HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

Dương Đỗ Nhuận

TÓM TẮT
LUẬN VĂN THẠC SỸ KỸ THUẬT
(Theo định hướng ứng dụng)

HÀ NỘI – 2020

Luận văn được hoàn thành tại:

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

Người hướng dẫn khoa học: TS. Hoàng Xuân Dậu

Phản biện 1: PGS. TS. Đỗ Trung Tuấn

Phản biện 2: TS. Phùng Văn Ổn

Luận văn đã được bảo vệ trước Hội đồng chấm luận văn thạc sĩ tại Học viện Công nghệ Bưu chính Viễn thông

Vào lúc: 9giờ 15 ngày 09 tháng 1 năm 2021

Có thể tìm hiểu luận văn tại:

- Thư viện của Học viện Công nghệ Bưu chính Viễn thông.

LỜI CẢM ƠN

Em xin gửi lời cảm ơn và tri ân tới các thầy cô giáo, cán bộ của Học viện Công nghệ Bưu chính Viễn thông đã giúp đỡ, tạo điều kiện tốt cho em trong quá trình học tập và nghiên cứu chương trình Thạc sĩ.

Em xin gửi lời cảm ơn sâu sắc tới TS. Hoàng Xuân Dậu đã tận tình hướng dẫn, giúp đỡ và động viên em để hoàn thành tốt nhất Luận văn "**PHÁT HIỆN XÂM NHẬP MẠNG SỬ DỤNG HỌC MÁY** ".

Do vốn kiến thức lý luận và kinh nghiệm thực tiễn còn ít nên luận văn không tránh khỏi những thiếu sót nhất định. Em xin trân trọng tiếp thu các ý kiến của các thầy, cô để luận văn được hoàn thiện hơn và có những hướng phát triển sau này.

Trân trọng cảm ơn.

Tác giả

Dương Đỗ Nhuận

Danh mục các ký hiệu, các chữ viết tắt

Từ viết tắt/ Ký hiệu	Ý nghĩa/ Từ đầy đủ
ANN	Mạng Nơron nhân tạo - Artificial Neural Network
GD	Một thuật toán tối ưu lặp (iterative optimization algorithm) được sử dụng trong các bài toán Machine Learning và Deep Learning Gradient Descent
SAE	Phương pháp học đặc trưng đầu bằng cách xếp chồng các Autoencoder lên nhau - Stacked Autoencoder
DAE	Một phương pháp học đặc trưng đầu vào phát triển từ Autoencoder - Denoise Autoencoder
SDAE	Một phương pháp học đặc trưng đầu vào bằng cách xếp chồng các Denoise Autoencoder lên nhau - Stacked Denoise Autoencoder
IDS	Hệ thống phát hiện xâm nhập - Intrusion Detection System
SVM	Thuật toán học có giám sát liên quan đến nhau để phân loại và phân tích hồi quy - Support Vector Machine
RF	Thuật toán học có giám sát tạo ra cây quyết định trên các mẫu dữ liệu được chọn ngẫu nhiên, được dự đoán từ mỗi cây và chọn giải pháp tốt nhất bằng cách bỏ phiếu - Random Forest
KNN	Một kĩ thuật học có giám sát dùng để phân loại quan sát mới bằng cách tìm điểm tương đồng giữa quan sát mới này với dữ liệu sẵn có - K-nearest Neighbors
SGD	Thuật toán được dùng để tối ưu hàm mục tiêu $J(\theta)$, với tham số tương ứng $\theta \in R^d$, bằng cách dần dần cập nhật tham số θ theo hướng ngược lại với với gradient của tham số tại hàm mục tiêu $\nabla_\theta J(\theta)$ - Stochastic Gradient Descent
AUC	Một phương pháp tính toán hiệu suất của một mô hình phân loại theo các ngưỡng phân loại khác nhau - Area Under The Curve
TP	True Positive
FN	False Negative
FP	False Positive
TN	True Negative
NIDS	Hệ thống phát hiện ở mức mạng – Network Intrusion Detection System
HIDS	Hệ thống phát hiện xâm nhập ở mức máy trạm chủ – Host Intrusion Detection System
R2L	Hình thức tấn công từ xa - Remote to Local
DoS	Hình thức tấn công từ chối dịch vụ - Denial of Service
U2R	Hình thức tấn công leo thang đặc quyền - User to Root
DT	Thuật toán cây phân cấp có cấu trúc, được dùng để phân lớp các đối tượng dựa vào dãy các luật - Decision Tree
NB	Thuật toán Naive Baves

Danh mục các bảng

Bảng 3.1 Các thuộc tính của tập dữ liệu Phishing Website Data
Bảng 3.2 Các kiểu tấn công trong tập dữ liệu NSL-KDD
Bảng 3.2 Các thuộc tính của tập dữ liệu NSL-KDD
Bảng 3.3 Bảng so sánh AUC giữa sử dụng SAE, SDAE và không sử dụng của Phishing Data Website
Bảng 3.5 Bảng so sánh AUC giữa sử dụng SAE, SDAE và không sử dụng của NSL-KDD

Danh mục các hình

Hình 2.1 Sơ đồ cấu trúc mạng Autoencoder
Hình 2.2 Sơ đồ cấu trúc mạng Stacked Autoencoder
Hình 2.3 Sơ đồ cấu trúc mạng Denoise Autoencoder
Hình 2.4 Sơ đồ cấu trúc mạng Stacked Denoise Autoencoder
Hình 2.5 Mô hình ứng dụng SAE và SDAE vào hệ thống IDS

MỤC LỤC

LỜI CẢM ƠN **2**
Danh mục các ký hiệu, các chữ viết tắt **2**
Danh mục các bảng **3**
Danh mục các hình **3**
CHƯƠNG I **5**
1.1 Khái quát về tấn công xâm nhập mạng **5**
1.2 Một số dạng tấn công xâm nhập điển hình vào hệ thống CNTT. **5**
1.2.1 Asymmetric Routing 5
1.2.2 Buffer Overflow Attacks (Tấn công tràn bộ đệm) 5
1.2.3 Common Gateway Interface Scripts 5
1.2.4 Protocol-Specific Attacks (Tấn công theo giao thức mạng) 5
1.2.5 Traffic Flooding (Tấn công tràn lưu lượng mạng) 5
1.2.6 Trojans 5
1.2.7 Worms (Sâu máy tính) 6
1.3 Các biện pháp phòng chống tấn công, xâm nhập mạng **6**
1.4 Khái quát về phát hiện xâm nhập mạng **6**
1.4.2 Phân loại **6**
CHƯƠNG II **8**
2.1.Khái quát về học máy và học sâu **8**
2.1.1 Khái quát về học máy 8
2.1.2 Khái quát về học sâu 8
2.2. Học sâu sử dụng Autoencoder và ứng dụng trong tiền xử lý dữ liệu **8**
2.2.1 Học sâu sử dụng Autoencoder 8
2.2.2 Phân loại Autoencoder 9
2.2.3. Ứng dụng Autoencoder trong tiền xử lý dữ liệu 11
2. 3. Xây dựng mô hình phát hiện xâm nhập dựa trên Autoencoder **11**
CHƯƠNG III **13**
3.1 Phương pháp cài đặt thử nghiệm **13**
3.2 Giới thiệu tập dữ liệu **13**
3.2.1 Phishing Website Data 13
3.2.2 NSL-KDD 13
3.3 Trích chọn đặc trưng sử dụng AE **13**
3.4 Huấn luyện và phát hiện **13**
3.5 Kết quả và nhận xét **13**
3.5.1 Kết quả của bộ dữ liệu Phishing Website Data 13
3.5.2 Kết quả của bộ dữ liệu NSL-KDD 14
KẾT LUẬN **16**

CHƯƠNG I
TỔNG QUAN VỀ PHÁT HIỆN XÂM NHẬP MẠNG

1.1 Khái quát về tấn công xâm nhập mạng

Tấn công xâm nhập mạng là mọi hành vi trên mạng máy tính không có được sự cho phép. Phát hiện tấn công xâm nhập mạng dựa trên người phòng thủ có sự hiểu biết rõ ràng về cách tấn công được thực hiện.

1.2 Một số dạng tấn công xâm nhập điển hình vào hệ thống CNTT.

1.2.1 Asymmetric Routing

Trong phương pháp này, kẻ tấn công cố gắng sử dụng nhiều hơn một đường dẫn (route) đến thiết bị mạng được nhắm mục tiêu. Ý tưởng là để cuộc tấn công tổng thể tránh bị phát hiện bằng cách để một phần đáng kể các gói tin vi phạm bỏ qua một số phân đoạn mạng nhất định và các cảm biến xâm nhập mạng của chúng.

1.2.2 Buffer Overflow Attacks (Tấn công tràn bộ đệm)

Cách tiếp cận này cố gắng ghi đè các phần cụ thể của bộ nhớ máy tính kết nối mạng, thay thế dữ liệu bình thường trong các vị trí bộ nhớ đó bằng một bộ lệnh mà sau này sẽ được thực thi như một phần của cuộc tấn công.

1.2.3 Common Gateway Interface Scripts

Giao diện cổng chung (CGI) thường được sử dụng trong mạng để hỗ trợ tương tác giữa máy chủ và máy khách trên Web. Nhưng nó cũng cung cấp các lỗ hổng dễ dàng - chẳng hạn như "backtracking" - thông qua đó những kẻ tấn công có thể truy cập các tệp hệ thống mạng được cho là an toàn.

1.2.4 Protocol-Specific Attacks (Tấn công theo giao thức mạng)

Khi thực hiện các hoạt động mạng, các thiết bị tuân theo các quy tắc và thủ tục cụ thể. Các giao thức này - chẳng hạn như ARP, IP, TCP, UDP, ICMP và các giao thức ứng dụng khác nhau - có thể vô tình để lại lỗ hổng cho các cuộc xâm nhập mạng thông qua mạo danh giao thức ("giả mạo") hoặc thông báo giao thức không đúng định dạng. Ví dụ: Giao thức phân giải địa chỉ (ARP) không thực hiện xác thực trên tin nhắn, cho phép kẻ tấn công thực hiện các cuộc tấn công "man-in-the-middle".

1.2.5 Traffic Flooding (Tấn công tràn lưu lượng mạng)

Một phương pháp xâm nhập mạng khéo léo chỉ đơn giản là nhắm vào các hệ thống phát hiện xâm nhập mạng bằng cách tạo ra tải trọng quá lớn để hệ thống không thể sàng lọc tất cả dữ liệu vào mạng. Trong môi trường mạng hỗn loạn và tắc nghẽn, những kẻ tấn công có thể thực hiện một cuộc tấn công không bị phát hiện và thậm chí gây ra tình trạng "không mở được" (fail-open) không bị phát hiện.

1.2.6 Trojans

Các chương trình này tự thể hiện là lành tính (không có những hành vi ăn cắp, phá hoại dữ liệu) và không tự tái tạo giống như vi-rút hoặc sâu. Thay vào đó, chúng mở các cửa

hậu cho các hành vi tấn công khác, cho phép những kẻ tấn công bên ngoài kiểm soát hệ thống.

1.2.7 Worms (Sâu máy tính)

Sâu là một loại phần mềm độc hại có khả năng tự lây nhiễm từ máy này sang máy khác mà không cần chương trình chủ, vật chủ, hoặc sự trợ giúp của người dùng. Khi sâu lây nhiễm vào một máy, nó sử dụng máy này làm "bàn đạp" để tiếp tục rà quét, tấn công các máy khác.

1.3 Các biện pháp phòng chống tấn công, xâm nhập mạng

- Chiến lược an toàn hệ thống
- Tính logic, khoa học, an toàn ở mức cao là những yếu tố hết sức cần thiết đối với hệ thống thông tin hiện nay.
- Quyền tối thiểu (Least Privilege)
- Phòng thủ theo chiều sâu (Defense in Depth):
- Điểm thắt (Choke Point): Chiến lược này buộc kẻ tấn công sử dụng một kênh hẹp, mà quản trị viên có thể giám sát và kiểm soát, tăng cường các hình thức giám sát, bảo đảm ATTT nâng cao.
- Liên kết yếu nhất (Weakest Link): Mỗi một hệ thống thông tin luôn có những điểm yếu. Ta cần phải liên tục gia cố, tăng cường bảo mật cho các yếu điểm hệ thống.
- Lập trường thất bại an toàn (Fail-Safe Stance): Trong cơ chế này, khi hệ thống bị sự cố mất ATTT hoặc bị lỗi, nó sẽ chặn các truy cập từ cả người dùng hợp pháp và người dùng bất hợp pháp đến khi vấn đề được xử lý xong.
- Phòng thủ đa dạng *(Diversity of Defense)*: Ý tưởng đằng sau sự đa dạng của hệ thống phòng thủ là việc sử dụng cân bằng các hệ thống bảo mật từ các nhà cung cấp khác nhau có thể làm giảm nguy cơ xảy ra lỗ hổng phổ biến hoặc lỗi cấu hình ảnh hưởng đến hệ thống.
- Đơn giản hóa (Simplicity): Đơn giản hóa là yếu tố cần thiết đối với hệ thống thông tin. Có 02 lý do cho sự cần thiết này như sau. Thứ nhất, hệ thống càng đơn giản thì càng dễ hiểu. Thứ hai, các chương trình, hệ thống càng phức tạp thì nguy cơ lỗi, nguy cơ tồn tại lỗ hổng bảo mật càng cao.

1.4 Khái quát về phát hiện xâm nhập mạng

1.4.1 Giới thiệu

Phát hiện xâm nhập mạng là quá trình theo dõi các sự kiện xảy ra trong một hệ thống thông tin và phân tích chúng để tìm ra các dấu hiệu xâm nhập trái phép hoặc các hành vi tấn công có thể xảy ra, đó là các hành vi hoặc các mối đe dọa sắp xảy ra, vi phạm các chính sách bảo mật, các chính sách sử dụng được chấp nhận hoặc vi phạm tiêu chuẩn bảo mật gây ảnh hưởng đến hệ thống.

1.4.2 Phân loại

Có 02 phương pháp phân loại chính các hệ thống IDS là phân loại theo nguồn dữ liệu và phân loại theo phương pháp phân tích dữ liệu.

Đối với phân loại theo nguồn dữ liệu, có 02 loại hệ thống IDS.

1.4.2.1 Hệ thống phát hiện xâm nhập ở mức mạng (Network – based IDS)

NIDS là một hệ thống độc lập, xác định các truy cập trái phép bằng cách kiểm tra các luồng thông tin trên mạng và giám sát nhiều máy. NIDS truy cập vào luồng thông tin trên mạng bằng cách kết nối vào các Hub, Switch được cấu hình Port mirroring hoặc sử dụng

Network tap để bắt các gói tin, phân tích nội dung các gói tin và từ đó sinh ra các cảnh báo hoặc phát hiện tấn công.

Nhược điểm của hệ thống NIDS là giới hạn băng thông và có thể xảy ra hiện tượng tắc nghẽn cổ chai khi lưu lượng mạng sử dụng ở mức cao.

1.4.2.1Hệ thống phát hiện xâm nhập ở mức máy (Host – based IDS)

HIDS thường là một phần mềm chạy trên các thiết bị đầu cuối làm việc để giám sát tất cả các hoạt động trên máy. Hệ thống này phân tích thông tin thu được trong nội bộ hệ thống.

Nhược điểm của HIDS là việc thu thập dữ liệu xảy ra trên mỗi máy và ghi vào log do đó có thể làm giảm hiệu năng mạng, ảnh hưởng đến tài nguyên sử dụng trên máy.

Đối với phân loại theo phương pháp phân tích dữ liệu thì IDS được chia làm 02 dạng phát hiện dấu hiệu dựa trên chữ ký (Signature-based IDS) và dựa vào bất thường (*Anomaly-based IDS*)

1.4.2.3 Phát hiện xâm nhập dựa trên chữ ký (Signature-based IDS)

Hệ thống IDS loại này dựa vào các dấu hiệu của các cuộc xâm nhập. Những dấu hiệu đó có thể là thông tin về các kết nối nguy hiểm đã biết trước. Hệ thống sẽ mô hình hóa các dấu hiệu của các cuộc xâm nhập đã biết và bằng việc so sánh thông tin của các gói tin đến với các dấu hiệu này để phát hiện ra các hoạt động đáng ngờ và đưa ra cảnh báo cho hệ thống.

Ưu điểm của kỹ thuật này là rất hiệu quả trong việc phát hiện tấn công đã biết với tỷ lệ cảnh báo sai thấp.

Tuy nhiên, ***nhược điểm*** chính của kỹ thuật này là chỉ phát hiện được những cuộc tấn công đã biết, không có khả năng phát hiện các tấn công mới hoặc chưa biết do chữ ký không tồn tại trong CSDL.

1.4.2.4 Phát hiện xâm nhập dựa vào bất thường (Anomaly-based IDS)

Ý tưởng của cách tiếp cận này xuất phát từ giả thiết "Dấu hiệu của các cuộc tấn công khác biệt với dấu hiệu của những trạng thái mạng được coi là bình thường". Khi đó, việc phát hiện sẽ được tiến hành qua hai giai đoạn: giai đoạn huấn luyện (pha huấn luyện) và gia đoạn phát hiện (pha phát hiện). Tại pha huấn luyện sẽ xây dựng một hồ sơ về các hoạt động bình thường (thông số chuẩn). Sau đó tại pha phát hiện sẽ tiến hành so khớp các quan sát (gói tin) với hồ sơ từ đó xác định dấu hiệu bất thường.

Ưu điểm của kỹ thuật này là hiệu quả trong việc phát hiện các mối nguy hiểm không được biết trước. Những năm gần đây, hướng tiếp cận này đang thu hút rất nhiều sự quan tâm của các nhà nghiên cứu. *Nhược điểm* của kỹ thuật phát hiện dựa trên bất thường là tỷ lệ cảnh báo sai thường cao và đòi hỏi lượng lớn tài nguyên tính toán cho xây dựng hồ sơ, hoặc mô hình phát hiện.

CHƯƠNG II
PHÁT HIỆN XÂM NHẬP DỰA TRÊN HỌC SÂU

2.1.Khái quát về học máy và học sâu

2.1.1 Khái quát về học máy

Học máy (*machine learning*) là một lĩnh vực của trí tuệ nhân tạo liên quan đến việc nghiên cứu và xây dựng các kĩ thuật cho phép các hệ thống "học" tự động từ dữ liệu để giải quyết những vấn đề cụ thể.

2.1.2 Khái quát về học sâu

Học sâu (Deep Learning) là một nhánh của lĩnh vực Học máy (Machine Learning) dựa trên một tập hợp các thuật toán để cố gắng mô hình hóa dữ liệu trừu tượng hóa ở mức cao bằng cách xử lý với cấu trúc phức tạp, hoặc bằng cách khác nhau bao gồm nhiều biến đổi phi tuyến. Cốt lõi của Deep Learning bao gồm mô hình mạng neural nhiều lớp và quá trình huấn luyện mạng để xác định tham số cho mô hình. Trong Deep Learning, có 03 dạng học chính là học có giám sát, học nửa giám sát và học không giám sát.

2.2. Học sâu sử dụng Autoencoder và ứng dụng trong tiền xử lý dữ liệu

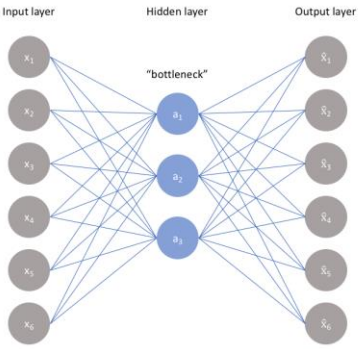
2.2.1 Học sâu sử dụng Autoencoder

Autoencoder là một loại ANN dùng để học không có giám sát thông qua các mã code với ý tưởng là nếu một mô hình mạng neural có số nút mã trung gian (hidden layer) nhỏ hơn số nút đầu vào thì mô hình đó sẽ học được các đặc tính ẩn (features) của dữ liệu. Chính vì vậy mà Autoencoder học được cách biểu diễn cho một tập dữ liệu giúp dự đoán đầu ra từ một đầu vào ban đầu. Trong thực tế Autoencoder đã được ứng dụng thành công để giảm chiều dữ liệu, tất nhiên không làm mất đi các đặc tính quan trọng của dữ liệu.

Cấu trúc Autoencoder được chia thành bộ mã hóa và bộ giải mã, bao gồm cả input layer, hidden layer, output layer. Autoencoder có một hoặc nhiều hidden layer có chức năng mã hóa để tạo ra dữ liệu có chứa các thuộc tính cơ bản nhất có thể mô tả đầy đủ dữ liệu đầu vào. Sau đó, bộ giải mã tạo ra một sự tái thiết của bộ mã hóa để tạo ra đầu ra giống với đầu vào nhất có thể. Một autoencoder có thể mã hóa dữ liệu dầu vào bằng cách có 01 hidden layer có số nút nhỏ hơn input layer vì vậy buộc nó phải tìm ra mối tương quan giữa các thành phần của dữ liệu để có thể tìm ra các thuộc tính chính. Điều này tạo điều kiện cho việc phân loại, trực quan hóa, giao tiếp và lưu trữ dữ liệu. Mục đích của autoencoder là thử và tìm hiểu hàm được hiển thị trong phương trình:

$$h(W, b(x)) \approx x \qquad (1)$$

Trong đó W là trọng số, b là bias.



Hình 2.1 Sơ đồ cấu trúc mạng Autoencoder

Trường hợp đơn giản nhất, khi có 01 tầng ẩn, tầng encoder của autoencoder lấy input $x \in R^d = X$ và ánh xạ tới $z \in R^p = F$: $z = \delta(Wx + b)$ (2)

z được gọi là code, biến tiềm ẩn hoặc biểu diễn tiềm ẩn. Còn δ là một hàm kích hoạt (activation function) như sigmoid function (*hàm toán học có biểu đồ hình chữ S nằm ngang*). W là một ma trận trọng số và b (bias) là vector sai lệch. Sau đó, giai đoạn giải mã của autoencoder là lấy z để tái thiết x' giống x: $x' = \delta'(W'z + b')$ (3)

Ở đây, δ', W' và b' đối với bộ giải mã có thể khác với δ, W, b của mã hóa tùy thuộc vào việc thiết kế autoencoder.

Mạng Autoencoder có thể tính toán trên bộ số liệu đầu vào mới dựa trên bộ trọng số đã lưu trong quá trình huấn luyện để đưa ra kết quả đầu ra tương ứng.

Denoise Autoencoder (DAE) được phát triển từ Autoencoder nhưng mạnh mẽ hơn. Đầu vào của DAE là dữ liệu bị làm nhiễu và chúng ta sẽ học các đặc trưng của dữ liệu từ dữ liệu nhiễu. Nhưng sau quá trình giải mã, đầu ra sẽ là dữ liệu ban đầu trước khi bị làm nhiễu.

Trước khi đào tạo mạng Autoencoder, ta cần thiết lập các tham số sau:

✓ Code size: Số lượng nút trong hidden layer (tầng ẩn) giữa. Kích thước hidden layer càng nhỏ thì dữ liệu càng nén nhiều.

✓ Số lớp: Ta có thể thấy nerural network có nhiều hidden layer hơn có thể biểu diễn các function phức tạp hơn. Tuy nhiên, nhược điểm là nó quá phức tạp để huấn luyện.

✓ Số lượng nút trên mỗi lớp: Số lượng nút trên mỗi lớp giảm theo từng lớp tiếp theo của encoder và tăng trở lại trong decoder.

✓ Loss function: Mục đích của quá trình huấn luyện mạng AE và DAE là để tìm được weight (*trọng số*) đúng, các thuật toán cần tìm weight để tạo đầu ra giống với đầu vào nhất có thể. Phương trình để tính độ sai lệch này được gọi là loss function.

✓ Learning rate (*tốc độ học*): Tốc độ hội tụ của GD phụ thuộc vào *learning rate*. Với learning rate quá nhỏ tốc độ hội tụ rất chậm. Với learning rate lớn, thuật toán tiến rất nhanh tới gần đích sau vài vòng lặp. Tuy nhiên, thuật toán không hội tụ được vì bước nhảy quá lớn, khiến nó cứ quẩn quanh ở đích.

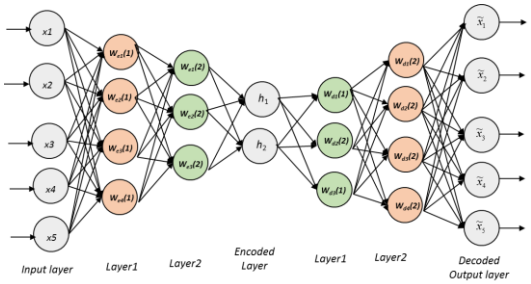
✓ Chỉ số AUC (Area Under the ROC Curve): AUC cung cấp phương pháp đánh giá kết hợp hiệu suất trên tất cả các ngưỡng phân loại khả thi. Khoảng giá trị AUC từ 0 đến 1. 01 mô hình mà sự dự đoán của nó là 100% sai có giá trị AUC là 0.0; một mô hình mà sự dự đoán của nó là 100% đúng thì có giá trị AUC là 1.0

2.2.2 Phân loại Autoencoder

- Autoencoder chưa hoàn thành (Undercomplete Autoencoder): Autoencoder có kích thước mã ít hơn kích thước input được gọi là undercomplete Autoencoder. Một tính chất undercomplete của autoencoder là nắm bắt các đặc tính đặc trưng nhất của dữ liệu huấn luyện.

- Regularized Autoencoder (Autoencoder đúng quy tắc): Ta có thể huấn luyện bất kỳ kiến trúc nào của autoencoder thành công khi kích thước mã và sức chứa của quá trình encoder và quá trình decoder được mô hình hóa dựa trên độ phức tạp của phân phối dữ liệu. Regularized autoencoder cung cấp khả năng như vậy. Regularized autoencoder có thể là phi tuyến và overcomplete nhưng vẫn có thể học được điều gì đó hữu ích về phân phối dữ liệu.

- Stacked Autoencoder (SAE): Stacked Autoencoder cũng giống với Autoencoder bình thường có đủ 03 thành phần chủ yếu trong mạng là: input layer, hidden layer, output layer, và có 02 quá trình là encoder và decoder.

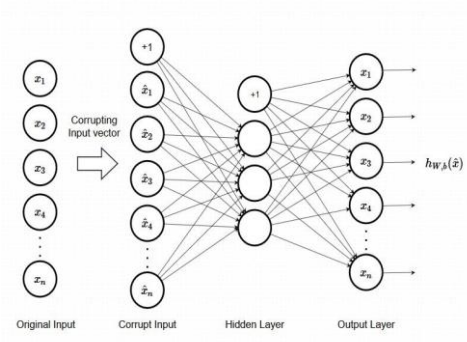


Hình 2.2 Sơ đồ cấu trúc mạng Stacked Autoencoder

Stacked Autoencoder (*SAE*) là sự xếp chồng lên nhau của AE vì vậy nó có nhiều hidden layer hơn Regularized Autoencoder và Undercomplete Autoencoder. Ta xem xét Stacked Autoencoder với n lớp. Sử dụng các ký hiệu $W_{1(k)}$,$W_{2(k)}$, $b_{1(k)}$, $b_{1(k)}$ biểu diễn các tham số $W_{(1)}$, $W_{(2)}$, $b_{(1)}$, $b_{(2)}$ cho hidden layer thứ k. Một cách tốt nhất để có được các tham số cho SAE là đào tạo theo layer-wise training (*đào tạo khôn ngoan*). Trước tiên, đào tạo lớp đầu tiên có được các tham số $W_{1(1)}$, $W_{2(1)}$, $b_{1(1)}$, $b_{2(1)}$. Sử dụng lớp đầu tiên để chuyển đổi data thành một vectơ. Vectơ đầu ra của lớp đầu tiên này sẽ làm đầu vào cho lớp thứ hai. Huấn luyện lớp thứ hai trên vectơ của lớp đầu thu được các tham số $W_{1(2)}$, $W_{2(2)}$, $b_{1(2)}$, $b_{2(2)}$. Cứ như vậy, sử dụng đầu ra của mỗi lớp làm đầu vào cho lớp tiếp theo.

- Denoise Autoencoder (DAE) được phát triển từ Autoencoder nhưng mạnh mẽ hơn. Đầu vào của DAE là dữ liệu bị làm nhiễu và chúng ta sẽ học các đặc trưng của dữ liệu từ dữ liệu nhiễu. Nhưng sau quá trình giải mã, đầu ra sẽ là dữ liệu ban đầu trước khi bị làm nhiễu. Từ đó, ta có thể thấy khả năng khái quát hóa cả DAE tốt hơn so với Autoencoder. Hơn nữa, DAE có thể xếp chồng lên nhau để có được feature tốt hơn. Đào tạo mạng SDAE theo layer-wise vì mỗi DAE với một hidden layer được đào tạo độc lập. Sau khi đào tạo mạng SDAE, các lớp giải mã được loại bỏ và các lớp mã hóa tạo ra các đặc trưng được giữ lại. Vì có khả năng phục hồi dữ liệu trước khi bị làm nhiễu nên DAE thường được dùng để khôi phục ảnh và các dữ liệu bị hỏng.

Denoise Autoencoder có chứa 03 lớp: input layer, hidden layer và output layer, trong đó input layer và hidden layer là lớp encoder còn output layer và hidden layer là lớp decoder. Số lượng nút trong input layer tương ứng bằng với số chiều của dữ liệu đầu vào. Encoder của DAE thu được bằng hàm biến đổi phi tuyến: $z = \delta(W\bar{x} + b)$ (4)



Hình 2.3 Sơ đồ cấu trúc mạng Denoise Autoencoder

Quá trình giải mã hoặc tái cấu trúc của DAE sử dụng mapping function (*thuật toán trong học máy hình thức hóa biểu thức ánh xạ dữ liệu đầu vào thành dữ liệu đầu ra*):

$$x' = \delta(W'z + b') \quad (5)$$

Trong đó, x' € R^d là đầu ra của quá trình giải mã của DAE, quá trình này tái cấu trúc dữ liệu ban đầu x trước khi bị làm nhiễu. Output layer có số nút bằng với input layer. Trong quá trình đào tạo DAE ta có loss function được tính theo lỗi bình phương trung bình nhưng chúng ta sẽ không tính theo dữ liệu đầu vào trong input layer mà tính theo dữ liệu trước khi bị làm nhiễu. Mục tiêu của quá trình học là giảm loss function tìm weight phù hợp vì vậy ta cần sử dụng thuật toán tối ưu Stochastic Gradient Descent.

- Stacked Denoise Autoencoder

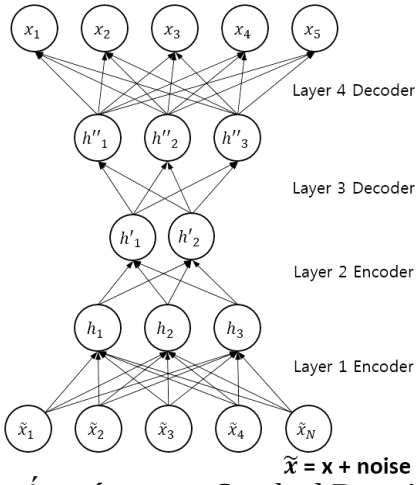
DAE có thể xếp chồng lên nhau để xây dựng deep network (mạng học sâu) có nhiều hơn 01 hidden layer. SDAE bao gồm hai phần: encoder và decoder. Trong phần encoder đầu ra của lớp đầu tiên đóng vai trò là dữ liệu đầu vào của lớp mã hóa thứ hai. Giả sử có L lớp ẩn trong encoder, chúng ta có hàm kích hoạt của lớp mã hóa thứ k:

$$z^{(k+1)} = \delta(W^{(k+1)}z^k + b^{(k+1)}, k = 0,1,\dots,L-1 \quad (16)$$

Trong phần decode, đầu ra của lớp thứ nhất là đầu vào lớp thứ hai, chúng ta có hàm kích hoạt của lớp decode thứ k:

$$x^{(k+1)} = \delta'(W'^{(k+1)}x^k + b'^{(k+1)}, k = 0,1,\dots,L-1 \quad (17)$$

Đầu vào của $x^{(0)}$ của decode là đầu ra của z^L. Đầu ra của decode là quá trình tái cấu trúc dữ liệu ban đầu x.



Hình 2.4 Sơ đồ cấu trúc mạng Stacked Denoise Autoencoder

2.2.3. Ứng dụng Autoencoder trong tiền xử lý dữ liệu

Trong đề tài này tôi sẽ sử dụng mạng học sâu là Autoencoder (AE) và một số thuật toán học máy để xác định tấn công xâm nhập mạng. Ứng dụng mạng học sâu có 02 ưu điểm chính:

- Thứ nhất, kết quả của các mạng học sâu không chịu chi phối của việc định nghĩa các đặc trưng của dữ liệu, điều đó có nghĩa là các dữ liệu đầu vào không cần phải trải qua công đoạn tiền xử lý và trích chọn các feature, chúng ta có thể đưa vào gần như là dữ liệu thô.

- Thứ hai, bản thân của các mạng học sâu vẫn sử dụng các thuật toán thống kê với quy mô siêu lớn, khi đưa vào càng nhiều dữ liệu thì độ chính xác càng cao.

Có hai giai đoạn trong quá trình phát hiện xâm nhập là: ***Learning Feature*** và ***Classifier***. Trong giai đoạn Learning Feature, các dữ liệu của mạng sẽ được đưa vào các mạng AE và DAE. Ta sẽ có được mã chứa các đặc trưng đại diện nhất của dữ liệu. Các đặc trưng này có thể mô tả được dữ liệu đầu vào, giúp cho việc phân loại nhanh hơn và chính xác hơn nhờ vào khả năng học của AE và DAE. Ngoài ra, ta cũng có thể sử dụng mạng SDAE để khôi phục được các dữ liệu bị hỏng. Trong giai đoạn Classifier, ta sẽ lấy các dữ liệu đã được trích xuất từ giai đoạn Learning Feature và sử dụng các thuật toán phân loại như SVM, RF, DT, KNN, NB để xác định dữ liệu đầu vào là bình thường hay bất thường.

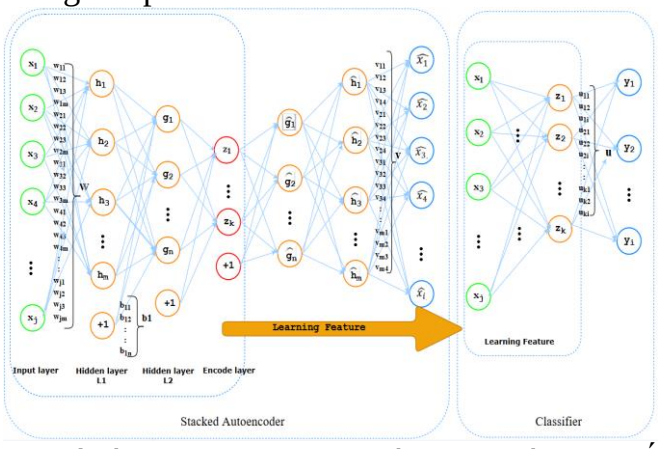
2. 3. Xây dựng mô hình phát hiện xâm nhập dựa trên Autoencoder

Trước khi quá trình Learning Feature có thể tạo ra các đặc trưng nhất của dữ liệu đầu vào, ta cần phải huấn luyện mạng để mạng có bộ trọng số phù hợp để cho dữ liệu xuất ra trong lớp output giống với dữ liệu đầu vào trong lớp input tức loss function nhỏ. Vậy quá trình huấn luyện sẽ tiến hành theo các bước sau:

- Bước 1. Chuẩn hóa dữ liệu đầu vào: Để huấn luyện mạng, ta sẽ dùng các bộ dữ liệu được dán nhãn NSS-KDD, Phishing. Sau đó, ta sẽ đưa tất cả dữ liệu số và chia dữ liệu theo tỷ lệ 70% để train và 30% để test. Dữ liệu train và test sẽ được chia thành các phần như X_train, Y_train, X_test, Y_test. Trong đó X_train là các data được dùng để train, Y_train là các label của X_train gồm 02 loại bao gồm normal: 0 và attack: 1, X_test là dữ liệu để test, Y_test là label của X_test.

- Bước 2. Xây dựng mạng SAE và SDAE: Từ những phương pháp huấn luyện mạng AE và DAE, loại Autoencoder và nội dung ứng dụng Autoencoder trong tiền xử lý dữ liệu, ta có mô hình mạng Noron phát hiện xâm nhập mạng dựa trên SAE và SDAE được xây dựng với cấu trúc:

Input Layer, 02 Hidden Layer và Output Layer (*như hình dưới*). Mục đích của việc huấn luyện là xác định bộ trọng số và giảm loss function. Trước khi huấn luyện, ta cần xác định các tham số cần thiết trong quá trình huấn luyện: learning rate = 1e - 4, batch_size = 100, num_epoch = 1000, step = 20 tức là mỗi lần chạy xong 20 epoch.



Hình 2.5 Mô hình ứng dụng SAE và SDAE vào hệ thống IDS

- Bước 3. Tiến hành huấn luyện: Đối với SDAE ta còn phải làm nhiễu dữ liệu trước khi huấn luyện. Mỗi lần huấn luyện 20 epoch, mỗi epoch huấn luyện 100 mẫu dữ liệu. Sau khi huấn luyện xong một lần, ta xem xét loss function đạt tiêu chuẩn không, sau đó sẽ sử dụng mạng SA và DAE để chuyển đổi X_train, X_test thành Z_train, Z_test chứa các đặc trưng của dữ liệu. Sau đó, Z_train, Y_train, Z_test, Y_test sẽ được classifier để học cách phân loại bằng các thuật toán phân loại.

Bước 4. Tiến hành phân loại: Sau khi sử dụng SAE và SDAE để có được các đặc trưng, ta sẽ sử dụng Z_train, Y_train cho các thuật toán phân loại để học. Sau đó, ta sẽ dùng nó để phân loại Z_test. Các thuật toán này sẽ tạo ra một Y_predict là kết quả phân loại của Z_test. Sau đó, ta sẽ so sánh với Y_test với Y_predict. Kết quả phân loại sẽ được tính theo chỉ số AUC.

CHƯƠNG III
CÀI ĐẶT VÀ THỬ NGHIỆM

3.1 Phương pháp cài đặt thử nghiệm

Để tiến hành quá trình thực hành và xây dựng hệ thống và huấn luyện tôi sử dụng ngôn ngữ lập trình Python làm ngôn ngữ để xây dựng hệ thống. Trong ngôn ngữ python, có hỗ trợ các thư viện dành cho deep learning như: Tensorflow, sklearn,…, các thư viện thao tác dữ liệu như pandas, numpy, .. và tôi sử dụng PyQt để xây dựng ứng dụng.

Tensorflow là một thư viện mã nguồn mở cung cấp khả năng xử lý tính toán dựa trên biểu đồ mô tả sự thay đổi của dữ liệu. Trong đó, các node là các phép tính toán học còn các cạnh biểu thị luồng dữ liệu. Ngoài ra, ta còn sử dụng thư viện Sklearn là thư viện mã nguồn mở hỗ trợ hầu hết các thuật toán học máy một cách đơn giản mà không cần phải cài đặt, lập trình lại. Các bước tiến hành thí nghiệm bao gồm:

- Bước 1: Lựa chọn bộ dữ liệu và chuẩn hóa dữ liệu;
- Bước 2: Xây dựng mạng SAE và SDAE bằng thư viện tensorflow;
- Bước 3: Huấn luyện mạng SAE và SDAE và các thuật toán phân loại;
- Bước 4: Cho dữ liệu test chạy qua mạng SAE và SDAE thu được mã chứa đặc trưng. Tiếp tục sử dụng mã để thực hiện quá trình phân lớp của dữ liệu;
- Bước 5: Đánh giá kết quả và kết luận;

3.2 Giới thiệu tập dữ liệu

3.2.1 Phishing Website Data

Phishing Website Data là bộ dữ liệu chứa các đặc trưng quan trọng trong việc phát hiện các trang bị tấn công phishing. Bộ dữ liệu có tổng cộng 30 đặc trưng và 2.456 mẫu dữ liệu, mỗi mẫu dữ liệu được gán nhãn dán là tấn công hoặc bình thường. Các đặc trưng bao gồm 03 trạng thái: Nghi ngờ, phishing và hợp pháp. Bộ dữ liệu này được chia chia thành 02 phần, một phần để huấn luyện chiếm 70% và một phần để testing chiếm 30%.

3.2.2 NSL-KDD

Tập dữ liệu NSL-KDD dùng để huấn luyện bao gồm 125.973 bản ghi và tập dữ liệu kiểm tra gồm 22.544 bản ghi. Mỗi bản ghi có 41 thuộc tính và được dán nhãn là bình thường hoặc cuộc tấn công một cách chính xác với một kiểu tấn công cụ thể. Tập dữ liệu huấn luyện chứa 22 kiểu tấn công và thêm 17 kiểu trong dữ liệu kiểm

3.3 Trích chọn đặc trưng sử dụng AE

Ta xây dựng mô hình mạng Noron phát hiện xâm nhập mạng dựa trên SAE và SDEA cấu trúc: Input Layer, 02 Hidden Layer, Output Layer. Số nút mỗi layer phụ thuộc vào số lượng đặc trưng của từng loại dữ liệu. Các tham số cần thiết trong quá trình huấn luyện: learning rate = 1e-4, batch_size = 100, num_epoch = 1000, step = 20 tức là mỗi lần chạy 20 epoch.

3.4 Huấn luyện và phát hiện

- Trong pha huấn luyện và phát hiện, ta sử dụng hàm có sẵn của thư viện tensorflow để chuyển đổi X_train, X_test thành Z_train, Z_test chứa các đặc trưng của dữ liệu.
- Sau đó, Z_train, Y_train, Z_test, Y_test sẽ được classifier để học cách phân loại bằng các thuật toán phân loại: SVM, DT, RF, NB, KN.

3.5 Kết quả và nhận xét

3.5.1 Kết quả của bộ dữ liệu Phishing Website Data

Sau quá trình xây dựng và huấn luyện mạng SAE và SDAE với bộ dữ liệu Phishing Website Data, tôi đã xem xét kết quả loss function, AUC của quá trình classifier và nhận thấy rằng: Khi sử dụng bộ dữ liệu Phishing Website Dataset có 30 feature thì cấu trúc mạng tối ưu nhất là mạng có 02 hidden layer và số lượng nút mỗi layer là [25, 15]. Tôi đã thử huấn luyện với số lượng hidden layer là 03 và 04 hidden và 01 hidden layer thì kết quả AUC không tốt bằng khi sử dụng 02 hidden layer.

Để so sánh việc sử dụng các mạng SAE và SDAE và không sử dụng mạng trong quá trình phân loại, tôi tiến hành so sánh chỉ số thời gian phân lớp, độ chính xác AUC của chúng. Sau khi so sánh, tôi nhận thấy rằng việc sử dụng mạng SAE và SDAE để học các đặc trưng của dữ liệu cho kết quả AUC tốt hơn việc không sử dụng mạng SAE và SDAE. Tuy nhiên, việc sử dụng mạng học sâu có kết quả khác nhau đối với các thuật toán phân loại khác nhau.

Qua bảng dữ liệu AUC của bộ dữ liệu Phishing Website Data bên dưới ta có thể nhận thấy rằng, việc phân loại của dữ liệu không có các đặc trưng nhận được từ SAE và SDAE có chỉ số UAC không cao bằng so với các dữ liệu được học các đặc trưng bởi SAE và SDAE. Đặc biệt, đối với thuật toán Naive Bayes chỉ số UAC tăng khoảng 20 % đối với mạng SAE còn các thuật toán khác chỉ tăng từ 02 đến 03%.

Đối với mạng SDAE, thuật toán Naive Bayes tăng khoảng 23% và các thuật toán khác cũng chỉ tăng từ 02 đến 03 %. Nhưng ta có thể nhận thấy rằng chỉ số AUC của mạng SDAE cao hơn mạng SAE trong bộ dữ liệu này.

Bảng 3.4 Bảng so sánh AUC giữa sử dụng SAE, SDAE và không sử dụng đối với bộ dữ liệu Phishing Data Website

Thuật toán Deep learning	**SVM**	**Random Forest**	**Naive Bayes**	**K-Neighbors**	**Decision Tree**
Không dùng	0.909	0.947	0.706	0.944	0.937
SAE	0.930	0.970	0.921	0.970	0.955
SDAE	0.931	0.972	0.930	0.974	0.954

Sau khi huấn luyện mạng SAE và SDAE với dữ liệu huấn luyện có epoch =20 (*thực hiện việc học toàn bộ dữ liệu 20 lần*), ta tiến hành thực hiện học các đặc trưng của dữ liệu test và phân loại chúng.

Tiếp theo, tiến hành tính AUC dựa trên kết quả phân loại. Ta thực hiện tổng cộng 1000 epoch. Ta nhận thấy quá trình phân loại cho kết quả tương đối tốt ngay từ những đợt huấn luyện đầu tiên. Với thuật toán NB và SVM chỉ số AUC của cả 02 mạng đều không biến động nhiều trong suốt quá trình nhưng ngược lại các thuật toán DT, KN và RF chỉ số AUC biến đổi liên tục không ổn định. Đặc biệt là thuật toán DT độ biến thiên rất nhiều.

Qua phân tích đồ thị giá trị hàm loss function của cả 02 mạng, ta nhận thấy rằng khả năng hội tụ của cả 02 mạng đều rất nhanh, ngay từ epoch đầu tiên và sau khi có kết quả tốt độ biến thiên của hàm loss rất ít. Điều này có nghĩa 02 mạng SAE và DAE đã học các đặc trưng rất tốt ngay từ những đợt huấn luyện đầu tiên.

Thời gian huấn luyện được tôi dựa trên thời gian phân loại một mẫu (*sample*) của dữ liệu. Đối với sử dụng mạng học sâu, tôi tính thời gian phân loại bằng thời gian học đặc trưng cộng với thời gian phân nhóm một mẫu. Tôi nhận thấy rằng các thuật toán KN và NB sử dụng mạng SAE và SDAE có thời gian phân loại tốt hơn khi không sử dụng. Còn đối với các thuật toán khác thì hoàn toàn ngược lại.

Đó chính là nhược điểm của việc áp dụng mạng học sâu đối với các thuật toán này.

3.5.2 Kết quả của bộ dữ liệu NSL-KDD

Bộ dữ liệu NSL-KDD bao gồm 41 feature nên trong quá trình huấn luyện, tôi đã xây dựng mạng SAE và SDAE với cấu trúc 02 hidden layer [30,15]. Bộ dữ liệu được dán 02 loại dãn nhán là tấn công và không tấn công.

Tương tự như đối với bộ dữ liệu Phishing Website Data, thuật toán NB khi sử dụng mạng SAE và SDAE với bộ dữ liệu NSL-KDD tăng rất nhiều khoảng 30% đối với cả 02 mạng. Các thuật toán khác cũng tăng từ 02 đến 07% ngoại trừ thuật toán SVM hầu như không tăng.

Bảng 3.5 Bảng so sánh AUC giữa sử dụng SAE, SDAE và không sử dụng đối với bộ dữ liệu NSL-KDD

Thuật toán Deep learning	SVM	Random Forest	Naive Bayes	K-Neighbors	Decision Tree
Không dùng	0.789	0.813	0.587	0.777	0.822
SAE	0.854	0.838	0.810	0.844	0.865
SDAE	0.855	0.846	0.815	0.836	0.870

Ta cũng nhận thấy giá trị hàm loss function của cả 02 mạng khi huấn luyện bằng bộ dữ liệu NSL-KDD cũng hội tụ về điểm cực tiểu rất nhanh nhờ thuật toán SGD và độ biến thiên của mạng không nhiều. Loss function của mạng SAE tốt hơn SDAE.

Thời gian phân loại của bộ dữ liệu NSL-KDD khi sử dụng mạng SAE và SDAE cũng giống như với bộ dữ liệu Phishing Website Data, thuật toán NB và KN là những thuật toán có thời gian phân loại tốt nhất khi sử dụng mạng SAE và SDAE. Các thuật toán khác không tối ưu về thời gian khi sử dụng mạng học sâu.

KẾT LUẬN

Trong quá trình nghiên cứu, tôi đã tìm hiểu chi tiết về các cơ sở lý thuyết để thực hiện đồ án như: Lý thuyết về mạng Noron, Deep Learning, phát hiện xâm nhập mạng. Trong học sâu, tôi đi vào tìm hiểu cấu trúc mạng Autoencoder. Đối với các bài toán phát hiện xâm nhập mạng, chúng ta thường sử dụng các thuật toán phân lớp để phân loại dữ liệu đầu vào, xem xét chúng để đưa ra kết luận chúng có phải là tấn công hay không. Việc đưa dữ liệu bình thường vào để phân loại đem lại hiệu quả không cao và tốn thời gian phân loại. Chính vì vậy, tôi đã sử dụng mạng học sâu AE. Khi sử dụng 01 hidden layer với AE, tôi nhận thấy rằng kết quả AUC không được tốt nên tôi đã sử dụng mạng với nhiều hidden layer. Tôi đã ứng dụng 02 mạng này vào hệ thống IDS và chia quá trình phát hiện thành 02 phần: Learning feature (học đặc trưng) và classifier (phân loại). Sau quá trình xây dựng và thực hiện huấn luyện mạng, tôi nhận thấy rằng: Sử dụng mạng học sâu SAE và SDAE đem lại hiệu quả cao đối với việc phát hiện xâm nhập mạng. Trong luận văn này, tôi sử dụng các thuật toán phân lớp SVM, RF, KNN, NB, DT. Kết quả tốt nhất khi sử dụng thuật toán NB. Với các thuật toán khác, kết quả đều tăng nhưng tăng rất ít. Về thời gian, có 02 thuật toán phân lớp rất tốt là NB và KNN khi sử dụng với mạng học sâu.

Tỷ lệ AUC của các thuật toán ngoại trừ thuật toán NB tăng rất ít và một số thuật toán (*RF, DT*) có thời gian phân loại không tốt hơn nhiều so với việc không sử dụng mạng SAE, SDAE. Đây là hạn chế trong nghiên cứu này của tôi. Trong thời gian tới, tôi sẽ tiếp tục nghiên cứu, tìm hiểu thêm để cải tiến, tối ưu mạng với mục tiêu giảm thời gian phân loại hơn nữa khi sử dụng mạng học sâu, tăng chỉ số AUC hơn nữa đối với từng thuật toán./.