TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

ĐỒ ÁN TỐT NGHIỆP

Hệ thống dự báo tài nguyên đám mây dựa trên mô hình cải tiến GAN

Nguyễn Tiến Thiện
thiennt@student.hust.edu.vn

Ngành Công nghệ thông tin và truyền thông
Chuyên ngành Hệ thống thông tin

Giảng viên hướng dẫn: TS. Nguyễn Bình Minh _____

Chữ kí của GVHD

Bộ môn: Hệ thống thông tin

Viện: Công nghệ thông tin và truyền thông

HÀ NỘI, 12/2019

ĐỀ TÀI TỐT NGHIỆP

Hệ thống dự báo tài nguyên đám mây dựa trên mô hình cải tiến GAN

Nội dung của bài nghiên cứu tập chung vào các công việc sau:

1. Thực hiện tiền xử lý biến đổi chuỗi thời gian về dạng mô hình dễ dàng học hơn.
2. Cải tiến mô hình Generative Adversarial Network - GAN để áp dụng cho bài toán phân tích chuỗi thời gian:
   (a) Xây dựng cấu trúc mạng nơ-ron hồi quy cho các thành phần của mạng GAN để học được các thông tin có sự phụ thuộc của chuỗi thời gian.
   (b) Cải tiến hàm chi phí mới trong quá trình học để áp dụng cho bài toán dự đoán dữ liệu chuỗi thời gian.
3. Thử nghiệm và so sánh các giải thuật tối ưu siêu tham số của mạng GAN đã đề xuất, đó là giải thuật tối ưu bayes và tối ưu bày đàn.
4. Áp dụng mô hình đề xuất vào bài toán dự đoán tài nguyên sử dụng của hệ thống điện toán đám mây.

Giáo viên hướng dẫn
Ký và ghi rõ họ tên

Lời cảm ơn

Đầu tiên, em xin được gửi lời cảm ơn chân thành đến các thầy giáo, cô giáo thuộc trường đại học Bách Khoa Hà Nội, đặc biệt là các thầy giáo, cô giáo thuộc Viện Công nghệ Thông tin và Truyền Thông đã tần tình dạy dỗ và trang bị cho em những kiến thức bổ ích trong những năm vừa qua.

Đồng thời em cũng xin được gửi lời cảm ơn đến TS. Nguyễn Bình Minh. Thầy là người đã chỉ dẫn tận tình, định hướng trong suốt quá trình em tham gia nghiên cứu và đặc biệt trong đồ án này.

Cuối cùng, Em xin gửi lời cảm ơn tới gia đình và bạn bè đã động viên và hỗ trợ em trong suốt những năm qua.

Tóm tắt nội dung đồ án

Điện toán đám mây đã ngày càng trở lên phổ biến và là sự lựa chọn tốt cho các doanh nghiệp muốn triển khai hệ thống tính toán. Một trong những thách thức của các hệ thống điện toán đám mây là vấn đề tự động mở rộng tài nguyên khi nhu cầu sử dụng của khách hàng thay đổi. Hiện nay phương pháp tự động mở rộng tài nguyên dựa trên giá trị ngưỡng được hầu hết các hệ thống đám mây sử dụng. Tuy nhiên, phương pháp mở rộng này vẫn gặp một số vấn đề, vì vậy phương pháp mở rộng dựa trên dựa đoán tài nguyên đã được nghiên cứu và đưa ra được các kết quả đầy hứa hẹn. Generative Adversarial Network - GAN là một mô hình học mới được khá nhiều sự quan tâm của các nhà nghiên cứu. GAN đã được áp dụng vào một số bài toán phân tích chuỗi thời gian nhưng đối với bài toán dự đoán chuỗi thời gian mô hình GAN vẫn gặp phải một số vấn đề dẫn đến kết quả dự đoán chưa được tốt. Vì vậy trong nghiên cứu này, chúng tôi đã cải tiến mô hình mạng GAN thông qua việc cải tiến hàm chi phí trong quá trình huấn luyện và sửa đổi cấu trúc của các thành phần trong mạng GAN để phù hợp hơn với bài toán phân tích chuỗi thời gian. Để các mô hình mạng có thể dễ dạng học hơn chúng tôi có thực hiện tiền xử lý, làm sạch và biến đổi chuỗi thời gian đầu vào. Đồng thời, do mạng GAN là mạng tổ hợp của 2 mạng nhỏ hơn nên số lượng tham siêu tham số rất lớn. Vì vậy chúng tôi sử dụng phương pháp tối ưu bày đàn để tối ưu các siêu tham số này để mô hình đạt kết quả tốt nhất. Thực nghiệm cho thấy sự cải tiến đáng kể độ chính xác của mô hình dự đoán và tính khả thi của mô hình khi áp dụng vào các mô hình điện toán đám mây.

Sinh viên thực hiện
Ký và ghi rõ họ tên

MỤC LỤC

DANH MỤC HÌNH VẼ

DANH MỤC BẢNG BIỂU

DANH MỤC THUẬT NGỮ VÀ TỪ VIẾT TẮT

Thuật ngữ	**Ý nghĩa**
ANNs	Mạng nơ-ron nhân tạo (Artificial Neural Networks)
BPTT	Lan truyền ngược liên hồi (Backpropagation through time)
CGAN	Conditional Generative Adversarial Networks
D	Bộ phân biệt
G	Bộ sinh
GAN	Generative Adversarial Networks
GRU	Gated Recurrent Unit
IaaS	Dịch vụ hạ tầng
JSD	Jensen–Shannon Divergence
LSTM	Long Short Term Memory
MAE	Mean absolute error
MAPE	Mean absolute percentage error
MLPs	Mạng nơ-ron nhiều lớp (Multi-layer Perceptrons)
MNIST	Modified National Institute of Standards and Technology
MSE	Mean Square Error
PaaS	Dịch vụ nền tảng
PSO	Particle Swarm Optimization
RMSE	Root mean square error
RNN	Mạng nơ-ron hồi quy (Recurrent Neural Network)
SaaS	Dịch vụ phần mềm

CHƯƠNG 1. GIỚI THIỆU ĐỀ TÀI

1.1 Đặt vấn đề

Hiện nay, điện toán đám mây hiện nay đang là một xu hướng mới cho ngành công nghiệp công nghệ thông tin. Các doanh nghiệp đã chuyển dần sang chọn giải pháp sử dụng điện toán đám mây thay cho việc tự xây dựng cơ sở hạ tầng phần cứng. Điện toán đám mây mang lại rất nhiều lợi ích như tiết kiệm chi phí, nhanh chóng và linh hoạt. Hiện nay, các nhà cung cấp dịch vụ điện toán đám mây thường dùng phương pháp mở rộng tài nguyên dựa trên chu kỳ sử dụng tài nguyên, giá trị ngưỡng hoặc khách hàng cần dùng bao nhiêu thì cấp bấy nhiêu tài nguyên. Tuy nhiên các phương pháp trên lại có những nhược điểm nhất định đó là tài nguyên cung cấp chậm, khi doanh nghiệp cần thì hệ thống chưa kịp cung cấp tài nguyên. Ngoài ra việc sử dụng tài nguyên chưa chính xác dẫn đến lãng phí do khó xác định được các thời điểm cung cấp thêm tài nguyên. Những điều trên làm giảm chất lượng dịch vụ và tăng chi phí của nhà cung cấp và người dùng.

Từ các nhược điểm của các phương pháp trên, các nhà nghiên cứu đã đưa ra một phương pháp mới có tính hiệu quả cao hơn đó là xây dựng mô hình dự báo tài nguyên sẽ sử dụng trong tương lai để có thể tự động tăng/giảm tài nguyên nếu nhu cầu tăng/giảm. Các dữ liệu thông tin tài nguyên sử dụng như CPU, bộ nhớ, lưu lượng mạng,... được theo dõi và lưu lại dưới dạng dữ liệu chuỗi thời gian. Từ dữ liệu đó các nhà nghiên cứu xây dựng các mô hình dự đoán dữ liệu chuỗi thời gian. Phương pháp này có tính hiệu quả cao, tránh được việc cung cấp tài nguyên chậm nhưng tính hiệu quả phụ thuộc vào độ chính xác của mô hình dự đoán. Nếu mô hình dự đoán có độ chính xác thấp thì có thể dẫn đến tình trạng thiếu hoặc thừa tài nguyên, làm ảnh hưởng đến chất lượng của dịch vụ cung cấp.

Trong những năm gần đây đã có rất nhiều nhà nghiên cứu đưa ra các mô hình dự đoán tài nguyên điện toán đám mây như các mô hình thống kê tuyến tính [1]. Tuy nhiên các mô hình thống kê tuyến tính lại có độ chính xác thấp. Cùng với sự phát triển của học máy và tốt hơn là học sâu, các mô hình dự đoán sử dụng học máy và học sâu cũng được áp dụng như trong [2] và [3]. Nhưng các mô hình dự đoán hiện tại chỉ học được với dữ liệu huấn luyện, và giá trị dự đoán thường là giá trị trung bình. Hiện nay, một mô hình đang rất được quan tâm của các nhà nghiên cứu, đó là mô hình Gernerative Adversarial Network - GAN. GAN đã rất thành công và được áp dụng rất nhiều để học và sinh dữ liệu ảnh. GAN là mô hình sinh được huấn luyện thông qua sự đối kháng của bộ sinh và bộ phân biệt, cả bộ sinh và bộ phân biệt đều là các mô hình mạng nơ-ron nhân tạo. Thông qua việc học đối kháng

của 2 bộ trong mạng GAN nên GAN học được phân phối của dữ liệu. Từ đó đã có một số nhà nghiên cứu đã áp dụng mô hình GAN cho bài toán chuỗi thời gian như trong [4], [5]. Tuy nhiên các mô hình GAN đó khi áp dụng vào bài toán dự đoán chuỗi thời gian vẫn cho độ chính xác trong dự đoán còn khá thấp.

Trong đồ án này, chúng tôi sử dụng mô hình Recurrent Conditional Generative Adversarial Network - RCGAN đã được nêu trong [5] để áp dụng cho bài toán dự đoán tài nguyên sử dụng trong tương lai. RCGAN là mô hình mạng GAN có điều kiện với bộ sinh và bộ phân biệt được xây dựng trên mạng nơ-ron hồi quy. Tuy nhiên khi áp dụng vào bài toán dự đoán thì do mô hình chỉ học phân phối của dữ liệu mà không chú trọng việc kết quả dự đoán có sát với thực tế hay không. Vì vậy kết quả dự đoán của mô hình chưa được tốt. Để khắc phục nhược điểm trên, chúng tôi cải tiến hàm chi phí của mô hình học là kết hợp hàm chi phí của GAN, hàm chi phí trung bình bình phương lỗi và hàm chi phí theo hướng. Sau đó, do mô hình GAN có 2 mạng nơ-ron nối tiếp nhau nên số lượng siêu tham số rất nhiều nên chúng tôi đề xuất sử dụng phương pháp tối ưu theo bầy đàn để tối ưu siêu tham số của mô hình. Ngoài ra, chúng tôi còn đưa ra các phương pháp biến đổi chuỗi thời gian để mô hình mạng có thể học dễ dàng hơn. Với hệ thống đã đề xuất chúng tôi chạy thực nghiệm trên các tập dữ liệu theo dõi tài nguyên sử dụng của cụm máy chủ Google, dữ liệu lưu lượng của EU và UK và dữ liệu về số lượng truy cập vào trang web của ban tổ chức World Cup năm 1998. Kết quả thực nghiệm cho thấy, mô hình đề xuất có độ chính xác tương đối tốt so với các mô hình khác.

1.2 Bố cục đồ án

Bố cục các phần còn lại của báo cáo đồ án tốt nghiệp này được tổ chức như sau:

- **Chương 2** sẽ giới thiệu về điện toán đám mây, bài toán tự động mở rộng trong dịch vụ điện toán đám mây và Các phương pháp dựa đoán tài nguyên. Chúng tôi còn nêu một số lý thuyết cơ bản về mô hình GAN và vấn đề tối ưu tham số của các mô hình học máy.
- **Chương 3** sẽ mô tả chi tiết về hệ thống và mô hình đề xuất để giải quyết bài toán dự đoán tài nguyên sử dụng.
- **Chương 4** sẽ trình bày kết quả thực nghiệm của mô hình hệ thống đã đề xuất và đưa ra các đánh giá đối với các kết quả đã có.
- **Chương 5** sẽ đưa ra kết luận cho những kết quả đạt được trong đồ án và định hướng phát triển cho bài toán.

CHƯƠNG 2. TỔNG QUAN CÁC NGHIÊN CỨU LIÊN QUAN VÀ CƠ SỞ LÝ THUYẾT

2.1 Tổng quan về điện toán đám mây

Theo phương pháp truyền thống trước đây, để triển khai một hệ thống mới chúng ta phải triển khai từ cơ sở hạ tầng đến triển khai hệ thống phần mềm. Việc triển khai như vậy khiến cho nhà phát triển tốn rất nhiều thời gian, công sức cũng như tiền bạc để triển khai cũng như bảo trì hệ thống cơ sở hạ tầng. Năm 2007, một thuật ngữ mới được đưa ra là **điện toán đám mây**, thuật ngữ này khái quát lại hướng đi của cơ sở hạ tầng thông tin vốn đã và đang diễn ra trong mấy năm qua. Điện toán đám mây là một mô hình tính toán, nó làm thay đổi phương pháp triển khai hệ thống truyền thống. Mô hình điện toán đám mây là nơi tập chung rất nhiều hệ thống được kết nối với nhau trong mạng riêng tư hoặc mạng công khai, để cung cấp khả năng mở rộng cơ sở hạ tầng một cách dễ dàng cho các ứng dụng, dữ liệu lưu trữ. Với lợi ích của công nghệ này, chi phí tính toán, triển khai ứng dụng, lưu trữ và phân phối nội dung được giảm khá nhiều.

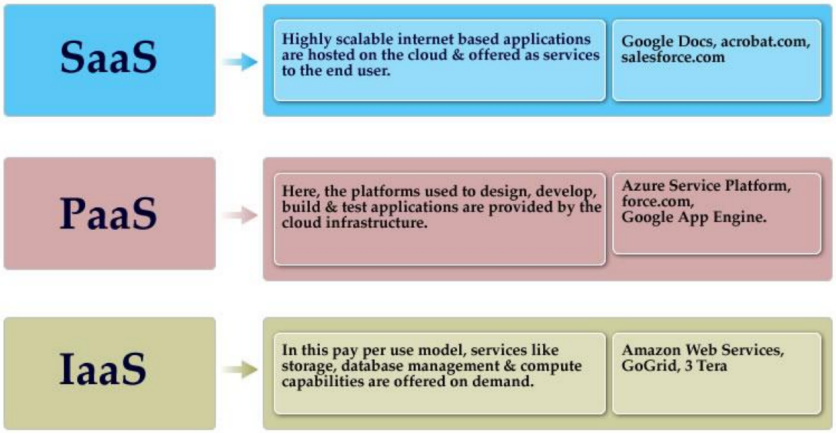
Ý tưởng chính của điện toán đám mây là dựa trên một nguyên lý rất cơ bản là khả năng tái sử dụng. Sự khác biệt của điện toán đám mây so với mô hình truyền thống về các nội dung tính toán lưới, tính toán phân tán, tính toán linh hoạt hoặc tính toán tự động được mở rộng ra quy mô lớn hơn, không còn chỉ trong một tổ chức.

Các mô hình cung cấp dịch vụ điện toán đám mây có thể chia thành 3 nhóm được thể hiện trong hình 2.1:

- **Dịch vụ hạ tầng (Infrastructure as a Service - IaaS)**: IaaS cung cấp nơi lưu trữ và tính toán cơ bản như một dịch vụ tiêu chuẩn trên môi trường mạng. Máy chủ, hệ thống lưu trữ, trang thiết bị mạng, không gian trung tâm dữ liệu... được gộp chung lại và sẵn sàng đáp ứng xử lý các công việc. Khách hàng có thể triển khai các phần mềm, môi trường như hệ điều hành, cơ sở dữ liệu của họ trên các cơ sở hạ tầng này. Một số ví dụ điển hình như Amazon, GoGrid,....
- **Dịch vụ nền tảng (Platform as a Service - PaaS)**: Đây là mô hình dịch vụ có mức độ cao hơn IaaS. Trong môi trường này, các môi trường phát triển được đóng gói vào trong một dịch vụ. Khách hàng không cần quan tâm đến môi trường chạy phần mềm mà chỉ cần cấu hình môi trường và triển khai phần mềm trên đó. Để đáp ứng các yêu cầu quản lý và mở rộng của phần mềm,

PaaS cung cấp các tổ hợp định nghĩa từ trước của hệ điều hành và các phần mềm trên máy chủ như LAMP platform (Linux, Apache, MySql và PHP), Google's App Engine, Force.com....

- **Dịch vụ phần mềm (Software as a Service - SaaS)**: Đây là mô hình dịch vụ ở mức độ cao nhất, một ứng dụng hoàn thiện được triển khai bởi nhà cung cấp và chạy trên môi trường đám mây. Nhiều khách hàng có thể sử dụng dịch vụ cùng lúc, và họ cũng không cần trả phí cho hệ thống máy chủ, cũng như giấy phép phần mềm, trong khi chi phí của bên cung cấp dịch vụ cũng nhỏ do chỉ có một ứng dụng cần được triển khai và bảo trì. Hiện nay, SaaS được phát triển bởi khá nhiều công ty như Google (gmail, docs), Microsoft (outlook, office online),....



Hình 2.1: Các mô hình dịch vụ trong điện toán đám mây. Nguồn [6]

Để triển khai dịch vụ, khách hàng có thể chọn các phương pháp triển khai khác nhau phù hợp với nhu cầu.

- Đám mây công khai (Public cloud): mô hình này được sở hữu và vận hành bởi một tổ chức bên thứ ba, dịch vụ được cung cấp trên mạng internet và mọi người muốn sử dụng điều có thể sử dụng chúng.
- Đám mây nội bộ (Private cloud): các hạ tầng đám mây được cung cấp chỉ trong một tổ chức. Mục đích của mô hình này là để đảm bảo tính bảo mật dữ liệu và kiểm soát truy cập.
- Đám mây lai (Hybird cloud): là mô hình kết hợp cả hai mô hình đám mây công khai và đám mây nội bộ. Với đám mây lai, các dịch vụ của bên cung cấp có thể sử dụng dịch vụ đám mây của bên thứ ba khác, vì vậy nó giúp tăng khả năng tính toán. Đám mây lai có thể cung cấp theo yêu cầu, tăng khả năng mở rộng của hệ thống. Mô hình này có thể tăng tính nội bộ trong đám mấy công khai thông qua việc quản lý bất kì sự gia tăng nào trong khối lượng công việc.

Các mô hình điện toán đám mây đem lại cho người dùng rất nhiều lợi ích:

- Giảm chi phí: Khi người dùng sử dụng dịch vụ đám mây, người dùng sẽ không tồn chi phí cho các công việc về triển khai vào bảo trì hệ thông cơ sở hạ tầng. Từ đó, người dùng được giảm rất nhiều chi phí để triển khai hệ thống của mình.
- Tăng khả năng lưu trữ: dịch vụ đám mây giúp người dùng có thể lưu trữ một lượng dữ liệu rất lớn. Và người dùng cũng không cần quan tâm đến dữ liệu được lưu trữ trên các phần cứng nào, ở đâu.
- Tăng khả năng mở rộng: khi người dùng có nhu cầu tăng tài nguyên, do không phụ thuộc vào cơ sở hạ tầng bên dưới nên việc mở rộng chỉ cần gửi yêu cầu đến bên cung cấp dịch vụ.

Mặc dù điện toán đám mây tăng trưởng một cách rất nhanh nhưng mô hình này vẫn có những thách thức.

- Bảo mật dữ liệu: Dữ liệu của người dùng được đưa lên hệ thống điện toán đám mây có thể được lưu chung trong một tài nguyên lưu trữ. Vì vậy, có thể có những truy cập trái phép. Từ đó, các nhà cung cấp dịch vụ điện toán cần đảm bảo tính bảo mật của khách hàng.
- Đảm bảo tính sẵn có và phục hồi dữ liệu: Các dữ liệu được người dùng đưa lên các hệ thống điện toán đám mây sẽ được các nhà cung cấp lưu trữ và bảo vệ. Các nhà cung cấp cũng cần có cơ chế phục hồi dữ liệu khi phần cứng hỏng nhằm đảm bảo tính sẵn có của dữ liệu.
- Quản lý khả năng cung cấp: Mặc dù có nhiều nhà cung cấp dịch vụ đám mây, nhưng việc quản lý của nền tảng và cơ sở hạ tầng vẫn còn khá đơn giản. Các tính năng tự động mở rộng là những yêu cầu khá quan trọng đối với nhiều doanh nghiệp. Nó có tiềm năng rất lớn để cải thiện khả năng mở rộng và cân bằng tải.
- Các điều lệ và luật cấm: Ở một số nước European, chính phủ không cho phép thông thông tin dữ liệu cá nhân của khách hàng và các thông tin nhạy cảm khác được đặt một cách vật lý ở nước ngoài. Do yêu cầu đó, các nhà cung cấp dịch vụ đám mây cần phải thiết lập các trung tâm dữ liệu hoặc các điểm lưu trữ trong nước.

2.2 Tự động mở rộng trong điện toán đám mây

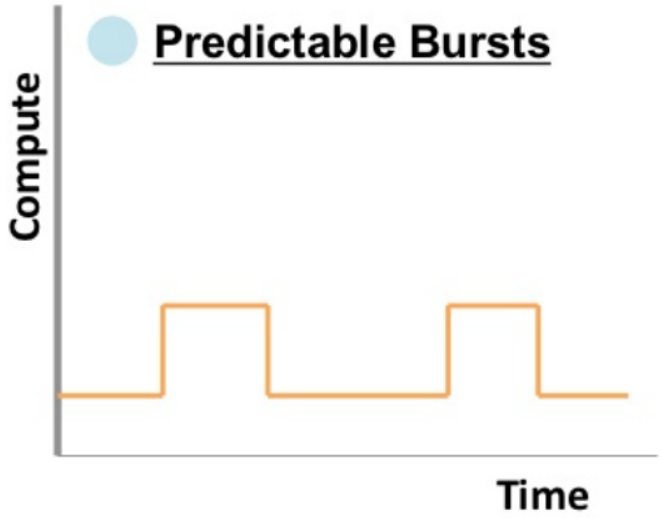
Khi một doanh nghiệp phát triển dịch vụ điện toán đám mây thì một bài toán lớn mà doanh nghiệp phải nghĩ đến là bài toán tự động mở rộng tài nguyên. Nếu

hệ thống cung cấp thừa tài nguyên (tài nguyên cung cấp lớn hơn nhu cầu sử dụng) thì doanh nghiệp sẽ tốn nhiều chi phí để duy trì trong khi thực tế nhu cầu nhỏ hơn nhiều so với cung cấp. Ngược lại, nếu hế thống cung cấp thiều tài nguyên sẽ dẫn đến hệ thống của người dùng chậm chạp, có thể dẫn đến ngừng hệ thống, từ đó doanh nghiệp cung cấp dịch vụ có thể mất khách hàng. Vì vậy, doanh nghiệp cung cấp dịch vụ cần đưa ra các phương pháp để hệ thống có thể tự động mở rộng - thu hẹp tài nguyên cung cấp.

Để giải quyết vấn đề cung cấp tài nguyên đã nêu trên, chúng ta cần có một cơ sở hạ tâng có khả năng mở rộng, thu hẹp một cách tự động, và có các chiến lược tự động mở rộng phù hợp. Hiện nay, có 3 chiến lược tự động mở rộng chính là Periodicity, Threshold, và Prediction.

2.2.1 Phương pháp chu kì (periodictiy)

Trong quá trình cung cấp tài nguyên, nhà cung cấp theo dõi nhu cầu tiêu dùng của khách hàng và phát hiện ra được các chu kỳ tiêu dùng tài nguyên. Các chu kỳ này có thể theo phút, giờ, ngày hay tháng được mô tả trong hình 2.2. Thông qua đặc điểm này nhà cung cấp đưa ra một chiến lược tăng hay giảm tài nguyên phù hợp. Nhược điểm lớn nhất của phương pháp này là không thể đáp ứng được các yêu cầu tài nguyên ngay tức thì từ ứng dụng.



Hình 2.2: Phương pháp chu kỳ. Nguồn [7]

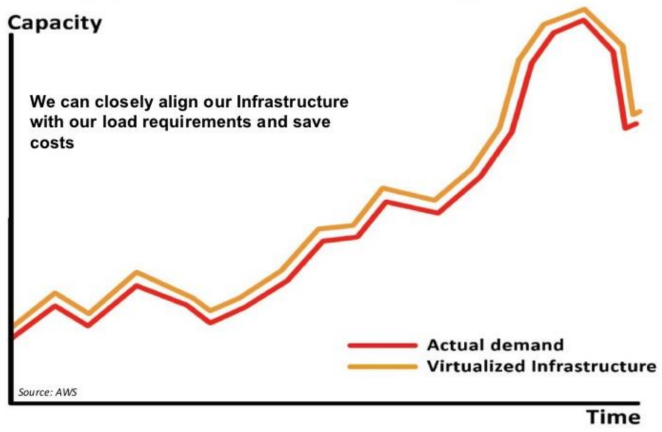
2.2.2 Phương pháp dựa theo ngưỡng (thresholds)

Phương pháp này sử dụng một vài giá trị ngưỡng thiết lập trước để xác định thời điểm nào cần tăng hay giảm tài nguyên. Nhà cung cấp thiết lập trước các ngưỡng trên và các ngưỡng dưới. Khi lượng tài nguyên sử dụng của hệ thống đạt tới giá trị ngưỡng trên, hệ thông sẽ tự động mở rông thêm máy ảo để đáp ứng nhu

cầu của khách hàng và ngược lại nếu lượng tài nguyên sử dụng chạm tới giá trị ngưỡng dưới thì hệ thống tự động giải phóng bớt máy ảo mà không ảnh hưởng đến nhu cầu của người dùng. Tuy nhiên, phương pháp này có một số nhược điểm như là khó xác định được giá trị ngưỡng phù hợp, có thời gian trễ khi mở rộng do để mở rộng thêm máy ảo cần một khoảng thời gian, do đó không đáp ứng được nhu cầu ngay lập tức.

2.2.3 Phương pháp dự đoán (prediction)

Đây là một phương pháp có tính hiệu quả cao, đó là dự đoán trước tài nguyên sẽ sử dụng của hệ thống trong tương lai. Phương pháp này áp dụng như trong bài toán phân tích chuỗi thời gian, sử dụng các dữ liệu thu thập các tài nguyên đã sử dụng trong thời gian trước, sau đó dự đoán các tài nguyên sẽ sử dụng trong thời gian tới. Hình 2.3 mô tả tài nguyên sử dụng thực tế và tài nguyên sử dụng dự đoán. Từ hình ta có thể thấy 2 đường này rất sát nhau, từ đó nhà cung cấp sẽ biết được tương lai nhu cầu sử dụng tài nguyên của khách hàng. Từ đó có thể giúp hệ thống biết khi nào tăng hay giảm tài nguyên. Tuy nhiên, tính hiệu quả của phương pháp này phụ thuộc phần lớn vào mô hình dự đoán. Vì vậy cần phải xây dựng một mô hình dự đoán hiệu quả. Hiện nay đã có một số bài nghiên cứu dựa trên phương pháp này áp dụng cho hệ thống điện toán đám mây như sử dụng mô hình tuyến tính [1], sử dụng mô hình học máy [8], hay tốt hơn là sử dụng mô hình học sâu [3]. GAN là một mô hình mới trong những năm gần đây và đã có rất nhiểu thành công trên nhiều lĩnh vực. Tuy nhiên chưa có bài nghiên cứu nào áp dụng mô hình GAN cho bài toán tự động mở rộng trong hệ thống điện toán đám mây.



Hình 2.3: Phương pháp dự đoán trước tài nguyên sử dụng. Nguồn [7]

2.3 Các phương pháp dự đoán chuỗi thời gian

2.3.1 Tổng quan

Trong phần này, chúng tôi sẽ trình bày các nghiên cứu đã có liên quan đến dự đoán tài nguyên. Phần hiệu quả của mô hình được thể hiện thông qua tính chính

xác và hữu dụng. Với một chuỗi thời gian có dạng

$$x_0, x_1, x_2, ..., x_n$$

là các giá trị được theo dõi và ghi lại và các giá trị này cách đều nhau về mặt thời gian. Mô hình dựa đoán lấy đầu vào là chuỗi các k giá trị gần nhất với thời điểm dự đoán $x_{t-k+1}, x_{t-k+2}, x_{t-k+3}, ..., x_t$ và đầu ra dự đoán sẽ là giá trị x_{t+1}. Bài toán dựa đoán chuỗi thời gian đã được đông đảo các nhà nghiên cứu tham gia và đã có nhiều thành tựu. Mô hình đơn giản nhất là các mô hình thống kê tuyến tính như trong [8] đã sử dụng các mô hình tự hồi quy (Autoregressive - AR), trung bình trượt (Moving Average - MA), tự hồi quy và trung bình trượt.

Trong các năm gần đây, mạng nơ-ron nhân tạo đã phát triển mạnh mẽ và được áp dụng rộng rãi, được biết đến nhiều nhất là mạng nơ-ron truyền thẳng (Feed-forward neural network - FFNN) sử dụng giải thuật lan truyền ngược để cập nhật trọng số của mạng. Trong nghiên cứu [9] F. Jabari và các cộng sự đã ứng dụng mạng nơ-ron nhiều tầng để dự đoán năng lượng bức xạ của mặt trời sử dụng bộ dữ liệu năng lượng bức xạ ở San Jose, Califolia từ 2005 đến 2010. Gần đây các mô hình học sâu được phát triển mạnh mẽ với khả năng học các bài toán phức tạp một cách hiệu quả. Trong các mô hình mạng học sâu thì mạng nơ-ron hồi quy được áp dụng rất nhiều trong bài toán phân tích chuỗi thời gian do khả năng học mối quan hệ theo thời gian giữa các điểm dữ liệu. Mạng nơ-ron hồi quy cũng tương tự như mạng truyền thẳng nhưng được phát triển để có thể nhớ được các thông tin trong quá khứ. Để cải thiện được khả năng nhớ dài hạn của RNN, Hochreiter và các cộng sự đã đề xuất mô hình mạng long short term memmory - LSTM, và đây cũng là mô hình được dùng phổ biến nhất trong lớp mô hình mạng nơ-ron hồi quy. Mô hình LSTM đã được áp dụng trong hệ thống đám mây như trong [10] đã áp dụng mô hình LSTM cho việc dự đoán tải trên các trung tâm dữ liệu đám mây sử dụng bộ dữ liệu NASA. Cùng với LSTM, mô hình mạng nơ-ron hồi quy Gated Recurrent Unit - GRU cũng đã được đề xuất bởi Cho, et al. vào năm 2014 với ưu điểm số lượng tính toán ít hơn mạng LSTM. GRU cũng đã được áp dụng trong [11] cho bài toán phân tích các giá trị theo dõi của hệ thống Internet of Thing.

2.3.2 Các phương pháp tuyến tính

Chuỗi thời gian trong các mô hình tính tính dưới đây được biểu diễn dưới dạng $X = x_1, x_2, ..., x_n$. Trong đó, x_t là giá trị tại thời điểm t của chuỗi thời gian X.

a) Mô hình tự hồi quy (Auto Regressive)

Giả sử ta có các giá trị $x_1, x_2, ..., x_{t-1}$ và cần dự đoán giá trị x_t, mô hình hồi quy giải thiết x_t là một tổ hợp tuyến tính của các giá trị từ 1 đến $t-1$. Mô hình hồi quy bậc p, ký hiệu $AR(p)$ được biểu diễn theo công thức như sau:

$$x_t = c + \sum_{i=1}^{p} \phi_i x_{t-i} + \epsilon$$

Trong đó:

$$\begin{cases} c & \text{là hằng số} \\ p & \text{là bậc của mô hình} \\ \phi_i & \text{là tham số thứ i của mô hình} \\ \epsilon & \text{là sai số ngẫu nhiên, có giá trị trung bình bằng 0} \\ & \text{và phương sai không đổi } \sigma^2 \end{cases}$$

Ví dụ với $p = 1$, giá trị tại thời điểm t chỉ phụ thuộc vào giá trị tại thời điểm trước đó $t-1$, ta có mô hình $AR(1)$:

$$x_t = c + \phi x_{t-1} + \epsilon$$

b) Mô hình trung bình trượt (Moving Average)

Mô hình trung bình trượt là mô hình xây dựng dựa trên giá trị trung bình của chuỗi và sai số ngẫu nhiên ở thời điểm hiện tại và q thời điểm trước đó. Mô hình trung bình trượt bậc q, ký hiệu $MA(q)$, được thể hiện theo công thức như sau:

$$x_t = \mu + \sum_{t=1}^{q} \theta_i \epsilon_{t-i} + \epsilon_t$$

Trong đó:

$$\begin{cases} \mu & \text{là giá trị trung bình của chuỗi thời gian} \\ p & \text{là bậc của mô hình} \\ \phi_i & \text{là tham số thứ i của mô hình} \\ \epsilon & \text{là sai số ngẫu nhiên} \end{cases}$$

Với $q = 1$, ta có mô hình trung bình trượt bậc nhất $MA(1)$ như sau:

$$x_t = \mu + \theta\epsilon_{t-1} + \epsilon_t$$

c) Mô hình tự hồi quy và trung bình trượt (Autoregressive Moving Average - ARMA)

Đây là mô hình kết hợp của cả 2 mô hình tự hồi quy và trung bình trượt. Với mô hình ARMA bậc p và q, kí hiệu ARMA(p, q), ta có công thức:

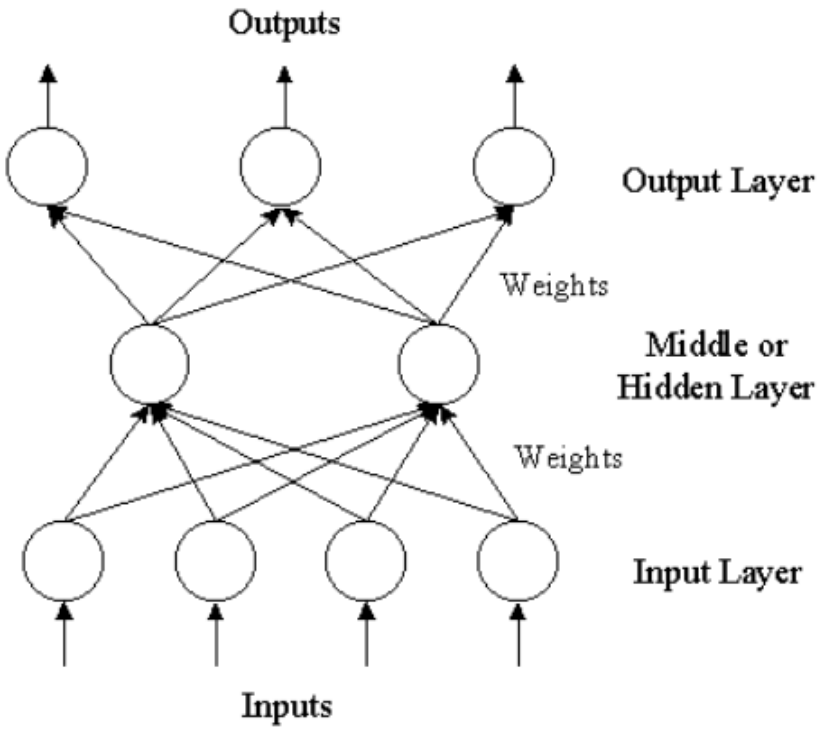
$$x_t = c + \theta_t + \sum_{i=1}^{p} \phi_i x_{t-i} + \sum_{t=1}^{q} \theta_i \epsilon_{t-i}$$

2.3.3 Mô hình mạng nơ-ron nhân tạo

Mạng nơ-ron nhân tạo (Artificial Neural Networks - ANNs) là một mô hình được đưa ra dựa trên ý tưởng mô phỏng lại não người. ANNs được sử dụng phổ biến hiện nay do mô số đặc trưng sau:

- ANNs có thể tự điều chỉnh, thích ứng mà không cần biết trước thông tin phân phối thông kê trên dữ liệu.
- ANNs là mô hình phi tuyến, từ đó nó có thể học được các mẫu dữ liệu phức tạp - dữ liệu thường có trong thực tế.
- ANNs là hàm xấp xỉ tổng quát, nó có thể xấp xỉ các hàm liên tục. Từ đó, ANNs đạt được độ chính xác cao hon.

Mạng nơ-ron đơn giản và được sử dụng rộng rãi là Mạng nơ-ron nhiều lớp (Multi-layer Perceptrons - MLPs). Mô hình được đặc trưng bởi một mạng gồm có các tầng khác nhau. Tầng đầu tiên được gọi là tầng đầu vào (input layer), đây là tầng nhận thông tin dữ liều đầu vào. Tầng cuối cùng là tầng đầu ra (output layer), đây là tầng đưa ra thông tin mong muốn học. Giữa tầng đầu vào và tầng đầu ra là các tầng được gọi là tầng ẩn (hidden layers), các tầng này chứa thông tin của mô hình. Mỗi tầng có nhiều nút tương ứng với các nơ-ron, các nút ở các tầng kề nhau thường được kết nối toàn bộ với nhau. Hình 2.4 mô tả mạng nơ-ron với 1 tầng ẩn.



Hình 2.4: Cấu trúc mạng nơ-ron nhiều lớp. Nguồn [12]

Với các giá trị đầu vào $X = x_1, x_2, ..., x_n$ mạng nơ-ron sẽ biểu diễn xấp xỉ hàm quan hệ giữa đầu vào X và đầu ra thực tế y

$$y = f(X) = f(x_1, x_2, ..., x_n)$$

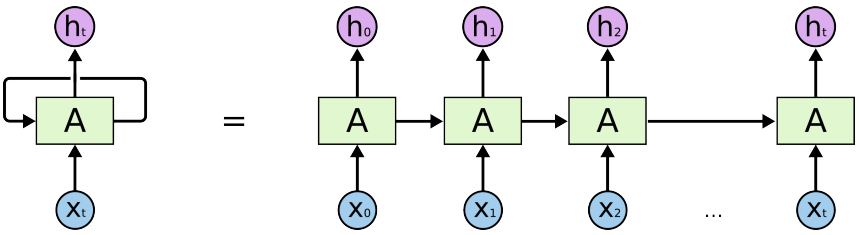
Đối với bài toán chuỗi thời gian thì mạng nơ-ron sẽ học mối quan hệ giữa giá trị hiện tại và các giá trị trong quá khứ $x_t = f(x_1, x_2, ..., x_{t-1})$.

2.3.4 Mô hình học sâu

a) Mạng nơ-ron hồi quy

Dữ liệu thực tế phụ thuộc rất nhiều vào các dữ liệu trong quá khứ, đặc biệt với dữ liệu chuỗi thời gian thì yếu tố này rất quan trọng. Với mô hình mạng nơ-ron nhiều lớp thông thường thì không thể biểu diễn yếu tố đó. Từ đó Mạng nơ-ron hồi quy (Recurrent Neural Network - RNN) sinh ra để giải quyết vấn đề trên. Mạng này chứa các vòng lặp bên trong cho phép thông tin có thể lưu lại. Hình 2.5 mô tả mạng nơ-ron hồi quy với đầu vào x_t và đầu ra là h_t. Một vòng lặp cho phép thông tin có thể được truyền từ bước này qua bước khác của mạng nơ-ron.

Mạng nơ-ron hồi quy cũng giống với mạng nơ-ron truyền thẳng, cũng có các tầng ẩn. Tuy nhiên, RNN khác mạng truyền thẳng ở chỗ các tầng ẩn có kết nối vòng lại chính nó, vì vậy RNN cho phép trạng thái của các tầng ẩn tại một thời



Hình 2.5: Mạng nơ-ron hồi quy. Nguồn [13]

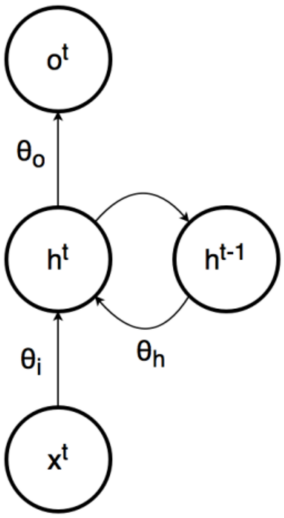
điểm được sử dụng lại như là một đầu vào cho các tầng ẩn ở thời điểm tiếp theo. Và nếu được huấn luyện, các trạng thái ẩn này học được các thông tin về mối quan hệ thời gian giữa chuỗi đầu vào và chuỗi đầu ra. RNN được gọi là mạng nơ-ron hồi quy vì chúng thực hiện việc tính toán (xét trên trọng số, bias và hàm kích hoạt) là như nhau cho mỗi phân tử của chuỗi đầu vào. Sự khác nhau giữa các đầu ra đến từ sự khác nhau giữa các phần tử của chuỗi đầu vào và từ trạng thái ẩn khác nhau. Đầu ra phụ thuộc vào giá trị phần tử đầu vào hiện tại và giá trị cuối cùng của trạng thái ẩn:

$$o_t = f(h_t; \theta)$$
$$h_t = g(h_{t-1}, x_t; \theta)$$

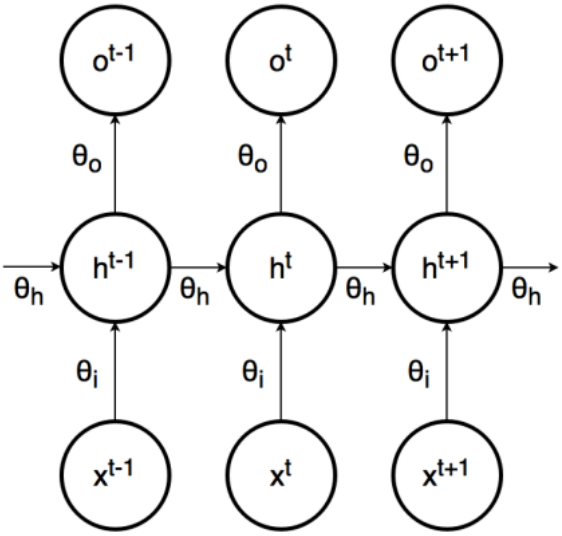
Trong đó:

$\begin{cases} o_t & \text{là đầu ra của RNN tại thời điểm } t \\ x_t & \text{là đầu vào của RNN tại thời điểm } t \\ h_t & \text{là trạng thái của các tầng ẩn tại thời điểm } t \\ \theta & \text{là các tham số của mô hình} \end{cases}$

Hình 2.6 dưới đây thể hiện sơ đồ đơn giản thể hiện mối quan hệ giữa 3 loại biến trong đồ thị tính toán của RNN.



Hình 2.6: Sơ đồ mô hình của một mạng hồi quy. Nguồn [14]



Hình 2.7: Mạng nơ-ron hồi quy đã trải ra. Nguồn [14]

Biểu thức thứ 1 cho ta thấy rằng, khi nhận vào các tham số θ, đầu ra tại thời điểm t chỉ phụ thuộc vào trạng thái của các tầng ẩn tại thời điểm t, phần này khá giống với mạng truyền thẳng. Biểu thức thứ 2 cho ta thấy rằng, khi nhận vào cùng các tham số θ, tầng ẩn tại thời điểm t phụ thuộc vào tầng ẩn tại thời điểm $t-1$ và đầu vào tại thời điểm t. Biểu thức thứ 2 chứng minh rằng RNN có thể nhớ được quá khứ thông qua truyền h_{t-1} để tính toán h_t.

Huấn luyện mạng hồi quy rất giống huấn luyện mạng truyền thẳng. Các giải thuật lan truyền ngược của mạng truyền thẳng đều làm việc tốt cho RNN, gọi là Lan truyền ngược liên hồi (Backpropagation through time - BPTT). BPTT hoạt động khi áp dụng giải thuật lan truyền ngược với RNN đã trải ra (hình 2.7). Khi trải RNN ra, ta thấy nó tưng tự mạng nơ-ron truyền thẳng với tất cả phần tử o_t như là tầng đầu ra, tất cả phần tử x_t từ chuỗi đầu vào x như là tầng đầu vào, toàn bộ chuỗi đầu vào x và chuỗi đầu ra o đều cần có trong úa trình huấn luyện.

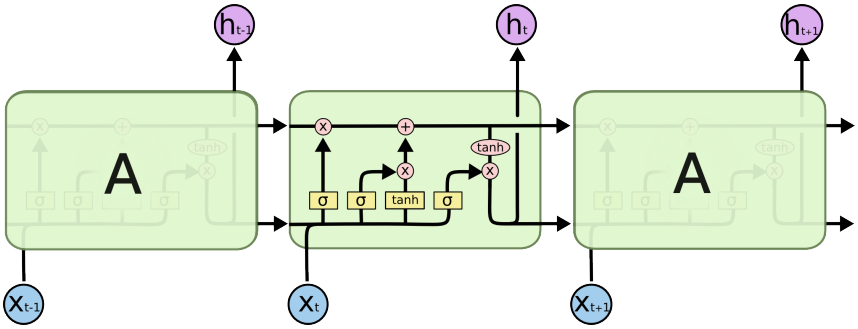
BPTT bắt đầu tương tự như lan truyền ngược, tính toán pha lan truyền tiến để xác định giá trị o_t và sau đó thực hiện lan truyền ngược từ o_t đến o_1 để xác định phần đạo hàm của hàm lỗi để cập nhật các tham số θ. Các tham số được nhân rộng theo các lát theo thời gian khi trải ra, đạo hàm được tính cho mỗi tham số tại mỗi lát cắt t. Cuối cùng các đạo hàm đầu ra của BPTT được tính bằng trung bình các đạo hàm của các lát. Việc huấn luyện mạng RNN được thực hiện cho đến khi hết số vòng lặp hoặc hàm lỗi hội tụ.

Mô hình mạng nơ-ron hồi quy đạt được hiệu quả tốt khi kết hợp được thông tin trong khá khứ, tuy nhiên mô hình RNN cũng gặp một số một vấn đề lớn ảnh hưởng đến hiệu năng sử dụng mạng, đó là vấn đề phụ thuộc xa. Trong quá trình học,

mạng RNN phải đạo hàm trên các tầng, khi số tầng quá nhiều có thể dẫn đến các vấn đề triệt tiêu đạo hàm (vanishing gradients) hoặc bùng nổ đạo hàm (exploding gradients). Triệt tiêu đạo hàm là khi giá trị đạo hàm quá nhỏ dẫn đến cập nhật trong số quá nhỏ, từ đó việc học mô hình kém hiêu quả. Ngược lại, bùng nổ đạo hàm là khi giá trị đạo hàm quá lớn dẫn đến cập nhật trọng số quá lớn khiên mô hình học không ổn định.

b) Mạng Long short term memory neural network - LSTM

Để giải quyết vấn đề phụ thuộc xa trên của mạng RNN, Hochreiter và Schmidhuber đã đề xuất mô hình LSTM. LSTM có khả năng loại bỏ hoặc bổ sung các thông tin cần thiết. Việc loại bỏ và cập nhật thông tin được thể hiện trong cấu trúc của LSTM thông qua các cổng (hình 2.8).



Hình 2.8: Cấu trúc của mạng LSTM. Nguồn [13]

Cổng đầu tiên là cổng quên (forget gate), cổng này quyết định thông tin nào trong quá khứ được giữ lại hay bỏ đi.

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

Cổng tiếp theo là cổng vào (input gate), cổng này quyết định thông tin mới nào được lưu lại.

$$\begin{aligned} i_t &= \sigma\left(W_i \cdot [h_{t-1}, x_t] + b_i\right) \\ \tilde{C}_t &= \tanh\left(W_C \cdot [h_{t-1}, x_t] + b_C\right) \\ C_t &= f_t * C_{t-1} + i_t * \tilde{C}_t \end{aligned}$$

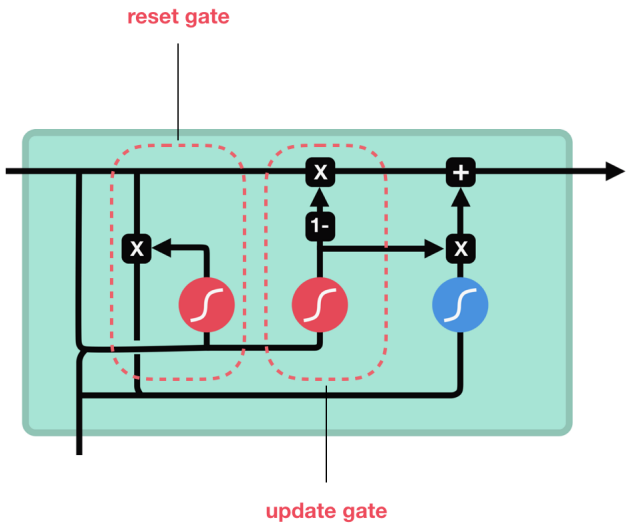
Cổng cuối cùng là cổng ra (output gate), cổng này quyết định thông tin đầu ra là gì.

$$\begin{aligned} o_t &= \sigma(W_o[h_{t-1}x_t] + b_o) \\ h_t &= o_t * \tanh(C_t) \end{aligned}$$

c) Mạng Gate Recurrent Unit - GRU

GRU là một phiên bản mới của mạng nơ-ron hồi quy, GRU khá giống với LSTM. GRU loại bỏ trạng thái tế bào và sử dụng trạng thái ẩn để truyền thông tin trong quá khứ. GRU chỉ có 2 cổng là cổng thiết lập lại (reset gate) và cổng cập nhật (update gate), cấu trúc của một nhân GRU được mô tả trong hình 2.9. Cổng cập nhật trong GRU có vai trò tương tự cổng quên và cổng đầu vào của LSTM. Nó quyết định thông tin nào được bỏ qua và thông tin nào được thêm vào. Cổng thiết lập lại được sử dụng để quyết định bao nhiêu thông tin sẽ bị quên đi.

$$
\begin{aligned}
z_t &= \sigma(W^{(z)}x_t + U^{(z)}h_{t-1}) \\
r_t &= \sigma(W^{(r)}x_t + U^{(r)}h_{t-1}) \\
h_t^{'} &= \tanh(Wx_t + r_t \cdot Uh_{t-1}) \\
h_t &= z_t \cdot h_{t-1} + (1 - z_t) \cdot h_t^{'}
\end{aligned}
$$



Hình 2.9: Cấu trúc của 1 nhân trong mạng GRU. Nguồn [15]

2.4 Mô hình mạng Generative Adversarial Network và các mô hình cải tiến

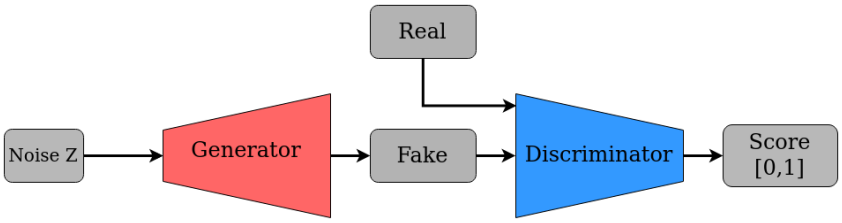
Các mô hình mạng nơ-ron nhân tạo trên đã đạt được nhiều thành tựu, tuy nhiên các mô hình này cũng có những vấn đề cần giải quyết đó là các mô hình này chỉ học được từ dữ liệu huấn luyện nếu phân phối của dữ liệu đầu vào có sự thay thổi thì kết quả của mô hình không còn đúng nữa. Để giải quyết vấn đề trên, các mô hình dự đoán xác suất đã được nghiên cứu rất nhiều. Các mô hình dự đoán xác suất được xây dựng trên lý thuyết bayes như trong [16], [17], [18], [19]. Tuy nhiên các phương pháp trên tốn chi phí tính toán khi lượng tham số tăng lên, hơn nữa các mô hình bayes khá nặng khi cần tính toán phân phối tiên nghiệm. Gần đây, Gal et

al. [20] đã sử dụng dropout [21] giúp mô hình học được phân phối. Trong khi kỹ thuật dropout được đưa ra để ngăn chặn tình trạng overfitting khi huấn luyện mạng nơ-ron, Gal đã chỉ ra trong bài báo của ông rằng mạng nơ-ron nhân tạo khi kết hợp với dropout có thể xấp xỉ được phân phối.

GAN là [22] là một loại mô hình mạng nơ-ron nhân tạo mới được sử dụng rất nhiều trong những năm gần đây. GAN có thể học được phân phối của dữ liệu khi nhận vào một tập dữ liệu và sinh ra được dữ liệu có cùng phân phối với dữ liệu huấn luyện. Với khả năng sinh dữ liệu, GAN đã thu hút được rất nhiều sự chú ý của các nhà nghiên cứu và đã có khá nhiều cải tiến như GAN có điều kiện [23], Wassertein GAN (WGAN) [24], WGAN với phạt đạo hàm (WGAN-GP) [25], least square GAN (LSGAN) [26].... Đối với bài toán phân tích chuỗi thời gian cũng đã có một số ngiên cứu như mô phỏng chuỗi thời gian sử dụng mạng GAN [4], phát hiện ngoại lệ sử dụng mạng GAN [27], sinh dữ liệu y tế thời gian thực sử dụng mạng GAN [28]. Tuy nhiên các mô hình đã có áp dụng cho bài toán dự đoán cho độ chính xác trong dự đoán vẫn còn thấp và đầu ra của mô hình còn chưa ổn định.

2.4.1 Mô hình Generative Adversarial Networks - GAN

GAN [22] là một mô hình sinh, mô hình này cố gắng học để sinh phân phối của dữ liệu sao cho giống thực tế nhất có thể. GAN bao gồm 2 mạng nơ-ron nhân tạo:



Hình 2.10: Mô hình GAN

1. Mạng nơ-ron thứ nhất được gọi là "**bộ sinh (Generator - G)**", mạng này sinh ra các điểm dữ liệu mới từ đầu vào là các vec-tơ ngẫu nhiên (thường theo phân phối chuẩn). Mục tiêu là tạo ra kết quả giả giống với thực tế nhất.
2. Mạng nơ-ron thứ hai được gọi là "**bộ phân biệt (Discriminator - D)**", mạng này xác định dữ liệu đầu vào của mạng là dữ liệu giả do "**Generator** " sinh ra hay là dữ liệu thực tế. Mục tiêu của mạng này là phân biệt được dữ liệu thật và dữ liệu giả.

Ý tưởng chính của GAN là huấn luyện 2 mạng nơ-ron khác nhau cạnh tranh nhau với 2 hàm mục tiêu khác nhau:

- Bộ sinh G cố gắng đánh lừa bộ phân biệt tin ràng đầu vào được sinh ra bởi bộ

sinh là thật.

- Ngược lại, bộ phân biệt cố gắng phát hiện được dữ liệu do bộ sinh G sinh ra là giả.
- Sau khi nhận được giá trị phạt từ bộ phân biệt D, bộ sinh G học để sinh ra dữ liệu giống với dữ liệu thực tế hơn.
- Và quá trình học lặp lại cho đến khi bộ sinh G học được phân phối của dữ liệu và bộ phân biệt không thể phân biệt được dữ liệu do bộ sinh G sinh ra là giả.

Hàm mục tiêu của GAN:

Như chúng ta đã biết, bộ phân biệt D học để phân biệt dữ liệu đầu vào là thật hay giả, nên D là một bộ phân loại nhị phân. Mô hình càng tốt khi phân tách được hoàn toàn dữ liệu giả và dữ liệu thật. Và ta xét mô hình tốt khi có xác suất cao (càng gần 1) khi đầu vào là dữ liệu thật và xác xuất càng thấp khi dữ liệu đầu vào là dữ liệu giả (đầu ra của bộ sinh G). Gọi z là vec-tơ ngẫu nhiên (thường theo phân phối chuẩn). Từ z sau khi truyền qua bộ sinh G ta có $x_{fake} = G(z)$. Và từ tập dữ liệu học x ta có dữ liệu thật x_{real}. Từ đó, đầu ra của bộ phân biệt là $D(x) = P(y|x_{real}) \in [0-1]$ đối với dữ liệu thật và $D(G(x)) = P\left(y|x_{fake}\right) \in [0-1]$ đối với dữ liệu giả. Như vậy, bộ phân biệt học để tăng giá trị $D(x)$, giảm giá trị $D(G(z))$ và bộ sinh học để tăng giá trị $D(G(z))$.

Hàm chi phí của bộ phân biệt D:

$$D_{loss_{real}} = \log(D(x))$$

$$D_{loss_{fake}} = \log(1 - D(G(z)))$$

$$\Rightarrow D_{loss} = D_{loss_{real}} + D_{loss_{fake}} = \log(D(x)) + \log(1 - D(G(z)))$$

$$\Rightarrow D_{loss} = \frac{1}{m}\sum_{i=1}^{m} \log(D(x^i)) + \log(1 - D(G(z^i)))$$

Hàm chi phí của bộ sinh G:

$$G_{loss} = \log(1 - D(G(z)))$$

hoặc

$$G_{loss} = -\log(D(G(z)))$$

$$\Rightarrow G_{loss} = \frac{1}{m}\sum_{i=1}^{m}\log(1 - D(G(z^i)))$$

hoặc

$$\frac{1}{m}\sum_{i=1}^{m} -\log(D(G(z^i)))$$

$\Longrightarrow$ Hàm mục tiêu chung $V(G, D)$:

$$\begin{aligned}\min_G \max_D V(D, G) =& \mathbb{E}_{x\sim\rho_{\text{data}}(x)}[\log D(x)] + \\ & \mathbb{E}_{z\sim\rho_{\text{noise}}(z)}[\log(1 - D(G(z)))]\end{aligned} \tag{2.1}$$

Phân tách hàm mục tiêu ta có:

$$\begin{aligned}\max_D V(D) =& \mathbb{E}_{x\sim\rho_{\text{data}}(x)}[\log D(x)] + \\ & \mathbb{E}_{z\sim\rho_{\text{noise}}(z)}[\log(1 - D(G(z)))] \\ \min_G V(G) =& \mathbb{E}_{z\sim\rho_{\text{noise}}(z)}[\log(1 - D(G(z)))]\end{aligned} \tag{2.2}$$

Chi tiết giải thuật theo giải thuật 1 dưới đây:

Algorithm 1: Giải thuật huấn luyện GAN. Số lần học bộ phân biệt D , k là một siêu tham số.

for *Các vòng lặp huấn luyện* **do**
  for *k bước* **do**
    - Lấy mẫu một gói m vec-tơ nhiễu $\left\{z^{(1)}, \ldots, z^{(m)}\right\}$ từ phân phối tiên nghiệm $p_g(z)$.
    - Lấy mẫu m ví dụ $\left\{x^{(1)}, \ldots, x^{(m)}\right\}$ từ phần phối dữ liệu thực tế $p_{data}(x)$.
    - Cập nhật trọng số của bộ phân biệt D theo stochastic gradient:

$$\nabla_{\theta_d}\frac{1}{m}\sum_{i=1}^{m}\left[\log D\left(x^{(i)}\right) + \log\left(1 - D\left(G\left(z^{(i)}\right)\right)\right)\right]. \tag{2.3}$$

  end
    - Lấy mẫu một gói m vec-tơ nhiễu $\left\{z^{(1)}, \ldots, z^{(m)}\right\}$ từ phân phối tiên nghiệm $p_g(z)$.
    - Cập nhật trọng số của bộ sinh G theo stochastic gradient:

$$\nabla_{\theta_d}\frac{1}{m}\sum_{i=1}^{m}\log\left(1 - D\left(G\left(z^{(i)}\right)\right)\right). \tag{2.4}$$

end

2.4.2 Mô hình Conditional Generative Adversarial Networks - CGAN

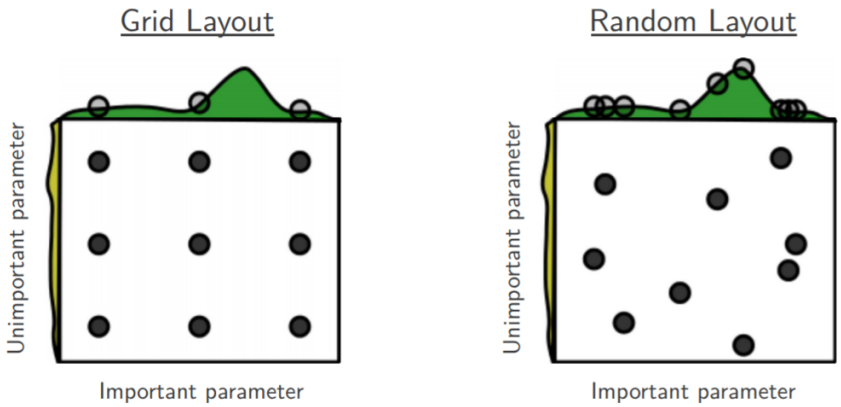
Sau khi học được phân phối dữ liệu của dữ liệu, GAN sinh ra dữ liệu mới trong phân phối đã học được nhưng chúng ta không thể xác định chính xác dữ liệu sinh ra là gì. Ví dụ, khi áp dùng GAN vào học trên tập dữ liệu MNIST, mô hình GAN sinh ra dữ liệu mới nhưng không xác định dữ liệu mới sinh đó là số mấy.

Conditional Generative Adversarial Networks - CGAN [23] là một mô hình mở rộng từ GAN. CGAN cho phép ta bổ sung thêm thông tin điều kiện đầu vào y cho GAN thay vì chỉ để mẫu ngẫu nhiêu z. Thông tin điều kiện rất có ích cho việc học và áp dụng, ví dụ bổ sung điều kiện là nhãn phân loại, thông tin giúp cho việc sinh dữ liệu the phân loại định trước. Chúng ta có thể thêm điều kiện này như là đầu vào cho cả bộ sinh và bộ phân biệt. Từ đó ta có hàm giá trị $V(G, D)$ cho biểu diễn mới:

$$\begin{aligned}\min_G \max_D V(D,G) =& \mathbb{E}_{x\sim\rho_{\text{data}}(x)}[\log D(x|y)]+ \\ & \mathbb{E}_{z\sim\rho_{\text{noise}}(z)}[\log(1-D(G(z|y)))]\end{aligned} \tag{2.5}$$

2.5 Tối ưu siêu tham số

Kết quả đầu ra của các mô hình mạng nơ-ron phụ thuộc vào các trọng số đã học được trong quá trình huấn luyện. Trong quá trình huấn luyện, các trọng số của mô hình được cập nhật và việc cập nhật này được điều khiển bởi các siêu tham số mà chúng ta thiết lập. Các siêu tham số có thể điều khiển việc học, cập nhật trọng số trở nên tốt hơn và cũng có thể trở nên xấu đi. Nếu chúng ta chọn được đúng bộ tham số phù hợp thì mô hình sẽ học được bộ trọng số tối ưu, từ đó cho kết quả tốt. Vì vậy chúng ta cần tìm ra bộ siêu tham số tối ưu để có thể đạt được kết quả tốt. Các phương pháp tối ưu siêu tham số thường sử dụng hiện nay là tìm kiếm thủ công, tìm kiếm lưới (grid search) và tìm kiếm ngẫu nhiên (random search).



Hình 2.11: Ví dụ không gian tham số của tìm kiếm lưới và tìm kiếm ngẫu nhiên. Nguồn [29]

Tìm kiếm thủ công là phương pháp người sử dụng tự đưa ra các bộ tham số và thử nghiệm để đánh giá. Phương pháp này phụ thuộc rất nhiều vào kinh nghiệm của người thử nghiệm và khó đạt được điểm tối ưu toàn cục. Phương pháp này thường dùng cho các mô hình có ít siêu tham số.

Tìm kiếm lưới là phương pháp liệt kê các trường hợp của từng siêu tham số, ghép tổ hợp thành toàn bộ không gian siêu tham số và thực hiện chạy lần lượt mô hình trên tất cả bộ siêu tham số. Sau khi chạy hết bộ tham số thì ta đánh giá kết quả của tất cả bộ tham số và chọn ra bộ tham số tốt nhất. Nhưng phương pháp này có nhược điểm là các bộ tham số do là liệt kê nên phụ thuộc rất nhiều vào ý kiến chủ quan của người sử dụng, và khi số lượng siêu tham số tăng lên thì số lượng tổ hợp tăng lên theo hàm mũ. Từ đó, ta có thể thấy không gian tham số rất lớn, việc chạy hết không gian tham số là rất tốn chi phí và thời gian. Mặt khác, việc tìm kiếm trên không gian dạng lưới rất khó có thể đạt được điểm tối ưu toàn cục. Điều này được mô tả như trong hình 2.11 bên trái, phương pháp tìm kiếm lưới chỉ xét trên các điểm đã định nghĩa từ trước, và các điểm đó có thể không là điểm tối ưu toàn cục.

Tìm kiếm ngẫu nhiên [30] là phương pháp thử ngẫu nhiên các bộ siêu tham số trong khoảng giá trị định trước của từng chiều để thử nghiệm đánh giá mô hình. Với phương pháp này ta có thể đạt được điểm tối ưu toàn cục, không gian tham số được mô tả như trong hình 2.11 bên phải. Hiệu quả của tìm kiếm ngẫu nhiên phụ thuộc vào số lần thử, khi tối ưu mô hình có nhiều siêu tham số thì số lần thử cũng rất lớn.

Với các phương pháp trên chúng ta chỉ chú ý thử nghiệm khi chọn ra các bộ tham số mà không quan tâm đến mối quan hệ giữa bộ tham số và kết quả của mô hình. Khi không gian siêu tham số lớn thì việc thử nghiệm này rất tốn thời gian và chi phí. Sau đây, chúng tôi xin giới thiệu hai phương pháp tối ưu có sử dụng mối quan hệ giữa các bộ tham số và kết quả của mô hình.

2.5.1 Phương pháp tối ưu bayesian

Tối ưu bayesian là phương pháp dựa trên lý thuyết bayes, nó thực hiện ghi lại các thông tin đánh giá khi chọn tổ hợp siêu tham số tiếp theo. Với việc chọn tổ hợp siêu tham số theo đúng hướng tốt hơn, nó có thể tập chung vào các vùng không gian tham số mà nó tin rằng sẽ mang lại điểm đánh giá tốt hơn. Phương pháp này thường yêu cầu ít lần đánh giá mô hình hon để đạt điểm tối ưu trong không gian siêu tham số. Hai thành phần chính của tối tưu bayesian là hàm thay thế và hàm lợi ích.

Hàm thay thế (Surrogate function) được sử dụng để xấp xỉ ánh xạ từ tập mẫu

ví dụ đầu vào đến giá trị điểm đầu ra. Theo quan điểm xác suất, hàm này tổng hợp xác xuất có điều kiện của hàm mục tiêu f với điều kiện dữ liệu D: $P(f|D)$

Có khá nhiều phương pháp có thể được sử dụng cho hàm thay thế, nó tương tự như bài toán hồi quy, dữ liệu được biểu diễn như đầu vào và điểm được biểu diễn như đầu ra. Mô hình tốt nhất được sử dụng là rừng ngẫu nhiên (random forest) hoặc Gaussian Process. Gaussian Process - GP là mô hình xây dựng phân phối xác suất đồng thời trên các biến. Nó có thể tổng hợp hiệu quả một số lượng lớn các hàm.

Hàm lợi ích (Acquisition functions) có nhiệm vụ đưa các điểm mẫu trong không gian tìm kiếm. Hàm này cần phải cân bằng giữa khai thác và khám phá. Khai thác có nghĩa là lấy mẫu thông qua hàm thay thế dự đoán có độ tin cậy cao và khám phá là lấy mẫu tại các vị trí mà sự dự đoán có độ không chắc chắn cao.

Algorithm 2: Giải thuật tối ưu bayesian

Thiết lập tiến trình Gausian tiên nghiệm trên f Quan sát f tại n_0 điểm trong không gian tham số định trước ban đầu. Gán $n = n_0$.
while *n* <= *N* **do**
    Cập nhật phân phối xác suất hậu nghiệm trên f sử dụng các dữ liệu đã có.
    Chọn x_n sao cho hàm lợi ích cực đại theo x, trong đó hàm lợi ích được tính theo phân phối hậu nghiệm hiện tại.
    Tính $y_n = f(x_n)$
    n += 1
end
return điểm mà có $f(x)$ lớn nhất nếu điểm đó đã được đánh giá, hoặc điểm có phân phối hậu nghiệm lớn nhất.

2.5.2 Phương pháp tối ưu dựa trên giải thuật bày đàn

Particle Swarm Optimization - PSO [31] là một trong những giải thuật heuristic có thể giải được các vấn đề tối ưu có số chiều lớn. Nó là một giải thuật tìm kiếm dựa trên toàn bộ quần thể và tìm kiếm trên từng nhóm cá thể nhỏ hơn. Được đề xuất vào năm 1995 bởi Kennedy and Eberhart, PSO là một giải thuật tối ưu lấy cảm hứng từ các tập tính của động vật trong môi trường tự nhiên, giải thuật được thiết kế theo tập tính kiếm ăn của đàn chim hay sự di chuyển của đàn cá. Trong PSO, mỗi cá thể đưa ra quyết định dựa trên kinh nghiệm của chính bản thân nó và kinh nghiệm của cả quần thể. Ưu điểm chính của giải thuật PSO là đơn giản, dễ cài đặt, dễ dàng để điều chỉnh các tham số và hiệu quả tính toán cao.

Ý tưởng chính của giải thuật được thể hiện thông qua việc cập nhật vị trí của mỗi cá thể bằng 2 công thức cập nhật vị trí (2.6) và công thức tính vận tốc (2.7).

$$x_{k+1}^i = x_k^i + v_{k+1}^i \tag{2.6}$$

$$v_{k+1}^i = w_k v_k^i + c_1 r_1 \left(p_k^i - x_k^i\right) + c_2 r_2 \left(p_k^q - x_k^i\right) \tag{2.7}$$

Trong đó:

$$\begin{cases} x_k^i & \text{là vị trí của cá thể } i \text{ tại thời điểm } k \\ v_k^i & \text{là vận tốc của cá thể } i \text{ tại thời điểm } k \\ p_k^i & \text{là vị trí tốt nhất của riêng cá thể } i \text{ tại thời điểm } k \\ p_k^g & \text{là vị trí tốt nhất của quần thể tại thời điểm } k \\ w_k & \text{là trọng số quán tính di chuyển cá thể tại thời điểm } k \\ c_1, c_2 & \text{là hệ số theo vị trí tốt nhất của riêng cá thể} \\ & \text{và vị trí tốt nhất của cả cộng đồng} \\ r_1, r_2 & \text{là các số ngẫu nhiên trong khoảng 0 đến 1} \end{cases}$$

Từ công thức tính vận tốc (2.7), ta thấy có 2 nhóm quan trọng là

- Phần kinh nghiệm cộng đồng: $c_2 r_2 (p_k^g - x_k^i)$
- Phần kinh nghiệm cá nhân: $c_1 r_1 (p_k^i - x_k^i)$

Chi tiết giải thuật được trình bày trong giải thuật 3. Hướng di chuyển của mỗi cá thể là vec-tơ tổng hợp từ 3 thành phần, đó là quán tính (vận tốc trước đó), khoảng các từ vị trí tốt nhất của mỗi cá thể và khoảng các từ vị trí của cả quần thể. Ba hướng trên được kiểm soát bởi các trọng số w_k, c_1, c_2 và các giá trị ngẫu nhiên r_1, r_2.

Algorithm 3: Giải thuật tối ưu bày đàn

A. Khởi tạo

1. Thiết lập các hằng số k_{max}, w_k, c_1, c_2
2. Khởi tạo ngẫu nhiên vị trí của các cá thể.
3. Khởi tạo vận tốc ngẫu nhiên của các cá thể.
4. k=1

B. Tối ưu

1. Đánh giá hàm mục tiêu f_k^i tại mỗi vị trí của mỗi cá thể x_k^i
2. Nếu $f_k^i \leq f_{best}^i$ thì $f_{best}^i = f_k^i$ và $p_k^i = x_k^i$
3. Nếu $f_k^i \leq f_{best}^g$ thì $f_{best}^g = f_k^i$ và $p_g^i = x_k^i$
4. Nếu thỏa mãn điều kiện dừng thì chuyển đến bước C
5. Cập nhật vận tốc của tất cả cá thể theo công thức (2.7)
6. Cập nhật vị trí của tất cả cá thể theo công thức (2.6)
7. k = k+1
8. Chuyển đến B(1).

C. Kết thúc

CHƯƠNG 3. MÔ HÌNH ĐỀ XUẤT

Trong chương trước chúng tôi đã giới thiệu về điện toán đám mây cũng như bài toán mở rộng trong hệ thống điện toán đám mây. Chúng tôi cũng đã nêu một số phương pháp dự đoán tài nguyên hỗ trợ cho việc mở rộng tài nguyên, các phương pháp này đều có những ưu điểm và nhược điểm riêng. Sau đó chúng tôi đã trình bày nội dung lý thuyết của mạng GAN và các phương pháp tối ưu siêu tham số. Trong chương này, chúng tôi trình bày mô hình hệ thống dự đoán tài nguyên sử dụng trong tương lai của hệ thống điện toán đám mây dựa trên mô hình cơ sở là mô hình GAN. Và từ những hạn chế của mô hình GAN hiện tại, chúng tôi đã đưa ra một số cải tiến giúp mô hình đạt độ chính xác cao hơn trong bài toán dự đoán.

3.1 Tổng quan hệ thống

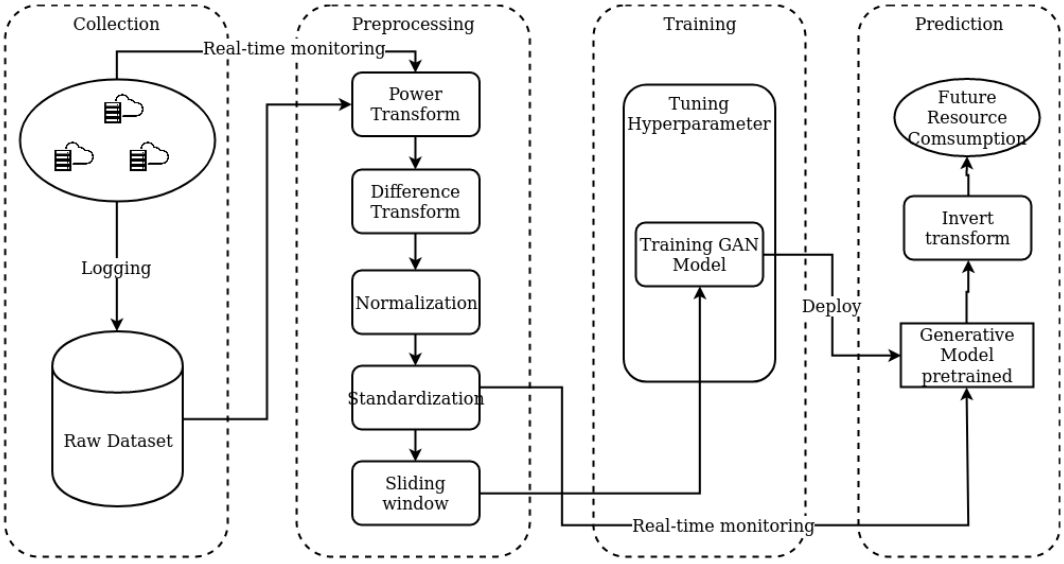
Tổng quan của hệ thống mà chúng tôi thiết kế được mô tả trong hình 3.1. Trong đó có 4 mô đun chính là: thu thập dữ liệu, tiền xử lý dữ liệu , huấn luyện mô hình và dự đoán.

Theo luồng dữ liệu, giai đoạn đầu tiên là thu thập dữ liệu. Trong giai đoạn này, các thông số kĩ thuật về tài nguyên sử dụng của hệ thống sẽ được theo dõi và ghi lại.

Sau khi được thu thập dữ, dữ liệu được chuyển sang bước tiền xử lý dữ liệu. Tại bước này dữ liệu được làm sạch và biến đổi sang một dạng mới phù hợp với mô hình học.

Dữ liệu sau khi đã tiền xử lý sẽ là đầu vào của cho bước huấn luyện mô hình. Chúng tôi đã lựa chọn mô hình RCGAN đề áp dụng cho bài toán này. Mô hình sẽ được huấn luyện và tối ưu siêu tham số, sau đó đưa ra được một mô hình bộ sinh đã học được phân phối của dữ liệu.

Cuối cùng là bước dự đoán, tại bước này phần bộ sinh Generative trong mô hình GAN đã huấn luyện sẽ được lấy ra để dự đoán. Dữ liệu sau khi dự đoán sẽ được chuyển ngược lại mẫu thực tế như trước khi biến đổi ở bước tiền xử lý.



Hình 3.1: Kiến trúc tổng quan hệ thống

3.2 Thu thập dữ liệu

Để xây dựng được mô hình dự báo tài nguyên sử dụng, các hệ thống điện toán đám mây cần có các mô đun theo dõi và ghi lại dữ liệu về các thông số kĩ thuật về tài nguyên như CPU, bộ nhớ RAM, lưu lượng vào ra dữ liệu ổ cứng, thông tin lưu lượng trên mạng..... Có 2 loại dữ liệu là dữ liệu lịch sử và dữ liệu thời gian thực.

Dữ liệu lịch sử đã thu thập trong quá khứ được sử dụng cho các giai đoạn tiền xử lý và huấn luyện mô hình. Dữ liệu thời gian thực được thu thập lại, được tiền xử lý và đưa qua mô hình dự báo để đưa ra dự báo tài nguyên sử dụng trong tương lai ngay lập tức.

3.3 Tiền xử lý dữ liệu

Mục tiêu của mô đun này là khởi tạo bộ dữ liệu phù hợp cho mô đun huấn luyện mô hình. Với dữ liệu dữ liệu chuỗi thời gian chúng tôi đưa ra các phép biến đổi sau: power transform, difference transform, starndardization, normalization và kỹ thuật cửa sổ trượt để tạo dữ liệu học có giám sát. Đối với các phép biến đổi, chúng tôi cũng đưa ra các phép biến đổi ngược lại tương ứng để áp dụng khi dự đoán thực tế.

3.3.1 Các phép biến đổi dữ liệu

- **Power transform**: Đây là phép biến đổi giá trị riêng từng điểm của dữ liệu của dữ thông qua các hàm như căn bậc 2, bình phương, $\log$.... Trong đồ án này, dữ liệu là thông số các tài nguyên nên giá trị các giá trị điểm dữ liệu là lớn hơn hoặc bằng 0, vì vậy chúng tôi sử dụng hàm $\log$ để co dải giá trị của dữ liệu lại. Nếu để các điểm dữ liệu có sự chênh lệch quá lớn có thể dẫn đến các

giá trị nhỏ có thể trở nên vô nghĩa trong quá trình học. Và để tránh điểm dữ liệu 0, điểm làm cho hàm log không xác định, chúng tôi sử dụng công thức:

$$x'_t = \log(x_t + 1)$$

Công thức biến đổi ngược lại thông qua hàm $\exp$:

$$x_t = \exp(x'_t) - 1$$

- **Difference transform**: Đây là phép biến đổi trạng thái của dữ liệu từ giá trị thông số sang giá trị chỉ sự thay đổi so với điểm dữ liệu trước đó.

$$x'_t = x_t - x_{t-1}$$

Phép biến đổi này giúp ta loại bỏ được yếu tố xu hướng và chu kỳ của dữ liệu, 2 yếu tố này thường có trong dữ liệu chuỗi thời gian. Khi loại được 2 yếu tố này, mô hình học sẽ học một cách tổng quát hơn. Ví dụ, khi không biến đổi, có thể dữ liệu huấn luyện mô hình có xu hướng đều, hoặc tăng nhẹ nhưng dữ liệu đánh giá lại có xu hướng tăng nhanh hơn. Nếu mô hình học trên dữ liệu như vậy thì sẽ không có tính tổng quát. Với các biến đổi này ta sẽ bỏ qua một điểm dữ liệu đầu tiên do không có điểm dữ liệu trước đó. Để biến đổi ngược lại ta cần lưu lại các điểm dữ liệu lịch sử và áp dụng công thức:

$$x_t = x'_t + x_{t-1}$$

- **Standardization**: Đây là phép biến đổi dữ liệu về phân phối Gausian. Phép biến đổi này giúp cho mô hình học với giải thuật lan truyền ngược nhanh hội tụ hơn, nội dung này đã được giải thích trong [32].

$$x'_t = (x_t - x_{mean})/x_{std}$$

Công thức biến đổi ngược lại:

$$x_t = x'_t * x_{std} + x_{mean}$$

Trong đó:

$$\begin{cases} x_t & \text{là giá trị tại thời điểm } t \\ x_{mean} & \text{là giá trị trung bình của tập dữ liệu} \\ x_{std} & \text{là giá trị độ lệch chuẩn của dữ liệu} \end{cases}$$

- **Normalization**: Đây là phép biến đổi dữ liệu về dải từ 0 đến 1. Phép biến đổi này giúp cho mô hình học với giải thuật lan truyền ngược nhanh hội tụ hơn, nội dung này đã được giải thích trong [32].

$$x'_t = (x_t - x_{min})/(x_{max} - x_{min})$$

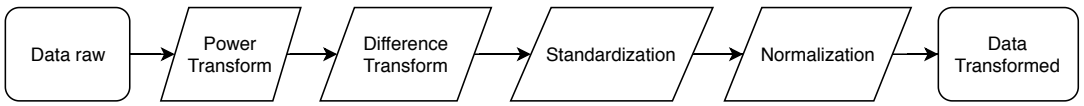
Biến đổi ngược lại theo công thức:

$$x'_t = x_t * (x_{max} - x_{min}) + x_{min}$$

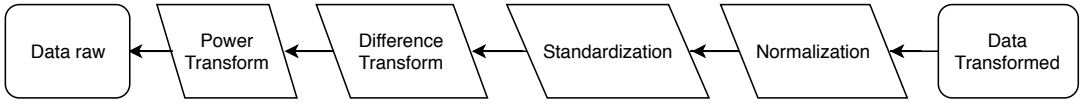
Trong đó:

$$\begin{cases} x_t & \text{là giá trị tại thời điểm } t \\ x_{min} & \text{là giá trị nhỏ nhất của tập dữ liệu} \\ x_{max} & \text{là giá trị lớn nhất của dữ liệu} \end{cases}$$

Khi kết hợp các phép biến đổi trên, ta áp dụng theo thứ tự như trong hình 3.2 và biến đổi ngược lại theo thứ tự như trong hình 3.3



Hình 3.2: Thứ tự biến đổi chuỗi thời gian

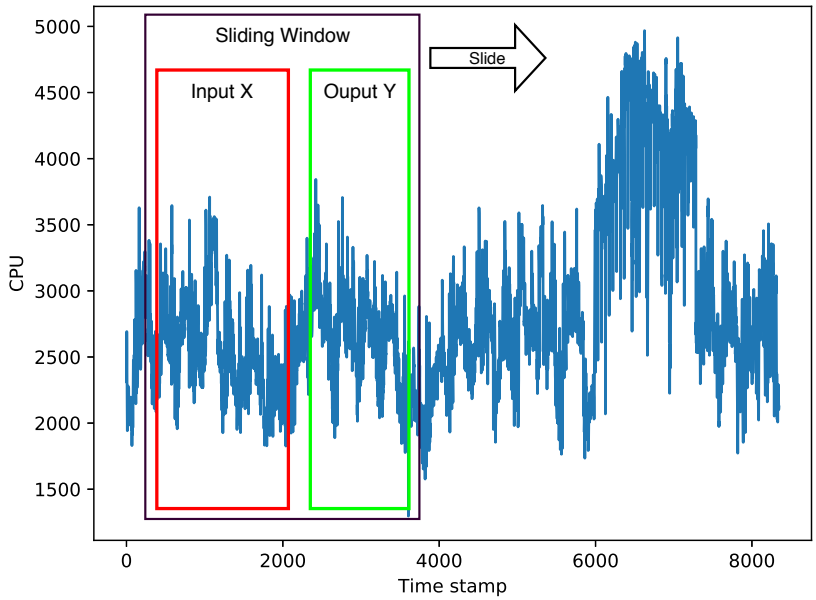


Hình 3.3: Thứ tự biến đổi ngược chuỗi thời gian

3.3.2 Kỹ thuật cửa sổ trượt

Khi áp dụng vào mô hình, chúng ta cần biểu diễn dữ liệu đúng với yêu cầu đầu vào của mô hình. Ví dụ, khi ta huấn luyện mô hình mạng hồi quy LSTM học cho bài toán dự đoán chuỗi thời gian thì dữ liệu cần được biểu diễn dưới dạng học có giám sát. Trong đó mỗi điểm dữ liệu có 2 thành phần là chuỗi các giá trị trong quá khứ gần $x_{t-k}, x_{t-k+1}, ..., x_t$ và nhãn là giá trị tiếp theo của chuỗi giá trị đầu vào

x_{t+1}. Hai phần này được lấy dựa trên 1 cửa sổ trượt trên dữ liệu chuỗi thời gian như trong hình 3.4. Mỗi lần cửa sổ này trượt ta có được một điểm dữ liệu. Trong nghiên cứu này cũng như vậy, chúng tôi áp dụng mô hình RCGAN thì dữ liệu cũng cần 2 thành phần là chuỗi các giá trị trong quá khứ gần được biểu diễn là điều kiện thông tin đầu vào của mô hình và giá trị tiếp theo của chuỗi là giá trị tương lai do bộ sinh G sinh ra.



Hình 3.4: Biểu diễn dữ liệu

3.4 Xây dựng mô hình GAN

Huấn luyện mô hình là phần quan trọng nhất của hệ thống. Để có một hệ thống tốt chúng ta cần một mô hình dự đoán chính xác. Trong phần này, chúng tôi trình bày mô hình RCGAN áp dụng cho bài toán dự đoán chuỗi thời gian, cải tiến hàm chi phí mới trong quá trình huấn luyện và đưa ra phương pháp tối ưu tham số.

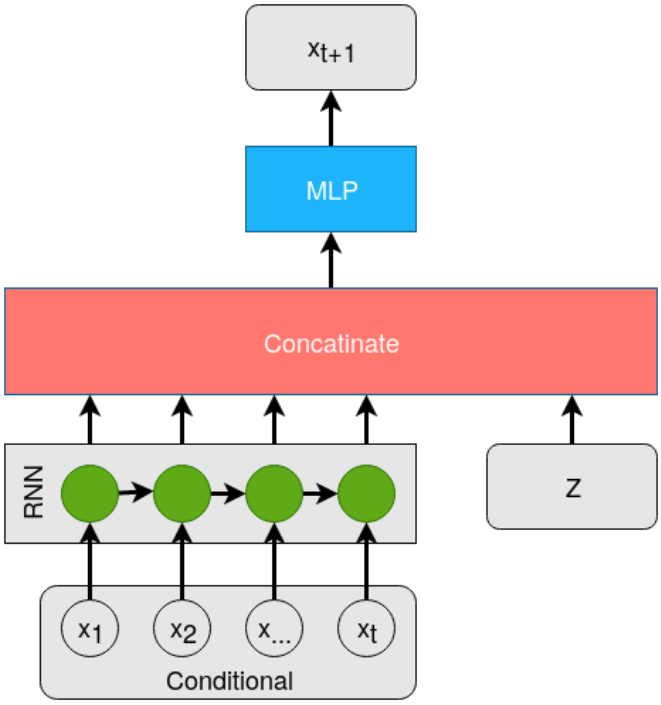
3.4.1 Mô hình học

Mô hình GAN đã được áp dụng vào rất nhiều lĩnh vực và đã đạt được rất nhiều thành công, đặc biệt là trong lĩnh vực sinh dữ liệu ảnh. Dưới đây là phần thiết kế mô hình học dựa trên mô hình GAN khi áp dụng cho bài toán dự đoán chuỗi thời gian. Với mô hình GAN gốc, dữ liệu được học theo phương pháp học không giám sát, sau khi học được phân phối của dữ liệu, mô hình có thể sinh ra chuỗi thời gian bất kỳ tùy thuộc vào vec-tơ nhiễu đầu vào. Việc sinh dữ liệu như vậy không phù hợp với bài toán dự đoán chuỗi thời gian do dữ liệu mới sinh ra có thể cùng phân phối với dữ liệu thực tế nhưng chúng ta không thể xác định chuỗi đó là thuộc thời điểm nào. Vì vậy, chúng ta cần bổ sung thêm thông tin để mô hình có thể đưa ra

dự đoán phù hợp, đúng thời điểm đang xét. Thay cho mô hình GAN gốc, chúng tôi sử dụng mô hình GAN có điều kiện (conditional generative adversarial network - CGAN). Trong bài toán dự đoán chuỗi thời gian thì thông tin điều kiện được bổ sung thêm là chuỗi các giá trị trong lịch sử ngay trước đó: $c = x_{t-k+1}, x_{t-k+2}, ..., x_t$ và đầu ra mong muốn là phân phối của dữ liệu $\rho(x_{t+1}|c)$.

Dữ liệu chuỗi thời gian thường có sự phụ thuộc khá nhiều vào các giá trị lịch sử trước đó. Hiện nay các mô hình học sâu như mạng nơ-ron hồi quy LSTM, GRU đã học được rất tốt mối quan hệ trong chuỗi thời gian, do khả năng nhớ thông tin trong quá trình học của các mô hình mạng nơ-ron hồi quy. Do vậy trong mô hình CGAN này chúng tôi có sử dụng mạng nơ-ron hồi quy LSTM hoặc GRU để xây dựng các thành phần bộ sinh và bộ phân biệt.

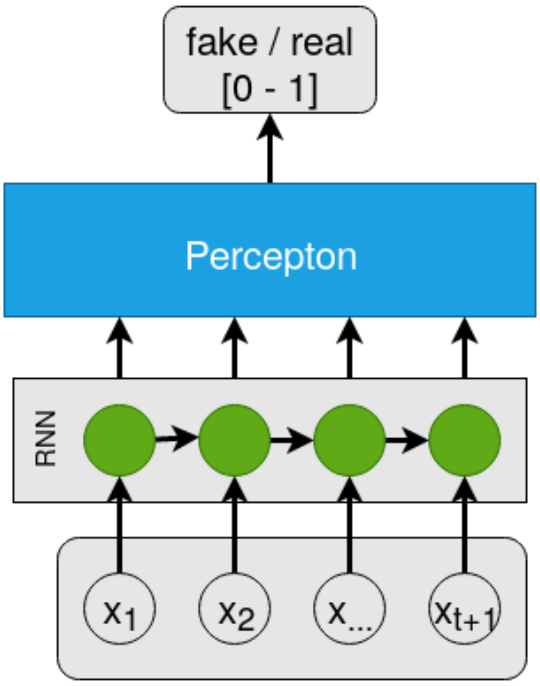
Kiến trúc của bộ sinh G được mô tả trong hình 3.5. Thông tin điều kiện đầu vào $c = x_{t-k+1}, x_{t-k+2}, ..., x_t$ được đưa qua một mạng nơ-ron hồi quy để học đặc trưng của chuỗi đầu vào. Sau đó, đặc trưng học được từ mạng nơ-ron hồi quy được nối với vec-tơ ngẫu nhiên Z và được truyền qua một mạng nơ-ron truyền thẳng. Đầu ra cuối cùng của bộ sinh G là giá trị x_{t+1}.



Hình 3.5: Kiến trúc bộ sinh G

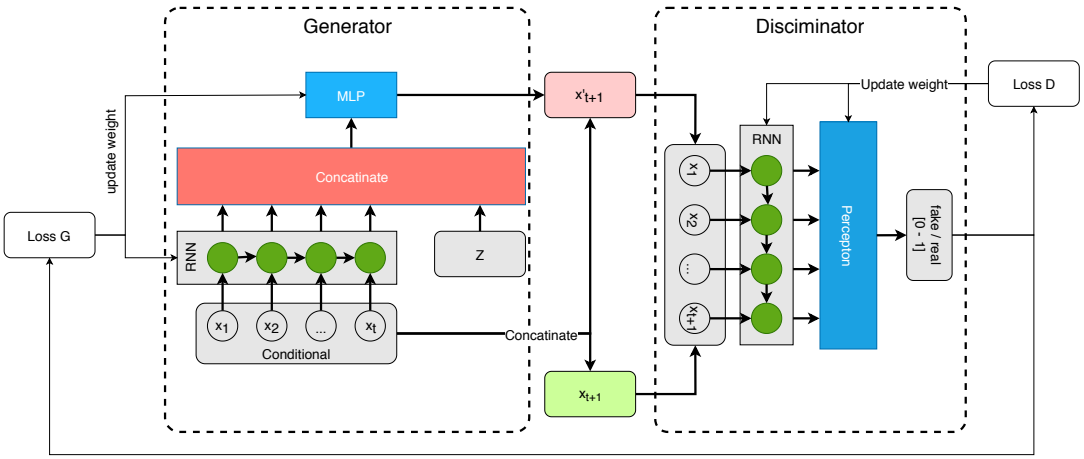
Và kiến trúc của bộ phân biệt D như trong hình 3.6. Dữ liệu đầu vào của bộ phân biệt D là chuỗi điều kiện $c = x_{t-k+1}, x_{t-k+2}, ..., x_t$ nối với giá trị x_{t+1}. Giá trị x_{t+1} do bộ sinh G sinh ra thì điểm dữ liệu có nhãn là 0 (dữ liệu giả) và có nhãn là 1 (dữ liệu thật) nếu x_{t+1} lấy từ tập dữ liệu thực tế. Kiến trúc của bô phân biệt có

mạng LSTM/GRU để học đặc trưng của chuỗi đầu vào và một nơ-ron để đưa ra xác suất dữ liệu đầu vào là thật hay giả.



Hình 3.6: Kiến trúc bộ phân biệt D

Mô hình học tổng thể được thể hiện trong hình 3.7. Với đầu vào là chuỗi điều kiện $c = x_{t-k+1}, x_{t-k+2}, ..., x_t$ và vec-tơ ngẫu nhiên Z, G đưa ra giá trị dự đoán $\hat{x}_{t+1}$. Giá trị này được kết nối với chuỗi điều kiện ban đầu tạo thành đầu vào cho D với nhãn là 0 (dữ liệu giả). Giá trị x_{t+1} thực tế cũng được kết nối với chuỗi điều kiện tạo thành đầu vào cho D với nhãn là 1 (dữ liệu thật). Sau đó hàm chi phí của G và D được tính toán và áp dụng phương pháp đạo hàm để cập nhật trọng số của G và D



Hình 3.7: Kiến trúc mô hình GAN tổng thể cho bài toán dự đoán chuỗi thời gian

3.4.2 Cải tiến hàm mục tiêu mới của mô hình GAN áp dụng cho bài toán chuỗi thời gian

Với mô hình đã đề xuất trên, ta sử dụng hàm mục tiêu (3.1) trong mô hình CGAN. Nhưng đối với bài toán chuỗi thời gian các thông tin trong lịch sử có mỗi liên hệ với nhau thông qua mô hình mạng hồi quy RNN. Từ đó, ta có hàm mục tiêu như sau:

$$\begin{aligned}\min_G \max_D V(D,G) =& \mathbb{E}_{x\sim\rho_{\text{data}}(x)}[\log D(x_{t+1}|(x_1,x_2,...,x_t)]+ \\ & \mathbb{E}_{z\sim\rho_{\text{noise}}(z)}[\log(1-D(G(z|(x_1,x_2,...,x_t))))]\end{aligned} \tag{3.1}$$

Khi huấn luyện mô hình, bộ phân biệt D có mục tiêu là tối đa khả năng phân biệt chuỗi đầu vào là thật hay do bộ sinh G tạo ra. Bộ phân biệt D được cập nhật theo phương pháp Gradient descent với hàm mất mát:

$$D_{loss} = \frac{1}{m}\sum_{i=1}^{m}\log(D(x_{t+1}|(x_1,x_2,...,x_t)) + \log(1-D(G(z^i|(x_1,x_2,...,x_t)))) \tag{3.2}$$

Ngược lại với bộ phân biệt D, mục tiêu của bộ sinh G là tối thiểu khả năng phân biệt của D. Do vậy bộ sinh G cũng được cập nhật theo phương pháp Gradient descent với hàm mất mát:

$$G_{loss} = L_{gan} = \frac{1}{m}\sum_{i=1}^{m}\log(1-D(G(z^i|(x_1,x_2,...,x_t))) \tag{3.3}$$

Tuy nhiên, với hai hàm mục tiêu đối nghịch nhau, mô hình GAN càng khó hội tụ hơn. Đặc biệt, khi áp dụng vào bài toán dự đoán chuỗi thời gian, giá trị dự đoán đầu ra yêu cầu phải càng gần giá trị thực tế càng tốt. Theo hàm chi phí trên của GAN, ta thấy bộ sinh G chủ yếu học theo hướng làm cho D tin rằng chuỗi giá trị do G sinh ra là thật mà không có mục tiêu là đưa ra dự đoán gần với thực tế. Do vậy nếu D là một mô hình yếu, có khả năng phân biệt thật giả kém hoặc nếu G đưa ra một chuỗi đúng trong quá khứ nhưng sai tại thời điểm đang xét có thể làm cho mô hình học sai. Lúc này D nhận định các chuỗi G sinh ra là thật trong khi các chuỗi đó sai lệch rất nhiều so với thực tế hiện tại. Mà việc cập nhật trọng số của G lại phụ thuộc vào kết quả đầu ra của D nên dữ liệu sinh ra bởi G không đảm bảo gần với giá trị thực tế $\hat{y}_{t+1}$.

Để giải quyết vấn đề này, bộ sinh G cần bị phạt trong trường hợp đưa ra đầu ra khác xa so với thực tế. Và chúng tôi đã bổ sung lỗi dự đoán trung bình bình

phương sai số Mean Square Error - MSE vào hàm mất mát của G.

$$L_{mse} = \frac{1}{m}\sum_{i=1}^{m}(\hat{y}_{t+1} - y_{t+1})$$

Và một yếu tố cũng khá quan trọng là khi dự đoán ta cần đoán được cả xu hướng (tăng hay giảm) của dữ liệu. Vì vậy chúng tôi áp dụng thêm mức phạt khi mô hình đưa ra xu hướng sai.

$$L_{direct} = \begin{cases} 1, & \text{Nếu}(x_{t+1} - x_t) \neq (\hat{x}_{t+1} - x_t) \\ 0, & \text{Nếu ngược lại} \end{cases} \tag{3.4}$$

$$G_{loss} = L_{gan} + L_{mse} + L_{direct} \tag{3.5}$$

Khi bổ sung hàm mất mát MSE, mô hình GAN có thể mất đi khả năng học được phân phối của dữ liệu. Nếu mô hình chỉ học theo MSE thì mô hình sẽ chỉ như các mô hình thông thường, mất đi khả năng học phân phối của dữ liệu. Còn nếu mô hình chỉ học theo hàm mất mát ban đầu của CGAN kết quả sẽ không ổn định như đã nói ở trên. Vì vậy, để điều chỉnh hàm mất mát, chúng tôi sử dụng hai hệ số hiệu chỉnh là α và β:

$$G_{loss} = \alpha * L_{gan} + \beta * L_{mse} + \gamma * L_{direct} \tag{3.6}$$

Như đã nói ở trên, chúng tôi bổ sung hàm mất mát MSE để tránh trường hợp G sinh ra dữ liệu quá khác so với thực tế. Nếu sau khi G đã học được sinh dữ liệu gần nhất với thực tế mà ta tiếp tục thực hiện cập nhật tham số thì có thể dẫn đến $L_{rcgan} = 0$ và mô hình tiếp tục học theo L_{mse}. Như vậy lại gặp phải nhược điểm như đã trình bày ở trên. Vì vậy, chúng tôi sử dụng thêm một tham số để kiểm soát việc sử dụng L_{mse}, đó là một giá trị ngưỡng $\theta >= 0$. Giá trị ngưỡng θ này sẽ giúp loại bỏ L_{mse} khi mà mô hình đã học được phân phối của dữ liệu trong quá trình huấn luyện.

$$G_{loss} = \begin{cases} \alpha * L_{gan} + \beta * L_{mse} + \gamma * L_{direct}, & \text{Nếu } L_{mse} \geq \theta \\ L_{gan}, & \text{Ngược lại} \end{cases} \tag{3.7}$$

Với các tham số α, β và γ, chúng tôi xác định thông qua việc tối ưu tham số sẽ trình bày ở phần sau. Còn đối với giá trị ngưỡng θ chúng tôi xác định thông qua

phần sai số chấp nhận được. Đối với mỗi bài toán chúng ta chấp nhận một sai số nhất định. Khi L_{mse} giúp mô hình học được dự đoán trong khoảng sai số chấp nhận được thì mô hình có thể tiếp tục học phân phối theo L_{gan}.

3.4.3 Tối ưu siêu tham số

Đối với các mô hình học máy nói chung, việc tối ưu siêu tham số là rất quan trọng vì nó ảnh hưởng nhiều đến kết quả của mô hình. Các mô hình đơn giản, ít siêu tham số thì có thể sử dụng các phương pháp đơn giản như tìm kiếm lưới hay tìm kiếm ngẫu nhiên. Nhưng đối với mô hình phức tạp, nhiều siêu tham số việc sử dụng tìm kiếm lưới hay tìm kiếm ngẫu nhiên là rất khó đạt được điểm tối ưu toàn cục vì số lượng tổ hợp của các siêu tham số quá lớn, chúng ta không thể đánh giá trên tất cả các tổ hợp được. Vì vậy, với các siêu tham số của mô hình đề xuất, chúng tôi sử dụng phương pháp tối ưu bayes để tối ưu siêu tham số và đề xuất sử dụng giải thuật tối ưu bầy đàn cải thiện phương pháp tối ưu siêu tham số.

Mô hình đã đề xuất ở trên có rất nhiều siêu tham số, các siêu tham số được mô tả trong bảng 3.1.

Siêu tham số	Ý nghĩa
layer size (G, D)	Mảng chỉ số nhân trong các tầng của mạng RNN trong G và D
cell type (G, D)	Loại mạng nơ-ron hồi quy: LSTM hoặc GRU trong G và D
activation (G, D)	Hàm truyền của mạng nơ-ron trong G và D
dropout (G, D)	Tỉ lệ loại bỏ trong mạng nơ-ron trong G và D
number in	Chiều dài chuỗi điều kiện đầu vào của G
noise shape	Kích thước vec-tơ nhiễu Z
learning rate (G, D)	Tốc độ học của G và D
number train d	Số vòng lặp huấn luyện D khi huấn luyện G 1 vòng
batch size	Kích thước gói dữ liệu trong một lần tính hàm chi phí
α, β, γ	Các tham số trong hàm chi phí

Bảng 3.1: Bảng các siêu tham số của mô hình

Mô hình mạng nơ-ron hồi quy LSTM là một mô hình có 5 siêu tham số như số nhân, hàm truyền, tỉ lệ dropout, tỉ lệ học, kích thước đầu vào. Với 5 siêu tham số như vậy việc tối ưu siêu tham số đã là rất khó khăn. Như vậy trong mô hình đề xuất với 18 siêu tham số, việc áp dụng phương pháp tối ưu thủ công là không phù hợp nên chúng tôi đã áp dụng giải thuật tối ưu bày đàn để tối ưu siêu tham số.

Trước khi áp dụng các phương pháp tối ưu siêu tham số, chúng ta cần biểu

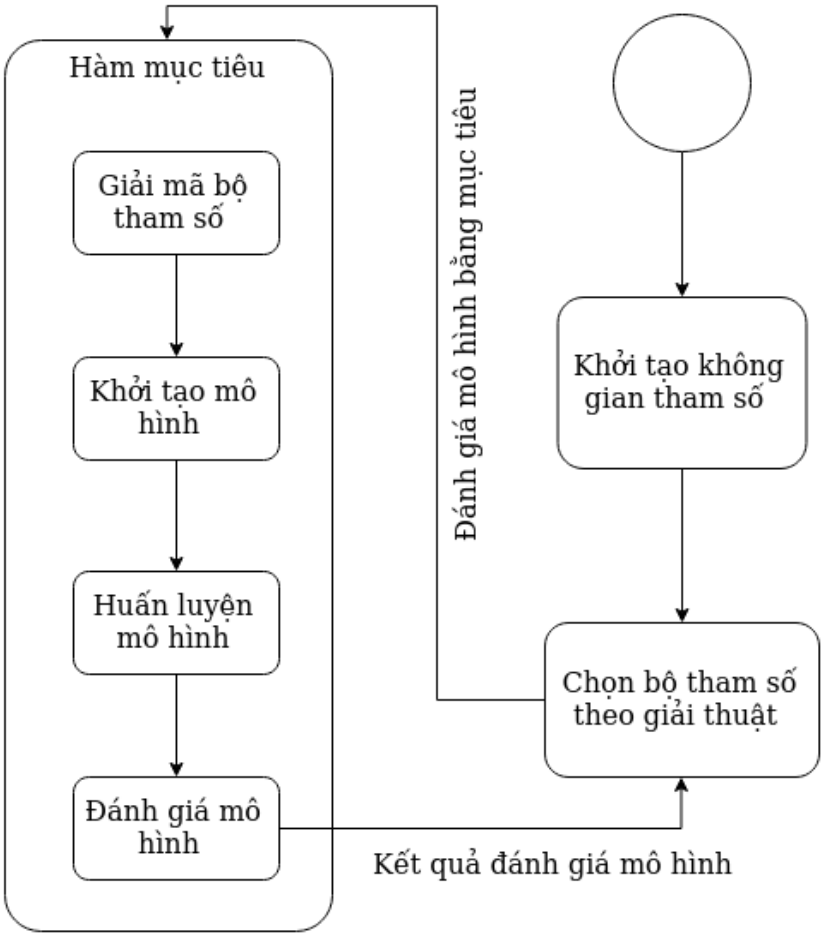
diễn được không gian siêu tham số. Chúng tôi biểu diễn không gian siêu tham số trên 3 loại kiểu dữ liệu là: liên tục (continuous), rời rạc (discrete), và phân loại(category). Các siêu tham số xác định trên một kiểu dữ liệu và cần xác định trước không gian biểu diễn của siêu tham số đó. Không gian các siêu tham số cho mô hình mà chúng tôi đã đề xuất được biểu diễn trong bảng 3.2.

Siêu tham số	Kiểu dữ liệu	Không gian giá trị
layer size (G, D)	discrete	[2, 4, 8, 16, 32, 64]
cell type (G, D)	category	LSTM, GRU
activation (G, D)	category	tanh, sigmoid, relu
dropout (G, D)	continuous	(0, 0.5)
number in	discrete	[1, 2, 3, 4, 5, 6, 7, 8]
noise shape	discrete	[2, 4, 8, 16, 32, 64]
learning rate (G, D)	continuous	(0.0001, 1)
number train d	discrete	[1, 2, 3, 4, 5]
batch size	discrete	[2, 4, 8, 16, 32, 64]
α, β, γ	continuous	(0, 1)

Bảng 3.2: Bảng biểu diễn không gian các siêu tham số

Tiếp theo, ta cần mã hóa các siêu tham số về dạng số học. Các tham số kiểu liên tục và kiểu rời rạc được biểu diễn bằng chính giá trị số của nó. Còn kiểu phân loại sẽ được biểu diễn dựa trên chỉ số tương ứng với giá trị loại được lưu trên một danh sách. Khi chuyển ngược lại thì với kiểu liên tục và rời rạc thì dữ nguyên giá trị còn kiểu phân loại thì từ chỉ số ta lấy được giá trị loại tương ứng.

Trước khi áp dụng các phương pháp tối ưu ta cần xây dựng một hàm mục tiêu. Khi tối ưu siêu tham số của mô hình thì hàm mục tiêu chính là hàm đánh giá của mô hình. Một hàm mục tiêu cần có giải mã bộ tham số đầu vào theo phương pháp vừa nói trên, khởi tạo mô hình, huấn luyện mô hình với tập dữ liệu huấn luyện, đánh giá mô hình với tập dữ liệu đánh giá và trả về giá trị đánh giá của mô hình. Trong mô hình GAN đã đề xuất, mục tiêu của mô hình là học được phân phối của dữ liệu nên độ đo để đánh giá mô hình là độ sai khác của phân phối. Và giá trị đánh giá này cũng là giá trị của hàm mục tiêu khi nhận đầu vào là bộ siêu tham số của mô hình.



Hình 3.8: Sơ đồ tối ưu siêu tham số

3.5 Dự đoán

Sau khi đã tối ưu siêu tham số và huấn luyện được mô hình có kết quả tốt trên tập đánh giá, chúng ta cần đưa mô hình vào dự đoán thực tế. Với mô hình GAN sau khi được học xong, chỉ có bộ sinh G được đưa ra để áp dụng vào dự đoán. Vì lúc này bộ sinh G đã học được phân phối dữ liệu nên G có thể đưa ra dự đoán trực tiếp mà không cần bộ phân biệt D. Dữ liệu thời gian thực được chuyển qua bộ tiền xử lý và thu được dữ liệu đã biến đổi. Dữ liệu biến đổi này được đưa qua G và G trả lại giá trị dự đoán trong tương lai. Giá trị mà G sinh ra ở trạng thái đã biến đổi nên ta cần chuyển đổi lại về dạng thực tế. Và giá trị sau chuyển đổi ngược lại là giá trị dự đoán cuối cùng.

CHƯƠNG 4. THỰC NGHIỆM VÀ ĐÁNG GIÁ

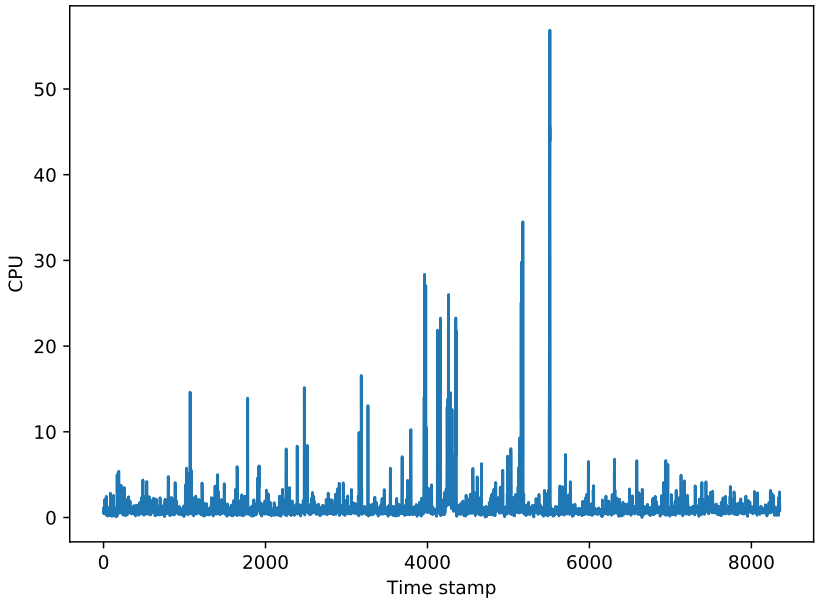
Với mô hình GAN cải tiến chúng tôi đã đề xuất ở trên, chúng tôi tiến hành thực nghiệm trên dữ liệu thực tế. Trong phần này, chúng tôi sẽ giới thiệu các dữ liệu mà chúng tôi đã thực nghiệm. Chúng tôi cũng đưa ra kết quả của việc tiền xử lý biến đổi chuỗi thời gian đã đề xuất. Tiếp đến, chúng tôi đưa ra các độ đo đánh giá và so sánh các mô hình. Các kết quả tối ưu siêu tham số cũng được thể hiện trong phần này. Và phần cuối cùng là kết quả của mô hình cải tiến GAN đề xuất, qua kết quả thực nghiệm chúng tôi đã đưa ra một số nhận xét về sự cải tiến sau thực nghiệm.

4.1 Dữ liệu

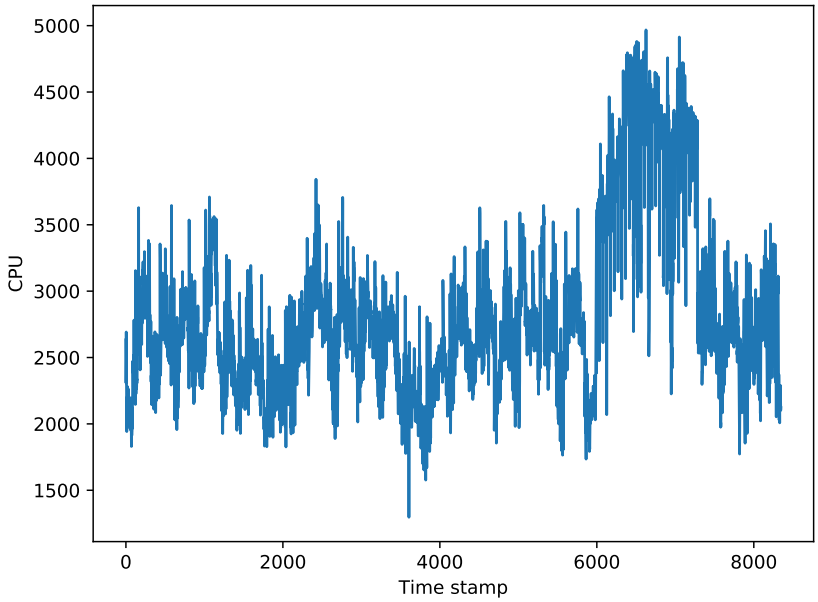
Trong nội dung đồ án này, chúng tôi thực hiện đánh giá mô hình thử nghiệm trên 5 bộ dữ liệu. Trong đó có 2 bộ dữ liệu được trích xuất từ bộ dữ liệu lịch sử của Google Cluster Trace, 2 bộ dữ liệu là dữ liệu lịch sử lưu lượng mạng tại EU và UK, và dữ liệu lịch sử số truy cập trong thời gian sự kiên FIFA World Cup năm 1998.

4.1.1 Google Cluster Trace

Google Cluster Trace là bộ dữ liệu thu thập lượng tài nguyên sử dụng trên cụm máy chủ được Google báo cáo năm 2011. Dữ liệu được thu thập với nhiều công việc trên 12,5 nghìn máy khác nhau trong vòng 29 ngày từ 01/05/2011. Mỗi công việc chứa nhiều task khác nhau và chạy đồng thời trên nhiều máy khác nhau. Bộ dữ liệu bao gồm có 20 thông số khác nhau như CPU, RAM, không gian lưu trữ.... Trong 29 ngày thu thập, dữ liệu có 1.232.799.308 bản ghi dữ liệu về tài nguyên sử dụng. Để tiến hành đánh giá sự hiệu quả của mô hình đề xuất với bộ dữ liệu các dấu vết của cụm máy chủ Google, chúng tôi tách thành 2 bộ dữ liệu trên thông số CPU là bộ dữ liệu trên một công việc (hình 4.1) và bộ dữ liệu trên tất cả các công việc (hình 4.2). Với bộ dữ liệu trên một công việc chúng tôi chọn công việc có ID là 6.176.858.948 vì công việc này có 25.954.362 bản ghi dữ liệu chạy trong suốt 29 ngày. Với bộ dữ liệu tất cả các công việc, chúng tôi tính tổng tài nguyên của tất cả các công việc tại một thời điểm.



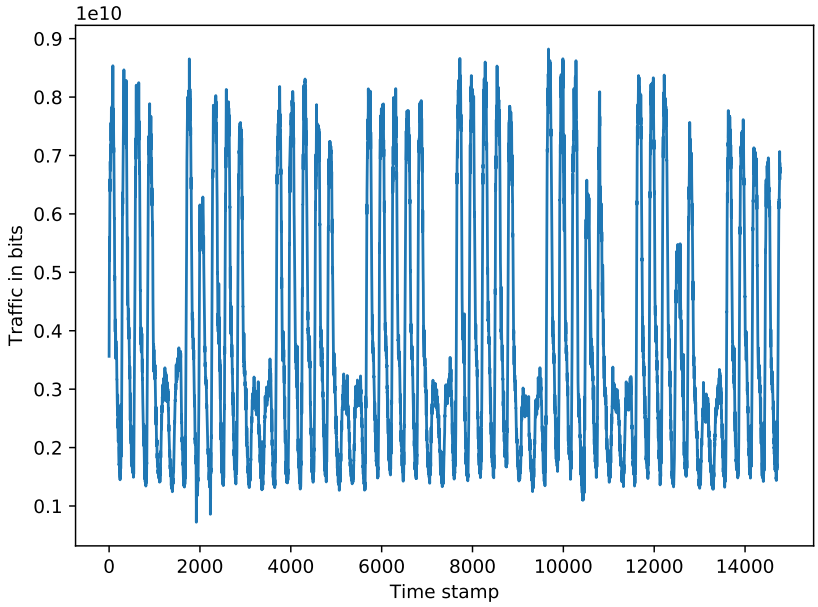
Hình 4.1: Bộ dữ liệu CPU trên một công việc



Hình 4.2: Bộ dữ liệu CPU trên tất cả công việc

4.1.2 Internet Traffic EU

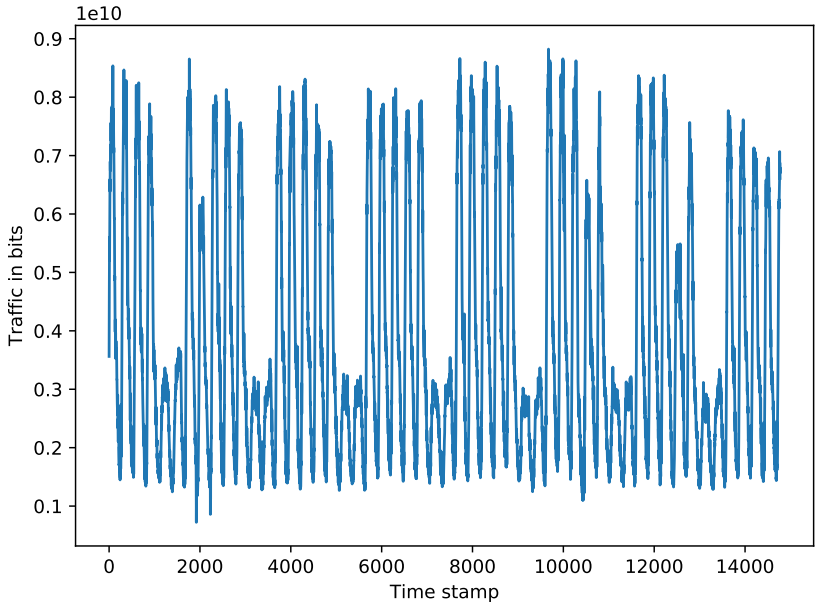
EU Internet traffic là bộ dữ liệu thu thập lưu lượng (theo đơn vị bits) từ một ISP nội bộ có trung tâm ở 11 thành phố châu Âu.Dữ liệu được thu thập từ 06:57 giờ ngày 7 tháng 6 đến 11:17 giờ ngày 31 tháng 7 năm 2005. Tập dữ liệu này có 14.772 điểm dữ liệu và các điểm dữ liệu được thu thập cách nhau 5 phút.



Hình 4.3: Bộ dữ liệu lưu lượng internet của EU

4.1.3 Internet Traffic UK

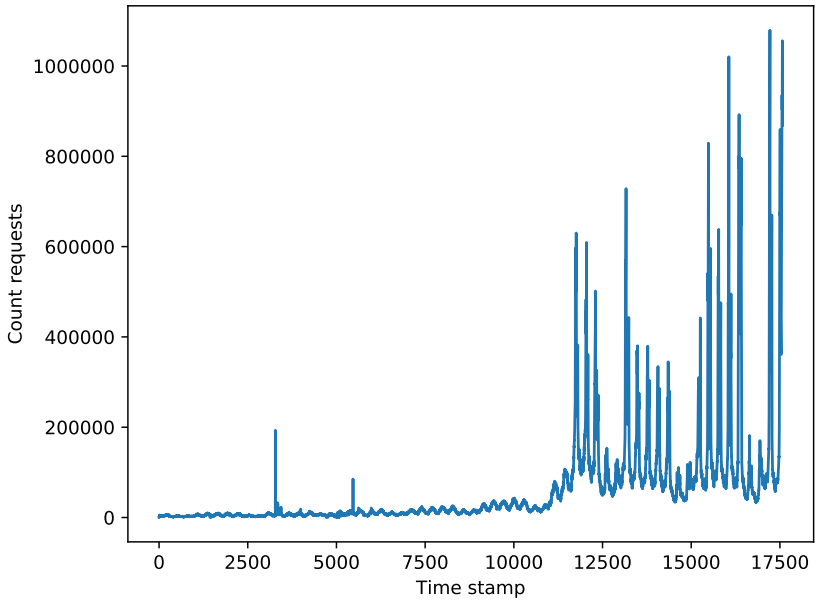
UK internet traffic là bộ dữ liệu lưu lượng internet (theo đơn vị bits) từ một ISP. Dữ liệu này là tổng lưu lượng internet của học viện Anh Quốc được thu thập từ 09:30 ngày 27 tháng 11 năm 2004 đến 11:11 ngày 27 tháng 1 năm 2005. Tập dữ liệu này có 19.888 điểm dữ liệu và các điểm dữ liệu được thu thập cách nhau 5 phút.



Hình 4.4: Bộ dữ liệu lưu lượng internet của UK

4.1.4 Wold Cup 98

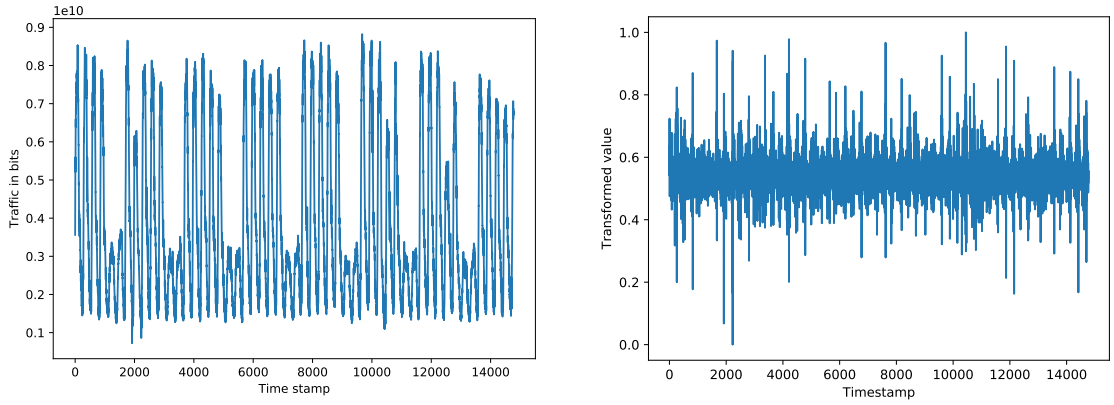
World Cup 98 là bộ dữ liệu lưu vết số lượng truy cập truy cập vào trang web của World Cup năm 1998 từ 30 tháng 4 đến 26 tháng 7 năm 1998. Trong khoảng thời gian diễn ra world cup 1998, trang web nhận được tổng cộng 1.352.804.107 lượt truy cập. Tập dữ liệu bao gồm 17573 điểm dữ liệu, mỗi điểm dữ liệu cách nhau 5 phút và có giá trị là số truy cập trong khoảng thời gian 5 phút.



Hình 4.5: Bộ dữ liệu World Cup 98

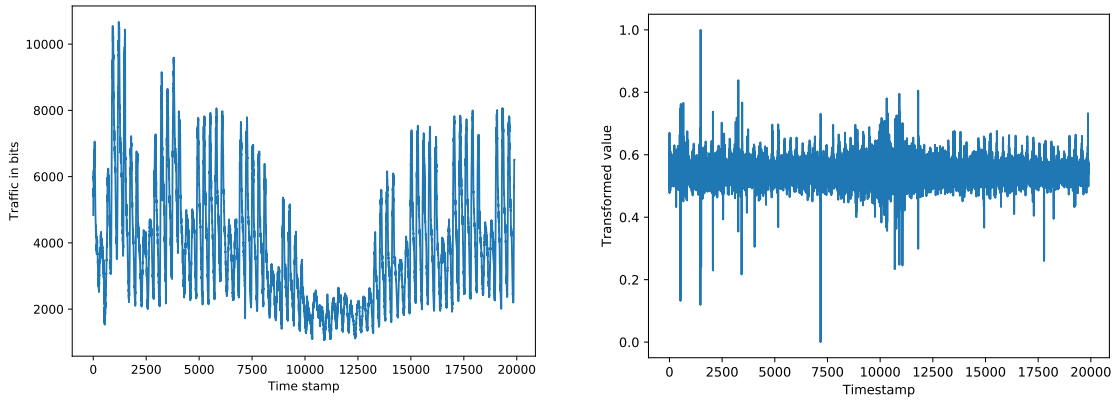
4.2 Tiền xử lý dữ liệu

Theo phương pháp tiền xử lý dữ liệu đã trình bày ở phần trước, chúng tôi đã biến đổi và thu được dữ liệu mới như trong hình các hình dưới đây. Thông qua các hình chuỗi sau biến đổi ta thấy rằng các thông tin về xu hướng của chuỗi đã được loại bỏ thể hiện rõ trong hình 4.10, và các thông tin về chu kỳ cũng đã được loại bỏ thể hiện rõ trong các hình 4.7, 4.6.



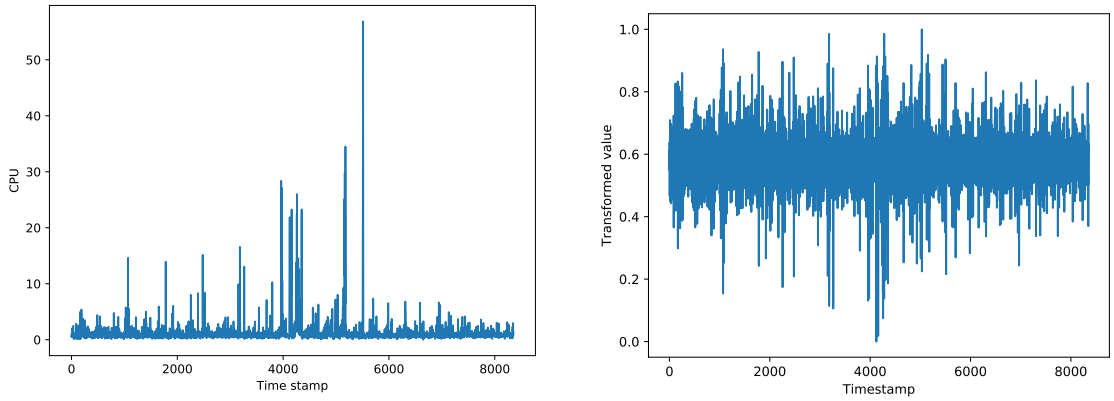
(a) Dữ liệu trước biến đổi (b) Dữ liệu sau biến đổi

Hình 4.6: Ví dụ biến đổi dữ liệu internet traffic EU



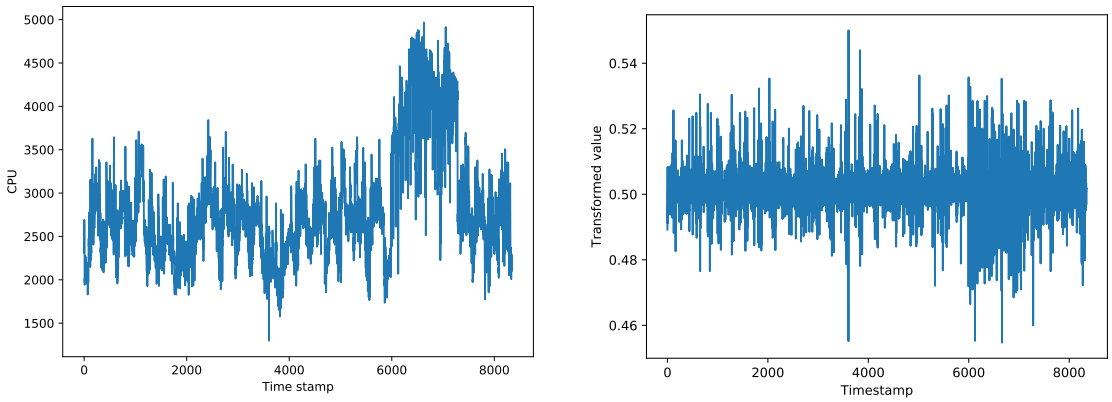
(a) Dữ liệu trước biến đổi (b) Dữ liệu sau biến đổi

Hình 4.7: Ví dụ biến đổi dữ liệu internet traffic UK



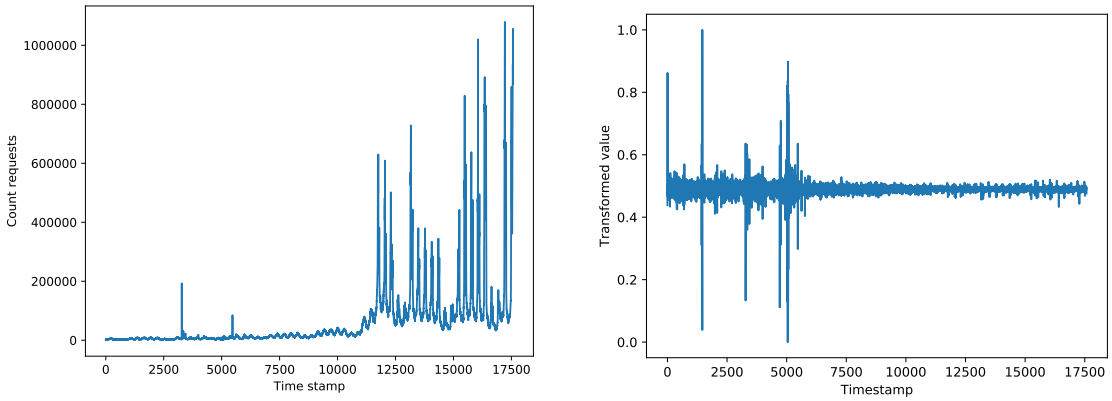
(a) Dữ liệu trước biến đổi (b) Dữ liệu sau biến đổi

Hình 4.8: Ví dụ biến đổi dữ liệu Google Trace trên một công việc



(a) Dữ liệu trước biến đổi (b) Dữ liệu sau biến đổi

Hình 4.9: Ví dụ biến đổi dữ liệu Google Trace tất cả công việc



(a) Dữ liệu trước biến đổi (b) Dữ liệu sau biến đổi

Hình 4.10: Ví dụ biến đổi dữ liệu World Cup 98

4.3 Độ đo đánh giá

Trong đồ án này để đánh giá và so sánh mô hình đề xuất với các mô hình khác, chúng tôi sử dụng 5 độ đo đánh giá.

- Trung bình sai khác (Mean absolute error - MAE):

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

- Căn trung bình bình phương sai khác (Root mean square error - RMSE):

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

- Trung bình phần trăm sai khác đối xứng(Mean absolute percentage error - MAPE):

$$\text{MAPE} = \frac{100}{n}\sum_{i=1}^{n}\frac{|y_i - \hat{y}_i|}{(|y_i| + |\hat{y}_i|)/2}\%$$

- Jensen–Shannon divergence - Jensen–Shannon divergence - JSD: Đây là độ đo thể hiện độ tương đồng giữa 2 phân phối

$$\text{JSD}(P\|Q) = \frac{1}{2}KL(P\|M) + \frac{1}{2}KL(Q\|M)$$

Trong đó:

$$\begin{cases} P, Q \text{ là phân phối của dữ liệu dự đoán và thực tế} \\ KL(P\|Q) = \sum_{x\in\mathcal{X}} P(x)\log\left(\frac{P(x)}{Q(x)}\right) \\ M = \frac{1}{2}(P+Q) \end{cases}$$

- Trung bình độ lệch chuẩn: để đánh giá độ hội tụ của các mô hình đối với các hàm chi phí đã đề xuất, chúng tôi sử dụng độ đo trung bình độ lệch chuẩn std

$$STD = \sum_{i=1}^{n} std_i$$

4.4 Kết quả

4.4.1 Kết quả tối ưu siêu tham số

Với mỗi bộ dữ liệu chúng tôi thực hiện tối ưu siêu tham số trong không siêu tham số đã nêu ở phần trước. Việc tối ưu siêu tham số được thực hiện trên 2 giải thuật là tối ưu bayes và giải thuật tối ưu bày đàn. Bảng 4.1 cho thấy kết quả đánh giá của bộ tham số tối ưu nhất tìm được của 2 giải thuật tối ưu. Qua kết quả trong bảng ta thấy rằng giải thuật PSO có thể tìm được lời giải cho các siêu tham số tốt hơn giải thuật tối ưu bayes. Và bảng 4.2 là kết quả các siêu tham số tối ưu nhất tương ứng với từng bộ dữ liệu.

Dataset	Metrics	Bayesian Optimization	PSO
Google Trace 1 job	mae	0.297	**0.295**
	smape	0.302	**0.300**
	rmse	0.472	**0.471**
	jsd	**0.167**	0.173
	std	0.259	**0.178**
Google Trace all job	mae	123.231	**123.018**
	smape	0.039	0.039
	rmse	**202.107**	202.900
	jsd	0.087	**0.086**
	std	80.661	**70.731**
Traffic EU	mae	9.22E+07	**9.11E+07**
	smape	0.028	0.028
	rmse	1.28E+08	**1.27E+08**
	jsd	**0.055**	0.065
	std	5.80E+07	**5.30E+07**
Traffic UK	mae	60.740	**59.224**
	smape	0.014	0.014
	rmse	83.416	**81.699**
	jsd	**0.044**	0.045
	std	59.042	**47.113**
World Cup 98	mae	7810.715	**7670.563**
	smape	0.042	0.042
	rmse	20321.138	**20090.935**
	jsd	0.066	**0.055**
	std	5714.144	**3488.192**

Bảng 4.1: Bảng so sánh kết quả của 2 giải thuật tối ưu siêu tham số Bayesian Optimization và PSO trên mô hình GAN với hàm chi phí kết hợp $L_{gan} + L_{mse}$

Hyperparameter	Google Trace 1 job	Google Trace all job	Traffic EU	Traffic UK	Wolrd Cup 98
layer_size_G	[2, 1]	[2, 1]	[2, 1]	[64, 1]	[16, 1]
layer_size_D	[16, 1]	[2, 1]	[8, 1]	[64, 1]	[64, 1]
cell_type_G	gru	lstm	lstm	lstm	gru
cell_type_D	gru	lstm	gru	lstm	gru
activation_G	tanh	tanh	tanh	tanh	tanh
activation_D	tanh	tanh	tanh	sigmoid	sigmoid
dropout_G	0.5	0.5	0.395	0.007	0.222
dropout_D	0	0	0.5	0.429	0
number_in	8	5	8	7	1
noise_shape	32	32	32	32	32
learning_rate_G	0.1	0.024	0.054	0.004	0.02
learning_rate_D	0.03	0.01	0.03	0.15	0.01
number_train_D	4	3	1	2	1
batch_size	16	32	16	8	32
α	0.6	0.4	0.7	0.6	0.3
β	0.4	0.1	0.3	0.4	0.5
γ	0.3	0.2	0.1	0.2	0.4

Bảng 4.2: Bảng kết quả tối ưu siêu tham số trên 5 tập dữ liệu.

4.4.2 Kết quả đánh giá mô hình

Như đã đề xuất, chúng tôi thực hiện mô hình GAN với 3 hàm chi phí trong quá trình huấn luyện mô hình. Hàm chi phí đầu tiên là hàm chi phí gốc của mạng GAN ban đầu, hàm chi phí thứ 2 là hàm chi phí của GAN kết hợp với hàm chi phí MSE và hàm chi phí cuối cùng là hàm chi phí kết hợp của hàm chi phí GAN , MSE và hàm chi phí theo hướng. Và chúng tôi thực nghiệm các mô hình mạng nơ-ron truyền thẳng và mạng nơ-ron hồi quy học sâu khá phổ biến hiện này để so sánh đánh giá với các mô hình đã đề xuất.

Kết quả thực nghiệm được thể hiện chi tiết trong bảng 4.3. Qua bảng này ta thấy rằng mô hình GAN đề xuất kết hợp với hàm chi phí mới cho kết quả tốt hơn so với các mô hình khác. Các kết quả dự đoán cũng được thể hiện trong các hình 4.11, 4.12, 4.13, 4.14, 4.15.

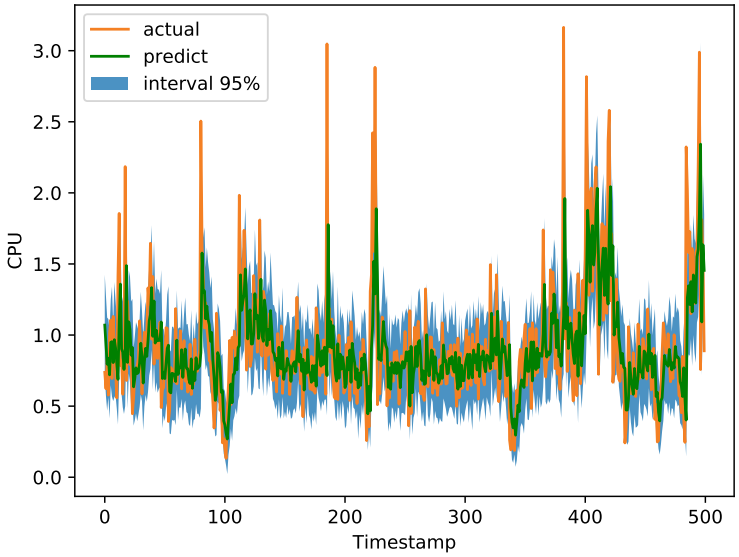
Với bộ dữ liệu google trace, mô hình chúng tôi đều xuất cho kết quả tốt hơn trên hầu hết các độ đo như trên bộ dữ liệu 1 công việc, mô hình GAN kết hợp 3 hàm chi phí cho mae tốt hơn mô hình LSTM 12%, smape tốt hơn 6%. Tuy nhiên trên dữ liệu tất cả công việc mô hình LSTM và GRU lại cho kết quả tốt hơn trên độ đo rmse và smape.

Với bộ dữ liệu lưu lượng mạng của EU và UK, kết quả cũng cho thấy mô hình chúng tôi đề xuất hoạt động hiệu quả hơn các mô hình khác. Cụ thể, mae của mô hình GAN kết hợp 3 hàm chi phí so với LSTM tốt hơn 6% hay rmse của mô hình GAN kết hợp 2 hàm chi phí 6%. Tuy nhiên trên độ đo jsd thì mô hình GRU cho kết quả tốt hơn.

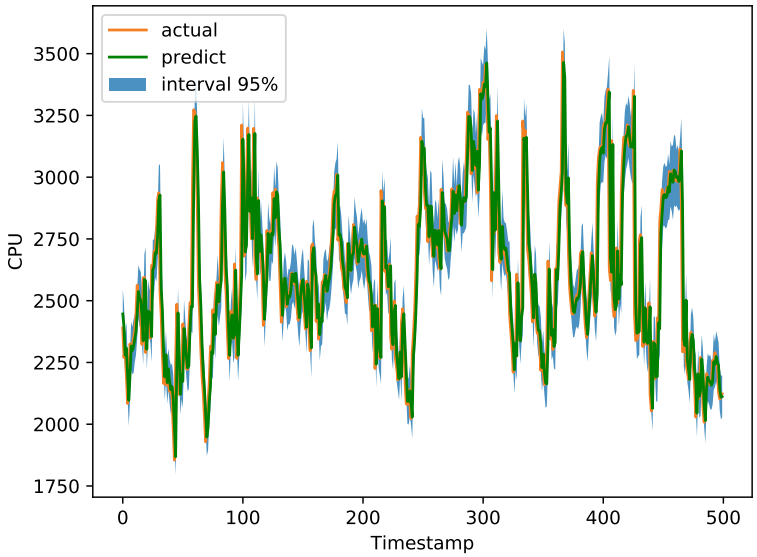
Cuối cùng với bộ dữ liệu world cup 98, mô hình chúng tôi đề xuất chỉ cho kết quả kém hơn ở độ đo smape và các độ đo khác đều cho kết quả tốt hơn.

Dataset	Metric	Model					
		LSTM	GRU	ANN	GAN		
					loss gan	loss gan + mse	loss gan + mse + loss direct
Google trace 1 job	mae	0.321	0.320	0.301	0.393	0.295	**0.294**
	smape	0.318	0.318	0.307	0.415	**0.300**	**0.300**
	rmse	0.502	0.503	0.479	0.574	**0.471**	0.472
	jsd	0.115	**0.112**	0.145	0.145	0.113	0.113
	std	-	-	-	2.597	**0.178**	0.202
Google trace all job	mae	123.675	123.426	139.524	1626.514	**123.018**	123.454
	smape	0.040	**0.039**	0.045	0.690	**0.039**	**0.039**
	rmse	**201.558**	202.151	212.173	1923.268	202.900	203.097
	jsd	0.108	0.098	0.119	0.375	0.086	**0.082**
	std	-	-	-	11919.603	**70.731**	140.863
Traffic EU	mae	9.61E+07	9.61E+07	9.55E+07	1.33E+08	9.11E+07	**9.07E+07**
	smape	0.029	0.029	0.029	0.038	**0.028**	**0.028**
	rmse	1.34E+08	1.34E+08	1.33E+08	1.83E+08	1.27E+08	**1.27E+08**
	jsd	0.023	**0.021**	0.033	0.082	0.025	0.026
	std	-	-	-	1.08E+09	5.30E+07	**4.85E+07**
Traffic UK	mae	62.434	62.531	66.063	102.550	**59.224**	59.309
	smape	0.015	0.015	0.016	0.024	**0.014**	**0.014**
	rmse	86.598	86.762	92.784	137.543	**81.699**	81.779
	jsd	0.014	**0.013**	0.045	0.081	0.015	0.014
	std	-	-	-	994.390	**47.113**	51.160
World cup 98	mae	7826.633	7874.528	9974.345	238530.213	7670.563	**7667.202**
	smape	**0.042**	**0.042**	0.050	0.806	**0.042**	**0.042**
	rmse	20271.649	20385.324	24807.277	427758.174	20090.935	**19817.535**
	jsd	0.059	0.059	0.071	0.424	**0.055**	0.065
	std	-	-	-	1424238.948	**3488.192**	9553.130

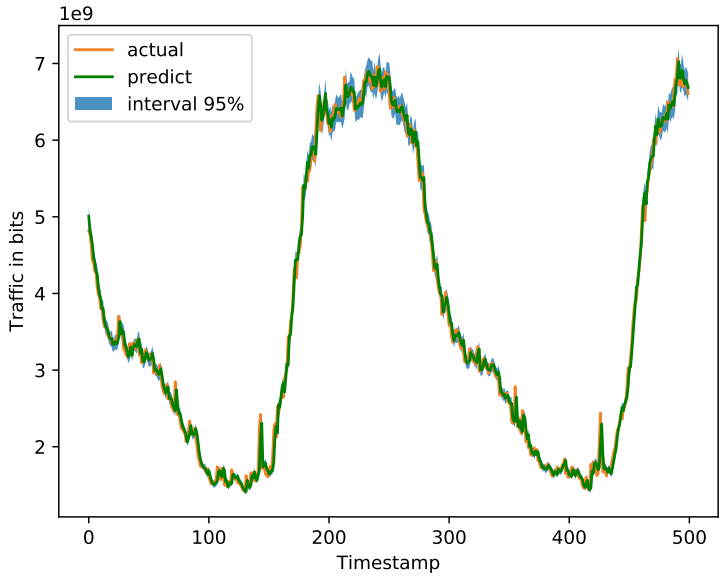
Bảng 4.3: Bảng kết quả



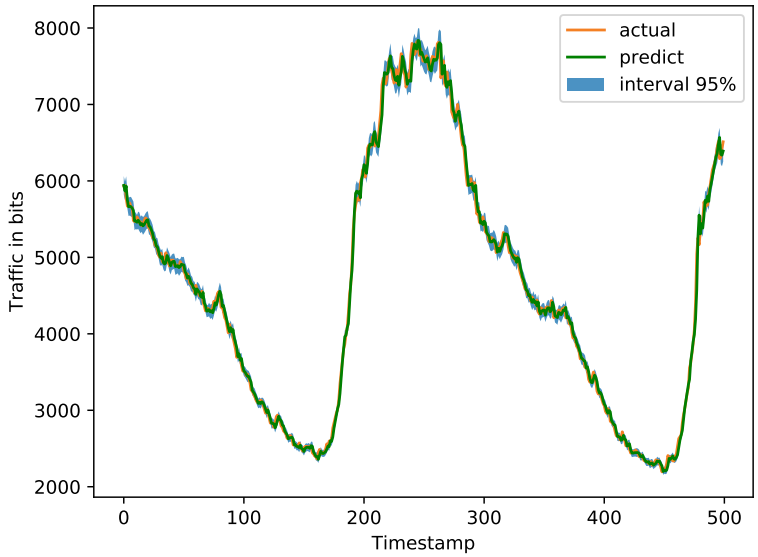
Hình 4.11: Dự đoán của tập dữ liệu Google Trace 1 công việc



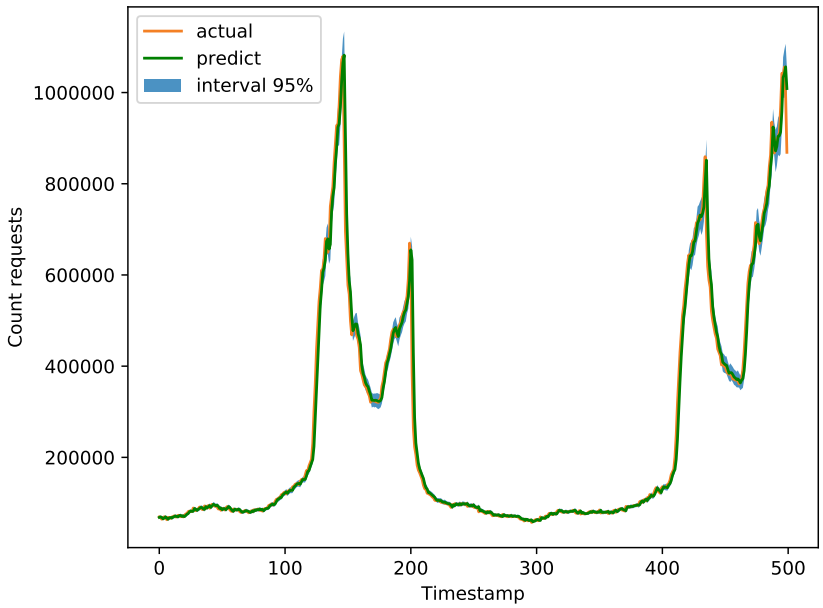
Hình 4.12: Dự đoán của tập dữ liệu Google Trace tất cả công việc



Hình 4.13: Dự đoán của tập dữ liệu lưu lượng của EU



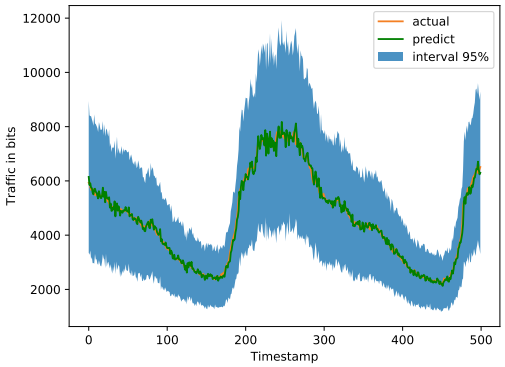
Hình 4.14: Dự đoán của tập dữ liệu lưu lượng của UK



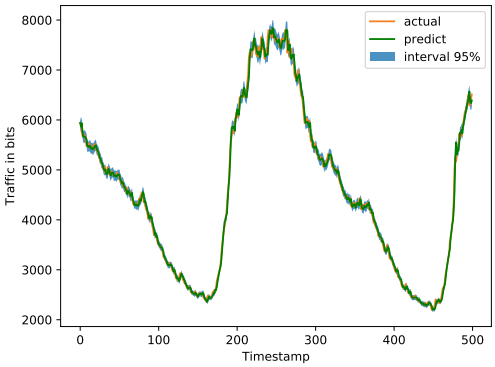
Hình 4.15: Dự đoán của tập dữ liệu World Cup 98

Hiệu quả của hàm chi phí mới được thể hiện rõ hơn trong hình 4.16. Khi thực hiện dự đoán chúng tôi thực hiện dựa đoán cùng 1 điều kiện đầu vào (khác vec-tơ nhiễu) lặp lại 100 lần, sau đó tính trung bình làm giá trị dự đoán và tính giá trị độ lệch chuẩn để vẽ vùng dự đoán màu xanh. Qua hình vẽ ta thấy rằng với hàm chi phí gốc của GAN, tuy đường dự đoán mà xanh lục khá sát với đường thực tế màu cam nhưng sai lệch của các dựa đoán vần rất lớn. Như vậy ta có thể thấy mô hình không ổn định. Còn đối với các mô hình có sử dụng hàm chi phí tổng hợp thì sai lệch khi

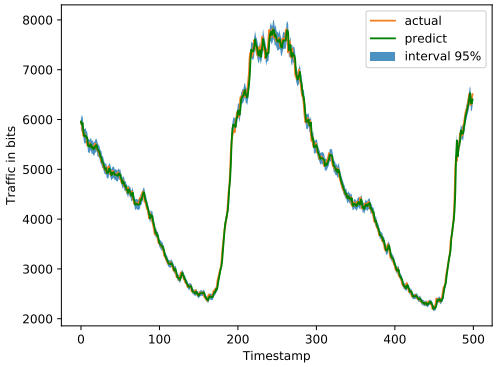
dự đoán đã nhỏ đi rất nhiều.



(a) Loss Gan



(b) Loss Gan + Loss Mse



(c) Loss Gan + Loss Mse + Loss direct

Hình 4.16: So sánh giữa mô hình với hàm chi phí gốc của GAN với các hàm chi phí đề xuất

CHƯƠNG 5. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

5.1 Kết luận

Trong đồ án này, chúng tôi đề xuất hệ thống dự đoán tài nguyên sử dụng trong tương lai cho các hệ thống cung cấp dịch vụ điện toán đám mây. Trong đó, chúng tôi đã đưa ra:

1. Các phép biến đổi tiền xử lý dữ liệu giúp cho việc huấn luyện mạng nơ-ron dễ dàng hơn.
2. Cải tiến mô hình Generative Adversarial Network - GAN để áp dụng cho bài toán phân tích chuỗi thời gian:
   (a) Xây dựng cấu trúc mạng nơ-ron hồi quy cho các thành phần của mạng GAN để học được các thông tin có sự phụ thuộc của chuỗi thời gian.
   (b) Cải tiến hàm chi phí mới trong quá trình học để áp dụng cho bài toán dự đoán dữ liệu chuỗi thời gian.
3. Thử nghiệm và so sánh các giải thuật tối ưu siêu tham số của mạng GAN đã đề xuất, đó là giải thuật tối ưu bayes và tối ưu bày đàn.
4. Áp dụng mô hình đề xuất vào bài toán dự đoán tài nguyên sử dụng của hệ thống điện toán đám mây.

Qua thực nghiệm với các bộ dữ liệu theo dõi tài nguyên sử dụng của cụm máy chủ Google, dữ liệu lưu lượng của EU và UK và dữ liệu về số lượng truy cập vào trang web của ban tổ chức World Cup năm 1998, mô hình cho kết quả khá tốt so với các mô hình dự đoán khác. Cụ thể mô hình chúng tôi đề xuất cho kết quả mae tốt hơn tất cả các mô hình khác trên tất cả các tập dữ liệu như trên tập dữ liệu google trace 1 công việc cho kết quả tốt hơn 12% so với LSTM, smape của mô hình đề xuất cho kết quả tốt hơn trên 4 tập dữ liệu, jsd của mô hình đề xuất cũng cho kết quả tốt hơn trên 3 tập dữ liệu. Từ kết quả dự đoán, các hệ thống điện toán đám mây có thể tăng hay giảm tài nguyên theo nhu cầu của khách hàng.

5.2 Hướng phát triển

Trong thời gian tới, chúng tôi sẽ tiếp tục pháp triển mô hình GAN áp dụng cho bài toán dự đoán chuỗi thời gian. Chúng tôi dự định xây dựng một bộ sinh G mới và sẽ áp dụng các giải thuật meta-heuristics vào bộ sinh G. Bên cạnh đó, chúng tôi cũng muốn xây dựng mô đun mở rộng tài nguyên cho hệ thống đã được đề xuất.

Tài liệu tham khảo

[1] J. Yang, C. Liu, Y. Shang, Z. Mao, and J. Chen, "Workload predicting-based automatic scaling in service clouds," in *2013 IEEE Sixth International Conference on Cloud Computing*, IEEE, 2013, pp. 810–815.

[2] M. Wajahat, A. Gandhi, A. Karve, and A. Kochut, "Using machine learning for black-box autoscaling," in *2016 Seventh International Green and Sustainable Computing Conference (IGSC)*, IEEE, 2016, pp. 1–8.

[3] N. Tran, T. Nguyen, B. M. Nguyen, and G. Nguyen, "A multivariate fuzzy time series resource forecast model for clouds using lstm and data correlation analysis," *Procedia Computer Science*, vol. 126, pp. 636–645, 2018.

[4] R. Fu, J. Chen, S. Zeng, Y. Zhuang, and A. Sudjianto, "Time series simulation by conditional generative adversarial net," *arXiv preprint arXiv:1904.11419*, 2019.

[5] E. L. Zec, H. Arnelid, and N. Mohammadiha, "Recurrent conditional gans for time series sensor modelling,"

[6] T. Harris, "Cloud computing-an overview," *Whitepaper, Torry Harris Bussiness Solutins (January 2010)*, 2009. [Online]. Available: `https://www.torryharris.com/downloads/Cloud-Computing-Overview.pdf`.

[7] H. Ganesan, "Scale new business peaks with amazon autoscaling," 2011.

[8] C. Vazquez, R. Krishnan, and E. John, "Time series forecasting of cloud data center workloads for dynamic resource provisioning.," *JoWUA*, vol. 6, no. 3, pp. 87–110, 2015.

[9] F. Jabari, A. Masoumi, and B. Mohammadi-ivatloo, "Long-term solar irradiance forecasting using feed-forward back-propagation neural network," in *3rd international conference of IEA technology and energy management. Shahid Beheshti University, Tehran*, 2017.

[10] J. Kumar, R. Goomer, and A. K. Singh, "Long short term memory recurrent neural network (lstm-rnn) based workload forecasting model for cloud datacenters," *Procedia Computer Science*, vol. 125, pp. 676–682, 2018.

[11] Y. Cheng, X. Zhou, S. Wan, and K.-K. R. Choo, "Deep belief network for meteorological time series prediction in the internet of things," *IEEE Internet of Things Journal*, 2018.

[12] R. Adhikari and R. K. Agrawal, "An introductory study on time series modeling and forecasting," *arXiv preprint arXiv:1302.6613*, 2013.

[13] C. Olah, "Understanding lstm networks," 2015. [Online]. Available: https://colah.github.io/posts/2015-08-Understanding-LSTMs/.

[14] C. W. John McGonagle and J. Khim, "Recurrent neural network," [Online]. Available: https://brilliant.org/wiki/recurrent-neural-network/.

[15] M. Nguyen, "Illustrated guide to lstm's and gru's: A step by step explanation," [Online]. Available: https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21.

[16] R. M. Neal, *Probabilistic inference using Markov chain Monte Carlo methods*. Department of Computer Science, University of Toronto Toronto, Ontario, Canada, 1993.

[17] M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul, "An introduction to variational methods for graphical models," *Machine learning*, vol. 37, no. 2, pp. 183–233, 1999.

[18] A. Doucet, N. De Freitas, and N. Gordon, "An introduction to sequential monte carlo methods," in *Sequential Monte Carlo methods in practice*, Springer, 2001, pp. 3–14.

[19] T. P. Minka, "Expectation propagation for approximate bayesian inference," in *Proceedings of the Seventeenth conference on Uncertainty in artificial intelligence*, Morgan Kaufmann Publishers Inc., 2001, pp. 362–369.

[20] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," in *international conference on machine learning*, 2016, pp. 1050–1059.

[21] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[22] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[23] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *arXiv preprint arXiv:1411.1784*, 2014.

[24] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein gan," *arXiv preprint arXiv:1701.07875*, 2017.

[25] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of wasserstein gans," in *Advances in neural information processing systems*, 2017, pp. 5767–5777.

[26] X. Mao, Q. Li, H. Xie, R. Y. Lau, Z. Wang, and S. Paul Smolley, "Least squares generative adversarial networks," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 2794–2802.

[27] D. Li, D. Chen, B. Jin, L. Shi, J. Goh, and S.-K. Ng, "Mad-gan: Multivariate anomaly detection for time series data with generative adversarial networks," in *International Conference on Artificial Neural Networks*, Springer, 2019, pp. 703–716.

[28] C. Esteban, S. L. Hyland, and G. Rätsch, "Real-valued (medical) time series generation with recurrent conditional gans," *arXiv preprint arXiv:1706.02633*, 2017.

[29] D. Senapati, "Grid search vs random search," 2018. [Online]. Available: `https://medium.com/@senapati.dipak97/grid-search-vs-random-search-d34c92946318`.

[30] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *Journal of Machine Learning Research*, vol. 13, no. Feb, pp. 281–305, 2012.

[31] J. Kennedy, "Particle swarm optimization," *Encyclopedia of machine learning*, pp. 760–766, 2010.

[32] Y. A. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller, "Efficient backprop," in *Neural networks: Tricks of the trade*, Springer, 2012, pp. 9–48.

[33] N. Rooy, "Particle swarm optimization from scratch with python," 2016. [Online]. Available: `https://nathanrooy.github.io/posts/2016-08-17/simple-particle-swarm-optimization-with-python/`.