o-', color='red', linewidth=2, markersize=8, label='Thực tế')
ax2.plot(range(1, len(results_df)+1), results_df['predicted_optimized'],
         's-', color='blue', linewidth=2, markersize=8, label='Tối ưu')

ax2.set_xlabel('Mẫu số')
ax2.set_ylabel('Tiêu hao điện (KWh/T)')
ax2.set_title('Xu hướng tiêu hao điện')
ax2.legend()
```

```
ax2.grid(True, alpha=0.3)

# Subplot 3: Mức cải thiện (%)
ax3 = plt.subplot(2, 3, 3)
colors = ['green' if x > 0 else 'red' for x in results_df['improvement_percent']]
bars3 = ax3.bar(range(1, len(results_df)+1), results_df['improvement_percent'],
               color=colors, alpha=0.7)

ax3.set_xlabel('Mẫu số')
ax3.set_ylabel('Cải thiện (%)')
ax3.set_title('Mức độ cải thiện (%)')
ax3.axhline(y=0, color='black', linestyle='-', alpha=0.3)
ax3.grid(True, alpha=0.3)

# Thêm giá trị lên bar
for i, bar in enumerate(bars3):
    height = bar.get_height()
    ax3.text(bar.get_x() + bar.get_width()/2., height,
            f'{height:.1f}%', ha='center', va='bottom' if height > 0 else 'top', fontsize=8)

# Subplot 6: Thống kê tổng quan
ax4 = plt.subplot(2, 3, 4)
ax4.axis('off')

# Tính toán thống kê
total_samples = len(results_df)
avg_improvement = results_df['improvement_percent'].mean()
max_improvement = results_df['improvement_percent'].max()
min_improvement = results_df['improvement_percent'].min()
```

```
    positive_improvements = (results_df['improvement_percent'] > 0).sum()

    stats_text = f"""
    THỐNG KÊ TỔNG QUAN

    Số mẫu tối ưu: {total_samples}
    Cải thiện trung bình: {avg_improvement:.2f}%
    Cải thiện tốt nhất: {max_improvement:.2f}%
    Cải thiện thấp nhất: {min_improvement:.2f}%
    Số mẫu cải thiện: {positive_improvements}/{total_samples}

    Tiêu hao điện trung bình:
    • Trước tối ưu: {results_df['actual_tieu_hao'].mean():.3f} KWh/T
    • Sau tối ưu: {results_df['predicted_optimized'].mean():.3f} KWh/T
    • Tiết kiệm: {(results_df['actual_tieu_hao'].mean() - results_df['predicted_optimized'].mean()):.3f} KWh/T
    """

    ax4.text(0.1, 0.9, stats_text, transform=ax4.transAxes, fontsize=12,
            verticalalignment='top', bbox=dict(boxstyle='round', facecolor='lightgray', alpha=0.8))

    plt.tight_layout()
    plt.savefig('optimization_comparison_results.png', dpi=300, bbox_inches='tight')
    plt.show()

    return results_df

# 10. Chương trình chính
def main():
```

```
# Thiết lập tham số
WINDOW_SIZE = 60
FORECAST_HORIZON = 20
FILE_PATH = 'D:/Downloads/solieu1234.xlsx'  # Đổi sang .xlsx
N_OPTIMIZATION_SAMPLES = 10  # Số mẫu để tối ưu hóa

print("=== CHƯƠNG TRÌNH TỐI ƯU HÓA TIÊU HAO ĐIỆN BẰNG LHS (PHIÊN BẢN CẢI TIẾN) ===\n")

# Đọc và chuẩn bị dữ liệu
print("1. Đang đọc và chuẩn bị dữ liệu...")
df = load_and_prepare_data(FILE_PATH)

# Kiểm tra các cột cần thiết
required_cols = ['TimeStamp', 'Toc_do_cap_lieu', 'Toc_do_cap_nuoc', 'San_luong', 'Tieu_hao_dien']
for col in required_cols:
    if col not in df.columns:
        print(f"Cảnh báo: Không tìm thấy cột {col}")

print(f"Dữ liệu có {len(df)} dòng và {len(df.columns)} cột")
print(f"Các cột chính: {required_cols}")

# Tách đặc trưng và mục tiêu
feature_cols = [col for col in df.columns if col not in ['TimeStamp']]
features = df[feature_cols]
target = df['Tieu_hao_dien']

# Chuẩn hóa dữ liệu
print("\n2. Đang chuẩn hóa dữ liệu...")
```

```
scaled_features, scaled_target, feature_scaler, target_scaler = scale_data(features, target)

# Tạo dữ liệu window
print("3. Đang tạo dữ liệu window cho dự báo...")
X, y = df_to_X_y_multi_var(scaled_features, scaled_target,
                           window_size=WINDOW_SIZE,
                           forecast_horizon=FORECAST_HORIZON)

# Chia tập dữ liệu (80% train, 20% test)
train_size = int(len(X) * 0.8)
X_train, y_train = X[:train_size], y[:train_size]
X_test, y_test = X[train_size:], y[train_size:]

print(f"Kích thước dữ liệu - Train: {X_train.shape}, Test: {X_test.shape}")

# Xây dựng và huấn luyện mô hình CNN-LSTM
print("\n4. Đang xây dựng và huấn luyện mô hình CNN-LSTM...")
model = build_cnn_lstm_model(WINDOW_SIZE, X.shape[2], FORECAST_HORIZON)

# Callbacks
cp = ModelCheckpoint('model_lhs_optimization.keras', save_best_only=True)
early_stop = EarlyStopping(monitor='val_loss', patience=10, restore_best_weights=True)

# Split train thành train và validation
val_size = int(len(X_train) * 0.2)
X_val, y_val = X_train[-val_size:], y_train[-val_size:]
X_train, y_train = X_train[:-val_size], y_train[:-val_size]

history = model.fit(
```

```
    X_train, y_train,
    validation_data=(X_val, y_val),
    epochs=30,
    batch_size=32,
    callbacks=[cp, early_stop],
    verbose=1
)

# Tải mô hình tốt nhất
print("\n5. Đang tải mô hình tốt nhất...")
best_model = load_model('model_lhs_optimization.keras')

# Dự đoán trên tập test để đánh giá
print("6. Đang đánh giá mô hình trên tập test...")
pred_test = best_model.predict(X_test)

# Nghịch chuẩn hóa
if len(y_test.shape) == 3:
    y_test_2d = y_test.reshape(y_test.shape[0], y_test.shape[1])
else:
    y_test_2d = y_test
if len(pred_test.shape) == 3:
    pred_test_2d = pred_test.reshape(pred_test.shape[0], pred_test.shape[1])
else:
    pred_test_2d = pred_test

y_test_original = target_scaler.inverse_transform(y_test_2d)
pred_test_original = target_scaler.inverse_transform(pred_test_2d)
```

```
# Đánh giá mô hình
print("\n=== ĐÁNH GIÁ MÔ HÌNH CNN-LSTM ===")
metrics = calculate_metrics(y_test_original[:, 0], pred_test_original[:, 0])
for metric, value in metrics.items():
    print(f"{metric}: {value:.6f}")

# In ra 10 giá trị so sánh thực tế và dự đoán
print("\n=== 10 GIÁ TRỊ SO SÁNH THỰC TẾ VÀ DỰ ĐOÁN (BƯỚC ĐẦU TIÊN) ===")
for i in range(10):
    print(f"Mẫu {i+1:2d}: Thực tế = {y_test_original[i, 0]:8.3f}, Dự đoán = {pred_test_original[i, 0]:8.3f}, Sai số = {abs(y_test_original[i, 0] - pred_test_original[i, 0]):8.3f}")

# Thực hiện tối ưu hóa cho nhiều mẫu
print(f"\n7. Bắt đầu tối ưu hóa {N_OPTIMIZATION_SAMPLES} mẫu bằng LHS...")
optimization_results = optimize_with_lhs_multiple(
    best_model, X_test, df, train_size, WINDOW_SIZE,
    feature_scaler, target_scaler, feature_cols, features,
    n_samples=N_OPTIMIZATION_SAMPLES
)

if optimization_results:
    print(f"\n=== KẾT QUẢ TỐI ƯU HÓA CHO {len(optimization_results)} MẪU ===")

    # In kết quả chi tiết
    for i, result in enumerate(optimization_results):
        print(f"\nMẫu {result['sample_idx']}:")
        print(f"  • Thời gian: {result['timestamp']}")
        print(f"  • Sản lượng: {result['san_luong']:.3f} T/h")
```

```
        print(f"    • Tốc độ cấp liệu: {result['original_toc_do_cap_lieu']:.3f} → {result['optimized_toc_do_cap_lieu']:.3f} T/h")
        print(f"    • Tốc độ cấp nước: {result['original_toc_do_cap_nuoc']:.3f} → {result['optimized_toc_do_cap_nuoc']:.3f} m³/h")
        print(f"  • Tiêu hao điện thực tế: {result['actual_tieu_hao']:.3f} KWh/T")
        print(f"  • Dự đoán ban đầu: {result['predicted_original']:.3f} KWh/T")
        print(f"  • Dự đoán tối ưu: {result['predicted_optimized']:.3f} KWh/T")
        print(f"    • Cải thiện: {result['improvement']:.3f} KWh/T ({result['improvement_percent']:.2f}%)")

    # Tính toán thống kê tổng quan
    results_df = pd.DataFrame(optimization_results)
    print(f"\n=== THỐNG KÊ TỔNG QUAN ===")
    print(f"Số mẫu được tối ưu hóa: {len(results_df)}")
    print(f"Cải thiện trung bình: {results_df['improvement_percent'].mean():.2f}%")
    print(f"Cải thiện tốt nhất: {results_df['improvement_percent'].max():.2f}%")
    print(f"Cải thiện thấp nhất: {results_df['improvement_percent'].min():.2f}%")
    print(f"Số mẫu có cải thiện dương: {(results_df['improvement_percent'] > 0).sum()}/{len(results_df)}")
    print(f"Tiết kiệm điện trung bình: {results_df['improvement'].mean():.3f} KWh/T")

    # Trực quan hóa kết quả
    print("\n8. Đang tạo biểu đồ so sánh trước và sau tối ưu hóa...")
    final_results_df = visualize_optimization_comparison(optimization_results)

else:
    print("Không có kết quả tối ưu hóa nào được tạo ra!")

print("\n=== HOÀN THÀNH TỐI ƯU HÓA ===")
print("Các file đã được lưu:")
```

```
    print("- model_lhs_optimization.keras: Mô hình CNN-LSTM đã huấn luyện")
    print("- optimization_comparison_results.png: Biểu đồ so sánh kết quả tối ưu hóa")

if __name__ == "__main__":
    main()
```

4. Kết quả chạy Gird Search để tìm tham số cho LSTM

```
=== GRID SEARCH CNN-LSTM OPTIMIZATION ===
Thời gian bắt đầu: 2025-05-03 13:04:28
Tham số tìm kiếm: {'lstm_units': [32, 64, 128], 'lstm_layers': [1, 2, 3], 'batch_size': [32, 64, 128]}
Tổng số tổ hợp: 27
Đang đọc và chuẩn bị dữ liệu...
Đang chuẩn hóa dữ liệu...
Đang tạo dữ liệu window cho dự báo...
Kích thước dữ liệu - Train: (35000, 60, 11), Val: (4500, 60, 11), Test: (5059, 60, 11)
BẮT ĐẦU GRID SEARCH
Bắt đầu Grid Search với 27 tổ hợp tham số...
Tổ hợp 1/27: lstm_units=32, lstm_layers=1, batch_size=32
R2 Score: 0.937956
*** MÔ HÌNH TỐT NHẤT MỚI! R2 = 0.937956 ***
------------------------------------------------------------
Tổ hợp 2/27: lstm_units=32, lstm_layers=1, batch_size=64
R2 Score: 0.956973
*** MÔ HÌNH TỐT NHẤT MỚI! R2 = 0.956973 ***
------------------------------------------------------------
Tổ hợp 3/27: lstm_units=32, lstm_layers=1, batch_size=128
R2 Score: 0.952597
------------------------------------------------------------
```

Tổ hợp 4/27: lstm_units=32, lstm_layers=2, batch_size=32

R2 Score: 0.937189

Tổ hợp 5/27: lstm_units=32, lstm_layers=2, batch_size=64

R2 Score: 0.956151

Tổ hợp 6/27: lstm_units=32, lstm_layers=2, batch_size=128

R2 Score: 0.921324

Tổ hợp 7/27: lstm_units=32, lstm_layers=3, batch_size=32

R2 Score: 0.943346

Tổ hợp 8/27: lstm_units=32, lstm_layers=3, batch_size=64

R2 Score: 0.955418

Tổ hợp 9/27: lstm_units=32, lstm_layers=3, batch_size=128

R2 Score: 0.923204

Tổ hợp 10/27: lstm_units=64, lstm_layers=1, batch_size=32

R2 Score: 0.974099

*** MÔ HÌNH TỐT NHẤT MỚI! R2 = 0.974099 ***

Tổ hợp 11/27: lstm_units=64, lstm_layers=1, batch_size=64

R2 Score: 0.957098

Tổ hợp 12/27: lstm_units=64, lstm_layers=1, batch_size=128

R2 Score: 0.943086

Tổ hợp 13/27: lstm_units=64, lstm_layers=2, batch_size=32

R2 Score: 0.964460

Tổ hợp 14/27: lstm_units=64, lstm_layers=2, batch_size=64

R2 Score: 0.949370

Tổ hợp 15/27: lstm_units=64, lstm_layers=2, batch_size=128

R2 Score: 0.915198

Tổ hợp 16/27: lstm_units=64, lstm_layers=3, batch_size=32

R2 Score: 0.935000

Tổ hợp 17/27: lstm_units=64, lstm_layers=3, batch_size=64

R2 Score: 0.951271

Tổ hợp 18/27: lstm_units=64, lstm_layers=3, batch_size=128

R2 Score: 0.928408

Tổ hợp 19/27: lstm_units=128, lstm_layers=1, batch_size=32

R2 Score: 0.899152

Tổ hợp 20/27: lstm_units=128, lstm_layers=1, batch_size=64

R2 Score: 0.943836

Tổ hợp 21/27: lstm_units=128, lstm_layers=1, batch_size=128

R2 Score: 0.909145

Tổ hợp 22/27: lstm_units=128, lstm_layers=2, batch_size=32

R2 Score: 0.963803

Tổ hợp 23/27: lstm_units=128, lstm_layers=2, batch_size=64

R2 Score: 0.943623

Tổ hợp 24/27: lstm_units=128, lstm_layers=2, batch_size=128

R2 Score: 0.940233

Tổ hợp 25/27: lstm_units=128, lstm_layers=3, batch_size=32

R2 Score: 0.944837

Tổ hợp 26/27: lstm_units=128, lstm_layers=3, batch_size=64

R2 Score: 0.953956

Tổ hợp 27/27: lstm_units=128, lstm_layers=3, batch_size=128

R2 Score: 0.922759

KẾT QUẢ GRID SEARCH

Tham số tối ưu: {'lstm_units': 64, 'lstm_layers': 1, 'batch_size': 32}

R2 Score tốt nhất: 0.97

Hoàn thành! Thời gian kết thúc: 2025-06-03 19:33:55