TRƯỜNG ĐẠI HỌC THỦY LỢI



NHÓM 03

BÁO CÁO BÀI TẬP LỚN MÔN HỌC

PHÂN TÍCH CHUỖI THỜI GIAN

Dự báo tải mạng viễn thông Quy mô quốc gia

|  |  |
| --- | --- |
| NGƯỜI HƯỚNG DẪN | 1. TRẦN ANH ĐẠT |

Hà Nội, năm 2025

MỤC LỤC

[Thành viên nhóm 3](#_Toc19223)

[MỞ ĐẦU 4](#_Toc14186)

[CHƯƠNG 1 GIỚI THIỆU BÀI TOÁN VÀ BỘ DỮ LIỆU 5](#_Toc5501)

[CHƯƠNG 2 CƠ SỞ LÝ THUYẾT 7](#_Toc24917)

[2.1 XGBoost 7](#_Toc7687)

[2.1.1. Decision Trees (Cây quyết định) 8](#_Toc20465)

[2.1.2. Hàm mất mát 8](#_Toc26657)

[2..1.2.1. Learning Rate (Tốc độ học) 9](#_Toc9209)

[2.1.2.2. Regularization (Điều chuẩn) 9](#_Toc30891)

[2.1.3. Quy trình huấn luyện 10](#_Toc24976)

[2.1.4. Ưu điểm 10](#_Toc13023)

[2.1.5. Các tham số chính 10](#_Toc6186)

[2.1.6. Ứng dụng 11](#_Toc9491)

[2.2. LSTM 11](#_Toc11946)

[2.3. PatchTST (Patch Time Series Transformer) 13](#_Toc3295)

[2.3.1. Giới thiệu 13](#_Toc7557)

[2.3.1. Quy trình hoạt động của mô hình PatchTST 13](#_Toc21812)

[CHƯƠNG 3 THỰC NGHIỆM VÀ KẾT QUẢ 16](#_Toc28396)

[3.1 Tiền xử lý dữ liệu 16](#_Toc9830)

[3.2 Kết quả thực nghiệm 17](#_Toc23877)

[3.2.1. XGBoost 17](#_Toc7985)

[3.2.2. LSTM 18](#_Toc27266)

[3.2.3. PatchTST 20](#_Toc18242)

[3.2.4. Tạo giao diện 26](#_Toc24341)

[KẾT LUẬN 27](#_Toc3801)

Thành viên nhóm

|  |  |
| --- | --- |
| Họ tên | Mã sinh viên |
| Nguyễn Văn Quân | 2251262627 |
| Nguyễn Ngọc Tân | 2251262637 |
| Nguyễn Đình Thành | 2251262643 |

MỞ ĐẦU

Trong thời đại chuyển đổi số và bùng nổ dữ liệu hiện nay, hạ tầng viễn thông đóng vai trò then chốt trong việc duy trì sự kết nối liên tục của hàng triệu người dùng và thiết bị. Các nhà mạng viễn thông không chỉ đối mặt với áp lực về mặt kỹ thuật trong việc đảm bảo chất lượng dịch vụ (QoS), mà còn cần có khả năng **dự báo chính xác tải mạng** để tối ưu hóa việc phân bổ tài nguyên, nâng cao hiệu suất vận hành và giảm thiểu rủi ro gián đoạn mạng.

Một trong những hướng tiếp cận hiệu quả để giải quyết bài toán này là sử dụng các mô hình **dự báo chuỗi thời gian (time series forecasting)** nhằm ước lượng trước lưu lượng mạng trong tương lai tại từng khu vực cụ thể. Tuy nhiên, tải mạng viễn thông là một loại dữ liệu có tính **đa biến, phi tuyến và dao động mạnh** theo thời gian (ngày/đêm, ngày thường/cuối tuần, lễ Tết,...), nên đòi hỏi các mô hình dự báo có độ chính xác cao và khả năng khái quát tốt.

Thông qua đề tài này, nhóm hướng tới mục tiêu không chỉ **ứng dụng các mô hình học máy hiện đại vào bài toán thực tiễn**, mà còn góp phần xây dựng nền tảng hỗ trợ ra quyết định trong việc quản lý và điều phối hệ thống mạng viễn thông quy mô lớn.

# GIỚI THIỆU BÀI TOÁN VÀ BỘ DỮ LIỆU

1. **Giới thiệu bài toán**

Trong bối cảnh nhu cầu kết nối viễn thông ngày càng gia tăng và hạ tầng mạng phải liên tục thích ứng với tải trọng không ổn định, việc **dự báo tải mạng** (bao gồm lưu lượng **Internet, cuộc gọi, và tin nhắn SMS**) đóng vai trò then chốt trong việc **quản lý tài nguyên mạng hiệu quả**, đảm bảo **chất lượng dịch vụ** và tối ưu hóa hoạt động vận hành của các nhà mạng.

Đặc biệt tại các quốc gia có **mật độ thuê bao cao**, tải mạng có thể thay đổi đáng kể theo **không gian**và **thời gian** (theo chu kỳ giờ, ngày, tuần,...), đòi hỏi những mô hình dự báo có khả năng xử lý **chuỗi thời gian đa biến và có tính phi tuyến cao**.

Đề tài này tập trung vào bài toán **dự báo tải mạng viễn thông** theo **quy mô toàn quốc**, dựa trên dữ liệu **CDRs (Call Detail Records)** là bản ghi chi tiết cuộc gọi, tin nhắn và dùng dữ liệu Internet do các nhà mạng thu thập mỗi khi người dùng thực hiện một hoạt động viễn thông. Dữ liệu được thu thập và tổ chức theo lưới không gian thời gian với độ phân giải cao (mỗi 10 phút), giúp phản ánh chi tiết hoạt động của người dùng trong từng vùng cụ thể.

1. **Giới thiệu tập dữ liệu**

Dữ liệu sử dụng trong đề tài được lấy từ một tập dữ liệu thực tế được công bố bởi công ty viễn thông Telecom Italia, bộ dữ liệu này sau đó đã được chuẩn hóa và lưu trữ trên nền tảng [Harvard Dataverse](https://dataverse.harvard.edu" \t "_new). Dữ liệu đã được thu thập trong hai tháng, từ ngày 1 tháng 11 năm 2013 đến ngày 1 tháng 1 năm 2014 ở hai thành phố là Trento và Milan. Với số lượng dữ liệu của thành phố Trento là gần 70 triệu dòng, của thành phố Milan là hơn 160 triệu dòng.

Bộ dữ liệu có 8 đặc trưng:

|  |  |  |
| --- | --- | --- |
| STT | Tên đặc trưng | Ý nghĩa |
| 1 | Square id | ID của ô vuông trong lưới không gian của thành phố |
| 2 | Time Interval | Khoảng thời gian ghi nhận mỗi 10 phút/lần. |
| 3 | Country code | Mã quốc gia của điện thoại. |
| 4 | SMS-in activity | Mức độ hoạt động của tin nhắn đến |
| 5 | SMS-out activity | Mức độ hoạt động của tin nhắn đi |
| 6 | Call-in activity | Mức độ hoạt động của cuộc gọi đến |
| 7 | Call-out activity | Mức độ hoạt động của cuộc gọi đi |
| 8 | Internet traffic activity | Mức độ sử dụng dữ liệu Internet. |

# CƠ SỞ LÝ THUYẾT

## XGBoost

XGBoost (Extreme Gradient Boosting) là một thuật toán học máy mạnh mẽ, thuộc nhóm **ensemble learning** (học kết hợp), dựa trên ý tưởng của **gradient boosting**. Nó được thiết kế để **hiệu quả, nhanh** và **có khả năng tổng quát hóa tốt**.

XGBoost được giới thiệu bởi **Tianqi Chen** và đã trở thành một công cụ phổ biến trong các cuộc thi dữ liệu (Kaggle, Data Science Bowl, v.v.) nhờ: Tốc độ huấn luyện nhanh, Hiệu quả cao với dữ liệu lớn, Hỗ trợ xử lý missing value, Giảm overfitting nhờ regularization

**Nguyên lý hoạt động**

XGBoost kết hợp nhiều cây quyết định (decision tree) theo cách **boosting tuần tự**, trong đó mỗi cây sau được huấn luyện để **sửa lỗi của cây trước**.

**Công thức dự đoán:**

Cho dữ liệu đầu vào xi​, mô hình là tổng các cây:

Với:

* K: Số cây
* ​: Một cây quyết định (cụ thể là CART - Classification and Regression Tree)
* F: Không gian của các cây

**Cấu trúc XGBoost:**

****

### 2.1.1. Decision Trees (Cây quyết định)

Trong kiến trúc của XGBoost, cây quyết định đóng vai trò là những **mô hình dự đoán đơn lẻ**, làm nền tảng cho toàn bộ hệ thống ensemble.

Về bản chất, một cây quyết định hoạt động như một lưu đồ, nơi dữ liệu được phân tách tại các nút dựa trên giá trị của các đặc trưng cụ thể. Mỗi nút đại diện cho một bài kiểm tra (phép chia) đối với một đặc trưng, và các nhánh phát sinh từ nút đó tương ứng với các kết quả có thể của bài kiểm tra.

Bằng cách đi theo các nhánh từ gốc đến lá, cây quyết định đưa ra một dự đoán. Cấu trúc này cho phép mô hình học và biểu diễn các mối liên hệ, đôi khi phức tạp, giữa các đặc trưng đầu vào và biến mục tiêu cần dự đoán.

### 2.1.2. Hàm mất mát

**Hàm mất mát** giữ một vị trí trung tâm và không thể thiếu trong quy trình huấn luyện của XGBoost. Nó đóng vai trò như một **thước đo định lượng mức độ sai lệch** giữa dự đoán của mô hình và giá trị thực tế.

Mục đích xuyên suốt của thuật toán là điều chỉnh các [tham số](https://interdata.vn/blog/tham-so-parameter-la-gi/), cụ thể là cấu trúc và trọng số của các cây quyết định, sao cho giá trị của hàm mất mát này đạt mức tối thiểu.

XGBoost hỗ trợ nhiều loại hàm mất mát khác nhau, và việc lựa chọn hàm mất mát phù hợp phụ thuộc chặt chẽ vào bản chất của bài toán đang giải quyết; ví dụ, hàm log-loss thường được dùng cho bài toán phân loại nhị phân, trong khi lỗi bình phương trung bình (Mean Squared Error – MSE) lại thích hợp cho các bài toán hồi quy.

### 2..1.2.1. Learning Rate (Tốc độ học)

**Tốc độ học ([learning rate](https://interdata.vn/blog/learning-rate-la-gi/))** là một [siêu tham số](https://interdata.vn/blog/hyperparameter-tuning-la-gi/) có tầm ảnh hưởng lớn, đóng vai trò **điều tiết trong quá trình huấn luyện của XGBoost**. Tham số này quy định “trọng lượng” hay mức độ đóng góp của mỗi cây quyết định mới được thêm vào mô hình tổng thể trong từng bước lặp.

Việc thiết lập một tốc độ học quá lớn có thể khiến mô hình học quá nhanh và trở nên quá khớp (overfitting) với dữ liệu huấn luyện, mất đi khả năng tổng quát hóa. Ngược lại, một tốc độ học quá nhỏ lại có thể làm cho quá trình hội tụ diễn ra rất chậm, đòi hỏi nhiều bước lặp hơn và có nguy cơ bị kẹt ở một giải pháp chưa phải là tốt nhất.

### Regularization (Điều chuẩn)

Cơ chế điều chuẩn (Regularization) là một thành phần then chốt được tích hợp trong XGBoost, đóng góp trực tiếp vào sự thành công của thuật toán này. Chức năng chính của nó là kiểm soát độ phức tạp của mô hình học được, ngăn chặn việc mô hình trở nên quá tinh vi và chỉ phù hợp với dữ liệu huấn luyện (tức là tránh overfitting).

XGBoost áp dụng hai kỹ thuật điều chuẩn phổ biến là L1 (còn gọi là Lasso) và L2 (còn gọi là Ridge). Các kỹ thuật này thêm “hình phạt” vào hàm mục tiêu dựa trên độ lớn của các trọng số trong mô hình, qua đó khuyến khích mô hình sử dụng các đặc trưng một cách cân bằng hơn và giảm thiểu sự phụ thuộc quá mức vào bất kỳ một hoặc một vài đặc trưng đơn lẻ nào.

### 2.1.3. Quy trình huấn luyện

Bước 1: Bắt đầu với mô hình đơn giản (thường là dự đoán hằng số).  
Bước 2: Ở mỗi vòng lặp, thêm một cây mới để giảm hàm mất mát.  
Bước 3: Cây mới học trên **gradient của hàm mất mát** (giống descent):

Hàm mục tiêu tại vòng t:

Bước 4: Tìm cấu trúc cây và giá trị lá tối ưu.

### 2.1.4. Ưu điểm

Regularization giúp giảm overfitting  
 Tự động xử lý missing value  
 Hỗ trợ song song hóa khi xây dựng cây  
 Có thể làm việc với dữ liệu lớn  
 Hỗ trợ sparse data  
 Có thể dùng cho hồi quy, phân loại, xếp hạng

### 2.1.5. Các tham số chính

* max\_depth: Độ sâu tối đa của cây.
* eta (learning\_rate): Tốc độ học.
* n\_estimators: Số lượng cây.
* subsample: Tỉ lệ mẫu dùng cho mỗi cây.
* colsample\_bytree: Tỉ lệ cột dùng cho mỗi cây.
* gamma: Minimum loss reduction khi chia node.
* lambda, alpha: L2, L1 regularization.

### 2.1.6. Ứng dụng

* Phân loại (classification): Dự báo khách hàng rời đi, phát hiện gian lận.
* Hồi quy (regression): Dự báo giá nhà, dự báo nhu cầu.
* Xếp hạng (ranking): Hệ thống gợi ý.

## **LSTM**

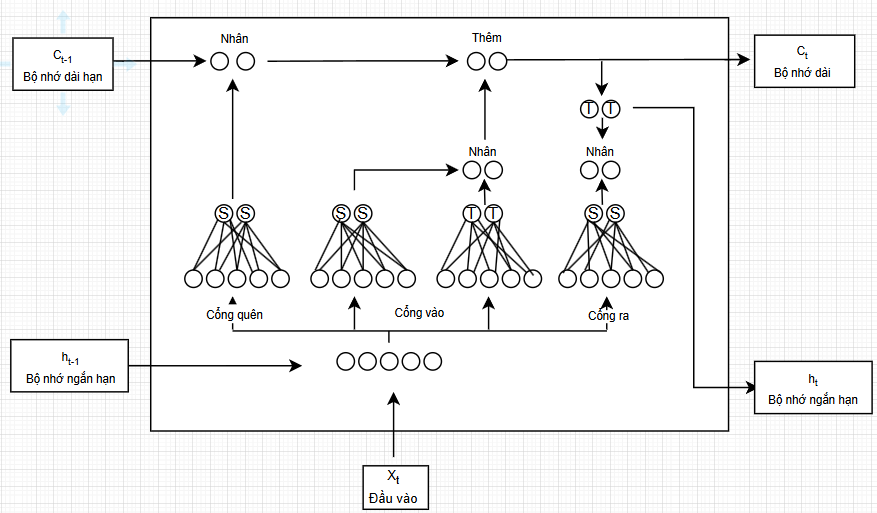
**2.2.1 Khái niệm**

Mạng bộ nhớ dài-ngắn (Long Short Term Memory networks), thường được gọi là LSTM - là một dạng đặc biệt của RNN, nó có khả năng học được các phụ thuộc xa.

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

Chìa khóa của LSTM là bộ nhớ dài hạn (cell state) một dạng giống như băng truyền. Nó chạy xuyên suốt tất cả các mắt xích (các nút mạng) và chỉ tương tác tuyến tính đôi chút. Vì vậy mà các thông tin có thể dễ dàng truyền đi thông suốt mà không sợ bị thay đổi.

**2.2.2 Cấu trúc**



****Cell state (bộ nhớ dài hạn)****: Như một "băng chuyền" chuyên chở thông tin xuyên suốt chuỗi thời gian

**Ba cổng điều khuyển**:

**+ Forget Gate** : Quyết định phần nào của cell state cũ cần bị loại bỏ, phần nào sẽ được giữ lại

**+ Input Gate và **Candidate Cell State****  : Tạo thông tin mới bằng hàm tanh và dùng thàm sigmoid để quyết định phần nào nên lưu vào bộ nhớ.

****Candidate Cell State****  : Tạo ra giá trị đề xuất cập nhật cho cell state, được “nén” trong khoảng [−1,1]bởi hàm tanh.

**+ Output Gate** : Quyết định phần nào của cell state sẽ được “trích xuất” ra làm hidden state.

**2.2.3 Các bước của mô hình**

Bước đầu tiên của LSTM là quyết định xem thông tin nào cần bỏ đi từ bộ nhớ dài hạn . Quyết định này được đưa ra bởi tầng sigmoid - gọi là “cổng quên” . Nó sẽ lấy đầu vào là  và rồi đưa ra kết quả là một số trong khoảng [0,1]  cho mỗi số trong .Đầu ra là 1 thể hiện rằng nó giữ toàn bộ thông tin lại, còn 0 chỉ rằng toàn bộ thông tin sẽ bị bỏ đi.

Bước tiếp theo là quyết định xem thông tin mới nào ta sẽ lưu vào bộ nhớ dài hạn. Đầu tiên là sử dụng một tầng sigmoid được gọi là “cổng vào” để quyết định giá trị nào ta sẽ cập nhập. Tiếp theo là một tầng tanh tạo ra một véc-tơ cho giá trị mới nhằm thêm vào cho bộ nhớ dài hạn Trong bước tiếp theo, ta sẽ kết hợp 2 giá trị đó lại để tạo ra một cập nhập cho bộ nhớ.

Cuối cùng, ta cần quyết định xem ta muốn đầu ra là gì. Giá trị đầu ra sẽ dựa vào bộ nhớ dài hạn, nhưng sẽ được tiếp tục sàng lọc. Đầu tiên, ta chạy một tầng sigmoid để quyết định phần nào của bộ ta muốn xuất ra. Sau đó, ta đưa bộ nhớ qua một hàm tanh để co giá trị nó về khoảng [−1,1], và nhân nó với đầu ra của cổng sigmoid để được giá trị đầu ra ta mong muốn.

## PatchTST (Patch Time Series Transformer)

### 2.3.1. Giới thiệu

PatchTST (Patch Time Series Transformer) là một mô hình dựa trên Transformer được đề xuất trong bài báo “A Time Series is Worth 64 Words: Long-term Forecasting with Transformers” bởi Nie et al. (2023). Mô hình này được thiết kế để giải quyết các bài toán dự báo chuỗi thời gian đa biến (multivariate time series forecasting) và học biểu diễn tự giám sát (self-supervised representation learning). PatchTST nổi bật nhờ khả năng cải thiện độ chính xác trong dự báo dài hạn so với các mô hình Transformer khác, đồng thời giảm đáng kể chi phí tính toán và bộ nhớ.

Mô hình sử dụng hai ý tưởng cốt lõi:

* Patching: Phân đoạn chuỗi thời gian thành các “miếng vá” (patches) nhỏ hơn, được sử dụng như các token đầu vào cho Transformer.
* Channel-Independence: Xử lý độc lập từng kênh (channel) của chuỗi thời gian đa biến, chia sẻ cùng embedding và trọng số Transformer giữa các kênh.

### 2.3.1. Quy trình hoạt động của mô hình PatchTST

Mô hình PatchTST bao gồm các bước xử lý chính như sau:

**Bước 1: Phân đoạn chuỗi thời gian (Patching)**

Chuỗi thời gian đầu vào (với B : kích thước batch, C : số kênh, L : độ dài chuỗi) được chia thành các patches. Mỗi patch có độ dài P và bước nhảy S:

Số lượng patches: .

Ma trận patches:, trong đó mỗi patch được biểu diễn là với

*i* = 0, 1, …, N-1

Sau đó, mỗi patch được làm phẳng thành vector và thêm positional encoding để giữ thông tin thứ tự:

**Bước 2: Channel-Independent Processing**

Mỗi kênh c (với c=1,2,...,C) được xử lý độc lập. Dữ liệu từ tất cả các kênh chia sẻ cùng embedding layer và trọng số Transformer. Ma trận đầu vào cho Transformer là: trong đó ​ là kích thước mô hình (thường là 128 hoặc 256).

**Bước 3: Transformer Encoder**

Mỗi lớp encoder trong Transformer áp dụng cơ chế multi-head self-attention (MHSA) và feed-forward network (FFN). Công thức cơ bản:

* **Multi-Head Self-Attention**: Trong đó:
  + , , là các ma trận truy vấn (query), khóa (key), và giá trị (value), với , , là trọng số học được.
  + là chiều của K (thường bằng ).
  + Kết quả từ đầu attention được nối lại: .
* **Feed-Forward Network**: FFN(X) = ​ Với ​ là trọng số, ​ là bias, và là chiều trung gian.
* **Output của một lớp encoder**:

Quá trình này lặp lại qua lớp.

**Bước 4: Tạo ra đầu ra dự báo**

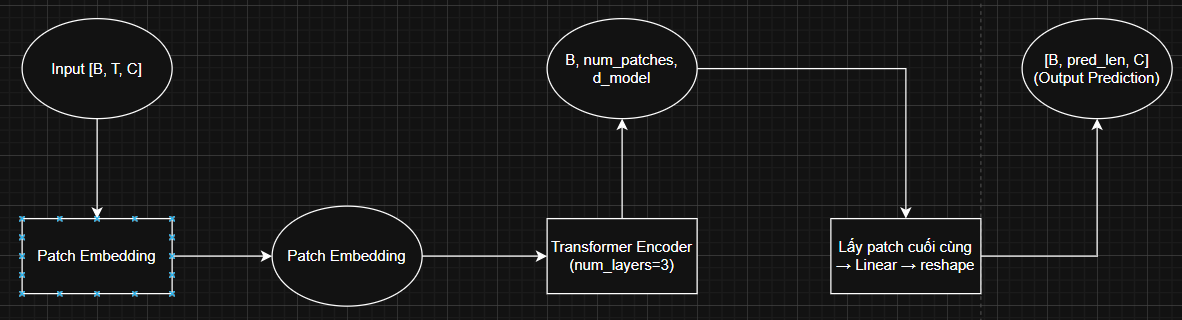
Đầu ra ​ được ánh xạ thành dự báo (với H: độ dài cửa sổ dự báo) bằng một lớp tuyến tính:

Trong đó và là trọng số và bias của lớp dự báo.

**Bước 5: Huấn luyện và tối ưu hóa**

Sử dụng hàm mất mát (thường là MSE) để so sánh dự báo với giá trị thực tế Y.

Tối ưu hóa bằng thuật toán như Adam, với các kỹ thuật tiền huấn luyện (self-supervised) hoặc fine-tuning tùy chọn.



# CHƯƠNG 3 THỰC NGHIỆM VÀ KẾT QUẢ

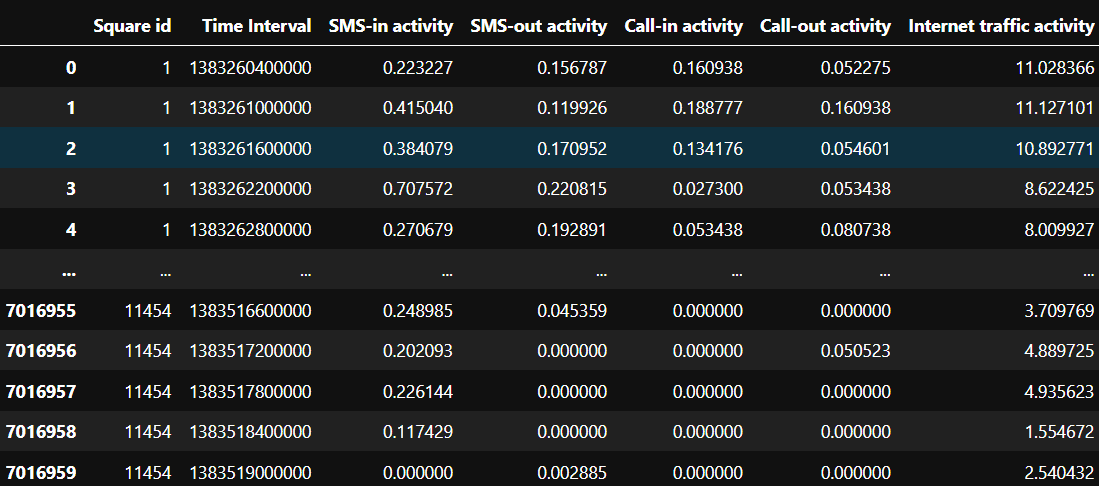
## 3.1 Tiền xử lý dữ liệu

Ban đầu dữ liệu gốc các Square id sẽ có nhiều giá trị Time Interval trùng nhau gây ra mất đi đặc trưng thời gian của dữ liệu nên cần phải gộp các giá trị có giá trị Time Interval trùng nhau bằng cách tính tổng các giá trị.

Nhóm nhận thấy rằng khi gộp các giá trị Country code không còn giống như ban đầu nữa và nó không có tác dụng khi thực hiện dự báo với đề tài của nhóm, nên sẽ xóa cột đó đi.

Dữ liệu gốc bao gồm 62 files text dữ liệu với mỗi file là 1 ngày dữ liệu. Và chúng em thấy rằng nếu gộp tất cả dữ liệu vào cùng một file sẽ rất lớn và không thể thực hiện gì được với mô hình (hơn 145 triệu dòng dữ liệu), nên chúng em chia dữ liệu thành các file batch dữ liệu (21 batch) để có thể dễ dàng thực hiện.

Dưới đây là 1 batch dữ liệu sau khi xử lý xong.

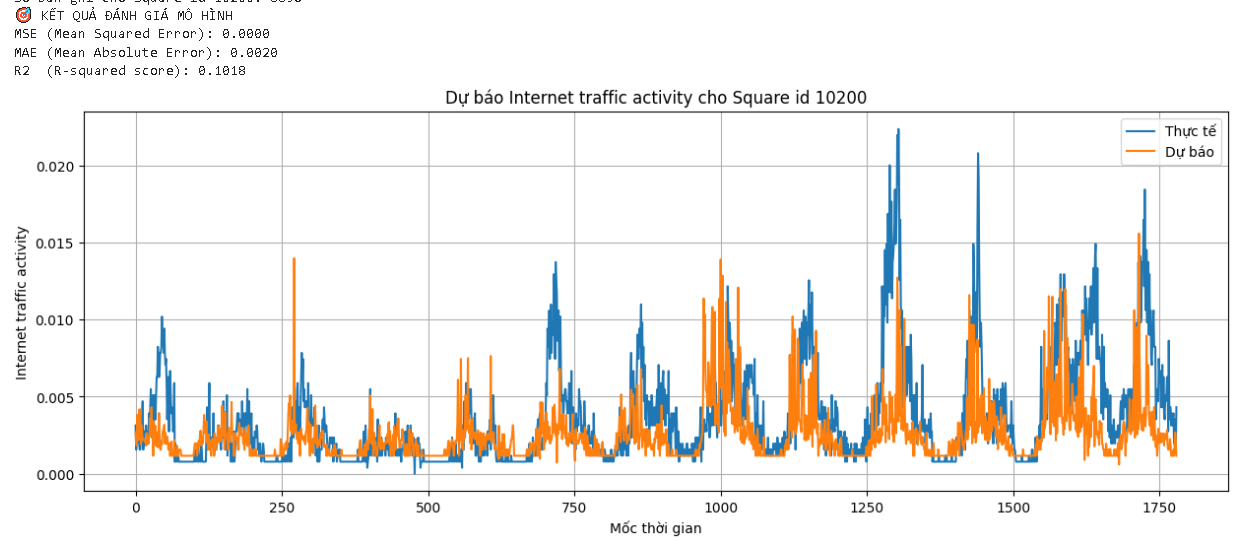


## 3.2 Kết quả thực nghiệm

### 3.2.1. XGBoost

Đầu vào :

VÍ dụ cho square ID 10200



Kết quả đánh giá cho thấy:

**MSE (Mean Squared Error)**: 0.0000  
Sai số bình phương trung bình gần bằng 0. Điều này cho thấy mô hình khớp khá tốt với dữ liệu huấn luyện/test tại hầu hết điểm

**MAE (Mean Absolute Error)**: 0.0020  
Sai số tuyệt đối trung bình thấp, dự báo nhìn chung gần đúng

**R² (R-squared score)**: 0.1018  
Giá trị R² thấp (≈ 10%), cho thấy mô hình giải thích được rất ít phương sai trong dữ liệu thực tế.

=> Mặc dù MSE và MAE nhỏ, nhưng R² thấp chứng tỏ mô hình **chưa bắt được biến động thực sự** của dữ liệu — nó có thể chỉ đang dự báo giá trị gần trung bình hoặc chưa nắm được các đỉnh, chu kỳ.

### **3.2.2. LSTM**

Đầu vào :

Input\_len : 432 số bước thời gian đầu vào (tức là 432 × 10 phút = 72 giờ dữ liệu trước).

Hidden\_dim : 64

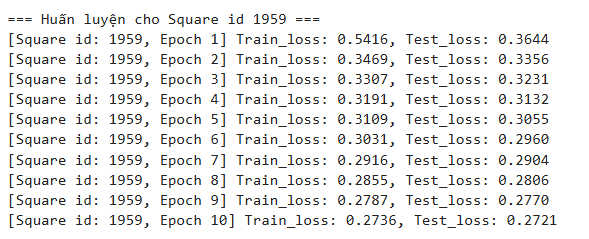
Learning\_rate : 0.01

Back\_size : 32

N\_epochs : 10 ( mỗi square id )

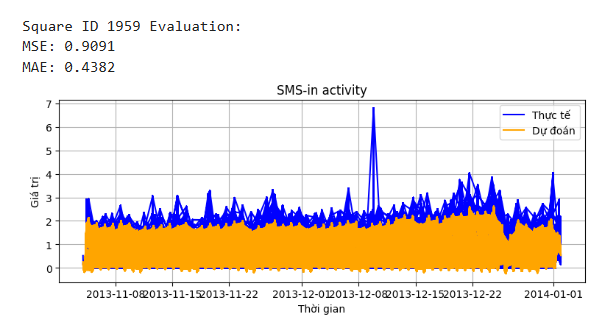
Num\_layer : 2

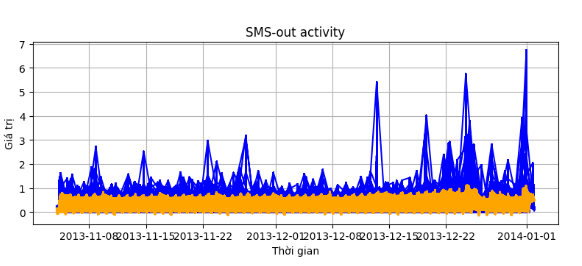
Chạy thử nghiệm trên square id 1959:

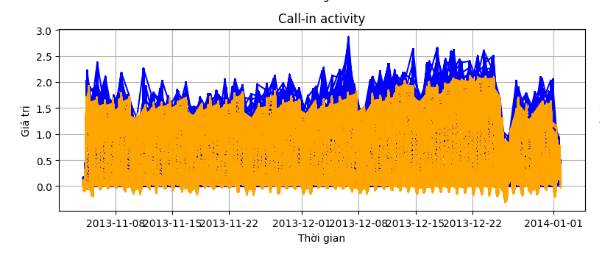


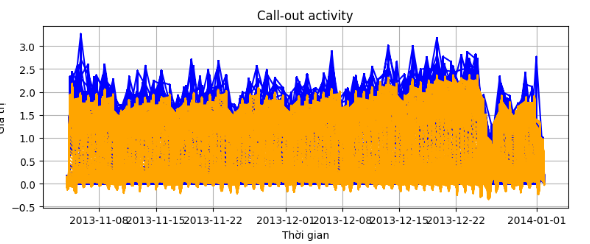
**Train loss và Test loss đều giảm đều qua các epoch** → cho thấy mô hình học được tốt, không overfit.

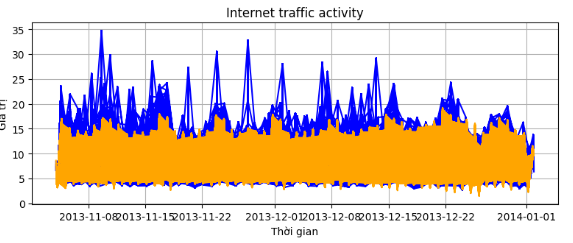
Không có hiện tượng tăng test loss ở cuối -> có thể tăng epoch để mô hình học tốt hơn











Mô hình **dự báo khá tốt các đặc trưng có xu hướng ổn định**, thể hiện rõ ở các biểu đồ như:, Call-out activity, Call-in activity

Mô hình **khó dự đoán các giá trị đột biến** như các đỉnh nhọn xuất hiện trong SMS-in, Call-in, và Internet traffic.

=> Mô hình LSTM hiện tại dự báo tốt với xu hướng dài hạn và các hoạt động ổn định, nhưng còn hạn chế với dữ liệu có độ dao động cao hoặc có nhiều điểm đột biến.

### 3.2.3. PatchTST

**Tiền xử lý đầu vào**: Với một Square ID cụ thể, mô hình truy xuất 1008 điểm dữ liệu gần nhất trước thời điểm yêu cầu (tương đương 7 ngày). Dữ liệu được chuẩn hóa theo scaler riêng cho từng trạm.

**Tách patch**: Chuỗi dài 1008 bước được chia thành các patch kích thước 32, trượt theo stride = 8.

**Embedding**: Mỗi patch được đưa qua lớp Linear để ánh xạ sang vector đặc trưng d\_model.

**Transformer Encoder**: Các vector patch được xử lý để học mối quan hệ phụ thuộc giữa các đoạn thời gian trong chuỗi.

**Dự báo**: Từ đầu ra cuối cùng, mô hình đưa ra dự đoán cho chuỗi 24 giờ tiếp theo (144 bước × 5 đặc trưng).

**Hồi quy giá trị gốc**: Kết quả đầu ra được inverse-transform (giải chuẩn hóa) để khôi phục về thang đo ban đầu.

**Đầu vào trọng số:**

Input\_len = 1008 số bước thời gian đầu vào (tức là 1008 × 10 phút = 7 ngày dữ liệu)

input\_dim = 5 (số đặc trưng)

patch\_len=32 (Độ dài của mỗi patch khi phân đoạn chuỗi thời gian)

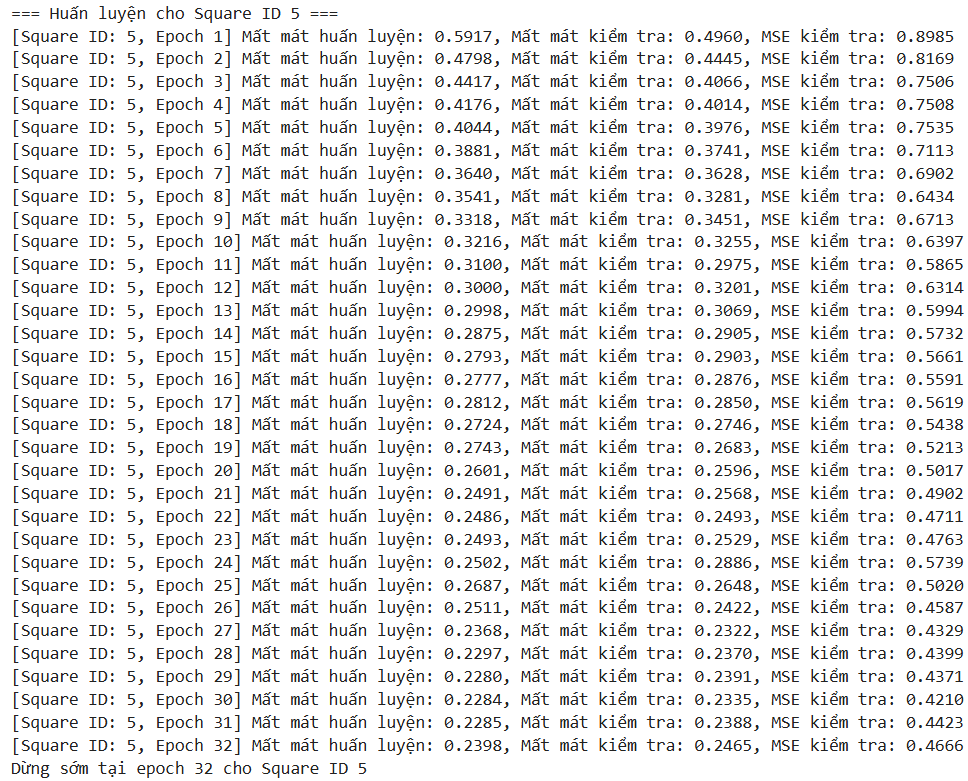
stride=8 (Bước nhảy (stride) khi trượt cửa sổ để tạo các patch. Nó xác định khoảng cách giữa các patch liên tiếp)

d\_model=64 (Kích thước của mô hình, tức là số chiều của các vector biểu diễn sau khi được ánh xạ bởi lớp tuyến tính đầu vào. Đây cũng là kích thước đầu vào/đầu ra của các lớp Transformer)

num\_layers=3 (Số lượng lớp encoder trong Transformer Backbone)

pred\_len=144 (Độ dài của chuỗi dự báo, tức là số bước thời gian mà mô hình sẽ dự đoán trong tương lai)

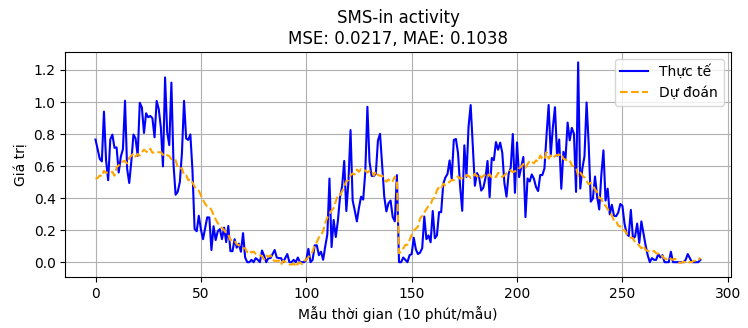
**Chạy thử nghiệm trên một square id ngẫu nhiên, mô hình sẽ chạy với tối đa 50 epoch và sẽ dừng sớm nếu như trong 5 lần huấn luyện liên tiếp không có thay đổi lớn:**



**Mất mát huấn luyện (Train loss)** giảm **đều đặn** từ 0.5917 (epoch 1) xuống còn 0.2398 (epoch 32), cho thấy mô hình học dần được các đặc trưng của dữ liệu.

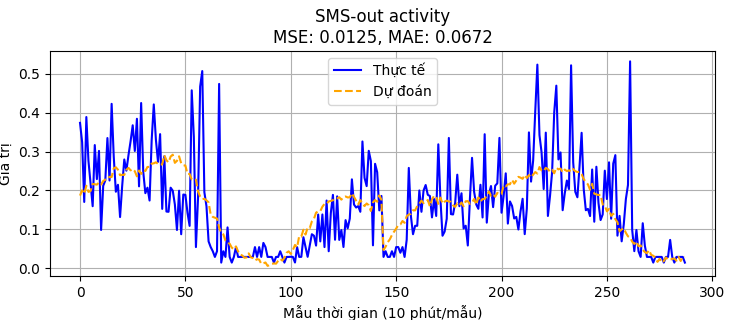
**Mất mát kiểm tra (Validation loss)** giảm từ 0.4960 → 0.2465, phản ánh sự **tổng quát hóa tốt** và không có dấu hiệu **overfitting** rõ rệt.

Mức giảm của train/val loss **ổn định đến khoảng epoch 25-32**, sau đó **dao động nhẹ**.



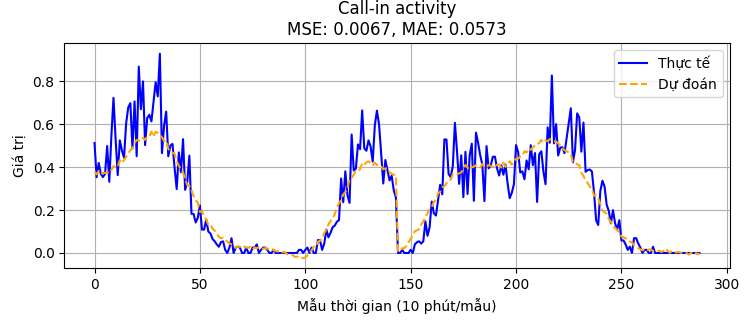
**SMS-in activity (MSE: 0.0217, MAE: 0.1038)**:

* MSE và MAE thấp, cho thấy dự báo khá chính xác.
* Đường dự đoán bám sát thực tế, nhưng có một số sai lệch ở các đỉnh cao.



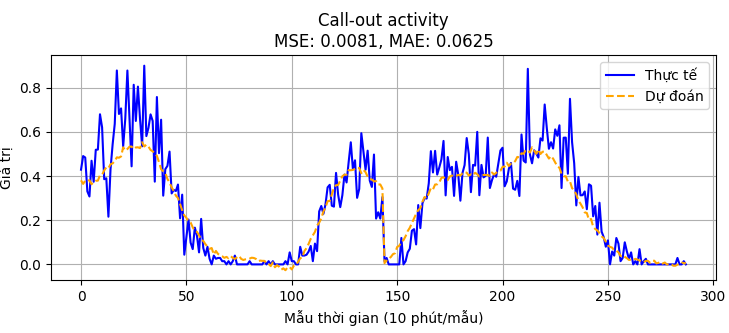
**SMS-out activity (MSE: 0.0125, MAE: 0.0672)**:

* MSE và MAE thấp hơn so với SMS-in, cho thấy hiệu suất tốt hơn.
* Đường dự đoán khá khớp với thực tế, với sai lệch nhỏ ở các dao động mạnh.



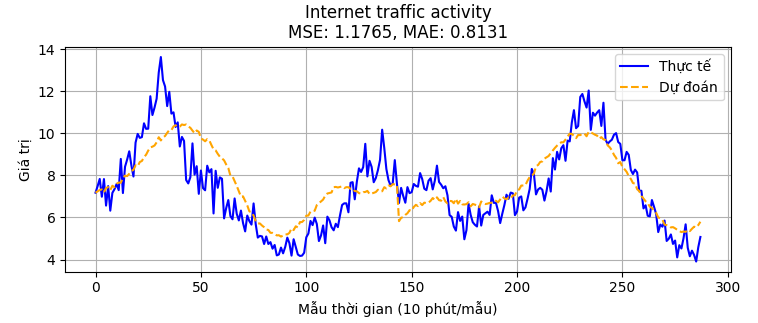
**Call-in activity (MSE: 0.0067, MAE: 0.0573)**:

* MSE rất thấp (0.0067), là giá trị tốt nhất trong 5 feature.
* MAE cũng thấp, cho thấy mô hình dự báo tốt với dữ liệu này.



**Call-out activity (MSE: 0.0081, MAE: 0.0625)**:

* MSE và MAE tương đối thấp, tương tự Call-in.
* Đường dự đoán gần khớp với thực tế, nhưng có một số điểm sai lệch nhỏ.



**Internet traffic activity (MSE: 1.1765, MAE: 0.8131)**:

* MSE và MAE cao nhất, cho thấy hiệu suất dự báo kém hơn so với các feature khác.
* Đường dự đoán có xu hướng theo xu hướng chung của thực tế nhưng không bắt được các biến động mạnh (đặc biệt ở các đỉnh cao).

**Kết luận:**

Xu hướng dự đoán:

Đường dự đoán (màu vàng) thường bám theo xu hướng chung của thực tế (màu xanh), nhưng không khớp hoàn toàn ở các điểm cực trị (đỉnh hoặc đáy).

Sai lệch lớn nhất xuất hiện ở 'Internet traffic activity', nơi mô hình có xu hướng làm mịn dữ liệu thực tế.

Ý nghĩa MSE và MAE:

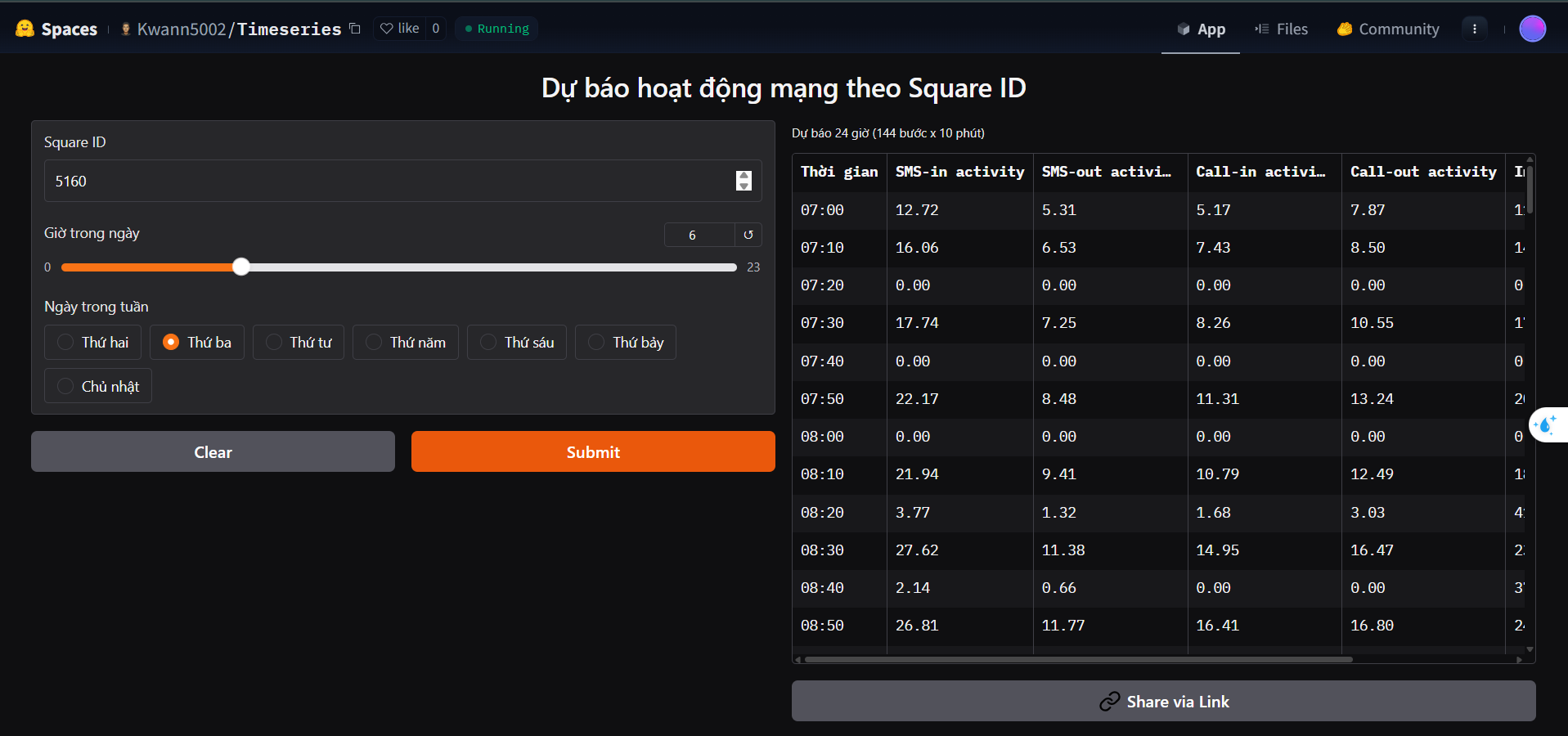
MSE cao hơn MAE trong tất cả các trường hợp, cho thấy sai lệch lớn chủ yếu tập trung ở một số điểm (các giá trị bất thường).

Giá trị MAE thấp hơn MSE cho thấy sai lệch trung bình không quá lớn, nhưng có một số điểm dự báo sai đáng kể.

### 3.2.4. Tạo giao diện

Trong quá trình xây dựng hệ thống dự báo tải mạng quy, nhóm chúng em đã triển khai một giao diện người dùng trực quan nhằm giúp người sử dụng dễ dàng tương tác với mô hình mà không cần hiểu sâu về kỹ thuật lập trình. Giao diện này được phát triển trên nền tảng Hugging Face Spaces, sử dụng thư viện Gradio – một công cụ mã nguồn mở hỗ trợ xây dựng giao diện web nhanh chóng cho các mô hình học máy.

Người dùng có thể truy cập giao diện thông qua đường dẫn: <https://huggingface.co/spaces/Kwann5002/Timeseries>



Mục tiêu chính của giao diện là cho phép người dùng:

Nhập **mã trạm viễn thông (Square ID)** tương ứng với một ô vuông địa lý (cell tower),

Chọn **mốc thời gian cụ thể** (giờ và ngày trong tuần),

Nhận kết quả **dự báo hoạt động mạng trong vòng 24 giờ tiếp theo**, tương ứng với 144 bước thời gian (mỗi bước dài 10 phút),

Trực quan hóa kết quả dự báo dưới dạng bảng.

# KẾT LUẬN

Trong bối cảnh nhu cầu viễn thông ngày càng gia tăng, đặc biệt tại các khu vực đô thị và quốc gia có mật độ thuê bao cao, việc dự báo tải mạng viễn thông đóng vai trò thiết yếu trong quản lý và tối ưu hóa tài nguyên hạ tầng mạng. Đề tài này đã thực hiện xây dựng hệ thống dự báo tải mạng quy mô toàn quốc, dựa trên phân tích dữ liệu CDRs thực tế với độ phân giải thời gian 10 phút và độ phân giải không gian theo lưới (Square ID).

Nhóm đã tiến hành khảo sát và thực nghiệm với ba mô hình học máy gồm: **XGBoost**, **LSTM** và **PatchTST (Transformer dành cho chuỗi thời gian)**. Qua các bước tiền xử lý dữ liệu, tạo batch, chuẩn hóa, huấn luyện mô hình và đánh giá kết quả, nhóm rút ra một số điểm nổi bật như sau:

Mô hình **XGBoost** tuy đơn giản và có tốc độ huấn luyện nhanh, nhưng hiệu quả chưa cao trong việc nắm bắt tính chu kỳ và sự biến động phức tạp của chuỗi thời gian viễn thông.

Mô hình **LSTM** cho kết quả tốt hơn với các đặc trưng có tính ổn định, nhưng gặp khó khăn với các tín hiệu dao động mạnh như lượng dữ liệu Internet hoặc các đỉnh hoạt động bất thường.

Mô hình **PatchTST** thể hiện ưu thế vượt trội trong dự báo chuỗi dài hạn và đa biến, đặc biệt là khả năng học xu hướng và sự ổn định của các đặc trưng như SMS và cuộc gọi.

Bên cạnh việc xây dựng mô hình, nhóm cũng phát triển **giao diện người dùng trực quan**, cho phép người dùng nhập thông tin (Square ID, giờ, ngày trong tuần) và nhận kết quả dự báo tải mạng trong 24 giờ tiếp theo. Giao diện này góp phần làm tăng tính ứng dụng thực tiễn và khả năng mở rộng của hệ thống.