HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

PHẠM VĂN HUẤN

NGHIÊN CỨU PHƯƠNG PHÁP PHÁT HIỆN BẤT THƯỜNG DỰA TRÊN DỮ LIỆU LOG HỆ THỐNG

ĐỀ ÁN

ĐỀ ÁN TỐT NGHIỆP THẠC SĨ KỸ THUẬT

(Theo định hướng ứng dụng)

HÀ NỘI – 2024

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

PHẠM VĂN HUẤN

NGHIÊN CỨU PHƯƠNG PHÁP PHÁT HIỆN BẤT THƯỜNG DỰA TRÊN DỮ LIỆU LOG HỆ THỐNG

CHUYÊN NGÀNH: HỆ THỐNG THÔNG TIN

MÃ SỐ: 8.48.01.04

ĐỀ ÁN TỐT NGHIỆP THẠC SĨ KỸ THUẬT

(Theo định hướng ứng dụng)

NGƯỜI HƯỚNG DẪN KHOA HỌC: TS NGUYỄN HUY TRUNG

HÀ NỘI – 2024

LỜI CAM ĐOAN

Tôi xin cam đoan theo hiểu biết tốt nhất của mình rằng đề án này không chứa bất kỳ tài liệu nào được xuất bản hoặc viết trước đây bởi người khác, ngoại trừ những tài liệu tham khảo thích hợp đã được trích dẫn đầy đủ và minh bạch trong nội dung của đề án.

Tác giả đề án tốt nghiệp

PHẠM VĂN HUẤN

LỜI CẢM ƠN

Em xin chân thành cảm ơn giảng viên hướng dẫn TS. Nguyễn Huy Trung đã giúp đỡ và định hướng cho em trong suốt quá trình học tập và thực hiện đề án tốt nghiệp.

Dưới sự hướng dẫn của TS. Nguyễn Huy Trung, em đã cố gắng hoàn thành tốt nhất có thể đề án tốt nghiệp này, tuy nhiên trong quá trình thực hiện không thể tránh được những thiếu sót, em rất mong nhận được sự góp ý của các thầy/cô trong hội đồng để em hoàn thiện hơn đề án tốt nghiệp này.

Em xin chân thành cảm ơn!

Học viên thực hiện

PHẠM VĂN HUẤN

MỤC LỤC

DANH MỤC CHỮ VÀ KÝ HIỆU VIẾT TẮT

Viết tắt	Tiếng Anh	Tiếng Việt
AI	Artificial intelligence	Trí tuệ nhân tạo
ANN	Artificial Neural Network	Mạng nơ-ron nhân tạo
CNN	Convolutional Neural Network	Mạng nơ-ron tích chập
DNN	Deep Neural Network	Mạng nơ ron sâu
GNN	Graph Neural Network	Mạng nơ ron đồ thị
GAT	Graph Attention Networks	Mạng chú ý đồ thị
LSTM	Long Short Term Memory networks	Mạng bộ nhớ ngắn dài
RNN	Recurrent neural networks	Mạng nơ ron hồi quy
SVM	Support vector machine	Thuật toán học máy

DANH MỤC BẢNG BIỂU

DANH MỤC HÌNH VẼ

LỜI MỞ ĐẦU

1. Lý do chọn của đề tài

Trong bối cảnh ngày nay, cùng với sự phát triển mạnh mẽ của công nghệ thông tin, quy mô và độ phức tạp của hệ thống thông tin ngày càng gia tăng, kéo theo những thách thức lớn về an toàn và bảo mật. Việc đảm bảo an toàn và bảo mật cho hệ thống thông tin là một yêu cầu cấp bách nhằm bảo vệ dữ liệu, ngăn chặn các hành vi xâm nhập và tấn công mạng. Đặc biệt, việc phát hiện sớm các sự kiện bất thường trong hệ thống thông tin đóng vai trò then chốt trong việc ngăn chặn và ứng phó hiệu quả với các mối đe dọa an ninh.

Trong nghiên cứu này, học viên tập trung vào việc nghiên cứu về phương pháp phát hiện bất thường dựa trên dữ liệu log hệ thống. Dữ liệu log hệ thống là nguồn thông tin hữu ích để giám sát và phát hiện bất thường, nó ghi lại trạng thái hệ thống và các sự kiện quan trọng tại các điểm then chốt khác nhau để giúp gỡ lỗi các vấn đề về hiệu suất và sự cố, đồng thời thực hiện phân tích nguyên nhân gốc rễ. Bên cạnh đó, log hệ thống cũng sẽ giúp nhận biết những biểu hiện tiềm ẩn của sự tấn công hoặc lỗi hệ thống.

Tuy nhiên, với sự phát triển mạnh mẽ của hệ thống thông tin hiện nay, đặc biệt trong các hệ thống thông tin quan trọng và cỡ lớn, lượng dữ liệu log hệ thống được tạo ra ngày càng nhiều và đa dạng. Do đó, việc phân tích thủ công là điều không khả thi, vì vậy cần phải có các phương pháp phân tích tự động, linh hoạt và chính xác, có khả năng phát hiện những biểu hiện bất bình thường trong dữ liệu log và đưa ra cảnh báo sớm, giúp quản trị hệ thống nhanh chóng ứng phó và ngăn chặn các rủi ro an ninh.

Dựa vào các phân tích trên, học viên tin tưởng đề án "***Nghiên cứu phương pháp phát hiện bất thường dựa trên dữ liệu log hệ thống"*** sẽ có ý nghĩa khoa học

và tính cấp thiết cao, từ đó góp phần nâng cao hiệu quả và an toàn cho các hệ thống thông tin.

2. Tổng quan về vấn đề nghiên cứu:

Có nhiều phương pháp khác nhau để phát hiện bất thường, một số phương pháp phổ biến bao gồm:

- Phân tích dữ liệu log hệ thống: Phương pháp này phân tích dữ liệu log để tìm ra các mẫu bất thường. Các mẫu bất thường có thể bao gồm các giá trị dữ liệu nằm ngoài phạm vi bình thường, các xu hướng dữ liệu bất thường hoặc các sự kiện không khớp với mô hình hành vi bình thường.
- Phân tích hành vi: Phương pháp này phân tích hành vi của người dùng hoặc hệ thống để tìm ra các hành vi bất thường. Các hành vi bất thường có thể bao gồm các truy cập hệ thống bất thường, việc sử dụng tài nguyên quá mức hoặc các thay đổi cấu hình bất thường.
- Phân tích mạng: Phương pháp này phân tích lưu lượng mạng để tìm ra các hoạt động bất thường. Các hoạt động bất thường có thể bao gồm các cuộc tấn công mạng, các cuộc truy cập từ xa không xác định hoặc các hoạt động sử dụng mạng bất thường.

Các nghiên cứu về phương pháp phát hiện bất thường dựa trên dữ liệu log đang tập trung vào các hướng sau:

- Sử dụng các kỹ thuật học máy: Các kỹ thuật học máy đang được sử dụng để phát triển các phương pháp phát hiện bất thường hiệu quả hơn. Các kỹ thuật học máy có thể học cách phân biệt dữ liệu bình thường và dữ liệu bất thường ngay cả khi dữ liệu log thay đổi theo thời gian.
- Sử dụng các kỹ thuật xử lý ngôn ngữ tự nhiên: Các kỹ thuật xử lý ngôn ngữ tự nhiên đang được sử dụng để phân tích dữ liệu log. Các kỹ thuật xử lý ngôn ngữ tự nhiên có thể giúp phát hiện các mối đe dọa bảo mật mới dựa trên các thông tin chi tiết trong dữ liệu log.

Tóm lại, phát hiện bất thường là một vấn đề nghiên cứu quan trọng trong lĩnh vực bảo mật thông tin. Nghiên cứu phương pháp phát hiện bất thường dựa trên dữ liệu log hệ thống đang được phát triển tích cực với nhiều hướng tiếp cận mới.

3. Mục đích nghiên cứu

Mục tiêu chính của đề án là đưa ra mô hình về phát hiện bất thường dựa trên dữ liệu log hệ thống, một số mục tiêu cụ thể như sau:

- Tiền xử lý dữ liệu
- Xây dựng mô hình phát hiện bất thường dựa trên dữ liệu log hệ thống (system log)
- Thực nghiệm và đánh giá kết quả

4. Đối tượng và phạm vi nghiên cứu

- Đối tượng: Dữ liệu log hệ thống (system log)

- Phạm vi nghiên cứu: Tập trung vào bài toán phát hiện bất thường dựa trên dữ liệu log hệ thống.

- Thời gian thực hiện: 15/12/2023 – 29/04/2024

5. Phương pháp nghiên cứu

- Nghiên cứu lý thuyết

Tiến hành nghiên cứu, khảo sát, tổng hợp, đánh giá các công trình nghiên cứu liên quan ở trong và ngoài nước để phân tích những vấn đề chưa giải quyết, những vấn đề cần tiếp tục nghiên cứu theo hướng của đề tài. Các công trình nghiên cứu được tìm kiếm tại các kho dữ liệu trực tuyến như:

- Google Scholar (https://scholar.google.com/)
- IEEE Xplore (https://ieeexplore.ieee.org/)

- Nghiên cứu thực nghiệm

Thu thập dữ liệu từ các kho lưu trữ công khai. Chia thành dữ liệu các tập huấn luyện và kiểm thử, sử dụng kỹ thuật kiểm thử chéo (cross-validation),… huấn luyện và thử nghiệm.

CHƯƠNG 1: TỔNG QUAN PHÁT HIỆN BẤT THƯỜNG VÀ DỮ LIỆU LOG HỆ THỐNG

1.1 Tổng quan về phát hiện bất thường

1.1.1 Định nghĩa về phát hiện bất thường?

Phát hiện bất thường là kỹ thuật xác định các hoạt động hoặc sự kiện khác biệt so với hành vi bình thường trong hệ thống. Mục tiêu của phát hiện bất thường là phát hiện sớm các mối đe dọa tiềm ẩn và ngăn chặn các vi phạm an ninh trước khi chúng gây ra thiệt hại.

1.1.2 Vai trò và ý nghĩa trong bảo mật hệ thống

Phát hiện bất thường là đóng vai trò quan trọng trong việc xây dựng một hệ thống thông tin an toàn và đáng tin cậy. Khi các hệ thống và các ứng dụng ngày càng trở nên phức tạp hơn bao giờ hết, chúng tiềm ẩn càng nhiều lỗi và lỗ hổng bảo mật có thể bị khai thác để tấn công gây hại hệ thống, đánh cắp dữ liệu. Do đó phát hiện bất thường kịp thời sẽ hỗ trợ bảo mật hệ thống thông tin một cách hiệu quả, các vai trò bao gồm như sau:

- Phát hiện sớm các mối đe dọa: Phát hiện bất thường có thể giúp phát hiện các dấu hiệu xâm nhập hoặc tấn công ngay từ giai đoạn đầu, khi kẻ tấn công mới bắt đầu thực hiện hành vi xâm hại. Điều này cho phép các tổ chức có thời gian để phản ứng và ngăn chặn mối đe dọa trước khi nó gây ra thiệt hại nghiêm trọng.
- Giảm thiểu thiệt hại: Việc phát hiện sớm các mối đe dọa giúp giảm thiểu thiệt hại do vi phạm an ninh gây ra. Ví dụ, nếu một cuộc tấn công được phát hiện sớm, tổ chức có thể ngăn chặn việc đánh cắp dữ liệu hoặc ngăn chặn sự lây lan của phần mềm độc hại.
- Cải thiện hiệu quả hoạt động: Phát hiện bất thường có thể giúp cải thiện hiệu quả hoạt động của hệ thống thông tin bằng cách xác định các vấn đề tiềm ẩn

trước khi chúng gây ra sự cố. Ví dụ, hệ thống có thể phát hiện các hoạt động bất thường của người dùng có thể dẫn đến lỗi hệ thống hoặc sự cố mạng.

- Tăng cường khả năng tuân thủ: Một số quy định và tiêu chuẩn tuân thủ yêu cầu các tổ chức phải triển khai hệ thống phát hiện bất thường để bảo vệ dữ liệu và hệ thống của họ.
- Bảo vệ dữ liệu: Dữ liệu là tài sản quý giá của các tổ chức. Phát hiện bất thường giúp bảo vệ dữ liệu khỏi bị đánh cắp, sử dụng trái phép hoặc bị phá hủy.
- Bảo vệ hệ thống: Hệ thống thông tin là cơ sở hạ tầng quan trọng cho hoạt động kinh doanh của các tổ chức. Phát hiện bất thường giúp bảo vệ hệ thống khỏi bị tấn công, xâm nhập hoặc phá hoại.

1.1.3 Thách thức và rủi ro

Mặc dù phát hiện bất thường mang lại nhiều lợi ích cho bảo mật hệ thống, nhưng nó cũng đi kèm với một số thách thức và rủi ro nhất định cần được xem xét như sau:

Thách thức:

- **Khối lượng dữ liệu khổng lồ và tốc độ gia tăng:**

  - Hệ thống thông tin hiện đại tạo ra một lượng dữ liệu khổng lồ với tốc độ gia tăng theo cấp số nhân. Việc xử lý và phân tích hiệu quả khối lượng dữ liệu khổng lồ này trong thời gian thực là một thách thức lớn đối với các công cụ phát hiện bất thường.
  - Các công cụ cần có khả năng mở rộng và thích ứng với tốc độ gia tăng của dữ liệu để đảm bảo hiệu quả phát hiện.
  - Việc thiếu hụt cơ sở hạ tầng và nguồn lực tính toán cũng có thể ảnh hưởng đến khả năng xử lý dữ liệu của các công cụ.

- **Thiếu dữ liệu tham chiếu và sự đa dạng của hành vi:**

- Việc thiếu dữ liệu tham chiếu về hành vi người dùng và hệ thống bình thường có thể khiến các công cụ phát hiện bất thường gặp khó khăn trong việc xác định các mẫu bất thường.
- Hành vi của người dùng và hệ thống có thể thay đổi theo thời gian, theo khu vực, văn hóa và ngữ cảnh sử dụng, dẫn đến sự đa dạng trong các mẫu hành vi bình thường.
- Việc thiếu dữ liệu tham chiếu cho các nhóm người dùng hoặc ngữ cảnh cụ thể có thể dẫn đến tỷ lệ báo động giả cao.

- **Sự tinh vi của các mối đe dọa và chiến thuật tấn công:**

- Các hacker và kẻ tấn công không ngừng phát triển các phương pháp tấn công mới, tinh vi hơn để lẩn tránh các hệ thống phát hiện bất thường.
- Các mối đe dọa có thể sử dụng các kỹ thuật che giấu, nhiễu loạn dữ liệu hoặc mô phỏng hành vi bình thường để qua mặt các công cụ phát hiện.
- Việc thiếu khả năng cập nhật và thích ứng với các mối đe dọa mới có thể khiến các công cụ trở nên lỗi thời và không hiệu quả.

- **Tính phức tạp của hệ thống thông tin và mối tương quan dữ liệu:**

- Hệ thống thông tin hiện đại thường có cấu trúc phức tạp với nhiều thành phần và mối tương quan dữ liệu đa dạng.
- Việc xác định chính xác nguyên nhân gốc rễ của các hành vi bất thường trong môi trường hệ thống phức tạp có thể là một thách thức lớn.
- Các công cụ cần có khả năng phân tích mối tương quan trong dữ liệu hiệu quả để xác định chính xác các hành vi bất thường và giảm thiểu tỷ lệ báo động giả.

- **Yêu cầu về chuyên môn và nguồn lực:**

- Việc triển khai, vận hành và bảo trì hiệu quả các công cụ phát hiện bất thường đòi hỏi chuyên môn cao về bảo mật hệ thống, phân tích dữ liệu và quản trị hệ thống.
- Các tổ chức cần có đội ngũ nhân viên có trình độ và nguồn lực tài chính để đầu tư, triển khai và vận hành các công cụ này.
- Việc thiếu hụt chuyên môn và nguồn lực có thể ảnh hưởng đến hiệu quả hoạt động và khả năng duy trì của hệ thống phát hiện bất thường.

Rủi ro:

- **Báo động giả và lãng phí tài nguyên:**

  - Tỷ lệ báo động giả cao có thể dẫn đến lãng phí thời gian và nguồn lực của các nhà phân tích bảo mật, khiến họ tập trung vào các sự kiện không quan trọng và bỏ sót các mối đe dọa thực sự.
  - Việc điều tra và xử lý các báo động giả liên tục có thể gây ra tình trạng quá tải cho nhóm bảo mật và ảnh hưởng đến hiệu quả hoạt động chung.
  - Báo động giả có thể gây hoang mang và lo lắng cho người dùng, ảnh hưởng đến niềm tin của họ vào hệ thống bảo mật.

- **Vi phạm quyền riêng tư và rủi ro pháp lý:**

  - Việc thu thập và phân tích dữ liệu người dùng cho mục đích phát hiện bất thường có thể dẫn đến vi phạm quyền riêng tư nếu không được thực hiện đúng cách và tuân thủ các quy định về bảo vệ dữ liệu.
  - Việc sử dụng dữ liệu người dùng cho các mục đích khác ngoài phát hiện bất thường mà không có sự đồng ý của họ có thể dẫn đến các vấn đề pháp lý và vi phạm quyền riêng tư.
  - Các tổ chức cần tuân thủ các luật bảo vệ dữ liệu như GDPR và CCPA khi thu thập, sử dụng và lưu trữ dữ liệu người dùng cho mục đích phát hiện bất thường.

- **Lạm dụng và sử dụng sai mục đích:**

  - Các công cụ phát hiện bất thường có thể bị lạm dụng cho các mục đích độc hại như theo dõi người dùng trái phép, thu thập các thông tin.

1.2 Dữ liệu log hệ thống

1.2.1 Định nghĩa về dữ liệu log hệ thống

Dữ liệu log hệ thống (hay còn gọi là log system, log file) là các tập tin văn bản ghi lại các hoạt động, sự kiện và trạng thái của hệ thống thông tin trong một khoảng thời gian nhất định. Dữ liệu log được tạo ra bởi các thành phần khác nhau của hệ thống, bao gồm hệ điều hành, ứng dụng, dịch vụ và phần mềm bảo mật. Dữ liệu log thường được lưu trữ trong các tệp văn bản, nhưng cũng có thể được lưu trữ trong cơ sở dữ liệu hoặc các định dạng khác. Dữ liệu log hệ thống là loại dữ liệu quan trọng trong bảo mật và giám sát, cung cấp lịch sử đầy đủ của các sự kiện theo thời gian. Ngoài dữ liệu log của hệ điều hành, log còn được sử dụng trong các ứng dụng, trình duyệt web, phần cứng và thậm chí cả email.

1.2.2 Cấu trúc và đặc điểm

Về cấu trúc của log hệ thống có thể khác nhau tùy thuộc vào hệ điều hành, ứng dụng và phần mềm tạo ra nó. Tuy nhiên, nhìn chung, các bản ghi log thường bao gồm các thông tin sau:

- Thời gian: Thời điểm xảy ra sự kiện được ghi lại.
- Mức độ nghiêm trọng: Mức độ nghiêm trọng của sự kiện (ví dụ: thông tin, cảnh báo, lỗi).
- Thành phần: Thành phần của hệ thống tạo ra bản ghi log.
- Sự kiện: Mô tả chi tiết về sự kiện đã xảy ra.
- Dữ liệu bổ sung: Thông tin bổ sung có thể hữu ích cho việc phân tích sự kiện

Ngoài ra, cấu trúc log hệ thống có thể có các thông tin bổ sung liên quan đến thiết bị hoặc người dùng sau:

- Địa chỉ IP: Địa chỉ IP của thiết bị hoặc người dùng liên quan đến sự kiện.
- Tên người dùng: Tên người dùng của người dùng liên quan đến sự kiện.
- Mã lỗi: Mã lỗi nếu có lỗi xảy ra.
- Dữ liệu yêu cầu: Dữ liệu được gửi đến hệ thống trong trường hợp yêu cầu HTTP.
- Dữ liệu phản hồi: Dữ liệu được trả về bởi hệ thống trong trường hợp yêu cầu HTTP.
- ID phiên: ID phiên duy nhất cho mỗi phiên sử dụng.
- ID truy vấn: ID truy vấn duy nhất cho mỗi truy vấn được gửi đến hệ thống.
- Tên máy chủ: Tên máy chủ của máy chủ tạo ra bản ghi log.
- Thông tin hệ thống: Thông tin về hệ điều hành, phiên bản phần mềm và các thông số cấu hình khác.

Thông thường, log thường là dữ liệu ở dạng văn bản được in ra bởi các câu lệnh log (ví dụ: printf(), Console.Writeline()) in mã nguồn). Hình 1.1 minh hoạ một log message ghi lại một sự kiện cụ thể trong hệ thống với các trường như: mốc thời gian xảy ra sự kiện (2008-11-09 20:46:55,556), mức độ nghiêm trọng của sự kiện (INFO) và mô tả chi tiết sự kiện.

```
2008-11-09 20:46:55,556 INFO dfs.DataNode$PacketResponder:
Received block blk_3587508140051953248 of size 67108864 fr
om /10.251.42.84
```

Hình 1.1: Minh hoạ một log message

Về đặc điểm của log hệ thống bao gồm như sau:

- **Khối lượng lớn:**
  - Log hệ thống có thể tạo ra một lượng lớn dữ liệu trong một thời gian ngắn.
  - Lượng dữ liệu log phụ thuộc vào nhiều yếu tố như kích thước hệ thống, mức độ hoạt động của người dùng, số lượng ứng dụng và các dịch vụ đang

chạy trên hệ thống. (Ví dụ: một hệ thống máy chủ web bận rộn có thể tạo ra hàng gigabyte dữ liệu log mỗi ngày)

- **Tốc độ thay đổi nhanh:**

  - Log hệ thống được tạo ra liên tục khi hệ thống hoạt động.
  - Mỗi khi có sự kiện xảy ra, một bản ghi log mới sẽ được tạo ra.
  - Tốc độ thay đổi của log hệ thống phụ thuộc vào mức độ hoạt động của hệ thống. (Ví dụ: một hệ thống xử lý giao dịch tài chính có thể tạo ra hàng triệu bản ghi log mỗi giây)

- **Tính đa dạng:**

  - Log hệ thống có thể chứa nhiều loại thông tin khác nhau từ các thành phần khác nhau của hệ thống.
  - Các loại thông tin phổ biến trong log hệ thống bao gồm:
    - Thông tin chung về sự kiện (thời gian, mức độ nghiêm trọng, thành phần, sự kiện)
    - Thông tin bổ sung về sự kiện (địa chỉ IP, tên người dùng, mã lỗi, dữ liệu yêu cầu, dữ liệu phản hồi, ID phiên, ID truy vấn, tên máy chủ, thông tin hệ thống)
    - Thông tin chi tiết về hoạt động của hệ thống (sử dụng CPU, sử dụng bộ nhớ, sử dụng mạng, trạng thái dịch vụ)
    - Thông tin về các sự kiện bảo mật (cố gắng đăng nhập thất bại, truy cập tệp trái phép, tấn công mạng)

- **Tính phức tạp:**

  - Việc phân tích log hệ thống có thể phức tạp do khối lượng dữ liệu lớn và tính đa dạng của thông tin.

- Việc xác định thông tin quan trọng và giải thích ý nghĩa của các bản ghi log có thể là một thách thức.
- Cần có kiến thức về hệ thống, ứng dụng và phần mềm để phân tích log hệ thống hiệu quả.

- **Tính thời gian thực:**

- Một số hệ thống có thể tạo ra log hệ thống trong thời gian thực.
- Điều này cho phép giám sát hệ thống và phát hiện các vấn đề tiềm ẩn trong thời gian thực.
- Tuy nhiên, việc thu thập và xử lý log hệ thống thời gian thực đòi hỏi nhiều tài nguyên tính toán.

- **Tính bảo mật:**

- Log hệ thống có thể chứa thông tin nhạy cảm về người dùng và hệ thống.
- Do đó, việc thu thập, lưu trữ và sử dụng log hệ thống cần được thực hiện một cách an toàn và tuân thủ các quy định về bảo mật dữ liệu.

- **Khả năng lưu trữ:**

- Log hệ thống cần được lưu trữ để có thể truy vấn và phân tích sau này.
- Việc lưu trữ log hệ thống có thể tốn kém vì khối lượng dữ liệu lớn.
- Có nhiều giải pháp lưu trữ log hệ thống khác nhau, bao gồm lưu trữ cục bộ, lưu trữ đám mây và lưu trữ lưu trữ.

- **Khả năng truy vấn:**

- Log hệ thống cần có thể truy vấn để có thể tìm kiếm thông tin cụ thể.

- Có nhiều công cụ truy vấn log hệ thống khác nhau, bao gồm các công cụ dựa trên văn bản, các công cụ dựa trên SQL và các công cụ dựa trên NoSQL.

- **Khả năng phân tích:**

- Log hệ thống cần được phân tích để có thể trích xuất thông tin hữu ích.
- Có nhiều công cụ phân tích log hệ thống khác nhau, bao gồm các công cụ thống kê, các công cụ học máy và các công cụ khai phá dữ liệu.

- **Khả năng hiển thị:**

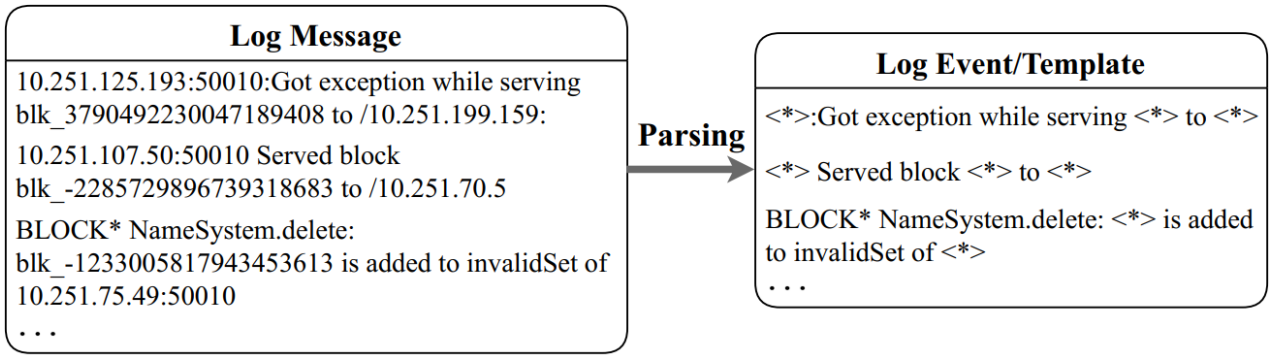
- Log hệ thống cần được hiển thị để có thể dễ dàng xem và hiểu.
- Có nhiều công cụ hiển thị log hệ thống khác nhau, bao gồm các bảng điều khiển, các biểu đồ và các báo cáo.

1.3 Liên kết phát hiện bất thường và dữ liệu log hệ thống

Các hệ thống và phần mềm ngày nay ngày càng trở nên lớn mạnh và lượng thông tin được lưu trữ bởi các hệ thống này là rất lớn. Điều này dẫn đến nhu cầu đảm bảo tính bảo mật, tính sẵn sàng và độ tin cậy của chúng. Một trong những thách thức của việc này là nhanh chóng xác định được lỗi trong hệ thống. Do đó, với việc theo dõi log hiệu quả, hệ thống có thể được phòng ngừa hoặc nhanh chóng phát hiện và ứng phó, sửa chữa các lỗi trong hệ thống. Có phương pháp theo dõi log tối ưu giúp giảm thời gian chết và giảm thiểu nguy cơ mất dữ liệu. Chính vì vậy đối với những hệ thống lớn, log thường được gửi đến một máy chủ an toàn hoạt động như một điểm thu thập chung trước khi được quản trị viên hệ thống xử lý thêm.

Dữ liệu log hầu hết là không có cấu trúc, điều này làm cho việc phát hiện bất thường dựa trên log trở nên khó khăn. Yêu cầu cơ bản để phát hiện bất thường chính xác là có thể cấu trúc hiệu quả các log template và phân loại chúng thành các nhóm.

Quá trình chuyển đổi các log từ này được gọi là phân tích cú pháp log – log parsing. Nội dung của log gồm 2 thành phần: Phần cố định (các từ khóa thể hiện các mẫu sự kiện - event template) và phần biến thay đổi (các tham số chứa các thông tin về thời gian chạy). Hình 1.2 cho thấy một đoạn log được tạo bởi HDFS (Hệ thống tệp phân tán Hadoop). Log parsing tự động chuyển đổi từng log message thành mẫu sự kiện cụ thể bằng cách loại bỏ các tham số và giữ lại các từ khóa. Ví dụ như hình 1.2, log template "<*> :Got exception while serving <*> to <*>" được phân tách được từ log message đầu tiên, trong đó <*> thể hiện vị trí của một tham số.



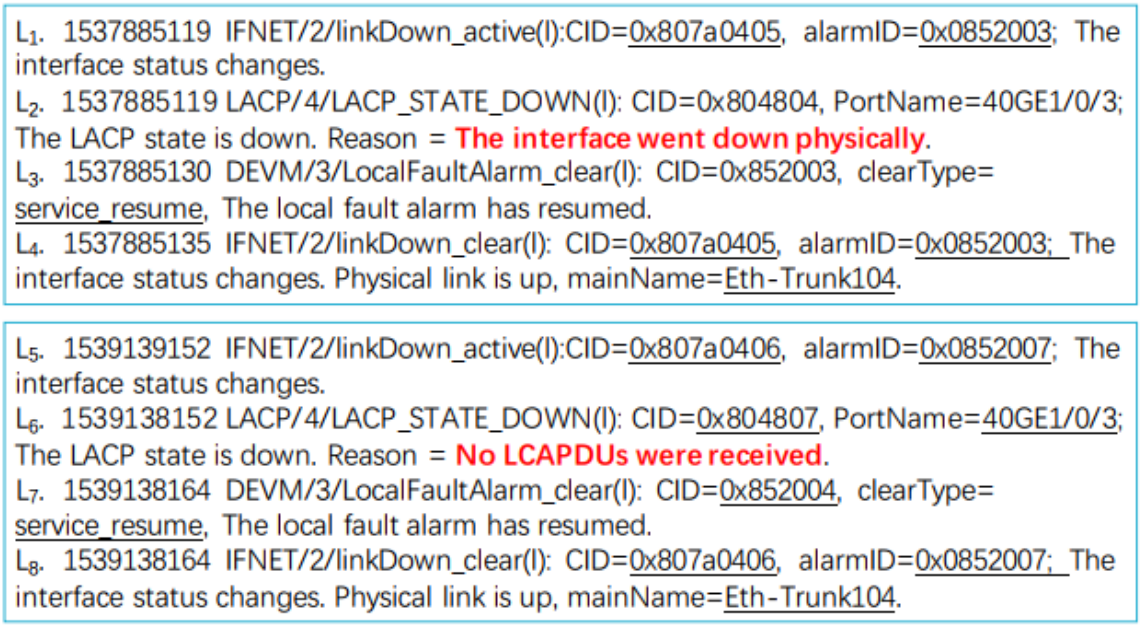
Hình 1.2: Ví dụ về thông báo log và log event/log template

Bằng việc sử dụng log parsing, dữ liệu log được nhóm lại thành các nhóm theo từng log template giúp làm giảm độ phức tạp, tạo thuận lợi cho việc phát hiện bất thường dựa trên phân tích log.

Log ghi lại các trạng thái hệ thống và các sự kiện quan trọng tại các điểm quan trọng khác nhau để giúp gỡ lỗi các sự cố và lỗi hiệu suất, đồng thời thực hiện phân tích nguyên nhân gốc rễ. Dữ liệu log như vậy có sẵn phổ biến trong hầu hết các hệ thống thông tin và là tài nguyên quý giá để hiểu trạng thái hệ thống. Hơn nữa, vì log ghi lại các sự kiện đáng chú ý khi chúng xảy ra từ các quy trình đang chạy, đây là nguồn thông tin tuyệt vời để theo dõi hệ thống theo thời gian thực và phát hiện bất thường.

Một số hệ thống lớn thường được phát triển/vận hành bởi hàng trăm kỹ sư/nhà phát triển. Thông thường một kỹ sư khi vận hành hệ thống, nếu không có đầy đủ

thông tin về tổng thể hệ thống có xu hướng xác định các bất thường từ góc độ cục bộ do đó dễ gặp lỗi. Ngoài ra, việc phát hiện bất thường bằng cách thủ công đang trở nên không khả thi do sự bùng nổ của số lượng log được sinh ra. Trên thực tế, các cách truyền thống để xử lý sự bất thường từ log rất kém hiệu quả. Người vận hành tiến kiểm tra nhật ký hệ thống theo cách thủ công bằng cách đối sánh biểu thức chính quy (Regular Expression – RegEx) hoặc tìm kiếm từ khóa (ví dụ: “fail”, “kill”, “down”) để phát hiện sự bất thường dựa trên kiến thức, kinh nghiệm của họ. Tuy nhiên, loại phát hiện bất thường này không thể áp dụng cho các hệ thống quy mô lớn.



Hình 1.3: Log của một switch với lỗi link flapping

Nhà phát triển phải tìm kiếm hàng triệu dòng log ngay sau khi một lỗi mới hoặc một lỗi tương tự xảy ra nhưng với một thông báo log khác, dẫn đến việc xác định bất thường bằng RegEx có thể không chính xác. Các sự kiện nhật ký được tạo thường có ở dạng text tự do, không theo cấu trúc, kèm theo thông tin về thời gian thực hiện, cấp độ của sự kiện và thông tin về thành phần sinh ra log. Để có thể phát hiện log, một người vận hành cần rất nhiều kinh nghiệm và hiểu biết, kiến thức vững chắc về hệ thống mới có thể phát hiện và khắc phục được lỗi, khiến cho là việc khắc phục sự cố rất tốn kém thời gian và cần nhiều nhân lực.

Phát hiện bất thường dựa trên log hệ thống thông qua tìm kiếm các từ khoá có thể cung cấp những gợi ý, thông tin về nguyên nhân và giúp mổ xẻ các sự kiện lỗi. Thông thường, các từ khoá, RegEx cụ thể của tên miền và các công cụ từ những thông báo log lỗi trước đó sẽ được lưu lại và sử dụng để tìm kiếm nhằm tìm kiếm bất thường. Tuy nhiên việc này cần được cập nhật liên tục để đảm bảo tính đúng đắn, phù hợp với hiện trạng của hệ thống. Ngoài ra, các thông báo log mới nếu chưa từng xuất hiện trước đó có thể dễ dàng bị bỏ qua với phương pháp này. Ngược lại, phát hiện bất thường hướng đến việc tìm kiếm những mẫu log chưa xuất hiện trước đó hay chứa các từ có nội dung khác thường có hiệu quả cao hơn.

Việc tìm kiếm dựa trên các từ khoá (ví dụ: “fail”, “kill”, “down”) và RegEx để phát hiện lỗi có thể bỏ qua một số lượng lớn hành vi bất thường. Những bất thường này chỉ có thể được suy ra dựa trên trình tự log của chúng có chứa nhiều log vi phạm các quy tắc thông thường. Ví dụ, bốn dòng log đầu tiên trong Hình 1.3 hiển lỗi link-flapping trên switch, được cho là không có bất thường. Nếu áp dụng so sánh từ khóa để phát hiện sự bất thường của log, cả L1 và L2, đều chứa từ khóa “down”, sẽ kích hoạt cảnh báo sai. Tuy nhiên, nó thực sự là một sự kiện bình thường vì switch tự động phục hồi nhanh chóng thể hiện tại dòng L4. Ở tình huống khác, phương pháp thủ công sẽ mất rất nhiều thời gian. Vấn đề thậm chí có thể không nằm trong thông báo lỗi và cảnh báo mà nó có thể bị ẩn trong các thông báo thành công được kích hoạt quá nhanh hoặc không theo thứ tự. Do đó, phát hiện bất thường dựa trên log thông qua các từ khoá và RegEx không thực sự hiệu quả và cần đến mô hình phát hiện bất thường dựa trên các chuỗi log.

Tuy nhiên, với sự phát triển của ứng dụng AI trong các bài toán thực tế, một số phương pháp sử dụng các thuật toán để phục vụ mục đích phát hiện các bất thường trên hệ thống. Các sự kiện nhật ký này sẽ được ánh xạ tới một không gian vectơ, từ đó các sự kiện nhật ký liên tiếp được chuyển đổi thành vectơ đầu vào cho thuật toán ML. Tuy nhiên, các phương pháp này thường không ổn định trên các tập dữ liệu khác

nhau. Bởi vì các sự kiện nhật ký này được tạo bởi các nhà phát triển nên mỗi dòng nhật ký đều có ý nghĩa ngữ nghĩa riêng. Việc ánh xạ như trên gây mất thông tin quan trọng.

Gần đây, các phương pháp học sâu sử dụng các mô hình nổi tiếng như CNN, LSTM, máy biến áp đã được áp dụng để phát hiện các điểm bất thường trong thiết bị. Các mô hình này đã đạt được kết quả tốt trên các tập dữ liệu khác nhau. Các phương pháp này lấy các sự kiện nhật ký tuần tự làm đầu vào để xác định các điểm bất thường bằng cách phát hiện các hành vi vi phạm các mẫu tuần tự. Tuy nhiên, các mô hình này có nhược điểm là chưa khai thác được thông tin về mối liên kết giữa các sự kiện log, điều này rất quan trọng. Ngoài ra, những mô hình này không thể bao quát hết tất cả các sự kiện trong một khoảng thời gian dài. Nếu các cuộc tấn công vào thiết bị xảy ra với tần suất thấp, thì các hoạt động tấn công sẽ cách xa nhau, khiến các phương pháp này không thể nhận ra chúng.

Hiện nay có những nghiên cứu sử dụng đồ thị như LogGD [6] cũng cho kết quả rất tốt. Tuy nhiên, việc tính toán chuyển đổi từ chuỗi log sang biểu đồ vẫn còn hạn chế khi bỏ qua chuỗi sự kiện log. Để giải quyết vấn đề trên, học viên đề xuất phương pháp phát hiện bất thường dựa trên đồ thị. Phương pháp này sẽ chuyển đổi dữ liệu từ dạng chuỗi sang dạng biểu đồ. Sử dụng thông tin về tính năng nút và thông tin thu được từ biểu đồ để phục vụ cho việc phân loại các điểm bất thường trong hệ thống. Kết quả thử nghiệm trên các bộ dữ liệu phổ biến hiện nay cho thấy mô hình đạt kết quả tốt trong việc phát hiện các bất thường.

1.4 Kết chương

Chương 1 trình bày tổng quan về phát hiện bất thường trong hệ thống thông tin, cấu trúc và đặc điểm dữ liệu log hệ thống. Từ đó, nội dung chương chỉ ra những vấn đề còn tồn tại, thách thức và đề xuất phương án giải quyết các vấn đề này.

Chương 2 sẽ trình bày tổng quan về học máy, học sâu và một số phương pháp phát hiện bất thường hiện nay đồng thời mô tả chi tiết phương án giải quyết các vấn đề này nhằm đưa ra mô hình đề xuất phù hợp.

CHƯƠNG 2: CÁC PHƯƠNG PHÁP PHÁT HIỆN BẤT THƯỜNG

2.1 Tổng quan về học máy và học sâu

2.1.1 Tổng quan về học máy

Định nghĩa về học máy: Arthur Samuel, một người Mỹ đi tiên phong trong lĩnh vực trò chơi máy tính và trí tuệ nhân tạo, đã đặt ra thuật ngữ "Machine Learning" vào năm 1959 khi còn ở IBM. Ông định nghĩa học máy là "lĩnh vực nghiên cứu cung cấp cho máy tính khả năng học hỏi mà không cần lập trình rõ ràng". Tuy nhiên, trên thực tế không có định nghĩa được chấp nhận rộng rãi cho học máy. Các tác giả khác nhau định nghĩa thuật ngữ khác nhau.

Học máy là khả năng của chương trình máy tính sử dụng kinh nghiệm, quan sát, hoặc dữ liệu trong quá khứ để cải thiện công việc của mình trong tương lai thay vì chỉ thực hiện theo đúng các quy tắc đã được lập trình sẵn. Chẳng hạn, máy tính có thể học cách dự đoán dựa trên các ví dụ, hay có thể học cách tạo ra các hành vi phù hợp dựa trên quan sát trong quá khứ.

Học máy là một nhánh nghiên cứu rất quan trọng của trí tuệ nhân tạo với khá nhiều ứng dụng thành công trong thực tế. Hiện nay, học máy là một trong những lĩnh vực phát triển mạnh nhất của trí tuệ nhân tạo. Mục tiêu của học máy nói chung là hiểu cấu trúc của dữ liệu và điều chỉnh dữ liệu đó thành các mô hình mà con người có thể hiểu và sử dụng được.

Sử dụng những dạng kinh nghiệm và dạng biểu diễn khác nhau dẫn tới những dạng học máy khác nhau. Có bốn dạng học máy chính như sau:

Học có giám sát (supervised learning): Là dạng học máy trong đó cho trước tập dữ liệu huấn luyện dưới dạng các ví dụ cùng với giá trị đầu ra hay giá trị đích. Dựa trên dữ liệu huấn luyện, thuật toán học cần xây dựng mô hình hay hàm đích để dự đoán giá trị đầu ra (giá trị đích) cho các trường hợp mới.

- Nếu giá trị đầu ra là rời rạc thì học có giám sát được gọi là phân loại hay phân lớp (classification).

- Nếu đầu ra nhận giá trị liên tục, tức đầu ra là số thực, thì học có giám sát được gọi là hồi quy (regression). Trong phần tiếp theo, ta sẽ xem xét chi tiết hơn về học có giám sát.

Học không giám sát (Unsupervised learning): Mô hình được huấn luyện trên một tập dữ liệu không có nhãn. Mục tiêu là tìm kiếm cấu trúc ẩn trong dữ liệu, chẳng hạn như phân cụm hay giảm chiều dữ liệu. Học không giám sát sẽ không đưa ra kết quả cụ thể cho câu trả lời có hay không, hoặc một nhãn cụ thể nào đó. Các bài toán thường dùng cho học không giám sát là:

- Clustering (phân cụm): bài toán phân loại với tập dữ liệu X, tìm ra điểm tương đồng/ sự liên quan dữ liệu để phân nhóm. Ví dụ như, việc phân loại gen thành các nhóm về màu da, màu mắt, màu tóc… giống nhau
- Non-clustering (không phân cụm): bài toán tìm cấu trúc hoặc quy luật, dựa trên một dữ liệu có trước. Một ví dụ điển hình cho bài toán này đó là 'Cocktail Party Algorithm' - việc nhận dạng giọng nói/ âm thanh trong môi trường tạp âm.

Học không giám sát tương tự như cách con người học, trong đó phương pháp này được sử dụng để nhận biết rằng các đối tượng hoặc sự kiện cụ thể thuộc cùng một nhóm, thường bằng cách quan sát sự tương đồng giữa chúng. Một số hệ thống mà bạn có thể gặp trên internet sử dụng loại học này để tự động hóa chiến lược tiếp thị trên thị trường

Học nửa giám sát (Semi supervised learning): học nửa giám sát sẽ sử dụng những dữ liệu chỉ có một phần được gắn nhãn, và phần còn lại chưa được gắn nhãn. Phương thức này sẽ kết hợp học có giám sát và học không giám sát để đưa ra kết quả dự đoán cuối cùng.

Học tăng cường (reinforcement learning): Trong học tăng cường, hệ thống không nhận được kinh nghiệm dưới dạng đầu vào/đầu ra cho mỗi trạng thái hoặc hành động một cách trực tiếp. Thay vào đó, nó nhận được một giá trị khuyến khích

(reward) là kết quả của một chuỗi hành động cụ thể. Thuật toán trong học tăng cường nhằm mục đích học cách hành động để cực đại hóa giá trị khuyến khích.

Ví dụ, trong một phòng lab với các chướng ngại vật, robot nhận dạng trạng thái của mình dựa trên tọa độ và thông tin từ cảm biến. Hành động của robot bao gồm di chuyển, quẹo trái, quẹo phải và dừng lại. Robot nhận được giá trị khuyến khích dựa trên khoảng cách đến điểm đích. Trong quá trình học, robot thực hiện các hành động, nhận được phản hồi, và điều chỉnh chiến lược để tối ưu hóa việc đạt được mục tiêu. Thông qua nhiều thử nghiệm, robot ngày càng cải thiện khả năng tự điều hướng và di chuyển một cách hiệu quả trong môi trường không gian, mặc dù không biết trước được bản đồ chi tiết của nó.

2.1.2 Tổng quan về học sâu

Học sâu là một nhánh của học máy hoàn toàn dựa trên mạng nơ ron nhân tạo, vì mạng nơ ron sẽ bắt chước bộ não con người nên học sâu cũng là một loại bắt chước bộ não con người. Nhiều mô hình học sâu được áp dụng trong các lĩnh vực như: Thị giác máy tính, xử lý ngôn ngữ tự nhiên, phát hiện bất thường...học sâu mô hình hóa các mối quan hệ và khái niệm phức tạp bằng cách sử dụng nhiều cấp độ biểu diễn.

Mô hình học sâu thường được xây dựng dựa trên các kiến trúc mạng nơ-ron sâu, mô phỏng cách mà não người xử lý thông tin. Các mô hình này có khả năng học từ dữ liệu thông qua nhiều lớp nơ-ron, tăng cường khả năng biểu diễn và hiểu các đặc trưng phức tạp của dữ liệu.

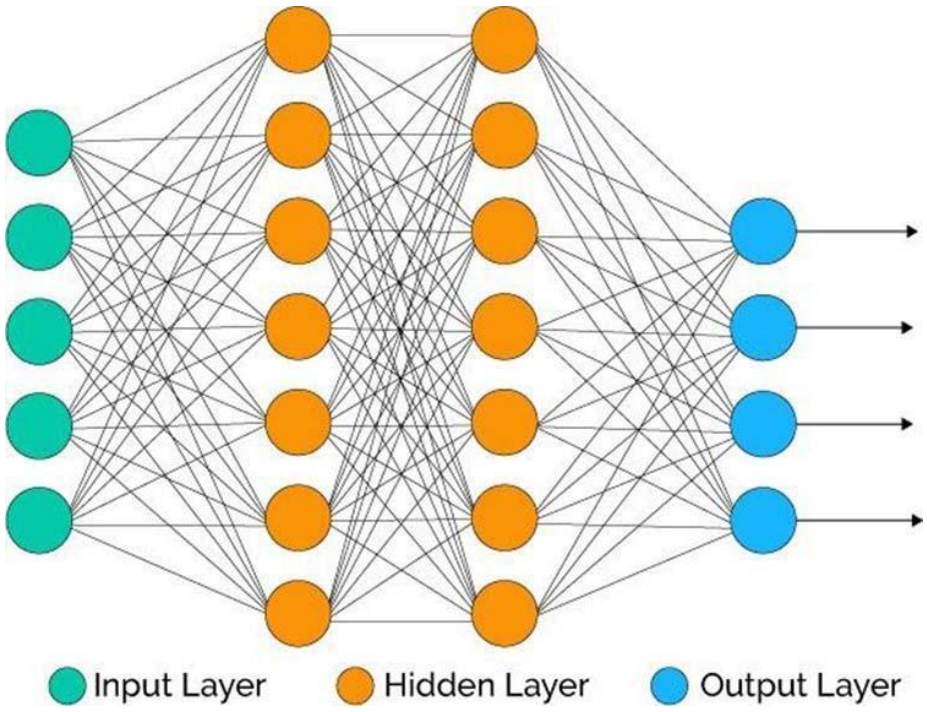
Ứng dụng của học sâu rộng rãi, từ nhận diện giọng nói, nhận diện hình ảnh, dịch ngôn ngữ tự nhiên đến tự động lái xe. Học sâu đã đưa ra những tiến bộ lớn trong nhiều lĩnh vực, mở ra những cánh cửa mới cho sự đổi mới và cải tiến công nghệ. Nó không chỉ là một công cụ mạnh mẽ cho các nhà nghiên cứu, mà còn là nguồn động viên lớn đối với những người mong muốn khám phá và ứng dụng sức mạnh của máy học trong thế giới thực.

Mạng nơ-ron nhân tạo (Artificial Neural Network - ANN) là một hệ thống tính toán có cấu trúc tương tự như mạng nơ-ron trong não người. Được thiết kế để mô phỏng cách nơ-ron làm việc, ANN là một phần quan trọng của lĩnh vực trí tuệ nhân tạo (AI).

Một ANN bao gồm các "nơ-ron" được tổ chức thành các lớp: lớp đầu vào, lớp ẩn (nếu có), và lớp đầu ra. Mỗi nơ-ron trong lớp được kết nối với tất cả các nơ-ron trong lớp liền kề bằng các trọng số. Các trọng số này được điều chỉnh trong quá trình huấn luyện để mô hình có thể học từ dữ liệu.

Quá trình học của ANN thường dựa trên giả định "học giám sát", nơi mô hình được đào tạo bằng cách so sánh đầu ra dự đoán với đầu ra mong muốn. Các thuật toán như lan truyền ngược (backpropagation) thường được sử dụng để điều chỉnh trọng số và cải thiện hiệu suất của mạng.

Kiến trúc chung của một mạng neuron nhân tạo gồm 3 thành phần đó là: Input Layer, Hidden Layer và Output Layer:



Hình 2.1: Kiến trúc mạng neuron

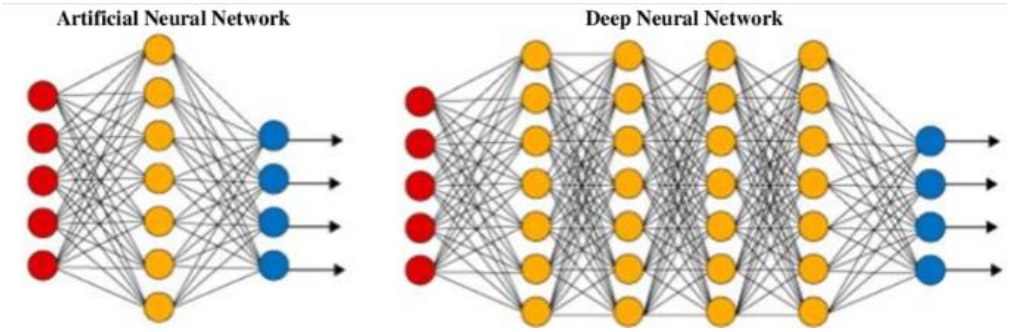
Trong đó, lớp ẩn (Hidden Layer) gồm các nơ ron nhận dữ liệu input từ các nơ ron ở lớp (Layer) trước đó và chuyển đổi các input này cho các lớp xử lý tiếp theo. Trong một ANN có thể có nhiều lớp ẩn.

Trong đó các Processing Elements (PE) của ANN gọi là nơ ron, mỗi nơ-ron nhận các dữ liệu vào (Inputs) xử lý chúng và cho ra một kết quả (Output) duy nhất. Kết quả xử lý của một nơ-ron có thể làm Input cho các nơ ron khác.

2.1.3 Một số phương pháp học sâu

- **Mạng nơ ron sâu (Deep Neural Network-DNN)**

Mạng nơ ron sâu (Deep Neural Network - DNN) là một dạng cụ thể của lĩnh vực học sâu. Mạng nơ ron sâu là một mạng nơ ron nhân tạo nhưng có kiến trúc phức tạp và "sâu" hơn nhiều so với kiến trúc của mạng nơ ron truyền thống. Mạng DNN tận dụng thành phần của mạng nơ ron nhân tạo ANN. Nghĩa là nó có số nút trong mỗi lớp và số lớp ẩn lớn hơn rất nhiều và cách thức hoạt động của nó phức tạp hơn so với kiến trúc mạng nơ ron truyền thống.



Hình 2.2: Mô hình mạng DNN

DNN cho phép mô hình thực hiện với độ chính xác cao hơn. Chúng cho phép mô hình thao tác với tập dữ liệu và trả về một kết quả với độ chính xác được cải thiện so với mạng nơ ron truyền thống.

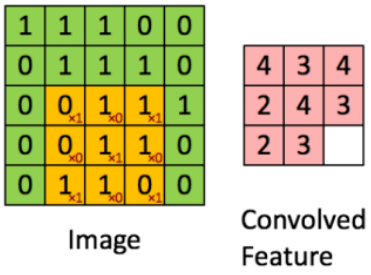
- **Mạng nơ ron tích chập (Convolutional Neural Network)**

CNN là từ viết tắt của cụm Convolutional Neural Network hay là mạng nơ ron tích chập. Đây là mô hình vô cùng tiên tiến được áp dụng nhiều trong lĩnh vực học sâu Deep learning. Mạng CNN cho phép người dùng xây dựng những hệ thống phân

loại và dự đoán với độ chính xác cực cao. Hiện nay, mạng CNN được ứng dụng nhiều hơn trong xử lý ảnh, cụ thể là nhận diện đối tượng trong ảnh.

Convolution (tích chập) Tích chập được sử dụng đầu tiên trong xử lý tín hiệu số (Signal processing). Nhờ vào nguyên lý biến đổi thông tin, các nhà khoa học đã áp dụng kĩ thuật này vào xử lý ảnh và video số.

Để dễ hình dung, ta có thể xem tích chập như một cửa sổ trượt (sliding window) áp đặt lên một ma trận. Cơ chế của tích chập qua hình minh họa bên dưới.



Hình 2.3: Minh họa cơ chế tích chập

Ma trận bên trái là một bức ảnh đen trắng. Mỗi giá trị của ma trận tương đương với một điểm ảnh (pixel), 0 là màu đen, 1 là màu trắng (nếu là ảnh grayscale thì giá trị biến thiên từ 0 đến 255).

Sliding window còn có tên gọi là kernel, filter hay feature detector. Ở đây, ta dùng một ma trận filter 3×3 nhân từng thành phần tương ứng (element-wise) với ma trận ảnh bên trái. Giá trị đầu ra do tích của các thành phần này cộng lại. Kết quả của tích chập là một ma trận (convoled feature) sinh ra từ việc trượt ma trận filter và thực hiện tích chập cùng lúc lên toàn bộ ma trận ảnh bên trái. Dưới đây là một vài ví dụ của phép toán tích chập.

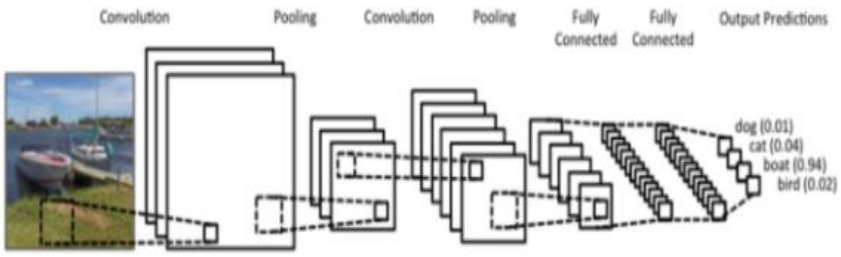
Ta có thể làm mờ bức ảnh ban đầu bằng cách lấy giá trị trung bình của các điểm ảnh xung quanh cho vị trí điểm ảnh trung tâm.

Bây giờ, Chúng ta đã biết thế nào là tích chập. Vậy CNNs là gì? CNNs chỉ đơn giản gồm một vài lớp của tích chập kết hợp với các hàm kích hoạt phi tuyến (nonlinear activation function) nhờ ReLU hay tanh để tạo ra thông tin trừu tượng hơn (abstract/higher-level) cho các layer tiếp theo.

Trong mô hình CNNs các layer liên kết được với nhau thông qua cơ chế convolution. Layer tiếp theo là kết quả convolution từ layer trước đó, nhờ vậy mà ta có được các kết nối cục bộ. Nghĩa là mỗi nơ-ron ở layer tiếp theo sinh ra từ filter áp đặt lên một vùng ảnh cục bộ của nơ-ron layer trước đó.

Mỗi layer nhờ vậy được áp đặt các filter khác nhau, thông thường có vài trăm đến vài nghìn filter như vậy. Một số layer khác như pooling/subsampling layer dùng để chắt lọc lại các thông tin hữu ích hơn (loại bỏ các thông tin nhiễu). Tuy nhiên, ta sẽ không đi sâu vào khái niệm của các layer này.

Trong suốt quá trình huấn luyện, CNNs sẽ tự động học được các thông số cho các filter. Ví dụ trong tác vụ phân lớp ảnh, CNNs sẽ cố gắng tìm ra thông số tối qua cho các filter tương ứng theo thứ tự raw pixel > edges > shapes > facial > high-level features. Layer cuối cùng được dùng để phân lớp ảnh.



Hình 2.4: Ví dụ về phân lớp ảnh

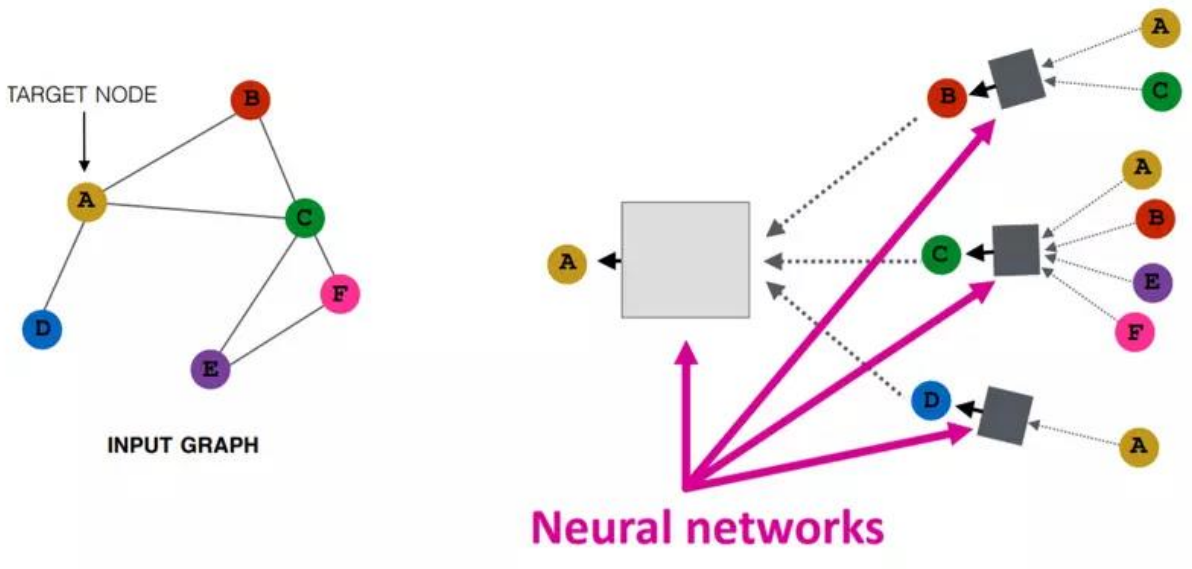
CNNs có tính bất biến và tính kết hợp cục bộ (Location Invariance and Compositionality). Với cùng một đối tượng, nếu đối tượng này được chiếu theo các góc độ khác nhau (translation, rotation, scaling) thì độ chính xác của thuật toán sẽ bị ảnh hưởng đáng kể. Pooling layer sẽ cho bạn tính bất biến đối với phép dịch chuyển (translation), phép quay (rotation) và phép co giãn (scaling).

Tính kết hợp cục bộ cho ta các cấp độ biểu diễn thông tin từ mức độ thấp đến mức độ cao và trừu tượng hơn thông qua convolution từ các filter. Đó là lý do tại sao CNNs cho ra mô hình với độ chính xác rất cao. Cũng giống như cách con người nhận biết các vật thể trong tự nhiên. Ta phân biệt được một con chó với một con

mèo nhờ vào các đặc trong từ mức độ thấp (có 4 chân, có đuôi) đến mức độ cao (dáng đi, hình thể, màu lông).

- **Mạng nơ ron đồ thị (Graph Neural Network - GNN)**

Mạng nơ-ron đồ thị (Graph Neural Network - GNN) là một loại mô hình học sâu được thiết kế đặc biệt để làm việc với dữ liệu đồ thị. GNN có khả năng mở rộng và áp dụng trên các đồ thị có cấu trúc phức tạp, như mạng xã hội, mạng lưới giao thông, hay bất kỳ hệ thống nào có mối quan hệ giữa các đối tượng. GNN hoạt động bằng cách truyền thông tin qua các đỉnh và cạnh trong đồ thị. Mô hình học thông qua việc cập nhật và kết hợp thông tin từ các hàng xóm của mỗi đỉnh, cho phép nắm bắt thông tin cấu trúc và tương tác giữa các đối tượng trong đồ thị. Một trong những đặc điểm đáng chú ý của GNN là khả năng tích hợp thông tin từ cả đặc trưng của các đỉnh và cấu trúc đồ thị. Điều này cho phép GNN học mô hình phức tạp và biểu diễn các mối quan hệ phức tạp giữa các đối tượng trong đồ thị. GNN đã chứng tỏ được hiệu quả trong nhiều nhiệm vụ, bao gồm phân loại đồ thị, phân loại nút, dự đoán liên kết và nhúng đồ thị. Các ứng dụng của GNN rất đa dạng, từ phân tích mạng xã hội, gợi ý người dùng, cho đến phát hiện và kiểm soát các hiện tượng trong các hệ thống phức tạp.



Hình 2.5: Mô hình mạng nơ ron đồ thị

Đồ thị đầu vào được đi qua một loạt mạng nơ-ron. Cấu trúc đồ thị đầu vào được chuyển đổi thành nhúng đồ thị, cho phép chúng ta duy trì thông tin về các nút, cạnh và ngữ cảnh toàn cục. Sau đó, vectơ đặc trưng của các nút A và C được thông qua lớp mạng neural. Nó tổng hợp những đặc trưng này và truyền chúng vào lớp tiếp theo.

Tuy GNN đã mang lại nhiều tiến bộ, nhưng vẫn còn nhiều thách thức trong việc khai thác toàn bộ tiềm năng của dữ liệu đồ thị và tăng tính khả chuyển của mô hình. Các nghiên cứu về GNN đang tiếp tục phát triển để nâng cao hiệu suất và ứng dụng của mô hình trong các lĩnh vực khác nhau.

Một số loại Graph Neural Networks

Có nhiều loại mạng nơ-ron đồ thị (Graph Neural Networks - GNN) được phát triển để làm việc với dữ liệu đồ thị. Dưới đây là một số loại GNN phổ biến:

- Graph Convolutional Networks (GCN): GCN là một dạng GNN đơn giản và phổ biến. Nó sử dụng cơ chế tích chập đồ thị để truyền thông tin qua các đỉnh và cạnh trong đồ thị. GCN kết hợp thông tin đặc trưng

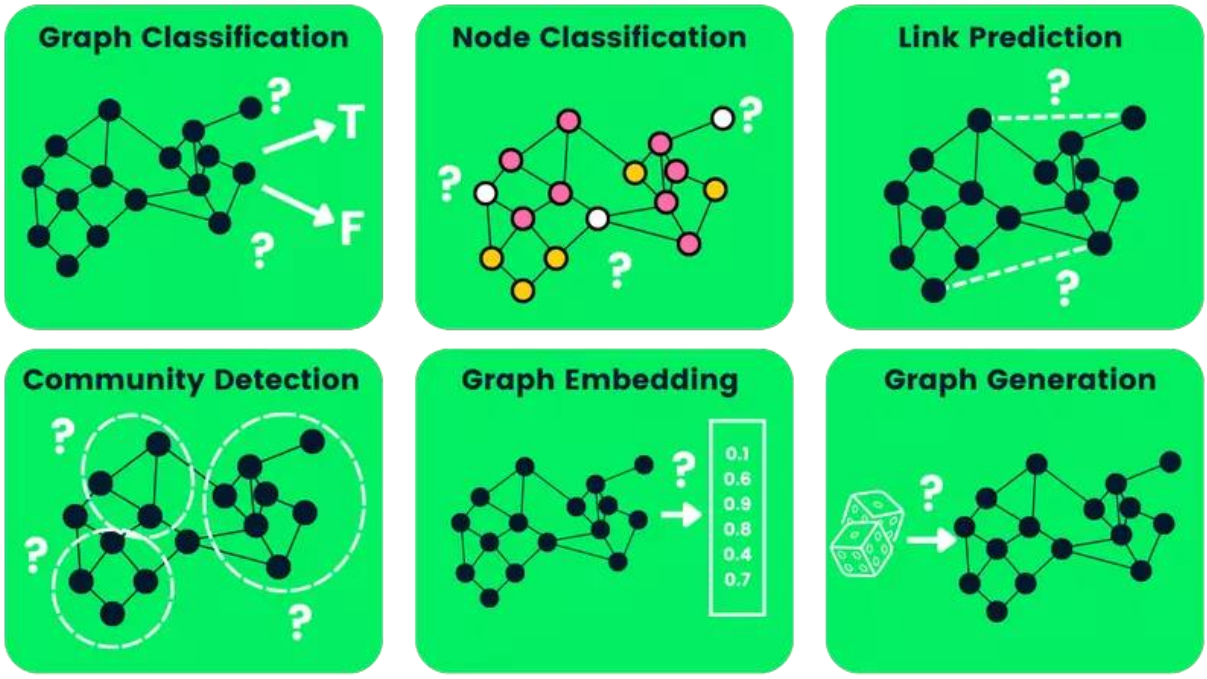
của các đỉnh và cấu trúc đồ thị để nhằm thực hiện phân loại hoặc dự đoán trên đồ thị.

- GraphSAGE: GraphSAGE (Graph Sample and Aggregated) sử dụng một quá trình lấy mẫu và tổng hợp thông tin từ hàng xóm của các đỉnh để cập nhật đặc trưng của mỗi đỉnh. Điều này giúp GraphSAGE học được biểu diễn đồ thị tổng thể và khả năng xử lý các đồ thị lớn.
- Graph Attention Networks (GAT): GAT sử dụng cơ chế chú ý (attention mechanism) để xác định mức độ quan trọng của các hàng xóm đối với mỗi đỉnh trong đồ thị. Bằng cách trọng số hóa thông tin từ các hàng xóm, GAT tập trung vào các đỉnh quan trọng và xử lý đồ thị một cách linh hoạt.
- Graph Autoencoders (GAE): GAE là một dạng GNN được sử dụng để học biểu diễn nhúng (embedding) của đồ thị. GAE cố gắng tái tạo đồ thị gốc từ biểu diễn nhúng, giúp học các đặc trưng ngầm của đồ thị và khám phá cấu trúc ẩn.
- Graph Recurrent Neural Networks (GRNN): GRNN sử dụng kiến trúc mạng nơ-ron hồi quy để xử lý dữ liệu đồ thị có thứ tự thời gian. GRNN có khả năng mô hình các quá trình động trên đồ thị như dự đoán chuỗi thời gian hoặc phân tích dữ liệu đồ thị theo thời gian.

Một số nhiệm vụ của Graph Neural Networks

- Graph Classification: chúng ta sử dụng phương pháp này để phân loại đồ thị vào các danh mục khác nhau. Ứng dụng của nó bao gồm phân tích mạng xã hội và phân loại văn bản.
- Node Classification: nhiệm vụ này sử dụng nhãn của các nút láng giềng để dự đoán nhãn của các nút bị thiếu trong đồ thị.

- Link Classification: dự đoán liên kết giữa một cặp nút trong đồ thị với ma trận kề không đầy đủ. Phương pháp này thường được sử dụng trong mạng xã hội.
- Community detection: chia các nút thành các cụm khác nhau dựa trên cấu trúc cạnh. Nó học từ trọng số cạnh, khoảng cách và đối tượng đồ thị tương tự.
- Graph Embedding: ánh xạ đồ thị thành các vectơ, bảo tồn thông tin quan trọng về các nút, cạnh và cấu trúc.
- Graph Generation: học từ phân phối đồ thị mẫu để tạo ra một cấu trúc đồ thị mới nhưng tương tự.



Hình 2.6: Một số nhiệm vụ của Graph Neural Networks

2.2 Các nghiên cứu về phát hiện bất thường

Trong những năm qua, đã có nhiều nghiên cứu đề xuất phát hiện bất thường từ dữ liệu log hệ thống. Các mô hình dựa trên học máy dựa vào thống kê log để phát hiện các bất thường của hệ thống. Đã có nhiều phương pháp học máy phổ biến áp dụng vào phát hiện bất thường dựa trên log. Tuy nhiên, các mô hình dựa trên học

máy này đều có nhiều hạn chế, các tính năng không linh hoạt, kém hiệu quả và khả năng thích ứng yếu.

Để giải quyết những vấn đề này, học sâu đã được áp dụng và đạt được những kết quả đầy hứa hẹn. Deeplog [13] đề xuất sử dụng mạng Long-Short Term Memory (LSTM) để học mẫu của các chuỗi log bình thường, từ đó xác định bất thường dựa trên các chuỗi log khác thường so với các mẫu bình thường đã học. LogRobust [18] và NeuralLog [14] trích xuất các vectơ ngữ nghĩa từ log để phát hiện bất thường. Meng và cộng sự đã đề xuất LogAnomaly[19] huấn luyện mạng LSTM để phát hiện bất thường sử dụng input là các vector đếm log. Họ cũng đề xuất template2vec, trong đó quan tâm đến các từ đồng nghĩa và từ trái nghĩa xuất hiện trong log template. Nhìn chung các phương pháp này lấy các sự kiện nhật ký tuần tự làm đầu vào để xác định các điểm bất thường bằng cách phát hiện các hành vi vi phạm các mẫu tuần tự. Tuy nhiên, các mô hình này có nhược điểm là chưa khai thác được thông tin về mối liên kết giữa các sự kiện log, điều này rất quan trọng. Ngoài ra, những mô hình này không thể bao quát hết tất cả các sự kiện trong một khoảng thời gian dài. Nếu các cuộc tấn công vào thiết bị xảy ra với tần suất thấp, thì các hoạt động tấn công sẽ cách xa nhau, khiến các phương pháp này không thể nhận ra chúng.

Hiện nay có những nghiên cứu sử dụng đồ thị như LogGD [6] cũng cho kết quả rất tốt. Tuy nhiên, việc tính toán chuyển đổi từ chuỗi log sang biểu đồ vẫn còn hạn chế khi bỏ qua chuỗi sự kiện log.

Bảng 2.1: Mô tả về ưu và nhược điểm của các phương pháp

	Deeplog	**LogRobust**	**NeuralLog**	**LogGD**
Ưu điểm	Hoạt động tốt với hệ thống có dữ liệu log ổn định	Hoạt động tốt với hệ thống có dữ liệu log ổn định và không ổn định	Hoạt động tốt với hệ thống có dữ liệu log ổn định và không ổn định	Hoạt động tốt với hệ thống có dữ liệu log ổn định và không ổn định
Nhược điểm	Hoạt động không tốt với hệ thống có dữ liệu log không ổn định - Cần dữ liệu lớn log để huấn luyện	Có độ chính xác chưa được cao so với phương pháp khác vì bỏ qua thông tin về mối liên kết giữa các sự kiện log	Có độ chính xác chưa được cao so với phương pháp khác vì bỏ qua thông tin về mối liên kết giữa các sự kiện log	Có độ chính xác chưa được cao so với phương pháp khác vì việc tính toán chuyển đổi từ chuỗi log sang biểu đồ vẫn còn hạn chế khi bỏ qua chuỗi sự kiện log

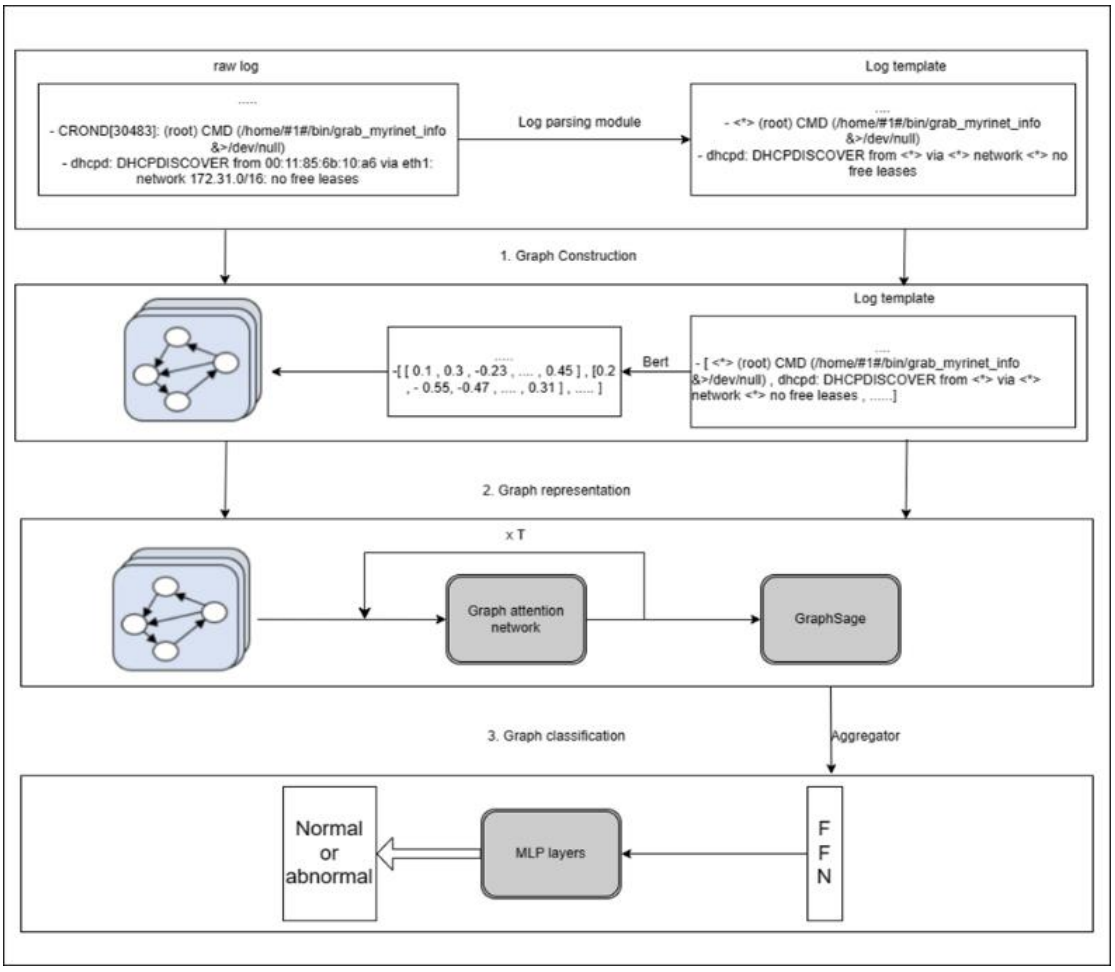
2.3 Đề xuất mô hình phát hiện bất thường

2.3.1 Giới thiệu mô hình

Để giải quyết vấn đề nêu trên, học viên đề xuất phương pháp phát hiện bất thường dựa trên đồ thị. Phương pháp này sẽ chuyển đổi dữ liệu từ dạng chuỗi sang dạng biểu đồ. Sử dụng thông tin về tính năng nút và thông tin thu được từ biểu đồ để phục vụ cho việc phân loại các điểm bất thường dựa trên mô hình mạng Graph Attention Networks (GAT) [7]. Phương pháp phát hiện bất thường dựa trên biểu đồ của học viên bao gồm ba bước chính:

- B1: Xây dựng biểu đồ: Là bước tính toán chuyển đổi dữ liệu thành dạng biểu đồ

- B2: Biểu diễn biểu đồ: Là bước tính toán thông tin của biểu đồ
- B3: Phân loại biểu đồ: Là bước phân loại nhãn của dữ liệu.



Hình 2.7: Các bước xử lý trong giải pháp

GraphSage (Graph Sample and Aggregated)

Việc biểu diễn các đặc điểm của đối tượng thành các đặc điểm của nút trong một biểu đồ lớn rất quan trọng để phân loại biểu đồ nhằm đạt được hiệu suất tốt nhất. Tuy nhiên, các phương pháp học chuyển đổi như DeepWalk không hiệu quả khi có các nút mới. Từ đó, GraphSAGE được đề xuất để giải quyết vấn đề này.

Cho $G = (V, E)$ là một đồ thị, trong đó V là tập các nút trong đồ thị, E là tập các cạnh trong đồ thị, X_v là tập đặc trưng nút của nút v, N_v là tập các nút lân cận của nút v , k là số lớp của mô hình, AGG_v là hàm tổng hợp ở lớp k. Ở trạng thái k, trọng số $h_v^1,\ h_v^{k+1}$Graphsage thực hiện các bước tính toán:

$$h_v^1 = X_v$$

$$h_v^{k+1} = AGG_v(h_v^k, \{h_{n_i}^k\} n_i \in N_v)$$

$$h_v^{k+1} = f(h_v^{k+1})(1)$$

Trong đó f là hàm kích hoạt. Trong nghiên cứu này đang sử dụng hàm sigmoid.

Mạng chú ý đồ thị(Graph Attention Networks - GAT)

Trong phần này, sẽ được trình bày tổng quan về mạng chú ý đồ thị (GAT). Trong mô hình GATS, mỗi nút có thể tham gia vào tính năng của vùng lân cận của chúng. Đặt $h = \{h_1, h_2, h_3, \dots h_n\}$ là tập hợp các đặc điểm nút của biểu đồ đầu vào, N là số nút trong biểu đồ, F là số lượng đặc điểm nút. Sau khi vượt qua từng lớp, sẽ thu được một tập hợp các tính năng nút mới có kích thước F', $h' = \{h_1', h_2', h_3', \dots, h_n'\}$ với $h_i' \in R^{F'}$. Hàm điểm e được xác định, $e : R^{F'} \times R^{F'} \to R$, e tính toán mức độ bất lực của đặc điểm của nút v_i và v_j.

$$e(h_i, h_j) = LeakyRelu(a^T || W\vec{h}_i || W\vec{h}_j) (2)$$

Trong đó cơ chế chú ý a là mạng nơ-ron tiếp liệu, $\overrightarrow{a} \in R^{2F'}$, T đại diện cho chuyển vị, W là ma trận trọng số, $W \in R^{F'xF}$, || là phép nối vectơ. Điểm chú ý giữa vi và vj được tính theo công thức:

$$\alpha_{ij} = \frac{exp(e(h_i,h_j))}{\sum_{k \in N_i} exp(e(h_i,h_j))}(3)$$

Trong đó N_i là tập hàng xóm của nút v_i. Sau đó, GAT tính giá trị trung bình có trọng số của các đặc điểm được chuyển đổi của các nút lân cận, kết quả của phép tính đóng vai trò là các đặc điểm đầu ra cuối cùng cho mỗi nút:

$$\overrightarrow{h'}_i = \sigma(\sum_{j \in N_i} \alpha_{ij} W\vec{h}_j) (4)$$

2.3.2 Các bước xử lý

Trong hệ thống máy tính, các mục nhật ký thường được phát triển và triển khai bởi các lập trình viên hệ thống. Các mục nhật ký này thường chứa cả thông tin ngữ nghĩa bằng ngôn ngữ tự nhiên và thông tin về các quy trình được ghi lại, chẳng hạn như ID tiến trình. Do đó, các công cụ phân tích dữ liệu nhật ký được thiết kế để trích xuất từng phần thông tin riêng lẻ từ nhật ký. Trong thử nghiệm này, sẽ sử dụng Drain [8], một công cụ phổ biến được nhiều mô hình sử dụng để phân tích dữ liệu nhật ký.

Quá trình xử lý của Drain gồm:

- Phân tích cú pháp: Chuyển đổi dữ liệu nhật ký sang định dạng có cấu trúc.
- Loại bỏ nhiễu: Loại bỏ các phần dữ liệu không liên quan hoặc không chính xác.
- Chuyển đổi kiểu: Chuyển đổi các kiểu dữ liệu sang định dạng phù hợp cho phân tích.
- Bổ sung trường: Thêm các trường mới vào dữ liệu nhật ký để hỗ trợ phân tích.

Sau khi thực hiện xong, dữ liệu có thể được sử dụng để phân loại. Dữ liệu cần được nhóm thành các cụm, chẳng hạn như phiên hoặc theo kích thước cửa sổ. Dựa trên phiên là quá trình nhóm các mục nhật ký của một quy trình thành một cụm và sau đó phân loại xem quy trình đó có bất thường hay không. Các thử nghiệm đã chỉ ra rằng phân loại bất thường dựa trên phiên đạt được kết quả tốt hơn so với việc sử dụng kích thước cửa sổ. Tuy nhiên, nhiều bộ dữ liệu đã xuất bản có thể khó thực hiện dựa trên phiên, vì vậy chúng ta cần sử dụng phân vùng dựa trên cửa sổ. Hiện nay có hai phương pháp phân vùng cửa sổ được sử dụng: cửa sổ cố định và cửa sổ trượt. Cửa sổ cố định là quá trình chia chuỗi nhật ký thành các cửa sổ có kích thước cố định, chẳng hạn như 20, 100, v.v. Cửa sổ trượt tương tự như cửa sổ cố định, nhưng các mục nhật ký được chia có thể chồng lên nhau giữa các cửa sổ, ví dụ: 20 nhật ký với một bước kích thước 1.

- **Xây dựng đồ thị**

Đầu vào của bài toán là chuỗi log sau khi phân tích và nhóm dữ liệu (S) theo từng phần hoặc cửa sổ trượt như đã trình bày trên. Với thông tin đầu vào này, sẽ xây dựng một biểu đồ có hướng để biểu diễn chúng. Cho $G = (V, E)$ là một đồ thị, trong đó V là tập hợp các nút trong đồ thị và E là tập hợp các cạnh có hướng. X_v là tập hợp các đặc điểm nút của đồ thị và F_v là tập hợp các đặc điểm cạnh của đồ thị. Mỗi nút trong biểu đồ G tương ứng với một sự kiện nhật ký trong chuỗi nhật ký. Tính năng nút sẽ được xây dựng dưới dạng thông tin ngữ nghĩa của các sự kiện nhật ký. Ở đây, việc thực hiện tính toán thông qua các mô hình NLP hiện có để phân tích các sự kiện nhật ký. Với xây dựng tập cạnh, mỗi e_{ij} tương ứng với sự xuất hiện của sự kiện V_i, theo sau là sự kiện V_j trong chuỗi nhật ký. Mỗi đặc điểm cạnh biểu thị thông tin cho cạnh có kích thước bằng độ dài của chuỗi nhật ký. Kích thước của tính năng cạnh bằng với độ dài của chuỗi nhật ký bởi vì sẽ đảm bảo rằng tất cả thông tin về thời gian xuất hiện của sự kiện nhật ký trong chuỗi nhật ký không bị mất trong quá trình xây dựng biểu đồ.

- **Biểu diễn đồ thị**

Biểu diễn đồ thị là một quá trình quan trọng trong các nhiệm vụ phân loại đồ thị và các nhiệm vụ liên quan khác. Để mô hình có thể phân biệt tốt giữa các sự kiện bình thường và bất thường, cần phải thiết kế cẩn thận các thuộc tính sẽ được sử dụng. Mỗi nút trong biểu đồ đại diện cho một sự kiện duy nhất trong chuỗi nhật ký thu được từ quá trình phân tích và xử lý. Mỗi sự kiện trong nhật ký đều do nhà phát triển xác định nên chúng đều có một ý nghĩa ngôn ngữ nhất định. Theo nhiều nghiên cứu trước đây, thông tin ngữ nghĩa từ các sự kiện nhật ký có thể rất hiệu quả cho nhiều mục đích khác nhau. Có nhiều mô hình NLP đạt hiệu suất cao trong trích xuất ngữ nghĩa như Word2vec [9], BERT [10], Fastext [11], v.v. Trong thử nghiệm này, sẽ sử dụng mô hình BERT(Bidirectional Encoder Representations from Transformers) là một mô hình ngôn ngữ mạnh mẽ được phát triển bởi Google AI,

có khả năng hiểu và biểu diễn văn bản ngôn ngữ tự nhiên một cách hiệu quả, với kích thước nhúng là 128 để biểu diễn log sự kiện. Các từ trong mỗi sự kiện nhật ký sẽ được trích xuất các đặc điểm thông qua BERT và các vectơ được mã hóa sẽ được tổng hợp để thu được vectơ biểu diễn thông tin 128 chiều cho mỗi sự kiện nhật ký.

Dưới đây là các bước chính trong hoạt động của mô hình:

- Sử dụng kiến trúc mạng nơ-ron Transformer để xây dựng mô hình BERT.
- Áp dụng hai nhiệm vụ học tập chính:
    - Dự đoán từ tiếp theo (Masked Language Modeling - MLM): Ẩn một số từ trong đoạn văn bản và yêu cầu mô hình dự đoán những từ bị ẩn dựa trên ngữ cảnh.
    - Dự đoán câu tiếp theo (Next Sentence Prediction - NSP): Cung cấp cho mô hình hai câu văn bản và yêu cầu mô hình xác định xem hai câu đó có liên quan với nhau hay không.
- Lặp lại quá trình học tập trên tập dữ liệu khổng lồ để tinh chỉnh mô hình BERT, giúp nó học cách hiểu các mối quan hệ ngữ nghĩa giữa các từ và câu.

Trong nghiên cứu đã phân tích ở trên, việc sử dụng chuỗi sự kiện theo trình tự nhật ký đã đạt được kết quả rất tốt. Trình tự của các sự kiện nhật ký này phản ánh mối tương quan giữa các mối quan hệ giữa các sự kiện nhật ký. Ví dụ: Neurallog [14] sử dụng mã hóa vị trí để biểu diễn chuỗi sự kiện nhật ký hoặc DeepLog [13] sử dụng các mô hình như LSTM để tính toán theo cách tuần tự. Để đảm bảo thông tin về thứ tự các sự kiện nhật ký không bị mất trong quá trình biểu diễn đồ thị, chúng ta biểu diễn thời gian xuất hiện của hai sự kiện nhật ký liên tiếp v_i và v_j vào e_{ij}.

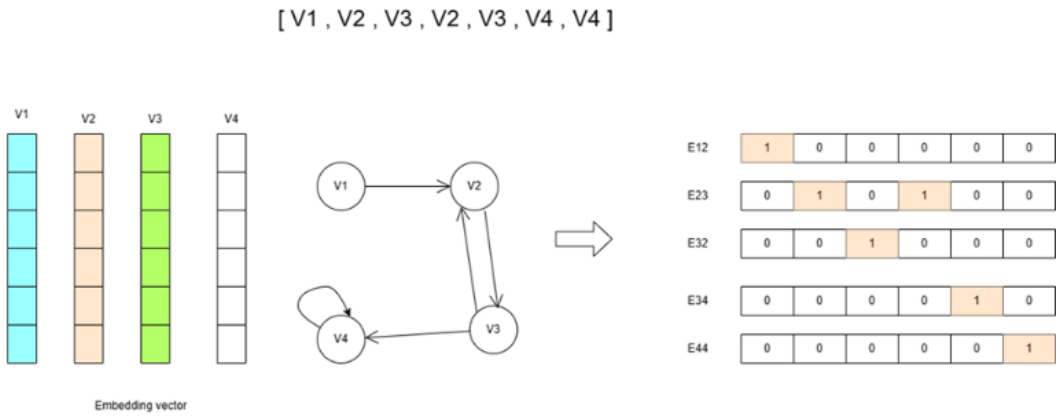
Đặt $e_{ij} = \{{e_{ij}}^{(0)}, {e_{ij}}^{(1)}, \dots {e_{ij}}^{(n)}\}$

$${e_{ij}}^{(t)} = \{1\ if\ S_t = (v_i, v_j)\ \ 0\ another \quad (5)$$

Trong đó $t\ in\ [1,\ldots,n]$ và S_t là một cặp sự kiện nhật ký liên tiếp trong S. Ví dụ như trên hình dưới. Sự kiện V2 và V3 xuất hiện liên tiếp hai lần trong dãy nhật ký nên ta có vectơ E_{23} = [0,1,0,1,0,0]. Cách biểu diễn này đảm bảo rằng biểu đồ vẫn duy trì tính tuần tự của dữ liệu gốc cũng như đảm bảo rằng việc biểu diễn dữ liệu ngắn gọn hơn khi các sự kiện trùng lặp chỉ cần được biểu diễn một lần. Trong thực tế, khi số lượng sự kiện chuỗi nhật ký lớn, các cuộc tấn công cũng có thể xảy ra trong thời gian dài, việc sử dụng kích thước cửa sổ nhỏ có thể dẫn đến khó phát hiện các sự kiện bất thường. Tuy nhiên, nếu nguồn là một chuỗi nhật ký dài thì việc biểu diễn nó theo cách này sẽ gây lãng phí tài nguyên khi có quá nhiều số 0 trong tính năng cạnh. Do đó, thay vì biểu thị số chiều của đối tượng địa lý cạnh bằng số chiều của chuỗi log, chúng ta sẽ sử dụng:

$$e_{ij}^{(t)'} = \sum e_{ij}^{(t\ mod\ m)} \quad (6)$$

Trong đó m là chu kỳ hoạt động của quá trình. Trong nghiên cứu này, sẽ sử dụng m = 60.



Hình 2.8: Xây dựng tính năng cạnh

- **Phân loại đồ thị**

Để có đầu vào cho phần phân loại đồ thị, phải cần chuyển đổi đầu ra từ phần biểu diễn đồ thị thành vectơ. Để có được vectơ này, cần sử dụng bộ tổng hợp để đọc. Có một số trình tổng hợp có thể được sử dụng, chẳng hạn như set2set, giá trị

trung bình, tổng, v.v. Kết quả sau khi tổng hợp thông tin sẽ được sử dụng làm đầu vào của khối MLP. Entropy chéo được sử dụng làm hàm mất mát cho nhiệm vụ.

$$h_g = Agg(h_n) \quad (7)$$

$$\hat{y} = MLP(h_g) \quad (8)$$

Cuối cùng, mô hình dựa trên đề xuất có thể dự đoán liệu đồ thị có bất thường hay không.

2.4 Kết chương

Chương 2 đã trình bày các khái niệm cơ bản về học máy, học sâu và một số phương pháp về học sâu, bên cạnh đó cũng đã nghiên cứu một số mô hình phát hiện bất thường trong hệ thống thông tin hiện nay. Từ đó, đưa ra mô hình đề xuất phù hợp để giải quyết được bài toán hiện nay.

Chương cuối sẽ trình bày về việc áp dụng mô hình đề xuất đã trình bày ở Chương 2 để cài đặt thử nghiệm và đánh giá kết quả.

CHƯƠNG 3: CÀI ĐẶT THỬ NGHIỆM VÀ ĐÁNH GIÁ KẾT QUẢ

3.1 Bộ dữ liệu thử nghiệm

Trong thử nghiệm này, học viên đã sử dụng 4 bộ dữ liệu phổ biến [15]: HDFS, BGL, Spirit và Thunderbird. Chúng đã được sử dụng trong nhiều nghiên cứu khác trước đây. Các bộ dữ liệu này được thu thập trong điều kiện thực tế ở các hệ thống lớn khác nhau và được chính quản trị viên hệ thống gắn nhãn. Để đảm bảo tính khách quan, việc sử dụng tập dữ liệu gốc sẽ được tải xuống từ trang web của nhà cung cấp. Chi tiết về các tập dữ liệu như sau:

- HDFS (Hệ thống tệp phân tán Hadoop) [16]:
    - Bao gồm 11.175.629 sự kiện được thu thập từ cụm Hadoop trong 10 ngày. Phân phối lớp khá mất cân bằng. Đây là tập dữ liệu tham khảo cho các sự cố hệ thống phát hiện xâm nhập.
    - Kích thước: 100 GB Loại sự kiện: 10, Định dạng: JSON
- BGL (Siêu máy tính BlueGene/L) [17]:
    - 4.748.092 sự kiện được ghi lại từ siêu máy tính BlueGene/L tại Phòng thí nghiệm quốc gia Lawrence Livermore. Bộ dữ liệu này thường được sử dụng trong các vấn đề phát hiện an ninh mạng và dự đoán lỗi hệ thống.
    - Kích thước: 1 GB Loại sự kiện: 5 loại Định dạng: CSV
- Spirit [17]:
    - Tập dữ liệu Spirit được thu thập từ siêu máy tính Spirit tại Phòng thí nghiệm quốc gia Sandia Hoa Kỳ. Trong nghiên cứu này, sẽ sử dụng 5 triệu dòng nhật ký của tập dữ liệu gốc để đào tạo và xác thực.
    - Kích thước: 10 GB Loại sự kiện: 20 loại Định dạng: JSON
- Thunderbird [17]:

- Bộ dữ liệu Thunderbird cũng được lấy từ siêu máy tính tại Phòng thí nghiệm quốc gia Sandia Hoa Kỳ. Tập dữ liệu rất lớn nhưng chỉ sử dụng 10 triệu dòng nhật ký để đào tạo và xác thực trong thử nghiệm này.
- Kích thước: 100 GB Loại sự kiện: 50 loại Định dạng: JSON

Bảng 3.1: Mô tả cơ sở dữ liệu

Bộ dữ liệu	**Nodes**	**Windows**
HDFS	48	section
BGL	1847	100 logs
		20 logs
Spirit	1229	100 logs
		20 logs
Thunderbird	4992	100 logs
		20 logs

3.2 Tiêu chuẩn đánh giá

Để đánh giá tính hiệu quả của mô hình được đề xuất, sẽ sử dụng các số liệu về F1, khả năng thu hồi và độ chính xác:

- False Negative (FN): Số trường hợp mà mô hình dự đoán là bình thường nhưng thực tế là bất thường.
- True Negative (TN): Số trường hợp mà mô hình dự đoán đúng là bình thường.
- False Positive (FP): Số trường hợp mà mô hình xác định sai, báo cáo là bất thường, trong khi chúng thực sự là bình thường.
- True Positive (TP): Số trường hợp mà mô hình phát hiện đúng và xác định là bất thường.

Chi tiết về các chỉ số như sau:

- Recall là thước đo để đánh giá hiệu suất của mô hình phân loại, tập trung vào khả năng mô hình tìm thấy tất cả các trường hợp tích cực:

$$Recal\ (rec.)\ =\ \frac{TP}{TP\ +\ FN}(9)$$

- Precision là thước đo để đánh giá hiệu suất của mô hình phân loại, tập trung vào khả năng của mô hình trong việc dự đoán chính xác các trường hợp tích cực

$$Precision\ (prec.)\ =\ \frac{TP}{TP\ +\ FP}(10)$$

- F1-score là thước đo để đánh giá hiệu suất của mô hình phân loại, kết hợp cả độ chính xác và khả năng thu hồi.

$$F1-score(prec.)\ =\ \frac{2}{\frac{1}{Recal}+\frac{1}{Precision}}(11)$$

3.3 Cài đặt, thử nghiệm

Các thử nghiệm được thực hiện trên máy tính với thông số cấu hình của mô trường được mô tả cụ thể trong bảng sau:

Bảng 3.2: Cấu hình môi trường thử nghiệm

Thông tin	Môi trường máy huấn luyện
Vi xử lý	CPU Intel(R) Xeon(R) E5-2650 v2 @ 2.60GHz
Dung lượng RAM	62GB
Dung lượng bộ nhớ	256GB
GPU	Quandro GTX 4000 8GB
Hệ điều hành	Ubuntu 22.04.3 LTS
Python	3.8.0
Pytorch	1.13.0
torchgeometric	2.4.0

Trong nghiên cứu của học viên, học viên đã thử nghiệm các tham số và đặt mạng chú ý đồ thị(Graph Attention Networks - GAT) trên biểu đồ với số lần lặp là 2 và kích thước của mạng chuyển tiếp nguồn cấp dữ liệu là 512 để tối ưu. Học viên sử

dụng trình tối ưu hóa Adam cho trạng thái đào tạo và không sử dụng tính năng giảm trọng số trong giai đoạn huấn luyện và tốc độ học được sử dụng là 0,001. Học viên đặt mini-batch là 32 và tỷ lệ dropout là 0,1. Trong nghiên cứu này, mô hình được huấn luyện trong 40 epochs. Học viên sử dụng entropy chéo để tính hàm mất mát.

Học viên đã tóm tắt kích thước của một số mô hình Sota hiện tại trong Bảng 3.3. Chúng ta có thể quan sát rằng kích thước của các mô hình như NeuralLog[14] hoặc LogGD[6] sử dụng một số lượng lớn tham số. Các mô hình như DeepLog[13] và LogRobust [18] sử dụng ít tham số hơn mô hình học viên đề xuất. Tuy nhiên, số lượng tham số ít dẫn đến kết quả phân loại với tập dữ liệu của hai mô hình này không cho kết quả tốt nhất. Mô hình học viên đề xuất có số lượng tham số nhỏ hơn NeuralLog và LogGD 20 lần. Điều này đảm bảo tính khả thi khi triển khai trên các thiết bị nhỏ.

Bảng 3.3: Thông số của mô hình

Mymodel	NeuralLog	LogRobust	DeepLog	LogGD
1156610	22067714	563602	181800	19371521

3.1 Phân tích và đánh giá kết quả

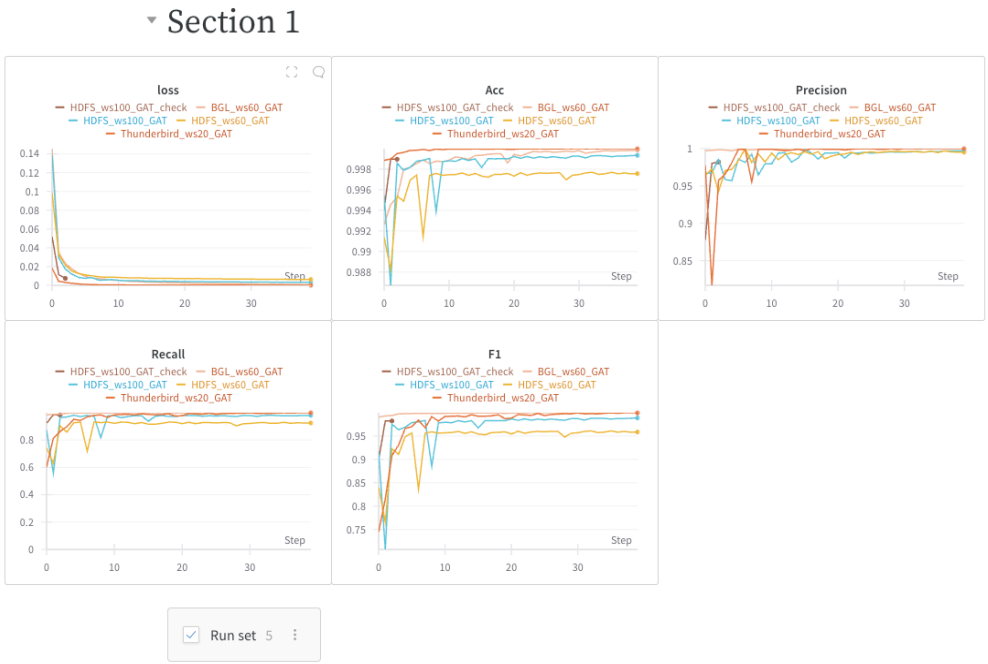
Kết quả trong Bảng 3.4 cho thấy mô hình đề xuất của học viên đạt được điểm F1 tốt nhất trên cả 4 bộ dữ liệu - HDFS, BGL, Spirit và Thunderbird. Cụ thể, mô hình của học viên đạt được điểm F1 là 0,9886, 0,9998, 0,995 và 0,992 trên các bộ dữ liệu này. Điều này chứng tỏ rằng mô hình của học viên hoạt động tốt trong việc phát hiện sự bất thường. So với LogGD, cũng sử dụng cách tiếp cận dựa trên biểu đồ, kết quả của học viên tốt hơn trong tất cả các thử nghiệm, điều này cho thấy rằng phương pháp biểu diễn biểu đồ mà học viên đề xuất đã trích xuất tất cả thông tin liên quan giữa các sự kiện nhật ký và chuỗi sự kiện xảy ra. Các mô hình chỉ sử dụng chuỗi sự kiện nhật ký như DeepLog, NeuralLog và LogRobust cũng đạt kết quả cao với các bộ dữ liệu như HDFS hoặc BGL. Tuy nhiên, với các bộ dữ liệu không chia thành các phiên như Spirit và Thunderbird thì kết quả không ổn định. Điều này có thể là do các

bộ dữ liệu này là sự kết hợp của các quy trình khác nhau trong hệ thống. Việc chỉ sử dụng trình tự khiến các mô hình thiếu sự kết nối giữa các tiến trình khác nhau, dẫn đến kết quả không cao bằng các mô hình sử dụng Graph-Based.

Bảng 3.4: Kết quả thực nghiệm

Model	**HDFS**			**BGL**			**Spirit**			**Thunderbird**		
	F1	*Rec.*	*Prec.*	*F1*	*Rec.*	*Prec.*	*F1*	*Rec.*	*Prec.*	*F1*	*Rec.*	*Prec.*
My model	0.9886	0.9803	0.997	0.9998	0.9996	1	0.995	0.992	0.999	0.992	0.986	0.998
NeuralLog	0.9827	1	0.96	0.9535	0.9586	0.9484	0.97	0.98	0.96	0.96	1	0.93
DeepLog	0.908	0.994	0.835	0.927	0.903	0.952	0.929	0.992	0.871	0.369	1	0.232
LogGD	0.9877	0.9982	0.9774	0.9719	0.9708	0.9731	0.979	0.989	0.969	0.9284	0.8889	0.9772
LogRobust	0.9819	1	0.9688	0.9402	0.9229	0.9596	0.9757	0.9957	0.9566	0.941	0.921	0.803

Nhìn chung, kết quả thử nghiệm thể hiện rõ ràng ưu điểm của mô hình đề xuất trong việc phát hiện sự bất thường từ nhật ký hệ thống. Với khả năng biểu diễn thông tin và trình tự kết nối, mô hình của học viên đạt kết quả tốt nhất về độ chính xác và điểm F1 được mô tả chi tiết dưới hình sau:



Hình 3.1: Biểu đồ dữ liệu chạy trên các tập dữ liệu

3.2 Kết chương

Chương cuối trình bày tập dữ liệu thử nghiệm, cài đặt và thử nghiệm từ đó tiến hành đánh giá hiệu quả mô hình đã được trình bày trong chương 2 và đề xuất phương pháp nghiên cứu, phát triển sau này.

KẾT LUẬN

Đề án nghiên cứu và đề xuất mô hình hình phát hiện bất thường trong các hệ thống thông tin thông qua dữ liệu log hệ thống dựa trên mô hình mạng chú ý đồ thị. Với mô hình đề xuất, các nghiên cứu đi kèm chứng minh tính hiệu quả của nó trong việc phát hiện các bất thường cho hệ thống thông tin. Học viên hy vọng rằng mô hình đề xuất có thể được áp dụng vào các ứng dụng thực tế trong tương lai. Dưới đây là một số kết quả đạt được và định hướng tương lai của học viên:

Kết quả đạt được:

- Giới thiệu và trình bày tổng quan về dữ liệu log hệ thống, phát hiện bất thường trong hệ thống thông tin, đồng thời tìm hiểu các vấn đề liên quan tới phát hiện bất thường dựa trên phân tích log.
- Giới thiệu và trình bày một số các mô hình sử dụng học máy và học sâu phát hiện bất thường dựa trên phân tích dữ liệu log hệ thống hiện nay.
- Nghiên cứu và trình bày chi tiết về mô hình phát hiện bất thường trong các hệ thống thông tin thông qua dữ liệu log hệ thống dựa trên mô hình mạng chú ý đồ thị (Graph Attention Networks – GAT).
- Tiến hành thực hiện và so sánh đánh giá kết với các một số phương pháp tiên tiến hiện có.

Hướng phát triển trong tương lai:

Với mô hình đề xuất kích thước không quá lớn và các nghiên cứu đi kèm chứng minh tính hiệu quả của nó trong việc phát hiện các bất thường cho các hệ thống thông tin. Học viên định hướng sẽ áp dụng và phát triển mô hình cho việc phát hiện bất thường dựa trên dữ liệu log hệ thống trong các thiết bị IoT vừa và nhỏ.

DANH MỤC TÀI LIỆU THAM KHẢO

[1] ‘Huy-Trung, N., & Viet Quoc, N. (2023, May). Anomaly Detection in Internet of Things Based on Logs Using Machine Learning and Deep Learning Techniques. In Proceedings of the 2023 4th International Conference on Computing, Networks and Internet of Things (pp. 969-974).’

[2] ‘Le, V. H., & Zhang, H. (2022, May). Log-based anomaly detection with deep learning: How far are we?. In Proceedings of the 44th international conference on software engineering (pp. 1356-1367).’

[3] ‘Raut, P., Mishra, A., Rao, S., Kawoor, S., Shelke, S., Deore, M., & Kumar, V. (2022, February). Review on Log-Based Anomaly Detection Techniques. In Proceedings of Second International Conference on Sustainable Expert Systems: ICSES 2021 (pp. 893-906). Singapore: Springer Nature Singapore.’

[4] ‘Forcher, B., Agne, S., Dengel, A., Gillmann, M., & Roth-Berghofer, T. (2011, September). Semantic logging: Towards explanation-aware das. In 2011 International Conference on Document Analysis and Recognition (pp. 1140-1144). IEEE.’

[5] ‘Chen, Z., Liu, J., Gu, W., Su, Y., & Lyu, M. R. (2021). Experience report: Deep learning-based system log analysis for anomaly detection. arXiv preprint arXiv:2107.05908.’

[6] ‘Xie, Y., Zhang, H., & Babar, M. A. (2022, December). LogGD: Detecting Anomalies from System Logs with Graph Neural Networks. In 2022 IEEE 22nd International Conference on Software Quality, Reliability and Security (QRS) (pp. 299-310). IEEE.’

[7] ‘Brody, S., Alon, U., & Yahav, E. (2021, October). How Attentive are Graph Attention Networks?. In International Conference on Learning Representations.’

[8] ‘He, P., Zhu, J., Zheng, Z., & Lyu, M. R. (2017, June). Drain: An online log parsing approach with fixed depth tree. In 2017 IEEE international conference on web services (ICWS) (pp. 33-40). IEEE.’

[9] ‘Ayyadevara, V. K., & Ayyadevara, V. K. (2018). Word2vec. Pro Machine Learning Algorithms: A Hands-On Approach to Implementing Algorithms in Python and R, 167-178.’

[10] ‘Chen, S., & Liao, H. (2022). Bert-log: Anomaly detection for system logs based on pre-trained language model. Applied Artificial Intelligence, 36(1), 2145642.’

[11] ‘Gniewkowski, M., Maciejewski, H., Surmacz, T., & Walentynowicz, W. (2023, March). Sec2vec: Anomaly Detection in HTTP Traffic and Malicious URLs. In Proceedings of the 38th ACM/SIGAPP Symposium on Applied Computing (pp. 1154-1162).’

[12] ‘Chen, Z., Gao, Q., & Moss, L. S. (2021). Neurallog: Natural language inference with joint neural and logical reasoning. arXiv preprint arXiv:2105.14167.’

[13] ‘Du, M., Li, F., Zheng, G., & Srikumar, V. (2017, October). Deeplog: Anomaly detection and diagnosis from system logs through deep learning. In Proceedings of the 2017 ACM SIGSAC conference on computer and communications security (pp. 1285-1298).’

[14] ‘Yu, Y., Si, X., Hu, C., & Zhang, J. (2019). A review of recurrent neural networks: LSTM cells and network architectures. Neural computation, 31(7), 1235-1270.’

[15] ‘Wu, X., Li, H., & Khomh, F. (2023). On the effectiveness of log representation for log-based anomaly detection. Empirical Software Engineering, 28(6), 137.’

[16] ‘Xu, W., Huang, L., Fox, A., Patterson, D., & Jordan, M. I. (2009, October). Detecting large-scale system problems by mining console logs. In Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles (pp. 117-132).’

[17] ‘Oliner, A., & Stearley, J. (2007, June). What supercomputers say: A study of five system logs. In 37th annual IEEE/IFIP international conference on dependable systems and networks (DSN’07) (pp. 575-584). IEEE.’

[18] ‘Zhang, X., Xu, Y., Lin, Q., Qiao, B., Zhang, H., Dang, Y., ... & Zhang, D. (2019, August). Robust log-based anomaly detection on unstable log data. In Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (pp. 807-817).’

[19] Weibin, M., Ying, L., Yichen, Z., Shenglin, Z., Dan, P., Yuqing, L., Yihao, C., Ruizhi, Z., Shimin, T., Pei, S. (2019), “Loganomaly: Unsupervised detection of sequential and quantitative anomalies in unstructured logs”, *IJCAI*, 19, pp. 4739–4745.

[20] "Introduction to deep neural networks" 7 2023. [Online]. Available: https://www.datacamp.com/tutorial/introduction-to-deep-neural-networks. [Accessed 12 1 2024].

[21] Andrew M Dai and Quoc V Le. 2015. Semi-supervised sequence learning. In Proc. Neural Information Processing Systems Conference (NIPS). 3079–3087.

[22] Min Du, Feifei Li, Guineng Zheng, and Vivek Srikumar. 2017. Deeplog: Anomaly detection and diagnosis from system logs through deep learning. In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security.ACM, 1285–1298.