**ĐẠI HỌC KINH TẾ TP.HỒ CHÍ MINH (UEH)**

**TRƯỜNG CÔNG NGHỆ VÀ THIẾT KẾ**

**KHOA CÔNG NGHỆ THÔNG TIN KINH DOANH**



**BÁO CÁO ĐỒ ÁN  
 TÍNH TOÁN HIỆU SUẤT CAO   
(HPC - High Performance Computing)**

**Đề tài: DỰ BÁO CHẤT LƯỢNG KHÔNG KHÍ THỜI GIAN THỰC BẰNG RNN SỬ DỤNG DOCKER**

**GVHD: TS.GVC Nguyễn Quốc Hùng**

**Nhóm thực hiện**: 08

Nguyễn Mai Hồng Trâm (Trưởng nhóm)

Trần Vọng Triển

Nguyễn Ngọc Thúy Anh

Đỗ Ngọc Phương Anh

Huỳnh Minh Phương

**TP. Hồ Chí Minh, Tháng 5/2025**

**MỤC LỤC**

Contents

[DANH MỤC HÌNH ẢNH. 4](#_Toc198978343)

[DANH MỤC BẢNG BIỂU. 5](#_Toc198978344)

[DANH MỤC TỪ VIẾT TẮT. 6](#_Toc198978345)

[Lời mở đầu. 7](#_Toc198978346)

[BẢNG PHÂN CÔNG CÁC THÀNH VIÊN 8](#_Toc198978347)

[Chương 1: GIỚI THIỆU TÍNH TOÁN HIỆU SUẤT CAO (HPC) VÀ DỰ ÁN THỰC HIỆN 9](#_Toc198978348)

[1.1. Tổng quan về tính toán hiệu suất cao 9](#_Toc198978349)

[1.2. Tổng quan về dự báo chất lượng không khí thời gian thực 10](#_Toc198978350)

[1.3. Lý do chọn đề tài 11](#_Toc198978351)

[Bảng 1. Tổng hợp các nghiên cứu đi trước 13](#_Toc198978352)

[1.4. Mục tiêu và phạm vi của dự án 15](#_Toc198978353)

[Chương 2: CƠ SỞ LÝ THUYẾT VỀ CHỦ ĐỀ NGHIÊN CỨU 16](#_Toc198978354)

[2.1. Dự báo chất lượng không khí 16](#_Toc198978355)

[*2.1.1. Nguồn thu thập dữ liệu* 16](#_Toc198978356)

[*2.1.2. Các chỉ số ô nhiễm không khí* 16](#_Toc198978357)

[2.2. Các mô hình học sâu để dự báo chuỗi thời gian 19](#_Toc198978358)

[*2.2.1. RNN* 19](#_Toc198978359)

[*2.2.2. LSTM* 20](#_Toc198978360)

[*2.2.3. GRU* 23](#_Toc198978361)

[2.3. Các công nghệ sử dụng 25](#_Toc198978362)

[*2.3.1. Docker* 25](#_Toc198978363)

[*2.3.2. Pytorch* 27](#_Toc198978364)

[*2.3.3. Fast API* 28](#_Toc198978365)

[2.4. Các chỉ số đánh giá 28](#_Toc198978366)

[*2.4.1. Mean Absolute Error - MAE* 28](#_Toc198978367)

[*2.4.2. Root Mean Square Error - RMSE* 29](#_Toc198978368)

[*2.4.3. Hệ số xác định R bình phương - R-squared* 30](#_Toc198978369)

[Chương 3: TRIỂN KHAI THỬ NGHIỆM HỆ THỐNG 32](#_Toc198978370)

[3.1. Chuẩn bị dữ liệu 32](#_Toc198978371)

[3.2. Thăm dò dữ liệu 34](#_Toc198978372)

[3.3. Thiết kế các mô hình học sâu 38](#_Toc198978373)

[3.4. Triển khai hệ thống 38](#_Toc198978374)

[3.4.2. Các bước triển khai 40](#_Toc198978375)

[Chương 4: BÀN LUẬN VÀ ĐÁNH GIÁ 44](#_Toc198978376)

[ChKẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 47](#_Toc198978377)

[TÀI LIỆU THAM KHẢO 49](#_Toc198978378)

# DANH MỤC TỪ VIẾT TẮT.

ANN (Artificial Neural Network): Mạng nơ-ron nhân tạo

API (Application Programming Interface): Giao diện lập trình ứng dụng

BiLSTM (Bidirectional Long Short-Term Memory): Mạng LSTM hai chiều

CMAQ (Community Multiscale Air Quality): Mô hình chất lượng không khí đa tỉ lệ

EDA (Exploratory Data Analysis): Phân tích dữ liệu khám phá

GEOS-Chem (Goddard Earth Observing System-Chemistry): Mô hình hóa khí quyển toàn cầu

GPU (Graphics Processing Unit): Bộ xử lý đồ họa

GRU (Gated Recurrent Unit): Mạng nơ-ron hồi tiếp có cổng điều khiển

HPC (High Performance Computing): Tính toán hiệu năng cao

LSF (Load Sharing Facility): Hệ thống chia sẻ tải, quản lý tài nguyên và lập lịch công việc

LSTM (Long Short-Term Memory): Mạng nơ-ron hồi tiếp có bộ nhớ dài ngắn

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

ORNL (Oak Ridge National Laboratory): Phòng thí nghiệm quốc gia Oak Ridge (Hoa Kỳ)

PBS Pro (Portable Batch System Professional): Hệ thống quản lý tác vụ cho siêu máy tính

PM (Particulate Matter): Bụi mịn

PSI (Pollutant Standards Index): Chỉ số tiêu chuẩn ô nhiễm

RNN (Recurrent Neural Network): Mạng nơ-ron hồi tiếp

RMSE (Root Mean Square Error): Căn bậc hai của sai số bình phương trung bình

RT-AQF (Real-Time Air Quality Forecasting): Dự báo chất lượng không khí thời gian thực

SLURM (Simple Linux Utility for Resource Management): Công cụ quản lý tài nguyên trong HPC

WHO (World Health Organization): Tổ chức Y tế Thế giới

WRF-Chem (Weather Research and Forecasting Model with Chemistry): Mô hình dự báo thời tiết tích hợp hóa học khí quyển

# Lời mở đầu.

Trong bối cảnh đô thị hóa nhanh chóng và các hoạt động công nghiệp không ngừng gia tăng, ô nhiễm không khí trở thành mối lo ngại lớn nhất, ảnh hưởng nghiêm trọng đến vấn đề sức khỏe cộng đồng và cản trở sự phát triển bền vững của xã hội. Việc giám sát và dự báo chất lượng không khí theo thời gian thực trở thành nhu cầu cấp thiết để đưa ra cảnh báo sớm và đề ra những biện pháp ứng phó kịp thời.

Sự phát triển mạnh mẽ của trí tuệ nhân tạo hiện nay, đặc biệt là các mô hình học sâu đã và đang chứng minh được hiệu quả vượt trội trong việc dự báo chất lượng không khí. Tuy nhiên, việc triển khai các mô hình này trên môi trường thực tế vẫn còn vướng phải nhiều thách thức. Docker - một nền tảng container hóa hiện đại - nổi lên như một giải pháp hiệu quả để triển khai và vận hành các hệ thống phức tạp này một cách dễ dàng và nhất quán.

Đề tài “Dự báo chất lượng không khí thời gian thực bằng Docker” được lựa chọn thực hiện để nghiên cứu và ứng dụng Docker để triển khai hệ thống dự báo chất lượng không khí một cách linh hoạt, đồng thời có thể mở rộng và tự động hóa. Điều này sẽ góp phần giải quyết được các thách thức thực tế trong việc triển khai mô hình dự báo chất lượng không khí.

# BẢNG PHÂN CÔNG CÁC THÀNH VIÊN

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Họ và tên** | **Công việc phụ trách** | **Mức độ hoàn thành** |
| **1** | Nguyễn Mai Hồng Trâm *(Trưởng nhóm)* | Code xây dựng mô hình, FastAPI.  Viết phần 3.4 | 100% |
| **2** | Nguyễn Ngọc Thúy Anh | Code EDA, Viết phần 2.4 | 100% |
| **3** | Đỗ Ngọc Phương Anh | Viết chương 1, phần 2.1 | 100% |
| **4** | Huỳnh Minh Phương | Viết 2.2, 2.3 | 100% |
| **5** | Trần Vọng Triển | Code tiền xử lý, Viết 3.2, 3.3, chương 4 | 100% |

# Chương 1: GIỚI THIỆU TÍNH TOÁN HIỆU SUẤT CAO (HPC) VÀ DỰ ÁN THỰC HIỆN

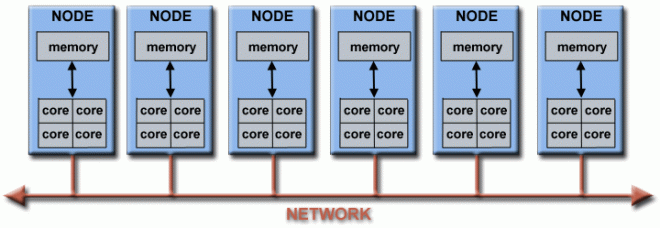
## 1.1. Tổng quan về tính toán hiệu suất cao

Tính toán hiệu suất cao (High Performance Computing - HPC) là một lĩnh vực chuyên môn liên ngành, tích hợp giữa khoa học máy tính, kỹ thuật phần cứng - phần mềm và các phương pháp luận, tập trung vào việc xây dựng và khai thác các hệ thống tính toán có khả năng xử lý đồng thời khối lượng lớn tác vụ với hiệu năng tính toán vượt trội. Các hệ thống này được thiết kế để giải quyết những bài toán quy mô dữ liệu lớn, độ phức tạp tính toán cao và đòi hỏi thời gian xử lý vượt xa khả năng của các hệ thống thông thường, đáp ứng được nhu cầu xử lý theo thời gian thực.

Hiện nay, kiến trúc hệ thống HPC được triển khai chủ yếu dưới hai dạng chính: siêu máy tính (supercomputers) và cụm máy tính hiệu năng cao (high performance clusters). Các hệ thống này đạt hiệu suất tính toán lên tới hàng tỷ tỷ mỗi giây (exaFLOPS), vượt xa giới hạn xử lý của các hệ thống máy tính truyền thống (T. Sterling và cộng sự, 2017). Một ví dụ tiêu biểu là siêu máy tính Frontier tại Phòng thí nghiệm Quốc gia Oak Ridge (ORNL), thuộc Bộ Năng lượng Hoa Kỳ, đạt hiệu suất 1.1 exaFLOPS, đánh dấu mốc quan trọng trong tiến trình hiện thực hóa kiến trúc tính toán ở cấp độ exascale (Oak Ridge National Laboratory, 2022).

Không như những hệ thống máy tính truyền thống chỉ có thể thực hiện các tác vụ tuần tự hoặc song song có giới hạn, HPC nổi bật với khả năng tính toán song song quy mô lớn, cho phép phân tách bài toán thành các tác vụ độc lập và xử lý đồng thời trên mạng lưới gồm hàng nghìn đến hàng triệu lõi xử lý (cores) (T. Sterling và cộng sự, 2017). Đây là yếu tố then chốt tạo nên ưu thế vượt trội của HPC trong việc rút ngắn đáng kể thời gian xử lý nhưng vẫn đảm bảo độ chính xác và hiệu quả cao trong việc giải quyết các bài toán phức tạp của nhiều lĩnh vực như vật lý, khí hậu, sinh học,.v.v.

Ngoài hiệu năng tính toán vượt trội, hệ thống HPC còn đặc trưng bởi kiến trúc tổ chức tài nguyên tối ưu hóa, khả năng mở rộng theo chiều ngang (horizontal scalability) và khả năng liên kết nội bộ (interconnect performance) giữa các thành phần trong hệ thống (T. Sterling và cộng sự, 2017). Kiến trúc được triển khai theo dạng song song phân tầng (hierarchical parallelism) với các tầng cơ bản bao gồm: các lõi xử lý (cores), các nút tính toán (nodes), cụm máy tính (cluster) và toàn bộ trung tâm dữ liệu (Lawrence Livermore National Laboratory, n.d.). Mỗi tầng trong hệ thống đảm nhiệm chức năng chuyên biệt với mục tiêu tối ưu hóa hiệu suất truyền thông liên tầng, đảm bảo cân bằng tải và đồng bộ hóa tiến trình nhằm duy trì tính toàn vẹn và hiệu quả của hệ thống phân tán.



#### Hình 1. Kiến trúc HPC cơ bản

*Nguồn: (Lawrence Livermore National Laboratory, n.d.)*

Bên cạnh kiến trúc phần cứng mạnh mẽ, hệ sinh thái phần mềm chuyên biệt cũng là một yếu tố quan trọng giúp hệ thống HPC hiện đại có thể phát huy tối đa hiệu quả, được thiết kế để điều phối tài nguyên, tối ưu hóa tiến trình xử lý và hỗ trợ lập trình song song hiệu quả trong môi trường tính toán phân tán. Hệ sinh thái này là trung gian quan trọng giữa phần cứng và ứng dụng, đảm bảo hệ thống vận hành đạt được hiệu suất cao, có thể mở rộng linh hoạt và ổn định lâu dài. Các thành phần cốt lõi của hệ sinh thái phần mềm HPC bao gồm: Hệ điều hành hiệu năng cao như CentOS, Rocky Linux hay SUSE HPC đóng vai trò giảm độ thiểu độ trễ I/O, quản lý tài nguyên chi tiết và hỗ trợ kiến trúc tính toán song song, trình lập lịch (Job Scheduler) như SLURM, PBS P ro hay LSF giữ nhiệm vụ quản lý hàng nghìn tiến trình thực thi đồng thời, hỗ trợ phân phối tải tối ưu và đảm bảo tính công bằng trong việc sử dụng tài nguyên hệ thống (Andy B. Yoo và cộng sự, 2003), hệ thống tập tin song song như Lustre, IBM Spectrum Scale (GPFS) hay BeeGFS được sử dụng để cung cấp thông lượng I/O cao, giảm thiểu tắc nghẽn và hỗ trợ truy cập đồng thời với quy mô hàng nghìn node tính toán (Peter Braam, 2019) và môi trường lập trình song song với các API như MPI, OpenMP, CUDA hay OpenACC cho phép các ứng dụng song song được phát triển hiệu quả, cũng như tận dụng tối đa đặc trưng song song của phần cứng HPC (William Gropp và cộng sự, 1999).

Tổng quan có thể thấy, nhờ vào sự tích hợp đồng bộ giữa phần cứng và phần mềm, HPC đã và đang dần trở thành một nền tảng hạ tầng chiến lược trong kỷ nguyên phát triển của trí tuệ nhân tạo và dữ liệu lớn, đóng vai trò tiên phong trong việc hiện thực hóa các hệ thống yêu cầu khả năng tính toán quy mô hàng tỷ tham số, xử lý theo thời gian thực và truy xuất dữ liệu với tốc độ cao trong nhiều lĩnh vực mũi nhọn.

## 1.2. Tổng quan về dự báo chất lượng không khí thời gian thực

Dự báo chất lượng không khí theo thời gian thực (Real-time Air Quality Forecasting - RT-AQF) là một lĩnh vực trọng yếu trong khoa học khí quyển hiện đại, đại diện cho sự phát triển mang tính ứng dụng thực tiễn cao của khoa học và kỹ thuật. Nhiệm vụ chính của RT-AQF là cung cấp kịp thời thông tin về các vấn đề ô nhiễm không khí để có thể đưa ra phương hướng bảo vệ sức khỏe cộng đồng và thúc đẩy việc hoạch định chính sách môi trường hiệu quả (Yang Zhang và cộng sự, 2012).

Trải qua nhiều thập kỷ, lĩnh vực dự báo chất lượng không khí theo thời gian thực đã chứng kiến sự thay đổi mạnh mẽ, từ các phương pháp thực nghiệm và thống kê đơn giản sang các mô hình mô phỏng vật lý - hóa học bán thực nghiệm, và tiên tiến hơn là các mô hình tích hợp khí tượng - hóa học hiện đại. Các mô hình này, tiêu biểu như WRF-Chem, CMAQ, GEOS-Chem, đã trở thành “xương sống” cho các hệ thống RT-AQF hiện đại, nhờ vào khả năng mô phỏng đồng thời và tương tác hai chiều giữa điều kiện khí tượng và quá trình biến đổi hóa học của các chất ô nhiễm trong môi trường không khí (Yang Zhang và cộng sự, 2012).

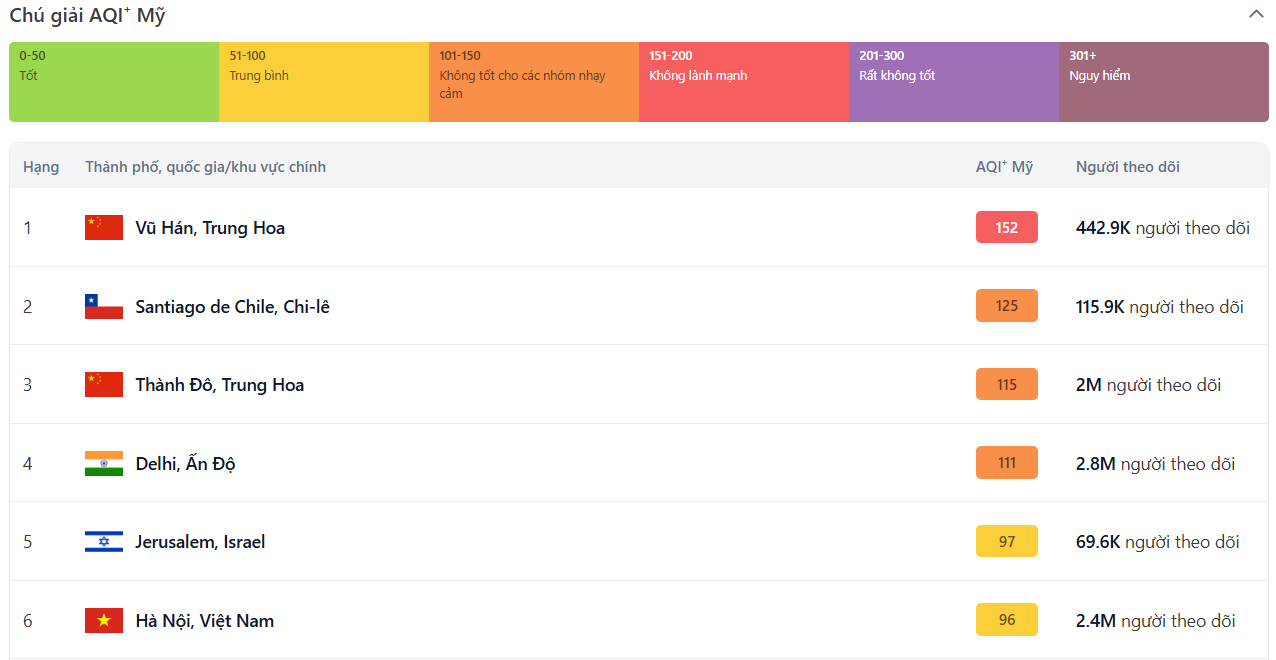
Sự chuyển mình này không chỉ là sự phát triển về mặt công nghệ kỹ thuật mà còn phản ánh xu hướng lớn trong khoa học khí quyển - sự tích hợp liên ngành giữa khí tượng học, hóa học khí quyển, mô hình hóa số và tính toán hiệu suất cao. Nhờ đó, RT-AQF ngày nay đang dần thu hẹp khoảng cách đến mục tiêu mô phỏng chính xác và toàn diện nhất trạng thái của không khí trong môi trường theo thời gian thực, cả về mặt không gian vi mô lẫn khu vực quy mô toàn cầu.

Trong bối cảnh ô nhiễm không khí đang ở mức báo động toàn cầu, dự báo chất lượng không khí theo thời gian thực RT-AQF ngày càng được nhìn nhận như một nền tảng hạ tầng thiết yếu để ra quyết định chiến lược quản trị môi trường, hỗ trợ triển khai các chính sách không khí sạch và hiện thực hóa quyền được sống trong môi trường không ô nhiễm - một trong những mục tiêu Phát triển Bền vững cốt lõi vẫn chưa thực sự giải quyết được triệt để - của Liên Hợp Quốc.

## 1.3. Lý do chọn đề tài

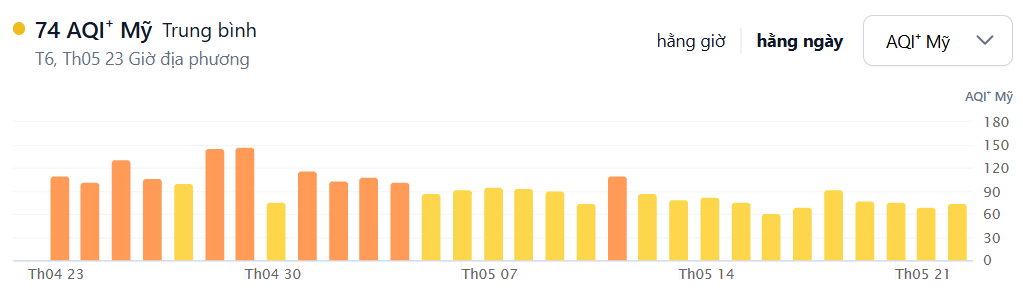
Ô nhiễm không khí đã là một trong những vấn đề nghiêm trọng trong nhiều thập kỷ và đến nay vẫn là một trong những mối đe dọa hàng đầu lên sức khỏe cộng đồng trên phạm vi toàn cầu. Theo Tổ chức Y tế Thế giới (WHO), ước tính mỗi năm có khoảng 7 triệu ca tử vong sớm do phơi nhiễm ô nhiễm không khí. Tại Việt Nam, tình trạng suy giảm chất lượng không khí là một vấn đề môi trường vô cùng cấp bách, nhất là ở những đô thị lớn như TP. Hồ Chí Minh và Hà Nội - nơi có mật độ dân cư cao, tốc độ đô thị hóa nhanh và hoạt động kinh tế - giao thông xảy ra với cường độ lớn.

Đặc biệt là ở Hà Nội - thủ đô Việt Nam, dù đã được cải thiện nhiều năm liên tiếp, nhưng nồng độ PM2.5 - một dạng chất hạt bụi mịn trong không khí có tác động xấu đến sức khỏe cộng đồng - vẫn cao hơn rất nhiều so với mục tiêu tiếp xúc mà WHO đề ra (10 µg/m³), cụ thể là trong năm 2020, mức ô nhiễm không khí ở thủ đô vẫn cao gấp 4 lần so với tiêu chuẩn. Theo số liệu xếp hạng trực tiếp từ IQAir, mức độ ô nhiễm không khí ở Hà Nội được ghi nhận đạt hạng 6 trong danh sách các thành phố ô nhiễm nhất. Cũng theo dữ liệu trên IQAir, có thể thấy chất lượng không khí thu được mỗi ngày ở Hà Nội luôn nằm ở mức trung bình đến mức không tốt cho nhóm nhạy cảm. Thông tin IQAir được truy cập vào ngày 22 tháng 5 năm 2025.



#### Hình 2. Xếp hạng trực tiếp thành phố lớn ô nhiễm nhất

*Nguồn: IQAir [[ref1]](https://www.iqair.com/vi/world-air-quality-ranking)*



#### Hình 3. Biểu đồ chất lượng không khí lịch sử cho Hà Nội

*Nguồn: (IQAir)*

Điều này đánh lên một hồi chuông cảnh báo mạnh mẽ về sự cấp thiết trong việc dự báo sớm chất lượng không khí trong môi trường, đặc biệt là ở những khu đô thị lớn như Hà Nội - nơi có mật độ phơi nhiễm đáng báo động. Việc dự báo sớm không chỉ để có thể chủ động ứng phó mà còn là bước đệm đầu cho những nhà hoạt động chính sách phát triển hiệu quả các chính sách kiểm soát phát thải, từ đó thúc đẩy để quy hoạch đô thị theo hướng phát triển bền vững.

Bài toán dự báo chất lượng không khí đã nhận được sự quan tâm rộng rãi từ cả cộng đồng nghiên cứu quốc tế lẫn trong nước, các nghiên cứu đã và đang khai thác nhiều mô hình với kiến trúc học máy và học sâu phức tạp hơn. Bảng dưới đây tổng hợp một số công trình nghiên cứu tiêu biểu có liên quan đến dự báo chất lượng không khí bằng các mô hình học máy và học sâu.

### Bảng 1. Tổng hợp các nghiên cứu đi trước

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Tác giả** | **Dữ liệu sử dụng** | **Phương pháp/ Mô hình** | **Kết quả chính** | **Mức độ liên quan đến nghiên cứu** |
| (Dang Thi Khanh Linh & Vu Van Huan, 2024) | Dữ liệu 6 chất ô nhiễm (PM2.5, PM10, NO2, O3, SO2, CO) được thu thập từ nhiều trạm quan trắc tại Hà Nội  từ ngày 5/7/2016 - 1/7/2020 | Mô hình Multi-step LSTM | Đạt độ chính xác cao, giá trị gần sát thực tế thu được từ các trạm quan trắc | Cùng sử dụng dữ liệu ô nhiễm không khí tại Hà Nội. |
| (Trang Pham và cộng sự, 2023) | Dữ liệu 6 chất ô nhiễm (PM2.5, PM10, NO2, O3, SO2, CO) được thu thập từ nhiều trạm quan trắc tại Hà Nội  từ ngày 1/1/2018 - 10/5/2022 | Mô hình LSTM, BiLSTM và mô hình lai Encoder LSTM | Mô hình đạt độ chính xác cao (hệ số tương quan > 0.95). Thời gian huấn luyện ngắn, sử dụng ít đặc trưng, vượt trội hơn so với các phương pháp khác. | Cùng sử dụng dữ liệu ô nhiễm không khí tại Hà Nội. |
| (Yi-Ting Tsai và cộng sự, 2018) | Dữ liệu 20 đặc trưng được lấy từ EPA của Đài Loan, từ năm 2012 đến năm 2016.  Dữ liệu kiểm thử là của  năm 2017 | Sử dụng Keras trên TensorFlow để xây dựng mô hình RNN với LSTM | Kết quả cho thấy phương pháp đề xuất có thể dự báo hiệu quả giá trị PM2.5 | Sử dụng mô hình RNN và mô hình LSTM |
| (Sagar V. Belavadi và cộng sự, 2020) | Dữ liệu mạng cảm biến không dây và nguồn dữ liệu mở từ Chính phủ Ấn Độ | Mô hình RNN với LSTM | Mức độ biến động tăng lên, hiệu suất mô hình bị suy giảm | Sử dụng mô hình RNN và mô hình LSTM |
| (Quý Minh & Hải Hồng, 2024) | Dữ liệu 7 chỉ số (PM1, PM10, PM2.5, SO2, O3, CO, NO2) được thu thập từ 18 trạm quan trắc ở Bắc Ninh. Thời gian chia thành 2 giai đoạn:  2000 - 20022 để huấn luyện, từ 2022 để kiểm thử | Mô hình ANN, RNN và LSTM | Mô hình LSTM có độ chính xác dự báo cao nhất với 5 chất ô nhiễm. | Dữ liệu thu thập được từ trạm quan trắc.  Có sử dụng mô hình RNN và mô hình LSTM, LSTM có độ chính xác dự báo tốt nhất |

Các nghiên cứu được liệt kê đã cho thấy tiềm năng lớn dự báo chất lượng không khí ở những khu đô thị lớn như Hà Nội bằng cách ứng dụng các mô hình học sâu như LSTM, BiLSTM, mô hình lai Encoder - LSTM và RNN. Các kết quả thu được mang lại tín hiệu khả quan trong việc dự báo các chất ô nhiễm trong không khí với độ chính xác cao.

Tuy nhiên, các nghiên cứu này tập trung vào dữ liệu từ nhiều trạm quan trắc, rất ít nghiên cứu khai thác chi tiết dữ liệu từ một trạm quan trắc duy nhất - điều này đặc biệt có ý nghĩa, bởi trong thực tế, nhiều địa phương hạn chế về điều kiện để có thể triển khai mạng lưới quan trắc rộng rãi, nên việc xây dựng một mô hình hiệu quả từ một trạm đơn lẻ là vô cùng cần thiết. Bên cạnh đó, hầu hết đều chỉ mới dừng lại ở việc xây dựng mô hình mà chưa triển khai thực tế dưới dạng hệ thống, cũng chưa có nghiên cứu tích hợp mô hình vào môi trường như Docker để có thể phát triển nhu cầu dự báo theo thời gian thực. Sự thiếu vắng những kết quả đánh giá hiệu quả khi triển khai mô hình học sâu trong hạ tầng Docker và tính toán hiệu suất cao là một khoảng trống lớn cần được chú ý nhiều hơn.

Chính vì vậy, nhóm nghiên cứu lựa chọn đề tài “Triển khai hệ thống trên Docker kết hợp với các mô hình học sâu để dự báo chất lượng không khí tại Hà Nội dựa trên dữ liệu từ một trạm quan trắc” để xây dựng một hệ thống có thể triển khai thực tế nhưng vẫn đạt độ chính xác cao. Nghiên cứu này không chỉ đánh giá khả năng dự báo của mô hình học sâu trên một trạm dữ liệu đơn lẻ mà còn cung cấp giải pháp công nghệ có thể mở rộng và tích hợp với các hệ thống dự báo sớm.

## 1.4. Mục tiêu và phạm vi của dự án

Mục tiêu của đề tài là ứng dụng mạng neural hồi tiếp RNN (Recurrent Neural Network) để dự báo chất lượng không khí AQI theo thời gian thực với độ chính xác cao.

Mục tiêu cụ thể bao gồm:

* Thu thập và xử lý dữ liệu thu từ nguồn dữ liệu mở của OpenAQ thông qua OpenAQ API và OpenAQ Python SDK.
* Thiết kế và huấn luyện mô hình RNN để dự báo chất lượng không khí.
* Đánh giá độ chính xác của mô hình dựa trên các chỉ số thống kê dành cho dự báo chất lượng không khí như MAE, RMSE, R2.

Phạm vi của nghiên cứu được giới hạn như sau:

* Dữ liệu giới hạn ở các chỉ số ô nhiễm bao gồm: PM2.5, PM10, CO, NO2 với độ phân giải thời gian theo giờ.
* Không gian dữ liệu được thu thập ở một trạm quan trắc duy nhất, có dữ liệu chất lượng không khí của Hà Nội.

# Chương 2: CƠ SỞ LÝ THUYẾT VỀ CHỦ ĐỀ NGHIÊN CỨU

## 2.1. Dự báo chất lượng không khí

### *2.1.1. Nguồn thu thập dữ liệu*

OpenAQ là một tổ chức phi lợi nhuận trong lĩnh vực công nghệ môi trường, chuyên thu thập và chuẩn hóa dữ liệu chất lượng không khí từ khắp nơi trên thế giới - từ dữ liệu của cơ quan chính phủ, trạm quan trắc độc lập đến các tổ chức nghiên cứu - lên một nền tảng dữ liệu mã nguồn mở cho phép truy cập tự do. Bất kỳ ai quan tâm đến dữ liệu chất lượng không khí đều có thể tiếp cận một cách dễ dàng.

OpenAQ API cho phép truy cập mở vào dữ liệu đo lường chất lượng không khí trên toàn cầu theo thời gian thực và cả dữ liệu lịch sử với các thông số phổ biến như: bụi mịn PM2.5 và PM10, các khí gây ô nhiễm NO2, CO, SO2, O3, cũng như các chỉ số khác như thời gian, vị trí địa lý, v.v. và phản hồi kết quả thành một file định dạng JSON. Ở một số khu vực nhất định, API của OpenAQ có thể cung cấp thêm dữ liệu khác như PM1, PM4, CO2, NO, NOx, CH4, UFP.

OpenAQ Python SDK là một bộ thư viện Python được phát triển để có thể dễ dàng gọi API của OpenAQ và truy cập, thu thập và xử lý dữ liệu chất lượng không khí. Có thể nói, Python SDK là một giao diện thân thiện, giúp gọi API một cách thuận tiện mà không cần phải thao tác với các yêu cầu HTTP phức tạp.

### *2.1.2. Các chỉ số ô nhiễm không khí*

Để đánh giá mức độ suy giảm chất lượng môi trường không khí, việc đo lường nồng độ các chất ô nhiễm trong không khí là vô cùng quan trọng. Những kết quả này là những chỉ số quan trọng để phản ánh thực trạng ô nhiễm và giúp theo dõi biến động chất lượng không khí, một số chỉ số quan trọng nhất có thể nhắc đến bao gồm: PM2.5, PM10, CO, NO2 và chỉ số AQI.

Bụi phân tử (Particulate Matter - PM), có mặt trong môi trường đô thị và phi đô thị, là hỗn hợp có thành phần và đặc tính hóa học vật lý đa dạng phức tạp. Các tổn thương sức khỏe do PM gây ra có sự khác nhau tùy thuộc vào kích thước, cũng như các đặc tính vật lý, thành phần hóa học và nguồn gốc của các hạt bụi. Tuy nhiên, các hạt bụi thường không phân loại theo những đặc tính kể trên mà được phân loại dựa trên đặc tính khí động học, một chỉ số tổng hợp, biểu thị kích thước hạt với đường kính khí động học tương ứng với kích thước hạt. Hai loại bụi phân tử được quan tâm nhiều nhất trong vài thập kỷ gần đây là PM2.5 và PM10.

* PM2.5 là những hạt bụi phân tử có đường kính khí động học nhỏ hơn hoặc bằng 2.5 µm. Do kích thước siêu nhỏ, những hạt này có khả năng xâm nhập vào vùng phế nang và hệ tuần hoàn máu, gây ra những bệnh lý nghiêm trọng ở người. Các nghiên cứu dịch tễ học đã chứng minh rằng, tiếp xúc lâu dài với PM2.5 có thể dẫn đến nguy cơ gia tăng mắc các bệnh như các vấn đề hô hấp mạn tính, ung thư phổi, bệnh tim mạch cũng như đột quỵ.
* PM10 là những hạt bụi phân tử có đường kính khí động học nhỏ hơn hoặc bằng 10 µm, trong đó bao gồm cả PM2.5. Do không thể đi sâu vào như PM2.5, phần lớn các hạt PM10 lắng đọng ở những phần trên của hệ hô hấp, gây nên kích ứng và các vấn đề hô hấp như ho, hắt hơi, viêm xoang.

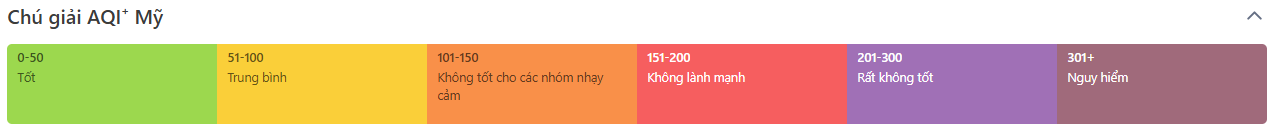
Trong “Hướng dẫn Chất lượng không khí Toàn cầu” 2021, Tổ chức Y tế Thế giới (WHO) đã thiết lập các ngưỡng khuyến nghị nghiêm ngặt, được xây dựng dựa trên những bằng chứng khoa học. Cụ thể, PM2.5 có ngưỡng trung bình năm khuyến nghị nhỏ hơn 5 µg/m3 và giá trị trung bình trong 24 giờ không vượt quá 15 µg/m3. Còn đối với PM10, ngưỡng trung bình năm khuyến nghị là không vượt quá 15 µg/m3 và trung bình 24 giờ không vượt 45 µg/m3. Mức khuyến nghị này là cần thiết để đảm bảo sức khỏe cộng đồng không bị ảnh hưởng quá nhiều bởi các hạt bụi mịn trong không khí, đồng thời cũng là chiến lược y tế thiết yếu giúp giảm thiểu các vấn đề bệnh tật liên quan đến ô nhiễm không khí.

Bên cạnh các hạt bụi mịn, carbon monoxide (CO) - một loại khí độc hại không màu, không mùi, không vị và không gây kích ứng, được sinh ra từ quá trình đốt cháy các nhiên liệu chứa carbon - cũng gây những tác động tiêu cực đến sức khỏe cộng đồng. Theo Tổ chức Y tế Thế giới (WHO), trong “Hướng dẫn Chất lượng không khí Toàn cầu”, CO khi kết hợp với các hemoglobin trong cơ thể người, hình thành nên carboxyhemoglobin (COHb), sẽ làm suy giảm khả năng vận chuyển oxy của máu. Phơi nhiễm CO lâu dài có thể gây ra hàng loạt các vấn đề sức khỏe ở hệ thần kinh (chóng mặt, đau đầu, buồn nôn, lú lẫn, mất ý thức, hôn mê), hệ tim mạch (đau ngực, hạ huyết áp, rối loạn nhịp tim), hệ hô hấp (khó thở, thở nhanh, suy hô hấp) và tử vong. Vì thế, để đảm bảo sức khỏe cộng đồng, WHO khuyến nghị giới hạn của nồng độ CO trung bình trong 24 giờ không được vượt quá 4 mg/m3.

NO2 (Nitrogen dioxide) cũng là một chất trong không khí rất được chú tâm bởi những ảnh hưởng tiêu cực lên sức khỏe cộng đồng. NO2 là một loại khí màu nâu đỏ có mùi hắc đặc trưng, đây là một chất oxy hóa mạnh, khi tiếp xúc với nước sẽ tạo thành acid nitric - thành phần chính trong mưa acid. Ngoài những tác động trực tiếp lên sức khỏe, NO2 còn tham gia vào quá trình quang hóa khí quyển, đóng vai trò quan trọng trong sự hình thành tầng đối lưu của ozone và các hạt bụi mịn PM2.5, PM10, điều này gây ra những vấn đề ảnh hưởng trực tiếp đến biến đổi khí hậu và giảm tầm nhìn trong môi trường. Vì những lý do trên, NO2 có thể xem là tiền chất quan trọng cho sự hình thành thứ cấp của nhiều chất ô nhiễm có ảnh hưởng đến sức khỏe con người.

Việc đánh giá và giám sát nồng độ các chất ô nhiễm trong không khí được nêu trên là vô cùng cần thiết để bảo vệ sức khỏe cộng đồng. Chỉ số chất lượng không khí (Air Quality Index - AQI) là một công cụ được sử dụng phổ biến để có thể cung cấp một thang điểm trực quan về dễ hiểu nhất, phản ánh nồng độ phức tạp của các chất ô nhiễm trong không khí. Được giới thiệu lần đầu tiên vào năm 1976 tại Hoa Kỳ, với tên gọi là chỉ số tiêu chuẩn ô nhiễm (Pollution Standard Index - PSI), AQI ban đầu được tính toán dựa trên giá trị cao nhất của 5 chất gây ô nhiễm chính: PM10, O3, SO2, CO và NO2. Sau này, phương pháp NowCast được phát triển để khắc phục nhược điểm của việc lấy trung bình AQI trong 24 giờ. Phương pháp này tạo sự cân bằng giữa các rủi ro tiếp xúc ngắn hạn và dài hạn, cũng như loại bỏ lỗi phát sinh do sự thay đổi nhanh chóng nồng độ ô nhiễm không khí, thường xuất phát từ các sự kiện như cháy rừng hoặc gió mạnh. Công thức tính chỉ số AQI như sau:

Trong đó, là chỉ số của chất ô nhiễm p, là nồng độ của chất ô nhiễm p, là ngưỡng nồng độ cao hơn gần nhất, lớn hơn hoặc bằng , là ngưỡng nồng độ thấp hơn gần nhất, nhỏ hơn hoặc bằng , là giá trị AQI tương ứng với và là giá trị AQI tương ứng với . Chỉ số sau cùng được làm tròn đến số nguyên gần nhất và giá trị lớn nhất của các chỉ số trong số các chất ô nhiễm được báo cáo là AQI tổng hợp. Trên thang điểm từ 0 - 500, các nhóm mức độ được chia thành: tốt, trung bình, không tốt cho nhóm nhạy cảm, không lành mạnh, rất không tốt và nguy hiểm.



#### Hình 4. Các nhóm mức độ AQI

*Nguồn: IQAir*

## 2.2. Các mô hình học sâu để dự báo chuỗi thời gian

### *2.2.1. RNN*

Mạng nơ-ron hồi quy (Recurrent Neural Network - RNN) là một kiến trúc học sâu được thiết kế để xử lý dữ liệu dạng chuỗi như văn bản, tín hiệu cảm biến, dữ liệu thời gian thực. Không giống như các mạng nơ-ron truyền thống vốn xử lý dữ liệu ở từng node độc lập, RNN lại có khả năng lưu giữ dữ liệu theo thời gian thông qua trạng thái ẩn được duy trì và cập nhật liên tục ở mỗi node, cho phép mô hình học được các mối quan hệ phụ thuộc trong chuỗi dữ liệu (Filippo Maria Bianchi và cộng sự, 2017).

Về mặt toán học, quá trình RNN cập nhật trạng thái được mô tả theo công thức:

,

,

Trong đó:

* : Trạng thái ẩn tại thời điểm t, chứa thông tin được “ghi nhớ” của mạng nơ-ron tại thời điểm đó.
* (⋅): Hàm kích hoạt phi tuyến cho trạng thái ẩn, thường là sigmoid hay tanh.
* : Đầu vào tại thời điểm t
* : Trạng thái ẩn từ khoảng thời gian trước.
* : Trọng số từ input-to-hidden và hidden-to-hidden.
* : Vectơ bias tương ứng cho input và hidden layer.
* : Đầu ra tại thời điểm t, tính bằng phép biến đổi (⋅) trên tổ hợp tuyến tính của trạng thái ẩn và trọng số đầu ra.
* : Trọng số từ hidden-to-output, kích thước .
* : Vectơ hiệu số điều chỉnh (bias) đầu ra.

Công thức này thể hiện cơ chế hoạt động của RNN cổ điển, được trình bày trong các giáo trình kinh điển về học sâu như “Recurrent Neural Networks Architectures” của (Filippo Maria Bianchi và cộng sự, 2017)

Quá trình huấn luyện RNN sử dụng thuật toán lan truyền ngược qua thời gian (Backpropagation Through Time - BPTT) nhằm tối ưu các trọng số bằng cách tính gradient toàn bộ chuỗi thời gian. Tuy nhiên, mô hình RNN cơ bản thường gặp hiện tương mất biến hoặc bùng nổ gradient khi chuỗi dữ liệu quá dài, khiến cho mo hình không thể học hiệu quả. Để giải quyết vấn đề này, các biến thể của RNN như LSTM và GRU ra đời, tích hợp thêm các cổng điều khiển (gating mechanisms) để kiếm soát dòng thông tin, giúp mô hình “ghi nhớ” được các mối quan hệ dài hạn.

Trong thực tiễn, RNN được ứng dụng rộng rãi trong hệ thống tính toán hiệu suất cao để xử lý dữ liệu thời gian thực. Ví dụ như nghiên cứu của (Ji Seo & Kyoung-Dae Kim, 2025) xây dựng mô hình kết hợp giữa AMES và RNN có thể làm mượt số mũ trong mô hình. Bên cạnh đó, RNN có thể nâng cao khả năng dự báo dữ liệu lái xe với đặc trưng tích lũy và không tuần hoàn, qua đó cho thấy được tiềm năng của RNN trong các hệ thống xử lý các chuỗi dữ liệu phức tạp có quy mô lớn và độ trễ thấp.

### *2.2.2. LSTM*

Để khắc phục khó khăn khi xử lý chuỗi dữ liệu có chiều dài quá lớn thì (Sepp Hochreiter & Jürgen Schmidhuber, 1997) đã giới thiệu kiến trúc LSTM được thiết kế để mô hình có thể học và ghi nhớ thông tin lâu hơn qua cơ chế các cổng kiểm soát thông tin. Kiến trúc của một tế bào (cell) trong LSTM được cấu tạo từ ba cổng kiểm soát chính như cổng quên (forget gate), cổng cập nhật và cổng ra, cùng với hai hàm kích hoạt như (⋅) và (⋅). Cấu trúc này cho phép thông tin sau khi được chọn lọc và lưu trữ hoặc loại bỏ thông qua từng phần từ (Hadamard product) nhằm đảm bảo luồng gradient ổn định theo thời gian và tăng cường khả năng ghi nhớ dài hạn.

Cụ thể, LSTM cập nhật trạng thái với các cổng sau:

* Cổng quên (forget gate):

,

Cổng quên nhằm loại bỏ các thông tin không cần thiết từ các trạng thái trước.

* Trạng thái đề xuất (candidate gate):

,

* Cổng cập nhật (hay còn biết tới là cổng vào - input gate)

,

* Trạng thái cell (cell state):

,

* Cổng đầu ra (output gate):

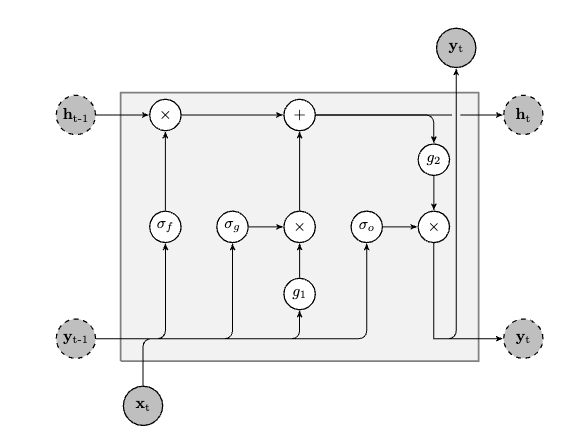
,

* Đầu ra (output):

.

Trong đó:

* : Đầu vào tại thời điểm t
* : Đầu ra tại cell tại khoảng thời gian
* : Trạng thái đề xuất tại thời điểm t, được tính từ (;) sau khi qua hàm phi tuyến (⋅); đây là thông tin “mới” mà cell sẽ xem xét để cập nhật trạng thái tiếp theo.
* : Trạng thái cell tại thời điểm t, phán ánh được bộ nhớ ngắn hạn được duy trì và cập nhật qua thời gian.
* : Đầu ra của các hàm sigmoid tại ba cổng
* tại cổng quên quyết định phần thông tin nào từ trạng thái trước cần được loại bỏ.
* tại cổng cập nhật (cổng vào) kiểm soát mức độ lưu trữ thông tin từ .
* tại cổng đầu ra điều chỉnh lại lượng thông tin từ trạng thái .
* , , , : các ma trận trọng số đầu vào kết nối đến các cổng và trạng thái đề xuất.
* , , , : các ma trận trọng số đầu vào kết nối đến các cổng và trạng thái đề xuất.
* , , , : các vectơ hiệu số điều chỉnh (bias) tương ứng.
* (⋅) ,(⋅): các hàm kích hoạt phi tuyến theo từng phần tử, thường là hàm tanh, giúp nén giá trị về khoảng [-1;1].
* : phép nhân từng phần tử giữa hai vectơ (Hadamart product)



Các vòng tròn xám đậm có viền liền thể hiện các biến trao đổi trực tiếp với đầu vào và đầu ra, trong khi các vòng tròn xám viền đứt biểu diễn trạng thái ẩn được truyền qua các bước thời gian. Các phép biến đổi phi tuyến và thường dùng hàm tanh. Các vòng tròn trắng chứa (+) và (x) biểu diễn các phép toán tuyến tính, trong khi đó là các hàm sigmoid tương ứng cho các cổng

#### Hình 5. Hình minh họa một cell trong kiến trúc LSTM

*Nguồn: (Filippo Maria Bianchi và cộng sự, 2017)*

Trong thực tế, các nghiên cứu gần đây như nghiên cứu của (Jerry Teleron, 2025) cho thấy LSTM khi kết hợp với CNN có khả năng cải thiện độ chính xác đáng kể trong việc dự báo dữ liệu chuỗi thời gian thực như nhu cầu năng lượng điện.

### *2.2.3. GRU*

Đơn vị hồi quy có cổng (Gated Recurrent Unit - GRU) là một phiên bản thu gọn của LSTM, được đề xuất bởi (Kyunghyun Cho và cộng sự, 2014) để tăng tốc độ huẩn luyện mô hình và giảm số lượng tham số mà vẫn giữa được hiệu suất cao trong việc học các chuỗi dữ liệu dài. Theo nghiên cứu của (Filippo Maria Bianchi và cộng sự, 2017), mô hình này tích hợp thêm cơ chế kiếm soát thông tin với hai cổng chính: cổng đặt lại (reset gate) và cổng cập nhật (update gate) hoặc được biết là đầu vào, thay vì ba cổng như LSTM. Nhờ cấu trúc nhỏ gọn hơn nên GRU sẽ hoạt động hiệu quả hơn trong các bài toán xử lý thời gian thực hoặc tránh được hiện tượng quá khớp mà vẫn duy trì được khả năng ghi nhớ các quan hệ dài trong chuỗi (Filippo Maria Bianchi và cộng sự, 2017).

Khác với LSTM, Gru không sử dụng trạng thái cell riêng biệt mà cập nhật trực tiếp lên trạng thái ẩn . Cơ chế của GRU có thể được mô tả theo chuỗi công thức sau:

* Cổng đặt lại (reset gate):

,

* Trạng thái tạm thời (current state):

,

* Trạng thái đề xuất (candidate state):

,

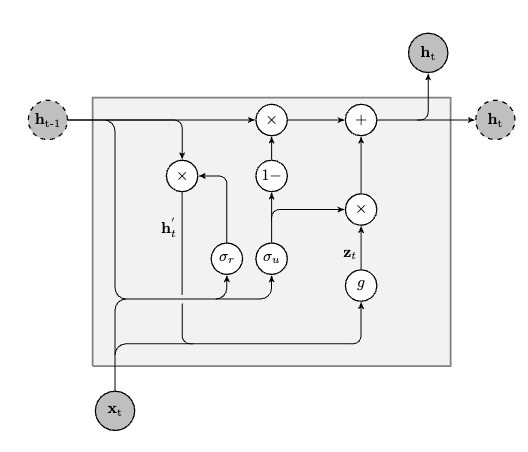
* Cổng cập nhật (update gate)

,

* Trạng thái mới (new state):

.

Trong đó:

* : Đầu vào tại thời điểm t.
* : Trạng thái ẩn từ khoảng thời gian trước
* : Trạng thái cell tại thời điểm t, phản ánh được bộ nhớ ngắn hạn được duy trì và cập nhật qua thời gian.
* , : Đầu ra của các reset gate và update gate.
* : Trạng thái đề xuất
* (⋅): Hàm kích hoạt sigmoid
* (⋅): Hàm phi tuyến, thường dùng hàm tanh.
* : phép nhân từng phần tử giữa hai vectơ (Hadamart product)
* , , : các ma trận trọng số hình chữ nhật từ đầu vào kết nối đến các cổng và trạng thái đề xuất.
* , , : các ma trận trọng số vuông từ đầu vào kết nối hồi quy từ .
* , , : các vectơ hiệu số điều chỉnh (bias) tương ứng.

Hình 6. Hình minh họa một cell trong kiến trúc GRU

*Nguồn: (Filippo Maria Bianchi và cộng sự, 2017)*

*Các vòng tròn xám đậm có viền liền thể hiện các biến trao đổi trực tiếp với đầu vào và đầu ra của mạng, trong khi các vòng tròn xám viền đứt biểu diễn trạng thái ẩn được truyền giữa các cell nội bộ trong lớp ẩn. Các phép biến đổi phi tuyến g(⋅) thường dùng hàm tanh. Các vòng tròn trắng chứa (+) và (x) biểu diễn các phép toán tuyến tính, trong khi đó r và u là các hàm sigmoid tương ứng cho các cổng.*

## 2.3. Các công nghệ sử dụng

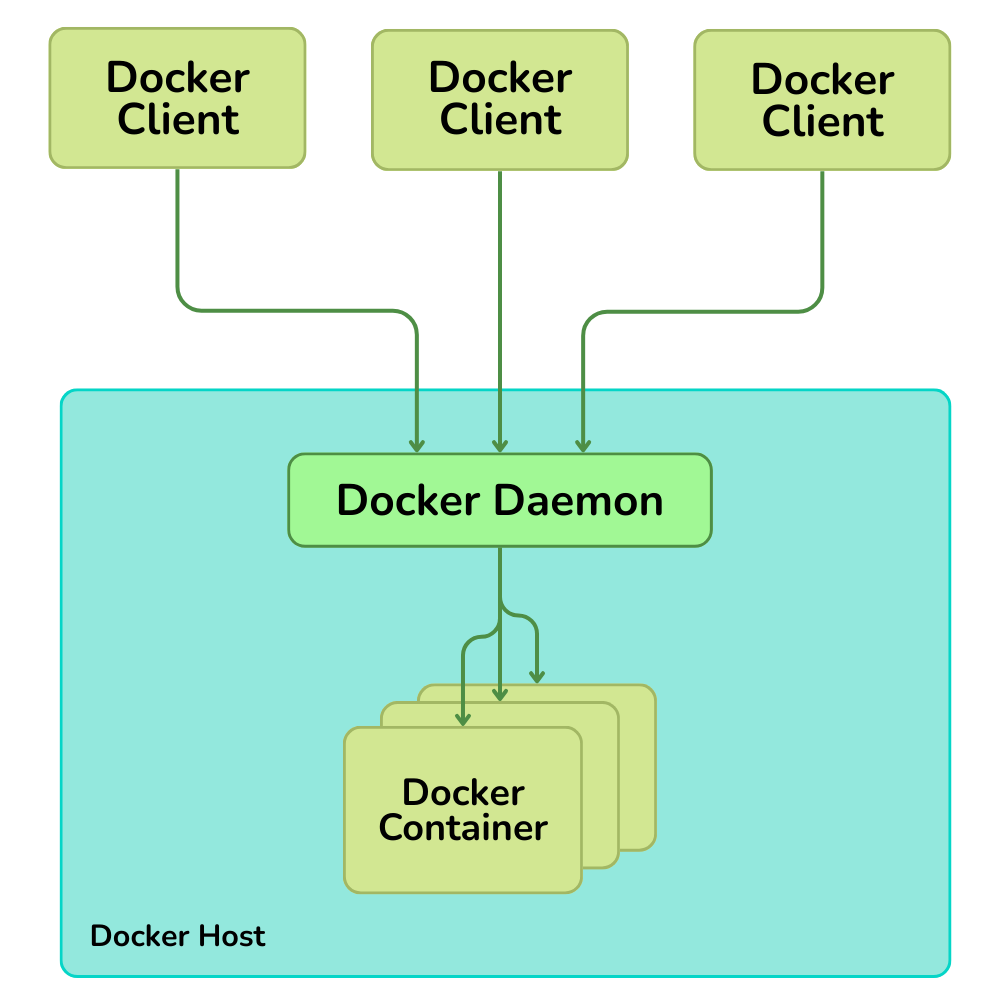
### *2.3.1. Docker*

Docker là một nền tảng container hóa mã nguồn mở cho phép đóng gói, phân phối và triển khai các ứng dụng trong các môi trường biệt lập linh hoạt và hiệu quả. Thay vì sử dụng nhiều máy ảo truyền thống nặng nề, Docker tận dụng công nghệ container để chia sẻ cùng một kernel hệ điều hành nhưng vẫn giữ được mức độ phân vùng cao, giúp tối ưu hóa hiệu năng và giảm tải tài nguyên hệ thống (Babak Bashari Rad và cộng sự, 2017).

Theo (Babak Bashari Rad và cộng sự, 2017), Docker gồm bốn thành phần bên trong quan trọng:

* Docker Client và Server được xem là cơ chế giao tiếp chính giữa người dùng và hệ thống Docker Engine, nơi xử lý các yêu cầu chính như build, run và quản lý các container.
* Docker Images là các mẫu hệ thống bất biến chứa đựng tất cả các thành phần cần thiết để khởi chạy ứng dụng.
* Docker Registries có vai trò là kho lưu trữ các image, một ví dụ điển hình là Docker Hub.
* Docker Containers là các thực thể được khởi tạo từ các image, có tính di động cao và triển khai đồng thời trên bất kỳ hệ thống nào có hỗ trợ Docker.

Trong bối cảnh sử dụng điện toán hiệu năng cao (HPC), Docker được sử dụng để đơn giản hóa quá trình thiết lập môi trường thực thi, đồng thời hỗ trợ việc tái sử dụng cấu hình trên nhiều node tính toán. Tuy vậy, (Theo Combe và cộng sự, 2016) đưa ra lời cảnh bóa về việc chia sẻ kernel giữa các container cũng tiềm ẩn nhiều nguy cơ về an ning nếu không được cấu hình đúng cách. Bên cạnh đó, nghiên cứu của (Jeeva Chelladhurai & Sathish Kumar, 2016) đưa ra các cảnh báo về những lỗ hổng ấy thường gặp bao gồm tấn công vượt quyền từ container, tấn công từ chối dịch vụ (DoS), hoạc rò rỉ thông tin qua các tài nguyên được chia sẻ. Các biện pháp như tối ưu bảo mật cấu hình, giới hạn quyền truy cập hoặc áp dụng các chính sách tối thiểu hóa quyền là rất quan trọng để bảo vệ môi trường làm việc Docker.



#### Hình 7. Kiến trúc tổng quan của Docker

### *2.3.2. Pytorch*

PyTorch được biết đến là một thư viện học sâu mã nguồn mở được phsat triển bởi Facebook AI Research, lần đầu được ra mắt vào năm 2016 và đến nay nó đã trở thành một trong những công cụ phổ biến nhất trong cộng đồng học máy.

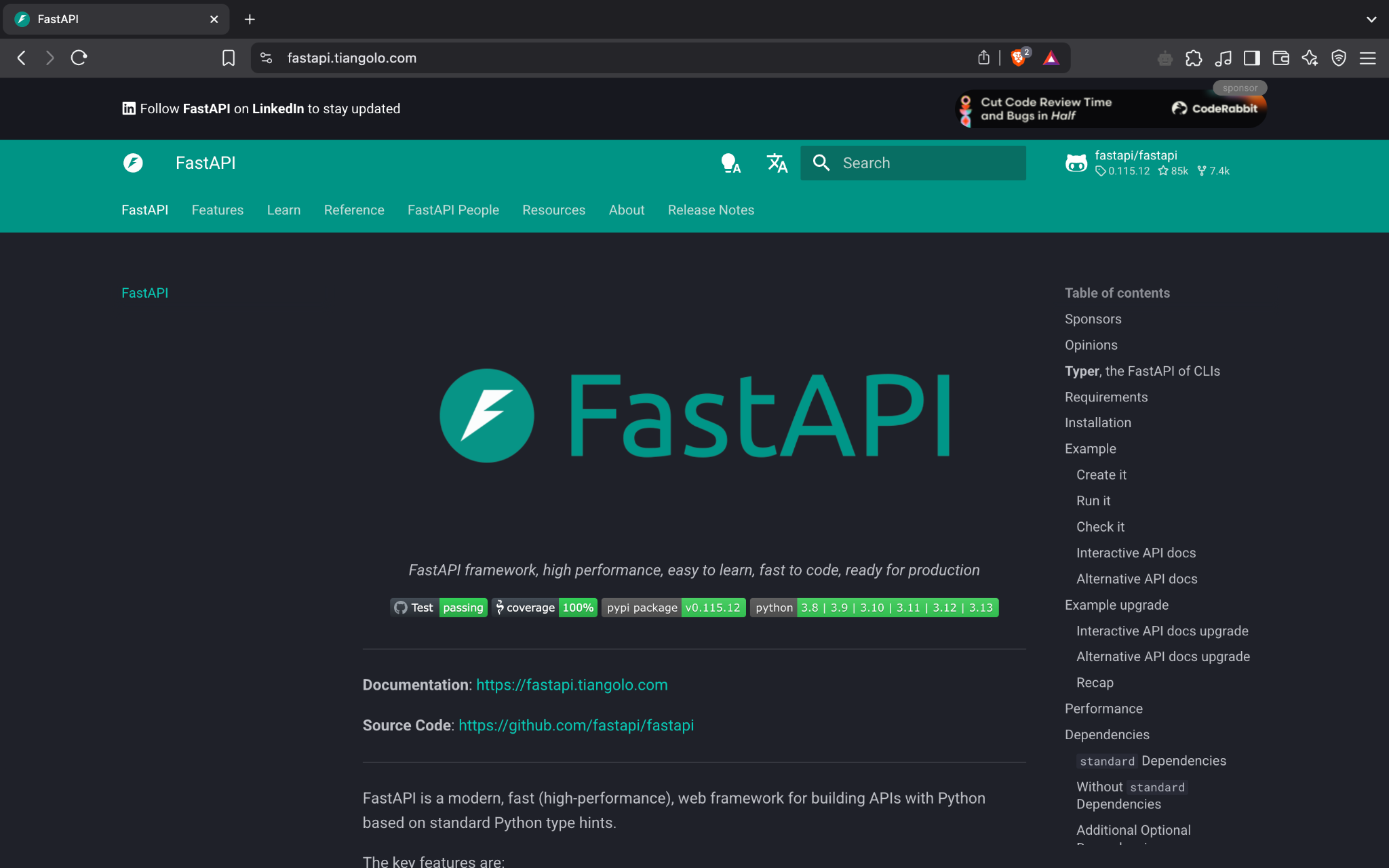
PyTorch được đánh giá cao nhờ thiết kế linh hoạt cùng với khả năng tích hợp với hệ sinh thái Python, đặc biệt là tính năng define-by-run, cho phép mô hình có thể được xây dựng và cập nhật theo từng command cụ thể, điều này phù hợp với các nghiên cứu mang tính tương tác cao hoặc mô hình động. Về mặt kiến trúc, PyTorch được tổ chức xoay quanh ba thành phần cốt lõi gồm:

* Tensor: cấu trúc dữ liệu tương tự NumPy nhưng hỗ trợ về mặt tính toán trên GPU.
* Autograd: Đây là hệ thống lan truyền ngược tự động giúp mô hình tính đạo hàm hiệu quả.
* Torch.nn : Bộ công cụ xây dựng mạng nơ-ron được thiết kế theo hướng mô-đun hóa.

Bên cạnh đó, PyTorch còn hỗ trợ các hệ sinh thái mạnh có thể sử dụng rộng rãi như TorchVission, TorchAudio hoặc TorchMetrics, đây là những thư viện mở rộng nhằm phục vụ cho các bài toán chuyên biệt trong xử lý dữ liệu dạng ảnh, âm thanh từ đó đánh giá mô hình (Nicki Detlefsen và cộng sự, 2022; Adam Paszke và cộng sự, 2019).

Trong bài nghiên cứu (Rima Palli, 2025), PyTorch được sử dụng như công cụ chính để xây dựng và huấn luyện mô hình học máy. Việc lựa chọn PyTorch không chỉ đến từ hiệu suất tính toán cao và khả năng tích hợp các thư viện khác như NumPy và Pandas mà còn nhờ vào sự linh hoạt và hỗ trợ rộng rãi của cộng đồng học máy. Đây là những yêu tố then chốt giúp PyTorch đảm bảo khả năng mở rộng và thực hiện lại pipeline nghiên cứu một cách hiệu quả.

### *2.3.3. Fast API*



#### Hình 8. Giao diện trang chủ FastAPI

Fast API là một framework trong Python hiện đại được Sebastián Ramírez thiết kế và cho ra mắt vào năm 2018 để xây dựng các API web hiệu năng cao nhanh chóng và trực quan hơn. Fast API thu hút nhiều sự chú ý từ những hôm đầu ra mắt nhờ khả năng xử lý bất đồng bộ mạnh tích hợp các thư viện như Pydantic và Starlette, đồng thời có khả năng tự động sinh tài liệu API theo chuẩn OpenAPI, một điều cần nhiều tài nguyên và công sức nếu được thực hiện thủ công.

Một điểm khác biệt của Fast API ở việc khai thác hệ thốgn gợi ý kiểu dữ liệu (type hints) của Python để tự động kiểm tra và xác minh dữ liệu đầu vào. Điều này không chỉ giúp giảm thiểu lỗi khi lập trình ở những bước tiếp theo mà còn tạo điều kiện cho việc tự sinh tài liệu, từ đó cải thiện trải nghiệm lập trình. Fast API cũng hỗ trợ cơ chế dependency injection, cho phép mô hình tái sử dụng kết nối cơ sở dữ liệu, xác thực người dùng hoặc cấu hình chung trong các ứng dụng có quy mô lớn và có thể xây dựng giao diện tương tác với người dùng như Swagger UI để lập trình viên có thể thử nghiệm trực tiếp API trong quá trình phát biểu.

## 2.4. Các chỉ số đánh giá

Đánh giá mô hình học sâu dự báo chuỗi thời gian:

### *2.4.1. Mean Absolute Error - MAE*

Mean Absolute Error (MAE) - Sai số trung bình tuyệt đối là chỉ số được dùng để đánh giá độ chính xác của các mô hình dự đoán, cũng là thước đo độ mất mát trong các bài toán hồi quy (Tianfeng Chai & R. Draxler, 2014; Johannes Fürnkranz và cộng sự, 2010). MAE đo lường sai số trung bình trong tập hợp các dự đoán, mà không tính toán đến hướng của chúng. Cũng như tên gọi, MAE có thể hiểu đơn giản là trung bình cộng của tất cả các sai số tuyệt đối của các điểm dữ liệu trong tập dữ liệu.

Để tính toán MAE, chúng ta tính hiệu giữa giá trị dự đoán và giá trị thực tế tương ứng. Kết quả tính toán này gọi là phần dư, đại diện cho sai số dự đoán của mô hình tại điểm dữ liệu đó. Sau đó, chúng ta lấy giá trị tuyệt đối của các phần dư để tránh việc các sai số có giá trị âm và dương triệt tiêu lẫn nhau. Cuối cùng là lấy tổng các giá trị tuyệt đối của các sai số và chia cho tổng sai số (số điểm dữ liệu).

Biểu thức tính sai số trung bình MAE:

Trong đó:

* n: Số lượng quan sát hay số điểm dữ liệu
* : Giá trị thực (True value)
* : Giá trị dự đoán (Predicted value)

Một số đặc điểm của MAE:

* MAE được tính toán trên thang đo tuyến tính, tức là tất cả sai số riêng lẻ đều đóng góp vào giá trị trung bình. Do đó, dù là sai số lớn hay nhỏ thì đều được gán trọng số như nhau.
* MAE không bị ảnh hưởng nhiều bởi outliers như các chỉ số đánh giá khác. Điều này chủ yếu là do MAE chỉ tính theo độ lớn tuyến tính của các sai số, khiến nó trở thành thước đo phù hợp với các tập dữ liệu có sự xuất hiện của outliers.
* MAE là chỉ số thống kê đơn giản và rõ ràng, đại diện cho giá trị trung bình của các sai số dự đoán. Chỉ số này đơn giản về cả khái niệm và cách thức tính toán, giúp cho những người không chuyên về kỹ thuật cũng có thể dễ dàng hiểu được nó.
* MAE không khả vi tại giá trị 0, vì vậy nó có thể gây ra hạn chế khi đánh giá các kỹ thuật tối ưu hóa để cải thiện hiệu suất mô hình.

### *2.4.2. Root Mean Square Error - RMSE*

Tương tự như MAE, Root Mean Square Error (RMSE) - Độ lệch bình phương trung bình gốc cũng là một chỉ số được sử dụng phổ biến trong học máy để đánh giá độ chính xác của các dự đoán từ mô hình (Tianfeng Chai & R. Draxler, 2014). Trong máy học phần dư (residual) biểu diễn khoảng cách giữa các điểm dữ liệu với đường hồi quy. Về cơ bản, RSME là độ lệch chuẩn của phần dư, đo lường mức độ phân tác của các phần dư này, cho biết độ phù hợp của dữ liệu với các dự đoán của mô hình.

Về định nghĩa, RSME là căn bậc hai của giá trị trung bình của bình phương các hiệu của giá trị dự đoán và giá trị thực tế. Cách tính RSME về cơ bản là khá giống với công thức tính độ lệch chuẩn và công thức tính phương sai của giá trị quan sát và giá trị kỳ vọng.

Biểu thức tính độ lệch bình phương trung bình gốc RSME:

Trong đó:

* n: Số lượng quan sát hay số điểm dữ liệu
* : Giá trị thực (True value)
* : Giá trị dự đoán (Predicted value)

Một số đặc điểm của RSME:

* RSME có cùng đơn vị với biến phụ thuộc, giúp dễ diễn giải và so sánh giữa các mô hình hoặc các tập dữ liệu khác nhau.
* RSME có khoảng biến thiên là từ 0 đến dương vô cực.
* Giá trị RSME càng thấp cho thấy mô hình dự đoán càng chính xác. Ngược lại, RSME có giá trị càng cao tức là có sai số đáng kể giữa giá trị dự đoán và thực tế, và mô hình dự đoán kém chính xác hơn.
* RSME rất nhạy cảm với outliers. Bình phương sai số càng lớn sẽ gây ảnh hưởng đáng kể đến tổng giá trị. Điều này cũng hỗ trợ nhận biết tác động của outliers đến hiệu suất mô hình.
* RSME trừng phạt rất nặng các sai số lớn do yếu tố bình phương. Chính vì đặc điểm này, RSME thường được dùng làm tiêu chí tối ưu hóa trong quá trình huấn luyện mô hình, đòi hỏi mô hình không chỉ cần đưa ra dự đoán chính xác mà còn phải giảm thiểu sự xuất hiện của các sai số lớn.

### *2.4.3. Hệ số xác định R bình phương - R-squared*

R-squared - Hệ số xác định R bình phương là chỉ số thống kê được dùng phổ biến để đánh giá chất lượng của mô hình hồi quy, cho biết độ phù hợp của mô hình với dữ liệu (James Colton & Keith Bower, 2002; Dalson Figueiredo và cộng sự, 2011).Về cơ bản, R-squared đo lường tỷ lệ phương sai của biến phụ thuộc có thể được giải thích bởi biến độc lập trong mô hình hồi quy, chỉ ra phần trăm biến thiên của biến mục tiêu có thể được giải thích bởi biến độc lập.

R-squared được tính toán bằng cách so sánh tổng bình phương sai số (Sum of Squares of Errors - SSE) hay tổng bình phương phần dư (Sum of Squared Residuals - SSR) với tổng bình phương toàn phần (Total Sum of Squares - SST). SSE là tổng bình phương hiệu giữa giá trị thực tế của biến phụ thuộc và giá trị dự đoán của mô hình, nó đại diện cho sự biến thiên không được giải thích bởi biến độc lập. SST là tổng bình phương của hiệu giá trị thực tế của biến phụ thuộc và giá trị trung bình của tất cả các giá trị thực tế, đại diện cho tổng biến thiên trong biến phụ thuộc.

Biểu thức tính R-squared:

Đặc điểm của R-squared:

* Giá trị R-squared nằm trong khoảng [0,1]. Giá trị R-squared càng gần 1 thì chứng tỏ mô hình có độ tương thích cao với dữ liệu, có thể giải thích càng nhiều cho sự biến thiên của biến phụ thuộc. Ngược lại, giá trị R-squared càng gần 0 tức là mô hình có thể giải thích ít hoặc gần như không thể giải thích cho bất kỳ sự biến thiên nào của biến phụ thuộc.
* Tuy nhiên, giá trị R-squared cao hay thấp không hoàn toàn nói lên việc mô hình được áp dụng là tốt hay xấu, cũng không đảm bảo được rằng mô hình sẽ hữu ích cho việc dự đoán với quan sát mới. Ngoài ra, giá trị càng cao cũng là dấu hiệu của overfitting và giá trị thấp cũng có thể báo hiệu cho việc underfitting.
* Việc đánh giá giá trị R-squared là tốt hay xấu còn tùy thuộc vào ngữ cảnh và lĩnh vực mà nó được áp dụng.
* R-squared giả định rằng tất cả các biến trong mô hình là độc lập, nhưng trên thực tế thì điều này không phải lúc nào cũng đúng. Đồng thời nó cũng có thể gặp khó khăn trong việc phát hiện quan hệ phi tuyến và cho kết quả lầm lẫn khi làm việc với dữ liệu nhỏ hơn.

# Chương 3: TRIỂN KHAI THỬ NGHIỆM HỆ THỐNG

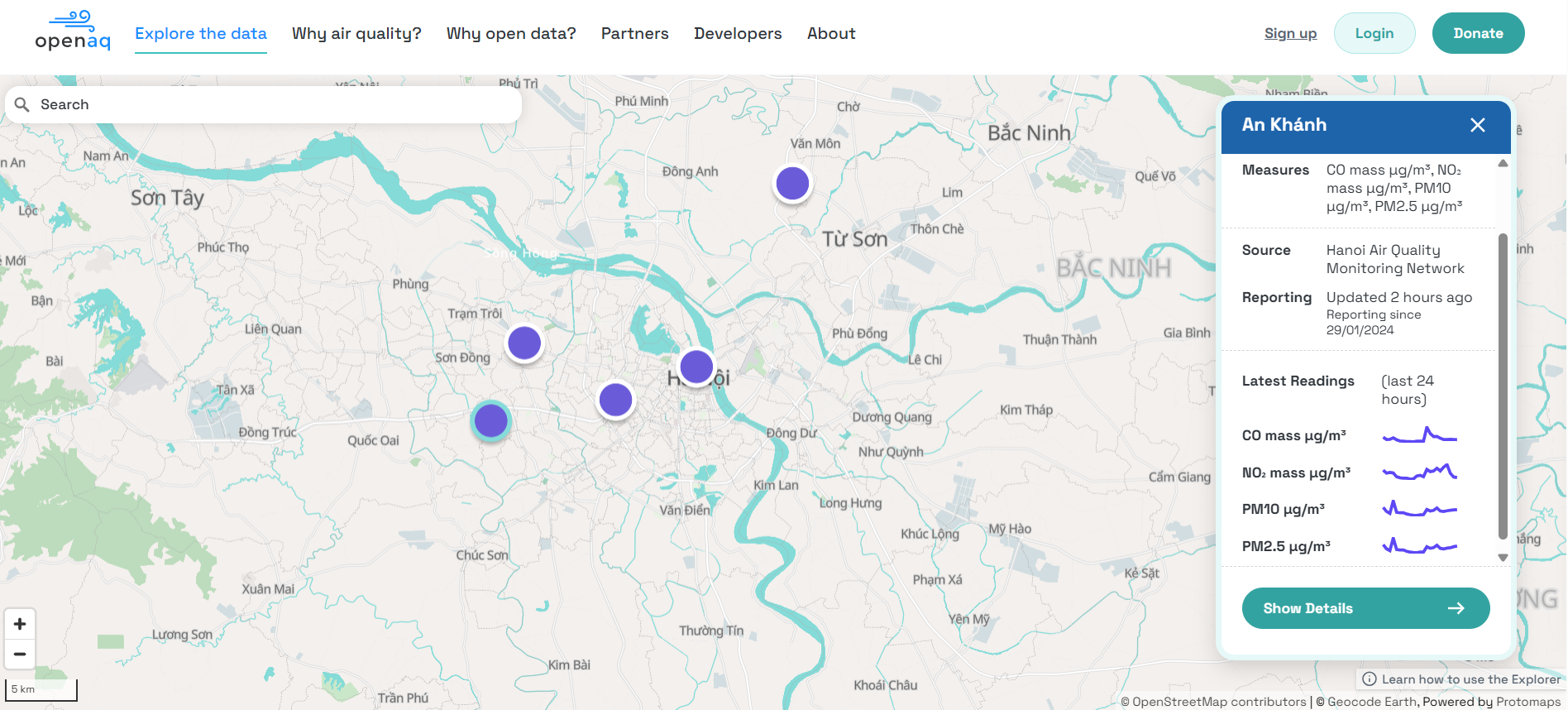
## 3.1. Chuẩn bị dữ liệu

OpenAQ có thông tin dữ liệu của 47 trạm ở Việt Nam. Nhóm chọn 1 trạm để lấy dữ liệu huấn luyện mô hình và xây dựng hệ thống dự báo.

Các tiêu chí lựa chọn trạm để lấy dữ liệu:

* Có khoảng thời gian đủ dài kể từ khi bắt đầu thu thập từ sensors => đảm bảo đủ dữ liệu để huấn luyện, không quá ít bản ghi, tránh mô hình bị overfitting
* Chọn trạm có nhiều hơn 1 sensor, phù hợp với dạng bài toán time series kiểu Multivariate-to-Multivariate hoặc Multivariate-to-Univariate
* Trạm vẫn còn cập nhật dữ liệu đến thời điểm hiện tại (18/5/2025)

Sau khi xem xét một số trạm có sensor còn đang hoạt động, nhóm chọn thu thập dữ liệu ở trạm An Khánh, Hà Nội. Dữ liệu ở trạm này có 8880 bản ghi, có 4 sensors được ghi nhận: CO mass µg/m³, NO₂ mass µg/m³, PM10 µg/m³, PM2.5 µg/m³. Trong đó, cả 4 sensors vẫn còn được cập nhật dữ liệu đến thời điểm hiện tại. Dữ liệu ở trạm An Khánh được cung cấp bởi Mạng lưới Giám sát Chất lượng Không khí Hà Nội (Hanoi Air Quality Monitoring Network) kể từ ngày 29/01/2024.



#### Hình 9. Vị trí của trạm An Khánh trên bản đồ

*Nguồn: (OpenAQ, 2025)*

#### Hình 10. Thông tin cơ bản của trạm An Khánh

*Nguồn: (OpenAQ, n.d.)*

Nhóm sử dụng thư viện openaq, một công cụ được Python hỗ trợ để truy xuất dữ liệu từ nền tảng OpenAQ. Nhóm kết nối với API với API key đã tạo để khởi tạo đối tượng OpenAQ, từ đó có thể truy cập được các endpoint như countries, locations, sensors, measurements.

Sau khi thiết lập kết nối, nhóm tìm country code của Việt Nam theo chuẩn ISO và tìm được ID của Việt Nam là 56. Tiếp đó nhóm truy xuất tất cả các địa điểm (locations) của các trạm đo chất lượng không khí tại Việt Nam bằng lệnh api.locations.list() với countries\_id=56. Mỗi địa điểm sẽ có các thông tin bao gồm tên trạm, tọa độ địa lý, thời gian hoạt động của các cảm biến và môt số thông số khác. Danh sách các địa điểm này được lưu trữ dưới dạng DataFrame để tiện xử lý và phân tích.

Từ danh sách các địa điểm thu được, nhóm tiến hành phân tích sâu hơn để đánh giá từng trạm dựa trên thời gian hoạt động và số lượng cảm biến. Nhóm tính độ dài thời gian đo lường tại mỗi trạm bằng cách lấy hiệu của đặc trưng datetime\_first và datetime\_last trong phần thông tin của các địa điểm. Đồng thời nhóm cũng thống kê và sắp xếp danh sách các trạm theo thứ tự giảm dần thời gian hoạt động và số lượng sensors nhằm hỗ trợ việc đưa ra quyết định lựa chọn trạm phù hợp.

Từ danh sách đó, nhóm xác định lựa chọn trạm An Khánh, Hà Nội (ID location = 2161290) là trạm phù hợp. Lựa chọn xong thì nhóm dùng lệnh api.locations.sensors() với location\_id = 2161290 để truy xuất toàn bộ thông tin của các sensors tại trạm này. Kết quả là nhóm thu được danh sách gồm 4 sensors là CO, NO₂, PM10 và PM2.5, cùng với đó là thời gian bắt đầu và kết thúc đo lường của từng sensor. Nhóm nhận thấy cả 4 sensor đều vẫn còn đang hoạt động.

Với danh sách sensor thu được, nhóm dùng endpoint measurements để lấy dữ liệu chi tiết từ từng sensor. Nhóm thực hiện lấy dữ liệu theo khoảng thời gian là từ ngày 29/01/2024 đến 18/05/2025, dữ liệu được truy xuất theo từng trang với tối đa 1000 bản ghi trong mỗi lượt gọi API, bao gồm các thông tin như thời gian đo, giá trị đo, đơn vị,… Sau đó, dữ liệu lấy được sẽ lưu trữ ở dạng DataFrame.

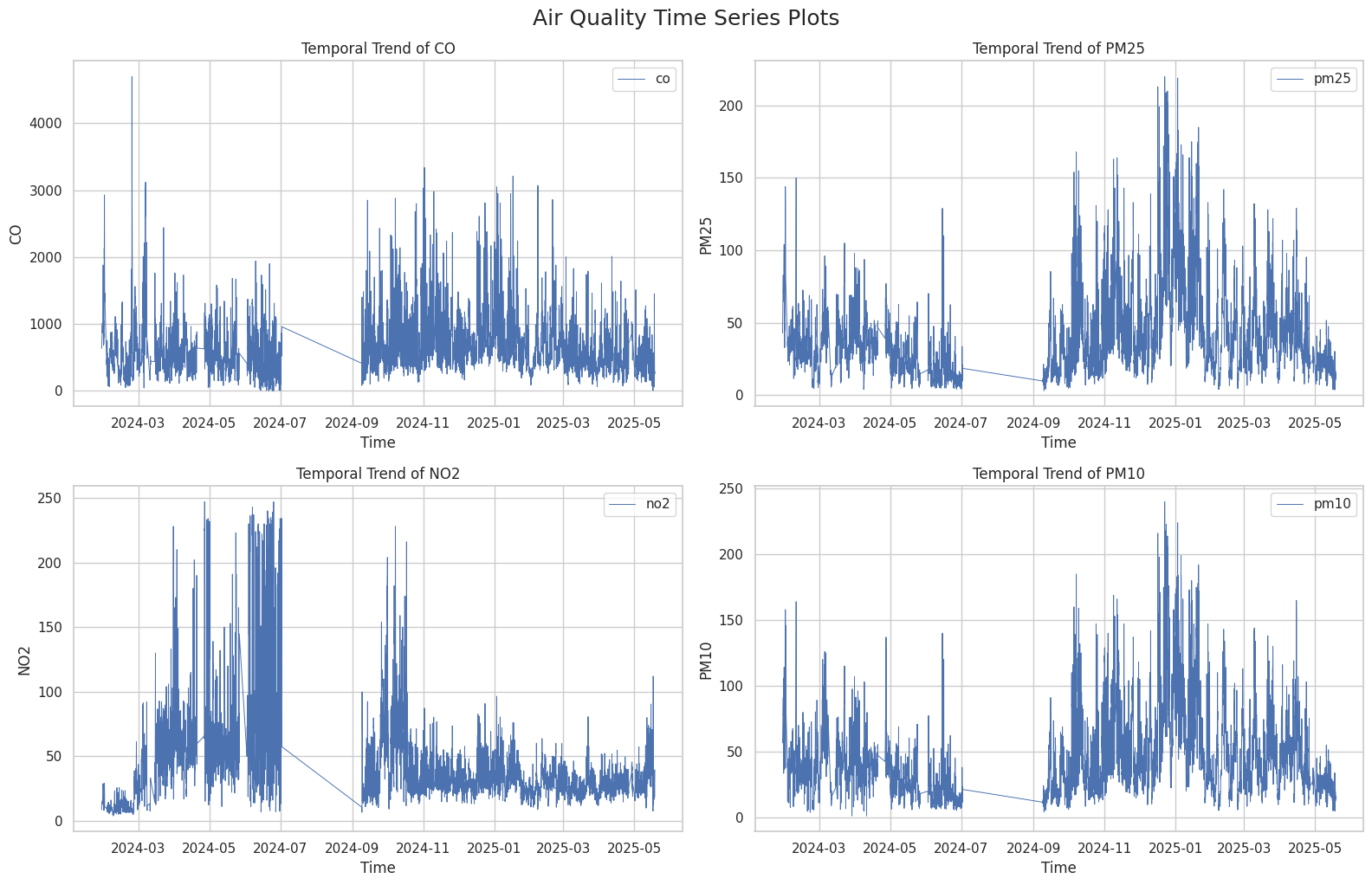
Sau khi lấy dữ liệu từ từng sensor, nhóm tiến hành gộp các bảng thành một bảng tổng hợp dựa trên trường thời gian (datetime). Cuối cùng, nhóm xây dựng hàm preprocess để thực hiện tiền xử lý dữ liệu. Nhóm loại bỏ đơn vị µg/m³ ra khỏi tên cột và xóa cột pm10 do không sử dụng trong huấn luyện. Để đảm bảo tính liên tục của chỉ số thời gian, nhóm đặt datetimeFrom\_local làm chỉ số thời gian (index), tạo dãy thời gian cách đều nhau với khoảng cách là 1 giờ, và dung foward fill để lấp vào các giá trị NaN. Nhóm sử dụng log-transform log1p cho các cột co, pm25, no2, điều này giúp ổn định phương sai và làm giảm ảnh hưởng của các giá trị cực đoạn. Ngoài ra, nhóm cũng có tạo ra các đặc trưng theo thời gian, chẳng hạn hour, dayofweek là trích xuất giờ và ngày trong tuần, is\_holiday đánh dấu các ngày lễ ở Việt Nam dựa trên thư viện holidays. Nhóm mã hóa các giá trị tuần hoàn, biểu diễn giờ trong ngày và ngàu trong tuần theo dạng đồ thị sin-cos. Kết quả cuối cùng trả về một DataFrame đã được xử lý và phù hợp để dùng cho huấn luyện mô hình dự báo.

## 3.2. Thăm dò dữ liệu

Trước khi thực hiện phân tích khám phá dữ liệu (EDA), nhóm tiến hành xử lý sơ bộ nhằm đảm bảo dữ liệu được nhất quán và phù hợp cho các bước phân tích tiếp theo. Nhóm đổi tên các cột dữ liệu để loại bỏ đơn vị đo lường, đồng thời định dạng thời gian được chuyển về múi giờ địa phương UTC+07:00, múi giờ chung của các quốc gia như Việt Nam, Thái Lan, Indonesia,… Việc sử dụng múi giờ địa phương là rất cần thiết trong các phân tích chuỗi thời gian, vì các hoạt động của con người thường tuân theo nhịp sinh hoạt của địa phương mà họ sinh sống. Với bài nghiên cứu về dự đoán chất lượng không khí mà nhóm thực hiện, điều này lại càng cần thiết hơn nữa vì các hoạt động tham gia giao thông, tạo ra các khí thải môi trường thường diễn ra theo các khung giờ khác nhau trong ngày, đặc biệt là vào các giờ cao điểm buổi sáng và buổi tối, dẫn đến có sự dao động về nồng độ khí thải, ô nhiễm môi trường không khí theo thời gian trong ngày. Ngoài ra sự dao động này cũng khác nhau đáng kể giữa ngày thường và các ngày cuối tuần, ngày lễ, phản ánh sự thay đổi trong sinh hoạt hằng ngày của người dân địa phương có tác động rõ rệt đến lưu lượng giao động và sự tạo sinh khí thải sinh hoạt.

Dữ liệu ban đầu bao gồm 8.881 bản ghi với 5 cột, trong đó cột datetimeFrom\_local có định dạng datetime64[ns, UTC+07:00]. Các biến đo lường chất lượng không khí bao gồm co, no2, pm10 và pm25, tất cả đều có kiểu dữ liệu là số thực (float64). Dữ liệu bị thiếu bản ghi hợp lệ khi các biến này thấp hơn so với tổng số thời điểm ghi nhận: co có 8.873 giá trị, pm25 có 8.866, no2 có 8.874 và pm10 có 8.847 giá trị.

Để đảm bảo tính liên tục cần thiết cho phân tích chuỗi thời gian, dữ liệu đã được tái cấu trúc bằng cách tạo ra một chuỗi thời gian liên tục với tần suất 1 giờ, bắt đầu từ thời điểm sớm nhất đến muộn nhất trong tập dữ liệu. Dữ liệu sau đó được tái lập chỉ mục theo chuỗi thời gian này, bổ sung các bản ghi thiếu (biểu diễn bằng giá trị NaN). Sau khi tái lập chỉ mục, tổng số bản ghi trong tập dữ liệu là 11.402, phản ánh rõ rệt mức độ thiếu dữ liệu ban đầu và cho thấy cần tiếp tục xử lý các giá trị bị thiếu.

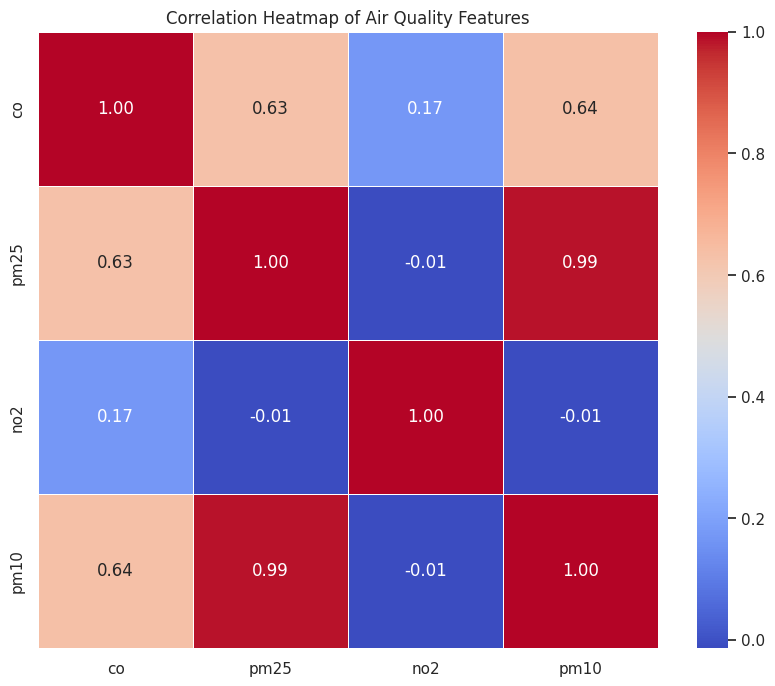


#### Hình 11. Biểu đồ chuỗi thời gian chất lượng không khí

*Nguồn: Nhóm tác giả*

Biểu đồ chuỗi thời gian cho thấy cả bốn biến đều có biên độ dao động lớn và biểu hiện xu hướng biến động theo thời gian. Mặc dù không phải là xu hướng tuyến tính nhất quán, nhưng có sự dịch chuyển về mức trung bình trong các giai đoạn khác nhau. Các biến pm25 và pm10 còn cho thấy dấu hiệu của tính mùa vụ trong khoảng từ tháng 2 đến tháng 6. Các khoảng thiếu dữ liệu tập trung chủ yếu trong giai đoạn từ tháng 7 đến tháng 9 năm 2024, tạo nên một khoảng trống dữ liệu lớn ở cả 4 biến

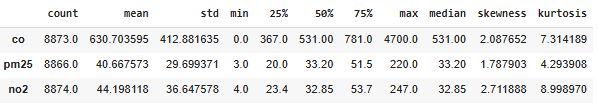
Kết hợp các quan sát từ biểu đồ, có thể nhận định rằng cả bốn biến đều không có tính dừng và cần được xử lý trước khi tiến hành các mô hình dự báo.



#### Hình 12. Biểu đồ nhiệt tương quan các đặc trưng chất lượng không khí

*Nguồn: Nhóm tác giả*

Biểu đồ nhiệt tương quan cho thấy mối quan hệ rất mạnh giữa pm10 và pm25 (hệ số tương quan đạt 0.99), hàm ý sự trùng lặp về thông tin. Ngoài ra, pm10 có tương quan vừa phải với co (0.64), tương tự như pm25 (0.63). Trong khi đó, no2 thể hiện tương quan yếu với co và có mối tương quan âm nhẹ với pm10 và pm25. Để tránh vấn đề đa cộng tuyến, biến pm10 đã được loại bỏ khỏi tập dữ liệu ở các bước xử lý sau đó. Biến no2 tuy có tương quan thấp nhưng vẫn được giữ lại do tiềm năng mang thông tin độc lập hữu ích cho mô hình dự đoán.



#### Hình 13. Thống kê mô tả các đặc trưng chất lượng không khí

*Nguồn: Nhóm tác giả*

Phân tích thống kê mô tả cho thấy các biến pm25, co, và no2 đều có phân phối lệch phải (Skewness > 1) và nhọn (Kurtosis > 4), điều này cho thấy cần thực hiện chuẩn hóa dữ liệu trước khi đưa vào mô hình.

Nhóm xây dựng lớp TimeSeriesDataLoader để load và xử lý dữ liệu, chuẩn bị đặc trưng đầu vào và nhãn cho việc huấn luyện mô hình.

Trong đó, các bước tiền xử lý bao gồm các bước biến đổi dữ liệu cũng như chuẩn hóa nhằm khắc phục các vấn đề của dữ liệu đã nêu:

1. Đổi tên cột và loại bỏ cột không cần thiết bao gồm cột pm10.
2. Chuẩn hóa thời gian và tái lập chỉ mục.
3. Bổ sung các đặc trưng theo thời gian gồm: tuần hoàn giờ trong ngày (hour\_sin, hour\_cos), ngày trong tuần (dayofweek\_sin, dayofweek\_cos) và biến nhị phân is\_holiday cho biết thời điểm đó có phải là ngày nghỉ ở Việt Nam hay không.
4. Biến đổi logarit: khắc phục vấn đề phân phối lệch phải và độ nhọn cao của các biến co, no2, pm25. Cùng với các đặc trưng tuần hoàn gián tiếp giúp giảm bớt tính không dừng của dữ liệu.
5. Lưu dữ liệu đã xử lý: dữ liệu sau tiền xử lý được lưu lại dưới dạng .csv trong thư mục processed.

Dữ liệu sau khi đã được tiến hành các bước tiền xử lý cơ bản sẽ tiếp tục được biến đổi để phù hợp với mô hình dự đoán đa biến đến đa biến (Multivariate to Multivariate). Cụ thể, mô hình sẽ nhận đầu vào gồm 8 đặc trưng với độ dài 72 bước thời gian, tương ứng với 72 giờ liên tiếp. Mục tiêu dự báo là 3 biến đầu ra (co, no2, pm25) trong khoảng thời gian 24 giờ. Nhóm cũng tạo khoảng cách offset 24 giờ giữa phần dữ liệu đầu vào và phần nhãn dự báo để cải thiện khả năng tổng quát hóa và tránh việc mô hình “nhìn trước” vào nhãn gần. Các chuỗi dữ liệu mẫu được trích xuất với bước trượt (stride) là 6 giờ, tạo ra sự chồng lấp vừa phải giữa các chuỗi dữ liệu, giúp mô hình học được nhiều mẫu đa dạng mà vẫn giữ được tính liên tục của dữ liệu chuỗi thời gian.

Quá trình trích xuất các chuỗi dữ liệu hợp lệ sau đó được thực hiện bằng cách áp dụng cửa sổ trượt (sliding window) với độ dài tổng cộng là input\_size + offset + label\_size = 120 trên toàn bộ dữ liệu, loại bỏ những cửa sổ có giá trị thiếu. Kết quả là tập các chỉ số bắt đầu của các chuỗi hợp lệ được lưu giữ để tạo tập dữ liệu đầu vào cho mô hình.

Như vậy, đầu vào của dữ liệu có kích thước (số mẫu, 72, 8) và nhãn có kích thước (số mẫu, 24, 3).

Sau đó, dữ liệu được chia với tỉ lệ 70:15:15 tạo ra tập huấn luyện gồm 522 mẫu, tập validation có 112 mẫu và tập test có 113 mẫu, đảm bảo đủ lượng dữ liệu cho các giai đoạn huấn luyện và đánh giá.

Cuối cùng, dữ liệu được chuyển đổi thành tensor PyTorch với kiểu dữ liệu float32 và đóng gói thành các DataLoader với kích thước batch là 32.

## 3.3. Thiết kế các mô hình học sâu

Đối với bài toán này, nhóm sử dụng 3 mô hình Mạng nơ-ron hồi tiếp (Recurrent Neural Networks - RNN) tiêu biểu là LSTM, RNN và GRU để dự đoán, với thư viện PyTorch trên môi trường VSCode.

Đầu tiên, class ModelManager được xây dựng để quản lý toàn bộ quy trình huấn luyện, đánh giá và lưu mô hình cho cả 3 kiến trúc. Mô hình được huấn luyện với hàm mất mát L1Loss và tối ưu bằng thuật toán Adam. Trong quá trình đánh giá, các chỉ số MAE, RMSE và R² được tính toán ở cả không gian log1p và đơn vị gốc. Ngoài ra, cơ chế early stopping được áp dụng dựa trên độ lỗi trên tập validation để tránh overfitting và rút ngắn thời gian huấn luyện.

Mỗi mô hình (RNN, LSTM, GRU) đều được định nghĩa dưới dạng một lớp kế thừa từ nn.Module, lần lượt là class RNN(nn.Module), class LSTM(nn.Module) và class GRU(nn.Module). Cả ba đều được triển khai theo kiến trúc tổng quát giống nhau, bao gồm các thành phần chính sau:

* Lớp RNN/LSTM/GRU: đảm nhiệm việc xử lý dữ liệu chuỗi.
* Batch Normalization: giúp ổn định quá trình huấn luyện bằng cách chuẩn hóa đầu ra.
* Dropout: giảm hiện tượng overfitting.
* Fully Connected Layers (trong một nn.Sequential): gồm hai lớp Linear kết hợp với ReLU và Dropout.

Các mô hình được huấn luyện trên dữ liệu đã chuẩn bị với 50 epochs.

## 3.4. Triển khai hệ thống

**3.4.1. Cấu trúc thư mục dự án**

Nhóm đã triển khai dự án dự báo chất lượng không khí thời gian thực bằng FastAPI và mô hình RNN, LSTM, GRU theo cấu trúc sau:

A screenshot of a computer

AI-generated content may be incorrect.

*Hình 14. Cấu trúc thư mục dự án*

*Nguồn: Nhóm tác giả*

Thư mục api chứa các thành phần liên quan đến API được xây dựng bằng FastAPI:

* main.py: định nghĩa endpoint /predict để phục vụ dự báo.
* data\_collector.py: định nghĩa lớp OpenAQProcessor để thu thập và xử lý dữ liệu từ API OpenAQ.

Thư mục data chứa dữ liệu đầu vào thô và dữ liệu đã xử lý:

* raw/: ankhanh\_measurements.csv - dữ liệu gốc thu thập từ OpenAQ API.
* processed/: processed\_data.csv

Thư mục docker chứa các file cấu hình để triển khai hệ thống dưới dạng container hóa:

* Dockerfile: xây dựng image chứa API và mô hình đã huấn luyện
* docker-compose.yml: định nghĩa dịch vụ để chạy ứng dụng với Docker Compose

Thư mục model chứa kiến trúc và trọng số các mô hình đã huấn luyện:

* models.py: định nghĩa các lớp mô hình học sâu RNN, LSTM và GRU bằng PyTorch
* config.py: cấu hình mô hình (số lớp, kích thước ẩn, số giờ dự báo,...).
* train.py: huấn luyện mô hình.
* best-\*.pth: các file trọng số mô hình tốt nhất đã huấn luyện.
* utils/: tiện ích cho việc load dữ liệu (data\_loader.py), quản lý mô hình (model\_manager.py)

Các file còn lại:

* requirements.txt
* [README.md](http://readme.md)
* .gitinore, LICENSE

### 3.4.2. Các bước triển khai

Quá trình triển khai hệ thống dự báo chất lượng không khí được thực hiện theo các bước chính sau:

*Bước 1: Huấn luyện mô hình*

Trước tiên, các mô hình học sâu như RNN, LSTM và GRU được huấn luyện trên dữ liệu lịch sử chất lượng không khí đã được xử lý sẵn. Quá trình huấn luyện được thực hiện bằng PyTorch, với các siêu tham số được cấu hình trong tệp config.py.

Sau khi quá trình huấn luyện hoàn tất, mô hình tốt nhất theo tiêu chí như giá trị loss  thấp nhất trên tập validation sẽ được lưu lại dưới dạng các tệp .pth trong thư mục model/

*Bước 2: Đóng gói hệ thống thành Docker image*

Hệ thống sau đó được đóng gói thành một Docker image để dễ dàng triển khai. File Dockerfile định nghĩa cách cài đặt các thư viện cần thiết, sao chép mã nguồn và khởi động server FastAPI bằng uvicorn.

Ngoài ra, một file docker-compose.yml cũng được xây dựng để quản lý việc khởi chạy các container. Tuy nhiên, nhóm chưa đủ thời gian và khả năng để triển khai hệ thống multi-service chạy trên nhiều node. Đây cũng là một hạn chế của nhóm.

*Bước 3: Khởi chạy hệ thống*

Sau khi Docker được cấu hình xong, có thể khởi chạy hệ thống bằng lệnh:

docker-compose up --build

Lệnh này sẽ build image từ mã nguồn và khởi động server API trên cổng 8000.

A screenshot of a computer program

AI-generated content may be incorrect.

*Hình 15: Docker Compose*

*Nguồn: Nhóm tác giả*

*Bước 4: Kiểm tra và truy cập API*

Sau khi hệ thống được khởi động thành công, người dùng có thể truy cập API thông qua trình duyệt hoặc các công cụ như Postman tại địa chỉ: <http://127.0.0.1:8000/predict> hoặc <http://localhost:8000/predict>

Tại đây, người dùng có thể yêu cầu dự báo chất lượng không khí trong 24 giờ tiếp theo bằng cách chọn mô hình (LSTM, GRU, RNN) thông qua 2 query parameters: ahead với giá trị mặc định là 24, yêu cầu nhập vào số giờ nhỏ hơn hoặc bằng 24; model\_name là tên của mô hình được huấn luyện bao gồm RNN, LSTM và GRU. Ví dụ như <http://localhost:8000/predict?ahead=24&model_name=lstm>.

Trang chủ (root endpoint) của ứng dụng FastAPI đã triển khai trên máy cục bộ có địa chỉ <http://127.0.0.1:8000/>

Cấu trúc thư mục dự án:

Nhóm đã triển khai dự án dự báo chất lượng không khí thời gian thực bằng FastAPI và mô hình RNN, LSTM, GRU

Các bước triển khai bao gồm:

* Huấn luyện mô hình
* Mô hình đã huấn luyện được lưu vào folder model
* Đóng gói thành docker image
* docker-compose
* Vào đường link <http://127.0.0.1:8000/predict> hoặc <http://localhost:8000/predict>

A white rectangular object with a white border

AI-generated content may be incorrect.

*Hình 16: Trang chủ của ứng dụng*

*Nguồn: Nhóm tác giả*

Trang tài liệu API tương tác (interactive API documentation) do FastAPI tự động tạo ra, sử dụng thư viện Swagger UI: <http://localhost:8000/docs>. Trang này giúp xem tất cả các endpoint mà API đang cung cấp. Thêm vào đó là gửi request trực tiếp đến các endpoint và nhận phản hồi ngay trên trình duyệt.

A black and white striped background

AI-generated content may be incorrect.

*Hình 17. Trang tài liệu API tương tác*

*Nguồn: Nhóm tác giả*

Kết quả trả về được lưu dưới dạng file json thể hiện các tham số nhập vào như model, ahead và các cột được dự đoán.

A screenshot of a computer

AI-generated content may be incorrect.

*Hình 18: Kết quả dự đoán được ghi nhận lúc 6h22 ngày 24/5/2025*

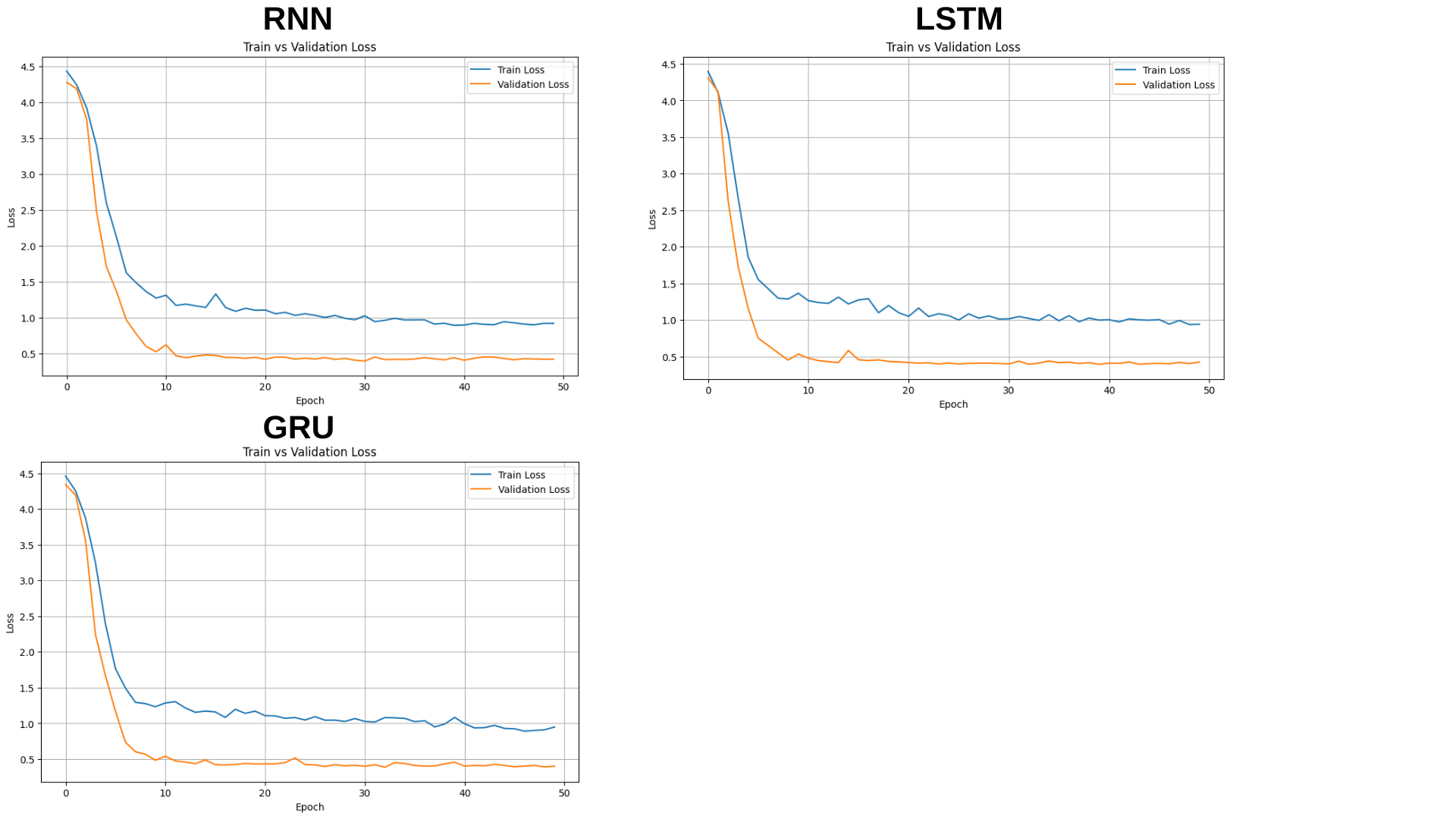
A screenshot of a computer

AI-generated content may be incorrect.

*Hình 19: Kết quả dự đoán được ghi nhận lúc 10h02 ngày 24/5/2025*

# Chương 4: BÀN LUẬN VÀ ĐÁNH GIÁ

Nhóm đã tiến hành huấn luyện các mô hình trong 50 epoch. Kết quả quá trình huấn luyện và đánh giá hiệu suất được trình bày chi tiết trong Hình x và Bảng x.



Hình: Biểu đồ minh họa loss của RNN, LSTM và GRU

Quan sát các biểu đồ loss cho thấy, cả ba kiến trúc mô hình đều thể hiện khả năng hội tụ rõ rệt, với giá trị loss trên cả tập huấn luyện và tập kiểm định giảm dần theo số lượng epoch. Sự sụt giảm loss diễn ra mạnh mẽ nhất trong khoảng 10-15 epoch đầu tiên, sau đó tốc độ giảm dần chậm lại và đạt trạng thái tương đối ổn định sau epoch 20-25. Điều này chỉ ra rằng các mô hình đã thành công trong việc học các đặc trưng cơ bản từ dữ liệu trong giai đoạn đầu của quá trình huấn luyện.

Về mức độ hội tụ cuối cùng, mô hình GRU cho thấy Training loss và Validation loss đạt giá trị thấp nhất sau 50 epoch. Tuy nhiên, có một khoảng chênh lệch giữa training loss và validation loss được quan sát ở cả ba mô hình. Đối với GRU và RNN, khoảng chênh lệch này có xu hướng mở rộng rõ rệt ở các epoch cuối cùng, cho thấy dấu hiệu của hiện tượng overfitting nhẹ. Ngược lại, từ epoch 30 trở đi, mô hình LSTM thể hiện sự hội tụ ổn định nhất, với chênh lệch giữa training loss và validation loss được kiểm soát tốt hơn so với hai mô hình còn lại.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **MAE (log1p)** | **rMSE (log1p)** | **R2 (log1p)** | **MAE (original)** | **rMSE**  **(original)** | **R2**  **(original)** |
| **RNN** | 0.46 | 0.62 | 0.79 | 79.27 | 169.81 | 0.61 |
| **LSTM** | 0.48 | 0.63 | 0.78 | 81.42 | 179.24 | 0.57 |
| **GRU** | 0.46 | 0.62 | 0.78 | 80.54 | 174.37 | 0.58 |

Bảng: Đánh giá định lượng hiệu suất của RNN, LSTM và GRU

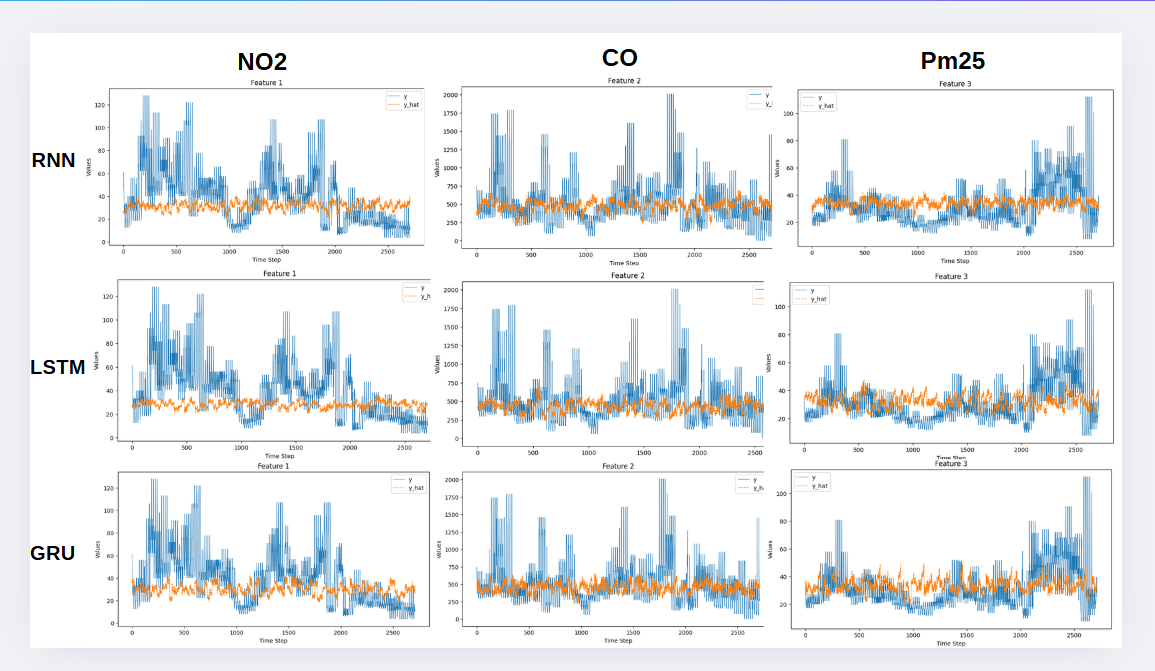
Trên thang đo log1p, ba mô hình RNN, LSTM, và GRU đạt hiệu suất khá tương đồng. Cụ thể, RNN và GRU có cùng MAE là 0.46 và RMSE là 0.62, trong khi LSTM có MAE là 0.48 và RMSE là 0.63. Về chỉ số R2, RNN đạt 0.79, còn LSTM và GRU đạt 0.78. Sự tương đồng này cho thấy khả năng của các mô hình trong việc nắm bắt xu hướng chung của dữ liệu sau khi chuẩn hóa.

Tuy nhiên, khi các chỉ số được tính toán trên dữ liệu gốc, sự khác biệt về hiệu suất trở nên rõ ràng hơn, phản ánh chính xác hơn sai số dự đoán thực tế. Mô hình RNN đạt MAE là 79.27 và RMSE là 169.81, cùng với R2 là 0.61. LSTM có MAE là 81.42, RMSE là 179.24, và R2 là 0.57. GRU ghi nhận MAE là 80.54, RMSE là 174.37, và R2 là 0.58. Từ các chỉ số này, RNN cho thấy sai số dự đoán thấp nhất và khả năng giải thích phương sai cao nhất trên dữ liệu gốc. Mặc dù giá trị R2 của cả ba mô hình không quá cao (từ 0.57 đến 0.61), chúng vẫn chỉ ra rằng các mô hình có khả năng giải thích được khoảng 57-61% sự biến động của dữ liệu thực tế, cho thấy khả năng dự đoán ở mức chấp nhận được. Sự khác biệt giữa kết quả trên dữ liệu log1p và dữ liệu gốc là do dữ liệu ban đầu có phân phối không đều, khiến các sai số nhỏ trên thang đo log1p có thể trở nên đáng kể hơn khi chuyển về thang đo gốc.

Tập dữ liệu chuỗi thời gian được nhóm thu thập có một số điểm hạn chế: không có chu kỳ rõ ràng, không dừng (non-stationary), chứa nhiều giá trị ngoại lai, phân phối bị lệch và nhọn đồng thời có nhiều khoảng dữ liệu bị thiếu. Những yếu tố này ảnh hưởng tiêu cực đến hiệu suất của hầu hết các mô hình học sâu, dẫn đến giá trị R² chưa cao trong kết quả.

Dựa trên các phân tích tổng hợp trên, mô hình LSTM được xác định là lựa chọn phù hợp nhất cho việc xây dựng pipeline phân tán với khả năng kiểm soát hiện tượng overfitting tốt hơn.

Kết quả dự đoán các chỉ số của 3 mô hình được trực quan hóa trong hình x.



Hình: Kết quả dự đoán của các mô hình

# ChKẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

5.1. Kết quả đạt được

Thu thập được một bộ dữ liệu từ OpenAQ gồm các chỉ số về chất lượng không khí ở một trạm quan trắc đặt tại Hà Nội, Việt Nam.

Phân tích khám phá dữ liệu và tiền xử lý để tạo ra một bộ dữ liệu huấn luyện cho bài toán dự báo chuỗi thời gian

Huấn luyện và so sánh, đánh giá các mô hình học sâu

Xây dựng REST API backend để dự đoán các chỉ số không khí thời gian thực, có tùy chọn mô hình và thời gian dự đoán trong 24h.

Đóng gỏi hệ thống bằng Docker

5.2. Hạn chế của hệ thống hiện tại

* Chưa chạy tối ưu hóa siêu tham số mô hình deep learning
* Chưa đánh giá hiệu suất hệ thống
* Mới chỉ có Backend dùng Fast API, chưa làm giao diện Frontend
* Hệ thống còn đơn giản, không có cơ sở dữ liệu, không có cấu hình mạng, không có nhiều service.
* Chưa thực hiện hệ thống phân tán
* Chỉ dự đoán giá trị liên tục của các chỉ số chất lượng không khí (PM2.5, CO, NO2), chưa tính AQI để đánh giá mức độ ô nhiễm

5.3. Đề xuất cải tiến và hướng phát triển tương lai

Đầu tiên là cải thiện hết tất cả những hạn chế ở phần trên. Ứng dụng với các khu vực khác ở Việt Nam hoặc trên nền tảng đám mây

# TÀI LIỆU THAM KHẢO

Bashari Rad, B., Bhatti, H., & Ahmadi, M. (2017). An Introduction to Docker and Analysis of its Performance. *IJCSNS International Journal of Computer Science and Network Security, 173*, 8.

Belavadi, S. V., Rajagopal, S., R, R., & Mohan, R. (2020). Air Quality Forecasting using LSTM RNN and Wireless Sensor Networks. *Procedia Computer Science, 170*, 241-248. doi:<https://doi.org/10.1016/j.procs.2020.03.036>

Bianchi, F. M., Maiorino, E., Kampffmeyer, M., Rizzi, A., & Jenssen, R. (2017). Recurrent Neural Network Architectures. In (pp. 23-29).

Braam, P. (2019). *The Lustre Storage Architecture*.

Chai, T., & Draxler, R. (2014). Root mean square error (RMSE) or mean absolute error (MAE)? *Geosci. Model Dev., 7*. doi:10.5194/gmdd-7-1525-2014

Chelladhurai, J., & Kumar, S. (2016). *Securing Docker Containers from Denial of Service (DoS) Attacks*.

Cho, K., Merrienboer, B., Gulcehre, C., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. doi:10.3115/v1/D14-1179

Colton, J., & Bower, K. (2002). Some Misconceptions about R-Square. *3*, 20-22.

Combe, T., Martin, A., & Pietro, R. (2016). To Docker or Not to Docker: A Security Perspective. *IEEE Cloud Computing, 3*, 54-62. doi:10.1109/MCC.2016.100

Detlefsen, N., Borovec, J., Schock, J., Jha, A., Koker, T., Di Liello, L., . . . Falcon, W. (2022). TorchMetrics -Measuring Reproducibility in PyTorch. *The Journal of Open Source Software, 7*. doi:10.21105/joss.04101

Figueiredo, D., Júnior, S., & Rocha, E. (2011). What is R2 all about? *Leviathan-Cadernos de Pesquisa Polútica, 3*, 60-68. doi:10.11606/issn.2237-4485.lev.2011.132282

Fürnkranz, J., Chan, P., Craw, S., Sammut, C., Uther, W., Ratnaparkhi, A., . . . De Raedt, L. (2010). Mean Absolute Error. In.

Gropp, W., Lusk, E., & Skjellum, A. (1999). *Using MPI - 2nd Edition: Portable Parallel Programming with the Message Passing Interface* (2nd Edition ed.): MIT Press.

Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation, 9*, 1735-1780. doi:10.1162/neco.1997.9.8.1735

Laboratory, L. L. N. (n.d.). Introduction to Parallel Computing Tutorial. Retrieved from <https://hpc.llnl.gov/documentation/tutorials/introduction-parallel-computing-tutorial>

Laboratory, O. R. N. (2022). Frontier supercomputer debuts as world’s fastest, breaking exascale barrier. Retrieved from <https://www.ornl.gov/news/frontier-supercomputer-debuts-worlds-fastest-breaking-exascale-barrier>

Linh, D. T. K., & Huan, V. V. (2024). 16. AIR QUALITY PREDICTION IN HANOI USING A DEEP LEARNING APPROACH. *Tạp chí khoa học Tài nguyên và Môi trường*(54), 172-179. doi:10.63064/khtnmt.2024.650

Minh, Q., & Hồng, H. (2024). Dự báo ô nhiễm không khí từ dữ liệu quan trắc sử dụng các mô hình trí tuệ nhân tạo. *Transport and Communications Science Journal, 75*, 1404-1412. doi:10.47869/tcsj.75.3.7

OpenAQ. (2025). Air Quality Data – Location ID: 2161290. Retrieved from <https://explore.openaq.org/?location=2161290#10.02/21.0191/105.9078>

OpenAQ. (n.d.). Monitoring Location Details – Location ID: 2161290. Retrieved from <https://explore.openaq.org/locations/2161290>

Palli, R. (2025). BirdSoundSegmentation: Spectrogram Segmentation with PyTorch.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., . . . Chintala, S. (2019). *PyTorch: An Imperative Style, High-Performance Deep Learning Library*.

Pham, T., Nguyen, V. T., Hằng, D., & Hoang, K. (2023). Enhancing Air Quality Prediction Accuracy Using Hybrid Deep Learning. *International Journal of Environmental Science and Development, 14*, 155-159. doi:10.18178/ijesd.2023.14.2.1428

Seo, J., & Kim, K.-D. (2025). An RNN based Adaptive Hybrid Time Series Forecasting Model for Driving Data Prediction. *IEEE Access, PP*, 1-1. doi:10.1109/ACCESS.2025.3554803

Sterling, T., Anderson, M., & Brodowicz, M. (2017). *High performance computing: Modern systems and practices*.

Teleron, J. (2025). Electrical Energy Demand Forecasting using Time Series in LSTM and CNN-LSTM Models in Deep Learning Applications. *Journal of Information Systems Engineering and Management, 10*, 718-724. doi:10.52783/jisem.v10i15s.2511

Tsai, Y.-T., Zeng, Y.-R., & Chang, Y.-S. (2018). *Air Pollution Forecasting Using RNN with LSTM*.

Yoo, A. B., Jette, M. A., & Grondona, M. (2003, 2003//). *SLURM: Simple Linux Utility for Resource Management.* Paper presented at the Job Scheduling Strategies for Parallel Processing, Berlin, Heidelberg.

Zhang, Y., Bocquet, M., Mallet, V., Seigneur, C., & Baklanov, A. (2012). Real-time air quality forecasting, part I: History, techniques, and current status. *Atmospheric Environment, 60*. doi:10.1016/j.atmosenv.2012.06.031

**PHỤ LỤC**

**Github:** [**https://github.com/nmhongtram/air-quality-forecasting**](https://github.com/nmhongtram/air-quality-forecasting)